Felipe Calliari Ribas

Caracterização de Algoritmos de Segmentação de Dígitos Manuscritos

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2010.

Felipe Calliari Ribas

Caracterização de Algoritmos de Segmentação de Dígitos Manuscritos

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de concentração: Ciência da Computação.

Orientador:
Prof^{o}. Dr^{o}. Alceu de Souza Britto Jr.
Co-orientador:
Prof^{o}. Dr^{o}. Luiz Eduardo Soares de Oliveira

Curitiba
2010.

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

R482c
2010

Ribas, Felipe Calliari
Caracterização de algoritmos de segmentação de dígitos manuscritos / Felipe Calliari Ribas ; orientador, Alceu de Souza Britto Jr. ; co-orientador, Luiz Eduardo Soares de Oliveira. – 2010.
xi, 68 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2010
Bibliografia: f. 66-68

1. Sistemas de reconhecimento de padrões. 2. Processamento de imagens - Técnicas digitais. 3. Algoritmos de computador. I. Britto Júnior, Alceu de Souza, 1966-. II. Oliveira, Luiz Eduardo Soares de. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 20. ed. – 005.1

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 04/2010

Aos 10 dias do mês de março de 2010 realizou-se a sessão pública de Defesa da Dissertação **"Caracterização de Algorítmos de Segmentação de Dígitos Manuscritos,"** apresentada pelo aluno **Felipe Calliari Ribas** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Alceu de Souza Britto Junior PUCPR (Orientador)	*(assinatura)*	APROVADO *(aprov/reprov.)*
Profª. Drª. Cinthia Obladen de Almendra Freitas PUCPR		APROVADO
Prof. Dr. Luiz Eduardo Soares de Oliveira UFPR		APROVADO
Profª. Drª. Aura Conci UFF/RJ		APROVADO

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Mauro Sérgio Pereira Fonseca
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax: (41) 3271 2121 www.ppgia.pucpr.br

Agradecimentos

Agradeço a meus pais Romy José Ribas e Ana Maria Calliari Ribas, por terem investido em minha educação e apoiado minhas escolhas, o que com certeza foi essencial para o sucesso deste trabalho.

Agradeço a meus orientadores, Prof. Dr. Alceu de Souza Britto Jr. e Prof. Dr. Luiz Eduardo Soares de Oliveira, pelo ensino, suporte, paciência e amizade durante esses anos.

Agradeço a Pontifícia Universidade Católica do Paraná, por ter me concedido a bolsa de estudos, valorizando assim meu esforço durante a graduação.

Agradeço também a todos que souberam compreender os momentos em que estive ocupado me dedicando à este trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AG	Algoritmo Genético
ASCII	American Standard Code for Information Interchange
CC	Componentes Conectados
CG	Centro de Gravidade
CEDAR	Center of Excellence for Document Analysis and Recognition
DDR2	Double Data Rate 2
Gb	Giga Bytes
HMM	Hidden Markov Model
k-NN	k-Nearest Neighbors
Mb	Mega Bytes
MLP	Multi-Layer Perceptron
NIST	National Institute of Standards and Technology
RAM	Random Access Memory
SD19	Base de dados NIST de formulários e caracteres manuscritos
SVM	Support Vector Machine

Resumo

Este trabalho apresenta um estudo comparativo de algoritmos de segmentação de pares de dígitos manuscritos onde diferentes tipos de conexões são considerados. O objetivo principal consiste em coletar estatísticas detalhadas sobre os algoritmos de modo que em um trabalho futuro seja possível desenvolver um sistema de segmentação que combine muitos algoritmos, levando em consideração os casos em que cada um possui melhor desempenho, para assim obter taxas de acerto melhores do que as que são alcançadas por algoritmos únicos.

O método de avaliação estabelece um critério justo de comparação de seus desempenhos, o que é um diferencial deste trabalho em relação a outros. Resultados interessantes foram obtidos, os quais mostram que um algoritmo que possui a melhor taxa de acerto geral da segmentação não é necessariamente o melhor para todos os tipos de conexão.

Analisando os resultados deste estudo nós observamos que seria possível obter uma melhora de até 34,33% na taxa de acerto da segmentação se fosse desenvolvido um sistema ideal de combinação de algoritmos de segmentação. Com esses resultados concluímos que um sistema que combine algoritmos de segmentação de pares de dígitos manuscritos, é uma idéia promissora à ser desenvolvida em trabalhos futuros.

Palavras-Chave: Segmentação, Dígitos Manuscritos, Comparação de Algoritmos.

Abstract

This work presents a comparative study of segmentation algorithms for handwritten digit pairs where different types of connections were considered. The main goal consists in collecting detailed statistics about the algorithms so in a future work would be possible to develop a segmentation system that combines many algorithms, taking into account cases where each one has better performance, thereby achieving better segmentation rates than those achieved by single algorithms.

The evaluation method provides a fair protocol for the comparison of their performance, which is a differential of this work compared to others. Interesting results were obtained, which show that an algorithm that has the best overall correct segmentation rate is not necessarily the best for all connection types.

Analyzing the results of this study we observed that it would be possible to obtain an improvement of up to 34.33% in accuracy rate in the segmentation if it would possible to develop a system with a perfect combination of segmentation algorithms. With these results we conclude that a system to combine segmentation algorithms for handwritten digit pairs is a promising idea to be developed in future works.

Keywords: Segmentation, Handwritten Digits, Comparison of Algorithms.

Capítulo 1

Introdução

Diversas atividades do dia a dia utilizam documentos manuscritos, dentre elas podemos citar o reconhecimento de envelopes postais, processamento automático de cheques bancários e formulários diversos, bem como a indexação de documentos históricos. Essas atividades tem muito a ganhar com a utilização de sistemas automatizados de reconhecimento de manuscritos. Esse tipo de sistema pode trazer diversos benefícios à sociedade, como por exemplo aumentar a velocidade do processamento de informações, proporcionar maior sigilo para documentos de acesso restrito e a possibilidade de substituir o trabalho humano pela máquina em atividades que são repetitivas e cansativas demais para o homem.

Reconhecimento de manuscritos é uma área de pesquisa bem antiga e que é bastante ativa. Porém, mesmo com a grande exploração dessa área, ela continua sendo muito atrativa devido à constante e rápida evolução da tecnologia, o que proporciona que os sistemas possam assumir um nível de complexidade cada vez maior em busca de um melhor desempenho. Principalmente nos dias de hoje com vários tipos de documento migrando para o formato digital, existem muitas aplicações para sistemas de reconhecimento de manuscritos. Isso se deve à necessidade de muitos documentos precisarem existir em forma de papel, seja por praticidade em atividades do dia a dia, por questões legais ou até mesmo documentos antigos que possuem conteúdo importante e precisam ser mantidos por muito tempo, criando então, a necessidade de serem processados e indexados.

Diversos autores sugerem diferentes estruturas para a construção de sistemas de reconhecimento de dígitos manuscritos, mas de modo geral, podemos exemplificar a arquitetura desse tipo de sistema pelas seguintes etapas: aquisição da imagem, pré-processamento, segmentação dos dígitos, reconhecimento e pós-processamento, como podemos observar na Figura 1.1.

A primeira etapa visa obter a imagem a ser processada, o que dependendo do tipo de sistema, pode representar diferentes processos como: capturar uma imagem

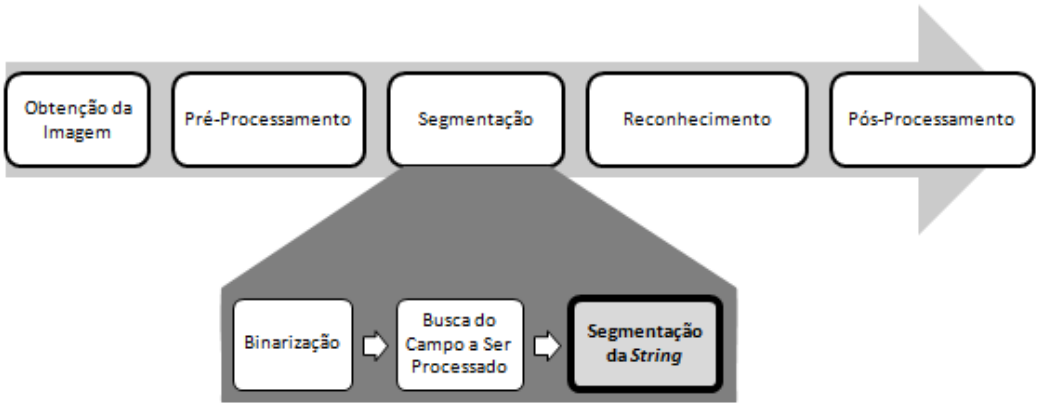


Figura 1.1: Etapas básicas de um sistema de reconhecimento.

com uma câmera de vídeo, digitalizar uma imagem com equipamento específico ou até simplesmente realizar a leitura de um arquivo já contendo a imagem em formato digital.

Figura 1.2: Exemplo de imagem de entrada para um Sistema de Reconhecimento de Dígitos Manuscritos, a qual corresponde à um Cheque Bancário.

Podemos classificar os sistemas de reconhecimento de manuscritos de acordo com o método de entrada utilizado, em dois tipos: *on-line* e *off-line*. *On-line* é o método que realiza a captura em tempo de execução, utilizando para isso, hardware específico, como mesas digitalizadoras ou canetas sensíveis à pressão. Já sistemas *off-line*, são os que processam um manuscrito que foi previamente digitalizado e já esta armazenado em formato de arquivo. No primeiro método, pode-se utilizar além da imagem em si, outras informações adicionais da escrita, como: pontos de início e fim, velocidade, pressão da caneta, dentre outros. Essas informações podem ser utilizadas para aumentar o desempenho do algoritmo de segmentação. Porém em muitos casos, não é possível utilizar um equipamento *on-line*, como no processamento automático de cheques ou envelopes postais.

Já a segunda etapa aplica algoritmos visando eliminar da imagem os componentes que não são importantes, os quais podem prejudicar a segmentação e o reconhecimento, como ruídos oriundos da captura da imagem, além de texturas e marcas d'água que possam estar presentes no fundo da mesma.

Por sua vez, a etapa de segmentação, pode ser sub-dividida basicamente em três partes: binarização da imagem, busca pelo campo a ser processado e segmentação da *string*.

A binarização visa separar na imagem o que é informação importante (texto), do plano de fundo, o qual pode conter outras informações que não puderam ser removidas pelo pré-processamento realizado na imagem. Após esta etapa teremos uma imagem binária (imagem contendo apenas duas cores, geralmente preto e branco) onde todas as informações de interesse serão exibidas por pixels pretos e os demais componentes da imagem serão transformados em pixels brancos. Tendo a imagem binarizada, é necessário localizar o campo a ser processado. Essa busca pode ser realizada de muitas formas, desde algoritmos simples que utilizam apenas a localização do campo a ser reconhecido até o uso de algoritmos de IA (Inteligência Artificial) que analisam características que diferenciam o campo de interesse dos demais. Tendo encontrado o campo de interesse, é necessário então realizar a segmentação da *string* (cadeia de caracteres), fase essa que é o foco deste trabalho. Esta etapa recebe como entrada a imagem contendo a *string* e realiza a separação dos caracteres, no caso deste trabalho, dígitos. Ou seja, teremos após esta fase, várias imagens, cada uma contendo apenas um dígito.

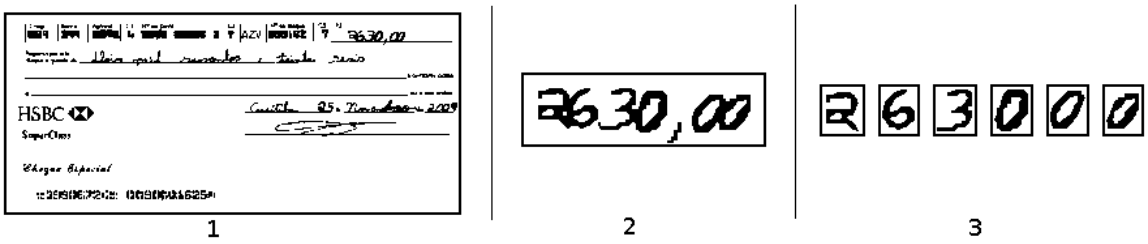


Figura 1.3: Imagens resultantes de cada uma das três sub-etapas da Segmentação. 1 - Binarização; 2 - Busca pelo campo de interesse; 3 - Segmentação da *string*

Após a segmentação, as imagens dos caracteres isolados são enviados para a etapa de reconhecimento, a qual é responsável por transformar uma imagem contendo um único componente, em um caracter ASCII. De posse do resultado do reconhecimento, este passa para a etapa de pós-processamento, a qual visa corrigir erros que possam ter ocorrido em etapas anteriores. Diferentes tipos de processamento podem existir nesta etapa. Para exemplificar, podemos citar a análise com dicionário de contexto, ou seja, tendo um sistema para reconhecimento de palavras manuscritas, por exemplo, pode-se utilizar um dicionário contendo todas as possíveis palavras (desde

um pequeno grupo, até um dicionário completo de determinado idioma) e com isso, detectar e corrigir pequenos erros da segmentação.

Resumindo ainda mais estas etapas, podemos falar que este tipo de sistema processa uma imagem de entrada (Figura 1.2) de modo à transformá-la em uma sequência de caracteres ASCII contendo a informação de interesse contida na mesma.

O escopo deste trabalho, são os algoritmos de segmentação de pares de dígitos manuscritos.

1.1 Definição do Problema

A segmentação é considerada a fase mais crítica em sistemas de reconhecimento de dígitos manuscritos, isso se dá tanto pela complexidade desta fase, quanto ao fato de ser na maioria dos casos crucial, pois um erro nessa fase geralmente acarreta um erro no reconhecimento.

Esta etapa vem sendo pesquisada há muito tempo e diversos algoritmos já foram desenvolvidos. O objetivo da segmentação é dividir a imagem de entrada em regiões que possuam uma e apenas uma entidade completa. No caso do reconhecimento de dígitos manuscritos, essa entidade seria um dígito. Então, para dividir uma imagem contendo uma cadeia de dígitos em imagens de dígitos isolados, precisamos encontrar os chamados "caminhos de segmentação" (Figura 1.4), que nada mais são que "cortes" que se realizados na imagem dividem a mesma, fazendo com que cada uma passe a ter apenas um dígito.

Figura 1.4: Exemplos de pares de dígitos e seus caminhos ótimos de segmentação.

A escolha da melhor abordagem para a segmentação de dígitos depende de um conjunto de variáveis, as quais necessitam de um profundo estudo para análise de qual pode ser mais eficaz para cada caso. Casei et al [CL96] propõem a classificação dos métodos de segmentação em dois tipos: segmentação implícita e explícita. Basicamente, a segmentação explícita, é aquela realizada antes do reconhecimento, tipo este correspondente a todos os algoritmos que serão abordados neste projeto. Já a segmentação implícita é realizada simultaneamente com o reconhecimento, sendo então a segmentação, o resultado do mesmo. Cita-se como exemplo de segmentação implícita

o método baseado em modelos ocultos de Markov (HMM) proposto por Britto et al [BSBS03].

Como mencionado no parágrafo anterior, para escolha do método de segmentação devemos levar em consideração vários fatores. O primeiro é o tipo de imagem de entrada. As representações mais comuns são: caracteres sólidos binários, esqueleto da imagem, imagem em nível de cinza e contorno. Chama-se de imagem binária a imagem contendo apenas duas cores, uma para o plano frontal (no caso o dígito) e outra para o fundo, de maneira geral utiliza-se: preto e branco, respectivamente. Caractere sólido entende-se por um caractere não vazado. Como representações não-sólidas podemos citar esqueleto e contorno da imagem.

Figura 1.5: Caractere sólido, imagem do contorno e do esqueleto, respectivamente.

Imagens coloridas dificilmente são utilizadas para este tipo de aplicação, devido à maior complexidade de processamento. De modo que até o momento nenhum método proposto demonstrou um ganho em desempenho que justificasse esta complexidade. Desta maneira, de modo geral transforma-se as imagens coloridas para uma das representações citadas anteriormente antes de realizar o processamento. Por outro lado, a transformação para uma maneira muito simplificada, como a imagem binária, dependendo do caso, pode resultar em uma imagem com qualidade insuficiente para uma boa segmentação. Existem também métodos que utilizam mais de uma representação [AYV98].

Outro ponto crucial dos métodos de segmentação são as características utilizadas. Ou seja, para descobrir os possíveis caminhos de segmentação, é necessário extrair da imagem algumas características, as quais quando analisadas, forneçam informações que possam ajudar na localização dos possíveis caminhos de segmentação. Como exemplos de características podemos citar: relação entre largura e altura da imagem, espessura do traço, pontos de máximo e mínimo no contorno da imagem, perfil da imagem, densidade de pixels em cada região da imagem (supondo uma divisão da mesma em várias regiões), pontos de conexão e pontos terminais ambos no esqueleto da imagem, dentre diversas outras características existentes.

O tipo de documento a ser tratado também é um fator importante na escolha do método a ser utilizado. Devemos levar em conta o nível de degradação que a imagem dos dígitos pode vir a apresentar, a variabilidade de fontes a ser considerada e o fato de podermos ter que tratar de manuscritos não delimitados. Manuscritos delimitados

são imagens nas quais existem áreas determinadas para a escrita, como por exemplo, formulários nos quais cada caractere tem um espaço pré-determinado. A diferença entre caracteres delimitados e não delimitados é representada pela Figura 1.6, na qual a sequência de dígitos 04618-003 correspondente a um código de endereçamento postal, é considerada uma sequência de dígitos manuscritos delimitados. Já o restante do texto é dito não-delimitado, pois não possui nenhuma marcação limitando o posicionamento dos caracteres.

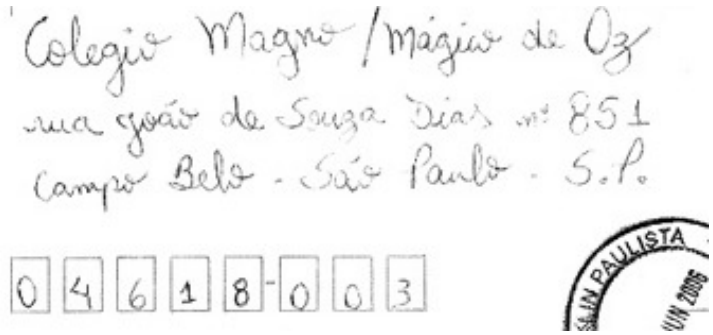


Figura 1.6: Exemplos de manuscritos delimitados e não-delimitados.

Não se deve esquecer também de observar se o método a ser escolhido é invariante as distorções que os documentos em questão podem vir a apresentar. Distorções estas que podem ser inclinações, rotações, dentre outras.

Diante de tamanha variabilidade encontrada nos diferentes casos onde se pode aplicar um sistema para reconhecimento de manuscritos, e também da diversidade de métodos existentes, a escolha da estratégia a ser utilizada em cada caso acaba se tornando também um problema. Baseado nesta dificuldade, fundamentamos este projeto, pois com ele pretendemos obter argumentos para a escolha do método a ser utilizado.

1.2 Objetivo

O objetivo principal deste trabalho é realizar um estudo comparativo de métodos de segmentação de pares de dígitos manuscritos, buscando caracterizar os tipos de conexão em que cada método apresenta melhor desempenho.

Já como objetivos específicos podemos citar:

- Definição de um protocolo experimental para comparação dos métodos de segmentação.
- Rotulação da base de dados de acordo com o tipo de conexão existente em cada imagem.

- Comparação dos métodos de acordo com suas características de funcionamento e desempenho obtido, dados esses que serão necessários em um trabalho futuro de desenvolvimento de um sistema de seleção dinâmica de algoritmos de segmentação.

1.3 Justificativa

Mesmo com toda pesquisa envolvida neste tema, os algoritmos de segmentação ainda estão longe da perfeição e são, na maioria dos casos, os responsáveis pela maior parte dos erros ocorridos em sistemas de reconhecimento de manuscritos. Por isso, consideramos a caracterização de métodos de segmentação visando uma futura combinação dos mesmos, um estudo bem fundamentado e de valor científico, visto que poderemos apenas combinando algoritmos já existentes, otimizar o desempenho final da fase de segmentação. Outro ponto que justifica e valoriza este trabalho está na dificuldade de se realizar comparações entre os algoritmos de segmentação existentes, devido a falta de um padrão para realização dos testes. Neste trabalho, realizamos uma comparação entre os diferentes algoritmos de forma justa, pois os testes foram realizados utilizando para todos os algoritmos testados, a mesma base de dados e os resultados avaliados com uma mesma rede neural.

1.4 Proposta

O foco deste trabalho é a segmentação de pares de dígitos conectados. Esta decisão foi baseada nos dados mostrados por Wang et al [WGS00], pois nesse artigo, os autores fazem uma análise e concluem que 85% dos casos de conexão encontrados correspondem a pares de dígitos conectados. Para realizar a avaliação dos algoritmos, foi utilizada uma base sintética de dígitos conectados proposta por Oliveira et al [OBJS05]. Esta base é dita "sintética", pois foi gerada automaticamente com base em imagens de dígitos isolados da base NIST. Como esta base possui uma quantidade muito grande de imagens, foi utilizada uma parte dela, parte esta que contém 79.466 imagens. Em uma primeira etapa estas imagens fora classificadas visualmente de acordo com o tipo de conexão existente. Foram então implementados e avaliados 6 diferentes algoritmos de segmentação e para dizer se os algoritmos obtiveram sucesso na segmentação de cada imagem, foi utilizado o classificador proposto por Oliveira [SO03], o qual se baseia em uma rede neural MLP(*Multi Layer Perceptron*). Os algoritmos foram avaliados tanto em relação ao desempenho global da segmentação, como em relação ao desempenho específico para cada tipo de conexão.

1.5 Contribuição

A principal contribuição deste trabalho é a caracterização de diferentes algoritmos de segmentação, a qual foi realizada considerando uma característica do próprio problema, que é o conhecimento a priori do tipo de conexão. Esta caracterização é uma etapa fundamental para que os algoritmos possam ser utilizados em um sistema de seleção dinâmica de algoritmos de segmentação onde o critério de escolha seria o tipo de conexão encontrado na imagem de entrada. Além dessa contribuição, que motivou nosso projeto, outras contribuições secundárias foram alcançadas. Dentre elas, cita-se o fato de os algoritmos terem sido avaliados sobre as mesmas condições experimentais, o que gerou uma comparação justa entre os mesmos. Essa comparação era uma necessidade para evolução das pesquisas na área, visto que comparações existentes utilizam diferentes bases de dados e métodos de avaliação, tornando-as tendenciosas. Podemos comentar também a classificação visual da base de dados, pois mais de 79 mil imagens foram classificadas visualmente de acordo com o tipo de conexão, e esta base de dados poderá ser utilizada em trabalhos futuros.

