部品合成による漢字スケルトンフォントの作成

田中哲朗[†] 岩崎英哉[†,☆]
長橋賢児[†,☆☆] 和田英一[†,☆☆]

多くの漢字は偏や旁などの基本的な部品の組合せからできている．本論文では，このことを利用して，漢字を具体的な座標を含まない抽象的な組合せ情報で定義しておき，プログラムによって部品を組み合わせてフォントを生成する方法を提案する．各部品は組合せによって大きさが変化するので，通常は様々な大きさの部品を用意しないと組み合わせて使うことができない．しかし，線の太さを自由に変えることのできるスケルトンフォントを使えば，あるサイズでデザインした部品をサイズを変更して使うことができる．そこで，本研究では論文1)で提案した複数書体に対応可能なスケルトンフォントの形式で部品のデザインを表現する．組合せの種類として，横方向，縦方向，片方の部品の空白部分に他方の部品を配置する嵌込みが考えられる．JIS X0208に含まれる全ての漢字をこの3種類の組合せで表現して，最初の2つの組合せを扱うだけでもある程度デザインコストを減らせるが，嵌込みを扱うと更に効果があることを確かめた．また，未知の漢字が現われた時に，既知の部品の組合せだけで表現できる可能性を，JIS X0212の漢字の何割がJIS X0208の漢字の部品で表現できるかによって見積もった．表現した漢字を表示，印刷するためには組合せアルゴリズムが重要である．そこで，一番容易な横方向の組合せを実現するために単純な方法をいくつか実装し，評価をおこなった．その中でもっとも良かった組合せアルゴリズムをもとに縦方向，嵌込みのプログラムを実装し，JIS X0208とJIS X0212のフォントセットを作成した．これが単純なアルゴリズムで作成されたものにもかかわらず，ある程度の品質に達していることを確かめた．

Making Kanji skeleton fonts through compositing parts

TANAKA TETSUROU,[†] IWASAKI HIDEYA,[†,☆]
NAGAHASHI KENJI[†,☆☆] and WADA EIITI[†,☆☆]

Most of Kanji characters consist of primitive parts, such as “hen”s and “tsukuri”s. We propose a font system which automatically generates Kanji fonts from their abstract joint definitions.

Although the size of parts have to be extended or reduced through the composition process, the use of skeleton fonts enables us to change their sizes without redesigning them for many sizes.

In this paper, we handle three types of composition, that is, horizontal, vertical and insertion compositions. We represented all characters in JIS X0208 by the combination of these three types, which reduces the design cost considerably.

We tried several horizontal composition methods to get better designed fonts, and extended the best method to vertical and insert compositions. Through these methods, we made a skeleton font set which includes all 12156 JIS X0208 and X0212 Kanji characters. Although the algorithm is simple, the quality of generated fonts is tolerably high.

1. はじめに

近年の計算機能力の向上により，日本語処理のための環境は飛躍的に整ってきた．文書印刷においては，高解像度のレーザープリンタの使用が一般的になり，使用するフォントもドットイメージフォントでの一種類の書体からアウトラインフォントによる多数の書体へと移行してきている．そのため，一般的に使われるJIS X0208の範囲内の漢字を使う限りは，以前の商業印刷に迫る品質の出力を個人ユーザレベルで得られるようになった．

しかし，古書や人名中のJIS X0208に含まれない文字や異体字を使うための環境は現在も貧弱である．ユーザがドットイメージやベクターデータでフォントをデザインする必要があるが，多数の文字を作成する場合は

† 東京大学工学部
Faculty of Engineering, University of Tokyo
☆ 現在，東京大学教育用計算機センター
Presently with Educational Computer Centre, University of Tokyo
☆☆ 現在，富士通研究所
Presently with Fujitsu Laboratories

フォントデザインのコストが膨大になる．JIS X0212が標準的にサポートされるようになっても，その5801文字に含まれない漢字も多数存在するのでこの問題は残る．

しかし，2章で示すように規格に含まれない漢字でも規格に含まれる漢字に使われている部品だけからなる場合が多い．本研究ではそのことを利用して，そのような漢字を部品の抽象的な組合せとして表現して，プログラムで自動的に部品を配置してフォントを作成することを試みる．

古くは明治初期に偏や旁などの部品ごとの活字を用意して活字職人が手で並べて使う試みがあるが，偏や旁の大きさが2種類（$\frac{1}{2}+\frac{1}{2}$ と $\frac{1}{3}+\frac{2}{3}$）しかなかったため，不自然な文字が多かった[2]．

本研究では，部品をストロークの骨組み（スケルトン）で表すことによって，この点を克服した．スケルトンで表現された部品ならば，線の太さを変えることなく部品の大きさを自由に変更することができる．組み合わせる部品の大きさや配置は組合せの種類と部品のスケルトンデータからプログラムで決定する．

以下，2章では部品の表現に用いたスケルトンデータについて説明し，3章では漢字の分解の基準と，分解の作業量，扱う分解によるフォントデザインのコスト軽減の見積りをおこない，フォントの製作コストを大幅に軽減するには，横方向の組合せ，縦方向の組合せだけでなく部品の中に別の部品が入り込む嵌込みも扱う必要があることを示す．

