

【適用事例】 文字認識における曲線の屈曲度の判定

《学修項目》

- フリーマンコード化を用いた曲線のコード化
- 曲線の外輪郭と内輪郭の速度差を用いた曲線のコード化
- フリーマンコード化を用いた曲線の識別実験
- 両輪郭の速度差によるコード化手法を用いた識別実験

《キーワード》

文字認識、曲線、屈曲度、フリーマンコード、特徴量、コード化、識別、類似文字、特徴ベクトル、認識率

1. はじめに

手書き文字の中には多くの類似文字が存在する。英数字を例にとれば、「5-S」、「U-V」、「2-Z」などの類似文字の組みが挙げられる。これに対し、多くの商用OCR(Optical Character Reader)では、これらの類似文字を簡単に区別できるように、書き手側に模範的な書き方の指針を与えている。そこでは、類似文字の一方、あるいは両方について慣習的な文字の書き方にわずかな変更を加えている。例として、図2は、「U-V」と「5-S」に対する筆記例を示す。

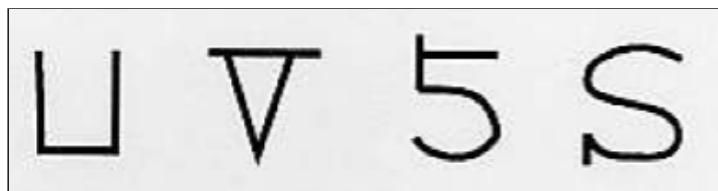


図2 手書き文字の筆記法に関する拘束条件の例

このような筆記法に関する拘束は、「O(アルファベットのO)-0(零)」の組に対しては必要不可欠なものであるかもしれない。しかし、これらの条件を与えて、必ずしも筆記者が、その全てを満足するような書き方をするとは限らず、印刷文字として伝統的に書かれてきた文字の形状に従って書いてしまう傾向があることは否定できない。そこでこれらの類似文字を含む手書き文字に対する従来の認識法では、段階的な手法が広く採用されている。ここでは、ほとんどの文字が第一段階で識別されるが、二つのカテゴリーのどちらにも採用できる文字に対しては、この段階では認識せず、第二段階の処理へと移行する。この第二段階にまわされる文字は上記のような類似文字群であり、第二段階では、これらの類似文字群の識別を行う。このような識別処理をズーミング(zooming)と呼んでいる。

中野[47]は、文字認識のエキスパートシステムに関する提案をしており、そこでズーミングシステムの必要性を指摘している。しかし、類似文字の組は多種多様に存在し、これら全てに対応するズーミングシステムを構築することは難しい。文字認識に精通している技術者であれば、適切な識別ルールを見出してプログラムを作成することは、不可能なことではないかもしれない。しかし、それらのプログラムが全ての組に適用できるとは限らない。たとえ全てに対応するプログラムを作成できたとしても、それらに含まれるパラメータの調整が必要であり、結局、大変な労力と時間を費すことになることは明らかである。このような理由から、ズーミングの自動設計あるいはコンピュータ支援の設計手法が必要となる。

そこで、ここでは、ニューラルネットワークを用いて、上記のズーミング手順を自動化するための事例についての実験を紹介していこう。今回、対象として行ったのは、曲線の尖鋭度に関する識別であ

る。これは、例えば「2-Z」、「5-S」、「U-V」、「D-O」のように、その文字に含まれる曲線の尖鋭度によって識別が可能な類似文字群が数多く存在しているためである。

「5-S」のような類似文字は、複数の曲線セグメントから構成されていると見ることができる。当然、類似文字の識別をするためには、双方の文字中の尖鋭度の異なる曲線セグメント部分を処理対象とするべきであり、それらの曲線セグメントを検出する前処理が必要となる。曲線セグメントの検出は、例えばOPLM(Outmost Point List Method)[50]手法を用いることにより解決できる。この方法は、入力文字パターンの輪郭を抽出後、それらの曲線を凸と凹に分類することができる。図3に類似文字「5-S」の場合の凸と凹セグメントの分割例を示した。このように分割した後、「5-S」の識別には、図3のセグメント番号1の凸曲線を対象にして尖鋭度の識別処理を行えば良い。

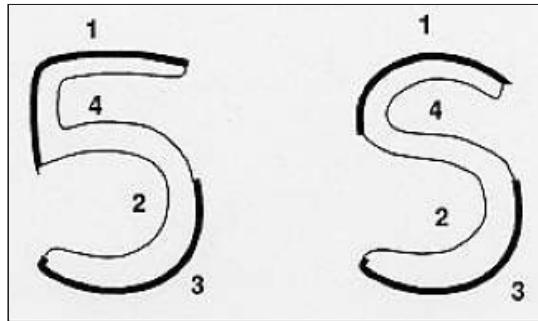


図3 輪郭線のセグメント分割の例（太線：凸の曲線部分、細線：凹の曲線部分）

ここでは、問題を簡単化するために、識別すべき一本の上に凸な曲線セグメントが上記のような方法であらかじめ抽出され、その曲線の両端点と屈曲点(頂点)の位置情報が既知であると仮定して処理を行ってみる。

凸曲線セグメントの曲がり具合の特徴をニューラルネットワークに入力するために、ここでは、次の二つの方法を用いて特徴のコード化を行う。

- 曲線の輪郭をフリーマンコードを用いてコード化する手法
- 曲線の外輪郭と内輪郭の速度差をコード化する手法

これらのコード化法を用い、ニューラルネットワークにより曲線の尖鋭度の識別が可能かどうかを調査する。

2. フリーマンコード化を用いた曲線のコード化

ニューラルネットワークを活用するためには、曲線パターンから曲がり具合についての何らかの特徴量を抽出して、ネットワークの入力とするためにそれらをコード化する必要がある。ここでは、曲線の屈曲部分の方向成分を特徴として採用するためにフリーマンコードを用いたコード化の手法を使用してみる。

