한글 트루타입폰트 및 손글씨의 자동 획 분할 알고리즘

곽윤석[1], 구상옥[2], 정순기[2]
경북대학교 전자전기컴퓨터학부[1], 경북대학교 컴퓨터공학과[2]
yskwak@vr.knu.ac.kr sokoo@vr.knu.ac.kr skjung@knu.ac.kr

Automatic Stroke Extraction of TrueType Font and Handwriting of Hangul

Yoon Seok Kwak[1], Sang Ok Koo[2], Soon Ki Jung[2]
Graduate school of Electrical engineering and computer science Kyungpook National University[1], Department of Computer Engineering Kyungpook National University[2]

요 약

본 논문에서는 한글 글립(glyph)의 형태학적 분석을 통해 자동으로 획을 분할하는 방법을 제안한다. 제안된 방법은 thinning된 한글 글립의 골격(skeleton) 이미지를 기반으로, 획 분리, 획 병합, 그리고 획 볼륨 복원의 세가지 단계를 거쳐 한글의 기본 획들을 추출해 낸다. 실험 결과, 트루타입폰트(TrueType Font)에 대해서는 80%, 손글씨(Handwriting) 글립에 대해서는 72%의 획 분할 정확도를 보였다. 본 논문에서 제안한 방법으로 획득된 획 정보를 이용하여, 향후 한글 손글씨 생성을 위한 연구를 하고자 한다.

1. 서 론

컴퓨터의 보급과 더불어 편지나 레포트, 논문, 메모 등의 글씨를 쓰는 것과 관련된 많은 일들이 여러 편리성 때문에 컴퓨터를 이용해 작성하는 일들이 대중화되었다. 디지털 폰트는 이러한 글을 쓰는 것에 대한 수단으로 사용되고 있다. 그러나 최근 디지털 폰트의 획일성과 표현의 한계로 인해 개인의 개성을 표현할 수 있는 손글씨에 대한 관심이 높아지고 있다. '캘리그래피(Calligraphy)'라는 서예를 응용한 디자인 장르가 영화포스터, 음반, 책표지, 광고 등에 널리 보급되면서, 단순히 정보를 전달하기 위한 텍스트로서 뿐만 아니라 감정이나 아름다움을 표현하기 위한 수단으로서 글씨체의 중요성이 재인식 되고 있다.

다양한 폰트에 대한 사용자의 요구가 높아지면서, 자신만의 개성이나 목적에 맞게 서체를 직접 개발하는 경우가 늘어나고 있다. 온라인 커뮤니티 상에서 웹폰트라는 컨텐츠의 인기가 이를 반증한다. 높은 인기에도 불구하고 서체 개발에는 많은 시간과 경험이 요구되기 때문에, 개인의 손글씨를 디지털 폰트화 하고 싶다면 디자인 서비스 업체에 주문을 하여 개발하는 식이다[1, 2, 3].

손글씨 생성 관련 연구로는 Lin의 스타일을 보존하는 영어 필기 생성에 대한 연구가 있다[4]. 이 연구는 온라인 필기 입력 샘플을 수집한 후, 수집한 샘플로부터 스타일을 추출해, 글자 및 글자군 순으로 단계적으로 스타일을 적용하는 방법으로 손글씨를 생성하였다. 하지만, 이 연구는 한글과 조형적 특징이 다소 다른 영어만을 대상으로 하였기 때문에, 한글에 적용되기 위해서는 한글에 맞는 스타일 추출 및 손글씨 생성 방법이 필요하다.

본 논문에서는 자동으로 손글씨를 생성해 내기 위한 이전 단계로서, 한글 글립의 조형적 특징과 형태학적 분석(Morphological Analysis) 방법을 이용하여 자동으로 획을 분할하는 방법을 제안한다. 제안한 획 분할 알고리즘은 컴퓨터 운영체제에서 가장 널리 사용되는 디지털 폰트인 트루타입 폰트 (TrueType Font)와 여러 사람들로부터 타블렛(Tablet) 입력장치를 통해 직접 입력 받은 손글씨들의 획을 자동으로 분할하는 데 적용되었다.

