文書処理とヒューマンインタフェース 21－5
（１９８８．１１．７）

# プレビュー用フォント作成の試み

大野 義夫
慶応義塾大学 情報科学研究所

文書清書システムにおけるプレビューアの重要性はいうまでもない．プレビューアで使用する文字には，少ないドット数でも読めること，同じ書体に属する文字に対しては，一様な印象を与えること，印刷時に使用する文字フォントのイメージを保存していること，などの性質が要求される．本研究では，漢字のもつ図形的な性質を利用して，こうした条件を満たすようなドット文字フォントを輪郭線フォントから自動的に作成するアルゴリズムを提案する．

このアルゴリズムは，輪郭線フォントの中から，水平線や垂直線を表わしている部分を抽出し，残りの部分をドットパターンに変換した後で，先に抽出した水平線や垂直線を一定のドット数で記入する，というものである．

実際の例を示すとともに，得られた結果の評価を行なう．

# An Algorithm to Generate Chracter Fonts for Previewers

Yoshio OHNO
Institute of Information Science, Keio University
4-1-1 Hiyoshi, Kohoku-ku, Yokohama-shi, 223 Japan

The previewer is an important component of the document production systems. The character fonts used in the previewer must satisfy such conditions as (1) they should be readable even when the number of dots is small, (2) they should give uniform impression when they belongs to a same font family, and (3) they should preseve the impression of the original (printer) font.

The shape of Kanji characters differs from Roman letters in that (1) Kanji shape is much more complex than Roman letters, and (2) most Kanjis have many horizontal and vertrical strokes.

This paper proposes an algorithm for the automatic generation of such character fonts from the contour font data using the above geometric characteristics of Kanji characters. This algorithm extracts horizontal and vertical strokes, converts the remaining strokes into dots, then write the horizontal and the vertical strokes in the uniform widths.

We show some examples, then evaluate the results.

## 1　はじめに

バッチ型の文書清書システムにとって，最終文書の印刷体裁を実際に印刷する前に画面で確認するプレビューアは，文書作成の効率を高める上で欠かすことができない．またWYSIWYG方式のシステムの場合には，プレビューアに相当するものがそもそもシステムの不可分の要素となっている．

文書の中心的な構成要素が文章である以上，プレビューアは多くの文字を表示しなければならない．ところが現在の表示装置の解像度を考えると，たとえばA4版の文書全体を一度に表示しようとしたとき，1文字あたりのドット数は $10 \times 10$ からせいぜい $20 \times 20$ ドット程度にしかならない．

プレビューアで用いる文字フォントは，読みやすくしかも，印刷に用いる文字のイメージを保存している必要がある．さらに，清書システムで使用可能なすべての書体，すべての文字サイズを画面上で区別できるように表示しなければならない．

本研究では，清書システムに使用する輪郭フォントから，上の条件を満たすドットフォントを自動的に作成することを試みており，これまでの実験結果を報告したい．

## 2　表示用フォントの見やすさ

漢字の字体と欧文文字の字体を幾何学的な図形として見たとき，その違いの1つは，漢字の中に斜め線や曲線の他，多くの水平線・垂直線が含まれていることである．しかも，それらの水平線や垂直線は，水平線は水平線同士，垂直線は垂直線同士で，それぞれがほとんど一定の太さを持っている．これが同じ書体に属する漢字が一様に見えるための大きな要素となっているものと考えられる．したがって，そうした漢字をドット書体で表示したとき，その書体がきれいに見えるかためには，水平・垂直線をそれぞれ同じ太さで表示することが必要となる．

一方，明朝体とゴチック体の違い，同じ明朝体の中でも，細明朝・中明朝・太明朝の違い，さらには，同じ中明朝でも，秀英書体・写研書体の違いなど，漢字フォントの個性は主として，水平線がどのような太さで描かれるか，垂直線がどのような太さで描かれるかということと，斜線や曲線のストロークがどのような形をしているのかによって生ずるものと考えられる (図1)．

輪郭情報を単にスキャン・コンバートしただけでは，丸めの誤差のため，幾何学的には同じ太さの線であっても，必ずしも同じ数のドットに変換されないことはよく知られている．また，印刷用のフォントをドット形式で保持している清書システムの場合には，それを適宜間引いたものを表示用フォントとして使用することがあるが，その場合には何本かの水平線・垂直線が完全に失われてしまうことも多い (図2)．欧文文字の場合には，あとで述べる理由により，こうしたことがそれほど深刻な問題とはならない．