フリーマンコードは、線図形を正方格子座標系において折れ線近似表現するコード化手法である。これは図4に示すように、各単位方向に図に示すような数値を与え、折れ線を0から7の数値の列として表現する。例えば、図5に示す線図形は「4564356707」の数値列にコード化される。

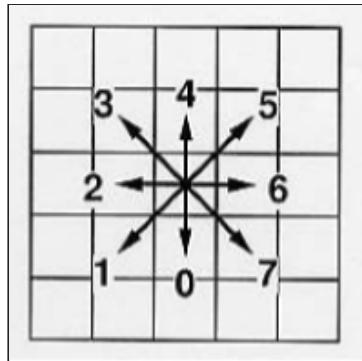


図4 フリーマンコードの方向と数値

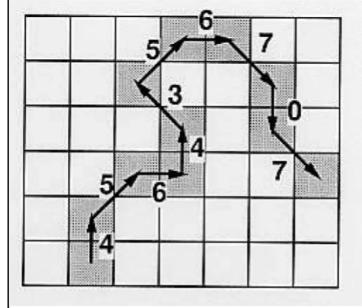


図5 曲線のフリーマンコード化の例

解像度の高い実際の曲線画像に対して各画素を単位としてフリーマンコード化を行うと、非常に大きな数列になってしまう。よって、ここでは曲線を大まかに水平分割して、フリーマンコード化することを考える。

図6に示すように、上に凸な曲線セグメントにおいて、左端点 $(x_{-10}, y(x_{-10}))$ 、屈曲点 $(x_0, y(x_0))$ 、右端点 $(x_{10}, y(x_{10}))$ の座標が既知であるとした場合のフリーマンコード化は次のように行う。

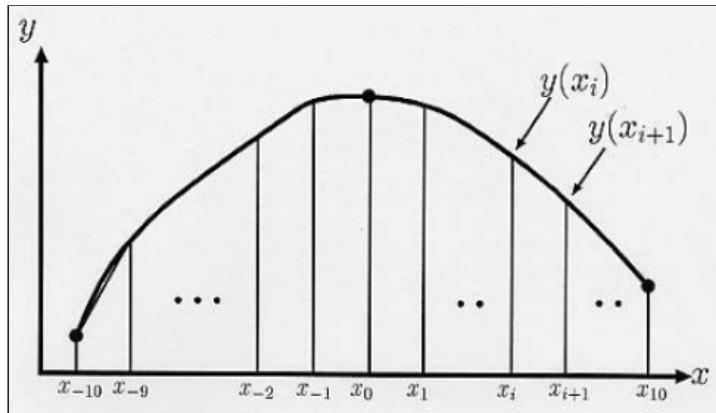


図6 曲線の分割

まず、軸の区間 $[x_{-10}, x_0]$ と $[x_0, x_{10}]$ をそれぞれ等間隔に10分割し、分割した x 軸の各点の値を $x_{-10}, x_{-9}, \dots, x_0, x_1, \dots, x_{10}$ で示す。そこで、各整数 $\{i | -10 \leq i \leq 9\}$ について座標 $(x_i, y(x_i)), (x_{i+1}, y(x_{i+1}))$ を結ぶ区別的な直線を考える。この直線を次のアルゴリズムを用いてフリーマンコードに変換する。

```

for i := -10 to 9 do begin
    loop := (y(x[i+1]) - y(x[i])) / (x[i+1] - x[i]) の絶対値の整数部分
    if(loop = 0) then フリーマンコード6を生成
    else begin
        if(i < 0) then begin
            フリーマンコード5を生成
            loop-1回分、フリーマンコード4を生成
        end{if}
        else begin {i >= 0 のとき}
    end{if}
end{for}

```

```

loop-1回分、フリーマンコード0を生成
フリーマンコード7を生成
end{else}
end{else}
end{for}

```

ここで、コード化の起点が上に凸な曲線の左端点であるため、上記のアルゴリズムでは1、2、3の方
向コードを持つ直線は存在しないと仮定している。例として、図7(a)の区分的直線は、(b)のようなフ
リーマンコードに変換される。

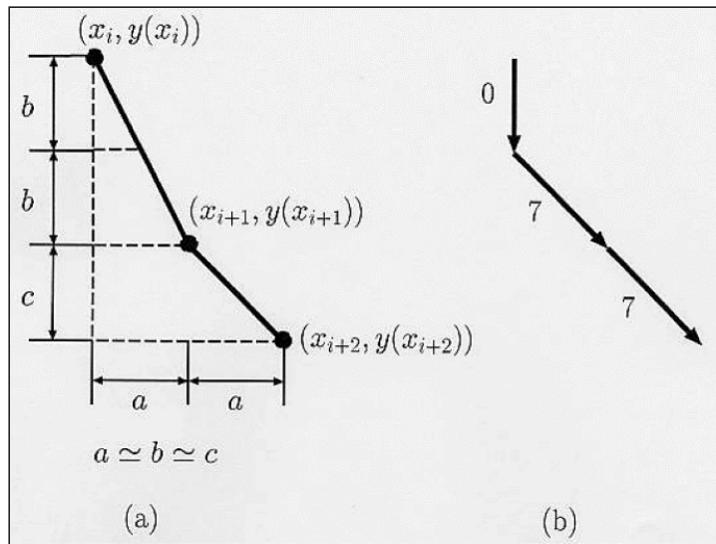


図7 フリーマンコード化の例

採用するフリーマンコード列は、曲線の屈曲点付近に重要な特徴が含まれることと、ニューラルネット
ワークに入力するコードの数を全ての曲線について同一にするために、曲線の屈曲点(座標
($x_0, y(x_0)$))を中心に、図8に示すように左右に各10個のフリーマンコードを採用する。よって、コ
ード化のアルゴリズムからもわかるように、曲線が鋭く屈曲するほど フリーマンコード列は、その屈曲
した頂点付近に集中することになる。