디지털 폰트 관련 연구는 1990년대 곡선(curve)에 관련된 연구들이 주를 이룬다[5,6,7,8]. 초기에는 임의의 점의 집합을 베지어 곡선(Bézier curve)으로 표현하는 문제에 관한 연구들이 많았고, 이러한 기존 연구를 바탕으로 한 최근 연구 중에는 Park이 베지어 커브 근사화를 이용하여 PDA 등의 장치에서 손으로 그린 그래픽 메시지를 표현하는 방법을 제안하였다[9]. 트루타입폰트 개발과 관련된 연구로는 자동 힌팅(Hinting)에 관한 연구가 있다. 힌팅 기술은 트루타입 폰트를 이용해 임의의 크기의 글자를 래스터라이저(Rasterizer)를 통해 생성할 때, 획의 굵기가 일정치 않거나, 획이 사라지거나, 기울어진 직선이나 곡선 부분의 중간이 끊어지는 현상을 해결하기 위한 기술이다. Zongker는 힌팅 정보가 들어있는 기존 영어 트루타입폰트의 힌팅 정보를 기반으로 새로운 트루타입폰트의 힌팅 정보를 생성하기 위한 방법을 제안하였다[10].

획 분할 (Stroke Segmentation) 관련 연구로는 Y. Kato & M. Yasuhara의 단일획(Single Stroke) 이미지의 획 순서를 복원하는 연구가 있으며[11], 스캐닝을 통해 입력 받은 손글씨 이미지를 그래프 이론으로 표현하고 중복으로 지나치는 에지에 대한 처리 후 한붓그리기를

이용해 순서를 복원한다. 그러나 커브가 많은 하나의 획에 대한 알고리즘 이므로, 한글 손글씨에 동일하게 적용할 수 없다.

그리고 Koo 등의 한자를 대상으로 한 사용자의 입력을 통해 정확한 획순을 얻어내는 연구가 있었다[12,13]. 이 방법은 결과가 정확하긴 하지만 사용자의 입력이 필요하다는 단점이 있다. 그 후 사용자 입력 없이 한자폰트에서 기본 획을 이루는 대응벡터를 찾고 기하학적 분석을 통해 획을 자동으로 추출하는 연구도 있었다[14,15]. 이 방법은 고딕체나 굴림체와 같이 획의 두께 변화가 적은 폰트의 획 분할 및 획 순서 부여에 대해서 매우 높은 정확도를 보인다. 그러나 한양해서체나 궁서체와 같이 획에 곡선이 많고 두께 변화가 심한 폰트에 적용하는데 한계가 있다.

본 논문에서는 트루타입폰트와 한글 손글씨 글립(Glyph)에 대한 형태학적 분석을 통해 글립의 골격(Skeleton)을 찾아낸 후, 자동으로 획을 분할하는 방법을 제안한다. 2장에서는 본 논문에서 제안하는 방법을 이해하기 위해 필요한 배경 지식을 설명하고, 3장에서는 제안하는 획 분할 방법에 대해 설명한다. 4장에서는 획 분할의 정확도 및 다양한 분할 결과를 보여주고, 마지막으로 5장에서는 결론 및 향후 연구를 제시한다.

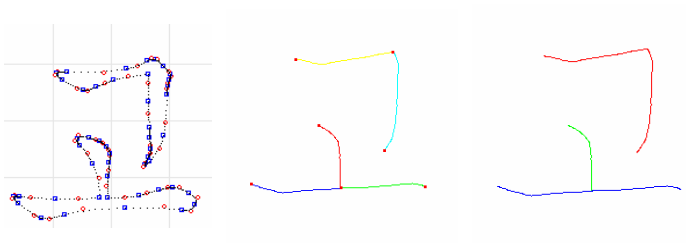
2. 연구 배경

본 논문에서 제안하는 한글 획 분할 방법은 한글이 가지는 형태학적 정보에 기반하여 이루어진다. 이는 임의의 폰트 외에도 스캐닝을 통해 입력 받은 한글 손글씨에도 적용할 수 있는 방법으로, 이 장에서는 한글 형태 정보를 분석하는데 필요한 기본 지식을 설명하고 기존의 형태적 정보를 이용하는 방식에 대해 설명한다.

2.1 트루타입폰트(TrueType Font)

트루타입폰트는 현재 윈도우 같은 대중적인 운영체제에서 가장 많이 쓰이는 폰트 기술로써, 글자 정보를 베지어곡선 형태로 담고 있는 벡터 폰트 또는 아웃라인 폰트 중 하나로 비트맵 폰트에 비해서 적은 용량에 많은 폰트 정보를 저장할 수 있고, 글자 크기가 달라져도 정보의 양이 달라지지 않는다. 트루타입폰트는 별도의 폰트 래스터라이저를 통해 글자를 렌더링 하게 된다.

