트루타입폰트 기반 한자 자동 획 분할 및 자동 획순 부여

장현규[O] 구상옥 정순기
경북대학교 컴퓨터 공학과
{seirion, sokoo}@vr.knu.ac.kr, skjung@knu.ac.kr

Automatic Stroke Extraction and Stroke Ordering of Chinese Characters Based on TrueTypeFont

Hyun Gyu Jang[0], Sang Ok Koo , Soon Ki Jung
Department of Computer Engineering, Kyungpook National University

요 약

이 논문에서는 트루타입 폰트(TrueType Font)의 글자 외곽선 데이터를 이용하여 자동으로 한자의 획을 분리하고 획 순서를 정하는 방법을 제안한다. 트루타입 폰트에는 글자의 외곽선 정보가 벡터 형식으로 저장되어 있으며, 이러한 벡터들은 일정한 규칙으로 배열되어 있다. 이와 같은 벡터들의 배치를 이용하여 한자의 획이 될 수 있는 벡터들의 집합을 조합하여 독립적인 획을 분리해 내고, 글자를 획 별로 분리하여 본래 트루타입 폰트의 저장 형식과 동일한 파일 형식으로 저장한다. 또한 분리된 모든 획에 대하여, 획 이름을 정의하고, 정의된 획들 간의 위치와 상관관계를 이용하여, 획 사이의 우선순위를 결정하여 획 순서를 부여한다. 이 작업들은 사람의 작업 없이 순수하게 자동으로 이루어지므로, 시간과 노력을 최소화 할 수 있다. 게다가, 획별로 분리되고 순서대로 정리된 한자들은 트루타입 폰트에 저장되어 있는 모양과 특성을 그대로 가지고 있으므로, 단순히 폰트 자체로써 사용할 수도 있을 뿐만 아니라, 한자 학습 컨텐츠로도 이용이 가능하며, 각종 애니메이션 효과 등 다양한 분야에서 융통성 있게 활용될 수 있다.

1. 서론

현대의 많은 정보들은 일반 PC나 이동 통신 기기 등을 통해서 쉽게 접근이 가능하다. 많은 사업자들은 여러 정보 통신 기기들의 이점을 이용할 수 있도록 하는 다양한 컨텐츠들을 만들고, 그것을 서비스 한다. 특히 이동 통신 기기의 제조 기술이 발달하여 휴대전화나 각종 휴대용 미디어 플레이어를 통하여 손쉽게 각종 컨텐츠들을 접할 수 있으며, 따라서 다양한 플랫폼에 적용할 수 있는 컨텐츠를 효율적으로 생산하는 것은 매우 중요하다.

최근 영어나 일본어뿐 아니라, 중국어 또는 한자를 배우려는 수요가 많아지고 있다. 때문에, 이와 관련한 학습물들이 증가하고 있으며, 웹이나 모바일 서비스 등을 통해서 다양한 컨텐츠가 소개되고 있다[1][2]. 이와 같은 컨텐츠 생산은 방대한 양에도 불구하고, 거의 수작업으로 이루어지고 있으며, 많은 시간과 노력을 투자하여 생산된 데이터들은 다른 기기들로 옮겨 다시 사용하는 것은 불가능하기 때문에 활용도가 매우 떨어진다. 특히, 글자의 모양과 크기가 고정되어 있으므로 특수한 효과나 변형을 주기 위해서는 모든 글자를 처음부터 다시 생성할 수밖에 없다. 따라서 계속해서 발달하고 있는 휴대용 미디어의 경우, 새로운 기기가 개발될 때마다 새로운 플랫폼에 적용될 수 있는 컨텐츠들의 생산은 매우 더디며, 기존의 데이터들은 쓸모없어 진다. 결국, 기기나 플랫폼에 독립적으로 사용될 수 있는 컨텐츠의 생산이 매우 필요하며, 여러 가지 해상도의 LCD에 적용될 수 있고, 다양한 소비자의 요구를 충족시킬 수 있는 여러 가지 효과들을 손쉽게 적용할 수 있는 데이터 형식이 필요하다.

