# 毛筆フォントの掠れ・滲み処理システムについて

中村剛士† 真野淳治†
世木博久† 伊藤英則†

本論文では，TrueTypeなどの既存の毛筆フォントを処理し，疑似的な掠れまたは滲みのある毛筆フォントに変換出力する手法について述べ，本手法により作成した掠れ・滲み毛筆フォントを例示する．本システムでは，まず，毛筆フォントを入力し，これを2値画像に変換する．次に，2値画像フォーマットに変換した毛筆フォントを細線化し，毛筆フォントの文字骨格を獲得する．さらに，文字骨格を構成する各画素上に，筆触カーソルと呼ぶドットパターンを配置していく．以上の処理を実行し，入力毛筆フォントを疑似的な掠れまたは滲みのある毛筆フォントに変換出力する．なお，掠れおよび滲みの変化については，それぞれパラメータを設定することで多彩に表現することが可能である．

# A System for Generating Various Calligraphy Characters with Scratched Look or Blurred Look

Tsuyoshi Nakamura,† Junji Mano,† Hirohisa Seki† and Hidenori Itoh†

In this paper, we describe the approach to generate artistic calligraphy characters which have scratched look or blurred. The artistic characters which our system generated are shown in this paper. The system generates artistic characters from the original calligraphy fonts which are TrueType, Bitmap, and so on. The process of the system is divided into the following stages. In the first place, an original calligraphy font is inputted, and it transforms into bitmap image. In the second place, the skeleton of the image is found by using thinning algorithm. Furthermore, brush-touch cursors are placed on each pixel of the skeleton. The brush-touch cursor is expressed as dot patterns. By the above process, scratched look and blurred are expressed on calligraphy characters. Some parameters which users can set make it possible to express artistic calligraphy fonts which have variety of the scratched look and blurred.

## 1. はじめに

近年，ワープロ機器はもちろん，パソコン上で使用できるワープロソフトや葉書印刷ソフトの普及が著しい．これにともない，単に文書が印刷できるだけでなく，様々な機能を備えた製品が登場している．とくに，ユーザからの要望が強い機能の1つとして，毛筆文字の使用がある．これについては，各ソフトとも早くから対応し，現在では標準装備ともいえるほどの機能になっている．

葉書印刷ソフト等で使用できる毛筆文字の大部分はTrueTypeに代表されるアウトラインフォントであり，印刷書体である明朝体やゴシック体と同列に扱われている．しかし，明朝体やゴシック体などの印刷書体と異なり，毛筆書体は本来手書き文字の1つである．この手書き文字という視点から毛筆書体を観察した場合，実際の手書き文字ならば筆の軌跡や掠れ・滲みなどが文字上に見てとれるのに対し，現在使用されているほとんどの毛筆フォントには，それがあまり表現されていない．

仮に，掠れおよび滲みのある毛筆文字をアウトラインフォントによって表現した場合，そのデータ量は莫大なものになる．さらに，掠れや滲みのパターンを数種類用意しようとすれば，さらなるデータ量の増加を招くことになる．一般に，アウトラインフォントはビットマップフォントに比べ，データ量が少なく，拡大縮小などの変形が容易である．しかし，掠れや滲みのある毛筆文字をアウトラインフォントによって表現しようとすれば，先に述べたようにデータ量が増加することはもちろん，拡大縮小などの変形によって掠れや滲みの美しさをそのまま保持することが可能であるか否

† 名古屋工業大学知能情報システム学科
Department of Artificial Intelligence and Computer Science, Nagoya Institute of Technology

