

A Heuristic Search Approach to Chinese Glyph Generation Using Hierarchical Character Composition

PAK-KEUNG LAI*, DIT-YAN YEUNG, and MAN-CHI PONG**

Department of Computer Science

Hong Kong University of Science and Technology

Clear Water Bay, Kowloon, Hong Kong

dyyeung@cs.ust.hk

*Department of Law, University of Hong Kong,

Pokfulam Road, Hong Kong

**Computer Centre, University of Hong Kong,

Pokfulam Road, Hong Kong

Abstract: When some Chinese characters used by the user are not represented in the standard character encoding scheme of a computer system, it would be desirable if the system can still support these characters. Based on the hierarchical relationships between character components, we propose structurally based character composition expressions (CCE) for composing these unrepresented Chinese characters unambiguously. We then describe a glyph generation process, which generates the corresponding glyph for a given CCE. The process makes use of glyph beauty evaluation metrics which are based on traditional Chinese calligraphy rules, and uses a beauty evaluation function to guide the search for nice-looking glyphs through iterations. Our approach can be applied to glyph composition components that correspond to bitmap, vector as well as outline base fonts, as compared to earlier work which can only use parametric base fonts [1]. A prototype system has been implemented to demonstrate our approach.

Keywords: Chinese font, Glyph generation, Character composition, Evaluation function, Heuristic search, Greedy algorithm.

1. Introduction

All commonly used Chinese character encoding schemes (e.g., GB and Big5) assign character codes to only a limited number of characters. Although Unicode [2] was designed in an attempt to represent “all” characters in the world, its two-byte format can represent at most 65,536 characters, yet the total number of characters in the world continues to increase.

In some applications, infrequently used and user-defined characters that are not represented in the adopted encoding scheme are encountered. It is necessary to come up with a satisfactory scheme for accommodating unrepresented characters unambiguously. Some Chinese systems allow the user to define the glyph for a new character and then assign to it a code from the unused code space. However, the encoding conflict problem may arise: the same internal code in two different systems may represent two different characters (and hence two different glyphs). As a consequence, a document created in one system may not be interpreted correctly in another system, even though both systems use the same encoding scheme for the standard set of characters.

(Received April 9, 1995, in revised form May 30, 1996, received for publication October 19, 1996.)



We have proposed to use character composition expressions (CCE) to encode unrepresented characters by specifying the structural relationships between character components [3]. Based on the use of CCE, this paper presents a method for generating Chinese glyphs, with the generation process formulated as a heuristic search problem. A prototype glyph generation system has been implemented. Although our current implementation is based on the GB encoding scheme, the techniques used are quite general and hence can be applied to other encoding schemes for both simplified and traditional Chinese characters.

The remainder of this paper is organized as follows. The terms used in the paper are defined in Sec. 2. Section 3 briefly describes the use of CCE for representing Chinese characters. Section 4 describes the generation of glyphs and the quantitative evaluation of glyph beauty. Section 5 presents some examples of generated glyphs, and the last section compares our approach with related work and discusses our future work.

2. Some Definitions

A character is an abstract symbol that represents some concept or object. The graphical representation of a character is called its glyph. In digital typography, each glyph is usually represented as a 2-dimensional bitmap. The position of a pixel in the bitmap can be represented with respect to some reference point in the bitmap. The encoding of a character under a certain encoding scheme is called its character code, or simply code, in that encoding scheme.

A Chinese radical is a simple graphical pattern composed of multiple strokes. There is no standard radical set for Chinese characters. Discussions on the evolution of radicals in Chinese dictionaries can be found in [4].

Some radicals can appear in more than one form. For example, the radical 火 has two different forms, 火 and 灬. The 灬 form can be considered as a variant of the 火 form. Historically, the two forms were used interchangeably in the composition of some characters. In this paper, these two forms will be treated as two different radicals as far as character composition is concerned.

Some radicals are also characters by themselves. For example, the radicals 火 (fire), 口 (mouth) and 气 (gas) are characters with their own meanings. These radicals (or characters) are also called radical characters.

A character component may be a character or any structural part of a character. Its glyph can be used for the composition of more complex glyphs. All radicals are character components.

The bounding box of a glyph is a rectangular region bounding the glyph. The bounding box includes the white borders, if any, surrounding the black pixels of the glyph. The tightest bounding box of a glyph is the extent of the black pixels of the glyph, excluding the white borders surrounding the black pixels. See Fig. 1 for illustration.

