卒業論文

文字認識用辞書の構成に関する研究

東北大学工学部通信工学科 阿曽研究室

風越 直紀

平成8年3月26日

目 次

図目次

表目次

第１章

序論

1.1 本研究の背景と目的

昨今、情報処理は飛躍的に向上し、計算機の扱う情報量は増大の一途をたどっている。また、そこで扱われる情報の質もますます複雑化してきている。しかし、計算機の急激な発展に伴ない、マン・マシンインタフェースの立ち遅れが如実に現われてきた。我々が普段情報交換のために用いている印刷された文字などを自動的に計算機に入力できるならば、それは人間にとって訓練の必要がなく、容易な手段であるため、省力化につながる。また近年 OA 化が進み、ワープロ、ファクス、コピー機といったの事務機器などによる大量の文書が氾濫している。そこで、このような大量の文書を計算機によって保存、変更、検索させるような社会的要求が高まっている。高度情報化社会において、文字認識技術はその重みをより増してゆくものと考えられる。

文字認識についての研究は約三十年前から行なわれ、種々の成果が得られている。現在、特定の分野においては OCR(Optical Character Reader) が実用化されている。しかし、現状では認識率 100%の文書認識システムは完成されておらず、どうしても誤認識を生じてしまう。また、認識率を上昇させるために認識時間が非常にかかり、その時間短縮のために専用のハードウエアの設計を必要とするようなものもある。

現在、文字認識における問題点は、認識精度と認識速度の 2 つに大別される。このうち認識精度においては、これまでにも様々な研究が成され、より高い認識精度を持つようなアルゴリズムが模索されてきた。例えば阿曽ら [1] は完全に認識が可能であるような範囲を検討し、パターンマッチング法に正確さの検証法を付加できるという方向性を示した。ま

た、西川ら [2] はいくつかの特徴量を合成した新しい特徴量による認識実験を行ない、認識精度が大幅に向上したという結果を得ている。

しかし、認識時間短縮に重点を置いた研究というものはほとんど成されていないのが現状である。阿曽らの研究 [1] で認識の高速化に対する試算は成されているものの、実験による検討が必要である。また、仙田ら [3] は ETL9B を対象とした手書き文字認識において大規模マルチテンプレート手法と K 最近傍法を用いることによって秒間 4 文字強の認識結果を得ているが、辞書が主記憶上に納まらないほどの大きさとなるため、実際に利用するには改良の余地があると述べている。

本研究では、認識率を落とさずに全体の認識時間を削減することを目的とする。まず、認識のために必要である辞書の高速検索可能な構成法を提案し、その辞書作成のためのアルゴリズムについて述べる。そして、提案手法による認識実験を行ない、その有効性を確認する。

1.2 本論文の構成

第 1 章 序論であり、本研究の背景と目的について述べる。

第 2 章 文字認識アルゴリズムについて述べる。

第 3 章 高速検索可能な辞書の構成法を提案する。

第 4 章 前章で提案した手法を評価するために実験を行なう。具体的には、提案手法に従って辞書を構成し、その辞書を用いた認識実験を行ない、認識速度、認識精度を既存の手法と比較する。

第 5 章 結論であり、まとめと今後の課題について述べる。

第 2 章

文字認識

2.1 はじめに

文字認識とは、入力されたイメージが何という文字であるかを判断する処理である。文字認識の手法は、大きく次の二つに分けられる。

1. パターンマッチング法

2. 構造解析法

パターンマッチング法では、あらかじめ読もうとする文字の一つ一つにテンプレート (template: 原形) を用意しておく。それに対し、入力された未知の文字パターンを重ね合せていき、最も近いテンプレートの文字をその入力文字であると決定する。すなわち、パターンマッチング法は、パターン同士の重なり具合で評価し、認識を行なう。このため、文字の多少の変形やノイズに強く、計算機上で容易に高速に実現できるが、類似文字の識別は難しい。普通は、計算量削減のため、文字画像そのものではなく、文字画像から得られる特徴ベクトルを用いて認識を行う。

構造解析法は文字がどのように構成されているかを解析して総合的な判断を下す。具体的に言うと、構造解析法は線分の接続関係や位置関係などの文字構造に着目し、構造の類似性で認識を行なう。この手法は類似文字の多い漢字や、手書きなどによる変形が大きい文字の認識に有効である。しかし、特徴量の定義や抽出が難しく、また計算機上での処理に時間がかかる等、問題も多い。本研究では、認識時間削減という目的のため、時間のか

