ADERLY TEREZINHA STRESSER BORGES

LOCALIZAÇÃO DO CEP NO BLOCO DO ENDEREÇO EM ENVELOPES POSTAIS

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

CURITIBA

2004

ADERLY TEREZINHA STRESSER BORGES

LOCALIZAÇÃO DO CEP NO BLOCO DO ENDEREÇO EM ENVELOPES POSTAIS

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

Área de Concentração: *Metodologias e Técnicas de Computação.*

Orientador: Prof. Dr. Flavio Bortolozzi
Co-orientadores: Prof. Dr. Dr. Alceu Britto Jr.
Prof. Dr. Robert Sabourin

CURITIBA
2004

Borges, Aderly Terezinha Stresser.

Localização do CEP no bloco endereço em envelopes postais. Curitiba, 2004. 92p.

Dissertação - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Localização do CEP. 2. Modelos Escondidos de *Markov*. 3. Meta-Classes. 4. *Level Building* I. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada II-t

Agradecimentos

A minha família e aos meus amigos pelo carinho, incentivo e conforto nos momentos difíceis.

Ao meu orientador, professor Dr. Alceu Britto, pela paciência, orientação e amizade desenvolvida durante o mestrado. Aos professores Dr. Robert Sabourin e Dr. Flávio Bortolozzi pela orientação e contribuições.

Ao professor Dr. Guilherme Vilar pela participação na banca examinadora. A todos os professores do curso de mestrado pelo conhecimento compartilhado.

A empresa Perform Informática por ter me permitido um horário de trabalho diferenciado. Aos colegas de trabalho Marcos e Gisele pela amizade e apoio decisivos para a realização deste trabalho.

Aos colegas de mestrado e de laboratório pela amizade e pelas trocas de informações que ajudaram no desenvolvimento deste trabalho.

A Pontifícia Universidade Católica do Paraná.

E a todos que de alguma forma contribuíram para a realização deste trabalho

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

CEDAR	**Center of Excellence in Document Analysis Recognition**
CEP	**Código de Endereçamento Postal**
CC	**Componentes Conexos**
ECC	**Estado, Cidade e CEP**
LBA	***Level Building Algorithm***
MEM	**Modelos Escondidos de Markov**
MLP	**MultiLayer Perceptrons**
MV	**Máxima Verossimilhança**
VQ	***Vector Quantization***

Resumo

Este trabalho propõe um método para a localização do CEP no bloco do endereço de envelopes postais manuscritos e avalia diferentes estratégias para a modelagem das informações que compõem o bloco do endereço. O método proposto consiste na segmentação do bloco do endereço em linhas, seguida pela segmentação e reconhecimento dos componentes de cada linha. Uma vez identificados os componentes das linhas, a localização do CEP consiste na busca de uma seqüência de 5 dígitos mais uma seqüência de 3 dígitos, separadas ou não pelo traço. Modelos Escondidos de *Markov* e o algoritmo *Level Building* são utilizados para encontrar as melhores hipóteses de segmentação-reconhecimento da linha que está sendo processada, utilizando uma estratégia de segmentação implícita. Num segundo estágio, as hipóteses de segmentação-reconhecimento são verificadas e re-classificadas. O bloco do endereço é composto de uma grande variedade de informações (palavras, dígitos, caracteres, traços, ruídos, etc.) dificultando o processo de modelagem destas informações. Para amenizar este problema, neste trabalho são avaliadas diferentes estratégias de modelagem para os componentes de uma linha do bloco do endereço, utilizando o conceito de Meta-Classes. Os melhores resultados foram obtidos com as seguintes Meta-Classes: $M_{0...9}$ (agrupa todos os dígitos de 0 a 9), M_{PC} (agrupa palavras e caracteres), M_H (o traço que separa os dígitos do CEP), M_{CEP} (a palavra CEP). Nos modelos criados para o primeiro estágio do sistema (segmentação-reconhecimento) foi obtida uma taxa de reconhecimento de 80,64%. Modelos similares treinados com informações complementares criados para o segundo estágio obtiveram uma taxa de reconhecimento de 85,38%.

Palavras-Chave: Localização do CEP, Modelos Escondidos de *Markov*, Meta-Classes, *Level Building*.

Abstract

This work presents a method for Zip Code location on address block of handwritten postal envelopes and also evaluates different strategies of modeling the components present in the address block. The proposed method starts by segmenting the address block into lines in order to execute the segmentation and recognition of the components of each line. Afterwards, the location of the ZIP CODE consists of searching for a sequence of 5 digits plus a sequence of 3 digits, separate or not by a dash. Hidden Markov Models and the Level Building Algorithm are used for finding the best segmentation-recognition hypothesis of the line being processed taking into account implicit segmentation-based strategy. In a second stage, the segmentation-recognition hypotheses are verified and re-classified. The address block presents a great variety of information (words, digits, character, lines, noises, etc.) hindering the modeling task. To deal with this problem, different modeling strategies for the address block components are evaluated. The best results were obtained by using the following Meta-Classes: $M_{0...9}$ (digits from 0 to 9), M_{PC} (words and characters), M_H (dash), and M_{CEP} (word CEP). The models constructed for the first stage of the system (segmentation-recognition) provide a recognition rate of 80.64%. Similar models trained with complementary information for the second stage of the system provide 85.38% of recognition rate.

Keywords: Zip Code Location, Hidden Markov Models, Meta-Classes, Level Building.

1. Introdução

Muitos estudos têm sido realizados nos últimos anos na área de análise e reconhecimento de documentos, mais especificamente na área de reconhecimento de documentos manuscritos. Este esforço é justificado devido ao grande número de aplicações em potencial, tais como a leitura de cheques, formulários, o endereço e o CEP em envelopes postais, verificação de assinaturas, etc.

Entre estas possíveis aplicações, o reconhecimento do CEP em envelopes postais é uma área de grande interesse devido ao seu grande valor prático, pois a partir da sua leitura automática pode-se melhorar o processo de triagem postal feito pelos correios.

O objetivo do presente trabalho é propor um método para a localização do CEP que poderá fazer parte de um sistema de triagem automática de envelopes postais.

1.1. Definição do Problema

Em um sistema para a triagem automática de envelopes postais, uma das etapas consiste na leitura automática do Código de Endereçamento Postal (CEP), pois este representa a parte mais importante do endereço, identificando qual a localidade, logradouro, etc., a correspondência se destina. Mas para a sua leitura automática o CEP deve ser primeiramente localizado no endereço. A falta de uma padronização na distribuição das informações no bloco do endereço, como pode ser observado na Figura 1.1, torna a sua localização uma tarefa desafiadora. Por este motivo faz-se necessário o desenvolvimento de um método que faça uma busca em todo o bloco do endereço, sem levar em conta informações posicionais do CEP.

Figura 1.1 - Exemplos de envelopes postais brasileiros.

Uma estratégia para a solução deste problema pode ser percorrer cada linha de texto que compõe o bloco do endereço em busca de uma seqüência de 5 ou 8 dígitos, separados ou não pelo traço. Seguindo esta estratégia, neste trabalho é proposto um método para a localização do CEP, onde o bloco do endereço é segmentado em linhas e cada linha é analisada em separado e os seus componentes são identificados. Uma vez identificados os componentes das linhas, a localização do CEP consiste na busca de uma seqüência de componentes que represente a estrutura do CEP.

No processo de identificação dos componentes da linha, são utilizados Modelos Escondidos de *Markov* (MEM). O bloco do endereço é composto de uma grande variedade de informações (palavras, dígitos, caracteres, traços, ruídos, etc.) dificultando o processo de modelagem destas informações utilizando MEM. Para amenizar este problema, neste trabalho são avaliadas diferentes estratégias de modelagem para os componentes do bloco do endereço, utilizando o conceito de Meta-Classes, onde as informações são agrupadas segundo algum critério.

Para o desenvolvimento do método foram encontrados os seguintes problemas:

- Por se tratar de documentos manuscritos, um mesmo caractere pode ser escrito de formas diferentes, como pode ser observado na Figura 1.2. Em alguns casos, um mesmo autor pode escrever o mesmo caractere de formas diferentes. Além disso,

no contexto de envelopes postais, existe uma grande variabilidade de tamanho e estilos de escrita.

- O bloco do endereço é composto de dígitos, palavras, caracteres isolados, traços, barras, ruídos, etc., sendo necessário levar em conta toda esta variabilidade.
- Muitos dígitos e caracteres são escritos de forma parecida, podendo facilmente ser confundidos. Alguns exemplos podem ser observados na Figura 1.3.
- Em documentos como formulários e cheques bancários existe um local específico para o preenchimento das informações. No caso de envelopes postais as informações podem ser preenchidas em qualquer posição, podendo estar sobrepostas ou misturadas, dificultando a etapa de segmentação. Mesmo quando existe um local específico para o preenchimento das informações nem sempre este local é utilizado.
- Uma característica da escrita manuscrita é a inclinação das linhas e das palavras. No bloco de endereço as palavras e as linhas podem estar inclinadas ou até mesmo sobrepostas.
- Outra característica da escrita manuscrita é que alguns dígitos podem apresentar-se fragmentados ou conectados interferindo no processo de segmentação. Uma estratégia para a solução deste problema é a utilização de uma estratégia de segmentação implícita.
- A base de dados disponível para treinamento, validação e teste do método é pequena, podendo não ser representativa de todas as variações entre os caracteres.

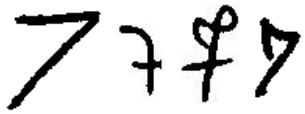

Figura 1.2 - Diferentes formas de escrita do número 7.

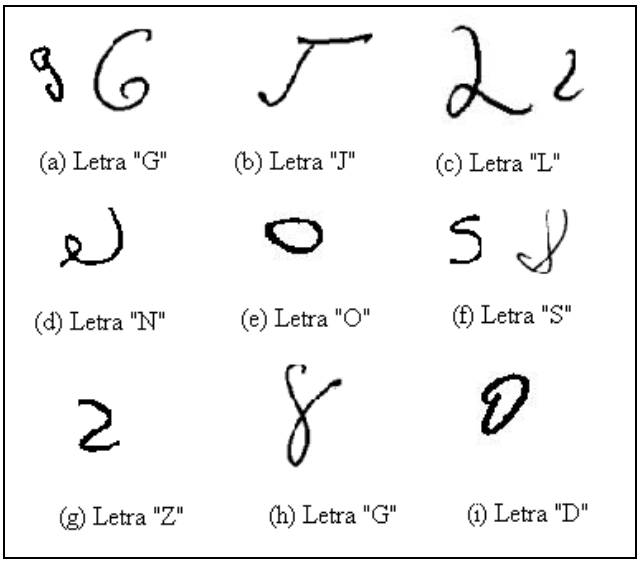

Figura 1.3 - Confusões entre dígitos e caracteres.

1.2. Motivação

Os Correios brasileiros distribuem um volume diário de 34 milhões de objetos e correspondências. No ano de 2001, o total da carga postal foi de mais de 9,5 bilhões e em 2002, de 9,4 bilhões [Correios 2004] . Desta forma, o desenvolvimento de um sistema que faça a triagem automática destas correspondências diminuirá o esforço manual de triagem e o tempo necessário para o seu encaminhamento.

Além disso, até o momento não temos informações da existência de trabalhos brasileiros com este objetivo, sendo motivador o desenvolvimento de um sistema voltado para as características dos envelopes postais brasileiros.

1.3. Objetivos

O objetivo deste trabalho é propor um método para a localização do CEP no bloco do endereço, sem utilizar informações posicionais do CEP, e avaliar diferentes estratégias de modelagem para os componentes do bloco do endereço, utilizando MEM. A Figura 1.4. mostra o diagrama geral do método proposto.

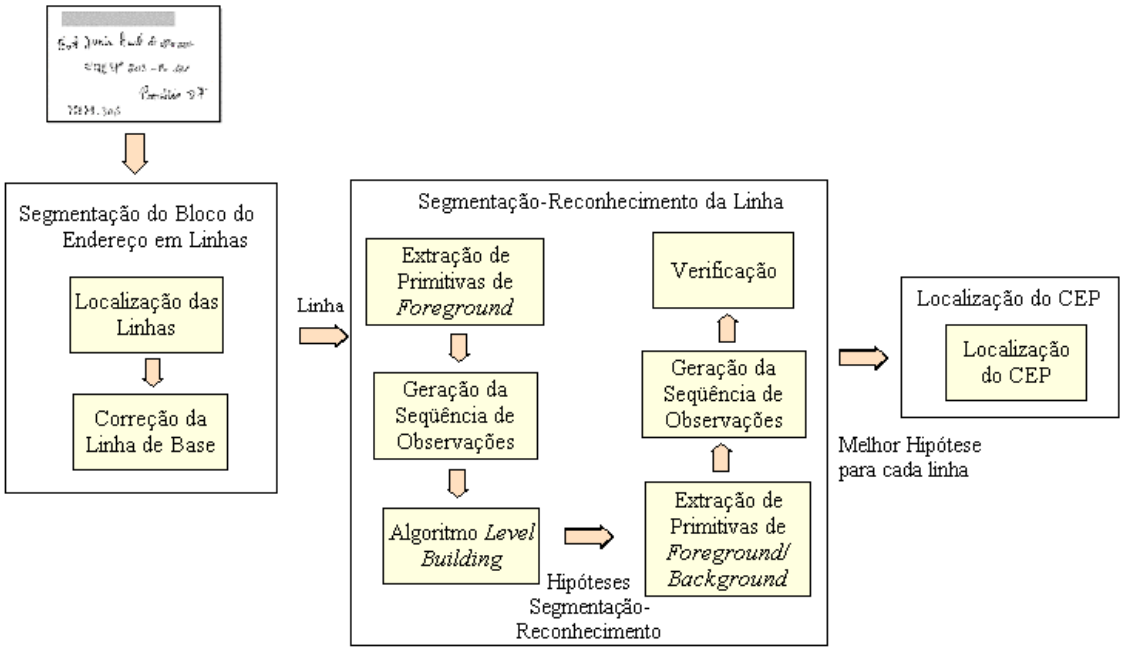

Figura 1.4 - Diagrama geral do método proposto

Primeiramente o bloco do endereço é segmentado em linhas e estas linhas são pré-processadas e enviadas para um classificador baseado no algoritmo *Level Building* e em MEM. Este classificador encontra as melhores hipóteses de segmentação-reconhecimento da linha utilizando uma estratégia de segmentação implícita, onde não existe uma segmentação a priori da linha de texto e a segmentação e o reconhecimento dos componentes da linha são feitos em um único processo. Esta abordagem tem apresentado bons resultados nos casos onde podem ser encontrados dígitos fragmentados, sobrepostos ou conectados, o que ocorre com freqüência no caso de envelopes postais.

No processo de segmentação-reconhecimento, para cada linha são extraídas primitivas locais e globais baseadas nas colunas da imagem. Para que estas primitivas possam ser utilizadas pelos MEMs, cada vetor de primitivas é quantizado em um dos símbolos disponíveis em um *codebook*, gerando uma seqüência de observações. O classificador encontra as melhores hipóteses de segmentação-reconhecimento utilizando a seqüência de observações e modelos MEM treinados com imagens de manuscritos isolados (dígitos, palavras, caracteres, ruídos, etc.). Na etapa seguinte, as hipóteses de segmentação-reconhecimento são verificadas e re-classificadas. Uma vez identificados os pontos de segmentação e os componentes das linhas, a localização do CEP consiste na busca de uma seqüência de elementos que represente a estrutura do CEP.

Para atingir o objetivo proposto foram necessários os seguintes passos:

- Criação de uma base de dados composta de dígitos, palavras, caracteres, etc., extraídos de envelopes postais brasileiros manuscritos.
- Escolha de um método de extração de primitivas.
- Avaliação da configuração de modelos que melhor representa os componentes do bloco do endereço, utilizando o conceito de Meta-Classes.
- Escolha da topologia dos modelos MEM.
- Escolha dos algoritmos para treinamento e teste dos modelos MEM.
- Treinamento dos modelos MEM.
- Teste dos modelos criados.
- Escolha dos algoritmos para segmentação-reconhecimento do CEP.

1.4. Contribuições

As principais contribuições do presente trabalho são:

- Propor um método que poderá ser utilizado na triagem postal, melhorando o processo de triagem postal feito pelos Correios.
- Criação e avaliação dos modelos MEM utilizados na etapa de segmentação-reconhecimento e no estágio de verificação.
- Avaliação do método proposto em [Britto 2001] no reconhecimento de dígitos manuscritos brasileiros e no reconhecimento de palavras e caracteres.
- Avaliação de Meta-Classes na representação das informações que compõem as linhas do bloco do endereço, e para amenizar o problema da base de dados disponível ser pequena.
- Construção de uma base de dados com exemplos de dígitos, palavras, caracteres, etc., extraídos de envelopes postais brasileiros manuscritos.

1.5. Estrutura do Documento

Este documento está dividido em 6 capítulos. O Capítulo 2 apresenta os principais trabalhos relacionados ao método proposto: reconhecimento de dígitos e caracteres manuscritos, localização do CEP em envelopes postais e Meta-Classes. Primeiramente são apresentados alguns métodos de extração de primitivas utilizados no reconhecimento de dígitos e caracteres manuscritos, destacando suas vantagens e desvantagens e listando alguns trabalhos onde estas primitivas foram utilizadas. Alguns métodos propostos para a localização e reconhecimento do CEP em envelopes postais são apresentados, bem como os resultados obtidos. Finalmente é apresentado um trabalho onde foram utilizadas Meta-Classes.

O Capítulo 3 descreve os principais tópicos necessários para o entendimento do método proposto. Primeiramente são apresentados os Modelos Escondidos de *Markov* (MEM), os quais são utilizados para modelar as informações que compõem o bloco do endereço. São descritos os principais conceitos sobre MEM, os parâmetros de um MEM discreto de primeira ordem, os tipos de MEM e os algoritmos necessários para treinamento e reconhecimento dos modelos. O algoritmo *Level Building* utilizado para segmentação-reconhecimento da linha é descrito e um algoritmo para o cálculo do melhor número de estados para os modelos também é apresentado. O segundo tópico descreve o processo de quantização vetorial, o qual é utilizado para o mapeamento do vetor de primitivas extraídas da imagem em observações discretas.

No capítulo 4, o método proposto é descrito detalhadamente. Primeiramente é dada uma visão geral do método e em seguida são descritas todas as etapas que o compõem: segmentação do bloco do endereço em linhas, segmentação-reconhecimento das linhas e localização do CEP. O processo de criação dos modelos MEM é descrito em detalhes.

O Capítulo 5 descreve a base de dados utilizada para o treinamento, validação e teste do método proposto, bem como os testes realizados durante a criação dos modelos MEM utilizados nos processos de segmentação-reconhecimento da linha e verificação dos resultados obtidos. O primeiro experimento avalia a eficiência dos modelos na distinção entre os dígitos de 0 a 9 e faz uma comparativo com os resultados obtidos em [Britto 2003]. Nos experimentos seguintes são avaliadas diversas configurações de Meta-Classes, com o objetivo de identificar qual a melhor configuração representa as informações que compõem o bloco do

endereço. Nos últimos experimentos, são avaliados os modelos MEM criados para o estágio de verificação. A última seção apresenta uma discussão dos resultados obtidos.

O Capítulo 6 apresenta as conclusões e trabalhos futuros.

2. Estado da Arte

Neste capítulo são apresentados os principais trabalhos relacionados ao método proposto: reconhecimento de dígitos e caracteres manuscritos, localização do CEP em envelopes postais e Meta-Classes. Na Seção 2.1 são apresentados alguns métodos de extração de primitivas utilizados para o reconhecimento de dígitos e caracteres manuscritos, destacando as suas vantagens e desvantagens. Alguns trabalhos onde estes métodos foram utilizados são listados e os resultados obtidos são descritos. Na Seção 2.2, alguns métodos propostos na literatura para a localização e reconhecimento do CEP em envelopes postais são apresentados, bem como suas limitações e os resultados obtidos. Um trabalho onde foram utilizadas Meta-Classes é apresentado na Seção 2.3. e a Seção 2.4 apresenta as considerações finais referentes a este capítulo.

2.1. Extração de Primitivas

A investigação de métodos para extração de primitivas é um tópico importante de pesquisa, uma vez que um conjunto de primitivas discriminante é um fator importante para a obtenção de altas taxas de reconhecimento.

Geralmente os métodos de extração de primitivas estão divididos em estatísticos e estruturais [Britto 2001]. Métodos estatísticos são derivados da distribuição estatística dos pontos, utilizando a manipulação de medidas, tais como variáveis numéricas e lógicas. Os métodos estruturais tratam da manipulação de informações baseadas na percepção humana dos objetos sendo baseados em propriedades topológicas e geográficas do caractere ou dígito.

Um trabalho completo sobre métodos de extração de primitivas pode ser encontrado em [Trier, et al. 1996]. Segundo este autor as primitivas podem possuir as seguintes propriedades:

- **Invariante:** primitivas são consideradas invariantes quando tem aproximadamente os mesmos valores para exemplos do mesmo caractere, não sofrendo influência de parâmetros como: translação, escala, rotação e espelhamento. Ex.: primitivas invariantes à inclinação podem ser utilizadas para textos manuscritos, onde os caracteres podem estar mais ou menos inclinados, e textos impressos, onde algumas fontes são inclinadas e outras não.

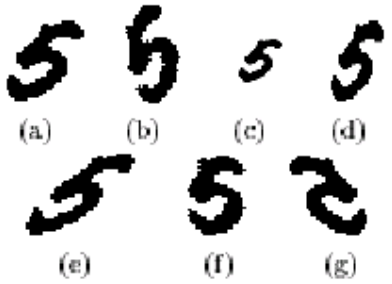

Figura 2.1 - Versões transformadas do dígito '5 '
(a) original, (b) rotação, (c) escala, (d) estiramento, (e) inclinação, (f) correção de inclinação, (g) espelhamento [Trier, et al. 1996].

- **Reconstrução:** Por alguns métodos de extração de primitivas, os caracteres podem ser reconstruídos a partir de uma primitiva extraída. Esta propriedade garante que a informação completa sobre a forma do caractere está presente nas primitivas extraídas. Pela reconstrução do caractere a partir das primitivas extraídas pode-se confirmar se o número suficiente de primitivas está sendo usado para capturar a estrutura essencial do caractere e também controlar se a implementação está correta.

A seguir serão descritos alguns métodos de extração de primitivas utilizados para o reconhecimento de dígitos e caracteres manuscritos, destacando as suas vantagens e desvantagens.

2.1.1. Modelos Deformáveis

Modelos deformáveis representam uma evolução do método de *Template Matching*.

Neste último, um modelo (*template*) para cada classe que se deseja reconhecer deve ser armazenado. Toda a imagem é usada como um vetor de primitivas e na etapa de

reconhecimento o caractere é comparado com os *templates* armazenados, através de alguma medida de similaridade ou dissimilaridade. O *template* com a mais alta medida de similaridade é identificado, e se esta similaridade está acima de um determinado limiar, o caractere é considerado pertencente à classe do *template*. Para se utilizar o *Template Matching* para o reconhecimento de caracteres manuscritos, um grande número de *templates* é necessário para a representação de todas as possíveis deformações do caractere. Uma alternativa é ter a possibilidade de deformar estes modelos.

A partir do modelo de uma classe, são utilizadas funções, deformando este modelo, tendo assim múltiplos modelos para cada classe, melhorando os resultados de reconhecimento. Uma desvantagem desta técnica é a grande complexidade computacional [Cheung, et al 1997] . Um exemplo de modelo deformável pode ser observado na Figura 2.2.