The bounding box of the glyph of a character component can be represented as (x_1, y_1, x_2, y_2) , where (x_1, y_1) and (x_2, y_2) correspond to the upper left and lower right corners of the



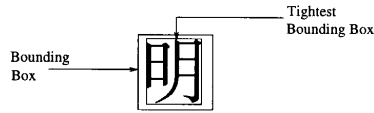


Figure 1. Bounding box and tightest bounding box.

	1 component	2 components	3 components	4 components
One-element	田 月 母			
Horizontal composition		伸 张 肌	湘 沼 邵	洲 祿 湖 朔 碗 鹁 溜 鄴 慢 鶉 疑
Vertical composition		宙 男 昌	曼 茄 盟	蔓 荔 葬 筭 露 簷 蔽 照 薇 樊 琵
Surrounding composition		历 习 旭 冈 凶 区 回	蔗 廂 逞 迳 司 闾 阔 匿 甌 圆 囧	Not shown

Figure 2. Structural categorization of Chinese characters.

bounding box, respectively. The origin (0, 0) is the upper left corner of the bounding box of the character glyph (of size x_{dim} by y_{dim}) to be composed from the glyphs of the character components.

3. Chinese Character Composition




3.1 Structure of Chinese characters

The structural categorization of Chinese characters has been studied by many researchers. Figure 2 (adapted from [5]) shows one scheme of classifying Chinese characters based on the character components. There are basically four different types, namely, one-element, horizontal composition, vertical composition, and surround composition. One-element characters are those characters that cannot be further decomposed.

Note that character composition can also be considered in a hierarchical manner. For example, instead of horizontally pairing up the three radicals 口, 冫 and 可 to form the character 啊, one may alternatively pair up 口 and 阿 where the character 阿 can be constructed from 冫 and 可.



Table 1. Construction operators supported by the current system.

Construction operator	Functional symbol	Graphical meaning
Horizontal composition	h	
Vertical composition	v	
Surround composition	s	

3.2 Our approach

In our design, a character composition expression (CCE) consists of a character construction operator followed by two components [3]. For example, the character 阿 is represented by the expression $h(\beta, \overline{\text{可}})$. A component is either another CCE, a Chinese character, or a radical. The construction operators supported by our current system are shown in Table 1.

3.3 Encoding principles

The representation of a character may not be unique. For example, 辨 can be represented as $h(\text{辛}, h(\text{讠}, \text{辛}))$ or $h(h(\text{辛}, \text{讠}), \text{辛})$. This may lead to problems in some applications. To ensure a unique representation for each character, the following encoding principles are applied to resolve ambiguities:

- (1) For a character or character component that already exists in the encoding scheme, its code should be used instead of composing the character structurally.
- (2) If a character component is a radical and it already exists in the standard radical set (i.e., it only has a radical code, but not a character code), e.g., 丨, 丿, 彡, 冫, 艹, 扌, the radical code should always be used. For a character component which has both a radical code and a character code, the exact choice depends on whether the component acts as a radical. For example, when 日 appears on the left, it is a radical and the radical code of 日 should be used. However, if it appears on the right, it is a character component but not a radical. In this case, the character code of 日 should be used.
- (3) We should always compose a character using an expression that requires the least number of components. Suppose the character 啊 does not exist in an encoding scheme. To compose it, it should be represented as $h(\text{口}, \overline{\text{阿}})$. The other two possible representations $h(\text{口}, h(\beta, \overline{\text{可}}))$ and $h(h(\text{口}, \beta), \overline{\text{可}})$ should not be used because they require more components.
- (4) In case there exist more than one composition with the least number of components, the composition with the simpler left subtree should be chosen. For example, the character 辨 should be expressed canonically as $h(\text{辛}, h(\text{讠}, \text{辛}))$ rather than $h(h(\text{辛}, \text{讠}), \text{辛})$.
- (5) The simpler-left-subtree rule is also applicable to the components that appear as the two operands of a composition expression. Suppose a character is to be composed by horizontally pairing up the three components 者, 冫 and 可. It can be composed by $h(\text{都}, \overline{\text{可}})$



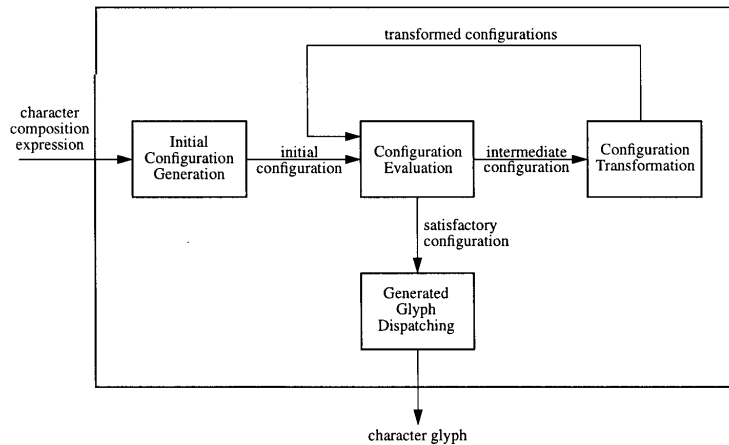


Figure 3. Glyph generation process.

or $h(\text{者}, \text{阿})$. The latter expression $h(\text{者}, \text{阿})$ should be used because its left operand is simpler.