かる構造解析法ではなく、方向線素特徴量 [4] を用いたパターンマッチング法を用いた。図2.1に文字認識のアルゴリズムを示す。

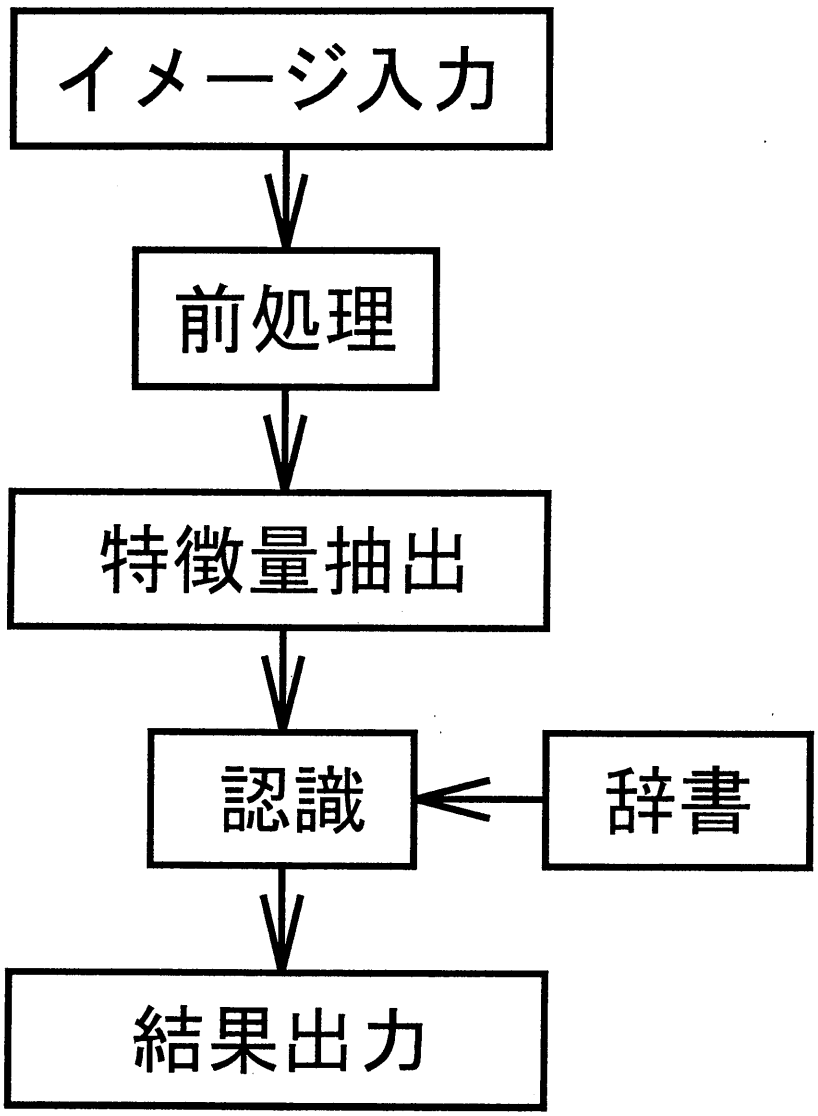


図 2.1: 文字認識のアルゴリズム

2.2 イメージ入力

イメージ入力とは、新聞・本・雑誌などの文書を二値画像として取り込む処理である。画像データはイメージスキャナによって入力される。入力された画像データは切り出しの処理により各文字ごとに切り出されたものとする。

2.3 前処理

文字を認識する場合の前処理とは、切り出されたイメージデータから特徴量を抽出するために必要な処理である。ここでは、前処理は、入力文字の線の輪郭を滑らかにするためのスムージング・ノイズ除去、文字の大きさの正規化、細線化と線素化の４つの処理によって構成されている。

図 2.2に各前処理後のイメージを示す。

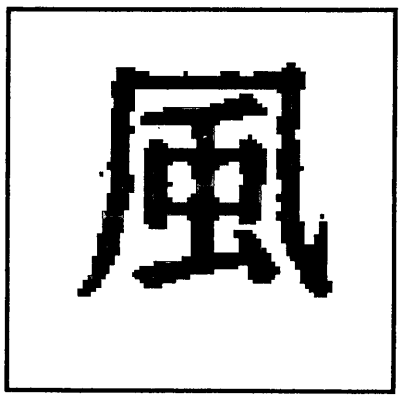
2.4 特徴量抽出

パターンマッチング法では、処理の高速化、パターン分離の効率化などのために、文字パターンを数値ベクトルに変換する。この過程を特徴量抽出という。

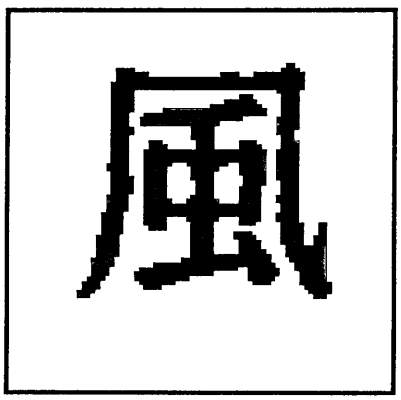
- 方向線素特徴量：

  方向線素特徴量の抽出法は図 2.3のように、まず、64 × 64 ドットの線素化画像の縦横を 8 ドット間隔に分割する。次に、16 × 16 ドットの領域を 49 個作成する (左上を 0 とし、8 ドットずつずらし、左から右、上から下へ順に並ぶ)。各領域ごとに縦、横、斜め 45 °、斜め 135 ° の４種類の方向 (線素と呼ぶ) の数を、重みつきでカウントし、４次元のベクトルを対応させる。これより、1 文字あたり 196(= 49 × 4) 次元のベクトルとなる。

