論　文

手書文字認識用辞書の自動作成*

門田彰三** 安田道夫**
山本真司** 岡　光宣***

Abstract

A new method for automatic generation of a character recognition dictionary is proposed. The composition was accomplished by use of minimization technique of incompletely specified sequential logics. We have reduced the computing time using the following techniques:

1) Reduction of the combination of the internal states. The merging begins from the final states to the starting state step by step.
2) Reduction of the internal states of each dictionary. In composition of each dictionary, only the learnig patterns of a given category and those which were confusible to the given category patterns were used.
3) Division of each learning pattern into several independent segments. Each dictionary was made up of several independent blocks.

With use of above techniques, we have succeeded in composing dictionaries with good accuracy in short time. The dictionaries so composed have been applied in recognition of hand-printed alphanumerics.

1. まえがき

手書文字を認識する方法の一つにオートマトンによる認識方式[1)~3)]がある．この方式は **Fig. 1**(次頁参照)に示すように，未知パターンを1次元的なコード列に表現し，これをオートマトンに入力し，オートマトンのアクセプト/リジェクトにより未知パターンを認識するものである．このようなオートマトンはしばしば「辞書」と呼ばれており，われわれも以下，認識用辞書または単に辞書と呼ぶことにする．辞書は **Fig. 2**(次頁参照)に示すように状態遷移図で表され，入力コードにより内部状態を変化させる．たとえば入力パターンとして5-7-1-6というコード列が入力されるとFig. 2の辞書は S_0 を初期状態として，S_2, S_3, S_4 と遷移し，最後に最終状態 F_2 に遷移する．最終状態 F はカテゴリ名「2」を出力する．

現在，手書文字認識用の辞書は人手により作られている．すなわち，膨大な量のサンプルパターンを集め，パターンのあらゆる変形を考慮に入れ，試行錯誤を繰り返して作られる．この過程で要求される莫大な時間と作業量とを節減する目的で，辞書の自動作成の試みがいくつかなされてきた[4)~8)]．

辞書を自動的に作る方法の一つとして，まず与えられた学習用サンプルから比較的簡単な規則で予備的な辞書を作り，それを再構成してコンパクトな辞書を作る方法[4)~5)]がある．この方法は，オートマトンの簡略化と同じ手法を用いるが，簡略化についてはすでに多くの方法[9)~11)]が発表されている．これらの方法に共通することは，与えられたオートマトンの中で冗長な内部状態を統合し，簡約化されたオートマトンを再構成することであり，統合のしかたにそれぞれ特色を有する．内部状態を統合するためには，任意の二つの内部状態を取り出し，統合できるかどうかをひとつずつ調

* Automatic Composition of Dictionaries for Hand-printed Character Recognition, by Shozo KADOTA, Michio YASUDA, Shinji YAMAMOTO (Central Research Laboratories, Hitachi Ltd.) and Mitsunori OKA (Odawara Works, Hitachi Ltd.)

** (株)日立製作所中央研究所
*** (株)日立製作所小田原工場

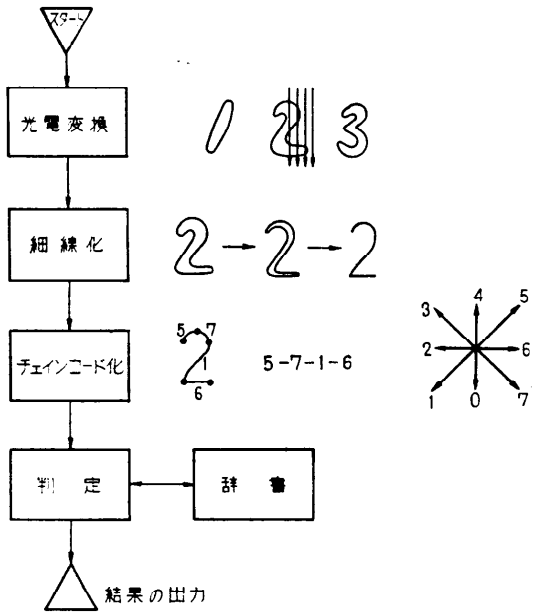

Fig. 1 Flow Chart of a Hand-printed character Recognition Method.

