一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
LOIS2013-60(2014-3)

手書き入力によるユーザの個人性を反映したフォントの生成

中村　孝司†　檜垣　泰彦†

† 千葉大学大学院工学研究科　〒 260–8522 千葉県千葉市稲毛区弥生町 1–33
E-mail: †higaki@tu.chiba-u.ac.jp

あらまし　現在，手書き文字に出会う機会は少なくなってきている。しかし，手書きの文字は年賀状や日記など各所で用いられ，手書き文字に対する需要は根強い。このため手書き文字風の字体の開発が行われている。しかし，日本語の文字数が膨大なためこの試みは困難である。実際，数学に基づいた方法が候補に挙げられている。我々の研究の目的はユーザー入力に基づく，本人の個性を反映するフォント作成システムの提案である。本研究の方法ではユーザーの入力が不可欠であり，一定の負担を必要とするが，特定の文字記号への高速マッピングを可能にすることでユーザーの不便を解消した。また，一般的にベクタ画像の作成にはラスタ画像のアウトラインをベジェ曲線化する作業を伴うが，本研究では入力データを直接生成する方針を取った。主観実験の結果，フォントがユーザーの個性を反映していることを確認した。

キーワード　手書き，フォント，文字認識，個人性，ベジェ曲線

Generation of Hand-Written Font Reflecting Users' Individuality

Koji NAKAMURA† and Yasuhiko HIGAKI†

† Graduate School of Engineering, Chiba University　1–33, Inage-ku, Chiba, 263-8522 Japan
E-mail: †higaki@tu.chiba-u.ac.jp

Abstract　Today, the chances we see hand-written characters are decreasing despite the constant demand for usage. Thus, attempts are being made to invent hand-written-like font. However, making Japanese fonts has much difficulty because of its amount of characters. In order to solve the problem, math-based generation methods are being proposed. The aim of our study is to propose input-based font generation system, which completely reflects users' individualities. Although our method needs input by users, UI that conducts character recognition enables fast mapping to specific character codes and less trouble to users. When vector image is converted from raster image, in general, it needs to traced outlines of image and to be expressed by Bezier Curve. However, our method generates vector images directly from touch panel coordinates. As a result of subjective experiment, we confirmed that our fonts were reflecting users' individuality.

Key words　Hand-Written, Font, Character Recognition, Individuality, Bezier Curve

1. はじめに

手書きの文字を日常生活で見る機会は減少傾向にある。したがって，手書き風のフォントを作成し利用する試みが多方面で行われているが，日本語は多くの文字から構成されるため，フォント生成には膨大な手間と時間を伴う。その問題を解決するため，SNS を利用したフォント生成の取り組み [1] や既に存在するストロークに数学的処理を加え手書き風にする研究 [2] [3] [4] [5] が行われている。

以上の例は集団でのフォント生成により個人性の保持ができないことと，手書きではないことによる不自然さを消し去ることは難しい。

本研究はフォント作成に掛かる手間・時間を最小限にしつつ，個人性を保持した自然なフォント作成するシステムの提案を目的とした。

2. 設　　計

2.1 要　　件

目的を達成するシステム設計を行うにあたり，以下の仮定から要件を設定した。

（1）　ユーザー人に対し一つのオリジナルフォントを提供
　　ー集団ではなく個人単位でフォントを作成することで，
　　個人性を色濃く反映したフォントができる

（2） 手書きベースで自然で高品質なフォントの生成
—直接入力をもとにしたフォントは自然である
（3） 高速な UI と内部アルゴリズム
—ユーザの手間・時間を最小限にとどめる
（4） 作成するフォントはアウトラインフォントである
—拡大縮小時に美麗さを保持することができる
—ブラウザで使える Web フォント(注1)への対応のため

2.2 関連研究・要素技術調査

要件 (1) について先行研究はなく，本研究は初の試みである。

要件 (2) に関してユーザの書き癖を分析し，ユーザ毎のオリジナルフォント生成を目的とした研究 [3] がある。

要件 (3) 現状直接フォントを作成・編集するにはフォントエディタと呼ばれるソフトを利用する必要がある。フリーのものには Font Forge [6]，有償のものには TTEdit [7] などがある。

