2次元表現入力のためのオンライン手書き文字認識†

松下武史†† 佐伯元司††

図面，数式などの2次元表現の入力に用いる実用的なオンライン手書き文字認識を開発した．これは2次元表現で使用される各種の文字を対象としている．さらに，本システムは任意の位置に任意の大きさで書いた文字を認識し，ユーザが文字の大きさや位置の情報も入力できるようになっている．マイクロコンピュータを使用し，練習用の援助機能を設けることで，コンパクトでかつ十分な認識率をもった実用的なシステムにまとめた．種々の文字，記号を扱うために，文字の特徴を分析した結果に基づき，効率よく特徴を抽出，分類している．また，変形した文字を統一的に処理するアルゴリズムを組み込んだ．書く位置の制限をなくすため，続けて書かれたストロークが有意な文字を構成するかどうかをストロークの型と位置関係から判断し，自動的に文字の区切りを判定する．誰でもが自由に書いた文字の認識率を 95% 以上にすることは容易ではない．これは大部分，書き順・字体の相違，筆記具に対する不慣れなどが原因であり，この対策として，練習援助機能を組み込み，システムが大きくなるのを避けた．ほとんどのユーザは1時間程度の練習で 98% の認識率を得ることができた．

1. まえがき

本論文は2次元情報の入力に使用することを目的とした手書き文字・シンボル（以下文字と略す）のオンライン認識について述べる．

図面や数式といった2次元表現においては文字だけでなく，それらが書かれている位置や大きさも重要な情報である．さらに2次元表現では各種の文字が用いられる．キーボード装置は1次元の文字列しか扱えず，また文字セットが限られている．このため2次元表現の入力にはキーボードは適当ではなく，オンライン手書き文字認識を用いて手で書いたものを直接入力することが望ましい．

このための文字認識は

1) 2次元表現で用いられるさまざまな文字が取り扱える．

2) 文字を書く大きさ，位置に制限がない．

3) どこででも利用できるコンパクトなシステムで，かつ認識率が高い．

ことが要求される．

2次元表現で使用される文字は英数字（小文字も含む）のほかに，応用分野によって，種々の形をしたシンボルが含まれる．認識システムは，あらかじめその形や種類を制限せず，これらを統一的に取り扱うことができなければならない．文字の特徴を分析した結果に基づき，本論文では人が文字を意識的に区別する際に注目する特徴を用いて識別を行う．これにより，文字を区別するための最適な特徴を使うことができ，正しい認識や効率的な分類を行うことができる．

文字が任意の場所に任意の大きさで自由に書けるためには，文字を続けて書いたとき，入力ストローク列のどこからどこまでが一つの文字を構成しているかを自動的に判定することが必要である．本研究ではストローク型の組合せを使って，効率的に文字の自動分離を行う方法を考案した．

入力装置としての実用性を上げるため，種々のシステムに組み込めるようにマイクロコンピュータを使用し，汎用性の高いコンパクトなシステムとした．このため，処理速度・メモリ容量の面からも十分実用になる簡単なアルゴリズムが必要である．このことから，システムに学習能力をもたせたり，文脈情報を識別に使用したりすることを避けた．したがって，ここでは認識対象を1文字だけ取り出しても誰にでも読める文字に限定している．

認識率をより向上させるためには，入力ペンや文字の書き方についてユーザが練習することが必要である．本システムではユーザが筆記練習をする際，これを支援する機能を組み込んだ．

オンライン手書き文字認識は従来から種々試みられてきた[1)~7)]．文字から抽出する特徴パラメータにはストロークの方向系列[3)~6)]やフーリエ係数[7)]などがある．しかし，これらは単一の特徴を用いているため，一般には，おのおのの文字の最も重要な特徴を表していない．また，特徴パラメータのほぼ等しい文字の区

† On-Line Recognition of Handwritten Symbols for Two-Dimensional Data Input by TAKESHI MATSUSHITA and MOTOSHI SAEKI (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