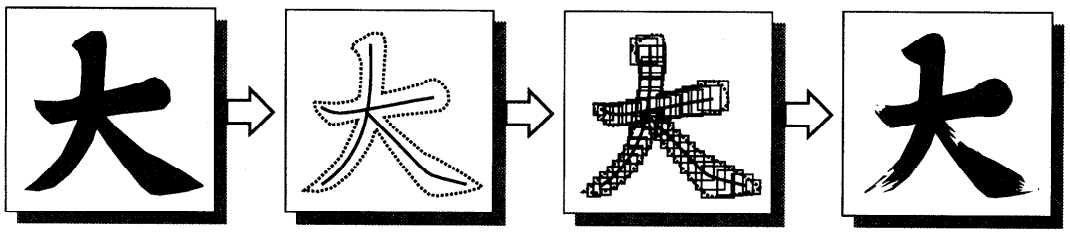



図1 システムの処理のながれ
Fig. 1 Process of the system.

か不明であり，アウトラインフォントの利点は薄れる．

一方，毛筆文字をフォントデータという形で保持するのではなく，異なる形の毛筆文字データを利用して毛筆文字を生成する研究がいくつか報告されている．その主なものとして，書字知識を利用する生成手法[11),12)]や，階層分解合成法による生成手法[13)~15)]，電子ペンを用いた一画ごとの描画入力による手法[2)~9)]が存在する．

これら数ある毛筆文字生成に関する研究の中で，電子ペンを用いた一画ごとの描画入力による手法は掠れや滲みの表現手法について詳しく触れている．この研究では，電子ペンを用いて漢字の一画を描画入力し，それを毛筆画に変換する処理を繰り返し行うことで，毛筆漢字を作成するものである．このとき，電子ペンの筆圧，筆速をファジィ推論することで，掠れや滲みの度合を決定し，毛筆文字上に表現する．この方式では，描画入力を用いているため，個人の筆記技能が文字に反映される．したがって，個性的な毛筆文字を作成可能である反面，万人が整った毛筆文字を作成できるわけではない．

本論文では，毛筆フォントのデータ量を増やすのではなく，また毛筆文字の基本的な形状に対して個人性を反映させることなく，掠れおよび滲みのある毛筆文字を生成する手法について述べる．

本システムは，毛筆フォントを入力し，それを処理することで疑似的な掠れおよび滲みのある毛筆フォントに変換出力する．入力する毛筆フォントのデータフォーマットは市販のアウトラインフォント，またはスキャナ等で取り込んだ毛筆文字であっても，2値画像（ビットマップ）にフォーマットを変換することで処理できる．したがって，データ量を増やすことなく，掠れや滲みのある毛筆フォントを適時作成することができる．

また，本システムでは，掠れおよび滲みのパターンの変化を多段階につけることができ，1つの入力毛筆フォントから様々な掠れおよび滲みのある毛筆フォントを作成でき，かつ掠れや滲みによる変化以外に文字形状を変化させることもない．

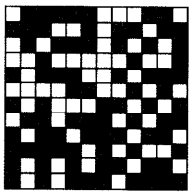


図2 筆触カーソルの例（2値カーソル）
Fig. 2 An example of brush-touch cursor (half-tone cursor).

## 2. システム概要

本システムは，毛筆フォントを入力し，それを処理して掠れ・滲み毛筆フォントとして変換出力するシステムである．

処理の流れを**図1**に示す．まず，入力した毛筆フォントを2値画像（ビットマップ）フォーマットに変換する．ここで，フォーマット変換した毛筆フォントを入力ビットマップフォントと呼ぶ．次に，Hilditch の細線化法[1)]により入力ビットマップフォントを細線化して，文字の骨格を獲得する．さらに，得られた文字骨格を構成する画素（骨格構成画素）それぞれを中心に適当なサイズの筆触カーソルを配置し，掠れまたは滲みのあるビットマップフォントを出力する．筆触カーソルとは，**図2**に例示するようなドットパターンであり，これを順次配置していくことで，疑似的な掠れおよび滲みを形成する．なお，図2に示すような黒（1）と白（0）の2値で構成される筆触カーソルを2値カーソルと呼び，黒（1）の部分が紙に墨のついた部分を表現し，白（0）の部分が墨のつかない部分を表現する．

