ヒューマンインタフェース 48－4
（1993．5．13）

オンライン手書き数式認識システム「*METAH*」の試作

村瀬敦史，佐藤俊，中川正樹
（東京農工大学大学院　工学研究科　電子情報工学）

本稿では、オンライン手書き入力による、数式認識システム「***METAH***」におけるUIに関する設計と、実際に試作したシステムの処理方式について述べる。数式は、その構造を示すために、二次元的な情報を有している。このような対象においては、手書き入力の利点が生かせると考えた。しかし、パタン認識には誤認識やリジェクトが伴う。UIの設計は認識エンジンの高い認識率と同等に重要である。本研究では、手書きは創造的な思考を妨害しないということに注目して、数式の考えながらの入力（創造入力）のためのUIを提示する。またそれと対をなす複写入力のためのUIの設計も論じる。一方、認識方式としては、数式を個々の記号パタンに分割し、それぞれを識別し、それらの記号間に幾何学的な連結を見い出し、それをもとに構造解析を行う。その処理方式についても述べる。

Prototyping of *METAH*, A Recognition system for On-line Handwritten Mathematical Expressions

Atsushi MURASE, Takashi SATOU, Masaki NAKAGAWA

This paper describes the design of user interfaces(UI), and pattern processing and recognition methods for an on-line recognition system '***METAH***' for the input of mathematical expressions by handwriting. Mathematical expressions possess geometrical information that expresses their structures. Handwriting is thus suitable for inputting such objects. However, pattern recognition entails misrecognition and rejection. Well designed UI's are as essential to the system as high recognition rates. Based on the fact that handwriting does not interrupt creative thinking, this paper presents UI's for the creative input of mathematical expressions. In contrast to the creative input, the design of UI's for their copy input is also discussed. As for the recognition processing, the prototype system partitions mathematical expressions into symbol patterns, identifies each symbol, recognizes their positional relationships and then parses them.

1．はじめに

飛躍的な低価格化、高性能化に伴い、現在計算機は幅広く普及している。身近なものとしては、文書の作成や管理に関する分野に広く普及していることに気づく。これには、ソフトウェアとしてはワードプロセッサ（以下ワープロ）や文書整形ソフトウェアが、ハードウェアとしては高解像度のプリンタが出現することで、手書きの文書とは比べものにならない高印字品質の文書を簡単に作成することが可能になったという背景がある。しかし、ここ数年多くの人々にこれらが使用された結果、ワープロなどの欠点が浮き彫りになってきたことも事実である。

文書の要素には、文章の他にも図、数式、イメージ、グラフなど、様々なものが存在する。文章の入力に関しては、仮名漢字変換技術の向上などにより、かなり快適なものとなっている。ところが、図や数式の入力に関しては、操作方法の学習や、入力の際に多大な労力を必要とするのが現状である。このような問題点は、その入力デバイスがキーボードとマウスであることに一因があると思われる。

ところで、最近スタイラスペンと表示一体型タブレットを用いた、手書き入力方式のシステムが注目されている。図や数式といったものは、二次元的な情報を有しており、手書き入力方式は、このような対象に有効的であると考えられる。特に、発想段階における入力については、一層手書きの利点が生かせると思われる。

そこで我々は、表示一体型タブレットを用いた手書き入力方式のシステムが実現されれば、計算機としての利点を生かしつつ、これらの問題点を解消できるのではないかと考えた。そして、文書要素の中でも、特に数式の入力において手書き方式が効果的であると考え、オンライン手書き数式認識システム *METAH* の構築を試みるにいたった。

ただし、計算機処理の恩恵を受けるためには、手書きパタンの認識が不可欠となる。ところが、この処理に 100% の認識率を期待することはできない。そこで、人間が苦にしない形で支援を引き出す UI の設計が重要となる。