4章以下は，組合せデータからプログラムによって部品の具体的な配置情報を決定する方法について論ずる．

4章では横方向の組合せのアルゴリズムを単純なものいくつかを実装し，評価をおこなう．そこで良好と判断されたアルゴリズムを5章では縦方向の組合せに，6章では嵌込みのアルゴリズムに拡張する．

7章では，これらのアルゴリズムに従って製作したフォントの評価をおこない，8章でまとめと今後の課題に関して論ずる．

2. スケルトンデータの表現

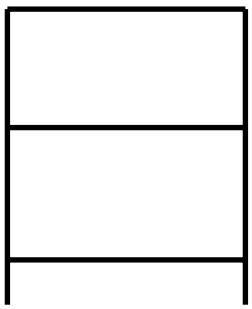
フォントの表現には，一般にドットイメージとアウトラインが用いられる．近年，高解像度の出力装置が広まるにつれてアウトラインフォントが一般的に使われるようになってきている．

しかし，ドットイメージもアウトラインのとちらも，文字をストロークとしてではなく黒白の2値画像として扱っているので，線の太さを変えずに文字の大きさだけを変えることができない．これらのフォントを部品の表現に用いると，偏と旁で線の太さが変化するなどバランスの取れない文字が出来てしまう．しかし，部品の大きさによって別々のデータをデザインするのはデザインコスト削減の立場から好ましくない．

本研究では，論文1）で筆者らが提案したスケルトンフォントで部品を表現することによって，この問題に対応する．これはストロークを16種類のエレメントに分類し，フォントをエレメントの位置を決める制御点（エレメントの種類に応じて2から4個指定する）と接続情報からなるスケルトンデータで表現し，プログラム内付けによって，明朝体やゴシック体など多様な書体のフォントを生成するというものである．

スケルトンフォントでは，線の太さは部品の大きさと独立なので，部品を任意の大きさ，任意の場所に配置することができる．制御点を動かすことにより任意の変形が可能だが，本研究では部品の縦横方向の拡大と平行移動のみを扱う．

論文1）のシステムはUtiLisp[3),4)]によって記述されていて，漢字は図1のような形式のLispのS式で表現される．S式の各部分はそれぞれ以下のような意味を持つ．



```
(defprimitive nil 日
   '(((316 50)(316 394)(312 350)(311 191) ; (1)
      (87 50)(87 390)(87 350)(87 191))
     ((tate (0 1) ; (2)
            (link 2 3)) ; (3)
      (tate (4 5) (link 6 7))
      (yoko (6 2))
      (yoko (7 3))
      (yoko (4 0)))
))
```

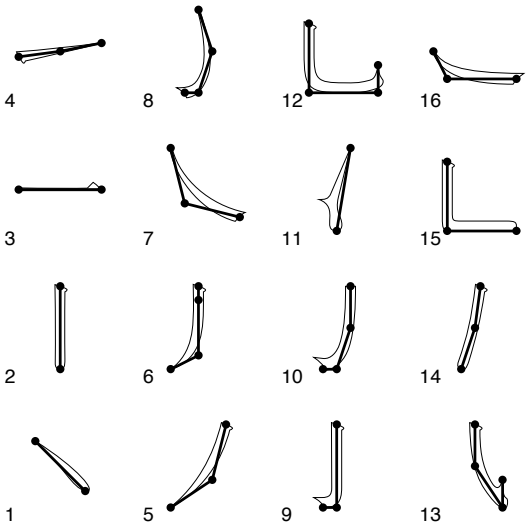
図1 スケルトンデータの表現
Fig. 1 Representation of skeleton data

(1) 制御点の座標

スケルトンデータ中のすべての制御点のX, Y座標の対のリスト．デザインする際にX-window上のエディタを使う関係で，X, Y座標の範囲は0から399までとする．

(2) 各エレメントの種類と制御点との対応

エレメントの種類は，tate, yokoなどのLispシンボルで表す．エレメントは種類に応じていくつかの制御点を持つ．図2で本システムで扱う16種類のエレメントを示す．図中の黒丸が制御点を表す．



1. ten, 2. tate, 3. yoko, 4. migiue, 5. hidari, 6. tatehidari, 7. migi, 8. kozato, 9. tatehane, 10. tsukurihane, 11. sanzui, 12. kokoro, 13. tasuki, 14. magaritate, 15. kagi, 16. shin-nyuu

図2 エレメント
Fig. 2 Elements of "Mincho" type faces

(3) 各エレメントの属性リスト

エレメントの肉付けプログラムに渡される属性リスト．エレメントの途中に接続する点を記述するlinkという属性は，エレメントの肉付けプログラムからは無視され，セリフ(エレメントの始点終点や連結部分につく飾り)の決定に用いられる．

(4) 属性リスト

スケルトンデータ全体の属性の定義に用いる．

スケルトンデータは，X-window上で対話的に編集するスケルトンエディタ[5]を用いて作成する．慣れた作業者がおこなうとエレメント数が10前後のスケルトンデータを5分程度でデザインすることができる．