別は例外的に扱っている．本システムでは各文字ごとに，その文字に適した特徴で分類する．種々の特徴を統一して取り扱うために，計算機による処理方式を検討し[8),9)]，任意の特徴が記述できるtree構造の辞書を使って，これを効率的に実現している．文字の分離方式の提案は少なく，従来の文字認識アルゴリズムはすでに文字分離が行われているということを前提にしたものがほとんどである．文字の分離は決められた枠内に1文字ずつ文字を書いたり[4),5)]，時間による方式[4)]が多い．Millerの比較的単純なストローク間の距離による判定[3)]は，2次元表現に使われるさまざまな形の文字に拡張するには不十分である．本研究のストロークの形と位置関係を組み合わせて文字の分離を判定する方式は，一般の文字に適用でき，これを効率的に行う方法を実現した．また，これまでのシステムは大型計算機もしくはミニコン上で実現されている[1)~7)]．本研究では特に演算時間を考慮し，マイクロコンピュータで十分実用になるようにアルゴリズムを工夫した．認識率向上のため，従来のシステムにはなかった筆記練習支援機能を組み込んだ．

2. 文字の特徴

2.1 特徴抽出の方針

文字は人間同志の通信の手段であり，人は一定の規則に従って文字を書き，それを解釈している．人が一般の2次元の形状を識別するプロセスは明白ではないが，文字ではそれぞれの形状の特徴を記述し，どのようにしてそれらを互いに区別するかを述べることができる．たとえば，'*a*'の場合では輪を描き，その後上から下へ進み，左回りで右上へはねる．'*g*'では前半までは同じであるが，最後は右に回って右上へはねる．文字の特徴にはストロークの方向・曲がり方・位置関係などいろいろあるが，どの特徴が重要であるかは個々の文字の形により，また似た文字があるか否かによっても異なる．多種の文字を扱い，その形をあらかじめ制限しないために，できるだけこの人が意識して区別する特徴を用いて分類・識別することを基本の方針とする．ここで問題は

1) 特徴を抽出するアルゴリズムの問題：たとえば，「輪を描く」などの特徴をどのように計算機で判定するか．

2) 人が暗黙のうちに使用している特徴をすべて明白に記述できるか．

2次元の特徴抽出を人と同じように行うことは現実にはむずかしく，計算機で処理しやすいやり方に置き換えなければならないこともある．このため，変形した文字では人には読めても計算機では認識できないことが起こる．これを最小限にするため，実験により，どのような変形が多いかを調べ，その処理を特徴抽出に組み込む．さらに文字の区別に必要な特徴だけでなく，追加のチェックを加え，予想外の変形をした文字はできるだけリジェクトする．オンライン認識ではリジェクトはユーザに知らせることでただちに入力し直すことができるからである．

2.2 抽出する特徴

人が文字を記述するときに用いるおもな特徴は

1) ストローク各部の曲がり方：直線，左回転，右回転，どれくらい曲がっているか，輪になっているか．

2) ストローク各部の方向：直線ならその方向，曲線ならどちらを向いているか．

3) 尖点の有無

4) 交点・接近個所の有無

5) 特徴点の位置：ストロークの始点，終点，尖点，交点，縦横方向の極大・極小点などの位置関係

6) ストローク各部の大きさ

さらに複数ストロークで構成される文字においては

7) 文字を構成するストロークの型

8) ストローク間の位置関係：上下左右の方向，距離，交差しているか．

9) ストロークの大きさの比

1)の曲線の曲がり方についてはストロークの接線方向の変化を調べ，それによって判定を行う．また，交点や接近個所は人にとっては容易に抽出でき，わかりやすい特徴であるが，計算機では時間のかかる処理である．ところが文字を観察した結果，ほとんどの場合，交点以外の特徴点の位置関係を調べることで交点の有無は判定でき，それで十分であることがわかった．その他の特徴は直接抽出する．