Figura 2.2 - Ilustração da extração de modelos deformáveis. O caractere pequeno próximo ao canto superior esquerdo de cada figura é o modelo [Cheung, et al 1997] .

Em [Cheung, et al 1997] foram utilizados modelos deformáveis para o reconhecimento de dígitos manuscritos da base especial NIST1. A Tabela 2.1 mostra os resultados obtidos considerando-se diferentes taxas de rejeição. Foram utilizados 11660 e 11791 dígitos para treinamento e teste respectivamente.

Tabela 2.1 - Resultados obtidos em [Cheung, et al 1997] utilizando Modelos Deformáveis

Confiabilidade	92.90%	93.01%	94.18%	95.50%	96.20%	96,44%
Taxa de Reconhecimento	92.90%	92.83%	89.82%	86.10%	80.30%	78.10%
Taxa de Rejeição	0%	0.20%	4.63%	4.63%	16.58%	19.02%
Taxa de Substituição	7.08%	6.98%	5.55%	5.55%	3.14%	2.88%

2.1.2. *Zoning*

No método *zoning*, um grid N x M é sobreposto sobre a imagem do caractere, dividindo esta imagem em zonas, e para cada uma dessas zonas são extraídas primitivas (um exemplo pode ser observado na Figura 2.3). Este método é geralmente utilizado para extrair informações topológicas do padrão, permitindo a análise de características locais. Em alguns casos a imagem é dividida em zonas de igual tamanho e em outros casos a imagem é dividida de acordo com a densidade do padrão. Em [Leche, et al. 2000] é apresentada uma nova técnica para *zoning*, determinando a distribuição estatística de características locais usando um conjunto de treinamento de padrões e utilizando entropia para determinar zonas "centrais" mostrando alta capacidade de discriminação. A Tabela 2.2 mostra os resultados obtidos no reconhecimento de dígitos manuscritos, extraídos da base de dados do CEDAR (*Center of Excellence in Document Analysis Recognition*).

Tabela 2.2 - Resultados obtidos em [Leche, et al. 2000]

		Taxa de Reconhecimento	
Classe	Número de exemplos para teste	Zoning (4X4) Tradicional	Nova Técnica
0	355	68%	77%
1	288	85%	93%
2	220	88%	93%
3	206	84%	96%
4	179	77%	88%
5	116	56%	83%
6	243	65%	87%
7	217	63%	76%
8	189	60%	75%
9	176	65%	76%

Alguns exemplos de primitivas extraídas pelo método *zoning* são a média dos tons de cinza em imagens em nível de cinza, o percentual de *pixels* pretos em imagens binárias e código de cadeia, o qual será descrito posteriormente.

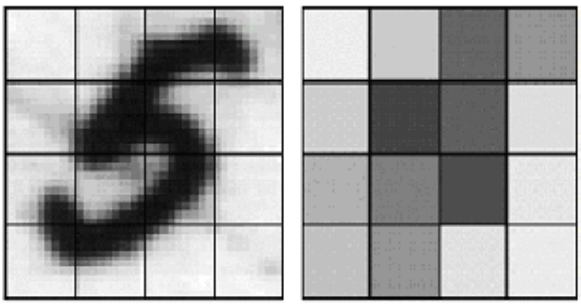

Figura 2.3 - Zoning de uma imagem em nível de cinza [Trier, et al. 1996].

2.1.3. Características Extraídas do Contorno Externo do Caractere

Primitivas extraídas do contorno externo dos caracteres são muito utilizadas e se baseiam na observação de que os caracteres podem ser distinguidos pelo seu contorno externo. Uma desvantagem deste tipo de primitiva é que o contorno externo de algumas letras como B e D podem ser muito similares, e conseqüentemente, são sensíveis a ruídos. Este problema pode ser parcialmente resolvido pelo uso do contorno interno ou outras características adicionais.

Características geralmente extraídas do contorno:

Perfil do Contorno

A medida do contorno esquerdo é a distância da borda esquerda da imagem até o contorno esquerdo do caractere em cada coluna. O contorno direito é formado correspondentemente. Isto resulta em dois vetores de valores inteiros, os quais tem a dimensionalidade igual à altura da imagem. Os perfis podem ser usados como vetores de primitivas em classificação estatística ou processados para formar primitivas heurísticas. Uma vantagem do perfil do contorno é sua imunidade a pequenas falhas (*gaps*) na imagem do caractere. Contudo, este é extremamente sensível à rotação e inclinação [Laaksonen 1997] . Na Figura 2.4 pode ser observado que é possível confundir o perfil do contorno do número 4 com o número 9.

Figura 2.4 - Perfil do contorno esquerdo e direito do número ‘4’ [Laaksonen 1997] .

Código de Cadeia

Códigos de cadeia são formados pela vetorização do contorno fechado do caractere. A cadeia externa é transformada em uma string na qual cada elemento indica a direção do contorno de um ponto particular. O número de níveis de quantização usados na representação das direções pode variar. Mais freqüentemente as 4 direções básicas que resultam de um modelo de 8 vizinhos conectados é suficiente. Um histograma direcional pode ser gerado da imagem inteira ou de alguma parte dela. Este método é freqüentemente combinado com o método *zoning* descrito anteriormente. A Figura 2.5 mostra as direções normalmente utilizadas para o cálculo do código de cadeia. Este método é amplamente utilizado no reconhecimento de caracteres manuscritos como em [Yamauchi, et al. 1997] e
[Hirano, et al. 1997].

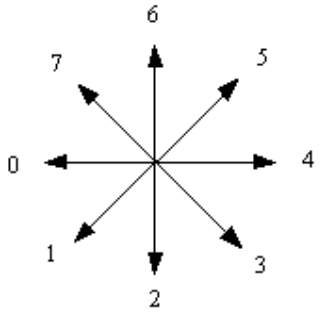

Figura 2.5 - Direções normalmente utilizadas para o código de cadeia.

Descritores de Fourier

Através da transformada de Fourier, pode-se decompor um sinal em seus componentes de freqüência (senos e co-senos), de forma que um coeficiente de Fourier reflete a importância de determinada freqüência para o sinal. Em sinais discretos, pode ser feita a redução de dimensionalidade (extração de características) através dos descritores de Fourier.

Descritores de Fourier têm sido uma poderosa ferramenta para o reconhecimento de padrões e são usados para caracterizar o limite externo de um caractere. Os coeficientes de Fourier capazes de representar a forma de um dígito manuscrito podem ser usados como primitivas estatísticas. Os primeiros coeficientes descrevem a maior parte da forma rústica do caractere. A quantidade de detalhes é aumentada quando mais coeficientes são incluídos.

Em [Trier, et al. 1997] foram utilizados descritores Elípticos de Fourier extraídos do contorno externo do caractere, para o reconhecimento de dígitos em mapas. Como os dígitos variavam em tamanho e rotação, foram escolhidos os Descritores de Fourier por não serem afetados por estes problemas. A Figura 2.6 mostra um exemplo do número 4 reconstruído por descritores elípticos de Fourier.

Figura 2.6 - Dígito '4' reconstruído por descritores elípticos de Fourier [Trier, et al. 1996].

2.1.4. Projeção Poligonal

Em [Silva 2002] é descrito um método de extração de primitivas baseado em um polígono posicionado em torno da imagem do caractere, destinado ao reconhecimento de caracteres manuscritos (dígitos e letras), provenientes de formulários de concursos. O método proposto por Silva, atingiu uma taxa de 94,57% de acerto no reconhecimento de dígitos, no conjunto de teste, utilizando uma rede neural, e uma taxa de 91,91% no reconhecimento de letras sobre o conjunto de teste, utilizando um time de redes neurais. Para treinamento, validação e teste com dígitos foi utilizada uma base com 4000, 500 e 3720 exemplares respectivamente, e 10400, 1300 e 8843 para o caso das letras.

Os vetores de primitivas são baseados no cálculo de um conjunto de distâncias tiradas a partir do contorno da imagem até um polígono de referência. O polígono deve ser regular, podendo ter qualquer número de lados e deverá ser posicionado em torno da imagem-alvo como ilustrado na Figura 2.7.

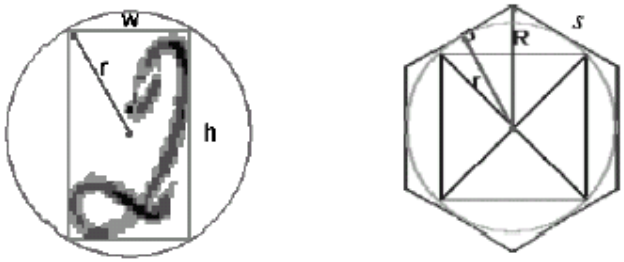

Figura 2.7 - Círculo circunscrito à imagem e o polígono circunscrito ao círculo [Silva 2002] .

O processo básico toma a distância de cada lado do polígono até o contorno da imagem e armazena esta distância em um vetor que contém também o número de lados e o número de pontos (tamanho do lado) para cada lado. Este método tem um bom poder de discriminação e pode resolver problemas relacionados com escala, rotação e translação da imagem original. Desde que a imagem do caractere esteja inicialmente centralizada dentro de um retângulo antes de ser circundada pelo polígono desejado, o problema de translação é automaticamente eliminado. A aplicação de um esquema de interpolação na tomada dos pontos para o cálculo da distância ao contorno resolve o problema de escala. O problema da rotação pode ser resolvido aplicando-se uma rotação circular no vetor resultante para a direita ou para a esquerda, conforme a inclinação do caractere [Silva 2002] .

Segundo o autor, os polígonos com números pares de lados, mais especificamente o quadrado e o hexágono, obtiveram melhores resultados nos experimentos realizados, sendo o quadrado o mais eficiente. A Figura 2.8 mostra uma seqüência de projeções para o quadrado e para o hexágono.

O autor extraiu também características do caractere com o quadrado rotacionado em 45º em relação à sua posição original, o que possibilitou considerar detalhes do caractere que não eram possíveis de ser extraídos com o método convencional do quadrado. Para evitar falhas quando a diferenciação entre os caracteres se faz principalmente na região interna dos mesmos, foram extraídas também características da região interna dos caracteres. As medidas que compõem o vetor de características internas partem do centro do caractere e terminam no contorno interno do mesmo. Este método é bastante útil para caracteres que possuem orifícios em sua forma como o 8 (oito) e a letra O. A Figura 2.9 mostra uma seqüência de projeções para o quadrado rotacionado, a Figura 2.10 ilustra as primitivas internas extraídas.

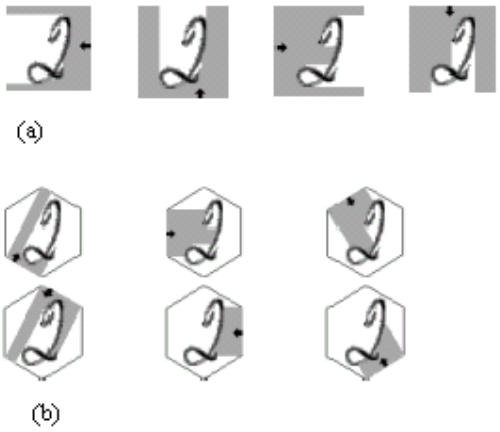

Figura 2.8 - (a) Seqüência de procedimentos de projeções para o quadrado. (b) seqüência de procedimentos de projeções para o hexágono [Silva 2002] .

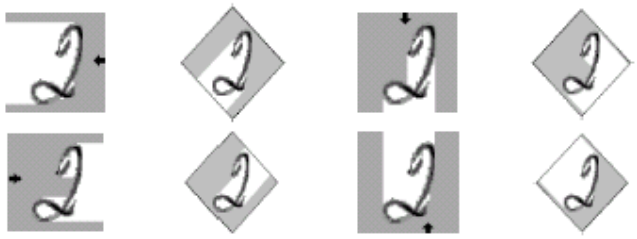

Figura 2.9 - Seqüência de projeções para o quadrado rotacionado [Silva 2002] .

Figura 2.10 - Extração de primitivas internas [Silva 2002] .

2.1.5. Características Topológicas

São consideradas topológicas primitivas como cavidades e *loops*. Em [Lee & Gomes, 1997] foram utilizadas características topológicas para o reconhecimento de dígitos manuscritos. As características extraídas foram:

1) O número de cavidades centrais, o número de cavidades do lado esquerdo, o número de cavidades do lado direito, a localização de cada cavidade central (esquerda, direita,

topo, base). Estas características foram obtidas pela contagem e identificação da localização das cavidades que aparecem em cada imagem do dígito.

2) As seqüências de intersecção dos numerais: duas seqüências de intersecção foram definidas: vertical e horizontal. Cada elemento da seqüência de intersecção refere-se ao número de blocos de *pixels* pretos consecutivos que aparecem na linha horizontal ou vertical correspondente.

3) Intersecção com os eixos principal e secundário: O eixo principal da imagem do numeral é definido como um segmento de linha vertical desenhado através do centro de gravidade da imagem. O eixo secundário é uma linha vertical eqüidistante do eixo principal e a borda esquerda da imagem.

Estas características foram extraídas do esqueleto da imagem do caractere [Trier, et al. 1996]. Uma desvantagem na utilização de características topológicas é que elas nem sempre podem ser extraídas devido às distorções e ruídos introduzidos durante o processo de digitalização ou causadas pelos instrumentos de escrita.

Como o método descrito acima leva em consideração a posição das primitivas encontradas na imagem (esquerda, direita), ele sofre interferência quando o caractere está inclinado ou rotacionado.

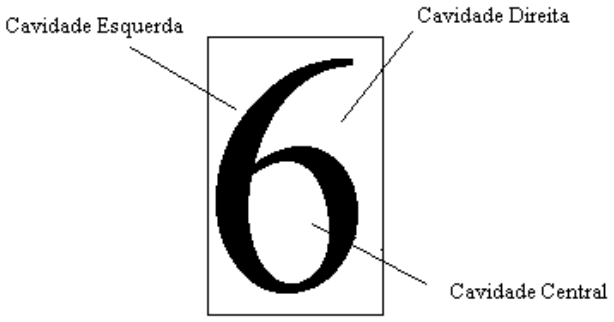

Figura 2.11 - Extração de cavidades (Adaptada de [Lee & Gomes, 1997])

A Tabela 2.3 mostra um comparativo entre os métodos descritos anteriormente, quanto à sensibilidade a rotação, inclinação e escala.

Tabela 2.3 - Comparativo entre os métodos, quanto a sensibilidade à rotação, inclinação e escala.

Primitiva	Rotação	Inclinação	Escala	Observações
Template Matching	variante	Variante	variante	Sensível a variações intraclasse
Modelos Deformáveis	variante	Variante	variante	Grande complexidade computacional
Zoning	variante	Variante	variante	
Perfil do Contorno	variante	variante	variante	Sensível a ruído
Código de Cadeia	variante	variante	variante	Sensível a ruído
Descritores de Fourier	invariante	invariante	invariante	
Projeção Poligonal	invariante	invariante	invariante	
Características Topológicas	variante	variante	variante	Sensível a distorções e ruídos.

2.2. Métodos para localização e reconhecimento do CEP

A seguir serão descritos alguns métodos propostos na literatura para a localização e reconhecimento do CEP em envelopes postais.

Em [Whichelo & Yan 1996] é apresentado um método para a localização automática do bloco do endereço e do código postal, em imagens de envelopes postais australianos, manuscritos e impressos. O método consiste em dividir a imagem em regiões com probabilidade de conter o endereço e examiná-las utilizando um processo de OCR. Primeiramente são identificadas as regiões "ocupadas" do envelope, através da localização das partes da imagem com alta freqüência, utilizando para isto um filtro passa alta [Marques & Vieira 1999]. Este processo é seguido por um processo de detecção de bordas formando regiões conectadas. As regiões candidatas a pertencer ao endereço são selecionadas pela detecção das *bounding boxes*. Para eliminar possíveis ruídos ou regiões improváveis de pertencer ao endereço, algumas regiões são descartadas de acordo com alguns critérios como, por exemplo, regiões cuja área esteja abaixo de um limiar estimado experimentalmente. Este processo retorna entre 10 a 30 regiões candidatas a pertencer ao bloco do endereço. Para facilitar a busca do código postal, evitando que todas as regiões retornadas sejam analisadas, as regiões são ordenadas de acordo com a sua posição. Além disso, estas regiões estão grosseiramente agrupadas juntas na face do envelope (como o endereço do destinatário, endereço de retorno) e por este motivo o seu agrupamento facilita o processo de ordenação.

Através da implementação do algoritmo *C-means* os blocos candidatos restantes neste ponto são agrupadas em 3 grupos (um exemplo pode ser observado na Figura 2.12). Os *clusters* resultantes do processo de agrupamento são examinados por um processo de OCR para determinação do código postal. O *cluster* que é encontrado a direita do centro nos 2/3 mais abaixo do envelope é determinado como a primeira escolha para a busca do CEP, pois o bloco do endereço é mais provavelmente encontrado nesta região.

Figura 2.12 - Blocos candidatos extraídos da imagem de teste, após a aplicação do algoritmo C-Means [Whichelo & Yan 1996]

Foram realizados testes com 72 imagens de envelopes e em 86% dos casos o primeiro *cluster* retornado pelo método continha o CEP, e em apenas um caso todos os *clusters* tiveram que ser analisados. Os autores não fornecem informações de como é feita análise das regiões pelo processo de OCR.

Em [Kimura, et al. 1995] é descrito um método para o reconhecimento do CEP usando um algoritmo para reconhecimento de palavras, onde uma string numérica é reconhecida como uma palavra. No método descrito, o bloco do endereço é pré-processado e dividido em linhas. Em seguida, os componentes das linhas são segmentados em palavras. Palavras são assumidas como sendo separadas por um espaço, uma vírgula ou um período. Um algoritmo de detecção de espaços detecta os espaços através da classificação dos intervalos entre os segmentos do caractere dentro de duas classes: intervalos “entre campos” e intervalos “intra campos” respectivamente. Se o intervalo é mais largo do que um limiar, o intervalo é assinalado como intervalo “entre campos”, caso contrário, como intervalo “intra campos”. O intervalo “entre campos” corresponde ao espaço, por conseguinte especifica o limite entre dois campos adjacentes.

Para a localização do CEP, é assumido primeiramente que o CEP é o último campo da última linha. Se a probabilidade do CEP detectado é menor que um limiar, até duas linhas precedentes são assumidas sucessivamente como sendo a linha do CEP, até que o CEP com probabilidade suficiente seja detectado. A taxa de reconhecimento do CEP foi de 82,20%. Os autores não fornecem os resultados obtidos na localização do CEP.

Em [Dzuba, et al. 1999] é descrito um método para a segmentação do bloco contendo o estado, a cidade e o CEP (ECC). Primeiro o bloco do endereço é pré-processado e dividido em linhas. Estatísticas mostraram que o bloco contendo o ECC ocupa as duas últimas linhas do bloco do endereço em cerca de 97% dos casos. Desta forma, o maior problema é decidir se o bloco com o ECC está localizado na última linha ou nas duas últimas linhas. O comprimento das linhas e a sua localização são levadas em conta para decidir se o bloco contendo o ECC consiste de uma, duas ou três linhas. O processo de segmentação do ECC gera uma lista de cidade, estado e CEP candidatos para posterior reconhecimento. A saída do processo é um conjunto de "variações" de segmentação, e cada uma destas "variações" contém 3 caixas: cidade, estado e CEP respectivamente. O processo de segmentação tem os seguintes passos:

- Primeiro, uma nova imagem é criada "borrando" a imagem de entrada em direções horizontais. Com este procedimento, freqüentemente cada palavra é representada por um único componente conexo (um *spot*);
- Se o *spot* contém potenciais vírgulas e traços ele é dividido em partes;
- As seguintes características de cada *spot* são estimadas: tamanho, localização na string, distância entre vizinhos na mesma string, complexidade vertical e horizontal da imagem original na área do *spot*;
- Os *spots* são classificados em: palavras, grupos de letras, letras separadas, vírgulas, traços ou ruídos;
- Hipóteses para a localização do CEP são geradas começando com a hipótese de que o CEP está escrito no final da última string;
- Para cada hipótese do CEP os *spots* remanescentes são interpretados como cidade, estado, os dígitos adicionais do CEP (é esperado que o CEP contenha 5 dígitos, em alguns casos o CEP pode ter 4 dígitos adicionais) ou ruídos;
- Pontuações para cada hipótese são calculadas;

O sistema gera até 4 “variações” de segmentação dependendo da complexidade da imagem, e produz em média 2.4 hipóteses. A hipótese correta está entre os candidatos em 91% dos casos. As hipóteses com mais alta pontuação são processadas primeiro. Se o valor de confiança do reconhecimento do CEP para alguma hipótese é alto o suficiente o processo é terminado, caso contrário a próxima hipótese de segmentação é investigada. Um exemplo com as hipóteses de segmentação pode ser observado na Figura 2.13. A Tabela 2.4 mostra a performance do método para vários valores de confiança. Taxa de reconhecimento é o percentual de respostas aceitas do CEP. A taxa de erro é o percentual de erros entre as respostas aceitas do CEP. Estima-se que 40% dos erros no reconhecimento é devido à segmentação errada dos campos de interesse (cidade, estado e CEP).

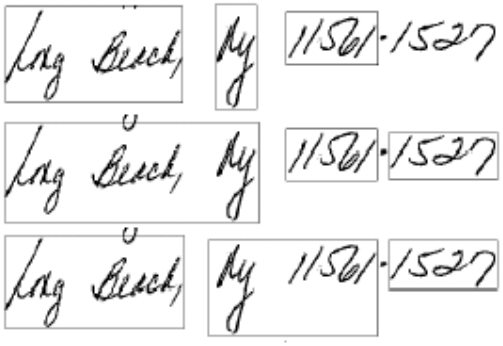

Figura 2.13 - Hipóteses de segmentação [Dzuba, et al. 1999].

Tabela 2.4 - Performance do método proposto em [Dzuba, et al. 1999] com vários valores de confiança.

Bloco do endereço		Strings da base NIST	
Taxa de Reconhecimento	Taxa de Erro	Taxa de Reconhecimento	Taxa de Erro
100.00	18.3	100.00	5.3
77.8	2.0	93.3	2.0
73.1	1.0	91.4	1.0
66.7	0.5	90.7	0.5

Em [Matan, et al. 1992] é descrito um algoritmo baseado em redes neurais para a leitura de ***strings*** e dígitos manuscritos, o qual faz parte de um sistema para a leitura do CEP em envelopes postais. O sistema foi treinado e testado com aproximadamente 10.000 imagens de CEP, com 5 ou 9 dígitos, extraídos de envelopes reais. A localização dos dígitos do CEP foi feita manualmente. O sistema é composto dos seguintes estágios:

- Pré-Processamento: consiste na remoção de ruídos e correção da inclinação dos dígitos.
- Segmentação e reconhecimento dos Componentes Conexos (CC): cada CC é avaliado para determinar se este é pequeno ou grande o bastante para ser um dígito. Se for muito pequeno ele é descartado. Se for muito grande este provavelmente consiste de dois ou mais dígitos conectados e deve ser enviado para o próximo nível de segmentação. Os CCs que passam nesta avaliação são avaliados pelo reconhecedor. Se o resultado é maior do que um determinado limiar, o CC é considerado como resolvido. Segmentos com resultados abaixo do limiar são enviados ao próximo estágio. Quando todos os CCs obtêm uma taxa de reconhecimento acima do limiar e o número de segmentos é 5 ou 9, a segmentação é concluída.
- Estimação dos pontos de corte vertical e segmentação: partes da imagem que ainda não foram consideradas resolvidas são segmentadas utilizando cortes verticais. Os CCs considerados resolvidos na etapa anterior são apagados da imagem. A estimação dos pontos de corte é baseada na projeção vertical dos *pixels* da imagem remanescente. Um ponto de corte candidato com um valor de projeção maior que um certo limiar é considerado como um ponto óbvio de segmentação. É necessário encontrar um certo número de cortes, determinado pelo número de dígitos que podem ser encontrados na imagem menos o número de CCs já encontrados. Se o número de cortes óbvios é igual ao número de cortes necessários, então o conjunto de cortes óbvios é a solução. Se mais cortes são necessários, o sistema usa o reconhecedor para encontrá-los. Este reconhecedor é baseado em redes neurais. Cada segmento é avaliado pelo classificador. O segmento que resultar na melhor combinação de resultados é escolhido.
- Diretório de *Lookup*: as respostas do sistema que não sejam encontradas no diretório de CEPs são rejeitadas e o processo de estimação dos pontos de corte vertical e segmentação é chamado novamente para gerar o próximo segmento candidato.