部品の大きさの変更は(1)の制御点にアフィン変換を施すことによっておこなう．

3. 漢字の分解と部品組合せ

漢字を部品に分解することによって，デザインコストの削減に役立つことは直感的には明らかだが，本章ではどのような分解を扱えば効果があるかを見積もり，扱う分解の種類を決定する．

分解の基準として以下の前提条件をもうける．

(1) エレメント間の交叉や接続が存在するものは分解しない

「見」を「目」と「にんにょう(禿の下)」の組み合わせで定義すると，2つの部品間の接続が生ずる．部品間の接続を組合せで実現するためには，接続の位置を考慮した多種類の組合せを用意する必要があるので，このような分解はおこなわない．

(2) 組合せは抽象的なデータ(組合せの種類＋部品のリスト)だけで表現する

「れんが(然の下)」，「さんずい」のように点を微妙に配置する必要がある部品，「三」のように横棒の幅を変える必要がある部品を組合せで実現するためには，補助的な情報が必要になると思われるので，このような分解はおこなわない．

この方針にしたがってJIS X0208に含まれるすべての漢字(6355文字)を分解した．この作業はすべて人手でおこない，約2ヶ月人を要した．組合せは次の3種類に分類された．ただし，2番目の条件に人間の主観が入るので結果に一意性はない．

- 横方向
  「波」，「街」のような組合せ．組み合わせる部品数は最大3個(「街」など)．．
- 縦方向
  「栗」，「志」のような組合せ．組み合わせる部品数は最大5個(「櫚」の右側など)．
- 嵌込み
  「国」，「間」などの「構えと旁」や，「雁」，「虎」のような「たれ」，「道」，「処」のような「にょうと旁」などの，片方の部品の中の空白部分に他方の部品が入るような組合せ．組み合わせる部品数は最大3個(「巫」など)．

JIS X0208の漢字の最上位層がどのような組合せで表現されるかを表1に示す．

直感的には組合せプログラムの実現の困難さは，横方向 < 縦方向 < 嵌込みであると予想される☆．実現の困難な組合せからなる文字は分解しないとすると，ど

☆ われわれの用いたアルゴリズムでは実際にもそうなった

表1 最上位の組合せ (JIS X0208)
Table 1 Joint types of top level components(JIS X0208)

組合せ	横方向	縦方向	嵌込み	分解不能
文字数	3862	1584	551	358

表2 扱う組合せとデザインコスト (JIS X0208)
Table 2 Joint types and design cost(JIS X0208)

分解	プリミティブ数	エレメント数	プリミティブあたりのエレメント数
(1)	6355	90967	14.3
(2)	3160	39017	12.3
(3)	1532	14161	9.2
(4)	975	6022	6.2

表3 分解の範囲と作成可能性 (JIS X0212 5801 文字)
Table 3 Potential of generatng Unknown characters(JIS X0212)

分解	作成可能漢字数	割合 (%)	新たに必要な部品数	エレメント数
(1)	0	0	5801	91039
(2)	3722	64.2	1968	28818
(3)	5029	86.7	613	8189
(4)	5543	95.5	121	791

こで区切るかによって, (1) 組合せなし, (2) 横方向のみ, (3) 横方向＋縦方向, (4) 横方向＋縦方向＋嵌込み の4種類の表現法が考えられる. これを評価するため, 各表現法によってJIS X0208の6355文字をデザインする際の, デザインすべき部品数 (それ以上分解できない部品を以下ではプリミティブと呼ぶ) 及び, 総エレメント数を求めた (表2).

これにより, (2) でも大幅なデザインコストの削減を達成できるが, (4) では更にプリミティブ数で1/3, エレメント数で1/6ほどに減ることが分かる.

また, 本研究の目的である規格外漢字の表現能力を調べるために, JIS X0212の5801文字も同様に3種類の組合せに人手で分解した. こちらの作業も約2ヶ月人を要した. JIS X0208に使われる部品セットを元にJIS X0212の漢字をどの程度作れるかと, 部品セットをどれだけ増やすとすべての漢字を作れるかを, 先ほどの4種類の表現法について調べてみた結果を表3に示す. JIS X0212は偏や旁の字体などJIS X0208と関連が深いので, 規格外漢字のサンプルとするには不適当な面もあるが, 大体の目安になる.

以上により, 縦方向, 横方向, 嵌込みの3種類の組合せを扱えばかなりの規格外漢字を表現でき, 表現できない場合も, 新たにスケルトンデータでデザインする部品の数が大幅に減らせることが見積もれる. そこで, 本研究では3種類の組合せを扱うことにする. 組合せで表現される漢字は下のように階層的なLispのS式として扱う.

鯵 ⇒ (yoko うおへん 参)
参 ⇒ (tate むのじ大 かしらさんづくり)
うおへん ⇒ (tate くのじ田 四点)
超 ⇒ (hamekomi そうにょう 召)
召 ⇒ (tate 刀 口)

組合せによって定義された部品の組合せデータから, 以下の方法でスケルトンデータを生成する. 組合せプログラムもスケルトンフォントの肉付けシステムと同様にUtiLispで記述する.

(1) 組合せの種類に対応するシンボル (tate, yoko,hamekomi) の属性リストに収めた組合せプログラムを取り出す.
(2) その部品を構成する各部品のスケルトンデータ (部品が組合せで実現されている時は再帰的に適用する) を得る.
(3) 2で得られたスケルトンデータのリストを引数として, 1のプログラムを呼び出す.