要件 (4) 河村らはラスタ・ベクタ画像変換ツール Potrace の最適化を検討した [8]。Potrace は 2 値ラスタ画像の図形のベジェ曲線化時に，多角形図形に分割してからベクタ変換を行う手順をとり，スムーズな曲線を生成する。しかしながら，Potrace で作成されたベクター画像は端点が多くデータ量も多くなるため，Web フォントとしての利用には不向きである。また，ストロークから単純な一定幅の文字を生成するのにはベクター変換とラスタ画像からの変換では状況・最適化の方策も異なる。

2.3 設計の特徴

要件 (1) を満たすため，クライアントサーバモデルを採用，ユーザアカウントで各ユーザの作成したフォントを管理することとした。

要件 (2) を達成するためには，精度の高い入力端末が必要と捉え，アンドロイド端末用のネイティブアプリとして作成し，ユーザの入力をリアルタイムで処理するようにした。生成されたグリフをユーザに提示，ユーザの意向に沿わないできである場合はインタラクティブに字形処理を行える。

要件 (3) のため，文字入力と文字認識を連携させ文字コードへのフォントグリフのマッピングにおける手間を省く。

要件 (4) スケーラブルで簡便なことから，多くのシステムでアウトラインフォントが標準利用されてきており，今研究でも，作成したフォントの今後利用性を考えて，アウトラインフォントとした。

2.4 構　　成

以上を踏まえたシステム設計を提案する。

2.4.1 フォントの作成

サーバ認証のためクライアント端末上でアカウントを作成し，ユーザ名パスワードをサーバ側でデータベースに登録する。端末で入力したストロークデータはベクタアウトライン化され，文字認識で候補に挙げられた文字のコードにマッピングを行う。マッピングされたデータはある程度バッファされたのち，サーバに SVG 形式で転送する。SSL 通信を用いて秘匿性を確保しつつ，meta ノードに付加されたユーザ名とパスワードを絡めデータベースに保存される。

(注1)：組み込みフォントではなく，WWW 上からフォントを表示する技術

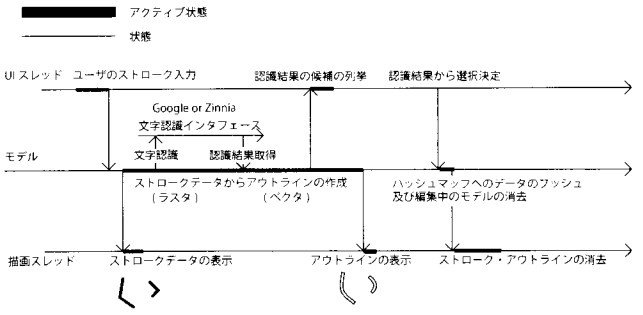

図 1 クライアントアプリにおける各スレッドの関係
Fig. 1 The Relationship Between Each Thread in Client Application.

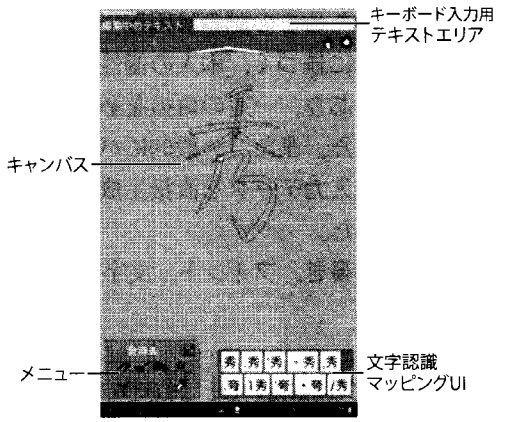

図 2 クライアントアプリのユーザインタフェース
Fig. 2 UI on Client Application.

