한글 외곽선 폰트의 자소 분할

구상옥[O] 정순기*

경북대학교 IT대학 컴퓨터학부

sokoo@vr.knu.ac.kr skjung@knu.ac.kr

Hangul Component Decomposition in Outline Fonts

Sang Ok Koo[O] Soon Ki Jung*

School of Computer Science and Engineering, Kyungpook National University

요 약

본 논문은 한글 외곽선 폰트를 입력으로 글자의 초성, 중성, 종성 요소(컴포넌트)를 통계적-구조적 정보를 이용하여 분할하는 방법을 제안한다. 한 폰트 내에서 한글 컴포넌트는 통계적으로 일정한 위치에 나타나며, 각 컴포넌트를 이루는 획 간의 관계는 그 컴포넌트의 구조적 특징을 나타낸다. 우리는 먼저 각 컴포넌트의 위치를 저장하는 컴포넌트 히스토그램을 생성하여 컴포넌트 위치에 관한 통계 정보를 저장하였다. 그리고 글자의 구조적 정보를 반영하기 위해 픽셀의 방향성 확률을 기반으로 픽셀클러스터를 만들고, 클러스터의 위치, 방향 및 크기, 클러스터간 인접성 정보를 이용하여 후보 획을 추출하였다. 마지막으로 릴렉세이션 레이블링을 통해 후보 획 집합과 미리 정의된 글자 모델 간의 가장 적합한 구조적 매치를 구하였다. 본 논문에서 제안한 컴포넌트 분할방법은 한글 폰트의 조형적 특징에 관한 연구 및 이를 활용한 폰트분류 및 폰트검색에 활용될 수 있다.

Abstract

This paper proposes a method for decomposing a Hangul glyph of outline fonts into its initial, medial and final components using statistical-structural information. In a font family, the positions of components are statistically consistent and the stroke relationships of a Hangul character reflect its structure. First, we create the component histograms that accumulate the shapes and positions of the same components. Second, we make pixel clusters from character image based on pixel direction probabilities and extract the candidate strokes using position, direction, size of clusters and adjacencies between clusters. Finally, we find the best structural match between candidate strokes and predefined character model by relaxation labeling. The proposed method in this paper can be used for a study on formative characteristics of Hangul font, and for a font classification/retrieval system.

키워드: 한글, 폰트, 글립분석, 컴포넌트분할, 폰트특징추출

Keywords: Hangul, font, glyph analysis, component decomposition, font feature extraction

* 교신저자: 정순기 (skjung@knu.ac.kr)

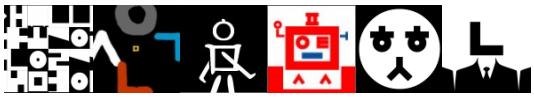
투고일 : 2011.08.30 / 심사일 : 2차 2011.09.09 / 게재확정일 : 2011.09.09

1. 서론

디지털 문서 환경에서 가독성과 심미성을 위해 고안된 디지털 폰트는 단순한 문서작업에 뿐만 아니라 개인의 개성을 나타내는 도구이자 예술작품 및 디자인의 재료로 활용되고 있다. 과거 다양한 폰트의 주된 이용자는 서적, 방송, 영화 등의 매체에 종사하는 디자이너였으나 디지털매체가 일상적인 커뮤니케이션의 도구로 사용되면서, 일반인들도 보다 다양한 디지털 폰트를 원하고 소비하게 되었다. 개인컴퓨터와 스마트폰 등의 개인화된 정보기기의 보급, 개인 블로그(blog)와 홈페이지 등의 일상화로 인해 폰트의 소비는 빠르게 증가하였고, 최근에는 폰트 제작에 있어서도 양적인 비약을 보이고 있다.

이에 비슷한 모양과 느낌을 가진 폰트들끼리 분류하고, 어떤 폰트와 가장 비슷한 폰트를 찾아주는 폰트검색 시스템이 필요하게 되었다. 그러나 현재 한글폰트에 관해서는 폰트파일의 헤더(header) 정보, 즉 폰트패밀리이름(font family name)과 폰트파일의 크기 정보만을 이용하는 폰트검색 도구만 존재할 뿐[9], 글자의 모양에 기반 한 폰트검색시스템은 없는 실정이다. 현재 폰트 사용자가 자신이 원하는 폰트를 찾기 위해서는 폰트 이름과 실제 글자의 모양을 연결하여 기억해 두었다가 필요할 때 폰트이름으로 검색하여 찾거나, 폰트이름을 모르는 경우 폰트데이터베이스를 일일이 뒤져서 원하는 폰트를 찾아야 한다. 요즘 유명인의 글씨체나 일반 개인의 손글씨까지 폰트화하는 등 한글폰트의 수가 기하급수적으로 늘어나는 추세를 감안해볼 때, 글자모양에 기초한 폰트분류 및 폰트검색 시스템은 폰트제작자와 폰트이용자들에게 유용한 도구가 될 것이다.

글자는 다양한 예술영역의 재료로 활용되고 있다. 글자에 관한 디자인 영역인 타이포그래피(typhography)나 캘리그래피(calligraphy)를 넘어서, 애니메이션, 회화, 조각, 의상, 건축에 이르기까지 글자 자체의 조형적 아름다움을 활용한 작품들이 많이 존재한다. 특히 한글은 그 꼴이 기하학적인 모습에서 출발했기 때문에, 아래 그림과 같이 한글의 기하학적 아름다움을 표현 및 활용한 작품들이 많다[1].



한글 폰트를 문서 작성을 위한 실용적 이용 뿐 아니라, 다양한 예술작품에 자유롭게 활용하기 위해서, 그리고 수많은 한글 폰트들을 글자의 모양으로 분류·검색하기 위해서는 먼저 한글 폰트의 특징을 분석해야 한다. 한글은 초성, 중성, 종성의 독립된 구성요소가 하나의 글자에 함께 나타나고, 그것들은 또 다시 몇 개의 기본 획으로 구성되는 계층적인 구조를 가진다. Biederman[1]의 요소에 의한 인식(recognition-by-components) 이론에 따르면, 사람은 어떤 시각적 입력이 들어오면 뇌에 저장된 물체에 대한 구조적 표현들-'부분'이 되는 기본 모양들과 그것들의 상호관계들-에 입력 영상을 일치시킴으로서 물체를 인식한다고 한다. 이를 글자구조에 적용한다면 '부분'은 획 또는 보다 상위의 구성요소(한글의 초·중·종성, 한자의 부수 등)를 나타내고, 그것들의 공간적인 배치가 글자의 '모양'을 결정한다고 볼 수 있다. 이 개념은 한글이나 한자와 같은 계층적인 구조를 가진 글자의 인식(character recognition)에 관한 연구에 많이 이용되었다[2, 3].

본 논문에서도 위 이론에 입각하여, 먼저 한글의 세 구성요소를 분할하고 분할된 각 구성요소의 모양, 크기, 획의 개수와 특징, 그리고 한 글자 내에서 세 구성요소의 위치를 고려하여 한글 폰트의 특징을 추출한다. 이는 폰트 안의 모든 글자(완성형 2,350자, 조합형 11,172자)를 독립적인 객체로 보고 분석하는 것보다 훨씬 효율적이며 폰트의 특징을 보다 정확히 파악할 수 있다. 마찬가지로 세 구성요소를 정확히 분할하기 위해서 각 구성요소를 그것의 기본 단위인 획으로 분할하고, 획의 방향과 크기, 그리고 한 구성요소 내에서의 획의 위치와 획들 간의 관계를 이용한다.

