트루타입 폰트 기반 한자 자동 획 분할 및 획 순서 부여

장현규[1],구상옥[2],정순기[3]
경북대학교 컴퓨터 공학과[1 2 3]
{seirion[1],sokoo[2]}@vr.knu.ac.kr , skjung@knu.ac.kr[3]

Automatic Stroke Extraction and Stroke Ordering Based on TrueTypeFont

Hyun Gyu Jang[1], Sang Ok Koo[2] , Soon Ki Jung [3]
Department of Computer Engineering, Kyungpook National University [1 2 3]

요 약

이 논문에서는 트루타입 폰트의 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, 획 별로 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.

Keyword : Digital Contents, TrueTypeFont, Chinese Characters, Relaxation Labeling

1. 서론

현대의 많은 정보들은 PC 나 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. 최근 중국어 또는 한자를 배우려는 수요가 많아지고 있다. 때문에, 이와 관련한 학습 물들이 증가하고 있으며, 웹이나 모바일 서비스 등을 통해서 다양한 컨텐츠가 소개되고 있다 [1][2][3][4][5]. 이와 같은 컨텐츠 생산은 방대한 양에도 불구하고, 거의 수작업으로 이루어지고 있으며, 많은 시간과 노력을 투자하여 생산된 데이터들은 다른 플랫폼에서 사용이 불가능하기 때문에 활용도가 매우 떨어진다. 특히, 글자의 모양과 크기가 고정되어 있으므로 특수한 효과나 변형을 주기 위해서는 모든 글자를 처음부터 다시 생성할 수밖에 없다.

이 논문에서는 플랫폼에 독립적으로 사용되는 트루타입 폰트의 데이터를 이용하여 한자의 획을 자동 분리하고, 그 결과를 트루타입 폰트에 저장된 글자와 동일한 방식으로 저장하는 방법을 제안한다. 즉, 컨텐츠의 데이터를 벡터 폰트 방식으로 사용하여 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 편집해야 하는 수고로움을 덜 수 있으므로 매우 효율적으로 컨텐츠를 생산해 낼 수 있다.

이 논문의 구성은 다음과 같다. 2 장에서는 우

리가 제안한 방법을 사용하기 위하여 기반이 되는 트루타입 폰트에 대해 설명하고 기존 한자 애니메이션을 위한 데이터 생성 방법에 대하여 서술한다. 3 장과 4 장에서는 자동으로 한자의 획을 분리하고, 획 순서를 결정하는 알고리즘에 대하여 다루며, 5 장에서 제안한 방법을 이용한 실험 결과를 정리하고, 마지막으로 6 장에서 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

트루타입 폰트는 현재 맥 OS 나 윈도우즈와 같은 운영체제에서 가장 많이 쓰이는 폰트 기술로써, 하드웨어에 독립적으로 사용되는 매우 효율적인 폰트 포맷이다 [6]. 폰트 파일 내의 글자 정보는 외곽선을 베지어 곡선(Bézier Curve)[7]의 형태로 담고 있는 벡터 폰트로써 비트맵 폰트에 비해서 적은 공간에 많은 정보를 저장할 수 있고, 글자의 변형과 응용이 매우 자유롭다는 큰 장점을 지닌다.

그림 1 은 굴림 폰트에서 '家' 자를 출력한 결과이며, 그림 2 는 '家' 자의 획을 분할한 결과의 일부이다. 그림 1 에서 나타나 있는 조절점들을 그대로 활용한 것을 볼 수 있다. 따라서 각 획 별로 분리된 결과들을 조합하여 처음 폰트로부터 얻어왔던 글자를 다시 완성해 낼 수 있을 뿐만 아니라, 글자 손실 없이 확대나 회전 등과 같은 여러 가지 변형을 가할 수 있다.

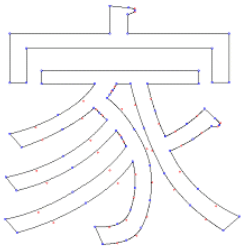


그림 1. 트루타입 폰트 외곽선 정보

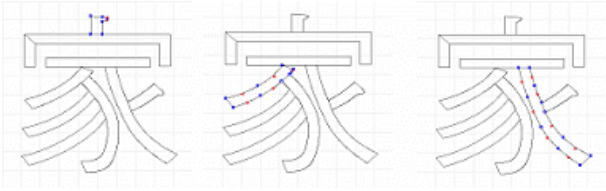


그림 2. '家' 자를 획 분리한 결과

2-2 한자 획 애니메이션

[4]는 웹 상에서 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][8][9]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, [8][9]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.