2.3 文字分離の方針

ストローク間の距離は，文字の区切りを判定する重要な基準であるが，人は単純に一定の距離以上離れているかどうかで区切りを判定しているのではない．図1(a)の'a','b'や'L','T'はかなり接近し，多少重なっていても二つの文字に読むが，同図(b)の'E'や'H'はストロークが離れていても一つの文字に読む．すなわち，人は複数のストロークの形と位置関係を見て，それらが一つの有意な文字を構成しうるかど

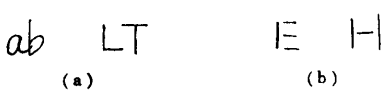


図 1 文字の分離
Fig. 1 Character separation.

うかで文字の区切りを判断している．本システムではこの方式を使って文字の自動分離を行う．

2.4 識別の手順

本システムの識別の手順は以下のとおりである．

1) 入力ストロークに沿って長さと接線方向を求める．

2) 短区間で方向が急変している個所を見つけ，尖点とする．ストロークを尖点で分割する．これはストローク型の判定を行いやすくするためである．分割したものをセグメントと呼ぶことにする．

3) 各セグメントごとに，方向の変化と大きさ，特徴点における方向と位置関係，ストローク中でのセグメントの出現順といった特徴を抽出し，セグメントを分類する．

4) セグメント型の大きな特徴を用いてストロークを分類する．

5) ストロークの出現列，およびその位置関係などの特徴から文字の区切りを判定し，識別する．

3. 文字の変形

認識する文字にはそれぞれ標準となる字体を定義し，それを基準として識別する．前章で述べた特徴抽出は人のそれと完全には同じでないため，標準字体から変形した文字を正しく認識できないことがある．

相違の第1は，人はストローク全体を同時に見るが，ストロークに沿って方向変化を調べていく方法では局所的な方向変化に影響を受けやすいことである．人が無視するような局所的なノイズを尖点と判定したり，筆記者の手ブレやハネなどのために回転方向の判定を誤ったりする．とくにセグメントの始点・終点の近傍では，ペンの押えやハネなどの急激な方向変化がよく現れる．

第2は，人にとっては尖点の重要度は文字によって違うことである．**図2**(a)のように尖点が丸くなって消えてしまっても，同図(b)のように丸い曲線中に尖点が出現しても認識できる．ところが 'U' と 'V' は尖っているか否かで区別する．尖点はどのように変形するかを調べると，**図2**(c)のように，もともと方向変化の小さい尖点（弱尖点）では，なまって消える場合と逆に尖点の前後で逆方向に曲がり強調される場合とがある．方向変化の大きい尖点（強尖点）では，同図(a)の '3' のように小ループになってしまうことがあるが，その個所の方向変化が急であるという性質は保存される．

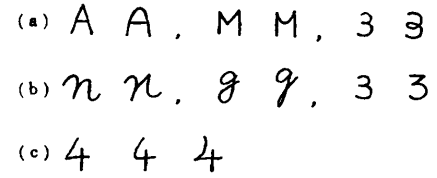


図 2 尖点の変形
Fig. 2 Distortion of cusp points.

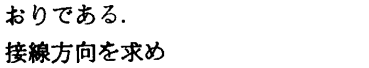


図 3 直線部の湾曲
Fig. 3 Distortion of lines.

第3は，**図3**のように，標準パターンでは直線である個所が変形して曲線に書かれる場合である．同図の 'H' の標準パターンでは第1ストロークは '|' であるが，変形パターンでは ')' である．単独ではもちろんこれらのストロークは，直線と括弧などに区別して扱わなければならないが，この後に第2，第3ストロークが続くと，同一視される．つまり ')' というストロークは状況により，右括弧か直線のどちらかに解釈される．