## 3　表示用フォントの作成手順

以上の考察から，始めに輪郭線フォントの中から水平線・垂直線を分離し，残りの部分をスキャン・コンバートし (必要があればそこからドットを間引いてから)，さらに一定の太さの水平線・垂直線を書き加えることを考えた．これらそれぞれの手順をまとめて比較したものが図3である．

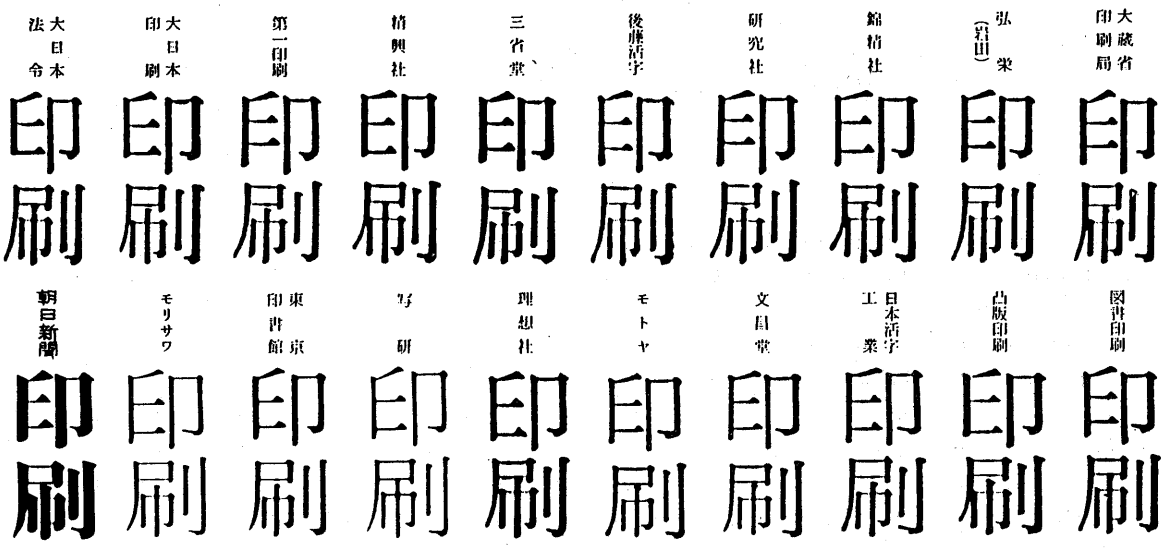
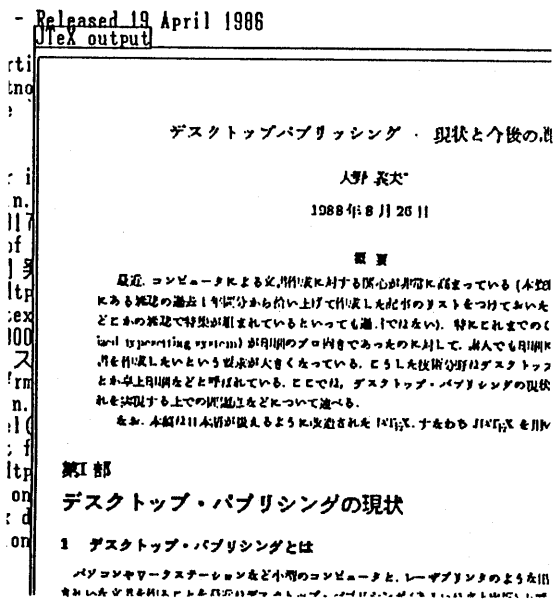



図1. 漢字フォントの個性. [1,2] より.




GMWウィンドウシステムで TEX の文書をプレビューしているところ．「今後」の「今」や「日本語」の「日」の字などに注意．

図 2. プレビュー画面の例.

