Path Following에 의한 자모추출 한글인식 Algorithm

황도찬* · 김성식*

Hangul Recognition Using The Path Following Algorithm

To-Chan Hwang* and Sung-Shick Kim*

Abstract

본 연구는 컴퓨터에 의한 인쇄체 한글의 인식방법을 제안하고 있다. 일반적인 인식방법에서는 세선화과정 후의 이미지를 처리하고 있으나, 본 연구는 이 과정을 거치지 않고 원 이미지로부터 직접 패턴점들을 찾아내고, 이들을 이용하여 획을 결정하고 자모를 분리하였다. 문자 판별시에는 한글 의사 결정 나무(Decision-Tree)를 이용하여 자소를 분리하고 판별하였다. 본 연구는 자형에 관계없는 인식 방법을 제안 하였으므로 필기체 한글 인식에 기초를 제공하게 된다.

1. 서 론

정보화 사회라 일컬어지는 현대사회에 있어서 보다 빠르고 신속하고 정확한 정보의 처리가 더욱 요구되어지고 있고, 또한 컴퓨터의 급속한 발달로 인하여 빠르고 정확한 정보의 처리가 가능해지면서 정보처리시 인간의 개입을 줄이는 자동화에 대한 요구도 비례하여 커져 왔다.

우리가 일상생활에서 쓰는 문자의 처리에 대한 자동화의 필요성도 이런 맥락에서 대두되어 왔고, 현재 여러가지 방법[1, 2, 4, 5, 6, 7, 8]으로 이에 대한 연구가 활발히 진행되고 있다. 우리나라의 한글에 대한 자동인식문제는 약 20년전부터 연구되어 왔는데 영어나 일어와는 달리 한글의 독특한 특성으로 인해 그 연구가 매우 어려운것이 사실이다. 한글은 조합문자라는 점에서 일반적인 언어와는 다른 특성을 가지고 있기 때문에 인식에 어려움이 있다. 한글에서는 초성으로 가능한 자모가 19자, 중성으로 21자이고, 종성으로는 무음소를 포함하여 28자가 사용이 가능하기 때문에 이들을 조합할 경우 글자의 수는 약 만 4천여자에 이르므로 [3] 이를 글자별로 인식하는 것은 그 방대한 정보량으로 인해 거의 불가능하다. 이는 결국 한글의 경우는 어떠한 형태로든 모든 문자를 유사군(類似群)으로 대분류를 시켜야 한다는 것을 의미한다. 일반적으로는 한글에 있어서는 구성 형태에 따른 6형식 분류를 하거나, 초성, 중성, 종성으로 자모별 분리를 시켜 문자를 판별하는 방법을 취하고 있다. 인식에 어려움을 주는 또다른 점으로는 자형에 따라 다소의 차이는 있겠으나 일반적으로 한글은 전 문

* 고려대학교 산업공학과

자의 25% 이상의 자모들이 수직과 수평으로 복잡하게 연결되어 있고 이는 자소의 분리를 매우 어렵게 하는 요소가 된다. 즉, 자모의 분리를 통한 인식방법의 경우 연결된 자모의 처리여부가 결국 문자 인식률에 커다란 영향을 미치며 그 방법의 성패 여부를 결정짓는 다고 해도 과언은 아니다.

이러한 어려움들을 극복하기 위하여 여러가지 인식방안이 제시되고 있으나 아직 상품화되어 사용되는 경우는 실질적으로 없다.

본 연구는 인쇄체 문자 인식을 목표로 해서 전처리과정(Preprocessing)인 세선화(Thinning)없이 자모를 분리(Segmentation)해내는 방법의 세선화 과정을 전제로 한다. 이 경우 세선화방법에서의 내용의 차이가 존재하고 이것은 다음의 처리 과정에 영향을 끼치게 된다. 세선화는 차후의 처리과정에서 커다란 간편성을 주는 면도 있지만, 문자의 본래의 이미지에 정보의 상실을 가져오기 때문에 일반성을 상실하게 되는 면도 있다. 인쇄체 문자든, 필기체 문자든 한글의 경우에는 그 복잡성과 글자간의 유사성 때문에 가능한 한 본래의 이미지로부터 정보의 상실을 최소한으로 줄여야 하며 이 과정은 많은 처리시간을 소요한다는 점과 함께 세선화 과정의 단점이기도 하다. 여기서 사용된 문자 이미지는 32×32 비트 맵 방식을 기본(실험대상)으로 하였으나 다소의 변화에는 무관하도록 고안되었다. 각 자모의 분리방법은 기본 8방향 선분 추출 방법으로 구성된 각 자소의 고유 경로(Path)를 따라 형성한 의사 결정 나무(Decision-Tree)와 추출한 획과의 비교 추출을 통해서 이미지로부터 자소 분리를 시도하였다. 최종글자 판단은 추출된 자소에 고유의 식별 번호를 부여하여 판단 하였다.

