修士学位論文

オンライン数式認識手法に関する研究

東北大学大学院工学研究科 電気・通信工学専攻
日下部　淳

目 次

図目次

表 目 次

第1章

序論

1.1 背景

安価で性能の良いパソコンが続々と発売されたことや、多くの機能を持ち、初心者から上級者まで誰にでも使える文書作成ソフトが開発されたことにより、コンピュータを用いて文書作成をするという場面はごくありふれたものになった．これは各研究機関の研究者にもあてはまり、自らの成果を世に公開するための学術論文を書く際には、ほぼコンピュータが使用される．これはもはや通例になっていると言ってもいいであろう．この、学術論文の作成に当たり、テキストの入力に比べて数式の入力には長い時間がかかる場合が多い．これは、キーボード上に数式に対応するキーがないためである．そのため、数式を入力するためには専用のシステムを使用するが、これは大きく分けて2種類に分類される．1つはマークアップ言語によるもので、もう1つは数式エディタによるものである．前者は、キーボードから直接数式に対応するコマンド入力をすることができるため、スムーズな入力が可能である．しかし、コマンドを覚えるためにはある程度の熟練が必要であり、初心者にとって平易なものとはいえない．また、コマンドは基本的に記号の羅列であり、入力中の数式を視覚的にイメージすることは難しい．これとは逆に、後者の場合は画面を見ながら各シンボルを二次元的に配置していくことから、視覚的なイメージは容易であるが、マウスとキーボードを併用するためスムーズな入力という点では

問題が残る.

その一方で、ここ約10年、インターネットの利用者数は爆発的な伸びをしてきた.その結果、ＰＤＡに代表されるモバイル機器や、タブレットのようなオンライン入力装置の技術の発展に結びつくようになった.それに伴い、近年オンライン文字認識の研究が盛んになっている[1].

これらを背景に、手書きの数式をオンラインで認識させるための研究が行なわれている[2][3].このようにタブレット等で入力した数式を認識してコード変換するシステムは、視覚的にも操作性の上でも優れた数式認識システムと言えるだろう.青島ら[4]はストロークの位置関係と解析結果をもとに数式における二次元構造の認識をストローク単位で行なうことにより数式認識を行なう、数式入力支援システムを提案した.この手法の評価すべき点は、前述のとおり逐次認識を行なったところにある.現状では手書き数式認識の精度は一般にあまり良いとは言えず、個々の要素の認識率は良いものの、構造の完全一致率という観点からは未だに改善の余地を残している.しかし、逐次認識を行うことによりエラー箇所をその都度修正することが可能で、認識に要する時間は最終的に短縮されることになる.

このような数式認識の研究はオフライン認識の分野でも行なわれているが[5][6][7]、この場合の目的の多くは、過去に印刷物の形で出版された文書を電子化すること、すなわち文書画像データをテキストデータに変換することにより、データ容量を減らし、検索性を向上させる点にある.これまでの研究例を見てみると、オンラインとオフラインに関わらず、数式内の数学記号に注目して記号と文字との垂直・水平関係を調べ、数式を木構造に置き換えて表現することにより認識を行なっている.ところが、学術論文作成の際に時折現れる行列構造については、これを示す明確な記号が存在しないことから、認識は困難とされ、多くの研究では認識対象とはされなかった.そこで豊住ら[8]は、同グループが以前に構築した既存の数式認識システムに、行列認識機構を付加したシステムを提案した.このシステムでは、各要素間の垂直関係を検出することにより、行列の認識を行なっている.しかし、これは行列内の全ての要素が埋まっている場合に限定されており、行列の表記法の一つである連続記号や領域要素を用いた「略記」には対応していない(この「略記」を含め、本論文内で用いる用語については、図1.1で説明する).略記にも対応した行列認識手法は金堀ら[9]が提案しているが、これは活字で書かれた行列を対象としたオフライン認識であり、その性格上、行列全体を見ることにより解析を行なっ

ている．金堀らの手法を逐次認識に適用することができれば、略記を含む行列に対応したオンライン数式認識が可能となる．

1.2 本論文の目的

本論文では、略記を含む行列を対象として行列構造を逐次認識するシステムを開発することを目的とする．このシステムでは、文字認識等はすでになされているものとして、行列構造の解析に主眼を置く．ここで、本論文内で用いる用語を説明しておく．図1.1に示すように、連続記号、領域要素を含む行列を略記を含む行列と呼ぶ．

1.3 本論文の構成

本論文の構成は以下のとおりである．

第1章 序論

本研究の背景と目的、論文の構成を述べる．

第2章 提案する認識手法

提案するシステムの各部の処理について述べる．

第3章 構造認識実験

提案したシステムを用いて構造認識実験を行なう．

第4章 考察

実験結果についての考察を行なう．

第5章 結論

本研究の成果と今後の課題について述べる．

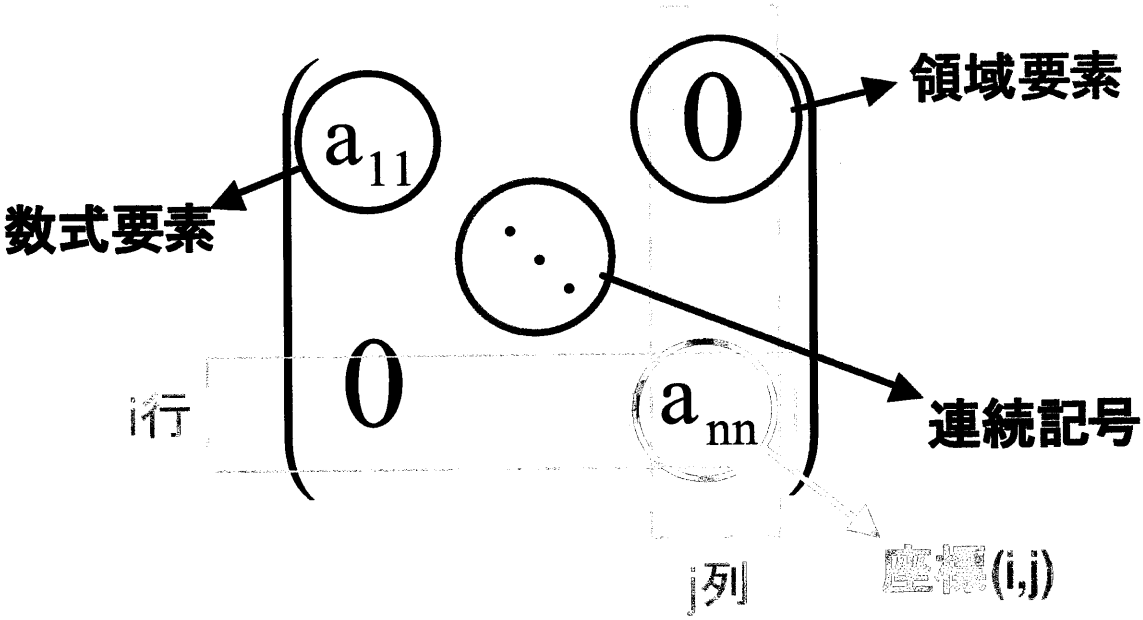


図 1.1: **本論文内で用いる用語**
i 行 j 列に存在する要素の座標を (i,j) とする．また、行列を構成する一般要素を数式要素、連続する数式要素を省略した記号を連続記号、同じ数式要素が存在する領域を省略したものを領域要素と呼ぶ．連続記号、領域要素を略記と呼ぶ．

第2章

提案する認識手法

提案するシステムのフローチャートを図 2.1 に示す．本システムでは、ストロークが入力される度に文字認識を行ない、そのタイプによって数式要素処理、連続記号処理、添え字処理のいずれかの処理を行なう．連続記号処理後は条件により入力待ちモードとなり、このときの流れは他の場合とは異なる．この処理を全てのストロークの入力が終了するまで続け、最後に領域要素処理を行なう．以下、各処理ブロックにおける処理について説明する．

2.1 ストローク入力

入力はマウスによる描画によって行なう．この軌跡の (x, y) 座標は時系列データ (図 2.2) であるが、それをもとに、ストローク毎に外接矩形を作成する．直前のストロークとの重なりがある場合には、複数画の 1 つの文字と判定し、これらを統合して新たな外接矩形を作成する (図 2.3)．以降は、ストロークそのものではなく、この外接矩形を参照することにより各処理を行なう．

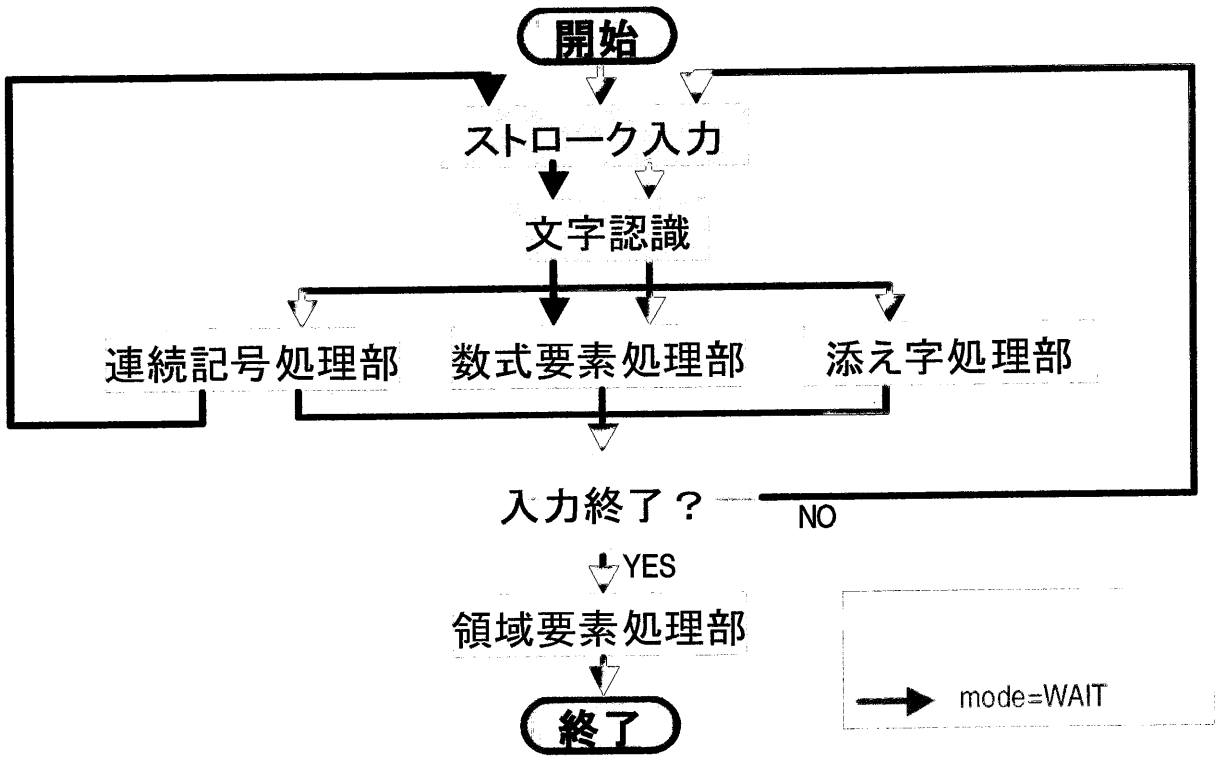

開始
ストローク入力
文字認識
連続記号処理部
数式要素処理部
添え字処理部
入力終了？
NO
YES
領域要素処理部
終了
mode=WAIT

図 2.1: システムの流れ

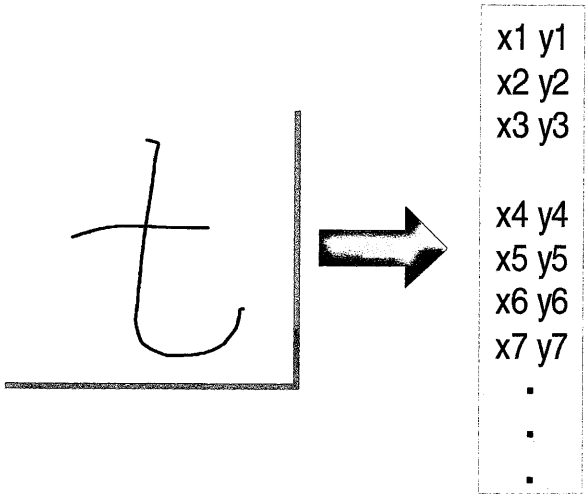


図 2.2: **時系列データへの変換**
時系列データはマウスの軌跡の座標を並べたものである．マウスのボタンを離すと空白行が書き込まれる．

2.2　文字認識

ここでは、入力された文字のタイプを数式要素、連続記号、添え字の3つに分類する．まず、時系列データを参照し、画素数によってストロークの長さを調べる．これが閾値より小さいものをドットとみなす．そしてこのドットが3つ以上連続して入力された場合、これらを連続記号とする．この様子を図2.4に示す．

次に、入力された文字とその直前に入力された文字を参照して、添え字の判定を行なう．この判定には2つの文字の外接矩形の位置関係を用いる．入力された文字がその直前に入力された文字の右側にあれば以下の処理を行なう．右側の文字の中心の高さが左側の文字の下端から、左の文字の高さ H の l%のところより低く、かつ右側の文字の上端の高さが左側の文字の上端から、左の文字の高さ H の l%のところより低い場合、右側の文字は左側の文字の添え字であると判定する．(図2.5) この領域の選び方については、文献 [4] を参考にした．

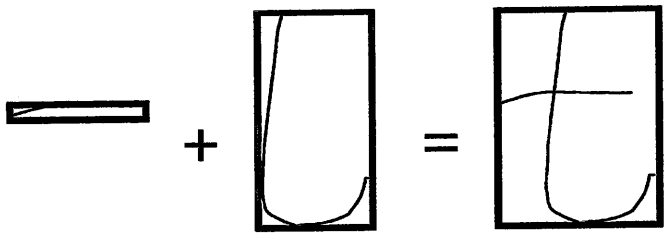


図 2.3: **外接矩形の統合**
"t"のような複数画の文字については外接矩形を統合し、新たな外接矩形を作成する.

