EG UK Theory and Practice of Computer Graphics (2006)
M. McDerby, L. Lever (Editors)

Automatic Stroke Extraction and Stroke Ordering Based on TrueType Font

Sang Ok Koo[†] , Hyun Gyu Jang, Kwang Hee Won and Soon Ki Jung

Virtual Reality Laboratory, Kyungpook National University, South Korea

Abstract

In this paper, we suggest a method that extracts strokes of Chinese characters and orders them automatically based on glyph data from TrueType Font. TrueType Font contains glyph of characters whose format is a series of bézier curves and they are arranged according to a rule. Using clues, we extract the vectors which consist of each stoke and reconstruct each stroke to the format similar to a TrueType Font. In addition, we label every stroke and define the orders of strokes using stroke labels and indicate the relations among them. Because all of the processes are performed automatically, we can minimize the time and the effort to make stroke database of Chinese characters. Moreover, because the final data has vector graphics format, it can be applied to the study of graphical contents using glyphs as well as to simple font generation for the Chinese education.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Geometric algorithms, languages, and systems

1. Introduction

There are many people who learn the Chinese language. When we are willing to learn Chinese, the first step is memorizing Chinese characters. However, Chinese characters are so complicate and have many different characters that we cannot learn them easily. Nowadays, we can contact a lot of education contents through web or multi media devices [1][2]. Despite an increasing amount of content, vendors still generate them manually. In addition, the contents are not compatible because of the limitation of the data format. To change the attributes of contents such as size, color, and others, they have to do it all over again. So it is very important that we have to define the data format of the contents that is device-independently.

In this paper, we extract stokes of Chinese characters using TrueType Font data, which is able to be employed device-independently. This stores data in vector format, which is more efficient and more compatible. Moreover, we describe how to order stokes that are extracted from the previous step. We employ a labeling method. We name arrangements of strokes as labels, and define the orders among them. Since all parts of the process are performed automatically, we can simply build up the Chinese character stroke database. This database can be used within any application.

This paper is organized as follows. We describe the TrueType Font format which we use, and previous works in Section 2. We describe the automatic algorithm of stroke extraction and ordering in Section 3 and Section 4, Experimental results are shown in Section 5. Finally, the conclusion and future work are discussed in Section 6.

2. Background

The method of stroke extraction we propose is based on the TrueType Font data, which is more effective at expressing glyphs than using bitmap data. In this section, we discuss the properties of the TrueType Font and previous methods to extract strokes and order them.

2.1 TrueType Font and Bézier Curves

[†] sokoo@vr.knu.ac.kr

TrueType Font is a popular font format used in operating systems such as Mac OS and Windows and was designed to be efficient in storage and processing, and extensible [3]. The information from a TrueType Font is the glyph of a character, which is composed of control points of bézier curves [4]. This kind of vector font needs a font rasterizer which renders the final shape of the character, but it is very efficient structure to change certain attributes including shape or size.

Figure 1 shows the glyph of '家'. There are some points shown, that are the control points of the TrueType Font. The algorithm of stroke extraction we propose keeps these points when making up each stroke. Figure 2 shows the result of stroke extraction. The resulting points are all originally from TrueType Font data. Therefore, the result of our method has the characteristics of a TrueType Font.

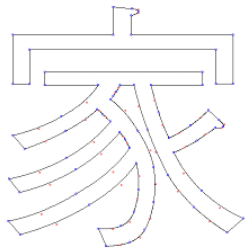

Figure 1: *The glyph of '家'*

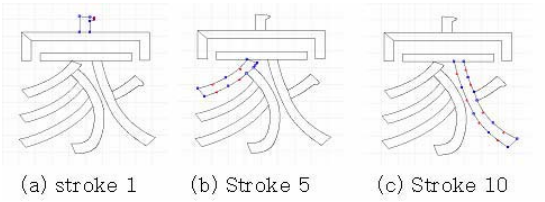

Figure 2: *The part of the result of stroke extraction*

2.2 Acquisition of Stroke Animation Data

i-hanja shows GIF animation of Chinese characters [2]. To make the GIF animation data, they have to paint stroke by stroke manually. That is simple and easy, but very time consuming work.