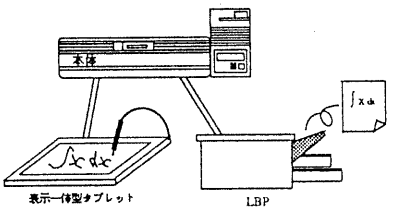

図１　システム概念図

本稿では、まず、数式を手書きで入力する際にどのようなユーザインタフェース（以下 UI）が最適であるのかということについて考察する。そして、その後我々が実際に構築した *METAH* プロトタイプにおける UI の設計、および、手書きパタン処理を含めた実現方式について述べる。

2．数式と手書き入力方式の適合性

既存の文書整形システムの中には、数式についても綺麗に整形し出力するソフトウェアがいくつか存在する。しかしながら、我々は我々自身の経験を通じ、特にその入力環境において、次に挙げるような不満を抱いた。

・数式を解きながら入力できる環境がない
・入力するのに、多大な労力と時間を要する

数式の作成形態には、その数式を解きながら筆記するような場合もある。このような場合に求められることは、入力のために生じる思考の中断を最小に止どめるということである。しかし、キーボードやマウスによる入力環境は、どうしてもある程度の学習や慣れが必要であり、
たとえ慣れたとしても、余計な操作が思考の中断を生じさせてしまう（もっとも、既存のシステムでは、数式を解きながら入力するということは、コンセプトとして持っていなかったのだろうが）。

また、数式は二次元的情報を有しており、それがその構造を示している。したがって、入力の際には、これらの情報を自由に表現できる必要がある。現在数式を整形出力するシステムには、TeX、Expressionist、Rof などが存在する。しかし、数式整形言語で文字列として表現する場合は、構造の表現のために余分な文字列（コマンド）を必要とする。一方、WYSIWYG のシステムでは、上記の問題は生じないが、非常に多くのモード変換が必要になり、多大な労力と時間を要する。

さらに、これらのシステムでは、上に述べたようにモード変換などの操作が頻発し、思考の継続を極めて難しくする。

以上のことをまとめると、

・キーボードやマウス、さらには入力ソフトウェアに関する学習や慣れが必要である
・モード変換などの、数式の入力以外に必要となる操作が多い

ということが原因となり、上記の問題点が生じていたのではないかと考えられる。特に、後者に関しては、結果として思考の中断が生じ、ユーザが何かを考えながら入力するといった UI の形態を実現する際に致命傷となっていたといえる。

次に、手書き入力方式がこれらの問題点を解消できるのかということについて考えてみる。まず、手書きの場合、特に意図的に制限を加えない限り（入力用の枠など）、自由な大きさの文字や記号を意図した配置で筆記することができる。このことにより、数式に存在する表記上の規則に沿って、自由にその構造を表現

することが可能となる。

また、手書きは、人間が生まれたときから親しんできた方式であり、その作業をほとんど無意識に行うことができる。さらに、モード変換など存在しないので、筆記という作業自体で生じる思考の中断は非常に少なくなる。これらのことにより、ユーザは、頭で何かを考えながら数式を入力することが可能となる。そして、その結果として、数式を解きながら、あるいは展開しながら計算機への入力が同時に行えることになる。

以上のことから、上記の問題点は、その入力方式を手書きにすることによって解消できるのではないかと考えることができる。

ただし、手書き入力にすることにより、新たな問題も生じる。それは、手書き記号パタン、そして、数式としての構造認識処理が必要不可欠になるということである。前述のように、人々が文書作成に計算機を用いるようになったのは、高印字品質な出力が得られるためである。したがって、手書きで入力されたパタンをそのまま出力するのでは、計算機を用いる大きな利点の一つを欠いてしまうことになり、「計算機の利点を生かしつつ」という当初の思想から逸脱してしまう。

また、これらの認識処理に伴う、誤認識、リジェクトに対する修正作業が必要となる。そのため、これらの作業により、思考の中断が生じてしまうおそれがある。しかしながら、この問題に関しては、システムのUIとして後処理型という方式を採用することで回避できると考えている（第3章参照）。

以上のことをふまえ、結論としては、上記の認識処理が自動化でき、最終的に高品質の出力が得られるならば、数式の入力方式という観点からは、手書き入力方式が適しているのではないかと考えることができる。