1.6 Organização

Esta dissertação de mestrado está organizada em cinco capítulos. Neste primeiro capítulo foi apresentada uma introdução ao tema proposto. O Capítulo 2 traz algumas definições que são citadas neste trabalho, como segmentação, classificador, e vários tipos de características utilizadas pelos algoritmos, além de um apanhado geral com os principais métodos de segmentação que foram pesquisados, dentre estes estão os que selecionamos para utilizar em nosso trabalho de caracterização, todos os métodos contém uma breve análise de suas particularidades e desempenho reportados pelos respectivos autores. No Capítulo 3 é descrita a metodologia adotada para o desenvolvimento do trabalho. Já no Capítulo 4 são exibidos os resultados obtidos ao longo deste trabalho, juntamente com a análise comparativa dos mesmos. Por último, no Capítulo 5, apresentamos a conclusão bem como os trabalhos futuros.

Capítulo 2

Estado da Arte

Neste capítulo apresentamos com mais detalhes a segmentação de dígitos manuscritos. Essa é uma etapa muito importante em sistemas de reconhecimento de dígitos manuscritos e é o foco deste trabalho. Nas seções seguintes, além do conceito de segmentação, são descritos alguns dos principais métodos existentes na literatura, destacando suas particularidades e comparando os desempenhos obtidos nos testes realizados pelos autores.

2.1 Segmentação

Em sistemas de reconhecimento de manuscritos, a etapa de pré-processamento da imagem influencia diretamente a fase de segmentação. E a segmentação acaba por influenciar o reconhecimento, pois em casos de imagens mais complexas o pré-processamento muitas vezes não consegue eliminar totalmente os ruídos e componentes não desejados da imagem, como texturas presentes no fundo ou marcas d'água. Já o reconhecimento é prejudicado quando a fase de segmentação falha, enviando nesses casos ao classificador, imagens contendo mais de um dígito ou contendo apenas algumas partes de um dígito. O algoritmo base utilizado pela etapa de reconhecimento (Figura 1.1) em um sistema de reconhecimento de manuscritos é chamado de classificador. Este algoritmo pode ser por exemplo, uma rede neural, um SVM, dentre outros.

O objetivo da segmentação é processar uma imagem de entrada contendo um ou mais caracteres e dividi-la em várias imagens, de forma que cada uma contenha apenas um caractere. Para realizar essa divisão da imagem, o algoritmo busca os prováveis "caminhos de segmentação", esses nada mais são que "cortes" realizados na imagem visando dividi-la. Alguns métodos de segmentação optam por uma estratégia conhecida por *over-segmentation* ou sobre-segmentação, a qual consiste em gerar mais "caminhos de segmentação" do que o necessário, e depois disso com a ajuda de um classificador, escolher a melhor hipótese de segmentação. Na Figura 2.2 é exibido um grafo que

representa as 16 hipóteses de segmentação que foram geradas pela combinação dos quatro caminhos representados na Figura 2.1.

Figura 2.1: Imagem contendo dois dígitos conectados e os quatro caminhos de segmentação encontrados na tentativa de segmentá-los.

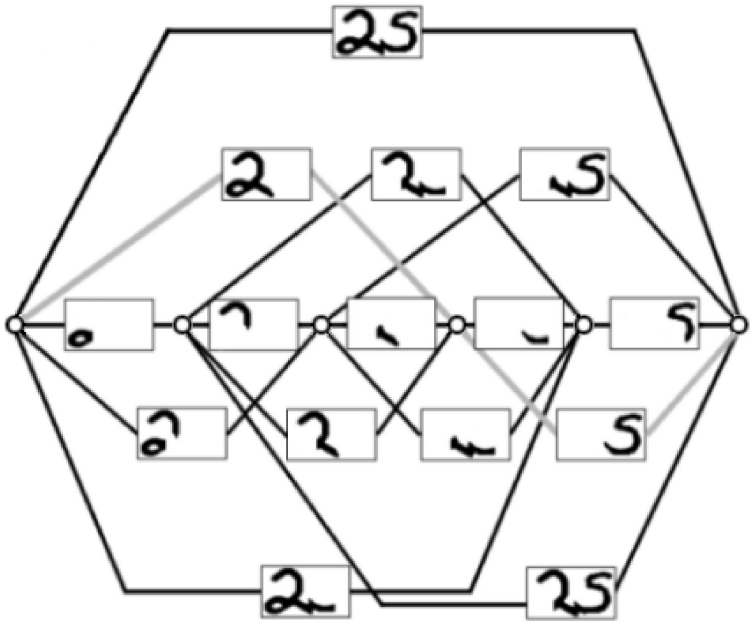


Figura 2.2: Exemplo onde a combinação de 4 caminhos de segmentação geraram um grafo com 15 sub-imagens e 16 hipóteses de segmentação (o caminho destacado no grafo, representa a melhor hipótese).

O problema de se utilizar um método baseado em *over-segmentation* é que para cada sub-imagem gerada, é necessária uma chamada ao classificador, e estas chamadas são computacionalmente caras. No exemplo da imagem acima, necessitaríamos de 15 chamadas ao classificador para avaliar qual seria a melhor dentre as 16 hipóteses de segmentação.

Para calcular a quantidade de hipóteses de segmentação, pode-se utilizar a equação 2.1, na qual n é o número de sub-imagens formadas quando a imagem é segmentada por todos os caminhos de segmentação encontrados. No caso do exemplo acima teríamos $n = 5$.

$$QtdHip = 2^{(n-1)} \tag{2.1}$$

De acordo com a classificação proposta por Casei et al [CL96], os métodos de segmentação podem ser divididos em dois tipos, segmentação implícita e explícita. A segmentação explícita é aquela realizada antes do reconhecimento e que utiliza características da imagem para determinar os pontos que serão utilizados para segmenta-la. Para exemplificar essas características utilizadas, podemos citar: informações de curvatura no traçado, pontos de máximo e mínimo locais (também chamados de colina e vale)(Figura 2.3), pontos no esqueleto (Figura 2.4), perfil da imagem, dentre outras. Já a segmentação implícita é realizada simultaneamente com o reconhecimento, sendo então a segmentação, nada mais que o resultado do mesmo. Esse tipo de algoritmo determina o caminho de segmentação de forma arbitrária, geralmente cortando a imagem verticalmente de ponta à ponta, não utilizando nenhum tipo de característica utilizada na segmentação explícita. Podemos citar como exemplo de segmentação implícita o método baseado em modelos ocultos de Markov (HMM) proposto por Britto et al [BSBS03]. A grande desvantagem desse tipo de método é o maior custo computacional, pois é necessário um grande número de chamadas ao classificador.

Figura 2.3: Dezena "38" e as representações de seu vale mais profundo (profundidade P_v) e sua colina mais alta (altura A_c).

Figura 2.4: Imagem original à esquerda. À direita imagem esqueletizada com os respectivos pontos de característica do esqueleto (pontos finais e pontos de intersecção)

A segmentação é necessária pois a maior parte dos classificadores utilizados em sistemas de reconhecimento classifica apenas dígitos de forma isolada, ou seja, não consegue classificar uma imagem se esta possuir dois ou mais dígitos conectados. Existem também na literatura, métodos para o reconhecimento de manuscritos nos quais a segmentação não é necessária e pares de dígitos conectados são enviados diretamente

ao classificador, como exemplo deste tipo de método temos o proposto por Choi e Oh [CO99], o qual utiliza um classificador que possui 100 classes na saída (00 à 99). Como problemas deste tipo de método temos: a dificuldade de treinar um classificador com tão alto número de classes e que possua bom desempenho, além do alto custo computacional do mesmo.

A maioria dos métodos de segmentação utiliza o conceito de Componentes Conectados (CCs) em seu algoritmo. Um CC nada mais é que um conjunto de pixels nos quais a partir de qualquer ponto, é possível chegar a qualquer outro ponto, percorrendo somente pontos deste mesmo conjunto, isto é, se não existir pelo menos um caminho entre quaisquer dois pontos deste conjunto não podemos dizer que estes pontos pertencem ao mesmo CC.

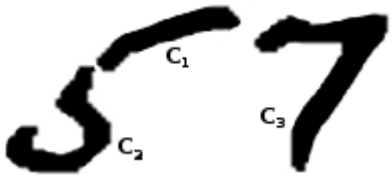


Figura 2.5: Imagem contendo dois dígitos, "5" e "7", compostos por três CCs C_1, C_2 e C_3).

Veremos nas seções seguintes que alguns métodos realizam remoção dos chamados "Traços-Ligadores". Esse tipo de traço aparece em algumas imagens de dígitos conectados e se caracterizam por serem traços que não pertencem a nenhum dos dígitos da imagem. Estes são originários do modo de escrita de determinadas pessoas, as quais por descuido proveniente da velocidade da escrita, acabam por deslizar a caneta sobre o papel produzindo um traço que liga dois dígitos consecutivos e não pertence a nenhum dos mesmos.

Esta seção apresentou uma explanação geral sobre a segmentação de dígitos manuscritos e alguns conceitos principais.

2.2 Algoritmos de Segmentação

Um grande número de algoritmos de segmentação existe na literatura atualmente. Nesta seção apresentamos vários destes algoritmos, descrevendo seu funcionamento básico, terminando com uma tabela comparativa das características dos mesmos.

2.2.1 Fujisawa et al

Fujisawa et al [FNK92] propõem um método de segmentação baseado em reconhecimento, ou seja, o algoritmo utiliza um classificador para selecionar uma dentre as hipóteses de segmentação geradas.

Basicamente o algoritmo num primeiro passo identifica através de perseguição de contorno, todos os componentes conectados (CCs) presentes na imagem. Então estes CCs são classificados em dígitos isolados ou dígitos conectados, através de dois limiares baseados na largura comum de dígitos isolados. Caso todos os CCs sejam classificados como dígitos isolados, então a segmentação não é necessária.

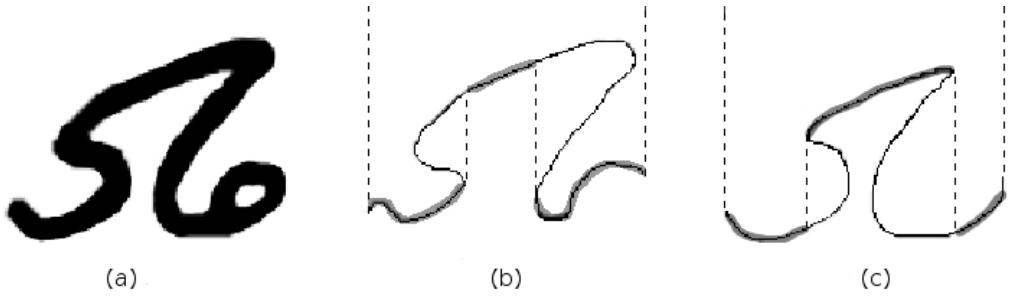


Figura 2.6: a) Imagem de Entrada; b) Contorno Superior; c) Contorno Inferior. Sombreados no contorno, os pontos selecionados para cada coordenada x. Figura adaptada de [FNK92]

Quando o algoritmo encontra um CC correspondente a dígitos conectados, é necessária então a segmentação, para isso, o contorno é dividido horizontalmente em duas partes: contorno superior e contorno inferior, conforme mostrado na Figura 2.6. Esta separação é feita com base nos ponto mais a esquerda da imagem e no ponto mais a direita. Analisando individualmente cada uma das partes (superior e inferior) do contorno, para algumas coordenadas x podem existir mais de um ponto. Então é aplicada uma operação, sobre cada um dos contornos, que faz com que para cada coordenada x exista apenas um ponto no contorno. Para isso, sempre que existir mais de um ponto, é selecionado o ponto mais baixo ou mais alto, para cada um dos contornos superior e inferior, respectivamente. O resultado desta operação é representado pela parte sombreada do contorno dos dígitos, na imagem 2.6. As coordenadas y dos pontos obtidos do contorno superior, são representadas pela função $H_u(x)$, e as obtidas do contorno inferior representadas pela função $H_l(x)$. Com essas funções, determina-se a largura vertical do traço para cada coordenada x ($H(x)$), de acordo com a equação 2.2.

$$H(x) = |H_u(x) - H_l(x)| \tag{2.2}$$

A largura vertical é então comparada com um limiar, visando encontrar as possíveis regiões de toque. Para limitar a região de busca, o autor sugere que seja determinado um intervalo $[X1, X2]$ onde seja mais provavel que esteja a área de toque.

Para as regiões que possuem conexão entre dois ciclos fechados (*"loops"*), é realizado um tratamento diferenciado. O algoritmo extrai os contornos destes ciclos e divide a imagem verticalmente, separando os ciclos existentes em dois grupos: pertencentes ao dígito da esquerda ou ciclos pertencentes ao dígito da direita. Para que este

algoritmo seja válido, a seguinte condição deve ser satisfeita: existe uma distância horizontal (Figura 2.7) mínima entre os ciclos que pertencem ao grupo da esquerda e os ciclos que pertencem ao grupo da direita. Esta distância mínima é dada por um limiar pré-determinado. Caso a distância seja menor do que o limiar, então o CC é tratado como sendo um único componente, o qual possui mais de um ciclo. Poderia ser, por exemplo, o dígito 8. Porém se a distância for maior que o limiar, então ali existe uma região de conexão e os pontos são comparados com pontos obtidos do contorno externo, para se chegar aos pontos de segmentação candidatos. Os caminhos de segmentação são gerados através da ligação de pares de pontos de segmentação com segmentos de reta.

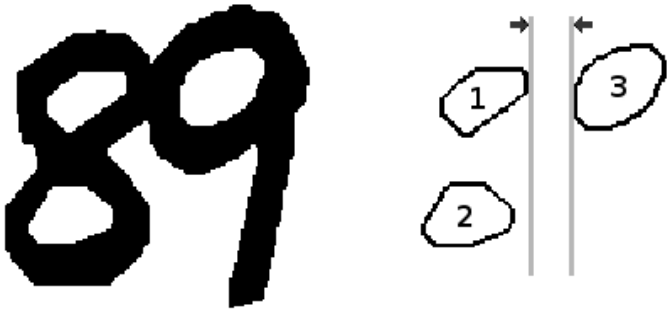


Figura 2.7: Linhas verticais mostrando a distância horizontal dos ciclos (Imagem original à esquerda).

Depois de encontrados os caminhos de segmentação candidatos, a imagem é segmentada utilizando-se todos eles (técnica conhecida como "segmentação em excesso"). Os componentes conectados (CCs) obtidos após a segmentação, são então colocados em uma lista ordenada de acordo com o posicionamento dos mesmos em relação à coordenada x da imagem original. Tendo esta lista formada, é então criado um grafo com as possibilidades de segmentação da imagem, podendo ter desde todos os CCs correspondendo à elementos isolados, até vários componentes consecutivos (segundo a ordem anterior) constituindo um só componente. Para determinar qual o caminho do grafo de hipóteses representa a melhor segmentação que pode ser obtida com base nos pontos gerados, é utilizada uma função de classificação do CC. Esta função adota três comparações de medidas de tamanho dos CCs em relação à alguns limiares. Cada comparação pode retornar três tipos de resultado: Componente precisa ser agrupado; componente precisa ser segmentado ou componente indefinido. Então se realiza uma "votação" sobre os resultados fornecidos por cada uma das comparações. O resultado mais votado é utilizado como sendo o ideal. Os autores propuseram também uma segunda maneira de realizar o teste, tentando reconhecer cada um dos componentes do grafo de hipóteses. Depois, de acordo com os escores de reconhecimento, avalia-se qual dos caminhos possui maior probabilidade de reconhecimento.

A avaliação do algoritmo foi realizada sobre base de dados própria do autor, contendo 46 classes, sendo essas as que mais aparecem nos dígitos conectados obtidos pelos autores com auxílio de um formulário. Para cada uma das classes foram utilizados 20 exemplos. Contabilizando assim 920 imagens de teste. Vale lembrar que esta base de dados não possui casos de conexão múltipla. O resultado obtido foi uma taxa de acerto de 95% com 5% de rejeição.

Como limitações deste método podemos citar os casos nos quais o algoritmo falha ao determinar a largura do traço vertical. Outra limitação que deve ser destacada é que o algoritmo não consegue segmentar casos de conexão múltipla (dígitos que se tocam em mais de um ponto) nem cadeias numéricas contendo três ou mais dígitos conectados. Um ponto negativo do algoritmo é a necessidade de determinação manual de alguns limiares, os quais poderiam necessitar de uma redefinição caso a base de dados utilizada para testar o algoritmo fosse substituída por outra.

Como ponto positivo podemos citar a ausência de algoritmos de alta complexidade e custo computacional, como esqueletização ou operações morfológicas. Podemos citar também, apesar da alta taxa de rejeição (5%), que a taxa de erro é nula para este algoritmo, o que é de extrema importância para um caso de aplicação do algoritmo na solução de problema real.

2.2.2 Shi e Govindaraju

Shi e Govindaraju [SG97] propõem um método que se diferencia dos tradicionais por não determinar os pontos de segmentação unicamente pelo traço que conecta os pares de dígitos, mas por identificar regiões com potencial para serem utilizadas como pontos de segmentação. Essas regiões são determinadas basicamente pela análise da trajetória do traçado dos dígitos. A tarefa de busca do ponto de segmentação ou do traço-ligador não pode ser considerada trivial, principalmente devido à largura do traço não ser uniforme. A esqueletização seria uma boa solução para este problema, entretanto não foi utilizada devido ao alto custo computacional deste tipo de algoritmo.

De acordo com os autores, métodos heurísticos poderiam ser utilizados, porém apesar da eficiência, sua precisão é limitada. Os autores informam também que evitaram o uso de Histogramas verticais, pois, segundo eles, estes são muito propensos a erro. Eles propõem então a utilização de informações do traço, como direção, pontos de curvatura e pontos finais. Estas informações são obtidas da representação de contorno em forma de cadeia de códigos.

Essa cadeia de códigos que representa o contorno da imagem, além das coordenadas do ponto, a inclinação e a curvatura, também possui o tipo do ponto, o ponto oposto correspondente (Figura 2.8) e outro ponto calculado a partir do ponto oposto, o qual é utilizado para calcular a largura do traço e para a reconstrução das imagens

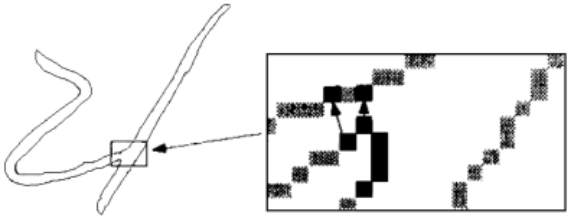


Figura 2.8: Exemplos de pontos de curvatura e respectivos pontos opostos. [SG97]

dos caracteres segmentados.

Os autores puderam observar através de um estudo dos pontos de toque e dos traços-ligadores entre dois dígitos, que o contorno faz curvaturas significantes à direita (Figura 2.9), em cada ponto de toque. O método utilizado para encontrar os pontos de curvatura à direita, utiliza para isso a espessura do traço e o vetor normal. Depois de encontrados esses pontos de curvatura, é utilizado um limiar (o qual deve ser obtido experimentalmente), para determinar se a curvatura é ou não significante.

Figura 2.9: (a) Imagem Original, (b) Pontos de curvatura à direita, (c) Dígitos segmentados. [SG97]

Os pontos significantes de curvatura à direita, junto com seus pontos opostos correspondentes, dividem o contorno em pedaços. Estes pedaços são então classificados como pertencentes ao dígito da esquerda ou da direita. A maneira mais fácil de realizar a classificação é utilizando uma linha de decisão. Esta é uma vertical, centralizada na imagem, a qual divide a imagem em duas partes de igual tamanho. Então se o centro de massa de determinado pedaço do contorno se encontra à esquerda da linha, este pertence ao dígito da esquerda, senão ao da direita. Porém este método falha se os dígitos possuírem uma diferença muito grande de largura. Devido a este motivo outra maneira de determinar a linha de decisão é proposta, a qual utiliza o histograma vertical da imagem.

Na Figura 2.9 podemos visualizar a imagem original, os pontos de curvatura encontrados e o resultado da segmentação.

O método proposto foi testado sobre uma base contendo 1966 imagens do CEDAR, base esta que contém pares de dígitos conectados. Os autores exploram apenas a segmentação de pares de dígitos conectados, não tratando cadeias de três ou mais dígitos conectados. O método obteve 78% de acerto na segmentação. Se para os

casos em que existe um traço-ligador for adicionado um classificador, para auxiliar na tomada de decisão da segmentação, então a taxa de acerto obtida é de 80,5%. A necessidade de determinar experimentalmente o ângulo para que uma curvatura à direita seja considerada "significante" é um ponto negativo deste método.

2.2.3 Oliveira et al

Oliveira et al [OLBS00] propõem um novo método de segmentação para dígitos manuscritos, baseado no método proposto por Fenrich [Fen91]. O método utiliza uma combinação de dois tipos de características estruturais e foi desenvolvido para ser utilizado em um sistema de segmentação baseado em reconhecimento. O algoritmo recebe como entrada uma imagem binária dos dígitos. Como resultado o algoritmo fornece uma lista com os melhores caminhos de segmentação e o número de dígitos a serem segmentados.

O contorno e o perfil encontram-se entre as características mais comumente encontradas na literatura. São então extraídos os mínimos locais do contorno (Figura 2.10a) e do perfil da imagem (Figura 2.10b).

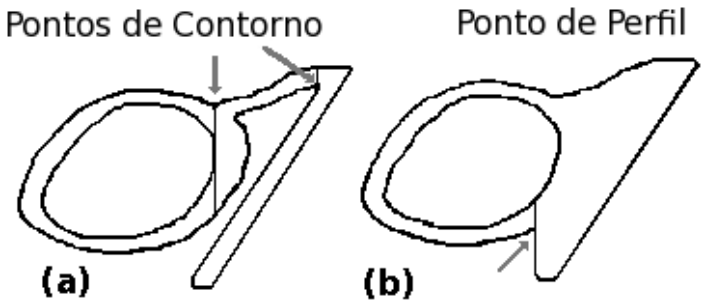


Figura 2.10: Pontos de (a) contorno e (b) perfil. Figura adaptada de [OLBS00]

O algoritmo busca relações entre todos os pontos encontrados na tentativa de agrupá-los de modo que sejam formadas várias hipóteses de segmentação. Para avaliar as várias hipóteses, foi utilizada uma rede neural treinada com uma base de dados contendo 8500 imagens de dígitos manuscritos isolados. O melhor experimento de aprendizagem resultou em uma taxa de 99,9% no treinamento e 98,5% no teste.