Koo et al. shows more natural and smooth stroke animation [5]. Their method depends on user's input as shown Figure 3. The number of stroke, the direction of each stroke and the order of strokes are determined by the user's input. This is faster and easier than making image sequences directly, but it is still cumbersome to make a stroke for each character. We propose the method to extract and to order strokes automatically without user interaction.

Figure 3: *User's input for manual stroke extraction*

3. Automatic Stroke Extraction

In this section, we describe how to extract stokes from the vectors in the TrueType Font data. As shown in Figure 4, a stroke consists of the set of vectors. There are two vector sets; one consists of vector ***a***, ***b*** and ***c***, the other consists of vector ***i***, ***j***, and ***k***. We name one vector set "corresponding vector set" to each other. We have to find vector sets as shown in Figure 4. The process of stroke extraction consists of four steps (contour grouping, extracting vectors, composing strokes and control point recovery) as shown in Figure 5.

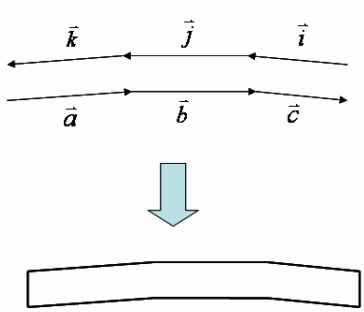

Figure 4: *A stroke is composed of vectors*

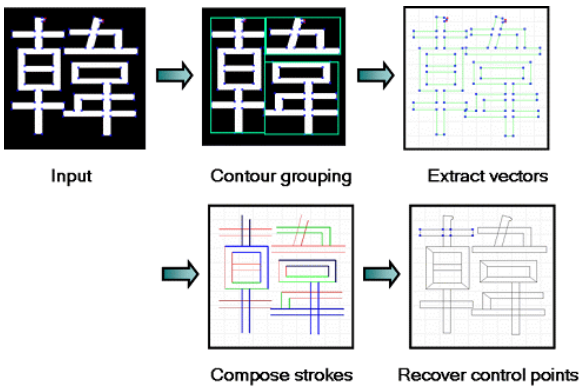

Figure 5: *The process for extracting strokes*

3.1 Contour Grouping

It is not efficient to search for corresponding vectors from all vectors. So, we make contour group, which does not affect others when finding a corresponding set. The left of Figure 6 shows the result of grouping contours of a character. Every group does not overlap any group. A group can have one contour or more. If a contour is surrounded by an outer one, they belong to same group. In this case, we have to pay attention to the fact that the direction of an inner contour is the reverse of the outer one.

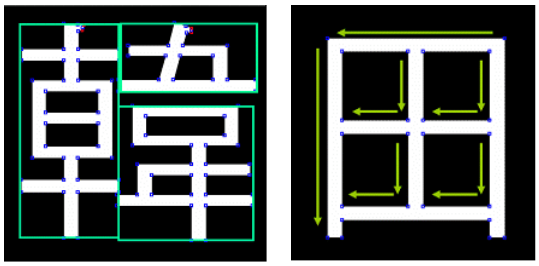

Figure 6: *Contour grouping and direction*

3.2 Extracting Vectors

The initial data from a TrueType Font is a series of bézier curves. It is hard to calculate with bézier curves. For this reason, we replace them by simple vectors as shown in Figure 7. These simple vectors keep the feature that they were originally bézier curves taking the coordinates of control points. We employ only operations between 2 dimensional vectors. This is very efficient and easy.

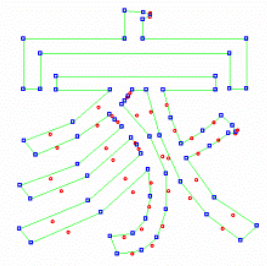

Figure 7: *Extracting vectors*

3.3 Composing Strokes

To compose stokes, we need two vector sets, which are parallel to each other. To do this, we use the distance and the angle between two vectors.