本システムでは以下のような方法でこれらの変形を取り扱う．

1) ペンの押えやハネの対策：回転方向の判定では，セグメントの始点・終点付近の一定範囲内の方向変化は判定から除外し，短いセグメントは無視する．

2) 尖点が消失する変形の対策：(a)セグメント型の識別ができなかったときは尖点がなまって検出できなかったとみて，セグメント中で最も方向変化の激しい個所で分割し，再度調べる．(b)短区間に変曲点が二つ出現したときは，その中点を尖点として分割する．(c)消失の可能性のある弱尖点をもつ文字については，両方のパターンを辞書に登録しておく．

3) 余計な尖点が出現する変形の対策：曲線部に尖点が出現する可能性のあるセグメントについては，以下の修正規則を使ってセグメントの合成を行う．たとえば，変曲点を含むパターンにおいてはよく尖点が出現し，図2(b)の 'n' のように，'∩' 型のセグメントは '↑' と '∪' の二つのセグメントに分割されることが多い．そこで '↑' と '∪' というパターンが連続

したならば，これを合成し，'∿' に置き換える．修正規則は全部で四つあり，字体のバリエーションを減らすためのものもある．修正規則が適用できない変形については辞書を2通り用意する．

4) 直線部が湾曲する変形の対策：曲がり方の小さい曲線は曲線の性質を記録しておくが，いったん直線と判定し，ストローク型の判定，文字の識別を行う．その結果，セグメントが直線か曲線かの判定が必要になったならば，記録しておいた特徴を使って判定する．

4. 特徴抽出のアルゴリズム

本章では，タブレットから入力されたストロークから特徴を抽出し，型を判定する手順について述べる．

4.1 前処理

タブレットから入力された座標データに以下のような前処理を施す．

1) ストロークの始点の補正：ペンスイッチの動作の遅れにより生じるストロークの始点近傍のデータの欠けを補うため，ペンダウン直前の1点をデータに付け加える．

2) 平滑化：筆記者の手ブレやディジタル化の誤差を除くために，隣り合う2点の移動平均をとる．

3) 標本化：冗長な点を取り除くため，ウィンドゥフィルタリング[2),6),9)]を施す．曲がりの急激なところではウィンドゥを小さくし，逆に直線部では大きくする．またフィルタリングと同時に各点の接線方向と始点からの長さを求める．

4) 終点付近のハネの除去：終点の近傍で方向変化の激しい個所を捜し，もしあればその個所から終点までをハネとみなして除去する．

図4 に前処理後データと接線方向の変化のグラフを示す．

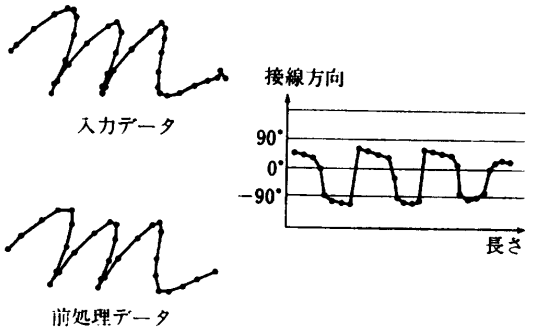
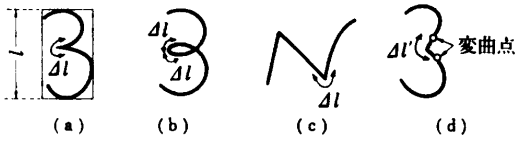


図 4 前処理後データと接線方向の変化
Fig. 4 Preprocessed data and direction of tangent.



l：ストロークの大きさ

図 5 尖点の検出
Fig. 5 Detection of cusp points.