## 3. 掠れ・滲み処理

### 3.1 文字骨格の獲得

文字骨格の獲得方法には，2 章で述べたように，Hilditch の細線化法を用いる．Hilditch の細線化法とは，2 値画像（ビットマップ）を処理の対象とし，画像部分を黒（1），背景部分を白（0）として，画像部分の最外郭に位置する画素を 1 層（1 ドット）削除する処理を線幅が 1 になるまで反復する手法である．たとえば，2 値画像の片仮名「ノ」を Hilditch の細線化法によって処理した場合の例として**図 3** に示す文字骨格をあげる．

ここで，2 値画像の画像部分の最外郭に位置する画素を 1 層削除する処理を 1 回の細線化処理として数えることとする．**図 4** は数回の細線化処理を繰り返し，文字骨格を獲得するまでの過程を示す．0 回目の細線化処理の結果，すなわち細線化処理を 1 度も実行していない段階で最外郭に位置する画素に細線化回数 0 を記録する．さらに，1 回目の細線化処理を実行し，その結果初めて最外郭に位置した画素には細線化回数 1 を記録する．同様にして，2 回目，3 回目の細線化処理の結果，初めて最外郭に位置した画素にも細線化回数 2，3 を記録する．図 4 では，各画素に記録した細線化回数を濃淡によって表現した．図 4 右上に濃淡と細線化回数との対応を示す．このように，$t$ 回目の細線化処理の結果，初めて最外郭に位置した画素には細線化回数 $t$ を記録する．なお，文字骨格の獲得が完了した時点であっても，最外郭に位置しない画素については最後に実行した細線化回数 $T$ を記録する．

Hilditch の細線化法は画像の最外郭から順次，1 層ずつ画素を削除していく処理であることから，各骨格構成画素に記録した細線化回数 $t(\geq 0)$ は，その骨格構成画素から元画像（入力ビットマップフォント）の最外郭までの距離に相当する．本システムは，各骨格構成画素に記録した細線化回数 $t(\geq 0)$ を利用し，配置する筆触カーソルのサイズを決定するが，詳細については次節で述べる．

### 3.2 2 値カーソルの生成

本システムは筆触カーソルデータとして，$m \times m$ サイズの多値カーソル☆を 1 つデータとして持ち，多値カーソルから 2 値カーソルを変換生成し（**図 5**），文字骨格上に 2 値カーソルを配置する．2 値カーソルは，それを配置する骨格構成画素ごとにサイズが異なり，

☆ 筆触カーソルには，カーソルを構成する各成分が 0 から $HIGH(=255)$ までの値をとる多値カーソルと，0 または 1 の 2 値をとる 2 値カーソルがある．

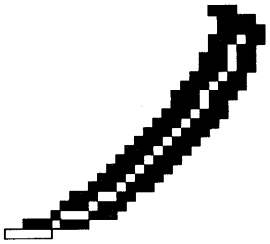


**図 3** 片仮名「ノ」の文字骨格
Fig. 3 The skeleton superimposed over the character image.

そのサイズを $n \times n$ と表すと，$n$ は 2 値カーソルを配置する骨格構成画素ごとに次式により決定する．なお，式中の $t$ は，各骨格構成画素に記録した細線化回数を表す．また，$t = T$ のときの $n$ は最大値をとるが，そのときの $n$ を最大サイズ $N$ とする．

$$n = 2(t+1)+1 \qquad 0 \leq t \leq T \tag{1}$$

本システムは，線形補間法によって，多値カーソルを上式で決定した $n \times n$ のサイズに拡大または縮小し，それを 2 値カーソルに変換する．$n \times n$ サイズに変更後の多値カーソルの $(i,j)$ 成分を $D_n(i,j)$ と表し，さらに，その多値カーソルから生成した 2 値カーソルの $(i,j)$ 成分を $d_n(i,j)$ と表す．