그림 1(a)는 한양해서 폰트에서 '고' 자를 출력한 결과이다. 파란색 점은 커브 상(on-curve)의, 빨간색 점은 커브 외(Off-curve)의 조절점(Control Point)들로, 트루타입폰트에 저장되어 있는 글자 글립의 외곽선(Contour) 정보이다. 본 논문에서 제안하는 획 분할 방법은 그 입력으로서 트루타입폰트의 글자를 래스터라이징(Rasterizing)한 이미지를 thinning 하여 얻은 글립의 골격 정보를 이용한다. 그림 1(b)는 '고'자를 thinning한 결과이며, 그림 1(c)는 이러한 골격 상에서 획을 분할한 결과이다.



(a) (b) (c)

그림 1. '고' 자의 트루타입폰트(a)와 골격정보(b) 및 분할된 획 정보(c).

2.2 형태학적 분석(Morphological Analysis)

평면상에서 어떤 모양(shape)의 형태적 정보를 알아내는 방법에는 전통적으로 thinning, MAT(Medial Axis Transform, Voronoi diagram 등의 많은 방법들이 있다. Thinning은 평면상에서 대상 물체의 표면을 조금씩 벗겨내어 골격(skeleton)을 찾는 방법이고, MAT은 어떤 모양의 경계선을 이용해 중심축을 찾는 방법인데, 계산시 많은 시간이 소요되고, 노이즈에 민감하다는 단점을 가지고 있다.

본 논문에서는 Zhang & Suen[16]의 thinning 알고리즘을 적용해 각각의 한글 글립에 대한 골격을 구한다. 이 방법은 이진화된 이미지를 대상으로 하며, 빠르고 정확한 결과를 얻을 수 있어 널리 사용되는 thinning 알고리즘 중 하나로, 이미지 크기만큼의 버퍼를 추가적으로 사용해 현재의 처리 결과가 다음의 처리 결과에 전혀 영향을 미치지 않는 병렬 처리 알고리즘 중의 하나이다. 기본적인 알고리즘은 다음과 같은 2개의 루프로 이루어진다.

첫번째 루프

1) $2 \le B(P_1) \le 6$

2) $A(P_1) = 1$

3) $P_2 = 0$ or $P_4 = 0$ or $P_6 = 0$

4) $P_4 = 0$ or $P_6 = 0$ or $P_8 = 0$

두번째 루프

1) $2 \le B(P_1) \le 6$

2) $A(P_1) = 1$

3) $P_2 = 0$ or $P_4 = 0$ or $P_8 = 0$

4) $P_2 = 0$ or $P_6 = 0$ or $P_8 = 0$

여기에서 $A(P_1)$는 P_1 주변의 픽셀이 그림2와 같이 순서가 정해져 있을 때, 픽셀 값이 0→1로 변화되는 패턴의 수를 나타낸다. 이는 '1'의 값을 가지는 하나의 픽셀 주위에 몇 개의 동일한 값을 가지는 픽셀군의 수(connectivity)를 나타낸다. 그림 3은 connectivity 값이 2인 픽셀의 예이다.

$B(P_1)$는 P_1의 이웃하는 0이 아닌 픽셀의 수이다.

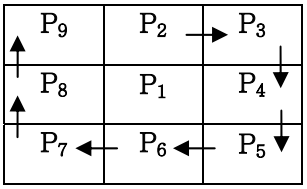


그림2. 주변 픽셀의 구조.

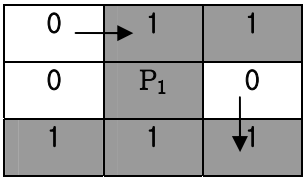


그림 3. Connectivity 2인 픽셀.

각 루프에서는 위의 조건을 모두 만족하는 픽셀을 제거하고, 더 이상 지울 픽셀이 남아 있지 않을 때까지 두 단계의 루프를 계속한다.

3. 한글 글립의 획 분할

한글 글립의 획 분할은 트루타입폰트와 한글 손글씨를 대상으로 이루어진다. 여기에서, 한글 손글씨는 정자체로 쓰여진 것을 주로 이용하며 지나치게 흘려 쓴 글씨나 일반적인 모양을 이루지 않는 글자는 사용하지 않는다.

본 논문의 획 분할 방법은 크게 획 분리, 획 병합, 그리고 획 볼륨 복원의 세 가지 과정으로 이루어진다.

3.1 획 분리(Stroke Splitting)

한글은 5개의 기본 획(basic stroke)들로 이루어져 있다. 다음의 그림은 한글의 5가지 기본 획들을 보여준다.