組合せの種類に応じた3種類の組合せプログラムの実現法を次章以下で検討する.

4. 横組合せのアルゴリズム

この節では, 横方向の組合せのいくつかの単純なアルゴリズムを提案し, 評価をおこなう. 説明では2つの部品からなる組合せを用いるが, それより多くの部品からなる組合せも扱えるようにプログラムを作成する.

部品の配置の自由度は, 幅, 高さとX, Y方向の移動の4だが, 横方向の組合せでは, 幅とX方向の移動のみを扱う. 高さと上下移動は部品固有のデータとして固定する. 横方向の組合せにおいては, 部品の高さの変化と, 上下の移動を必要とする文字も存在し, 商用レベルのフォントを作成する場合は避けてはならないが, 本研究では現時点ではそこまでの品質は必要としていないので扱わない.

したがって, 横組合せのアルゴリズムは組み合わせる部品の種類によるそれぞれの部品の幅の決定と部品間の間隔という独立な2つの要素によって決められる. 以下で, それぞれに関して簡単なアルゴリズムをいくつか提案し, 評価をおこなう.

4.1 部品の幅

横方向の組合せで漢字を生成する場合の部品の幅に関しては, 2) などに述べられているように, 次のような規則が成り立つ.

規則1.複雑な部品ほど幅が広い
規則2.縦棒の間隔はほぼ一定

これらの規則を反映させた幅の決定法として以下の2

種類のアルゴリズムを試みる.

アルゴリズムA線密度一定

規則1に注目する. 文字の複雑度の定義として自然に考えられるのが, 文字の面積当たりのストローク長 $\frac{\Sigma L_n}{(X_{max}-X_{min})*(Y_{max}-Y_{min})}$ (L_n は n 番目のストロークの長さ, X_{max} はストロークのX座標の最大値, 以下同様に定義) を用いる方法である. ストロークの長さとしては近似的に制御点を結んだ線分の長さを用いる. 以下, 面積当たりのストローク長のことを線密度と呼ぶ.

線密度を左右の部品で等しくなるようにそれぞれの幅を決定すると, 複雑な部品ほど広い幅になると考えられる. 部品の線密度を揃えるために, 同じ幅の組合せから始めて, 線密度の大きい側を広く, 逆を狭くして再び組み合わせるという反復計算を収束するまでおこなう.

アルゴリズムB縦棒間隔一定

規則2に注目する. 部品中に含まれる縦棒のX座標の差の最小値を基準幅として, 左右の部品の基準幅を揃えるように幅を決定する. 部品中に複数の縦棒がない場合は人間が判断した複雑度に基づき, 部品ごとに基準幅を設定する

さらに, 同じ部品の組合せで定義される文字(「弱」, 「朋」など) のフォントを見ると右側の部品の幅が1割程度広くデザインされている事実から,

規則3.左側の部品よりも右側の部品の方が大きめ

という規則も成立するので, それに従った幅の補正を両方のアルゴリズムについて適用する.

4.2 間隔の決定

部品間の間隔を決定するアルゴリズムを作成する場合の基礎となる経験的規則は次のようなものがある.

規則4.ストロークの交叉がない

部品への分解の前提条件から導かれる. 左払いや右上はねの終点や横棒の始点などが縦棒と重なることはあるが突き抜けることはない.

規則5.縦棒の間隔はほぼ一定

左側の部品に含まれる縦棒と, 右側の部品に含まれる縦棒の間隔は, それぞれの部品の中における縦棒の間隔 (基準幅) よりも狭くはならない.

これらの規則を考慮して, 次の2種類のアルゴリズムを試みる.

アルゴリズムC外接長方形の接触

それぞれの部品の外接長方形同士が接触するように左右の部品を配置する. ただし, 肉付けの結果の外接長方形を得るのは手間がかかるので, 部品中のエレメントの制御点の外接長方形を用いる. 外接長方形が交わらない限りエレメントの交叉もないので規則4は満たされる. 規則5に関しては, 縦棒が外側に来る部品は外接長方形を人手で大きめに補正することである程度対応が可能である.

アルゴリズムDエレメント間制約

規則4は, それぞれの部品を構成するエレメントの制御点を結んだ線分が交叉しないという条件で近似する. 更に, 規則5を実現するために, 縦棒同士を基準幅程度離すなどエレメント間の制約条件をif-then規則として記述する.

規則は, 以下のようなLispのS式で記述する.

```
(deflimit ( elements1  elements2)
  ( condition1  rule1 )
  ( condition2  rule2 )
  ...
)
```

elements1 は片方の部品に含まれるエレメントの種類を, *elements2* は他方の部品に含まれるエレメントの種類をあらわす. これらのエレメント間に *condition* が満たされた時の制約条件を *rule* で記述する.