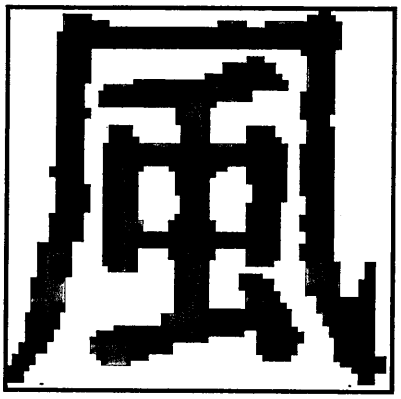
  1 領域内の重みは図 2.3のようになっている。



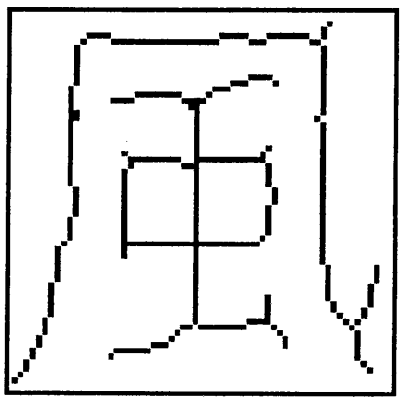
(1) イメージ



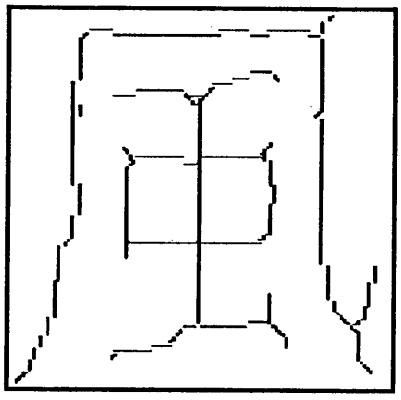
(2) ノイズ除去・スムージング



(3) 正規化

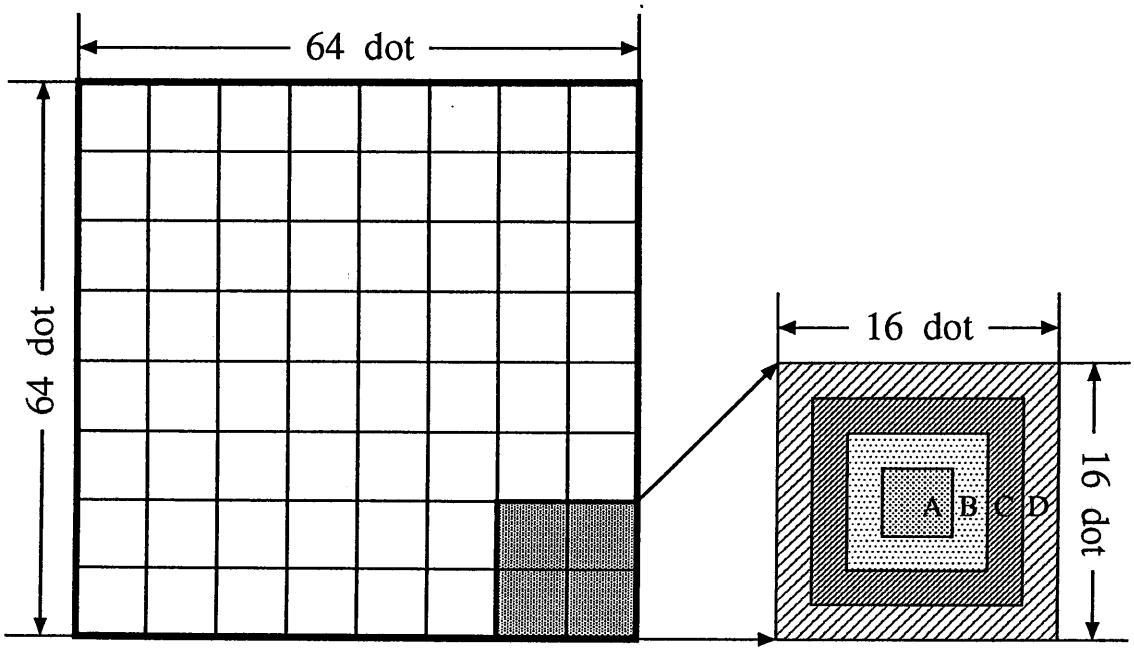


(4) 細線化



(5) 方向線素化

図 2.2: 各前処理後のイメージ



線素の種類：“ | ”, “—”, “／”, “＼”

一部分領域：16 × 16 ドット

部分領域数：7 × 7 = 49 個

次元数：4 × 49 = 196 次元

重みの値： 領域 A → 4 倍

領域 B → 3 倍

領域 C → 2 倍

領域 D → 1 倍

図 2.3: 特徴ベクトル作成時の重み

2.5 認識

ここでは最も一般的な全数整合法を説明する。全数整合法は、全認識対象文字の辞書ベクトルと特徴抽出で得られた未知入力文字の特徴量で評価値 (距離) を求め、小さいものから順に認識結果とする方法である。アルゴリズムの簡潔性やある程度の高い認識率を得られることから、一般的に用いられている。認識で用いられる辞書・評価値について以下で説明する。

2.5.1 辞書

辞書ベクトルを作成するとき、各字種ごとに、あらかじめ多数の学習パターンを用意しておき、そのパターンから求めた特徴ベクトルの平均をその字種の辞書ベクトルとする。

2.5.2 評価値

認識を行う際、近さの尺度 (評価値) としてはユークリッド距離の 2 乗を用いる。字種 k の辞書ベクトルを

$$\boldsymbol{m}^k = (m_1^k, m_2^k, ..., m_n^k) \tag{2.1}$$

とし、入力パターンから求めた特徴ベクトルを

$$\boldsymbol{v} = (v_1, v_2, ..., v_n) \tag{2.2}$$

とすると、評価値 e_k は、

$$e_k = \sum_{i=1}^{n} (v_i - m_i^k)^2 \tag{2.3}$$

となる。

2.6 候補出力

前節で求めた e_k の小さいものから候補字種とする。そして、その順に、第 1 位候補、第 2 位候補、… 、第 n 位候補として出力する。

第 3 章

高速検索可能な辞書の構成法

3.1 はじめに

本章では、認識精度をできるだけ落とさずに高速検索が可能な辞書データの構成法について考え、主成分分析法を用いた辞書クラスタリング手法を提案する。

辞書検索を高速化するにはいくつかの方法が考えられるが、今回提案する手法の基本的思想は検索すべき辞書空間を狭めるというものである。手順としては辞書空間に対して主成分分析を行ない、主成分方向に垂直な超平面で辞書空間を分割する。そして、分割された後の 2 つの辞書空間それぞれに対してこの作業を繰り返し、辞書空間 (以下、クラスタと呼ぶ) が十分小さくなった時点で分割を停止する。認識時には、まず入力された文字のベクトルに対して探索対象となるべきクラスタを決定し、このクラスタにある辞書ベクトルとのみ距離を計算する。もし最終クラスタ (分割終了時の最終層のクラスタ) に属する辞書ベクトルの数が全辞書ベクトルの 10 分の 1 になっていれば、単純計算では認識時間も 10 分の 1 になるはずである。

3.2 主成分分析

辞書ベクトル空間を分割する際には主成分分析法を用いる。ここでは、主成分分析法について簡単に述べる。

主成分分析法は、多くの変量の値をできるだけ情報の損失なしに主要な変動に要約して、特徴を把握する統計的手法である。一般に同一のサンプルについて何らかの相関関係があ

る n 種の変量 $(x_1, x_2, \ldots, x_n)$ の測定された a 組のデータ

$$x_{1\lambda}, x_{2\lambda}, \ldots, x_{n\lambda} \qquad (\lambda = 1, 2, \ldots, n) \tag{3.1}$$

が得られたとする。これらのデータが n 変量相互に関係のある変動を示していると考えて、n 変量の一次結合

$$z = l_1 x_1 + l_2 x_2 + \cdots + l_n x_n \tag{3.2}$$

を考え、$l_1, l_2, \ldots, l_n$ を変えて $\sum_{i=1}^{n} l_i^2 = 1$ の条件の下で z の分散が最大になるときの z を第一主成分という。これを z_1 と書き、係数を $l_{1i}(i = 1, 2, \ldots, n)$ で表わすと、次のようになる。

$$z_1 = l_{11} x_1 + l_{12} x_2 + \cdots + l_{1n} x_n \tag{3.3}$$

次に z_1 とは無相関な z のうちで $\sum_{i=1}^{n} l_i^2 = 1$ の条件を満たす最大の分散を持つ z_2 を決定する。これを第二主成分といい、次のように書ける。

$$z_2 = l_{21} x_1 + l_{22} x_2 + \cdots + l_{2n} x_n \tag{3.4}$$

以下同様にして第三、第四、…、第 n 主成分が決定される。このとき、各主成分はベクトル

$$\boldsymbol{l}_m = \begin{bmatrix} l_{1m} \\ l_{2m} \\ \vdots \\ l_{nm} \end{bmatrix} \tag{3.5}$$

とベクトル変量

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3.6}$$

を用いて次のように表わせる。

$$z_m = \boldsymbol{l}_m' \boldsymbol{x} \qquad (m = 1, 2, \ldots, n) \tag{3.7}$$

これらの $\boldsymbol{l}_m$ は互いに直交する単位ベクトルで、それぞれ第 m 主成分ベクトルと呼ぶ。また、その方向をそれぞれ第 m 主成分方向と呼ぶ。以下、第一主成分ベクトルを単に主成分ベクトルと呼び、その方向を主成分方向と呼ぶことにする。

本手法では、n が特徴量の次元数 (本手法で使用する方向線素特徴量では n=196)、a が注目するクラスタ内の全辞書ベクトル数に当たる。

3.3 辞書クラスタリングの流れ

前節で述べた主成分分析法を用いてクラスタの分割を行なう。主成分分析法を用いた理由としては

- 辞書空間の分散が最も大きい方向に垂直に分割を行なうことで各辞書ベクトルが認識すべき範囲 (各字種のベクトル空間) の重なりが少なくなることが予想される
- 計算機における計算方法が確立している