Os resultados obtidos podem ser observados na Tabela 2.5.

Tabela 2.5 - Resultados obtidos em [Matan, et al. 1992]

Versão do Sistema	Erro	Erro com 40% de Rejeição
Todos os dígitos do CEP	26,15	5,47
Os primeiros 5 dígitos	24,53	4,64
Os primeiros 5 dígitos + Diretório de avaliação do CEP	23,83	3,22

Como visto nos métodos descritos, o CEP tem grande probabilidade de ser o último campo da última linha do bloco do endereço, ou de estar em uma das 2 últimas linhas do endereço. Na análise dos exemplos de envelopes brasileiros verificou-se que isso nem sempre acontece e o CEP pode ser encontrado em qualquer posição. Por este motivo, faz-se necessário o desenvolvimento de um método que não leve em conta informações posicionais do CEP e faça uma busca em todo o bloco do endereço.

2.3. Meta-Classes

Em [Morita, et al. 2002] é apresentado um sistema híbrido MEM-MLP (*MultiLayer Perceptrons*) para reconhecimento de imagens de datas complexas, escritas em cheques bancários brasileiros. O sistema primeiro segmenta implicitamente a imagem da data em sub-campos através de um processo baseado em MEM. Em seguida, os três sub-campos obrigatórios da data são processados pelo sistema (dia, mês, ano). A abordagem neural é utilizada para o reconhecimento dos dígitos e a abordagem MEM é utilizada para o reconhecimento e verificação de palavras. Foram utilizadas Meta-Classes de dígitos para reduzir o tamanho do léxico dos campos dia e ano e para melhorar a precisão do processo de segmentação e reconhecimento.

Meta-Classes podem ser utilizadas quando o objetivo não é o reconhecimento e sim a localização dos dígitos em um documento, evitando certas confusões entre os dígitos (como os dígitos 4 e 9 e os dígitos 8 e 0, por exemplo) e aumentando a base de treinamento. Esta é uma das hipóteses a serem testadas nesta dissertação. Morita, com o uso deste conceito no reconhecimento de dígitos, melhorou a taxa de reconhecimento de 97,1% para 99,2%, usando um subconjunto da série hsf_7 da base NIST SD19 [Grother, 1995], a qual contém 986 imagens de strings com dois dígitos relativos ao léxico dos dois dígitos do dia.

A autora definiu 4 Meta-Classes de dígitos ($C_{0,1.2,3}$, $C_{1,2}$, $C_{0,9}$ e $C_{0,1,2,9}$) baseados nas classes de dígitos presentes em cada posição dos dígitos do dia (que pode ser composto de 1 ou dois dígitos) e dos dígitos do ano (que pode ser composto de 2 ou 4 dígitos), Figura 2.14. Isto é possível porque o léxico do dia e do ano é conhecido e limitado. Enquanto a classe de dígitos C_{0-9} lida com 10 classes de numerais, as Meta-Classes de dígitos trabalham com classes específicas de dígitos.

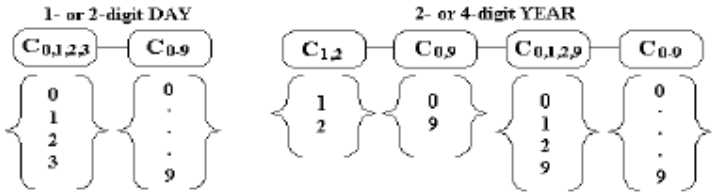

Figura 2.14 - Classes de dígitos presentes em cada posição dos 1 ou dois dígitos do dia e 2 ou 4 dígitos do ano [Morita, et al. 2002] .

2.4. Considerações Finais

Neste capítulo foram apresentados os principais trabalhos relacionados ao método proposto: reconhecimento de dígitos e caracteres manuscritos, localização do CEP em envelopes postais e Meta-Classes. Alguns métodos de extração de primitivas utilizados para o reconhecimento de dígitos e caracteres manuscritos foram apresentados, bem como suas vantagens e desvantagens.

Alguns métodos propostos na literatura para a localização do CEP em envelopes postais também foram apresentados. Como visto nestes métodos, o CEP tem grande probabilidade de ser o último campo da última linha do bloco do endereço, ou de estar em uma das 2 últimas linhas do endereço. Na análise dos exemplos de envelopes brasileiros verificou-se que isso nem sempre acontece e o CEP pode ser encontrado em qualquer posição. Por este motivo, faz-se necessário o desenvolvimento de um método que não leve em conta informações posicionais do CEP e faça uma busca em todo o bloco do endereço, o qual é objetivo do método proposto.

Finalmente, foi descrito um trabalho baseado em Meta-Classes, as quais são utilizadas no nosso método para diminuir a quantidade de classes a serem investigadas e aumentar o

número de exemplos para treinamento, validação e teste, uma vez que a base de dados disponível é pequena.

O próximo capítulo descreve os principais conceitos necessários para a descrição do método proposto: Modelos Escondidos de *Markov*, *Level Building* e Quantização de Vetores.

3. Fundamentação Teórica

Este capítulo descreve os principais tópicos necessários para a descrição do método proposto. A Seção 3.1 apresenta os Modelos Escondidos de *Markov* (MEM), os quais são utilizados para modelar as informações que compõem o bloco do endereço: dígitos, palavras, caracteres, etc. São descritos os parâmetros de um MEM discreto de primeira ordem, os tipos de MEM e os algoritmos necessários para o treinamento e reconhecimento dos modelos. A distribuição da probabilidade de observação para cada estado: discreta, contínua e semi-contínua e um algoritmo para o cálculo do melhor número de estados para os modelos também são apresentados. Na Seção 3.2 é descrito um processo de quantização vetorial, o qual é utilizado para o mapeamento do vetor de primitivas extraídas da imagem em observações discretas. A Seção 3.3 apresenta as considerações finais referentes a este capítulo.

3.1. Modelos Escondidos de Markov (MEM)

Os processos no mundo real geralmente produzem sinais (seqüência de observações). Os sinais podem ser discretos por natureza (caracteres de um alfabeto finito, vetores quantizados de um alfabeto ou *codebook*) ou contínuos por natureza (exemplos de vozes, medidas de temperatura, música, etc). A fonte do sinal pode ser estacionária (suas propriedades estatísticas não variam com o tempo) ou não estacionária (propriedades estatísticas variam ao longo do tempo). Além disso, os sinais podem ser puros (vem somente de uma fonte restrita) ou não puros (ruído, distorções de transmissão, outras fontes de sinais) [Rabiner 1989] .

Os sinais podem ser modelados utilizando-se as classes deterministas ou estatísticas. Os modelos deterministas exploram geralmente algumas propriedades específicas do sinal, sendo necessário determinar (estimar) valores dos parâmetros do modelo do sinal (amplitude, freqüência, etc.). Os modelos estatísticos tentam caracterizar somente propriedades estatísticas dos sinais (processos de *Gauss*, *Poison*, *Markov*, Modelos Escondidos de *Markov*, entre outros) [Oliveira & Morita 1998].

Os Modelos Escondidos de Markov (MEM) são modelos estatísticos e têm sido utilizados com sucesso no reconhecimento da fala [Rabiner 1989] , no reconhecimento de strings numéricas manuscritas [Britto, et al. 2001b] , no reconhecimento de palavras manuscritas [Morita, et al. 2002] , [Freitas 2001] , [Yacoubi, et al. 1999] e no reconhecimento de texto impresso [Aas, et al. 1996].

Uma Cadeia de *Markov* é um método para modelagem de um sinal, através de uma seqüência de observações de saída produzidas por algum processo chamado de fonte de *Markov*. Os símbolos gerados a partir dessa fonte são dependentes apenas de observações anteriores, as quais foram geradas da mesma forma e assim sucessivamente. O número de seqüências anteriores consideradas para o próximo símbolo a ser produzido define a ordem da Cadeia de *Markov*. Cadeias de *Markov* de primeira e segunda ordem são mais utilizadas, uma vez que a complexidade do modelo cresce exponencialmente com sua ordem.

Um MEM tem a mesma estrutura de uma Cadeia de *Markov*. A diferença é que cada estado de uma Cadeia de *Markov* representa uma observação única de um símbolo correspondendo a uma observação física de um evento, enquanto nos MEMs cada estado é uma probabilidade sobre todos os símbolos. Desta forma, em uma Cadeia de *Markov*, dada uma seqüência de símbolos produzidos por um modelo, é possível calcular a seqüência de estados que a produziu. Porém, na maioria dos problemas reais, cada estado de um modelo não pode ser definido como um único símbolo, pois mais de um símbolo pode ser observado por estado. Os estados em um MEM são considerados "escondidos", isto é, cada estado representa uma probabilidade sobre todos os símbolos. Desta forma, MEMs podem não fornecer exatamente qual seqüência de estados produziu uma dada seqüência de símbolos gerados por um modelo, mas possibilitam calcular a seqüência de estados com maior probabilidade de ter gerado a seqüência observada de símbolos.

3.1.1. Elementos de um MEM

Um conjunto completo de parâmetros de um MEM discreto de primeira ordem é descrito pela seguinte notação:

$$l = (N, M, T, A, B, p) \tag{3.1}$$

onde:

N: Número de estados distintos em um modelo. Os estados individuais são rotulados como:

$$S = \{s_1, s_2, \ldots, s_N\} \tag{3.2}$$

e o estado no tempo *t* é rotulado como q_t.

M: Número de símbolos distintos observados por estado (alfabeto), os quais representam a saída física de um modelo. O conjunto de símbolos individuais pode ser denotado por:

$$V = \{v_1, v_2, \ldots, v_N\} \tag{3.3}$$

T: Comprimento da seqüência de observações. Uma seqüência de observações pode ser denotada por:

$$O = \{o_1, o_2, \ldots, o_T\} \tag{3.4}$$

e o_t denota as observações no tempo *t*;

A: Matriz de transição de estados. $A = \{a_{ij}\}$, onde:

$$a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i], \quad 1 \le j \le N, \tag{3.5}$$

isto é, a probabilidade do estado S_j no tempo $(t+1)$ dado o estado S_i no tempo t. Os elementos em A devem apresentar as seguintes propriedades:

$$a_{ij} \ge 0, \qquad \forall j, i \tag{3.6}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i \tag{3.7}$$

B: A distribuição da probabilidade das observações no estado j. $B = \{b_j(k)\}$, onde:

$$b_j(k) = P[o_t = v_k \mid q_t = S_j], \qquad 1 \le k \le M, \quad 1 \le j \le N \tag{3.8}$$

A probabilidade do símbolo v_k sendo observado no tempo *t*, dado o estado S_j no tempo *t*, $b_j(k)$ deve obedecer as seguintes propriedades:

$$b_j(k) \geq 0 \qquad 3.9$$

$$\sum_{K=1}^{M} b_j(k) = 1 \qquad 3.10$$

p : Probabilidade do estado inicial $p = \{p_i\}$, onde:

$$p_i = P[q_1 = S_i] \qquad 1 \leq i \leq N \qquad 3.11$$

Esta probabilidade deve ser não negativa e

$$\sum_{i=1}^{N} p_i = 1 \qquad 3.12$$

3.1.2. Tipos de MEMs

De um modo geral existem três tipos de estruturas de MEM: os modelos sem restrição chamados de ergóticos, os modelos seqüenciais e os modelos paralelos. Nos modelos ergóticos (Figura 3.1(a)) todas as transições possíveis entre os estados são permitidas e qualquer estado pode ser atingido por qualquer outro estado em um número finito de passos.

Os modelos seqüenciais e os modelos paralelos fazem parte dos modelos que seguem a topologia de Bakis, também chamados "*left-right*". Os modelos seqüenciais (Figura 3.1(b)) funcionam segundo uma evolução em série do modelo através de seus estados, não sendo permitidas transições para estados que estão abaixo do estado corrente. Qualquer um destes estados pode ser saltado no curso do processo.

Nos modelos paralelos (Figura 3.1(c)), muitos caminhos são permitidos e cada um desses caminhos pode saltar um ou vários estados do modelo.

Neste trabalho, a maioria dos modelos utiliza a estrutura seqüencial, porque esta estrutura representa melhor o sinal da escrita, uma vez que esta ocorre da esquerda para a direita. Para a modelagem de palavras foram utilizados modelos ergóticos. A escolha desta topologia foi devido ao fato de que nesta Meta-Classe são combinadas uma variedade muito grande de informações: palavras, letras, ruídos, etc., com uma grande variação no tamanho da

seqüência de observações, e baseado no trabalho descrito em [Morita, et al. 2002] onde um modelo ergótico se mostrou mais adequado para a modelagem de palavras.

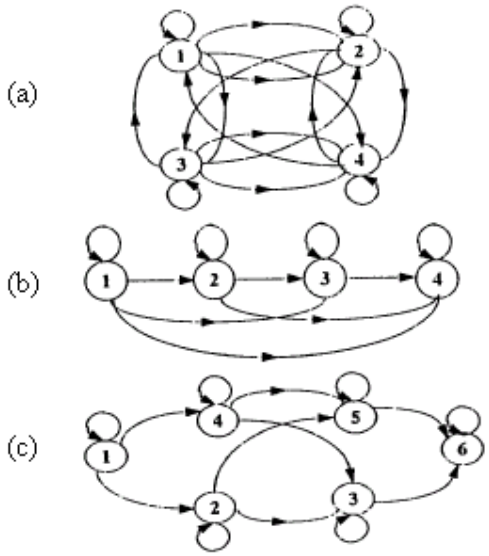

Figura 3.1. Tipos de estruturas de MEM
(a) modelo sem restrições; (b) modelo seqüencial; (c) modelo paralelo.
Adaptado de [Britto, et al. 2001a]

3.1.3. Densidade de Observação Discreta, Contínua e Semi-Contínua

Cada estado de um MEM tem uma distribuição da probabilidade de observação do símbolo (parâmetro B), que descreve a probabilidade de um símbolo v_k ser observado neste estado.

Em um MEM discreto, a probabilidade de observar o símbolo v_k durante o estado S_j, denotado por $b_j(k)$, é definida a partir da distribuição calculada sobre o conjunto de todos os possíveis símbolos no sistema. Esta distribuição é não paramétrica e quantificada, isto é, não é necessário um conhecimento prévio sobre a forma da distribuição para modelar um sinal, e cada observação pode receber um único valor discreto de um conjunto finito. Desta forma, é necessário apenas ter um conjunto de treinamento suficiente para modelar um sinal. No entanto, para a maioria das aplicações as observações são sinais contínuos. Nestes casos um MEM discreto pode ser usado somente após o processo de quantização de um sinal com o objetivo de criar um *codebook*. O custo disto é que o processo de quantização usualmente apresenta uma distorção que pode degradar a performance significativamente e,

além disso, para adicionar uma nova classe no sistema, são necessários a reconstrução do *codebook* e o re-treinamento de todos os modelos do sistema.

O uso de MEMs contínuos evita a distorção de quantização e o re-treinamento do sistema, visto que não existe *codebook*. No entanto, existem alguns custos: uma função de densidade da probabilidade contínua deve ser definida previamente, e mais exemplos de treinamento são requeridos para uma estimativa acurada dos parâmetros do MEM. Uma função mista de densidade de probabilidade, ponderada usando a soma do número de distribuições paramétricas, pode ser usada para encontrar a melhor forma de refletir a distribuição das observações.

Os MEMs semi-contínuos podem ser usados para evitar as distorções causadas pela quantização de sinais contínuos na modelagem de MEMs discretos, assim como para reduzir a quantidade de dados e a complexidade computacional para treinamento de MEMs contínuos. A idéia é que podem existir similaridades nos dados entre observações que não representam a mesma origem. Isto significa que agrupando os dados de uma forma não supervisionada poderão ser criados agrupamentos com limites de classes cruzados. Qualquer distribuição de estados pode ser representada com uma combinação ponderada de protótipos gaussianos conhecidos como densidades semicontínuas.

Em MEMs semi-contínuos a quantização de vetores (VQ), usada para modelar sinais contínuos com MEMs discretos é representada pelo conjunto de funções de densidade de probabilidade contínua, cujas distribuições são sobrepostas. Neste *codebook* cada *codeword* pode ser representado por uma função de densidade da probabilidade contínua. A operação de VQ produz valores de funções contínuas de probabilidade de densidade $f(x|v_j)$ para todos os *codewords* v_j.

O problema deste método é que seu foco está no agrupamento de observações similares, e não em observações que fornecem informações discriminantes.

No método proposto foi escolhido um MEM discreto, visto que o conjunto de dados disponível para treinamento é pequeno e em MEMs contínuos e semi-contínuos são necessários mais exemplos para treinamento.

3.1.4. Treinamento dos MEMs

O treinamento dos MEMs utilizados no método proposto foi feito utilizando-se o algoritmo *Baum_Welch*. Este algoritmo é baseado no critério de Máxima Verossimilhança

(MV). Neste critério, dada uma base de dados de treinamento composta de uma seqüência de observações, os parâmetros do MEM são primeiramente inicializados e então re-estimados iterativamente de tal forma que a probabilidade do modelo produzido pela seqüência de treinamento aumenta. O processo de treinamento é finalizado quando a probabilidade atinge o valor máximo.

O primeiro passo consiste no cálculo de $P(O \mid l)$, isto é, a probabilidade de uma seqüência de observação *O*, dado um modelo l. Este passo é chamado de problema de avaliação, isto é, dado um modelo e uma seqüência de observações, como calcular a probabilidade de a seqüência observada ser produzida pelo modelo? A forma mais eficiente de resolver este problema é utilizando os procedimentos *forward-backward.*

A variável *forward*, $a_t(i)$, é a probabilidade da seqüência de observações parciais $o_1, o_2, ..., o_t$ (até o tempo *t*) e estado S_i no tempo *t*, dado o modelo l, onde:

$$a_t(i) = P(o_1, o_2, ..., o_t,\ q_t = s_i \mid l), \tag{3.13}$$

$a_t(i)$ pode ser obtido através do seguinte algoritmo:

Inicialização:

$$a_1(i) = p_i b_i(o_1) \qquad 1 \le i \le N, \tag{3.14}$$

Indução:

$$a_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i)\, a_{ij}\right] b_j(O_{t+1}), \qquad 1 \le t \le T-1, \quad 1 \le j \le N, \tag{3.15}$$

Término:

$$P(O \mid l) = \sum_{i=1}^{N} a_T(i) \tag{3.16}$$

A variável *backward* $b_t(i)$ é definida como a probabilidade da seqüência de observações parciais de *t+1* até o fim, dado o estado S_i no tempo *t* e o modelo l, onde:

$$b_t(i) = P(o_{t+1}, o_{t+2}, ..., oT \mid q_t = s_i, l), \tag{3.17}$$

$b_t(i)$ pode ser obtido através do seguinte algoritmo:

Inicialização:

$$b_T(i) = 1, \quad 1 \le i \le N, \tag{3.18}$$

Indução:

$$b_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) b_{t+1}(j), \quad \begin{matrix} t = T-1, T-2, \ldots 1 \\ 1 \le i \le N, \end{matrix} \tag{3.19}$$

Término:

$$P(O \mid l) = \sum_{i=1}^{N} p_i B_i(O_1) b_1(i) \tag{3.20}$$

O *Baum_Welch* (BW) é um algoritmo iterativo baseado nas probabilidades de *forward* e *backward* e tem o objetivo de ajustar os parâmetros do modelo $l = (A, B, p)$ para maximizar $P(O|l)$.

Após a definição da variável *backward*, é possível definir a probabilidade de estar no estado S_i no tempo *t*, e no estado S_j no tempo *t+1*, dado o modelo e a seqüência de observação, a qual é denotada por:

$$x(i, j) = P(q_t = S_i, q_{t+1} = S_j \mid O, l), \tag{3.21}$$

e é calculada usando:

$$\zeta_t(i, j) = \frac{a_t(i) a_{ij} b_j(o_{t+1}) b_{t+1}(j)}{P(O \mid l)}, \tag{3.22}$$

Considerando $g_t(i)$ como a probabilidade de estar no estado S_i no tempo *t*, dada a seqüência de observação *O*, e o modelo l.

$$g_t(i) = P(q_t = S_i \mid O, l), \tag{3.23}$$

a qual pode ser calculada usando as variáveis *forward* e *backward*:

$$\gamma_t(i) = \frac{a_t(\mathrm{i})\,\beta_t(\mathrm{i})}{\mathrm{P}(\mathrm{O} \mid l)}, \tag{3.24}$$

O número esperado de transições de S_i é denotado por:

$$\sum_{t=1}^{\mathrm{T-1}} \gamma_t(\mathrm{i}), \tag{3.25}$$

e o número esperado de transições de S_i para S_j é denotado por

$$\sum_{t=1}^{T-1} \xi_t(i, j), \tag{3.26}$$

A re-estimação dos parâmetros *p, A e B* é possível utilizando-se as fórmulas abaixo:

1) Freqüência esperada (número de vezes) no estado S_i no tempo $t = 1$

$$\overline{p} = g_1(i), \tag{3.27}$$

2) Coeficiente de transição: número esperado de transições do estado S_i para o estado S_j, divido pelo número esperado de transições do estado S_i.

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} x(i, j)}{\sum_{T}^{T-1} g_t(i)}, \tag{3.28}$$

3) Probabilidade de observação do símbolo: número esperado de vezes no estado *j*, enquanto observa-se o símbolo v_k, dividido pelo número esperado de vezes no estado *j*.

$$\overline{b_j}(k) = \frac{\sum_{t=1, s.t. ot=vk}^{T} g_t(j)}{\sum_{t=1}^{T} g_t(j)} \tag{3.29}$$

3.1.5. Reconhecimento

O reconhecimento dos modelos criados no método proposto é feito pelos algoritmos *Viterbi* e *Level Building.* O algoritmo *Viterbi* é utilizado para a avaliação dos modelos de forma isolada e o algoritmo *Level Buiding* é utilizado para a concatenação dinâmica destes

modelos com o objetivo de segmentar e reconhecer linhas de texto extraídas do bloco do endereço.

Algoritmo *Viterbi*

O algoritmo *Viterbi* é um método de programação dinâmica para estimação da seqüência de modelos de estados com maior probabilidade de ter produzido as seqüências de observações. Dada uma seqüência de observações $O = \{o_1, o_2, ...o_T\}$ e o modelo l, este método encontra a melhor seqüência de estados $O = \{q_1, q_2, ...q_T\}$.

São consideradas as seguintes variáveis:

- $d_t(i)$: é o melhor resultado (probabilidade mais alta) ao longo de um caminho simples no tempo *t*, o qual leva em consideração as *t* primeiras observações e termina no estado *i*;
- $y_t(i)$: *array* usado para traçar o caminho de máxima probabilidade. Isto permite armazenar os estados com a máxima probabilidade do tempo 1 até *T*.