- (6) If it is possible to compose a character by either horizontal or vertical composition using the same number of operands, then the horizontal composition is preferred. For example, the character that is composed of four radicals \square arranged in four quadrants should be expressed as $h(v(\square, \square), v(\square, \square))$ rather than $v(h(\square, \square), h(\square, \square))$.

4. Generation of Glyphs by Character Composition

4.1 The glyph generation process

In this section, we describe the process of generating the glyph of a character to be composed given its CCE. Here the term configuration (of a composed glyph) refers to a graphical representation of the components arranged in a certain manner. A composed character glyph can have many configurations. Sometimes we may just refer to the configuration as glyph when the context is clear.

The glyph generation process is basically a greedy search (Fig. 3). The target of the search is a satisfactory configuration. Starting with an initial configuration, each intermediate configuration is modified by different transformation operators to generate some possible transformed configurations. These transformed configurations are evaluated using a beauty evaluation function. The configuration which gets the highest score is chosen to be the next intermediate configuration, if its score is higher than that of the current configuration. This process repeats until no further improvement can be obtained using a greedy algorithm.

In the following subsections, different subproblems involved in the process will be discussed.

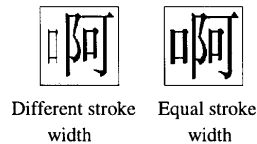


Figure 4. The need for stroke width normalization.

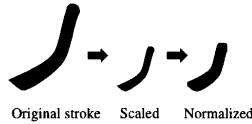


Figure 5. Stroke width normalization after scaling.

4.2 Representing the configurations

Since a CCE corresponds to a hierarchical structure, each configuration is represented by a tree, called glyph generation tree. In each node of the tree, the bounding box of the glyph rooted at the node must be maintained by the glyph generation process.

4.3 Choosing the base font

To be used as a base font for character composition, a font should satisfy the following conditions:

- (1) It should be rescalable, so as to avoid the boundary aliasing effect when it is scaled up.
- (2) It should be possible to control the stroke width, so as to keep roughly equal stroke width in different components. For a glyph composed of components of different sizes, different stroke widths will be resulted if the components are shrunk equally. Figure 4 shows a glyph without stroke width normalization and one with stroke width normalization.

Dong and Li's parametric graphics approach [6], which is used in Fan's font generation algorithm [1], can handle the stroke width normalization problem by using different parametric subroutines to generate the structural components of a Chinese font down to individual strokes. However, building the whole base font using this approach is apparently very tedious.

Another approach is to use an ordinary outline font or a high resolution bitmap font as the base font. When the character components are shrunk, the strokes are thickened to compensate for the reduction in stroke width due to shrinking. A simple method to handle this is to replace each pixel in the shrunk glyph by a rectangular region of black pixels to normalize the stroke width. Figure 5 illustrates this process. We adopt this approach in our system.

4.4 Generating the initial configuration

We define the distance between two configurations to be the smallest number of transformations required to transform one configuration to the other. The time for the search to converge is proportional to the distance between the initial and final configurations. In other words, the better the initial configuration is, the faster is the search to converge.

We choose to preprocess the initial configuration so that the white space in the components is roughly evenly distributed. This follows from one of the Chinese calligraphy rules



(see Sec. 4.6). Our experiments show that this method is quite effective and requires relatively little computation.

A simple heuristic for better allocation of space in the initial glyph is to assign space proportional to the density of a component. Such space allocation scheme keeps the densities of the two resultant components to be approximately equal. The boundary is then normalized to eliminate excess boundary white space. We can then eliminate excessive gaps between components. For the surround composition operator, the initial configuration is found by searching for the maximum empty rectangular region enclosed by the surrounding component. This region is assigned to be the initial location of the surrounded component.

4.5 Choosing the transformation operators

In our current study, transforming a configuration only affects the bounding boxes of its components. That is, no nonlinear deformation within a component is considered since this requires stroke-based information in the base font. In general, the more powerful the set of transformation operators is, the wider is the fan-out at each level of the search tree and hence the bigger is the search space. Although this implies a greater chance to reach the optimal configuration, searching a huge tree could be computationally expensive. Instead, a greedy approach is used here to find a satisfactory (but not necessarily optimal) configuration.