が挙げられる。

分割超平面は、クラスタ内の各辞書ベクトルの重心を通り、かつ主成分ベクトルに垂直な超平面とする。こうすることで、分割の際にクラスタ内の辞書ベクトルがほぼ半分ずつ次層のそれぞれのクラスタに入ることが予想され、クラスタの最深層までの層数が最も小さくなる (認識時間が短縮される) と考えられる。

実際には、分割超平面 Pは 1 つのスカラで以下のように表現される。ここで、Nはクラスタ内の辞書ベクトル数、$\boldsymbol{d}_i$はクラスタ内の i 番目の字種の辞書ベクトル、$\boldsymbol{v}_c$はクラスタの主成分ベクトルである。

$$P = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{d}_i \cdot \boldsymbol{v}_c) \tag{3.8}$$

しかし、一超平面を基準とし、クラスタ内の辞書ベクトルがそのどちら側にあるかという条件のみで辞書ベクトルを次層のクラスタに分配すると、分割超平面付近に辞書ベクトルがある字種については誤認識が増える恐れがある (図 3.1に示す)。そこで、分割時に各クラスタに重なりを持たせることを考える。具体的には、辞書作成に用いた各字種の全ベクトルを調べ、各字種のベクトル空間の位置と範囲を求め、それが分割超平面にまたがって分布しているならその字種は分割後両方のクラスタに含ませるという作業を行なう。これにより、少なくとも辞書作成に用いた文字については新たに誤認識が生じないことが保証される。この様子を図 3.2に示す。

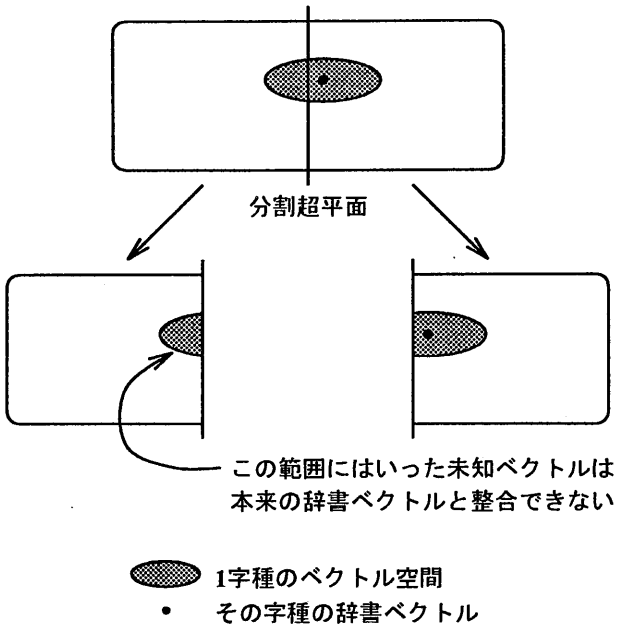


図 3.1: 分割によって誤認識が起こる例

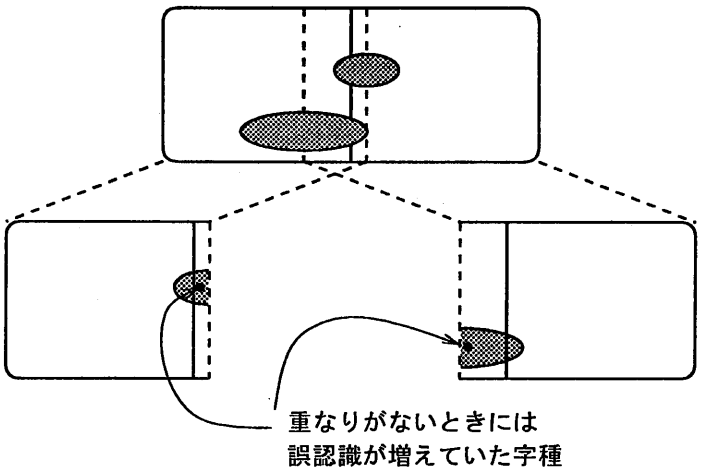


図 3.2: 重なりを持たせた分割

この作業を分割終了条件が満たされるまで再帰的に繰り返し、最終的には図3.3のようなクラスタの二分木構造が作られることになる。

分割の終了条件は次のどちらかが満たされることとする。

(1) $N < K_1$

(2) $\max(l_1, l_2) > K_2$

ここで、n は注目しているクラスタにある辞書ベクトル数、l_iはもしそのクラスタを分割した場合、次層のクラスタの双方に含まれるべき辞書ベクトル数をそれぞれ N_1, N_2 としたときに

$$l_i = \frac{N_i}{N} \qquad (i = 1, 2) \tag{3.9}$$

で表わされる数である。この l_iを以下重複度と呼ぶ。また、$K_i (i = 1, 2)$ は最適な分割ができるようにするためのパラメータである。

また、分割終了時に各クラスタが持っている情報は次のようになる。

- 次の層のクラスタ (子クラスタ) がある → 分割超平面 (実際には1スカラ)、そのクラスタの主成分ベクトル
- 次の層のクラスタがない → そのクラスタ内の辞書ベクトルの字種番号