3. 자동 획 분할 알고리즘

이 논문에서 제안한 획 분리 방법은 트루타입 폰트의 글자 외곽선 데이터로부터 획을 구성하는 벡터 집합을 찾는 것이다. 그림 3 에서 보는 바와 같이 하나의 획은 여러 개의 벡터들로 구성된다. 그림 3 에서 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 대응 벡터 집합이라 일컫는다.

이 장에서는 모든 획에 대한 2 개의 대응벡터 집합을 찾아 획을 분리하는 과정을 네 단계를 통하여 설명한다.

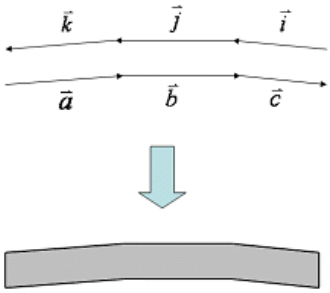


그림 3. 획은 벡터로 구성된다.

3-1 컨투어 그룹 분리

획을 구성하는 벡터를 검색하기 위해 하나의 글자를 구성하는 모든 벡터들을 대상으로 삼는 것은 매우 비효율적이다. 예를 들어, '韓' 자는 그림 4 와 같이 3 개의 작은 단위의 글자로 이루어져 있고, 그것들을 각각 독립적으로 획을 분할 하는 작업을 할 수 있다.

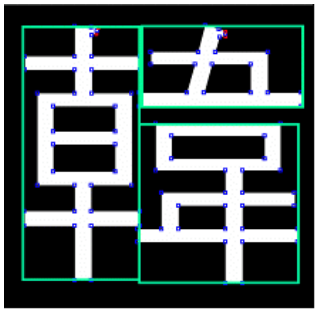


그림 4. 컨투어 그룹 분리

3-2 벡터 추출

폰트로부터 얻은 최초의 데이터는 베지어 곡선의 조절점들의 연속으로 이루어져 있다. 하지만 획을 추출하는 과정에서 베지어 곡선을 그대로 사용하여 계산하는 것은 매우 어렵다. 따라서 입력으로 받은 베지어 곡선에서 곡선의 성분을 제거하고 간단하게 직선으로 취급하여 계산하면 쉽고 간단하다. 베지어 커브가 2 차 이상의 곡선인 경우, 곡선의 차수를 높이는 조절점을 제외하여 모든 곡선을 직선으로 변환한다. 이 작업을 실행한 결과 예는 그림 5 와 같다.

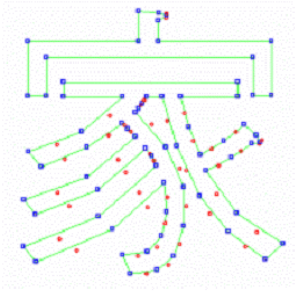


그림 5. 베지어 커브를 벡터로 변환

3-3 획을 구성하는 벡터 집합 찾기

획을 구성하기 위해서는 일정 간격을 두고 평행하게 놓여 있는 2 개의 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. 즉, 획을 구성하는 대응 벡터들이 서로 일정한 간격을 두고 평행하게 놓여 있다는 것을 의미한다. 일정한 거리 내에 있는 벡터들을 찾기 위해서 점과 직선 사이의 거리 계산법을 이용한다. 그림 6 의 (a)와 (b)의 경우 한 벡터의 양 끝점을 다른 벡터에 수직으로 내린 경우, 벡터와 만나는 지점이 반드시 존재해야 하며, 6 의 (c)와 같이 양 벡터가 서로 어긋나 있는 경우는 고려 대상에서 제외된다.