2.3 数式要素処理部

入力された文字のタイプが連続記号と添え字のどちらでもない場合、文字は数式要素であると判定して以下の処理を行なう. ここで行なわれる処理のフローチャートを図 2.6 に示す.

まず、現在入力待ちモードであるかどうかを調べる. 入力待ちモードである場合 ($mode = WAIT$)、モードを通常モードにし ($mode \leftarrow NORMAL$)、入力された文字を入力待ちとなっている連続記号の始点または終点に設定する (図 2.7). 入力待ちモードについては 2.4 節で述べる.

次に入力された要素が属する座標を調べる. 入力されたストロークから列方向と行方向に走査をして、同一ライン上に存在する要素を調べる. 既に入力されている要素は座標情報を持っているため、ライン上に要素があればその要素と一致する行または列を座標とする. 同一ライン上に要素がない場合は、まず行方向について、既に決定された各行を構成する要素について y 方向の中心線の平均をとり、この行ごとの平均値と入力された要素の y 方向の中心線の値とを比較する. 中心線の値がある行の平均値とその次の行の平均値との間にあった場合、それらの行の間に新たな行を挿入し、入力された要素はこの行に属するものとする. 挿入された行より下の行に属する要素については、これらが持つ行情報を 1 つずつシフトする. 中心線の値が 1 行目の平均値より小さかった場合も同様の操作を行なう. 中心線の値が最も下にある行の平均値より大きかった場合は、その行の下に新たな行を生成し、入力された要素をこの行に配置する. これらの処理を列方向についても同様に行ない、入力要素の座標を決める.

ここで設定した座標が連続記号の始点と終点の間の座標 (連続記号領域と呼ぶ) であった場合、始点ないし終点の変更を行なう. 例を図 2.8 に示

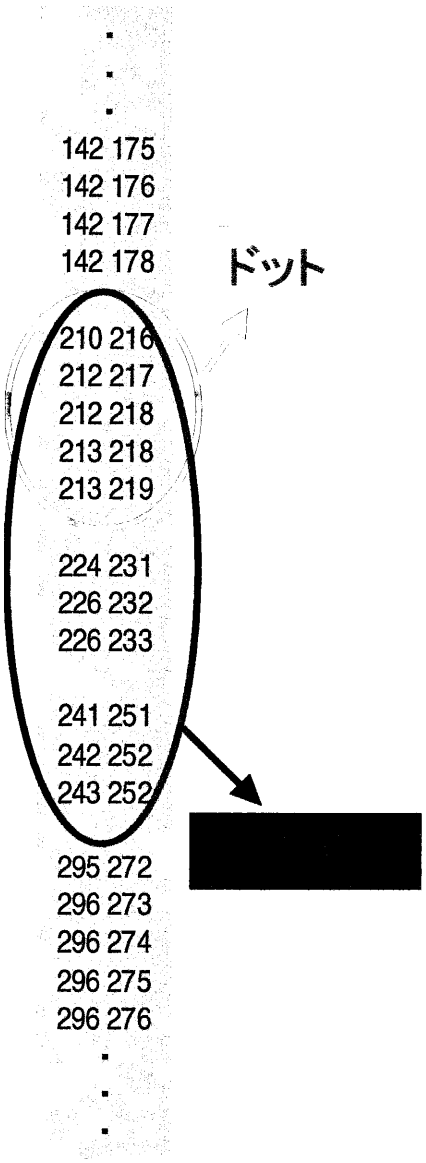


図 2.4: **連続記号の抽出**
この時系列データにはドットが 3 つ含まれている．ドットが 3 つ連続で入力されているため、これらを連続記号とする．

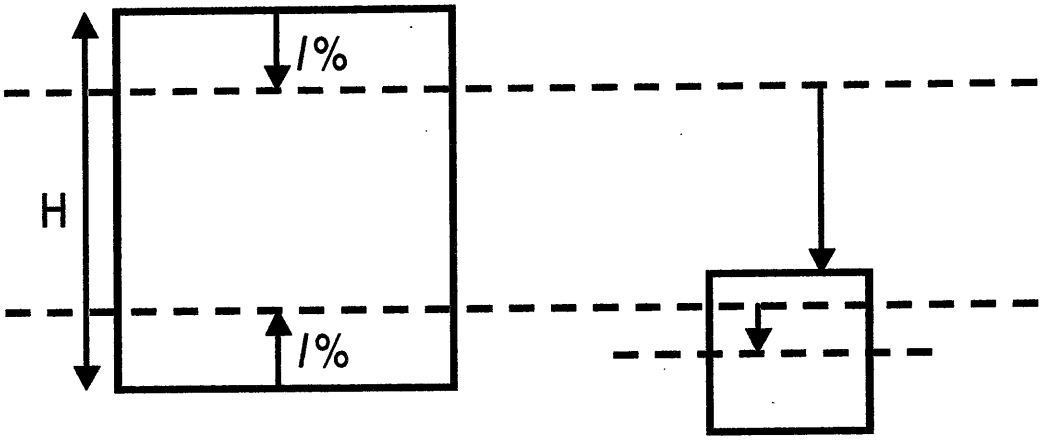


図 2.5: **添え字領域**
左側の文字が親要素、右側の文字が添え字となる.

す. (a) の状況で (b) のように連続記号が入力さると、連続記号の始点と終点は図のようにそれぞれ第1行目、第3行目 (この時点では) の座標に設定される. 連続記号領域は第2行目になる. ここで (c) に示す位置に新たな要素が入力されたとき、行の挿入がおき、(d) に示すように連続記号の終点 (または始点) を変更する処理を行なう.

2.4 連続記号処理部

ドットが3つ以上連続して入力された場合、以下のように連続記号処理を行なう. 流れ図を図 2.9 に示す.

まず、図 2.10 のように連続記号を構成する各点の x, y 方向の差分から傾きを求める. 傾きを求める計算式は次のようになる.

$$\text{傾き} = \frac{\frac{\Delta y_1}{\Delta x_1} + \frac{\Delta y_2}{\Delta x_2}}{2}$$

この値が図 2.11 に示す数直線のどの範囲に属するかによって連続記号の向き (右・下・右下・左下) を決定する.

次に、連続記号の座標を決定する. これは数式要素の座標を決定したときと同様の方法で行なう.

続いて、連続記号の傾きをもとに、その方向に続く数式要素を調べ、始点と終点にあたる数式要素を決定する. どちらかに要素が存在しない場合には、入力待ちモードとする ($mode \leftarrow WAIT$). 図 2.12 では (a) の場合

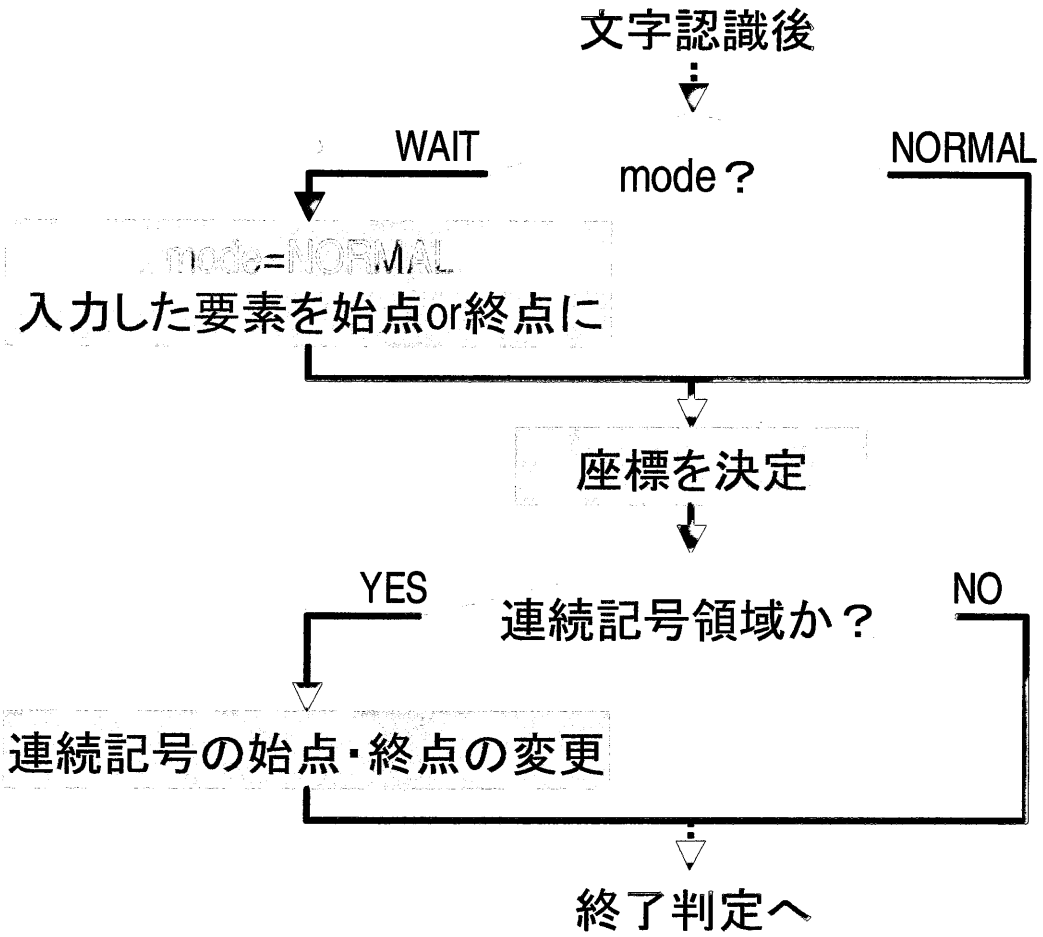


図 2.6: 数式要素処理部の流れ

がこれにあたる．終点にあたる要素がないため、入力待ちモードとなる．一方、(c) の場合には始点と終点がすぐに決定できる．

2.5　添え字処理部

添え字と判定された文字は、座標情報はもたず、その親となる文字の情報のみを持つ．添え字が 2 つ以上連続で入力された場合、2 番目以降の添え字は直前の添え字と同じ情報を持つものとする．たとえば a_{12} という数式要素があったとする．この場合、添え字"1"の親は a である．また、添え字"2"の親は、直前に入力された添え字"1"と同じものになるため、a となる．

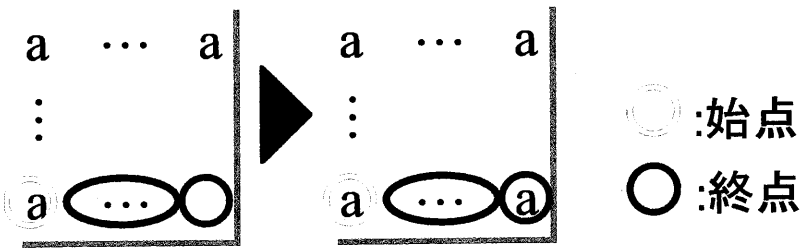


図 2.7: **入力待ちモード**
左側の図では、緑色の丸で示された連続記号の終点は入力されておらず、入力待ちモードとなる．右側の図のように数式要素が入力されると、この数式要素を入力待ちとなっている連続記号の終点とする．