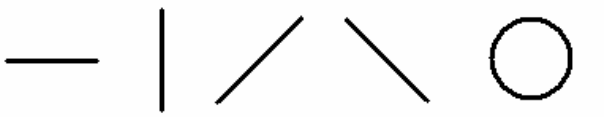
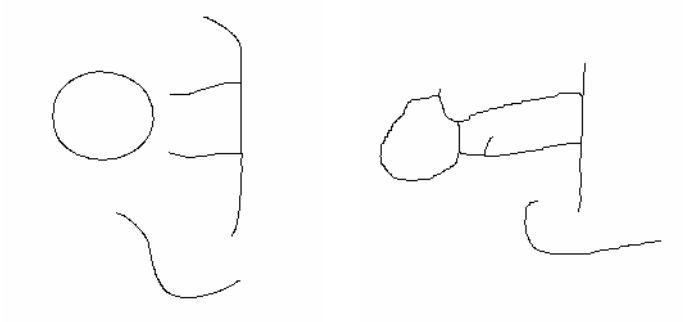


그림 4. 한글의 기본 획

이러한 기본 획을 한글 글립에 대한 외곽선 정보로부터 추출해 내기 위해, 먼저 2.2절에서 설명한 thinning을 래스터라이징된 트루타입폰트 이미지 또는 손글씨 이미지에 적용한다. 그림 5는 thinning 알고리즘을 트루타입폰트와 손글씨 이미지에 각각 적용한 결과를 나타낸다.



(a) (b)

그림 5. '연' 자의 thinning 결과 예: 폰트 (한양해서체) 이미지(a)와 손글씨 이미지(b)

다음은 thinning 후 얻어진 글립의 골격 이미지 에서 각 픽셀의 connectivity를 평가해, 아래와 같은 세 가지 종류의 특징점을 찾는다.

① *End point* : connectivity의 값이 1인 경우
② *Branch point* : connectivity의 값이 3인 경우
③ *Cross point* : connectivity의 값이 4인 경우

이렇게 특징점을 찾는 것이 획 분할에 있어 유용한 이유는 end point간의 연결로 이루어진 선을 한글에서의 기본 획으로 생각할 수 있기 때문이고, 이러한 기본 획들이 서로 닿아 있거나 교차하는 경우를 판단할 때, thinning된 이미지의 connectivity 계산으로 쉽게 알아낼 수 있기 때문이다.

이렇게 구해진 특징점에 두 개 이상의 획이 교차하거나 합쳐져 있는 경우를 생각해 볼 수 있으나, 한글 글립에서는 이런 경우는 생각하지 않는다.

위 세가지 특징점에 더하여, 'ㄱ', 'ㄴ' 의 굽어진 획에서 기본 획을 분리하거나 'ㅇ'을 구분하기 위해 *corner point*라는 특징점을 추가로 정의한다. Corner point는 임의의 픽셀에서의 기울기 변화가 심해, 미리 정해 놓은 임계값(threshold)을 넘는 점으로서, 이러한 점의 connectivity 는 2가 된다.

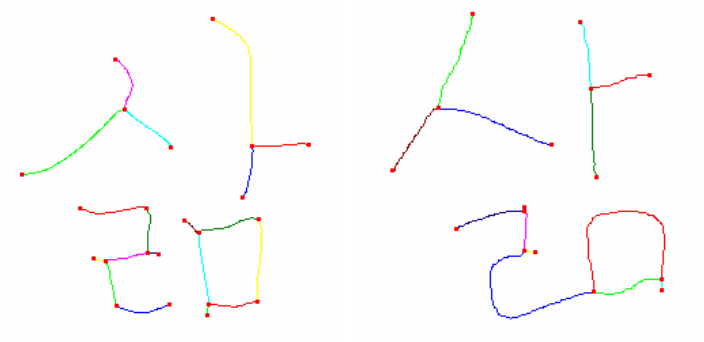
④ *Corner point*: 기울기 변화가 임계치 이상인 경우

다음은 골격 이미지를 특징점에서 잘라줌(splitting)으로써 부분 획(sub stroke)들을 추출한다. 특징점은 어떤 부분 획의 시작점 또는 끝점이 될 수 있다. 부분 획의 방향(direction), 즉 시작점과 끝점을 정하는 것은 표 1과 같이 한글 기본 획에 미리 부여된 획 방향을 따른다.