본 논문에서는 자소의 공간적인 정보를 획득하기 위해서 통계적 방법을 도입한다. 같은 폰트 내에서 '각'의 초성 'ㄱ'과 '같'의 초성 'ㄱ'은 동일한 위치와 크기를 가질 가능성이 높다. 이는 폰트를 제작할 때, 주로 자소별로 모양을 만들고 나서, 이를 조합하여 각각의 글자를 만드는 과정을 거치기 때문이다. 조합 과정에서 글자의 전체적인 모양을 고려하여 자소의 모양을 미세하게 변형시키거나 위치를 조정할 뿐이므로 자소의 모양과 위치의 변화가 매우 적은 편이다. 손글씨 이미지를 보정 없이 그대로 폰트로 만드는 경우도 있는데, 이 경우에도 한 사람의 손글씨 내에서는 같은 자소의 모양과 위치가 일관성을 가진다. 이는 우리가 사람의 필체를 신원확인(identification)에 이용하는 이유이기도 하다. 따라서 본 논문에서는 '핵'을 분석할 때, 'ㅎ'이 초성으로 나오는 다른 글자들(예를 들어, '함')의 초성, 중성이 'ㅐ'인 글자들(예를 들어, '밴')의 중성, 그리고 종성이 'ㄱ'인 글자들(예를 들어, '각')의 종성의 위치에 대한 확률정보를 이용한다. 그러나 컴포넌트 위치에 관한 확률정보만 이용할 경우, 꼭 필요한 획이 생략되거나 있을 수 없는 획이 추가되는 경우가 발생될 수 있다. 본 논문에서는 위치 기반으로 분할된 컴포넌트의 검증(verification)과 정확한 재분할을 위해 컴포넌트의 구조적 정보 즉 컴포넌트를 구성하는 획 간의 관계를 이용한다.

본 논문의 2절에서는 기본적인 폰트 데이터 표현과 전처리 과정, 즉 자소의 위치와 크기에 대한 통계정보 획

1) 2010년 10월 9일 한글날 기념 기획 전시회 작품 중 일부: 왼쪽에서부터 '한글통통'(노승관), '한글과 나무'(이상현), '한글픽토그램', '한글로봇'(이정훈), '표정','양복입은 사내들'(허한솔).

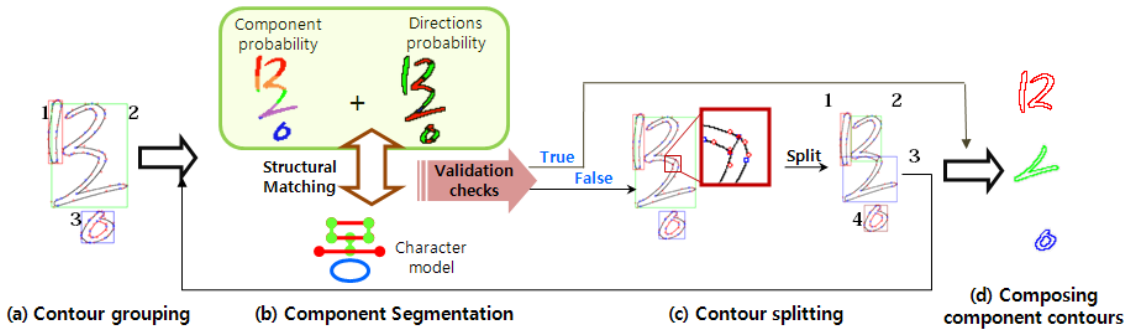


그림 1. 한글 외곽선 폰트의 컴포넌트 분할 과정 (예: 봄날체 '몽').

득과 획 추출 방법에 대해 기술한다. 3절에서는 자소분할 결과의 검증 및 릴렉세이션 레이블링을 통한 구조적 매칭 알고리즘을 제안한다. 4절에서는 여러 한글 폰트의 구성요소 분할 결과에 대해 토의하고, 마지막으로 5절에서는 결론을 짓고 향후 연구에 대해 언급한다.

2. 한글 폰트와 통계적 글자 모델

본 절에서는 외곽선폰트 파일(*.ttf, *.ttc, *.otf)을 읽어, 한글 자소의 모양, 크기 및 위치에 관한 통계정보를 획득하고, 각 픽셀의 방향성을 기반으로 획을 추출하는 방법을 기술한다. 그림 1은 2절과 3절에 걸쳐 설명할 컴포넌트 분할 과정을 그린 것이다. 외곽선 폰트가 입력으로 들어오면 컨투어를 그룹핑하고, 컴포넌트의 모양에 대한 통계정보와 픽셀 방향성에 대한 확률정보에 기반 하여 컴포넌트를 분할한다. 컴포넌트 검증을 통해 완전한 분할로 판별되지 않으면 컨투어 분리(split)를 수행하고, 분할 과정을 반복한다. 그림 1의 '몽'은 초성과 중성 컴포넌트가 하나의 컨투어 그룹으로 이루어지는데, 이처럼 하나의 컨투어 그룹에 두 개 이상 컴포넌트의 획들이 들어있는 경우, 컨투어를 분리해야만 컴포넌트 분할이 가능하다. 본 논문의 최종적인 분할 결과는 기존 글자인식(recognition)에서와 같이 스켈레톤(skeleton) 또는 픽셀 세그먼트가 아니라, 외곽선 폰트와 같은 베지어 컨투어 형태이다.

2.1 오픈타입폰트와 베지어곡선

트루타입(True Type)과 포스트스크립트(Postscript) Type1 폰트를 통칭하는 오픈타입(Open Type)폰트는 2차 이상의 베지어곡선(Bézier curve)으로 글자의 외곽선을 표현한다[4, 5]. 이는 비트맵 폰트에 비해 폰트의 크기를 획기적으로 줄이며 해상도에 관계없이 좋은 품질의 글자를 렌더링 할 수 있는 표현방법이다. 그러나 외곽선폰트는 각 글자의 모양정보만 가지고 있을 뿐 글자를 이루는 컴포넌트나 획 등에 대한 정보를 가지고 있지 않다. 이를 보완하기 위해 메타폰트(Metafont)나 획 기반 폰트(Stroke-based Font) 등이 고안되었으나 일반적으로 사용되지는 않고 있다[6, 7, 8]. 본 논문은 현재 PC 및 스마트폰 등 모바일 기기에서 가장 널리 사용되고 있는 트루타입/오픈타입폰트를 입력대상으로 한다.

폰트에서 캐릭터(character)란 동일한 의미나 형태를 갖도록 선언된 모양 클래스를 표현하는 추상화된 표식이며, 글립(glyph)은 캐릭터의 특정한 인스턴스이다. 글립정보는 베지어 커브로 표현된 N개의 폐곡선형태로 주어진다. 그림 1의 (a)는 캐릭터 '몽'에 대한 봄날체 폰트의 글립을 그린 것이다. 이 경우 총 4개의 3차 베지어곡선으로 표현된 외곽선으로 이루어졌음을 알 수 있다.

2.2 컨투어, 컨투어 그룹 그리고 컴포넌트

본 논문에서는 베지어곡선으로 외곽선을 표현하는 점의 집합을 컨투어(contour)라 하고, 단일 컨투어 또는 'ㅁ, ㅇ, ㅎ, ㅂ' 등에서 나타나는 것처럼 외부컨투어와 내부컨투어로 이루어진 컨투어 집합을 컨투어그룹(contour group)이라 한다. 마지막으로 본 논문에서 분할하고자 하는 자소들을 각각 초성(initial), 중성(medial), 종성(final) 컴포넌트(component)라고 부르겠다.