2.4.2 フォントのブラウズ・利用

フォントブラウズ機能の大部分は html・css で記述し，アンドロイド端末だけでなくその他の端末で確認を行える汎用性の高い仕様とした。SVG 形式でデータベースに保存している為，Web フォントとして読み込み，必要とする量に応じた利用が簡易に実装できる。フォントをダウンロード利用する際には SVG→TTF 変換ライブラリ [9] を用いて，PC 上で組み込みフォントとして扱えるフォーマットとする機能を提供する。

3. 実　　装

3.1 入力の処理

本システムはユーザビリティ・高速化のため図 1 のようにマルチスレッドシステムとして設計している。UI スレッド，モデル，描画スレッドの 3 スレッドで構成される。アプリケーションが起動されると UI パーツが作られそれに伴って画面にレンダリングされる。その後，ユーザの入力を受け付ける状態になり，入力されるとストロークデータからアウトラインの作成を開始する。これには数百ミリ秒かかるので，並行して文字認識を行う。文字認識エンジンは設定によって WebAPI である Google 文字認識もしくは，オフラインでも利用可能であるライブラリ Zinnia を選択できるようにした。アウトライン化及び文字認識が完了すると，ユーザにその結果を提示し，その中からユーザが意図したものを選択させる。選択した文字の文字コードに対して，マッピングを行い，計算に使用していたモ

デルオブジェクトを廃棄する。

3.2 ユーザインタフェース

以上の処理を円滑に行うユーザインタフェースを設計した。図2に概形を示す。中央部キャンパスに文字を入力すると右下部に文字認識結果が表示される。「一 (ハイフン)」「一 (いち)」など紛らわしい文字は上部テキストエリアにキーボードで入力する。左下部には消去ボタン・端点操作ボタン・簡易ブラウズボタン等を含むメニューバーを配置した。

3.3 フォント生成アルゴリズム

グリフのベクタ形式アウトラインの生成には手書き入力値からラスタ画像を作成してから，ベクタ変換ツールを用いる方法が考えられる。広い用途で行う場合はこの方法は好ましいが，筆記に限った本研究の環境においては，線的入力に特化したアルゴリズムを考案し，高速化を検討していくこととした。提案手法は，以下の手順を踏む。

（1） ストロークの取得
（2） 端点設置位置の決定
（3） フォントのアウトライン構成点の計算
（4） アウトラインのベクタデータ化

(1) 端末のタッチパネルから取得した値を逐一配列に格納し，n 個の連続する点を得る。

$$S = [Q_0, \ldots, Q_n] \tag{1}$$

(2) 端点設置位置を決定する。端点の数は多ければ多い程複雑な図形を描くことを可能にするが，計算時間・データ量の肥大化，入力時のノイズをそのまま残すという副作用ももたらす。以上の理由より，必要最低限の端点で構成されるグリフの生成を目指すこととした。

まず，連続する前後の点 Q_{i+1}, Q_i に対する単位ベクトルを計算し，

$$a_i = \frac{Q_{i+1} - Q_i}{|Q_{i+1} - Q_i|} \tag{2}$$

とし，さらに a_{i-1} の a_i に対してベクトルの内積を求めて，

$$\alpha_i \cdot \alpha_{i-1} = \frac{Q_{i+1} - Q_i}{|Q_{i+1} - Q_i|} \cdot \frac{Q_i - Q_{i-1}}{|Q_i - Q_{i-1}|} = \cos(\chi) \tag{3}$$

$$\chi = \arccos\left(\frac{Q_{i+1} - Q_i}{|Q_{i+1} - Q_i|} \cdot \frac{Q_i - Q_{i-1}}{|Q_i - Q_{i-1}|} \right) \tag{4}$$

求まる χ は離散値によるものであるが，本研究では曲率と呼称することとする。急な曲がりを表現するために，曲率が閾値 σ よりも大きい点に端点を置き，これを条件1とする。

また，S字型のストロークに対応するため，曲率の符号が変化する点を制御点設置条件2とする。

同じく筆で書かれた文字のアウトライン抽出を行っている [10] でも，この条件1は行われているが，条件2については行われていなかった。条件1($\sigma = 0.25$) 及び条件2により検出される点 Q の例をそれぞれ□,× として図3に示す。求められたそれぞれの点がお互いに隣り合う場合は平均をとって代表的な Q とする。これに，端点 Q_0, Q_n を含めて最終的な曲線 P_{ij} の端点の基準点とする。

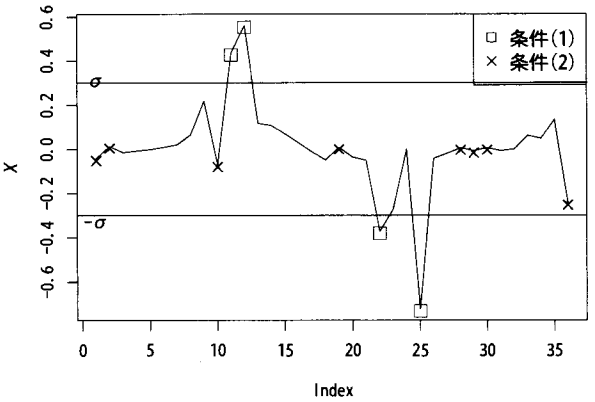

図3 条件1，2により検出される端点設置位置の決定
Fig. 3 Calculated Anchor Point by Rule 1 and 2.