3．数式の筆記形態を考慮したＵＩの設計

3．1　充実したＵＩの必要性

従来人間の労力に対するコストより高価なものとされていた計算機のコストが逆転し、今ではいかに人間の労力を効率よく使用できるかということが注目点となっている。また、計算機でありながら、計算以上にUIが多くCPUやメモリを消費するようになってきている。

UIの充実が重要視されるのには、エンドユーザの層が拡大したことも原因となっている。そして、エンドユーザが快適に計算機システムを利用していくためには、より自然で使い勝手のよいUIを提供する必要があると考えられる。

次に、手書き入力方式のシステムであるという関連からも、UIの充実が必要となる理由がある。

手書きで入力された数式の構造解析という観点からは、過去にも研究報告がいくつかなされている[1,2,3,4]。しかし、それらの研究では、修正のUI、あるいは、それを考慮した処理方式について触れているものは見受けられなかった。ところが、手書きで入力し、それを認識する以上、誤認識などに対する修正作業が問題となる。この作業は、わずらわしいものには違いないが、パタン認識が不完全な以上、計算機処理の恩恵を受けるためには必要不可欠である。したがって、いかに手書きの長所を殺さないUIを実現するか、ということも手書き入力方式のシステムにとっては重要なことである。

ところで、最近の計算機業界では、「手書き入力」が非常に注目を浴びている。このような動きは、アメリカから火がついたが、現在では、日本においても様々なところで研究開発が行われている[5,6]。ただし、パタン認識という観点からは20年も前から様々な研究が行われ、それなりの成果も挙げている。しかしながら、最近では、単にパタン認識の入力手段というだけではなく、使い勝手のよいUIの実現、特にエンドユーザ向けのシステムとして手書き入力方式が脚光を浴びている。我々も、手書きがより自然な方式であるという事実をもとに、手書き入力方式である本システムが使い勝手のよいものとなる可能性が高いということを信じ、その利点を生かしたUIを実現したいと考えた。そして、修正の作業においても、手書きを用いることが効果的と考え、そのような方法を採用することを考えた。

以上本節では、充実したUIの必要性について述べた。やはり、たとえあるシステムが、その機能として非常に高度な処理を行えるとしても、そのUIが貧弱では、ユーザはいずれそのシステムを使わなくなってしまう。逆に、多少機能的には劣っていても、その操作インタフェースなどが充実していれば、ユーザが機能の足りない部分を補うことも期待できる。

3．2　数式の筆記形態を考慮したＵＩ

本節では、数式を入力する際のUIについて述べる。

手書き数式認識システムのUIとして、どのようなものがよいUIといえるだろうか。我々は、その指針として、従来人々が紙とペンを用いて数式を筆記していたときと同等、または（計算機の利点を生かすことで）それ以上の快適さを求めることが必要であると考えた。これは、もし計算機を用いることで快適さが損なわれるならば、そのシステムの存在価値が半減してしまうと考えたためである。そこで、そのようなUIを実現するために、まず従来紙とペンを用いて数式を筆記していたときのことを振り返ってみる。

通常人間がペンで紙に数式を書くときは、どんな書き方があるのだろうか。言い換えれば、どんな状況で「数式を書く」という動作を行うだろうか。結論としては、大きく分けて、

- 複写系 … 従来書かれた、または印刷された数式を新たに書き写す場合
- 創造系 … 数式を解いたり、展開する場合

の二つの形態が存在するといえる。紙とペンという環境では、これら二つのどちらの形態においても対応することができた。ところが、従来のキーボードとマウスを用いたシステムは、複写系の作業をねらいとしたもので、創造系における作業は対象とされていなかった。したがって、もし数式を解いたものを綺麗に出力したい場合は、一度紙面上に書いてから、改めて複写系の作業を行うといった、いわば二度手間な作業を強いられた。これでは、紙とペンの環境と同等の快適さとはいえない。そこで *METAH* では、創造系の作業が可能となる UI の実現を目指すことを考えた。