2. 인식 방법

2-1. 인식 단계 :

본 연구는 Binary 이미지로 들어오는 문자에 대한 데이타를 인식 알고리즘에 넣어 문자를 판별해내는 과정에 대한 연구를 하였다. 입력 데이타를 32×32 매트릭스(Matrix) 형태의 이진 데이타로 변환한 후 이를 기본으로 인식 알고리즘을 수행하여 보았다. 보통 인식 방법에 많이 쓰이는 입력 데이타에 대한 세선화(Thinning) 과정을 생략하고, 패턴의 추출시에 원 데이타로부터 필요한 패턴을 뽑아낼 수 있는 방법을 사용하였다. 또한 글자의 두께에 따라 불필요하게 뽑아진 패턴들을 제거하도록 하였으나, 글자의 크기에 비해 획이 매우 두꺼울 경우 올바른 추출에 좋지 않은 영향을 끼칠것으로 생각된다.

이렇게 하여 추출된 점들은 그 성격에 따라 각기 다른 점들과 인터링크(Interlink)를 시키는데, 이렇게 인터링크된 선분들은 문자의 획으로 판단하기에는 매우 부족하며, 이런 문제를 해결하기 위하여 2차에 걸쳐서 구부러진 선분의 펴는 작업과 불필요한 선분의 제거 작업을 수행 하였다.

이러한 선분들로부터 자모 추출시에, 본 연구에서 만든 각 자소에 대한 의사 결정 나무(Decision-Tree)를 이용해서 비교 추출 방식으로 자소를 판별하였다. 먼저 초성을 추출한 후 결정된 초성의 위치와, 다음 발견되는 선분의 위치와 방향을 토대로 중성 또는 종성을 추출한다.

이렇게 하여 추출된 각 자소에는 고유의 식별번호가 부여되는데 이것을 이용하여 KSC-5601-1989 코드와 비교하는 방식으로 문자를 판별하였다.

그림 1은 위의 내용을 도시화한 것이다.

2-2. 문자 형상으로부터 패턴 추출 :

한글 이미지로부터 패턴 추출시 본 연구에서는 추출하고자 하는 패턴의 형태를 일반적으로 많은 연구에서 사용하는 다음의 것을 이용하였다.

추출 패턴 종류 :

시점(Starting point), 종점(Termination point), 굴곡점(Bending point), 분기점(Branching point).

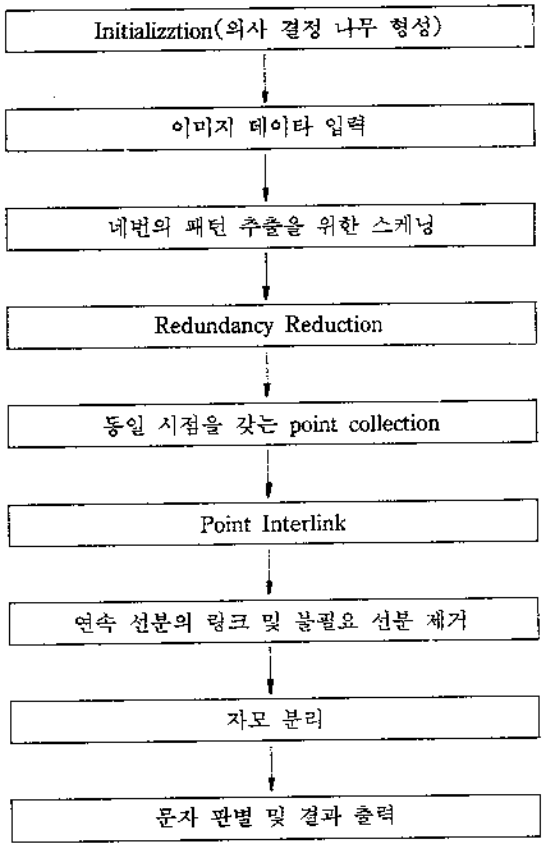


그림 1. 인식 단계

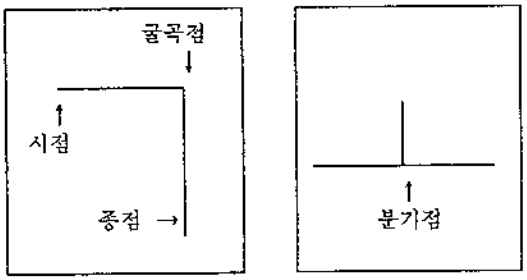
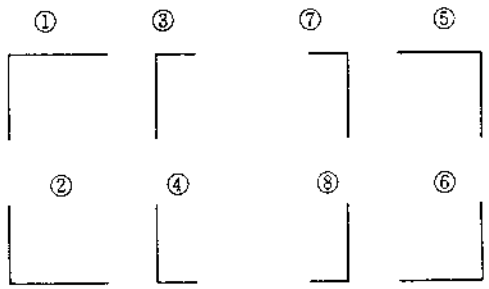


