グラフィクスとCAD 62－2
（1993. 5. 21）

文字フォントのための曲線の自動あてはめ手法

伊藤 浩市　大野 義夫
慶応義塾大学 理工学部

本論文では，スキャナで取り込んだ原字の濃淡画像から，自動的に 3 次 Bézier 曲線によるアウトラインフォントを生成する手法を提案する．提案する手法の手順は，(1) 濃淡画像から輪郭点列を計算し，(2) 角点において輪郭点列を分割する，(3) 分割された点列それぞれについて曲線あてはめを行う，というものである．

ここで曲線あてはめには最小二乗法を用いたが，従来法のように曲線の両端点を固定せず，曲線の両端点も他の輪郭点列と同一条件で曲線あてはめを行い，曲線がすべて計算できた時点でとなりあう曲線同士の交点を計算し端点とするという方法をとった．この方法により従来法の問題点であった，角付近における曲線の不自然さを改善することができた．

An Algorithm for Automatically Fitting Curves for the Outline Font Generation

Koichi ITOH　Yoshio OHNO
Faculty of Science and Technology, Keio University

This paper presents an algorithm that automatically generates outline fonts from a gray-level image of a character obtained by a scanner. Our algorithm begins by extracting edge points from the image and dividing the points into a number of segments at the corner points. The next step is to fit a piecewise cubic Bézier curve to each segment.

To fit cubic Bézier curves to segments, we use the least-squares fitting, without fixing endpoints of the curves. We locate the endpoints by computing the intersection of the adjoining curves. This algorithm improves the shape of the corner of the outline fonts very much.

1 はじめに

近年の小型計算機の処理能力の向上や LBP(Laser Beam Printer) の低価格化, 高解像度化等により DTP (Desktop Publishing) と呼ばれる, 計算機による文書処理システムが非常に身近なものとなってきている.

しかし日本の DTP システムはその表現力において欧米のシステムに比べ大きな遅れをとっている. その大きな原因の一つに日本語フォントの書体の少なさがあげられる.

通常このようなシステムにおいて用いられる文字フォントの表現形式は, 文字の輪郭を関数により記述するアウトラインフォントが用いられており, その関数としては 3 次 Bézier 曲線を用いるのがほぼ標準的となっている.

ところで現在, この Bézier 曲線によるアウトラインフォントは, ほとんどの場合, 紙上原字を取り込み, ディスプレイ上に表示した上で対話的編集を行うことにより作成されている. しかしこのような手段で JIS 第二水準まで含め約 7000 字もあり, また一つ一つの字形も複雑である日本語のフォントを作成するのは容易なことではない. これが日本語の DTP システムの書体の少なさの最大の原因となっている.

そこで, 紙上原字を用いて自動的にアウトラインフォントを生成する技術の開発が必要となる. 特に, 紙上原字をスキャナ等で読みとった濃淡画像に対し, 画像中の原字の輪郭線に自動的に 3 次 Bézier 曲線をあてはめるアルゴリズムの開発が重要である.

このような要求に対し, これまでにいくつかの研究がなされているが [4][8][10], いずれも, 原字への忠実性, あるいは生成される曲線数などの点において十分満足のいくものではない.

本稿では, スキャナで取り込まれた原字の濃淡画像をもとに 3 次 Bézier 曲線によるアウトラインフォントを自動的に生成する方法について報告する.

2 本研究の手法

まず始めに理想的なアウトラインフォントの条件について考えてみると, (1) 原パターンに忠実であること, (2) できるだけ少ない区分曲線数であること, などがあげられる.

そこで本研究では以上 2 点に留意し, 輪郭点列の計算を従来法では考慮されていない濃淡画像中の輪郭付近の中間調を用いることでより厳密に計算し, また曲線あてはめにおいては, 従来法のように端点を固定せずに曲線あてはめを行うことで, 角の周辺における質の劣化を避ける, というアルゴリズムにより自動的にアウトラインフォントを作成する手法を提案する.