이 논문에서는 플랫폼에 독립적으로 사용되는 트루타입 폰트의 데이터를 이용하여 한자의 획을 자동 분리하

고, 그 결과를 트루타입 폰트에 저장된 글자와 동일한 방식으로 저장하는 방법을 제안한다. 이는 글자 모양을 이미지로 저장하는 것이 아니라, 글자 외곽선에 대한 정보를 저장하는 것이므로, 그 데이터를 해석할 수 있는 어떠한 기기에서도 다시 사용될 수 있다. 또한 벡터 폰트의 방식을 사용하므로 글자의 크기 변형이 자유롭고, 다양한 변형이 가능하다. 또한 이 논문에서는 한자의 획 순서를 정해진 규칙에 따라 자동으로 결정하는 방법을 제안한다. 자동 획 분할과 획 순서 결정은 사람이 모든 글자에 대하여 일일이 편집해야하는 수고로움을 덜 수 있으므로 매우 효율적으로 컨텐츠를 생산해 낼 수 있다.

이 논문의 구성은 다음과 같다. 2절에서는 우리가 제안한 방법을 사용하기 위하여 기반이 되는 트루타입 폰트에 대해 설명하고 기존 한자 애니메이션을 위한 데이터 생성 방법에 대하여 서술한다. 3절과 4절에서는 자동으로 한자의 획을 분리하고, 획 순서를 결정하는 알고리즘에 대하여 다루며, 5절에서 제안한 방법을 이용한 실험 결과를 정리하고, 마지막으로 6절에서 결론 및 향후 연구 계획에 대하여 기술한다.

2. 배경

우리가 제안하는 한자 획 분할 방법은 트루타입폰트 파일 내의 한자 외곽선 정보를 기반으로 하여 이루어진다. 이는 기존에 한자 획 애니메이션 데이터를 생산하기 위해 이루어졌던 비트맵 이미지 편집 방식에 비해 매우 효과적이고 경제적인 방법이다. 이 절에서는 트루타입 폰트가 지니는 특성에 대하여 서술하고, 기존의 한자 획 분할 방식에 대하여 설명한다.

2.1. 트루타입 폰트와 베지어 곡선 (Bézier Curve)

이 논문에서 제안하는 획 분할 방법은 기본적으로 트루타입 폰트의 글자 정보를 이용한다. 트루타입 폰트는 현재 맥OS나 윈도우즈와 같은 운영체제에서 가장 많이 쓰이는 폰트 기술로써, 하드웨어에 독립적으로 사용되는 매우 효율적인 폰트 포맷이다 [3]. 트루타입 폰트 파일 내의 글자 정보는 외곽선을 베지어 곡선의 형태로 담고 있는 벡터 폰트로써 비트맵 폰트에 비해서 적은 공간에 많은 폰트 정보를 저장할 수 있다. 이러한 외곽선 기반 폰트는 별도의 폰트 래스터라이저(font rasterizer)를 사용하여 글자를 렌더링 해야 하는 단점이 있지만, 반면 글자의 변형과 응용이 매우 자유롭다는 점에서 굉장히 큰 장점을 지닌다.

그림 1은 굴림 폰트(gulim.ttc)에서 '家' 자를 출력한 결과이다. 파란색 점과 빨간색 점은 폰트에 저장되어 있는 베지어 곡선의 조절점을 그대로 출력한 것이며, 글자 외곽선은 이 조절점들을 이용하여 생성한 것이다.

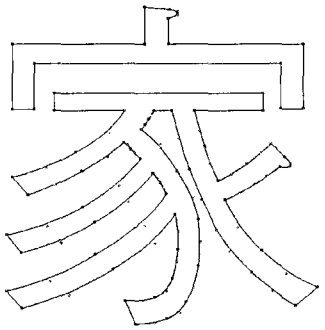
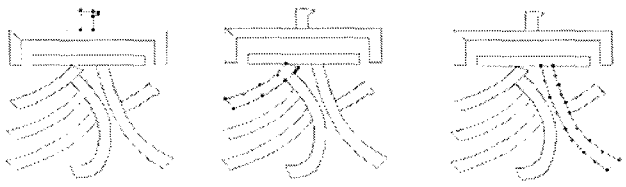


그림 1. 트루타입 폰트
외곽선 정보

우리가 제안한 획 분할 방식은 트루타입 폰트의 글자 외곽선을 이루는 조절점들을 그대로 활용한다. 그림 2는 '家' 자의 획을 분할한 결과의 일부이다. 그림 1에서 나타나 있는 조절점들을 그대로 활용한 것을 볼 수 있다. 따라서 각 획별로 분리된 결과들을 조합하여 처음 트루타입 폰트로부터 얻어왔던 글자를 다시 완성해 낼 수 있을 뿐만 아니라, 글자 손실 없이 확대나 회전 등과 같은 여러 가지 변형을 가할 수 있다.