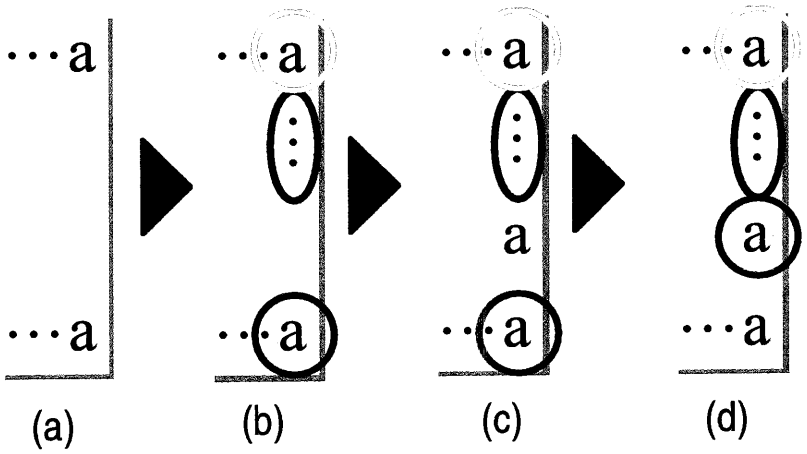


図 2.8: **連続記号の始点と終点の変更**
(c) のような誤った構造になるのを防ぐため、連続記号の終点を変更する．

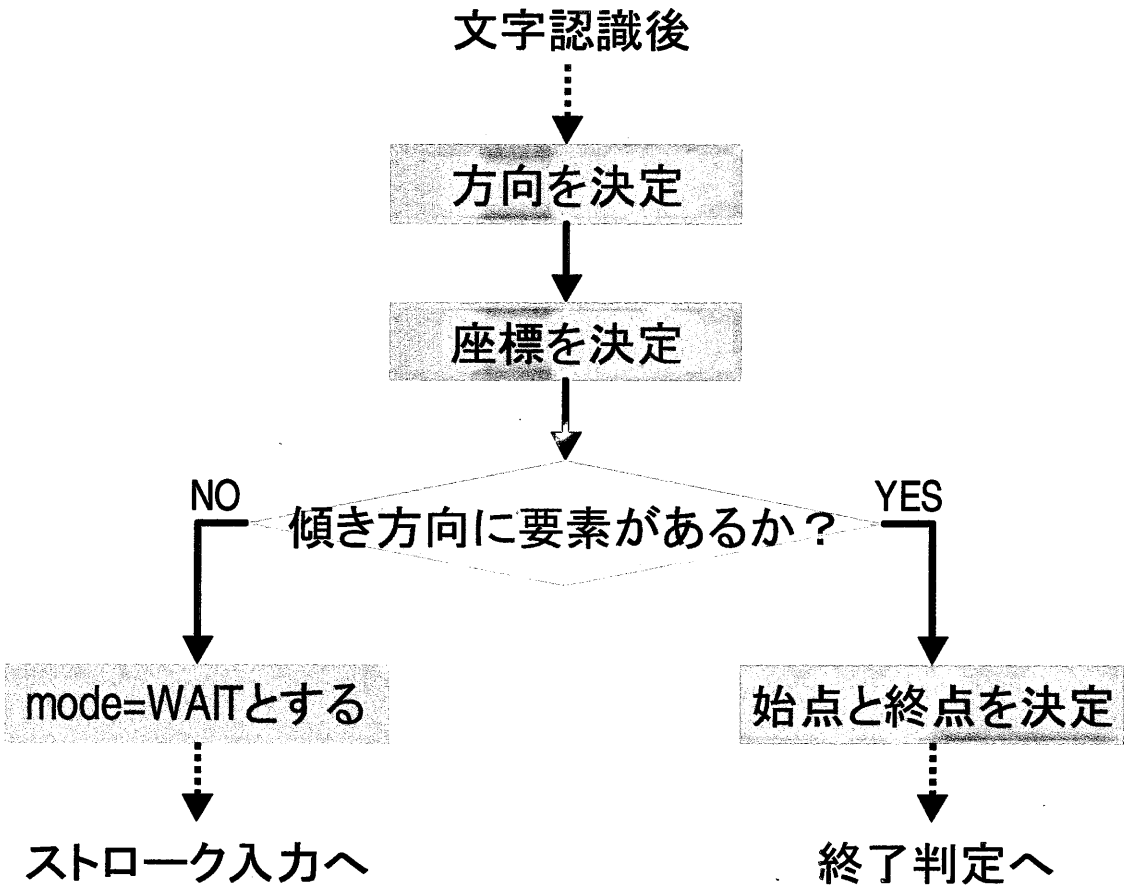


図 2.9: 連続記号処理部の流れ

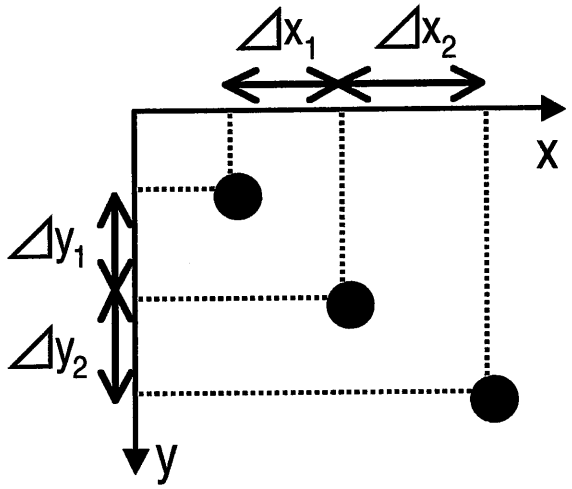


図 2.10: **連続記号の傾きの求め方**
1 番目と 2 番目のドットの座標の差分を $(\Delta x_1, \Delta y_1)$、2 番目と 3 番目のドットの座標の差分を $(\Delta x_2, \Delta y_2)$ とし、連続記号の傾きを求める.

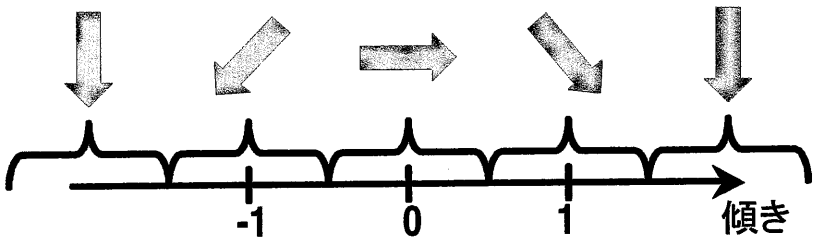


図 2.11: **連続記号の方向の求め方**
傾きの値によって連続記号の方向が決定される.

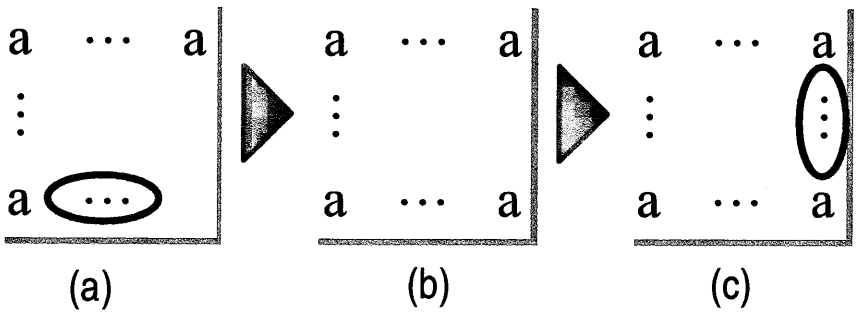


図 2.12: **連続記号の始点と終点の決定**
(a) に示された連続記号では終点が決定できないため、入力待ちモードとする．一方、(c) に示された連続記号では始点と終点を決定することができる．

2.6 領域要素処理部

全てのストロークの入力が終了した後、領域要素処理を行なう．これまでの処理では領域要素も数式要素として扱われていたため、この処理が正当だったかを改めて確認するステップになる．

数式要素と領域要素の違いは、数式要素が行・列・斜めラインのいずれかの方向に関して要素が全て埋まっている状態を形成するのに対して、領域要素はある領域を省略して１つの要素で表現しているため、これを含むラインには要素がない「空白」部分が生じる点にある．そこで、領域要素処理部では、各行と列、斜め成分について全ての要素が埋まっているかどうかを調べる．しかし、全ての斜め成分を調べると行列の四隅の成分は高々1個しか要素を持たないので、要素があるときは必ず埋まっていることになってしまう．したがって斜め成分については、調べるライン上の要素数が、行列の行と列のうちの短いほうの数の 1/2 より大きい数だけあるラインにする．例えば、短辺が 3 であった場合は 3 の 1/2 である 1.5 より大きい 2 個以上の要素数を含む斜めラインを、同様に短辺が 4 であった場合は 2 より大きい 3 個以上の要素数を含む斜めラインを調べることになる (図 2.13)．ラインを調べたとき、要素が全て埋まっていれば、それらにマークをつける．最終的にマークされずに残っていた要素が領域要素となる (図 2.14)．領域要素に隣接する空白部分は、領域要素と同じ値を持つものとして処理を行なう．この様子を図 2.15 に示す．

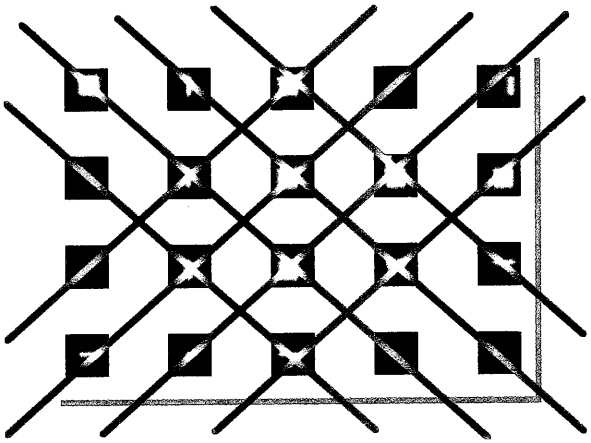


図 2.13: 4 × 5 行列の斜めの有効走査ライン
この例では 3 個以上の要素を含む計 8 ラインを走査する.

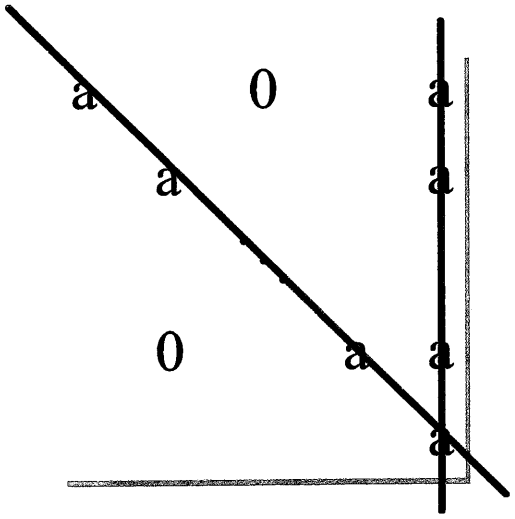


図 2.14: 領域要素の判定
2 つの”0” が領域要素となる.

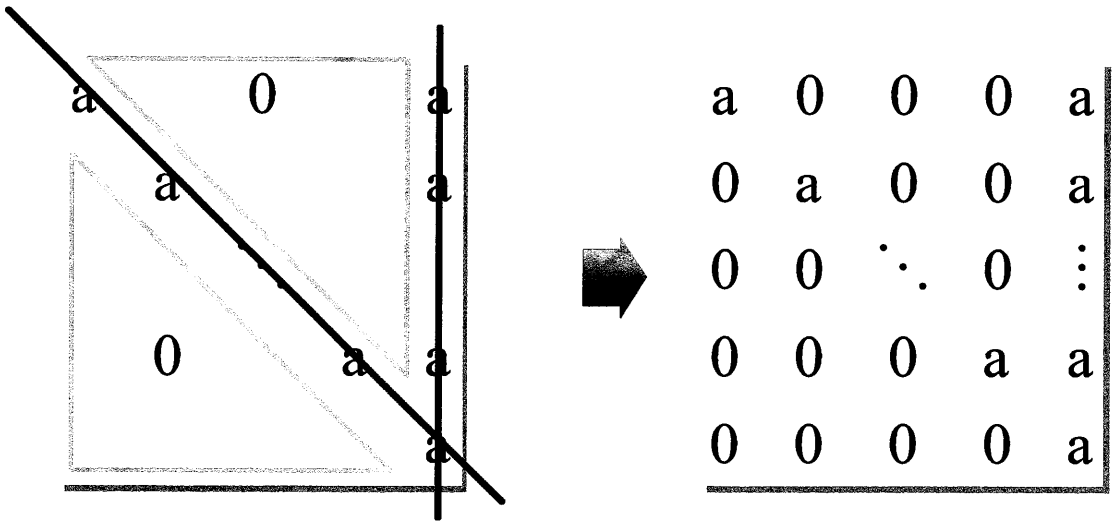
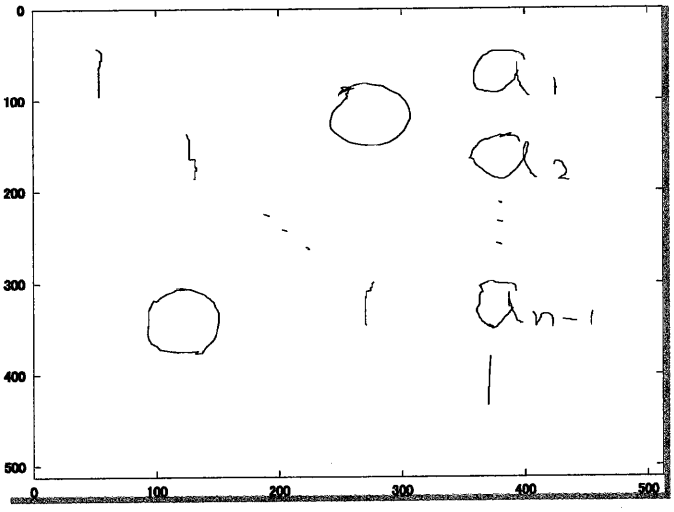


図 2.15: **空白部分の処理**
三角で囲まれた領域の要素は全て 0 である.