ここで作業を大まかな手順に分けると

1. 濃淡画像からの輪郭点列の計算
2. 輪郭点列中の角点の計算および輪郭点列の分割
3. 曲線あてはめ

となる. 以下ではそれぞれについて詳しく説明する.

2.1 濃淡画像からの輪郭点列の計算法

スキャナにより取り込んだ原字の濃淡画像から, 輪郭点列を計算する方法として, 本研究ではノイズの影響を避けることのできない二値化による輪郭点抽出法は用いることはせず, 1991 年に Avrahami らによって発表された輪郭点列計算法 [2] を用いた.

この方法は, 従来法で二値化により捨てられていた画像中の濃淡に着目, 追跡することによって, (1) 画像の 1 pixel 以下の精度で輪郭点列を計算することができ, (2) 濃淡の差の激しい領域を抽出することができる, という特徴をもつ. したがって本研究のようにアウトラインフォントの自動生成に非常に適している方法である.

しかし, 発表された方法では初期追跡点を与えなければならないなど, 全輪郭点を求める手段が自動化されていないため, 本研究では初期追跡点を計算により求め, 全輪郭を自動的に求められるようにアルゴリズムの拡張を行い [1], 輪郭点の計算に用いた.

またこの方法の欠点として, 角のように輪郭線が集中しているような部分で輪郭点の精度が落ちる場合があるが, その対策については次節で述べる.

2.2 輪郭点列中の角点の計算法

次に前節のアルゴリズムにより得られた点列のなかから, 角と見なせる点を計算する方法について説明

する．後にこの点において点列を分割することとなる．

輪郭点列中の角点の計算に，本研究では Davis のアルゴリズム [3] を用いた．簡単にこのアルゴリズムについて説明すると，輪郭点列を $O = \{(x_i, y_i)\}_{i=1}^{n}$ とするとき O の各点において近似曲率 $C^k(i)$ を次式により計算する．

$$
\begin{aligned}
C^k(i) &= \frac{a_{ik} \cdot b_{ik}}{|a_{ik}| \cdot |b_{ik}|}, \\
a_{ik} &= (x_i - x_{i+k}, y_i - y_{i+k}), \\
b_{ik} &= (x_i - x_{i-k}, y_i - y_{i-k}).
\end{aligned}
$$

この $C^k(i)$ が極大となる点を角点とみなすという方法である．ここでこの k の値についてはもとの画像の解像度や図形の形に応じて変える必要がある．また単純に極大点をすべて角点としてしまうと角に対して非常に敏感なアルゴリズムとなってしまうため，本研究ではあるしきい値 T を用いて，$C^k(i) > T$ となる極大点についてのみ角点と見なすこととした．

2.3 曲線あてはめ

2.3.1 本研究における曲線あてはめの概念

従来の曲線あてはめ法 [4][8][10] は，どれをとっても曲線あてはめを行う際に与えられた点列の中から節点 (すなわち曲線の端点) を計算し，その点を固定した上で曲線あてはめを行う方法をとっている．その概念図を図 1 に示す．

この考え方を用いた場合，図 2 (a) のような点列が与えられた場合，節点をどう計算したとしても図 2 (b) のような曲線しか得ることができない．

どのような輪郭点計算法を用いたとしても，角付近においてこのような点列が得られることは決して少なくない．なぜなら，原パターンにおいて角であった点が必ずしも輪郭点として計算されるとは限らないためである．そのため従来法により作成したフォントは角付近の形状が不自然な場合が多い．

また両端点固定による曲線あてはめでは，それだけ曲線としての自由度が低くなってしまう．これは曲線を決定しやすくする反面，曲線に対する拘束が強く，与えられたセグメントを一定以下の誤差で近似しようとする場合，不要に曲線数を増やしてしまう原因ともなる．

