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

Reconhecimento de Caracteres Japoneses

TRABALHO DE GRADUAÇÃO

Recife, julho de 2008

Reconhecimento de Caracteres Japoneses

TRABALHO DE GRADUAÇÃO

MARCONI EMANUEL MADRUGA FILHO

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do Grau de Bacharel em Ciências da Computação.

Orientador: Prof. George Darmiton da Cunha Cavalcanti

Recife, julho de 2008

Resumo

O reconhecimento de caracteres manuscritos, é uma área desafiadora. A grande variabilidade intra-classe é uma barreira a ser vencida. Além disso, no domínio de caracteres orientais o grande número de classes torna o problema ainda mais complexo. Enquanto os alfabetos ocidentais trabalham com, no máximo, 52 classes de caracteres (diferenciando-se maiúsculas de minúsculas), o alfabeto japonês possui mais de 5000 classes.

Devido a essa clara disparidade, as abordagens *on-line* e/ou *off-line* utilizadas são, em sua maioria, diferentes das que tratam de problemas ocidentais. Elas se aproveitam da estrutura hierárquica e do traçado regular dos caracteres para criar classificadores que realcem essas características.

O objetivo do trabalho desenvolvido é estudar e implementar todas as fases do processo que envolve um sistema de reconhecimento de caracteres japoneses. Um banco de dados próprio foi gerado, e diversas técnicas de processamento de imagens e de suporte ao reconhecimento foram implementadas.

Sumário

Índice de Figuras

Índice de Tabelas

Índice de Algoritmos

1 Introdução

O reconhecimento de caracteres é uma área que ainda está em aberto. Há diversas técnicas e soluções satisfatórias para reconhecimento de caracteres digitados (*machine-printed*), mas a área de reconhecimento de caracteres manuscritos ainda representa um grande desafio para os pesquisadores. Isso se deve à grande variação que há na maneira de escrever de cada pessoa.

Para entender o estado-da-arte das técnicas de reconhecimento de caracteres, é necessário, a princípio, fazer uma distinção entre os dois tipos de técnicas de reconhecimento: *on-line* e *off-line*.

O reconhecimento *on-line* se refere a técnicas que podem reconhecer uma letra pelo modo como ela foi escrita. São técnicas que não utilizam a imagem de uma letra, mas informações como: pressão aplicada em determinado ponto, largura e ordem de cada traço, tempo em que a caneta esteve na superfície e fora da superfície, entre outras. Aplicações que envolvem reconhecimento em *tablet PCs* e *hardwares* que possuam dispositivos de entrada semelhante podem utilizar esse tipo de técnica de forma eficiente.

O reconhecimento *off-line*, por sua vez, trata-se de um conjunto de métodos que utilizam a imagem em sua forma final, após o término da escrita. Não possui informações extras como ordem dos traços, apenas a imagem pura do caracter. É aplicado facilmente a domínios como identificação de endereços em cartas de correios e digitalização de documentos.

Para o caso de caracteres japoneses, ou orientais em geral, a situação é outra, dadas as grandes diferenças entre eles e os caracteres de línguas ocidentais. O contexto da língua e caracteres japoneses, assim como uma visão de algumas técnicas específicas que existem, atualmente, para esse domínio são explicados a seguir.

1.1 Escrita Japonesa

A língua japonesa é considerada uma das línguas de mais difícil aprendizado para falantes não-nativos. Isso não é apenas devido à sua gramática diferente da maioria das línguas ocidentais, mas ao seu sistema de escrita. O japonês possui 3 alfabetos com caracteres próprios: Hiragana, Katakana e Kanji.

O Hiragana e o Katakana são alfabetos de 46 caracteres fonéticos (fonogramas) cada, semelhantes ao alfabeto latino. Hiragana são usados, em sua maioria, para representar terminações e variações de palavras, bem como partículas. Os Katakana, por sua vez, são utilizados para representar palavras de origem estrangeira, através do seu som.

Já os Kanji são ideogramas baseados em caracteres chineses. O alfabeto de Kanji possui mais de 5000 caracteres, dos quais 2000, aproximadamente, são exigidos para leitura de jornais e revistas no dia-a-dia.

Para o propósito de processamento computacional, os caracteres japoneses são divididos em 2 classes: JIS First Level e JIS Second Level (JIS é

Japanese Industrial Standard). Enquanto o JIS First Level contém apenas os caracteres básicos, o JIS Second Level é composto por caracteres mais raros, alguns usados apenas em nomes próprios [8].

Além da grande diferença em relação aos outros alfabetos em termos de cardinalidade, os Kanji possuem uma escrita característica, com formas padrão, que se repetem em várias classes de caracteres, e grande número de traços.

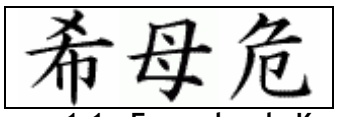

Figura 1.1 - Exemplos de Kanji

1.2 Reconhecimento de Caracteres Japoneses

Somando-se os 3 alfabetos japoneses aos caracteres ocidentais (numéricos ou não), que também são utilizados na escrita japonesa, há aproximadamente de 6000 a 7000 caracteres em uso na língua japonesa, incluindo escrita técnica e literária [8]. As técnicas para reconhecimento são, obviamente, influenciadas pela grande disparidade em relação ao número de classes entre caracteres ocidentais e orientais. Há diversas técnicas próprias para reconhecimento de caracteres japoneses, dentre *off-line* e *on-line*, algumas das quais foram estudadas e estão explanadas a seguir.

1.2.1 Reconhecimento *On-line*

O fato de caracteres japoneses serem bem estruturados e possuírem padrões característicos de escrita dá à área de reconhecimento *on-line* desses caracteres abordagens diferentes das ocidentais. Na escrita japonesa, informações como ordem e sentido dos traços são partes constituintes do aprendizado de cada caracter.

Além disso, os kanji possuem padrões recorrentes chamados de radicais. Em [9], essa característica é bem aproveitada para o reconhecimento a partir de uma gramática de radicais. Cada kanji é visto como uma composição hierárquica de traços.

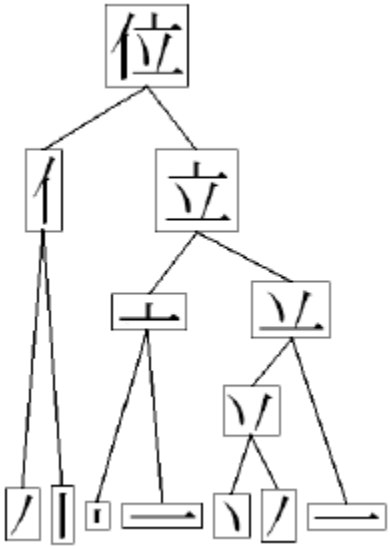

Figura 1.2 - Representação Hierárquica do Kanji 位 (“grau”).

A partir daí, os traços, combinados com rótulos que indicam o seu posicionamento relativo são capazes de compor todos os caracteres, como no exemplo abaixo:

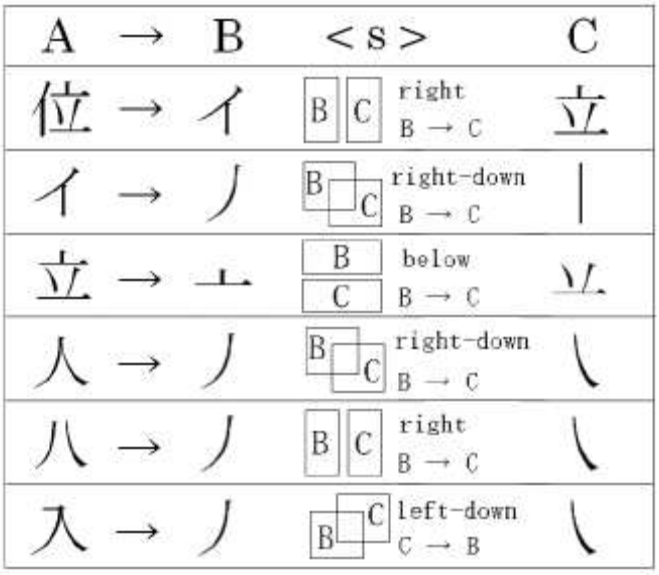

Figura 1.3 - Combinação de traços e rótulos formando novos caracteres.

Já a técnica citada em [10] utiliza um dicionário de traços que é independente de ordenação, sendo robusto a caracteres que sejam escritos de forma incorreta. Os elementos dos dicionários são “subtraços” curtos e longos divididos em pen-down (quando a caneta está tocando a superfície) e pen-up (quando a caneta está suspensa, entre a escrita de 2 traços).

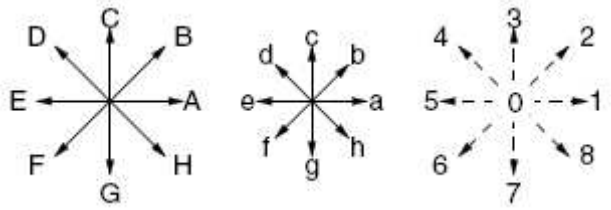

Figura 1.4 - Valores representativos de subtraços na composição do dicionário.

Há também técnicas que utilizam a característica estrutural dos kanji para a geração de protótipos como em [11], dentre outras.

1.2.2 Reconhecimento *Off-line*

No reconhecimento *off-line*, a hierarquia e estrutura bem definida dos kanji não pode ser tão bem aproveitada quando no *on-line*. No entanto a forma geral dos traços dos kanji, sempre muito característica, acaba direcionando o estudo das técnicas para alternativas de extração de características.

Em geral, há técnicas de extrações de características que consideram detecção de linhas em diversas direções na imagem, como em [12]. Este artigo mostra uma técnica chamada de Gabor Filter Extraction, que utiliza filtros de Gabor para gerar imagens secundárias e usá-las como características. A Figura 1.5 mostra os componentes reais e imaginários dos filtros de Gabor em 2 escalas.

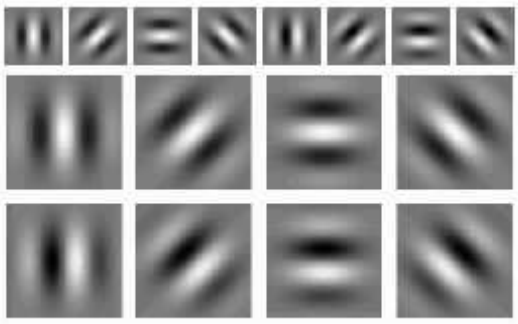

Figura 1.5 - Filtros de Gabor em 2 escalas e 4 direções. Extraído de [12].

Liu [17] propõe uma estrutura bem definida de um sistema de classificação completo de reconhecimento de caracteres orientais. E estrutura do sistema ideal proposto está exposta na figura 1.5.

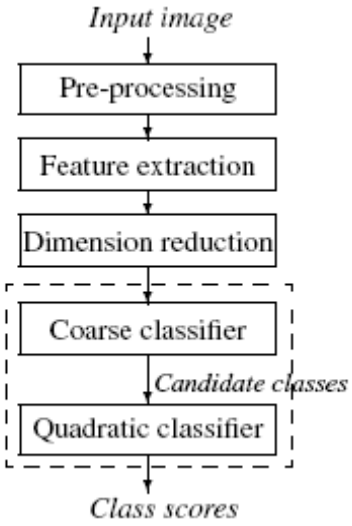

Figura 1.6 - Diagrama de um sistema de reconhecimento de caracteres orientais. Extraído de [17]

Ele utiliza estratégias das mais diversas etapas dos problemas de reconhecimento de padrões, dentre elas: extração de características pelo gradiente e redução de dimensionalidade utilizando o Discriminante Linear de Fisher (FDL), além de uma medida de classificação chamada MQDF (*Modified Quadratic Discriminant Function*). Essa medida é discriminante, ou seja, ela indica o quão próximos são 2 conjuntos de dados. No entanto, ao contrário da maioria das medidas discriminantes, ela utiliza conceitos de espaços vetoriais como autovalores e autovetores no seu cálculo.

Há também técnicas que tentam utilizar a normalização de caracteres em prol da melhoria do desempenho. Enquanto [13] lista um conjunto grande de técnicas de normalização, Liu [14] usa algumas dessas técnicas para contribuir na extração de características.

1.3 O trabalho

Este trabalho tem como objetivo estudar e entender todo o processo de um sistema de reconhecimento de caracteres *off-line*, desde a geração de um banco de dados próprio, até a etapa final, de classificação. A abordagem escolhida foi a *off-line*, devido à maior portabilidade para testes, pois não demanda um hardware específico. Além disso, um banco de dados desse tipo exige muito menos esforço para coleta de dados tanto por parte do usuário quanto de quem estiver montando o banco.