(a) 1획 (b) 5획 (c) 10획

그림 2. '家'자를 획 분리한 결과

2.2. 한자 획 애니메이션

[2]는 웹상에서 한자 학습을 위한 획 정보를 gif 애니메이션으로 보여준다. 이 방법은 비트맵 글자를 일일이 편집하여 획의 순서대로 모든 이미지를 만든 다음, 이것을 gif 파일로 만든 것으로 가장 쉽고 흔히 쓰이는 방법이다. [1]과[4]는 비트맵 방식이 아닌 한자의 획이 부드럽게 애니메이션 되는 효과를 구현하였다. 이는 기존의 gif 애니메이션에 비해 보다 뛰어난 애니메이션 효과를 보이며, 비트맵 이미지 기반보다 데이터 크기도 매우 작다. 특히, [4]는 트루타입 폰트를 기반으로 하여 한자의 획을 분할하고, 그 결과를 획 순서대로 애니메이션 할 수 있도록 하는 저작 툴을 구현하였다. 이것은 사용자의 획 입력을 토대로 획 수와 획 순서를 결정하는 방식이다. 이 방식은 사용자의 입력이 글자마다 있어야 한다는 단점이 있다. 이 논문에서는 사용자의 입력 없이 자동으로 획을 구분하고 획 순서를 결정하는 방법을 제안한다. 이는 기존의 방법보다 훨씬 빠르며, 적은 노력으로 한자 획순 애니메이션 데이터를 생산한다.

3. 자동 획 분할 알고리즘

이 논문에서 제안한 획 분리 방법은 트루타입 폰트의 글자 외곽선 데이터로부터 획을 구성하는 벡터 집합을 찾는 것이다. 그림 1에서 보는 바와 같이 하나의 획은 여러 개의 벡터들로 구성된다. 그림 1에서 벡터 집합 $\{\vec{a}, \vec{b}, \vec{c}\}$와 벡터 집합 $\{\vec{i}, \vec{j}, \vec{k}\}$는 각각 획의 양 축을 구성하고 있다. 이 두 개의 벡터 집합들은 하나의 획을 구성하는 서로 대응되는 벡터 집합으로써, 서로에 대한 대응벡터 집합이라 일컫는다.

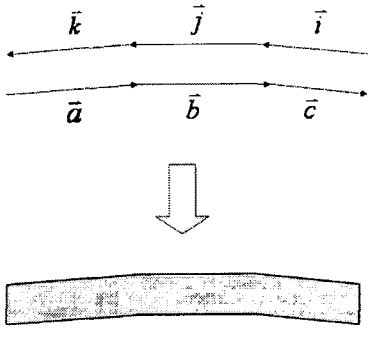


그림 3. 획은 벡터로 구성된다.

이 절에서는 모든 획에 대한 2개의 대응벡터 집합을 찾아 획을 분리하는 과정을 설명한다.

3.1. 컨투어 그룹 분리

획을 구성하는 벡터를 검색하기 위해 하나의 글자를 구성하는 모든 벡터들을 대상으로 삼는 것은 매우 비효율적이다. 두 개의 벡터가 만약 서로 다른 외곽선 상에 놓여 있다면 둘은 절대 하나의 획을 이루는 같은 벡터 집합에 포함되지 않는다. 따라서 서로 연관 있는 벡터만을 모아 두고 그들 내에서 대응 벡터를 찾는 것이 좋다.

컨투어(contour)란 글자 외곽선을 이루는 단위로써, 하나의 독립적인 폐곡선이다. 트루타입 폰트로부터 얻어온 글자 데이터는 하나 또는 여러 개의 컨투어들로 이루어져 있고, 그룹은 컨투어 단위로 분리된다. 예를 들어, '韓' 자는 그림 2와 같이 모두 8개의 컨투어로 구성되어 있으며, 크게 3개의 그룹으로 분리될 수 있다. 일반적으로 하나의 컨투어는 하나의 그룹을 이루게 되며, 만약 한 컨투어의 내부에 다른 컨투어가 존재하는 경우, 그것은 외부 컨투어와 같은 그룹으로 취급된다. 컨투어를 구성하는 벡터들은 항상 반시계방향으로 배치되어있다.