Descrição do algoritmo:

Inicialização:

$$\delta_1(\mathrm{i}) = \pi_\mathrm{i} \mathrm{b}_\mathrm{i}(\mathrm{O}_1), \quad 1 \le \mathrm{i} \le \mathrm{N} \qquad 3.30$$
$$\psi_1(\mathrm{i}) = 0,$$

Recursão:

$$\delta_\mathrm{t}(\mathrm{j}) = \max_{1 \le i \le N}[\delta_{\mathrm{t-1}}(\mathrm{i}) a_{ij}] b_j(o_t), \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \qquad 3.31$$

$$\psi_\mathrm{t}(\mathrm{j}) = \arg\max_{1 \le i \le N}[\delta_{\mathrm{t-1}}(\mathrm{i}) a_{ij}], \qquad \begin{matrix} 2 \le t \le T, \\ 1 \le j \le N \end{matrix} \qquad 3.32$$

Término:

$$P^* = \max_{1 \ge i \le N}[d_T(i)], \qquad 3.33$$

$$q_T{}^* = \arg\max_{1 \le i \le N}[d_T(i)], \qquad 3.34$$

Caminho de retorno (*Backtracking*):

$$q_t^* = \psi_{t+1}(q^*_{t+1}), \qquad t = T-1, T-2, \ldots 1. \tag{3.35}$$

Algoritmo *Level Buiding*

O algoritmo Level Building (LBA) é utilizado no método proposto para encontrar a seqüência de modelos MEM que melhor representa uma linha de texto desconhecida, extraída do bloco do endereço. Este algoritmo foi utilizado em [Britto 2001] para a segmentação-reconhecimento de strings numéricas manuscritas e por [Aas, et al. 1996] na segmentação e reconhecimento de textos impressos. A Figura 3.2 mostra uma estrutura *trellis* que descreve o LBA.

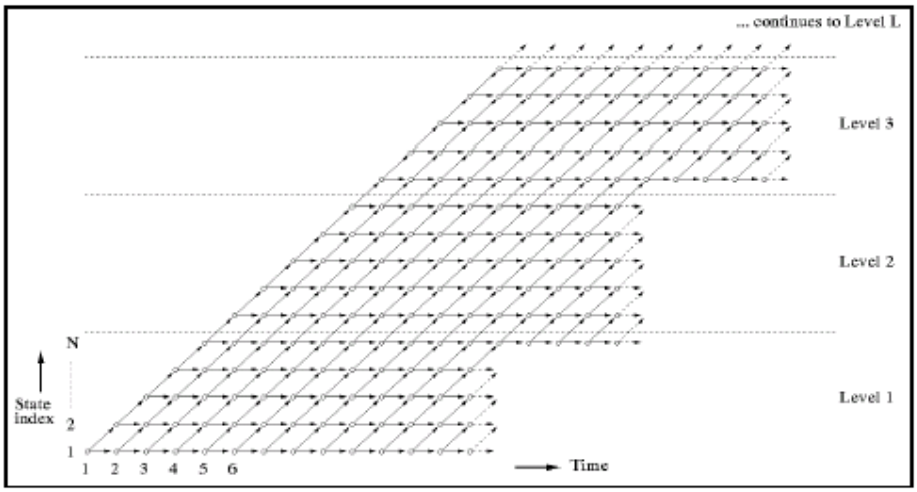

Figura 3.2 - Estrutura *trellis* para o LBA [Britto 2001].

Para descrever este algoritmo vamos considerar $l = \{l_1, l_2, \ldots, l_m\}$ como um conjunto de *M* MEMs *left-right*, cada um composto de *N* estados; *O* = {o_1, o_2, ..., o_T} como uma seqüência de observações de tamanho *T*; *t* como uma função tempo *(1 £ t £ T);* e *L* como o número máximo de níveis, o qual deve ser grande o suficiente para permitir uma seqüência correta de modelos a ser descoberta. Por exemplo, 10 níveis permitem um total de 10 modelos de caracteres a ser comparados a uma seqüência de entrada.

No primeiro nível (*l* = 1), cada modelo l_k (*1 ≤ k ≤ M*) é comparado a seqüência de observações do tempo (*t* = 1).

Inicialização:

$$d_1(1) = b_1^{\,l}(o_1) \tag{3.36}$$

$$d_1(1) = 0 \qquad 2 \le i \le N \tag{3.37}$$

Recursão:

$$d_t(j) = \max_{1 \le i \le N}[d_{t-1}(i) a_{ij}^{\,l}] b_j^{\,l}(o_t) \quad 2 \le t \le T \tag{3.38}$$

$$1 \le j \le N$$

Término

$$P^*(l,t,l) = d_t(N), \quad 2 \le t \le T \tag{3.39}$$

$$B^*(l,t,l) = -1 \tag{3.40}$$

Um nível de redução é executado no fim do nível. Neste ponto as equações 3.41 e 3.42 são utilizadas para manter, respectivamente, a probabilidade do melhor modelo comparado e o rótulo do modelo correspondente. A Equação 3.43 representa um ponteiro de retorno para o nível anterior.

$$P^{**}(l,t) = \max_{l} [P^*(l,t,l)] \tag{3.41}$$

$$\Omega(l,t) = \arg\max_{l}[P^*(l,t,l)] \tag{3.42}$$

$$B^{**}(l,t) = B^*\left[1, t, \arg\max_{l}[P^*(l,t,T)]\right] \tag{3.43}$$

Para níveis mais altos (l ≥ 2), o processo de inicialização deve ser:

$$d_t(1) = 0 \tag{3.44}$$

$$d_t(1) = \max[P^*(l-1,t-1), a_{11}^{\,l} d_{t-1}(1)] b_1^{\,l}(o_t), \quad 2 \le t \le T \tag{3.45}$$

Além disso, o array de ponteiros de retorno deve ser adicionado para manter o caminho do tempo nos níveis anteriores quando a comparação do modelo anterior é finalizada. Este array deve ser inicializado levando em conta a seguinte condição:

$$a_t(i) = \begin{cases} t-1 \; se \; P^*(l-1, t-1) > a_{11}{}^{l} d_{t-1}(1) \\ a_{t-1}(1) \text{ caso contrário} \end{cases} \quad 3.46$$

Durante o processo de recursão este array de ponteiros de retorno é atualizado, como:

$$a_t(j) = \arg\max_{1 \le i \le N} [d_{t-1}(i) a_{ij}{}^{l} \quad 3.47$$

Finalmente, no processo de término este array de ponteiros de retorno é reiniciado em ***B****, como:

$$B^*(l, t, l) = a_t(N), \quad 1 \le t \le T \quad 3.48$$

Depois de construir todos os níveis, ***P*****(*L,T*) representa a probabilidade da melhor comparação dos *L* modelos para a seqüência de observações. Com o objetivo de encontrar a melhor comparação, o array ***B**** poder ser usado no processo de retorno. A melhor ***string*** comparada é o máximo de ***P***** sobre todos os níveis.

3.1.6. Definição do número de estados de um MEM.

Wang, [1994] descreve um método para a definição do número possível de estados (*N*) de um MEM levando em conta estatísticas duracionais calculadas da base de treinamento.

A média de duração μ e a variação σ^2 coletadas de todas as seqüências de observações no conjunto de treinamento definem o possível *N* para cada numeral MEM:

$$\frac{\mu(\mu - 1) + \sigma^2}{\mu - 1 + \sigma^2} < N < \mu + 1 - \sqrt{2\sigma^2 + 1} \quad 3.49$$

Este método é utilizado para a estimação do número de estados dos MEMs utilizados no método proposto.

3.2. Processo de Quantização de Vetores

Para a utilização de MEMs discretos é necessário representar a imagem de um dígito ou ***string*** como uma seqüência de observações discretas. Para este fim, cada valor real das

primitivas extraídas deve ser quantizado para um número de símbolos discretos disponíveis em um *codebook* previamente calculado. Para criar este *codebook* é necessário aplicar o conceito de quantização de vetores (VQ).

Vamos assumir que $x = [x_1, x_2, ..., x_Z]$ é um vetor Z-dimensional cujos componentes $\{x_k, 1 \pounds k \pounds Z\}$ são valores reais, variáveis randômicas de amplitudes continuas. Na quantização de vetores, o vetor *x* é mapeado sobre outro valor real ou amplitude discreta, o vetor Z dimensional *y*. É usualmente dito que *x* é quantizado como *y*, e *y* é a quantização do valor de *x*. Isto pode ser denotado por:

$$y = q(x) \tag{3.50}$$

Onde $q(\ )$ é o operador de quantização e *y* é o vetor de saída correspondente para *x*. O valor de *y* é um conjunto finito de valores $\{y_i, 1 \le i \le L\}$, onde $y_i [y_{i1}, y_{i2}, ... y_{iZ}]$. O conjunto *Y* é chamado *codebook*, *L* é o tamanho do *codebook* (ou número de níveis), e $\{y_i\}$ é o conjunto de *code vectors*. Para construir o *codebook*, nós dividimos o espaço Z-dimensional do vetor randômico *x* em *L* regiões ou células e associamos a cada célula C_i um vetor y_i. O quantizador então determina o *code vector* y_1, se *x* está em C_i.

$$q(x) = y_i, \quad se \ \ x \in C_i \tag{3.51}$$

O mapeamento de *x* em *y* resulta em um erro de quantização, e uma medida de distorção *d*(*x*, *y*) pode ser definida entre eles. A medida de distorção *d*(*x*, *y*) é também conhecida como dissimilaridade ou medida de distância. A medida de distorção mais comum é o erro médio quadrado, onde:

$$d(x, y) = \frac{1}{K} \sum_{k=1}^{N} \left(x_k - y_k \right)^2 \tag{3.52}$$

A quantização é ótima quando a distorção é minimizada sobre todos os *L*-níveis quantizados. Existem duas condições necessárias para a optimalidade. A primeira condição é que o quantizador ótimo é realizado usando uma distorção mínima ou regra de seleção do vizinho próximo.

$$q(x) = y_i, \ \ se \ d(x, y_i) \le d(x, y_j), \quad j \ne i, \ \ i \le j \le L \tag{3.53}$$

Isto significa que o quantizador seleciona o *code vector* que resulta em uma distorção mínima com respeito à *x*. A segunda condição necessária para a optimalidade é que cada *code vector* y_i, é escolhido para minimizar a média de distorção na célula C_i. Considerando $\{x(z), 1 \leq z \leq M\}$ como o conjunto de vetores de treinamento, e M_i como um subconjunto destes vetores na célula C_i. A distorção média D_i é então dada por:

$$D_i = \frac{1}{M_i} \sum_{x \in C_i} d(x, y_i) \qquad 3.54$$

O vetor que minimiza a média de distorção na célula C_i é chamado de centróide de C_i, e é denotado como:

$$Y_i = cent(C_i) \qquad 3.55$$

Um método bastante conhecido para a construção de *codebooks* é o algoritmo iterativo chamado *K-Means*, onde $K = L$ (tamanho do *codebook*). O algoritmo divide o conjunto de vetores de treinamento $\{x(k)\}$ em L agrupamentos ou clusters C_i de uma forma que as duas condições de optimalidade sejam satisfeitas. Na descrição do algoritmo, m é o índice de iteração e $C_i(m)$ é o i^o agrupamento da iteração m, com $y_i(m)$ seu centróide. O algoritmo é descrito abaixo:

1) Passo de Inicialização:

   Setar m = 0.

   Escolha através de um método adequado um conjunto inicial de *code vectors* $y_i(0)$, $1 \leq i \leq L$.

2) Passo de Classificação:

   Classifique o conjunto de vetores de treinamento $\{x(z), 1 \leq z \leq M\}$ dentro de um agrupamento C_i, pela regra do vizinho próximo.

   $x \in C_i(m), \quad se\ d(x, y_i) \leq d(x, y_j), \quad para\ todo\ j \neq i,\ 1 \leq j \leq L$

3) Passo de atualização dos *Code Vectors*:

   m = m +1.

   Atualize o *code vector* de cada agrupamento pelo cálculo do centróide dos vetores de treinamento em cada agrupamento.

   $Y_i(m) = cent(C_i(m)),\ 1 \leq i \leq L$

4) Passo de teste do término:

Se o decréscimo em todas as distorções $D(m)$ da iteração m relativo a $D(m-1)$ está abaixo de um certo limiar, pare; caso contrário vá para o Passo de Classificação.

3.3. Considerações Finais

Neste capítulo foram descritos os Modelos Escondidos de Markov e um processo de Quantização Vetorial. Os MEMs são utilizados no método proposto para a modelagem da informações que compõem o bloco do endereço. Foram descritos os principais conceitos referentes aos MEMs, o conjunto de parâmetros de um MEM discreto de primeira ordem, os tipos de MEMs levando em conta a topologia: ergótico e *left-right*, e a distribuição da probabilidade de observação para cada estado: discreta, contínua e semi-contínua. Os principais algoritmos para treinamento e reconhecimento dos modelos foram apresentados.

Um processo de quantização vetorial, baseado no algoritmo *k-means* foi descrito. Este processo é utilizado para o mapeamento do vetor de primitivas extraídas da imagem em observações discretas. Ele é necessário por causa do uso de MEMs discretos.

No próximo capítulo, o método proposto é descrito detalhadamente.

4. Método Proposto

Este capítulo descreve em detalhes o método proposto. A Seção 4.1 apresenta uma visão geral do método, enquanto na Seção 4.2 é descrita a primeira etapa, na qual o bloco do endereço é segmentado em linhas e pré-processado para a correção da linha de base da escrita. A Seção 4.3. descreve o processo de criação dos modelos MEM que são utilizados nas próximas etapas do método. São descritas as primitivas extraídas da imagem, as Meta-Classes criadas, a topologia dos modelos e os processos de treinamento e avaliação dos modelos. Na Seção 4.4. são apresentadas a etapa de segmentação-reconhecimento das linhas do bloco do endereço e a etapa de verificação e re-classificação dos resultados obtidos. O processo de localização do CEP é descrito na Seção 4.5 e a Seção 4.6 apresenta as considerações finais deste capítulo.

4.1. Visão Geral do Método Proposto

O CEP atualmente utilizado no Brasil é composto de 8 (oito) dígitos, divididos em duas partes, uma de 5 (cinco) algarismos e outra de 3 (três), separadas por um traço. Cada dígito pode assumir valores entre 0 e 9. Embora o CEP seja composto de 8 dígitos, em alguns exemplos extraídos da base de dados dos correios foi observado que somente os 5 primeiros dígitos são preenchidos e que os 3 últimos dígitos nem sempre são separados pelo traço. Alguns exemplos podem ser observados na Figura 4.1.

Uma visão geral do método proposto pode ser observada na Figura 4.2. O método consiste na análise das linhas que compõem o bloco do endereço em busca de uma seqüência de componentes que represente a estrutura do CEP: 5 ou 8 dígitos separados ou não pelo traço, podendo ser precedidos da palavra CEP.

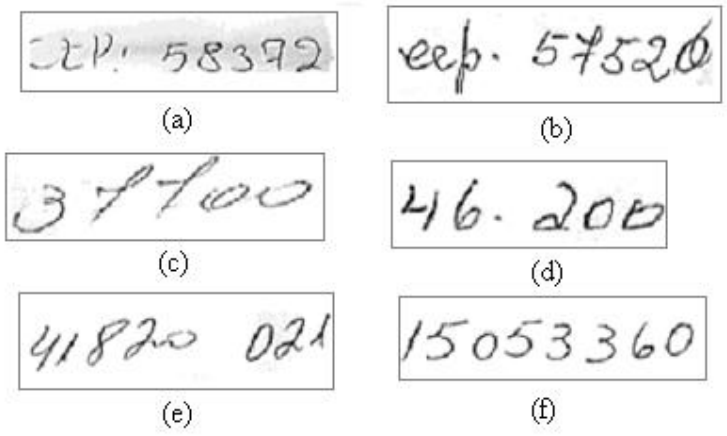

Figura 4.1 - Exemplos de CEP, extraídos da base de dados dos Correios.

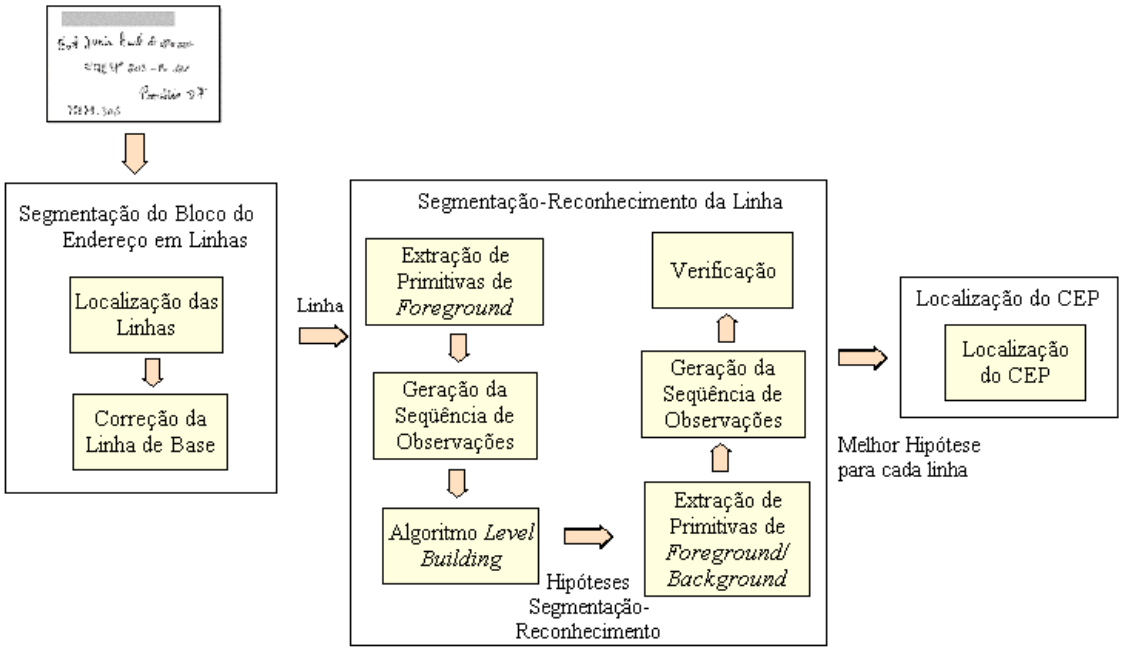

Figura 4.2 - Visão geral do método proposto

A primeira etapa do método consiste na segmentação do bloco do endereço em linhas, seguida por um processo de pré-processamento para correção da linha de base da escrita.

No próximo estágio, cada linha é enviada para um classificador baseado no algoritmo *Level Building*, descrito no capítulo 3. Este classificador encontra as melhores hipóteses de segmentação-reconhecimento utilizando uma estratégia de segmentação implícita e modelos MEM treinados com exemplos de manuscritos isolados (dígitos, palavras, caracteres, etc.). Na etapa seguinte, as hipóteses de segmentação-reconhecimento são verificadas e re-classificadas. Uma vez identificados os pontos de segmentação e os componentes das linhas, a

localização do CEP consiste na busca de uma seqüência de elementos que represente a estrutura do CEP e a localização deste na linha já é fornecida pelo classificador.

4.2. Segmentação do bloco do endereço em linhas

O objetivo é dividir o bloco do endereço em linhas. Este processo é composto das seguintes etapas:

- Localização das linhas no bloco do endereço: este processo será baseado nas projeções horizontais da imagem. Projeções Horizontais da imagem do documento é a técnica mais comumente utilizada na segmentação de linhas. Os picos e vales encontrados a partir desta técnica são utilizados para localizar os limites entre linhas. Um exemplo da detecção de linhas através de histogramas horizontais pode ser observado na Figura 4.3.
- Correção da linha de base: a correção da linha de base da escrita visa detectar e corrigir o ângulo de inclinação com a horizontal de modo a deixar a escrita horizontal. Um exemplo desta correção pode ser observado na Figura 4.4. Nesta etapa será utilizado o método proposto por [Yacoubi, et al. 1999] onde a correção é feita alinhando os mínimos locais do contorno inferior da palavra após a filtragem das partes da palavra que correspondem aos descendentes.

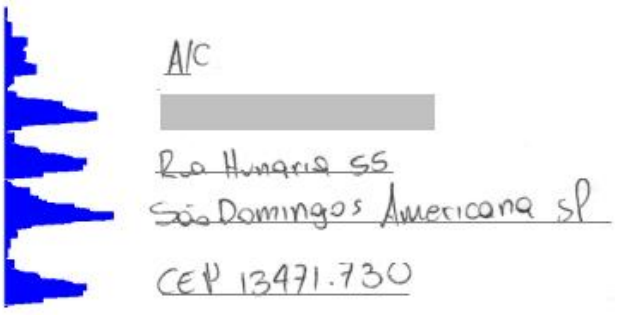

Figura 4.3 - Detecção de linhas utilizando histogramas horizontais.

Figura 4.4 - Correção da linha de base. (a) Imagem Original, (b) Correção da linha de base. [Yacoubi, et al. 1999].

4.3. Criação dos modelos MEM

O objetivo é a criação de modelos MEM que representem as informações que compõem o bloco do endereço: dígitos, palavras, caracteres, hífen, a palavra CEP, etc, que serão utilizados pelo algoritmo *Level Building* para segmentação-reconhecimento das linhas e no processo de verificação dos resultados. A Figura 4.5 apresenta as etapas deste processo.

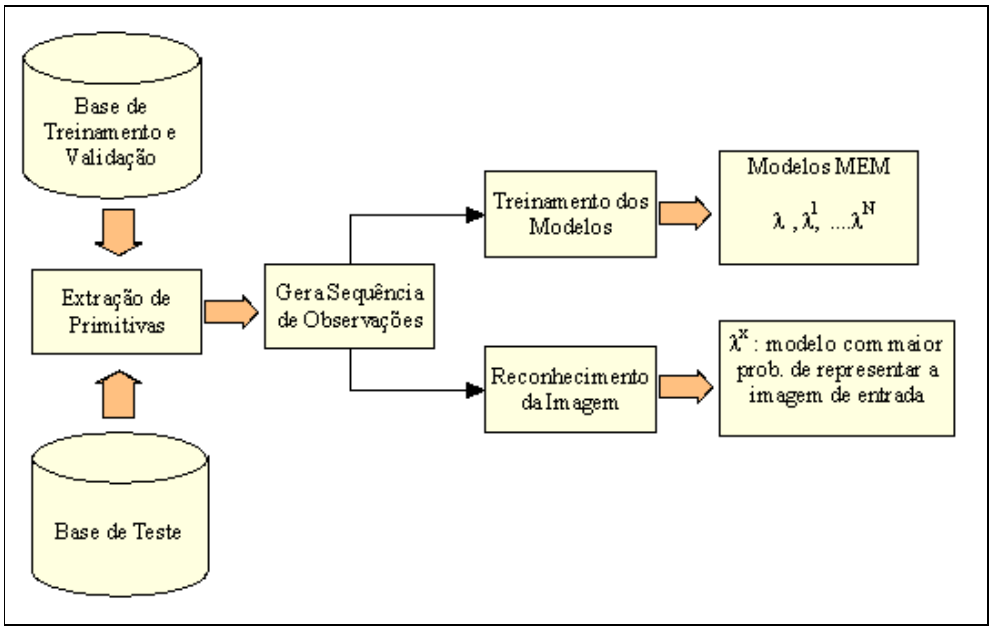

Figura 4.5 - Etapas para a criação dos modelos MEM.