べる操作が必要である．内部状態が N 個あれば，任意の状態の組み合わせは $N(N-1)/2$ であり，N の増加にしたがい組み合わせは N^2 オーダーで増加する．したがって従来の方法は内部状態の少ないオートマトンには使えても，内部状態の多い認識用辞書に適用するには処理時間の制約が大きな障害となってくる．また上記の方法で作った辞書は，内部状態数を最少にすることはできるが，認識用辞書としては必ずしも最良のものであるとは言えない．一つは上記の方法で作った辞書は人が直観的に作った辞書とくらべて，修正が非常に難しいことである．もう一つは学習用に用いたのと同じサンプルであればまちがいなく正しく認識するが，それ以外の未知パターンについてはほとんど予測することができないことである．

上記の欠点を解決するため，本論文では，オートマトンの簡略化の手法を基礎にして，状態数の多い認識用辞書にも適用できるように改良を加え，処理手数が少なく，辞書の自動修正が容易にでき，かつ誤認識の少ない辞書の作成方法について述べる．また自動辞書作成法によって作成した辞書を手書英数字認識に適用した結果についても報告する．

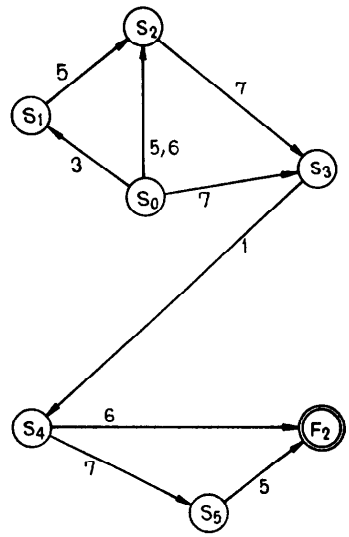

Fig. 2 An Example of Dictionary for "2"

2. 辞書の作成方法

辞書を有限状態オートマトンにより構成する．オートマトンは入力コードを入力するたびに内部状態を変化させる．入力コード a_j により内部状態 S_i から S_k へ遷移することを，

$$\delta(S_i, a_j)=S_k$$

と書き表わすことにする．内部状態の集合のうち，初期状態を S_0，最終状態の集合を F，$\{F_1, F_2, \cdots, F_N\}$ とあらわす．最終状態はカテゴリ名を出力する内部状態である．

オートマトンはコード列を入力し終ったときに内部状態が最終状態にあるときのみ入力コード列を受理し，最終状態にない場合，あるいは途中で遷移先が無くなったときにはリジェクトする．

冗長なオートマトンは次の手順で作成する．まず，学習用サンプルを一つずつ取り込みコード列を作成する．第1番目のサンプルについては，S_0 を初期状態として，入力コードが一つ入力されるたびに一つずつ状態を遷移するように，内部状態 S_1, S_2, …, S_l を付け加える．l は入力サンプルの長さである．第 i 番目のサンプルについては i-1 個のサンプルを使ってできたオートマトンで遷移できるところまで遷移し，遷移できなくなったら，そこから新らしくパスをつけ加える．すなわちすでに k 個の状態ができているとすると $k+1$, $k+2$, … をつけ加え，入力サンプルが遷移

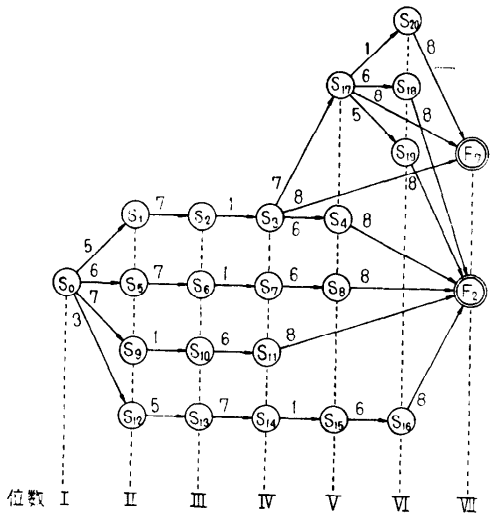

Fig. 3 A Redundant Automation

できるようにする.