2.7　出力

出力は、各数式要素の座標、領域要素が表す領域、連続記号の座標、添え字情報を記述するフォーマットで行なう. これは金堀ら [9] が提案した行列表現フォーマットに準ずるものである. 実際の出力例を図 2.16 に示す.



*****PROFILE*****

5*5 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (2,2)
6 : (4,4)
7 : (5,5)
8 : (1,5)
#10 : (2,5)
#15 : (4,5)
#19 : Region (1,2) (1,3) (2,3) (1,4) (2,4) (3,4)
#20 : Region (2,1) (3,1) (4,1) (5,1) (3,2) (4,2) (5,2) (4,3) (5,3) (5,4)

SUBSCRIPT LIST
1 : subscript of 8.
2 : subscript of 10.
3 : subscript of 15.
4 : subscript of 15.
5 : subscript of 15.

CONNECTION LIST
1 : Connects (2,2) -> (4,4)
2 : Connects (2,5) -> (4,5)

図 2.16: **実際の出力例**
本システムでは具体的な文字認識を行なっていないため、入力された順番をラベルとして出力を行なう.

第3章

構造認識実験

3.1 実験条件

第2章で提案した行列構造認識システムを用いて構造認識実験を行なう．実験に用いる画像はマウスを用いて入力した512×512サイズの手書き画像である．被験者には図3.1に示す5種類の行列を入力してもらう．ただし、括弧は入力しないものとする．パラメータは$l = 25$とした．評価は、サンプル行列の構造と出力されたフォーマットから読み取れる構造とが一致するかどうかを調べることにより行なう．構造解析結果がサンプル行列と完全一致したと見なされる場合のみ認識成功とし、それ以外は認識失敗とする．

本手法では、入力の際に、いくつかの制約条件を課した．それらの詳細は次節に示すが、通常の入力においてはユーザが違和感を感じない程度のものである．

3.2 実験結果

3.2.1 行・列間挿入を行なわない場合

初めに、簡単のために行・列間挿入を行なわない場合における構造認識実験を行なった．この場合の制約条件は次のようになる．

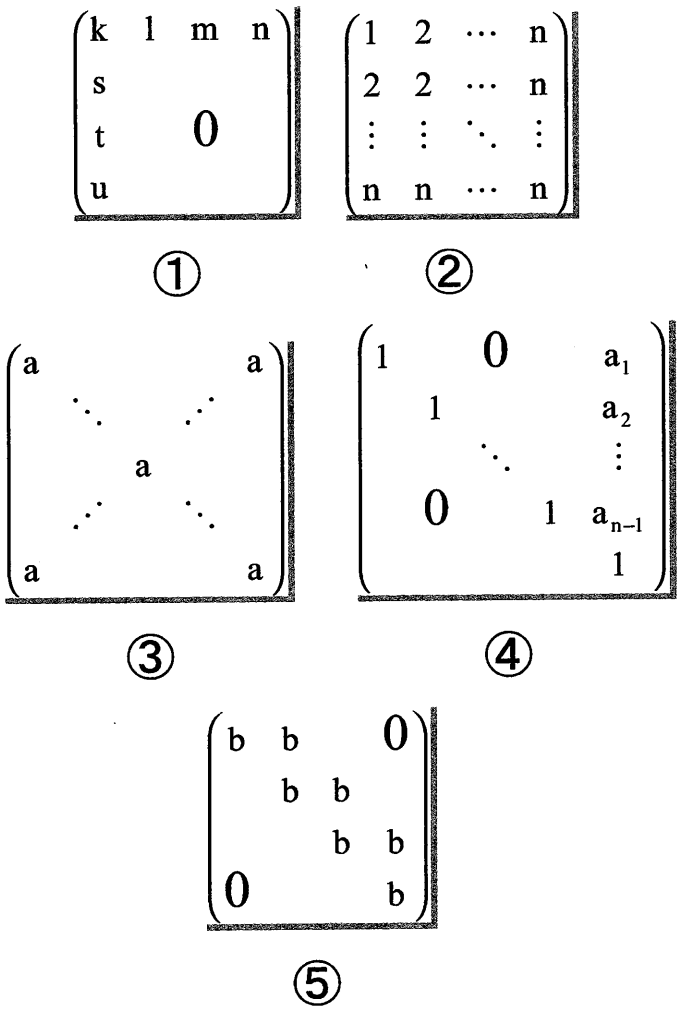


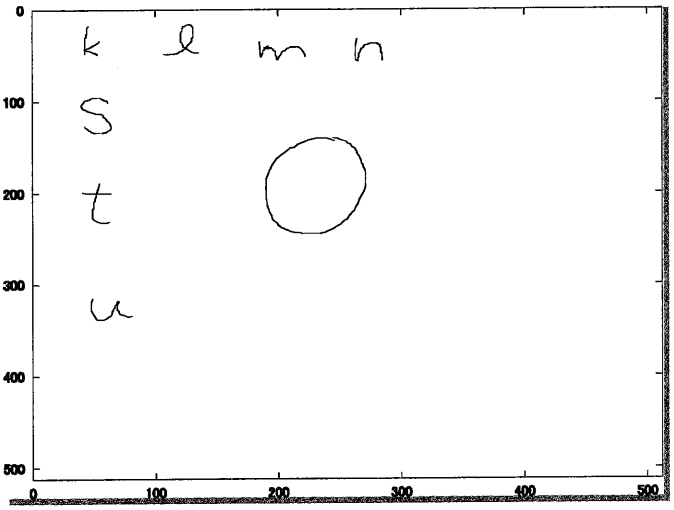
図 3.1: **認識対象とした行列**
(1) は領域要素を含む例、(2) は連続記号を含む例、(3) は連続記号と空白領域を含む例、(4) は連続記号、領域要素、添え字を含む例、(5) は斜めの走査ラインを 2 つ含む例である.

- 行と列は空白によって格子状に区切られている
- (1,1) 要素から隣接する順に入力する
- 連続記号を書いた後にその始点または終点に対応する要素を入力する場合の入力のタイミングは、連続記号入力の直後とする
- 添え字の入力は親要素を入力した直後に行なう
- 領域要素は最後に入力する

実験の結果、認識率は表 3.1 のようになった．実際の認識成功例と認識失敗例をそれぞれ図 3.2 ～図 3.6 と図 3.7 ～図 3.9 に示す．

サンプル No.	1	2	3	4	5	合計
認識率	93%	93%	73%	13%	90%	71%

表 3.1: **構造解析結果**



*****PROFILE*****

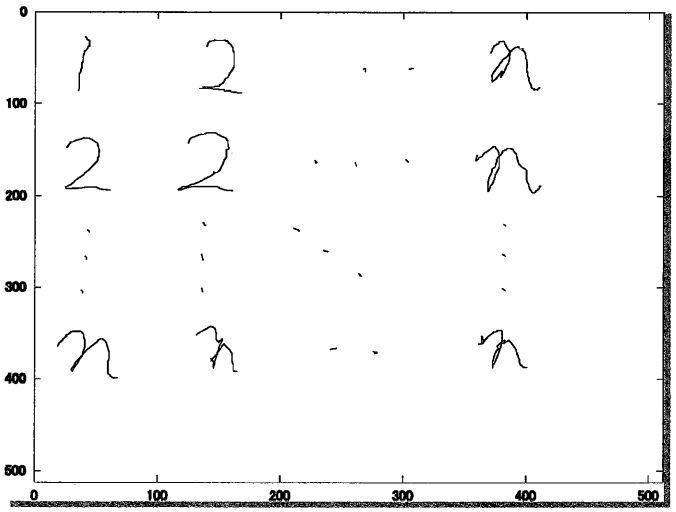
4*4 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (1,2)
3 : (1,3)
4 : (1,4)
5 : (2,1)
6 : (3,1)
7 : (4,1)
8 : Region (2,2)(3,2)(4,2)
(2,3)(3,3)(4,3)(2,4)(3,4)
(4,4)

SUBSCRIPT LIST

CONNECTION LIST

図 3.2: **認識成功例** 1
領域要素を表す領域が正確に抽出されている.



*****PROFILE*****

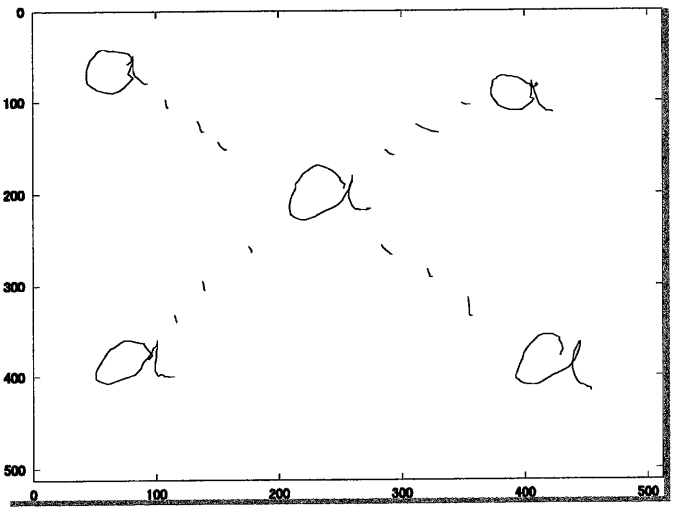
4*4 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (1,2)
6 : (1,4)
7 : (2,1)
8 : (2,2)
#12 : (2,4)
#16 : (4,1)
#20 : (4,2)
#24 : (4,4)

SUBSCRIPT LIST

CONNECTION LIST
1 : Connects (1,2) -> (1,4)
2 : Connects (2,2) -> (2,4)
3 : Connects (2,1) -> (4,1)
4 : Connects (2,2) -> (4,2)
5 : Connects (2,4) -> (4,4)
6 : Connects (2,2) -> (4,4)
7 : Connects (4,2) -> (4,4)

図 3.3: **認識成功例** 2
連続記号が複雑に入り組んでいる場合でも、正しく認識できた.



*****PROFILE*****

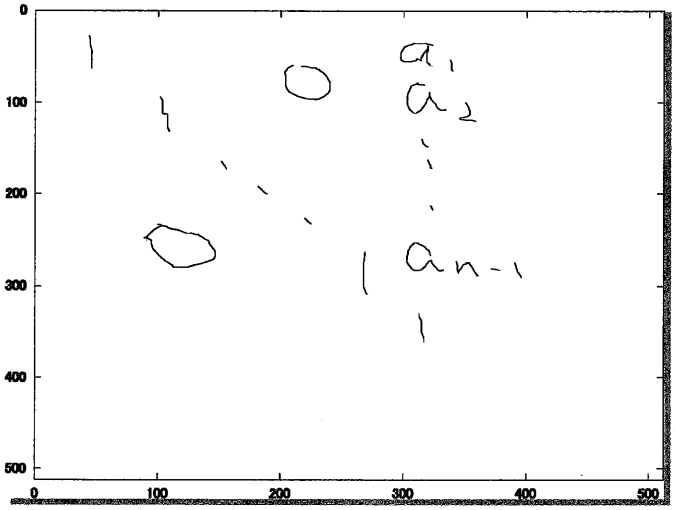
5*5 MATRIX.

ELEMENT LIST
1 : (1,1)
5 : (3,3)
9 : (1,5)
#13 : (5,1)
#17 : (5,5)

SUBSCRIPT LIST

CONNECTION LIST
1 : Connects (1,1) -> (3,3)
2 : Connects (1,5) -> (3,3)
3 : Connects (3,3) -> (5,1)
4 : Connects (3,3) -> (5,5)

図 3.4: **認識成功例 3**
空白要素の影響を受けていないことが確認できる.



*****PROFILE*****

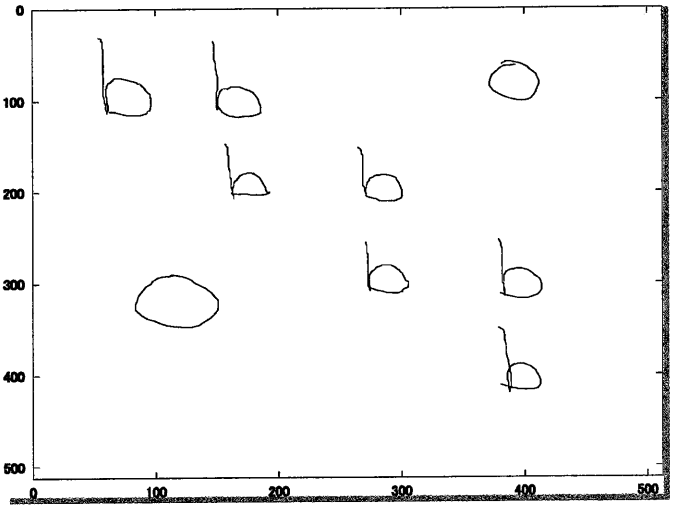
5*5 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (2,2)
6 : (4,4)
7 : (5,5)
8 : (1,5)
#10 : (2,5)
#15 : (4,5)
#19 : Region (2,1)(3,1)(4,1)
(5,1)(3,2)(4,2)(5,2)(4,3)(5,3)
(5,4)

SUBSCRIPT LIST
1 : subscript of 8.
2 : subscript of 10.
3 : subscript of 15.