O sistema proposto foi então testado sobre 900 imagens de dígitos conectados, extraídas de 2000 imagens de cheques bancários brasileiros. Vale lembrar que o algoritmo só se aplica a pares de dígitos conectados. A melhor hipótese de segmentação é determinada pelo maior valor de produto dos escores dos elementos segmentados que compõem a hipótese. Estes escores são atribuídos pela rede neural. Em uma primeira analise visual dos resultados conclui-se que 98,5% dos dígitos foram corretamente segmentados. Em um segundo momento, o algoritmo foi avaliado com o uso de uma rede

neural para classificar os dígitos segmentados. Nesta segunda avaliação, a taxa de acerto foi de 90,8%. Dentre o percentual de erro de 9,2% verificou-se que 1,5% era erro de segmentação, e 7,7% era erro de reconhecimento. Esse erro referente à segmentação pode ser diminuído se a rede neural for treinada com dígitos provenientes da saída de um algoritmo de segmentação, e não apenas dígitos naturalmente isolados.

A técnica de segmentação desenvolvida em [OLBS00] possui um algoritmo simples e com poucas regras. O algoritmo atingiu bons resultados e realizou uma boa segmentação na maioria dos dígitos conectados, mesmo para aqueles em que os dígitos estavam sobrepostos ou inclinados.

Como ponto negativo deste algoritmo, podemos destacar o fato de que na maioria dos casos este irá gerar mais de uma hipótese de segmentação. Para escolher a melhor dentre as hipóteses, seria necessário o uso de um classificador, como uma rede neural. Porém o uso de classificador para seleção da melhor hipótese é um processo computacionalmente caro.

2.2.4 Chen e Wang

Chen e Wang [CW00] propõem uma abordagem para segmentação de cadeias de pares de dígitos conectados. Esta abordagem visa à segmentação tanto de dígitos simplesmente conectados, quanto dígitos com múltiplas conexões. O método combina características tanto do primeiro plano da imagem (dígito propriamente dito) como do plano de fundo. Os pontos de segmentação são obtidos com a análise dos esqueletos da imagem, esqueleto dos dígitos e do fundo (Figura 2.11). Vários caminhos de segmentação são gerados e os traços-ligadores removidos.

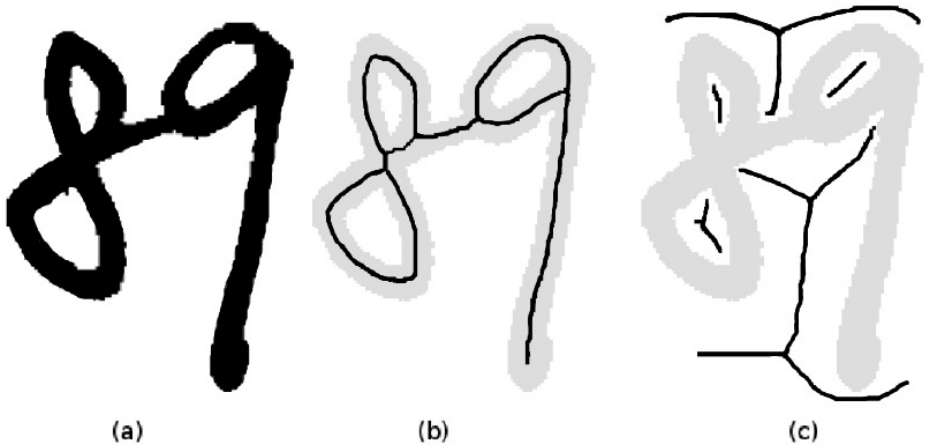


Figura 2.11: (a) Imagem Original. (b) Esqueleto dos Dígitos. (c) Esqueleto do Fundo. Imagem adaptada de [CW00]

Grande parte dos métodos encontrados na literatura apresenta dificuldade para segmentar dígitos com conexão múltipla ou dígitos com conexão simples que possuem

uma grande área de toque. Segundo os autores, este método conduz a bons resultados nesses casos, se comparado a outros métodos existentes.

As conexões entre dígitos são classificadas pelos autores em cinco tipos (Figura 2.12). De modo geral, os algoritmos que trabalham com o plano de fundo ou plano principal, ambos isoladamente, segmentam com sucesso conexões dos tipos 1 e 2. Porém apresentam dificuldade com conexões dos tipos 3, 4 e 5. Por isso neste método, características de ambos os planos são extraídas, pois se espera com isso, melhorar o desempenho em relação aos demais métodos que utilizam apenas um dos tipos de característica.

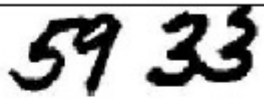
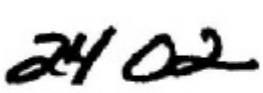
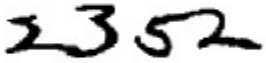
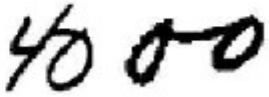
Categoria	Tipo	Tipo de Conexão	Exemplo
Conexão simples	1		
	2		
	3		
	4		
Conexão múltipla	5		

Figura 2.12: Cinco possíveis tipos de conexão, segundo Chen et al [CW00].

Os pontos de características extraídos dos esqueletos da imagem são: pontos de bifurcação, de curvatura e terminais. O algoritmo combina então essas características extraídas tanto dos dígitos como do fundo, para gerar os possíveis caminhos de segmentação.

É feita uma classificação dos caminhos de segmentação, utilizando-se uma função de mistura de probabilidades Gaussianas. Esta função foi obtida com base em 823 imagens da base NIST SD19[1]. Então se a probabilidade Gaussiana do caminho de segmentação melhor classificado for maior que um valor pré-determinado, este é tido como o melhor caminho de segmentação. Caso contrário, este caminho é rejeitado.

Após a identificação de todos os possíveis caminhos de segmentação é feita a remoção dos traços-ligadores quando os mesmos existem. Esses traços são caracterizados por dois caminhos de segmentação com mesmo início e fim, além disso, algumas

[1]http://www.nist.gov/srd/nistsd19.htm

heurísticas devem ser satisfeitas para que esse tipo de traço seja realmente caracterizado. A Figura 2.13 ilustra uma imagem contendo traço-ligador, sua identificação através dos caminhos de segmentação com início e fim em comum e a imagem final após removido o traço.

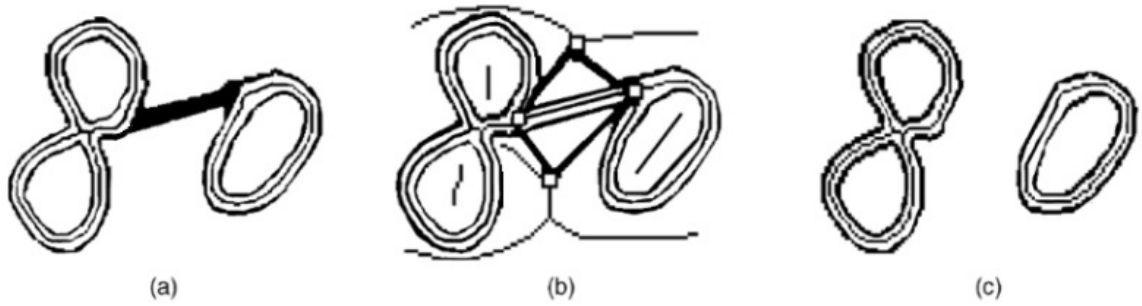


Figura 2.13: (a) Imagem contendo traço-ligador, (b) identificação do traço, (c) imagem após a remoção do mesmo. [CW00]

Para testar o desempenho do método foram utilizadas 4500 cadeias de dois dígitos, destas, 4178 foram extraídas da base NIST SD19, as 322 imagens restantes foram coletadas pelos próprios autores. No experimento foram separadas 832 imagens para determinar a função de mistura de probabilidades Gaussianas, e o restante utilizado para testar o algoritmo de segmentação. Das imagens utilizadas no teste, 8% foram rejeitadas, e se considerarmos somente as imagens aceitas, é obtida uma taxa de acerto de 95,7%. As principais razões causadoras de rejeição foram: quando o centro do caminho de segmentação está muito deslocado em relação ao centro da imagem ou quando a largura de um dos dígitos separados é muito maior que do outro.

Como pontos positivos do método podemos citar o bom desempenho em alguns casos específicos que dificilmente são segmentados por outros métodos presentes na literatura, como por exemplo, conexões nas quais a área de toque é muito grande, ou imagens com mais de dois pontos de conexão. Além da boa capacidade de remoção dos traços-ligadores. O principal ponto negativo do método é o número muito alto de caminhos de segmentação gerados, os quais aumentam bastante o tempo de processamento e impossibilitam o uso do mesmo em aplicações reais.

2.2.5 Yu e Yan

Yu e Yan [YY01] desenvolveram um método capaz de segmentar cadeias de dígitos, ou seja, cadeias que podem possuir mais de dois dígitos. O método funciona segmentando a cadeia em várias sub-cadeias de dois dígitos e processando cada uma delas separadamente.

O método utiliza características morfológicas estruturais. São obtidos os pontos

de características estruturais na imagem de bordas suavizada. Um ponto estrutural é o ponto que define uma mudança morfológica no contorno. São definidos 16 diferentes padrões de pontos estruturais (Figura 2.14). A imagem é dividida em quatro regiões. Durante a análise dos pontos estruturais é levado em conta além do padrão do ponto, a região que este se encontra. Também são extraídas características das concavidades da imagem. Com base nestas características foram construídos diversos modelos, os quais são representados por sequências de pontos estruturais. Então com esses modelos são construídas as diversas heurísticas.

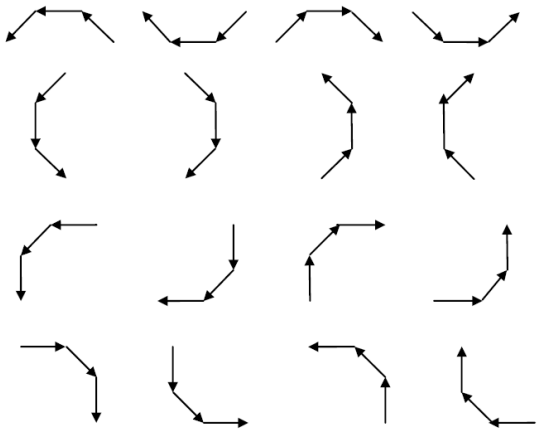


Figura 2.14: Diferentes padrões estruturais que representam as possíveis mudanças morfológicas no contorno da imagem. [YY01]

Para construir os modelos foram utilizadas 600 imagens da base NIST. O conjunto de teste era constituído de 3287 imagens de cadeias constituídas de dois dígitos, para as quais, a taxa de segmentação alcançada foi de 94,8%. Para as 256 imagens restantes, as quais eram constituídas de três dígitos, a taxa de segmentação correta foi de 84,7%. Para os testes de desempenho foi utilizado um k-NN otimizado, de modo que se ambos os dígitos segmentados são reconhecidos por este k-NN, então a segmentação é considerada correta. Os autores enfatizam que podem existir casos em que uma boa segmentação possa ter sido classificada como incorreta devido a um erro de classificação do dígito por parte do k-NN.

O uso do k-NN pode ser visto como um ponto negativo deste método, visto a quantidade e disposição dos pontos utilizados pode o deixar computacionalmente caro, além de precisar armazenar todos os dados utilizados no treinamento e também não fornecer uma saída probabilística.

2.2.6 Pal et al

Pal et al [BCP03] propõem um algoritmo de segmentação livre de reconhecimento (não utiliza um classificador para a validação dos pontos de segmentação). O

algoritmo se baseia na idéia principal de que se dois dígitos se tocam, é formado entre eles um grande espaço, chamado pelos autores de "reservatório" (Figura 2.15). Os pontos de segmentação tendem a estar na base destes reservatórios. Sendo assim, temos uma redução da área de busca dos pontos, o que contribui para um menor esforço computacional.

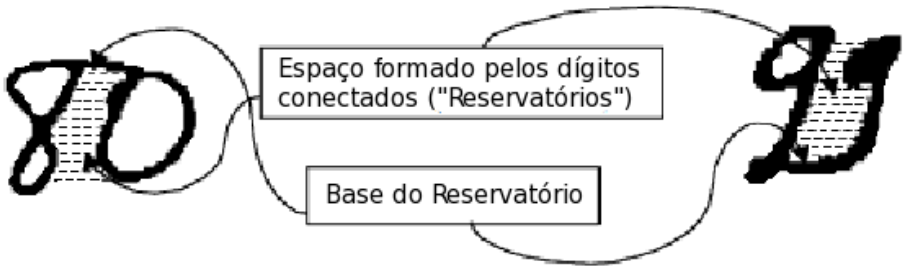


Figura 2.15: Espaços ("Reservatórios") criados pela conexão entre os dígitos, e suas respectivas bases. Adaptada de [BCP03]

Para iniciar a busca pelos pontos de segmentação, é necessário determinar o melhor reservatório. Para isso, a imagem é dividida em regiões, como mostra a Figura 2.16. É também calculado o Centro de Gravidade (CG) de cada um dos reservatórios existentes na imagem. O maior reservatório que possuir o seu CG na região vm da imagem, é então chamado de "Melhor Reservatório". O algoritmo realiza alguns cálculos utilizando a posição da base do "Melhor Reservatório" e as coordenadas das regiões da imagem e determina a posição da área de toque, que pode ser: superior, central ou inferior.

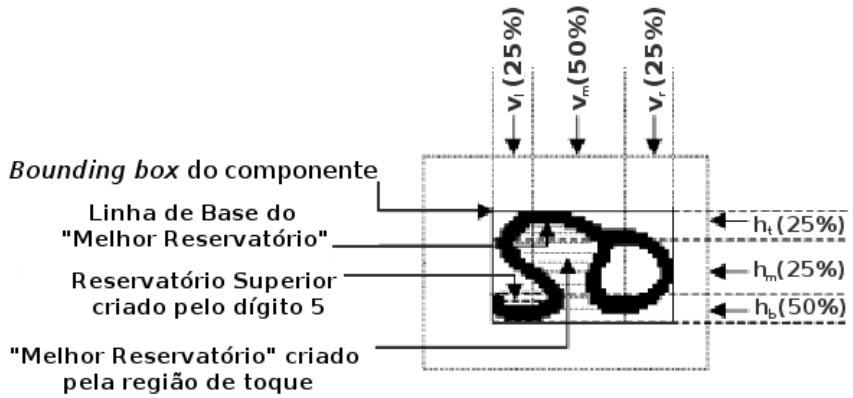


Figura 2.16: Regiões da imagem são utilizadas para determinar o melhor reservatório. Adaptada de [BCP03]

De acordo com a região de toque, o algoritmo determina quais reservatórios serão utilizados na busca pelos pontos de segmentação. Para essa busca, o método utiliza características como: quantidade de reservatórios, suas posições em relação à

Bounding Box do componente, seus tamanhos e formas, centro de gravidade, posição relativa e padrão morfológico da área de conexão. Também são utilizadas características extraídas dos ciclos fechados, são elas: posição, centro de gravidade e a razão $\frac{AlturadoCiclo}{AlturadoComponente}$.

Dependendo do tipo de conexão existente, o caminho de segmentação gerado pode ser de dois tipos: uma curva ou um segmento de reta. O caminho em forma de curva é utilizado quando a área de conexão se encontra entre dois ciclos fechados, de modo que esse caminho é realizado em uma posição central em relação aos ciclos. Nos demais casos, um segmento de reta é utilizado.

Como pontos positivos deste método, podemos apontar o funcionamento independente da inclinação da imagem, suporte a conexões simples e múltiplas e a não utilização de algoritmos de alto custo computacional, como por exemplo, esqueletização da imagem. Outro ponto positivo é o fato do método gerar sempre apenas uma hipótese de segmentação, descartando assim o uso de algoritmos mais complexos, como SVM ou Redes Neurais para a escolha da melhor hipótese de segmentação.

Figura 2.17: Exemplos de segmentação incorreta gerada pelo método proposto. [BCP03]

Um diferencial deste algoritmo é que antes de tentar segmentar ele classifica os dígitos da imagem em isolados ou conectados. Esta classificação possui uma taxa de acerto de 98,81%. Como problemas deste algoritmo temos o fato de não conseguir segmentar imagens que contenham falha no contorno do dígito em um ponto que constitua a parede de um reservatório e também o fato de suportar apenas cadeias de dois dígitos. Dois exemplos de segmentação incorreta deste algoritmo podem ser observados na Figura 2.17. Os autores testaram o método sobre uma base de dados contendo 2250 pares de dígitos conectados extraídos de cheques bancários franceses. Neste teste, o algoritmo alcançou 94,8% de acerto com uma taxa de rejeição de 3,4%.

2.2.7 Elnagar e Alhajajj

Elnagar e Alhajajj [EA03] propõem um método para segmentação de pares de dígitos simplesmente conectados. O método utiliza características de esqueleto e contorno para encontrar os possíveis pontos de segmentação. Para determinar os pontos com maior potencial, o método utiliza heurísticas baseadas no posicionamento destes pontos com relação aos pontos de máximo e mínimo locais.

Antes da extração dos pontos de características a imagem é normalizada quanto à inclinação, linha de base e tamanho. Quanto ao tamanho, a imagem é redimensionada para 30 x 60 pixels, independente do fato de ter dimensões originais maiores ou menores que estas. Após este processamento inicial a imagem é então esqueletizada (Figura 2.18). O processo de esqueletização da imagem é caro computacionalmente, porém, os autores justificam seu uso argumentando que de posse de uma imagem com traços uniformes (neste caso, traço com um pixel de espessura), a extração de características torna-se mais simples.

Figura 2.18: Imagens das fases do método de segmentação: imagem original, imagem esqueletizada, imagem segmentada e imagem restaurada. [EA03]

Para a extração de características são utilizadas máscaras de tamanho 3x3 (Figura 2.19). Estas máscaras são deslizadas sobre a imagem em todas as suas rotações múltiplas de $\frac{\pi}{2}$. Desta forma são então obtidos os pontos finais, pontos de ramificação e de cruzamento. Outro ponto é encontrado obtendo-se o ponto mais alto do histograma do eixo y. Depois de encontradas as características, é então utilizado um processo de remoção de ruído, também baseado em uma máscara 3x3. Este processo visa à remoção dos pontos de características redundantes.

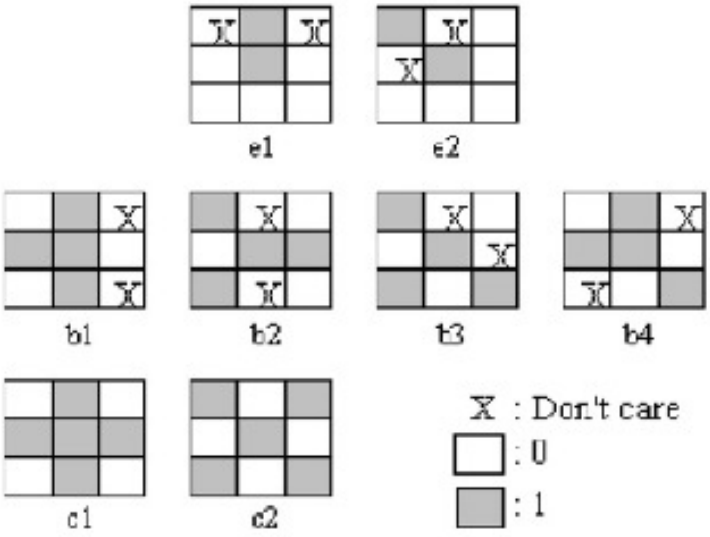


Figura 2.19: Modelos utilizados na extração de características. Pontos Finais (e1;e2), pontos de ramificação (b1;b2;b3;b4) e pontos de cruzamento (c1;c2). [EA03]

O método foi desenvolvido para trabalhar com imagens de dígitos simplesmente

conectados. As conexões simples foram classificadas em quatro tipos:

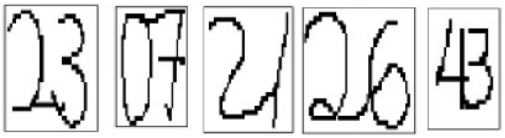


Figura 2.20: Conexões com ponto em comum. [EA03]

1. Conexão com ponto em comum (Figura 2.20): conexão onde os dígitos compartilham um ponto. Ocorre geralmente em pontos de ramificação ou cruzamento, no esqueleto da imagem.

2. Conexão com segmento em comum (Figura 2.21): neste tipo os dígitos compartilham parte de um traço (contorno) da imagem. É identificado por uma combinação de pontos de ramificação e/ou cruzamento.

Figura 2.21: Conexões com segmento em comum. [EA03]

3. Conexão suave (Figura 2.22): é a conexão onde os dígitos compartilham um traço, ou seja, a conexão é tão suave que não é encontrado nenhum ponto de característica na região de toque.

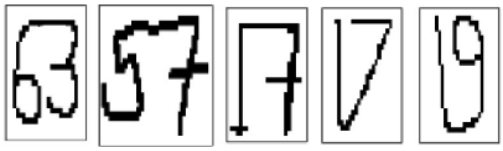


Figura 2.22: Conexões suaves. [EA03]

4. Conexão com Traço-Ligador (Figura 2.23): tipo onde os dígitos se conectam por um traço "extra"presente na imagem, de modo que este traço precisa ser removido. Este traço pode ser identificado pelos pontos de características.

Os pontos de segmentação são então escolhidos de acordo com algumas heurísticas que basicamente analisam os posicionamentos dos pontos de características encontrados com relação a mais alta colina e o mais profundo vale encontrado na imagem (Figura 2.24).

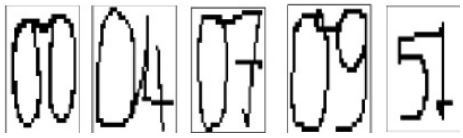


Figura 2.23: Conexões com traço-ligador. [EA03]

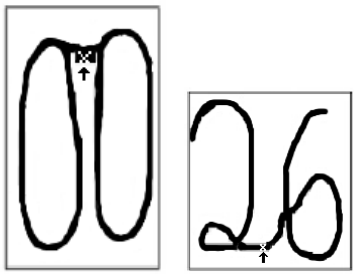


Figura 2.24: Na esquerda temos uma imagem exemplificando um ponto de colina e na direita um ponto de vale.

No caso específico de conexões suaves (Figura 2.22), o algoritmo não consegue encontrar os pontos de características e conseqüentemente não determina nenhum caminho de segmentação.

Embora tenha sido desenvolvido para tratar de conexões simples, os autores afirmam que o algoritmo é capaz de segmentar casos de conexão múltipla se estes pontos de conexão se encontrarem próximos um ao outro.

O método foi testado com imagens da base CEDAR, base NIST 19 e imagens de propriedade dos autores. Os testes resultaram em uma taxa de acerto de 96%.