では、創造系の作業を行うためには、どのような UI とすればよいだろうか。重要なことは、思考の中断をできるだけ少なくするということである。そこで、次に認識のタイミングに着目し、創造系の作業に適した方式について考えてみる。認識のタイミングとは、数式を入力（筆記）しているときに、どれだけの量を書いたところで認識処理を走らせ、結果を表示するかということである。

認識のタイミングを検討するために、ここで、仮名漢字変換の UI を考察する。我々がワープロなどを用いて文章を入力するときには、頭の中で文章を考えながら、同時進行でそれを入力するときがある。これは、上記の創造系の作業といえる。このとき、ほとんどの人は、ある単位で変換キーを押し、そこまでの仮名列を漢字混じりのものに変換する。このとき、誤変換が生じるので、確認、そして修正という作業が入る。先に、創造系の作業においては、思考の中断が大敵であると述べた。それにもかかわらず、我々は文章を入力している途中で次々と変換キーを押す。これは、仮名だけの文章を読むのが困難なためと思われる。我々が考えながら文章を入力するときには、少し前に入力した部分を読み返すという作業と、続きを入力するという作業を繰り返す。したがって、直前に入力した文章をスムーズに読めることが重要となる。そこで、ユーザは思考が中断しないように注意しながら、変換、修正、確定をよぎなくされる。ただし、これらをきらい、一段落ぐらい何もしないで（あるいは自動変換のままで）入力するユーザもいる。しかし、いずれにしても、ユーザは苦肉の策をとっているといえる。

手書き数式認識の際にも、仮名漢字変換のように、ある程度の単位で逐次認識処理を行う方法も考えられる（逐次処理型）。

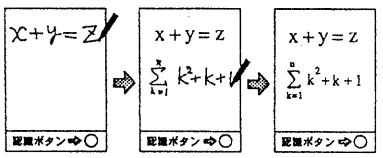

図２　逐次処理型

しかし、手書きの場合、特に認識処理を行わなくても、ユーザが入力した手書きパタンを見れば、直前に何を入力したかのかがわかる。したがって、入力するときには認識処理を一切行わず、ただ書くことだけに専念し、一段落したところでまとめて後から認識処理をかけ、その修正を行うといった方法が可能となる（後処理型）。後処理型であれば、筆記中に思考の中断は生じない。その結果、創造系の作業、つまりタブレット上で数式を解きながら入力するといったことが可能となるわけである。したがって、創造系の作業に対応できる UI という観点からは、後処理型を採用することが適当であると考えられる。

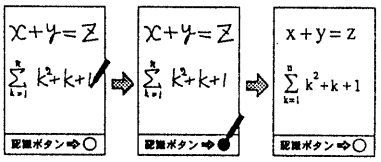

図３　後処理型

これまでは、創造系の作業に適した UI について述べた。確かに、創造系の作業を行える環境の実現が本システムの第一目標ではあるが、前にも述べたように、数式の筆記形態には複写系も存在する。そして、本システムが完成したときも、複写系の作業に用いられることが十分に考えられる。そこで次からは、複写系の作業に適した UI について考えてみる。

複写系の作業では、何を入力するかが決定されているので、入力の際に思考の中断が生じることにあまり問題はない。したがって、この場合問題となるのは、誤認識結果に対する修正作業である。つまり、複写系においては、いかに修正の手間を軽減できるかということが重要なこととなる。そこで、数式の構造の作成方法に着目し、どのような UI にすることにより修正の手間を軽減できるかということについて考えてみる。

創造系における UI では、思考の中断を生じさせないために、筆記中に余計な表示や認識処理を行わないこと、また、ユーザに不必要な操作をさせないことが必要である。したがって、入力された数式を綺麗に出力するためには、そのようにして書かれた数式の記号の大きさや位置関係ともとにして、計算機自身が自動的にその構造を抽出する必要がある（構造抽出型）。しかし、この場合、数式の構造解析をすべて計算機が行うことになるので、誤認識の確率は増すことが懸念される。ただし、創造系の場合は、筆記中に思考の中断が生じないことが必須条件なので、それでも構わないと思われる。

