卒業論文

サンプルによらないフォントの自動生成に関する研究

東北大学 工学部 情報知能システム総合学科
大町研究室 ４年
景山 竣

目 次

図 目 次

表 目 次

第1章

まえがき

1.1 研究の背景

私達の身の回りにはたくさんのフォントで溢れている.

雑誌やチラシ, 看板から web サイトまで, 幅広い範囲で様々な種類のフォントが使用されている. デザイン的な観点はもちろん, 情報を伝えやすくするために用途にあわせてフォント選択することもあり, フォントには大きな需要があるといえる.

特に近年では web ページにおけるフォントの利用がよく見られる. というのは, web ページ上でサーバ上に置かれたフォントを使用する「web フォント」が, CSS3 において規定されていて [1] 世界的に一般的な技術になっていたり, 通信速度やデバイス容量の増加によってフォントの大量使用が可能になっていることから, web 上における様々なフォントの使用が可能になったためである. 実際に多様なフォントを駆使した web サイトもめずらしくない [2].

そんな中, 日本語のフォントの需要も高まっているが, 日本語のフォントとは制作が大変で, さほど出回っていないというのが現状である. これは日本語フォントがひらがな, カタカナに加えて大量の漢字を含んでいるためである.

英語と日本語のフォントの収録文字数を比較してみると, 英語はアルファベットの大文字小文字あわせて計 52 文字, 記号を含めても 100 字程度あれば十分である. 一方日本語は, ひらがな, カタカナあわせただけですでに 169 文字あることに加え膨大な量の漢字がある. 例として, 一般的に商用フォントに収録される, JIS 漢字コードにおける第 1 水準漢字と第 2 水準漢字をあげると, 6355 文字もの漢字がある.

しかも, 漢字は着本的に複雑な構造をしているため 1 文字 1 文字手作業でデザインすることが普通であり, 1 つのフォントを完成させるのに数年かかると言われている.

このフォント制作の困難さから, 日本語のフォントは英語のフォントと比べて非常に高価である.Adobe 社のフォントを例にあげると, 英語のフォントが 4 千円程度で販売されているのに対し, 日本語のフォントは数万円が相場である [3].

今後, さらに日本語フォントの需要が高まることが予想されるため, 日本語フォントの制作をより容易にするための技術が求められているといえる.

1.2　研究の目的

本研究では日本語のフォント制作を容易にするための先行研究 [4] に改良を加え, 任意の文字を入力可能にするための手法を提案する.

日本語フォントを自動生成する研究には先行研究が存在する.15 字程度のサンプルをもとに大量の漢字デザインを自動生成するというもので, ひとつのフォントを作るのに必要な漢字を高精度に生成することができる.

しかし先行研究は入力に制限があり, 任意の文字を入力とできない. そこで本論文では, これを解決する手法の提案をし, 先行研究を改良することを目的とする.

第2章

先行研究

研究の目的はフォントの自動生成である. 研究にあたって土屋の手法「パーツへの分解を用いた漢字フォントの自動生成に関する研究」[4] を参考にした. 本章ではこの先行研究に関して大まかではあるが説明していく. また, 本章で使用した画像に関しては, 土屋の手法 [4] の論文から引用したものであることをご了承いただきたい.

2.1 先行研究の概要

先行研究の全体の流れを説明する.

先行研究は文字のサンプル画像とその骨格データを入力として用い, そこから新しく文字を生成するという流れになっている.

入力データとは別に, GlyphWiki と呼ばれる文字の骨格データを管理したデータベースも利用している. 詳細に関しては後述する.

先行研究は前処理と文字生成の 2 段階構成になっている. 前処理では入力データをもとに文字をパーツに分解し, さらに骨格データの変形の推定を行う. 文字生成では, 分解したパーツと学習した骨格データを利用して文字を生成する.

2.2 GlyphWiki

先行研究の内容を説明する前に, 先行研究で利用しているデータベースである GlyphWiki について説明する.

2.2.1　GlyphWiki について

GlyphWiki とは, 明朝体の漢字の骨格データを Wiki 形式で管理したデータベースである. 利用者は GlyphWiki のデータを完全にフリーなライセンスで利用でき, 骨格データを作成・登録することも可能である. 現在数十万の漢字の骨格データが登録されている.

2.2.2　GlyphWiki のデータ形式

GlyphWiki では各文字がストロークの組として表現されている. ストロークは「線種」「頭形状」「尾形状」の 3 つのカテゴリ情報を持っている.「線種」は文字通り直線, 曲線といったそのストロークの線の種類を表している.「頭形状」,「尾形状」はストロークの両端の状態を表しており, 払い, ハネ, 他のストロークに接しているか否か, といった情報が記録されている.

ストロークの形は, 始点, 終点と 0 - 2 個の制御点の点郡によって表される. この点群のデータをもとに描いた軌跡がストロークの形となる. 制御点の数は「線種」によって決まっており, 詳細は以下の通りである. 制御点が 2 つある場合は, 2 つの制御点を制御点 1, 制御点 2 と区別している.