As técnicas utilizadas e estudadas são detalhadas no Capítulo 2. O processo de geração do banco de dados é detalhado no Capítulo 3. No Capítulo 4, são mostrados experimentos feitos e seus resultados. E por fim, no Capítulo 5, a conclusão e possíveis vertentes para geração de trabalhos futuros.

2 Técnicas

Um sistema de reconhecimento *off-line* de caracteres possui um processo próprio e bem definido. Engloba técnicas que tratam desde a imagem original até o vetor de dados utilizado para classificação. O diagrama abaixo ilustra esse fluxo:

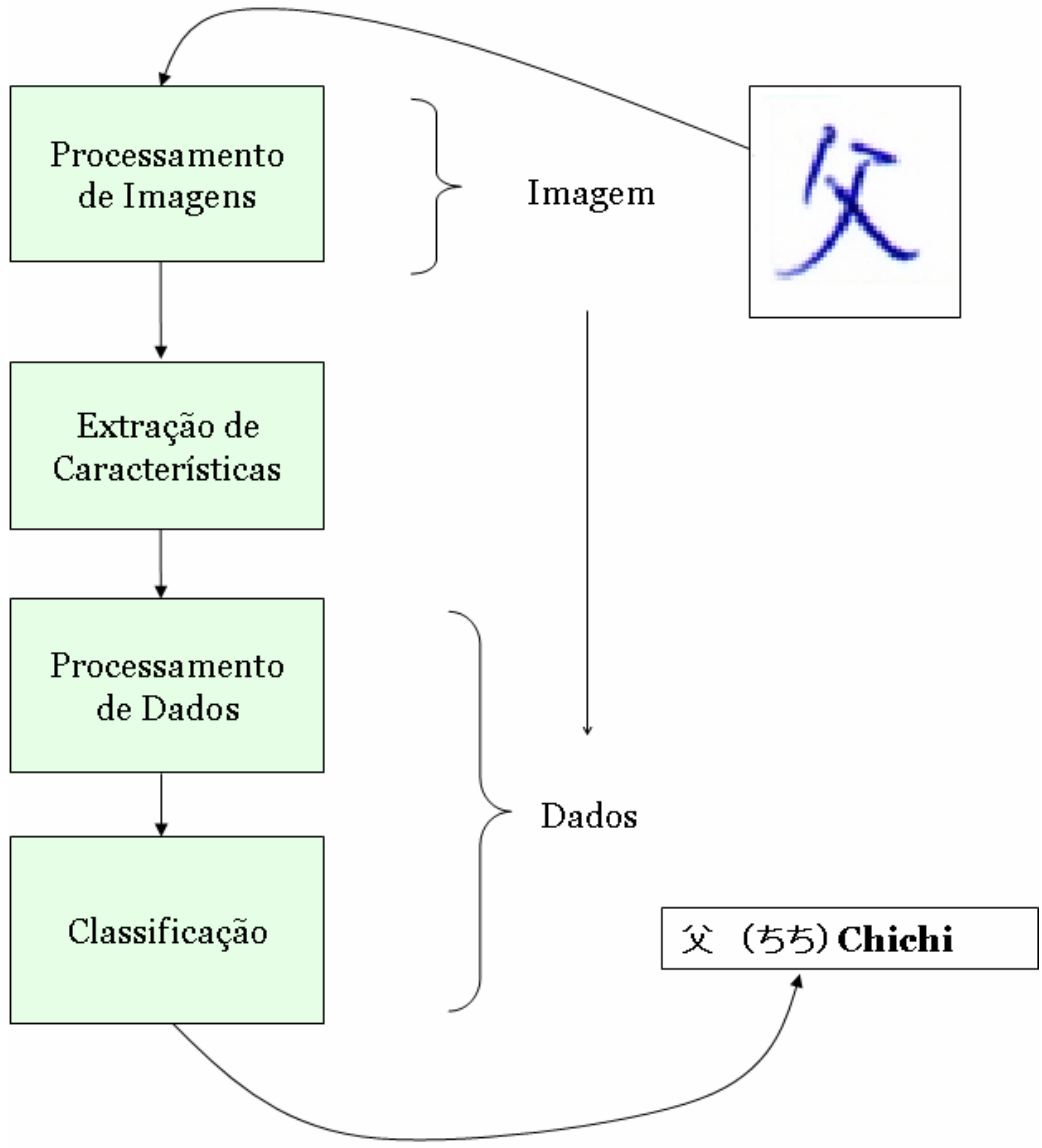

Figura 2.1 - Processo de Classificação de Caracteres.

Os passos do fluxo são detalhados a seguir, assim como as técnicas implementadas em cada um deles.

2.1 Processamento de Imagens

O Processamento de Imagens se refere ao processamento de imagens digitais através de um computador. Em geral possui duas grandes áreas de aplicação: melhoria da qualidade da informação visual da imagem e processamento de imagens para armazenamento, transmissão e representação para percepção computacional autônoma [5]. No caso deste trabalho, as técnicas aplicadas têm a função de pré-processamento da imagem, com o intuito final de melhorar a qualidade da classificação. Alguns dos métodos descritos têm como objetivo a **uniformização** das imagens, eliminando informações que não são relevantes para a etapa de classificação. Outros pretendem **melhorar a qualidade** da imagem, eliminando ruídos ou realçando-as. As técnicas são descritas a seguir:

2.1.1 Image Trim

Apesar do espaço fornecido para a escrita do caracter ser o mesmo para todos os escritores, não se pode determinar onde e nem qual tamanho ele terá dentro do espaço fornecido. A Figura 2.2 ilustra alguns exemplos comuns de disparidades:

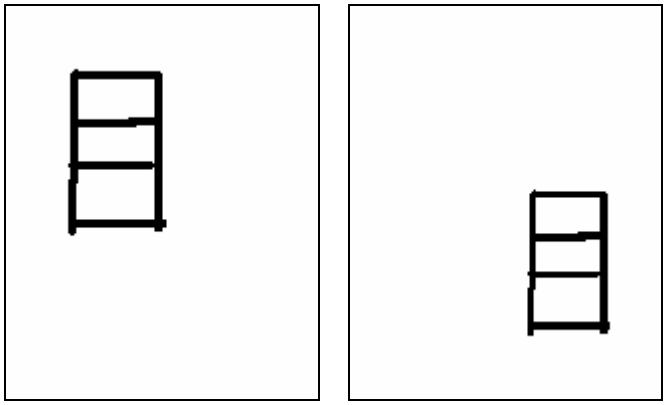

Figura 2.2 - Caracter '目'("olho"), escrito de 2 formas diferentes. Observe que as 2 formas estão bem semelhantes, mas devido às variações de escala e posição, podem dificultar a geração de um padrão para classificação.

Essas disparidades podem ser corrigidas, ou minimizadas, na etapa de pré-processamento. O posicionamento do caracter no quadrado fornecido para a escrita é uma informação pouco relevante no momento da classificação. Para desprezar essa característica, foi utilizada uma técnica de Image Trim.

A técnica consiste em eliminar da imagem tudo o que não pertença ao menor retângulo contendo todos os pixels de objeto da imagem. Para encontrar esse retângulo, a imagem é percorrida de diversas formas:

1. A partir do canto superior esquerdo, lendo uma linha por vez, sempre da esquerda para a direita. Para encontrar y_1.
2. A partir do canto superior esquerdo, lendo uma coluna por vez, sempre de cima para baixo. Para encontrar x_1.
3. A partir do canto inferior direito, lendo uma linha por vez, sempre da direita para a esquerda. Para encontrar y_2.
4. A partir do canto inferior direito, lendo uma coluna por vez, sempre de baixo para cima. Para encontrar x_2.

O retângulo é definido pelos pontos (x_1, y_1) e (x_2, y_2). A Figura 2.3 ilustra a operação:

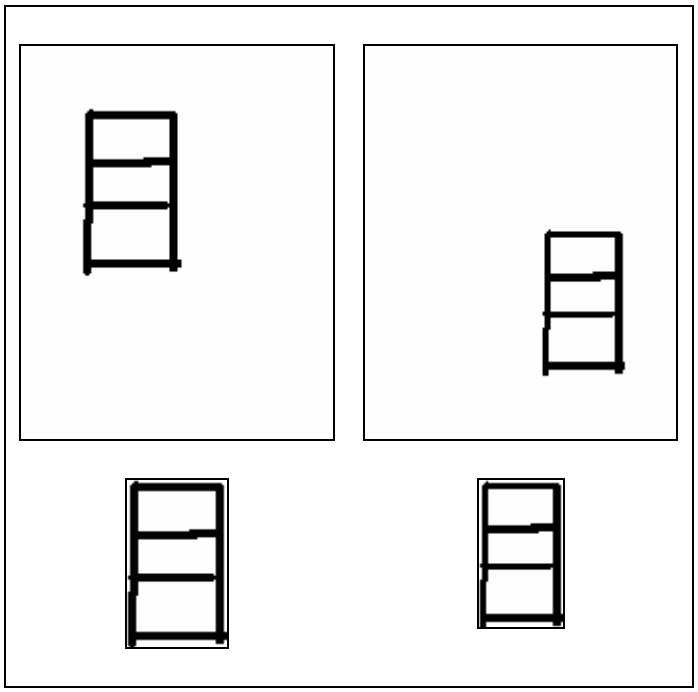

Figura 2.3- Caracter '目'("olho"), escrito de 2 formas diferentes e suas respectivas imagens após o Image Trim. Nas imagens resultantes, o posicionamento relativo do caracter não possui mais influência.

Com essa operação, é possível desconsiderar a informação do posicionamento do caracter no espaço fornecido, ainda persistindo, no entanto, a diferença de escala. Essa é amenizada pela normalização.

2.1.2 Normalização – Lanczos Resampling

A normalização visa redimensionar uma imagem tentando evitar a sua distorção, ou seja, mantendo o máximo de características da imagem original. Neste caso, a normalização não só é útil, como é necessária, pois todas as imagens precisam ter o mesmo tamanho para serem comparadas. Além disso, devido à grande quantidade de pixels que uma imagem grande possui, um redimensionamento para uma imagem menor pode contribuir para a melhoria da performance.

A técnica de normalização escolhida foi o Lanczos resampling, amplamente utilizado para redimensionamento de imagens digitais. Nas imagens abaixo, percebe-se claramente o ganho em suavidade ao usar Lanczos contra um método que usa ferramentas semelhantes à compressão JPEG.

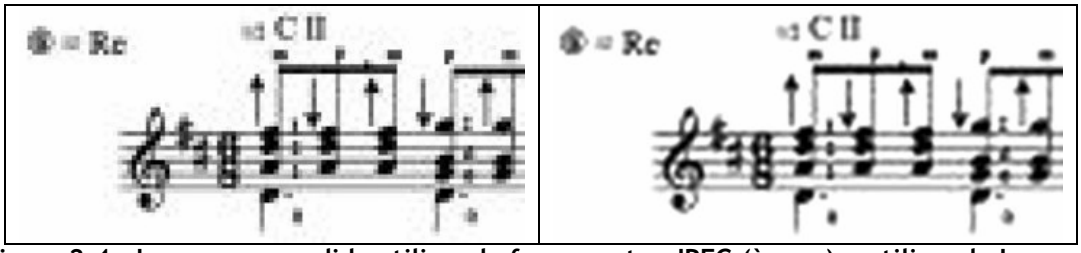

Figura 2.4 - Imagem expandida utilizando ferramentas JPEG (à esq.) e utilizando Lanczos resampling (à dir.). Fonte [4]

Cada pixel da imagem resultante é formado por um conjunto de contribuições dos pixels da imagem original. O método de Lanczos informa quais os pixels e em quais proporções comporão o valor de um pixel na imagem final. Isso é feito através de uma máscara de convolução. Os valores da máscara são dados pela função de Lanczos:

$$L(x) = \begin{cases} \operatorname{sinc}(x)\operatorname{sinc}\left(\frac{x}{a}\right) & -a < x < a, x \neq 0 \\ 1 & x = 0 \\ 0 & \text{do contrário} \end{cases}$$

Sabendo que o x é a distância entre os centros do pixel do kernel e o pixel desejado. O valor de *a*, tamanho da máscara, foi escolhido como 3. O pixel central sobrepõe o pixel desejado, tendo distância 0 e peso máximo (1). A função sinc (sinoidal cardinality) é definida por:

$$\text{sinc}(x) = \text{sen}(\pi x)/\ \pi x$$

As imagens abaixo ilustram um exemplo comparativo da aplicação do método de Lanczos para redimensionamento.