この図の中で，手順 (A) が単純に幾何学的なスキャン・コンバージョンを行なう場合を表わす．本報告に示す例では，文字フォントが少しでもかかっているピクセルはすべて黒としてスキャン・コンバージョンを行なった．このため，生成するドット・フォントのドット数が少なくなるほど，オリジナルの輪郭線フォントと比べて，全体的に黒っぽくなる傾向がある．

手順 (B) はその黒さを補正する場合である．はじめにやや多いドット数でスキャン・コンバートする．この時点では，濃淡はかなり正確に保存されているはずである．次にそのビット・マップから縦横一定間隔でドットのサンプリングを行なって最終的なドット・フォントを得る．したがってこの方法は (A) に比べると，輪郭線フォントの黒さをより正確に保存する．しかし欠点として，ドットのサンプリングを行なうとき，文字の１ストロークがすっぽり脱落してしまう場合がしばしば生じる．明朝体であれば，水平のストロークがなくなっても，ウロコの部分が残って判読できる場合もあるが，細いゴチック体では極端な場合には文字全体がなくなってしまうことも考えられる．

たとえば TEX のプレビューアのほとんどは，この手順によってリアルタイムで表示用フォントを作成しているようである (図 2 参照)．

(C) が本報告で提案する手順である．これについては次節と次々節でやや詳しく述べる．

## 4 漢字の字形とドット・フォント

はじめに述べたように，欧文文字を印刷したブロックと漢字がまとまっている領域とは一見してかなり印象が異なる．その理由としては

1. 漢字の方が字形が複雑である
2. このため，１つ１つのストロークは漢字の方が細い
3. 漢字には水平線・垂直線のストロークが非常に多い

などが考えられる．また，個々の漢字の識別には文脈が大きな貢献をしているであろうが，その他には，

1. 水平線・垂直線ストロークそれぞれの大体の本数
2. 水平線・垂直線以外のストロークの形状
3. ストローク相互の相対的な位置関係

などに頼っているものと思われる．このため，こうした特徴を保存し，しかも水平線ストローク，垂直線ストローク

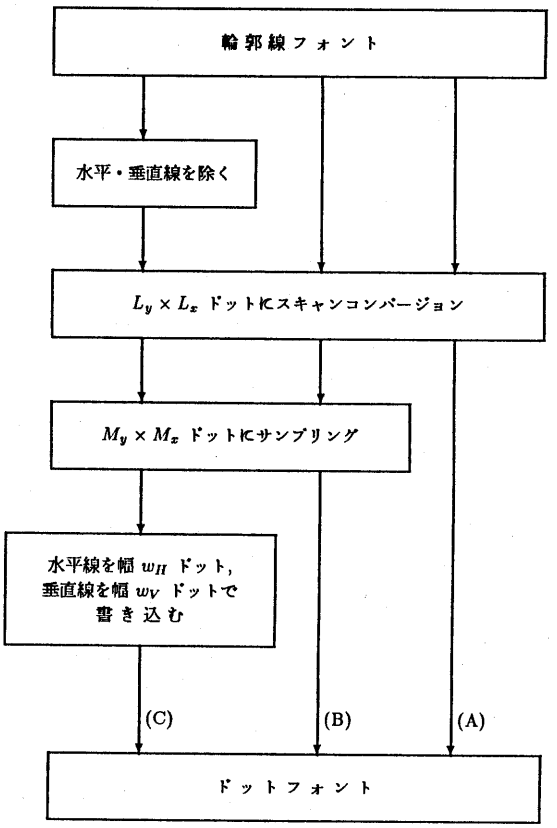



図 3. ドットフォントを作る手順

それぞれを一定のドット数で表現したドット・フォントを作成すれば，見た目にきれいで，一様な印象を与え，読みやすく，しかもオリジナルの輪郭線フォントがもつ印象を可能な限り保存したドット・フォントを得ることが期待できる．漢字は字数が多いので，このようなドット・フォントは輪郭線情報から自動的に作成できることが望ましい．

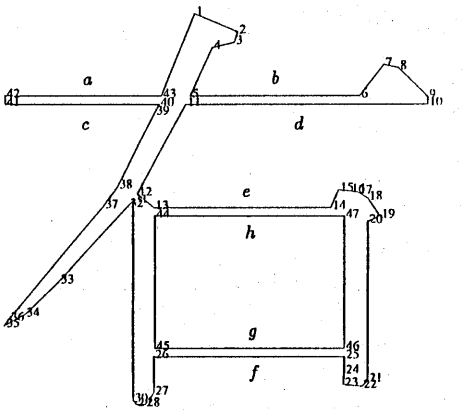
## 5 ドット・フォントの作成手順

本研究では，折れ線による輪郭線データを材料として実験を行なった．このデータでは，字の外形部分では右回り，穴の形状は左回りになるような順序で頂点の位置データが与えられている．言いかえると，与えられた順序にしたがって輪郭線上を進むと，常に右側が黒領域となるようになっている．また，座標系は左上スミを原点とし，$x$ 軸が右向き，$y$ 軸が下向きであったが，本報告では左下スミを原点とし，$y$ 軸を上向きとして説明する．輪郭線データの例を図 4(a) に示す．