本章で提案した辞書クラスタリングの流れを図3.4に示す。

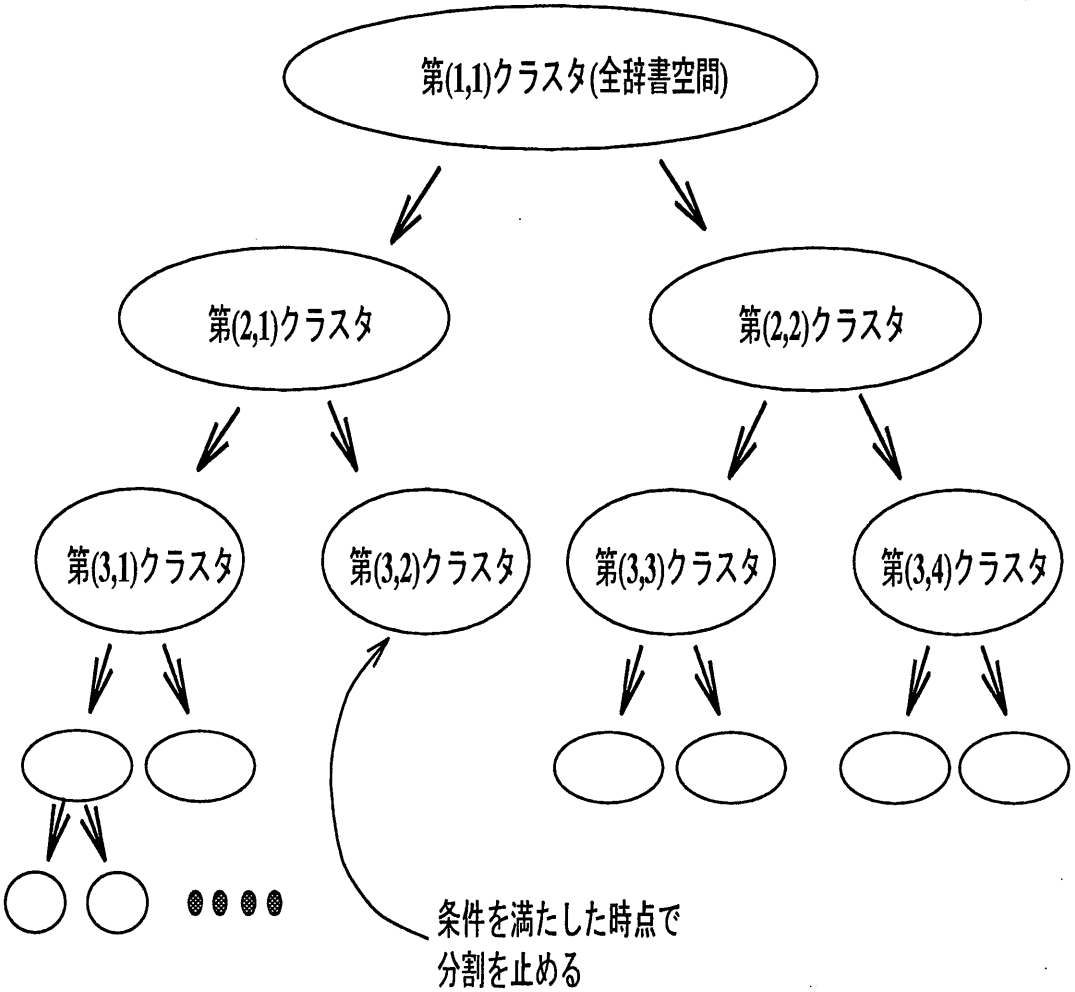


図 3.3: クラスタのデータ構造

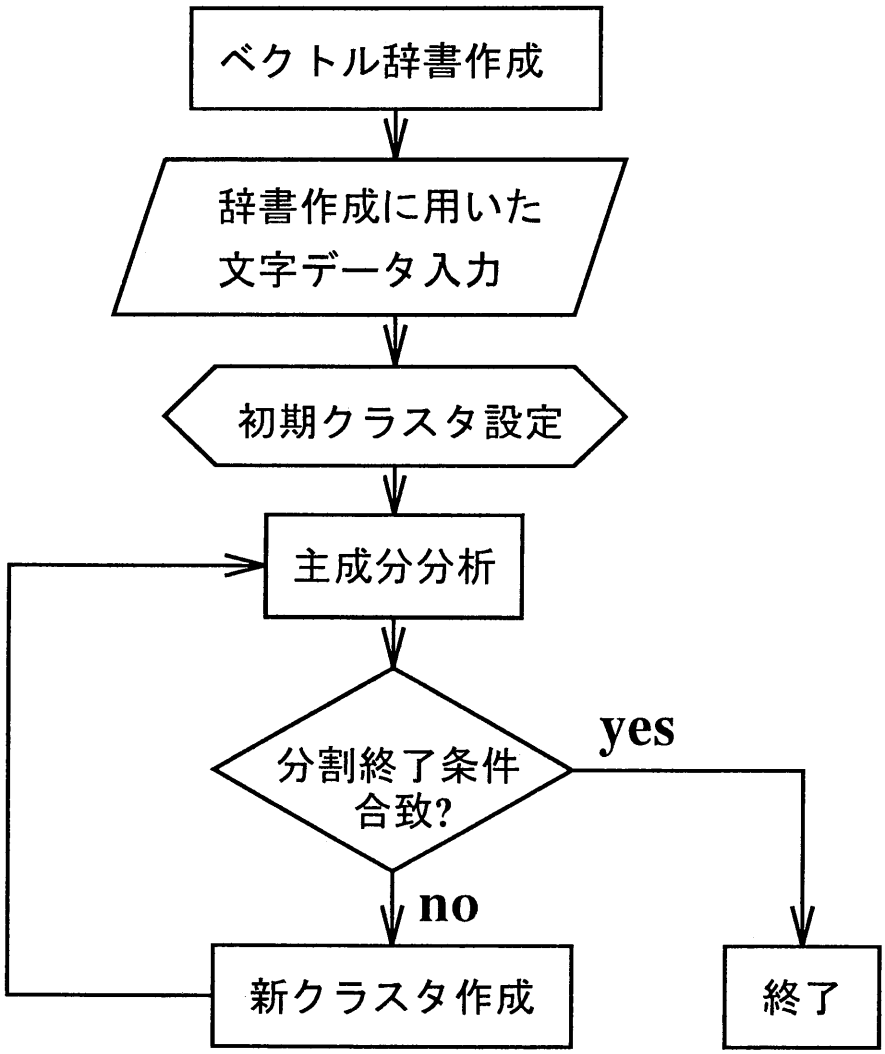


図 3.4: クラスタリングの流れ

3.4 認識の流れ

認識時はまず入力ベクトルが第1層のクラスタ(全辞書ベクトル空間)の分割超平面のどちら側にあるかを調べ、次層のクラスタのどちらに属するかを決定する。具体的には、入力ベクトル$\boldsymbol{x}$とクラスタの主成分ベクトル$\boldsymbol{v}_c$との内積$\boldsymbol{x}\cdot\boldsymbol{v}_c$と分割超平面を表わす値$P$を比較し、その大小によって判断する。本研究では、クラスタの配置を階層数sとその層にあるクラスタの番号tで(s,t)と表現し(図3.3参照)、

$\boldsymbol{x}\cdot\boldsymbol{v}_c \leq P$
$\Rightarrow$ 次層左側のクラスタ$(s+1, 2t+1)$に属する

$\boldsymbol{x}\cdot\boldsymbol{v}_c > P$
$\Rightarrow$ 次層右側のクラスタ$(s+1, 2t+2)$に属する

としている。

この作業を最終層のクラスタに到達するまで繰り返し、最終層のクラスタに到達した時点でそのクラスタ内にある辞書ベクトルとのみ距離(式(2.3)参照)を計算して出力を行なう。分割時にクラスタに冗長性を持たせているため、認識時には単純なアルゴリズムを用いることができる。

3.5 提案手法を用いた試算

提案手法を用いた場合、認識時にどのくらい計算量が削減されるかを考える。全字種数をU、最終層のクラスタ内字種数$K_1 = \frac{1}{10}U$とし、重複度は常に$l = 0.75$であると仮定する。