線種 直線

軌跡は始点, 終点を結んだ直線

線種 曲線

軌跡は始点, 終点, 制御点の 3 点で描いた 2 次ベジェ曲線

線種 複曲線

軌跡は始点, 終点と 2 つの制御点で描いた 3 次ベジェ曲線

線種 縦払い

軌跡は, 始点と制御点 1 を結んだ直線＋制御点 1 と制御点 2 と終点で描いた 2 次ベジェ曲線

線種 折れ

軌跡は, 始点と制御点 1 を結ぶ直線＋制御点 1 と終点を結ぶ直線

線種 乙線

軌跡は, 始点と制御点 1 を結ぶ直線＋制御点 1 と終点を結ぶ直線

2.2.3　GlyphWiki データの分類

先行研究では, ストロークを扱いやすくするために, カテゴリ情報をもとにストロークを 12 種類のタイプへと分類している. 本論文ではこれをストロークカテゴリと呼ぶこととする. ストロークカテゴリの一覧を図 2.1 に示す.

図 2.1: ストロークカテゴリの一覧

2.3　入力データについて

入力データは文字のサンプル画像と, その骨格情報からなる.

サンプル画像は文字通り文字のサンプル画像である.

サンプル画像の骨格情報とは GlyphWiki のデータの始点や終点, 制御点をサンプル画像に合わせてマッチングしたものである.

2.4　パーツへの分解

パーツ分解はまず骨格情報を用いて大まかに分割したあと, 動的輪郭モデルを用いてよりきれいに分解するという流れになっている. 以下, それぞれについて詳しく説明していく.

2.4.1　骨格を用いた領域分割

まずは骨格情報をもちいておおまかに領域分割を行う. 画像中の各画素について最も近いストロークを記録していく. これによって, 各ストロークのおおまかな境界線が求まる.

2.4.2　動的輪郭モデルを用いた領域分割

続いてより高精度に領域分割を進めていく. ここで, 動的輪郭モデル [5] と呼ばれるものを利用する.

動的輪郭モデルとは物体の領域分割やエッジ抽出などに使用される手法であり, Snakes とも呼ばれる.Snakes は制御点群と制御点を結んだ曲線で表現されている. 適当な初期位置を与えた曲線についてエネルギー最小化を行うことで, 領域分割のための境界線や物体のエッジを求める手法である. エネルギーは外部エネルギーと内部エネルギーの 2 つのエネルギーが存在する.

内部エネルギーとは, 与えられた曲線の張力や剛性を保持する役割をもつ. 内部エネルギーを小さくすることで滑らかな曲線を得られるようになっている.Snakes の各点の位置ベクトルを $u(s), s \in [0, 1]$

とすると, 内部エネルギーは次式で表される.

$$E_{int} = \int_0^1 \alpha \left| \frac{\partial \boldsymbol{u}}{\partial s} \right|^2 + \beta \left| \frac{\partial^2 \boldsymbol{u}}{\partial^2 s} \right|^2 \tag{2.1}$$

外部エネルギーは, 実際に画像のデータを利用して求めるものであり, 目的によって定義は異なる. 例えばエッジ抽出の場合は, 画像のエッジ部分のエネルギーが低くなるような画像を用意し, Snakes を適用させることで, Snakes がエッジに沿うように移動する.

先行研究では, 先ほど述べた骨格情報による領域分割で求めた境界線に対して動的輪郭モデルを適用する. そうすることでエネルギーを最小したときに描く曲線がパーツを抽出するための切り取り線となる.

そのために先行研究では, サンプル画像を反転・平滑化した画像と, 骨格画像を反転・平滑したものを足し合わせた画像の画素値を外部エネルギーとしている.

こうして Snakes を適用させてエネルギー最小化を行った境界線は, 文字の実線・骨格を避けるような曲線になる.

以上の工程をもって, パーツを綺麗に切り取っている.

2.5　骨格の変形の推定

もう一つの前処理の内容である, 骨格の変形の推定について説明する.

先行研究では分解したパーツを骨格に合わせて配置することで文字を生成するが, このとき配置する骨格が GlyphWiki のデータそのままだと, 明朝体のような字形になってしまう（GlyphWiki は明朝体の字形を管理したデータベースであるため）.

そこで先行研究では, 入力として受け取った骨格データと GlyphWiki の骨格データを用いて, 骨格の変形の推定を行っている. 具体的には, GlyphWiki の骨格データから入力として受けとった骨格データへの変形を, アフィン変換を用いて推定する. 文字生成の際は生成したい文字の GlyphWiki の骨格データにアフィン変換を適用し, サンプル画像のような字形を求める.

先行研究では 2 種類のアフィン変換を推定する.1 つめは文字全体の大まかな形を推定するアフィン変換であり, 文字全体の大きさや縦横比に関係するものである.2 つ目は文字を小さな骨格の集まり（サブ骨格と呼ぶこととする）に分解し, サブ骨格ごとに推定したアフィン変換であり, 骨格の傾きなどに関係するものである.