We use the following 18 transformation operators:

- (1) move (left, right, up, down).
- (2) enlarge (left, right, top, bottom, vertical, horizontal).
- (3) shrink (left, right, top, bottom, vertical, horizontal).
- (4) boundary shift (to-child1, to-child2).

Figure 6 shows the effect of each operator on the radical 日 of the character 明.

For a character composed of two components, we have to consider up to $16 \times 2 + 2 = 34$ possible transformed configurations. The first term 16×2 refers to the four *move* operators, six *enlarge* operators and six *shrink* operators for each component. The second term 2 refers to the two *shift* operators for the boundary between the two components.

The first three categories of transformation operators are self-explanatory. The boundary shift operators affect two components at a time. For example, the *shift to-child1* operator causes the boundary between the two components to shift towards the first child. Similarly, the *shift to-child2* operator causes the boundary to shift towards the second child.

Another factor we have to consider is the granularity (or step size) of the operators. For a glyph of size 128 X 128, it is not efficient to perform all transformation operations in the granularity of one pixel change at a time. Currently we set the initial granularity to be 1/16 of the dimension of the square glyph, which is then reduced by half every time for fine tuning until it becomes zero.



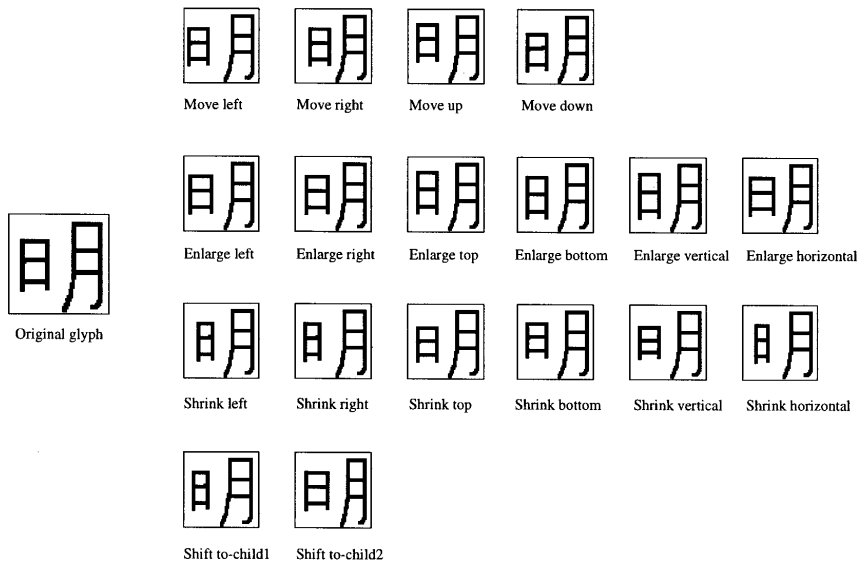


Figure 6. Effects of the 18 transformation operators (step size = 4).

4.6 Defining the beauty evaluation metrics

The beauty evaluation function is a weighted sum of the glyph beauty evaluation metrics, which are quantitative interpretations of several traditional Chinese calligraphy rules:

四平八穩 (good alignment and geometric stability).

布白均勻 (even distribution of white space).

穿插避让 (appropriate elimination of gap between components).

风格统一 (consistent style throughout the whole font).

These rules are suggested by books on calligraphy [7, 8], and are relatively easy to quantify. Each metric takes the glyph generation tree as input. The glyph is rendered on a bitmap of width x_{dim} and height y_{dim} . A number, called its beauty metric value, is returned as output. A metric is an attempt to quantify some visual quality of a glyph. In our system, the smaller the metric value is, the better is the quality. The best possible value for each metric is equal to zero.

We have devised 10 different beauty evaluation metrics: (1) component alignment metric; (2) center of gravity metric; (3) white space evenness metric; (4) density range metric; (5) feature stroke spacing evenness metric; (6) shortest distance between components metric; (7) aspect ratio difference metric; (8) size change metric; (9) border elimination metric; and (10) rule-based beauty metric.

Some metrics are only applicable to some of the horizontal, vertical and surround composition operators. Details of the metrics can be found in [9]. As an illustration, We describe metrics 2 and 10 below.