The thickness of fonts is regular. Therefore, two vectors that are far from one another cannot be corresponding vectors. A vector placed within a specified distance to another is considered as a candidate. We employ the distance between a point and a line on Euclidean 2-space.

As shown in Figure 8, there are three vector relations. In the relation of (a) and (b), we can estimate the distance. But in case of (c), the intersection is not within the line segment. Hence, we do not consider it as a possible candidate.

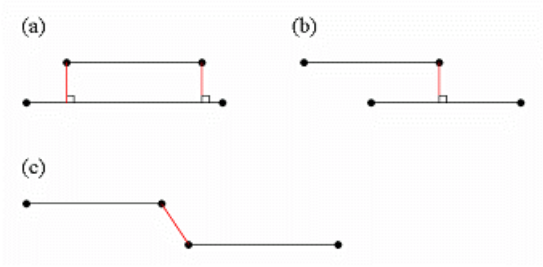

Figure 8: *Vector relations*

To examine whether a vector corresponds to another, we have to check the angle between two vectors. Two main axes which make up a stroke are nearly parallel each other. Therefore we think of two vectors that make an angle between them close to 180 degree as a possible candidate. We define the angle (θ) between two vectors $\vec{u}$ and $\vec{v}$ as taking arc cosine of the dot product of the two vectors as Equation 1.

$$\theta = \cos^{-1}\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|}\right). \qquad (1)$$

If two vectors of a stroke move counter-clockwise or right-hand direction as Figure 9-2, we can make up a stroke. But if vectors move clockwise as Figure 9-3, we cannot compose a stroke.

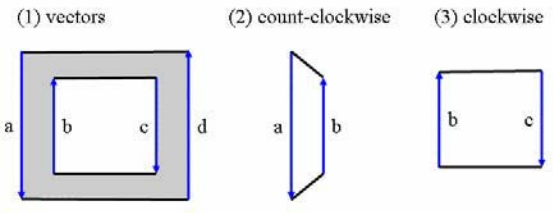

Figure 9: *Contour of a stroke moves counter-clockwise*

In order to determine the direction of two vectors, we estimate the cross product of two vectors. If there are two vectors $\vec{u}$ and $\vec{v}$, we can make a vector $\vec{w}$, connecting the head of $\vec{u}$ and the tail of $\vec{v}$ as you can see in Figure 10.

In the following Equation 2, we augment dimension, adding the value zero as a third entry, because $\vec{u}$ and $\vec{v}$ are vectors in the 2D coordinate system. If the third entry of $\vec{u}\times\vec{w}$ is greater than 0, this means $\vec{u}$ and $\vec{v}$ move counter-clockwise according to the right-hand rule.

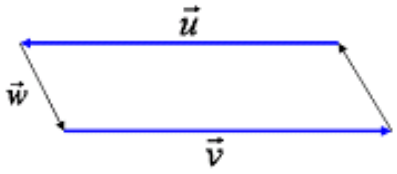

Figure 10: *Arrangement of vectors*

$$u \times w = (x_u, y_u, 0) \times (x_w, y_w, 0) = (0,0, x_u y_w - y_u x_w) \quad (2)$$

Figure 11 shows the algorithm for finding corresponding vectors for composing strokes.

T_d : threshold distance

T_a : threshold angle

```
MakeStroke( vector1, vector2)
{
   if( Distance(vector1, vector2) < Td &&
       Angle(vector1, vector2) < Ta &&
       CrossProduct(vector1, vector2) = CCW )
    {
       if( SameDirection(vector1, vector2) )
           add_same_vector_set …
       else add_opposite_vector_set …
    }
}
```

Figure 11: *The algorithm for finding corresponding vectors*

To complete making a stroke, finally it needs merging some segments. The glyph as shown in Figure 12, where there is a '+' shape, has four segments, A, B, C and D that have been composed by previous stages. However, actually we want to get strokes, A+B and C+D.