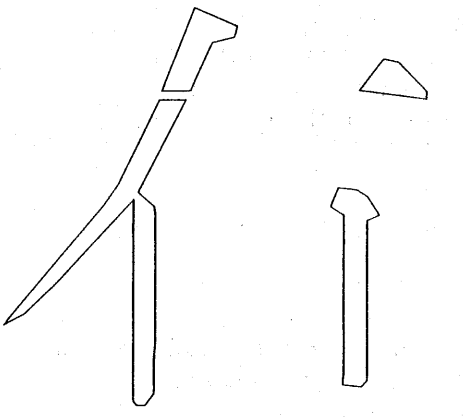
始めに，与えられた輪郭線の情報から水平線ストローク，垂直線ストロークを表現している部分を抽出する．水平線ストロークの場合，この作業は次の手順で行なった．垂直線ストロークの場合も同様である．

1. 輪郭線を構成する線分を順に調べ，水平なものを抽出する[1]．このとき，右に向かう線分と左に向かう線分とを区別する．たとえば図 4(a) では，辺 $a$, $b$, $e$, $g$ が右に向う水平線，辺 $c$, $d$, $f$, $h$ が左に向う水平線として認識される．
2. 右に向かう線分を $y$ 座標によってソートし，同じ $y$ 座標をもつものをグループとしてまとめる．例では，辺 $a$ と $b$ が 1 グループになる．その他の辺 $e$, $g$ はそれぞれが 1 本ずつでグループを構成する．
3. それぞれのグループの中の線分を $x$ 座標に従ってソートする．辺 $a$ と $b$ はこの順序にソートされる．
4. 1 グループ内の線分を順に調べ，連続する線分の間がすべて黒領域である場合には，それらの線分を合併させて 1 本にまとめる．この例では辺 $a$ と $b$ が合併させられる．
5. 左に向かう線分についても同様の作業を行なう．この結果，辺 $c$ と $d$ が合併させられる．
6. 右に向かう線分とそれよりやや下にあって左に向かう線分とを組合わせて，それらが 1 本の水平線ストロークの上縁・下縁を形成する線分とみなしてよいかどうかを判定する．この例では，合併辺 $a$, $b$ が合併辺 $c$, $d$ とペアになり，他に辺 $e$ と $h$, 辺 $g$ と $f$ もそれぞれペアを構成する．

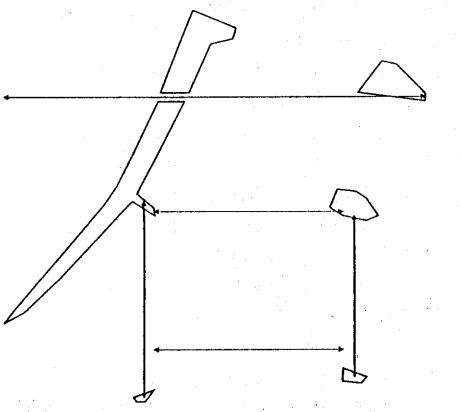
[1]水平であるかどうかの判定に適当な閾値を定めていることは言うまでもない　以降の手順についても同様である．閾値によっては，明朝体とゴチック体，水平線ストロークと垂直線ストロークで別々の値を設定したものもある．




(a)



(b)



(c)

図 4. 水平線・垂直線ストロークの分離

7. 得られた水平線ストロークを表わす台形を輪郭線データから除去する (図 4(b)).

8. 垂直線ストロークに対しても同様の作業を行なう (図 4(c)).

このようにして水平線・垂直線以外の部分が残るので, これをやや多いドット数でスキャン・コンバートする. さらにそのドット・パターンから縦方向, 横方向それぞれ一定間隔でサンプリングして最終的に必要なドット数にした後, 上で抽出した水平線ストローク, 垂直線ストロークをそれぞれ一定のドット数の線として加筆し (水平線の幅を $w_H$ ドット, 垂直線の幅を $w_V$ ドットとする), 最終的なドット・フォントとする. この手順を図 3 では (C) として示してある.