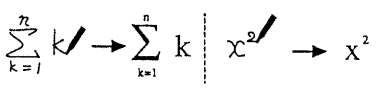

図４　構造抽出型

ところが、複写系の場合は、逆に入力の際に多少の手間暇をかけても、できるだけ誤認識による修正の手間を少なくしたい。そこで、筆記の際にユーザが計算機の手助けをすることで、誤認識を減らすことを考えた。具体的には、数式の構造の一部を定義しながら入力するというものである（**構造定義型**）。例えば、ユーザが「Σ」を記入した時点で、あるいはメニューから【シグマ】というボタンを選んだときに、「Σ」の上下と右にそれぞれの部分のを書き込む場所として矩形を表示する。そして、ユーザは、その矩形の中に内容を記入するといった方法である。

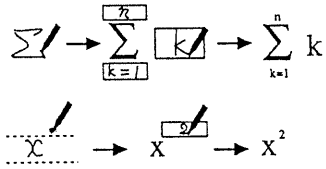

図５　構造定義型

この方法を用いても、それぞれの記号の認識処理は行う必要があるので、それの修正作業は必要となる。しかしながら、二次元的な情報を用いる数式の構造認識処理の負担は非常に軽減される。したがって、全体の誤認識が少なくなり、ユーザによる修正の手間も軽減されると考えられる。そこで、複写系の作業には、構造定義型の UI を採用することが適しているといえる。

ところで、以上で述べたように二つの筆記形態に要求されることは相反するものとなっている。そして、その要求を満たす UI の仕様も異なる。これらを統合した UI を設計することは不可能ではないかもしれない。しかし、我々は、とりあえずプロとタイプとして、それぞれの筆記形態を想定した２種類の UI を用意し、そのときに応じて好みの方をユーザが自由に選択できる方式を採用することにした。これは、単純な発想ではあるが、紙以上の快適さを求めるという当初の目的を果たすためには最適であると考えている。なお、これら二つ形態は、童話『うさぎとかめ』の両者にその性質が非常に似ているので、*METAH* における二つの UI をそれぞれ「うさぎ」（創造系）と「かめ」（複写系）と名付けた。

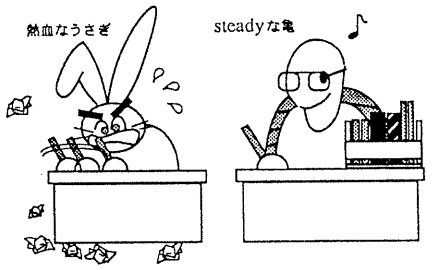

図６　２種類のＵＩ「うさぎ」と「かめ」

３．３　修正作業におけるＵＩ

前述のように、誤認識に対する修正作業は、ユーザにとってはできれば行いたくないものである。しかしながら、本システムが認識処理を行う以上、避けて通ることはできない。そこで、我々ができることは、ユーザが修正作業を行っていても、なるべく不快感を感じないような UI を実現することである。そして、そのような UI を実現するために、次に挙げる二つの点が重要であると考えた。

① 誤認識の発生箇所が即座にわかる
② 簡単な操作で修正が行える

そこで、次からは、それぞれの点にどのように対応するのかということについて述べる。

(1) ２種類のウィンドウモード

項目①を実現するためには、認識前の手書きパタンと認識後の結果の対応づけが、瞬時に行えることが必要である。認識結果の表示方法については、入力されたパタンの付近に出力する方法も考えられるが、この方法では、画面が繁雑になってしまうおそれがある。そこで、本システムでは、入力パタンをそのまま表示するウィンドウと、それとは別のウィンドウを横に用意し、それに認識結果を表示するという方法を用いる。ただし、位置の対応が即座に行えるように、二つのウィンドウにおけるスクロールは連動させる。このことにより、例えば手書きパタンをスクロールしても、常にその領域に対応する結果がもう一つのウィンドウに表示されることになる（図１３参照）。