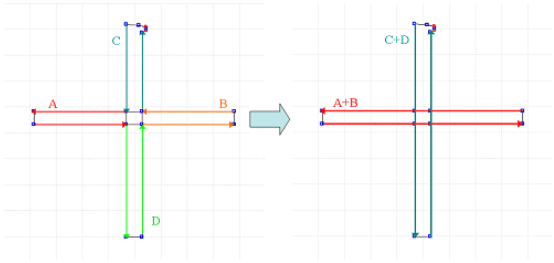

Figure 12: *Stroke segmentation*

It is easy to know which segments have to be considered. At the right part of A in Figure 13, the vectors move clockwise. On the other hand, at the left part of A, the vectors move counter-clockwise. The stroke that has the parts that vectors move counter-clockwise is a candidate. And if there is another candidate, they are merged. We can examine the fact that two vectors are arranged clockwise by using Equation 2.

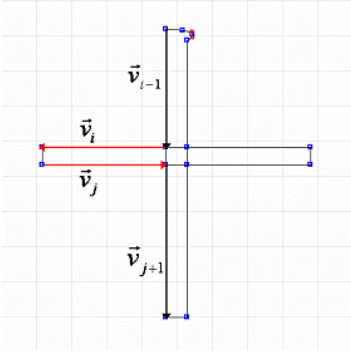

Figure 13: *Movement of vectors at '+' shape strokes*

3.4 Control Point Recovery

We have to substitute control points for vectors that have been extracted during the previous steps as shown in Figure 14.

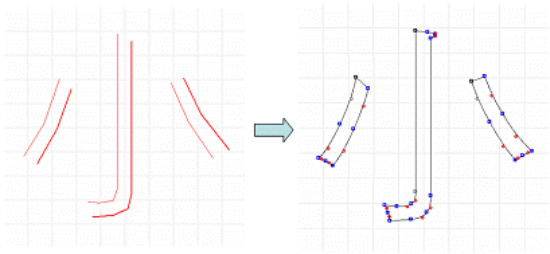

Figure 14: *Control point recovery*

In the case of Figure 15, two strokes intersect each other at 'T' or '+' shape characters. There is no vector that connects all other vectors, so we simply add line there. It is easy to know when and where these cases exist. That is the point where two or more stroke share one control point.

At the end of a stroke, we add all control points between the ends of vectors which make up a stroke as shown in Figure 16. This ensures that there is no control point which shares two or more strokes at the end of stroke.

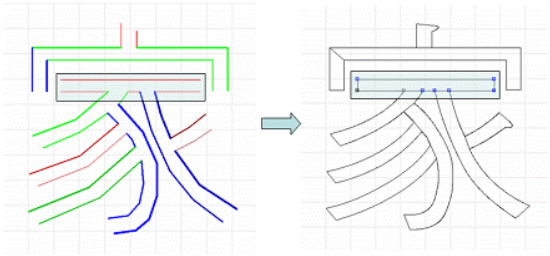

Figure 15: *Control point recovery at general part*

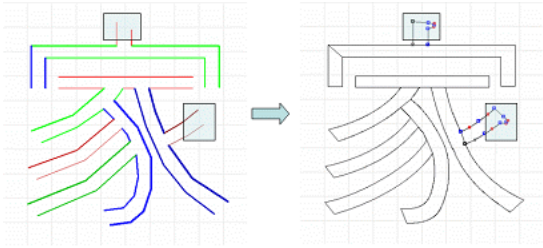

Figure 16: *Control point recovery at the end of a stroke*

4. Automatic Stroke Ordering

In this section, we describe how to estimate the order of stokes. We label all strokes, and determine the order according to the relation of them.

4.1 Stroke Classification

Table 1 shows the basic rule for ordering Chinese stroke [6], but these are exceptions. To resolve the exception, we deal with the problem as a special case. If there are stroke sets which are against the basic rule, we define them as labels and order them correctly.

Before we set the order of strokes, we should determine the stroke level. There are 7 stroke types as you see in table 2. All strokes map into that pre-defined types. The length of strokes, the slope of strokes and the angle between vectors in strokes decide the label of strokes. The algorithm for labeling is shown in Figure 17.