Figura 2.5 - Imagem original (acima) e redimensionamento utilizando método de interpolação de nearest-neighbour (Esq.) e utilizando método de lanczos (Dir.)

2.1.3 Dilatação ⊕

A dilatação é uma operação morfológica que, aplicada sobre uma imagem binária, tem um efeito de expansão da área de objeto da imagem. O tamanho e as condições para essa expansão dependem de uma máscara que será utilizada como o objeto dilatador. É, portanto, uma operação que envolve dois objetos. Dado o seguinte exemplo:

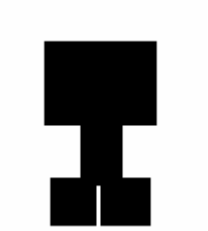

Figura 2.6 - Objeto A (que será dilatado)

Figura 2.7 - Objeto B (que será utilizado como máscara de dilatação)

Assume-se que o Objeto B é um quadrado com 3x3 pixels e cujo ponto de referência é o ponto central. O efeito da dilatação é o mesmo de colar cópias do Objeto B sobre cada pixel do objeto A, usando o ponto de referência de B para passar por cada pixel. Ou seja, cada pixel do objeto A é sobreposto com o ponto de referência do Objeto B, o restante do objeto é mapeado para a Imagem A, acrescentando ao Objeto A os pixels que eram de background, mas que foram sobrepostos por B.

O resultado da dilatação de A por B é o seguinte:

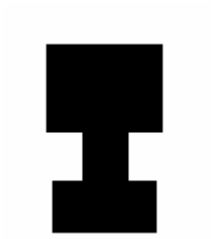

Figura 2.8 - Objeto C, resultado de A ⊕ B.

Outros exemplos de dilatação seguem abaixo:

Figura 2.9 - Exemplos de dilatação.

2.1.4 Erosão Θ

Como a dilatação, a erosão também é uma operação morfológica. No entanto, ao ser aplicada sobre uma imagem binária tem o efeito de encolhimento da área do objeto da imagem. A erosão também é feita utilizando outro objeto como máscara, o objeto erodente. Exemplo:

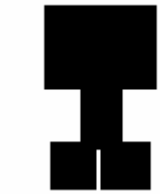

Figura 2.10 - Objeto A (que será erodido)

Figura 2.11 - Objeto B (que será utilizado como máscara de erosão)

Assume-se que o Objeto B é um quadrado com 3x3 pixels e cujo ponto de referência é o ponto central. A erosão ocorre da seguinte forma: removem-se os pixels de objeto de A que, se sobrepostos com o ponto de referência de B, não tiverem uma cópia de B correspondente em A. Mais detalhes na Figura 2.12:

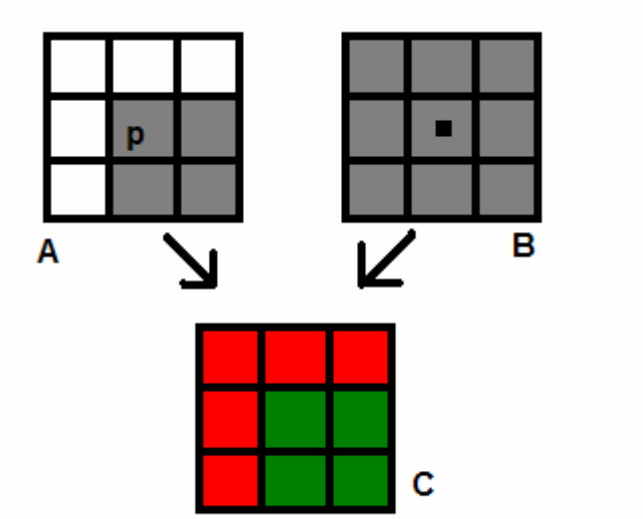

Figura 2.12 - Erosão - pixel que será removido.

Ao sobrepor o pixel p em A com o objeto B, tem-se o resultado mostrado em C. Onde os pixels **verdes** são casamentos (objeto em A e objeto em B) e os **vermelhos** são disparidades (*background* em A e objeto em B). Como p gera disparidades ele será removido. A Figura 2.13 abaixo mostra um exemplo onde um pixel t **não** é removido:

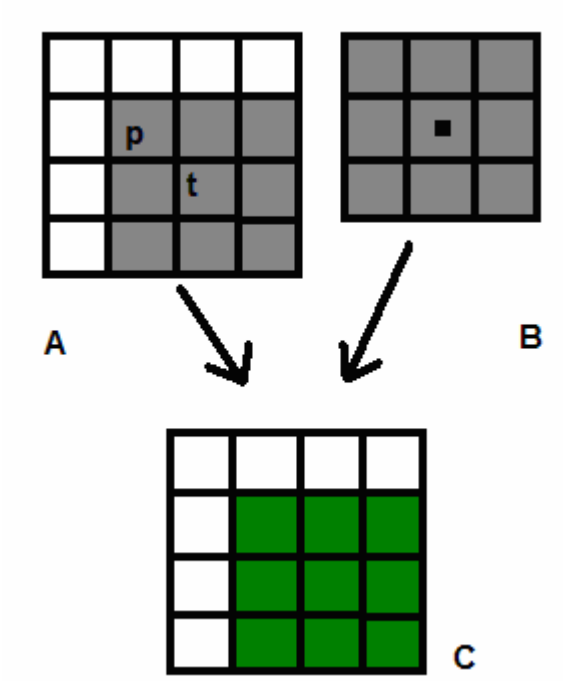

Figura 2.13 - Erosão - pixel que não será removido.

O resultado da erosão de A por B é o seguinte:

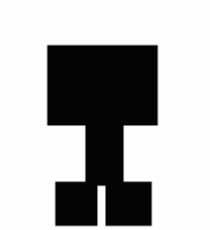

Figura 2.14 - Objeto C, resultado de A Θ B.

Outros exemplos de erosão seguem abaixo:

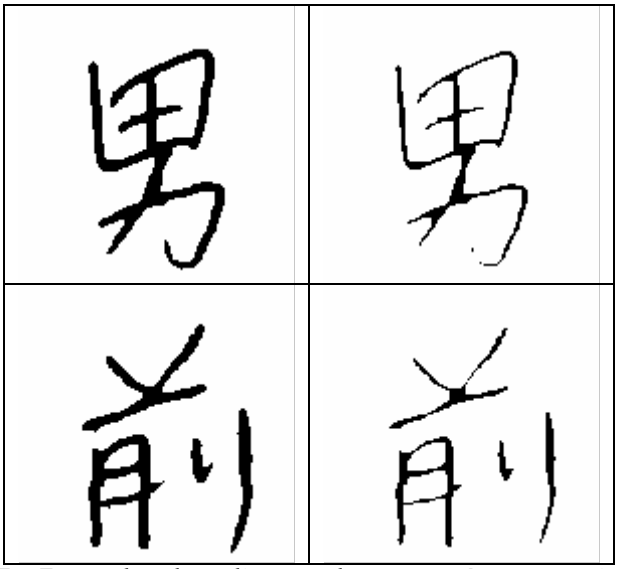

Figura 2.15 - Exemplos de aplicação de erosão. Imagem original à esq.

2.1.5 Abertura e Fechamento

Abertura e fechamento são combinações das técnicas morfológicas citadas acima: a erosão e a dilatação. Abertura é a aplicação da erosão, seguida pela dilatação. Como o próprio nome já diz, tem o efeito de abertura do objeto eliminando linhas finas e pequenas pontas que existiam na imagem original, sem alterar de maneira excessiva a espessura da imagem, ao contrário da erosão.

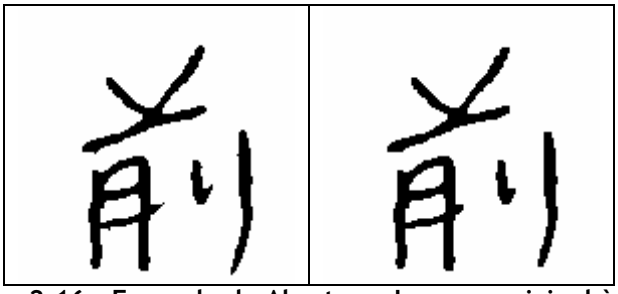

Figura 2.16 - Exemplo de Abertura. Imagem original à esq.

Já o fechamento (dilatação seguida de erosão) tem o efeito de preencher pequenos buracos no objeto. Isso também é feito de forma a manter o formato da imagem original, diferentemente da dilatação.

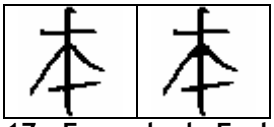

Figura 2.17 - Exemplo de Fechamento.
Imagem original à esq.

2.1.6 Esqueletização (NWG)

Ainda na etapa de pré-processamento, pode ser utilizada a esqueletização. A esqueletização tem como principal objetivo obter uma representação simplificada do(s) objeto(s) contido(s) na imagem, eliminando a sua informação de espessura. Isso é feito de forma que a estrutura do objeto seja preservada. A esqueletização pode contribuir no desempenho do reconhecimento, evitando que objetos semelhantes com espessuras diferentes sejam classificados erroneamente.

A técnica de esqueletização escolhida para implementação foi uma versão melhorada do *NWG Thinning* (Nagendraprasad-Wang-Gupta) [3]. É um algoritmo iterativo rápido e eficiente, que mantém a conectividade dos objetos nos esqueletos gerados. A princípio, a partir da imagem original, cada pixel de objeto é analisado. A análise determinará se o pixel será ou não removido (pixel redundante) e é feita avaliando o pixel e seus 8-vizinhos. Os 8-vizinhos de 1 pixel são numerados em sentido horário, segundo o esquema a seguir:

p(7)	p(0)	p(1)
p(6)	p	p(2)
p(5)	p(4)	p(3)

Numeração dos 8-vizinhos

Além disso, são considerados também:

- b(p) – Número de vizinhos de p que são “ligados” (pixels de objeto)
- a(p) – Número de transições “desligado-ligado” quando os pixels são visitados em ordem ao redor de p. A Figura 2.18 ilustra um pixel com a(p) = 2:

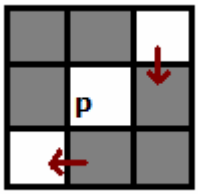

Figura 2.18 - Pixel com a(p) = 2.

As condições:

- c(p) = $\begin{cases} 1 & \text{se } p(0) = p(1) = p(2) = p(5) = 0 \text{ e } p(4) = p(6) = 1 \\ 1 & \text{se } p(2) = p(3) = p(4) = p(7) = 0 \text{ e } p(6) = p(0) = 1 \\ 0 & \text{caso contrário} \end{cases}$
- d(p) = $\begin{cases} 1 & \text{se } p(1) = p(4) = p(5) = p(6) = 0 \text{ e } p(0) = p(2) = 1 \\ 1 & \text{se } p(0) = p(3) = p(6) = p(7) = 0 \text{ e } p(2) = p(4) = 1 \\ 0 & \text{caso contrário} \end{cases}$
- e(p) = (p(2) + p(4)) * p(0) * p(6)
- f(p) = (p(6) + p(0)) * p(4) * p(2)

De posse das funções definidas acima, pode-se compreender o funcionamento do algoritmo NWG:

Algoritmo 2.1 - Esqueletização NWG

```
1   entrada : Q (mapa de pixels)
2   saída : Q
3   g = 1; h = 1; Q0 = Q; (configurações iniciais)
4   do (while h = 1)
5           h = 0;
6           Q = Q0;
7           g = 1 − g;
8           for (cada pixel p ∈ Q)
9                   if (1 < b(p) < 7 and (a(p) = 1 or (1 − g) * c(p) + g * d(p) = 1)) then
10                          if (g=0 and e(p)=0) then
```

```
11                              apague p em Q0
12                              h = 1
13                      endif
14                      if (g=1 and f(p)=0) then
15                              apague p em Q0
16                              h = 1
17                      endif
18              endif
19      endfor
20  enddo
21  end NWG
```

O algoritmo, a cada iteração, remove pixels que atendem a certas condições. Essas condições mudam se a iteração for par ou ímpar (variável g).

Condições fixas:

- Número de vizinhos maior que 1 e menor que 7;
- Apenas 1 transição "desligado-ligado".

Iterações pares (g = 1):

- d(p) =1 e f(p)=0.