Os modelos MEM são criados a partir de uma base de dados de treinamento, composta de imagens extraídas de envelopes postais e cheques bancários brasileiros. Para cada exemplo da base de treinamento, são extraídas primitivas locais e globais baseadas nas colunas da imagem. Na criação dos modelos para verificação são extraídas também primitivas baseadas

nas linhas da imagem e na etapa de reconhecimento estes dois conjuntos de primitivas são combinados.

Para que estas primitivas possam ser utilizadas pelos MEMs, cada vetor de primitivas é quantizado em um dos símbolos disponíveis em um *codebook*, gerando uma seqüência de observações. Para a criação dos modelos, esta seqüência de observações é enviada para o algoritmo de *Baum-Welch*, descrito do capítulo 3. Para a avaliação dos modelos criados, a seqüência de observações é enviada para o algoritmo *Viterbi*, descrito no capítulo 3, e este indica qual o modelo tem a maior probabilidade de representar a imagem de entrada.

4.3.1. Extração de Primitivas

Embora existam muitos métodos de extração de primitivas propostos na literatura, sua eficiência depende do tipo de informação que está sendo analisada: palavras, caracteres, números, etc. Como o bloco do endereço pode conter um grande léxico de informações e com muitas variações no tamanho e no estilo da escrita, fica difícil escolher um método que seja eficiente para toda esta variabilidade.

O conjunto de primitivas *Foreground*/*Backgound Features* foi proposto em [Britto, et al. 2001b] e foi utilizado no método proposto devido aos seguintes motivos:

- Este método tem apresentando bons resultados no reconhecimento de strings numéricas manuscritas, mesmo quando os dígitos se apresentam conectados, fragmentados ou sobrepostos, o que pode ser encontrado nos envelopes postais.
- Este método pode ser utilizado em um processo de segmentação implícita, onde não há a necessidade de uma segmentação a priori da string, a qual é a estratégia do método proposto.

O problema da grande variabilidade de informações poderá ser resolvido com o uso de Meta-Classes, as quais serão descritas posteriormente.

Foreground Features

O conjunto de primitivas é composto de primitivas locais e globais extraídas de cada coluna da imagem. As primitivas locais são baseadas em transições dos pixels de primeiro para segundo plano e vice-versa. Para cada transição, a direção média e variância correspondente são obtidas por meio de estimadores estatísticos. Estes estimadores são mais adequados para observações direcionais, uma vez que eles são baseados em uma escala

circular. Por exemplo, dadas as observações direcionais $a_1 = 1^0$ e $a_2 = 359^0$, eles fornecem uma direção média $(\overline{a})$ de 0^0 ao invés de 180^0 calculada pelos estimadores tradicionais.

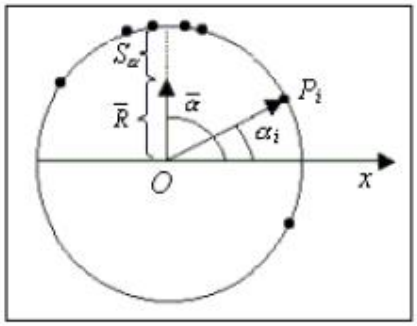

Figura 4.6 - Média circular de direção $\overline{a}$ e variância S_a para a distribuição $F(a_i)$ [Britto 2001].

Considerando $a_1,...,a_i,...,a_N$ como sendo um conjunto de observações direcionais com distribuição $F(a_i)$ e tamanho *N*. A Figura 4.6 mostra que a_i representa o ângulo entre o vetor unitário $\overline{OP_i}$ e o eixo horizontal, enquanto P_i é o ponto de interseção entre $\overline{OP_i}$ e o círculo unitário. As coordenadas cartesianas de P_i são definidas como:

$$(\cos(a_i),\ sin(a_i)) \qquad 4.1$$

A média circular de direção $\overline{a}$ das N observações direcionais no círculo unitário correspondem à direção do vetor resultante $(\overline{R})$ obtido pela soma dos vetores unitários $(\overline{OP}_1,...,\overline{OP}_i,...,\overline{OP}_N)$. O Centro de gravidade $(\overline{C},\overline{S})$ das N coordenadas $(\cos(a_i),\ sin(a_i))$ é definido como:

$$\overline{C} = \frac{1}{N}\sum_{i=1}^{N} F(a_i) * \cos(a_i) \qquad 4.2$$

$$\overline{S} = \frac{1}{N}\sum_{i=1}^{N} F(a_i) * sin(a_i) \qquad 4.3$$

Estas coordenadas são utilizadas para estimar o tamanho médio de $\overline{R}$, com a seguinte equação:

$$\overline{R} = \sqrt{(\overline{C}^2 + \overline{S}^2)} \qquad 4.4$$

A média circular da direção pode ser obtida pela resolução de uma das seguintes equações:

$$\cos(\overline{a}) = \frac{\overline{C}}{\overline{R}} \qquad 4.5$$

$$sin(\overline{a}) = \frac{\overline{S}}{\overline{R}} \qquad 4.6$$

A variância circular de $\overline{a}$ é calculada com a seguinte equação:

$$S_a = 1 - \overline{R} \qquad 0 \le S_a \le 1 \qquad 4.7$$

Para o cálculo de $\overline{a}$ e S_a para cada transição da imagem da ***string***, considera-se $\{0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0\}$ como o conjunto de observações direcionais. $F(a_i)$ é calculado pela contagem do número de pixels pretos sucessivos na direção a_i de uma transição até o encontro de um pixel branco. Na Figura 4.7 as transições em uma coluna do número 5 são enumeradas de 1 a 6, e as possíveis observações direcionais das transições 3 e 6 são mostradas.

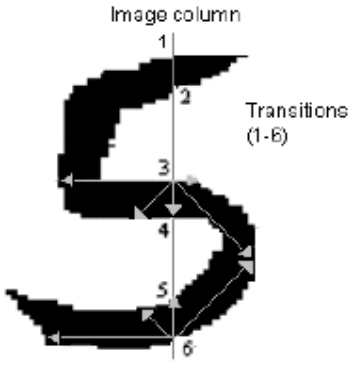

Figura 4.7 - Transições em uma coluna da imagem do número 5, e as observações direcionais usadas para estimar a direção média para as transições 3 e 6 [Britto 2001].

Além das informações direcionais, são extraídas duas outras primitivas locais:

- Distância relativa de cada transição, levando em conta o topo da *bounding box* da string. Esta primitiva é utilizada para compensar a falta de um esquema de zoning;
- Se a transição pertence ao contorno externo ou interno, o que mostra a presença de *loops* na imagem da *string*;

Para cada coluna da imagem da string são consideradas 8 possíveis transições, resultando neste ponto em um vetor de 32 primitivas, 8 (direção média) + 8 (variância) + 8 (posição em relação a *bounding box*) + 8 (se a transição pertence ao contorno externo ou interno).

As primitivas Globais são:

- Projeções Verticais dos pixels pretos para cada coluna;
- Diferença entre as Projeções Verticais de colunas adjacentes;

No final do processo tem-se um vetor de 34 primitivas normalizadas ente 0 e 1, para cada coluna da imagem analisada.

Background Features

Este conjunto de primitivas é adicionado ao conjunto de primitivas de *Foreground* e é utilizado somente para a criação dos modelos MEM que farão parte do estágio de verificação, onde os resultados obtidos na segmentação-reconhecimento das linhas de texto são verificados e re-classificados. Estas primitivas são baseados em informações de concavidade e são utilizadas para colocar em evidência propriedades topológicas e geométricas da imagem. Cada vetor de concavidades representa o número de pixels brancos que pertencem a uma configuração de concavidade específica. O rótulo para cada pixel branco é escolhido baseado nas 4 direções do código de Freeman. Cada direção é explorada até o encontro de um pixel preto ou dos limites impostos pela *bounding box* da imagem. Um pixel branco é rotulado se, pelo menos, duas direções consecutivas encontram pixels pretos (observe a Figura 4.8).

Desta forma, temos 9 possíveis configurações de concavidade (observe a Figura 4.9). Além disso, são considerados mais 4 configurações, para detectar mais precisamente a presença de *loops*. O tamanho do vetor de concavidades é 13 e este é normalizado entre 0 e 1 pela divisão de cada valor pelo total de códigos de concavidade calculados para cada coluna ou linha da imagem.

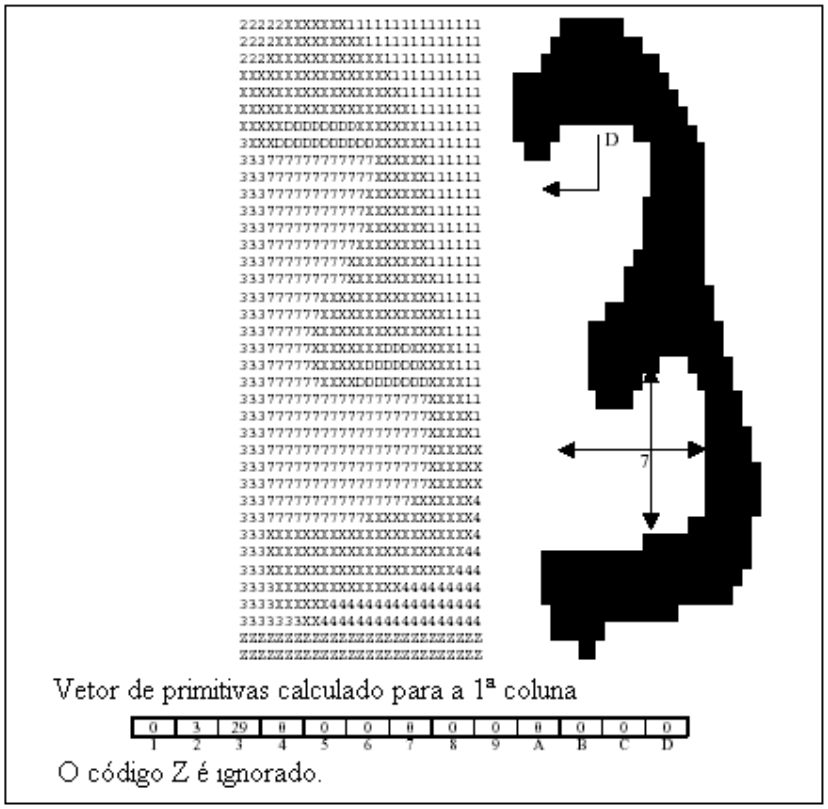

Figura 4.8 - Primitivas de concavidade calculadas para o número 3, adaptada de [Britto 2001].

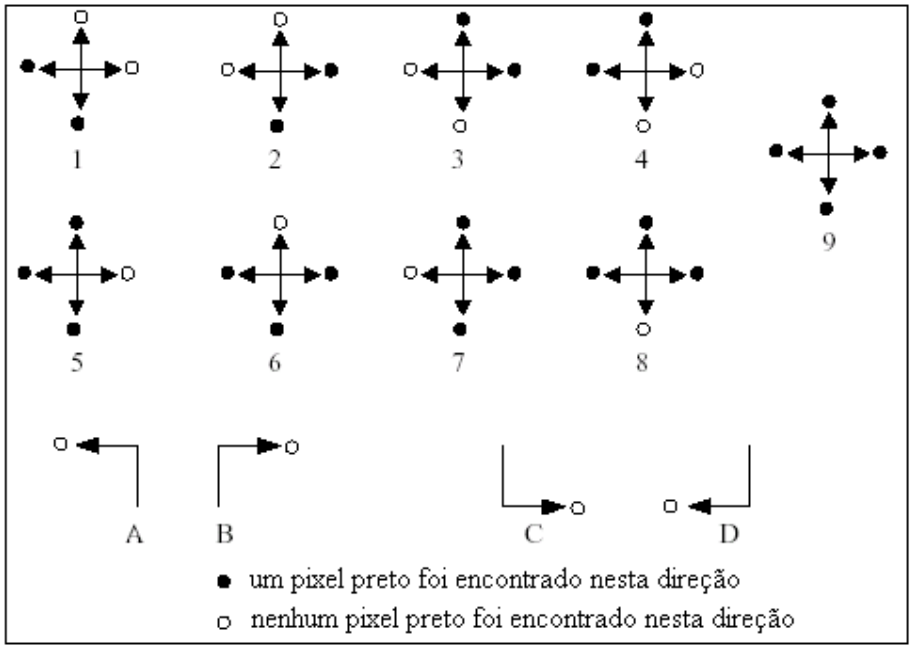

Figura 4.9 - Configurações de concavidade utilizadas para rotular os pixels brancos, adaptada de [Britto 2001].

Neste ponto temos um vetor de 47 primitivas, 13 primitivas de *background* e 34 primitivas de *foreground*, calculado para cada coluna ou linha da imagem.

4.3.2. Geração da Seqüência de observações

Como visto no Capítulo 3, quando são utilizados MEMs, é preciso representar uma imagem como uma seqüência de observações. Cada vetor gerado na etapa anterior deve ser mapeado em um símbolo discreto disponível em um *codebook*. O comprimento da seqüência de observações corresponde ao número de colunas da *bounding box* da string.

Para a criação deste *codebook* foi utilizado um processo de quantização vetorial, descrito no Capítulo 3. Neste processo, foram extraídas primitivas de todos os exemplos da base de treinamento, e utilizando-se o algoritmo *K-Means*, estas primitivas foram agrupadas em um número definido de símbolos, formando um *codebook*. Este *codebook* é composto de 256 entradas, definido experimentalmente.

4.3.3. Meta-Classes

Como descrito no capítulo 2, Meta-Classes têm sido utilizadas para reduzir o tamanho do léxico de determinados campos que estão sendo analisados, como em [Morita, et al. 2002] onde Meta-Classes foram utilizadas para reduzir o tamanho do léxico dos campos dia e ano na segmentação de datas manuscritas, presentes em cheques bancários brasileiros.

A idéia consiste em agrupar determinadas classes em uma única Meta-Classe segundo algum critério.

O uso de Meta-Classes no método proposto deve-se aos seguintes fatores:

- No bloco do endereço existe uma grande quantidade de campos que podem ser encontrados, tais como palavras, caracteres, dígitos, traços, barras, ruídos, etc., tornando inviável a modelagem de cada um destes campos. Uma alternativa é agrupá-los em Meta-Classes.
- Como o objetivo do método proposto é a localização do CEP e não o seu reconhecimento, a identificação dos componentes que são dígitos já é suficiente não sendo necessário identificar qual dígito este representa. Desta forma os dígitos podem ser agrupados em única Meta-Classe.
- A base de dados disponível é pequena, podendo não ser representativa de todas as variações que podem ser encontradas. Com o agrupamento dos campos, o número de exemplos necessários para treinamento, validação e teste aumenta.

A divisão dos componentes presentes no bloco do endereço em Meta-Classes foi feita após uma análise dos exemplos de envelopes postais disponíveis. As Seguintes Meta-Classes foram investigadas:

- Agrupamento de todos os dígitos em uma única Meta-Classe ($M_{0...9}$).
- Agrupamento de todas as palavras em uma única Meta-Classe (M_P): são consideradas palavras os componentes conexos com mais de um caractere.
- Agrupamento de todos os caracteres em uma única Meta-Classe (M_C): são incluídas nesta Meta-Classe todas as letras de A a Z, barras, vírgula, etc.
- Agrupamento das Meta-Classes palavras e caracteres em uma única Meta-Classe (M_{PC}).
- Hífen (M_H): o traço que separa os dígitos do CEP.
- A palavra Cep (M_{CEP}): como em 66% dos casos os dígitos do CEP vem precedidos da palavra CEP, a sua localização pode auxiliar na busca do CEP.

Foram realizados testes com várias combinações destas Meta-Classes, os quais serão descritos no Capítulo 5, e para cada um destes testes o *codebook* foi recalculado.

4.3.4. Definição, Treinamento e Avaliação dos Modelos

Os modelos MEMs são criados a partir de uma base de dados de treinamento, composta de imagens extraídas de envelopes postais e cheques bancários brasileiros e baseados nas Meta-Classes definidas.

Os MEMs foram escolhidos porque estes têm sido utilizados com sucesso na modelagem de palavras e cadeias numéricas e devido a possibilidade de se modelar caracteres ou dígitos de forma isolada e facilmente agrupá-los com o objetivo de representar palavras ou frases. Outro ponto positivo está na possibilidade de se inserir informações de contexto nos modelos, como em [Britto, et al. 2001b] onde os MEMs foram treinados com dígitos isolados mas considerando informações contextuais da string a que estes pertenciam, tais como a inclinação da *string* e a variação do tamanho das intra-*strings*.

Foram escolhidos MEMs discretos porque o número de exemplos disponível para o treinamento dos modelos é pequeno. Em MEMs contínuos e semi-contínuos são necessários um grande número de exemplos para treinamento.

Definição dos MEMs

Para a modelagem das Meta-Classes palavras e CEP foi utilizado um modelo ergótico (Figura 4.10(a)). A escolha desta topologia é devido ao fato de que em uma única Meta-Classe são combinadas uma variedade muito grande de informações: palavras, letras, ruídos, etc., com uma grande variação no tamanho da escrita, e baseado no trabalho descrito em [Morita, et al. 2002] onde um modelo ergótico se mostrou mais adequado para a modelagem de palavras.

Para as demais Meta-Classes foi utilizada uma topologia "*left-right*" (Figura 4.10(b)), porque esta representa melhor o sinal da escrita, uma vez que esta ocorre da esquerda para a direita.

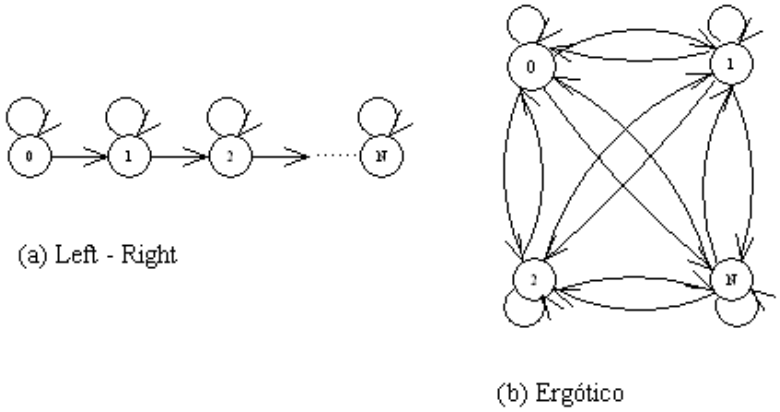

Figura 4.10 - Topologia dos modelos

Em todos os casos, a definição do número de estados foi feita utilizando o método proposto em [Wang 1994] , descrito na Seção 3. A Tabela 4.1 mostra o número de estados dos modelos.

Tabela 4.1 - Número de estados dos modelos

Classe/Meta- Classe	Modelo baseado em colunas	Modelo baseado em linhas
0	14	14
1	5	14
2	12	15
3	13	17
4	11	14
5	11	16
6	12	15
7	12	15
8	10	16
9	14	16
$M_{0...9}$	10	14
M_P	4	5
M_C	3	4
M_{PC}	2	4
M_H	5	4
M_{CEP}	12	12

Treinamento e avaliação dos modelos

Para o treinamento dos modelos, as seqüências de observações são enviadas ao algoritmo *Baum-Welch* e o processo de treinamento é feito utilizando validação cruzada.

Para a avaliação dos modelos criados, as seqüências de observações são enviadas ao algoritmo *Viterbi*, e este indica qual o modelo tem a maior probabilidade de representar a imagem de entrada.

4.4. Segmentação-reconhecimento das linhas do bloco do endereço e verificação dos resultados

A segmentação-reconhecimento das linhas do bloco do endereço e o processo de verificação são baseados em dois estágios: segmentação-reconhecimento e verificação dos resultados.

4.4.1. Segmentação-reconhecimento das linhas

O objetivo desta etapa é encontrar as melhores hipóteses de segmentação-reconhecimento utilizando uma estratégia de segmentação implícita, onde não existe uma

segmentação a priori da linha de texto e a segmentação e o reconhecimento dos componentes da linha são feitos em um único processo. Esta abordagem tem apresentado bons resultados nos casos onde podem ser encontrados dígitos fragmentados, sobrepostos ou conectados, o que ocorre com freqüência no caso de envelopes postais [Britto 2001].

Para cada hipótese de segmentação-reconhecimento são fornecidas as seguintes informações: o número de componentes da linha, os pontos de segmentação, o resultado de reconhecimento de cada segmento e a probabilidade global de reconhecimento da linha. Este processo é composto das seguintes etapas:

- Extração de primitivas de *Foreground* para cada coluna da imagem.
- Conversão dos vetores de primitivas em uma seqüência de observações.
- Os modelos MEM treinados com manuscritos isolados e descritos na seção anterior são utilizados pelo algoritmo *Level Buiding* para encontrar os pontos de segmentação e a melhor seqüência de modelos que representa a imagem de entrada, através de uma concatenação dinâmica dos modelos MEM. Um exemplo do funcionamento do algoritmo *Level Building* para o exemplo da Figura 4.11 pode ser observado na Figura 4.12. Os níveis representam os componentes da linha de texto. No primeiro nível foi encontrada a palavra CEP, nos níveis 2 a 6 foram encontrados os 5 primeiros dígitos do CEP, no nível 7 foi encontrado o separador e nos níveis 8 a 10 foram encontrados os 3 últimos dígitos do CEP.

O objetivo do *Level Building* é determinar a seqüência ótima de MEMs. No primeiro nível os possíveis candidatos para o primeiro componente da linha são encontrados, no segundo nível os possíveis candidatos para o segundo nível são encontrados e assim sucessivamente. Este processo é repetido com um número de níveis correspondente ao número máximo de componentes esperados na linha (L), a ser definido experimentalmente. O uso de Meta-Classes diminui o número de MEMs a serem analisados por nível, como no exemplo da Figura 4.12, onde sem a Meta-Classe $M_{0...9}$ existiriam mais 9 modelos a serem analisados.

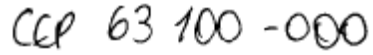

Figura 4.11 – Exemplo de linha extraída do bloco do endereço

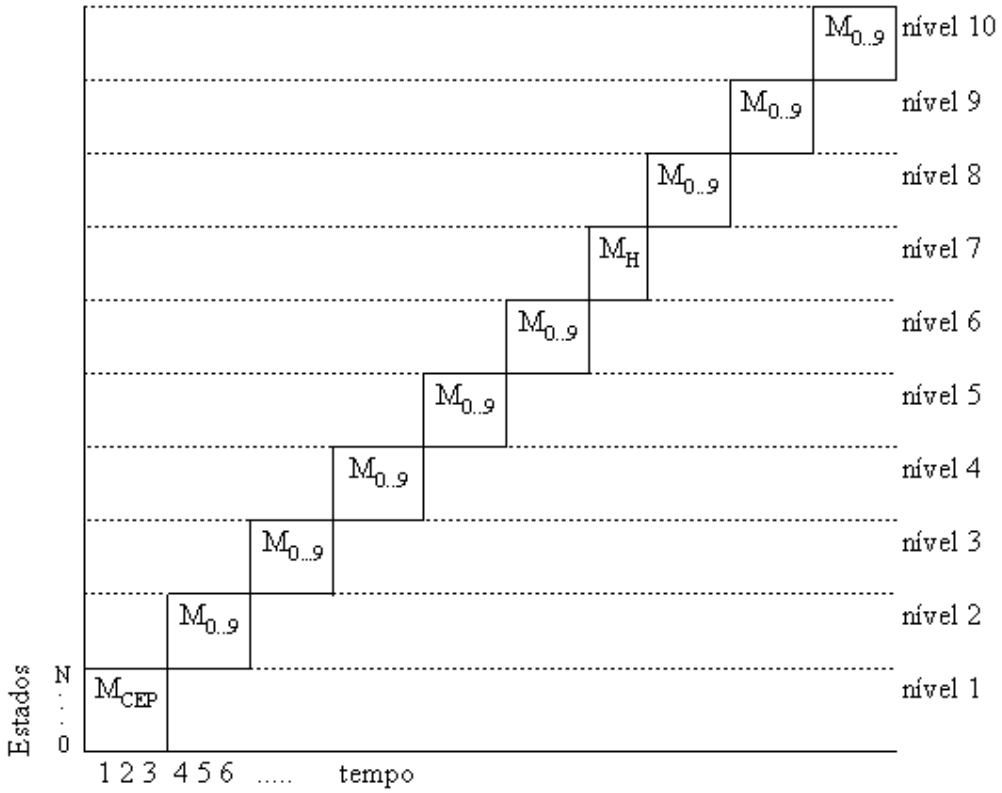

Figura 4.12 - Estrutura *trellis* para o exemplo da Figura 4.11.