CONNECTION LIST
1 : Connects(2,2)->(4,4)
2 : Connects(2,5)->(4,5)

図 3.5: **認識成功例** 4
数式要素"a_{n-1}"の添え字のうち、"−"と"1"は認識できなかったが、全体的な構造は抽出できていると考えられるため、認識成功とした.



*****PROFILE*****

4*4 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (2,2)
3 : (3,3)
4 : (4,4)
5 : (3,4)
6 : (2,3)
7 : (1,2)
8 : Region (1,3)(1,4)(2,4)
9 : Region (2,1)(3,1)(4,1)
(3,2)(4,2)(4,3)

SUBSCRIPT LIST

CONNECTION LIST

図 3.6: **認識成功例 5**
領域要素の判定において、要素を 4 つ含むラインと 3 つ含むラインが走査
された.

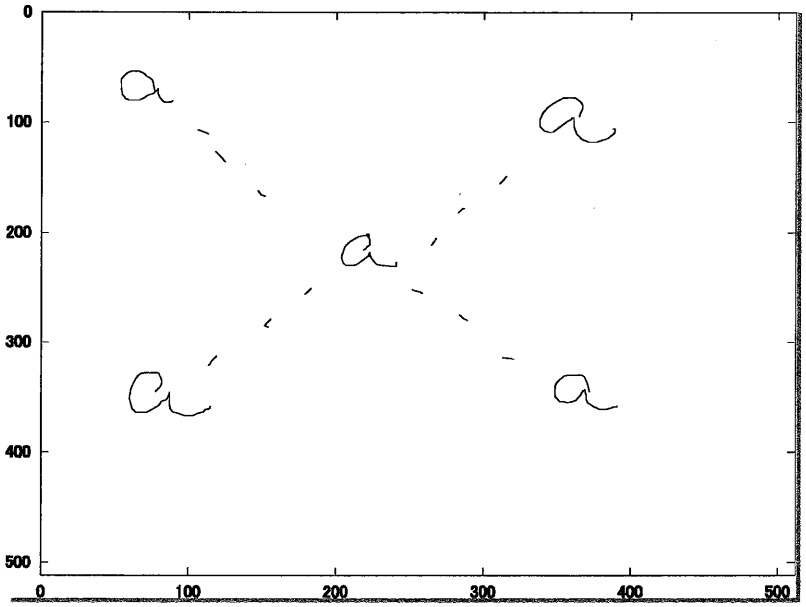


図 3.7: **認識失敗例** 1
右上の連続記号の終点 (左下にあたる要素) が、走査領域になかったため検出できず、エラーとなった.

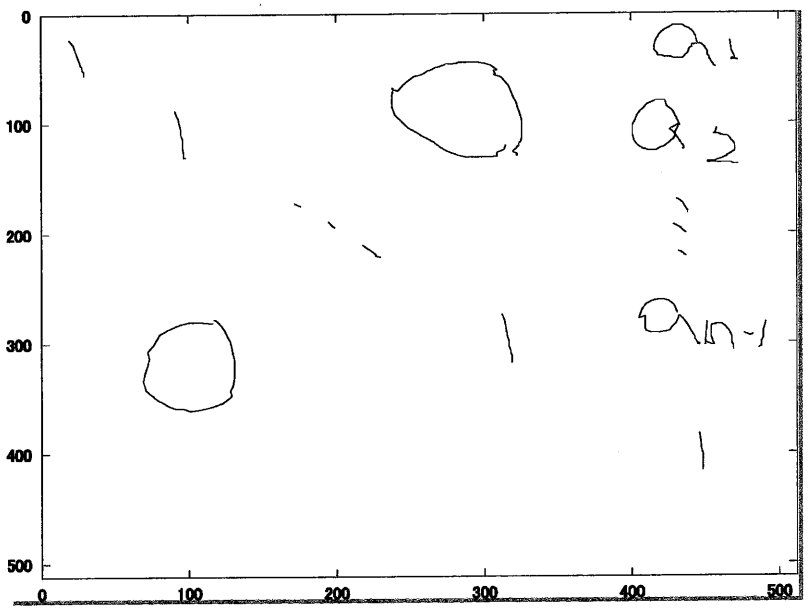


図 3.8: **認識失敗例 2**
添え字のサイズが大きかったため、数式要素として認識され、正しい構造を得ることができなかった.

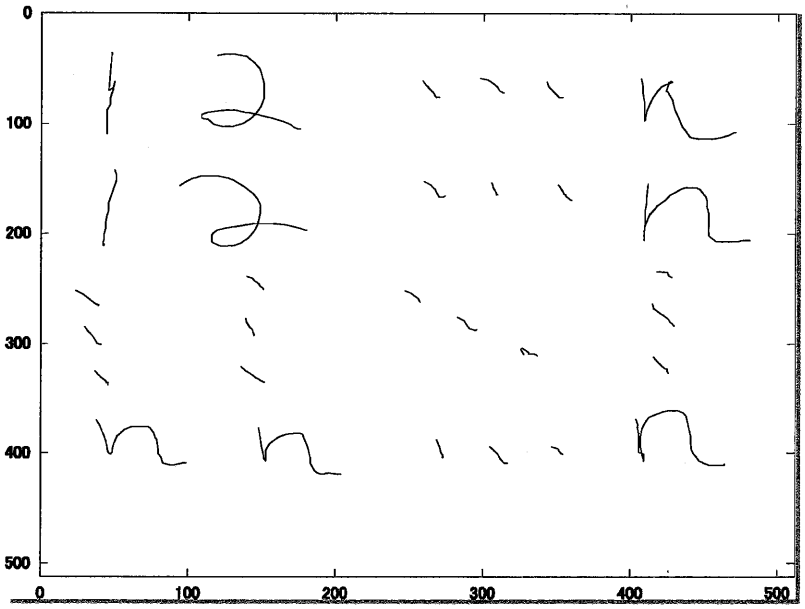


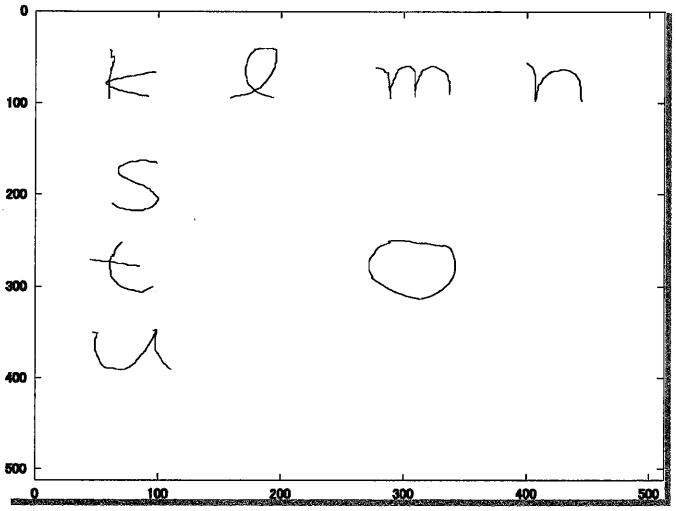
図 3.9: **認識失敗例 3**
(4,1) 要素と (2,2) 要素に列方向の重なり部分があるため、列の構造が正しく検出できなかった.

3.2.2　行・列間挿入を可能にした場合

次に、このシステムを拡張し、行・列間挿入を可能にしたものを用いて実験を行なった．この場合の制約条件は次のようになる．

- 行と列は空白によって格子状に区切られている
- 連続記号を書いた後にその始点または終点に対応する要素を入力する場合の入力のタイミングは、連続記号入力の直後とする
- 添え字の入力は親要素を入力した直後に行なう
- 領域要素は最後に入力する

実験の結果、前節で用いたサンプル画像については、行・列間挿入を認めなかった場合と同様の認識率であった．新たな制約条件に基づいて作成したサンプル画像を用いたときの認識成功例については、図 3.10 と図 3.11 に示す．



*****PROFILE*****

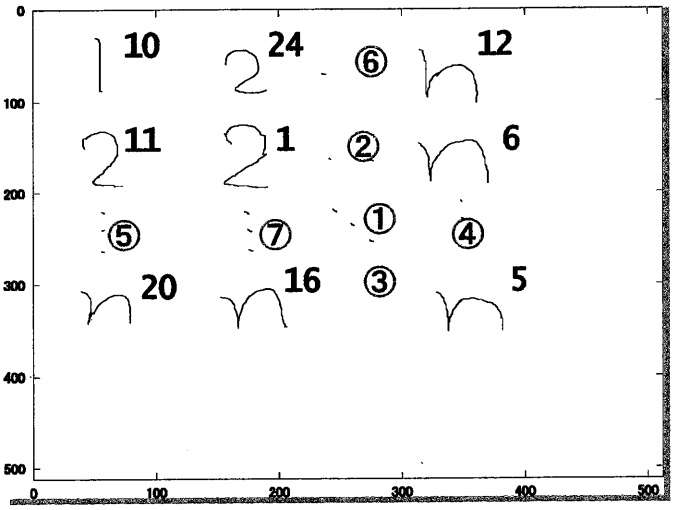
4*4 MATRIX.

ELEMENT LIST
1 : (1,3)
2 : (4,1)
3 : (1,4)
4 : (2,1)
5 : (1,1)
6 : (3,1)
7 : (1,2)
8 : Region (2,2)(3,2)(4,2)
(2,3)(3,3)(4,3)(2,4)(3,4)
(4,4)

SUBSCRIPT LIST

CONNECTION LIST

図 3.10: **行・列間挿入を含む画像の認識成功例** 1
要素の入力順によらず、構造が正しく認識できている.




```
*****PROFILE*****

 4*4 MATRIX.

**ELEMENT LIST**
# 1 : (2,2)
# 5 : (4,4)
# 6 : (2,4)
#10 : (1,1)
#11 : (2,1)
#12 : (1,4)
#16 : (4,2)
#20 : (4,1)
#24 : (1,2)

**SUBSCRIPT LIST**

**CONNECTION LIST**
# 1 : Connects (2,2) -> (4,4)
# 2 : Connects (2,2) -> (2,4)
# 3 : Connects (4,2) -> (4,4)
# 4 : Connects (2,4) -> (4,4)
# 5 : Connects (2,1) -> (4,1)
# 6 : Connects (1,2) -> (1,4)
# 7 : Connects (2,2) -> (4,2)
```

図 3.11: **行・列間挿入を含む画像の認識成功例** 2
制約条件に沿った入力を行なえば、複雑な入力順に対しても対応できていることが確認できる．青字で示した番号は要素の入力順であり、出力で用いたラベルと対応している．2～4 番目の要素はドットであり、これらは連続記号であるとして認識されたため、1 つ目の連続記号として赤字の 1 で示した．これ以下も同様で、全部で 30 文字分が入力されている．

第4章

考察

構造認識実験の結果、領域要素の抽出や連続記号の解析などは概ね上手く働いていたといえるだろう．ところが、全体の認識率は71%とあまり良い結果は得られなかった．中でも、連続記号や添え字を含む行列構造に関しては認識率が悪かった．この理由として、マウスを使った入力のため、小さな文字が書きづらかったことや、添え字や連続記号の判定方法が厳しすぎたことが挙げられる．添え字の判定方法については、青島らが提案した添え字抽出手法を用いたが、彼らの実験では添え字認識率が約90%になっていることを考えると、入力時の問題が大きく影響していることが考えられる．また、画素数と閾値の関係により、数式要素がドットと判定されたケースや、逆にドットが数式要素と判定されてしまったケースも多かった．ドットの判定の際には画素数情報のほかに文字のサイズも考慮するなど、より厳密な判定法が必要となる．連続記号(ドット列)においても、傾き方向にあるストロークを調べるだけでなく、座標情報なども併用することにより、より正確な解析が可能になると思われる．また、今回は文字認識を行なっていないため、それぞれの文字の特徴(例：同じ文字サイズのaとbでは後者の高さのほうが高くなる)を反映できなかったことも挙げられる．添え字の影響を調べるため、添え字を含むサンプル4から添え字を消去して再実験を行なったところ、認識率は53%(全体で83%)となった．

認識率を向上させるための手法の一つに、複数行(または列)にまたがっ

てしまった数式要素の処理が考えられる．制約条件の中に、行と列が空白によって区切られなければならないという項があるが、手書きの入力ではこの条件が守られない場面がたまに見受けられる．入力する行列が大きなものであったり、筆記者に文字列を右上がり (または右下がり) に書く癖がある場合などには、この現象は顕著になるであろう．この問題の解決案として、ある要素が2つ以上の行 (または列) にまたがった場合、それらの行 (または列) を構成する要素について外接矩形の y(または x) 方向の中心線の平均をとり、その平均値と入力された要素の外接矩形の y(または x) 方向の中心線とを比較する．比較の結果、最も値の近い行 (または列) に入力された要素を配置すればよい．入力した要素が目的のものと隣接する行 (または列) にわずかに接触してしまった程度のエラーについては、この手法によって十分対応できると考えられる．

本論文で提案した手法は、略記を含む行列構造の認識を行なうもので、これだけでは数式認識システムとしては不完全である．しかし、領域要素の判定法などは行列特有の性質を利用したものであり、まだ明確な認識手法が確立されていないオンライン数式認識分野において、このような手法が行列認識にも対応した数式認識システムを構築する際の一助となることが期待される．

最後に、今回は入力順等に関していくつかの制約条件を課したが、ユーザの利便性を考えれば、このような条件は無いに越したことはない．本来紙の上にペンで行列を書く際には何の制約も受けずに書けるためであり、そのような状況で入力ができる数式認識システムこそが理想であるからである．そこで、今後は入力順に関する制約条件を必要としないアルゴリズムを導入するなど、より自由度の高い入力にも対応可能な手法を開発することが課題となる．

第5章

結論

本論文では、これまでの関連研究では対応していなかった連続記号や領域要素などの略記にも対応可能な行列構造認識システムを提案した．その上で、手書き行列画像を用いた実験を行ない、提案手法の有効性を確認した．行列構造の認識率は 71%であった．また、提案するシステムを拡張し、既に入力された行 (または列) 間への要素の挿入が可能であることを確認した．

今後は、筆記者による行列の書き方の違いに対して頑健であり、かつより自由な環境で入力できるシステムに改善していくことが課題となる．

謝辞

本研究を行なう機会を与えて下さり、研究の楽しさから厳しさまで情熱を持って御指導御鞭撻を頂きました、東北大学工学研究科 阿曽弘具教授に心より感謝致します.