Como limitações deste método podemos citar o fato de não conseguir segmentar casos de conexão suave. Além disso, o algoritmo utiliza processos computacionalmente caros, como esqueletização e deslizamento de máscaras para a extração de características. Um ponto positivo do método é que com a utilização das máscaras propostas para encontrar os pontos de segmentação, a implementação do algoritmo se torna simples e é possível realizar uma implementação paralela para reduzir o tempo de processamento.

2.2.8 Lei et al

Lei et al [LLDF04] propõem um método baseado no conceito de sobre-segmentação e reconhecimento. Método este que realiza um pré-processamento para normalização do tamanho, suavização do contorno e uma operação morfológica de fechamento, esta visando resolver o problema de pedaços quebrados e ruídos. Antes de iniciar a segmentação da imagem, o método decide se a mesma é de dígito isolado ou conectado, assim como em Pal et al [BCP03], através de um limiar aplicado ao resultado de um

classificador de dígitos isolados.

Para identificar os possíveis pontos de segmentação é realizada uma análise do contorno interno e externo, divididos em superior e inferior (Figura 2.25), além de projeções de histograma. A divisão do contorno em superior e inferior é realizada à partir dos pontos extremos esquerdo e direito da imagem, assim como é feito no método proposto por Fujisawa et al [FNK92]. O resultado ótimo da segmentação é determinado pela máxima probabilidade de reconhecimento. Visando a diminuição do número de caminhos de segmentação a serem testados é utilizada uma condição de poda, tendo assim uma consequente redução do tempo de processamento.

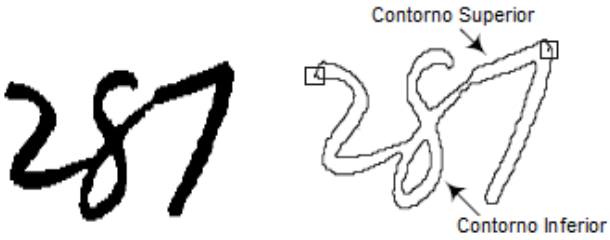


Figura 2.25: Contorno inferior e superior de uma cadeia de dígitos conectados. [LLDF04]

Este método foi testado pelos autores em 3359 exemplos da base NIST SD19, e atingiu uma taxa de acerto de 97,72% (sem rejeição), em cadeias de dois dígitos. Para cadeias de três dígitos o acerto foi de 93,33%, para os 525 exemplos testados. Vale ressaltar que as imagens utilizadas sempre possuíam os três dígitos conectados, o que pode-se considerar um fato de aumento da complexidade em relação a testes realizados por outros autores, os quais contém entre as imagens, casos em que apenas dois dígitos se tocam.

Podemos destacar como ponto interessante do algoritmo a condição de poda utilizada no grafo contendos as hipóteses de segmentação, pois o teste das hipóteses é uma etapa que penaliza métodos baseados em sobre-segmentação. Como ponto negativo, temos o fato do algoritmo utilizar uma operação morfológica durante o pré-processamento, operação esta que eleva o custo computacional do método.

2.2.9 Suwa e Naoi

Suwa e Naoi [SN04] propõem um algoritmo capaz de segmentar tanto dígitos com conexão simples, quanto dígitos com conexão múltipla. O algoritmo proposto trabalha apenas com pares de dígitos e segmenta inclusive casos em que há um traço ligador, utilizando para a identificação e remoção dos traços ligadores, o método proposto por Elnagar et al [EA03].

Antes da segmentação o algoritmo executa um pré-processamento, o qual realiza remoção de ruídos, suavização e correção de inclinação. Após isso é realizada a

esqueletização da imagem (Figura 2.26b) e identificação dos vértices de graus um, três e quatro (pontos finais, junção em T e pontos de cruzamento, respectivamente). Os vértices são unidos por arestas formando um grafo conectado (Figura 2.26c). Algumas arestas são eliminadas de acordo com algumas heurísticas.

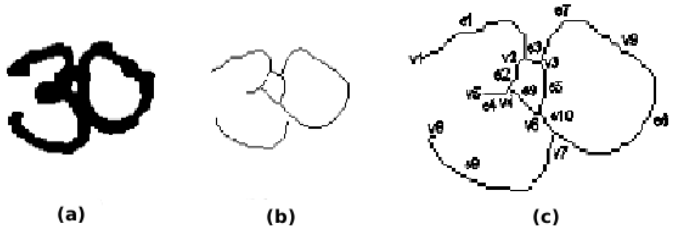


Figura 2.26: Par de dígitos e suas diferentes representações. (a) Imagem original, (b) Imagem do esqueleto, (c) Representação de grafo no contorno da imagem. [SN04]

Para realizar a segmentação, é necessário a obtenção de dois sub-grafos. Então para realizar o agrupamento dos vértices/arestas que pertencem a um mesmo dígito, são utilizados conceitos de teoria dos grafos e algumas heurísticas. Segundo os autores os pontos de segmentação geralmente são encontrados próximos ao ponto mais baixo do "vale" formado no contorno superior e no mais alto da "colina" encontrada no plano inferior. Esses pontos são encontrados pelo algoritmo com ajuda de dois histogramas verticais.

Os autores classificam as conexões entre dígitos em quatro tipos (Figura 2.27): Conexão em um ponto, conexão em um segmento, ponto de conexão em traço-ligador e conexão múltipla. Então o algoritmo identifica primeiramente o tipo da conexão para depois definir as heurísticas a serem utilizadas.

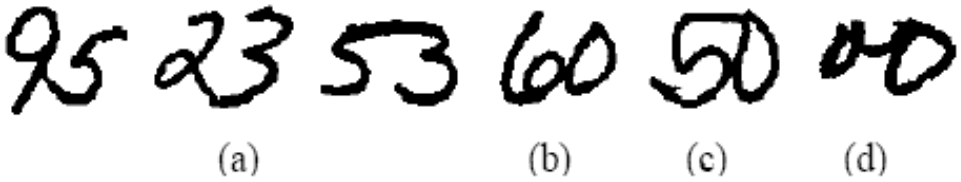


Figura 2.27: Tipos de conexão: (a) Conexão em um ponto, (b) Conexão em um segmento, (c) Conexão múltipla, (d) Conexão com traço-ligador. [SN04]

O desempenho do algoritmo foi avaliado nos testes realizados sobre 2000 pares de dígitos conectados, extraídos da base NIST 19. Outras 1000 imagens procedentes da mesma base foram utilizadas para construção das heurísticas. O algoritmo atingiu nos testes um desempenho de 88,7% de acerto com taxa de rejeição de 2,45%. As imagens segmentadas produzidas pelo algoritmo possuem forma mais natural do que as imagens geradas por algoritmos que utilizam uma linha reta para separar os dígitos.

O fato do método tentar classificar a conexão quanto ao tipo antes de segmentar, o torna um método interessante para ser utilizado em nossos trabalhos futuros (vide seção 5.1).

2.2.10 Sadri et al

Sadri et al [SSB07] propõem um método de segmentação e reconhecimento de strings numerais de tamanho variável, o qual utiliza um algoritmo genético para encontrar o melhor caminho de segmentação. Assim como em Lei et al [LLDF04], o método também é baseado no conceito de sobre-segmentação.

Antes da segmentação da imagem, a mesma passa por um pré-processamento constituído de suavização, remoção de ruídos e correção de inclinação. Na fase de segmentação o método trabalha com uma análise de componentes conectados (CCs). Desta forma os CCs podem ser classificados em três tipos: pedaços de dígitos quebrados ou traços-ligadores, dígitos isolados e dígitos conectados. Dentre estes, apenas o terceiro tipo necessita de segmentação. Por isso, o método realiza esta classificação analisando informação contextual (basicamente tamanho relativo do componente em relação à cadeia).

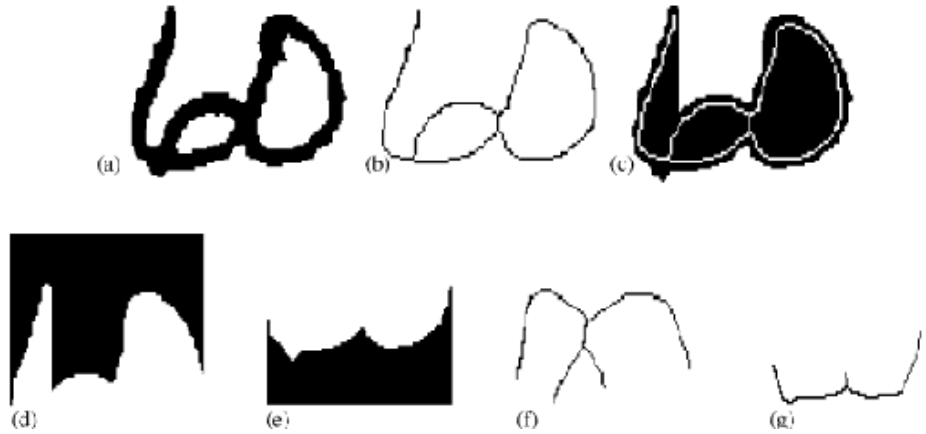


Figura 2.28: (a) Imagem pré-processada, (b) Esqueleto do plano frontal, (c) Plano de fundo (pixels brancos fora do objeto composto por pixels pretos, (d) projeção do perfil superior, (e) projeção do perfil inferior, (f) esqueleto superior do plano de fundo, (g) esqueleto inferior do plano de fundo. [SSB07]

Feita a classificação dos CCs, o método então vai buscar os caminhos de segmentação para todos os CCs classificados como dígitos conectados. Para isto são utilizadas características tanto do plano frontal (CC), como do fundo da imagem. As características do primeiro tipo (plano frontal) são os pontos de intersecção extraídos do esqueleto da imagem. Os autores propõem uma nova maneira de extrair características do plano de fundo, a qual não considera todos os pixels brancos da imagem. Primeiramente é obtida a projeção do perfil da imagem, logo após é então obtido o

esqueleto desta imagem da projeção. Segundo os autores pelo fato deste método cobrir apenas a parte essencial do fundo da imagem, os pontos encontrados são mais estáveis e informativos, além de ser encontrado um menor número de características de fundo, o que propicia uma maior simplicidade e menor custo computacional para avaliar estas características. Todas as etapas do método proposto para extração das características do plano de fundo são demonstradas pela Figura 2.28.

As características encontradas são então combinadas e os possíveis caminhos de segmentação gerados. Se existirem n caminhos de segmentação, então existirão 2^n hipóteses de segmentação para a cadeia de dígitos em questão. Nos casos em que $n <= 5$ (até 32 hipóteses), são avaliadas todas as hipóteses através de uma busca exaustiva. Porém se $n > 5$ então é feita a busca com auxílio de um algoritmo genético, o qual nem sempre encontra a solução ótima, porém encontra uma solução próxima a esta com custo computacional menor do que da busca exaustiva.

Para a avaliação das hipóteses de segmentação é proposta a combinação de um reconhecedor de dígitos manuscritos isolados com informações contextuais, informações estas que compõem um novo conjunto de valores, chamado de escore da segmentação. Apesar de a maioria dos algoritmos encontrados na literatura utilizarem apenas um reconhecedor de dígitos isolados, optou-se por utilizar também este escore de segmentação para prevenir alguns erros comuns (Figura 2.29) encontrados em sistemas de reconhecimento de cadeias manuscritas. Este escore expressa o grau de confiança de que determinado CC pertence ou não a cadeia de dígitos em questão e é formado basicamente por duas partes: posição relativa e razão de aspecto.

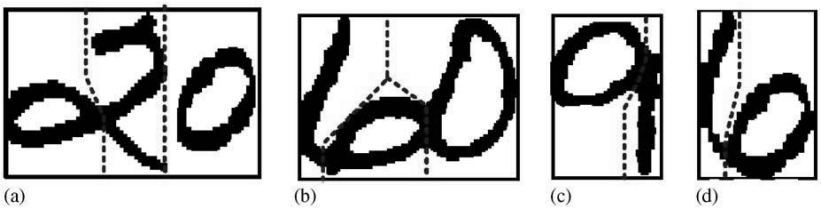


Figura 2.29: Casos nos quais provavelmente um reconhecedor de dígitos isolados cometeria erro sem a ajuda de informações de contexto. Por exemplo, (a) poderia ser facilmente reconhecido como 020, (b) como 101, (c) como 01 e (d) como 10. [SSB07]

Então para determinar se determinado CC é ou não valido dentro da cadeia analisa-se o mínimo entre o escore da segmentação e o escore do reconhecimento, sobre este mínimo aplica-se um limiar. Se o dígito for maior que um limiar (determinado empiricamente) então este é considerado válido.

Foram realizados dois testes do algoritmo, utilizando dois métodos de reconhecimento diferentes. Um deles utiliza uma rede neural MLP e o outro um máquina de vetores de suporte (SVM). Antes dos testes do sistema como um todo, foram realizados testes do módulo de segmentação, o qual, sobre uma base de dados contendo 5000

imagens de pares de dígitos conectados da base NIST, obteve uma taxa de acerto de 96,5%. Já para as 1800 imagens extraídas da base NIST NSTRING SD19, imagens estas de cadeias de tamanho dois, três, quatro, cinco, seis e dez, a taxa de acerto foi de 98,04%. Chegou-se a estes números através de avaliação visual dos resultados da segmentação. Porém estes números não podem ser comparados aos métodos encontrados na literatura, pois os autores consideraram como acerto todos os casos em que todos os caminhos de segmentação necessários estavam presentes dentre os vários caminhos encontrados pela sobre-segmentação da imagem. Ou seja, estas taxas de acerto podem diminuir caso o algoritmo que decide quais caminhos de segmentação dentre os encontrados deve utilizar, não seja livre de erros, o que é provável.

Destes primeiros testes foram extraídos os CCs que não constituem um dígito, para auxiliar na determinação de alguns parâmetros do calculo do escore de segmentação e também foram utilizados no treinamento da rede neural e do SVM.

Os testes finais foram então feitos, a versão utilizando MLP obteve uma taxa de acerto (segmentação e reconhecimento) de 95,28% e a que utilizou SVM alcançou 96,42%. Os testes também mostraram que houve uma melhora de 7,85% para o MLP e 6,35% para o SVM, referente à utilização do escore de segmentação.

Podemos considerar como contribuições significativas deste método o uso do escore de segmentação, o qual trouxe melhoria considerável no desempenho do algoritmo. O método diferenciado de análise do plano de fundo, o qual análise apenas os pixels considerados essenciais, reduzindo assim a quantidade de pixels a serem analisados e obtendo com isso um menor custo computacional. Devemos citar também o bom desempenho alcançado na segmentação de conexões que possuem traços-ligadores, os quais são responsáveis por boa parte dos erros em grande parte dos métodos encontrados na literatura. O uso de algoritmo genético também é um diferencial do método e é o principal ponto comentado no artigo, porém não demos destaque aos detalhes do AG, pois seu uso não faz sentido na segmentação de cadeias numéricas contendo apenas dois dígitos, caso este que é o foco do nosso projeto.

2.2.11 Britto et al

Um método que utiliza segmentação implícita para reconhecimento de cadeias de dígitos manuscritos, baseado em HMM, é proposto por Britto et al [BSBS03]. O método possui dois estágios, o primeiro é dividido em 3 módulos: pré-processamento, extração de características de traço e classificação/segmentação. Primeiramente o pré-processamento faz a correção da inclinação, suaviza o contorno e calcula o *bounding box* (local onde o objeto se encontra) da cadeia de dígitos. O módulo de extração de características varre a cadeia de dígitos, da esquerda para a direita, enquanto um vetor baseado em informações do traço é calculado para cada coluna da palavra. No módulo

de classificação/segmentação, dez HMMs que representam as classes de dígitos manuscritos treinados utilizando também informações contextuais, são concatenados em um processo dinâmico, baseado em um algoritmo de construção de níveis, utilizando a sequência de características provinda do módulo de extração das mesmas. Segundo o autor, neste estágio há uma perda em termos de reconhecimento, uma vez que as características e modelos utilizados buscam contemplar a segmentação e reconhecimento em um único processo.

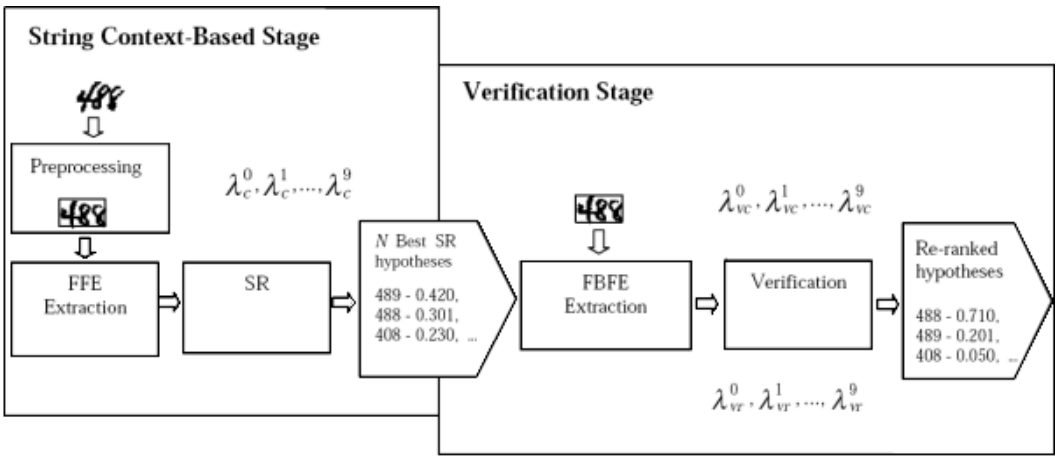


Figura 2.30: Esquema de funcionamento do método. [Bri01]

Como o resultado do primeiro estágio pode não ser preciso, os pontos de segmentação e o resultado de reconhecimento da cadeia fornecidos pelo primeiro estágio são utilizados em um procedimento de verificação das hipóteses de segmentação (segundo estágio). Neste, os pontos de segmentação são utilizados para definição dos limites de cada dígito na cadeia, o que torna possível a extração de novas características baseadas no fundo da imagem. Estas novas características são combinadas com as anteriores em outros vinte HMMs representando colunas e linhas. Neste segundo estágio o autor combina informações complementares como características dos dígitos e do plano de fundo, extraídas das colunas e linhas da imagem reconhecida no primeiro estágio sem utilizar nenhuma informação de contexto. Com base nesta verificação, a probabilidade do conjunto de dígitos é calculada e é adicionada à probabilidade das hipóteses obtidas no estágio anterior. O resultado final é utilizado para reclassificar a pontuação do conjunto de dígitos das hipóteses.

Nos experimentos, o autor relata uma taxa de reconhecimento de 89,61% para 2069 pares de dígitos conectados extraídos da base NIST SD19. Se o método for utilizado para reconhecimento de dígitos isolados, a taxa de acerto é de 98.02%

Uma característica deste método que merece destaque é o fato de não ser necessário definir heurísticas para realizar a segmentação, além de segmentar cadeias numéricas de tamanho variável. Como ponto negativo temos o custo computacional,

pois o método é composto por dois estágios, ambos utilizando HMMs.

2.3 Comparação dos Métodos

A tabela 2.1 estabelece uma comparação entre os métodos de segmentação descritos neste capítulo. As informações apresentadas na tabela são as seguintes:

- **Método:** Método de segmentação em questão (vide Referências Bibliográficas)
- **Primitivas:** Tipos de primitivas utilizadas na extração de características para determinar os caminhos de segmentação (ex: contorno, esqueleto, dentre outras).
- **Traço Lig:** Remove ou não Traços-Ligadores.
- **Pré-Proc:** Possui ou não algoritmo de pré-processamento.
- **Pré-Class:** Executa ou não pré-classificação da imagem em isolada ou conectada, antes da segmentação.
- ≥ 2**:** Segmenta ou não cadeias contendo mais que 2 dígitos.
- **Sobre-Seg:** Algoritmo é ou não baseado no conceito de Sobre-Segmentação
- **Abordagem:** O método é baseado em reconhecimento ("Base Reco"); é um método de Segmentação e Reconhecimento ("Seg / Reco") ou é um método com segmentação implícita ("Implícita").
- **Base:** Base de dados utilizada pelos autores para realizar o teste do método de segmentação
- **Qtd:** Quantidade de imagens presentes na base de dados utilizada pelos autores.
- **Acerto:** Percentual de acerto reportado pelos autores.
- **Rejeição:** Percentual de imagens rejeitadas.

Os percentuais de acerto da segmentação exibidos, são os reportados pelos autores. Lembrando que não podemos tirar conclusões concretas, olhando apenas para o desempenho exibido nesta tabela, devido principalmente a dois fatores. O primeiro é que os métodos não foram testados com as mesmas imagens, então estes podem ter sido submetidos à imagens com diferentes graus de dificuldade. Outro fator, é que as formas de avaliação também não foram as mesmas, dentre as utilizadas temos: avaliação visual, redes MLP, HMMs e SVM. Portanto apesar desta tabela ser uma forma de comparação, não é possível tirar conclusões justas, quanto ao desempenho, somente com base nela. Esperamos ao fim deste projeto, poder realizar uma comparação mais precisa.

Tabela 2.1: Tabela comparativa dos métodos. O desempenho corresponde ao reportado pelos autores.

Método	**Primitivas**	**Traço Lig**	**Pré-Proc**	**Pré-Class**	≥ 2	**Sobre-Seg**	**Abordagem**	**Base**	**Qtd**	**Acerto**	**Rejeição**
[FNK92]	Contorno	Não	Não	Sim	Não	Sim	Base Reco	Imagens próprias	920	95,0%	5,0%
[SG97]	Contorno	Sim	Não	Não	Não	Sim	Seg / Reco	CEDAR	1966	80%	-
[OLBS00]	Contorno / Perfil	Não	Não	Não	Não	Não	Base Reco	Cheques Brasileiros	900	98,5%	0%
[CW00]	Esqueleto (Frente e Fundo)	Sim	Sim	Não	Não	Não	Seg / Reco	NIST/ próprias	4500	96%	7,8%
[YY01]	Contorno / Concavidades	Não	Sim	Não	Sim	Não	Seg / Reco	NIST SD19	3287	94,8%	-
[BCP03]	Reservatórios / Concavidades	Não	Não	Sim	Não	Não	Seg / Reco	Cheques Franceses	2250	94,8%	3,4%
[EA03]	Esqueleto / Contorno	Sim	Sim	Não	Não	Não	Seg / Reco	NIST/ CEDAR/ próprias	-	96%	-
[LLDF04]	Contorno	Não	Sim	Sim	Sim	Sim	Base Reco	NIST SD19	3359	97,72%	0%
[SN04]	Esqueleto	Sim	Sim	Não	Não	Não	Seg / Reco	NIST SD19	2000	88,7%	2,45%
[SSB07]	Esqueleto / Fundo	Sim	Sim	Sim	Sim	Sim	Base Reco	NIST SD19	5000	96,5%	-
[BSBS03]	Transições	Não	Sim	Não	Sim	Não	Implícita	NIST SD19	2069	89,61%	-

Capítulo 3

Metodologia Proposta

Como vimos na introdução, a segmentação é uma fase crítica em sistemas de reconhecimento de manuscritos. Além disso foi mostrado no Capítulo 2 que existe atualmente uma dificuldade em comparar os métodos existentes, devido à grande quantidade deles que pode ser encontrada na literatura e a falta de um protocolo para a análise dos mesmos. Neste capítulo, descrevemos a metodologia utilizada neste trabalho de caracterização de algoritmos de segmentação, o qual visa a solução do atual problema de comparação de algoritmos e também gera dados que serão necessários para em um trabalho futuro, desenvolver um novo método de segmentação, mais eficiente que os atuais.