2 値カーソルの変換生成には，**図 6** に示すようなしきい値関数を用いる．横軸に 2 値カーソルのサイズ $n$，縦軸にしきい値 $th(n)$ をとる．また，$n=3$ のときのしきい値を $th(3) = th_3$，$n = N$ のときのしきい値を $th(N) = th_N$ と表し，サイズ 3 から $N$ の範囲で線形に変化するとし，次の関係を満足する．

$$0 \leq th_N < th_3 < HIGH\,(=255) \tag{2}$$

このしきい値関数を用いて $n \times n$ サイズの多値カーソルを同サイズの 2 値カーソルに変換する．2 値カーソルの $(i,j)$ 成分 $d_n(i,j)$ は，次式によって黒（1）または白（0）に設定する．ただし，$0 \leq i,j < n$ である．

$$d_n(i,j) = \begin{cases} 1 & D_n(i,j) \geq th(n) \\ 0 & D_n(i,j) < th(n) \end{cases} \tag{3}$$

一般に，毛筆文字においては筆の入り，払いなどの文字の細い箇所などについては掠れを生じやすい傾向がある[10]．本システムでは，元画像（入力ビットマップフォント）における文字の細い箇所付近に位置する骨格構成画素では，元画像の最外郭までの距離すなわち記録した細線化回数 $t$ が比較的小さい値をとることから，その細線化回数 $t$ から算出する筆触カーソルは縮小サイズとなる．

図 6 から分かるように，筆触カーソルのサイズ $n$ が

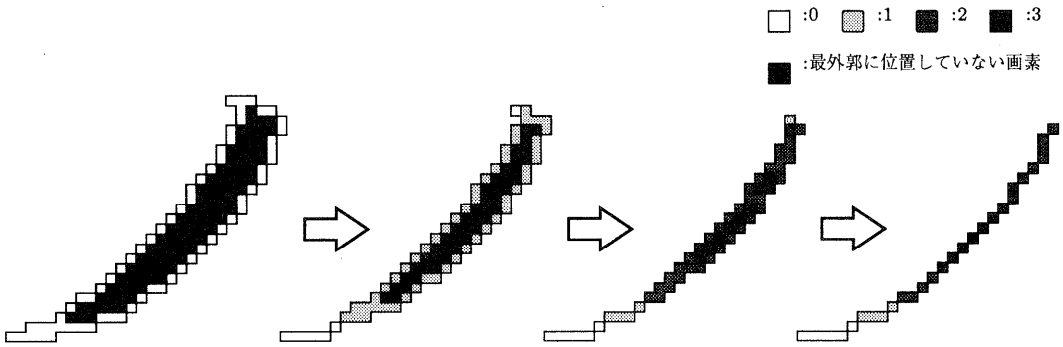



図4 細線化処理の過程
Fig. 4 Process of the line thinning.

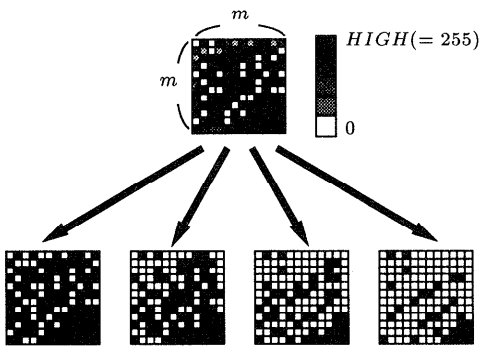



図5 多値カーソルから2値カーソル生成
Fig. 5 Transforming from a multi-tone cursor to half-tone cursors.

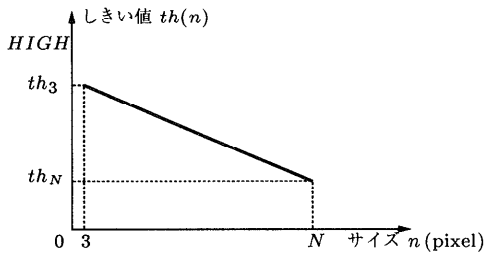



図6 しきい値関数
Fig. 6 Threshold function.