ここで, 本論文におけるアフィン変換 T は, 以下のように点 $p(x, y)$ を $p'(x', y')$ へと変換するものとする.

$$(x', y', 1) = (x, y, 1)\boldsymbol{T} \tag{2.2}$$

2.5.1　文字全体に適応させるアフィン変換

文字全体の大まかな形を推定するアフィン変換を求める. 推定には骨格の外接矩形を用いる.GlyphWiki の骨格データの外接矩形から, 入力として与えられた骨格データの外接矩形へと変換するアフィン変換が, 求めたいアフィン変換である.n 番目のサンプルの骨格の外接矩形の大きさを縦 $Y(n)$,

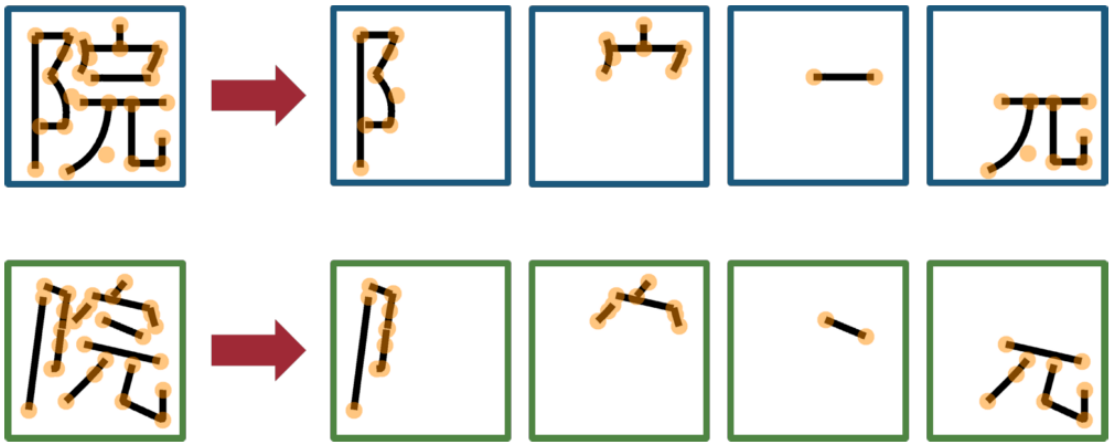


図 2.2: サブ骨格への分解

横 $X(n)$, またそのサンプルの文字の GlyphWiki の骨格データに関して, 外接矩形の大きさを縦 $Y_{gw}(n)$, 横 $X_{gw}(n)$ とする. このとき, 次式のようにして文字全体に対するアフィン変換 T_1 を求める.

$$\boldsymbol{T}_1 = \frac{1}{N}\sum_{i=1}^{\boldsymbol{N}} \begin{pmatrix} \frac{X(i)}{X_{gw}(i)} & 0 & 0 \\ 0 & \frac{Y(i)}{X_{gw}(i)} & 0 \\ I_x \frac{X(i)}{2X_{gw}(i)} & I_y \frac{Y(i)}{2Y_{gw}(i)} & 1 \end{pmatrix} \tag{2.3}$$

I_x, I_y はそれぞれ出力する画像の横, 縦のサイズである.

2.5.2　サブ骨格ごとに適用させるアフィン変換

文字全体に適用させるアフィン変換を求めた後はサブ骨格ごとに適用させるアフィン変換を求める.

アフィン変換を求める前に, まずは骨格をサブ骨格に分解する.

GlyphWiki のデータを利用して, 互いに接触, 交差または連続しているストロークを一つの組としてまとめることで, サブ骨格を得る. サブ骨格への分解を行った例が図 2.2 である. この分解したサブ骨格ひとつひとつに対して, 最小二乗法を用いてアフィン変換を推定する.

この推定は, GlyphWiki の骨格データからサンプルの骨格データへの変形を, それぞれの骨格からサンプリングした点群を用いた最小二乗法よって予測するというものである. 具体的には, GlyphWiki データの骨格情報を一定数でサンプリングした点群を $\{p_{gw\_1}, p_{gw\_2}, \cdots, p_{gw\_n}\}$, サンプルの骨格データを一定数でサンプリングしたものを $\{p_1, p_2, \cdots, p_n\}$ とすると, アフィン変換は次式で表される.

$$\boldsymbol{T_2} = \arg\min_{\boldsymbol{T}} \sum_{i=1}^{N} |p_i - \boldsymbol{T}(p_{gw\_i})|^2 \tag{2.4}$$

ここで $T(p)$ は p に対してアフィン変換 T を適用した点を表す.

2.6　文字生成

前処理を終えたあとは, 骨格にパーツを配置して文字を生成する段階に入る.

まずは生成したい文字の GlyphWiki の骨格データを取得し, 前処理で推定したアフィン変換を適用させる. 次にその変形した骨格に, 分解したパーツを適切に配置して文字を生成する.

以下, それぞれの詳細な流れを説明する.

2.6.1　骨格の変形

生成したい文字の骨格データに対して, 前節で述べた骨格の変形の推定において求めたアフィン変換を適用する.

まずは骨格全体に対するアフィン変換を適用する. そのあと前節と同様の方法でサブ骨格に分解し, 各サブ骨格に対してサブ骨格に対するアフィン変換を適用する.

これによって, サンプルの字形の特徴を取り入れた字形を得ることができる.