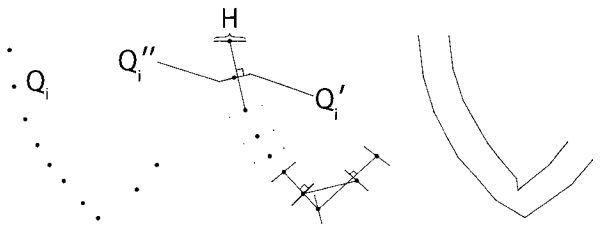

図4 アウトライン構成点の作成手順
Fig. 4 Procedure for Making Outline Dots.

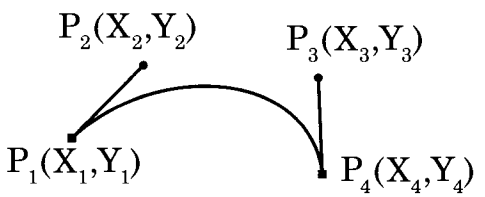

図5 ベジェ曲線
Fig. 5 Bezier Curve.

(3) 図4のようにストロークからアウトラインの構成点を計算する。内側・外側のアウトラインを構成する点はそれぞれ以下で求める。

$$Q' = Q_i + \frac{H}{2}N,$$
$$Q'' = Q_i - \frac{H}{2}N \tag{5}$$

(4) グリフのベクタデータ化のため，アウトラインをベジェ曲線(注2)で近似する。(3) によって求めた点 Q'_i, Q'_j を端点とするベジェ曲線を計算する。3次ベジェ曲線は図5のように $P_1 - P_4$ の4点で構成され，一般式は，

$$P(t) = (1 - t)^3 P_1 + 3(1-t)t^2 P_2 + 3(1-t)^2 t P_3 + t^3 P_4 \qquad (0 \leqq t \leqq 1) \tag{6}$$