$N$ に近付き拡大サイズになると，しきい値 $th(n)$ は減少することから，生成される2値カーソルの各成分に黒（1）が増え，筆触カーソルのサイズが縮小すると，生成される2値カーソルの各成分には白（0）が増加する．すなわち，縮小サイズの筆触カーソルを配置した箇所には，疑似的な掠れ表現を生じるが，拡大サイズの筆触カーソルを配置した箇所については，縮小サイズを配置した箇所に比べ，掠れ表現が比較的生じないことになる．

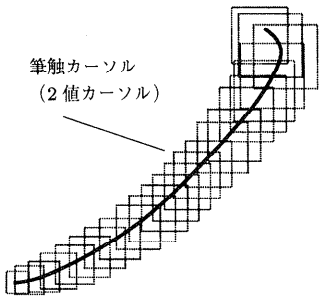



図7 筆触カーソルの配置
Fig. 7 Placing brush-touch cursors.

### 3.3 筆触カーソルの配置と描画

図7に示すように，各骨格構成画素を中心に2値カーソルを順次配置し，描画することで，掠れ・滲み毛筆文字は形成される．

いま，座標 $(X_0, Y_0)$ に位置する任意の骨格構成画素を中心に $n \times n$ の2値カーソルを配置するとする．このとき，2値カーソルの $(i,j)$ 成分である $d_n(i,j)$ は1または0の値をとるが，$d_n(i,j) = 1$ の場合についてのみ座標 $(x_i, y_j)$ に黒点を描画し，$d_n(i,j) = 0$ の場合については何も描画しない（図8）．座標 $(x_i, y_j)$ は次式で表される．

$$x_i = X_0 - \frac{n-1}{2} + i \tag{4}$$

$$y_j = Y_0 - \frac{n-1}{2} + j \tag{5}$$

### 3.4 滲み処理

滲み処理を実行しない場合は前述のように，$n \times n$ サイズの2値カーソルの $(i,j)$ 成分が $d_n(i,j) = 1$ であるとき，黒点を描画する．これに対して，滲み処理を実行する場合は黒点を描画するのではなく，半径 $r$ の黒塗りの円を描画し，滲みを疑似的に表現する（図9）．このとき，黒塗りの円の半径 $r$ については一様乱数

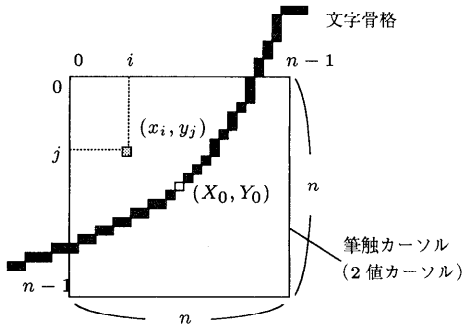



**図8** 筆触カーソルの描画
Fig. 8 Drawing a brush-touch cursor.

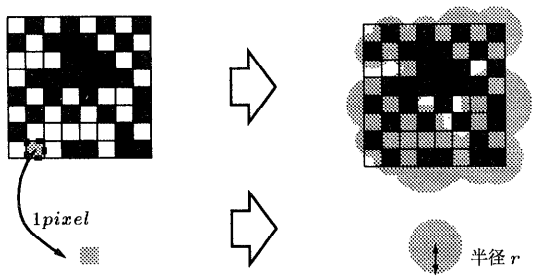



**図9** 滲み表現
Fig. 9 Blurred-look representation.

によって決定するが，その範囲は次式で示す範囲に限定する．ただし，式中の $r_{\max}(\geq 0)$，$HIGH(=255)$ は定数であり，$th(n)$ は多値カーソルから2値カーソルを生成する際のしきい値である．

$$0 < r \leq r_{\max}\frac{HIGH - th(n)}{HIGH} + 1 \qquad (6)$$

しきい値 $th(n)$ は0から $HIGH(=255)$ の範囲で値をとるが，カーソルサイズが小さくなるにしたがって，$th(n)$ が $HIGH$ に近づくので，半径 $r$ の選択範囲は狭く抑えられ，滲み表現があまり現れないことになる．逆に，カーソルサイズが大きくなると，半径 $r$ がとることのできる選択範囲が広がり，滲み表現が際だつ．さらに，$r_{\max}$ を変化させることによって滲みの度合を変化させることができる．