Table 1: *Basic rules for stroke order [6]*

Rule	Ex.	Stroke order
First horizontal, then vertical	十	一 十
First left-falling, then right-falling	人	丿 人
From top to bottom	三	一 二 三
From left to right	州	丶 丿 丬 州 州 州
First outside, then inside	月	丿 冂 月 月
Finish inside, then close	四	丨 冂 冂 四 四
Middle, then the two sides	小	亅 小 小

Table 2: *Labels of strokes*

Label	Stroke
DOT	丶
HORIZONTAL	一
VERTICAL	丨
HOOK	亅乚㇄
LEFTFALLING	丿
RIGHTFALLING	㇏
TURNING	𠃌乛

Table 3 shows subdivided stokes, which are labeled by relation of their positions. They can resolve the problem of exception.

4.2 The Algorithm for Stroke Ordering by Relaxation Labeling

Relaxation labeling is a way to define a unique label of an entry after one or more labels of an entry are given according to its attributes [7][8]. We define the set of labels (Table 3), and the relation as the position of them (see table 3). For any stroke, if we deal with the set of strokes $Q = \{s_1, s_2, ..., s_n\}$, s_i will be labeled by its relation with other strokes. The algorithm is shown in Figure 18. We consider the clues, which are intersection and adjacency of strokes and so forth.

Table 3: *Stroke subdivision*

Base	Label	Ex.	Description
DOT			Not divided
HORIZONTAL	NONE		General
	CROSS		Intersection
	LEFT		Left of other strokes
	RIGHT		Right of other strokes
	CLOSE		口 or 日
VERTICLAL	NONE		General
	T		Vertical of 王
	CROSS		Intersection
	LEFT		Vertical of 口
HOOK	NONE		General
	CROSS		Intersection
	CROSS 2		Hook of 子
LEFTFALLING	NONE		General
	SYM		Symmetry
	SYM2		Above the 一
	RIGHT		Intersection
RIGHTFALLING	NONE		General
	SYM		Symmetry
	SYM2		Above the 一
TURNING	NONE		General
	F		Turning of 民

T_L : threshold of stroke length (100 pixel)
T_{var} : the maximum angle between two vectors in a stroke
$T_{\max}$: maximum slope of stroke
$T_{\min}$: minimum slope of stroke

for all strokes in the set of strokes $Q = \{s_1, s_2, ..., s_n\}$ {
  if (length of s_i < T_L) label of s_i = DOT
  else if (vector angle of s_i > T_{var})
    label of s_i = HOOK
  else if(slope of s_i < $T_{\min}$)
    label of s_i = VERTICAL
  else if(slope of s_i > $T_{\max}$)
    label of s_i = HORIZONTAL
  else if(x-value of s_i > 0)
    label of s_i = RIGHTFALLING
  else label of s_i = LEFTFALLING

  for all strokes in the set of strokes $Q = \{s_1, s_2, ..., s_n\}$
    if (label of s_i = VETICAL &&
      label of s_j = HORIZONTAL &&
      Intersect(s_i , s_j) = TRUE)
      Label of $s_i + s_j$ = TURNING
}

Figure 17: *Stroke classification algorithm*

$Q = \{s_1, s_2, ..., s_n\}$: the set of strokes
$L = \{l_1, l_2, ..., l_n\}$: the set of labels

1. For all strokes in $Q = \{s_1, s_2, ..., s_n\}$, determine the label of s_i (by Figure 17).
2. Subdivide the label of s_i (by table 3) (s_i may have several labels).
3. Delete the labels of s_i which do not keep consistency.
4. Repeat 3, until the label of s_i is unique.

Figure 18: *Relaxation labeling algorithm*

4.3 Stroke Ordering

We make up the order after all labels of strokes are determined. We define the operator ' $\leq$ ' between two strokes. We consider the ordering strokes as a problem of sorting strokes. Operator ' $\leq$ ' is reflexive, antisymmetric, and transitive. Hence the set of strokes Q has totally ordered relation for the operator ' $\leq$ '. Operator ' $\leq$ ' is defined by the rule of table 4.