で表される。ここで P_1, P_4 を端点，P_2, P_3 を制御点と呼ぶ。

また P_2, P_3 は曲線の接線帖にあるので，$|P_2 - P_1| =$

(注2)：ベジェ曲線 (Bezier Curve) は滑らかな曲線を数式表現するために考案された形式である。2次及び3次のベジェ曲線がよく用いられる。

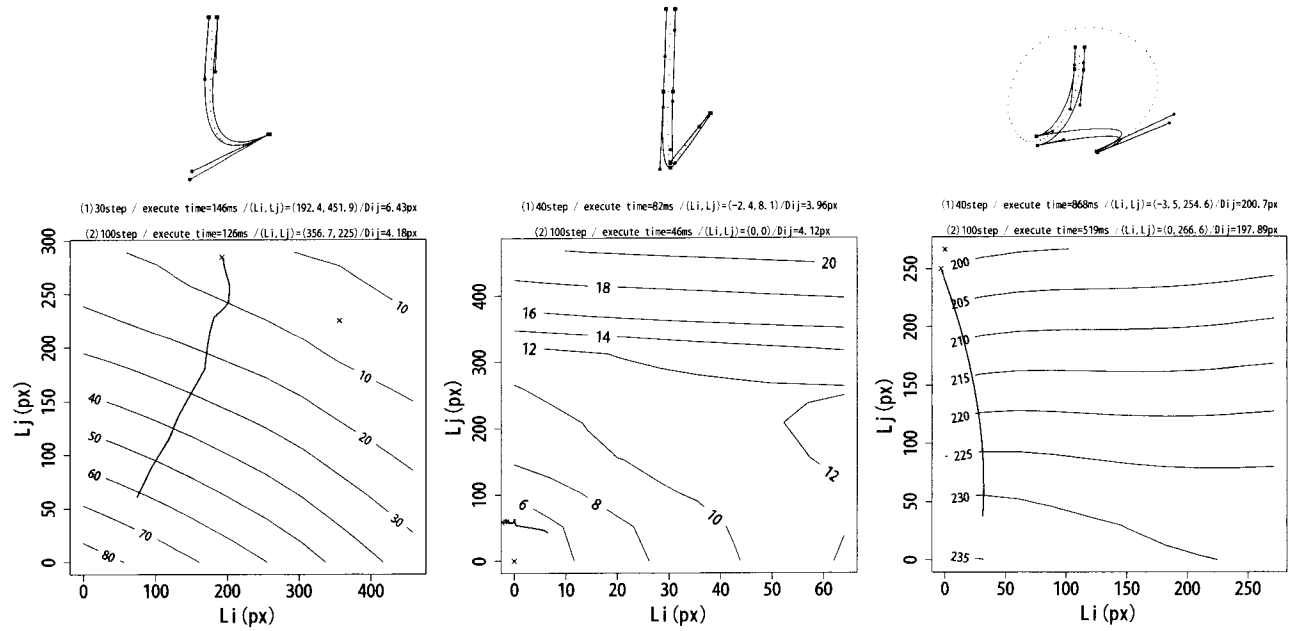

図 6　演算結果例 上から「し」「レ」「の」パターン

Fig. 6　Example of Calculation for Each Pettern "し","レ", and "の".

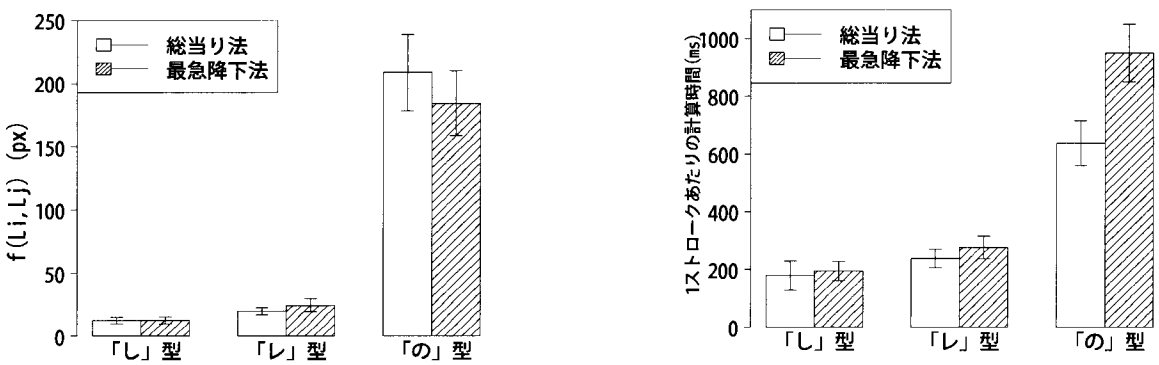

図 7　ストロークの演算に掛かる $f(L_i, L_j)$(左) と平均時間 (右)

Fig. 7　Average Reputation and Time for Calculation.

$L_1, |P_3 - P_4| = L_4$, とおいて，接線の単位ベクトルを (2) 式のように近似すると P_2, P_3 は

$$P_2 = P_1 - \mathrm{L}_1\alpha_1,$$
$$P_3 = P_4 - \mathrm{L}_4\alpha_4 \qquad (7)$$

のようにそれぞれ L_1, L_4 をパラメータとした表現に書き直すことができ，(6) 式もこれを用いて書き直せる。

そこで，(2) の $Q'_i, Q'_j (0 \leqq i < j \leqq n)$ を端点とする最適な曲線 $P_{ij}(t)$ を表現する L_i, L_j を求める。今回は，ベジェ曲線が成す点集合と $i \leqq k \leqq j$ を満たす Q'_k の距離の和 ((8) 式) を最も小さくする曲線が最適だとし，条件を満たす L_i，L_j の長さを探索していく ((9) 式)。これを (2)(3) で求めた全てのアウトラインで行うことでベジェ曲線化されたアウトラインを得る。

$$f(L_i, L_j) = \sum_{k=i}^{j} \min_{0 \leqq t \leqq 1} \sqrt{Q_k^2 + P(L_i, L_j; t)^2} \qquad (8)$$

$$\min_{L_i, L_j} f(L_i, L_j) \qquad (9)$$

4. 評　　価

4.1 アルゴリズム評価

図 6 に示したように 10×10 の領域で $f(L_i, L_j)$ を求めた。以降これを総当りと称する。総当りの解法により，評価関数 $f(L_i, L_j)$ はほぼ単峰性であることが明らかになった。そこで，L_i，L_j の最適化には最急降下法を用いて高速化を図った。図 6 の内部線は最急降下法の計算で辿った道筋である。幾つもの文字の入力の中で，十分な結果が得られる形状及びそうでない形状が判明したため，代表的な 3 パターンに分けて示した。以降詳細を述べる。