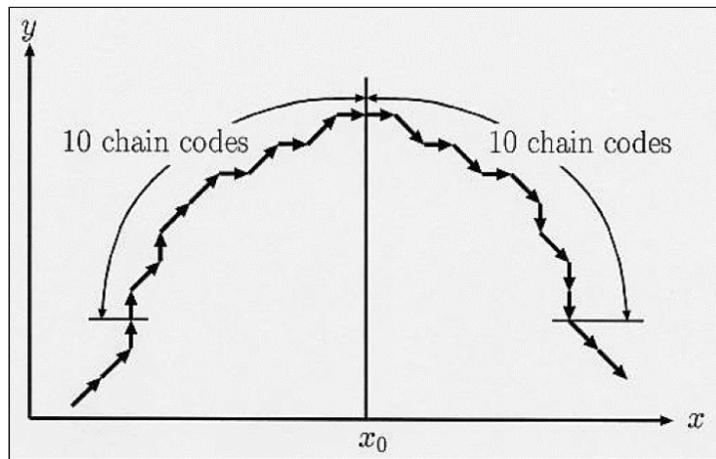


図8 採用するフリーマンコード列

さらに、フリーマンコード列を回転不变な量とするために、隣合ったコードの一階差分を求め、新た
に一階差分化したフリーマンコード列として採用する。隣合ったフリーマンコードの値 a 、 b から、
その一階差分 c を求めるアルゴリズムは次のとおりである。

```

if (|b - a| > 4) then c := {8 - |b - a|} * sign(b - a) * (-1);
if (|b - a| = 4) then c := 4;
if (|b - a| < 4) then c := b - a;

```

ここで、 $sign(x)$ は x の符号を求める関数であり、 x が正なら 1、負なら -1、0 なら 0 を返す。一階差分化したフリーマンコードの値を 0 から 7 の値にするために実際には一階差分 $c + 3$ の値を用いる。また、ニューラルネットワークの入力層には、0 から 7 の値を 3 ビットで表現し、各ビットの値を入力することとする。

上記のコード化をビット列で表現した場合、隣り合った方向のコードのビット表現において、必ずしも同じだけの数字の変化があるわけではない。つまり、0 から 7 を 2 進数で 000, 001, 010, 011, 100, ..., 111 と表したとき、000-001 間はビットが 1 つ変化するだけだが、001-010 間は 2 つ変化し、かつ 111-000 間は隣り合った方向コードであるにも関わらず、3 つのビット変化が生じる。そこで、コード変換部分の画像誤差に対する耐性を向上させる目的で、隣り合う方向のフリーマンコードのハミング距離が常に 1 となる Gray 符号化を採用してみる。

Gray 符号化は、0 から 7 までの数値を 000, 001, 011, 010, 110, 111, 101, 100 で表現するものである。

3. 曲線の外輪郭と内輪郭の速度差を用いた曲線のコード化

曲線のコード化のための二つ目の特徴として、曲線の外輪郭と内輪郭の速度差に注目してみる。これは、曲線の輪郭を陸上競技のトラックに置き換えると理解しやすい。曲線の内側の輪郭(内輪郭)をトラックの内側のコースとし、外側の輪郭(外輪郭)をトラックの外側のコースとする。そこで、二人の選手が同一ラインから同時にスタートして各自のコースを走り、同じゴールに向かって同タイムで走らなければならないとすると、トラック(曲線)が鋭く曲がっていると、外側のコースを走る選手は大きな不利を感じる。反面、なだらかなトラックの場合は、あまりハンディキャップを感じない。これは当然ながらトラックが鋭く曲がっているほど、外側の選手は内側の選手より速く走らねばならないからである。この観点から、内と外の二つのコース(輪郭上)の選手の速度差を曲線の尖鋭度の特徴として採用することにする。

実際の曲線画像上で速度差を表現するために、西田ら[49]が提案した細線化アルゴリズムの手法を応用してみよう。そこでは、図9のように、内輪郭と外輪郭上に等間隔に点を定め、各点と、反対側の点列からその点に最も距離が近い点とを対応させる。そして、両側の輪郭線上の点を端から辿り、一方が次の点に移行する間に、他方がどれだけ移動するかを 0, 1 の値によってコード化する。ここで、0 は進度がなく、1 は輪郭線上の点を 1 つ分、先に進むことを意味する。

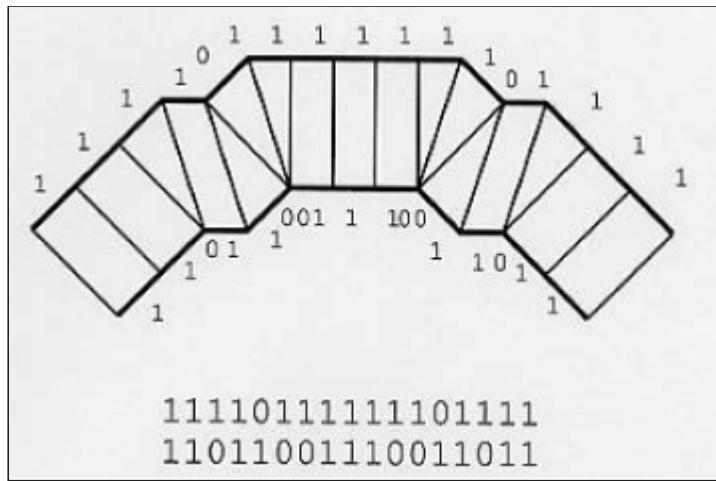


図9 内輪郭と外輪郭の対応点とそのコード化

曲線の尖鋭度を表現する特徴ベクトルとして、このビット列を使用する。この特徴ベクトルは、両側の輪郭の差分処理によって生成しているため、曲線パターンの回転に不变な量となる。