## 4. 掠れ・滲み毛筆文字の作成

### 4.1 掠れおよび滲みの度合の変化

本システムはパラメータの設定で，掠れおよび滲みを様々に変化させることが可能である．掠れの変化については，図6に示すしきい値関数の $th_N$，$th_3$ を自由に設定することで掠れの変化をつけることができる．また，滲みについては，$r_{\max}$ の設定によって滲みの変化をつける．なお，$th_N$ と $th_3$ の2変数について式(2)の関係があるほかは，$th_N$，$th_3$ および $r_{\max}$ はそれぞれ独立変数であり，自由な値を設定可能である．したがって，様々な掠れと滲みの混在した変化のある毛筆文字を作成することができる．

本システムによって作成した掠れ・滲み毛筆ビットマップフォントを**図11**から**図14**に例示する．なお，作成に用いた入力毛筆フォントを**図10**に示す．

### 4.2 処理速度について

**図15**は本システムの処理時間を示すグラフである．入力毛筆フォントには，図10に示す4つの毛筆フォントを用い，図11を作成した場合と同じパラメータ設定で実験を行った．横軸は出力ビットマップフォントのサイズ（$k \times k$）を表し，縦軸の処理時間については図10の4つの入力毛筆フォントを処理した場合の平均値である．図15から分かるように，本システムの処理時間は，出力する掠れ・滲み毛筆フォントのサイズにかなり依存する．これは本システムのアルゴリズムの性質上予想できることである．

まず，文字骨格獲得に用いるHilditchの細線化法については，入力ビットマップフォントのサイズが拡大すれば，細線化処理回数が増加し，処理量が増加するため，処理に時間がかかるのは明らかである．また，入力ビットマップフォントのサイズが $n$ 倍になれば，それを細線化して得た文字骨格の骨格構成画素の総数もほぼ $n$ 倍となり，配置する2値カーソル数が増大し，処理量が増加する．さらに，各骨格構成画素を中心に配置する2値カーソルのサイズについてもそれぞれ約 $n$ 倍となるが，筆触カーソルの面積でいえば $n^2$ 倍である．したがって，$n$ 倍になった文字骨格上に，面積比 $n^2$ 倍の2値カーソルを配置することになり，配置および描画の処理量は約 $n^3$ 倍程度に増加する．このように，細線化処理および2値カーソルの配置，描画すべてにおいて処理量が増加するため，図15に示すように，入力ビットマップフォントサイズの拡大にしたがって，処理時間が増加する．なお，図15に示す処理時間については，Sun Sparc Station 5の互換機であるJS5/85 MHzを使用し，評価を行った．

## 5. おわりに

本稿では，毛筆フォントを入力し，それに掠れ・滲み処理を施し出力するシステムについて述べた．入力する毛筆フォントは既存のアウトラインフォントはもちろん，スキャナ等によって取り込んだ毛筆文字であっても容易なフォーマット変換によって掠れ・滲み処理が可能である．また，入力する毛筆フォントのサイズや形状も掠れ・滲み処理には関係しない．