O desenvolvimento deste trabalho se dará seguindo as seguintes etapas:

1. Seleção dos algoritmos de segmentação à serem implementados.
2. Criação da base de dados à ser utilizada nos testes.
3. Definição dos critérios de avaliação que serão utilizados.
4. Implementação dos métodos.
5. Análise comparativa dos resultados obtidos.

3.1 Seleção dos Algoritmos

Optamos por focar este trabalho na segmentação de pares de dígitos conectados. Esta decisão foi baseada nos dados mostrados por Wang et al [WGS00], pois nesse artigo, os autores analisaram uma base de dados contendo 200.000 cadeias de dígitos correspondentes a códigos de endereçamento postal, números de rua, números de apartamentos e números de caixas postais. Com esta análise, concluíram que 85% dos casos de conexão encontrados correspondem a pares de dígitos conectados.

Decidimos realizar o trabalho de caracterização apenas com métodos de segmentação explícita, pois esses tem menor custo computacional e representam a maioria dos métodos encontrados na literatura. Além disso, para analisar também métodos de segmentação implícita, teríamos que utilizar uma outra metodologia, pois esses realizam a segmentação juntamente com o reconhecimento, não sendo então possível a comparação do modo em que esta será proposta nas seções seguintes.

Devido à grande quantidade de métodos existentes na literatura, foi necessário selecionar alguns deles para serem implementados e analisados neste trabalho. Dentre os métodos apresentados no capítulo anterior, foram selecionados para implementação e caracterização, os propostos por Fujisawa et al [FNK92], Shi e Govindaraju [SG97], Oliveira et al [OLBS00], Chen e Wang [CW00], Pal et al [BCP03] e Elnagar e Alhajajj [EA03].

O método proposto por Lei et al [LLDF04] é bastante parecido com o proposto por Fenrich [Fen91], ambos utilizando características similares extraídas do contorno da imagem, como pontos de mínimo locais, além de características do perfil da imagem, para identificação dos pontos de segmentação. Os dois métodos se baseiam em estratégia baseada em reconhecimento, onde as hipóteses de segmentação são colocadas em forma de grafo e avaliadas com um classificador. Como implementamos o método proposto por Oliveira et al [OLBS00], o qual nada mais é que uma evolução do método do Fenrich, achamos que a implementação do método do Lei et al não era necessária nesse primeiro momento e então este algoritmo não foi implementado.

Já o método proposto por Suwa e Naoi [SN04] utiliza basicamente pontos de esqueleto e analisa os pontos de máximo e mínimo no contorno da imagem, ou seja, muito parecido com o método proposto por Elnagar et al [EA03], o qual também utiliza essas características. Além disso, ambos os métodos conseguem remover traços-ligadores, realizam pré-processamento e pré-classificação da imagem, segmentam apenas pares de dígitos, não utilizam sobre-segmentação e utilizam abordagem baseada em segmentação e reconhecimento em etapas distintas, sem uso de classificador na etapa de segmentação. Devido a todas essas similaridades, optamos por selecionar apenas um deles, não implementando então o método de Suwa e Naoi.

O foco dos métodos propostos por Sadri et al [SSB07] e por Yu e Yan [YY01] é a segmentação de cadeias de dígitos manuscritos o que foge do escopo deste projeto, que como já foi descrito, visa trabalhar com algoritmos que segmentam pares de dígitos manuscritos. O primeiro evidencia o ambiente proposto o qual é baseado em algoritmo genético para segmentação das cadeias de dígitos manuscritos, mas esse ambiente não é útil para pares de dígitos. Além disso, o método proposto por Sadri et al ainda utiliza pontos de esqueleto, características de fundo e contorno. Características essas parecidas com as utilizadas por Elnagar et al [EA03] e Chen e Wang [CW00]. Por

todos esses motivos, esses dois métodos (Sadri et al e Yu e Yan) também não foram implementados.

Foram implementados métodos que utilizam diferentes características (contorno, perfil, esqueleto, reservatórios e concavidades), com diferentes abordagens, como foi exposto na tabela 2.1 exibida no capítulo anterior.

3.2 Criação da Base de Dados

Para realizar a avaliação dos algoritmos, será utilizada uma base sintética de dígitos conectados proposta por Oliveira et al [OBJS05]. Dizemos que é uma base "sintética", pois a mesma foi gerada automaticamente com base em imagens de dígitos isolados da base NIST. O algoritmo utilizado para a construção desta base seleciona dois dígitos isolados de mesmo escritor (a informação de escritor está disponível na base NIST), e os aproxima até que aconteça um toque. A base contém 273.452 pares de dígitos conectados, alguns exemplos podem ser vistos na Figura 3.1. Um detalhe importante desta base de dados, é que para sua criação, só foram utilizados dígitos isolados os quais foram classificados com sucesso pelo nosso classificador, o qual será descrito nas seções seguintes deste capítulo. Este fato minimiza os casos de erro ocorridos por conta da classificação, credibilizando ainda mais este trabalho.

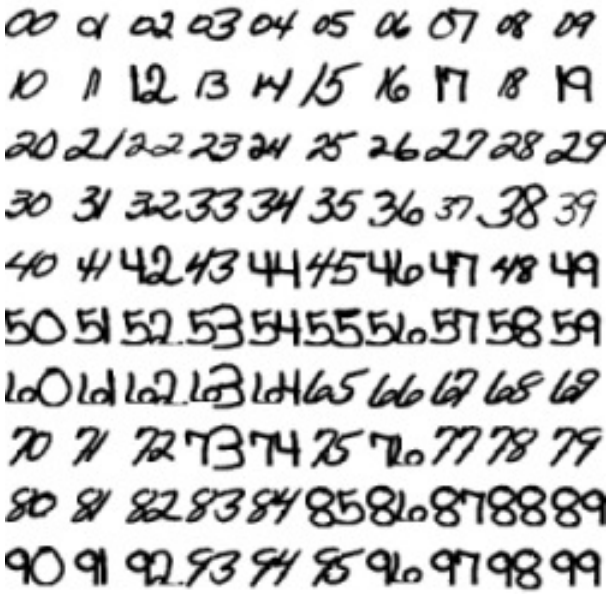


Figura 3.1: Exemplos de imagens geradas pelo algoritmo de geração automática. [OBJS05]

O algoritmo gerador da base de dados, após criar cada uma das imagens, armazena um arquivo de informações, em formato texto, contendo as imagens isoladas que originaram o par de dígitos conectados e também o que chamamos de "caminhos

ótimos de segmentação". Esses caminhos nada mais são que as coordenadas dos pontos que conectam as duas imagens isoladas, ou seja, se a imagem for dividida em duas, utilizando para isso segmentos de retas nessas coordenadas, teremos uma segmentação correta dos pares de dígitos conectados, produzindo assim imagens isoladas muito próximas das imagens da base NIST que as originaram. Não podemos dizer que as imagens são exatamente iguais as imagens que as originaram, pois para isso teria que ser armazenado ponto a ponto no arquivo texto o caminho de segmentação, e o que existe são apenas as coordenadas do primeiro e último pontos, fornecendo assim uma aproximação do caminho real. Outro fator é que em alguns casos existem pixels sobrepostos e então após segmentada a imagem, um dos dígitos ficará sem esses pixels. Porém a quantidade de pixels sobrepostos é mínima, sendo na maioria das imagens nula.

Figura 3.2: Na esquerda, exemplo de imagem da base de dados contendo os dígitos "65". À direita a imagem depois de traçado seu "caminho ótimo de segmentação".

Na Figura 3.2 é exibido um exemplo de imagem presente na base de dados. O arquivo de informações correspondente à esta imagem é exibido pela Figura 3.3. Podemos observar que a imagem foi formada pelos arquivos `cdf0014_19_1_0.tif` e `cdf0035_19_1_0.tif` da base NIST SD19, onde o primeiro corresponde ao dígito "6" e o segundo ao dígito "5". As coordenadas dos pontos ótimos de segmentação para esta imagem são $(37, 30)$ e $(37, 40)$.

```
/isolated/6/cdf0014_19_1_0.tif
/isolated/5/cdf0035_19_1_0.tif
x1 - y1    x2 - y2
37 - 30    37 - 40
```

Figura 3.3: Exemplo de arquivo de informações referente a imagem da base exibida na Figura 3.2

Neste trabalho foi utilizada uma parte desta base de dados, o que corresponde a aproximadamente 80.400 imagens (não será utilizada a base de dados inteira, pois não julgamos necessária quantidade tão grande de imagens). Todas essas imagens, como descrito nos parágrafos anteriores, possuem como informação adicional as coordenadas

dos pontos de conexão, facilitando assim a posterior avaliação do desempenho dos algoritmos de segmentação.

Para que seja possível a análise dos casos de acerto e erro de cada algoritmo de segmentação, de acordo com o tipo de conexão existente na imagem, foi preciso classificar a base de dados de acordo com o tipo de conexão. Para isso foi utilizada a classificação proposta por Chen e Wang [CW00] (Figura 2.12). Porém, devido ao fato de nossa base de dados possuir pouquíssimos casos de conexão por traço-ligador (tipo 4 na classificação de Cheng e Wang), optamos por não considerar este como um tipo de classificação e classificar as imagens que pertenceriam a este tipo, como sendo do tipo 1 ou 3, dependendo do caso. Um exemplo dessa classificação alternativa para os casos de dígitos conectados por traço ligador é mostrado na Figura 3.4.

Figura 3.4: Exemplo de imagem contendo traço-ligador, classificada alternativamente como tipo 1.

A classificação da base de dados foi realizada de acordo com os quatro possíveis tipos (Figura 3.5), em duas etapas. Uma primeira etapa automática classifica as imagens uma a uma de acordo com a localização do ponto de segmentação. Essa localização podendo ser: superior, central ou inferior. Para isso foram mapeadas manualmente cada uma das 100 classes (00 à 99), informando assim ao algoritmo qual tipo de conexão deveria ser atribuído para cada uma das possíveis posições do ponto de conexão em cada classe de dígitos. Por exemplo, de modo geral uma conexão de classe "02" é do tipo 1 quando a conexão acontece na parte superior da imagem, e de tipo 2 quando acontece na parte central ou inferior da imagem.

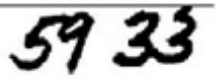
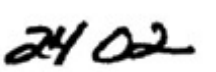
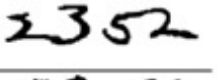
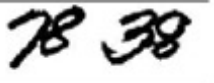
Categoria	Tipo	Tipo de Conexão	Exemplo
Conexão simples	1		
	2		
	3		
Conexão múltipla	4		

Figura 3.5: Tipos de conexão utilizados para a classificação da nossa base de dados.

Porém apesar desta classificação automática realizada ter atribuído os tipos corretamente à maioria das imagens, muitas não se encaixam nessas regras básicas

baseadas na posição do ponto de toque e foram erroneamente classificadas. Para resolver este tipo de problema, foi realizada uma segunda etapa. Etapa esta manual, na qual todas as imagens passaram por uma avaliação visual para correção dos casos de classificação incorreta.

Figura 3.6: Exemplos de imagens removidas da base de dados. Classe “16” e “10” respectivamente.

Nesta segunda etapa também foram removidas da base de dados algumas imagens que não condizem com a realidade, ou seja, imagens com conexões que nunca seriam vistas em casos naturais (Figura 3.6), mas que aparecem na base devido ao fato das imagens terem sido geradas artificialmente. Também foram removidas da base de dados, todas as imagens pertencentes à classe “11”, pois conexão entre dois dígitos “1” só existe quando o mesmo é escrito “com perna” (Figura 3.7). Nossa base foi criada com dígitos isolados retirados da base NIST SD19, a qual é uma base americana. O padrão americano de escrita do número “1” é “sem perna”.

Figura 3.7: Dígito “1”, com e sem “perna”, respectivamente.

Após esta remoção das imagens indesejadas, obtivemos uma base de dados contendo 79.466 imagens. A distribuição das imagens de acordo com o tipo de conexão pode ser visualizado na Figura 3.8. Podemos notar que os tipos 2 (53,6%) e 1 (34,8%) são os predominantes em nossa base.

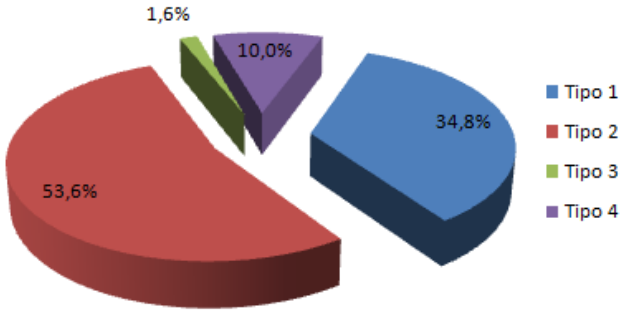


Figura 3.8: Gráfico da distribuição da Base de Dados de acordo com os tipos de conexão.

3.3 Definição dos Critérios de Avaliação

Nesta seção é definido o critério de avaliação a ser utilizado na comparação dos algoritmos de segmentação que serão implementados, os quais foram definidos na seção 3.1. Dois pontos serão avaliados, o mais importante deles é a taxa de acerto, quesito este que julgamos o mais importante deste trabalho de caracterização e que por isso daremos maior ênfase. Outro quesito a ser comparado é o custo computacional.

Como cada imagem da base possui as coordenadas dos pontos ótimos de segmentação, parece óbvio que a verificação dos pontos de segmentação gerados pelos algoritmos seja feita através de uma comparação com esses pontos ótimos presentes na base de dados. Porém, dependendo da forma que esta comparação seja feita, podemos enfrentar alguns problemas. Por exemplo, digamos que para uma determinada imagem, os pontos fornecidos pela base de dados como pontos ótimos de segmentação, sejam em número de seis e que um determinado algoritmo encontre para esta mesma imagem, apenas dois pontos de segmentação. Apesar do algoritmo ter gerado apenas $\frac{1}{3}$ dos pontos de segmentação ideais, não podemos afirmar que o algoritmo não obteve sucesso na segmentação. Esta situação é representada pela Figura 3.9. Percebemos então que apesar das duas hipóteses de segmentação mostradas por esta Figura possuírem pontos de segmentação muito parecidos, elas conduziriam à resultados diferentes, a primeira conduzindo ao resultado "38" e a segunda ao resultado "33", que neste caso é o correto.

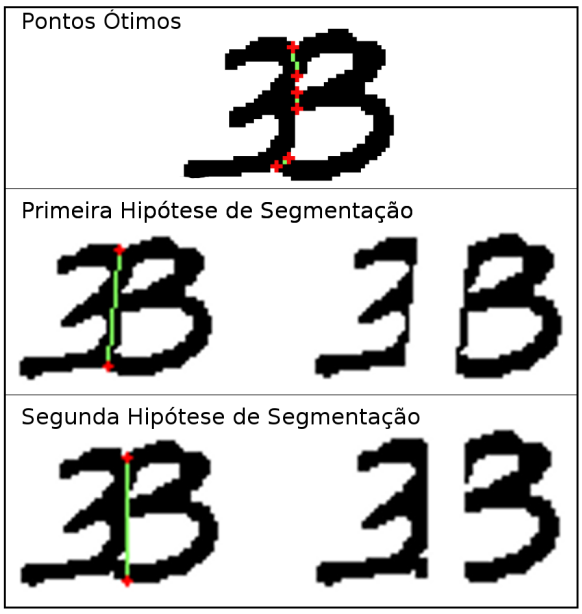


Figura 3.9: Imagens da classe 33, seus caminhos ótimos de segmentação e duas hipóteses. Sendo a primeira hipótese incorreta e a segunda correta.

Um detalhe que deve ser percebido é que a hipótese de segmentação que conduziria ao erro, possui seus pontos de segmentação mais próximos dos pontos ótimos

do que a outra hipótese que obteve um resultado correto.

Foram feitos alguns experimentos visando encontrar uma forma de melhor comparar os pontos encontrados com os pontos ótimos, como por exemplo, em um dos experimentos adicionamos à comparação, a quantidade de cortes realizados na imagem. Obtivemos esta informação da quantidade de cortes realizados, contando as transições branco/preto que ocorrem no(s) caminho(s) de segmentação gerado(s) e no(s) caminho(s) ótimo(s) de segmentação. Isto resolveu o problema exemplificado pela imagem 3.9, porém mesmo assim ainda restaram muitas imagens que foram erroneamente classificadas.

Devido à este motivo consideramos que não seria adequado realizar a avaliação dos algoritmos de segmentação comparando a localização dos pontos de segmentação encontrados, com a localização dos pontos ótimos de segmentação. Para termos certeza de que não teríamos qualquer tipo de erro na avaliação dos pontos encontrados pelos algoritmos, sendo estes decorrentes de quaisquer outros casos que possam configurar exceções e não sejam corretamente classificados pelo método de comparação da localização descrito nos parágrafos anteriores, avaliamos os pontos encontrados utilizando uma rede neural, e quando necessário, utilizando também a localização para determinar quais pontos testar com a rede neural.

Nem todos os métodos produzem resultado da segmentação em mesmo formato. Fujisawa et al [FNK92] e Oliveira et al[OLBS00] produzem uma ou mais hipóteses de segmentação, sendo cada uma delas um segmento de reta, ou seja, dois pontos de segmentação. Nesse caso, se for gerada uma quantidade maior de caminhos de segmentação, do que o número de caminhos ótimos, são escolhidos quais são utilizados de acordo com a "proximidade do caminho ótimo" (O cálculo da proximidade é descrito no parágrafo seguinte). Se a quantidade de caminhos gerada for igual ou menor do que a quantidade de caminhos ótimos, então todos os caminhos gerados são utilizados na segmentação.

A proximidade p é calculada utilizando a seguinte equação:

$$p = min(d_1, d_2) \tag{3.1}$$

onde $min(a, b)$ é o mínimo entre os valores a e b, já d_1 e d_2 são somas de distâncias Euclidianas, conforme equações 3.2 e 3.3, nas quais $d = (a, b)$ é a distância Euclidiana entre dois pontos, a e b, por sua vez, s_{1A} e s_{1B} correspondem ao primeiro e segundo ponto pertencentes ao segmento s_1 (caminho de segmentação encontrado pelo algoritmo) e de forma análoga s_{2A} e s_{2B} os pontos pertencentes ao segmento s_2 (caminho ótimo de segmentação).

$$d_1 = d(s_{1A}, s_{2A}) + d(s_{1B}, s_{2B}) \tag{3.2}$$

$$d_2 = d(s_{1A}, s_{2B}) + d(s_{1B}, s_{2A}) \tag{3.3}$$

Temos então que d_1 é a soma das distâncias Euclidianas de s_{1A} e s_{2A} e de s_{1B} e s_{2B}. E d_2 é a soma das distâncias Euclidianas entre s_{1A} e s_{2B} e entre s_{1B} e s_{2A}.

Calculadas as proximidades de todos os caminhos de segmentação encontrados pelo algoritmo, com os caminhos de segmentação ótimos, são escolhidos os de menor proximidade para serem testados na próxima etapa pela rede neural. Uma alternativa mais simples para esse cálculo de proximidade seria utilizar a equação que calcula diretamente a distância entre duas retas, porém realizamos testes com ambas as alternativas de cálculo e a que está apresentada acima obteve melhor desempenho do que o cálculo direto de distância entre retas.

Selecionados os pontos a serem utilizados, segmenta-se a imagem. Obtidas as duas imagens, sendo uma supostamente pertencente ao dígito da esquerda e outra contendo o dígito da direita. Então essas imagens obtidas são fornecidas como entrada para o classificador, e o resultado fornecido pelo mesmo é comparado com a informação real dos dígitos (também fornecida pela base de dados). Nos casos em que os dois dígitos forem reconhecidos corretamente pela rede neural, dizemos que o algoritmo de segmentação obteve sucesso. Todos os demais casos são considerados casos de erro de segmentação.

Já o algoritmo proposto por Chen e Wang [CW00], gera caminhos de segmentação compostos por n pontos. Como os caminhos ótimos de segmentação são compostos cada um por dois pontos apenas, para este algoritmo comparamos de maneira análoga à descrita nos parágrafos anteriores, porém comparando os pontos do caminho ótimo com o primeiro e último ponto do caminho de segmentação encontrado pelo algoritmo. Ou seja, os pontos intermediários de cada caminho são desconsiderados para o cálculo da proximidade dos caminhos encontrados. Porém após determinado quais são os caminhos a serem testados com o classificador, todos os n pontos que compõem o caminho são utilizados.

Tanto para o método proposto por Elnagar e Alhajajj [EA03] que retorna como resultado apenas um caminho de segmentação, como para os métodos propostos por Shi e Govindaraju [SG97] e Pal et Al [BCP03] que já retornam a imagem segmentada, não é necessária a comparação com os caminhos ótimos. O resultado da segmentação é passado diretamente para o classificador.

O classificador citado neste capítulo, foi proposto por Oliveira [SO03], o mesmo se baseia em uma rede neural MLP. Vale lembrar que durante a criação da base de dados sintética, só foram utilizados dígitos isolados que foram classificados com sucesso por este classificador. Este fato incrementa o nível de confiança no resultado dos testes que foram realizados sobre todos os algoritmos que foram implementados. Isso se deve ao fato de sabermos que caso o algoritmo segmente a imagem com um caminho

ótimo de segmentação, ela será com certeza classificada corretamente pelo classificador. Com isso saberemos que imagens que forem consideradas erroneamente segmentadas, dificilmente o foram devido à erro de classificação (problema frequentemente encontrado nas avaliações de desempenho de métodos existentes na literatura).

Além da taxa de acerto na segmentação, também será feita uma análise básica do custo computacional de cada um dos algoritmos Para isso analisaremos o tempo gasto para segmentar cada imagem. Esta medida será feita para cada um dos quatro tipos de conexão propostos.