4. 評価実験(1)：フリーマンコード化を用いた曲線の識別実験

フリーマンコード化による手法では、計算によって求めたモデル曲線と、手書き曲線に対する評価実験を行ってみる。また、外輪郭と内輪郭の速度差を特徴ベクトルとする手法では、手書き曲線の他に実際の手書き文字「U-V」、「(-<)」に対する識別を行ってみよう。

ニューラルネットワークの構成は、3層型のネットワークを採用している。ここで、入力層のユニット数は、特徴ベクトルのビット列分の数を用意し、中間層のユニット数は5、出力層のユニットは対象の曲線が「まるい」とき「0」、「するどい」ときには「1」を指す一つのユニットとする。中間層は種々の実験から最適と思われるユニット数に設定している。また、ネットワークの学習過程での各パラメータの値は、学習定数 $\eta = 0.2$ 、安定化定数 $\alpha = 0.9$ 、中間層と出力層の重みの初期値 $w_{int} = 0.5$ のように設定した。

4.1 モデル化曲線の識別

手書き曲線の識別をする前準備としてコンピュータによって生成した曲線(モデル曲線)についての識別を行ってみよう。モデル曲線は、次の関数を用いて生成する。

$$|x|^n + |y|^m = 1 \quad (-1 \leq x \leq 1, \quad n, m > 0) \quad (7)$$

この関数によって生成される曲線は図10に示すように $(-1, 0)$ 、 $(1, 0)$ を端点、 $(0, 1)$ を頂点(屈曲点)とし、 y 軸に対称な曲線となり、 n, m の値が大きくなるほどなめらかな曲線となり、 n, m の値が小さくなるほど頂点で鋭く屈曲した曲線となる。

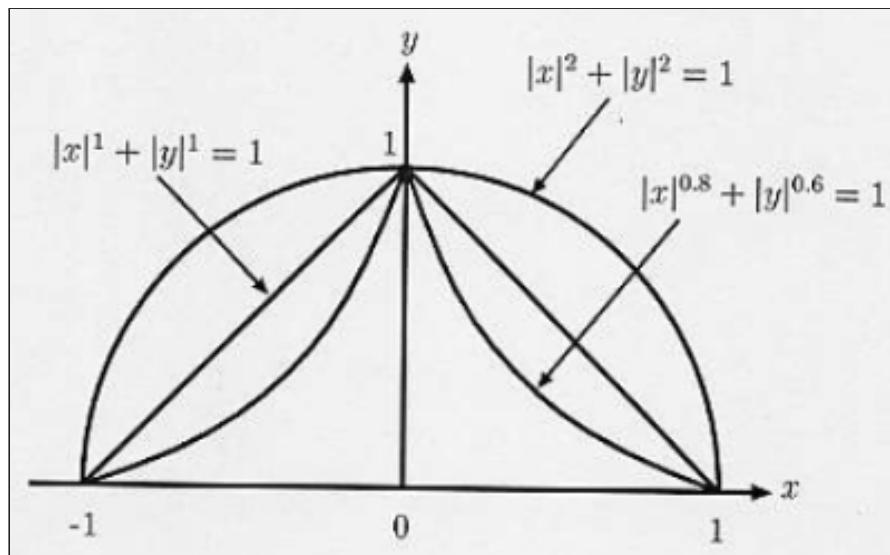


図10 モデル曲線の例

次に、生成した曲線を図11に示すシステムを用いて、被験者に提示する。実験で提示した曲線は、式(7)の n の値を0.2刻みで0.6~2.4、 m の値を0.2刻みで0.2~2.0まで変化させたときの、合計100本の曲線である。被験者はこれらの曲線を見て、それが鋭く曲がっているか、なめらかに曲がっているかを判断して、その判定値を入力した。その結果、鋭く曲がっていると判定された曲線が49本、なめらかに曲がっていると判定された曲線が51本であった。

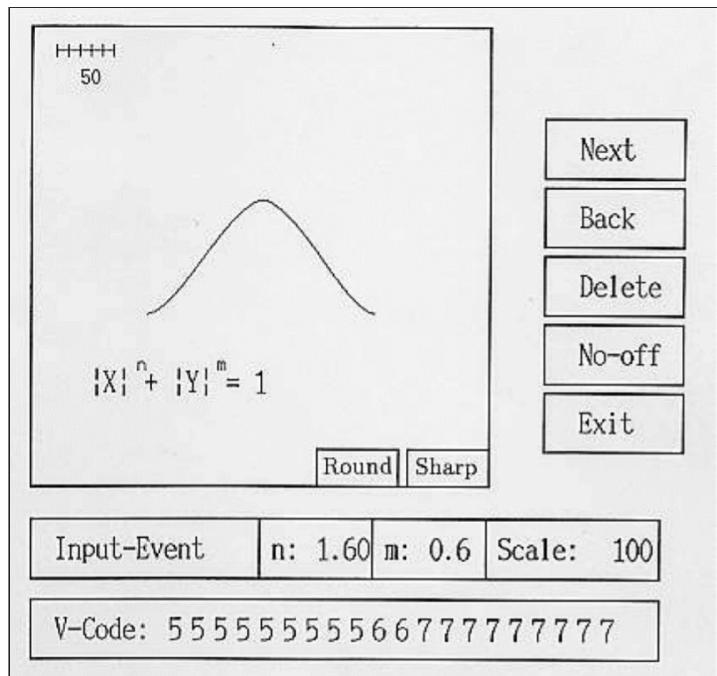


図11 モデル曲線の提示画面

ニューラルネットワークには、上記の100本の曲線とその判定値を与えて学習を行わせた。その際、曲線は「フリーマンコードを用いた曲線のコード化」で述べたコード化手法を用いて、次の4つのビットコード列に変換をし、ネットワークの入力としている。