以下に例を示す. x_{ij} は i 番目の部品中のエレメントの j 番目の制御点のX座標を表し, xunit は基準幅を表す.

```
(deflimit  ((tate tatehidari tatehane
             tsukurihane kagi)
            (tate tatehidari tatehane
             tsukurihane kagi))
 ((or (<= y00 y10 y01)(<= y00 y11 y01)
      (<= y10 y00 y11)(<= y10 y01 y11))
  (>= (diffabs (+ x00 x01) (+ x10 x11))
      (* 1.6 xunit))))
```

この例は, Y座標で重なりがある縦棒と縦棒の間は, 基準幅の0.8倍以上間隔を開ける必要があることを定義している. *condition* と *rule* には2次までの不等式及び, その否定, 論理和, 論理積などが記述できる.

制御点の座標が $\mathbf{x}+t\mathbf{p}$ の形で表された際に, 制約を満たす t の範囲を求める簡単な数式処理プログラムを作成した. そのプログラムを使って, 右側の部品をX方向に移動していって左側の部品との制約を満たす範囲でもっとも近付くように配置する.

4.3 評　　価

提案したアルゴリズムを評価するために, 以下の4種類の方法を実装した.

(1) 配置固定

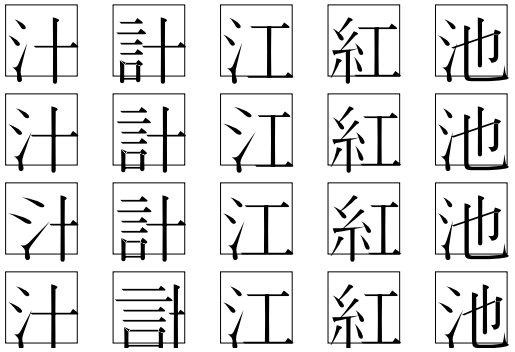


図3 テストシートの一部
Fig. 3 Sample of test-sheets

表4 評価結果
Table 4 Results of horizontal joint

	1	2	3	4
判別困難	2	24	1	0
著しく汚い	75	170	62	59
水準以下	204	174	155	134
許容範囲内	195	115	228	257
満点	24	17	54	50
計	500	500	500	500

部品ごとに配置位置の情報を定義しておいて, 組み合わせる相手の部品に関係なく同じ位置に配置する. 部品の配置位置は, 人手で調節しておく.

(2) 線密度一定＋外接長方形(アルゴリズムA+アルゴリズムC)

(3) 基準幅一定＋外接長方形(アルゴリズムB+アルゴリズムC)

(4) 基準幅一定＋エレメント間制約(アルゴリズムB+アルゴリズムD)

これらの方法で, 2つの部品の横方向の組合せで実現される文字100文字を作成しランダムに並べたテストシート(図3はその一部)を作成した. このテストシートをフォントの専門家ではない通常のユーザ5名に提示し, 次の5段階の基準で一文字ずつに点数をつけてもらった. その結果を表4に示す.

(1) 判別困難

(2) 著しく汚い

(3) 印刷用の水準には達しない

(4) 印刷用フォントとしての許容範囲内

(5) 満点

被験者によって点の付け方の基準に違いがあり, また文字によって順序が入れ替わることがことがあったが, 本研究が目標とする許容範囲内と満点の評価を得た文字数で見ると, おおむね$2 < 1 < 3 < 4$という評価になった.

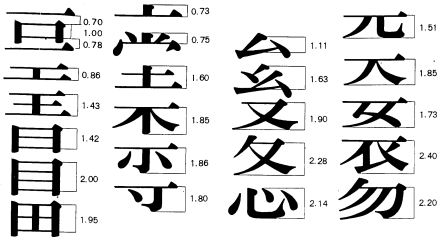


図4 パターンと高さとの対応(2)より引用)
Fig. 4 Relations of patterns and heights

2の評価が低いのは, 図3の「計」の字の一番下のように, 組み合わせる部品の横棒の数に違いがあると極端にバランスがくずれた文字が作成されてしまうためである. 線密度の考え方を縦横異なる比率での拡大/縮小に用いるのは不適当なようだ.

3と4は同じ評価がついた文字が多かった. ただ, 「白」の外接長方形を狭く設定すると, 「伯」の字が接近し過ぎ, 広く設定すると「拍」が離れ過ぎるなどの理由で, 水準以下や著しく汚いと判断された文字が3の方が多かった.

以上の評価結果に基づき, 本システムでは横方向の組合せに4の方法を採用した. 縦方向や嵌込みの組合せアルゴリズムも, 4の方法の拡張で実現する.

5. 縦方向の組合せ

縦方向の組合せは, 以下の手順によって行なう.

(1) 両方の部品の中心を合わせる

(2) 部品の幅を決める

(3) 部品の高さの比率を決定する

(4) 上下の部品の間隔を決定する

このうち1, 2は部品ごとに固有なデータであり, 部品の属性として実現しても良かったが, デザインコストの削減のために, プログラムで決定することにした.

3では横方向の縦棒間隔と同様に, 横棒の間隔(基準高さ)をそろえることによっておこなう. ただし, 基準高さの決定は横棒の間隔だけでなく, それ以外のエレメントも含めたパターン(図4)とプログラムでマッチングを取り決定する. 部品中にマッチする部分がなく決定できない場合は人間が判断して与える.

4はエレメント間の制約条件を満たす範囲で下の部品を上の部品になるべく近づけることで実現する. 以下では縦方向の組合せに固有の1, 2の問題について述べる.