컴포넌트 분할을 위해서는 먼저 컨투어 그룹핑이 선행되어야 한다. 독립적인 개별 컨투어를 하나의 그룹으로 보되, 'ㅇ', 'ㅂ', 'ㅁ', 'ㅎ' 과 같이 내부컨투어가 존재할 경우, 내부컨투어는 외부컨투어의 자식(child)으로 규정하고, 부모-자식관계의 컨투어 집합을 묶어 하나의 컨투어 그룹으로 만든다. 그림 1은 (a)'몽'에 대한 4개의 컨투어가 입력으로 들어오면 (b)3개의 컨투어그룹으로 그룹핑되고, 컨투어 분리과정을 통해 4개의 컨투어그룹으로 나뉘었다가 (c)총 3개의 컴포넌트로 분할된 결과를 보여준다.

2.3 컴포넌트 히스토그램

네모체 음절 한글 글립의 전체적인 모양은 세 컴포넌트가 하나의 사각형 안에 비교적 일관적인 위상(topology)을 가지고 배치된 형태이다. 한글 글꼴의 배치는 크게 중성의 모양과 종성의 유무에 따라 결정되는데, 본 논문에서는 표 1과 같이 18가지 계열로 구분하였다. 표 1은 일반적인 한글 컴포넌트의 영역을 나타낸 비트맵이며, 컴포넌

표 1. 한글 글립의 컴포넌트 위상에 따른 분류.

종성유무		중성타입	초성위치	중성위치	종성위치
종성없음	1	ㅏ, ㅑ, ㅐ, ㅒ			
	2	ㅓ, ㅕ, ㅔ, ㅖ			
	3	ㅣ			
	4	ㅗ, ㅛ			
	5	ㅜ, ㅠ			
	6	ㅡ			
	7	ㅘ, ㅙ, ㅚ			
	8	ㅝ, ㅞ, ㅟ			
	9	ㅢ			
종성있음	10	ㅏ, ㅑ, ㅐ, ㅒ			
	11	ㅓ, ㅕ, ㅔ, ㅖ			
	12	ㅣ			
	13	ㅗ, ㅛ			
	14	ㅜ, ㅠ			
	15	ㅡ			
	16	ㅘ, ㅙ, ㅚ			
	17	ㅝ, ㅞ, ㅟ			
	18	ㅢ			

트 분할을 위한 초기값으로 사용한다. 비트맵에서 흰색은 해당 컴포넌트의 확률이 1임을, 검은색은 0임을 나타낸다. 초기에는 0 또는 1의 값을 가지지만, 한 글자의 컴포넌트 분할이 완료되면, 해당 비트맵의 각 컴포넌트에 대한 확률을 갱신한다. 본 논문에서는 이를 컴포넌트 히스토그램(component histogram)이라 부르겠다. 히스토그램의 버킷(bucket) 크기는 비트맵의 한 픽셀이다. 그림 2는 입력글자 집합에 대한 컴포넌트 히스토그램의 생성 과정을 보여준다.

표 1에서처럼 중성타입 <1,2,3>은 동일한 위상을 가지고 있지만, 중성 컴포넌트의 크기와 위치가 다소 달라지므로 이에 따라 초성의 위치와 크기가 달라지고, 어떤 경우에는 글자를 이루는 컨투어들의 위상이 크게 달라지기도 한다. 예를 들어 중성타입<1>의 '가'와 달리 타입<2>의 '거'는 한 개의 컨투어에 초성과 중성이 연결된 구조이기 때문에 컴포넌트 분할을 위해서는 'ㄱ'의 세로획과 'ㅓ'의 가로획이 만나는 지점에서 컨투어 분리(split)가 필요하다. 반면 '가'는 초성, 중성 컴포넌트가 각각 독립적인 컨투어를 가지기 때문에 컨투어 분리없이 바로 컴포넌트 분할이 가능하다. 이런 이유로 각각 다른 히스토그램을 사용하도록 하였다. 이는 중성타입 <4,5,6>과 <7,8,9>에 대해서도 마찬가지다.

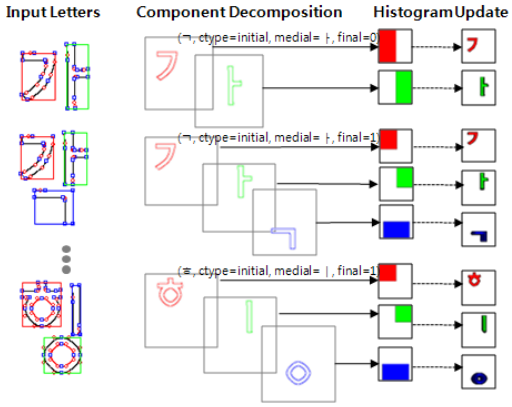


그림 2. 컴포넌트 히스토그램 생성 개요.

초성 컴포넌트는 중성의 타입과 종성유무에 따라 모양 변화가 큰 편이며, 중성 컴포넌트는 종성유무에 따라 크기 및 위치 변화가 있다. 따라서 초성 컴포넌트의 히스토그램은 $N_{initial} \times N_{medial} \times 2 = 19 \times 9 \times 2 = 342$ 개, 중성 컴포넌트는 $N_{medial} \times 2 = 9*2 = 18$개의 히스토그램을 만든다. 종성 컴포넌트는 그 위치가 상대적으로 안정적이므로, 한 벌 즉 $N_{final} = 28$개의 종성 히스토그램만 생성한다.

각 픽셀의 확률 값의 갱신은 다음과 같다. 먼저 하나의 글자 입력에 대해서 초·중·종성 컴포넌트를 분할 후, 분할된 각 컴포넌트별로 컨투어 그룹들의 내부를 1로 외부를 0으로 채운 크기가 I인 컴포넌트 비트맵 $B = \{B_1, ..., B_I\}$를 구한다. 컴포넌트의 문자코드가 c이고, 컴포넌트 타입이 t (initial=1, medial=2, final=3)이며, 종성의 존재유무가 e (true 또는 false)일 때 우리는 컴포넌트 히스토그램 $H(C=c, T=t, E=e) = \{h_1, h_2, ..., h_{I-1}, h_I\}$의 값을 식(1)과 같이 계산한다.

$$h_i = \frac{1}{|K|} \sum_{k=1}^{|K|} B_i(k) \,. \qquad \text{식(1)}$$

식(1)에서 $K = \{\forall components \mid C=c, T=t, E=e\}$는 현재까지 분할된 모든 컴포넌트집합의 부분집합이다. $B_i(k)$는 집합 K의 k번째 글자의 i번째 픽셀 값(0 또는 1)을 의미한다. 예를 들어, 한글 완성형 2350글자를 모두 컴포넌트 분할 한 후 얻을 수 있는 $H(C=ㄹ, initial, true)$는 그림 3과 같다. 'ㄹ'이 초성인 글자는 총 23개, 즉 $|K| = 24$이다.

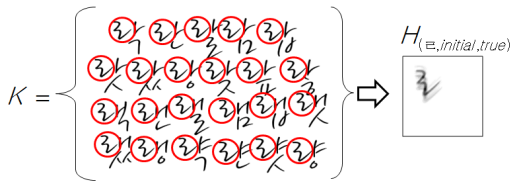


그림 3. 종성이 있는 초성 'ㄹ'의 히스토그램.