- 通常のフリーマンコード(コード長60)
 - 通常のフリーマンコードをGray符号化(コード長60)
 - 一階差分化したフリーマンコード(コード長57)
 - 一階差分化したフリーマンコードをGray符号化(コード長57)

よって、入力層のユニット数は、通常のフリーマンコード化を用いた場合は60であり、一階差分化した場合は57である。出力層には被験者によって入力された尖鋭度の判定値を教師信号として与え(曲線が「まるい」と判定された場合は「0」、「するどい」と判定された場合は「1」を与える)、「3層型ニューラルネットワークによる学習と識別」で述べた誤差逆伝搬アルゴリズムを用いて、ネットワークの重みの学習を行わせた。学習は1サイクルのネットワークの出力値と教師信号の値の平均誤差が0.0001に達したところで停止させた。ここで、1サイクルとは、用いる全ての学習パターン(曲線パターンと、対応する教師信号)をネットワークに与えて、重みの更新をする一回の処理過程のことである。

以上の学習によって得られたネットワークの重みを用いて、学習パターンと未知パターンの識別を行った。学習パターンはネットワークの学習に用いたものと同一の100本の曲線である。未知パターンは式(7)の n の値を 0.2 刻みで 0.5 ~ 2.3、 m の値を 0.2 刻みで 0.3 ~ 2.1 まで変化させたときの、合計100本の曲線である。未知パターンの各曲線の尖鋭度の判定値は、学習パターンの判定をしたと同一の被験者に入力してもらった。その結果、未知パターンに関しては鋭く曲がっていると判定された曲線が50本、なめらかに曲がっていると判定された曲線が50本であった。

識別は学習同様、4種類の各コードについて行った。識別結果となるニューラルネットワークの実際の出力は実数であり、0.5をしきい値として、0.5未満をなめらかな曲線、0.5以上を鋭い曲線として判定を行っている。この識別結果を表1に示す。

表1 モデル曲線の識別実験の結果

符号化手法	Gray符号化	データ	学習サイクル[回]	認識率[%]
通常のフリーマンコード	×	学習パターン(100本)	554	100
通常のフリーマンコード	×	未知パターン(100本)	-	99
通常のフリーマンコード	○	学習パターン(100本)	684	100
通常のフリーマンコード	○	未知パターン(100本)	-	96
一階差分化コード	×	学習パターン(100本)	5445	100
一階差分化コード	×	未知パターン(100本)	-	97
一階差分化コード	○	学習パターン(100本)	4481	100
一階差分化コード	○	未知パターン(100本)	-	97

4.2 手書き曲線の識別

次に手書き曲線を対象にして識別実験を行ってみよう。識別までの処理過程は図12に示すとおりである。

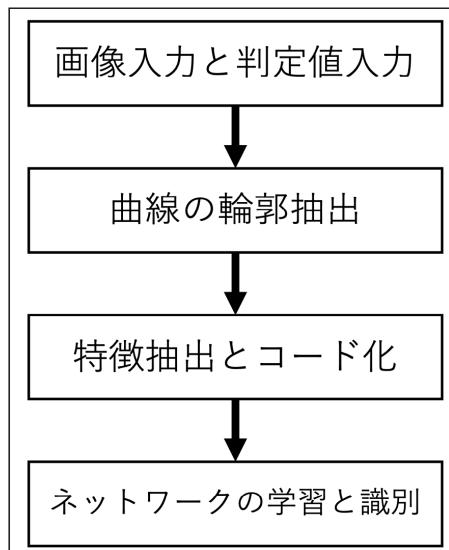


図12 手書き曲線識別の処理過程

まず、図13に示すように曲線の端点 a 、 b と屈曲点 c の位置を被験者に与え、その点を通る、上に凸な曲線を手動で書いてもらう。これは、モデル曲線の認識に用いたネットワークの重みを使って、手書き曲線の尖鋭度の判定が可能かどうかを検証するために、なるべく手書き曲線をモデル曲線に合わせるためにある。また、曲線の大きさや位置の正規化処理を省くことができ、処理を簡略化できるため、このような入力手法を採用してみた。

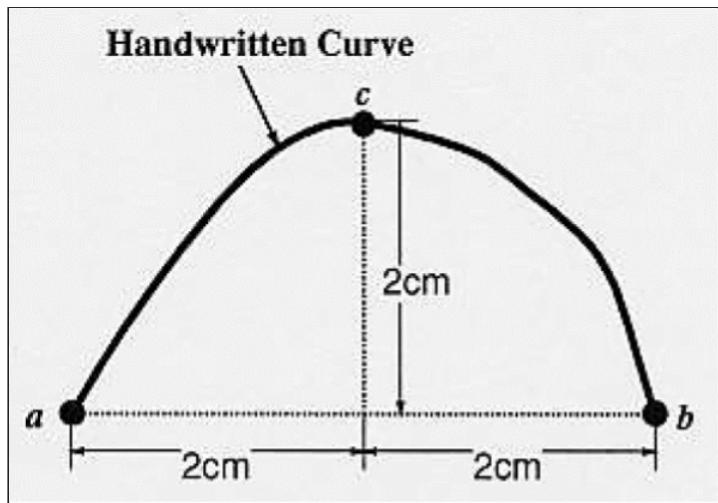


図13 手書き曲線の書き方

手書き曲線が書かれた画像をイメージスキャナで取り込む。この時のスキャナの解像度は400dpiであり、20分割程度のフリーマンコード列を得るには十分な解像度である。画像濃度は、適當なしきい値によって2値化(白黒画像)処理を行う。また、この画像入力と同時に、その曲線の尖鋭度の判定値も被験者に入力してもらった。