この方法で, 水平線・垂直線以外のストロークをはじめに余分のドット数でスキャン・コンバートするのは, 前にも述べたように, 全体的な黒さを保存するためである. サンプリングによってこうしたストロークがぬけてしまうことも考えられる. しかし明朝体の漢字では斜めのストロークが太く作図されており, また図 5 からわかるように, 45° の線では, 水平線や垂直線とくらべると, サンプリングを完全にかいくぐってしまう確率が $1/\sqrt{2}$ に減少するので, 水平線の場合ほど深刻な問題とはならない.

## 6 例

それぞれの方法によって得られたドット・フォントを例示する.

図 6 がオリジナルの輪郭線フォントの一部である. これをもとにして, 単純なスキャンコンバージョン (手順 (A)) によりドット・フォントにしたものを図 7~11 に示す.

図 12, 13 はいずれも手順 (B) によるドット・フォントで, はじめに単純なスキャン・コンバージョンによって $100 \times 100$ のドット・フォントを作成したのち, サンプリングによってそれぞれの大きさのドット・フォントとしたものである.

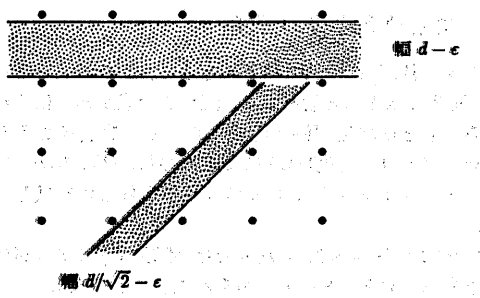



図 5. 一定間隔のサンプリングをかいくぐる確率 (● がサンプリングされる点. サンプリングの間隔を $d$ とする)