4.2 尖点の検出

ストローク中の尖点を検出し，その個所で分割する．尖点の前後では回転方向などの特徴が不安定であり，分割することにより，安定した特徴を容易に抽出でき，型の判定が行いやすくなる．

ストロークに外接する方形の長辺の長さをストロークの大きさ l とする．尖点の検出はまず，筆跡に沿って距離 Δl（ストロークの大きさ l の16分の1）の区間で方向変化が90°以上ある個所を見つける．そのような個所のうち，

1) Δl 以内で方向の変化が十分大きくて，140°以上ある（**図5** (a)）．

2) 条件を満たす区間が連続し，全体として140°以上の方向変化がある（小ループの尖点：図5 (b)）．

3) その直前あるいは直後の筆跡が逆方向に曲がって，方向変化が100°以上ある（図5 (c)）．

また，回転方向を調べていったときに

4) $\Delta l'$（ストローク長の10分の1）以内の区間に変曲点が続く（図5 (d)）．

のうち，どれか一つを満たすものを尖点とする．各パラメータ値は実験により定めた．

4.3 セグメント型の判定

検出した尖点でストロークをセグメントに分割する．各セグメントの回転方向，始点・終点の位置関係や方向，ストローク中での出現順を求める．さらに縦横方向の増減を調べることにより，セグメントの向きと極値点を求める．ただし，各セグメントの始点・終点付近のストローク長の15分の1以内の区間での方向変化は無視し，ストローク長の20分の1以下の長さのセグメントは無視する．直線か曲線かの判定は方向変化が45°以上あるかどうかで行う．

型を判定するための辞書は処理を高速に行うため，decision tree 構造とし，まず回転方向，セグメントの向きといった特徴で大きく分類し，しだいに各セグメント固有の特徴で細かく分類する．

セグメント型に分類できなかったときは，そのセグメント内で最も方向変化の大きい個所で分割し，もう

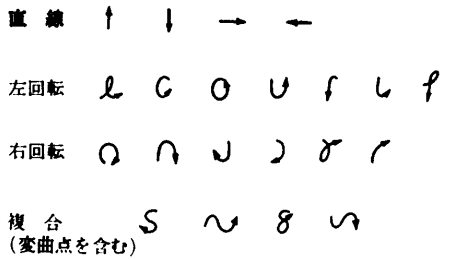
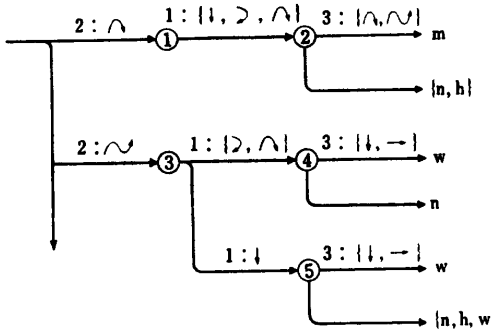


図 6 セグメント型
Fig. 6 Types of segments.



ノードの左に付けられた番号とシンボルはセグメント順と型を表す．たとえば 2: ∩ ならば「第2セグメントが ∩ である」ことを意味している．セグメント番号は必ずしも必要ではない．ノードの下から出ている枝は入力データが番号で指定されたセグメントをもっていない場合である．

図 7 ストローク型判定 tree
Fig. 7 A part of the tree for classification of stroke types.

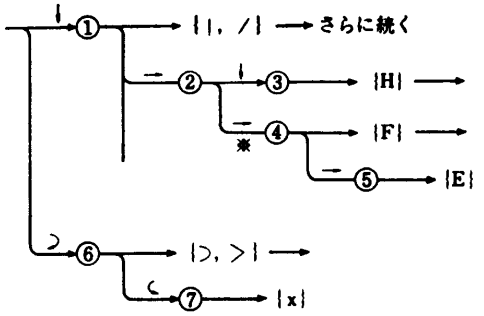
一度判定をやり直す．

図6に認識対象文字を英数字にした場合のセグメント型を示す．たとえば，'a','g' の場合は，それぞれ 'ↄ' と 'ɾ','ↄ' と 'ɾ' という二つのセグメントから成っている．

4.4 ストローク型の判定

判定されたセグメント型を使って，ストローク型を分類する．まず，セグメントの出現列に対し，修正規則が適用可かどうか調べる．適用可能ならば，その規則に基づいてセグメント型を変換する．変換前のセグメント型も分類に使用する場合があるので，捨てないで残しておく．