このように仮定すると、1回の分割においてクラスタ内の辞書ベクトル数はその1層前のクラスタのものの0.75倍になる。これを繰り返してクラスタ内の辞書ベクトル数が全辞書ベクトルの$\frac{1}{10}$になる層数sは

$$(0.75)^s = \frac{1}{10} \tag{3.10}$$

より、

$$s \simeq 8 \tag{3.11}$$

となる。すなわち第8層までの分割が行なわれると入力ベクトルに対する候補数は約$\frac{1}{10}$になるということが分かる。

このクラスタ情報を用いると、まず入力ベクトルが最終層に到達するまでの計算は7回、そして最終クラスタでの整合回数は$\frac{1}{10}U$回になる。単に全数整合を行なった場合の計算回数はU回であるから、全体として計算回数の比は

$$\frac{\frac{1}{10}U+7}{U} \tag{3.12}$$

となる。実際は$U \gg 7$である場合がほとんどであるので、認識時間は全数整合に比べて約$\frac{1}{10}$になると考えられる。

第 4 章

実験

4.1 はじめに

前章で提案した高速検索可能な辞書の構成手法の有効性を評価するために実験を行なう。具体的には、全辞書ベクトルをクラスタリングし、そのデータを用いて認識実験を行ない、認識速度と認識精度を既存の手法と比較、検討する。比較対象としては、全辞書ベクトルに対して全数整合を行なうようなプログラムとした。

4.2 予備実験

字種ごとのベクトル空間の分布状況を把握するため、字種を 231 に限定してクラスタリング実験及び認識実験を行なった。以下に実験の条件を挙げる。

- 調査対象：写植用明朝体 14 種類 (数字・アルファベット・平仮名・片仮名)
- 特徴量：方向線素特徴量
- 認識時の距離尺度：ユークリッド距離
- 実行環境：SunSPARCstation10 互換機

4.2.1 クラスタリング実験

クラスタ内の辞書ベクトルの最大数 K_1について、$K_1 = 20$ と $K_1 = 10$ の 2 つの場合について実験を行なった。その結果を表 4.1に示す。ただし重複度の最大数 K_2は 1.0 とした。

K_1	作成されたクラスタ数	最深層数	平均重複率
20	131	8	0.632
10	337	10	0.680

表 4.1: 231 字種でのクラスタリング実験の結果

4.2.2 認識実験

231 字種用の認識プログラムを作成し、前節で作成したクラスタ情報を用いた場合と用いなかった場合のそれぞれについて認識実験を行ない、認識時間と認識精度を求めた。以下にその結果を示す。

認識ファイル	sys. time (sec)	user time (sec)	累積認識率		
			1	2	3
教科書体	0.2	29.8	0.723	0.853	0.862
明朝体 1	0.3	30.2	0.957	0.996	1.000
明朝体 2	0.2	29.7	0.922	1.000	1.000

表 4.2: クラスタ情報を使用しない場合

認識ファイル	sys. time	user time	累積認識率		
			1	2	3
教科書体	0.2	2.5	0.693	0.800	0.801
明朝体 1	0.2	2.7	0.961	0.996	1.000
明朝体 2	0.1	2.6	0.926	1.000	1.000

表 4.3: $K_1 = 20$ の場合

認識ファイル	sys. time	user time	累積認識率		
			1	2	3
教科書体	0.2	1.7	0.680	0.779	0.784
明朝体 1	0.1	1.7	0.961	0.996	1.000
明朝体 2	0.2	1.6	0.926	1.000	1.000

表 4.4: $K_1 = 10$ の場合

4.2.3 考察

まずクラスタリングの結果であるが、$K_1 = 10$、$K_1 = 20$ のどちらの場合でもクラスタの二分木構造が作成され、最終層のクラスタのほとんどが $N < K_1$の条件を満たすものであった。すなわち、重複度が 1 になってしまいクラスタ内の字種数が多いにもかかわらず分割を停止したものがほとんどないということが言える。よって、その情報を用いて認識を行なった場合に認識時間が K_1によって操作可能になると考えられ、良好な結果であると思われる。

認識実験の結果を見ると、認識時間は従来の方法と比べて相当短縮できていることが分かる。これは参照すべき辞書ベクトルの数にほぼ比例するであろうと思われる。また、明朝体においてはクラスタに含まれる字種の最大数 K_1 を小さくしたときでも認識率の低下が起こらず、逆に上昇さえ見られるという結果が得られた。これはクラスタリング時に、誤認識の原因となる類似文字のうちのいくつかが分離されているためと考えられる。この結

果を見ると、この手法は認識時間短縮にかなりの有効性を持つものと考えられる。これをふまえ、次の実験を行なった。

4.3 3303 字種での実験 (1)

予備実験での結果をふまえて3303 字種で同様の実験を行なった。実験の条件を以下に挙げる。

- 調査対象：写植用明朝体 13 種類
- 特徴量：方向線素特徴量
- 認識時の距離尺度：ユークリッド距離
- 実行環境：SunSPARCstation10 互換機
- 許容最大重複度 K_2：0.95

何回かの実験の結果、$K_2 = 1.0$ では作成されるクラスタの数が膨大になり作業に時間がかかりすぎることと主記憶容量を圧迫することが分かったため、この実験から K_2の値を変化させた。

4.3.1 クラスタリング実験

クラスタ内の辞書ベクトルの最大数 K_1について、$K_1 = 300$ と $K_1 = 100$ の 2 つの場合について実験を行ない、作成されたクラスタ数と重複度を把握した。その結果を表 4.5に示す。

K_1	作成されたクラスタ数	最深層数	平均重複率
300	139	8	0.672
100	1169	13	0.714

表 4.5: 3303 字種でのクラスタリング実験の結果

4.3.2 認識実験

前節での実験で作成したクラスタ情報を用いて認識実験を行なった。以下に結果を示す。

認識ファイル	sys. time (sec)	user time (sec)	累積認識率		
			1	2	3
教科書体	0.7	1:28:04	0.882	0.938	0.953
明朝体1	0.6	1:27:35	0.991	0.996	0.997
明朝体2	0.5	1:27:39	0.989	0.995	0.996

表 4.6: クラスタ情報を使用しない場合

認識ファイル	sys. time	user time	累積認識率		
			1	2	3
教科書体	0.4	7:48	0.676	0.712	0.718
明朝体1	0.5	7:48	0.976	0.980	0.981
明朝体2	0.5	7:50	0.979	0.983	0.984

表 4.7: $K_1 = 300$ の場合

認識ファイル	sys. time	user time	累積認識率		
			1	2	3
教科書体	1.6	2:48.4	0.614	0.641	0.646
明朝体1	1.2	2:46.8	0.963	0.967	0.968
明朝体2	1.1	2:45.9	0.970	0.974	0.975