2.6.2　パーツの配置

推定した骨格にパーツを配置していく. このとき, パーツはコスト関数と呼ばれるものに従って, 最もコストが低いパーツを選択して配置する.

コストは 3 種類のコストの合計値である. 各コストについて説明する.1 つ目は生成したい文字のストロークの骨格データと配置候補のパーツの骨格データを比較して算出するコストである.2 つ目は生成したい文字のストロークと配置候補のパーツの GlyphWiki 上の骨格データを比較して算出するコストである.3 つ目は生成したい文字のストロークに隣接しているストロークと配置候補のパーツに隣接しているストロークを比較して算出するコストである.

骨格データが似ているほどコストは低くなるため, 最も似ているパーツが配置されるようになっている.

コスト関数に関するより具体的な詳細はここでは省略させていただく.

以上が先行研究による文字生成の全行程である.

第3章

提案手法

3.1 先行研究の問題点

提案手法に関する説明の前に先行研究の問題点について述べる.

前章では詳しく触れなかったが、先行研究には入力文字の制限がある.

先行研究では、2.2.3 で述べたように漢字のパーツが 12 のストロークカテゴリに分けられている. 入力文字はこの 12 のストロークカテゴリのパーツを含んでいる必要がある. 出力したい漢字のパーツに入力文字に含まれていないストロークカテゴリがあると、文字生成をすることができない.

少数の文字デザインを作成してフォントを自動生成したいときは、すべてのストロークカテゴリの含むサンプル文字をデザインすればすればよい. よってこの入力文字の制限はさほど問題にはならない.

しかし、スマートフォンやタブレットなどの小型電子機器の発達が著しい昨今、「文字画像を撮影してその画像をもとにフォントを生成する」といったアプリなどの形で利用されることが十分に考えられる. 撮影する文字画像は全てのストロークカテゴリを含んでいるとは限らないので、サンプルによらない任意の文字を入力可能とする必要がある.

このように、入力文字に制限があることによって先行研究で実現出来ることが限定されているといえる. そのため本論文では、先行研究の問題点を解決するために「サンプルによらないフォントの自動生成」をテーマとしている.

3.2 解決手段

サンプルによらないフォントの自動生成のために私が考案したアイデアがある. それは, 入力文字に対して特定のストロークカテゴリのパーツが不足した場合に, 他のパーツから補完するという

ものである. 例えば「しんにょう」カテゴリのパーツが不足したとき, 「直線」カテゴリのパーツを用いて「しんにょう」の代用をさせるといった具合である.

処理の具体的な内容としては, 文字の骨格データを用いてパーツ変形を行い, パーツの補完を行う, というものになっている. これを「骨格データを用いた変形」と称し, 次節で詳しく述べていく.

3.3　骨格データを用いた変形

提案手法「骨格データを用いた変形」に関して説明していく.

この手法は, 前処理とパーツ生成の 2 段階から構成されている. 前処理ではパーツ画像からデータを取得し, パーツ生成では前処理で取得したデータと生成したいパーツの骨格データを用いてパーツの生成を行う. 以下, それぞれの処理内容について詳しく述べる.

3.4　前処理

前処理は以下のような流れになっている.

(1) パーツの切り出し

パーツの切り出しは先行研究のパーツ切り出しの手法をそのまま用いる.

(2) 輪郭点を求める

パーツの輪郭点を求める. パーツ画像は白黒画像であり黒い部分がパーツに相当するので, この周りの点を輪郭点として求める.

(3) 各輪郭点と骨格データの関係を求める

各輪郭点と骨格データの関係を求める. その前に骨格データに関する少し詳しい話を述べる.

骨格は制御点データを用いて表現できるようになっているが, 基本的には直線, 2 次ベジェ曲線, 3 次ベジェ曲線及びそれらの線の組み合わせである. 式で表すと以下のようになる.

(a) 直線

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \end{bmatrix} + t \begin{bmatrix} x_e - x_s \\ y_e - y_s \end{bmatrix} \tag{3.1}$$

(b) 2 次ベジェ曲線

$$\begin{bmatrix} x \\ y \end{bmatrix} = (1-t)^2 \begin{bmatrix} x_s \\ y_s \end{bmatrix} + 2t(1-t) \begin{bmatrix} x_{c1} \\ y_{c1} \end{bmatrix} + t^2 \begin{bmatrix} x_e \\ y_e \end{bmatrix} \tag{3.2}$$

(c) 3 次ベジェ曲線

$$\begin{bmatrix} x \\ y \end{bmatrix} = (1-t)^3 \begin{bmatrix} x_s \\ y_s \end{bmatrix} + 3t(1-t)^2 \begin{bmatrix} x_{c1} \\ y_{c2} \end{bmatrix} + 3t^2(1-t) \begin{bmatrix} x_{c2} \\ y_{c2} \end{bmatrix} + t^3 \begin{bmatrix} x_e \\ y_e \end{bmatrix} \tag{3.3}$$

ここで, 始点 (x_s, y_s), 終点 (x_e, y_e), 制御点 $(x_{c1}, y_{c2}), (x_{c2}, y_{c2})$ である.