英小文字には，'m','m','m' などのように書き始めや書き終わりのセグメントにさまざまなバリエーションがあるため，不変のセグメントにまず注目して分類を行っていく．'m' の場合では第2セグメント '∩' が不変である．このような大きな特徴を用いることにより，ストロークを確実に分類し，範囲を狭めることができ，処理の高速化に有効である．ストローク型分類用の辞書も tree 構造になっている．図7に tree の一部を示す．



ノードの左上のシンボルはストローク型を表す．各枝に付けられている結合条件は省略したが，たとえば※には「第1,2ストロークを囲む方形領域内に第3ストロークの重心が含まれる」という条件が付加されている．

図 8 文字の分離・識別 tree
Fig. 8 A part of the tree for character separation and identification.

5. 文字の分離と識別

文字の自動分離はストローク型の出現列から有意な文字を構成しうるかどうかを調べることによって行う．このために図8のような識別 tree を用意した．この tree の各枝にはストローク型と以下のような条件が付加されている．

1) ストローク間の位置関係

2) 似た形状の文字を区別するために必要な追加の条件—ストロークの回転角，始点から終点への方向，始点・終点付近の方向，ストロークの縦横比など．

次に入力されたストロークの型に合う下位への枝がなかったり，あってもその枝に付けられた条件を満足していなければ，そこを文字の区切りと判定し，分類を行う．唯一の文字に分類できても，そこでチェックをやめず，文字として正しい形をしているかどうかの余分のチェックを行い，あいまいな文字をリジェクトする．

文字の分離に失敗したとき，つまり図8のノード②のような有意な文字が全く存在しない個所が文字の区切りと判定されたときは，そこまでに通ってきた経路中で有意な文字を含む最も近いノードで分離し直す．この場合，ノード②に最も近いノードは①であるから，ここで分離が起こり，識別結果は '|' か '/' になる．

この他に，一部のストローク型に2重の意味をもたせている．たとえば，回転角が 90° 以下の '⊃' には本来の形状の意味以外に '↓' という直線ストロークの意味も与える．'⊃' というストロークが入力された

場合，最初はこれを‘↓’とみなして分類を行っていく．分類を行っていった結果，どの複数ストローク文字にもならなかったときは，本来の形状に戻して再び識別を行っていく．この例では，最初ノード①に達するが，識別に失敗すると，tree の根に戻り，ノード⑥に達する．

文字の分離を行う手段としてはこの他にストローク間の入力待ちの時間を使用している．ストロークを書き終えてから 0.8 秒以上経っても次のストロークが入力されないときは，そこで無条件に文字の分離を行う．

6. 練習機能と認識実験

6.1 練習機能

誰が書いた文字でも 90% くらいまでの認識率は比較的容易であるが，それを越えると実現がむずかしくなる．実験の結果，これは書き順・字体が異なったり，筆記具に不慣れなことが主な原因であるとわかった．タブレットのペンは使いにくく，慣れないと正しくデータが入力できないことがある．これらの処理を全て組み込むと，システムは非常に大きなものとなる．また，文字のなかには，大文字・小文字の‘O’などのように，文脈を使わない限り判断できないものもある．これらは，‘O’,‘𝒪’のように，ある程度字体を変えざるをえない．したがって，最初にユーザに若干の練習をしてもらい，システムに慣れてもらうのが現実的である．そのために練習を援助する機能をシステムに組み込むことにした．

練習援助機能の目的は，ユーザが

1) 自分のくせを知り，標準の書き方を覚える．
2) タブレットペンに慣れる．

ことである．書いた文字が正しく認識されなかったとき，計算機が読み込んだデータと抽出した特徴をわかりやすく図形で表示して，ユーザに書き方の悪い所がただちにわかるようにした．図 9 に表示例を示す．これは抽出した特徴のうち，ストローク数とセグメント型を使って合成したもので，必要なら，他の特徴も表示することができる．また表示だけからはわからないときのために標準字体表も用意している．

(a) (b)

(a) 入力データ
(b) 抽出した特徴による表示 □は尖点を表す

図 9 練習援助機能
Fig. 9 Display of user practicing mode.