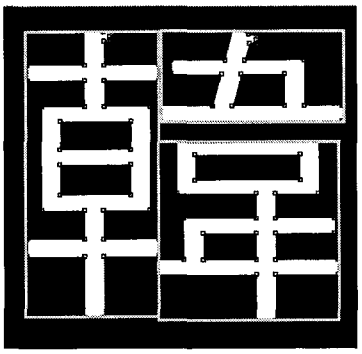


그림 4. '韓' 자의 그룹 분리

그림 5와 같이, 하나의 컨투어 내부에 다른 컨투어가 하나 이상 존재하는 경우, 내부 컨투어 상의 벡터들은 외부 컨투어와 반대로, 즉 시계방향으로 배치된다. 한 그룹에 속한 컨투어들이 서로 다른 벡터 방향을 가진다는 사실은 획 분리에 매우 중요한 단서가 된다. 이후의 모든 작업들은 각각의 컨투어 그룹에 대해서 독립적으로 수행된다.

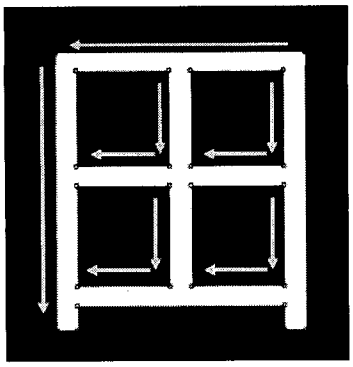


그림 5. 컨투어 내부에 다른 컨투어가 있는 경우

3.2 벡터 추출

트루타입 폰트로부터 얻은 최초의 데이터는 베지어 곡선의 조절점들의 연속으로 이루어져 있다. 하지만 획을 추출하는 과정에서 베지어 곡선을 그대로 사용하여 계산하는 것은 매우 어려운 일이다. 따라서 입력으로 받은 베지어 곡선에서 곡선의 성분을 제거하고 간단하게 직선으로 취급하여 계산하면 쉽고 간단하다. 베지어 커브가 2차 이상의 곡선인 경우, 곡선의 차수를 높이는 조절점을 제외하여 모든 곡선을 직선으로 변환한다. 이 작업을 실행한 결과 예는 그림 6과 같다. 글자 모양을 구성하는 일부 조절점이 빠졌지만 글자의 전체 모양은 거의 변함이 없으며, 모든 벡터들이 남아 있는 조절점을 지나가므로, 벡터 사이의 거리나 벡터 사이의 각도를 구하는 과정이 간단해 진다.

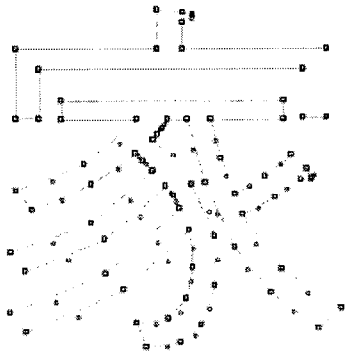


그림 6. 베지어 곡선을 단순 벡터로 변환

3.3 획을 구성하는 벡터 집합 찾기

획을 구성하기 위해서는 일정 간격을 두고 평행하게 놓여 있는 2개의 대응 벡터 집합이 필요하다. 이러한 대응 벡터 집합을 찾아내기 위해서 각각의 벡터들의 위치와 방향 정보를 이용한다.

3.3.1 두 벡터 사이의 거리 계산

폰트 내에 있는 글자의 획은 일정한 두께를 가지고 있다. 그것은 곧, 획을 구성하는 대응 벡터들이 서로 일정한 간격을 두고 평행하게 놓여 있다는 것을 의미한다. 일정한 거리 내에 있는 벡터들을 찾기 위해서 우리는 그림 7과 같이 2차원 좌표 상에서 점과 직선 사이의 거리 계산법을 이용한다. 이 경우 한 점과 직선의 최단거리를 계산하게 되므로, 점을 직선에 수선으로 내린 경우 거리 계산이 성립된다. 즉, 그림 8의 (a)와 (b)의 경우 한 벡터의 양 끝 점을 다른 벡터에 수직으로 내린 경우, 벡터와 만나는 지점이 반드시 존재해야 하며, (c)와 같이 양 벡터가 서로 어긋나 있는 경우는 고려 대상에서 제외된다.