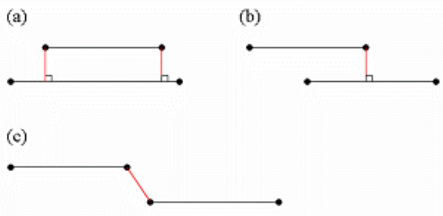


그림 6. 두 벡터 사이의 배치 관계

3-3-2 두 벡터 사이의 각도 계산

획의 양 축을 이루는 벡터들은 서로 거의 평행하다. 따라서 두 벡터 간의 각도가 180 도에 가깝다면, 두 벡터들은 서로 대응 벡터가 될 수 있다. 그림 7 과 같이, 두 벡터 사이의 각 θ 는 벡터

$\vec{u}$와 벡터 $\vec{v}$사이 각으로 정의한다.

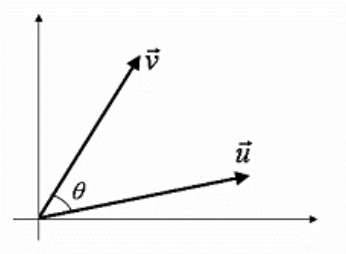


그림 7. 두 벡터 사이의 각도

3-3-3 오른손 법칙에 따른 벡터 방향

그림 8 와 같이, 글자의 이루는 벡터들은 반 시계방향으로 배치 된다. 하나의 컨투어 내에 다른 컨투어가 포함되어 있는 경우는 반대 방향으로 벡터가 구성되어 있다. 그림 8 의 (1)과 같이 컨투어가 구성된 경우, 그림 8 의 (2)와 같이 벡터가 반 시계방향으로 놓여있는 경우는 정확하게 획을 만들 수 있으며, 반면 벡터가 시계방향으로 놓여 있는 경우는 획을 구성하지 못 한다(그림 9 의 (3)).

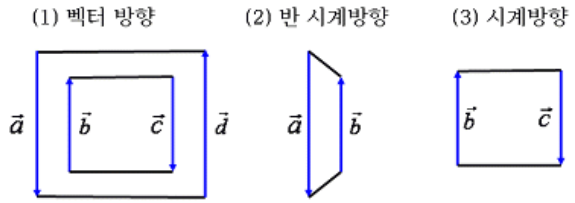


그림 8. 획을 구성하는 벡터는
반 시계방향으로 구성된다.

벡터들이 반 시계방향으로 놓여 있다는 사실은 그림 9 와 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 벡터 $\vec{w}$를 만들어 낼 수 있다.

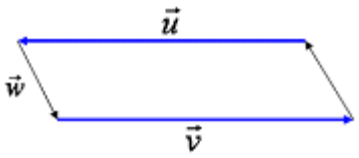


그림 9. 반 시계방향으로 구성된 벡터 관계

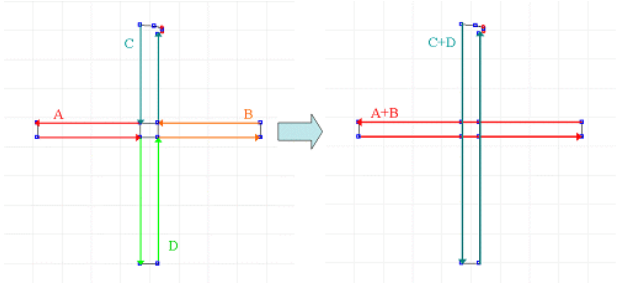
두 벡터 $\vec{u}$와 $\vec{v}$가 놓여 있는 방향은 벡터 $\vec{u}$와 $\vec{v}$의 외적을 통하여 구해 낼 수 있다. 두 벡터는 2 차원 벡터이지만, 각 벡터의 세 번째 성분에 0 을 추가하여 3 차원 벡터로 만든다. 두 벡터를 식(1)과 같이 외적 해서 구한 마지막 성분의 값이 0 보다 큰 경우, 오른손 법칙에 의해서 두 벡터 $\vec{u}$와 $\vec{v}$가 반 시계방향으로 되어있다는 것을 의미한다.

$$u \times w = (x_u, y_u, 0) \times (x_w, y_w, 0) = (0,0, x_u y_w - y_u x_w) \quad (1)$$