4.1.1 「し」パターン–少ない曲線で構成される場合

図 7 上部はもっともシンプルな計算である。長さは初期値から延長され，最急降下法を用いた場合，各曲線共 30step 程度の計算で終了し，収束・計算時間も比較的速い。解は平均 13px 程度の差に収まった。

4.1.2 「レ」パターン–急な曲がりを伴う場合

図 7 中央に 2 ストローク目内側の演算結果を示した。急な曲

がり近辺ではノイズが多くベジェ曲線による近似が歪む。曲率の変位が増大し始めた箇所に端点を配置し，ノイズが大きい箇所を避けて計算を行うべきである。

4.1.3 「の」パターン–緩やかな曲線で構成される場合

図6下部は3ストローク目の計算を示した。閾値 σ の設定が適切でなく，本来必要な場所に端点が置かれないためである。このパターンを解決するには，σ を下げて制御点を増やすことであるが，先に述べたように制御点が増加すると，計算時間とデータ量も増える。今後，十分な σ の値を検証もしくは，全く別の方法を用いての対応が求められる。

4.1.4 総　　評

図7左側に各パターンのストロークの入力からアウトライン化終了までの平均計算時間と評価関数 $f(L_i, L_j)$ の計算結果を誤差95%の信頼区間で示した。計算時間は総当りで計算したほうが計算時間は短くなる傾向にあることが分かった。全体の計算コストに関わる課題点として二つ挙げる。

- 適当な評価の探索に関して総当りで計算する場合の方が多くなるよう設定したが，最急降下法は x 軸,y 軸の二次元での計算が必要となることから結果的に計算量が多くなった。ある程度勾配があり，微分値が得られる場合は最急降下法の方が速かった。解の質については，最急降下法の方がより細かい解が求められられるが，極解に陥りやすく多様に変化するこの関数に対応できない。
- 今回使用した評価関数 $f(x)$ は，ベジェ曲線の構成点 P と $Q_k(i < k < j)$ との距離を全て計算していたため，膨大な演算時間時間がかかった。[10] は P と Q から代表的な点を抽出し最低4点比較することで，計算量に削減に成功しているため，本研究でも取り入れて改善を行っていく予定である。

これらから，1ストロークの入力あたり 200-1000ms の遅延が生じ，後述するアンケート調査でユーザの不満に直結した。少なくとも，3倍のパフォーマンスを出すことが今後の課題である。今回の研究では至らなかったが，5×5 程度荒くメッシュ状に計算したのちに細かい計算を最急降下法で行うなど，両者の利点を活かした解法を用いることで，より高速で良質な計算が可能になることが予想される。

4.2 認識エンジン

認識エンジンとして，Google 文字認識及び Zinnia の利用を検討した。前者は高い認識率を誇り，高速なグリフマッピング UI を実現できた。後者は利用に耐えうる性能が出なかった。これは，SVM ベースの同ライブラリにおいて十分なストロークモデルが用意できず，学習が足りなかった事に起因する。ストロークデータの収集を同時並行で続け，学習させていくことで性能が上がることが期待される。

4.3 主観評価

4.3.1 個人性実験

個人性の評価のため，アプリケーションのテスターが作成したフォントとゲルインクボールペンを使って書いた文字との比較検証を行った。手順は以下のとおりである。

(a) 任意文字をアプリケーションで作成する

(b) 同じ大きさの文字をシートに記入してもらう

表1　アンケート調査の質問項目と点数

質問項目	点数
デザインの良さ	**3.6**
何らかの場面で使えそうですか?	2.8
反応速度への満足度	2.0
文字は自然な出来栄えに仕上がりましたか?	2.8
あなたの職で使えそうですか?	2.8
アプリのアイデアは面白かったですか?	**3.8**
総合点をつけるならば	3.2