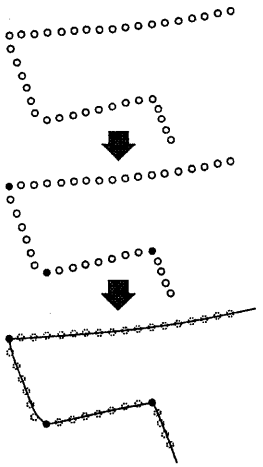


図 1. 従来法の概念図

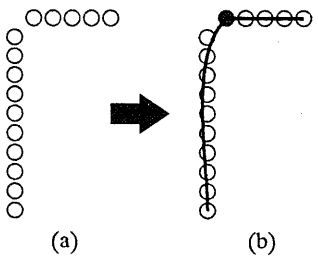


図 2. 角付近における品質の劣化

以上のような理由から，曲線あてはめを行う際に従来法のように曲線の両端点を固定するという方法は避けるべきである．そこで本研究における曲線あてはめの概念図を図 3 に示す．

本研究で提案する曲線あてはめは以下のような手順となる．まずこの図に示すように角であると見なされた点において輪郭点列を一旦分割する．分割された点列のことをセグメントと呼ぶこととする．それぞれのセグメントに対して曲線をあてはめるのであるが，この際セグメントの両端点についても他の点列と同じ条件 (重み) をあたえる．そして全ての曲線が得られた時点で曲線と曲線との交点を計算することにより曲線の端点を決定する．

このアルゴリズムにより曲線あてはめを行うため，2.1 節で得られた点列について 2.2 節の方法で角点

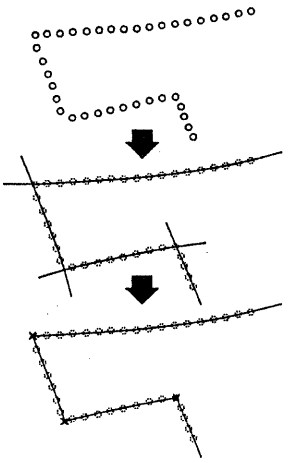


図 3. 本研究における曲線あてはめの概念図

を計算し, 角点ごとに輪郭点列を分割しておく. なおこの際, 角点と認識された輪郭点については精度が他の点に比べ落ちる可能性が高いので, 曲線あてはめのデータとしては用いないこととし, セグメントにはそれら両端点を除いた点列を登録することとした.

2.3.2 セグメントへの曲線のあてはめ法

各セグメントへ曲線をあてはめる方法には最小二乗法を用いた. まず初めに一つのセグメントに対し 1 本の曲線をあてはめる方法について述べ, 次に複数の滑らかに接続された曲線をあてはめる方法について述べる.

ここで与えられたセグメントを $\{P_i = (x_i, y_i)\}_{i=1}^{n}$ と表し, 求める Bézier 曲線の式を計算の便宜上以下のようにべき基底を用いて表す.

$$\begin{cases} B_x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ B_y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \end{cases} \quad (0 \le t \le 1)$$

与えられた点列を用いてこの各係数を決定する. ここで従来法のように P_1 および P_n を固定することはせず, すべての点を同一条件におく.

まず最初に, 各 P_i に対応するパラメータ値 t_i を点列の弦の長さに応じて決定する. 具体的には

$$t_i = \begin{cases} 0 & (i = 0) \\ 1 & (i = n) \\ \dfrac{\text{折れ線 } P_1P_2\cdots P_i \text{ の長さ}}{\text{折れ線 } P_1P_2\cdots P_n \text{ の長さ}} & (\text{それ以外}) \end{cases}$$

という値を用いる (ただしこれはあくまで初期値であり, この値は後に何度も補正を行う).

次に曲線 $B(t)$ 上で P_i に対応する点 $B(t_i)$ と P_i との二乗距離の総和 S を考えると以下のようになる.

$$\begin{aligned} S &= \sum_{i=1}^{n} [B(t_i) \text{ と } P_i \text{ の距離}]^2 \\ &= \sum_{i=1}^{n} (a_x {t_i}^3 + b_x {t_i}^2 + c_x t_i + d_x - x_i)^2 \\ &\quad + \sum_{i=1}^{n} (a_y {t_i}^3 + b_y {t_i}^2 + c_y t_i + d_y - y_i)^2 \end{aligned}$$