実際の練習では，計算機が文字を指示し，筆記者の書いた文字が正しいかどうか自動的に判定する．

6.2 認識実験

練習の効果がどれくらいあるかを調べるために以下のような実験を行った．全く予備知識の与えられていない複数の被験者に本システムの練習機能を使ってもらい，練習過程での認識率を測定した．実験対象文字は英数字 62 文字＋特殊記号 10 文字とした．また数日後，同じ被験者に対し練習効果がどれだけ残っているかを調べるために同じ実験を行った．

8 人の被験者に対し実験を行い，認識率の変化を測定した結果を図 10 に示す．1 回の練習で全対象文字

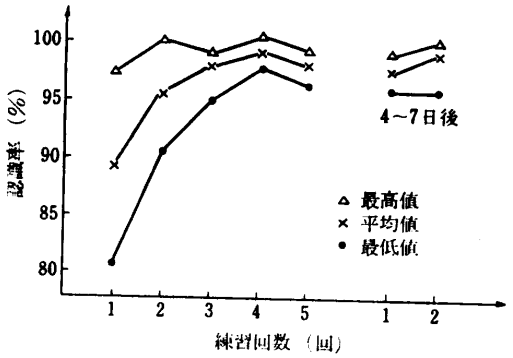


図 10 練習過程における認識率の変化
Fig. 10 Recognition rate in user practicing.

表 1 誤りの原因
Table 1 Causes of errors.

	第1回目		第4回目	
	誤認識	リジェクト	誤認識	リジェクト
1. 書き方があいまい	3.9	0	0.47	0
2. 書き順，字体が違う	2.4	1.9	0.21	0.1
3. ペンの不慣れによるデータ欠損	0.9	0.9	0.12	0.35
4. 尖点が検出できなかった	0.7	0.3	0	0.12
5. 手ブレによる尖点出現	0	0.8	0	0.12
合 計（誤った割合）	7.9	3.9	0.8	0.7

表中の数値は全入力データ数に対する割合 (%)

(誤りの例)

1. r を v と誤認識
   O (小文字) を大文字と誤認識
2. G を ①G→② と書いた
   E を ②→③→① と書いたなど

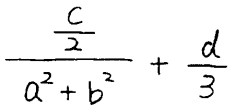


図 11 数式入力例
Fig. 11 Examples of input of algebra expression.

を1回ずつ書いた．**表1**に第1回目および第4回目の主な誤り原因を示す．

第1回目の認識率は 80～97% と幅があった．英字や特殊記号は，字体や書き順が統一されていないため，実験中にもさまざまな書き方が見られた．文字の形がもともと似ており，本システムでは字体を変えて区別している文字の誤り（たとえば，小文字のoと大文字のO）やタブレットペンに慣れていないことからくる誤りも多い．これらを合わせると，誤り原因の80%以上になり，筆記練習が必要であることがわかる．

第4回目では，これらが改善され，認識率が 97～100% となった．ここまでの練習時間は30分～1時間，文字数にして200～300文字程度であった．逆に4回目を過ぎると，認識率が若干落ちてくる．これは筆記者の手が疲れてしまったためと考えられる．

練習の効果は数日後でも十分に残っており，模準字体を忘れているという現象は二，三見られただけであった．

6.3 文字の自動分離機能の評価

文字の分離はよい結果が得られ，2次元表現の入力に十分使用できることが確認できた．**図11**に数式を入力した例を示す．数式中の文字は完全に分離でき，正しい認識ができた．