ところで、修正作業を行う上では、上記の方法が適しているのだが、数式を手書きで入力する際には、問題が生じる。上述のように、入力の際には、直前に筆記したものを即座に見ることが可能な環境が必要である。しかし、液晶画面に二つのウィンドウを用意すると、入力用ウィンドウの大きさがやや狭くなってしまう。そこで、入力の際には、一つの大きなウィンドウ、修正の際には、パタンと結果が表示される二つのウィンドウという２種類のウィンドウモードを設けることとした（図１２参照）。これにより、入力と修正の両者に適した表示方法が可能となる。なお、本システムの開発においては、特にウィンドウシステムを用いていないので、これらの管理を行うモジュールも自作した。

(2) 手書きを用いた修正作業

次に、上記②についてだが、これは、ペンの特性を生かすことにより実現可能であると考えられる。一般的に修正作業は、対象の指定とコマンドの選択の二つの動作からなる。ところが、キーボードやマウスによる操作では、対象の指定が即座に行えなかったり、コマンドを選択するために、いくつかのメニューを出して選ぶなどの段階的な操作が必要であった。しかも、

複雑な形をした対象を選択するために、何回も同じ作業を繰り返す必要が生じたりしていた。これに対しペンの場合は、直接対象を指定できるということが、大きな特徴の一つとなっている。また、ジェスチャを用いることにより、対象の指定とコマンドの選択を一度に行うことが可能となる。つまり、修正作業にペンを用いること自体が、上記②を実現することとなるわけである。

ところで、ペンを用いた対象の指定についてだが、対象をペン先でポインティングする他に、囲み線を用いて対象の指定を行う方法が考えられる。つまり、指定したい対象の周りをペンでなぞることにより、対象を特定するわけである。この方法を用いることで、

・複数の対象を同時に指定する
・多彩な形で指定する

といったことが可能となる。

この方法を用いるには、対象が線で囲んだ領域の中にあるか否かという判定を行う処理が必要となる。そして、ユーザに不快感を与えないために、この処理を高速に行うことが重要となる。そこで我々は、この処理を高速に行うアルゴリズムについて研究、実験を行い、独自の手法を考案した[7]。

なお、ペンを用いた修正作業例については、第４章の第１節と第２節を参照されたい。

４．*METAH* プロトタイプの構築

第３章で述べた構想をもとに、*METAH* プロトタイプの構築を行った。ただし、今回の構築では、「うさぎ」の UI だけを実現した。また、最終的に数式をフォーマットする処理が実現されていないので、現在では手書きで入力された数式の構造を示すものが最終的な出力となっている。

METAH は、次に示す三つの処理から構成されており、番号の順に処理が行われる。

(1) 記号分割処理
(2) 記号認識処理
(3) 構造化処理

次からは、それぞれの処理の概要について述べるが、記述の都合上 (2) の記号認識処理の概要を最初に述べる。。

４．１　記号認識処理

METAH が認識対象とする数式は、文字と記号によって構成されている。そこで、入力された数式の構造を認識するために、まず、文字と記号を文字コード（符号）に変換する必要がある。この処理を行うのが、**記号認識処理**である。なお、今「文字と記号」と述べたが、これ以降では、符号化する対象すべてを一意に「記号」と呼ぶ。

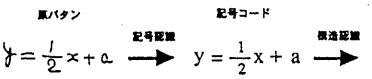

図７　記号認識処理

我々の研究室には、*JOLIS-2E* というオンライン手書き文字認識システムが存在している[8]。そこで、*METAH* における記号認識処理には、このシステムを組み込むことを考えた。ただし、*JOLIS-2E* は、漢字などを含む 2265 カテゴリを対象としたシステムであり、数式に用いられるような、小規模、低画数な記号を指向して構築されたものではない。そこで、数式固有の記号に適した処理にするための、各種変更、実験を行った[9,10]。しかし、ここでは、その詳細については割愛する。