2.4 획 추출

실제 입력 글자와 글자모델 간의 구조적 매칭을 위해, 그리고 매치 결과가 거짓일 경우 적절한 컨투어 분리(split) 지

점을 찾기 위해 획을 추출하는 과정이 필요하다. 기존의 손글씨 인식을 위한 연구에서는 주로 글자 이미지를 thinning하여 스켈레톤(skeleton)의 접합점(junction point)에서 모두 분할한 후, 각 부분 획들을 연결 정보에 따라 합치는 과정을 통해 획을 추출하였다[2, 3]. 그림 4(a)는 [2]의 획 추출 및 컴포넌트 인식 결과를 보여준다. 이들은 획 추출을 위한 전처리 단계인 thinning 프로세스에서 시간이 많이 걸리며, thinning으로 인한 왜곡 등 상당부분의 에러가 여기에서 발생한다고 언급하고 있다. 본 논문의 경우 최종 목적이 매칭을 통한 인식이 아니라, 컨투어의 분리 지점을 찾는 것이기 때문에 그림 4(b)와 같이 글자의 스켈레톤으로부터 획을 추출한 후에도, 획의 볼륨을 복원하고, 스켈레톤 상의 획 분할점(1개)으로부터 가장 적합한 컨투어의 분리점 쌍(2개)을 찾아야 한다. 이를 위해 또 다른 휴리스틱이 적용되어야 하므로, 계산양이 많아진다.

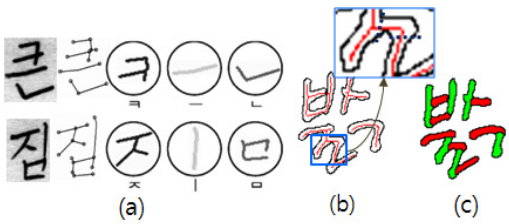


그림 4. 획 추출 방법 및 결과 비교.
(a)(b)스켈레톤 기반 방법, (c)제안한 방법.

본 논문에서는 2.3절에서 생성한 컴포넌트 히스토그램과의 융합까지 고려하여, 다음과 같은 픽셀 기반의 획 추출 방법을 제안한다.

1. N개의 컨투어로 이루어진 글자의 모든 컨투어 이미지 B^n에 대해 Distance Transform(DT)[10]을 수행한다.
2. 1의 결과에서 각 픽셀의 방향성 확률을 구한다.
3. 릴렉세이션 레이블링(Relaxation Labeling) 알고리즘을 통해 각 픽셀의 방향을 레이블링한다.
4. 같은 방향성을 가진 연결된 픽셀들을 묶어 그림 4(c)와 같은 픽셀 클러스터 집합을 만든다.
5. 4의 결과를 가지고, 입력 글자의 모든 획 레이블에 대한 후보 획을 구한다.

그림 5의 (b)와 같이 컨투어 이미지 B^n을 DT한 결과는 컨투어의 외부는 0이고, 경계로부터 내부로 갈수록 큰 값을 가지는 비트맵 T^n이다. T^n의 각 픽셀마다 그림 5의 (b-2)처럼 J(=8)개 방향으로 거리 d만큼 떨어진 픽셀의 값 t_i를 구하고, 획 레이블의 초기 확률을 구한다. 한글 획의 방향은 크게 가로, 세로, 왼쪽 내침, 오른쪽 내림, 그리고 'o'과 같은 원(circle)이 있다. 본 논문에서는 원에 대해서는 레이블링하지 않고, 4개 방향 {HORIZONTAL, VERTICAL, LEFTFALLING, RIGHTFALLING}에 대해서만 고려한다.

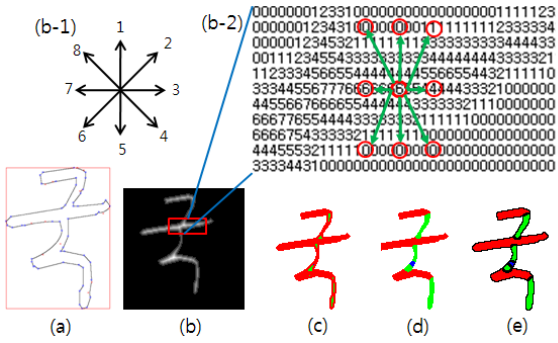


그림 5. 민윤희체 '국'의 획 추출 결과. (a) 입력컨투어, (b) Distance Transform결과, (b-1)픽셀의 8방향 확률구하기(d=4), (c) 초기 픽셀방향성 확률, (d)릴렉세이션 레이블링 결과(s=30), (e) 픽셀 방향 클러스터.

따라서 그림 5의 (b-1)의 8방향 벡터 중 1-5, 3-7, 2-6, 4-8은 같은 방향 레이블로 할당된다. T^n의 i번째 픽셀에서 방향 레이블 w_k의 확률 값은 식(2)와 같이 구한다.

$$P^1(\theta_i = w_k) \leftarrow \frac{1}{S}(t_m + t_{m'}), \quad S = \sum_{j=1}^{8} t_j . \qquad \text{식(2)}$$

m, m'은 같은 레이블로 할당되는 방향벡터 쌍을 의미하며, S는 거리 d만큼 떨어진 모든 방향의 픽셀 값의 합이다.

위와 같이 초기 확률을 구한 후에는 알고리즘 1의 릴렉세이션 레이블링을 수행한다. 알고리즘 1은 초기 픽셀 방향성 확률값의 노이즈를 제거하기 위함이므로, 별다른 호환 함수(compatibility function) 을 정의하지 않고, 8방향으로 인접한 이웃 픽셀들의 값을 참조하여 가장 빈도가 높은 방향이 그 픽셀의 방향으로 선택되도록 설계하였다. 방향 레이블링이 끝난 후에 각 픽셀에서 거리가 임계값 TH_1보다 작은 인접한 픽셀들 가운데 같은 레이블을 가진 픽셀들을 반복적으로 연결하여 픽셀 클러스터를 생성한다. TH_1의 기본 값은 2이다. 획의 두께가 매우 작은 폰트의 경우 TH_1의 값을 3으로 조정하여 너무 획이 가늘어서 클러스터가 필요이상으로 많이 생성되는 경우를 막을 수 있다.

픽셀 방향성 클러스터 집합 $C = \{c_1, \ldots, c_I\}$의 각 원소 c_i는 방향, 위치, 그리고 길이에 관한 특성(feature) 벡터 $[c_i^D, c_i^P, c_i^L]$로 표현된다. 방향특성 c_i^D는 방향 레이블링에 사용된 4개 방향에 다음과 같이 각도를 대입하였다.

HORIZONTAL=0, VERTICAL=90,
LEFTFALLING=45, RIGHTFALLING=135.

c_i^P는 클러스터의 무게중심 좌표이다. c_i^L은 클러스터의 가장 먼 두 점 사이의 거리이다.

위와 같은 과정으로 만들어진 픽셀 방향성 클러스터 중에는 하나의 획으로 보기에는 너무 작은 조각인 경우도 있고,

알고리즘 1. 픽셀 방향 클러스터링 위한 레이블링.

input: $B=\{b_1,b_2,\ldots,b_I\}$, $\Omega=\{\omega_1,\cdots,\omega_K\}$
output: P^*
initialize: $P^1(b_i,w_k)\leftarrow initial\ probability$
begin
  $s=1$
  Repeat
    for $k=1$ to K do
      for $i=1$ to I do
        $q_j^s(b_i,\omega_k)\leftarrow P^s(b_j,w_k)$;
        $Q^s(b_i,\omega_k)\leftarrow\sum_{j=1}^{J} q_j^s(b_i,\omega_k)$;
        $P^{s+1}(b_i,\omega_k)\leftarrow\frac{P^s(b_i,\omega_k)Q^s(b_i,\omega_k)}{\sum_i P^s(b_i,\omega_k)Q^s(b_i,\omega_k)}$;
      end
    end
    $s=s+1$;
  until $difference(P^{s+1},P^s)>\varepsilon$
  for $k=1$ to K do
    $i\leftarrow \arg max_i P^s(b_i,\omega_k)$;
    $P^*(b_i,\omega_k)\leftarrow 1$, $\sum_i P^*(b_i,\omega_k)\leftarrow 1$;
  end
end

하나의 획이지만, 중간에 끼어든 작은 조각 때문에 다른 클러스터로 나누어진 경우도 있다. 그래서 다음 단계에서 우리는 입력 글자의 획 레이블 집합 $L=\{l_1,\ldots,l_J\}$의 각 원소 l_j에 할당 가능한 클러스터 집합을 구하고, 이들을 연결하여 후보 획(candidate stroke) s_i를 구한다. 우리는 알고리즘 2에 기술된 바와 같이 그리디 탐색(greedy search) 알고리즘으로 획 레이블 l_j의 후보 획을 구하였다.