3-3-4 획 조각 모음

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

이러한 경우는 정확하게 두 개의 획이 교차하는 부분에서 발생하며, 획의 끝 부분의 두 벡터 축 모두가 그림 11 과 같이 반 시계방향으로 진행된다. 즉, 벡터 $\vec{v}_{i-1}$과 $\vec{v}_i$의 관계, 그리고 $\vec{v}_j$ 벡터와 $\vec{v}_{j+1}$과의 관계가 모두 벡터가 시계방향으로 진행되는 관계이다. 두 벡터가 시계방향으로 배치된 것은 두 벡터의 외적을 통하여 알 수 있다. 그림 10 의 (a)에서 A 획과 B 획이 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 10. 분리 된 획 조각 모음

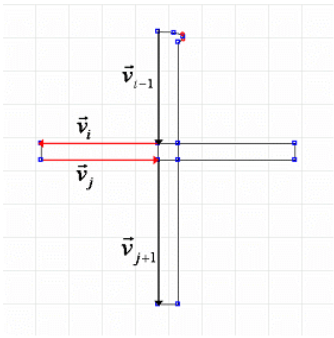


그림 11. '+' 자 모양 획에서 벡터의 배치

3.4 조절점 회복

획을 구성하는 벡터들의 집합이 최종적으로 결정 되면, 그것들을 이용하여 다시 원래의 획 모양을 복원하기 위하여 획을 베지어 곡선의 조절점들로 구성되도록 해야 한다. 그림 12 와 같이 벡터로만 구성되어 있는 획 사이에 적절히 곡선을 그릴 수 있는 조절점들을 삽입한다.

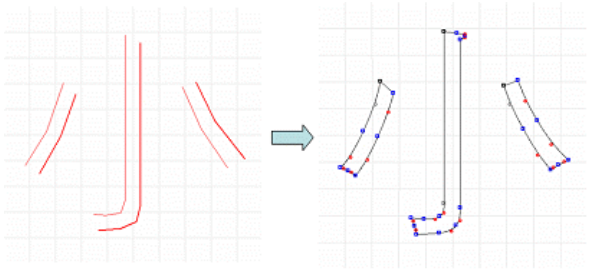


그림 12. 조절점 회복 결과

그림 13 의 경우와 같이 획을 이루는 벡터가 서로 연결되어 있지 않는 때에는, 비어있는 곳을 직선으로 채워 넣게 된다. 또한 그림 14 과 같이 획의 끝 부분에서는 벡터를 이루는 벡터의 양 끝 부분에 채워 넣지 않은 모든 조절점들을 채움으로써 획의 끝 부분을 완성한다. 획의 교차 지점에 있는 조절점은 두 개 이상의 획에 포함되게 되며, 이런 경우는 단순히 조절점 사이를 직선으로 연결할 수 있도록 고정점을 연속해서 배치하면 된다. 그러나 그림 14 와 같이 획을 이루는 벡터들의 끝 부분이 다른 획과 조절점을 공유하지 않는 경우는, 본래의 글자에서 배치되어 있는 조절점 순서대로 시작점과 끝점 사이에 있는 모든 조절점들을 추가해야만 본래 글자 모양 그대로를 회복할 수 있다.