この S を最小とするような a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y を求めれば良いのであるから $\partial S/\partial a_x$, $\partial S/\partial b_x$, $\partial S/\partial c_x$, $\partial S/\partial d_x$, $\partial S/\partial a_y$, $\partial S/\partial b_y$, $\partial S/\partial c_y$, $\partial S/\partial d_y$ を計算しそれぞれを 0 とすればよい.

$$\begin{cases} a_x \sum_{i=1}^{n} {t_i}^6 + b_x \sum_{i=1}^{n} {t_i}^5 + c_x \sum_{i=1}^{n} {t_i}^4 + d_x \sum_{i=1}^{n} {t_i}^3 = \sum_{i=1}^{n} x_i {t_i}^3 \\ a_x \sum_{i=1}^{n} {t_i}^5 + b_x \sum_{i=1}^{n} {t_i}^4 + c_x \sum_{i=1}^{n} {t_i}^3 + d_x \sum_{i=1}^{n} {t_i}^2 = \sum_{i=1}^{n} x_i {t_i}^2 \\ a_x \sum_{i=1}^{n} {t_i}^4 + b_x \sum_{i=1}^{n} {t_i}^3 + c_x \sum_{i=1}^{n} {t_i}^2 + d_x \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} x_i t_i \\ a_x \sum_{i=1}^{n} {t_i}^3 + b_x \sum_{i=1}^{n} {t_i}^2 + c_x \sum_{i=1}^{n} t_i + d_x n = \sum_{i=1}^{n} x_i \end{cases}$$

(y についても同様)

図 4. 最小二乗法

その上で式を整理すると図4に示すような2組の4元1次方程式となる．これを解いて a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y を求めれば，S を最小とする曲線を決定することができる．なおこの方程式の解法にはもっとも単純な Gauss 消去法を用いたが，安定して解を求めることができた．

しかしこの方法で求められる曲線は必ずしも最適なものではない．なぜなら，曲線を決定するにあたって用いた t_i の値が必ずしも最適なものではないためである．

そこで再パラメータ化を行う．再パラメータ化とは 1983 年 Plass らによって発表された方法 [5] で，一旦求めた曲線をもとによりよいパラメータ t_i の値を計算する方法である．具体的には次のような手順となる．

まず次式により，各点に対する新しいパラメータの値を計算する．

$$[B(t) - P_i] \cdot B'(t) = 0$$

ここでこの式は t に関する5次方程式となるが，すべての点 P_i においてこの方程式を t_i を初期値として Newton-Raphson 法で解く．(なおすべての t_i が計算できた時点で $t_0 = 0$, $t_n = 1$ となるように t_i の値を正規化しておく．) このようにして得られた t_i をもとに，再び曲線を求めるのである．これを繰り返し行うことにより曲線を点列に収束させることができる．

続いて複数の曲線をあてはめる方法について述べる．まず簡単のため2本の曲線を考える．1本のときと同様，式で表す．

$$\begin{cases} B_x^{(1)}(t) &= a_x^{(1)} t^3 + b_x^{(1)} t^2 + c_x^{(1)} t + d_x^{(1)} \\ B_y^{(1)}(t) &= a_y^{(1)} t^3 + b_y^{(1)} t^2 + c_y^{(1)} t + d_y^{(1)} \end{cases} \quad (0 \le t \le 1)$$

$$\begin{cases} B_x^{(2)}(t) &= a_x^{(2)} t^3 + b_x^{(2)} t^2 + c_x^{(2)} t + d_x^{(2)} \\ B_y^{(2)}(t) &= a_y^{(2)} t^3 + b_y^{(2)} t^2 + c_y^{(2)} t + d_y^{(2)} \end{cases} \quad (0 \le t \le 1)$$

次に2本の曲線の接続条件を考える．ここで接続の際の自由度を上げるため，G^1 連続の条件で考える．まず1本目の曲線の終点と2本目の曲線の始点が共通であることから

$$\begin{cases} B_x^{(1)}(1) &= B_x^{(2)}(0) \\ B_y^{(1)}(1) &= B_y^{(2)}(0) \end{cases}$$