いったん文字の分離個所を誤ると，その後の文字の分離にまで影響がおよぶ可能性があるが，実験では1文字後までしか影響が現れなかった．これは，分離を誤るとほとんどリジェクトになり，そのため筆記者が気づいて筆を休めるからである．文字分離を誤り，誤認識をひき起こしたのは，人が見てもまぎらわしい場合だけである．

7. むすび

本研究ではマイクロコンピュータ上で実現可能なオンライン手書き文字認識の一手法を示した．実験には，分解能 0.1 mm，サンプル速度 64 点/秒，大きさ 30 cm×30 cm の磁歪式タブレットと 8080 マイクロコンピュータ（クロック 2MHz）を使用した．使用メモリ量はプログラム（作業領域も含む）が約 24 kB，辞書が 6 kB である．処理時間は文字によっても違うが，0.5～1.5秒であった．実験では人の筆記速度はこれよりも若干速かったが，2次元表現に用いられるデータは短く区切られることが多いこと，最近のマイクロコンピュータはもっと処理が速いことなどの理由で問題はないと思われる．実験では筆記文字の大きさは高さで 3～8 mm であった．4～5 mm 程度の文字までなら十分認識できる．より小さい文字では，タブレットの入力ペンが使いにくく，一般には入力がむずかしい．

ストロークの分類・識別は人が文字を区別するときに注目していると考えられる幾何学的特徴を抽出して行った．この手法は英数字・特殊記号だけでなく，ループや交点の検出機能を付け加えれば，ひらがなやカタカナの認識にも拡張可能である．また文字分離のアルゴリズムは漢字のヘンやツクリを分離・識別するのにそのまま応用でき，漢字の認識にも有効である．

認識実験の結果や数式入力例より，本システムが2次元表現の入力に有効であることが確認できた．認識率は1時間程度の練習でほとんどの人が 98% 以上となった．入力装置として使用したタブレットペンは非常に使いづらいため，一度に大量の文字を入力するのは筆記者に対してかなりの負担になるであろう．使いやすい筆記具の開発が今後の課題である．

謝辞 本研究に対し，日ごろ，ご指導いただいた東京電機大学の穂坂衛教授，本学の榎本肇教授，ならびに熱心にご討論いただきました研究室の皆様に深く感謝します．

参考文献

1) Suen, C. Y., Berthod, M. and Mori, S.: Automatic Recognition of Handprinted Characters, *Proc. of the IEEE*, Vol. 68, No. 4, pp. 469-487 (1980).
2) Hosaka, M. and Kimura, F.: An Interactive Geometrical Design System with Handwriting Input, *Proc. IFIP '77*, Vol. 7, pp. 167-172 (1977).
3) Miller, G. M.: On-line Recognition of Hand-generated Symbols, *AFIPS Proc.* FJCC, Vol. 35, pp. 399-412 (1969).
4) Groner, G. F.: Real-time Recognition of Hand-printed Text, *AFIPS Proc.* FJCC, Vol. 29, pp. 591-602 (1966).
5) 寺井秀一，中田和男：手書き漢字，片仮名文字のオンライン実時間認識，信学会論文誌，Vol. 56-D, No. 5, pp. 312-319 (1973).
6) 藤原塩和，池田克夫，富永善治，清野 武：接線ベクトル列を用いたオンライン手書き文字の認識，情報処理，Vol. 17, No. 3, pp. 191-199 (1976).

7) 荒川弘熙，増田　功：手書き文字のオンライン認識，信学会論文誌，Vol. J59-D, No. 11, pp. 809-816 (1976).
8) 松下武史，佐伯元司，依田文夫，穂坂　衛：手書き文字・シンボルのオンライン入力，第21回情処学会予稿集，pp. 755-756 (1980).
9) 佐伯元司，松下武史：2次元データ入力のためのオンライン手書き文字認識，情処学会人工知能と対話技法研究会，23-3 (1981).

(昭和56年11月4日受付)
(昭和57年2月16日採録)