	摺	擷	擗	擗	擗	擗	擗	擗	擗	擗	擗	擗	擗	弋	忒
貳	弑	吓	叱	吼	叩	叨	叻	吒	叮	咄	呖	吮	呖	呖	呖
呃	吡	呗	另	心	叫	咂	咂	咂	咂	咂	咂	咂	咂	咂	咂
噍	哏	咕	晒	咳	哒	咧	嚏	哧	哧	哧	哧	哧	哧	哧	哧
唵	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞
啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞	啞

Figure 7. Relatively fixed position of the radical “mouth”.

4.6.1 Center of gravity metric

In the calligraphy rule “good alignment and geometric stability”, the second half of it is “good geometric stability”. It means that if a glyph is treated as a rigid body, the center of gravity of the glyph should be positioned at the center.

To compute the center of gravity of a glyph, each black pixel is treated as a unit mass. Suppose the glyph consists of n black pixels denoted as (x_i, y_i) , $1 \leq i \leq n$, the center of gravity (CG_x, CG_y) is defined as:

$$CG_x = \frac{\sum_{i=1}^n x_i}{n} \quad CG_y = \frac{\sum_{i=1}^n y_i}{n}$$

A metric derived from the center of gravity is

$$M_2 = \left| CG_x - \frac{x_{dim}}{2} \right| + \left| CG_y - \frac{y_{dim}}{2} \right|$$

where x_{dim} and y_{dim} are the width and height of the glyph, respectively. A zero value of M_2 means that the center of gravity is located exactly at the center of the glyph, and is thus “geometrically stable”.

The dimensionally normalized metric is

$$MI_2 = \left| \frac{CG_x}{x_{dim}} - \frac{1}{2} \right| + \left| \frac{CG_y}{y_{dim}} - \frac{1}{2} \right|$$

4.6.2 Rule-based beauty metric

This metric is related to the calligraphy rule “consistent style throughout the whole font”. For a set of glyphs in a font to look consistent, the style over different glyphs should be uniform. The style can be represented as some global properties of the entire font which can be stated as specific rules. An example of such a rule is the relatively fixed position of the radical 口 (mouth) whenever it appears in the left side of a glyph, as shown in Fig. 7.

We have designed a rule-based system for maintaining uniform global font style. The rules are expressed below in BNF form, where nonterminal symbols are delimited by angular brackets:

<Rules> → <Rule> <Rules>



<Rule>	→ <Op> <Component1> <Component2> <NewComponent1> <NewComponent2> <Location1> <Location2> <Weight> “NEWLINE”
<Location1>	→ <BoundingBox>
<Location2>	→ <BoundingBox>
<BoundingBox>	→ (“(” bbx1 bby1 bbx2 bby2 “)”) “ABSENT”
<Weight>	→ floatingPointNumber
<Op>	→ horizontalOp verticalOp surroundingOp
<Component1>	→ chineseCode “ABSENT”
<Component2>	→ chineseCode “ABSENT”
<NewComponent1>	→ chineseCode “ABSENT”
<NewComponent2>	→ chineseCode “ABSENT”

The semantics of a rule <Rule> is as follows: whenever a CCE in the form of <Op> <Component1> <Component2> is encountered, the radical <Component1> will be replaced by <NewComponent1> and the radical <Component2> will be replaced by <NewComponent2> if they are present (i.e., not ABSENT). <NewComponent1> and <NewComponent2> are called replacement components for the original two components. The replacement components are used so that it leaves room for improvement. For example, the radical character 火 can be replaced by a better designed radical (new component) of 火 as in the character 烧. Another example is the character component 皮 in the composition of characters like 颞. If a location is specified in the rule, it will be used as the desired location for the component. The transformation operators will try to move the component to this desired location in the final glyph to achieve consistent style, subject to the influence of other metrics in finding the final configuration.

For a given rule with the desired location of the components specified, the following metric is defined:

$$M_{10} = \frac{|bbx1 - \hat{bbx}1|}{x_{dim}} + \frac{|bbx2 - \hat{bbx}2|}{x_{dim}} + \frac{|bby1 - \hat{bby}1|}{y_{dim}} + \frac{|bby2 - \hat{bby}2|}{y_{dim}}$$

where $(bbx1, bby1, bbx2, bby2)$ and $(\hat{bbx}1, \hat{bby}1, \hat{bbx}2, \hat{bby}2)$ are the current and desired locations of the component, respectively. By minimizing M_{10} , a replacement component will be moved to its desired location as much as possible.

4.7 Defining the beauty evaluation function

In the previous subsection, we have described several metrics for quantifying the beauty of a glyph. However, when a metric M_i has value m , how can we tell whether this value m



represents a good or bad result? Also, since more than one metric may be applied during the glyph generation (refinement) process, we also need a method to normalize the metric values so that different metrics have comparable effects.