読み込んだ画像において、曲線を含む800×800画素の正方形領域を手動で切り出し、その領域のごま塙状の雑音を除くために孤立点除去を行った後、曲線の輪郭線抽出を行った。簡単のために輪郭線の両端(図14の a, b)の点の位置を手動で入力し、2点間を結ぶ内側の輪郭線(図14の太線)を手書き曲線画像として抽出、採用した。

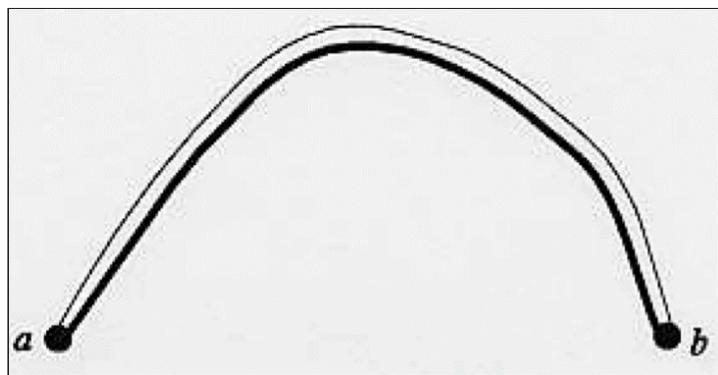


図14 手書き曲線の輪郭線抽出

抽出した曲線に対して、垂直方向の最上点を検出し、屈曲点として定める。屈曲点の左右の各曲線に対し、水平方向に等間隔に10分割し、「フリーマンコード化を用いた曲線のコード化」節で述べた手法を用いて4種類のフリーマンコード列を生成する。

上記の方法によって200本の手書き曲線を生成し、被験者に判定値を入力してもらったところ、その曲線中、96本が鋭い曲線と判定され、104本がまるい曲線として判定された。第一の実験として、モデル曲線の識別で用いたネットワークの重みを用いて、これらの手書き曲線の識別を試みた。その結果を表2に示す。

表2 モデル曲線の識別に用いた重みを使った手書き曲線(200本)の識別結果

符号化手法	Gray符号化	認識率[%]
通常のフリーマンコード	×	83.5
通常のフリーマンコード	○	87.5

符号化手法	Gray符号化	認識率[%]
一階差分化コード	×	88.5
一階差分化コード	○	91.0

次に第二の実験として、手書き曲線のパターンを再学習させ、その重みを用いて、同様の手書き曲線の識別を行った。ここで、学習用に用いたのは、被験者に新たに書いてもらった100本の手書き曲線である。上述と同様の方法で、画像入力、尖鋭度の判定値(教師信号)入力を行い、輪郭線抽出後、曲線を4種類のフリーマンコード列に変換した。この曲線中、鋭く曲がっていると判定された曲線は46本であり、なめらかに曲がっていると判定された曲線は54本であった。これらの手書き曲線の識別をモデル曲線で用いたニューラルネットワークの学習と同様の方法で学習させた。これによって得られた結果を表3に示す。ここで、学習パターンとは、学習に用いた100本の手書き曲線であり、未知パターンとは、第一の実験で用意した200本の手書き曲線である。

表3 モデル曲線の識別実験の結果

符号化手法	Gray符号化	データ	学習サイクル[回]	認識率[%]
通常のフリーマンコード	×	学習パターン(100本)	3148	99.0
通常のフリーマンコード	×	未知パターン(200本)	-	95.5
通常のフリーマンコード	○	学習パターン(100本)	2692	99.0
通常のフリーマンコード	○	未知パターン(200本)	-	96.0
一階差分化コード	×	学習パターン(100本)	4262	99.0
一階差分化コード	×	未知パターン(200本)	-	88.5
一階差分化コード	○	学習パターン(100本)	4416	99.0
一階差分化コード	○	未知パターン(200本)	-	90.0

4.3 識別実験の評価

学習パターンの識別に関して、ニューラルネットワークは、モデル曲線と手書き曲線の両方ともに、ほぼ完全に識別できることがわかる。一方、未知パターンに関しては、被験者が明らかに判定可能な曲線パターンについては、十分満足のいく結果を得た。すなわち、誤認識となった曲線を観察すると、被験者が見ても、まるいか鋭いかの判定がしづらいものが多かったのである。例えば、表2の通常のフリーマンコード化をしたパターンの認識は200パターン中、正解が167パターンであるが、残りの誤認識パターン33本中、27本の曲線は人間が見ても、鋭く曲がった曲線か、まるい曲線かのどちらともとれるような曲線であった。

通常のフリーマンコードと一階差分化したコードによる認識率の違いに関しては、大きな差異はなかった。よって、曲線の回転に不变な特徴量である一階差分化したフリーマンコードを用いる方が有利であると言える。

Gray符号化に関して結果を見ると、Gray符号化しないパターンより、したパターンの方が全てにおいて、若干、認識率が向上していることがわかる。よって、Gray符号化を用いることにより、コード変換部分の画像誤差に対する耐性をある程度、向上させることができたと考えられる。