先に、ペンを用いた修正作業について述べたが、次に、記号認識処理によって生じた誤認識結果に対する修正作業の様子を示す。

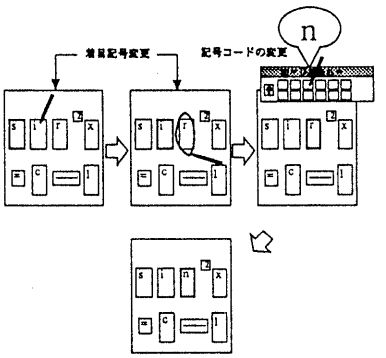

図８　記号認識処理の誤認識修正作業

４．２　記号分割処理

表記規則に沿ってタブレット上に筆記された数式は、システムからみれば、単なるストロークの列でしかない。そこで、上記の記号認識処理を行う前に、ストローク列を記号ごとに分割処理が必要となる。これを**記号分割処理**と呼ぶ。

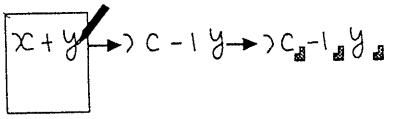

図９　記号分割処理

記号分割処理では、入力されたストローク列が持っているなんらかの情報を用いて、自動的にストローク列を記号の単位に分割する。これには、次に示すような情報を用いることが有効ではないかと考えた。

・ストローク間に生じる時間
・ストロークが占める面積の重なり関係
・ストロークの接触関係
・ストロークの重心間の距離

我々は、これらの情報を用いた処理が本当に有効なものとなるか、また、その有効性がみられ、実現を考えたときに、その処理に必要となるパラメータなどの測定も兼ね、様々な実験を行った。そして、最終的に、ストロークの重心間距離を用いた記号分割処理を実現した[11]。なお、この処理では、約 90% の分割率、そして、実用的な処理速度を実現している。

また、記号分割処理においても、誤分割結果の修正にペンを用いた修正方式を採用している。次に、その様子を示す。

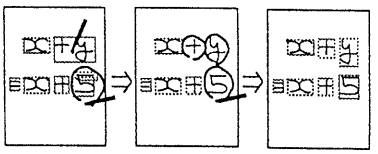

図１０　記号分割処理の誤分割修正作業

４．３　構造化処理

入力されたストローク列に対し、記号分割処理と記号認識処理を施すことにより、入力されたパタンは、様々な大きさの記号群となる。数式の構造認識では、これらの記号の大きさや位置をもとに、その構造を表現する構造木を作成する。この処理は、*METAH* において核となるもので、これを構造化処理と呼ぶ。

先に、数式の構造は、記号の大きさや位置で表現されると述べた。これは、具体的にいえば、「右上に小さく」といったように、記号の近傍にどのような幾何学的関係で他の記号が存在しているかという情報が積み重なったものであるといえる。この幾何学的関係は、目に見えない鎖のようなもので、数式全般に存在している。

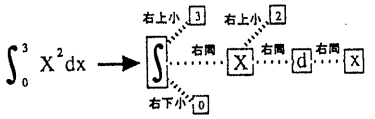

図１１　記号間の幾何学的関係

この鎖が、数式全般に連ねられていることにより、数式全体をたどれるようになる。我々は、これを用いて、その数式の構造を解析できるのではないかと考えた。そこで、*METAH* における構造化処理では、この手法（連結探索手法）を用いて数式の構造解析を行う[12]。また、従来の数式の構造解析手法においては、