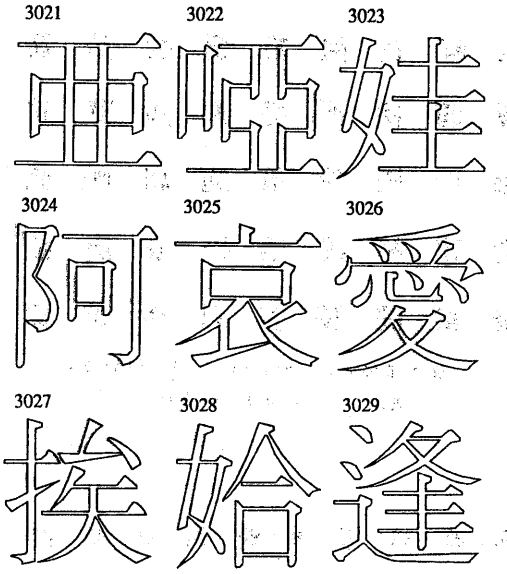



図 6. オリジナルの輪郭線フォントの例

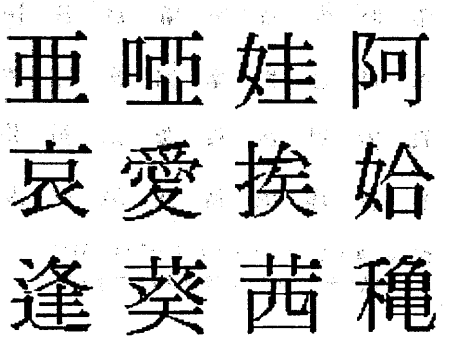


図 7. 単純なスキャン・コンバージョン (手順 (A)) で作成した $L_x = L_y = 45$ ドットのフォント

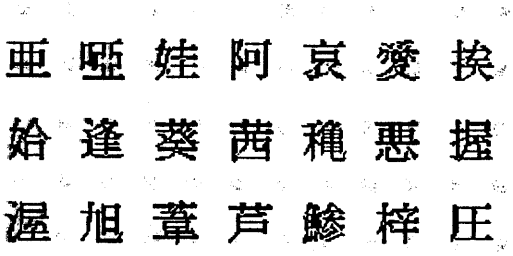


図 8. 単純なスキャン・コンバージョン (A) で作成した $L_x = L_y = 24$ ドットのフォント

図 9. 単純なスキャン・コンバージョン (A) で作成した $L_x = L_y = 16$ ドットのフォント

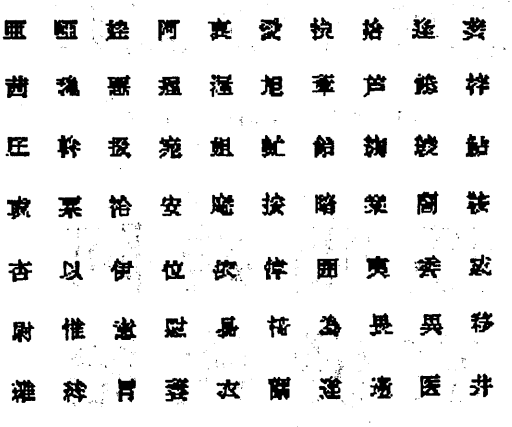


図 10. 単純なスキャン・コンバージョン (A) で作成した $L_x = L_y = 12$ ドットのフォント

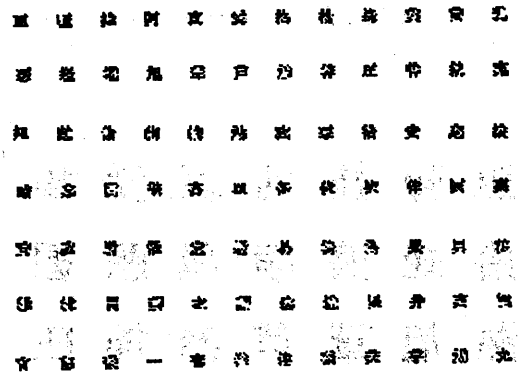


図 11. 単純なスキャン・コンバージョン (A) で作成した $L_x = L_y = 8$ ドットのフォント

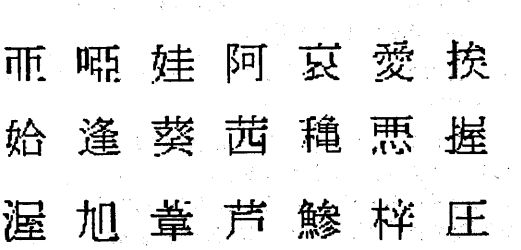


図 12. 手順 (B) で $L_x = L_y = 100$ ドットから作成した $M_x = M_y = 24$ ドットのフォント

図 13. (B) で $L_x = L_y = 100$ ドットから作成した $M_x = M_y = 12$ ドットのフォント

図 14 は図 6 から本報告の手順によって水平線・垂直線ストロークを分離した結果であり，これをもとにして作成したドット・フォントを図 15〜20 に示す (手順 (C)).

## 7 まとめと今後の課題

ドット数が 10 × 10 以下になると，どの方法を使っても，漢字はほとんど読めない．読める仮名があればそれを読み，あとは前後関係や記憶にたよって判読せざるを得ない．それでも手順 (C) によるものは，他の方法で作ったドット・フォントにくらべると漢字らしさを残している．

10×10 から 30×30 あたりでは，手順 (C) の効果がはっきりあらわれる．ドット数がそれ以上に多くなると，手順 (A) で十分である．いずれの場合も，手順 (B) ではよい結果が得られない．

ドット数が少ないときには，水平線や垂直線を最後に書き込むときに，どちらも 1 ドットずつとなり，太さの区

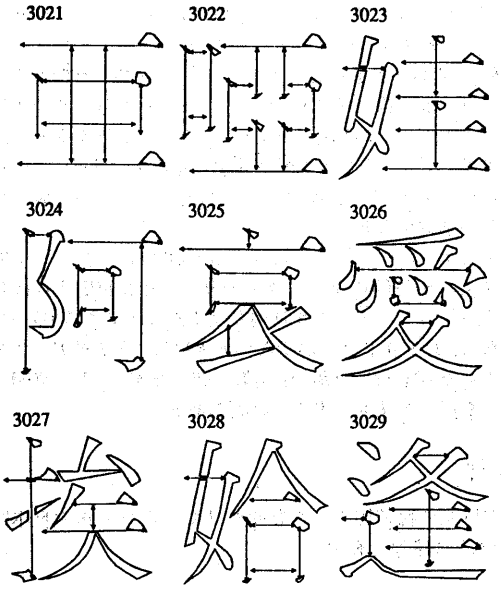



図 14. 図 6 の輪郭線から水平線・垂直線を分離した結果

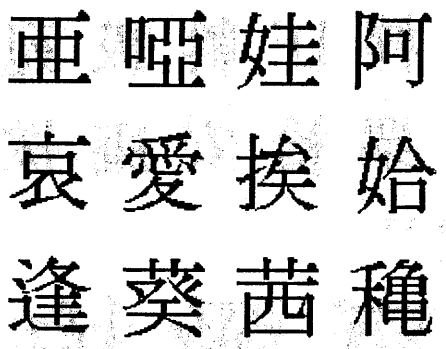


図 15. 手順 (C) で $L_x = L_y = M_x = M_y = 45$, $w_H = 2$, $w_V = 3$ として得られたドット・フォント

