Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 415–424

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Automatic Segmentation of Chinese Characters as Wire-Frame Models

Antoine Bossard[1]

Graduate School of Science, Kanagawa University
2946 Tsuchiya, Hiratsuka, Kanagawa, Japan 259-1293

Abstract

There exist thousands of Chinese characters, used across several countries and languages. Their huge number induces various processing difficulties by computers. One challenging topic is for example the automatic font generation for such characters. Also, as these characters are in many cases recursive compounds, pattern (i.e. sub-character) detection is an insightful topic. In this paper, aiming at addressing such issues, we describe a segmentation method for Chinese characters, producing wire-frame models, thus vector graphics, compared to conventional raster approaches. While raster output would enable only very limited reusing of these wire-frame models, vector output would for instance support the automatic generation of vector fonts (Adobe Type 1, Apple True Type, etc.) for such characters. Our approach also enables significant performance increase compared to the raster approach. The proposed method is then experimented with a list of several Chinese characters. Next, the method is empirically evaluated and its average time complexity is assessed.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: vector, graphics, scalable, script, kanji, glyph, Japanese, font

1 Introduction

Chinese characters are numerous (thousands) and used in several languages and writing systems such as in Japanese and Korean. Because of their huge number, various computer processing difficulties arise. For instance, one challenging topic is the automatic font generation for such characters. Also, interestingly, Chinese characters are often recursive compounds: for instance, the character 榎 *hackberry tree* consists of the character 木 *tree* horizontally combined with the character 夏 *summer*. This way, several "layers" of sub-characters can be often identified from the characters [7, 12]. Automatic and efficient character pattern (i.e. sub-character) recognition is an important topic with various applications (font generation, compression, etc.).

A conventional approach for two-dimensional image segmentation is thinning [10, 11]. This image processing technique is usually based on either convolution matrix application or decision tree, anyway pixel-based methods: this is the raster (bitmap) approach. It obviously requires going through each of all the pixels making the image to be segmented. Such a raster approach thus has for disadvantage that 1) the segmented result is only valid for the original image, no scaling up or down is possible, or at

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.122

significant precision and accuracy loss, and 2) a high time complexity, at least linear in the size of the source image, and 3) induced further complexity for segmented model analysis: raster analysis again required, or intermediate vectorisation.

Our objective in this paper is to propose a segmentation method for Chinese characters aiming for instance at the fast automatic generation of a character pattern database for facilitated character decomposition [5, 4, 3]. Segmentation here means that a two-dimensional wire-frame model of a character will be produced by extracting the character "spine". Also, as other application example, once characters have been segmented, it will provide a base for fully automatic Chinese character font generation, a famously difficult topic [13].

Our strategy is as follows: instead of conventionally segmenting Chinese characters from a bitmap image (raster graphics), we first generate the character list using vector graphics, concretely with the SVG format [14], before going on with segmentation. Compared with those of previous works, this approach has for significant merit that the resulting wire-frame models are also vector graphics, and that the time complexity required to segment a character is significantly reduced compared with that required by the conventional raster graphics approach. Informally, the time complexity is reduced from linear to logarithmic in the size of the character raster image. Furthermore, analysis (e.g. character pattern analysis) of the obtained vector graphics wire-frame model is greatly facilitated compared with that of a raster wire-frame model, and additional processing of the wire-frame model, such as font generation, is significantly better since a raster wire-frame model would induce a bitmap font (PCF, FON, etc.) with its well-known shortcomings compared to an outline font, i.e. a "modern" vector (scalable) font (e.g. Adobe Type 1 [8], Apple TrueType [1], OTF, etc.), or would first require vectorisation of the obtained raster wire-frame to then produce an outline font, but then at a higher total complexity since in two steps: rasterisation, followed by vectorisation. In addition and importantly, our method enables instant mapping of a segmented character to its Unicode (or any other encoding: Shift JIS, EUC, etc.) representation.

2 Preprocessing: character list as vector graphics

The aim of this preprocessing step is to produce vector graphics from Chinese characters for further processing. This is achieved as follows (we rely on the Inkscape vector drawing software [2]).

1. Create a new text object and type-in or paste the desired Chinese characters. A common scenario is to first copy an extensive list of Chinese characters to be processed, like the list of the Japanese regular-use characters [9], and then to paste this list inside the Inkscape text object.
2. Select the newly created text object and convert it to a path with the *Object to Path* command.
3. Save the file (SVG format by default, which is suitable). In the example of the list of the Japanese regular-use characters, a 2MB SVG file is generated.

The generated SVG file is structured as follows: each character corresponds to one SVG `path` element. All such path elements are grouped under an SVG `g` element. Each path element (i.e. path for one character) is defined using the `d` attribute which declares the path nodes and node connection (straight lines and Bézier curves in our case). As example, the path element for the character 圧 is given below.

```
<path
 d="m 355.3125,119.50507 30.46875,0 0,2.96875 -27.34375,0 0,17.5 q 0,7.34375 -3.75,11.5625 l
      -2.34375,-2.5 q 2.96875,-3.125 2.96875,-9.6875 l 0,-19.84375 z m 19.0625,5.46875 q
      1.25,0.46875 -0.15625,0.9375 l 0,7.1875 10.46875,0 0,2.96875 -10.46875,0 0,12.34375
      13.28125,0 0,2.96875 -28.59375,0 0,-2.96875 12.1875,0 0,-12.34375 -9.375,0 0,-2.96875 9.375,0
      0,-8.125 3.28125,0 z"
 style=""
 id="path3595" />
```