Para garantir que os algoritmos serão avaliados sobre as mesmas condições, todos serão testados utilizando o mesmo computador. Computador este que possui um processador Intel Core 2 Duo, modelo T5200 de 1.60Ghz e 2Mb de *cache* L2 e 2Gb de memória RAM DDR2 667Mhz. Este computador roda um sistema operacional Linux, sendo a distribuição Ubuntu versão 8.04.

Além do tempo necessário para segmentar cada imagem, será analisada também a quantidade de caminhos de segmentação gerados pelos métodos, pois no caso dos métodos baseados em reconhecimento, um maior número de caminhos de segmentação gera um maior número de possibilidades no grafo de hipóteses de segmentação, gerando com isso um maior número de chamadas ao classificador, aumentando consideravelmente o custo computacional do mesmo.

3.4 Implementação dos Métodos

Os seis métodos selecionados na seção 3.1 (Fujisawa et al [FNK92], Shi e Govindaraju [SG97], Oliveira et al [OLBS00], Chen e Wang [CW00], Pal et al [BCP03] e Elnagar e Alhajajj [EA03]), foram implementados utilizando a linguagem C++.

O módulo da verificação, o qual utiliza os caminhos ótimos de segmentação e rótulos das imagens, ambos contidos na base de dados, juntamente com a Rede Neural, foi implementado e é compartilhado por todos os métodos. O mesmo foi realizado para outras funções que são comuns à mais de um algoritmo. Com isso reduzimos o risco de diferenças na implementação causarem uma avaliação injusta entre os métodos de segmentação.

Os métodos processam todas as imagem da base de dados selecionada e geram os resultados em arquivos formato texto, de modo que não foi necessária a implementação de interface gráfica.

3.5 Resumo

Neste capítulo foi apresentada toda a metodologia proposta para o desenvolvimento deste trabalho de caracterização de algoritmos de segmentação de dígitos manuscritos, incluindo seleção dos algoritmos à serem utilizados, criação da base de dados e definição dos critérios utilizados na avaliação dos algoritmos, faltando apenas a análise comparativa dos resultados a qual é realizada no capítulo seguinte através de exibição de gráficos, tabelas e imagens comparativas dos resultados obtidos por cada um dos métodos implementados.

Capítulo 4

Resultados Experimentais

Conforme descrito no Capítulo 3, os algoritmos foram implementados e seus desempenhos testados com 79.466 imagens da base de dados sintética de pares de dígitos conectados. Para isto, avaliamos cada uma destas imagens, verificando se os pontos de segmentação gerados por cada algoritmo segmentaram corretamente o par de dígitos conectados em questão.

O objetivo dos experimentos é identificar pontos fortes e fracos de cada um dos métodos de segmentação analisados, utilizando para isso os quatro possíveis tipos de conexão de dígitos manuscritos descritos anteriormente.

Além da comparação global dos algoritmos, iremos compará-los quanto ao desempenho específico para cada tipo de imagem de entrada, conforme a classificação proposta no Capítulo 3, a qual é exemplificada pela Figura 3.5 da página 39.

No capítulo anterior foram descritos duas formas que seriam utilizadas para a avaliação dos algoritmos. Porém acabamos por não utilizar o método baseado na comparação direta do posicionamento dos pontos encontrados pelo algoritmo em relação aos pontos ótimos de segmentação. Esta decisão ocorreu pelo fato de que no primeiro teste utilizando este método, percebemos através de análise visual das imagens resultantes da segmentação que ele não é confiável, já que muitas imagens que deveriam ter sido classificadas como incorretamente segmentadas eram tidas como acertos e vice-versa.

Foram comparados também os algoritmos quanto ao número de caminhos de segmentação encontrados. Esta análise da quantidade de caminhos de segmentação gerados além de ser feita globalmente, foi realizada separadamente para cada um dos tipos (Figura 3.5) de conexão existentes. Esta comparação da quantidade de caminhos de segmentação gerados é um bom parâmetro para comparar o custo computacional dos algoritmos. Logicamente para se ter uma comparação completa e justa deste custo entre todos os algoritmos, teríamos que analisar todo o método e algoritmos intermediários utilizados. Porém como este não é o foco do projeto vamos nos restringir a compará-los de acordo com a quantidade de caminhos gerada e alguma análise superficial da

complexidade do algoritmo de forma geral.

Nas seções a seguir são detalhados os resultados obtidos para cada um dos métodos de segmentação e é também feita uma comparação entre todos os resultados.

4.1 Fujisawa et al

O método proposto por Fujisawa et al [FNK92] tenta primeiramente classificar os dígitos em "dígitos isolados" ou "dígitos conectados" antes de iniciar a segmentação. Como na base de dados utilizada nos testes todas as imagens correspondem a dígitos conectados, esta etapa foi eliminada.

Este método tem como particularidade extrair o contorno da imagem e o dividir em duas partes (superior e inferior) antes de extrair suas características. Além disso, este algoritmo antes de encontrar os pontos de segmentação, identifica as possíveis regiões de toque. Para identificar estas regiões o algoritmo calcula a chamada "Largura Vertical do Traço" e a compara com um limiar(H_x) pré-determinado. Isso pode ser considerado um ponto negativo deste algoritmo, pois o limiar precisa ser determinado de acordo com a base de dados à ser trabalhada. Para determinar este limiar foram realizados alguns testes variando o valor do mesmo, os quais resultaram em $H_x = 17$ como um bom valor à ser utilizado para a nossa base de dados.

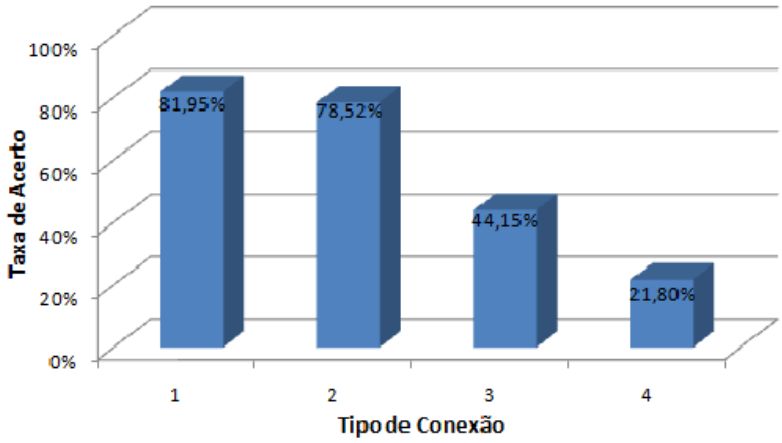


Figura 4.1: Desempenho obtido pelo método para cada um dos tipos de conexão.

No teste de desempenho obtivemos uma taxa global de 73,50% de acerto na segmentação. Vale lembrar que segundo o autor do método, ele não segmenta dígitos com conexão múltipla. Mesmo assim o método conseguiu atingir uma taxa de acerto de 21,8% para as imagens com esse tipo de conexão (tipo 4) como pode ser observado na Figura 4.1, a qual contém o detalhamento da taxa de acerto por tipo de conexão.

O algoritmo gera em média de 3,91 caminhos de segmentação para cada imagem processada, podemos visualizar a quantidade média e desvio padrão para cada tipo de

conexão na Figura 4.2

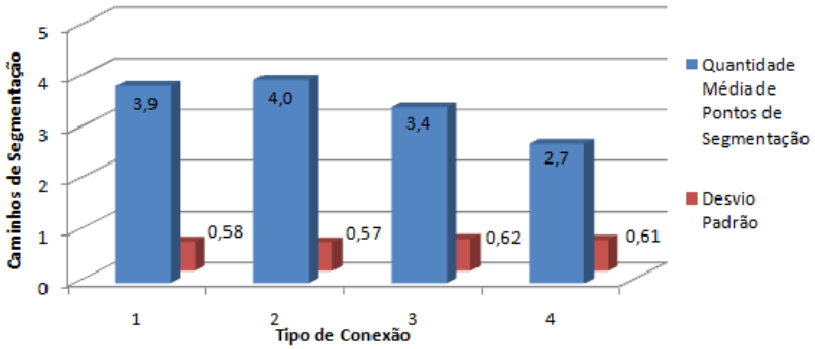


Figura 4.2: Quantidade média de caminhos de segmentação gerados pelo algoritmo, para cada tipo de conexão.

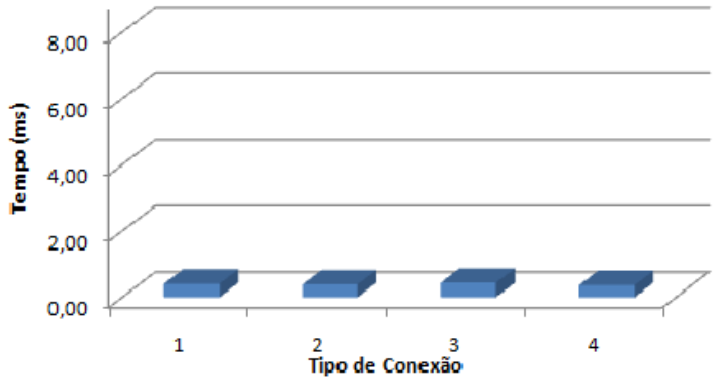


Figura 4.3: Tempo médio necessário para o algoritmo encontrar os caminhos de segmentação de cada imagem, exibidos de acordo com o tipo de conexão.

Analisando o tempo que este algoritmo precisa para encontrar os caminhos de segmentação de cada imagem pertencente à nossa base de dados, obtivemos um valor médio de 0,39ms por imagem, valor este que consideramos muito bom. Além de um tempo baixo de processamento, podemos observar na Figura 4.3 que este tempo varia muito pouco em relação ao tipo de conexão das imagens. Devemos lembrar que este tempo corresponde apenas à busca pelos caminhos de segmentação, fase esta que é o foco deste projeto. Ou seja, o tempo gasto para realizar a pré-classificação das imagens e o tempo necessário para escolher a melhor dentre as hipóteses de segmentação, não estão incluídos neste valor, pois para este projeto, não se faz necessária a utilização e comparação destas etapas do método.

4.2 Shi e Govindaraju

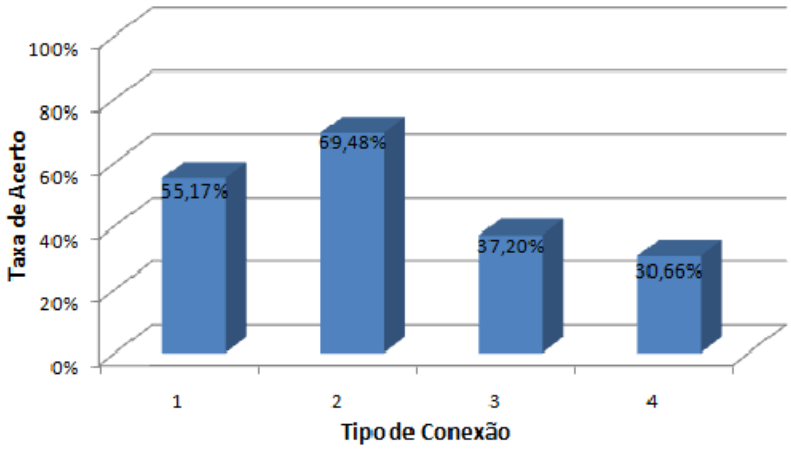


Figura 4.4: Desempenho obtido pelo método para cada um dos tipos de conexão.

O método proposto por Shi e Govindaraju [SG97] utiliza informações do traço dos dígitos e encontra curvaturas significantes no contorno da imagem para determinar os pontos de segmentação. O algoritmo utiliza a largura do traço e um limiar para determinar quando uma curvatura é ou não significante. Para determinar o valor deste limiar foi realizada uma sequência de testes com vários valores. Após os testes, resolveu-se por utilizar um valor que se convertido para graus corresponde a 75. Ou seja, as curvaturas com mais de 75 graus serão consideradas.

Este método alcançou um percentual de acerto global de 50,77%. O desempenho para cada tipo de conexão é exibido pelo gráfico presente na Figura 4.4.

Este método precisa em média de 1,31ms para encontrar os caminhos de segmentação para cada imagem de nossa base de dados. Assim como no método anterior, o tempo aferido para os diferentes tipos de conexão são muito próximos, o que mostra que o algoritmo mantém um mesmo comportamento, independente do tipo de imagem. O tempo de processamento tem variação quase nula, o valor médio, por tipo de conexão, e é exibido pela Figura 4.5.

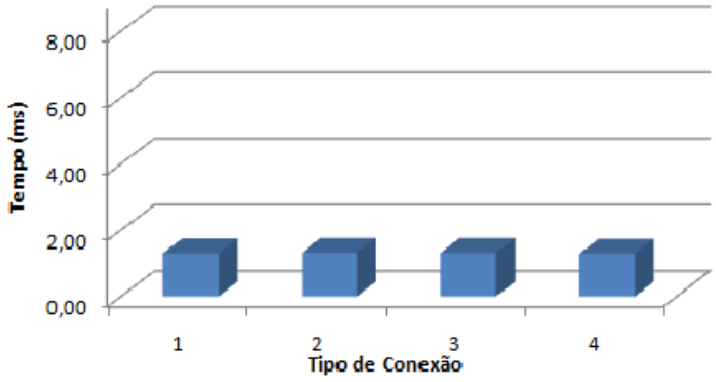


Figura 4.5: Tempo médio necessário para o algoritmo encontrar os caminhos de segmentação de cada imagem, exibidos de acordo com o tipo de conexão.

4.3 Oliveira et al

O algoritmo que utiliza basicamente informações de contorno e perfil para extração dos pontos de segmentação, proposto por Oliveira et al [OLBS00], atingiu um percentual de acerto global de 84,55%. O desempenho para cada tipo de conexão é exibido pelo gráfico presente na Figura 4.6. Em média o algoritmo gerou 3,09 caminhos de segmentação para cada imagem processada. A quantidade média de caminhos de segmentação gerados por tipo de conexão é exibida pelo gráfico presente na Figura 4.7.

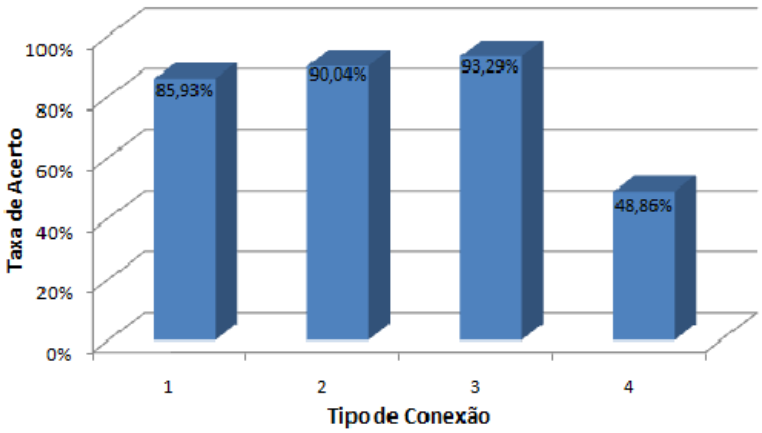


Figura 4.6: Desempenho obtido pelo método para cada um dos tipos de conexão.

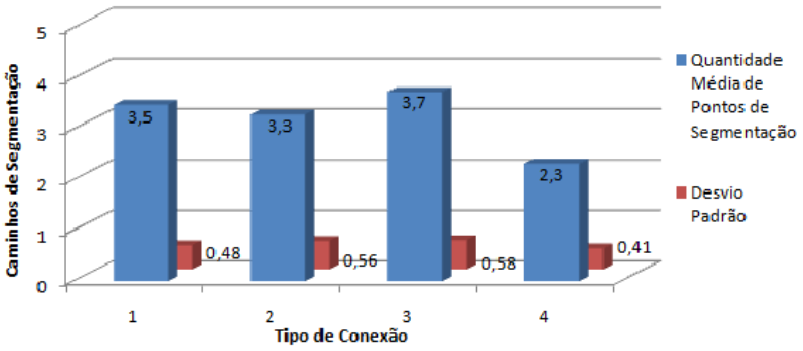


Figura 4.7: Quantidade média de caminhos de segmentação gerados pelo algoritmo, para cada tipo de conexão.

Este algoritmo precisou em média de 4,07ms para encontrar os caminhos de segmentação. Este tempo corresponde somente ao tempo de busca dos caminhos, ou seja, como este é um método baseado em reconhecimento, para um uso real deste algoritmo seria ainda necessária a utilização de um classificador para ajudar na escolha da melhor hipótese de segmentação. Este fato acarretaria em um aumento significativo do custo computacional. O tempo médio para encontrar os caminhos de segmentação, por tipo de conexão é exibido na Figura 4.8.

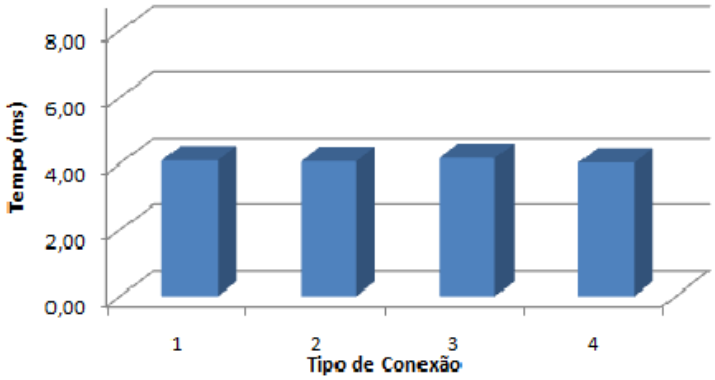


Figura 4.8: Tempo médio necessário para o algoritmo encontrar os caminhos de segmentação de cada imagem, exibidos de acordo com o tipo de conexão.

4.4 Chen e Wang

O algoritmo proposto por Chen e Wang [CW00] utiliza informações do plano de fundo e do contorno da imagem para obter os pontos de segmentação. Ele obtém a imagem de esqueleto de ambos, dígitos e fundo. Após encontrados os pontos de segmentação, um grande número de caminhos de segmentação é gerado. De acordo com os testes realizados, em média, são gerados 45,46 caminhos para cada imagem processada. Sendo cada um desses caminhos, compostos por dois ou mais pontos de segmentação. A quantidade de caminhos gerada por tipo de conexão pode ser observada na Figura 4.9.

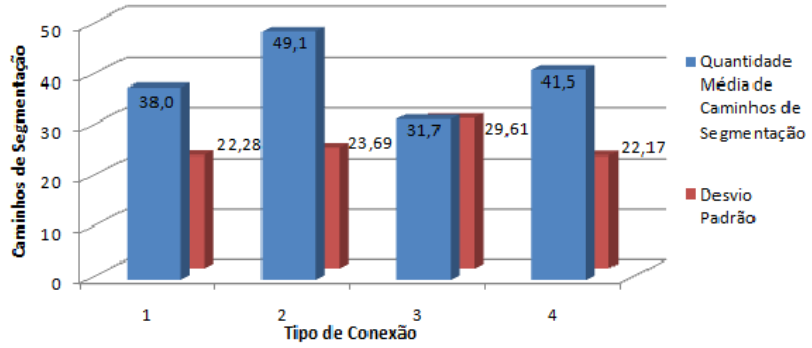


Figura 4.9: Quantidade média de caminhos de segmentação gerados pelo algoritmo, para cada tipo de conexão.

Para escolher o melhor caminho dentre os vários gerados os autores utilizam uma mistura de Gaussianas, fazendo assim com que o algoritmo gere uma única hipótese de segmentação como saída. O método de escolha baseado em Gaussianas necessita de treinamento, para isso os autores utilizaram uma base com 823 imagens, sendo que a base de testes possuía 3677. Para treinar a mistura de Gaussianas seria necessário realizar uma escolha visual dos melhores caminhos de segmentação, para todas as imagens da base de treinamento. Como nossa base de dados possui a informação de quais são os caminhos ótimos de segmentação, optamos por não utilizar essa mistura de Gaussianas. Ao invés disso, utilizamos a informação presente em nossa base para escolher o melhor caminho, da mesma maneira utilizada para os algoritmos que geram mais de um caminho de segmentação como saída.

Os testes desse algoritmo resultaram em um percentual de acerto global de 80,04%. O desempenho para cada tipo de conexão é exibido na Figura 4.10.

Destaca-se o fato de que este método utiliza duas esqueletizações (uma para o fundo e outra para a imagem) e além disso encontra uma grande quantidade de caminhos de segmentação no decorrer do processo, caminhos estes que terão que ser analisados pela mistura de Gaussianas para escolha do melhor dentre eles. Isto faz com que este método tenha um alto custo computacional.

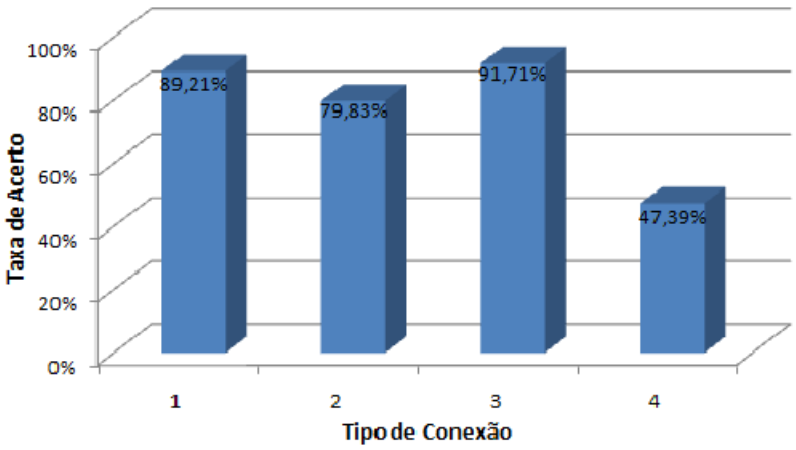


Figura 4.10: Desempenho obtido pelo método para cada um dos tipos de conexão.

Este algoritmo precisou em média de 74,8ms para encontrar os caminhos de segmentação de cada imagem. Este tempo não engloba a pré-classificação descrita pelos autores nem a seleção da melhor hipótese de segmentação. Não foi implementado o módulo de pré-classificação, devido ao fato de nossa base de dados conter apenas pares de dígitos conectados. Diferente dos métodos anteriormente descritos, este não apresentou uniformidade no tempo de processamento necessário para cada tipo de conexão, como pode ser observado na imagem 4.11. Este fato é explicado pelo alto e não uniforme número de caminhos de segmentação encontrados pelo algoritmo. Se observarmos a Figura 4.9, veremos que o tempo de processamento variou de acordo com o número de caminhos de segmentação encontrados durante o processo.