次に認識手法の違いによる結果の違いを検討してみる。モデル曲線に関する認識では、学習、未知パターンともに96%以上の高い認識率を示した。これに対し、モデル曲線を用いてネットワークの学習をした重みを利用して、手書き曲線の認識を行ったところ、認識率は83.5%以上となり、かなり低下

する。これは、モデル曲線が屈曲点を中心に対称であり、歪みがないのに対して、手書き曲線は非対称で、歪みがあるため、それらが影響していると考えられる。そこで、手書き曲線を学習した重みを使って、上記の手書き曲線を認識した結果は88.5%以上となった。よって、手書き曲線の学習により、曲線の非対称性や歪みを、ある程度補正する効果が得られたことになる。また、より多くの曲線パターンの学習をすることにより、更に認識率を上げることも可能であると期待できる。

5. 評価実験(2)：両輪郭の速度差によるコード化手法を用いた識別実験

5.1 手書き曲線の識別実験とその評価

最初の実験として200本の手書き曲線を用いて識別実験を行った。これらの手書き曲線は、フリーマンコード化手法で被験者に書いてもらった曲線と同一のものである。曲線画像はイメージスキャナでサンプリング、量子化され、白黒の2値画像としてコンピュータに取り込む。

各曲線に対して、「曲線の外輪郭と内輪郭の速度差を用いた曲線のコード化」で述べたコード化手法を用いて、特徴ベクトルの抽出を行った。例として、このコード化の処理過程を図15に示す。左図は、入力した曲線パターンであり、右図はその外輪郭と内輪郭の対応点を結び、コード化した結果を示している。ここで、屈曲点付近に重要な特徴が含まれることと、ニューラルネットワークに入力するコードの数を同一にするために、採用するコード列は、屈曲点を中心として、両側に10ビット分のコードを採用した。つまり、外輪郭、内輪郭の屈曲点の両側に10ビット分、合計40ビット分のコードとなる。ここで、屈曲点の位置は、簡単のため、手動で入力した。

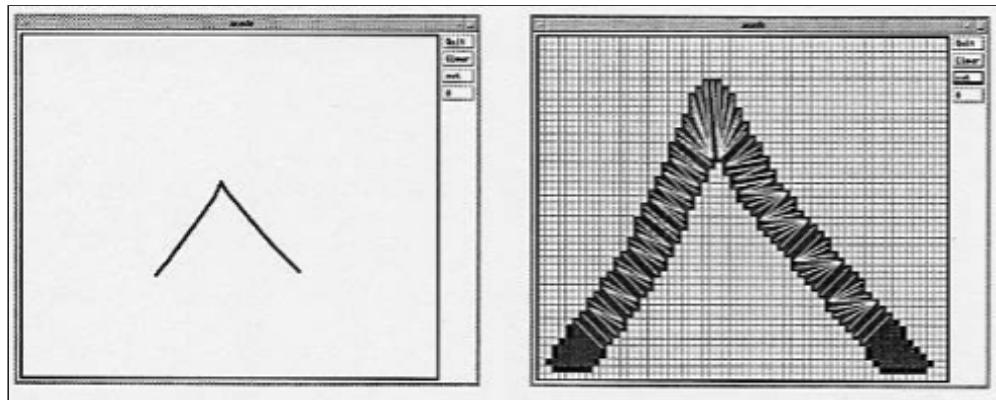


図15 曲線の外輪郭と内輪郭の対応点の生成過程（左図：入力した曲線パターン(鋭く屈曲している曲線)、右図：輪郭間の対応点の生成。各点の対応線を白線で表示した。この曲線パターンに対して、生成されたコード列は、外輪郭 10111111111111111111、内輪郭 110000000010000000001）

コード化した200本の手書き曲線のうち、100本をネットワークの学習用パターンとし、残りの100本を未知パターンとして使用した。ニューラルネットワークの入力層は、コード化した40ビットの値を入力するため、40個のユニットとしている。ネットワークの学習は、上記の100本の学習用曲線パターンのコードを入力層に入力し、出力層には教師信号として、尖鋭度の判定値(0:まるい、1:鋭い)を与え、前述の誤差逆伝搬学習法を用いて行った。

学習した重みを用いて、100本の未知パターンの識別を行った。ネットワークの入力には学習時と同様に、曲線をコード化したビット列を与え、出力として得られた値によって、識別を行う。ここで、出力値が0.5以上ならば鋭い曲線とし、0.5未満の時はまるい曲線として識別を行った。つまり、強制的にどちらかに判定する方式を採用した。

識別の結果、学習パターンに対する認識率が96%に達したところで学習を終了させた場合、未知パターンに対する認識率は84%であり、100%に達したところで、終了させた場合は74%の認識率となっ

た。ここで、やや不十分な認識結果となった理由を解析すると、サンプル曲線の境界線が滑らかでなく、雑音を含んでいること、サンプルが少ないとによるそれらの曲線パターンへのオーバーフィッティング(あてはめすぎ)が主な原因として考えられる。また、誤認識となった曲線パターンに対するネットワークの出力値はしきい値の0.5付近であることが挙げられ、強制的にどちらかに判定する方式にも問題点があると考えられる。そこで、前処理として曲線の境界部分を滑らかにする処理を施し、更に学習パターンの数を300本に増やすとともに、ネットワークの出力値が0.4～0.6の場合は識別不能とする範囲を設けて、再実験を試みた。その結果、認識率は、学習パターン、先程の実験と同一の未知パターンに対して、それぞれ100%、96%となった。