그림 6은 트루타입폰트와 손글씨에서 부분 획을 찾아낸 결과이다.

표 1. 한글 기본 획 및 방향 정보

Label	Basic Stroke	Start Point	End Point
Horizontal	ㅡ	Left	Right
Vertical	ㅣ	Top	Bottom
LeftFalling	/	Right top	Left bottom
RightFalling	\	Left top	Right bottom
ClosedCurve	ㅇ	Center top	Center top



(a) (b)

그림 6. 획 분리 결과

3.2 획 병합(Stroke Merge)

획 병합은 획 분리 과정을 거쳐 얻어진 부분 획을 획의 방향성과 획간의 연결성을 이용해 하나의 기본 획으로 합치는 과정이다.

획 병합을 위해서 연결되어 있는 특징점 간의 Adjacency matrix A를 다음과 같이 정의한다.

a_{ij} = 특징점 i과 특징점 j를 잇는 직선의 기울기

Adjacency matrix A를 이용한 획분할은 다음의 과정을 거쳐 이루어진다.

(1) End point에서 시작해 trace하면서 만나는 특징점 간의 기울기와 다음 특징점 까지의 기울기의 변화가 임계값 미만인 특징점을 골라 획을 합친다.
(2) 연결한 특징점의 기울기 값을 A에서 지운다.
(3) A에서 end point가 남아 있지 않을 때까지 (1)~(2)의 과정을 반복한다.
(4) 남아있는 부분획은 'ㅇ'로 간주한다.

다음의 그림 7과 표2는 'ㄷ'의 획을 분할하는 과정에서 생기는 특징점과 Adjacency matrix를 보여준다.

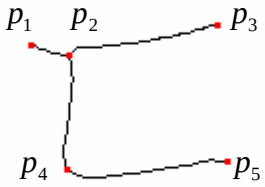


그림7. 획 분할의 예

표2. Adjacency matrix의 예(degree)

	p_1	p_2	p_3	p_4	p_5
p_1		15			
p_2	15		10	89	
p_3		10			
p_4		89			2
p_5				2	

3.3 획 볼륨 복원(Stroke Volume Recovery)

획 분리, 획 병합 과정을 거쳐 추출된 획 정보는 글립의 골격 상에서의 분할 정보이다. 획의 원래 모양을 얻기 위해서는 thinning 전 이미지에서 원래의 외곽선 정보를 얻어오는 볼륨 복원 과정이 필요하다.

볼륨 복원 과정에서는 thinning의 각 단계에서 지워지는 픽셀이 어느 픽셀로 수렴하는지, 즉, thinning의 결과로 나온 픽셀이 각 글립의 어느 픽셀로부터 나온 결과인지를 기록해 두었다가 이를 다시 복원하는 방법을 사용했다.

4. 실험 및 결과

구현한 알고리즘을 시험하기 위하여 총 50개의 글립에 대해 맑은 고딕체와 한양해서체의 트루타입폰트 이미지 및 10명의 대학(원)생으로부터 얻은 손글씨 이미지를 사용했다.

알고리즘을 평가하기 위한 정확도 P 를 다음과 같이 정의한다.

$$P = \frac{S_1 \cap S_2 \cap S_3}{N}$$

여기서 S_1 은 획이 분리된 글립집합, S_2 는 획이 병합된 글립집합, S_3 는 볼륨 복원된 글립집합, N 는 전체 실험 글립집합을 의미한다.

표 2는 트루타입 이미지 100개와 손글씨 이미지 500개를 가지고 알고리즘을 시험한 결과이다. 획 분할을 위한 각 처리단계별 및 전체 정확도를 보여준다.

표 2. 획 처리 단계별 및 전체 획 분할 정확도(%)

	Stroke Splitting	Stroke Merge	Volume Recovery	The whole Processing
TrueType Font	86	83	81	80
Handwriting	81	76	74	72

그림 8은 다양한 글자 모양에 대한 한글 트루타입폰트 및 손글씨의 획 분할 결과를 보여준다.

(a) 글립이미지 (b) 획 병합 결과 (c) 획 볼륨 복원 결과

그림 8. 획 분할 결과 예

다음의 경우에는 획 분할이 제대로 되지 않았다. 첫번째로 Corner point가 제대로 찾아지지 않은 경우가 있었다. 특히 받침 'ㄴ'의 경우 폰트와 손글씨 모두 느슨하게 구부러진 경향이 강해 제대로 찾지 못 했다.