Table 4: *The order of priority between labels*

Ahead	Later
DOT	HORIZONTAL_CLOSE
HORIZONTAL_CROSS	VERTICAL_CLOSE
HORIZONTAL_CROSS	HOOK_CROSS
VERTICAL_NONE	HORIZONTAL_LEFT
VERTICAL_NONE	HORIZONTAL_RIGHT
VERTICAL_T	HORIZONTAL_CROSS
VERTICAL_LEFT	TURNING_NONE
VERTICAL_NONE	LEFTFALLING_SYM
VERTICAL_NONE	RIGHTFALLING_SYM
HOOK_NONE	HORIZONTAL_LEFT
HOOK_NONE	HORIZONTAL_RIGHT
HOOK_CROSS2	HORIZONTAL_CROSS
HOOK_NONE	LEFEFALLING_SYM
HOOK_NONE	RIGHTFALLING_SYM2
LEFEFALLING_NONE	RIGHTFALLING_NONE
LEFEFALLING_SYM	HORIZONTAL_NONE
LEFEFALLING_SYM2	HORIZONTAL_CROSS
LEFEFALLING_RIGHT	RIGHTFALLING_NONE
RIGHTFALLING_SYM2	HORIZONTAL_NONE
RIGHTFALLING_SYM2	HORIZONTAL_CROSS
TURNING_F	VERTICAL_NONE
TURNING_F	HOOK_NONE

5. Experimental Results

In this section, we made an experiment to estimate the accuracy of our method for 1000 Chinese characters. The parameters used in this experiment are:

- Font name: gulim.tcc
- Font size: 500 x 500 pixel
- Minimum of thickness of strokes: 35 pixel
- Maximum of thickness of strokes: 45 pixel
- Tolerance of slope of corresponding vectors: 15 degree
- T_L (threshold of stroke length): 100 pixel
- T_{var} (the maximum angle between two vectors in a stroke): 75 degree
- $T_{\max}$ (maximum slope of stroke): 75 degree
- $T_{\min}$ (minimum slope of stroke): 15 degree

The result of the experiment is shown in table 5. Stroke extraction is successful for as many as 97% of all the characters. The ordering experiment was taken for the characters which are extracted successfully. The ordering is successful for about 77%. Figure 19 and Figure 20 show the result of stroke extraction and stroke ordering for two characters (國 and 靑).

6. Conclusions

We proposed a method of automatic stroke extraction and stroke ordering for glyph data within a TrueType Font. We verified that about 97% of characters are well extracted and 77% of the well extracted characters are ordered correctly. We made the order of strokes using relaxation labeling. We defined the labels of strokes according to the relation of strokes, and determined a priority amongst them. It is possible to generate stroke animation data for Chinese characters automatically using our method.

Our method does not work well for any fonts whose thickness or shapes are variable. We will study to apply our method how to these kind of fonts as future work.

References

[1] USC Chinese Department Homepage, http://www.usc.edu/dept/ealc/chinese/newweb/home.htm

[2] i-hanja Homepage (Korean), http://www.ihanja.com/

[3] Microsoft Typography, http://www.microsoft.com/typography

[4] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, New York, 1988.

[5] Sang Ok Koo, Hyun Gyu Jang and Soon Ki Jung: Efficient Stroke Order Animation of the Chinese Character, *KCJC*, 2005.

[6] How To Write Chinese Characters, http://www1.esc.edu/personalstu/jli/How to Write Chinese Characters.pdf

[7] Dana Harry Ballard, Christopher M. Brown: *Computer Vision*, Prentice Hall Professional Technical Reference, 1982.

[8] Hummel R.A. and Zucker S.W.: On the foundations of relaxation labeling processes, *IEEE Transaction on Pattern Analysis and Machine Intelligence*., Vol.5, No. 3, 1983.