**図10** 入力毛筆フォント
Fig. 10 Input calligraphy fonts.

**図11** 出力例1 ($th_N = 128$, $th_3 = 254$, $r_{\max} = 0$)
Fig. 11 Output examples 1.

**図12** 出力例2 ($th_N = 169$, $th_3 = 254$, $r_{\max} = 0$)
Fig. 12 Output examples 2.

これまで，掠れまたは滲みのある毛筆フォントを葉書印刷ソフト等で利用しようとする場合，新たなフォントデータベースを用意しなければならなかったが，本システムを利用することにより，既存の毛筆フォントから疑似的な掠れおよび滲みのある毛筆フォントを生成可能である．したがって，データ量を増やす必要がなく，従来と同じデータベースを用意するのみで済む．

本システムでは，$th_3$，$th_N$ の2つのパラメータによって掠れ表現に変化をつけることができ，また $r_{\max}$ によって滲み表現に変化をつけることができ，それによって多彩な掠れ・滲み毛筆フォントを作成することが可能である．掠れ表現については，図6に示した線形に変化するしきい値関数によって2値カーソルを生成しているが，しきい値関数の形状を線形以外の形状に変えることで，さらに多様な掠れ表現の変化をつけることができると考えられる．同様に，多値カーソルデータの各成分の持つ値を変化させることで，さらなる掠れ表現の変化も可能であると考えられる．

今後の課題としてあげられるものには処理速度の向上がある．処理速度については，図15に示すような結果が出ており，出力フォントサイズが大きい場合のパフォーマンスは実用的とは言い難い．最近のプリンタでは1200 dpiの性能を持つ機種も登場してきていることからも，かなり大きなサイズのフォントをプリントアウトする需要も出てくると考えられ，処理速度


図 13 出力例 3 ($th_N = 169$, $th_3 = 254$, $r_{\max} = 32$)
Fig. 13 Output examples 3.


図 14 出力例 4 ($th_N = 169$, $th_3 = 254$, $r_{\max} = 48$)
Fig. 14 Output examples 4.

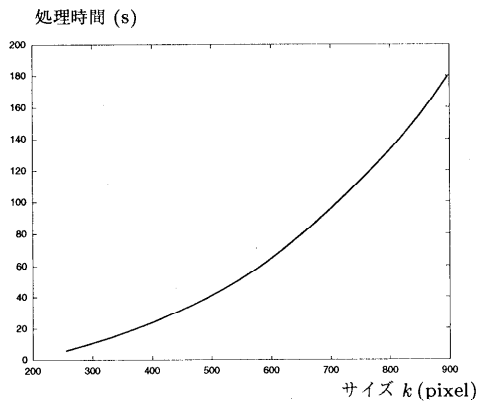



図 15 処理時間
Fig. 15 Process time.

の問題は早急に改善する必要がある課題である.

なお, この方式により作成されたプログラムの一部は株式会社リコーの製品「筆百彩」に使用されている. また, 本方式については以下に示すとおり, 現在特許出願中である.

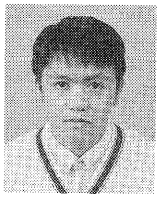
出願番号: 特願平 8 - 21763
出願番号: 特願平 8 - 257559

## 参 考 文 献

1) Hilditch, C.: Linear Skeleton from Square Cupboards, *Machine Intelligence*, Vol.6, pp. 403–420 (1969).
2) Mano, J., Nakamura, T., Seki, H. and Itoh, H.: A Scratched-look Expression of Calligraphy Characters Using Renormalization Group, *International Fuzzy System and Intelligent Control Conference*, pp.93–102 (1996).
3) 真野淳治, 中村剛士, 世木博久, 伊藤英則: 毛筆書体におけるくりこみ群を用いたかすれ・にじみ表現, 情報処理学会論文誌, Vol.38, No.4, pp.806–814 (1997).
4) Nakamura, T., Itoh, H., Seki, H. and Law, T.: A Writing System for Brush Characters Using Neural Recognition and Fuzzy Interpretation, *International Joint Conference on Neural Networks*, pp.2901–2904 (1993).
5) Nakamura, T., Kuroda, T., Itoh, H. and Seki, H.: Λ Calligraphy System Based on Analyzing User Writing Speed, *3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pp.189–190 (1994).
6) Nakamura, T., Kuroda, T., Itoh, H. and Seki, H.: Fuzzy-based Writing System for Acquiring Good Writing Skill of Brush Characters Based on the Analysis of Writing Speed, *3rd Pacific Rim International Conference on Artificial Intelligence*, Vol.2, pp.822–827 (1994).
7) 中村剛士, 黒田 崇, 伊藤英則, 世木博久: 筆記