두번째로 손글씨의 경우 'ㅍ'이나 'ㅎ' 같이 많은 획이 밀집하게 되는 경우 thinning 과정에서 outlier가 많이 발생해 제대로 된 결과를 얻을 수 없었다.

5. 결론

본 논문에서는 트루타입 폰트와 손글씨의 글립을 thinning을 이용한 형태학적 분석을 통해 자동으로 획을 분할하는 방법을 제안하였다. 제안한 방법은 글자의 모양과 획의 굵기 변화에 무관하게, 다양한 형태의 글자에 적용할 수 있는 장점을 가지고 있으면서도, 한글 획을 분할할 경우, 트루타입폰트에 대해서는 80%, 손글씨 글립에 대해서는 72%의 높은 정확도를 보였다.

하지만 여러 개의 획이 완만한 곡선을 이루거나 뭉쳐져 있는 경우, 그리고 글자를 심하게 흘려 쓴 경우는 획을 제대로 분할하지 못했다. 향후 연구에서 이를 개선하는 방법에 대해 연구하고자 한다.

우리는 궁극적으로 자동 한글 손글씨 생성을 위한 전 단계로서 자동 획 분할을 수행하였으므로, 획 분할된 결과를 이용하여 다양한 한글 손글씨를 생성하는 연구를 계속 진행할 예정이다.

Acknowledgement

"이 논문 또는 저서는 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임" (KRF-2007-521-D00425)

참고문헌 목록

[1] FontGad Corpoeration, http://www.fontgod.com

[2] 아이엠나@문자동맹, http://www.iamna.com/

[3] 직지소프트, http://www.smfont.com/

[4] Zhouchen Lin, Liang Wan, "Style-preserving English handwriting synthesis," IEEE Pattern Recognition, Vol.40, Issu.7, pp.2097-2109, July 2007.

[5] Thomas W. Sederberg and Rida T. Farouki, "Approximation by interval Bézier curves," IEEE Computer Graphics and Animation, Vol.12, Issu.5, Sept. 1992.

[6] Koichil Itoh and Yoshio Ohno, "A curve fitting algorithm for character fonts," Electronic Publishing, Vol.6(3), pp.195-205, Sept. 1993.

[7] Thierry Pudet, "Real Time Fitting of Hand-Sketched Pressure Brush strokes," Computer Graphics Forum, Vol.3, Issue3, pp.205-220, August 1994.

[8] Huang, Z. and Cohen, F.S., "Affine-invariant B-

spline moments for curve matching," IEEE Transactions on Image Processing, Vol.5, Issue.10, pp.1473-1480, Oct. 1996.
[9] Jaehwa Park and Young-Bin Kwon, "An Efficient Representation of Hand Sketch Graphic Messages Using Recursive Bézier Curve Approximation," LNCS, Vol.3211 pp.392-399, 2004.
[10] Douglas E. Zongker, Geraldine Wade and David H. Salesin, "Example-based hinting of TrueType fonts," Proc. of the 27th annual conference on Computer graphics and interactive techniques, pp. 411-416, 2000.
[11] Yoshiharu Kato and Makoto Yasuhara, "Recovery of Drawing Order from Single-Stroke Handwriting Images", IEEE PAMI, Vol. 22, pp. 938~949, 2000.
[12] 구상옥, 장현규, 정순기, "모바일 한자 학습 애니메이션 생성", 정보과학회, Vol. 33, No. 12, pp. 894-906, 2006.12.
[13] Sang Ok Koo, Hyun Gyu Jang and Soon Ki Jung, "Efficient Stroke Order Animation of the Chinese Characters", The First Korea-China Joint Conference on Geometric and Visual Computing, pp 8-15, 24~26 Aug 2005.
[14] 장현규, 구상옥, 정순기, "트루타입폰트 기반 한자 자동 획 분할 및 자동 획순 부여", 한국컴퓨터그래픽스학회 논문지(Journal of The Korea Computer Graphics Society), Vol. 11, No. 3, pp.10-18, SEPT 2005.
[15] Sang Ok Koo, Hyun Gyu Jang, Kwang Hee Won and Soon Ki Jung, "Automatic Stroke Extraction and Stroke Ordering Baseed on TrueType Font", Theory and Practice of Computer Graphics 2006, Eurographics UK Chapter Proceedings, Page 123~131, University of Teesside, Middlesbrough, UK, 20~22 June 2006.
[16] Zhang, Suen, "A Fast Parallel Algorithm For Thinning Digital Patterns", Communications of the ACM, 1984.