Iterações ímpares (g = 0):

- c(p) =1 e e(p)=0.

Implementação

O algoritmo proposto considera pixels de objeto como os que têm valor 1. Na implementação do algoritmo, ele foi ligeiramente adaptado para suportar ambos os casos (objeto = 1 ou objeto = 0). As imagens abaixo são exemplos da aplicação do algoritmo:

Figura 2.19 - Exemplos da aplicação da esqueletização NWG.

2.1.7 Binarização (Otsu)

A informação de cor (ou nível de cinza), assim como a espessura, também é uma informação que pode ser desprezada em prol de uma melhora na classificação. Representar a imagem na forma binária, além disso, abre as portas para uma nova gama de técnicas, como a esqueletização, que recebem como entrada apenas imagens binárias.

A binarização de Otsu é uma técnica que visa encontrar o limiar que maximize a variância entre classes. Isso é feito da seguinte forma:

Dados:

- n_i – número total de pixels com tom de cinza i;
- L – número de níveis de cinza (geralmente, 256);
- $N = n_1 + n_2 + n_3 + ... + n_L$, o número total de pixel na imagem;
- $p_i = n_i / N$, a probabilidade de um pixel ter intensidade i.

Dado um limiar k, os pixels com tom de cinza l, $l < k$, são classificados em uma classe e o restante em outra classe. A variância entre os dois grupos pode ser definida por:

- $\sigma_B^2 = \pi_0\pi_1(\mu_1 - \mu_0)^2$

Sabendo que πi é o somatório das probabilidades em cada classe:

$\pi_0 = \sum_{i=1}^{k} p_i$, $\pi_1 = \sum_{i=k+1}^{L} p_i = 1 - \pi_0$

e μi é a média de cada classe:

$\mu_1 = \sum_{i=1}^{k} ip_i / \pi_0$, $\mu_0 = \sum_{i=k+1}^{L} ip_i / \pi_1$

O problema é maximizar a razão entre a variância entre os grupos (σ_B^2) e a variância total (σ_T^2).

$\sigma_T^2 = \sum_{i=1}^{L} (i - \mu_T)^2 p_i$

A razão é calculada para cada valor de k e o limiar que obtiver o valor máximo é escolhido como o limiar global da imagem. Todos os tons de cinza abaixo desse limiar são mapeados para 0 e o restante para 1.

2.2 Extração de Características

2.2.1 Sobel Edge-Masked Maps

Os operadores de Sobel são conhecidos operadores para detecção de bordas e regiões de alta freqüência na imagem. São capazes de detectar elementos direcionais na imagem, como linhas horizontais e/ou verticais. Tais

elementos são bastante representativos, visto que o próprio córtex do olho humano leva em consideração características semelhantes [19].

De posse desse conceito, esta técnica [19] tenta modelar os mapas topográficos formados no córtex do olho humano. Isso é feito através do uso de máscaras de convolução, neste caso os operadores de Sobel. As máscaras mencionadas têm o seguinte formato:

-1	-1	-1
2	2	2
-1	-1	-1

Máscara horizontal

-1	2	-1
-1	2	-1
-1	2	-1

Máscara vertical

-1	-1	2
-1	2	-1
2	-1	-1

Máscara de diagonal direita

2	-1	-1
-1	2	-1
-1	-1	2

Máscara de diagonal esquerda

Figura 2.20 - Máscaras de Sobel

Com cada máscara é gerado um *Sobel Edge-Masked Map* para a imagem. A geração de cada mapa é feita em 3 estágios:

1. **Convolução** – a máscara é convoluída sobre a imagem, gerando uma matriz temporária.
2. ***Thresholding*** – É feito um *thresholding* nos elementos da matriz, para eliminar possíveis componentes negativos. Todo valor da matriz que seja negativo é truncado para zero.
3. **Normalização** – todos os elementos da matriz são divididos por 9, para normalização dos dados.

A partir daí, são geradas quatro matrizes de resultado, que serão transformadas num único vetor de características. O tamanho desse vetor é dado por:

$$t = h * w * 4$$

Sabendo que:

h – altura da imagem
w – largura da imagem

Como justificativa para a escolha desta técnica, há a grande semelhança entre certas classes de caracteres, diferenciadas apenas pela direção dos traços. Uma técnica que leve em consideração essas disparidades direcionais pode ter uma maior capacidade discriminativa neste domínio.

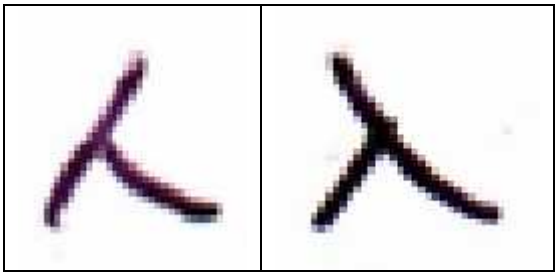

Figura 2.21 - Caracteres 人 ("pessoa") e 入 ("entrar"), respectivamente. São idênticos, a não ser pela direção do traço maior.

2.3 Processamento de Dados

Em muitos casos, é necessário, após ter o vetor de características em mãos, um pré-processamento dos dados para classificação. Muitas abordagens podem ser utilizadas como normalização dos dados, remoção de características constantes e etc. Apesar de ter sido iniciada a implementação de uma técnica de seleção de protótipos, nenhum método de processamento de dados foi efetivamente implementado e utilizado neste trabalho.

2.4 Classificação

A etapa de classificação é quando a imagem, ainda de classe desconhecida, é reconhecida como pertencente a um determinado grupo. Para a classificação é necessário ter uma base de comparação, ou base de treinamento. Alguns classificadores usam a base apenas para treinamento e ajuste e, posteriormente, tornam-se independentes dela. Outros, no entanto, usam a base para determinar a classe de cada padrão de entrada. A técnica escolhida para classificação foi o KNN, utilizando distância euclidiana.

2.4.1 KNN com distância euclidiana

O KNN (K-*nearest neighbours*) é um método de classificação que se baseia na distância entre padrões no espaço de características. Dado um conjunto de padrões classificados $V = \{v_i = (x_i, c_j) \mid cj \in C\}$ onde C é o conjunto total de classes e x é um vetor de características, a classe de um novo vetor y pode ser definida como a classe mais recorrente nos k-vizinhos de y. Os k-vizinhos são os k padrões mais próximos de y, segundo uma medida de distância previamente definida.

A medida de classe mais recorrente é apenas uma forma de determinar, dentre os k-vizinhos, qual é a classe de y. Pode-se ponderar cada vizinho por um fator para que a classe de vizinhos muito próximos tenha uma chance maior de ser escolhida. Esse fator pode ser dado, por exemplo, pelo inverso da distância (1/distância).

Dentre os métodos de classificação, o K-NN com distância euclidiana foi escolhido por ser simples de implementar e por possuir resultados satisfatórios em geral.

2.4.2 KNN com distância ponderada

O extrator de características utilizado, *Sobel Edge-Masked Maps*, gera um único vetor *v*, concatenando 4 outros (v_H, v_V, v_{DD}, v_{DE}), cada um referente a um plano de detecção de bordas. Dada a possibilidade de cada plano ter significâncias diferentes na classificação, uma nova medida de distância foi pensada.

Dados os pesos:

- w_H – taxa de acerto ao utilizar apenas o resultado da aplicação da **máscara horizontal** (v_H) de Sobel como vetor de características;
- w_V – taxa de acerto para a **máscara vertical** (v_V);
- w_{DD} – taxa de acerto para a **máscara diagonal direita** (v_{DD});
- w_{DE} – taxa de acerto para a **máscara diagonal esquerda** (v_{DE}).

A medida de distância ponderada entre dois padrões x e y será:

- $dp(x,y) = d_H(x,y)*w_H + d_V(x,y)*w_V + d_{DD}(x,y)*w_{DD} + d_{DE}(x,y)*w_{DE}$

Sabendo que:

- $d_E(x,y)$ – é a distância euclidiana entre $v_E(x)$ e $v_E(y)$, para $E \in \{H, V, DD, DE\}$.

Presume-se que está seja uma medida de distância no mínimo tão boa quanto a original, possivelmente melhor, pois as medidas que obtiverem um melhor potencial classificatório se sobressairão, contribuindo assim para a taxa de classificação total.

3 Banco de Dados

3.1 Bancos de Dados Existentes

Foi feita uma pesquisa na literatura para descobrir que bancos de dados são comumente utilizados em artigos e pesquisas no reconhecimento de caracteres orientais. Além de artigos que usam bancos próprios, criados em prol do experimento em questão, há algumas bases que são recorrentes. São elas:

1) **Nakayoshi e Kuchibue**

Nakayoshi e Kuchibue [1] são duas bases de dados pagas de caracteres on-line. A base Kuchibue, no entanto, possui uma versão de graça que é um subconjunto da base total para fins de pesquisa. Este subconjunto contém exemplos de 10(dez) escritores dos 120 presentes nas bases completa. Oferece, além disso, uma biblioteca de acesso aos dados.

2) **ETL-8 e ETL-9**

As bases de dados ETL [2] foram coletadas no Laboratório Eletrotécnico (ETL) do Instituto Nacional de Ciência Industrial e Tecnologia Avançada do Japão (AIST) sob a cooperação de várias entidades de pesquisa japonesas. As bases de ETL-1 a ETL-9 contém aproximadamente 1,2 milhão de caracteres manuscritos e digitados (machine-printed). As imagens ilustram caracteres japoneses, chineses, latinos e numéricos para fins de reconhecimento. São abertas para propósito de pesquisa. As bases ETL-8 e ETL-9, referenciadas em diversos artigos [6][17][14], são as únicas pertencentes ao conjunto total formada exclusivamente por imagens binárias.

Nenhuma das bases encontradas está disponível para livre download na internet. As que não exigem compra de licença, exigem o envio de um formulário de solicitação para o Japão, com dados sobre a pesquisa, antes da liberação.

Obtenção

Para a obtenção dos bancos foi feito contato eletrônico, sem sucesso. Tentou-se, então, através do Consulado Geral do Japão em Recife, um novo contato. Obteve-se resposta do AIST e a solicitação formal das bases ETL-8 e ETL-9 foi feita. A comunicação, no entanto, foi intermitente e levou certo tempo para ser concluída. Para poder fazer testes neste período, foi iniciado um processo de geração de uma base própria.

3.2 Base de dados própria

Dada a necessidade de realização de testes e a ausência de uma base para tal, foi gerada a demanda pela criação de uma base de dados própria.

Para isto, foi definido um fluxo de passos que deveriam ser seguidos para a geração da base, ilustrado na Figura 3.1 abaixo.

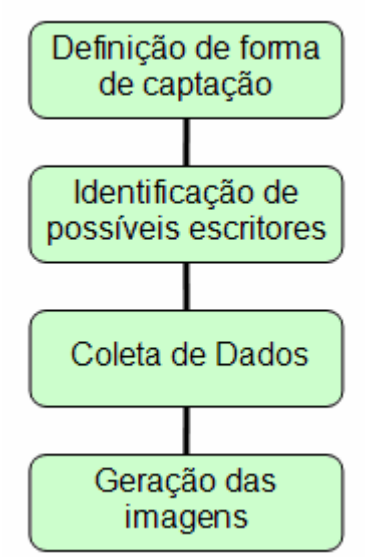

Figura 3.1 - Processo de geração da base de dados.

Cada um dos passos é definido a seguir:

3.2.1 Definição de forma de captação

O primeiro passo foi a definição de "Como os dados serão coletados?". Para responder a essa questão é necessário, primeiro, responder outras:

- A base será *on-line* ou *off-line*?
- Quantas classes de caracteres serão coletadas?
- E quantos caracteres por classe serão coletados?

Com essas perguntas respondidas, é muito mais fácil e determinístico propor um modelo de captação. Como a abordagem escolhida será a off-line não é estritamente necessário usar algum hardware especial para a captação. Isso não seria verdade caso a base escolhida fosse *on-line*, que considera informações como pressão de escrita, orientação e ordem dos traços, exigindo algum aparelho que possa lidar com tais dados. No caso off-line, só é necessária a obtenção das imagens.

Das mais de 2 mil classes, quantas serão consideradas na coleta? Obviamente é impossível, em curto prazo, incluir todas ou ao menos metade do conjunto total. O JLPT (Exame de Proficiência em Língua Japonesa) divide os kanji em 4 níveis, de acordo com o nível de dificuldade e uso na língua. Foi escolhido então como um subconjunto razoável para coleta, os caracteres do nível 4 (nível mais básico), que contém aproximadamente 100 classes.