冗長なオートマトンの各状態に位数を定義する. 位数とは初期状態 S_0 から何個の入力コードで遷移できるかを示すものである.

Fig. 3 に冗長なオートマトンの1例を示す. このオートマトンを作るのに使用したサンプルは, カテゴリ「2」に属するものとして

5-7-1-6-8
6-7-1-6-8
7-1-6-8
3-5-7-1-6-8
5-7-1-7-6-8
5-7-1-7-5-8

カテゴリ「7」に属するものとして,

5-7-1-8
5-7-1-7-8
5-7-1-7-1-8

を使用した. ここでコード列は 0〜8 であらわされ, 0〜7 は方向コードを 8 は枝の終端を現わす. パターンがいくつかの枝で構成されるときには, 特定の方向から走査し, 最初に見つかった端点を始点とし, 始点につながる枝から順につないでいく. 他端が分岐点の場合には左まわりにさがして最初に見つかる枝を次につなぎ, 枝のつなぎめに8を挿入する. 他端が端点の場合には, もとの分岐点にもどり左まわりに次の枝をさがす. このようにして, 枝が複数のパターンも1次元のコード列に変換することができる.

これから述べる辞書の作成方法は, 冗長なオートマトンの中から任意の二つの状態 S_i と S_j を取り出し, 統合できるかどうかを調べ, 統合できるものだけを統合し, コンパクトなオートマトンを再構成する方法である. S_i と S_j を統合しようとするとき, 統合できない場合がある. 一つは Fig. 3 に示す F_7 と F_2 のように相異なるカテゴリ名を出力する最終状態間の統合, 一つは S_{18} と S_{20} を統合した場合のように一つの入力コードで二つ以上の状態へ遷移する場合である. このような状態は, 統合禁止関係にあると言い, $S_i \neq S_j$ と書き表わす. 統合禁止条件は,

(i) $F_i \neq F_j$, $(i \neq j)$

(ii) $\delta(S_i, a)=S_i'$, $\delta(S_j, a)=S_j'$ において, $S_i' \neq S_j'$ ならば $S_i \neq S_j$.

(iii) $\delta(S_i, a)=S_j$, $\delta(S_j, a)=S_k$ において, $S_j \neq S_k$ ならば $S_i \neq S_j$.

と書きあらわされる. (i)は最終状態間の統合, (ii)と(iii)は一つの入力コードで二つ以上の状態へ遷移するような統合を禁止している. (iii)の条件は一般的には(ii)へ含まれるが, ここでは特に, (ii)を同一位数の状態間の統合禁止条件, (iii)を異なった位数の状態間の統合禁止条件を示しているものとする.

統合禁止条件(ii), (iii)が生じるのは(i)の結果である. 手書文字認識用辞書のように, 出力 C_i を出す最終状態がコード列の最後のコードで遷移する状態(位数の最も大きい状態)に限られている場合には, 最初に統合禁止条件が成立するのは, 位数の大きい状態間の統合のみであり, 位数の小さい状態からの影響を考慮する必要はない. 位数の小さい状態間の統合が統合禁止条件を満たすのは, 位数の大きい状態間の統合が進んだ結果である. したがって最終状態に近い状態間の統合からはじめ, 順に初期状態に近い状態へと統合を進めて行けば, 無駄に状態間の統合可能性を調べることなく統合でき, 処理時間を短縮できる.