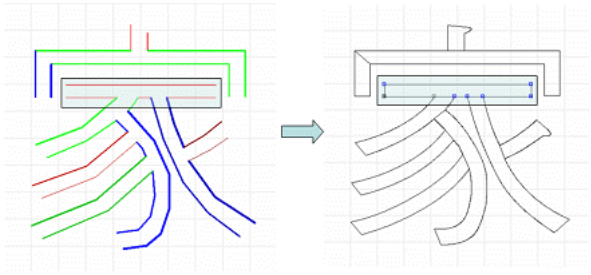


그림 13. 벡터가 연결 되지 않은 경우

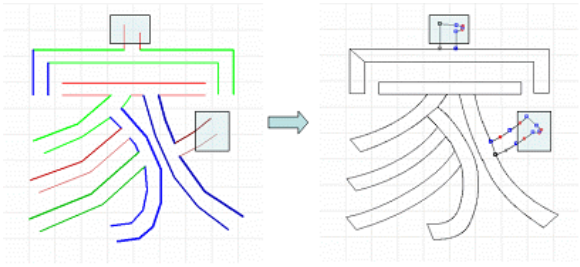


그림 14. 획의 끝부분에서 조절점 회복

4. 자동 획 순서 결정 알고리즘

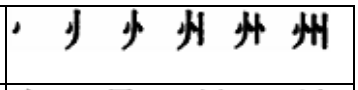
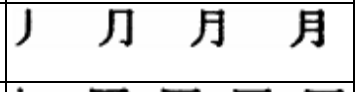
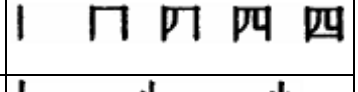
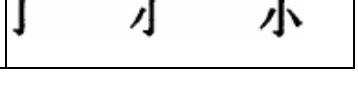
이 장에서는 앞 장에서 분리한 각 획에 대하여 순서를 결정하는 방법에 관하여 서술한다. 분리된 모든 획은 각각의 획 종류에 따라 획 이름을 부여하고, 정해진 획 이름과 획의 위치 그리고 획과 획 사이의 상관관계를 이용하여 획의 순서를 비교하게 된다.

4-1 획 분류

[10]에서 정의한 획 순서의 기본 규칙은 표 1 과 같다. 이 규칙은 가장 일반적으로 적용될 수 있는 규칙이지만 예외가 많다. 표 1 의 규칙들을 기반으로 하여 특정 부수 글자나 글자의 일부에 대해서 예외적인 규칙에 대해, 획을 따로 분리하여 적용하는 방식으로 획의 순서를 정하고자 한다.

표 1. 획 순서 결정 규칙[10]

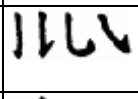
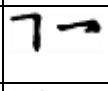
규 칙	모양	획 순
가로획 → 세로획	十	一 十
왼쪽삐침→오른쪽삐침	人	丿 人
위 → 아래	三	一 二 三

왼쪽 → 오른쪽	州	
바깥 획 → 안쪽 획	月	
안쪽 획 → 닫는 획	四	
중간 획 → 양 쪽 획	小	

획의 순서를 정하기에 앞서 우선 되어야 할 것은 모든 획들에 대하여 레이블(label)을 정하는 것이다. 획 순서 규칙은 획 종류와 획과 획 사이의 상대적인 위치관계 등에 의하여 결정되므로, 획을 종류에 따라 분류하는 것이 중요하다.

획은 표 2 에서 정의된 바와 같이 모두 8 가지 획의 형태로 나눈다[10]. 획은 획을 구성하고 있는 벡터의 방향, 벡터들 간의 관계 등에 의해서 결정된다.

표 2. 획 모양에 따른 분류[10]

레이블 (label)	획 모 양
DOT	
HORIZONTAL	
VERTICAL	
HOOK	
LEFTFALLING	
RIGHTFALLING	
TURNING	
RISING	

[10]에서 분류한 바와 같이 8 가지의 획으로 분리하여 표 1 의 규칙을 적용하는 것만으로는 예외적인 규칙을 적용하기 힘들므로, 획과 획들 사이의 상관관계를 이용하여 더 세분화하여 획을 정의한다. 표 3 은 표 2 에서 정의한 획을 기반으로 하여, 획들 간의 상관관계에 따른 세부적인 획들을 분류한 것이다. 세분화된 획들은 표 1 의 규칙에 대해 획의 모양에 따라 다르게 획 순서가 결정될 수 있는 경우를 분류한 것이며, 이는 주변 획들과의 관계에 의해 결정된다.

표 3. 획 세분화

기본 획	세부 레이블	획모양	비 고
DOT			나누지 않음
HORIZONTAL	NONE		관계없음
	CROSS		다른 획과 교차되는 획
	LEFT		다른 획의 왼쪽에 붙음
	RIGHT		다른 획의 오른쪽에 붙음
	CLOSE		口자 日자 등의 닫는 획
VERTICAL	NONE		관계없음
	T		王자의 세로획
	CROSS		다른 획과 교차
	LEFT		口자 日자 등의 세로획
HOOK	NONE		관계없음
	CROSS		다른 획과 교차
	CROSS2		子자의 세로획
LEFTFALLING	NONE		일반적인 획
	SYM		일반적 대칭형 획
	SYM2		一자 위의 대칭형 획
	RIGHT		오른쪽 삐침
RIGHTFALLING	NONE		일반적인 획
	SYM		일반적 대칭형 획
	SYM2		一자 위의 대칭형 획
TURNING	NONE		일반적인 획
	F		民자 부분의 오른쪽 상단 획