Pensar em quantos caracteres por classe pode estar relacionado ao número de classes escolhidas. Seria desejável ter, por exemplo, ao menos uma quantidade de exemplares por classe maior ou igual ao número de classes. Pode-se então tentar estabelecer um mínimo de 100 caracteres por classe.

De posse desses dados, já se pode designar um modelo de captação de caracteres mais facilmente. Visto que é uma abordagem off-line, são

utilizadas fichas, que serão preenchidas pelos escritores. É importante que essas fichas possibilitem a **simulação de uma escrita natural**, ou seja, é importante que o escritor não precise adaptar sua maneira de escrever para se ajustar ao modelo. Por isso, deve-se evitar um espaço de escrita muito restritivo ou amplo. Em uma análise empírica, observou-se que quadrados de área igual a 1 cm^2 seriam adequados. Chegou-se, então ao formato de ficha proposto no Apêndice. Conseguiu-se comprimir as 100 classes em apenas 2 folhas, utilizando uma divisão de 9 caracteres por classe. Assim, com apenas 11 escritores, já se alcança a marca desejada de exemplares por categoria.

3.2.2 Identificação de possíveis escritores

Estabelecido o modelo, é necessário definir um conjunto de possíveis escritores para a coleta de dados. A única restrição imposta na escolha dos escritores é que tenham domínio e experiência, senão com o a língua japonesa como um todo, com o subconjunto de kanji estabelecido. A escolha do grupo de caracteres do nível 4 do exame de proficiência contribuiu para uma divisão mais clara a respeito dos que podem ou não participar do experimento (são os que têm ou não o nível 4 de proficiência na língua). Foram contatados pessoas da Associação Cultural Japonesa do Recife e da Escola de Língua Japonesa do Recife, para a contribuição com a pesquisa, bem como alguns indivíduos residentes no Japão.

3.2.3 Coleta dos dados

O ideal é que a coleta dos dados seja feita de forma que as fichas possam, a posteriori, ser digitalizadas e convertidas em um banco de dados de forma automática. Para isso, foram necessárias algumas recomendações aos escritores para o preenchimento da ficha:

- Usar caneta escura, de preferência azul ou preta;
- Se houver erro, não rasurar, usar corretivo de papel;
- Escrever dentro da área do quadrado, não cruzando a grade da ficha;

As fichas foram, então, escaneadas com uma resolução de 1215 x 1530. Quanto às fichas que foram escaneadas no Japão, não há informação sobre a configuração utilizada, apenas sobre a resolução das imagens: 3308 x 4676.

3.2.4 Geração de Imagens

Na etapa de geração de imagens, recebe-se como entrada as fichas digitalizadas e o banco de dados é gerado como saída. Para mais detalhes sobre a geração das imagens e sobre o tratamento de certos problemas como ruído e inclinação vide a Seção 3.3.

3.3 Geração de Imagens a Partir das Fichas de Captação

As fichas digitalizadas precisavam ser convertidas em imagens individuais de cada caracter escrito. Para isso, foi implementado um algoritmo que recebe uma versão binarizada da ficha e trabalha com ela de forma a permitir uma operação automatizada. Este algoritmo visa encontrar os pontos de fronteira de cada caracter para que seja feito o recorte ("*cropping*") adequado e posterior geração do banco. Observe a Figura 3.2 - Pontos de fronteira de um caracter:

Figura 3.2 - Pontos de fronteira de um caracter

A Figura 3.2 mostra um caracter da ficha e seus quatro pontos de fronteira, que serão buscados pelo algoritmo.

Algoritmo inicial

A rotina, em pseudocódigo, encontra o ponto de partida do algoritmo principal. Recebe como entrada a imagem da ficha binarizada, o valor da cor de objeto e um parâmetro *offset* que indica o quão afastados da grade serão os pontos de fronteira.

Algoritmo 3.1 - Busca do Ponto de Partida.

```
1   entrada: Ficha binarizada [F], cor de objeto (1 ou 0) [C], offset
2   saída: ponto inicial do algoritmo
3
4   altura = F.altura; // altura de F
5   largura = F.largura; // largura de F
6   y_inicial = altura/2 (um valor central);
7   //primeira etapa – encontrar borda esquerda
8   x_borda_esquerda = 0
9   for (x = 0; x < largura; x++)
10          pi = (x, y_inicial)
11          if (cor(pi) = C)
12                  x_borda_esquerda = x
13                  break
14          end if
15  end for
16  //segunda etapa – encontrar borda superior
```

```
17  y_borda_superior = y_inicial
18  for (y=y_inicial; y > 0; y--)
19          pi = (x_borda_esquerda, y)
20          if (cor(pi) !=  C) // cor de background
21                  y_borda_superior = y
22                  break
23          end if
24  end for
25  // terceira etapa, encontrar ponto inicial do algoritmo
26  borda = 0; cruzou = 0;
27  x = x + offset;
28  pi = (x,y)
29  while (borda = 0 and cruzou = 0)
30          if ( cor(pi) = C )
31                  borda = 1;
32          else if (borda = 1 and cor(pi) != C)
33                  cruzou = 1;
34          else
35                  y--;
36                  pi = (x, y);
37          end if
38  end while
39  retorna pi
```

O ponto pi é o primeiro ponto interno da grade, como ilustrado na Figura 3.3 a seguir:

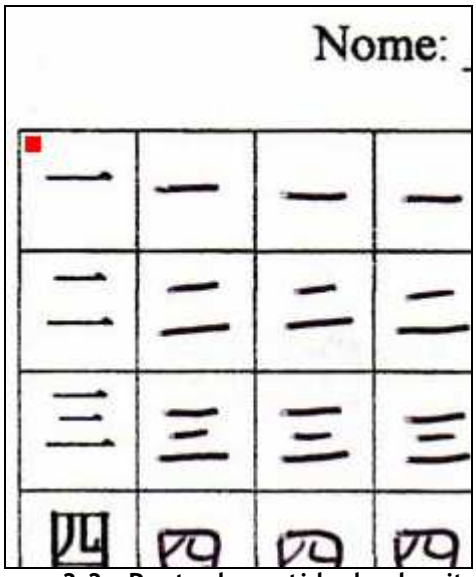

Figura 3.3 - Ponto de partida do algoritmo

Após encontrado esse ponto, o algoritmo passa para a fase de busca dos pontos de fronteira de cada caracter da grade. Isso é descrito nos algoritmos a seguir:

Algoritmo 3.2 - Busca de Valores Verticais para *Cropping*.

```
1   entrada: Ficha Binarizada [F], cor de objeto (1 ou 0) [C], ponto de
2   partida [pi], [offset], quantidade de valores verticais [ty]
3   saída: Array de valores para y
4
5   y0 = pi.y //inicializa com y do ponto inicial
6   x = pi.x
7   verticais = [].append(y0)
8   procura_linha = 1
9   for (i = y0; (i < F.altura and tamanho(verticais) < ty ); i++)
10          pt = (x, i)
11          if ( procura_linha = 1 and cor(pt) = C )
12                  //encontrou linha, guarda ponto anterior
13                  verticais = verticais.append(i – offset)
14                  procura_linha = 0
15          else if ( procura_linha = 0 and cor(pt) != C )
16                  // encontrou background, guarda ponto adiante
17                  verticais = verticais.append(i + offset)
18                  procura_linha = 1
19          end if
20  end for
21  retorna verticais
```

Algoritmo 3.3 - Busca de Valores Horizontais para *Cropping*.

```
1   entrada: Ficha Binarizada [F], cor de objeto (1 ou 0) [C], ponto de
2   partida [pi], [offset], quantidade de valores horizontais [tx]
3   saída: Array de valores para x
4
5   x0 = pi.x //inicializa com x do ponto inicial
6   y = pi.y
7   horizontais = [].append(x0)
8   procura_linha = 1
9   for (i = x0; (i < F.largura and tamanho(horizontais) < tx ); i++)
10          pt = (i, y)
11          if ( procura_linha = 1 and cor(pt) = C )
12                  //encontrou linha, guarda ponto anterior
13                  horizontais = horizontais.append(i – offset)
14                  procura_linha = 0
15          else if ( procura_linha = 0 and cor(pt) != C )
16                  // encontrou background, guarda ponto adiante
17                  horizontais = horizontais.append(i – offset)
18                  procura_linha = 1
19          end if
20  end for
21  retorna horizontais
```

Os dois algoritmos são idênticos, mudando apenas onde é necessário para que a varredura seja horizontal ou vertical. Os algoritmos foram feitos de forma independente para permitir uma implementação em paralelo, caso fosse necessário um ganho de performance. Assim como o algoritmo de busca

pelo ponto de partida, recebem como entrada a imagem e a cor de objeto, além do ponto de partida pi. Os valores *tx* e *ty* para quantidade de valores horizontais e verticais são:

- *tx* – (o número de exemplares por classe + 1) * 4, visto que em uma linha há duas classes e cada caracter precisa de dois valores de x (vide figura Figura 3.2). O número de exemplos é somado a 1, devido aos caracteres de referência na primeira e última colunas.
- *ty* – o número de classes por ficha. Apesar de cada caracter também precisar de 2 valores verticais, só há metade do número de classes por ficha em 1 coluna.

Dados os vetores $x = \{x_i \mid x_i = \text{horizontais}[i], 2 \leq i < tx - 2\}$ e $y = \{y_j \mid y_j = \text{verticais}[j], 0 \leq j < ty\}$ representando os valores relevantes de x e y para *cropping*. Os valores que envolvem os caracteres de referência na primeira e última colunas são eliminados pela condição ($2 \leq i < tx - 2$). Os pontos de fronteira de cada caracter pertencem, então, à matrix *M* que pode ser definida como:

$$M = \{p_{ij} = (x_i, y_j) \mid x_i \in x, y_j \in y\}$$

Logo, para um caracter manuscrito Ki ($0 \leq i <$ número de caracteres por classe), pertencente à classe Cj, pode-se definir seus pontos de fronteira como:

- Se ($Cj \mid j < ty/2$), ou seja, a classe está na metade esquerda da ficha, os pontos serão $\{ p_{2i,2j}; p_{2i+1,2j}; p_{2i,2j+1}; p_{2i+1,2j+1} \}$

- Se ($Cj \mid j \geq ty/2$), ou seja, a classe está na metade direita, os seguintes ajustes são feitos: $j = j - ty/2$ e $i = i +$ número de caracteres por classe. Com isso, os pontos serão: $\{ p_{2i,2j}; p_{2i+1,2j}; p_{2i,2j+1}; p_{2i+1,2j+1} \}$

Iterando por todos os possíveis K, e utilizando os pontos de fronteira encontrados, é possível gerar cada imagem separadamente e converter a ficha para um banco de imagens. Pode-se observar que nos algoritmos acima, uma variável de ajuste, chamada *offset*, foi utilizada. Esta variável tem como finalidade inserir um pequeno afastamento entre os pontos de fronteira e a grade da ficha, para evitar problemas com ruído e inclinação. Vide a Figura 3.4, abaixo:

Figura 3.4 - Pontos de fronteira. Observe que há um afastamento entre os pontos e a borda, introduzido pela variável *offset*.

A pesar da introdução do *offset*, problemas ainda surgiram. A descrição desses problemas e como eles foram contornados estão detalhados a seguir.

3.4 Problemas encontrados

Abaixo serão descritas dificuldades encontradas ao aplicar os algoritmos descritos na seção anterior. Para cada um dos problemas listados abaixo, serão informadas estratégias de prevenção (utilizadas antes de fornecer a imagem para os algoritmos) e de contingência (incluídas no próprio funcionamento dos algoritmos) utilizadas.