統合の具体的方法を **Fig. 4**(次頁参照)のフローチャートに示す.

この方法で辞書を作ると, 統合は位数の大きい状態から順に位数の小さい状態へと, 波が岸に押し寄せるように行われ, 波の通過後には統合された状態のみが残される. 残された状態を整理し, 状態番号をつけなおせば, コンパクトな形の辞書が得られる.

この方法では同一位数の状態間および位数の一つ異なる状態間でのみ統合するので処理手数は大幅に減少する. 簡単のために, 最大位数を M とし, 各位数の

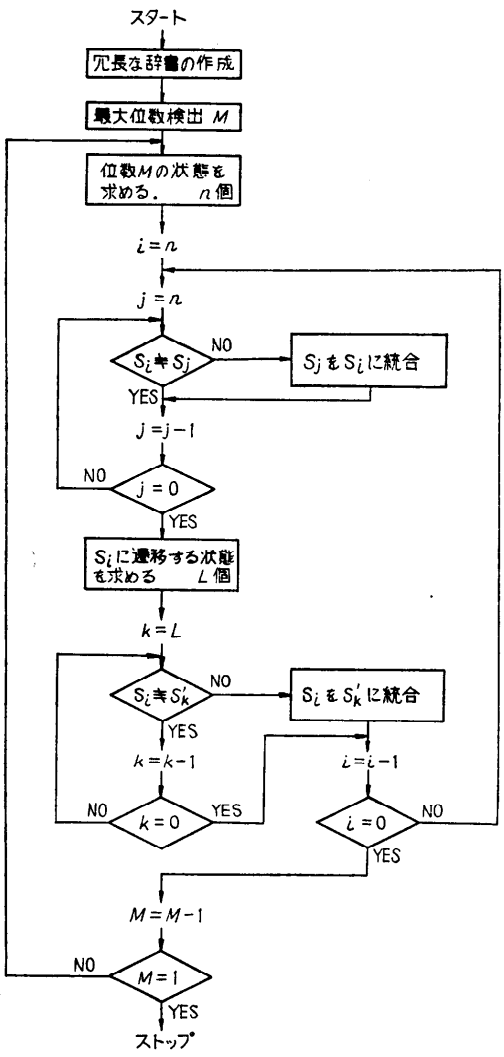

Fig. 4 Flow chart of composition of a dictionary.

内部状態は等しく n 個であるとしよう．内部状態の総数を N とすると，$N=M\times n$ である．n 個の内部状態のすべての組み合わせを考えると n の2乗あるので，処理手数はおよそ $M\times n^2/2$ のオーダーである．一方，内部状態 N 個すべての組み合わせは $N^2/2$ のオーダーであるので，本方式では従来の方式にくらべて $1/M$ のオーダーで処理手数を減少することができる．最大位数は 10 ないし 20 なので，1ケタ以上処理スピードを向上させることができる．

3. 辞書の修正

以上の方法により辞書は比較的簡単に作成できるが，未知パターンの中に学習用サンプルの中には無かったパターンが存在すると，この辞書はいつも正しくそれらを認識できるとは限らない．認識率をさらに向上させるには，すでにできた辞書を修正することが必要になるであろう．自動的に作成された辞書は人が作った辞書と異なり，直観的な表現がなされていないので人による辞書の修正は非常に面倒になる．したがって辞書の修正も自動的に行うことが望ましい．

辞書の修正にあたっては，修正した後にも内部状態数最少の条件が満たされていることが望ましいが，この条件を満たそうとすると，学習時に用いたのと同程度のサンプルを必要とする．ここでは簡単に修正することをねらい，修正サンプルのみを用いて修正する方法を考えてみる．