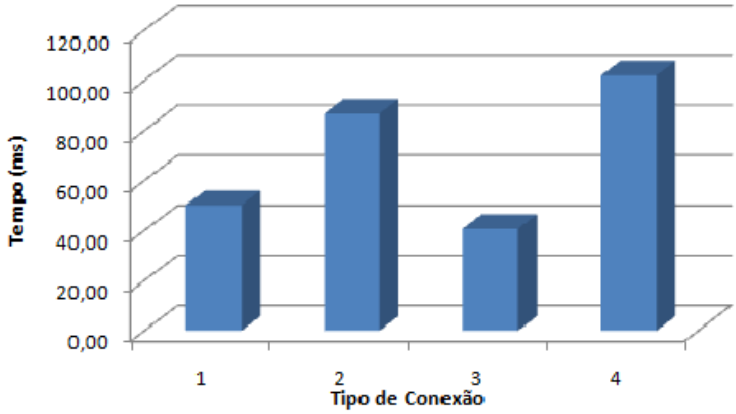


Figura 4.11: Tempo médio necessário para o algoritmo encontrar os caminhos de segmentação de cada imagem, exibidos de acordo com o tipo de conexão.

4.5 Pal et al

O método baseado no conceito de reservatórios, o qual foi proposto por Pal et al [BCP03], atingiu um percentual de acerto global de 64,50%. O desempenho para cada tipo de conexão é exibido pelo gráfico presente na Figura 4.12. Esse algoritmo gera apenas uma hipótese de segmentação.

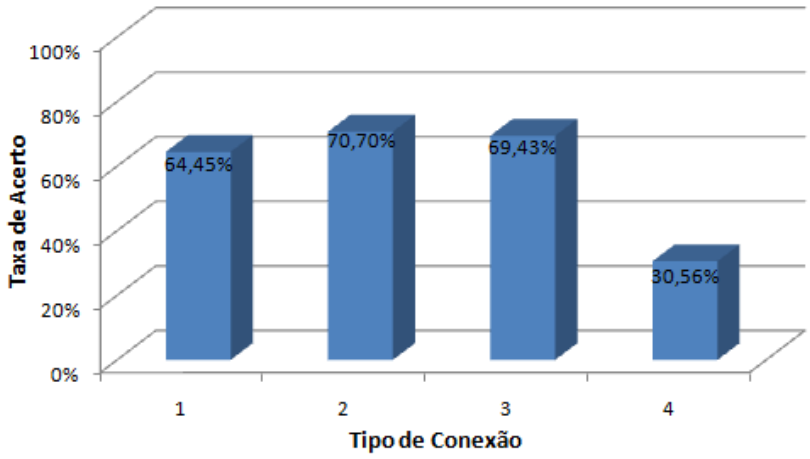


Figura 4.12: Desempenho obtido pelo método para cada um dos tipos de conexão.

Este algoritmo apresenta um novo conceito de característica, o qual se diferencia dos demais encontrados na literatura. Os autores chamam a região de onde serão obtidos os pontos de segmentação de "Reservatórios". Apesar do método proposto não ser um destaque com relação à taxa de acerto geral da segmentação este demora em média apenas 0,7ms para encontrar os pontos de segmentação de cada imagem. Este fato indica que estas características são alternativas econômicas quando se fala de custo computacional. Os tempos médios de processamento por tipo de conexão são exibidos na Figura 4.13. Deve-se lembrar que além do tempo de processamento ter sido baixo, este algoritmo gera apenas uma única hipótese de segmentação, o que elimina um custo computacional adicional para escolha do melhor caminho. Não podemos esquecer que este tempo de processamento citado, como nos métodos anteriormente descritos, não considera o algoritmo de pré-classificação.

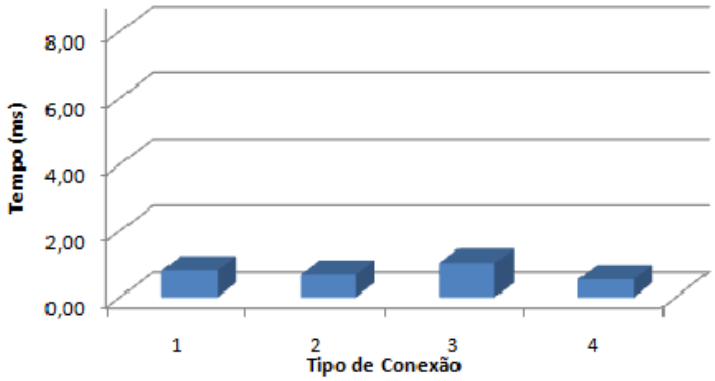


Figura 4.13: Tempo médio necessário para o algoritmo encontrar os caminhos de segmentação de cada imagem, exibidos de acordo com o tipo de conexão.

4.6 Elnagar e Alhajajj

O método proposto por Elnagar e Alhajajj [EA03] realiza esqueletização da imagem, e com o auxílio de de oito máscaras que são deslizadas sobre a imagem,

busca os pontos de características. Como cada máscara deve ser utilizada com todas as rotações múltiplas de $\frac{\pi}{2}$, o algoritmo precisa então realizar o processamento de 32 máscaras.

O método utiliza uma função para remoção de pontos de característica redundantes, a qual na maioria dos casos remove o "traço-ligador", quando este existe. Porém esta função não está bem descrita no artigo e por esse motivo sua implementação não pode ser realizada. A remoção de pontos redundantes não é essencial para o funcionamento do algoritmo, porém sua falta pode ter prejudicado o resultado final, principalmente casos onde exista o "traço-ligador". Como já citado no Capítulo 3, nossa base contém poucos casos de "traço-ligador", devido ao fato de ser uma base sintética. Por esse motivo, foi dado continuidade à implementação e avaliação deste algoritmo, mesmo não sendo possível a implementação da função de remoção dos pontos redundantes.

Para selecionar quais entre os pontos de característica devem ser utilizados, o algoritmo utiliza algumas regras para comparar esses pontos com o posicionamento dos pontos de mínimo e máximo locais, chamados pelos autores de "Ponto mais baixo do vale" e "ponto mais alto da colina".

Este método atingiu um percentual de acerto global de 62,25%. O desempenho para cada tipo de conexão é exibido pelo gráfico presente na Figura 4.14. Esse algoritmo sempre um único caminho de segmentação (dois pontos), independente da imagem e tipo de conexão.

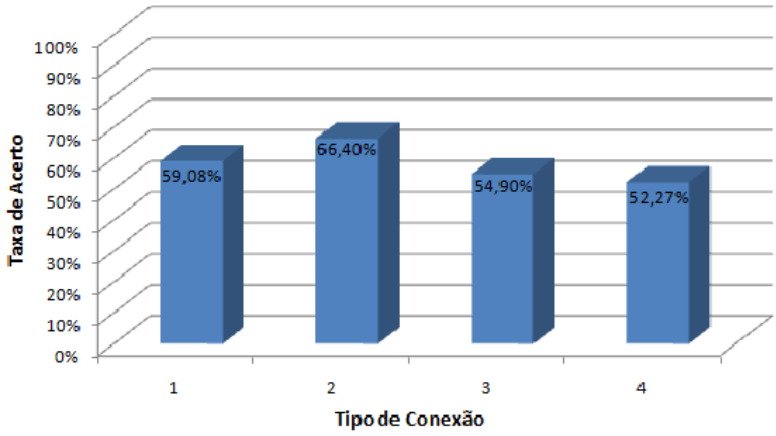


Figura 4.14: Desempenho obtido pelo método para cada um dos tipos de conexão.

Em média o algoritmo precisou de 7,5ms para encontrar o caminho de segmentação de cada imagem. O tempo médio por tipo de conexão é exibido na Figura 4.15. Apesar do tempo de processamento ser alto se comparado ao de alguns métodos já citados neste capítulo, os autores citam que a forma como foi implementada a busca pelos pontos de características (deslizamento de máscaras sobre a imagem) permite que a utilização destas máscaras seja paralelizada, o que causaria uma melhora no custo

computacional.

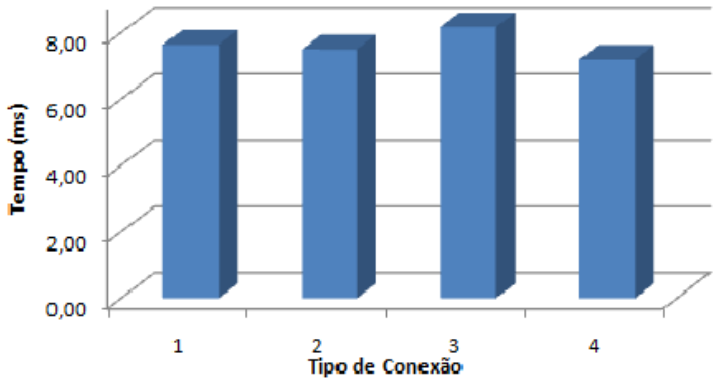


Figura 4.15: Tempo médio necessário para o algoritmo encontrar os caminhos de segmentação de cada imagem, exibidos de acordo com o tipo de conexão.

4.7 Avaliação e Comparação dos Resultados

Quando iniciamos o projeto esperávamos obter taxas de acerto globais bem próximas entre os algoritmos, pois era isso que indicava a tabela comparativa (tabela 2.1 da página 34) contendo os resultados reportados por cada autor. Sabíamos que o resultado dessa tabela não era preciso e não poderia ser utilizado como base para comparação dos métodos de segmentação. Porém não esperávamos que quando comparados os métodos com a mesma base de dados e utilizando a mesma forma de avaliação o resultado fosse tão diferente.

Tabela 4.1: Tabela contendo quantidade de caminhos de segmentação, tempo de processamento, taxa de acerto de acordo com nossos testes e o acerto informado pelos autores.

Métodos	**Aferido**	**Autor**	**Qtd Caminhos**	**Tempo (ms)**
Oliveira et al [OLBS00]	84,55%	98,5%	Média 4,07	3,9ms
Chen e Wang [CW00]	80,04%	96%	Sempre 1	74,8ms
Fujisawa et al [FNK92]	73,50%	95,0%	Média 3,66	0,4ms
Pal et al [BCP03]	64,50%	94,8%	Sempre 1	0,7ms
Elnagar e Alhajajj [EA03]	62,25%	96%	Sempre 1	7,5ms
Shi e Govindaraju [SG97]	50,77%	80%	Sempre 1	1,2ms

Na tabela 4.1 podemos ver a diferença entre os desempenhos citados pelos autores e o desempenho encontrado nos nossos testes. Para a realização dos nossos testes, a base de dados utilizada foi a mesma para todos os algoritmos. Além disso, para avaliar cada resultado de segmentação foi utilizado o mesmo classificador (rede neural). Nessa tabela é também exibida a quantidade de caminhos de segmentação gerados por cada um dos algoritmos e o tempo médio que cada um deles precisa para encontrar as hipóteses de segmentação.

No estágio atual deste trabalho, o dado que mais nos interessa é a taxa de acerto dos algoritmos de acordo com os diferentes tipos de conexão. Porém nem por isso deixamos de realizar uma análise básica do custo computacional dos mesmos, visto que esta informação pode vir à ser importante em trabalhos futuros.

Ainda observando a tabela 4.1, podemos ver que o método que foi mais rápido na tarefa de buscar os pontos de segmentação, foi o proposto por Fujisawa et al[FNK92]. Este método obteve um tempo 43% menor que o segundo método mais rápido, o proposto por Pal et al [BCP03]. Porém o segundo método entrega como resultado apenas uma hipótese de segmentação, enquanto o primeiro entrega em média 3,66 caminhos de segmentação.

O custo computacional atribuído à escolha da melhor hipótese de segmentação, varia de acordo com o tipo de classificador escolhido e número de chamadas realizadas ao classificador. A quantidade destes acessos, varia dependendo de quantas sub-imagens foram geradas no grafo que representa todas as hipóteses de segmentação, conforme já foi explicado no Capítulo 2 através da Figura 2.2 localizada na página 10. Para termos uma idéia mais concreta do custo que o processo de escolha da melhor hipótese representa, tomemos como referência a Rede Neural utilizada em nossos testes, a qual leva em média 28ms para avaliar cada imagem. Ou seja, independente de quantas chamadas seríam necessárias à rede neural, o método proposto por Fujisawa et al perderia em tempo de processamento para o método proposto por Pal et al, devido ao fato do tempo de processamento do classificador ser muito maior que o tempo necessário para localização dos caminhos de segmentação.

Deste modo, percebemos que se considerado um caso real de uso dos métodos de segmentação, o método que proporcionaria o menor custo computacional do processo como um todo, seria o proposto por Pal et al, o qual precisa em média de apenas 0,7ms para encontrar o caminho de segmentação. Sempre produzindo como saída um único caminho. Porém a taxa de acerto geral deste método é 20,05% menor que do método proposto por Oliveira et al [OLBS00].

Percebe-se que para escolher um método de segmentação para determinada aplicação, existe uma "balança" onde deve ser pesado o desempenho do algoritmo com relação ao seu custo computacional. E a melhor relação vai depender das exigências de cada aplicação.

Agora com relação apenas à taxa de acerto dos métodos, podemos ver que na Figura 4.16 é exibido um gráfico de comparação desta taxa de todos os métodos implementados de acordo com cada tipo de conexão.

Podemos observar que apesar do método proposto por Oliveira et al [OLBS00] ter alcançado uma taxa de acerto global da segmentação superior aos demais (entenda-se taxa de acerto global, como a taxa de acerto obtida para todas as imagens da nossa

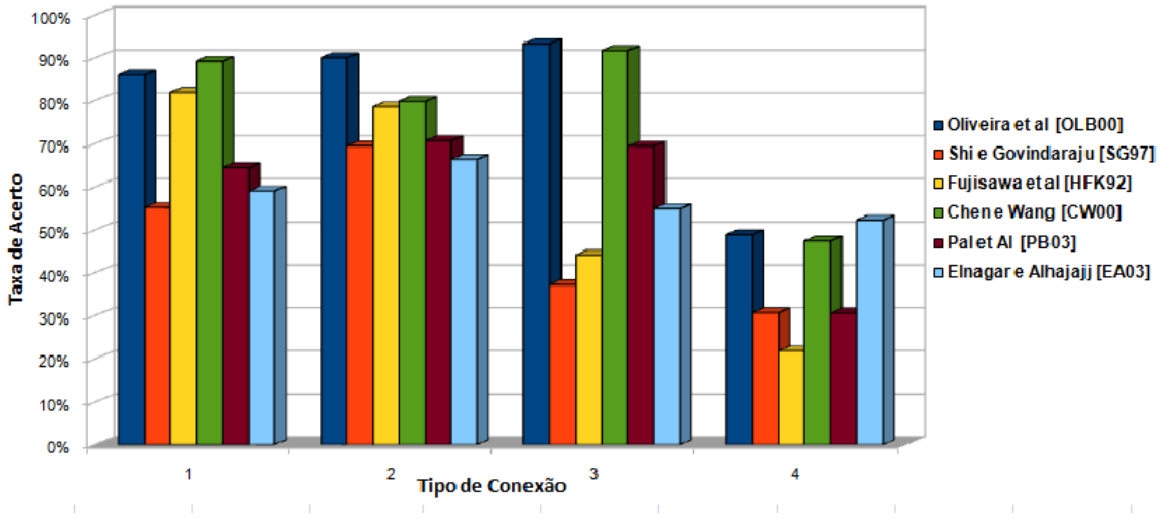


Figura 4.16: Comparativo das taxas de acerto dos algoritmos implementados.

base de dados), para dois dos tipos de segmentação (tipos 1 e 4) existem algoritmos que alcançaram uma taxa de acerto específica maior que o método proposto por Oliveira.

Para as imagens com conexão de tipo 1, o algoritmo com melhor desempenho foi o proposto por Chen e Wang [CW00], o qual ocupa a segunda posição no ranking do acerto global. Já para o tipo 4 de conexão, o melhor desempenho veio do algoritmo proposto por Elnagar et al [EA03], o qual no desempenho global ocupa a quinta posição. Isto é explicado pelo fato de que apenas 10% das imagens da nossa base de dados pertence ao tipo 4 (Figura 3.8).

Na tabela 4.2 a taxa de acerto por tipo de conexão é detalhada em função de quantos métodos acertaram determinado número de imagens, por exemplo "2/6" significa a quantidade de imagens que apenas dois dos seis algoritmos acertou. Podemos observar na última linha da tabela, 97,82% das imagens foram segmentadas por ao menos um dos algoritmos.

Tabela 4.2: Detalhamento das quantidades de acerto dos algoritmos.

	Geral	%	Tipo 1	%	Tipo 2	%	Tipo 3	%	Tipo 4	%
	79.466		27.656	34,80%	42.609	53,62%	1.266	1,59%	7.935	9,99%
0/6	1.734	2,18%	95	0,34%	576	1,35%	1	0,08%	1.063	13,40%
1/6	3.711	4,67%	444	1,61%	1.398	3,28%	34	2,69%	1.835	23,13%
2/6	6.239	7,85%	1.720	6,22%	2.615	6,14%	116	9,16%	1.788	22,53%
3/6	10.809	13,60%	3.884	14,04%	5.201	12,21%	260	20,54%	1.464	18,45%
4/6	18.511	23,29%	7.224	26,12%	9.850	23,12%	385	30,41%	1.052	13,26%
5/6	23.160	29,14%	8.777	31,74%	13.500	31,68%	328	25,91%	555	6,99%
6/6	15.302	19,26%	5.512	19,93%	9.470	22,23%	142	11,22%	178	2,24%
Min 1	77.732	97,82%	27.561	99,66%	42.034	98,65%	1.265	99,92%	6.872	86,60%

Os casos que apenas um dos seis algoritmos conseguiu segmentar corretamente (3.711 imagens), linha "1/6" da tabela acima, são detalhados de acordo com qual dos

algoritmos segmentou, na tabela 4.3 abaixo.

Tabela 4.3: Detalhamento das quantidades de acerto das imagens que foram segmentadas por apenas um dos algoritmos.

Algoritmo	Geral	%	Tipo 1	%	Tipo 2	%	Tipo 3	%	Tipo 4	%
[EA03]	1.062	28,62%	27	6,08%	420	30,04%	2	5,88%	613	33,41%
[OLBS00]	945	25,62%	57	12,84%	498	35,62%	15	44,12%	375	20,44%
[CW00]	840	22,64%	157	35,36%	194	13,88%	9	26,47%	480	26,16%
[FNK92]	339	9,14%	134	30,18%	138	9,87%	4	11,76%	63	3,43%
[BCP03]	277	7,46%	24	5,41%	86	6,15%	1	2,94%	166	9,05%
[SG97]	248	6,68%	45	10,14%	62	4,43%	3	8,82%	138	7,52%

Com esta tabela podemos perceber que todos os métodos, independente de seu desempenho, possuem casos de acerto que nenhum outro método segmentou corretamente. Ou seja, todos eles tem capacidade de fornecer alguma contribuição no caso de um sistema que combinasse vários métodos de segmentação.

Outro dado interessante que pode ser observado na tabela 4.3 é que o método proposto por Elnagar [EA03], mesmo tendo o pior desempenho dentre os algoritmos para o tipo 2 de conexão, é o segundo método que mais possui imagens que só ele segmentou corretamente para este tipo.

Na Figura 4.17 podemos observar um exemplo de imagem que nenhum algoritmo conseguiu segmentar corretamente e o resultado fornecido por cada um dos algoritmos. A Figura (a) representa a imagem original. Cada par de Figuras subsequentes representa os dígitos esquerdo e direito produzidos por cada um dos algoritmos.

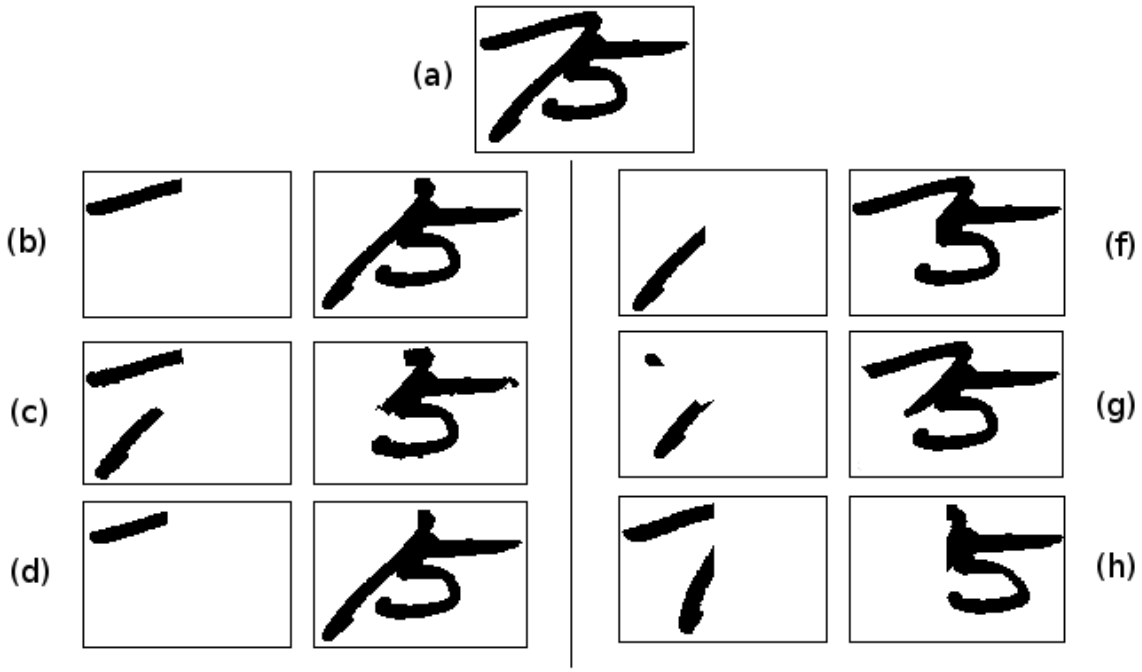


Figura 4.17: Exemplo de imagem que nenhum algoritmo segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

Nas Figuras seguintes (4.18, 4.19, 4.20, 4.21, 4.22 e 4.23) podemos observar exemplos de imagens que apenas um dos algoritmos avaliados conseguiu segmentar corretamente bem como os resultados fornecidos por cada um dos métodos de segmentação, utilizando o mesmo padrão de exibição da imagem anterior, porém adicionado uma borda mais espessa ao resultado que exibe uma segmentação correta da imagem.