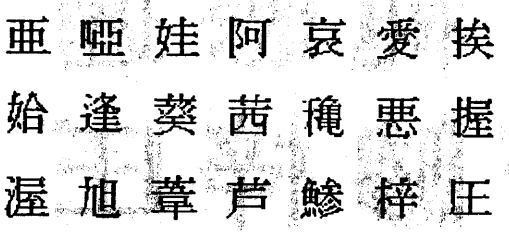


図 16. (C) で $L_x = L_y = 50$, $M_x = M_y = 24$, $w_H = 1$, $w_V = 2$ として得られたドット・フォント

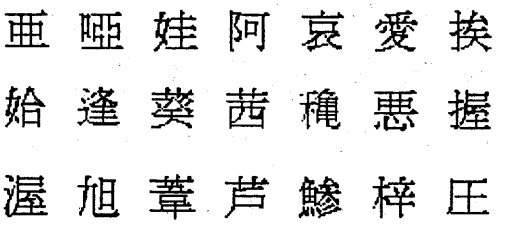


図 17. (C) で $L_x = L_y = 50$, $M_x = M_y = 24$, $w_H = w_V = 1$ として得られたドット・フォント

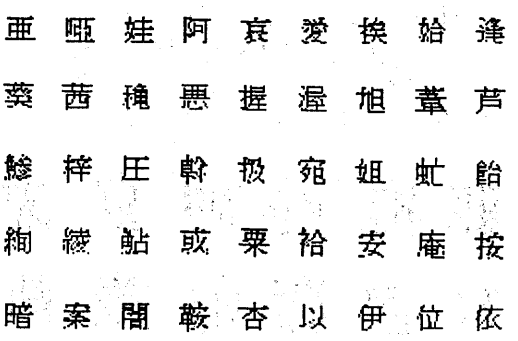


図 18. (C) で $L_x = L_y = 50$, $M_x = M_y = 16$, $w_H = w_V = 1$ として得られたドット・フォント

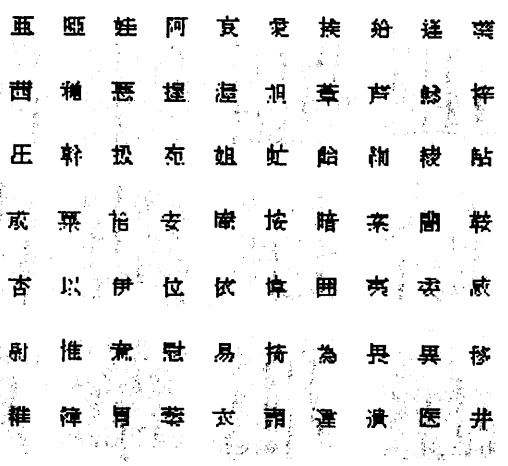


図 19. (C) で $L_x = L_y = 40$, $M_x = M_y = 12$, $w_H = w_V = 1$ として得られたドット・フォント

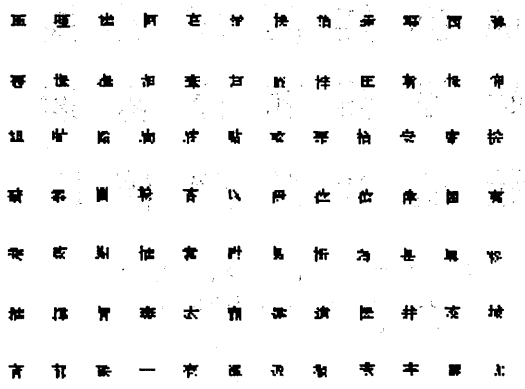


図 20. (C) で $L_x = L_y = 20$, $M_x = M_y = 8$, $w_H = w_V = 1$ として得られたドット・フォント

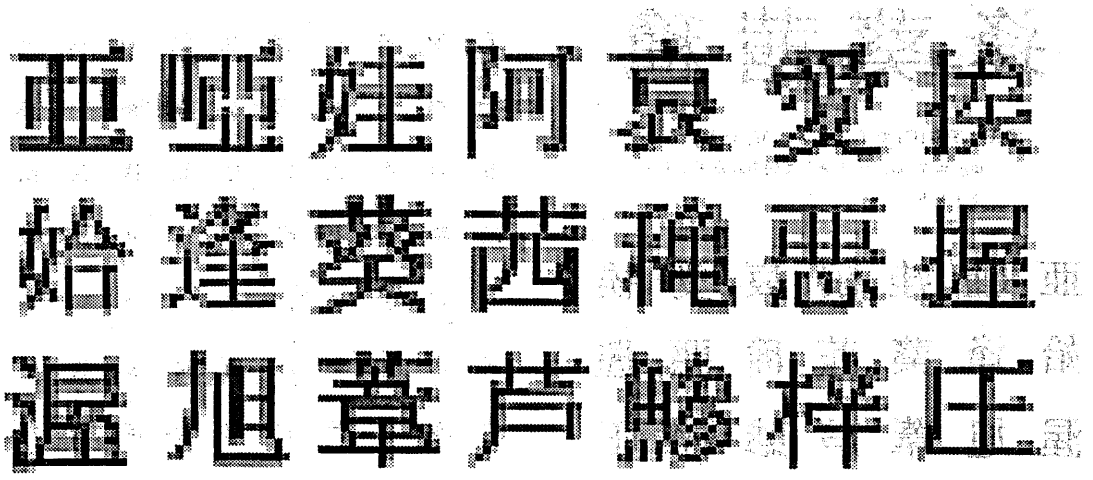
別がつかない．また明朝体の特長であるウロコが非常に小さく，水平線に吸収されてしまって，ゴチック体と区別がつけにくくなることがある．こうした点を改善するためには，水平線・垂直線以外にウロコもあらかじめ抽出することが考えられる．ただし，こうしたことをあまりやりすぎると，オリジナルの字体から離れてしまうので，いずれにしてもドット数の少ない場合については，表示に厳しい制約がある．