式より, パーツの始点を $t = 0$, 終点を $t = 1$ とした式となっていることが分かる. この t のことを骨格データ上における相対距離と呼ぶこととする. これを踏まえて処理の内容を説明していく.

各輪郭点から骨格に対して垂線を引く. この垂線に関して, まずは垂線の長さを求める. また, この垂線の足すなわち Glyph と垂線の交点について, Glyph 上の相対距離を求める. この垂線の長さと相対距離の 2 つのデータは「輪郭点は骨格データ上のこのくらいの位置からこれくらい離れたところにある」ということを表すデータである. これらのデータがあれば別の骨格に沿って輪郭点を配置することができる.

前処理のおおまかな流れを図 3.1 に示した.

3.5　パーツ生成

パーツ生成は以下のような流れになっている.

(1) 生成したいパーツの骨格データを用意

パーツを生成する際は Glyph のデータが必要であるため, まずはこれを準備する必要がある.

(2) 前処理で求めたデータを使って輪郭点を配置

骨格データ上の相対距離を求め, そこから垂線の長さのぶんだけ動かした位置に輪郭点を配置する.

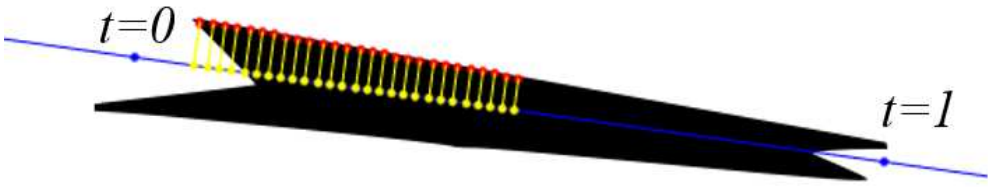
(3) 輪郭点に囲まれた範囲を塗りつぶす

輪郭点で囲まれた範囲を塗りつぶすことで, パーツを生成する.

パーツ生成のおおまかな流れを図 3.2 に示した.
以上がパーツの補完に関する提案手法の全行程である.

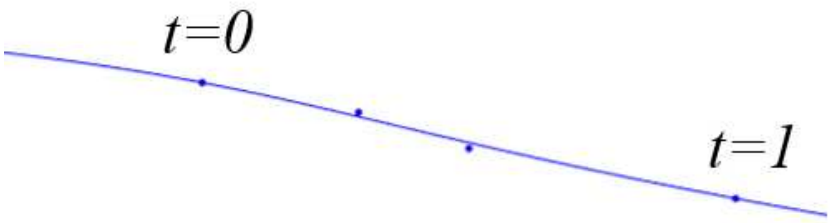
先行研究の方法を用いてパーツ分解

輪郭点を求める

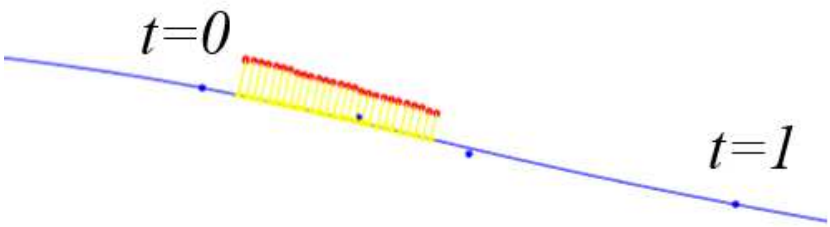


各輪郭点と骨格の関係を求める

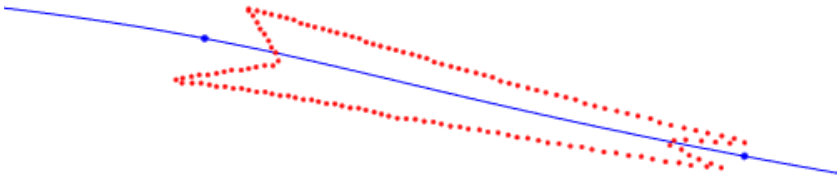
図 3.1: 前処理



骨格データを用意



輪郭点を配置



輪郭点配置終了

塗りつぶし

図 3.2: パーツ生成

第4章

実験

実験は文字生成実験と主観評価実験の 2 種類の実験を行った.
実験内容の説明の前に実験環境について説明する.

4.1 実験環境

文字生成にあたって, 既存のフォントの文字画像を入力用のサンプルとして用意した.
その際, 使用したフォントは次の 4 種類である.

- イバラ字[1]
- g_ コミック古印体[2]
- SR 角田さん[3]
- ジンペン糸-R[4]

4.2 生成実験

文字生成実験に関して述べる.
先行研究では「下八汽応究考字従熟述将続代望冷」の 15 文字を入力として使用している. そこで今回は, 提案手法を用いて「下八汽」の 3 文字から「応究考字従熟述将続代望冷」の 12 文字を生成し, これらを合わせた 15 文字を入力として使用することにした.

[1]fub 工房 http://www2s.biglobe.ne.jp/ fub/
[2]よく訓練された素材屋 http://material.animehack.jp/
[3]SRH 手作りフォント http://a17-s.net/srh/m/f/h694.html
[4]尋スタジオ http://zinsta.net/

不足しているストロークカテゴリのパーツの補完は提案手法を用いる.