本研究をまとめるにあたり、極めて有益な御質疑、御意見を賜りました、東北大学工学研究科 牧野正三教授、ならびに東北大学情報科学研究科 根元義章教授に心より感謝致します.

本研究を進めるに当たり、指導教官として常日頃より親身になって懇切丁寧な御指導を頂きました、東北大学工学研究科 大町真一郎助教授に心より感謝致します.

研究室における日常生活全般においてお世話になりました、東北大学工学研究科 菅谷至寛助手に心より感謝致します.

私の実験の進むべき方向性を示して下さり、また自らの身を以って研究者としての姿を示して頂きました、東北大学工学研究科博士課程後期3年加藤毅氏に心より感謝致します.

日頃のゼミにおいて、本研究に対する有益なアドバイスをしてくださった東北大学工学研究科博士課程後期3年 岩村雅一氏、東北大学工学研究科博士課程後期2年 下村正夫氏に心より感謝致します.

同じく修士論文をまとめる身として何かとお世話になりました、阿曽研究室M2の諸氏、また認識ゼミにおいてお世話になりました認識班の諸氏、実験を行なう上でのサンプル集めに協力して下さった諸氏に心より感謝致します.

最後に、就職で新天地に移った者を含め、大学入学から今まで共に学び、遊び、心の支えとなってくれた友人達、そして現在に至るまでの長きに渡る学生生活を温かく見守り、励まして下さった両親に、心より感謝致します.

「ありがとう.」

参考文献

[1] K.Chan and D.Yeung: "Recognizing on-line handwritten alphanumeric characters through flexible structural matching," Pattern Recogition**32**, pp. 1099–1114 (1999).

[2] K.Chan and D.Yeung: "Error detection,error correction and performance evaluation in on-line mathematical expression recognition," Pattern Recogition**34**, pp. 1671–1684 (2001).

[3] R.Zanibbi, D.Blostein and J.R.Cordy: "Recognizing mathematical expressions using tree transformation," IEEE Trans.Pattern Analysis and Machine Intelligence, **24**, 11, pp. 1455–1467 (2002).

[4] 青島, 鈴木, 森, 末永 : "実時間手書きストローク解析による数式入力システム," 信学論 D-II, **J-83-D-II**, 5, pp. 1232–1245 (2000).

[5] 岡本, 東 : "記号のレイアウトに注目した数式構造認識," 信学論 D- II, **J78-D-II**, 3, pp. 474–482 (1995).

[6] 能隅, 福田, 玉利, 鈴木: "絞り込み法による数式文字認識とその日本語/数式領域切出しへの応用," 信学論, **J83-D-II**, 3, pp. 895–906 (2000).

[7] 江藤, 福田, 鈴木 : "最小コスト全域木探索を用いたオフライン数式構文認識," 信学技報, **PRMU99-236**, pp. 37–43 (2000).

[8] 豊住, 鈴木, 森, 末永 : "オンライン手書き数式認識システムにおける行列認識機構の検討," 信学技報, **PRMU2001-239**, pp. 47–52 (2000).

[9] 金堀, 鈴木 : "可変ブロックパターンによる矩形領域分割を用いた行列の認識," 信学技報, **PRMU2000-201**, pp. 1–6 (2001).

研究業績

学会発表等

日下部淳, 大町真一郎, 阿曽弘具, 加藤毅

"オンライン手書き行列の構造認識," 電子情報通信学会総合大会,PRMU,2003年3月 (発表予定).

修士論文本審査

オンライン数式認識手法に関する研究

東北大学大学院工学研究科　電気・通信工学専攻
博士課程前期2年
日下部　淳

第1章
序論

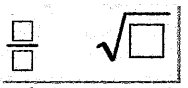
1.1 背景

- コンピュータを用いて学術論文等を書く際、数式の入力には多くの時間と手間がかかることがある

¥[¥frac{y}{x}¥]
¥[¥sqrt{x+y}¥]

TEXによる入力例

操作性・・・○
視覚的・・・×



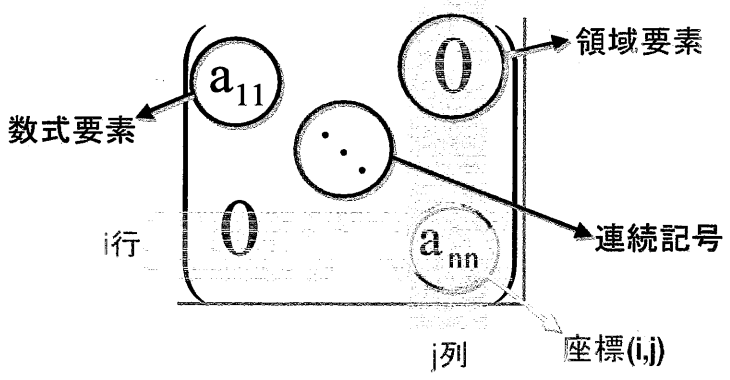
エディタによる入力例

操作性・・・×
視覚的・・・○

- 手書きの数式をオンラインで認識するための研究が行なわれている
- しかし、行列構造については、認識対象とされないことが多い

用語の解説

本論文では、行列各部の名称を次のように定義する



1.2 関連する研究

①『オンライン手書き数式認識システムにおける行列認識機構の検討』
豊住・鈴木・森・末永
<信学技報, PRMU2001-239(2002-2)>

②『可変ブロックパターンによる矩形領域分割を用いた行列の認識』
金堀・鈴木
<信学技報, PRMU2000-201(2001-3)>

	入力方式	認識対象	略記
①	オンライン	手書き文字	非対応
②	オフライン	活字	対応

1.3 目的

- 数式認識において、個々の要素の認識率は良いが、数式全体の完全認識率はあまり良くない
- 逐次認識を行なうことにより、エラー箇所をその都度修正し、結果的に認識率を上げることができる

⇨ オンライン手法を採用

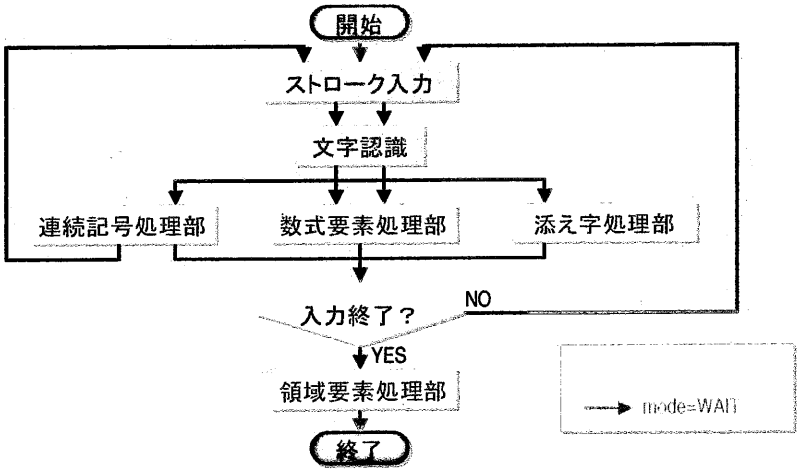
略記にも対応した、オンライン手書き行列構造認識システムの提案

本論文の構成

第1章　序論
第2章　提案する行列構造認識手法
第3章　構造認識実験
第4章　考察
第5章　結論

第2章
提案する認識手法

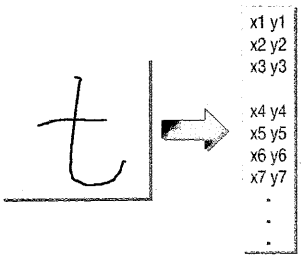
2.1 システムの流れ



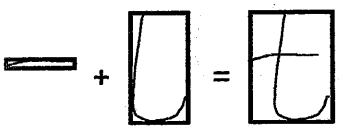
修士論文本審査資料 9

2.2 ストローク入力

- マウスの軌跡の座標を時系列データで表す
- 1ストローク毎に外接矩形を作成
- 直前のストロークとの重なりがある場合には、複数画の1つの文字と判定し、これらを統合して新たな外接矩形を作成



時系列データ



外接矩形の統合

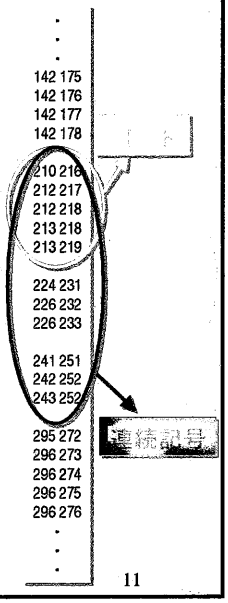
修士論文本審査資料 10

2.3 文字認識－連続記号の抽出－

- 入力されたストロークのタイプを、連続記号、数式要素、添え字の3つに分類

－連続記号の抽出－

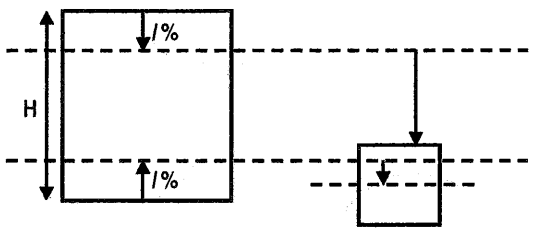
- ストロークの長さ(画素数)が極端に少ないものはドットとみなす
- ドットが3つ以上連続で入力された場合、これらを連続記号とする



2.3 文字認識－添え字の抽出－

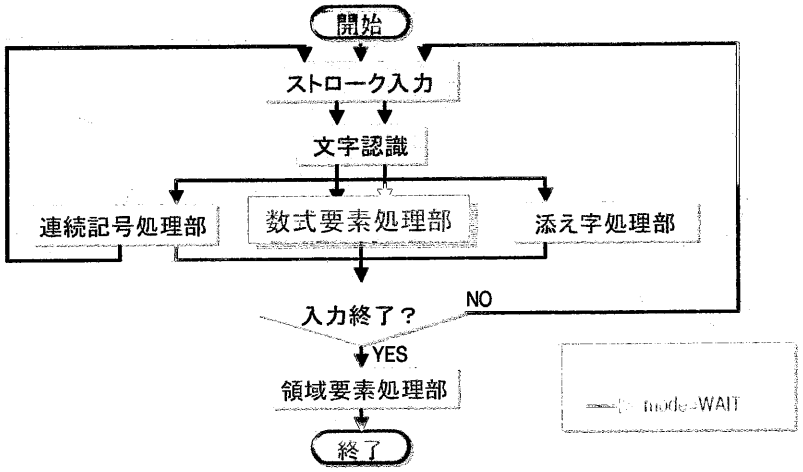
－添え字の抽出－

- 右側の文字の中心の高さが左側の文字の下端から、左の文字の高さHのl%のところより低く、かつ右側の文字の上端が左側の文字の上端から、左の文字の高さHのh%のところより低い場合、右側の文字は左側の文字に対し下付きの関係にあるとする