알고리즘 2에서 그래프 $G=(C,E)$의 노드는 각 픽셀 방향성 클러스터로서 $C=\{c_1,\ldots,c_I\}$이며, E는 클러스터들 간의 연결성 정보에 대한 행렬(adjacency matrix)이다. 클러스터c_i와 $c_{i'}$는 다음 조건을 모두 만족할 때 연결 가능한 클러스터로 정의된다.

(1) c_i와 $c_{i'}$의 최소 거리가 임계값 TH_2 보다 작고,

(2) $\| c_i^D - c_{i'}^D \| < TH_3$ 이고,

(3) $c_i^L + c_{i'}^L - \mu_j^L \le TH_4$ 일 때.

우리는 컨투어 이미지 크기가 120×120 일 때, 클러스터 사이의 적절한 거리 차는 $TH_2=5$ 임을 경험적으로 구할 수 있었다. 두 클러스터의 각도 차이에 관한 임계값은 $TH_3=\angle 90$로 정하였다. 위의 마지막 조건에서 $TH_4=1.5\,\mu_j^L$이다. μ_j^D, μ_j^P와 μ_j^L은 글자모델에서 주어지는 획의 표준 방향, 위치, 길이벡터이다. μ_j^P와 μ_j^L은 각 획이 속하는 컴포넌트의 위치 및 크기에 상대적인 비율로 주어진다. $OPEN$ 집합은 l_j의 위치와 방향 조건을 만족하는 초기 부분 획을 저장한다. $OPEN$ 집합의 초기 부분 획에 대해서 연결 가능한 부분 획들을 연결해 나간다. 최종적으로 $CLOSED$ 의 마지막 원소가 획 레이블 l_j에 대한 후보 획 s_i이 된다.

알고리즘 2. 획 레이블의 후보 획 검색 알고리즘.

input: $OPEN$, $CLOSED$, $G=(C,E)$.
output: $CLOSED$.
initialize: $OPEN$, $CLOSED\leftarrow\varnothing$.
begin
  for $j=1$ to J do
    for $i=1$ to I do
      if $\| c_i^P-\mu_j^P \| \le 0.5*CompSize$ and $\| c_i^D-\mu_j^D \| < \angle 90$ then
        $OPEN\leftarrow c_i$
      end
    end
  end
  for $k=1$ to $|OPEN|$ do
    $c_{new}\leftarrow c_k \in OPEN$;
    repeat
      $c_{old}\leftarrow c_{new}$;
      $CLOSED\leftarrow c_{old}$;
      $c_{new}\leftarrow c_{old}+c_{k'},E$;
    until $difference(c_{old}^L,c_{new}^L)>\varepsilon$
  end
end

3. 통계적-구조적 컴포넌트 분할

본 절에서는 2절에서 설명한 컴포넌트 히스토그램과 입력 글자의 획 레이블에 대한 후보 획 집합을 기반으로 한글 글립의 초성, 중성, 종성 컴포넌트를 분할하는 방법을 구체적으로 기술한다. 그림 1은 전체 컴포넌트 분할 과정의 예를 보여준다.

3.1 위치에 기반 한 컴포넌트 분할

한글의 초·중·종성 컴포넌트는 한 글자 안에서 상대적으로 일정한 위치에 나타난다. 그러므로 가장 기본적인 컴포넌트 분할은 각 컴포넌트가 가장 많이 나타나는 최소 경계사각형(Minimum Bounding Rectangle 또는 Bounding Box) 영역이 기준이 된다. 이 정보를 이용하기 위해 우리는 2.3절에서 컴포넌트 히스토그램을 생성하였다.

컴포넌트 히스토그램 기반의 분할과정은 다음과 같다. 먼저 입력 글자의 글자 코드를 초성, 중성, 종성으로 분해하여, 해당 자소의 중성 타입과 종성유무 조건에 맞는 히스토그램을 가져온다. 예를 들어 입력 글자가 '국'이라면 한글코드 조합규칙을 역으로 적용하여 초성 'ㄱ', 중성'ㅜ', 종성'ㄱ'으로 분해됨을 알 수 있다. 초성 'ㄱ'에 대한 히스토그램 중에서 중성타입이 'ㅜ'이고 종성이 존재하는 경우에 해당하는 히스토그램 $H_{initial}$을 가져온다. 이 때 컨투어 그룹 B_k가 초성이 될 확률은 식(3)과 같이 구한다.

$$P_k^{initial} = \sum_i^I h_i b_i. \qquad \text{식(3)}$$

각 픽셀에서의 확률 값은 단순히 히스토그램 값 h_i와 컨투어 비트맵의 값 b_i(내부 1, 외부 0)의 곱 $p_i = h_i b_i$이 된다. 최종적으로 컨투어 그룹 B_k의 컴포넌트 타입 T_k는 다음과 같다.

$$T_k \leftarrow \arg max_{ctype}\, P_k^{ctype}.$$

즉 $P_k^{initial}$, P_k^{medial}, 그리고 P_k^{final} 중 가장 큰 확률을 가지는 컴포넌트로 할당한다.

3.2 컴포넌트 분할 검증(Validation)

우리는 분할된 컴포넌트들이 다음과 같은 조건을 만족하면 완전한 분할(complete decomposition)이 이루어진 것으로 결정한다.

(1) 컨투어 하나가 두 개 이상의 컴포넌트로 중복 할당 되지 않으며,
(2) 각 컴포넌트가 필수 획들을 모두 가지고 있고, 불필요한 획을 가지지 않을 때.
(3) 2를 만족하지 않지만, 세 컴포넌트의 히스토그램 매칭 스코어가 임계값 이상 일 때 (1에 가까울 때).

히스토그램과 컴포넌트 간의 매치 스코어(match score)는 [0,1]의 값을 가지며, 다음과 같이 구한다.

$$match\,score(H,B) = \frac{H \cap B}{H \cup B}.$$

그러나 조건 3은 히스토그램이 초기값을 가지고 있을 때는 적용할 수 없고, 데이터 개수($|K|$)가 2이상일 때만 적용한다.

본 논문에서 컴포넌트 분할 및 검증 시 순서는 중성-종성-초성 순이다. 중성 컴포넌트의 획들이 일반적으로 가장 길이가 길고, 획 수도 적으며, 획 간의 관계도 상대적으로 명확하다. 또한 중성 컴포넌트 히스토그램이 가장 빈도가 높기 때문에 가장 신뢰할 수 있다고 볼 수 있다. 다음으로 종성 컴포넌트, 초성 컴포넌트 순으로 분할 및 검증을 수행하는데, 이 또한 히스토그램에 누적된 데이터의 양이 평균적으로 종성이 많기 때문이다.