辞書の修正が必要な場合は，リジェクトした場合と他のカテゴリに誤まって認識した場合である．リジェクトの原因には三つあり，一つはコード列を入力して遷移した先が最終状態でなかった場合，一つはリジェクトを出力する最終状態へ遷移した場合，一つは遷移の途中で遷移先が無くなった場合である．ここでは途中で遷移先が無くなった場合についてのみ修正する．この場合には修正サンプルが受理されるような冗長なオートマトンをあらたに追加する．すなわち，遷移可能な所までは古いオートマトンで遷移し，遷移先の無くなった所から冗長な形のオートマトンを追加し，最後の状態を最終状態へ遷移させる．追加された冗長なオートマトンの部分のみについて，辞書作成時と同様に他の状態と統合を行う．この処理により，新しく遷移する道ができ，リジェクトされたパターンを認識することができる．

誤読になるのは，異なったカテゴリのパターンで互いに似かよったパターンが学習用サンプル中に含まれていず，たまたま統合の過程で誤まったカテゴリに遷移するようになった場合である．誤読の場合には，誤読されたパターンが誤まったカテゴリに受理されないように，遷移の途中を切断し正しいカテゴリに遷移するよう新しく道を作ってやればよい．しかし，やみくもに道を切断していたのでは，今まで正しく認識されていたサンプルまで読まれなくなるので，誤読したサンプルだけが通る道を抽出し，それの最後の道を切断し，正しいカテゴリの最終状態へ遷移させればよい．

Fig. 5（次頁参照）の例により，もう少し詳しく説明しよう．カテゴリ C_j のサンプル，1-7-0-3-4-3 というコード列が，誤まってカテゴリ C_i と認識された

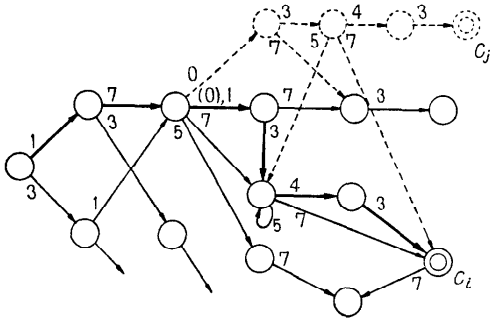

Fig. 5 Revision of a dictionary (in case of substitution error)

とする．このサンプルが通る道は図の太い実線で印した道である．この道には他のサンプルも通るので，誤読したサンプルしか通らない遷移図を Fig. 5 の破線で示したように浮き上がらせる．この時に他のサンプルが通ってもリジェクトされないようにすべての道を残しておく必要がある．この状態で，最後の道をカテゴリ C_j の最終状態へ遷移させれば修正は完了する．

遷移先が最終状態でなくリジェクトされた場合も誤読の場合と同様にして修正される．

4. 認識率の向上

以上の方法により辞書は自動的に作成できるが，認識精度の良い辞書を作るにはどうすれば良いかを考えてみよう．学習に使ったサンプルについては 100% 完全に認識するが，学習に用いなかった未知パターンについてはすべて正しく認識するとは保証されていない．

誤まりを無くすには徹底的に学習サンプルを増やして，あらゆる可能な変形パターンを含むような学習パターンを用いて辞書を作れば良いが，現実的には，むやみに学習サンプルを増やすことは不可能である．

以上に説明した辞書の作成方法では **Fig. 6(a)** に示すように，複数個のカテゴリを同時に考えるため，カテゴリ間の干渉による誤認識が生じやすい．これを避けるため，Fig. 6(b) に示すように，カテゴリごとに N 個の辞書に分割し，それぞれの辞書は一つの入力と一つの出力を持つようにする．

カテゴリ C_i のサンプルを受理するオートマトンを作るにはカテゴリ C_i に属するパターンのみ最終状態 F_i に遷移し，他カテゴリのすべてのパターンはリジェクトを出力する最終状態 F_R に遷移するようにする．このような方法だと 1 カテゴリのオートマトンを作るのに全サンプルを使用するために，カテゴリの数が N 個あれば全体で N 倍の処理時間を必要とするが，後で述べるように，あまり処理時間を増やすことなく辞書を作ることが可能である．辞書をカテゴリごとに分割することにより誤読率は大幅に減少することができる．分割のしかたはカテゴリ単位に限定する必要はなく，さらに細かく分割することにより誤読率は一層減少するであろう．