続いて節点において接線方向のみ共通であることから，ある変数 α を用いると次のように表すことができる．

$$\begin{cases} \alpha B_x^{(1)\prime}(1) &= B_x^{(2)\prime}(0) \\ \alpha B_y^{(1)\prime}(1) &= B_y^{(2)\prime}(0) \end{cases}$$

ここで α は2本の曲線の長さの比で表すことができると考え，以下の式で近似することとした．

$$\alpha = \frac{\text{折れ線 } P_m \cdots P_n \text{ の長さ}}{\text{折れ線 } P_1 \cdots P_m \text{ の長さ}}$$

ただし m は点列を分割する点を示す．

以上により未知数は $a_x^{(1)}$, $b_x^{(1)}$, $c_x^{(1)}$, $d_x^{(1)}$, $a_x^{(2)}$, $b_x^{(2)}$, $a_y^{(1)}$, $b_y^{(1)}$, $c_y^{(1)}$, $d_y^{(1)}$, $a_y^{(2)}$, $b_y^{(2)}$ の12個となる．そこで曲線1本のときと同様，二乗誤差の総和 S を計算し，$\partial S/\partial a_x^{(1)}$, $\partial S/\partial b_x^{(1)}$, $\partial S/\partial c_x^{(1)}$, $\partial S/\partial d_x^{(1)}$, $\partial S/\partial a_x^{(2)}$, $\partial S/\partial b_x^{(2)}$, $\partial S/\partial a_y^{(1)}$, $\partial S/\partial b_y^{(1)}$, $\partial S/\partial c_y^{(1)}$, $\partial S/\partial d_y^{(1)}$, $\partial S/\partial a_y^{(2)}$, $\partial S/\partial b_y^{(2)}$ をそれぞれ 0 とし整理すると，2組の6元1次方程式となる．これを Gauss 消去法により解き，再パラメータ化を行なうことで2本の曲線を滑らかに接続したままあてはめることができる．

3本以上の曲線に対しても，以上のアルゴリズムの延長上で対応できる．そこで1つのセグメントに対する曲線あてはめをまとめると以下のようになる．

1. 始めに与えられたセグメントに対し弦長に応じて決定したパラメータを用いて曲線あてはめを行ない，曲線とセグメントとの相違度がある一定以下であれば曲線を出力してアルゴリズムを終了する．そうでなければ2へ．
2. 再パラメータ化を行ない再び曲線をあてはめる．得られた曲線の相違度がある一定以下であれば，曲線を出力して終了する．
3. 2を繰り返し，曲線を収束させる．十分な回数繰り返したところでまだ収束しないようであれば，曲線を分割し複数の曲線に対して1からの操作を行なう．

なお曲線とセグメントの相違度については，セグメント中の点と曲線との二乗誤差のうち最大の値をその曲線の相違度とした．また収束がうまくいかなかったときの曲線の分割位置についてであるが，これには対象とするセグメント中の点と曲線との距離を計算し，これが最大となった点において曲線を分割することとした．

2.3.3 曲線の端点決定

本研究では 2 節はじめで述べたように端点を固定せずに曲線あてはめを行なうため, すべての曲線が得られた時点で曲線同士の交点を求め, 端点を決定しなければならない.

ところで前節においてあてはめた曲線は, すべてセグメントごとに途切れているため, 曲線をパラメータ空間で延長し曲線同士に交点を持たせる. その上で Bézier Clipping 法 [6] により曲線同士の交点を求める. この Bézier Clipping 法は, 1990 年に西田らによって発表された方法で, 数値的に安定してパラメトリック曲線同士の交点を求めることができる.

このようにして得られた交点が, 曲線の端点となるように曲線を正規化する. 以上の方法により, アウトラインフォントが完成する.

3 適用例

まず, 本研究の手法によりフォントを作成する過程を示す. はじめに入力例として角ゴシック体の「あ」を図 5 に, それに対し輪郭点列および角点を計算した例を図 6 に示す. 続いて曲線を延長し交点を計算した例を図 7 に, 完成したフォントを図 8 に示す.