入力文字の「下八汽」に含まれているパーツは「直線」「左曲線」「右曲線」「短点」「右曲線＋ハネ」である.

また,「応究考字従熟述将続代望冷」を生成する際に足りないパーツは「直線＋ハネ」「左曲線＋ハネ」「しんにょう」「折れ」「折れ＋ハネ」「縦払い」である.

これらの足りないパーツの補完を以下のように行う.

- 「直線＋ハネ」「しんにょう」「折れ」「折れ＋ハネ」は「直線（”下”の 1 画目)」を用いて補完
- 「左曲線＋ハネ」「縦払い」は「左曲線（”八”の 1 画目)」を用いて補完

なるべくパーツの形が近いもので補完したかったため, このような補完方式となった.

ここで文字生成の際に使用した骨格は, 先行研究の手法を用いて推定したものではなく, オリジナルのフォントの文字にマッチングさせた骨格データを使用している. これは今回の実験はパーツの補完がメインであることを考慮したためである.

入力文字 3 文字と作成した 12 文字を合わせた 15 文字を基にして文字生成を行う.

出力した文字は「歌奮散眼毒然詞忠結危便幹苦心鉱」. これは教育漢字（小学生が習う漢字のことで, 全部で 1006 字）からランダムに選択したものである.

生成した文字を図 4.1 に示す. 比較のために, オリジナルの文字画像と 15 文字のサンプルから自動生成した文字画像と共に並べている. 先行研究がサンプル数 15, 提案手法がサンプル数 3 である.

結果を見ると, サンプル数が少ないにもかかわらず非常にきれいな文字が生成できている. 提案手法は不足パーツを上手に補完できているといえる.