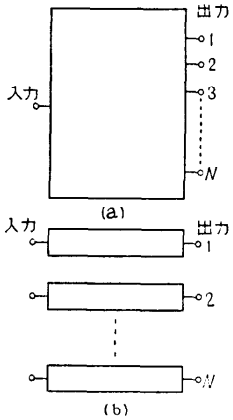

Fig. 6 Configuration of Dictionary (a) A Dictionary with one Input and Many Outputs. (b) Dictionaries with one Input and one Output.

すなわち，一般に文字パターンは，同一カテゴリのものであってもいくつかのサブカテゴリを形成しているため，それぞれのサブカテゴリごとに分割して辞書を作ればよい．

5. 辞書作成時間

辞書の作成に要する処理時間は，冗長なオートマトンの内部状態を k 個とすると，ほぼ k の 2 乗に比例して増大する．内部状態は辞書作成に用いる学習用サンプルの数にほぼ比例すると思われるので，学習用サンプルを増やすことにより処理時間は急激に増大することが予想される．**Fig. 7** (次頁参照) は手書数字用辞書を HITAC 5020 F を使用して作成したときの処理時間を実測しプロットしたものである．手書数字のみならずアルファベット用の辞書を同様な方法で作成するとなると，10^4 から 10^5 の学習用サンプルを使用するため，多くの計算時間を必要とする．以下では処理時間をいかにすれば減らせるかについて検討してみよう．

内部状態数を k とすると処理時間はほぼ k の 2 乗に

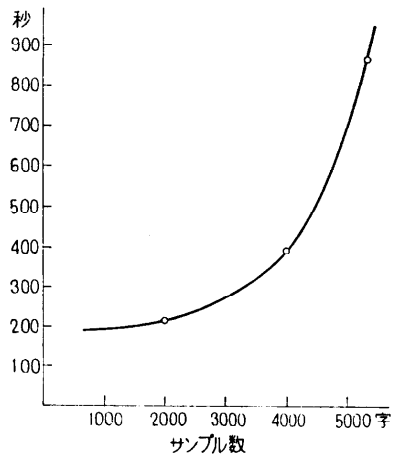

Fig. 7 Change of computer use time to make dictionaries with increase of samples for learning.

比例するので k を減らすことを考える．カテゴリごとに辞書を作る場合にはすべてのカテゴリのサンプルを使用する．カテゴリ C_i の辞書を作るのにカテゴリ C_i のサンプルしか使わなければ明らかに誤認識が多く使用に耐えない．カテゴリ C_i に属さないパターンをリジェクトするために，他カテゴリのサンプルが必要である．しかし本当にすべてのサンプルが必要であろうか．一般にカテゴリごとにコード列の先頭コードは特定のコードに限定できる．たとえばカテゴリ C_i のサンプルの先頭コードが8方向コードのうち0と1しか取り得ないとすれば，初期状態 $S_0{}^i$ からの遷移を，$\delta(S_0{}^i, 0)$ と $\delta(S_0{}^i, 1)$ しか許さなければ，0，1 以外のコードで始まるパターンはすべてリジェクトされる．もしも先頭コードの出現確率がコードによらず一定であれば，これら二つのコードで始まるサンプルだけを用いることにより，使用すべきサンプル数は 2/8 =1/4 ですむことになる．

さらにコード列が枝ごとのようにたがいに独立ないくつかのブロックに分割できる場合には，内部状態をさらに減少させることができる．辞書を **Fig. 8** に示すように構成する．

すなわちカテゴリごとの辞書は l 個のブロックをつないだものとして表わされ，それぞれのブロックを構成するオートマトンの初期状態には前段のブロックの最終状態が接続されている．

この辞書を作るときの処理時間を見積ってみよう．説明を簡単にするために，各コード列の長さはすべて

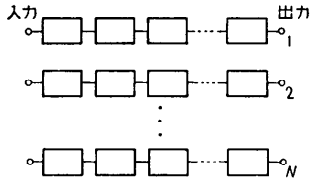

Fig. 8 Comfiguration of Dictionaries with sectional blocks.