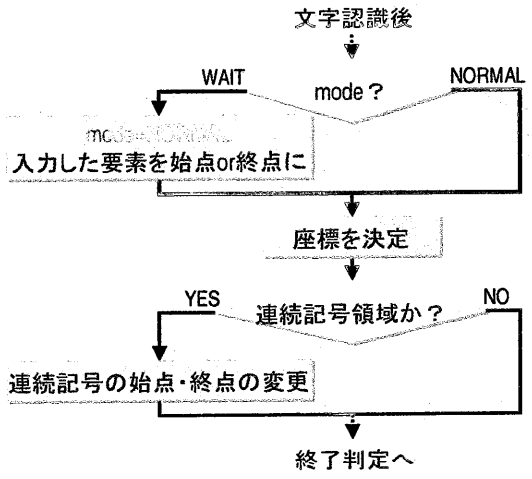
添え字判定領域

2.4 数式要素処理部



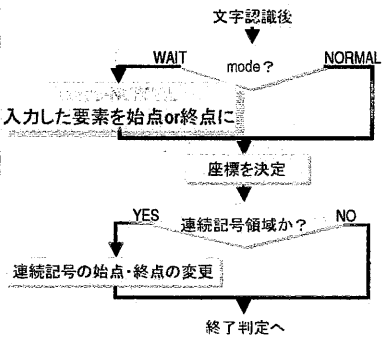
修士論文本審査資料 13

2.4 数式要素処理部－流れ図－

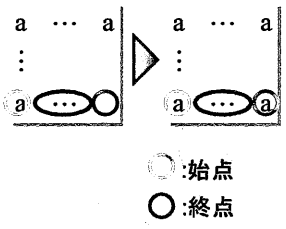


修士論文本審査資料 14

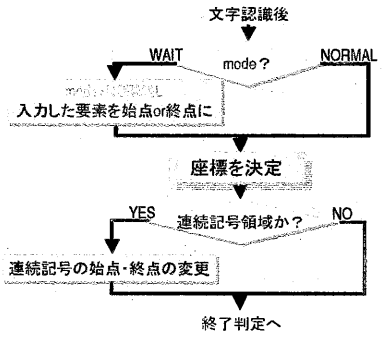
2.4 数式要素処理部－入力待ちモード－



- 入力待ちモードである場合、この要素を入力待ちとなっている連続記号の始点または終点にする



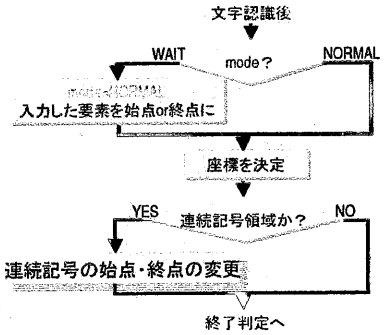
2.4 数式要素処理部－座標の決定－



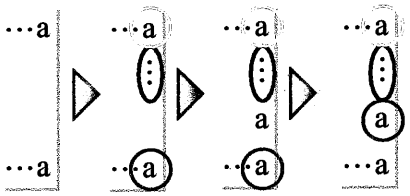
- 入力されたストロークから列方向と行方向に走査をして、同一ライン上に存在する要素を調べる
  - 要素があれば、その要素と一致する行または列を座標とする
  - なければ、新たな行または列を生成する

a	b
c	d

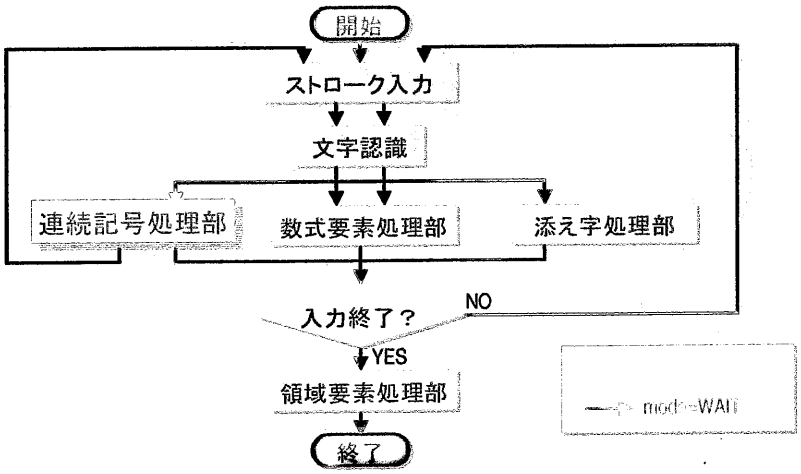
2.4 数式要素処理部－連続記号の変更－



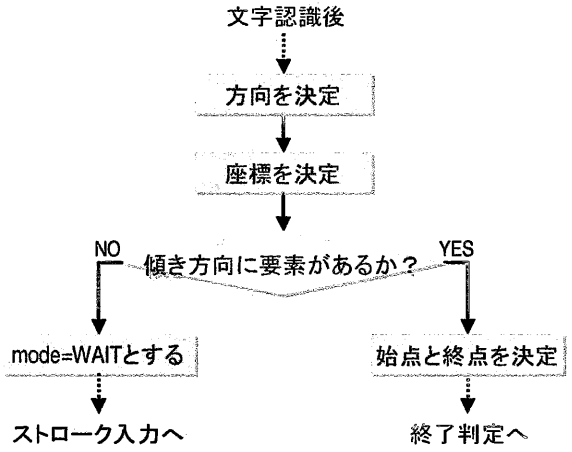
- 決定された数式要素の座標が連続記号領域であった場合、その連続記号の始点または終点にあたる数式要素を変更する



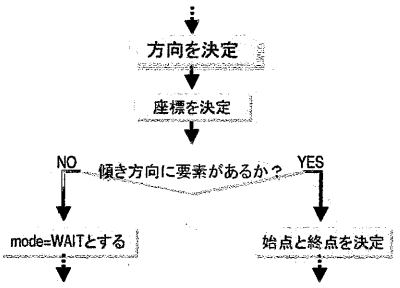
2.5 連続記号処理部



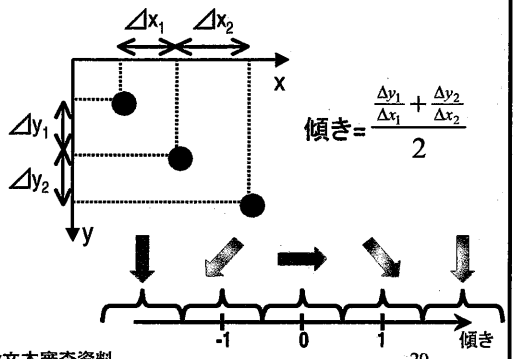
2.5 連続記号処理部－流れ図－



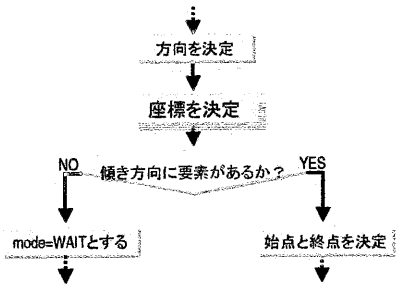
2.5 連続記号処理部－方向の決定－



- 連続記号を構成する各点のx,y方向の差分から傾きを求める
- この値により連続記号の向きを決定する（右・下・右下・左下）

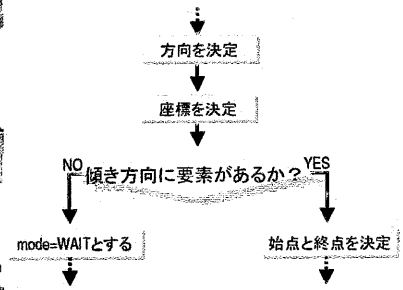


2.5 連続記号処理部－座標の決定－

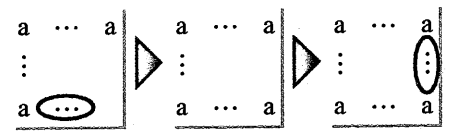


- 入力されたストロークから列方向と行方向に走査をして、同一ライン上に存在する要素を調べる
  - 要素があれば、その要素と一致する行または列を座標とする
  - なければ、新たな行または列を生成する

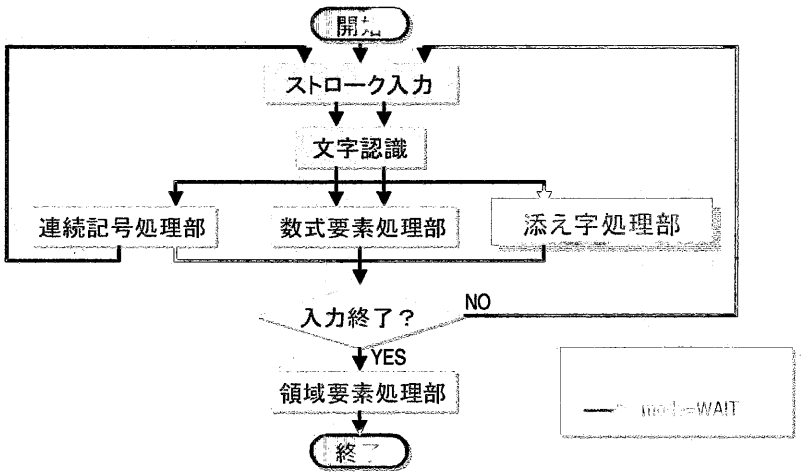
2.5 連続記号処理部－始点と終点の決定－



- 傾き方向に続く数式要素を調べ、始点と終点に当たる数式要素を決定する
- どちらかに要素が存在しない場合、入力待ちモード(mode=WAIT)とする



2.6 添え字処理部



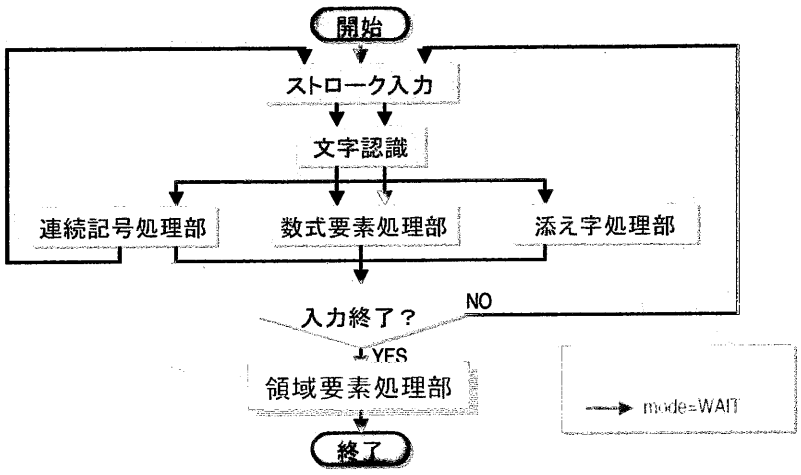
2.6 添え字処理部

- 添え字と判定された文字は、その親になる文字の情報のみを持つ
- 添え字が2つ以上連続で入力された場合、2番目以降の添え字は直前の添え字と同じ情報を持つ

$$a_{12}$$

```
PARENT[1] = a
PARENT[2] = PARENT[1]
          = a
```

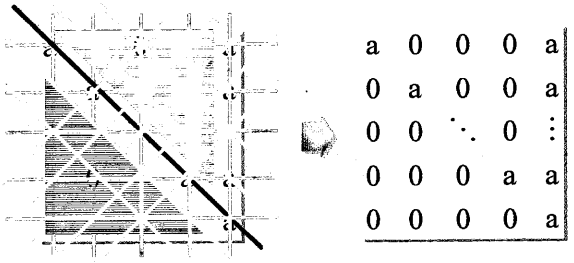
2.7 領域要素処理部



修士論文本審査資料 25

2.7 領域要素処理部

- 各行と列、対角成分について全ての要素が埋まっているかどうかを調べる
- 埋まっていれば、それに属する要素は数式要素または連続記号であると考え、マークを付ける
- 全てのラインを調べ終わったときにマークされずに残っていた要素を領域要素とみなす
- 領域要素に隣接する空白要素も、領域要素と同じ値を持つものとして処理する

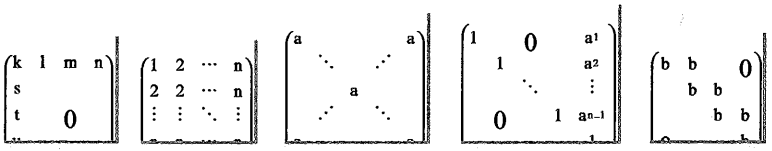


修士論文本審査資料 26

第3章
構造認識実験

3.1 実験条件－実験の概略－

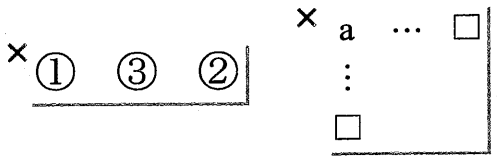
- マウスを用いて入力した512×512サイズの手書き画像を用いる
- 被験者には下記の5種類の行列を入力してもらう
- 評価はサンプル行列の構造と出力された構造を目視で比較
- 構造解析結果がサンプル行列と完全一致したとみなされる場合のみ、認識成功とし、それ以外は認識失敗とする