3.3 릴렉세이션 레이블링(Relaxation Labeling)

위치에 기반 한 컴포넌트 분할이 검증 결과 완전한 분할이 이루어지지 않았다면, 하나의 컨투어에 두 개 이상의 컴포넌트의 부분들이 합쳐져 있거나, 어느 컴포넌트의 획이 초기 컴포넌트 히스토그램의 위치를 벗어나 존재하는 경우에 해당한다. 이를 보완하기 위하여 본 논문에서는 알고리즘 3과 같은 릴렉세이션 레이블링 알고리즘을 적용한다.

2.4절에서 추출한 후보 획 집합 $S = \{s_1, s_2, ..., s_I\}$ 와 획 레이블 집합 $L = \{l_1, l_2, ..., l_J\}$ 이 존재할 때, 컨투어 s_i가 각 레이블에 대해 가지는 확률값의 집합은 다음과 같이 나타낼 수 있으며, 그 합은 1이다.

$$\{P(s_i,l_1), P(s_i,l_2), ..., P(s_i,l_J)\}, \qquad \sum_{j=1}^{J} P(s_i,l_j) = 1.$$

$P(s_i,l_j)$는 후보 획 s_i가 l_j로 레이블링 될 확률을 나타낸다. 알고리즘 3에서 $P_i^t(j)$은 릴렉세이션 레이블링 과정을 t번 반복했을 때의 $P(s_i,l_j)$를 나타낸다. 초기 확률 $P_i^1(j)$은 2.4절에서 추출한 획의 초기 레이블에 대해 1, 나머지 레이블에는 0을 넣어준다. 단, 이 때 그림 6과 같이 획 내부의 각 픽셀들에 컴포넌트 히스토그램 상의 확률을 적용했을 때, 하나의 가로획 또는 세로획이 명백하게 확률분포가 이분화 될 때, 획을 다시 분할한다.

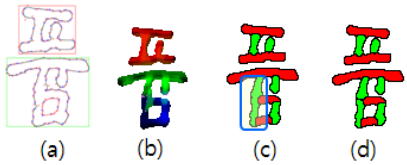


그림 6. (a)입력컨투어, (b)히스토그램의 확률분포, (c) 획 추출 결과, (d) b를 고려한 획 재분할.

$R_i(j)$은 후보 획 s_i의 레이블 l_j에 대한 호환성(compatibility)을 나타낸다. 이미 알고리즘 2에서 호환성 검사를 마쳤고, 초기 레이블로 반영되어 있으므로, 알고리즘 3에서 실제로 계산하지는 않는다. $R_{i,i'}(j,j')$는 s_i와 s_i의 이웃 획 사이의 관계에 관한 함수이다. 이웃 획이란 두 획 사이의 최소거리가 $\min dist(s_i,s_{i'}) < TH_2(=5)$인 인접한 획들을 의미한다. 만약 획 레이블간의 관계와 입력 후보 획 간의 관계가 서로 불일치할 때(inconsistent), 그러한 관계에 대해서는 값을 감쇄시킨다.

$$W(j,j') = \begin{cases} 0, & \text{if } j,j' \textit{ consist with } i,i', \\ P(j'|j), & \text{if } j,j' \textit{ inconsist with } i,i'. \end{cases}$$

알고리즘 3. 획 레이블링 및 컴포넌트 레이블링.

input: $S = \{s_1, s_2, ..., s_I\}$, $L = \{l_1, l_2, ..., l_J\}$
output: P^*
initialize: $P_i^1(j) \leftarrow$ *initial labeling*,
$R_{i,i'}(j,j') \leftarrow 2 - W(j,j') - W(s_i,s_{i'}|j,j')$.
begin
$t = 1$
Repeat
for $j = 1$ to J do
for $i = 1$ to I do

$$Q^t(b_i,\omega_k) \leftarrow R_i(j) + \sum_j \max_{i'} R_{i,i'}(j,j') P_{i'}(j');$$

$$P_i^{t+1}(j) \leftarrow \frac{P_i^t(j) Q_i^t(j)}{\sum_i P_i^t(j) Q_i^t(j)};$$

```
    end
  end
  t = t + 1;
  until   difference(P^{t+1}, P^t) > ε

  for j = 1 to J do
    i ← argmax_i P_i^t(j);
    P_i^*(j) ← 1,  Σ_i P_i^*(j) ← 1;
  end
end
merge l_j to component type
```

3.4 외곽선 분리 (Contour Splitting)

두 개 이상의 컴포넌트가 하나의 외곽선으로 이루어진 경우, 외곽선을 분리해야만 컴포넌트 분할이 가능하다. 외곽선 분리 과정은 다음과 같다. 먼저 컴포넌트 간 경계 지점에서 거리가 임계치 이하인 점들의 집합 중 다음을 만족하는 컨투어 분리점 쌍 (p_1,p_2)을 찾는다. 컨투어 분리점 쌍은 베지어 커브 상의 모든 컨트롤 포인트들로 라인벡터를 생성했을 때, 서로 대응관계 있는 라인 상에 존재하는 온 커브 컨트롤 포인트 쌍(on-curve control points pair)이다. 우리는 두 라인이 이루는 각도의 절대치가 임계값(30도) 이하이고, 두 라인 사이의 거리가 임계값(20) 이하일 때, 서로 대응하는 라인으로 본다. 두 점이 대응라인벡터 상에 존재할 때, 아래 식을 가장 크게 만드는 점의 쌍(p_1,p_2)을 구한다.

$$\arg max_{(p_1,p_2)} \frac{\tan(p_1)+\tan(p_2)}{dist(p_1,b)+dist(p_2,b)+dist(p_1,p_2)}$$

위 식에서 b는 컨투어 안에 존재하는 컴포넌트 경계(border)를 의미한다. 이와 같이 하나의 경계라인에 대해 가장 적합한 컴포넌트 분리점 한 쌍 씩을 구한다.

분리점 쌍이 구해지면, 다음과 같이 컨투어를 분리한다. p_1에서 시작하여 반시계방향(CCW)으로 p_2까지 이어지는 컨투어를 생성하고, 닫힌 컨투어를 위해 컨투어 마지막 점으로 첫 번째 컨트롤 포인트인 p_1을 추가한다. 마찬가지로 p_2에서 시작하여 반시계방향으로 p_1까지 이어지는 컨투어를 생성한다.

$$\begin{cases} p_1 \rightarrow CCW \rightarrow p_2 \rightarrow p_1 : contour1 \\ p_2 \rightarrow CCW \rightarrow p_1 \rightarrow p_2 : contour2 \end{cases}$$

반면에 분리점 p_1, p_2 중 하나가 외부컨투어의 점이고, 다른 하나는 내부컨투어의 점이되는 경우, 즉 두 개 이상의 컨투어로 이루어진 컨투어 그룹을 분리할 경우에는 분리 결과 1개의 컨투어만 생성하게 된다. 아래과 같이 p_1에서 시작하여 시계반대방향으로 진행하다가 다시 p_1, p_2를 거쳐 시계방향으로 진행하다가 마지막으로 p_1에서 컨투어가 닫히게 된다.

$$p_1 \rightarrow CCW \rightarrow p_1 \rightarrow p_2 \rightarrow CW \rightarrow p_1$$

분리된 외곽선들은 닫힌(closed) 상태가 아니므로, 컨투어의 시작점과 끝점을 연결하여 닫힌 컨투어로 만들어 주어야 한다. 이 때 적절한 위치에 오프 커브 컨트롤 포인트를 추가하면 끝모양이 보다 자연스러운 획으로 복원할 수 있다. 본 논문에서는 아래 그림 7과 같이 획 분리점 쌍이 $(p_1,\ p_2)$ 일 때, 선분 $\overline{p_1p_2}$중점인 $d/2$ 지점에서 수직으로 $w=5$ 만큼 이동한 직선 $\overrightarrow{p_1p_2}$ 에 평행한 직선과 p_1, p_2의 각 tangent 벡터가 만나는 점 off_p_1과 off_p_2를 p_1, p_2 사이에 오프 커브 컨트롤 포인트로 추가하는 방법을 제안한다.