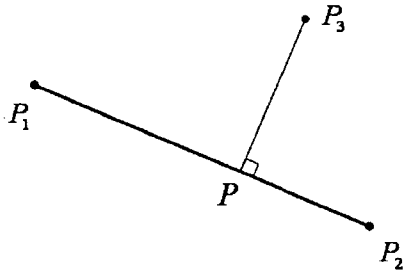


그림 7. 점과 직선 사이의 거리

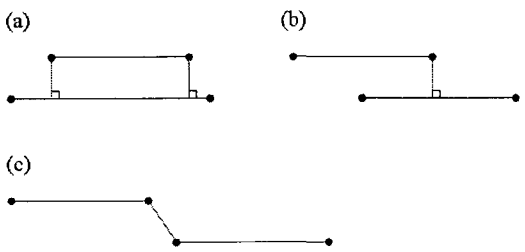


그림 8. 두 벡터간의 위치 관계

그림 5에서 P_1과 P_2를 지나는 벡터를 선분 $\overline{P_1P_2}$로 나타내면, 선분 $\overline{P_1P_2}$와 점 P_3사이의 최단 거리는 선분과 점이 수직으로 만나는 지점인 P에서 P_3사이의 거리, 즉 선분 $\overline{PP_3}$의 길이와 같다. P는 P_1과 P_2를 지나는 직선 위에 존재하므로 P는 다음과 같이 P_1과 P_2에 대한 식으로 바꾸어 쓸 수 있다.

$$P = P_1 + u(P_2 - P_1). \tag{1}$$

여기서 u는 임의의 상수로써 0과 1사이의 값을 가지게 되며, 만약 그림 8의 (c)와 같이 벡터가 놓여있는 경우에는 u가 그 외의 값을 가지게 된다.

선분 $\overline{P_1P_2}$과 선분 $\overline{P_1P_2}$는 서로 수직으로 만나므로,

$$(P_3 - P)\cdot(P_2 - P_1) = 0. \tag{2}$$

식(2)의 P를 식(1)로 치환하면,

$$[P_3 - P_1 - u(P_2 - P_1)]\cdot(P_2 - P_1) = 0. \tag{3}$$

식(3)을 u에 대한 식으로 바꾸면,

$$u = \frac{(x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1)}{\| P_2 - P_1 \|^2}. \tag{4}$$

식(1)에 식(4)를 치환하면 $P(x,y)$를 구할 수 있으며, 선분 $\overline{PP_3}$의 길이는 식(6)을 통해 구할 수 있다.

$$\begin{aligned} x &= x_1 + u(x_2 - x_1), \\ y &= y_1 + u(y_2 - y_1). \end{aligned} \tag{5}$$

$$d = \sqrt{(x - x_3)^2 + (y - y_3)^2}. \tag{6}$$

3.3.2 두 벡터 사이의 각도 계산

획의 양 축을 이루는 벡터들은 서로 거의 평행하다. 따라서 두 벡터 간의 각도가 180도에 가깝다면, 두 벡터들은 서로 대응 벡터가 될 수 있다. 그림 9와 같이, 두 벡터 사이의 각 Θ는 벡터 $\vec{u}$와 벡터 $\vec{v}$ 사이 각으로 정의한다.

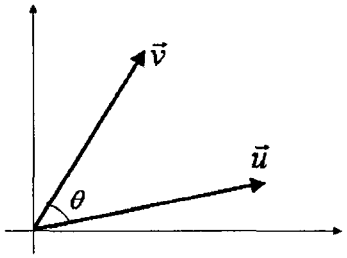


그림 9. 두 벡터 사이의 각도

두 벡터가 이루는 각은 식(7)과 같이 벡터의 내적의 arc cosine 함수를 통해 구할 수 있다.

$$\Theta = \cos^{-1}\left(\frac{\vec{u}\cdot\vec{v}}{\| \vec{u} \| \| \vec{v} \|}\right). \tag{7}$$