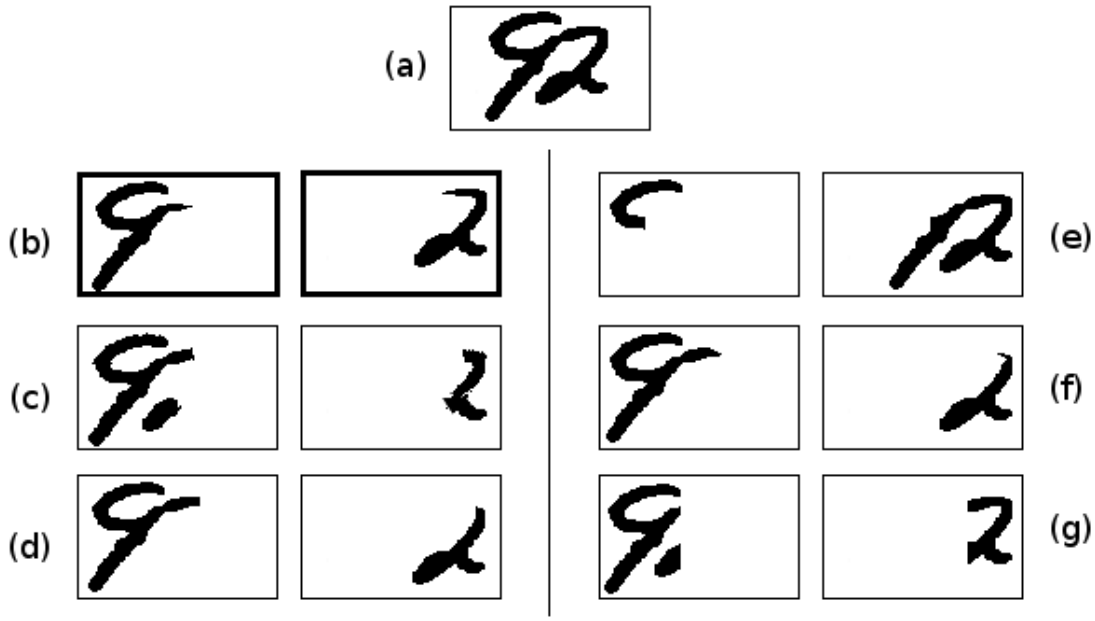


Figura 4.18: Exemplo onde apenas o algoritmo proposto por Oliveira et al segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

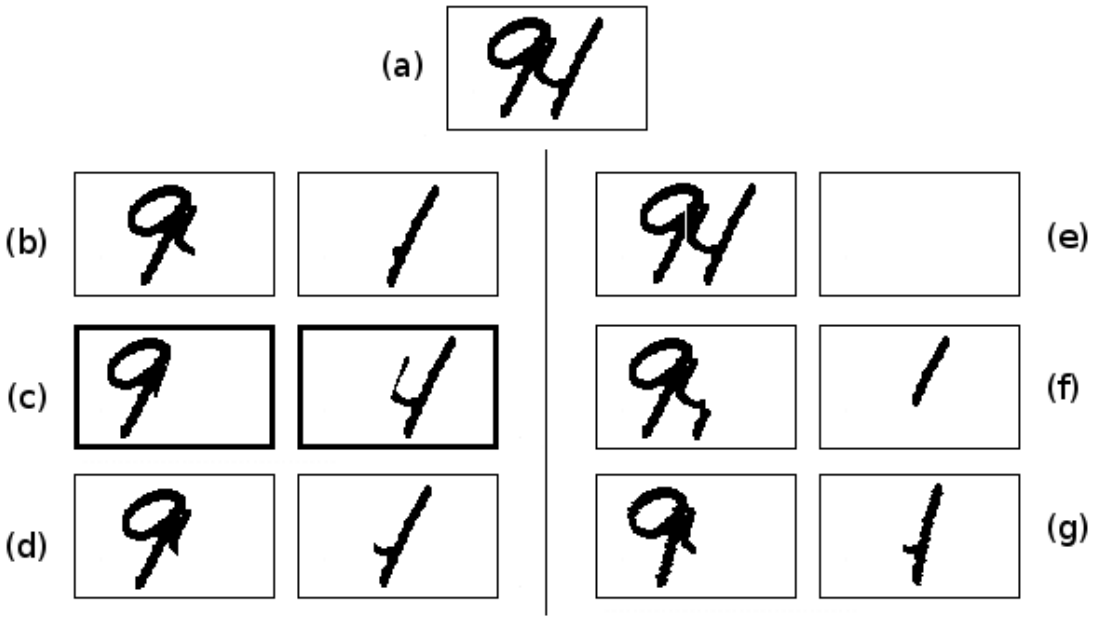


Figura 4.19: Exemplo onde apenas o algoritmo proposto por Shi e Govindaraju segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

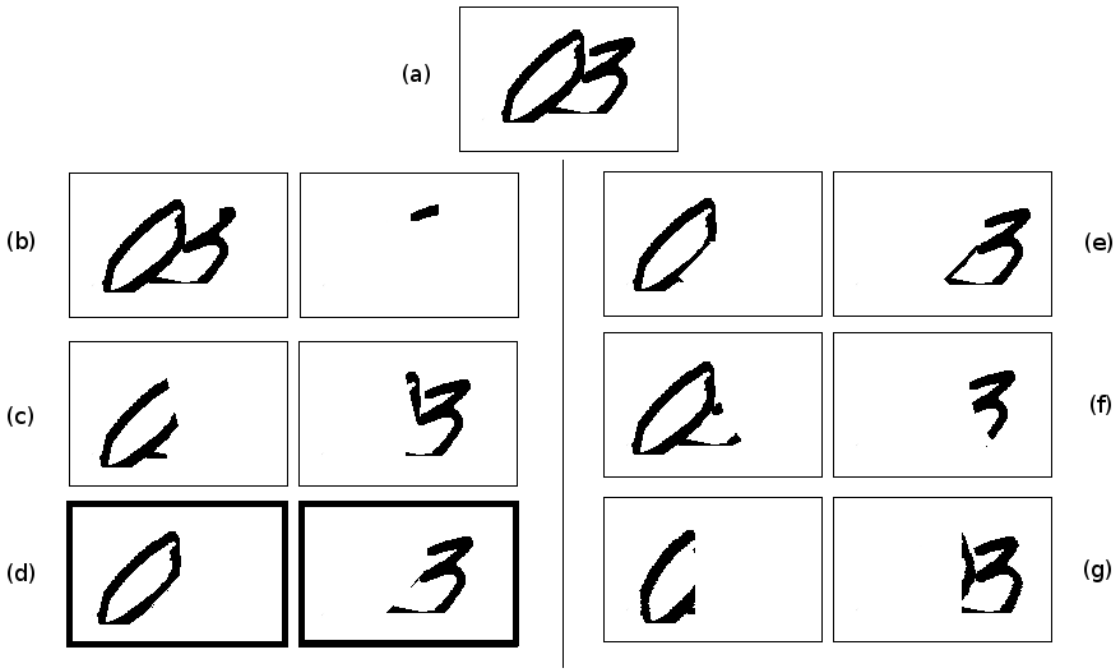


Figura 4.20: Exemplo onde apenas o algoritmo proposto por Fujisawa et al segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

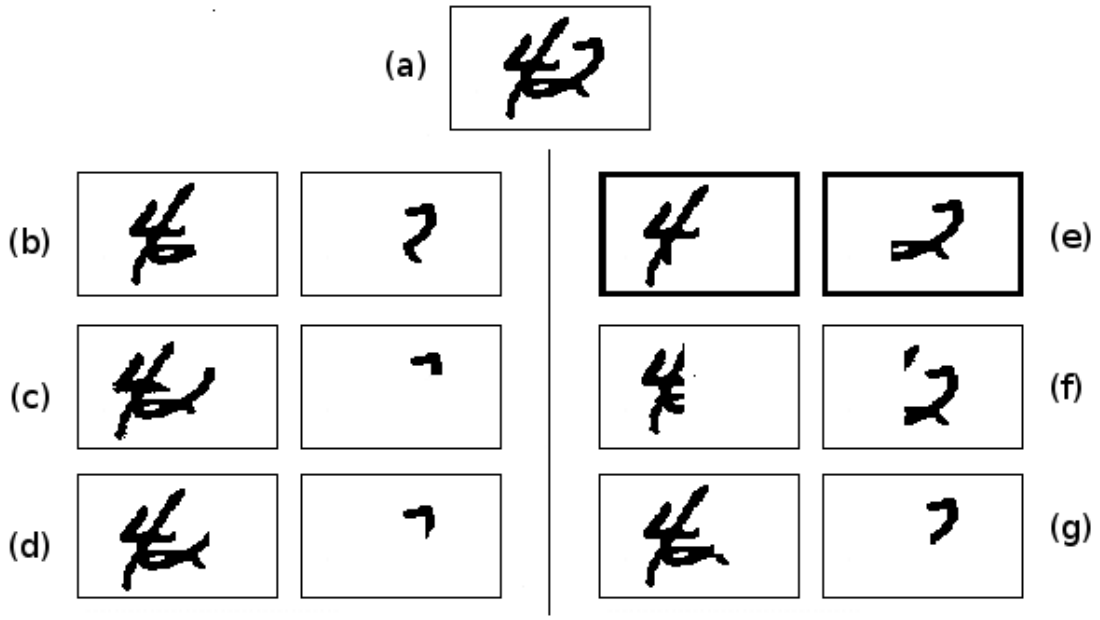


Figura 4.21: Exemplo onde apenas o algoritmo proposto por Pal et al segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

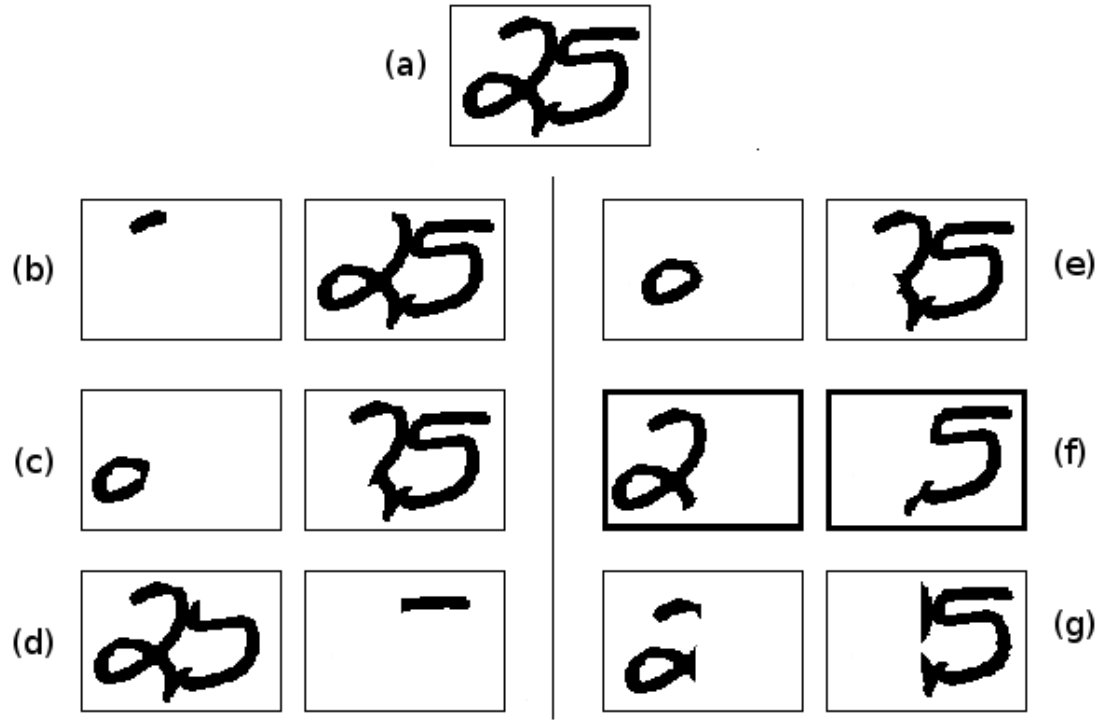


Figura 4.22: Exemplo onde apenas o algoritmo proposto por Chen e Wang segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

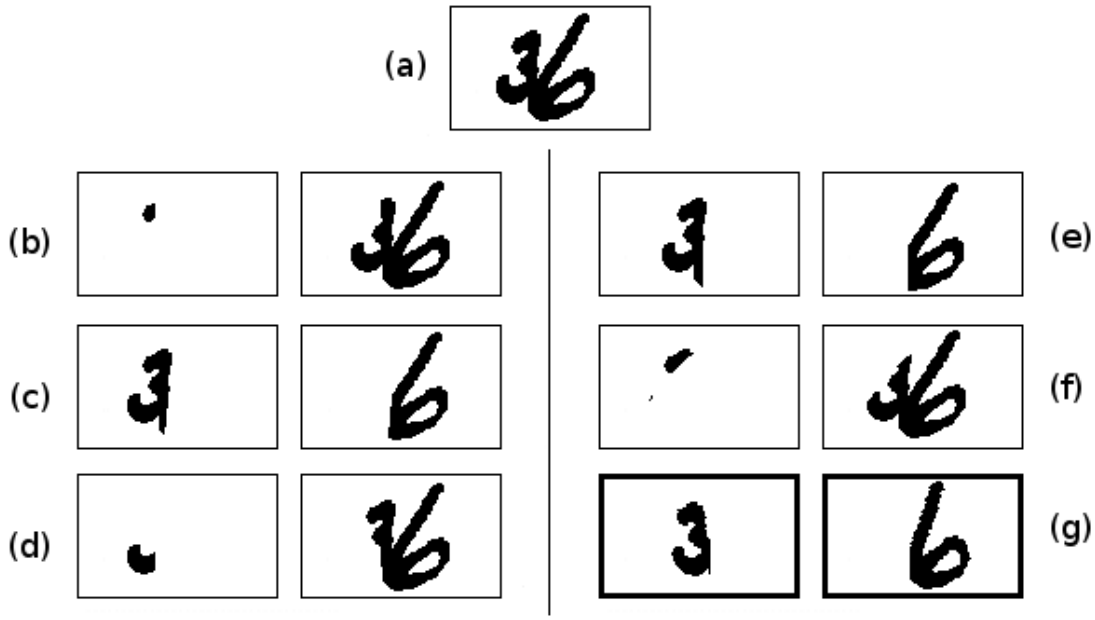


Figura 4.23: Exemplo onde apenas o algoritmo proposto por Elnagar e Alhajajj segmentou corretamente. (a) Original. (b) [OLBS00], (c) [SG97], (d) [FNK92], (e) [BCP03], (f) [CW00], (g) [EA03]

Neste capítulo foram apresentados os resultados obtidos nos testes comparativos dos métodos de segmentação implementados. Os métodos foram comparados quanto à taxa de acerto na busca pelos pontos de segmentação, quantidade de caminhos de segmentação encontrados e tempo de processamento. Esses quesitos foram avaliados levando em conta os quatro tipos de segmentação propostos no capítulo anterior. No próximo capítulo é apresentada a conclusão do trabalho e uma descrição dos trabalhos futuros.

Capítulo 5

Conclusão

Neste documento apresentamos uma comparação de alguns algoritmos de segmentação de pares de dígitos manuscritos. Diversas comparações já foram realizadas por vários autores, porém elas comparam os métodos sem estipular uma base de dados padrão para isso. Apesar de alguns algoritmos serem testados sobre imagens de bases de dados conhecidas, como a base NIST SD19, nem sempre são utilizadas as mesmas quantidades de imagens para os testes, o que acaba prejudicando a comparação. Os algoritmos pesquisados, os quais foram descritos de maneira sucinta no capítulo 2, utilizam entre 900 a 5000 imagens em seus testes, o que consideramos um número pequeno para uma boa comparação de desempenho. Outro ponto que prejudica as comparação existentes é que muitas vezes são utilizados diferentes classificadores para avaliar o resultado da segmentação dos dígitos. Isto faz com que as comparações não sejam precisas, tornando seus resultados muito discutíveis.

Por isso neste trabalho foi realizada a comparação dos algoritmos utilizando a mesma base de dados. Base esta que contém 79.466 imagens de pares de dígitos conectados. Esta quantidade torna os experimentos mais próximos do que se encontraria em casos de uso real dos algoritmos, aumentando assim a credibilidade da comparação. Para deixar a comparação mais detalhada, classificamos nossa base de dados de acordo com o tipo de conexão da imagem, em quatro diferentes tipos. Outro diferencial do nosso trabalho é que a base de dados utilizada possui informação de quais são os caminhos ótimos de segmentação, fazendo assim com que pudéssemos melhor avaliar os resultados dos algoritmos.

Após a comparação dos métodos implementados, pudemos ter uma boa visão do desempenho de cada um. O método proposto por Oliveira et al [OLBS00] foi o que alcançou o melhor desempenho geral na segmentação dos dígitos, com uma taxa de acerto de 84,55%. Porém, mesmo tendo alcançado o melhor desempenho global, não foi o melhor em todos os casos quando analisado o desempenho isolado para cada um dos tipos de segmentação. Para os tipos de conexão um e quatro, os

melhores métodos foram os propostos por Chen e Wang [CW00] e Elnagar et al [EA03], respectivamente. Outra informação interessante que pudemos obter observando os resultados comparativos, foi que mesmo o método com desempenho mais baixo (Shi and Govindaraju [SG97]) segmentou algumas imagens que não haviam sido segmentadas por nenhum dos outros métodos avaliados.

Diante de todos esses dados obtidos sobre cada um dos métodos de segmentação avaliados, podemos concluir que tivemos sucesso em nosso trabalho e que a idéia de utilizar os tipos de conexão de dígitos faz sentido e pode ser explorada em trabalhos futuros.

5.1 Trabalhos Futuros

Visto que nosso método de comparação dos algoritmos de segmentação gerou resultados bastante interessantes e existe um vasto número de algoritmos presentes na literatura, seria muito interessante dar continuidade a este trabalho, adicionando outros algoritmos nesta comparação. Isto com certeza traria resultados valiosos para todos que trabalham na área de segmentação de dígitos manuscritos.

Outra coisa que pode ser feita para incrementar este trabalho seria a adição de outros parâmetros na comparação dos algoritmos de segmentação, como por exemplo, uma análise minuciosa da complexidade de cada método.

Seria interessante também verificar se os tipos de conexão por nós utilizados na avaliação dos algoritmos são os mais indicados, pois tendo a relação completa das imagens que cada algoritmo segmentou podemos analisar se existe outra forma de melhor classificar nossa base de dados, do que os quatro tipos que utilizamos. Os resultados obtidos levam a crer que a forma que dividimos a base de dados é bastante adequada, mas mesmo assim não se pode descartar a hipótese de existir outra maneira ainda mais interessante de classificá-la.

Após a adição de mais métodos de segmentação nesta comparação, um excelente trabalho a ser feito será o desenvolvimento de um sistema para combinar os métodos de segmentação já existentes buscando uma elevação do desempenho. Observando os resultados atuais podemos observar que se os seis algoritmos de segmentação analisados fossem combinados de maneira perfeita, alcançaríamos uma taxa de acerto geral de 97,82%, o que representaria um aumento na taxa de acerto de 13,27% em relação ao algoritmo com melhor desempenho geral. Se considerarmos apenas os casos de imagens com conexão múltipla (tipo 4 da nossa comparação), o ganho obtido com esse "sistema ideal" seria ainda mais expressivo, 34,33% de aumento na taxa de acerto da segmentação.

Referências Bibliográficas

[AYV98] N Arica and F T Yarman-Vural. A new scheme for off-line handwritten connected digit recognition. In *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition*, volume 2, pages 1127–1129, Washington, DC, USA, 1998. IEEE Computer Society.

[BCP03] A. Bela, C. Choisy, and U. Pal. Water reservoir based approach for touching numeral segmentation. *Pattern Recognition Letters*, 24(1):261–272, January 2003.

[Bri01] Alceu S. Jr. Britto. *A Two-Stage HMM-Based Method for Recognizing Handwritten Numeral Strings*. PhD thesis, Pontifícia Universidade Católica do Paraná, Departamento de Informática, Curitiba, Junho 2001.

[BSBS03] A. S. Britto, R. Sabourin, F. Bortolozzi, and C. Y. Suen. The recognition of handwritten numeral strings using a two-stage hmm-based method. *International Journal on Document Analysis and Recognition*, 5(2-3):102–117, 2003.

[CL96] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):690–706, 1996.

[CO99] Soon-Man Choi and Il-Seok Oh. A segmentation-free recognition of two touching numerals using neural network. In *ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 253–256, Washington, DC, USA, 1999. IEEE Computer Society.

[CW00] Yi-Kai Chen and Jhing-Fa Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1304–1317, 2000.

[EA03] Ashraf Elnagar and Reda Alhajj. Segmentation of connected handwritten numeral strings. *Pattern Recognition*, 36(3):625–634, March 2003.

[Fen91] Richard K. Fenrich. Segmentation of automatically located handwritten numeric strings. In *2nd International Workshop on Frontiers in Handwriting Recognition*, pages 33–44, Bonas, France, 1991.

[FNK92] H. Fujisawa, Y. Nakano, and K. Kurino. Segmentation methods for character recognition: from segmentation to document structure analysis. volume 80, pages 1072–1092. IEEE Computer Society, 1992.

[LLDF04] Yun Lei, C. S. Liu, X. Q. Ding, and Qiang Fu. A recognition based system for segmentation of touching handwritten numeral strings. *International Workshop on Frontiers in Handwriting Recognition*, 00:294–299, 2004.

[OBJS05] Luiz S. Oliveira, Alceu S. Britto Jr., and Robert Sabourin. A synthetic database to assess segmentation algorithms. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, volume 1, pages 207–211. IEEE Computer Society, 2005.

[OLBS00] Luiz E. Soares Oliveira, Edouard Lethelier, Flávio Bortolozzi, and Robert Sabourin. A new segmentation approach for handwritten digits. In *ICPR '00: 15th International Conference on Pattern Recognition*, pages 2323–2326, 2000.

[SG97] Zhixin Shi and Venu Govindaraju. Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition*, 30(9):1501–1504, 1997.

[SN04] Misako Suwa and Satoshi Naoi. Segmentation of handwritten numerals by graph representation. In *IWFHR '04: Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 334–339, Washington, DC, USA, 2004. IEEE Computer Society.

[SO03] Luiz E. Soares Oliveira. *Automatic Recognition of Handwritten Numerical Strings*. PhD thesis, University of Quebec, ETS, Montreal, QC, Canada, Julho 2003.

[SSB07] Javad Sadri, Ching Y. Suen, and Tien D. Bui. A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. *Pattern Recognition*, 40(3):898–919, 2007.

[WGS00] Xian Wang, Venu Govindaraju, and Sargur N. Srihari. Holistic recognition of handwritten character pairs. *Pattern Recognition*, 33(12):1967–1973, 2000.

[YY01] Donggang Yu and Hong Yan. Separation of touching handwritten multi-numeral strings based on morphological structural features. *Pattern Recognition*, 34(3):587–599, 2001.