1. Ruído

Algumas fichas chegavam com um nível de ruído elevado, provavelmente proveniente de falha de impressão. Após a aplicação da binarização, então, surgiam situações como a ilustrada abaixo:

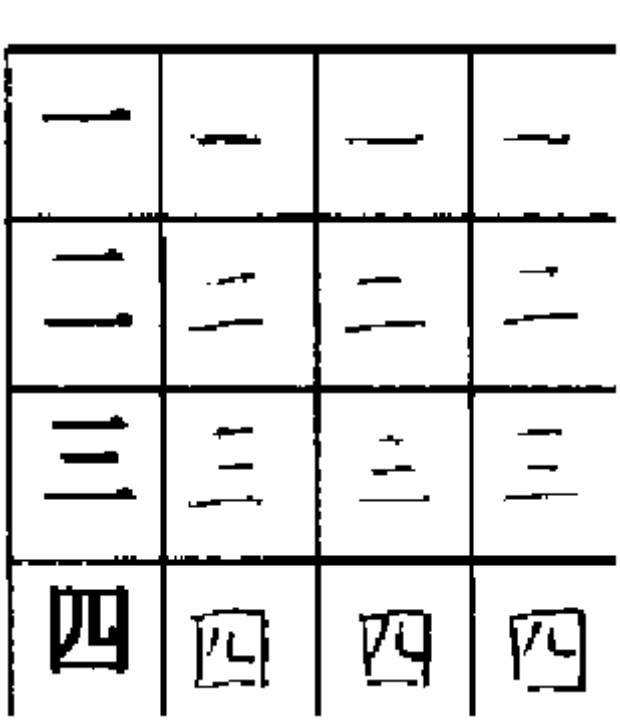

Figura 3.5 - Imagem binarizada com ruído

É possível observar que, dada a lógica do algoritmo de busca pelo ponto de partida previamente descrito, esse tipo de problema pode atrapalhar no seu funcionamento. O problema é ilustrado no esquema abaixo:

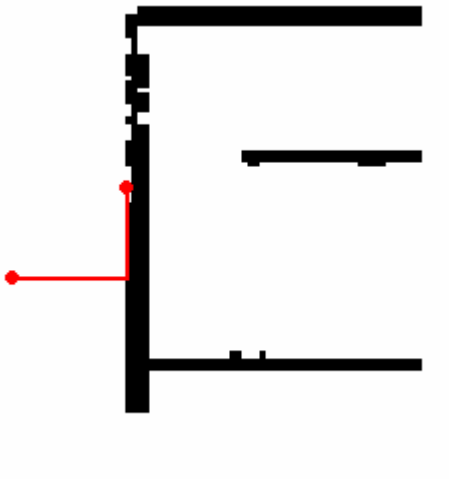

Figura 3.6 - trajetória da varredura em busca do ponto inicial. Imagem com ruído.

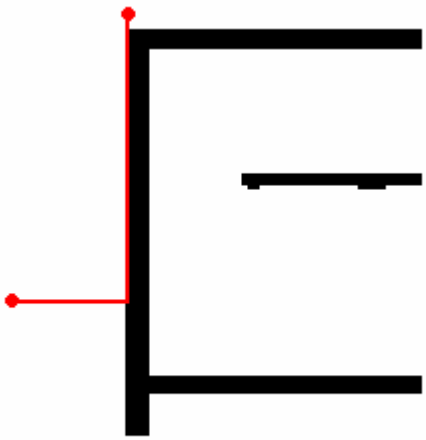

Figura 3.7 - Trajetória da varredura em busca do ponto inicial. Imagem sem ruído.

Estratégias de prevenção

- Suavização – Aplicando um filtro de suavização (descrito no Capítulo 2) antes da binarização ajuda a diminuir mudanças bruscas de cores na imagem (altas freqüências, características do ruído) e contribui para a binarização, apesar de não eliminar todo o ruído.

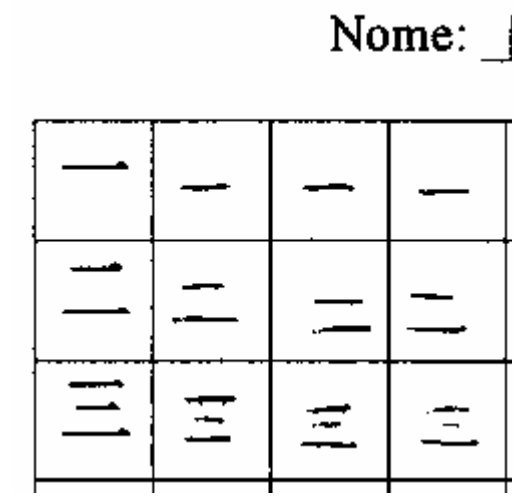

Figura 3.8 - Imagem binarizada sem aplicação de suavização prévia

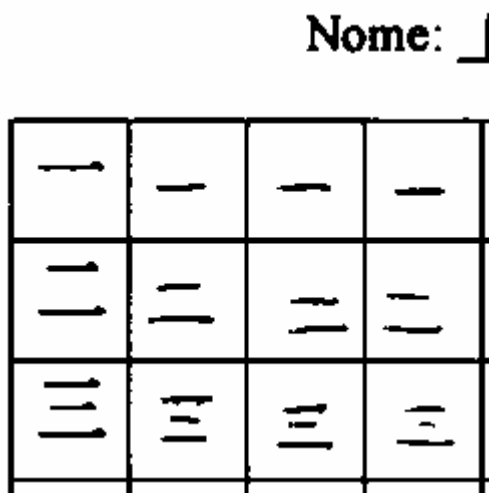

Figura 3.9 - Imagem binarizada com aplicação de suavização prévia

- Fechamento – Aplicando um filtro morfológico de fechamento (descrito no Capítulo 2) após a binarização, contribui para a eliminação de pequenas reentrâncias nas linhas da grade.

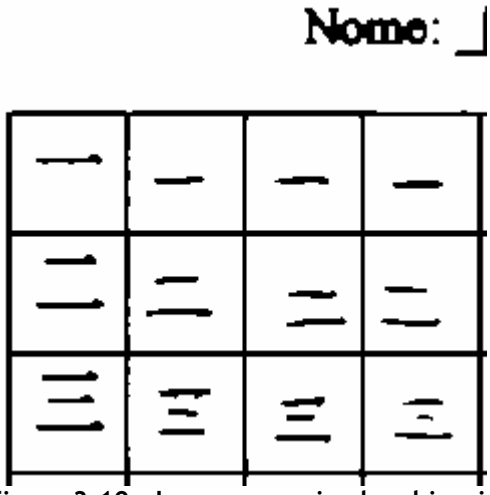

Figura 3.10 - Imagem suavizada e binarizada com posterior aplicação de fechamento.

Estratégias de contingência

Apesar das estratégias de prevenção contribuírem para a diminuição do ruído, elas não são suficientes para eliminá-lo por completo. Tornou-se necessária, então, a criação de uma estratégia de contingência para aumentar a robustez do algoritmo de busca do ponto de partida ao ruído.

Uma solução simples pode ser encontrada, apenas alterando a linha 30 do algoritmo. Na versão original a linha continha a seguinte condição:

```
30   if (cor(pi) !=  C) // cor de background
```

A condição acima verifica se a varredura já saiu da grade da ficha e chegou até o *background*, encontrando assim a borda superior. Devido ao ruído, a borda da grade ganha um aspecto "serrilhado", contendo pontos de background em meio aos pontos de objeto. Para resolver este problema, basta substituir a condição por:

```
30   if (bg(pi) = 1) // cor de background
```

Sabendo que:

Condição bg	
1	**entrada**: ponto [pi]
2	**saída**: 1 – se for ponto de background, 0 – do contrário
3	n = {8-vizinhos de pi} ∪ {pi}
4	**if** (x = (1- C), para todo x ∈ n) // (1 - C) cor de background
5	**retorna** 1
6	**else**
7	**retorna** 0
8	**end if**

Para satisfazer a condição bg acima, o ponto vai ter que estar cercado de pontos de *background*, evitando assim que pontos "intrusos" possam ser confundidos.

2. Dados inadequados

Alguns escritores, apesar das recomendações, acabavam gerando 2 problemas extras nas fichas: rasuras e caracteres encostando e até cruzando a grade. Exemplos nas imagens abaixo:

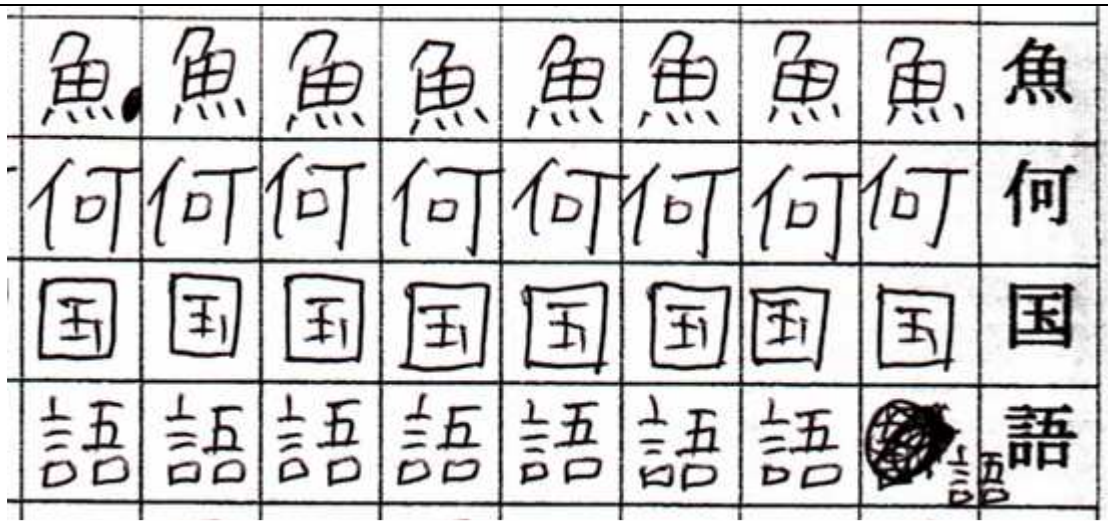

Figura 3.11 - Na imagem pode-se ver exemplos de rasuras graves e de caracteres encostando e até cruzando a grade.

Para as **rasuras**, não foi encontrada nenhuma solução automática, seja preventiva, ou de contingência. As imagens, então, foram editadas para a remoção das rasuras e, quando necessário, os caracteres rasurados foram substituídos por cópias de outros caracteres corretos.

Para os **caracteres que cruzam a linha**, também foi necessária edição manual, para que não interferissem na semântica dos seus vizinhos. Já os **caracteres que encostam na linha** não demandariam por nenhuma alteração brusca, a não ser os caracteres da primeira linha. A imagem abaixo ilustra o problema:

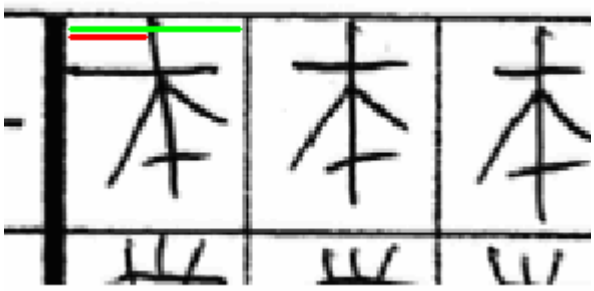

Figura 3.12 - A linha verde representaria a trajetória natural da varredura, no algoritmo de busca de valores horizontais para cropping. Mas devido ao traço estar encostado na grade, o algoritmo acaba esbarrando nele e tratando-o como a próxima borda.

Para amenizar esse tipo de problema, um "desvio" é calculado no quadrado que contém o primeiro caracter de referência e utilizado nos quadrados seguintes. Logo, ao invés de percorrer a direção horizontal pixel a pixel, a partir do segundo quadrado é dado um incremento inicial grande, ignorando possíveis barreiras no centro. Vide Figura 3.13 abaixo:

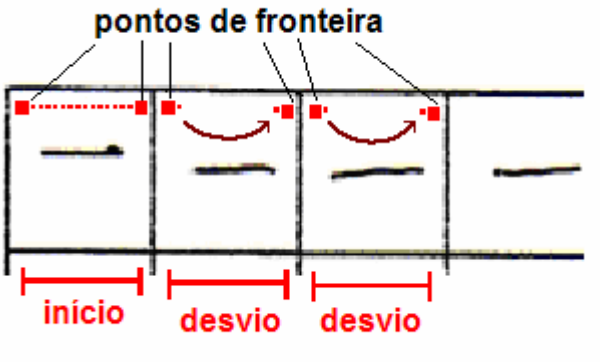

Figura 3.13 - O primeiro quadrado (início) é percorrido pixel a pixel para o cálculo do desvio. Nos próximos, os pixels do centro são "pulados" somando-se o valor do desvio à variável que está varrendo a imagem horizontalmente. O algoritmo não verifica os pixels do centro.