4.4.2. Verificação dos resultados

O objetivo é a verificação e reclassificação das hipóteses de segmentação-reconhecimento obtidos na etapa anterior. Este processo é composto das seguintes etapas:

- Extração de primitivas de *Foreground* e *Background* para cada coluna e para cada linha dos segmentos encontrados no processo de segmentação-reconhecimento.
- Os vetores de primitivas resultantes são convertidos em duas seqüências de observações: uma baseada nas colunas e outra baseada nas linhas do segmento.
- Verificação e re-classificação dos resultados.

A verificação dos resultados é feita utilizado-se o algoritmo *Viterbi*, descrito no capítulo 3 e modelos MEM treinados com manuscritos isolados.

Cada Meta-Classe é representada por dois modelos MEM: um baseado nas colunas da imagem e outro baseado nas linhas da imagem. Os resultados obtidos no processo de segmentação-reconhecimento são utilizados para selecionar os modelos de coluna e linha correspondentes. O par de modelos selecionados é avaliado utilizando-se o algoritmo *Viterbi* e

as seqüências de observações extraídas. A saída é a probabilidade dos modelos de colunas e linhas combinados pela soma dos seus *logs*. O resultado corresponde à probabilidade do segmento estar correto. A probabilidade de toda a linha é obtida pela soma das probabilidades de cada segmento da linha. A probabilidade calculada é adicionada ao resultado obtido no estágio de segmentação-reconhecimento. O resultado final é utilizado para re-classificar as hipóteses de segmentação-reconhecimento.

Os modelos MEM utilizados no estágio de verificação têm a mesma topologia e o mesmo processo de criação dos modelos descritos na seção 4.3. A única diferença é que foram adicionadas as primitivas de *Background* ao conjunto de primitivas de *Foreground* e que foram extraídas primitivas das colunas e das linhas da imagem.

4.5. Localização do CEP

Uma vez localizados os pontos de segmentação das linhas e identificados os seus componentes, a localização do CEP consiste na localização de uma seqüência de componentes que melhor representa a estrutura do CEP, uma seqüência de 5 ou 8 dígitos separados ou não pelo traço, podendo ser precedidos da palavra CEP. Como neste momento já se conhecem os pontos de segmentação dos componentes, já sabemos onde o CEP começa e onde ele termina na linha de texto.

4.6. Considerações Finais

Este capítulo descreveu o método proposto para a localização do CEP em envelopes postais e o processo de criação dos modelos MEM utilizados no método. Primeiramente foi dada uma visão geral do método e na seqüência foram descritas as etapas que o compõem.

A seção 4.2. descreveu o processo de segmentação do bloco do endereço em linhas, que consiste na localização das linhas no bloco e da correção da linha de base da escrita.

Na Seção 4.3. foi descrito o processo de criação dos modelos MEM utilizados nas demais etapas do método. Foram descritas as primitivas extraídas da imagem (*Foreground/Background Features*), as Meta-Classes criadas, a topologia dos modelos e os algoritmos para treinamento e avaliação dos modelos.

As etapas de segmentação-reconhecimento das linhas e verificação dos resultados obtidos foram descritas na Seção 4.4. Finalmente, na Seção 4.5 foi descrito o processo de localização do CEP, que consiste na busca de uma seqüência de 5 ou 8 dígitos, separados ou não pelo traço e podendo ser precedidos da palavra CEP.

No próximo capítulo serão descritos os experimentos realizados e os resultados obtidos na criação dos Modelos MEM utilizados na segmentação-reconhecimento da linha e na verificação dos resultados.

5. Resultados Experimentais

Este capítulo descreve os experimentos realizados durante a criação dos modelos MEM que serão utilizados para a segmentação-reconhecimento das linhas do bloco do endereço e verificação dos resultados. Na Seção 5.1 é apresentada uma descrição da base de dados utilizada para treinamento, validação e teste. A Seção 5.2 descreve os experimentos realizados na criação dos modelos MEM para segmentação e reconhecimento da linha, enquanto a seção 5.3 descreve os experimentos realizados durante a criação dos modelos para verificação dos resultados. A discussão dos resultados obtidos é apresentada na Seção 5.4.

5.1. Base de Dados

Para treinamento, validação e teste foi utilizada uma base com 12758, 3332 e 2784 imagens respectivamente. Esta base é composta de imagens de dígitos isolados, caracteres isolados, palavras, hífen e a palavra CEP, extraídas de imagens de envelopes postais e de cheques bancários brasileiros.

A base de dados dos Correios é composta de aproximadamente 50.000 imagens de envelopes reais com endereços impressos e manuscritos. Destas imagens, somente aproximadamente 900 exemplos são de envelopes manuscritos. Alguns exemplos podem ser observados na Figura 5.1.

A partir dos exemplos de envelopes manuscritos, foi criada uma base contendo imagens de dígitos isolados, palavras, caracteres, hífen e a palavra CEP. A criação desta base foi feita de forma semi-automática, utilizando um software de rotulação pertencente ao Laboratório de Análise e Reconhecimento de Imagens de Documentos (LARDOC) da PUC-PR, gerando uma imagem binária em formato *Tiff*. A remoção de ruídos presentes na imagem

do envelope ou provenientes da etapa de conversão da imagem para binário foi feita manualmente. Não foi executado nenhum processo de normalização do tamanho da imagem ou correção de inclinação. Alguns exemplos de imagens pertencentes à base de dados criada podem ser observados nas Figuras 5.2 e 5.3.

Figura 5.1 - Exemplos de envelopes postais manuscritos.

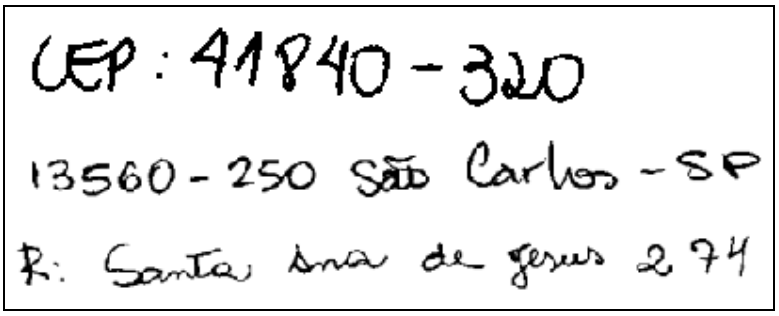

Figura 5.2 - Exemplos de linhas do bloco do endereço.

Figura 5.3 - Exemplos da base de dados criada.

Como o número de exemplos de envelopes manuscritos é pequeno, foram incluídos nos conjuntos de treinamento e validação, exemplos de dígitos isolados provenientes de cheques bancários brasileiros. A descrição da base de cheques pode ser encontrada em [Freitas, et al. 2000].

A Tabela 5.1 apresenta a distribuição das classes e o número de exemplos usados para treinamento, validação e teste, enquanto a Tabela 5.2 apresenta o número de exemplos disponíveis de cada caractere. A classe caracteres contém imagens de caracteres, barra, vírgula, aspas, pontos, etc.

Tabela 5.1 - Distribuição das Classes.

Classe / Meta-Classe	Treinamento			Validação			Teste		
	Base Cheque	Base Envelope	Total	Base Cheque	Base Envelope	Total	Base Cheque	Base Envelope	Total
0	900	100	1000	150	100	250	-	200	200
1	900	100	1000	150	100	250	-	200	200
2	900	100	1000	150	100	250	-	200	200
3	900	100	1000	150	100	250	-	200	200
4	900	100	1000	150	100	250	-	200	200
5	900	100	1000	150	100	250	-	200	200
6	900	100	1000	150	100	250	-	200	200
7	900	100	1000	150	100	250	-	200	200
8	900	100	1000	150	100	250	-	200	200
9	900	100	1000	150	100	250	-	200	200
Palavras	-	1000	1000	-	250	250	-	200	200
Caracteres	-	1240	1240	-	412	412	-	412	412
Hífen	-	380	380	-	126	126	-	126	126
Palavra Cep	-	138	138	-	44	44	-	46	46
Total	12758			3332			2784		

Tabela 5.2 - Número de exemplo de caracteres. A Classe outros agrupa “,”, “/”, “.”, “:”, etc.

Caractere	Nº Exemplos	Caractere	Nº Exemplos	Caractere	Nº Exemplos
A	208	J	28	S	120
B	47	K	3	T	24
C	300	L	42	U	21
D	64	M	49	V	28
E	300	N	59	W	8
F	19	O	65	X	7
G	26	P	300	Y	6
H	18	Q	3	Z	6
I	26	R	113	Outros	174

5.2. Experimentos realizados durante a criação dos modelos MEM para segmentação-reconhecimento das linhas

Os primeiros experimentos buscam o treinamento e avaliação dos modelos a serem utilizados no primeiro estágio do método (segmentação-reconhecimento das linhas do bloco do endereço) e para tal são baseados somente nas colunas da imagem e em primitivas de *Foreground*, descritas na Seção 4.3.

5.2.1. Experimento com dígitos isolados manuscritos

O objetivo deste experimento foi verificar a eficiência dos modelos na distinção entre os dígitos de 0 a 9 e fazer um comparativo com os resultados obtidos em [Britto 2003]. Para treinamento, validação e teste foram utilizados 1000, 250 e 200 exemplos por classe respectivamente. A Tabela 5.3 apresenta a matriz de confusão e os resultados obtidos.

Tabela 5.3 - Matriz de confusão - modelo baseado em colunas

Classe	0	1	2	3	4	5	6	7	8	9	Taxa Rec.
0	**180**	0	6	1	6	0	1	0	6	0	90,00
1	0	**159**	8	3	16	1	0	9	1	3	79,50
2	0	4	**169**	4	5	3	1	4	6	4	84,50
3	0	0	23	**144**	3	8	2	5	9	6	72,00
4	0	5	2	0	**188**	0	0	4	0	1	94,00
5	0	6	3	4	11	**152**	17	0	2	5	76,00
6	0	1	3	4	4	14	**159**	0	14	0	79,50
7	0	7	8	0	31	0	0	**148**	4	2	74,00
8	0	1	7	0	6	1	5	2	**166**	12	83,00
9	0	1	9	1	17	6	1	5	5	**155**	77,50
										Total	81,00

Em [Britto 2003], foi obtida uma taxa de reconhecimento de 94,00%, utilizando MEM e o mesmo conjunto de primitivas. Os experimentos foram realizados com 50.000, 10.000 e 10.000 exemplos para treinamento, validação e teste respectivamente, extraídos da base NIST SD19 [Grother, 1995]. A base NIST SD19 é mais bem comportada que a base de envelopes postais, pois foi criada a partir de *strings* extraídas de páginas de formulários, onde existia um local apropriado para o preenchimento das informações. O mesmo não ocorre na base de envelopes postais, onde não há um local reservado para o preenchimento das informações e estas podem ser preenchidas em qualquer posição, com qualquer inclinação, sobrepostas ou misturadas. Esta diferença pode ter feito com que a taxa de reconhecimento obtida seja menor que a obtida em [Britto 2003].

Outro fato que pode ter influenciado nas taxas de reconhecimento é que alguns dígitos, como o 1 e o 2, são escritos de forma muita parecida, podendo ser confundidos, como pode ser observado na Figura 5.4. Este problema pode ser amenizado agrupando os dígitos em uma única Meta-Classe.

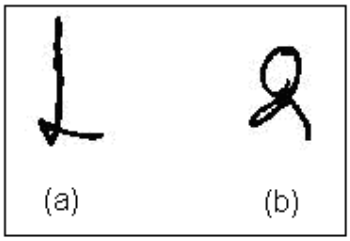

Figura 5.4 - Confusões treinamento. (a) Exemplo do nº 1 classificado como 2. (b) Exemplo do nº 2 classificado como 3.

5.2.2. Experimentos com as classes que compõem o bloco do endereço

O objetivo destes experimentos foi verificar qual a configuração de Meta-Classes tem os melhores resultados na representação dos componentes do bloco do endereço.

Experimento 1

Neste experimento foram avaliadas as seguintes Classes/Meta-Classes:

- Os dígitos de 0 a 9;
- Palavras: $M_{P;}$
- Caracteres: $M_{C;}$
- Hífen: $M_{H;}$

Os resultados obtidos podem ser observados na Tabela 5.4. Como pode ser observado, a taxa de reconhecimento dos dígitos diminuiu em relação ao experimento anterior, havendo confusão com a Meta-Classe Caracteres. Isto pode ser explicado porque muitas letras e dígitos são escritos de forma parecida, podendo facilmente ser confundidos. Outro motivo, é que o número de exemplos de caracteres é pequeno (2064 exemplos), podendo não ser representativo de todas as variações entre as letras. Na Figura 5.5 são apresentados alguns exemplos de caracteres classificados como dígitos, na Figura 5.6 são apresentados alguns exemplos de dígitos classificados como a Meta-Classe caracteres enquanto a Figura 5.7 apresenta alguns exemplos classificados corretamente.

Tabela 5.4 - Matriz de confusão - modelos baseados em colunas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_P	M_H	M_C	Taxa Rec.
0	**186**	0	2	0	0	0	0	0	2	0	0	0	10	93,00
1	0	**142**	3	4	13	1	0	9	1	1	4	1	21	71,00
2	0	2	**132**	0	0	4	0	0	6	2	7	0	47	66,00
3	0	0	11	**146**	0	11	0	2	8	2	2	0	18	73,00
4	0	4	2	0	**160**	1	0	4	0	0	19	0	10	80,00
5	0	1	1	5	2	**162**	3	0	1	2	2	0	21	81,00
6	0	0	2	2	0	14	**142**	0	7	0	6	0	27	71,00
7	0	4	1	1	14	2	0	**136**	3	1	6	0	32	68,00
8	1	0	2	2	0	2	3	1	**154**	7	1	0	27	77,00
9	1	0	1	3	11	5	0	8	2	**147**	0	0	22	73,50
M_P	0	0	2	1	2	0	0	0	0	0	**175**	0	20	87,50
M_H	0	3	0	0	1	0	0	0	0	0	0	**121**	1	96,03
M_C	14	30	20	10	14	47	6	11	21	10	37	1	**191**	46,36
													Total	72,83

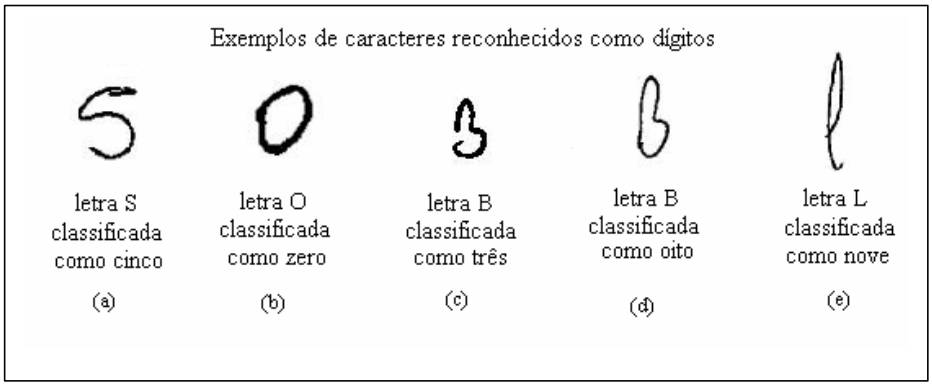

Figura 5.5 – Exemplos de caracteres classificados como dígitos

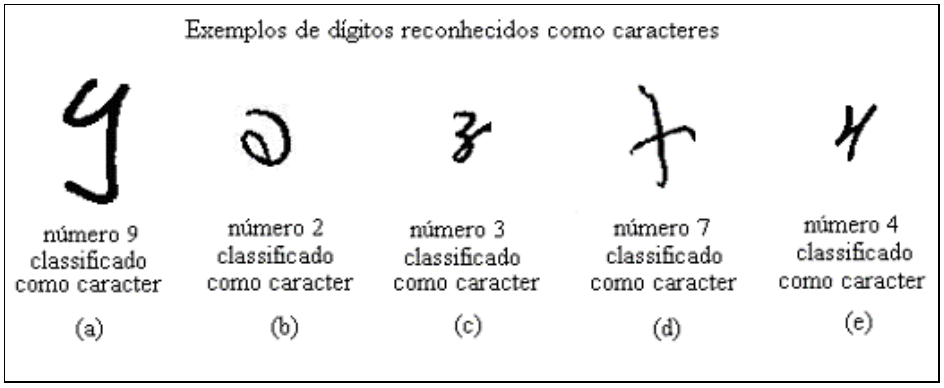

Figura 5.6 – Exemplos de dígitos classificados como caracteres

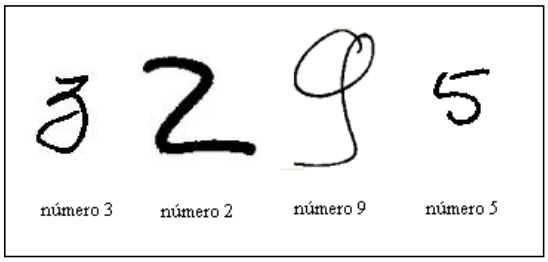

Figura 5.7 - Exemplos classificados corretamente.

Experimento 2

Neste experimento foi incluída a Meta-Classe CEP (M_{CEP}). Em 66% dos casos os dígitos do CEP vem precedidos da palavra CEP e conseqüentemente a sua localização no bloco do endereço poderá auxiliar na localização dos dígitos do CEP e ser um critério de seleção em casos em que existam mais de uma seqüência de dígitos que representem a estrutura do CEP.

Os resultados obtidos são apresentados na Tabela 5.5. Conforme se observa, embora a taxa total de reconhecimento não tenha aumentado significativamente, em 65,22 % dos casos a palavra CEP foi identificada corretamente. Além disso, houve poucos casos em que outras classes foram classificadas incorretamente como a palavra CEP. A Figura 5.8 apresenta alguns exemplos de erros e de acertos. Na Figura 5.8(a) temos um exemplo da palavra CEP classificado como a Meta-Classe palavra, na Figura 5.8(b) temos um exemplo de caractere classificado como a Meta-Classe da palavra CEP enquanto na Figura 5.8(c) temos um exemplo da palavra CEP classificado corretamente.

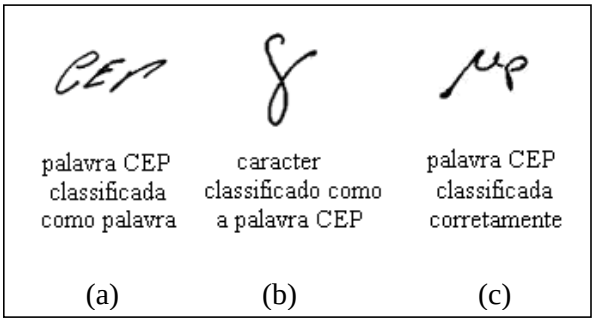

Figura 5.8 - Exemplos de erros e acertos.

Tabela 5.5 - Matriz de confusão - modelos baseados em colunas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_P	M_H	M_C	M_{CEP}	Taxa Rec.
0	**175**	0	0	0	0	0	0	0	4	0	0	0	21	0	87,50
1	0	**147**	6	8	11	0	0	8	1	1	4	0	14	0	73,50
2	0	2	**162**	1	0	2	0	3	6	2	4	0	18	0	81,00
3	0	0	7	**162**	0	3	1	5	4	1	2	0	15	0	81,00
4	1	4	2	0	**163**	0	0	4	0	0	13	0	12	1	81,50
5	0	2	1	7	2	**152**	4	0	1	1	0	0	30	0	76,00
6	0	0	1	2	2	11	**139**	0	10	0	7	0	28	0	69,50
7	0	2	2	0	9	1	0	**155**	0	2	2	0	27	0	77,50
8	1	0	6	1	1	1	1	2	**157**	7	1	0	22	0	78,50
9	1	0	1	9	12	3	0	9	11	**125**	0	0	29	0	62,50
M_P	0	0	3	0	2	0	0	0	0	0	**178**	0	16	1	89,00
M_H	0	4	0	0	0	0	0	0	0	0	0	**121**	1	0	96,03
M_C	13	35	21	3	6	26	10	13	24	5	35	2	**217**	2	52,67
M_{CEP}	0	0	0	0	0	0	0	0	0	0	15	0	1	**30**	65,22
														Total	74,82

Experimento 3

Procurando diminuir a confusão entre dígitos e letras, foi feito um experimento agrupando palavras e caracteres em uma única Meta-Classe. Os resultados obtidos podem ser observados na Tabela 5.6. Conforme se observa houve uma baixa nos resultados.

Um comparativo entre os três experimentos pode ser observado na Tabela 5.7. Os melhores resultados foram obtidos no experimento 2, com as Classes/Meta-Classes: 0, 1, 2, 3, 4, 5, 6, 7,8, 9, M_P, M_C, M_H e M_{CEP}.

Tabela 5.6 - Matriz de confusão - modelos baseados em colunas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_{PC}	M_H	M_{CEP}	Taxa Rec.
0	**181**	0	2	0	2	1	0	0	4	0	10	0	0	90,50
1	0	**152**	6	2	13	1	1	9	1	2	11	1	1	76,00
2	1	2	**130**	2	0	7	0	5	8	3	42	0	0	65,00
3	0	0	7	**138**	1	14	0	10	6	6	18	0	0	69,00
4	1	5	1	0	**161**	0	0	6	0	1	22	0	3	80,50
5	0	4	0	10	4	**153**	7	3	2	4	13	0	0	76,50
6	0	0	0	2	3	17	**130**	0	14	0	34	0	0	65,00
7	0	4	1	0	10	1	0	**169**	0	2	12	0	1	84,50
8	1	0	0	1	1	3	3	6	**151**	10	24	0	0	75,50
9	1	0	1	4	16	4	0	8	3	**151**	11	0	1	75,50
M_{PC}	14	47	20	6	34	35	28	27	23	7	**365**	2	4	59,64
M_H	0	2	0	0	3	0	0	0	0	0	1	**120**	0	95,24
M_{CEP}	0	0	0	0	3	0	0	0	0	0	16	0	**27**	58,70
													Total	72,84

Tabela 5.7 - Comparativo entre os três experimentos.