表 4.8: $K_1 = 100$ の場合

4.3.3 考察

クラスタリングの結果、231 字種の場合とは違ってかなりデータ構造が大きくなったことが分かる。ただし認識時間には全体のクラスタ数より K_1 と最大層数が影響するため、クラスタ数がある程度多くなっても認識時間が増大し過ぎることはないと思われる。しかし実際問題として計算機の主記憶容量を圧迫し過ぎてはならないため、それに応じて K_2 を操作することが必要であろう。

認識実験の結果を見ると、確かに認識時間の大幅な短縮はなされているものの認識率が相当落ちてしまっている。特に未知フォントである教科書体の認識率は劣悪である。理由としては、辞書作成に用いたフォント数が少なく、かつ単一フォントのみを扱っているため、分割超平面にまたがって分布している字種を重ねきれていないことが挙げられるであろう。そこで、分割によって受動的に決定されていた重複度を任意に操作し、より重なりを持たせて分割することが必要であると考えられる。次節でその手法について述べる。

4.4 より重なりを持たせた分割手法

前節までの実験で重複度を任意に操作する必要性が出てきたため、その手法について考える。今回は単純ではあるが、分割超平面付近に辞書ベクトルがある字種については無条件で次層の両方のクラスタに含ませるという方法をとった。「分割超平面付近」の範囲は式(4.1)で表現される。

$$\underline{\text{分割超平面付近の範囲}：[P - C \times \sigma, P + C \times \sigma]} \tag{4.1}$$

ここで、Cは正の任意実定数、Pはクラスタ内の全辞書ベクトルの重心で分割超平面を表現する値 (式 (3.8)参照) である。また、σはクラスタ内の全辞書ベクトルと主成分ベクトルとの内積の標準偏差であり、次のように表わされる。

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{d}_i \cdot \boldsymbol{v}_c)^2 - P^2} \tag{4.2}$$

ただし、Nはクラスタ内の辞書ベクトル数、$\boldsymbol{d}_i$はクラスタ内の i 番目の字種の辞書ベクトル、$\boldsymbol{v}_c$はクラスタの主成分ベクトルである。

範囲として式(4.1)を考えた理由は、クラスタ内の辞書ベクトルが主成分方向に正規分布

を成していると仮定した場合、Cを変化させることで分割超平面からの範囲を任意の割合で指定できるということである (図 4.1参照)。

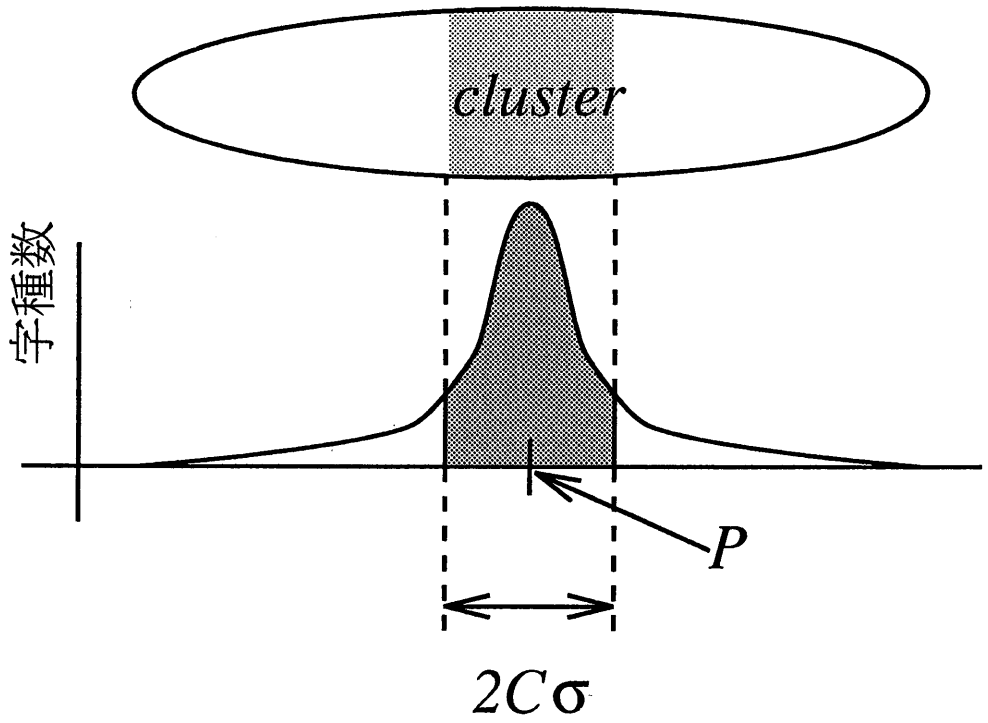


図 4.1: 強制的に重なりを持たせる範囲

4.5 3303 字種での実験 (2)

前節で導入したσの係数 Cを変化させてクラスタリング実験を行ない、その結果を用いて認識実験を行なった。ただし、Cについては 0.10 , 0.13 , 0.19 の 3 種類とした。これは、クラスタ内の辞書ベクトルの分布がその主成分ベクトル方向に対して正規分布を成している場合、それぞれ分割超平面付近の 8%,10%,15%が強制的に重なる値である。他の実験の条件は 4.3 と同様なものとした。

4.5.1 クラスタリング実験

クラスタ内の最高辞書ベクトル数 K_1 を 300 と 100 の二種類にし、クラスタリング実験を行なった。以下にその結果を示す。

C	作成されたクラスタ数	最深層数	平均重複率
0.10	225	9	0.717
0.13	453	11	0.737
0.19	1085	13	0.764

表 4.9: $K_1 = 300$ の場合

C	作成されたクラスタ数	最深層数	平均重複率
0.10	4213	13	0.761
0.13	6246	15	0.771
0.19	13525	18	0.785

表 4.10: $K_1 = 100$ の場合

4.5.2 認識実験

4.5.1で作成されたクラスタ情報を用いて再び認識実験を行なった。特徴的な結果として、$K_1 = 300, C = 0.19$ の場合と $K_1 = 100, C = 0.13$ の場合の認識時間と認識率を以下に示す。強制的重なりの度合である Cは大きなほど認識率が上昇すると考えられるが、$K_1 = 100, C = 0.19$ の場合ではクラスタ情報が大きくなりすぎ、主記憶容量が不足したために認識実験は行なえなかった。

認識ファイル	sys. time	user time	累積認識率		
			1	2	3
教科書体	2.5	6:46	0.782	0.823	0.831
明朝体1	1.8	6:46	0.986	0.990	0.991
明朝体2	1.8	6:47	0.985	0.989	0.990

表 4.11: $K_1 = 300, C = 0.19$ の場合

認識ファイル	sys. time	user time	累積認識率		
			1	2	3
教科書体	8.3	3:11	0.710	0.745	0.751
明朝体1	9.2	3:05	0.978	0.982	0.983
明朝体2	8.5	3:06	0.982	0.986	0.987