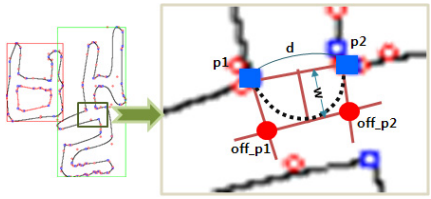


그림 7. 획 분리 시 획 끝 모양 복원.

4. 실험 및 토의

본 논문에서는 컴포넌트 분할 실험과 결과 검증을 위해 한글 폰트의 컴포넌트의 자동 및 수동분할을 위한 인터페이스를 그림 8과 같이 제공한다.

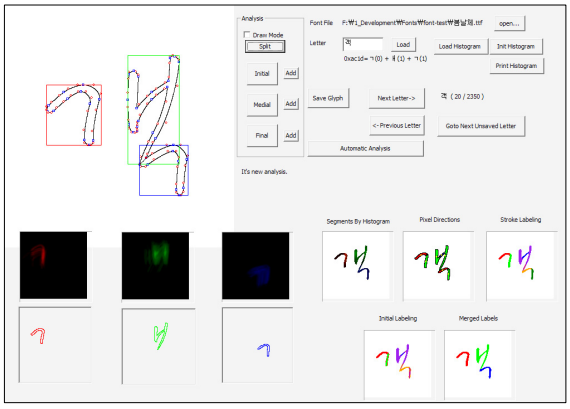


그림 8. 폰트의 컴포넌트 분할 사용자 인터페이스.

한글 폰트로부터 해당 코드의 글립을 로딩할 때, 글자의 크기는 100×100으로 가져오고, 컨투어가 글자 이미지 영역을 벗어나지 않도록 너비와 높이에 각 20 픽셀씩의 마진을 두어 120×120 크기의 글자 이미지를 생성하였다. 폰트마다 글자의 평균 크기와 중심점이 다르지만, 그것 또한 그 폰트의 특징으로 볼 수 있으므로, 별도의 이미

지 정규화는 하지 않았다.

수동 분할에서는 입력글자에 대한 시스템의 초·중·종성 결과가 잘못된 분할인 경우, 사용자가 컨투어 분리점쌍을 직접 찾아 라인을 그려주고 초성, 중성, 종성에 해당하는 컨투어 그룹을 직접 선택할 수 있게 하였다.

자동 분할의 경우, 폰트 파일이 입력으로 들어오면 한글 완성형 글자 2,350개에 대해 순차적으로 분할을 시작한다. 만약 어떤 글자의 분할이 검증 결과 완전한 분할이 이루어지지 않았다면, 건너뛰고 명백하게 분할이 이루어지는 글자들부터 분할하여 히스토그램을 업데이트 시킨다. 입력 글자들에 대해 한 번의 수행을 다 끝내고 나서는 분할이 되지 않았던 글자들에 대해 다시 한 번 분할 프로세스를 수행한다. 두 번째 루프에서는 처음보다 히스토그램의 데이터 수가 많기 때문에 거의 대부분의 글자에서 매치 스코어를 사용한 검증이 허용된다.

표 2는 2.4절의 획 추출 결과 및 컴포넌트 분할 예를 보여준다. 표의 지영체, 세희체, 크로키체처럼 일반적으로 획의 굵기가 가늘고 균일한, 즉 직선의 성질이 강한 획을 가진 폰트일수록 획 추출 정확도가 높다. 반면 획

표 2. 픽셀 방향성클러스터 및 컴포넌트 분할 결과 예.

폰트명	픽셀 클러스터 예				분할 예
봄날체	글	죽	팃	핵	
소녀체	계	콤	뱀	필	
민윤희체	냇	탉	콕	퐝	
지영체	겋	끊	벱	툭	
세희체	겜	궜	젤	뻥	
흥수체	숩	뺌	쿳	태	
형오체	뻠	쑥	엘	짝	
지희체	릴	칩	펙	홈	
아혜체	뻥	콤	필	훌	
소설가체	껙	돌	줌	쌉	
효봉개똥체	갈	봄	툭	팔	
크로키체	갈	곽	몲	퓐	

이 굵고, 한 획 내에서 굵기 차이가 크며, 곡선의 성질이 강한 폰트의 경우, 그림 9의 (a)산돌02체와 (b)효봉개똥체와 같이 픽셀 클러스터가 제대로 구해지지 않아 획 추출에 실패하는 경우가 있다. 이 경우에도 이전의 글자를 분석하면서 축적된 히스토그램 정보에 의해 초·중·종성컴포넌트는 정확히 분할될 수 있다. 그림 9(b)의 '갗'은 '갖'이나 '돛'과 같이 컴포넌트 분할이 제대로 이루어진 다른 글자들에 의해 만들어진 초성'ㄱ', 중성'ㅏ', 종성'ㅊ' 컴포넌트 히스토그램에 의해 분할된다. (a)의 '횐'의 경우에도 '획', '괸'과 같이 명확히 분리되는 컴포넌트의 모양정보를 바탕으로 분할이 될 수 있다. 그러나 본 시스템에서는 '횐'과 '획'은 중성이 다르므로 그 모양, 크기, 위치가 약간씩 차이가 나서 서로 다른 히스토그램에 저장하고 있다. '획'의 'ㅎ'을 가져와서 '횐'의 분할에 사용할 때에는 적절한 정규화가 필요하다.

그림 9. 획 추출 실패 예: (a)산돌02체, (b)효봉개똥체.

산돌02체와 같이 분할 정확도가 떨어지는 폰트에 대한 가장 손쉬운 해결방법은 사용자가 분할이 실패한 글자 집합의 대표적인 글자 몇 개를 수동으로 분할하여 정확한 컴포넌트 모양 정보가 히스토그램에 입력되도록 하는 것이다. 예를 들어, '횐'을 수동으로 분할하면, 같은 문제로 분할에 실패했던 '획','휠','횝','횟','횡'은 모두 히스토그램에 의해 제대로 분할된다.

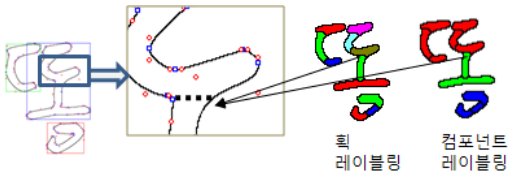


그림 10. 적절한 컨투어 분리점이 없는 경우.

획 추출까지 성공했다 하더라도 그림 10처럼 적절한 컨투어 분리점이 없어 획 분리에 실패하는 경우도 있다. 이 경우, 분리 지점에 온 커브 컨트롤 포인트를 추가해 주어야 컨투어 레벨에서 제대로 된 분할이 이루어질 수 있다.

맑은고딕체, 굴림체, 바탕체과 같은 한글 조합형 폰트는 운영체제 및 워드프로세서에서 제공하며 그 수가 제한적이다. 반면 손글씨 폰트 및 최근에 웹을 통해 배포되는 대부분의 한글폰트는 한글 완성형 2,350자에 대한 글립만 가지고 있다. 그래서 예를 들면 '똠'과 같은 글자는 완성형 폰트로는 표현할 수 없다. 하지만, 우리는 분할된 각 컴포넌트를 재조합함으로써 그림 11과 같이 완

성형 폰트에 없는 글자 글립을 만들어 낼 수 있다.