4.7.1 Normalization of metric values

Let us consider a font of n glyphs c_1, c_2, \dots, c_n , where the metric values of the glyphs are $MI_i(c_j)$ for $1 \leq j \leq n$. Suppose the theoretical optimal value of metric MI_i is zero, and the mean and standard deviation of the metric values for the font are \bar{m}_i and σ_i , respectively. We can normalize a metric so that its values are in approximately equal ranges:

$$MN_i(c) = \left(\frac{MI_i(c)}{\bar{m}_i + \sigma_i} \right)^2$$

By squaring the normalized value, a higher penalty is given to glyphs with unsatisfactory values.

4.7.2 Evaluation function

The evaluation function is defined as:

$$h(c) = \sum_i w_i MN_i(c)$$

where w_i is the weight of metric MN_i in the linear combination. The weight of a metric corresponds to the relative importance of the metric among all metrics. It is a difficult task to assign weights to the metrics. In our current system, the weights are set by trial-and-error.

4.8 Postprocessing the satisfactory configuration

After a satisfactory configuration has been found by the glyph generation process, a postprocessing step called border normalization is performed by scaling the glyph and adding to it a fixed-width border. This is important for maintaining a globally consistent style of the whole font. The method used in our system is simply to keep the tightest bounding box of the final glyph fixed. Another algorithm for border normalization can be found in [10].

5. Examples of Generated Glyphs

Figure 8 illustrates the glyph generation process that successively improves the glyph of a character until it cannot be refined any further.

For the character 鞍 which is well balanced, the appearance of the glyph looks reasonably good even at the initial phase. The preprocessing step does not help much. Later steps are mainly for fine-tuning the glyph, like further improving the alignment and separation between the components of the character.

For some characters, like 啊 which is expressed as h(口, 阿) with one small component and one big component, the situation is more complicated and the preprocessing step becomes important. For the example shown in Fig. 9, size hinting information *small* about left



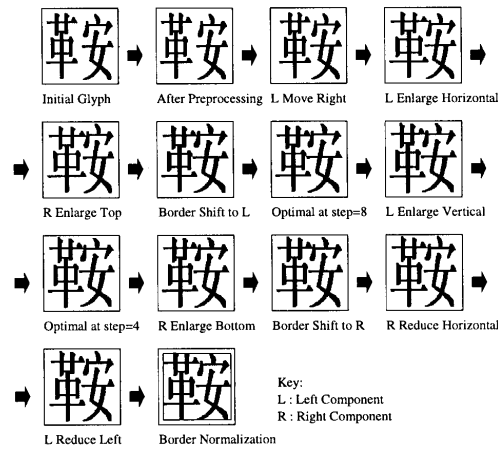


Figure 8. Example 1 – Iterations in the font generation process.

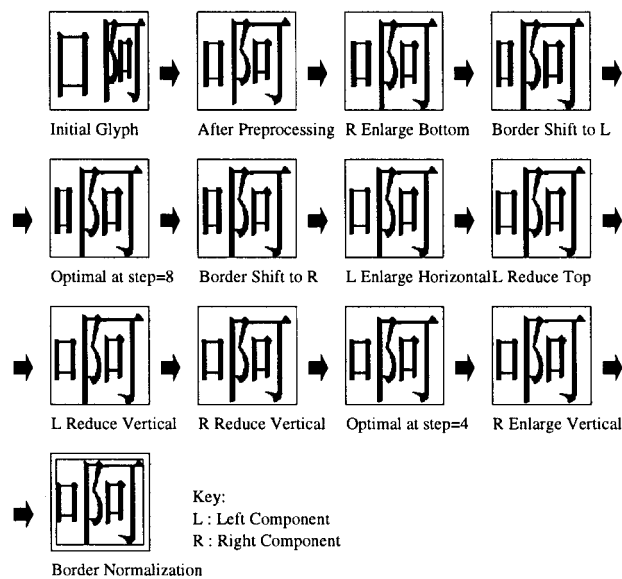


Figure 9. Example 2 – Iterations in the font generation process.

radical 冂 is specified in the CCE, suggesting that the left radical should be shrunk. After the preprocessing step, the alignment is improved, and the difference between the densities of the two components is much reduced. Other attributes, like the center of gravity of the glyph, are also improved during the search. When no better glyph can be generated, the border is normalized to give the final glyph.