等しく k，サンプル数を m とし，冗長なオートマトンの中で同一状態を共用しないと仮定すると内部状態数 n は $m \times k$ で与えられる．もしもコード列がたがいに独立した l 個のブロックに分割でき，それぞれの長さを k_i，各ブロックごとに互いに異なるコード列が m_i 個存在するとする．その場合の内部状態は，

$$n' = \sum_{i=1}^{l} m_i \times k_i \quad \left(m_i \leq m,\ \sum_{i=1}^{l} k_i = k \right)$$

で与えられ，$n' \ll n$ である．一般に文字パターンはいくつかの枝で構成され，各枝は他の枝とあまり相関がないことから，コード列をいくつかの独立したブロックに分割することは意味のあることである．

以上述べた方法がどの程度効果的であるかを知るために次のような実験を行った．平均コード長 25 のサンプル 49 個を用い，

（1） すべてのサンプルを用いてカテゴリごとに辞書を作る，

（2） 先頭コードで他カテゴリのサンプルを選択して辞書を作る，

（3） コード列を枝ごとに独立したブロックに分け，ブロックごとにオートマトンを作り，後でつなぎあわせて辞書を作る，

の三つの方法で辞書を作成し，その内部状態数と処理時間を求めたのが **Table 1** の結果である．実験に使用したサンプルにはカテゴリが四つあり，カテゴリ C_4 にはかなり多くのサンプルが含まれている．（1）の方法ではすべてのサンプルを使用しているためにカテゴリによる状態数の変化はみられない．（2）の方法ではかなり内部状態を減らせたが，カテゴリ C_4 では制限

Table1 Number of internal states and computer use time to make dictionaries

カテゴリ / 作成方法	カテゴリごとの状態数				処理時間
	C_1	C_2	C_3	C_4	
1	719	719	719	719	300秒
2	23	21	92	677	102
3	22	20	22	72	13

がうまく働いていない．(3)の方法によればすべてのカテゴリについて状態数を大幅に減らすことができ処理時間も減少している．

6. 手書アルファベットの認識

以上の考察から認識精度の良い辞書が簡単に作成できる見通しが得られたので，手書英数字の認識に適用してみた．手書文字は 5×6 mm の枠内に自由に書かれたもので，ビジコンカメラで光電変換し，クリップレベル 70% で2値化したものである．各文字の大きさは 30×25 メッシュである．アルファベット・数字および記号あわせて 52 種を各カテゴリごとにおよそ 400 サンプルずつ集めて学習用サンプルとして用いた．光電変換された文字は，細め処理により1ビットの太さの線図形に変換される．このパターンを4象限方向コードでコード化し特徴コード列とする．4象限方向コードとは，線図形を端点，分岐点および極値点で分割し，分割された曲線がどの象限に属するかによりコード化したものである．コード化された線図形は文字パターンを構成する成分数，枝数，ループ数などを使っておよそ 50 のクラスに分けられる．

辞書は各クラスともカテゴリごとに一つ作るのを原則とするが，同一カテゴリのサンプルであってもその中でいくつかのクラスタを構成することが明らかな場合には各クラスタごとに辞書を作成する．