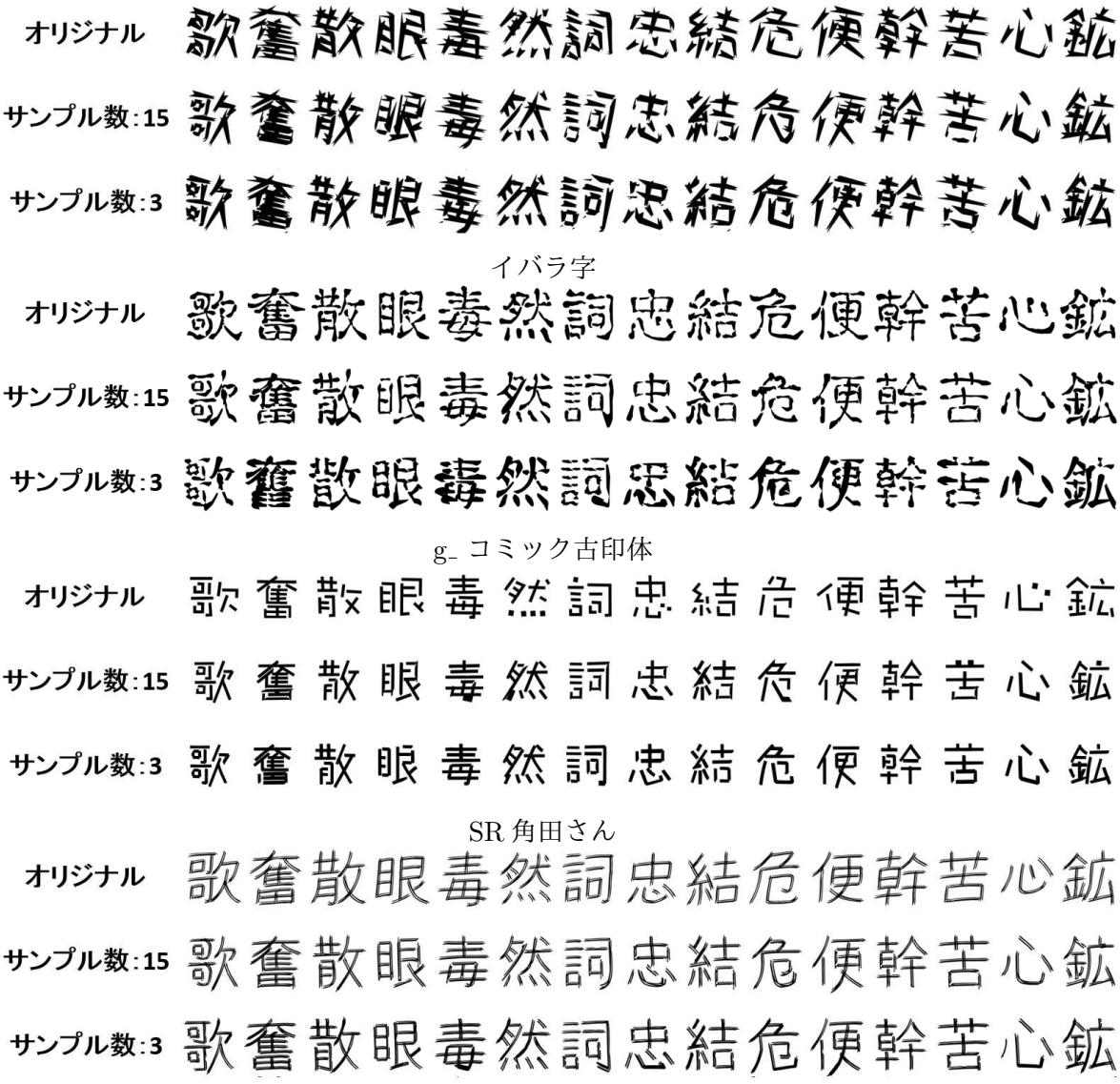


図 4.1: 文字生成実験結果

4.3　評価実験

生成実験で生成した文字について主観評価実験を行い, 提案手法の数値的評価をする. 本論文では 9 人の参加者に 2 種類の主観評価実験をしてもらった. 以下で結果について述べる.

4.3.1　平均オピニオン評点（MOS）による評価

生成された文字に対して 1（非常に悪い）から 5（非常に良い）の 5 段階評価をしてもらう, 平均オピニオン評点と呼ばれる指標を用いて評価を行った.

オリジナルのサンプル文字画像 10 枚を提示した上で, 与えられた四字熟語の文字デザインがそのサンプルと「同じフォントとしてふさわしいかどうか」という問いに対して 5 段階評価してもらった.

実験中のスクリーンショットを図 4.2 に示す.

四字熟語の文字デザインは, オリジナルのフォントから作成したもの, サンプル 15 文字から生成したもの, 提案手法を用いてサンプル 3 文字から生成したもの, の 3 種類から構成されており, それぞれの場合についてランダムで 10 個ずつ用意した.

この実験は 4 種類すべてのフォントに対して行った.

結果を表 4.1 と図 4.3 に示す.

サンプル数 15 と 3 であまり大きな差がないことや, フォントによってはオリジナルに近い評価を得られているものもあることから, 提案手法はパーツの補完がきれいに行えていることがわかる.

表 4.1: MOS による評価実験の結果

	イバラ字	g_ コミック古印体	SR 角田さん	ジンペン糸-R
オリジナル	3.6	4.0	4.1	4.3
サンプル数:15	3.4	3.3	2.9	3.8
サンプル数:3	3.4	2.8	2.8	3.5

図 4.2: MOS による評価実験の様子

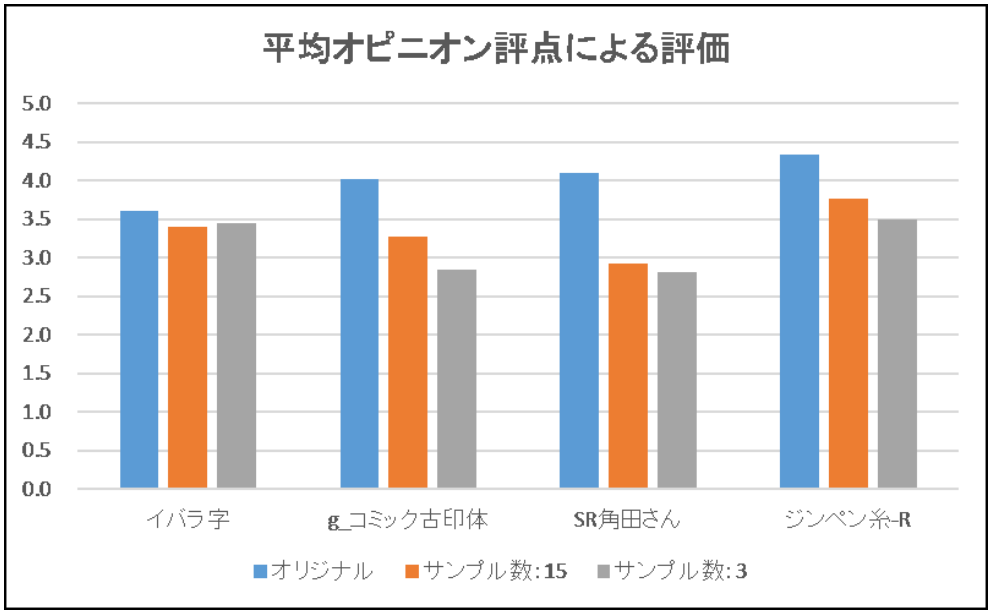


図 4.3: MOS による評価実験の結果

図 4.4: 自動生成と手動のデザインを判別する評価実験の様子

4.3.2　自動・手動の判別による評価

オリジナルのフォントの文字, サンプル 15 文字から生成した文字, 提案手法を用いてサンプル 3 文字から生成した文字, の 3 種類の文字をランダムに並べ, その中から「機械によって自動生成された」と感じる文字を選んでもらうという評価実験を行った.

実験中のスクリーンショットを図 4.4 に示す.

それぞれの場合について 50 字ずつ文字を用意し, 合計 150 文字をランダムに並べて評価してもらった.

結果を表 4.2 と図 4.5 に示す.

全体的にサンプル 3 文字でもサンプル 15 文字に並ぶほどの良い結果が得られており, フォントによってはサンプル 15 文字よりも良い結果が得られていたりオリジナルに近い結果であったりすることから, 提案手法ではパーツ補完がきれいに行えていることが示せた.