表 4.12: $K_1 = 100, C = 0.13$ の場合

4.5.3 各条件での認識結果の比較

前節までの結果を比較すると表4.13のようになる。ただし特徴的な結果として、$K_1 = 300$ の場合についてのみ示す。また、表中「使用」は4.3.1で作成されたクラスタ情報を使用したもの、「重なり強化」は4.5.1で作成された強制的な重なりを持たせたクラスタ情報 ($C = 0.19$) を使用したものである。

認識ファイル	クラスタ情報	sys. time	user time	累積認識率		
				1	2	3
教科書体	未使用	0.7	1:28:04	0.882	0.938	0.953
	使用	0.4	7:48	0.676	0.712	0.718
	重なり強化	2.5	6:46	0.782	0.823	0.831
明朝体1	未使用	0.6	1:27:35	0.991	0.996	0.997
	使用	0.5	7:48	0.976	0.980	0.981
	重なり強化	1.8	6:46	0.986	0.990	0.991
明朝体2	未使用	0.5	1:27:39	0.989	0.995	0.996
	使用	0.5	7:50	0.979	0.983	0.984
	重なり強化	1.8	6:47	0.985	0.989	0.990

表 4.13: 各条件での認識結果の比較

4.5.4 考察

まずクラスタリングの結果であるが、表4.5、表4.9、表4.10を比較すると、Cの増加に伴ないクラスタ数が非常に増加していることが分かる。これは重複度を強制的に増加させたために、クラスタ内の辞書ベクトルが一定値K_1以下に落ちるまでに要する層数が増加したためである。

認識率について見ると、重なりを強化することでどのファイルに対しても認識率低下が抑制されたことが分かる。これはクラスタリング時に分割超平面付近をまたいで分布している字種をより多く分割後の両方のクラスタに含ませることができたということを意味し、予想通りの結果と言える。しかし、辞書作成に用いていない教科書体については改善は見られるものの認識率はかなり低い。やはりマルチフォントで辞書作成とクラスタリングを行なうことが必要であろう。

また、認識時間についてはクラスタ情報未使用の場合に比べ、クラスタ情報を使用した場合は大幅に短縮ができることが示された。ここで、重なりを強化した場合に更に認識時間が短縮されているという結果となったのは、重なりを強化したことでクラスタ内の辞書ベクトル空間の主成分方向が変化し、重なりを持たせなかった時に重複度の上限の条件か

ら分割できなかったクラスタが分割できたためと考えられる。予想していたものとは違っていたが、結果的には良い実験結果が得られた。

認識時間と認識率はトレードオフの関係にあることが予想されるが、重なりを強化した場合は認識率低下を抑制できることが分かった。しかし、クラスタ内の最大字種数 K_1 を小さく (=100) した場合には認識率の低下が著しく、全字種数の 10 分の 1 よりかなり小さい値に K_1 を設定することは適切でないと思われる。

第5章

結論

本研究では、文字認識における認識時間の短縮を目的とし、その手法として、主成分分析を用いた辞書クラスタリング手法を提案した。また、提案手法を評価するためにクラスタリング実験を行ない、そこから得られたクラスタ情報を使用して認識実験を行なった。辞書作成時と認識時の特徴量としては方向線素特徴量を用いた。

クラスタリング実験の結果としては、平均重複度が0.6〜0.8程度の値になり、ほぼ初期段階の試算に値する結果が得られた。また、認識実験では、従来の認識手法である全数整合法と本手法との比較を行ない、認識時間を大幅に(平均91.1%)短縮できるという結果が得られた。しかし最初の実験では既存の手法と比べ、教科書体については20.6ポイント、明朝体についても1〜1.5ポイントの認識率低下が起こってしまった。そこで認識率低下を抑制するためにより重なりを持たせた分割手法を提案し、それに基づく実験を行なった。最終的には、認識時間は従来法から平均92.3%短縮され、かつ認識率低下は教科書体で10.0ポイント、明朝体では0.5ポイントまで抑えることができ、特に辞書作成フォントについては本手法は非常に有効であるという結論を得た。認識時に単純な距離尺度を用いていることを考えれば、実験結果は評価できるものと考える。

しかし、本手法ではクラスタ分割の際に第一主成分しか考慮しておらず、かつ各字種のベクトル空間がその重心(辞書ベクトル)の周りに超球の分布を成していることを期待している。クラスタ内の各字種のベクトル空間がそのクラスタの主成分方向に偏った分布を成している場合は本手法において適切な分割が行なわれない恐れもあるため、その分布については解析が必要であると思われる。また、辞書作成に単一フォントのみを用いたため、それ以外のフォントについては非常に認識精度が悪かった。これは辞書作成時にもっと多

くのフォントを用いることで解消されると思われるが、その分重複度が上昇することも予想される。これらは今後の課題である。

謝辞

本研究を進めるにあたり、全般的な御指導を賜わりました東北大学工学部阿曽弘具教授に心より感謝致します。

また、御討論、御協力をいただいた阿曽研究室の後藤英昭氏、森大毅氏、鈴木基之氏、黒岩丈介氏、山中清氏、孫方氏に心から感謝します。

最後に多面に渡り御意見、御協力をいただき、また日頃の生活においてお世話になった阿曽研究室の皆様に感謝致します。

参考文献

[1] 阿曽 弘具, 越後 和徳, 木村 正行：
「文字特徴量空間の性質と特徴抽出法の性能評価法」
電子情報通信学会論文誌 (D-II), **J76-D-II**, No.11, pp.2285–2294 (平成 5 年 11 月).

[2] 西川 修史, 若林 哲史, 木村 文隆, 三宅 康二, 提田 敏夫：
「手書き数字認識における特徴量の合成」
電子情報通信学会技術研究報告,PRU95-114 (平成 7 年 9 月).

[3] 仙田 修司, 美濃 導彦, 池田 克夫：
「高速な大規模マルチテンプレート手書き文字認識」
電子情報通信学会技術研究報告,PRU95-116 (平成 7 年 9 月).

[4] 孫 寧, 田原 透, 阿曽 弘具, 木村 正行：
「方向線素特徴量を用いた高精度文字認識」
電子情報通信学会論文誌 (D-II), **J74-D-II**, No.3, pp.330–339 (平成 3 年 2 月).

[5] 阿部 斉：
「応用数理統計学入門」
培風館 (昭和 60 年 1 月).

[6] 河口 至商：
「多変量解析入門 I」
森北出版 (昭和 47 年 6 月).

[7] 孫 方：
「文字認識における辞書のマルチテンプレート化に関する研究」
東北大学工学部通信工学科 平成 6 年 卒業論文 (平成 6 年 3 月).