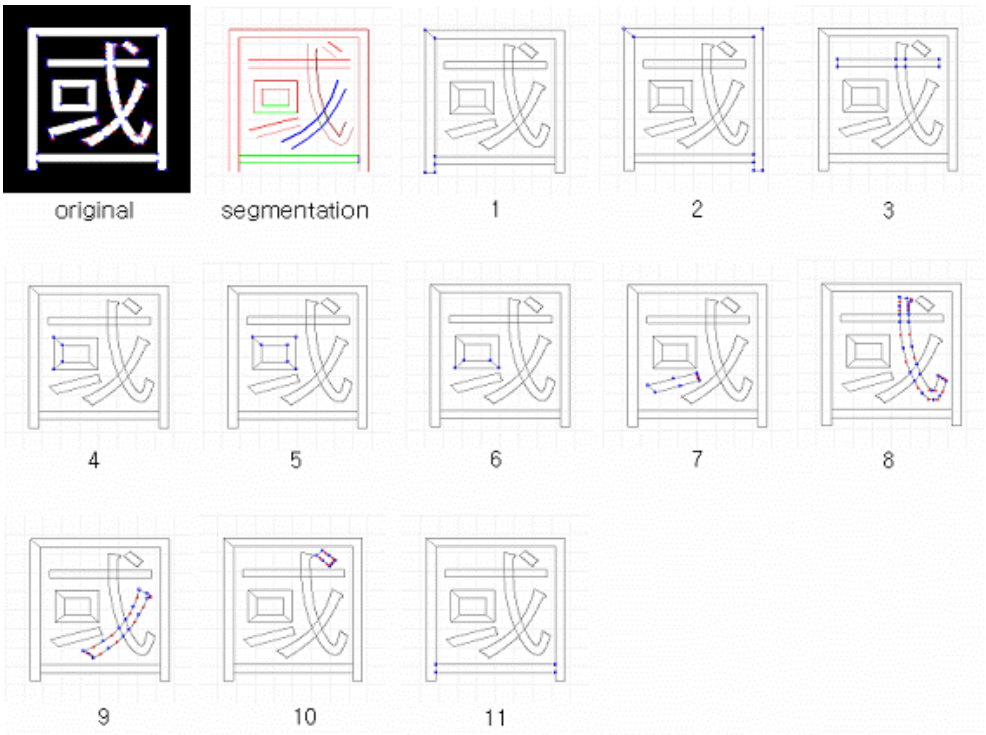

Figure 19: *The result of '國'*

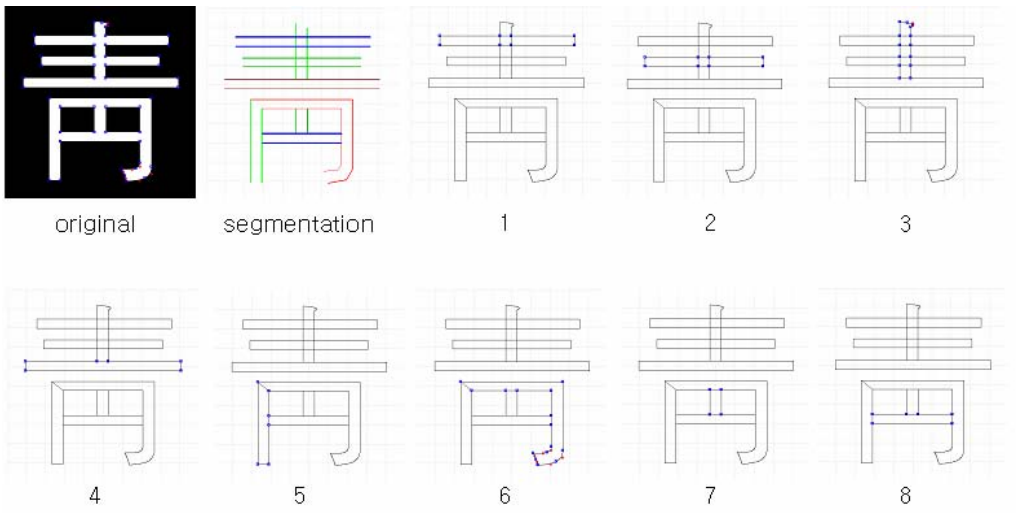

Figure 20: *The result of '靑'*

Table 5: *The result of the experiment*

<table>
<tr><th colspan="2">The extracted well</th><th>fail</th></tr>
<tr><td colspan="2">972 (97.2%)</td><td rowspan="3">28 (2.8%)</td></tr>
<tr><th>The ordered well</th><th>fail</th></tr>
<tr><td>751 (77.3%)</td><td>221 (22.7%)</td></tr>
<tr><td>家歌價可加假街暇角各刻間看干減甘甲江强講降開改個客車去巨拒居件傑檢儉格激見堅犬決結潔京敬景輕競經境警傾更鏡界計係階戒季鷄系高苦古告固故孤庫曲穀困骨工空公功共孔攻果課觀官管光廣鑛校教敎交橋口區具句構國局軍郡君群宮權勸貴歸規均極筋勤金今禁給氣記旗己基技汽期器起奇紀 寄吉暖難南男納內年努農能多短團壇端單檀談擔答堂當大代待隊帶圖度到都盜讀獨督東動洞同冬童銅頭豆得等羅落亂覽朗冷略良量糧慮力歷練列烈領令例禮老綠錄論龍類流留六陸輪律利李離林立萬滿末望亡賣買脈面勉名命明毛模木目牧墓務無舞門文問物米美味未民朴博拍反班髮方放房防訪倍配背 