Classe / Meta-Classe	Dígitos / M_P / M_C / M_H	Dígitos/ M_P / M_C / M_H / M_{CEP}	Dígitos/ M_{PC} / M_H / M_{CEP}
0	93,00	87,50	90,50
1	71,00	73,50	76,00
2	66,00	81,00	65,00
3	73,00	81,00	69,00
4	80,00	81,50	80,50
5	81,00	76,00	76,50
6	71,00	69,50	65,00
7	68,00	77,50	84,50
8	77,00	78,50	75,50
9	73,50	62,50	75,50
M_P	87,50	89,00	-
M_{PC}	-	-	59,64
M_C	46,36	52,67	-
M_H	96,03	96,03	95,24
M_{CEP}	-	65,22	58,70
Total	72,83	74,82	72,84

5.2.1. Experimentos agrupando os dígitos em uma única Meta-Classe

O objetivo deste experimento foi avaliar a eficiência do método agrupando os dígitos em uma única Meta-Classe, visando aumentar as taxas de reconhecimento, visto que existem confusões entre algumas classes de dígitos.

Experimento 4

Foram refeitos os experimentos descritos anteriormente. Os resultados obtidos, quando refeitos os experimentos 1, 2 e 3, podem ser observados nas Tabelas 5.8, 5.9 e 5.10, respectivamente. Um comparativo entre os experimentos pode ser observado na Tabela 5.11, incluindo os experimentos da seção anterior.

Tabela 5.8 - Matriz de confusão - experimento 1 refeito agrupando dígitos em uma única Meta-Classe.

Meta-Classe	$M_{0..9}$	M_P	M_H	M_C	Taxa Rec.
$M_{0..9}$	**1691**	62	15	232	84,55
M_P	10	**175**	0	15	87,50
M_H	0	0	**121**	5	96,03
M_C	161	40	8	**203**	49,27
				Total	79,99

Tabela 5.9 - Matriz de confusão - experimento 2 refeito agrupando os dígitos em um única Meta-Classe.

Meta-Classe	$M_{0...9}$	M_P	M_C	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	**1747**	77	156	3	17	87,35
M_P	14	**168**	17	0	1	84,00
M_C	229	43	**130**	5	5	31,55
M_H	5	1	2	**117**	1	92,86
M_{CEP}	0	12	4	0	**30**	65,22
					Total	78,74

Tabela 5.10 - Matriz de confusão, experimento 3 refeito agrupando os dígitos em um única Meta-Classe.

Meta-Classe	$M_{0...9}$	M_{PC}	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	**1826**	127	9	38	91,30
M_{PC}	328	**270**	7	7	44,12
M_H	3	0	**121**	2	96,03
M_{CEP}	0	18	0	**28**	60,87
				Total	80,64

Tabela 5.11 - Comparativo entre os experimentos

Meta-Classe	Dígitos / M_P/ M_C / M_H	$M_{0...9}$ / M_P / M_C / M_H	Dígitos/ M_P / M_C / M_H / M_{CEP}	$M_{0...9}$ / M_P / M_C / M_H / M_{CEP}	Dígitos/ M_{PC} / M_H / M_{CEP}	$M_{0...9}$ / M_{PC} / M_H / M_{CEP}
$M_{0..9}$	75,35*	84,55	76,85*	87,35	75,80*	91,30
M_P	87,50	87,50	89,00	84,00	-	-
M_{PC}	-	-	-	-	59,64	44,12
M_C	46,36	49,27	52,67	31,55	-	-
M_H	96,03	96,03	96,03	92,86	95,24	96,03
M_{CEP}	-	-	65,22	65,22	58,70	60,87
Total	72,83	79,99	74,82	78,74	72,84	80,64

* O cálculo da taxa de reconhecimento das classes de 0 a 9 foi feita dividindo o número de dígitos classificados corretamente pelo número de exemplos de dígitos disponíveis para teste.

Conforme pode ser observado na Tabela 5.11, houve uma melhora nos resultados agrupando os dígitos em uma única Meta-Classe, havendo uma melhora significativa nas taxas de reconhecimento dos dígitos. Além disso, os melhores resultados foram obtidos agrupando palavras e caracteres em uma única Meta-Classe, diferente do observado nos experimentos anteriores.

De acordo com os resultados apresentados nas seções anteriores, os melhores resultados para os modelos da etapa de segmentação-reconhecimento foram obtidos com as seguintes Meta-Classes: $M_{0...9}$, M_{PC}, M_H e M_{CEP}. Neste experimento os exemplos de caracteres classificados incorretamente na Figuras 5.5 e 5.6 (experimento 1) foram classificados corretamente. A Figura 5.9 apresenta alguns exemplos classificados incorretamente (nas Figuras 5.9(a) a 5.9(d) temos alguns exemplos de caracteres classificados como dígitos, na Figura 5.9(e) temos um exemplo da palavra “Av” classificado como dígito e nas Figuras 5.9(f) e 5.9(g) temos exemplos de caracteres classificados como a Meta-Classe hífen).

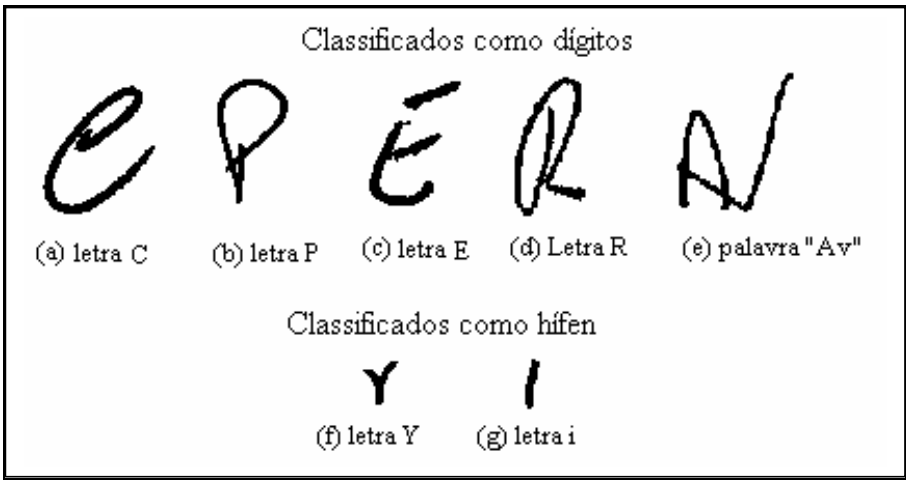

Figura 5.9 - Exemplos classificados incorretamente. (a), (b), (c), (d) - exemplos de caracteres classificados como dígitos. (e) - palavra "Av" classificada como dígito. (f) letra "y" classificada como hífen. (g) - letra "i" classificada como hífen.

A taxa de reconhecimento dos caracteres ainda continua baixa, existindo muitos casos em que os caracteres são classificados incorretamente como dígitos. Estas confusões poderão ser evitadas nos próximos experimentos, onde são avaliados os modelos que farão parte da etapa de verificação dos resultados de segmentação-reconhecimento. Espera-se que haja uma melhora nos resultados, pois será adicionado um novo conjunto de primitivas e serão extraídas também primitivas baseadas nas linhas da imagem.

5.3. Experimentos realizados durante a criação dos modelos MEM para verificação dos resultados

Nestes experimentos foram avaliadas informações complementares àquelas utilizadas nos experimentos anteriores. Ao conjunto de primitivas de *Foreground* foram adicionas primitivas baseadas em concavidades (*Background Features*, descritas na Seção 4.3). Além dos modelos baseados nas colunas da imagem, foram avaliados os modelos baseados nas linhas da imagem e a combinação destes dois modelos. Primeiramente serão descritos os experimentos incluindo os modelos baseados nas linhas e a combinação dos modelos de linhas e colunas. Na seqüência serão descritos os experimentos incluindo as primitivas de *Background.*

5.3.1. Experimentos incluindo os modelos baseados nas linhas da imagem

Nestes experimentos foram refeitos os testes descritos nas seções anteriores, incluindo os modelos baseados nas linhas da imagem.

Experimento com dígitos isolados manuscritos

Neste experimento foram avaliados somente os dígitos de 0 a 9. Os resultados obtidos podem ser observados nas Tabelas 5.12 e 5.13. Conforme se observa, obteve-se uma melhora nas taxas de reconhecimento em relação ao experimento da Seção 5.2.1, onde foi obtida uma taxa de reconhecimento de 81,00 %.

Tabela 5.12 - Matriz de confusão - modelos baseados em linhas

Classe	0	1	2	3	4	5	6	7	8	9	Taxa Rec.
0	**186**	0	4	1	0	0	2	0	7	0	93,00
1	0	**147**	3	1	6	1	0	38	4	0	73,50
2	1	4	**161**	0	2	2	1	11	15	3	80,50
3	0	10	4	**161**	1	5	0	8	9	2	80,50
4	1	3	0	0	**171**	0	0	22	0	3	85,50
5	1	13	0	0	4	**154**	1	11	15	1	77,00
6	0	1	0	1	0	3	**185**	1	9	0	92,50
7	0	6	0	0	4	0	0	**185**	3	2	92,50
8	5	2	0	1	1	0	1	2	**186**	2	93,00
9	2	1	1	0	18	2	0	13	6	**157**	78,50
										Total	84,65

Tabela 5.13 - Matriz de confusão - combinando os modelos baseados em colunas e linhas

Classe	0	1	2	3	4	5	6	7	8	9	Taxa Rec.
0	**184**	0	4	0	3	0	1	0	8	0	92,00
1	0	**173**	7	1	7	0	0	8	3	1	86,50
2	0	4	**180**	0	2	1	0	3	7	3	90,00
3	0	1	6	**167**	0	4	0	8	13	1	83,50
4	0	2	1	0	**187**	0	0	9	1	0	93,50
5	0	7	0	1	3	**176**	2	2	8	1	88,00
6	0	0	1	0	3	3	**182**	0	11	0	91,00
7	0	3	1	0	13	0	0	**180**	1	2	90,00
8	0	1	0	1	0	0	2	4	**190**	2	95,00
9	1	1	1	0	16	2	0	7	6	**166**	83,00
										Total	89,25

Experimentos com as classes que compõem o bloco do endereço

Nestes experimentos foram refeitos os experimentos realizados para determinar as Meta-Classes com melhor resultado na representação dos componentes do bloco do endereço.

Experimento 5

Neste experimento foram avaliadas as seguintes Classes/Meta-Classes:

- Os dígitos de 0 a 9;
- Palavras: $M_{P;}$
- Caracteres: $M_{C;}$
- Hífen: $M_{H;}$

Os resultados obtidos podem ser observados nas Tabelas 5.14 e 5.15. Neste caso também houve uma melhora nos resultados em relação aos modelos baseados somente nas colunas da imagem.

Tabela 5.14 - Matriz de confusão - modelos baseados em linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_P	M_H	M_C	Taxa Rec.
0	**176**	0	6	0	0	0	0	0	5	0	0	0	13	88,00
1	0	**123**	1	0	8	0	0	24	3	0	6	0	35	61,50
2	0	1	**129**	0	0	0	0	2	3	3	4	0	58	64,50
3	0	2	1	**123**	0	2	0	0	3	2	1	0	66	61,50
4	0	2	0	0	**156**	0	0	5	0	6	5	0	26	78,00
5	0	3	0	0	0	**129**	3	0	1	0	6	0	58	64,50
6	0	0	2	0	0	0	**155**	0	11	0	4	0	28	77,50
7	0	2	0	0	6	0	0	**155**	0	3	8	0	26	77,50
8	2	0	2	0	0	1	0	1	**163**	1	3	0	27	81,50
9	0	1	0	0	14	0	0	3	4	**146**	1	0	31	73,00
M_P	0	0	3	0	0	0	0	0	0	0	**185**	0	12	92,50
M_H	0	0	0	0	0	0	0	0	0	0	0	**121**	5	96,03
M_C	10	11	2	0	7	18	9	11	18	3	25	1	**297**	72,09
													Total	75,16

Tabela 5.15 - Matriz de confusão, combinando os modelos de colunas e linhas

Classe/Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_P	M_H	M_C	Taxa Rec.
0	**178**	0	4	0	0	0	0	0	4	0	0	0	14	89,00
1	0	**137**	2	0	8	0	0	5	1	0	5	0	42	68,50
2	0	0	**120**	0	0	0	0	0	1	2	4	0	73	60,00
3	0	1	0	**142**	0	3	0	1	8	1	2	0	42	71,00
4	0	2	0	0	**162**	0	0	2	0	0	12	0	22	81,00
5	0	2	0	0	1	**139**	2	0	2	0	2	0	52	69,50
6	0	0	0	0	0	2	**142**	0	14	0	7	0	35	71,00
7	0	0	0	0	4	0	0	**141**	0	1	7	0	47	70,50
8	0	0	1	0	0	0	0	0	**162**	0	1	0	36	81,00
9	0	0	0	0	13	2	0	4	1	**145**	0	0	35	72,50
M_P	0	0	1	0	0	0	0	0	0	0	**189**	0	10	94,50
M_H	0	0	0	0	0	0	0	0	0	0	0	**118**	8	93,65
M_C	11	13	1	1	4	24	4	6	13	1	27	0	**307**	74,51
													Total	76,04

Experimento 6

Neste experimento foi incluída a Meta-Classe CEP (M_{CEP}). Os resultados obtidos são apresentados nas Tabelas 5.16 e 5.17. Neste caso, embora tenha havido uma queda na taxa de reconhecimento da palavra CEP, com relação ao experimento 2, houve uma melhora na taxa total de reconhecimento.

Tabela 5.16 - Matriz de confusão - modelos baseados em linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_P	M_H	M_C	M_{CEP}	Taxa Rec.
0	**180**	0	3	0	0	0	1	0	6	1	0	0	9	0	90,00
1	0	**116**	1	0	6	1	0	28	0	0	4	0	41	3	58,00
2	0	1	**131**	0	0	1	0	5	4	1	3	0	53	1	65,50
3	0	1	3	**129**	1	6	0	4	3	1	1	0	51	0	64,50
4	0	1	0	0	**164**	0	0	11	0	4	4	0	16	0	82,00
5	0	0	0	0	0	**138**	2	1	1	1	8	0	49	0	69,00
6	0	0	2	0	2	3	**145**	0	11	0	2	0	35	0	72,50
7	0	6	0	0	2	0	0	**171**	0	1	2	0	17	1	85,50
8	1	1	1	0	0	0	0	0	**166**	0	2	0	28	1	83,00
9	0	1	0	0	9	2	0	8	5	**141**	1	0	33	0	70,50
M_P	0	0	3	0	0	0	0	0	0	0	**185**	0	12	0	92,50
M_H	0	0	0	0	0	0	0	0	0	0	0	**123**	1	2	97,62
M_C	10	16	6	0	7	12	8	10	17	0	26	0	**298**	2	72,33
M_{CEP}	0	0	0	0	0	0	0	0	1	0	6	0	13	**26**	56,52
														Total	75,90

Tabela 5.17 - Matriz de confusão - combinando os modelos de colunas e linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_P	M_H	M_C	M_{CEP}	Taxa Rec.
0	**177**	0	4	0	0	0	0	0	4	0	0	0	15	0	88,50
1	0	**140**	1	0	3	1	0	9	1	0	6	0	39	0	70,00
2	0	0	**138**	0	0	1	0	1	4	2	3	0	51	0	69,00
3	0	0	2	**147**	0	3	0	5	3	1	3	0	36	0	73,50
4	0	0	0	0	**156**	0	0	5	0	0	8	0	31	0	78,00
5	0	0	0	0	0	**130**	2	0	2	1	0	0	65	0	65,00
6	0	0	1	0	0	3	**141**	0	10	0	6	0	39	0	70,50
7	0	3	0	0	1	0	0	**162**	0	1	2	0	29	2	81,00
8	0	1	1	0	0	0	0	1	**163**	0	1	0	33	0	81,50
9	0	1	0	0	9	2	0	6	5	**129**	0	0	48	0	64,50
M_P	0	0	1	0	0	0	0	0	0	0	**193**	0	6	0	96,50
M_H	0	0	0	0	0	0	0	0	0	0	0	**119**	7	0	94,44
M_C	9	17	4	0	4	12	4	10	12	0	28	1	**311**	0	75,49
M_{CEP}	0	0	0	0	0	0	0	0	0	0	13	0	5	**28**	60,87
														Total	76,65

Experimento 7

Neste experimento foram agrupadas palavras e caracteres em uma única Meta-Classe. Os resultados obtidos podem ser observados nas Tabelas 5.18 e 5.19. Neste caso também houve uma melhora nos resultados.

Tabela 5.18 - Matriz de confusão - modelos baseados em linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_{PC}	M_H	M_{CEP}	Taxa Rec.
0	**180**	0	2	0	1	0	0	0	6	0	11	0	0	90,00
1	0	**121**	1	0	9	1	0	35	3	0	30	0	0	60,50
2	0	0	**134**	0	0	0	0	9	7	1	49	0	0	67,00
3	0	1	1	**141**	1	1	0	8	6	1	40	0	0	70,50
4	0	1	0	0	**174**	0	0	11	0	3	11	0	0	87,00
5	0	0	0	1	1	**121**	4	1	0	1	71	0	0	60,50
6	0	1	1	0	2	3	**157**	0	11	0	25	0	0	78,50
7	0	4	0	0	6	0	0	**176**	0	0	14	0	0	88,00
8	2	0	0	0	1	0	0	0	**173**	0	24	0	0	86,50
9	0	1	0	0	17	0	0	11	7	**141**	23	0	0	70,50
M_{PC}	11	25	24	2	17	11	18	24	31	0	**444**	1	4	72,55
M_H	0	0	0	0	0	0	0	0	0	0	1	**125**	0	99,21
M_{CEP}	0	0	0	0	0	0	0	0	0	0	20	0	**26**	56,52
													Total	75,90

Tabela 5.19 - Matriz de confusão - combinando os modelos de colunas e linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_{PC}	M_H	M_{CEP}	Taxa Rec.
0	**178**	0	2	0	0	0	0	0	6	0	14	0	0	89,00
1	0	**143**	1	0	8	0	0	11	1	0	36	0	0	71,50
2	0	0	**128**	0	1	0	0	5	5	4	57	0	0	64,00
3	0	0	0	**139**	0	1	0	12	6	1	41	0	0	69,50
4	1	0	0	0	**167**	0	0	7	0	0	25	0	0	83,50
5	0	0	0	1	1	**147**	3	0	1	1	46	0	0	73,50
6	0	0	0	0	3	7	**140**	0	12	0	38	0	0	70,00
7	0	1	0	0	4	0	0	**177**	0	0	18	0	0	88,50
8	0	0	0	0	1	0	0	1	**160**	0	38	0	0	80,00
9	0	1	0	0	14	2	0	6	7	**142**	28	0	0	71,00
M_{PC}	9	25	4	1	14	13	12	27	17	0	**488**	1	1	79,74
M_H	0	0	0	0	0	0	0	0	0	0	3	**123**	0	97,62
M_{CEP}	0	0	0	0	0	0	0	0	0	0	19	0	**27**	58,70
													Total	77,55

Um comparativo entre os resultados obtidos com os modelos para segmentação-reconhecimento e os modelos para verificação (combinando colunas e linhas) pode ser observado na Tabela 5.20. Conforme pode ser observado, houve uma melhora nas taxa de reconhecimento com a inclusão dos modelos baseados nas linhas da imagem, principalmente no reconhecimento dos caracteres (um aumento de aproximadamente 20%), o que era um problema nos experimentos anteriores.

Tabela 5.20 - Comparativo entre modelos para segmentação-reconhecimento e os modelos para verificação.

Classe / Meta-Classe	Dígitos / M_P / M_C / M_H		Dígitos/ M_P / M_C / M_H / M_{CEP}		Dígitos/ M_{PC} / M_H / M_{CEP}	
	Segmentação-Reconhecimento	Verificação	Segmentação-Reconhecimento	Verificação	Segmentação-Reconhecimento	Verificação
0	93,00	89,00	87,50	88,50	90,50	89,00
1	71,00	68,50	73,50	70,00	76,00	71,50
2	66,00	60,00	81,00	69,00	65,00	64,00
3	73,00	71,00	81,00	73,50	69,00	69,50
4	80,00	81,00	81,50	78,00	80,50	83,50
5	81,00	69,50	76,00	65,00	76,50	73,50
6	71,00	71,00	69,50	70,50	65,00	70,00
7	68,00	70,50	77,50	81,00	84,50	88,50
8	77,00	81,00	78,50	81,50	75,50	80,00
9	73,50	72,50	62,50	64,50	75,50	71,00
M_P	87,50	94,50	89,00	96,50	-	-
M_{PC}	-	-	-	-	59,64	79,74
M_C	46,36	74,51	52,67	75,49	-	-
M_H	96,03	93,65	96,03	94,44	95,24	97,62
M_{CEP}	-	-	65,22	60,87	58,70	58,70
Total	72,83	76,04	74,82	76,65	72,84	77,55

Experimentos agrupando os dígitos em uma única Meta-Classe

Nestes experimentos foram refeitos os experimentos 5, 6 e 7, agrupando os dígitos em uma única Meta-Classe. Os resultados obtidos são apresentados nas Tabelas 5.21 a 5.26.

Tabela 5.21 - Matriz de confusão experimento 5 - modelos baseados em linhas

Meta-Classe	$M_{0...9}$	M_P	M_H	M_C	Taxa Rec.
$M_{0...9}$	**1814**	12	0	174	90,70
M_P	2	**189**	0	9	94,50
M_H	0	0	**124**	2	98,41
M_C	134	26	0	**252**	61,17
				Total	86,89

Tabela 5.22- Matriz de confusão experimento 5 - combinando os modelos de colunas e linhas

Meta-Classe	$M_{0...9}$	M_P	M_H	M_C	Taxa Rec.
$M_{0..9}$	**1749**	32	0	219	87,45
M_P	3	**191**	0	6	95,50
M_H	0	0	**119**	7	94,44
M_C	125	29	0	**258**	62,62
				Total	84,62

Tabela 5.23 - Matriz de confusão experimento 6 - modelos baseados em linhas

Meta-Classe	$M_{0...9}$	M_P	M_C	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	**1809**	16	173	0	2	90,45
M_P	2	**183**	15	0	0	91,50
M_C	123	28	**256**	1	4	62,14
M_H	0	0	9	**115**	2	91,27
M_{CEP}	5	3	13	0	**25**	54,35
					Total	85,78

Tabela 5.24 - Matriz de confusão experimento 6 - combinando os modelos de colunas e linhas

Meta-Classe	$M_{0...9}$	M_P	M_C	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	**1811**	28	158	0	3	90,55
M_P	3	**186**	11	0	0	93,00
M_C	166	31	**214**	0	1	51,94
M_H	0	0	15	**111**	0	88,10
M_{CEP}	1	10	7	0	**28**	60,87
					Total	84,41

Tabela 5.25 - Matriz de confusão experimento 7 - modelos baseados em linhas

Meta-Classe	$M_{0...9}$	M_{PC}	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	1904	94	0	2	95,20
M_{PC}	217	387	2	6	63,24
M_H	0	2	123	1	97,62
M_{CEP}	4	18	0	24	52,17
				Total	87,57

Tabela 5.26- Matriz de confusão experimento 7 - combinando os modelos de colunas e linhas

Meta-Classe	$M_{0...9}$	M_{PC}	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	1897	98	1	4	94,85
M_{PC}	276	333	2	1	54,41
M_H	0	6	120	0	95,24
M_{CEP}	1	18	0	27	58,70
				Total	85,38

Como pode ser observado nas tabelas acima, houve uma melhora nas taxas de reconhecimentos dos dígitos em relação aos experimentos 5, 6 e 7. Porém houve uma queda

nas taxas de reconhecimento dos caracteres, de 79,74% (experimento 7) para 54,41%, no último experimento.

A Tabela 5.27 apresenta um comparativo com os resultados obtidos com os modelos para segmentação-reconhecimento, agrupando os dígitos em uma única Meta-Classe (experimento 4). Como pode ser observado houve um aumento nas taxas de reconhecimento.