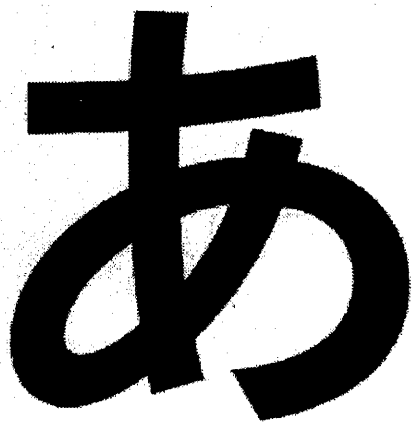


図 5. 入力画像の例

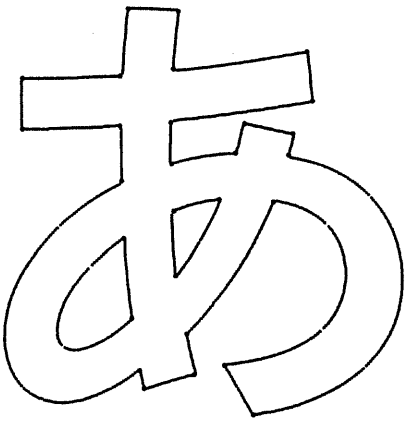


図 6. 輪郭点及び角点の計算例 (●印が角点)

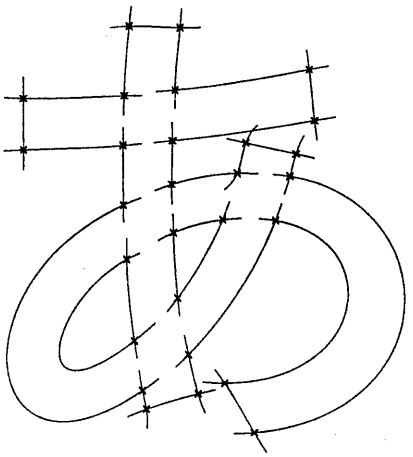


図 7. 曲線及び交点計算例

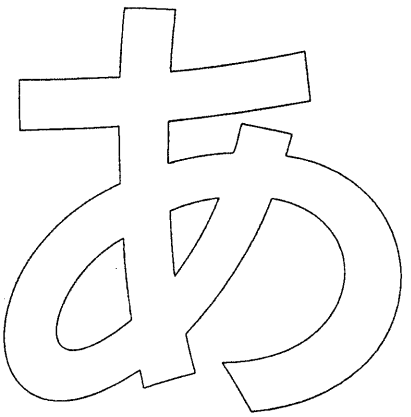


図 8. アウトラインフォントの完成例

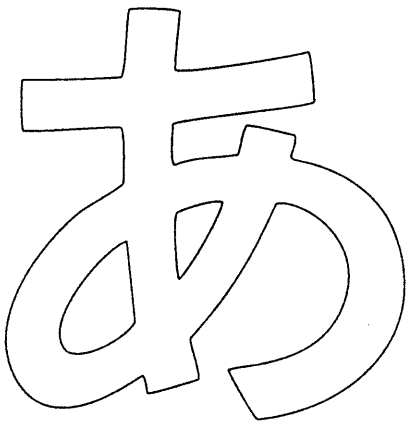


図 9. Schneider の手法によるアウトラインフォント

表 1. 角ゴシック体の「あ」の生成データ

	曲線数	最大二乗誤差
Schneider の手法	34	0.249529
本研究の手法	28	0.249840

4 評価

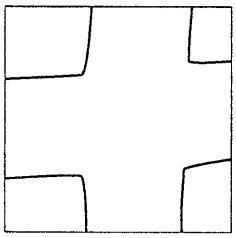
まず比較のために従来法の代表として Schneider の曲線あてはめ法 [4] を同じ入力に適用した例を図 9 に示す.

またそれぞれの手法によりフォントを生成した際のデータを表 1 に示す.