① ② ③ ④ ⑤

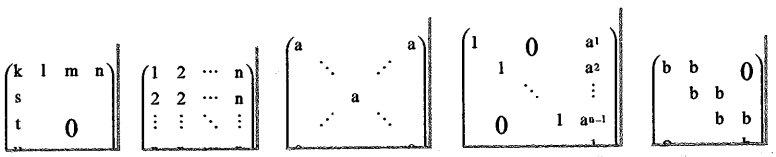
3.1 実験条件－制約条件－

- 入力の際には、次のような制約条件を課した．これらは、通常の入力においてはユーザが違和感を感じない程度のものである
  - 行と列は空白によって格子状に区切られている
  - (1,1)要素から隣接する順に入力する
  - 連続記号を書いた後にその始点または終点に対応する要素を入力する場合の入力のタイミングは、連続記号入力の直後とする
  - 領域要素は最後に入力

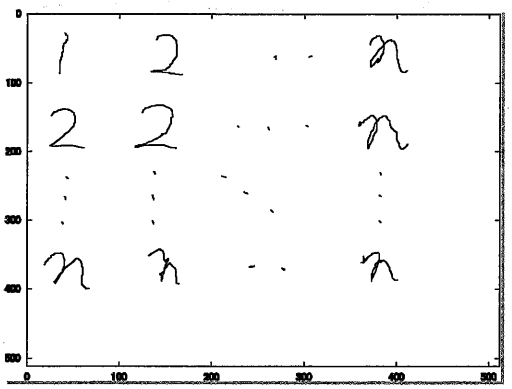


3.2 実験結果

サンプルNo.	①	②	③	④	⑤	合計
構造認識率	93%	93%	73%	13%	90%	71%



3.2 実験結果－成功例－



*****PROFILE*****

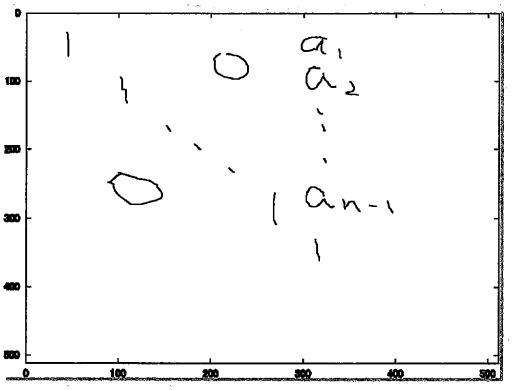
4*4 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (1,2)
6 : (1,4)
7 : (2,1)
8 : (2,2)
#12 : (2,4)
#16 : (4,1)
#20 : (4,2)
#24 : (4,4)

SUBSCRIPT LIST

CONNECTION LIST
1 : Connects (1,2) -> (1,4)
2 : Connects (2,2) -> (2,4)
3 : Connects (2,1) -> (4,1)
4 : Connects (2,2) -> (4,2)
5 : Connects (2,4) -> (4,4)
6 : Connects (2,2) -> (4,4)
7 : Connects (4,2) -> (4,4)

3.2 実験結果－成功例－



*****PROFILE*****

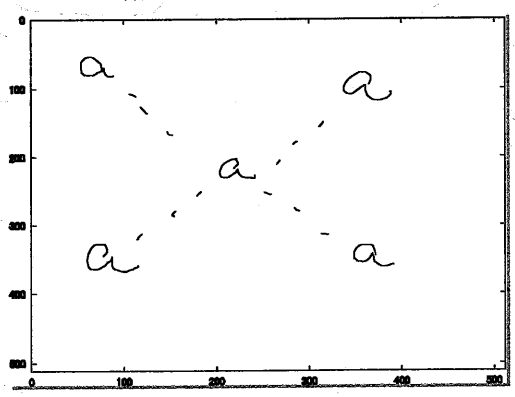
5*5 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (2,2)
6 : (4,4)
7 : (5,5)
8 : (1,5)
#10 : (2,5)
#15 : (4,5)
#19 : Region (1,2)(1,3)(2,3)((1,4)(2,4)(3,4)
#20 : Region (2,1)(3,1)(4,1)(5,1)(3,2)(4,2)(5,2)
(4,3)(5,3)(5,4)

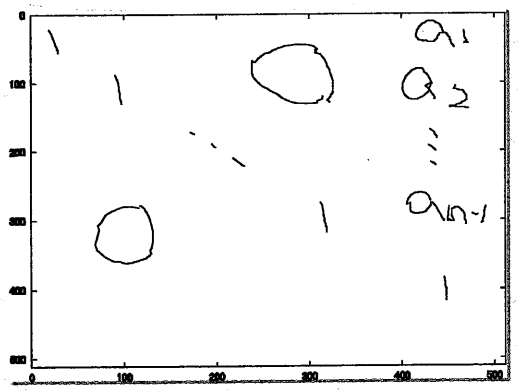
SUBSCRIPT LIST
1 : subscript of 8.
2 : subscript of 10.
3 : subscript of 15.

CONNECTION LIST
1 : Connects (2,2) -> (4,4)
2 : Connects (2,5) -> (4,5)

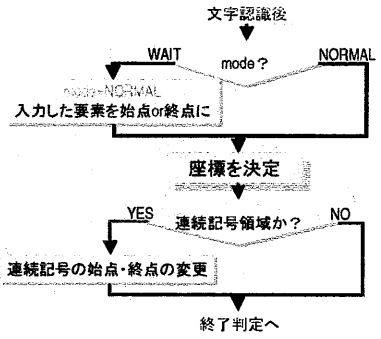
3.2 実験結果－失敗例－



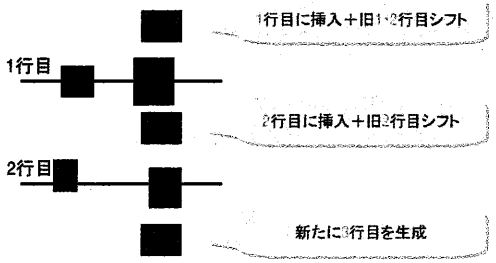
3.2 実験結果－失敗例－



3.3 システムの拡張－行・列の挿入－



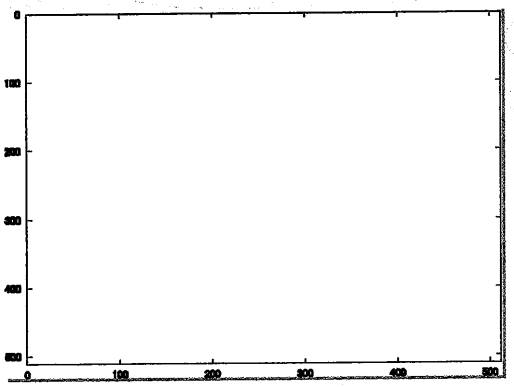
- 走査ライン上に要素がなかった場合、既に入力されている各行の要素のy方向中心線について平均をとり、この値と入力された文字のy方向の中心線の値とを比較する
- 挿入を行なった場合、それ以降の行情報はシフトする



3.3 実験条件－制約条件－

- 入力の際には、次のような制約条件を課した.
  - 行と列は空白によって格子状に区切られている
  - 連続記号を書いた後にその始点または終点に対応する要素を入力する場合の入力のタイミングは、連続記号入力の直後とする
  - 領域要素は最後に入力

3.3 実験結果－行・列間挿入を可能にした場合の成功例－




```
*****PROFILE*****

 4*4 MATRIX.

**ELEMENT LIST**
# 1 : (2,2)
# 5 : (4,4)
# 6 : (2,4)
#10 : (1,1)
#11 : (2,1)
#12 : (1,4)
#16 : (4,2)
#20 : (4,1)
#24 : (1,2)

**SUBSCRIPT LIST**

**CONNECTION LIST**
# 1 : Connects (2,2) -> (4,4)
# 2 : Connects (2,2) -> (2,4)
# 3 : Connects (4,2) -> (4,4)
# 4 : Connects (2,4) -> (4,4)
# 5 : Connects (2,1) -> (4,1)
# 6 : Connects (1,2) -> (1,4)
# 7 : Connects (2,2) -> (4,2)
```


第4章
考察

4 考察

- 添え字や連続記号を含む行列構造の認識率が良くない理由として、マウスを使った入力のため、小さな文字が書きづらかったことや、添え字や連続記号の判定の方法が厳しすぎたことなどが挙げられる
- 添え字を含むサンプルから添え字を消去して再実験を行なったところ、認識率は53%(全体で83%)であった
- 今回は入力順等に関していくつかの制約条件を課したが、今後はより自由度の高い入力にも対応可能な手法を考察する必要がある

第5章
結論

5 まとめと今後の課題

- 連続記号や領域要素などの略記にも対応可能な行列構造認識システムを提案した
- 手書き行列画像を用いた実験を行ない、提案手法の有効性を確認した
- 今後は筆記者による行列の書き方の違いや癖に対して頑健であり、またより自由な環境で入力できるシステムに改善していくことが課題となる

おしまい

豊住らの認識システム

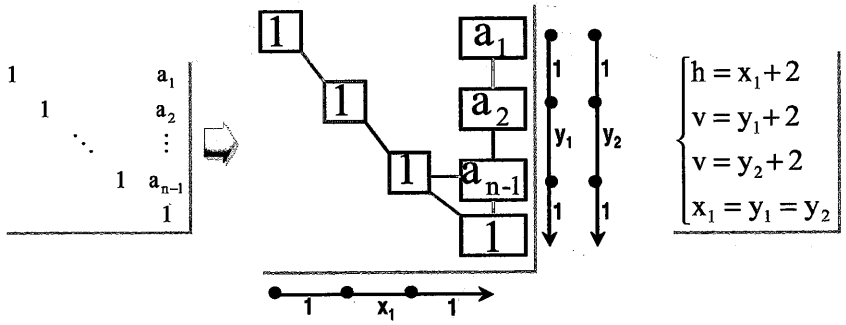
- 以前に同グループが提案した数式認識システムに行列認識機構を追加したもの
- ストロークの中心線のx座標とy座標に注目し、要素の座標を決定する
- 左括弧の入力で行列モードを開始し、右括弧の入力で終了
- 既存の行や列の間への新たなブロックの生成には対応せず
- 実験に用いた行列は次の2つ(15人×5回)

$$\begin{pmatrix} a+1 & b+2 \\ c+3 & d+4 \end{pmatrix} \qquad \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}$$

- 認識率は67%、行列が正しく検出された場合に限定すれば97%

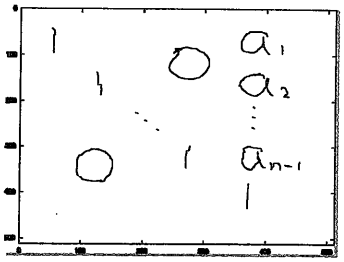
金堀らの認識手法

- 各要素を連結するネットワークを生成し、行列の高さ、幅について連立方程式をたて、可変長の部分を求めることにより、要素の位置を決定する



$$\begin{cases} h = x_1 + 2 \\ v = y_1 + 2 \\ v = y_2 + 2 \\ x_1 = y_1 = y_2 \end{cases}$$

出力例



*****PROFILE*****

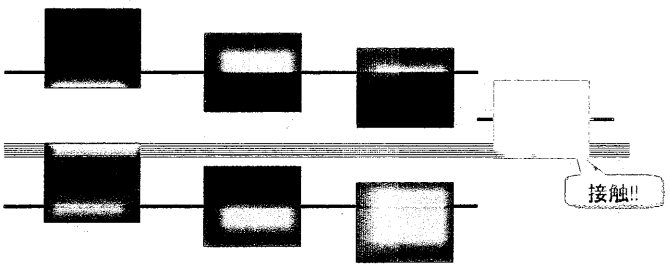
5*5 MATRIX.

ELEMENT LIST
1 : (1,1)
2 : (2,2)
6 : (4,4)
7 : (5,5)
8 : (1,5)
#10 : (2,5)
#15 : (4,5)
#19 : Region (1,2) (1,3) (2,3) (1,4) (2,4) (3,4)
#20 : Region (2,1) (3,1) (4,1) (5,1) (3,2) (4,2) (5,2) (4,3) (5,3) (5,4)

SUBSCRIPT LIST
1 : subscript of 8.
2 : subscript of 10.
3 : subscript of 15.
4 : subscript of 15.
5 : subscript of 15.

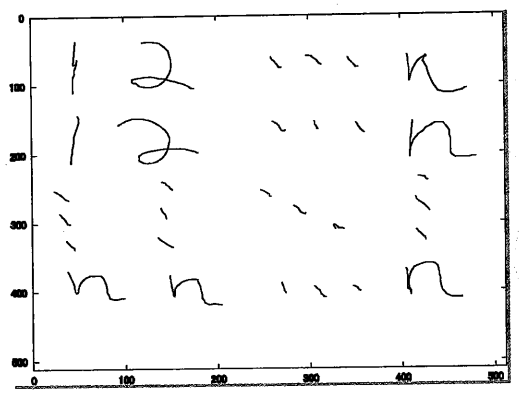
CONNECTION LIST
1 : Connects (2,2) -> (4,4)
2 : Connects (2,5) -> (4,5)

隣接する行・列に接触してしまった要素の処理



☆要素の中心線と2行目との距離より、1行目との距離のほうが短い
⇒1行目に配置

隣接する行・列に接触してしまった要素の処理



修士論文本審査資料

47