3. Inclinação

Apesar dos esforços para escanear e imprimir as fichas corretamente, algumas fichas ainda persistiam em ter um problema de inclinação. Esse pode ser considerado o maior problema encontrado, pois não pode ser contornado com filtros simples de processamento de imagem e também não pode ser resolvido facilmente com edição manual de imagens. Ainda assim, algumas estratégias para contorná-lo foram tentadas:

Estratégia de prevenção

Antes de escaneadas, as fichas foram aparadas lateralmente, evitando assim que inclinações da ficha em relação ao papel, em sua maioria, fossem captadas pelo scanner. Para as fichas que foram recebidas em papel, essa estratégia foi muito útil. No entanto, algumas fichas foram recebidas virtualmente, tornando impossível a aplicação dessa técnica.

Estratégia de contingência

Uma alteração em cada algoritmo de busca de valores para cropping é suficiente para lidar com pequenas inclinações. A idéia é guardar, além dos valores de x, um incremento do y para cada x. O mesmo é feito na vertical, além de guardar o valor de y, guardar um valor de incremento para x. Esse incremento indica o quanto o y (ou x, para o caso vertical) atual está abaixo do y original (ou x original), funcionando como uma informação de inclinação. O incremento é inicializado com 0 (zero) e é incrementado toda vez que o ponto sendo analisado satisfaça a seguinte condição:

Condição Topo	
1	**entrada**: ponto [pi]
2	**saída**: 1 – se for ponto do topo do quadrado, 0 – do contrário
3	pj = (pi.x , pi.y + 1) // ponto logo abaixo de pj
4	**if** (cor(pi) = C and cor(pj) = (1- C))
6	**retorna** 1
7	**else**
8	**retorna** 0
9	**end if**

Ao encontrar um pixel de topo, o incremento é aumentado e o algoritmo segue em frente. O valor do incremento é propagado para todos os próximos pixels. Para o caso vertical, a condição topo torna-se uma condição Borda Esquerda que checa se o pixel é um ponto lateral da grade, quando a busca quer um pixel inferior. A linha 3 passa, então, a ser:

```
pj = (pi.x+1, pi.y +1)
```

A Figura 3.14 ilustra a operação:

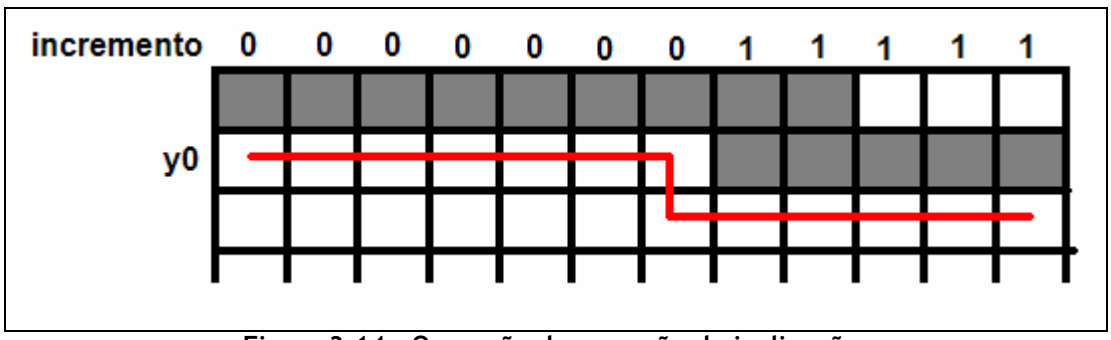

Figura 3.14 - Operação de correção de inclinação.

4. Qualidade das imagens geradas

Para analisar a qualidade das imagens geradas, o critério escolhido pode ser muito subjetivo. Não é possível atestar que uma imagem é melhor que outra sob todos os aspectos. Foram feitos diversos testes, e gerados bancos com propriedades diferentes. Algumas das técnicas utilizadas são mencionadas a seguir:

- Suavização e Fechamento – descritas na seção anterior;
- Abertura, Erosão e Dilatação;
- Esqueletização;
- Redimensionamento (Normalização).

3.5 Armazenamento do Banco

O banco foi armazenado em um sistema de arquivos seguindo a seguinte estrutura de pastas:

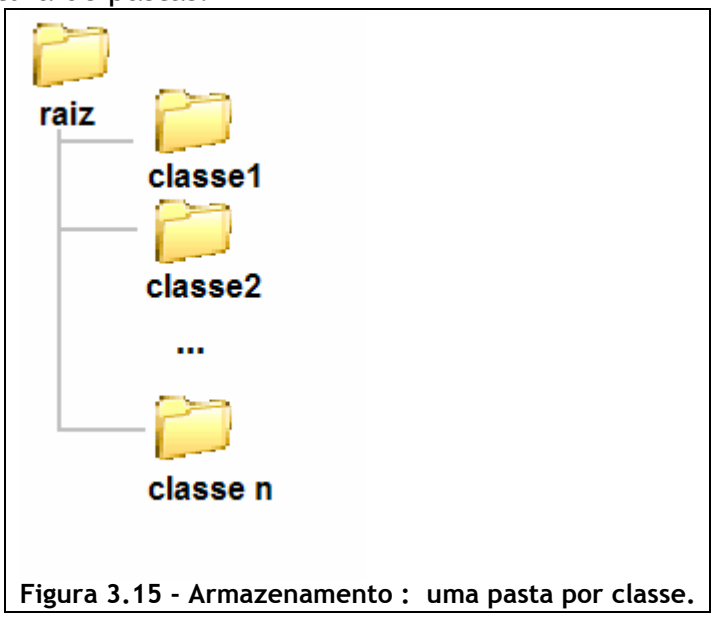

Figura 3.15 - Armazenamento : uma pasta por classe.

A princípio, pensou-se em armazenar as imagens do banco em formato JPEG. Mas por se tratar de um formato com perda de informação, ele foi substituído pelo PNG. Isso foi feito para evitar que o ligeiro ruído inserido pelo JPEG não se torne um fator negativo para a performance da classificação total.

4 Experimentos e Resultados

4.1 N-Fold Cross-Validation

Para a validação do classificador implementado, foi utilizado o método de N-Fold Cross Validation [18]. O método consiste em dividir o conjunto de dados em N partes e, em seguida, utilizar uma das partes como base de testes e as restantes N-1 como base de treinamento. Isso é feito N vezes, de forma que cada parte funcione como base de testes uma vez.

É importante ressaltar que é necessário manter o balanceamento de classes dentro de cada parte. Deve-se ter o cuidado para não colocar todas as instâncias de uma classe em uma parte só, por exemplo. Logo, se o conjunto total é balanceado, ou seja, com a mesma quantidade de elementos por classe, cada parte deve ser igualmente balanceada.

4.2 Experimentos

Os experimentos tiveram um caráter investigativo. Alguns parâmetros de implementação foram verificados, e avaliados como ganho ou não de performance no classificador. São eles:

- Esqueletização
- KNN – Número de vizinhos
- Medida de distância

Os experimentos utilizaram o banco de imagens próprio gerado. O banco, ao todo, possui 100 classes de caracteres, com exemplos de 20 escritores, totalizando 180 caracteres por classe, com um total de 18000 caracteres. Para os testes iniciais, foram utilizadas as 50 primeiras classes e 9 escritores, ou seja, 81 caracteres por classe, totalizando 4050 caracteres.

Para os testes, a banco de dados foi gerado e processado em Python, para geração dos vetores de características de cada imagem. As instâncias foram, então, convertidas para um formato ARFF e validadas no software Weka [15].

A princípio, as imagens foram ajustada utilizando a técnica de ImageTrim e normalizadas para um tamanho 16x16 pixels. Utilizando o extrator *Sobel Edge-Masked Maps*, foram gerados vetores de características de tamanho 1024. As instâncias foram, então, classificadas utilizando o KNN com distância euclidiana, variando os valores de K. (vide Tabela 4.1) Observou-se que o KNN com K = 1 obteve o melhor desempenho.

Tabela 4.1 - KNN tradicional com distância euclidiana.

	1NN	**3NN**	**5NN**
Taxa de acerto	74.42	61.85	67.46

Taxa de erro	25.58	38.15	32.54
# de instâncias classificadas corretamente	3014	2505	2732
# de instâncias classificadas incorretamente	1036	1545	1318

Foi avaliado também o uso do KNN incluindo um peso de (1/distância) para cada um dos k-vizinhos, para tornar a escolha da classe mais eficiente.

Tabela 4.2 – KNN com/sem ponderação dos vizinhos.

	1NN	**3NN**		**5NN**	
	--	**3NN (sem pesos)**	**3NN (com pesos)**	**5NN (sem pesos)**	**5NN (com pesos)**
Taxa de acerto	74.42	61.85	73.06	67.46	71.26
Taxa de erro	25.58	38.15	26.94	32.54	28.74
# de instâncias classificadas corretamente	3014	2505	2959	2732	2886
# de instâncias classificadas incorretamente	1036	1545	1091	1318	1164

Ao acrescentar uma etapa de esqueletização, antes da extração de características, obtiveram-se os resultados mostrados na Tabela 4.3. De maneira geral a utilização de imagens com esqueletização obteve um desempenho inferior ao da utilização de imagens completas.

Tabela 4.3 - KNN com etapa de esqueletização no pré-processamento.

	1NN	**3NN**		**5NN**	
	--	**3NN (sem pesos)**	**3NN (com pesos)**	**5NN (sem pesos)**	**5NN (com pesos)**
Taxa de acerto	66.54	61.85	66.17	61.19	64.89
Taxa de erro	33.46	38.15	33.83	38.81	35.11
# de instâncias classificadas corretamente	2695	2505	2680	2478	2628
# de instâncias classificadas incorretamente	1355	1545	1370	1572	1422

Tabela 4.4 – Taxas de acerto para KNN com/sem esqueletização.

	1NN	**3NN**	**5NN**
Sem esqueletização	**74.42**	73.06	71.26
Com esqueletização	66.54	66.17	64.89

Em seguida, foi avaliada a medida de distância proposta. Para tal, a princípio é necessário calcular os pesos para cada vetor de direção calculado pelo extrator de Sobel. As instâncias foram então, classificadas utilizando somente 1 dos 4 vetores por vez, (Vide Tabela 4.5) determinando assim os pesos utilizados.

Tabela 4.5 - Tabela comparativa - taxas de acerto para cada um dos 4 mapas de Sobel.

	1NN	**3NN**	**5NN**
Característica Horizontal	59.92	53.09	51.55
Característica Vertical	55.90	50.05	49.48
Característica Diagonal Direita	70.27	64.84	63.09
Característica Diagonal Esquerda	55.90	64.34	61.80

Como, apesar das diferenças de resultados entre a classificação com e sem esqueletização, o desempenho do 1NN prevaleceu em todos os casos. Logo, para o cálculo dos pesos, os resultados da classificação do 1NN foram utilizados.

Tabela 4.6 - Cálculo dos pesos para distância ponderada.

	Taxa de acerto (%)	Pesos proporcionais à taxa	
Característica Horizontal	59.92	0,25	w_H
Característica Vertical	55.90	0,23	w_V
Característica Diagonal Direita	70.27	0,29	w_{DD}
Característica Diagonal Esquerda	55.90	0,23	w_{DE}

Apesar do cálculo dos pesos ter sido feito, devido a problemas na integração com a ferramenta Weka, a avaliação utilizando o cálculo da medida de distância ponderada não pôde ser efetuado.

4.3 Análise dos Resultados

Quanto à configuração do KNN, observou-se que o 1NN obteve o melhor resultado de uma maneira geral. Enquanto não era utilizada uma ponderação dos vizinhos pela distância, a diferença entre o 1NN e os KNN com K > 1 era bastante considerável.

A esqueletização não mostrou um bom desempenho nos experimentos realizados. Isso, provavelmente, se deve à técnica de extração de

características utilizada. Por ser uma técnica que não valoriza a estrutura do objeto, mas sim a presença de linhas em determinadas direções, aplicar a esqueletização pode ter causado uma perda de informação não desejada.

Ao se analisar a matriz de confusão gerada pelo experimento utilizando o 1NN, pode-se visualizar alguns casos de caracteres que estão sendo classificados erroneamente de forma mais recorrente. As figuras abaixo ilustram alguns exemplos:

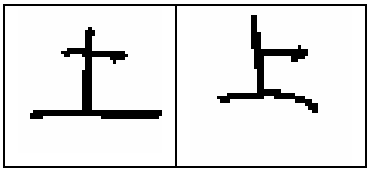

Figura 4.1 - Confusão entre os kanji 土 (“terra”) e 上 (“em cima, sobre”).

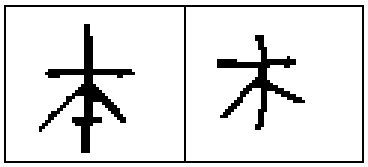

Figura 4.2 - Confusão entre os kanji 本 (“livro”) e 木 (“árvore”).