ワークステーションのカラー化が進むにつれて，中間調の表示が可能になってきており，これを利用すれば比較的少ないドットでもきれいな文字が表示できる (図 21)．ただし，あまりドットが少ないときには，文字がボケて見えるので (図 22)，今後，中間調を使いながら，かつ鮮明な表示を得る方法も工夫する必要がある．

## 謝辞

本研究に輪郭線データを使わせていただいた大日本印刷 (株) に感謝します．

(a) 16 × 16 ドット．各ドットは 4 階調．



(b) 16 × 16 ドット．各ドットは 16 階調．

図 21. 中間調を利用した漢字フォントの例．

(c) 32 × 32 ドット. 各ドットは 4 階調.

図 21. 中間調を利用した漢字フォントの例 (つづき).

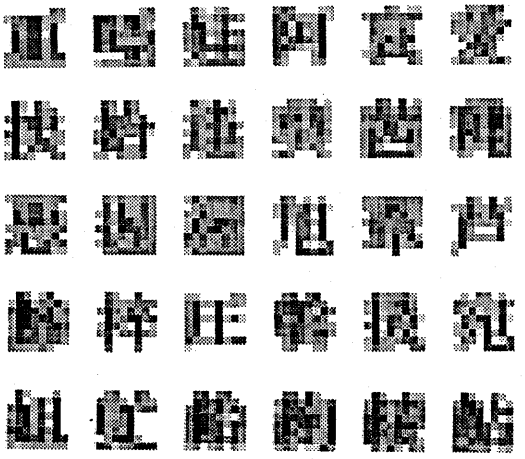


図 22. ドット数が非常に少ない場合の中間調フォント.
8 × 8 ドット. 各ドットは 16 階調.

## 参考文献


[1] 矢作 勝美, “明朝活字 — その歴史と現状,” 平凡社, 1976.

[2] 桑山 弥三郎, “レタリングデザイン,” グラフィック社, 1969.

[3] Peter Karow, “Digital Formats for Typefaces,” URW Verlag, 1987.

[4] Richard Rubinstein, “Digital Typography — An Introduction to Type and Composition for Computer System Design,” Addison-Wesley, 1988.