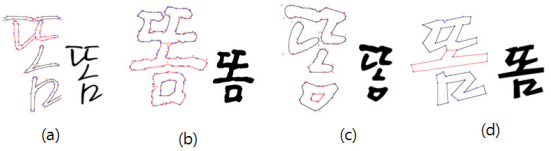


그림 11. (a)봄날체, (b)소녀체, (c)흥수체, (d)독도체의 새 글자 '똠'의 생성.

따라서 본 논문에서 제안한 방법으로 분할된 컴포넌트를 가지고 약간의 변형 및 공간 재배치를 통해 재구성하면, 이탤릭(italic)체나 볼드(bold)체처럼 같은 폰트패밀리지만 다소 다른 느낌의 폰트를 생성할 수 있다. 표 3과 같이 모든 글자에 대한 분석이 모두 끝난 후 얻을 수 있는 최종 컴포넌트 히스토그램은 그 폰트에서 각 컴포넌트의 위치와 모양의 분포를 보여준다. 이로부터 추출한 컴포넌트의 위치, 크기, 모양 특성 벡터 중 나머지는 고정하고 하나의 특성 벡터에 임의의 변형을 줌으로서, 새로운 느낌의 폰트를 만들 수 있다.

5. 결론

본 논문에서는 한글 폰트의 특성을 분석하기 위해 글자를 컴포넌트 단위로 분할하여 지역적인 모양특성과 컴포넌트 간 위상정보를 함께 고려함으로써 보다 효율적이고 정확한 폰트 분석을 수행하고자 하였다. 영문알파벳 등과 달리 한글은 단순히 글자 대 글자로 매칭을 수행하게 되면 유사도도 낮을뿐더러, 컴포넌트 조합에 의해 글자의 수가 기하급수적으로 많아지기 때문에 컴포넌트 분할이 꼭 필요하다.

분할된 컴포넌트들은 외곽선 폰트의 글자 글립과 마찬가지로 베지어곡선으로 표현된 컨투어 집합으로 표현되므로 분할 결과가 비트맵방식으로 저장되는 것에 비해 데이터의 양이 매우 적으며 해상도에 관계없이 높은 질의 컴포넌트 이미지를 얻을 수 있다는 장점을 가진다. 뿐만 아니라, 컴포넌트 재구성을 통해 바로 새로운 글자의 추가 및 새로운 폰트 생성이 가능하였다. 이 방법을 손글씨 이미지에 적용하면 개인 글씨체의 분석 및 손글씨 생성(handwriting synthesis)에 활용할 수 있다.

컴포넌트 분할이 끝난 후 가장 쉽게 적용할 수 있는 한글 폰트 분류 방법은 기존의 Eigen Faces 또는 Eigen Fonts 방법과 유사하다[11, 12]. 즉 수많은 폰트에 대해서 각 컴포넌트 별로 PCA(principal component analysis)를 적용, 그 컴포넌트를 가장 잘 표현하는 고유벡터인, Eigen Component를 구한다. 두 폰트 사이의 유사도(similarity)는 모든 컴포넌트들의 특징 벡터들 사이의 거리(Euclidean distance)로 계산한다. 우리는 향후 연구에서 대용량 폰트 데이터베이스에서의 폰트 분류 및 검색을 연구하고자 한다. 폰트 분류/검색 시스템은 폰트 제작 및 사용 시 뿐 아니라, 다양한 폰트관련 연구-예를 들면 폰트의 감성과 조형적 특징 간의 상관관계에 대한 연구 등-에도 활용될 수 있을 것이다.

본 논문에서는 미리 학습된(learned) 히스토그램을 사용하지 않았다. 그런데, 컴포넌트가 완전히 분할되면 히스토그램이 갱신되고, 갱신된 히스토그램이 다시 다음 글자 분석에 이용되므로, 일종의 학습 효과가 있었다고 볼 수 있다. 실제로 사용자가 몇 개의 샘플 글자를 수동으로 분할하여 초기 히스토그램을 만들어준 후, 나머지 글자들에 대해 수행을 하였더니 보다 빠르고 정확한 분할이 이루어졌다. 그래서 우리는 향후 연구로 RDT(Randomized Decision Tree) 또는 CRF(Conditional Random Field) 등의 방법을 적용하여 각 한글 컴포넌트의 구조적 정보(획의 위치, 길이, 방향, 다른 획과의 관계)를 학습함으로써, 분할 성능을 높이고자 한다.

표 3. 컴포넌트 픽셀 히스토그램 (봄날체, 중성컴포넌트).

중성	ㅏ		ㅐ		ㅑ		ㅒ		ㅓ		ㅔ		ㅕ	
종성유무	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음
빈도	19	223	16	153	17	60	4	11	19	193	19	121	18	113
봄날체														
중성	ㅖ		ㅗ		ㅘ		ㅙ		ㅚ		ㅛ		ㅜ	
종성유무	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음
빈도	15	29	19	165	18	63	13	29	19	82	17	48	19	158
봄날체														
중성	ㅝ		ㅞ		ㅟ		ㅠ		ㅡ		ㅢ		ㅣ	
종성유무	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음	없음	있음
빈도	17	39	16	25	18	87	17	67	19	147	8	22	19	166
봄날체														

참고 문헌

[1] I. Biederman, "Recognition-by-Components: A Theory of Human Image Understanding," *Psychological Rev.*, vol. 94, no. 2, pp. 115-147, Apr. 1987.

[2] K.W. Kang and J.H. Kim, "Utilization of Hierarchical, Stochastic Relationship Modeling for Hangul Character Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 9, pp. 1185-1196, Sep. 2004.

[3] J. Zeng and Z.Q. Liu, "Markov Random Field-Based Statistical Character Structure Modeling for Handwritten Chinese Character Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 5, pp. 767-780, May 2008.

[4] Apple Computer, Inc. The TrueType Font Format Specification, Version 1.0, 1990.

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

[6] D.E. Knuth, "METAFONT: The Program," *Addison Wesley*, 1986.

[7] Bitstream Inc., "Stroke-Based Fonts : Compact, High-Quality Chinese, Japanese, and Korean Fonts for Embedded Systems," Technical Paper of Bitstream Inc., 2005.

[8] E.J.Jakubiak, R.N. Perry, S.F. Frisken, "An Improved Representation for Stroke-based Fonts," *ACM SIGGRAPH*, Article 137, July 2006. (ACM Press, TR2006-119)

[9] 넥서스폰트 (NexusFont), http://xiles.net/downloads/#NexusFont.

[10] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance Transforms of Sampled Functions," Cornell Computing and Information Science Technical Report TR2004-1963, 2004.

[11] M. Turk, A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

[12] M. Solli and R. Lenz, FyFont: Find-your-Font in Large Font Databases, In Proceedings of the 15th Scandinavian conference on Image analysis, pp. 432-441, 2007.

〈저자소개〉

구상옥

- 2000년 경북대학교 컴퓨터공학과 공학사
- 2003년 경북대학교 컴퓨터공학과 공학석사
- 2004년~현재 경북대학교 컴퓨터공학과 박사과정
- 관심분야: Shape Analysis, Shape Deformation, Computer Graphics, Artificial Intelligence, Computer Vision.

정순기

- 1990년 경북대학교 컴퓨터공학과 공학사
- 1992년 한국과학기술원 전산학과 이학석사
- 1997년 한국과학기술원 전산학과 공학박사
- 1997~1998년 University of Maryland, Research Associate
- 2001~2002년 IRIS, University of Southern California, Research Assocoate
- 2008~2009년 IRIS, University of Southern California, Visiting Faculty
- 1998~현재 경북대학교 IT대학 교수
- 관심분야: Virtual Reality, Artificial Intelligence, Computer Vision, Image Processing, Computer Graphics.