5.2 手書き文字「UとV」と「(と＜」の識別実験

二つ目の実験として、手書き文字の「U」と「V」、それぞれ100文字を用いて識別実験を行った。これらの文字は、手書き曲線の識別と同様、イメージスキャナでコンピュータに読み込まれ、コード化を行う。

ネットワークの学習には50文字の「U」と、50文字の「V」を使用した。手書き曲線の識別と同様の方法で、識別実験を行った結果、認識率は学習パターン、未知パターンとともに100%となった。ここで、未知パターンは、学習で使用した以外の残りの50文字の「U」と、50文字の「V」を使用している。3つ目の実験は、手書き文字の「(」と「<」、それぞれ100文字に対する識別実験である。文字の種類が異なる点を除いては、前述の「U-V」の識別と同様の方法で実験を行った。その結果、認識率は学習パターン、未知パターンともに100%となった。

例として、図16と図17に対応点の抽出例を示した。図16は鋭い屈曲点を持つ「<」に対する例であり、図17はまるい曲線を持つ「(」に対する抽出例である。

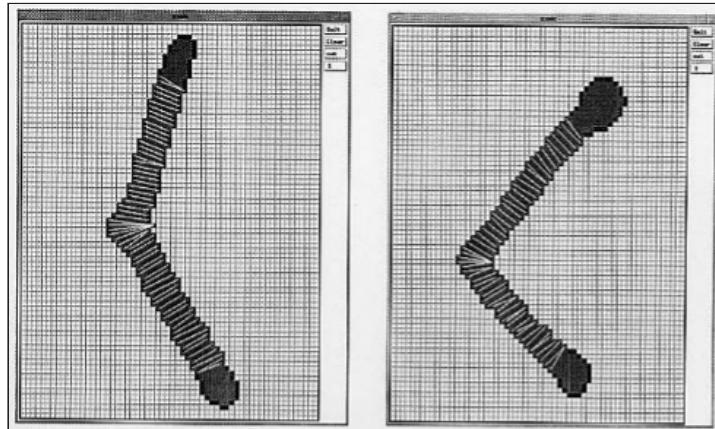


図16 文字「<」に対する外輪郭と内輪郭の対応点の抽出例

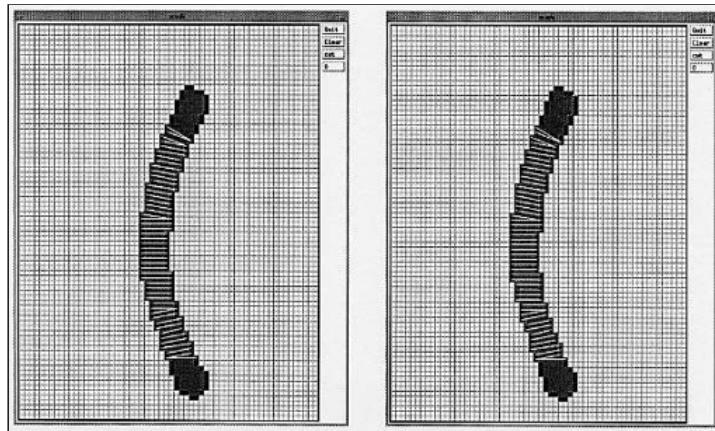


図17 文字「(」に対する外輪郭と内輪郭の対応点の抽出例

5.3 実験のまとめと評価

以上、両輪郭の速度差によるコード化手法を用いた識別実験の結果を表4にまとめた。この表から、手書き曲線に対する認識率より、手書き文字に対する認識率の方が良好であることがわかる。これは、カテゴリー分けされていない手書き曲線の尖鋭度の判定では、主観的な判定により強引に二分判定しているために、境界付近での判定が不安定になっていることに起因すると考えられる。すなわち、被験者のその時の気分によって、判定値が異なる場合が生じ、安定しない教師信号が生成される。一方、手書き文字のように最初からカテゴリーが分けられたパターンに対しては、判定が比較的安定しており、これが良い認識結果を導いた要因である。

表4 識別結果のまとめ

識別対象	サンプル数合計	学習パターン数	未知パターン数	認識率(学習パターン)	認識率(未知パターン)
手書き曲線	400	300	100	100%	96%
U-V	200	100	100	100%	100%
(- <	200	100	100	100%	100%

6. まとめ

ここでは、文字認識における類似文字の組を識別するためのズーミング手順の自動化のための一つの手段を説明した。ここで着目した類似文字の組は、その文字に含まれる曲線が鋭く屈曲しているか、あるいは、なめらかに曲がっているかで、区別されるものであり、これらを識別することを目的とした。

曲線から特徴ベクトルを抽出する方法として、フリーマンコードによる手法と曲線の外輪郭と内輪郭の速度差を用いる手法の二つを用い、階層型のニューラルネットワークを用いて、曲線の尖鋭度の識別を試みた。

その結果、フリーマンコードを用いる手法では、モデル曲線、手書き曲線の未知パターンに対して、それぞれ 96%、88.5%以上の認識率を得た。但し、この判定は強制判定である。これに対し、両輪郭の速度差を用いる手法では、手書き曲線に対し、学習パターン、未知パターンに対して、それぞれ 100%、96%の認識率を得ている。但し、ここでは、ネットワークの出力値に基づき、認識不能という出力も許している。次に、同様の方法により、手書き文字「U-V」と「(-<」の識別実験を行い、学習パターン、未知パターンの両方に対して 100%の認識率を得た。

これより、曲線から適切な特徴ベクトルを抽出、コード化し、ニューラルネットワークに学習させることにより、自動的に人間の能力に匹敵する識別処理を構築できることがおわかりいただけたと思う。

memo