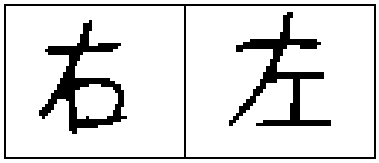

Figura 4.3 - Confusão entre os kanji 右 (“esquerda”) e 左 (“direita”), comum, inclusive, entre seres humanos.

5 Conclusão e Trabalhos Futuros

5.1 Conclusão

Foi feito um estudo de todo o processo de reconhecimento de caracteres japoneses. A pesquisa partiu do desenvolvimento de um banco de imagens próprio, passou pela implementação de diversas técnicas de processamento de imagens e reconhecimento de padrões e, enfim, gerou alguns experimentos e resultados.

Utilizando a base própria gerada, verificou-se que a performance da classificação utilizando a técnica de esqueletização no pré-processamento foi insatisfatória em relação à que não a utilizou. Isso se deveu, provavelmente, à combinação dela com a técnica de extração de características utilizada.

O uso de pesos para ponderar os vizinhos no K-NN, no entanto, provou-se eficiente na melhoria da taxa de acerto dos experimentos que procuravam por uma vizinhança de cardinalidade não unitária. Ainda assim, a vizinhança unitária, o 1NN, ainda continuou com o melhor resultado. A medida de distância ponderada pelas taxas de acerto dos planos direcionais, infelizmente, não pôde ser testada.

A performance apresentada nos experimentos ainda está muito aquém da exposta em alguns artigos. Não se pode, no entanto, fazer um estudo comparativo direto, pois as bases de dados utilizadas são diferentes. Pretende-se, *a posteriori*, realizar testes com as bases já existentes para uma melhor avaliação dos resultados.

Ainda há muito a ser desenvolvido neste trabalho, seja consertando problemas de integração, performance e geração de imagens para o banco, implementando novas técnicas e abordagens ou concluindo uma versão da interface gráfica, para reconhecimento de caracteres desenhados pelo usuário.

5.2 Trabalhos Futuros

5.2.1 Extração de Características

Um caminho a ser tomado neste momento é a escolha de outro método de extração de características. Algumas alternativas foram estudadas e estão listadas a seguir:

- LDA: Usar a própria imagem como vetor de características, sem redimensionamento para redução da dimensionalidade. O vetor de características seria, então, reduzido usando a técnica de LDA (Linear Discriminant Analysis) ou de PCA (Principal Component Analysis). A idéia das duas técnicas é encontrar uma projeção dos dados de forma a manter a variabilidade. Para o PCA, isso é feito apenas com base na variância total dos dados, já o LDA leva em conta, também, a

separabilidade das classes. O método descrito em [17] utiliza, dentre outro conjunto de técnicas, o LDA para a redução de dados e também para contribuir na medida de distância.

- Extração de Características de Gabor: Este método utiliza Filtros de Gabor [12] para gerar os vetores características. Os filtros são convoluídos na imagem original e, de forma semelhante ao Sobel Edge-Masked, ele gera imagens em diversas direções e as utiliza como vetor de características.
- Extração de Características pelo Gradiente: Também é semelhante ao Sobel Edge-Masked Maps, mas utiliza uma máscara diferente para geração dos mapas.

5.2.2 Estimativa de valor de confiança

Uma técnica estudada foi a estimativa de um valor de confiança para as classes mais próximas de um determinado elemento. A técnica proposta em [6] recebe um vetor de distâncias $x = \{x_1,x_2,...,x_M\}$ onde cada x_i, $1<i<m$, representa a distância do i-ésimo elemento mais próximo do padrão a ser classificado, e atribui um valor de confiança para os x_is [Vide Figura 5.1].

Rank	Candidate	Dist	ACV	CV
1	女	4300	19.67%	19.67%
2	幸	4328	31.95%	12.28%
3	前	4368	37.59%	5.64%
4	市	4371	42.58%	5.00%
5	字	4413	47.91%	5.33%
6	丘	4425	50.87%	2.97%
7	年	4478	55.23%	4.37%
8	赤	4536	59.36%	4.13%
9	希	4567	62.31%	2.95%
10	立	4590	64.19%	1.89%

Figura 5.1 - Valores de confiança (CV) para distâncias de um determinado caracter. Fonte: [6]

O CV do primeiro elemento é calculado como a probabilidade de, dado um vetor x, o primeiro elemento ser o elemento correto. O evento que representa o i-ésimo elemento como o correto é denotado por Xi = C. Dado que M é o número de elementos mais próximos, a probabilidade pode ser, então, calculada por:

$$P(X_j = C \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid X_j = C)P(X_j = C)}{\sum_{k=1}^{M+1} p(\boldsymbol{x} \mid X_k = C)P(X_k = C)}$$

A $P(Xj = C)$ é facilmente calculada como a taxa de acerto do j-ésimo elemento. Ou seja, após classificadas todas as instâncias, em número de casos em que o j-ésimo elemento era o correto, dividido pelo número total de classificações. A $P(X_{M+1} = C)$ é a probabilidade do elemento correto não ser nenhum dos M primeiros e é o complemento das probabilidades restantes.

A outra componente, $p(\boldsymbol{x} \mid Xj = C)$, não possui um cálculo tão trivial. É uma função de densidade de probabilidade (PDF) não conhecida e que precisa ser estimada. Essa estimativa é feita através do algoritmo EM (Expectation-Maximization) [16]. O algoritmo EM é capaz de, dado um conjunto de dados, encontrar uma ou mais distribuições normais que representam aquele conjunto.

De posse das duas componentes é possível calcular um valor de confiança para cada um dos elementos mais próximos. O artigo avaliado não exige que os elementos sejam de classes distintas, embora todos os exemplos mostrados possuam um vetor de classes diferentes. Para implementar a estimativa de CV para vizinhos de classes diferentes, seria interessante implementar, também, uma técnica de seleção de protótipos.

5.2.3 Seleção de protótipos

Alguns dos testes realizados, devido ao tamanho da base de dados e do vetor de características, levaram muito tempo para serem concluídos. Para diminuir o tamanho da base e amenizar, por conseguinte, o problema, pode-se implementar uma técnica de seleção de protótipos. Dentre as várias técnicas de seleção existentes, escolheu-se o LVQ (Linear Vector Quantization) [7] que é uma técnica capaz de gerar poucos protótipos com uma boa representação do conjunto original.

O tempo para geração dos protótipos, no entanto, acabou sendo muito extenso e, devido ao prazo, os experimentos com a técnica foram abandonados. Pretende-se, portanto, fazer uma avaliação mais aprofundada das técnicas de seleção existentes, para que uma técnica mais adequada para o problema seja escolhida e implementada.

5.2.4 Bancos de dados

Em relação aos bancos de dados utilizados, há, claramente, 2 caminhos a serem seguidos. O primeiro é testar utilizando as bases ETL8 e ETL9, bases que possuem uma cardinalidade muito maior de classes e elementos. Além disso, há a possibilidade de melhorar o banco de dados próprio, aumentando o número de classes e de contribuintes ou modificando os algoritmos de geração do banco, se necessário. Após um refinamento do banco de dados próprio, sua disponibilização online seria, também, uma possibilidade.

5.2.5 Interface Gráfica

Foi inicializado o desenvolvimento de uma interface gráfica para capturar entradas de caracteres de um usuário para posterior classificação. A classificação seria feita com base em uma base de dados e conjunto de técnicas também definido pelo usuário. A interface foi desenvolvida em Java

e se conectava com o programa em python para processamento da imagem e geração do vetor de características. A interface ainda precisa de muitos ajustes e melhorias, principalmente no que tange à integração com a ferramenta Weka.

5.2.6 Correção de Problemas

A correção de alguns problemas encontrados no decorrer do processo, principalmente no uso da medida de distância ponderada também é uma vertente de continuação deste trabalho.

6 Referências

[1] "Bases de dados Nakayoshi e Kuchibue", Disponível on-line em <http://www.tuat.ac.jp/~nakagawa/ipdb/Welcome.html>, acesso em 26/06/2008.

[2] "ETL Character Databases", Disponível on-line em <http://www.is.aist.go.jp/etlcdb/#English>, acesso em 26/06/2008.

[3] R. C. Carrasco e Forcada ML, "A note on the Nagendraprasad-Wang-Gupta thinning algorithm", *Pattern Recognition Letters*, Vol. 16, no.5, pp. 539–541, 1995.

[4] "Lanczos Resampling", Disponível on-line em <http://en.wikipedia.org/wiki/Lanczos_resampling>, acesso em 26/06/2008.

[5] R. C. Gonzalez e Richard E. Woods. Digital Image Processing, 2ª ed., Prentice Hall, 2002.

[6] E. Ishidera, D. Nishiwaki e A. Sato, "A confidence value estimation method for handwritten Kanji character recognition and its application to candidate reduction", *International Journal on Document Analysis and Recognition*. Vol. 6, no. 4, pp. 263–270, 2004.

[7] "Learning Vector Quantization", Disponível on-line em <http://www.borgelt.net//doc/lvqd/lvqd.html>, acesso em 26/08/2008.

[8] S. Jaeger, C.-L. Liu e M. Nakagawa, "The state of the art in Japanese online handwriting recognition compared to techniques in western handwriting recognition", *International Journal on Document Analysis and Recognition*. Vol. 6, no. 2, pp. 75–88, 2003.

[9] I. Ota, R. Yamamoto, S. Sako e S. Sagayama, "Online Handwritten Kanji Recognition Based on Inter-stroke Grammar", *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pp. 1188-1192, 2007.

[10] M. Nakai, H. Shimodaira e S. Sagayama, "Generation of Hierarchical Dictionary for Stroke-order Free Kanji Handwriting Recognition Based on Substroke HMM", *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pp. 514-518, 2003.

[11] A. Kitadai e M. Nakagawa. "Prototype learning for structured pattern representation applied to on-line recognition of handwritten Japanese characters", *International Journal on Document Analysis and Recognition*. Vol. 10, no. 2, pp.101–112, 2007.

[12] C.-L. Liu, M. Koga, e H. Fujisawa. "Gabor Feature Extraction for Character Recognition: Comparison with Gradient Feature" *Proceedings of the Eight International Conference on Document Analysis and Recognition*, pp. 121-125, 2005.

[13] C.-L. Liu e K. Marukawa. "Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition". *Pattern Recognition*, Vol. 38, no. 12, pp. 2242–2255, 2005.

[14] C.-L. Liu, "Normalization-Cooperated Gradient Feature Extraction for Handwritten Character Recognition", *Ieee Transactions On Pattern Analysis And Machine Intelligence*, Vol. 29, no. 8, pp. 1465-1469, 2007.

[15] "Weka Machine Learning Project", disponível on-line em <http://www.cs.waikato.ac.nz/~ml/index.html>, acesso em 26/06/2008.

[16] S. J. Russel e P. Norvig, *Artifical Intelligence: A Modern Approach*, Prentice Hall, 2003.

[17] C.-L. Liu, "High Accuracy Handwritten Chinese Character Recognition Using Quadratic Classifiers with Discriminative Feature Extraction." *Proceedings of the 18th International Conference on Pattern Recognition*, pp. 942–945, 2006.

[18] "Cross-validation". Disponível on-line em <http://en.wikipedia.org/wiki/Cross-validation>, acesso em 26/06/2008.

[19] C. K. Hiang, S. S. Erdogan, "A New Convolutional Map Feature Extraction for Character Recognition". *International Joint Conference on Neural Networks*, pp. 2887-2892, 1999.

7 Apêndice – Fichas de Captação

Nome: _____

一																			本
二																			学
三																			校
四																			先
五																			生
六																			父
七																			母
八																			年
九																			今
十																			毎
百																			週
千																			時
万																			半
円																			上
日																			中
月																			下
火																			外
水																			左
木																			右
金																			前
土																			後
人																			午
子																			東
男																			西
女																			南

Figura 7.1 - Ficha de captação - Primeira Página

北																			車
気																			名
天																			友
雨																			川
白																			山
大																			空
小																			花
高																			魚
長																			何
行																			国
来																			語
食																			目
飲																			口
入																			手
出																			足
立																			耳
見																			多
話																			少
聞																			店
読																			安
書																			古
言																			新
休																			会
分																			社
電																			道

Figura 7.2 - Ficha de captação - Segunda página

Orientador

_____

George Darmiton da Cunha Cavalcanti

Aluno

_____

Marconi Emanuel Madruga Filho