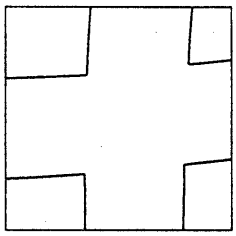
4.1 曲線形状について

まず生成された曲線形状の違いについて比較を行なうと, それぞれの手法における最大二乗誤差はほとんど変わらないにもかかわらず, 両者の角付近の形状がかなり違ったものとなっている. このことを顕著に表している例として, それぞれの一部を拡大した図を図 10 に示す.

この図を見ると明らかに本研究の手法により生成したフォントの方が自然である. したがって本研究の手法により角付近における曲線形状は大幅に改善できたものと思われる.



(a) Schneider の手法による輪郭曲線



(b) 本研究による輪郭曲線

図 10.「あ」の一部の拡大図

4.2 曲線数について

表 1 の曲線数を比べて見ると, 本研究の手法により作成したフォントはかなり少ない本数に抑えられていることがわかる. これは前述のように従来法では曲線の端点を固定して曲線あてはめを行なっていたために, 曲線の自由度が低く, 角付近で曲線あてはめに無理が生じてしまっているためで, その部分の埋め合わせをするために多くの曲線が必要となっているものと思われる.

5 まとめ

本研究の手法により, スキャナで取り込んだ紙上原字の濃淡画像から自動的に 3 次 Bézier 曲線によるアウトラインフォントを生成することができた. 本研究の両端点を固定せずに曲線あてはめを行なう手法により, 生成されたフォントは曲線数については従来法に比べ少なく, また曲線形状についても, 特に原字パターンにおいて角であった部分について従来法に比べ, より原字パターンに忠実な形状を得ることができた.

ただし本研究の手法は草書体のように, 文字の輪郭

が非常に長い滑らかな曲線となるような書体には向かない．その理由は，本研究の手法では一つの滑らかな区間に対して多数の曲線をあてはめる必要がある場合，非常に複雑な式を解かねばならず，それはあらかじめ実装されていなければならない．必要な曲線数が多いような書体に本研究の手法を適用しようとする場合には，本研究の手法を実装すること自体が非常に困難になってしまう可能性がある．

以上のことから本研究の手法は，角ゴシック体や楷書体といったように輪郭線の曲線部分と角とがはっきりしており，連続した曲線区間が比較的短いような書体に対して非常に有効であると思われる．

最後に今後の課題として，より単純な曲線計算法の開発が考えられる．

参考文献

[1] 伊藤，“文字フォントのための曲線の自動あてはめ手法，” 慶応義塾大学大学院 理工学研究科 計算機科学専攻 修士論文, 1993.

[2] G. Avrahami, V. Pratt, “Sub-Pixel Edge Detection in Character Digitization,” *RASTER IMAGING AND DIGITAL TYPOGRAPHY II*, Cambridge University Press, pp. 54–64, 1991.

[3] L. Davis, “Shape Matching Using Relaxation Techniques,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–1, pp. 60–72, 1979.

[4] P. Schneider, “An Algorithm for Automatically Fitting Digitized Curves,” *GRAPHICS GEMS*, Academic Press, pp. 612–626, 1990.

[5] M. Plass, M. Stone, “Curve-Fitting with Piecewise Parametric Cubics,” *Computer Graphics*, Vol. 17, No. 3, pp. 229–239, 1983.

[6] T. W. Sederberg, T. Nisita, “Curve Intersection Using Bézier Clipping,” *CAD*, Vol. 22, No. 9, pp. 538–549, 1990.

[7] T. W. Sederberg, D. C. Anderson, R. N. Goldman, “Implicit Representation of Parametric Curves and Surfaces,” *Computer Vision, Graphics and Image Processing*, Vol. 28, pp. 72–84, 1984.

[8] 鳥島, 山崎，“文字輪郭線の区分的多項式による近似，” 電子情報通信学会技術報告, PRU–87–107, pp. 65–72, 1988.

[9] 寅市, 関田, 森，“高品位文字フォントの自動圧縮，” 電子情報通信学会論文誌 (D), J70–D, 6, pp. 1164–1172, 1987.

[10] 山田, 佐藤，“線分近似表現から 3 次の Bézier 表現への変換手法，” 情報処理学会グラフィクスと CAD 研究会, 38–2, 1989.