4-2 레이블링을 이용한 획 분류 알고리즘

릴렉세이션 레이블링(Relaxation Labeling)은

각 개체들의 특성을 이용하여 모든 개체를 1 개 이상의 레이블로 정의하고 개체들 간의 관계에 일관성을 가지도록 레이블을 반복해서 갱신하는 방법이다[11][12]. 이 장에서는 레이블링 기법을 모든 획들에 대하여 수행하며, 표 3 에서 세분화한 획들의 레이블들을 전체 레이블들의 집합 정의하고, 표 3 에서 정의된 획들 간의 관계(relation)들은 모든 획에 대하여 세부적인 레이블을 결정할 수 있도록 하는 규칙이 된다. 임의의 글자에 대하여, 앞 장에서 분리한 모든 획의 집합을 $Q=\{s_1,s_2,...,s_n\}$ 라고 하면, 임의의 획 s_i 에 대하여 표 2 와 같이 획을 1 차 분류하고, 그 결과에 따라 표 3 에서 정의된 바와 같이 세분화된 레이블을 하나의 획 s_i 에 부여한다. 이렇게 정의된 획는 다른 임의의 획과의 관계를 이용하여, 서로 인접한 획인 지의 여부, 획의 교차 여부, 획 들의 배치 등의 단서를 이용하여 일관성을 따르지 않는 레이블을 반복해서 제거해 나간다. 이러한 과정은 그림 15 에서 요약된 바와 같이 모든 획들에 대해서 반복해서 적용한다.

$Q=\{s_1,s_2,...,s_n\}$: 모든 획들의 집합
$L=\{l_1,l_2,...,l_n\}$: 레이블들의 집합 (표. 3)

1. 모든 획의 집합 $Q=\{s_1,s_2,...,s_n\}$ 에 대하여, 임의의 획 s_i 의 레이블을 표. 2 의 획 모양에 따라 분류한다.
2. 임의의 획 s_i 는 1 의 결과에 따라 표. 3 과 같이 각 레이블에 따른, 세부 레이블들을 모두 가능성 있는 후보로 갖게 된다.
3. 획 s_i 와 임의의 획 s_j 와의 관계에 의해 가능성이 없는 s_i 의 세부 레이블을 제외한다.
4. 의 세부 레이블이 하나로 결정될 때까지 3 의 과정을 반복한다.

그림 15. 레이블링 알고리즘

4-3 획 우선순위 결정

모든 획의 레이블이 결정되고 나면, 획의 레이블과 획의 위치 관계에 의하여 획들 사이의 우선순위를 결정해야 한다. 모든 획들 간의 우선순위를 가리기 위해 두 획 사이의 연산자 '≤'를 정의한다. 이는 모든 획들의 집합 Q의 임의의 두 원소 s_i 와 s_j 사이의 우선순위를 정하기 위한 이항 연산자이다. 표 3 에서 정의된 획 우선순위 규칙은 표 1 의 획 순서 규칙에 우선하며, 표 3 의 규칙이 적용되지 않는 모든 두 획 사이의 관계는 표 1 의 규칙에 의하여 결정되며, 이 두 가지 규칙에 의해 연산자 '≤'가 정의된다.

표 4. 두 획 사이의 우선순위 관계

먼저 쓰는 획	나중에 쓰는 획
DOT	HORIZONTAL_CLOSE
HORIZONTAL_CROSS	VERTICAL_CROSS
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	LEFTFALLING_SYM
HOOK_NONE	RIGHTFALLING_SYM
LEFTFALLING_NONE	RIGHTFALLING_NONE
LEFTFALLING_SYM2	HORIZONTAL_NONE
LEFTFALLING_SYM2	HORIZONTAL_CROSS
LEFTFALLING_RIGHT	RIGHTFALLING_NONE
RIGHTFALLING_SYM2	HORIZONTAL_NONE
RIGHTFALLING_SYM2	HORIZONTAL_CROSS
TURNING_F	VERTICAL_NONE
TURNING_F	HOOK_NONE

5. 실험 결과 및 분석

이 장에서는 제안한 한자 획 분리 방법과 획 순서 결정 방식이 얼마나 정확한 성능을 보이는지를 보이기 위해 한자 능력 검증시험 8 급 50 자와 5 급 50 자(일부 발췌)에 대한 테스트를 수행하였다.