5.1 中心あわせ

縦方向に組み合わせることができる部品は左右対称に近い形をしたものが多く, 対称な部品同士を組み合わせる場合には対称の中心を合わせる必要がある. すべての

(a) 重心を中央にした失敗例

(b) 対称の中心を用いた例

図5 中心合わせ
Fig. 5 Centering

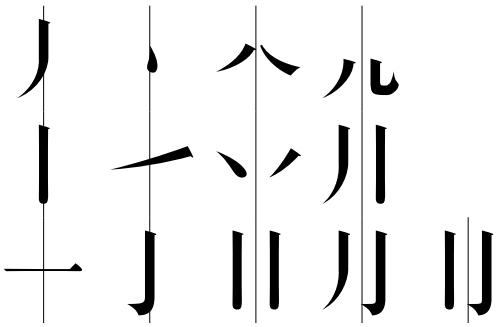


図6 対称とみなすエレメント，エレメント対
Fig. 6 Symmetric elements

対称なプリミティブについて部品の中心を定義するのは手間がかかるので，プリミティブのデータからプログラムによって対称中心を求める．対称の中心の近似としてストローク全体の重心を取って，そのX座標をそろえるように組み合わることを試みたが，図5の(a)のように失敗に終わった．

そこで，別の方法を採用した．まず，対称となりうるエレメント及びエレメントの対をテーブルとして用意する(図6)．プリミティブのすべてのエレメント，エレメント対についてそれが対称とみなせて，その対称の中心と重心との偏差がある程度以下のものを集め，それらの対称の中心の平均をそのプリミティブ全体の中心と定義する．以上のように定義した中心をそろえて作成した文字が図5の(b)である．

このアルゴリズムによって，正しい中心を求めることができないプリミティブに関しては人手でプリミティブの属性リストに中心のX座標を記述する．左右対称なプリミティブ488個のうち，この修正を必要としたプリミティブは24個だった．

合成の結果得られた部品は，合成に用いられた部品がすべて対称で，対称の中心がすべて等しい場合に限って対称とする．

(a) 幅の調節をしない場合

(b) 幅の調節をした場合

図7 幅の調節の有無
Fig. 7 Adjustment of width

5.2 幅の決定

縦方向の組合せでは，中心合わせと同様に幅の決定も重要である．部品の外接長方形が漢字の外枠一杯に接するように幅を広げると，図7の(a)のようにバランスの崩れた文字になってしまう．部品の幅に関しては，縦棒のような縦方向の構造線の数が多い部品の幅を広くすると，バランスの良い文字になるという経験的規則があるので，それを反映させて以下のようなアルゴリズムを採用した．

まずxlimitという仮想的なエレメントを用意して，xlimitと実際のエレメントとの位置関係の制約条件を以下のように宣言する．

```
(deflimit mincho
  ((tate tatehidari kokoro tatehane
    tsukurihane kagi tasuki magaritate)
    xlimit)
  ((or (<= y00 y10 y01)(<= y00 y11 y01)
       (<= y10 y00 y11)(<= y10 y01 y11))
   (>= (diffabs (+ x00 x01) (+ x10 x11))
       xunit)))
```

この定義は，xlimitという仮想的なエレメントと，縦線を含むエレメントが横に並ぶ際には，その縦棒とxlimitとの間に基準幅(xunit)の半分だけ離して配置する必要があるという制約を表している．

縦方向の組合せをする際は各部品について左右からxlimitを限界まで近付けておいて，そのX座標を調べる．縦方向の移動を行なう前に各部品の左右のxlimitが縦に並ぶように横方向の拡大/縮小，移動を行なう(図8)．

部品全体の幅に比して縦棒の数が多い部品は基準幅が小さくなるので，xlimitがより近くまで近付き，部品の幅が広くなる．このことは，図7の(b)で確かめることができる．