3.3.3 오른손 법칙에 따른 벡터 방향

그림 10에서 보는 바와 같이, 글자의 이루는 벡터들은 반시계방향으로 이동하며 외곽선을 구성하게 된다. 하나의 컨투어 내에 다른 컨투어가 포함되어 있는 경우는 반대 방향으로 벡터가 구성되어 있다. 이 점은 그림 8의 (2)와 같이 획이 구성된 경우에 획 각각에 대한 벡터들도 반시계방향으로 벡터가 구성되게 된다. 그림 8의 (1)과 같이 컨투어가 구성된 경우, 그림 8의 (2)와 같이 벡터가 반 시계방향으로 놓여있는 경우는 정확하게 획을 만들 수 있으며, 반면 벡터가 시계방향으로 놓여 있는 경우는 획을 구성하지 못 한다(그림 8의 (3)).

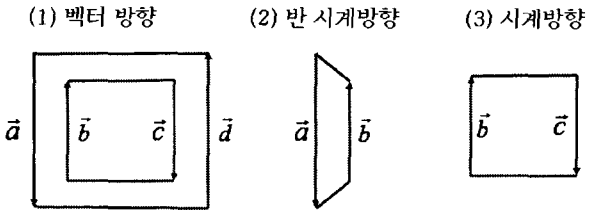


그림 10. 획을 구성하는 벡터의 방향

벡터들이 반시계방향으로 놓여 있다는 사실은 그림 11과 같이 두 벡터의 양 끝점을 이은 새로운 벡터를 이용하여 알 수 있다. 벡터 $\vec{u}$와 $\vec{v}$가 있다면, 각 벡터의 시작점과 끝점을 이은 새로운 백터 $\vec{w}$를 만들어 낼 수 있다.

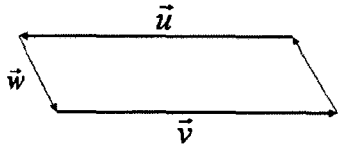


그림 11. 반 시계 방향으로
놓여있는 두 벡터

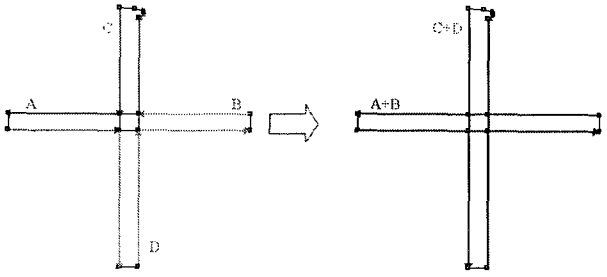
두 벡터 $\vec{u}$와 $\vec{v}$가 놓여 있는 방향은 벡터 $\vec{u}$와 $\vec{w}$의 외적을 통하여 구해 낼 수 있다. 두 벡터는 2차원 벡터이지만, 각 벡터의 세 번째 성분에 0을 추가하여 3차원 벡터로 만든다. 두 벡터를 식(8)과 같이 외적해서 구한 값은 항상 첫 번째와 두 번째 성분 값이 0이 되며, 마지막 성분의 값은 0이 아닌 값을 가지게 된다. 이 마지막 성분의 값이 0보다 큰 경우, 오른손 법칙에 의해서 두 벡터 $\vec{u}$와 $\vec{v}$가 반시계방향으로 되어있다는 것을 의미한다.

$$u \times w = (x_u, y_u, 0) \times (x_w, y_w, 0) = (0, 0, x_u y_w - y_u x_w). \quad (8)$$

3.3.4 획 조각 모음

획을 생성하기 위한 마지막 단계는 작은 획 조각들을 모아 하나의 완성된 획으로 만드는 것이다. 그림 12와 같이 '+'자 모양의 획이 있는 경우 A, B, C, D 등 모두 4개의 작은 획 조각들이 만들어졌다. 그러나 사실은 이 획들은 그림 10의 (2)와 같이 2개의 획이 되어야 한다.