표 5 와 표 6 은 한자 능력 검증시험 8 급과 5 급에서 발췌한 글자 각각 50 글자에 대한 획 분할 실험의 결과이다. 비교적 한자의 모양이 단순한 8 급 발췌 자와 비교적 한자의 모양이 복잡한 5 급 발췌 자 사이의 획 분할 성능차이는 거의 없다.

표 5. 획 분할 실험 (한자 능력 검증시험 8 급)

획 분할 성공 (47 자)	실패 (3 자)
校 九 國 軍 金 南 女 年 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 父 弟	敎 水 長

표 6. 획 분할 실험 (한자 능력 검증시험 5 급)

획 분할 성공 (47)	실패 (3)
價 加 角 强 件 見 固 果 過 果 區 念 多 當 待 童 來 領 禮 老 理 望 面 米 服 事 死 査 産 鮮 船 植 弱 場 前 正 種 體 致 則 風 夏 寒 形 號 話 會	男 數 幸

표 7 과 표 8 은 한자 능력 검증시험 8 급과 5 급 글자에 대한 실험으로, 앞에서 획 분할에 성공한 글자 각각 47 자에 대한 결과이다.

표 7. 획 순서 결정 실험
(한자 능력 검증시험 8 급)

획 순서가 바른 자(42 자)	실패(5 자)
校 九 國 軍 南 年 大 東 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 金 山 父	弟 女 母 民 學

표 8. 획 순서 결정 실험
(한자 능력 검증시험 8 급)

획 순서가 바른 자(40)	실패(7)
價 角 强 件 見 固 果 區 多 當 待 童 領 禮 老 理 望 面 米 服 事 死 査 産 鮮 雪 弱 場 前 體 致 則 風 夏 寒 形 號 話 會 加	過 念 來 船 植 正 種

한자 능력 검증시험 8 급의 글자보다 5 급의 글자가 보다 더 복잡하고 경우의 수가 많기 때문에, 획의 순서 결정의 정확성 향상을 위해서는 보다 더 상세히 레이블을 분류하여 정의할 필요가 있다. 특히 글자가 복잡해지고, 여러 기본 자들의 조합됨으로써, 글자의 크기가 작아지고 변형되는 경우가 생기기 때문에, 획들의 관계를 제대로 인식 못 하는 경우가 있었다.

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. 그리고 자동 획 순서 결정 알고리즘은 획 순서 규칙을 잘 따르는 글자에 대하여 잘 적용됨을 보였다. 획의 기본 순서를 잘 따르는 글자의 경우 일반적인 규칙을 통하여 쉽게 획 순서가 결정되며, 일반적 규칙을 따르지 않는 글자의 경우, 글자의 획이 구성되는 획들의 관계를 정의하여 레이블로 정의한 후, 레이블링 문제로써 획의 순서 문제를 해결하였다. 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.

참고 문헌

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

[2] 한자박사, http://www.hanjadoc.co.kr/

[3] 사이버서당, http://www.cybersodang.co.kr/

[4] 아이한자, http://www.ihanja.com/

[5] 마법천자문, http://www.magichanja.com/

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

[7] Farin, G., Curves and Surfaces for Computer Aided Ceometric Design, Academic Press, New York, 1988.

[8] 구상옥, 장현규, 정순기, "한자 학습을 위한 획순 애니메이션 및 모바일 컨텐츠 저작 도구 개발," HCI2005 학술대회논문집, Vol. 14, No. 1, pp. 636-642, January31-February3, 2005.

[9] Sang Ok Koo, Hyun Gyu Jang, Soon Ki Jung, Efficient Stroke Order Animation of the Chinese Character, KCJC, 2005

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

[11] Dana H. Ballard and Christopher M. Brown, “Computer Vision ”, Prentice Hall, pp.408-430, 1982.

[12] R. A. Hummel, S. W. Zucker., “On the foundations of relaxation labeling processes,” Trans. Pattern Analysis and Machine Intelligence, vol. 5, no. 3, pp 276-287, 1983.