拜白百番罰伐範犯法壁辯別兵報寶保普服福伏複本奉夫部富復副府否負北分憤粉不比鼻備非批碑貧四事社使死仕士史査師舍寺辭絲私算産散殺三上相商狀床象傷色生西書序夕石席先鮮善宣雪說設舌成省性誠聖城星盛世洗勢細稅小少所掃笑素束俗續屬孫損松頌手樹首修守秀宿叔順純術崇習市時示視試詩施式識信新神臣申室失實深十氏兒眼暗壓額夜弱藥約洋陽羊樣語魚漁言嚴餘易域煙演研緣鉛燃熱葉英營映豫五午誤屋玉溫完王往外曜謠浴勇用容右雨友牛郵雲雄園元願原院員圓援 源月偉位爲衛圍危威有由油乳儒育肉銀音飮陰 邑義議儀二以移異益人因印引仁一日任入自子字資昨作雜場章將障壯腸才在財材再爭貯低底 的赤敵籍賊績積電全前戰典傳田專轉節切絶折店點占情停精程丁靜制除帝祖朝調操助鳥早條組潮足族尊存卒終宗左座罪主住注周朱酒竹準中增紙地知至支指持智直織眞盡珍陣質集次着讚察參唱創採責冊川千天靑淸請體草初招寸村銃推祝築蓄縮春充蟲就測層致置治則親七侵寢針稱快打他卓彈歎脫探太宅擇土討統投特波破板判八敗便篇平評閉砲包布胞票標品風豊筆下夏河韓漢寒閑合港抗害解核幸行向香鄕許險革現賢血協兄形刑號湖護戶或混紅話花和畫化貨確歡活黃況會回孝效後候厚訓揮休凶希喜徊喧栗</td><td>覺簡感監敢康擧據建健 擊缺慶驚繼考科過關九 球舊救求究屈卷券根近急級女念怒斷段達黨對 德道島導徒逃毒斗登燈 樂卵來旅麗連路料柳里理馬每妹鳴母妙武聞密 半發妨變邊病婦佛費悲 飛祕思寫謝射山賞常想線仙船選姓聲歲消速送 數收授受肅勝承始是食植息身心惡安案愛液野養億業如與逆然延迎要 遇優運遠怨委慰遺遊恩 隱應意衣依耳認者姿殘長裝帳張適展錢點接正 庭定濟際聽總最出忠取 趣置齒炭態通痛鬪退派表疲避必學限恨航海憲 驗顯惠呼好婚患環黑吸 興誨逅誇段篤遁慮鸞萊療率躪卍妄媒魅麥汚繁魏</td><td>窮極劇機兩勞步父氷水燃永榮藝醫疑災秋暴爆虛火華灰囊髄懊鑿</td></tr>
</table>