学習用に使用したサンプルはおよそ2万サンプルあり，その中で全く同じコード列のものを除くと 6,300 のコード列が残る．これらのコード列の平均の長さは約 20，長いもので 30 である．これらを使って辞書を作成したが，辞書の総数はおよそ 400，各辞書の内部状態の総計はおよそ1万状態であった．一つの辞書はおよそ 25 の内部状態で構成されていることがわかる．手書数字の場合は辞書数約 50，全内部状態数 500，平均内部状態数 10 であったのに比べると，辞書数，内部状態ともに大きくなっている．このようにして作成した辞書を使って手書アルファベットおよそ1万字を認識したところ，誤認識 1.4%，リジェクト 4.2% であった．まだ誤認識が多いのはアルファベットの中にはまぎらわしい文字が多く，それらを区別するに十分なほど学習サンプルが多くなかったためである．なお認識用プログラムはフォートランで書かれ，5020F上でおよそ 32kW，処理時間は1字あたりおよそ 0.5 秒であった．

7. むすび

手書文字認識用辞書の自動作成の方法を検討し，手書英数字の認識に適用した．辞書の作成方法は，はじめに冗長なオートマトンを作り，内部状態を統合しながらコンパクトなオートマトンを作る方法を用いたが，辞書作成にあたっては処理時間を極力短縮すること，および誤読率を低減することに重点を置いた．処理速度を向上させるために，統合の方法を工夫するとともにパターンの性質を使って内部状態を少なくした．誤読を少なくするために，統合によりカテゴリの異なるパターン間の混同が生じないように，カテゴリごとに辞書を作った．以上の方法を用いることにより，大量の学習用サンプルを用いても短時間のうちに，手書英数字を認識する辞書を作成することができた．

このような方法で辞書を作成するときに注意しなければならない点は，この辞書が学習用に用いたサンプルと同一のものについては 100% 正しく認識するが，それ以外のサンプルについては何らの保証も与えられていないことである．すなわち学習用サンプルの中には起こり得るすべての変形パターンがほぼ完全に含まれているという前提が必要である．そのためにはかなり大量の学習用サンプルを必要とし，大量のサンプルを用いても短時間で処理できる方法を考慮しておかなければならない．本論文で処理時間の短縮を強調したのもそのためである．

最後に，本研究を進めるにあたり，御指導いただいた日立中央研究所 中田和男主管研究員，川崎淳6部部長，藤本ユニットリーダーおよび小田原工場入出力部森部長，佐野課長に深謝する．

参考文献

1) Genchi *et al.*: Recognition of handwritten numeral characters for automatic letter sorting, Proc. IEEE Vol. 56, No. 8, p. 1292 (1968).
2) 山本真司ほか：手書き数字認識論理の設計，信学論，Vol. 53-C, No. 16, p. 691 (1970. 10).
3) Hoshino *et al.*: Computer-aided design for a reader of handprinted characters, Int. Joint Conf. on artificial Intelligence. p. 153 (1969).
4) 榎本，ほか：句構造サンプルパターンを識別するオートマトンの構成について，信学会，オートマトン研資，A 69-76 (1970-03).
5) 榎本，ほか：オートマトンのサンプルパターンによる修正方法，信学会，オートマトン研資，A 70-20 (1970-07).
6) 門田，ほか：手書文字認識用辞書の自動作成，

信学会，オートマトン研資，A71-53 (1971-09).
7) 星野，ほか：認識のための順序論理の自動作成方法と自由手書文字認識への適用，信学会，オートマトン研資，A71-54 (1971-09).
8) 阿部：記号系列パターンを識別するあるオートマトンの構成について，信学会，パターン認識と学習研資，PRL 74-5 (1974-05).
9) M. C. Paull and S. H. Unger: minimizing the number of states in incompletely specified sequencial switching functions, IRE Trans. Vol. EC-8, p. 356 (Sep. 1959).
10) A. Grasselli and F. Luccio: a method for minimizing the number of internal states in incompletely specified sequential networks, IEEE Trans. Vol. EC-14, p. 350 (Jun. 1965).
11) J. Kella: State minimization of incompletely specified sequential machines, IEEE Trans. Vol. C-19, No. 4, p. 342 (April 1970).

(昭和51年2月19日受付)
(昭和51年5月20日再受付)