이러한 경우는 정확하게 두 개의 획이 교차하는 경우에 발생하며, 획의 끝 부분의 두 벡터 축 모두가 그림 13과 같이 반시계방향으로 진행된다. 즉, 벡터 $\overrightarrow{v_{i-1}}$과 $\overrightarrow{v_i}$의 관계, 그리고 벡터 $\overrightarrow{v_j}$와 $\overrightarrow{v_{j+1}}$과의 관계가 모두 벡터가 시계방향으로 진행되는 관계이다. 이점은 그림 12의 (a)에서 B, C, D 획에 대해서도 마찬가지이다. 두 벡터가 시계방향으로 배치 된 사실은 식 (8)에서 두 벡터의 외적을 통하여 알 수 있다. 그림 12의 A획과 B획이 서로 연속적으로 있으며, 양 끝에서 벡터의 방향이 시계방향이므로, 두 획이 '+'자 모양의 획 관계에 놓여 있다는 사실을 쉽게 알 수 있다.



(a) 획 조각 모음 이전 (b)획 조각 모음 이후

그림 12. 분리 된 획 조각 모음

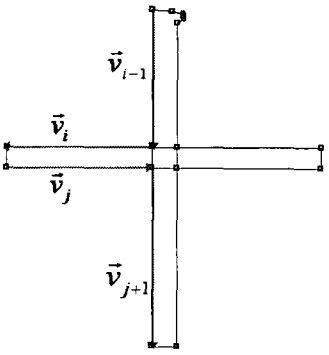


그림 13. '+'자 모양 획에서
벡터의 진행 방향

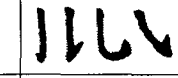
4. 자동 획 순서 결정 알고리즘

이 절에서는 앞 절에서 분리한 각 획에 대하여 순서를 부여하는 방법에 관하여 서술한다. 분리된 모든 획은 각각의 획 종류에 따라 획 이름을 부여하고, 정해진 획 이름과 획의 위치 그리고 획과 획 사이의 상관관계를 이용하여 획의 순서를 비교하게 된다.

4.1 획 정의

획은 표 1에서 정의된 바와 같이 모두 8가지 획의 형태로 나눈다[5]. 획은 그림 13과 같이, 획을 구성하고 있는 벡터의 방향, 벡터들 간의 관계 등에 의해서 결정된다. 여기서 결정된 획 이름들은 다른 획과의 우선순위를 비교하기 위하여 사용된다.

표 1. 획 모양에 따른 분류

획 이름	획 모양
DOT	
HORIZONTAL	
VERTICAL	
HOOK	
LEFTFALLING	
RIGHTFALLING	
TURNING	
RISING	

s_i 의 길이 : 벡터들의 길이 합

T_L : 벡터 길이의 임계값, 100 pixel

T_{var} : 획을 구성하는 벡터들이 이루는 최대 각, 75

T_{max} : 획의 기울기 최대 임계값, 75

T_{min} : 획의 기울기 최소 임계값, 15

모든 획의 집합 $Q = \{s_1, s_2, \ldots, s_n\}$ 에 대하여,

1. if(s_i ::길이 < T_L) s_i = DOT
2. if(s_i::벡터각 > T_{var}) s_i = HOOK
3. if(s_i::획기울기 < T_{min}) s_i = VERTICAL
4. if(s_i::획기울기 > T_{max}) s_i = HORIZONTAL
6. if(s_i::x > 0) s_i = RIGHTFALLING
7. else s_i = LEFTFALLING

모든 획의 집합 $Q = \{s_1, s_2, \ldots, s_n\}$ 에 대하여,

8. if(s_i == VERTICAL && s_j == HORIZONTAL && s_i와 s_j 공유 포인트 == 2) s_i + s_j = TURNING

그림 13. 획 종류 결정 알고리즘

4.2 획 우선순위 결정

모든 획 이름이 결정되고 나면, 획의 이름과 획의 위치 그리고 다른 획들 사이의 관계에 의하여 획들 사이의 우선순위를 결정하게 된다. 획의 우선순위를 결정하는 기본적인 규칙은 표 2와 같다 [5]. 이 규칙을 기반으로 하여 모든 획들 간의 우선순위를 가리기 위해 획 사이의 연산자 '≤'를 정의한다. 이는 모든 획들의 집합 Q의 임의의 두 원소 s_i와 s_j 사이의 우선순위를 정하기 위한 이항 연산자로서 반사성(reflexive), 비대칭성(antisymmetric), 추이성(transitive)을 모두 만족한다. 따라서 연산자 '≤'는 집합 Q 상의 완전 순서 관계(totally ordered relation)를 만족한다[6].