表 4.2: 自動・手動の判別による評価実験の結果

	イバラ字	g_ コミック古印体	SR 角田さん	ジンペン糸-R
オリジナル	16.4%	22.0%	7.8%	3.6%
サンプル数:15	17.6%	24.7%	30.2%	13.3%
サンプル数:3	25.1%	26.0%	28.9%	20.0%

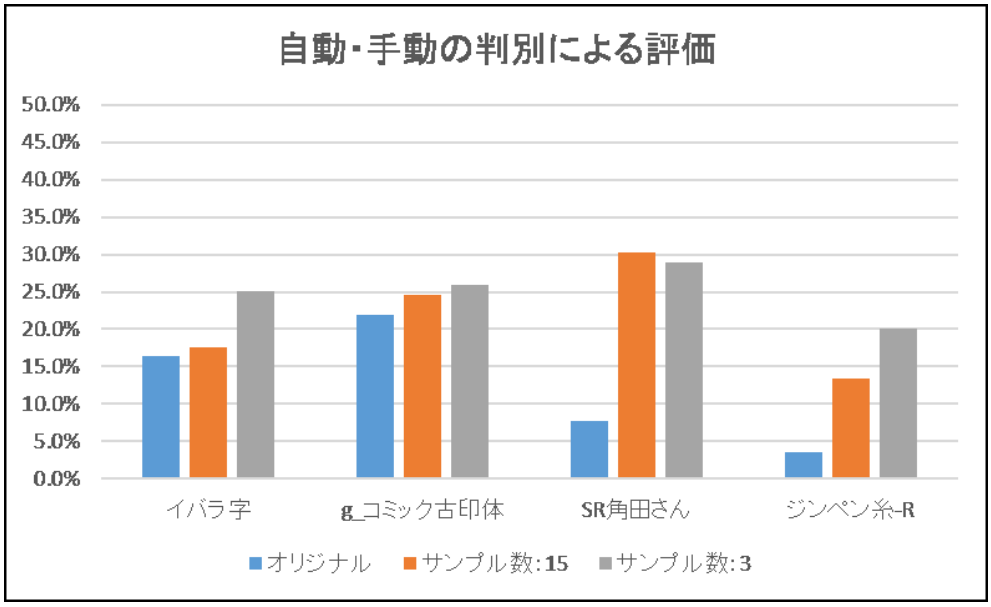


図 4.5: 自動・手動の判別による評価実験の結果

第5章

まとめと今後の課題

5.1 結論

本論文では, 不足しているカテゴリのパーツを補完する手法を提案した.

提案手法を用いて文字生成実験を行ったところ, サンプル数が少ないにもかかわらず非常にきれいな文字が生成できた. 骨格データは既存のものを使用したため, 純粋に少数サンプルからきれいな文字が生成できたわけではないが, つまりパーツの補完が上手く行えたことが確認できたということである.

また, 生成した文字に対して主観評価実験を行ったところ, 先行研究に並ぶほどの評価を得ることができた. フォントによってはオリジナルの文字とほとんど変わらないものもあった.

5.2 今後の課題

提案手法ではパーツの補完がかなり上手にできたので, それ以外の部分で課題が残されていないかを考える.

今後の課題としてまず挙げられるのは, 骨格情報のよりよい推定方法を考察することである.

先も述べた通り, 今回提案手法に関しては骨格情報の推定は特に行わず, オリジナルの骨格をそのまま利用した. これを少数サンプルにした場合, どのような結果になるか, またその結果を受けて今後どのようなアプローチをしていくかが, 考える余地のある内容だといえる.

また提案手法の課題点としてあげられていたのは, 骨格情報のマッチングの自動化である. 先行研究では手動で文字画像と骨格データのマッチングを行っているが, これは自動化を検討できる内容であるため, 今後の課題のひとつになりうるものだと考えられる.

謝辞

本研究を行うにあたり, 多大なるご指導を頂きました東北大学大学院工学研究科 大町真一朗教授に心より感謝致します.

研究を進めるにあたって, 研究方針や問題解決に関するアドバイスを頂いた東北大学工学研究科 菅谷至寛助教, 東北大学大学院工学研究科 宮崎智助教に心より感謝致します.

そして, 良い刺激を受けつつ日々の研究室生活を楽しく送らせていただいた大町研究室のメンバーの皆様に心より感謝致します.

参考文献

[1] W3C. Cascading style sheets (css) snapshot 2010. http://www.w3.org/TR/CSS/.

[2] グラフィック社編集部 (編集). TYPOGRAPHY 01 フォントをつくろう！グ ラフィック社, 5 2012.

[3] Adobe Systems Software Ireland Ltd. Adobe type. http://www.adobe.com/jp/products/type.html.

[4] 土屋達徳：“ パーツへの分解を用いた漢字フォントの自動生成に関する研究 , ” 2014 年度 修士学位取得論文, March 2015.

[5] Michael Kass, Andrew Witkin, and Demetri Terzopoulos : Snakes: Active contour models. Int. journal of computer vision, Vol. 1, No. 4, pp. 321-331, 1988.