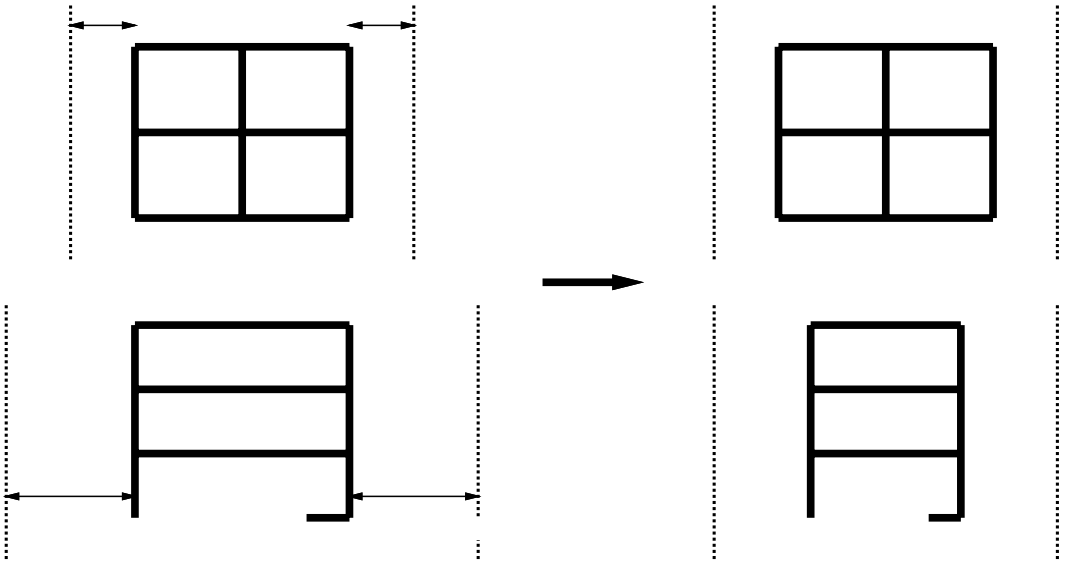


図8 幅の調節
Fig. 8 Process of adjustment of width

6. 嵌 込 み

嵌込みは片方の部品中の領域の一部に他方の部品を配置することによって実現する．どの領域に埋め込むかは，以下のようにプリミティブの定義の際に嵌込みの種類 (kamae) をタグとして属性リスト中にあらかじめ入れておく．

```
(setq くにがまえ
  '(((39 366)(361 366)(38 32)(38 392)
     (361 32)(361 392))
    ((yoko (0 1))(tate (2 3)(link 0)
     (tate (4 5)(link 1)(yoko (2 4)))
    (kamae 39 32 361 366)))
```

この領域に他方の部品を位置関係の制約を満たすようになるべく大きく配置することを目標とする．横方向の配置と違って自由度が 4 なので，制約条件を解く数式処理プログラムで直接扱うことが出来ない．そこで，以下のような反復演算で実現する (図 9)．

(1) 空白部分の中心に埋め込む部品を置く (a).
(2) 埋め込んだ部品に X, Y 等倍の拡大 / 縮小を施して制約をみたす限界を求める (b).
(3) 限界より小さめ (0.9 倍) にに拡大し，X 方向，Y 方向にとの程度平行移動できるかチェックする (c).
(4) それに応じて，空白部分を X, Y 方向独立に拡大し，中心位置をずらす (d).

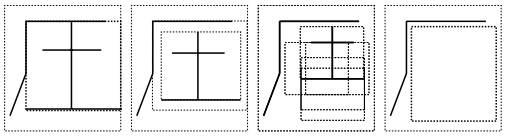


図9 嵌込みの組合せの手順
Fig. 9 Composition of including

(5) この結果，最初の繰り返しを除いて，ステップ 1 の時点よりも部品の面積が大きくなることが期待される．面積が小さくなったり，面積の伸びが 10% 以下になれば終了，それ以外の時はステップ 1 に戻る．

縦方向，横方向の組合せと比較して部品の移動，拡大の回数が多いので，組合せには数秒程度の時間がかかることもある．また，最初に制約を満たすように部品を置くことができれば，この反復演算はかならず収束する．

7. 評 価

以上の方針にしたがって，JIS X0208, 0212 に含まれる 12156 文字について試作をおこなった．プリミティブ 1095 個のデザインには約 3ヶ月人を要した．試作した文字のうち，JIS X0208 第一水準の文字からランダムに選び出した文字 100 文字☆のテストシート (図 10)

☆ 規格外の文字を製作するという意図から言えば JIS X0212 の文

図10 漢字のサンプル
Fig. 10 Samples of generated characters

を与え, 4章と同じ 5 名の被験者に 5 段階で評価してもらった. その結果を表 5に示す.

横方向の組合せだけのものと比べると全体的に評価が下がっているが, とりあえず判別困難と判断された文字はなかった. また, 被験者 2 は特に厳しい判断を下したが, それを除くと著しく汚いと判断された文字は少ない. 印刷用フォントとしての許容範囲内とされた文字がほぼ半分しかなかった点は不満が残るが, ある程度の品質には達しているとしてもよいだろう.

上の評価では 2 と評価された文字など組合せ結果に不満が残る文字については, 組合せデータ中の部品に次のようなアノテーションを人手で付加して対応する.

(xmove 移動量 部品) ; ただし移動量は部品の大きさに対する相対値
(ymove 移動量 部品)
(xscale 倍率 部品)
(yscale 倍率 部品)

これを用いて, 横方向の組合せの際に上下の移動も必要な文字「明」は次のように修正される.

```
(setq 明
 '(yoko (ymove -0.10 (yscale 0.95 にちへん)) 月))
```

図 11の左側が修正前の組合せ結果, 右側が修正後である.

8. おわりに

部品に分解して表現した場合の, デザインコストの削減に関しての見積りをおこない, かなり効果があることが判断できた. そして, 部品の組合せ情報からフォントをデザインするアルゴリズムを実現した. 得られたフォントの質には不満が残るものの, 新たな漢字を少ない手間で製作するための手段としては目的を果たしたと言えるだろう.