To verify our approach, the first 94 characters in the GB character set are analyzed here. Some of them, like 凹, 八, 巴, 白, 半, 办, and 包, are not composed of multiple components. All other characters are specified in CCE and their glyphs are generated with a relatively low-



	啊	阿	埃	挨	哎	唉	哀	皑	癌	藹	矮	艾	碍	爰	隘
鞅	氨	安	俺	按	暗	岸	胺	案	肮	昂	盎	凹	敖	熬	翱
袄	傲	奥	懊	澳	芭	捌	扒	叭	吧	芭	八	疤	巴	拨	跋
靶	把	耙	坝	霸	罢	爸	白	柏	百	摆	佰	败	拜	裨	斑
班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	力	绊	邦	帮
梆	榜	膀	绑	棒	磅	蚌	镑	傍	谤	苞	胞	包	褒	剥	

Figure 10. Examples of generated glyphs with rules applied.

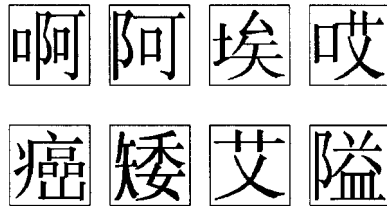


Figure 11. Examples of generated glyphs using a Song based font.

quality vector-based Hei style base font and with the rule-based beauty metric applied. The result is shown in Fig. 10.

A set of 64x64 bitmap Song style base font is also tested. Figure 11 shows some generated glyphs. Even with stroke width normalization, the generated glyphs have different stroke widths due to rounding errors caused by scaling bitmaps. This problem can be alleviated by using a vector base font or an outline base font.

6. Discussions

6.1 Comparison with other related work

Our approach to glyph generation has been inspired by a previous method proposed by Fan [1]. However, there are substantial differences between our approach and Fan's approach. We address the problem based on a different set of assumptions. Fan's approach is used for font design and generation, and is only applicable to fonts based on parametric graphics. Our approach is intended for using different existing fonts. As a result, we have chosen to put less restriction on the base font. Although our current system only handles bitmap and vector fonts, it can also be extended to handle outline and parametric fonts. This will make our techniques applicable to many existing Chinese fonts.

As we have made fewer assumptions about the base font, composing high-quality glyphs is more difficult in our system. For example, stroke-based information is absent in the bitmap or outline representation of a glyph, but it can be found in a parametric representation. As a result, stroke-based optimization can be done in Fan's approach, but not as easily in ours. On the other hand, our approach does not require a special font to be built. Existing software



	Fan's	Ours
Objective	Font design	Extensible font generation
Number of structural classes	9	3
Base font	Parametric	No assumption
Evaluation metrics	Primitive	Higher level

Table 2. Differences between our approach and Fan's approach.

can use our approach more readily, say, by using a font server where most existing fonts can be reused.

Fan classified Chinese characters into nine classes and 25 subclasses. We group the nine classes into three categories and use hierarchical character composition. This reduces the ambiguities caused by a large number of different classes. For example, if we have a three-horizontal-component operator h_3 , the character 啊 can then be expressed as either $h(\square, \text{阿})$ or $h_3(\square, \text{𠃉}, \text{𠃉})$. It thus violates the principle of enforcing unique character encoding. Fan's 25 subclasses are replaced by the technique of radical size hinting in our approach for greater flexibility.

Moreover, the metrics used by Fan are relatively simple. His metrics include the center of gravity metric, gap elimination metric, and size difference metric. Although these metrics seem to be sufficient in his approach, we need more metrics to control the quality of the generated glyphs because the quality of the base font used for composition is unknown in advance. Fan's metrics are augmented by some more sophisticated metrics in our approach, like the density difference metric, aspect ratio metric, alignment metric, stroke spacing evenness metric, and border elimination metric. The advantages of using these metrics are discussed in [9].

The differences between our approach and Fan's approach are summarized in Table 2.

If we decompose character components further, we will have strokes. Composition of characters using strokes as basic units is much more difficult. With only simple topological information like the horizontal, vertical and surround composition operators, it is difficult, if not impossible, to handle stroke-based character composition. In [11], Dürst tried to represent Kanji using a coordinate-independent description, where Kanji is represented using a box-bar model. Stroke segments are treated as bars, while radicals and character components are treated as boxes. Their topological relationships are described by the intersection points in the characters, as shown in Fig. 12. An algorithm for reconstructing the glyph of a character represented by the box-bar model has been implemented. However, the quality of the fonts generated using this approach is poor, since many features of Kanji (and hence Hanzi), like serif, length of strokes, location of intersection points, ratio of space occupied by the components, etc., will be missing if only the topological relationships of strokes are represented. Fig. 13 shows some characters generated using this approach.