Tabela 5.27 - Comparativo entre os experimentos agrupando os dígitos em uma única Meta-Classe.

Meta-Classe	$M_{0..9}$ / M_P / M_C / M_H		$M_{0..9}$ / M_P / M_C / M_H / M_{CEP}		$M_{0..9}$ / M_{PC} / M_H / M_{CEP}	
	Segmentação-Reconhecimento	Verificação	Segmentação-Reconhecimento	Verificação	Segmentação-Reconhecimento	Verificação
$M_{0..9}$	84,55	87,45	87,35	90,55	91,30	94,85
M_P	87,50	95,50	84,00	93,00	-	-
M_{PC}	-	-	-	-	44,12	54,41
M_C	49,27	62,62	31,55	51,94	-	-
M_H	96,03	94,44	92,86	88,10	96,03	95,24
M_{CEP}	-	-	65,22	60,87	60,87	58,70
Total	79,99	84,62	78,74	84,41	80,64	85,38

5.3.2. Resultados obtidos combinando as primitivas de *Foreground* e *Background*

Neste experimento foram refeitos os experimentos que obtiveram os resultados mais promissores na Seção anterior, incluindo agora as primitivas de *Background*.

Experimento 8

Neste experimento foram refeitos os testes com as seguintes Meta-Classes:

- Os dígitos de 0 a 9;
- Palavras e Caracteres em uma única Meta-Classe: $M_{PC;}$
- Hífen: $M_{H;}$
- A palavra CEP: $M_{CEP;}$

Os resultados obtidos podem ser observados nas Tabelas 5.28, 5.29 e 5.30.

Tabela 5.28 - Matriz de confusão - modelos baseados em colunas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_{PC}	M_H	M_{CEP}	Taxa Rec.
0	**183**	0	6	0	2	0	0	0	6	0	1	0	2	91,50
1	0	**153**	8	4	3	1	0	10	3	2	6	10	0	76,50
2	1	2	**176**	0	0	2	0	2	8	3	6	0	0	88,00
3	0	0	12	**156**	0	9	0	6	10	2	5	0	0	78,00
4	0	4	2	0	**150**	0	3	5	0	3	31	1	1	75,00
5	0	5	7	2	2	**161**	6	5	1	2	9	0	0	80,50
6	0	1	4	1	1	1	**171**	0	10	0	11	0	0	85,50
7	0	4	1	1	6	0	0	**176**	2	3	7	0	0	88,00
8	0	1	6	0	1	2	2	1	**171**	7	9	0	0	85,50
9	0	1	4	2	11	7	0	12	10	**138**	12	0	3	69,00
M_{PC}	26	36	24	3	13	33	77	24	39	5	**320**	7	5	52,29
M_H	0	5	0	0	0	0	0	0	0	0	2	**119**	0	94,44
M_{CEP}	0	0	0	0	0	0	0	0	0	0	19	0	**27**	58,70
													Total	75,47

Tabela 5.29 - Matriz de confusão - modelos baseados em linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_{PC}	M_H	M_{CEP}	Taxa Rec.
0	**184**	0	5	0	0	0	0	0	8	1	2	0	0	92,00
1	0	**164**	1	1	11	0	0	6	6	0	9	2	0	82,00
2	0	8	**158**	0	0	0	1	5	6	2	20	0	0	79,00
3	0	8	4	**160**	0	0	0	5	10	2	11	0	0	80,00
4	0	8	0	0	**175**	0	0	5	0	1	11	0	0	87,50
5	0	4	0	0	0	**144**	2	1	3	3	43	0	0	72,00
6	0	1	1	1	1	4	**164**	0	14	0	14	0	0	82,00
7	0	5	0	0	10	0	0	**169**	0	4	12	0	0	84,50
8	3	0	1	0	0	1	0	0	**166**	6	23	0	0	83,00
9	0	3	2	0	11	0	0	4	4	**163**	13	0	0	81,50
M_{PC}	16	47	39	3	23	19	30	17	25	16	**375**	2	0	61,27
M_H	0	0	0	0	0	0	0	0	0	0	1	**125**	0	99,21
M_{CEP}	0	0	0	0	0	0	0	0	0	0	36	0	**9**	20,00
													Total	77,47

Tabela 5.30 - Matriz de confusão - combinando os modelos de colunas e linhas

Classe / Meta-Classe	0	1	2	3	4	5	6	7	8	9	M_{PC}	M_H	M_{CEP}	Taxa Rec.
0	**186**	0	5	0	0	0	0	0	7	0	2	0	0	93,00
1	0	**164**	3	1	2	0	0	13	5	0	10	2	0	82,00
2	0	3	**164**	0	0	0	0	4	8	1	20	0	0	82,00
3	0	0	3	**162**	0	0	0	9	13	1	12	0	0	81,00
4	0	4	0	0	**174**	0	0	5	0	0	16	0	1	87,00
5	0	3	0	1	0	**157**	2	5	4	2	26	0	0	78,50
6	0	1	2	0	0	1	**171**	0	10	0	15	0	0	85,50
7	0	1	1	0	6	0	0	**178**	2	2	10	0	0	89,00
8	0	0	1	0	0	1	0	0	**179**	3	16	0	0	89,50
9	0	1	2	0	8	2	0	8	4	**154**	21	0	0	77,00
M_{PC}	16	33	15	3	6	25	30	22	30	4	**426**	1	1	69,61
M_H	0	0	0	0	0	0	0	0	0	0	4	**122**	0	96,83
M_{CEP}	0	0	0	0	0	0	0	0	0	0	20	0	**26**	56,52
													Total	81,29

Experimento 9

Neste experimento foram refeitos os experimentos com as seguintes Meta-Classes:

- $M_{0...9}$
- M_P
- M_H
- M_C

Os resultados obtidos podem ser observados nas Tabelas 5.31, 5.32 e 5.33.

Tabela 5.31 - Matriz de confusão - modelos baseados em colunas

Meta-Classe	$M_{0...9}$	M_P	M_H	M_C	Taxa Rec.
$M_{0...9}$	1846	27	29	98	92,30
M_P	17	163	0	20	81,50
M_H	2	0	121	3	96,03
M_C	219	27	13	153	37,14
				Total	83,38

Tabela 5.32 - Matriz de confusão - modelos baseados em linhas

Meta-Classe	$M_{0...9}$	M_P	M_H	M_C	Taxa Rec.
$M_{0...9}$	1848	0	2	150	92,40
M_P	2	160	0	38	80,00
M_H	0	0	125	1	99,21
M_C	143	9	6	254	61,65
				Total	87,18

Tabela 5.33 - Matriz de confusão - combinando os modelos de colunas e linhas

Meta-Classe	$M_{0...9}$	M_P	M_H	M_C	Taxa Rec.
$M_{0...9}$	1863	3	10	124	93,15
M_P	3	169	0	28	84,50
M_H	0	0	121	5	96,03
M_C	163	13	9	227	55,10
				Total	86,92

Experimento 10

Neste experimento foram refeitos os experimentos com as seguintes Meta-Classes:

- $M_{0..9}$
- M_{PC}
- M_H
- M_{CEP}

Os resultados obtidos podem ser observados nas Tabelas 5.34, 5.35 e 5.36.

Tabela 5.34 - Matriz de confusão - modelos baseados em colunas

Meta-Classe	$M_{0...9}$	M_{PC}	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	1922	53	20	5	96,10
M_{PC}	320	278	11	3	45,42
M_H	3	2	121	0	96,03
M_{CEP}	5	14	0	27	58,70
				Total	84,34

Tabela 5.35 - Matriz de confusão - modelos baseados em linhas

Meta-Classe	$M_{0...9}$	M_{PC}	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	1955	41	4	0	97,75
M_{PC}	319	290	3	0	47,39
M_H	0	1	125	0	99,21
M_{CEP}	0	37	0	9	19,57
				Total	85,45

Tabela 5.36 - Matriz de confusão - combinando os modelos de colunas e linhas

Meta-Classe	$M_{0...9}$	M_{PC}	M_H	M_{CEP}	Taxa Rec.
$M_{0...9}$	1949	40	11	0	97,45
M_{PC}	314	292	5	1	47,71
M_H	0	5	121	0	96,03
M_{CEP}	1	19	0	26	56,52
				Total	85,78

A Tabela 5.37 apresenta um comparativo com os resultados obtidos nos experimentos anteriores, combinando os modelos de colunas e linhas.

Tabela 5.37 - Comparativo com os resultados obtidos nos experimentos anteriores.

Meta-Classe	Dígitos/ M_{PC} / M_H / M_{CEP}		$M_{0..9}$ / M_P / M_C / M_H		$M_{0..9}$ / M_{PC} / M_H / M_{CEP}	
	Foreground	*Foreground*/ Background	Foreground	*Foreground*/ Background	Foreground	*Foreground*/ Background
0	89,00	93,00	-	-	-	-
1	71,50	82,00	-	-	-	-
2	64,00	82,00	-	-	-	-
3	69,50	81,00	-	-	-	-
4	83,50	87,00	-	-	-	-
5	73,50	78,50	-	-	-	-
6	70,00	85,50	-	-	-	-
7	88,50	89,00	-	-	-	-
8	80,00	89,50	-	-	-	-
9	71,00	77,00	-	-	-	-
$M_{0..9}$	73,05*	84,45*	87,45	93,15	94,85	97,45
M_P	-	-	95,50	84,50	-	-
M_{PC}	79,74	69,61	-	-	54,41	47,71
M_C	-	-	62,62	55,10	-	-
M_H	97,62	96,83	94,44	96,03	95,24	96,03
M_{CEP}	58,70	56,52	-	-	58,70	56,52
Total	77,55	81,29	84,62	86,92	85,38	85,78

* O calculo da taxa de reconhecimento da Meta-Classe $M_{0..9}$ foi feita dividindo o número de dígitos classificados corretamente pelo número de exemplos de dígitos disponíveis para teste.

Como pode ser observado houve uma melhora nas taxas de reconhecimento dos dígitos. Porém houve uma queda nas taxas de reconhecimento das palavras e caracteres. Uma explicação para esta queda nas taxas de reconhecimento é que estas Meta-Classes englobam uma grande variedade de informações (palavras, caracteres, ruídos, etc.) e com variação no

tamanho da escrita. Além disso, as primitivas de *Background* são baseadas em informações de concavidade, devendo levar em conta o contexto da palavra.

5.4. Discussão

Como pode ser observado nos experimentos descritos nas seções anteriores, com o uso de Meta-Classes para a modelagem das informações houve uma melhora nos resultados. Os melhores resultados foram obtidos agrupando os dígitos em uma única Meta-Classe. Além disso, com a utilização de Meta-Classe haverá uma diminuição no número de classes a serem investigadas pelo algoritmo *Level Building* no processo de segmentação-reconhecimento das linhas do bloco do endereço.

Outro ponto investigado foi que como em 66% dos casos os dígitos do CEP vem precedidos da palavra CEP, a sua localização no bloco do endereço pode auxiliar a busca do CEP e ser um critério de seleção nos casos em que forem encontradas mais de uma seqüência de dígitos que podem representar a estrutura do CEP (uma seqüência de 5 ou 8 dígitos, separados ou não pelo traço), como o número da rua, por exemplo. Nos testes realizados durante a criação dos modelos para segmentação-reconhecimento, no melhor caso foi obtida uma taxa de 60,87% no reconhecimento da palavra CEP, o que se significa que em mais da metade dos casos ela foi encontrada. Além disso, houve poucos casos em que outras classes foram classificadas incorretamente como esta Meta-Classe, podendo influenciar na localização do CEP. Nos casos em que a palavra CEP for incorretamente classificada como palavra ou caractere, não haverá influência na localização do CEP. Além disso, houve poucas confusões com dígitos (0,019%). Como o objetivo do método é a busca de uma seqüência de dígitos, é importante que não haja confusões dos dígitos com outras classes.

A localização do traço separando os dígitos do CEP também poderá auxiliar na localização do CEP. Embora o bloco do endereço contenha informações que possam ser confundidas com o traço (‘/’, por exemplo), foi obtida uma taxa de reconhecimento da classe hífen de 96,03% (no melhor resultado para os modelos para segmentação-reconhecimento).

O conjunto de Meta-Classes que obteve os melhores resultados na representação das informações que compõem o bloco do endereço, foi obtido com as seguintes Meta-Classes: $M_{0..9}$ (Agrupa todos os dígitos de 0 a 9), M_{PC} (agrupa palavras e caracteres), M_{H} (o traço que separa os dígitos do CEP), M_{CEP} (a palavra CEP). Foi obtida uma taxa de reconhecimento de

80,64% nos modelos para segmentação-reconhecimento e de 85,38% nos modelos para verificação dos resultados (extraindo somente primitivas de *Foreground*).

Com a inclusão das primitivas baseadas em concavidades (***Background Features***), houve uma melhora nas taxas de reconhecimento dos dígitos (97,45% no melhor caso). Porém houve uma diminuição nas taxas de reconhecimento das palavras e caracteres. Esta diminuição pode ser explicada pela grande variedade de informações que estas Meta-Classes englobam (palavras de diferentes tamanhos, caracteres, ruídos, etc.).

Embora os resultados mostrem que muitos caracteres são classificados incorretamente como dígitos, este problema poderá ser resolvido na próxima etapa do método, onde os modelos criados serão utilizados para reconhecer uma linha de texto e a busca do CEP será feita pela procura de uma seqüência de dígitos. Seguindo esta estratégia, mesmo que um caractere seja classificado incorretamente como dígito, para que este pertença ao CEP ele deverá pertencer a uma seqüência de 5 ou 8 dígitos. E nos casos onde uma seqüência de 5 ou 8 caracteres sejam classificados incorretamente como dígitos, a localização da palavra CEP poderá evitar este tipo de erro.

Até o momento não foram feitas estimativas em relação ao tempo gasto pelo método para a localização do CEP no bloco do endereço uma vez que nem todas as etapas do método foram desenvolvidas.

Não foram feitos estudos de viabilidade nem levantamento de requisitos, os quais não faziam parte do escopo do trabalho.

O próximo capítulo apresenta as conclusões e trabalhos futuros.

6. Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um método para a localização do CEP no bloco do endereço em envelopes postais manuscritos e foram avaliadas estratégias para a modelagem dos componentes de uma linha do bloco do endereço.

Os melhores resultados foram obtidos com a seguinte configuração de Meta-Classes: $M_{0...9}$ (Agrupa todos os dígitos de 0 a 9), M_{PC} (agrupa palavras e caracteres), M_H (o traço que separa os dígitos do CEP), M_{CEP} (a palavra CEP). Foi obtida uma taxa de reconhecimento de 80,64% nos modelos para segmentação-reconhecimento e de 85,38% nos modelos para verificação dos resultados (extraindo da imagem primitivas de *Foreground*).

Uma desvantagem dos modelos criados, é que muitas letras são classificadas incorretamente como dígitos. Isto se deve ao fato de que muitas letras são escritas de forma parecida aos dígitos e de que a base de dados disponível para treinamento, validação e teste do método é pequena, podendo não ser representativa de todas as variações que podem ser encontradas. Este problema poderá ser amenizado no momento da localização do CEP. Como a busca do CEP consiste na localização de uma seqüência de dígitos, mesmo que uma letra seja classificada incorretamente como dígito, esta será descartada se não pertencer a uma seqüência de dígitos. E no caso de uma seqüência de letras serem classificadas como dígitos, a presença da palavra CEP poderá evitar este tipo de erro.

Como contribuições do trabalho realizado podem ser citadas:

- A proposta de um método para a localização do CEP no bloco do endereço que não leva em conta informações posicionais do CEP, fazendo uma busca em todo o bloco do endereço.

- A avaliação de Meta-Classes na representação das informações que compõem as linhas do bloco do endereço e para amenizar o problema da base de dados disponível ser pequena.
- Avaliação de um conjunto de primitivas (*Foreground/Background Features*) no contexto de envelopes postais, onde pode ser encontrada uma grande variabilidade de informações e onde não existem linhas ou um local reservado para o preenchimento das informações e estas podem ser preenchidas em qualquer posição, com qualquer inclinação, sobrepostas ou misturadas.

Os trabalhos futuros são:

- Avaliar os modelos criados na identificação dos componentes de linhas extraídas do bloco do endereço. Para tal será necessário que sejam adicionadas informações de contexto aos modelos criados, tais como os espaços entre os componentes da linha.
- Definir uma estratégia de busca do CEP no bloco do endereço. Um exemplo seria começar a busca do CEP no bloco do endereço de baixo para cima, visto que o CEP na maioria dos casos é escrito nas últimas linhas. Outra estratégia seria descartar linhas cujo tamanho da seqüência de observações não permita a existência do CEP.
- Criar uma base de dados com exemplos de linhas extraídas do bloco do endereço e aumentar o número de exemplos de caracteres.

Referências

[Aas, et al. 1996] - AAS, K.; EIKVIL, L.; ANDERSEN, T. Text Recognition from Grey Level Images Using Hidden Markov Models. *Pattern Recognition,* Vol. 29, Nr. 6, p.503 1996.

[Britto 2001] - BRITTO, A. S.; A Two-Stage Hmm-based Method for Recognizing Handwritten Numeral Strings. PUCPR – Pontifícia Universidade Católica do Paraná, 2001. Tese de Doutorado.

[Britto 2003] - BRITTO, A. S.; SABOURIN, R.; BORTOLOZZI, F., SUEN, C. Y. The Recognition of Handwritten Numeral Strings Using a Two-stage HMM-based Method. International Journal of Document Analysis and Recognition (IJDAR), USA , vol.5, p.102 - 117, 2003.

[Britto, et al. 2001a] - BRITTO, A. S.; FREITAS, C. O. A., JUSTINO, E. J. R.; BORGES, D. L.; FACON, J.; BORTOLOZZI, F.; SABOURIN, R. Técnicas em Processamento e Análise de Documentos Manuscritos. RITA, vol. VIII, n. 1, 2001.

[Britto, et al. 2001b] - BRITTO, A. S.; SABOURIN, R.; BORTOLOZZI, F.; SUEN, C. Y. A two stage HMM-Based System for Recognizing Handwritten Numeral Strings. ICDAR, p. 396-400, 2001.

[Cheung, et al 1997] - CHEUNG, K.; YEUNG, D; CHIN, R. T. Recognition of Handwritten Digits Using Deformable Models. *Progress in Handwriting Recognition*, p. 145-150, 1997.

[Correios 2004] - www.correios.com.br – Acessado em 05/02/2004.

[Dzuba, et al. 1999] - DZUBA, G.; FILATOV, A.; VOLGUNIN, A. Handwritten Zip Code Recognition. *Proceedings of the 4th IEEE Int. Conf. on Document Analysis and Recognition*, p. 766-770, 1997.

[Freitas 2001] - FREITAS, C. O. A.; “O uso de Modelos Escondidos de Markov para Reconhecimento de Palavras Manuscritas”. PUCPR - Pontifícia Universidade Católica do Paraná, 2001. Tese de Doutorado.

[Freitas, et al. 2000] - FREITAS, C.O.A; MORITA, M.; OLIVEIRA, L.S.; JUSTINO, E.J.R.; EL YACOUBI, A.; LETHELIER, E.; BORTOLOZZI, F.; SABOURIN R. Base de Dados de Cheques Bancários Brasileiros. *XXVI Conferencia Latinoamerica de Informática*, (CLEI'2000), Atizapan de Zaragosa (México), CD vol. 1, p.18-22, 2000 (in portuguese).

[Garris 1992] - GARRIS, M. D. Design and Collection of a Handwriting Sample Image Database. *Social Science Computing Journal*, vol. 10, p. 196-214, 1992.

[Grother, 1995] - GROTHER, P. Handprinted forms and character database, NIST special database 19, March 1995. Tech. Rpt. and CDROM.

[Hirano, et al. 1997] - HIRANO, T.; OKADA, Y.; YODA, F. Structural Character Recognition Using Simulated Annealing. ICDAR, vol. 2, p. 507-510, 1997.

[Kimura, et al. 1995] - KIMURA, F.; MIYAKE, Y.; SHRIDHAR, M. Handwritten ZIP Code Recognition Using Lexical Free Word Recognition Algorithm. *Proceedings of the IEEE International Conference on Systems*. p. 906-910, 1995.

[Laaksonen 1997] - LAAKSONEN, J. Subspace Classifiers in Recognition of Handwritten Digits. Doctoral thesis. Acta Polytechnica Scandinavica Ma 84, Espoo, April, 1997.

[Leche, et al. 2000] - LECHE, V. D; DIMAURO, G.; GUERRIERO A; IMPEDOVO S; PIRLO, G.; SALZO, A. Zoning Design For Handwritten Numeral Recognition. IWFHR VII, p. 583-588, 2000.

[Lee & Gomes, 1997] - LEE, L. L; GOMES, N. R. Disconnected Handwritten Numeral Image Recognition. ICDAR, vol. 2, p. 467-470, 1997.

[Marques & Vieira 1999] - MARQUES, O. F.; VIEIRA, H. N. Processamento Digital de Imagens. Ed. Brasport. Rio de Janeiro - 1999.

[Matan, et al. 1992] - MATAN, O.; BAIRD, H. S.; BROMLEY, J.; BURGES, C.J.C.; DENKER, J.S.; JACKEL, L.D.; LE CUN, Y., PEDNAULT, E. P. D; SATTERFIELD, W.D.; STENARD, C.E.; THOMPSON, T.J. Reading Handwritten Digits: a ZIP Code Recognition System. *IEEE Computer*, Vol. 25, p. 59-63, 1992.

[Morita, et al. 2002] - MORITA, M.; SABOURIN, R.; BORTOLOZI, F.; SUEN, Y. Segmentation and Recognition of Handwritten Dates. IWFHR, p.105-110, 2002.

[Oliveira & Morita 1998] - OLIVEIRA, L. E. S. de; MORITA, M. E; Introdução aos Modelos Escondidos de Markov (HMM). Technical Report (PPGIA), PUCPR, November 1998.

[Rabiner 1989] - RABINER, L. R.; A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, vol. 77, n. 2, February 1989.

[Silva 2002] - SILVA, E. Reconhecimento Inteligente de Caracteres Manuscritos. Instituto Militar de Engenharia, Rio de Janeiro, 2002. Dissertação de Mestrado.

[Trier, et al. 1996] - TRIER, O. D.; JAIN, A.K.; TAXT, T. Feature extraction methods for character Recognition - a survey. Pattern Recognition, vol. 29, n. 4, 1996, p.641-662.

[Trier, et al. 1997] - TRIER, O. D; TAXT, T; JAIN, A. K. Recognition of Digits in Hydrographic Maps: Binary vs. Topographic Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, No. 4, p. 399-404, 1997.

[Wang 1994] – WANG, X. Durationally constrained training of HMM without explicit state duration *PDF*. Institute of Phonetic Sciences, University of Amsterdam, Proceedings 18, p. 111-130, 1994.

[Whichelo & Yan 1996] - WHICHELO, A. P.; YAN, H. Locating Address Blocks and Postcodes in Mail-Piece Images. ICPR 96, vol. 17, p. 1199-1214, 1996.

[Yacoubi, et al. 1999] – EL-YACOUBI, A.; GILLOUX, M.; SABOURIN, R. An HMM-Based Approach for Off-line Unconstrained Word Modeling and Recognition. *1999 IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, p. 752-760, 1999.

[Yamauchi, et al. 1997] - YAMAUCHI, T.; ITAMOTO, Y.; TSUKUMO, J. Shape based learning for a multi-template method, and its application to hand printed numeral recognition. ICDAR, vol. 2, p. 495-498, 1997.