표 2. 획 순서 결정 규칙

규 칙	예	획 순
가로획 -> 세로획	十	一 十
왼쪽 삐침 -> 오른쪽 삐침	人	丿 人
위->아래	三	一 二 三
왼쪽->오른쪽	州	丶 丿 丬 州 州 州
바깥획->안쪽획	月	丿 冂 月 月
안쪽획->닫는획	四	丨 冂 冂 四 四
중간획->양쪽획	小	亅 小 小

5. 실험 결과 및 분석

이 절에서는 제안한 한자 획 분리 방법과 획 순서 결정 방식이 얼마나 정확한 성능을 보이는 지를 보이기 위해 한자 능력 검증시험 8급 50자에 대한 테스트를 수행하였다. 트루타입폰트는 굴림, 폰트 사이즈는 500X500 픽셀이며, 획의 굵기를 35~40픽셀 사이로 정의하였다. 표 3은 테스트 대상 글자에 대한 획 분할 성공 글자와 실패 글자를 나열한 것이다.

표 3. 획 분할 성공 글자와 실패 글자

획 분할 성공 (45자)	획 분할 실패 (5자)
校 九 國 軍 金 南 女 年	敎 父 水 長 弟
大 東 六 母 萬 木 門 民	
白 北 四 山 三 生 西 先	
小 室 十 五 王 外 月 二	
人 一 日 中 靑 寸 七 土	
八 學 韓 兄 火	

획 분할에 실패한 글자의 경우, 하나의 획을 2개의 획으로 나누어지는 것(父)과, 2 개의 획이 되어야 하지만 하나의 획으로 나누어지는 것(水, 長), 획을 구성하는 벡터의 위치가 모호하여 획 분할에 실패하는 것(弟) 등이 있었다.

표 4는 표 3의 획 분할 성공 글자 45자에 대하여, 획의 순서가 바르게 된 것과 그렇지 못한 글자들에 대한 결과이다.

표 4. 획 순서가 바로 된 글자와 잘 못 된 글자

획 순서가 바른 자(39자)	획 순서가 틀린 자(6자)
校 九 國 軍 南 年 大 東	金 女 母 民 山 學
六 萬 木 門 白 北 四 三	
生 西 先 小 室 十 五 王	
外 月 二 人 一 日 中 靑	
寸 七 土 八 韓 兄 火	

획 순서가 잘 못 된 글자는 대부분 획의 순서가 표 2의 규칙에 벗어난 글자(母, 山)와 '+'자 획의 부분에서 세로획이 가로획 보다 우선되는 부분이 있는 글자(女, 學)이다.

6. 결론 및 향후 과제

이 논문에서는 트루타입 폰트의 데이터를 기반으로 하여 한자의 획을 자동 분리하고, 획의 순서를 규칙에 따라 결정하는 방법을 제안하였다. 제안된 방법을 이용하여 획 분할의 경우 90% 글자에 대하여 성공적인 결과를 보였다. 자동 획 순서 결정 알고리즘은 획 순서 규칙을 잘 따르는 글자에 대하여 잘 적용됨을 보였다. 제안한 방법을 이용하면 한자 획 애니메이션 데이터를 생산하기 위한 상당부분의 노력을 줄일 수 있을 것으로 생각된다.

이 논문에서 제안한 방법은 획의 굵기 변화가 심한 다른 폰트에 대해서는 잘 적용되지 않는 단점이 있다. 향후 연구로써, 제안한 방법으로 분리한 획의 정보를 다른 폰트에 적용시켜 다양한 폰트 모양의 결과를 얻는 방법에 대하여 연구하고자 한다.

감사의 글

본 연구는 경북대학교 BK21 유비쿼터스 컴퓨팅 과제 지원으로 수행되었음.

참고문헌

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

[2] 아이한자 http://www.ihanja.com/

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

[4] Sang Ok Koo et al., Efficient Stroke Order Animation of the Chinese Character, KCJC, 2005

[5] How To Write Chinese Characters, http://www1.esc.edu/personalstu/jli/ How%20to%20Write%20Chinese%20Characters.pdf

[6] Susanna S. Epp, Partial Order Relations, *Discrete Mathematics with Applications*, pp.632-640, Thomson Learning, Inc.