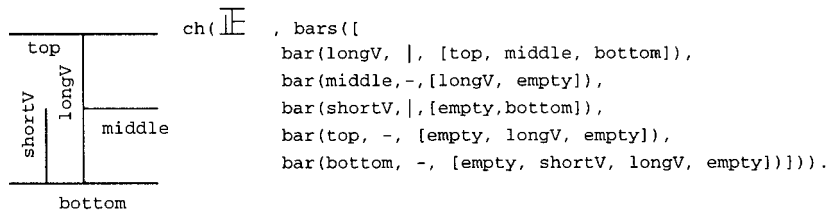


Figure 12. A coordinate-independent description for a Kanji (Hanzi).

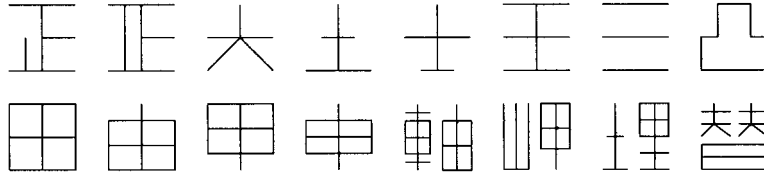


Figure 13. Examples of characters generated by a coordinate-independent description.

6.2 Future work

Our evaluation function is a weighted sum of the normalized beauty evaluation metrics. The means, standard deviations and weights of the metrics are font dependent. For a particular base font, we can determine the means and standard deviations of the metrics by using the glyphs of the base font as training data. The relative weights of different beauty evaluation metrics, however, are quite difficult to determine.

Right now, a trial-and-error approach is used for adjusting the weights of different metrics. More systematic methods, preferably with low computational requirements, will be studied in our future work.

Acknowledgement

The research work reported in this paper has been supported by the Research Infrastructure Grant RI92/93.EG08 of the Hong Kong University of Science and Technology.

References

- [1] J. Fan, "Towards intelligent Chinese character design," in R. Morris and J. Andre (eds.), *Raster Imaging and Digital Typography*, Cambridge University Press, Cambridge, 1991, pp. 166–176.
- [2] Unicode Consortium, *The Unicode Standard: Worldwide Character Encoding*, Vols. 1 and 2, Addison-Wesley, Reading, Massachusetts, 1991.
- [3] P. Lai and M. Pong, "Approaches to handle user-defined Chinese characters," in *Proceedings of the International Conference on Computer Computing*, Singapore, 1994, pp. 235–241.
- [4] 曹乃木, "部首查字法的历史演进," *语文建设* (Yuwen Jianshe), 中国, No. 2, 1993, pp. 29–32.



- [5] 汉字识别技术, 清华大学出版社, 1992.
- [6] Y. Dong and K. Li, "A parametric graphics approach to Chinese font design," in R. Morris and J. Andre (eds.), *Raster Imaging and Digital Typography*, Cambridge University Press, Cambridge, 1991, pp. 156–165.
- [7] 启功, 书法概论, 北京师范大学出版社, 1986.
- [8] 蒋彝, 中国书法, 上海书画出版社, 1986.
- [9] P. Lai, "Encoding and generation of extensible Chinese font," MPhil Thesis, Department of Computer Science, Hong Kong University of Science and Technology, 1994.
- [10] F. Uchio, T. Higuchi, et al., "A method of normalizing the appearance size of square styled brush-written Chinese characters," *Transactions of the Institute of Electronics, Information and Communication Engineers*, Vol. JF2D-II, No. 10, 1989, pp. 1650–1656.
- [11] M. J. Dürst, "Coordinate-independent font description using Kanji as an example," *Electronic Publishing: Origination, Dissemination and Design*, Vol. 6, No. 3, 1993, pp. 133–143.





Pak-Keung Lai received his bachelor degree from the University of Hong Kong and M.Phil. degree from the Hong Kong University of Science and Technology. His research interests include Chinese computing. Currently he is working for the Law-On-Line Project at the University of Hong Kong.



Dit-Yan Yeung received his B.Sc.(Eng.) degree in electrical engineering and M.Phil. degree in computer science from the University of Hong Kong, and the Ph.D. degree in computer science from the University of Southern California. He is currently an assistant professor in the Hong Kong University of Science and Technology. His research interests include Chinese computing, neural computation, statistical learning theory, and handwriting recognition.



Man-Chi Pong is currently a member of staff in the Computer Centre of the University of Hong Kong. He looks after the campus network and World Wide Web services. He has worked as a researcher, teacher, and member of the technical staff in the Hong Kong Hospital Authority, the Hong Kong University of Science and Technology, the University of Kent and the University of Edinburgh in the U.K., and the Institute of Software, Chinese Academy of Sciences, Beijing, China. He is a co-author of the widely used X-window Chinese terminal emulator called cxterm.