・人間が理解しにくい
・修正情報の維持を容易に行えない

といったことが問題点となっていたが、我々が考案した連結探索手法は、これらを解消できるものとなっている。

最後に、実現した *METAH* プロトタイプの実行画面例を次に示す。

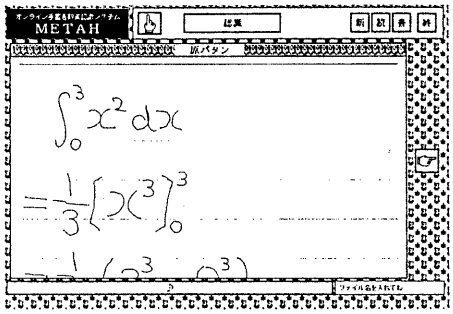

図１２　手書きパタンの入力

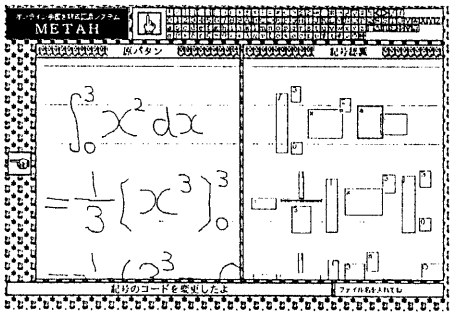

図１３　記号認識処理結果の修正

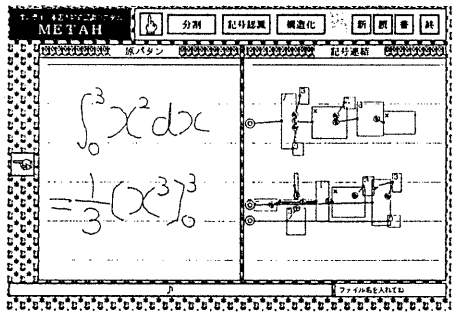

図１４　記号の連結関係の表示

５．おわりに

本研究では、次のような成果を挙げることができた。

・数式を解きながら入力が行える UI の実現
・記号間の幾何学的関係を利用した、数式の構造解析手法（連結探索手法）の考案

しかし、本研究は、まだ始めたばかりのものであり、今後も継続して行っていくつもりである。また、今後残された課題としては、

・数式フォーマッタ
・「かめ」の UI
・ジェスチャを初めとする手書きを生かした修正、編集操作の充実

などの実現が挙げられる。

また、将来的には、手書きの文章や図を認識するシステムと組合せ、統合的な手書き入力環境の構築を目指したい。

参考文献

[1] SHI-KUO CHANG : "A Method for the Structural Analysis of Two-Dimensional Mathematical Expressions", INFORMATION SCIENCES (1970)

[2] ROBERT H. ANDERSON : "Syntax-Directed Recognition of Hard-Printed Two-Dimensional Mathematics", Ph.D. Thesis, Div. of Eng. and Appl. Phys., Harvard Univ., Cambridge, Massachussetts (1968)

[3] CLAUDIE FAURE and ZI XIONG WANG : "AUTOMATIC PERCEPTION OF THE STRUCTURE OF HANDWRITTEN MATHEMATICAL EXPRESSIONS", Computer Processing of Handwriting, pp.337-361 (1990)

[4] ABDELWAHEB BELAID and JEAN-PAUL HATON : "A Syntactic Approach for Handwritten Mathematical Formula Recognition", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL.PAMI-6, NO.1(1984)

[5] R.Carr, D.Shafer : "The Power of PenPoint", Addison-Wesley Publishing Company (1991)

[6] 「ユーザインタフェースの新風 ペン入力用 2 大 OS の姿」, 日経マグロウヒル社,日経バイト 4 月号, pp.232-235 (1991)

[7] 佐藤俊他 : "手書きインタフェースのためのペンの囲みによる対象判定アルゴリズムの実現と評価", 信学技報 PRU92-88 (1992)

[8] 平松徹他 : "オンライン手書き文字認識システム JOLIS-2 の構成", 情報処理学会第 35 回全国大会 3H-5 (1987)

[9] 村瀬敦史他 : "手書き入力による数式認識システム", 情報処理学会ヒューマンインタフェース研究会報告, 36-1 (1991)

[10] 村瀬敦史他 : "手書き数式入力におけるヒューマンインタフェースの考察と認識システムの実現", 情報処理学会第 42 回全国大会5D-7(1990)

[11] 佐藤俊他 : "手書き数式認識のためのストローク間の位置関係を用いた記号分割処理の基本方式", 情報処理学会第 46 回全国大会4K-7(1993)

[12] 村瀬敦史他 : "オンライン手書き数式認識システム「METAH」の実現", 情報処理学会第 46 回全国大会 4H-5 (1993)

[13] 曽谷俊男他 : "遅延認識を用いた手書きユーザインタフェースの基本設計" : 情報処理学会論文誌 第 34 巻第 1 号別刷 (1993)

[14] 中川正樹 : "発想支援手書き環境の硬い技術と柔らかい技術", 情報処理学会第 34 回プログラミングシンポジウム報告 (1993)