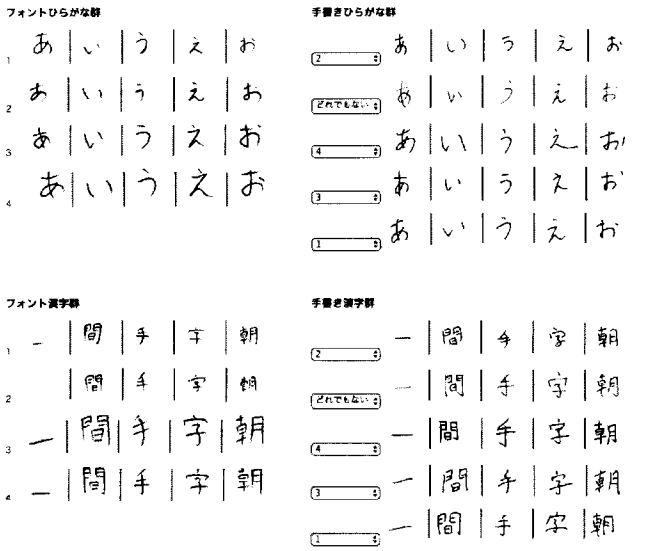

図8　個人性実験用ページ左列:手書きフォント 右列:手書き文字

Fig. 8　Web Page for Examining Individuality.
left:Hand-Written Font
right:Characters Written on a Paper.

(c) 数人の他被験者に (a) と (b) をシャッフルして見てもらい筆記者を当ててもらう。

図8に示す (c) で利用した試験ページである。この試験結果から (10) 式より平均費用価値を算出した。

$$\text{平均評価値} = \text{正解数}/\text{被験者数}\ (0 \leq \text{平均評価値} \leq 5) \quad (10)$$

ボールペンには 0.4mm 幅のものを用い，被験者は大学生41人を対象とした。実験結果を図9に示す。問題の形式上ランダムに回答した場合の期待値は1となるが，漢字の平均評価値は **1.8**，平仮名は漢字よりも高く **2.1** となり，両者とも期待値を上回る結果を得た。満点を獲得した学生もおり，ユーザの個人性を有したフォントの作成できたといえる。

4.3.2 アンケート調査

アンケート調査の結果，表1を得た。これによると，手書きフォントを作成するというアイデアや，デザインについての評価が高かった反面，フォント作成の速度に対する評価は低かった。また，自由記述欄を設けたところ以下の意見があった。

(1) 書いても文字にならない時があり煩わしい。

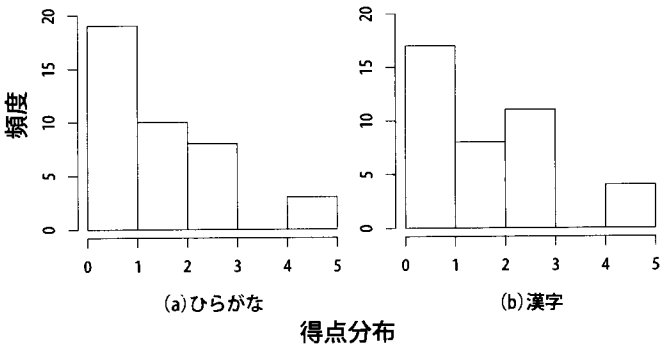

図 9　主観実験集計結果

Fig. 9　Results of Subjective Experiment.

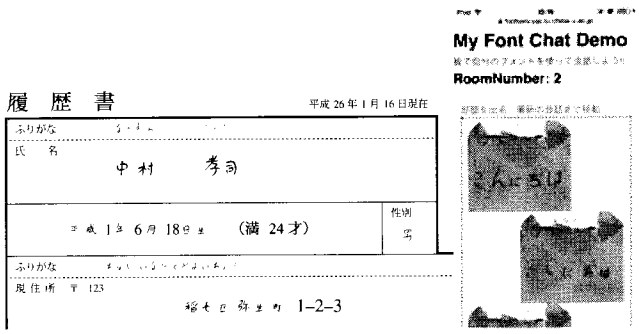

図 10　左:履歴書作成システム　右:手書きチャットシステム

Fig. 10　left:Personal History Generator
right:Hand-Written Font Chat System.