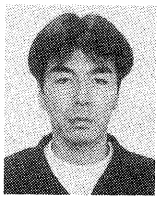
速度のファジィ評価方法を導入した毛筆文字生成システムについて，日本ファジィ学会誌，Vol.7, No.2, pp.371–379 (1995).
8) 中村剛士，松下政親，世木博久，伊藤英則：フラクタルを用いた毛筆文字のかすれ表現について，日本ファジィ学会誌，Vol.8, No.3, pp.558–566 (1996).
9) Nakamura, T., Nozaki, K., Seki, H. and Itoh, H.: A Fuzzy-based Calligraphy System Using Fractals, *Third European Congress on Intelligent Techniques and Soft Computing*, pp.1440–1444 (1995).
10) 若月敏明（編）：楷書百科，芸術新聞社 (1988).
11) 山本雅弘，山崎敏範，井口征士：書写技能知識を組み込んだCAIシステム，信学論，Vol.J72-D-II, No.9, pp.1493–1500 (1989).
12) 山崎敏範，山本雅弘，井口征士：筆記速度分析を導入したCAIシステム，信学論，Vol.J70-D, No.11, pp.2071–2076 (1987).
13) 張 憲栄，真田英彦，手塚慶一：漢字楷書毛筆字体の計算機による生成，信学論，Vol.J67-D, No.5, pp.599–606 (1984).
14) 張 憲栄，真田英彦，手塚慶一：階層分解合成法による隷書体漢字の生成，信学論，Vol.J68-D, No.8, pp.1489–1496 (1985).
15) 張 憲栄，真田英彦，手塚慶一：計算機による様々な書体生成に適合する筆触パターンの提案，信学論，Vol.J69-D, No.6, pp.885–892 (1986).

(平成8年10月2日受付)
(平成9年 3 月7日採録)



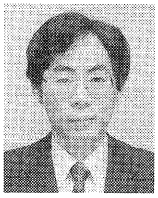
**中村 剛士**（学生会員）

1993年名古屋工業大学工学部電気情報工学科卒業．1995年同大学院博士前期課程修了．現在同大学院博士後期課程在学中．ニューラルネットワーク，ファジィ推論等に興味を持つ．ファジィ学会会員．



**真野 淳治**（学生会員）

1996年名古屋工業大学知能情報システム学科卒業．現在，同大学院博士前期課程在学中．毛筆文字生成システムの研究開発に従事．



**世木 博久**（正会員）

1979年東京大学工学部計数工学科卒業．1981年同大学院工学系研究科修士課程修了．同年4月より三菱電機（株）中央研究所に勤務．1985年～1989年（財）新世代コンピュータ技術開発機構に出向．1992年4月より名古屋工業大学工学部知能情報システム学科助教授．工学博士．論理プログラミング，演繹データベース等に興味を持つ．電子情報通信学会，人工知能学会，ACM，IEEE Computer Society 各会員．

**伊藤 英則**（正会員）

1974年名古屋大学大学院工学研究科博士課程電気電子専攻満了．工学博士号取得．1974年日本電信電話公社横須賀研究所勤務．1985年（財）新世代コンピュータ技術開発機構出向．1989年名古屋工業大学教授．現在知能情報システム学科所属．この間，数理言語理論，計算機ネットワーク通信，OS，知識ベースシステムなどの研究開発に従事．電子情報通信学会，人工知能学会，形の科学学会，ファジィ学会各会員．