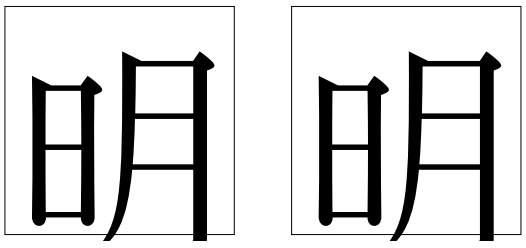


図11 アノテーションによる修正
Fig. 11 Modification with annotations

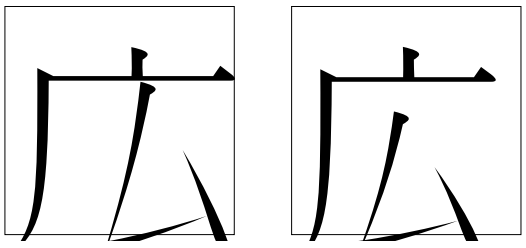


図12 アルゴリズムによる組合せの失敗例
Fig. 12 Bad sample of algorithmic composition

しかし, 個人の使用だけでなく印刷水準に達するような, 高品質のフォントを作成するためにはまだまだ多くの課題が残っている. 特に嵌込みに関しては, 不満の残る文字が多かった. 特に, 嵌込まれる部品の画数が少ない場合は, 図 12の左のように外側の部品と接近しすぎたり, 離れすぎたりなどでバランスの悪い組合せが生じることが多かった. このような文字は, アノテーションで右のように修正する必要がある.

アノテーションの付加なしに高品質の文字を生成するためには組合せアルゴリズムの改善が望まれる. また, 作成されたフォントを用いて文章を印刷する際に, できた文字の一文字一文字に問題がなくても文章中で並べると, 不自然に感じることがしばしば観察された. この方面の研究も必要になるだろう.

なお, 本システムによって作られた JIS X 0208, X0212 のアウトラインフォントは ftp.ipl.t.u-tokyo.ac.jp から anonymous FTP で入手することができるようになっている.

参考文献

1) 田中哲朗, 石井裕一郎, 竹内幹雄, 和田英一: プログラム内付けによる複数漢字書体間のスケルトンデータの共有, 情報処理学会論文誌 Vol. 36, No. 1, pp. 177-186(1995).

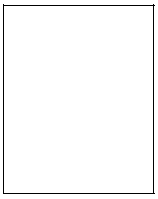
字からサンプルを選ぶべきだが, デザインの可否を判定するのには見慣れた文字でないと難しいと判断した

表5 総合評価
Table 5 Results

	被験者1	被験者2	被験者3	被験者4	被験者5	計
判別困難	0	0	0	0	0	0
著しく汚い	2	35	1	12	3	53
水準以下	26	44	41	43	37	191
許容範囲内	40	21	57	45	56	219
満点	32	0	1	0	4	37

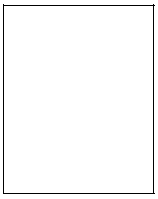
2) 佐藤 敬之輔：漢字 (上, 下), 丸善 (1973).
3) 近山隆：UtiLisp システムの開発, 情報処理学会論文誌, Vol. 24, No. 5, pp. 599-604(1983).
4) 田中哲朗：SPARC の特徴を生かした UtiLisp/C の実現法, 情報処理学会論文誌, Vol. 32, No. 5, pp. 684-690(1991).
5) 石井 裕一郎：Lisp による漢字フォント作成支援ツールの開発, 情報処理学会記号処理研究会資料, SYM 61-4 (1991).

(平成6年4月11日受付)
(平成7年6月12日採録)



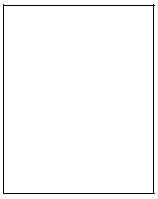
田中 哲朗（正会員）

1965 年生まれ．1987 年東京大学工学部計数工学科卒業．1992 年同大学院博士課程修了．博士 (工学)．東京大学工学部助手．記号処理言語, 漢字フォント, ゲームプログラミングに興味を持つ．ACM, 日本ソフトウェア科学会各会員．



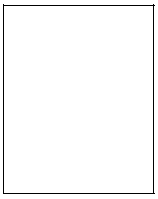
岩崎 英哉（正会員）

1960 年生．1983 年東京大学工学部計数工学科卒業．1988 年同大学院工学系研究科情報工学専攻博士課程修了．同年東京大学工学部計数工学科助手．1993 年 4 月より東京大学教育用計算機センター助教授．工学博士．記号処理言語，関数型言語，並列処理システムなどの研究に従事．日本ソフトウェア科学会，ACM 各会員．



長橋 賢児（正会員）

1966 年生．1990 年東京大学工学部計数工学科卒業．1993 年同大学大学院工学系研究科情報工学専攻修士課程修了、(株) 富士通研究所入社．以来ソフトウェアリバースエンジニアリング技術の研究開発に従事。プログラミング言語処理系、オブジェクト指向方法論などに興味をもつ．



和田 英一（正会員）

昭和 6 年生まれ．昭和 30 年 東京大学理学部物理学科卒業, 32 年 同大学院修士課程終了, 小野田セメント (株) 調査部統計課を経て昭和 39 年から東京大学工学部計数工学科助教授, 52 年同教授．この間昭和 48 年から 49 年まで M.I.T. 電気工学科助教授 併任．平成 4 年東京大学退官, 現在富士通研究所常任顧問, 東京大学名誉教授, 工学博士．情報処理学会, 計量国語学会, ACM 各会員