(2) そのまま手書き文字を作るのではなく，既存の書体に敢えて近づける方がいいのではないか?

(3) 写真をトリミングして文字にする機能が欲しい。

(4) 文字を入力したらすぐに文章にして，使えるようにして欲しい。

(5) メッセージアプリで手書きフォントを扱いたい。

これらの意見に対する考察を述べる。

(1) について，4.1 の結果がそのままユーザの満足度に関与していることが自明となった。引続きアルゴリズムの改善の必要がある。(2) について，個人性の保持のため，本研究はできる限り入力値をそのまま使ったが，既往研究 [2] を絡めた複合的フォント作成システムとして仕様変更の検討を行う。(3) について，本研究では行わなかったが，ラスタ画像からのフォント生成機能の要望であり，スマートデバイスを用いたアプリケーションでは考えられて当然の意見であった。文書をそのまま撮影，フォント化することは精度面で難しいと考えられるが，テキストを抽出できればフォント化すべき文字の推薦に利用可能であると思われる。

今後，[5] [10] 等のように画像そのものからのグリフの生成を検討し，この方法でユーザの個人性・満足度を得られるならば採用していく予定である。

(4)(5) から，手書きフォント利用環境拡充のため，組版処理ソフトウェア TeX を用いた履歴書作成システム・自作フォントチャットアプリ Font Chat の 2 種類を試作した。システムの外観を図 10 に示した。フォント作成・利用両アプリケーションを利用したテスターから「ただフォント作るより作る意欲が湧く」という評価を受けた。

5. おわりに

本研究では個人性の保持した自然なフォント作成するシステムを提案し，直接手書きした文字からフォントを作成しユーザアカウントとフォントを紐付け，フォントを提供するシステムを提案した。手書き文字と生成された手書きフォントの比較実験を行った結果，期待値以上の結果を取得し，個人性の維持を確認した。

文字認識と手書き文字入力を結びつけた，高速な文字マッピングインターフェースを提案した。しかし，グリフ生成速度に関してユーザの満足度は低かった。

テスターに対してのアンケート調査の結果，生成されたフォントは演算の状況によって不具合をもたらすことがあるが，自然で質が高いフォントが生成できた。

加えて，アンケート調査から手書きフォント環境の整備の必要性を確認し，履歴書作成システム，手書きチャットシステムを構築した。これにより，ユーザの手書きに対しての興味を喚起することができた。

今後，アルゴリズムの改良及び構成の見直し，オフライン時にも利用可能な文字認識ライブラリの実装，周囲の利用環境の拡充を行い，ユーザの満足度を向上していく予定である。

文　献

[1] 株式会社カヤック, "FONTA 一人一文字みんなでつくるフォント", http://fonta.kayac.com/, (2014/1/28 アクセス)

[2] 安本 護, "域的個人性と局所的個人性に基づく手書き風フォントの生成", 電子情報通信学会論文誌, Vol.J80-D-II, pp.2930-2939, Nov.1997

[3] 中西貴之, "書き癖パラメータ設定による手書き風フォントの自動生成", 電子情報通信学会技術報告 Vol33, No.37, Sep. 2009

[4] 和田章男, "多様なデザイン機能を備えた感性を反映する日本語フォント自動作成システム", デザイン学研究, Vol.52, No.6 , pp.9-16, Oct.2007

[5] 但馬文昭, "遺伝的アルゴリズムによる個性と変動を伴った手書き風文字の生成", 電子情報通信学会技術報告 教育工学, Vol.98, No.433, pp.125-132, Nov.1998

[6] "FontForge", http://fontforge.org, (2013/1/28 アクセス)

[7] "TTEdit", http://opentype.jp/ttedit.htm, (2013/1/28 アクセス)

[8] 河村 圭, " ベクター変換における曲線最適化アルゴリズムの一検討 ", 第 3 回情報科学技術フォーラム一般講演論文集, Vol.3, No.3, pp261-262, Aug. 2004

[9] "Fontello", https://github.com/fontello, (2014/1/28 アクセス)

[10] Hsi-Ming Yang, "A Bezier curve-based approach to shape description for Chinese calligraphy characters", Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE,pp.276-280,2001.