그림 2. 패턴 예

이와같은 패턴들을 찾아내기 위해 다음의 방법들을 사용하였다. 먼저 시점, 종점, 굴곡점을 찾아내기 위한 패턴 추출용 Tablet은 다음의 그림 3과 같은 것을 사용하였다. 물론 이러한 것은 32×32 매트릭스 형태의 이미지를 기준으로 하였으나, 다소의



세로길이 : 3 PIXEL.
가로길이 : ①, ②, ⑤, ⑥의 경우 3 PIXEL.
③, ④, ⑦, ⑧의 경우 2 PIXEL.

그림 3. 패턴 추출용 Tablet

변동된 크기에도 사용할 수 있다.

다음으로 굴곡점 또는 분기점을 찾기 위한 경우 이런 특성은 4번의 각기 다른 방향의 스캐닝을 통해서 결정되어 지는데, 각 방향의 스캐닝시에 위의 Tablet를 이용해 시점을 찾아내고 이후 이 시점에 연결되어 나타나는 점들을 찾아나간다. 연결되어 나가는 도중 서로 다른 시점을 가지는 점이 만나면 이점은 분기점내지는, 굴곡점이 될 가능성이 있는 점으로 보고 이를 표시한다.

4번의 스캐닝 방향은 그림 4와 같다. 이와 같은 방법으로 특성점을 찾아내는 경우 32×32 크기의 데이터에 대해서는 글자의 두께가 이미지의 크기에 비해 비정상적으로 크지 않은 한, 다른 인식 방법에서 일반적으로 적용하는 세선화 과정을 하지 않더라도 특성 패턴을 찾을 수 있고, 이러한 사실은 세선화 과정을 취하는 다른 방법보다 처리 작업량의 감소를 가져온다.

위의 패턴 추출 방법은 직선에 대해 특성점을 추출하는 경우에 사용되는데 한글은 이러한 직선만이 존재하는 것이 아니고 'ㅇ' 이나 'ㅎ'와 같이 원의 특성이 존재하는 자소가 있는데, 이러한 자소를 인지하기 위하여는 다른 방식을 사용하여야 하는데 원에 있어서 그 내부는 다른 연결 자소에 영향을 미치지 않으므로 다음 그림과 같이 3선분의

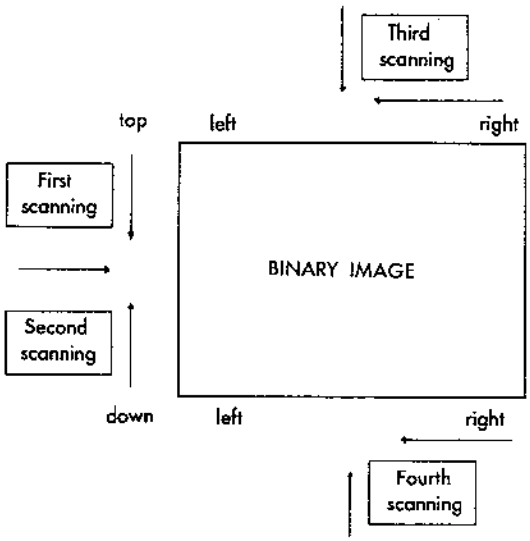


그림 4. 스케닝 방향

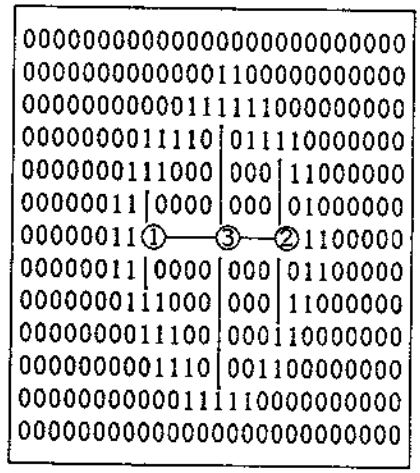


그림 5. 원에대한 패턴 추출

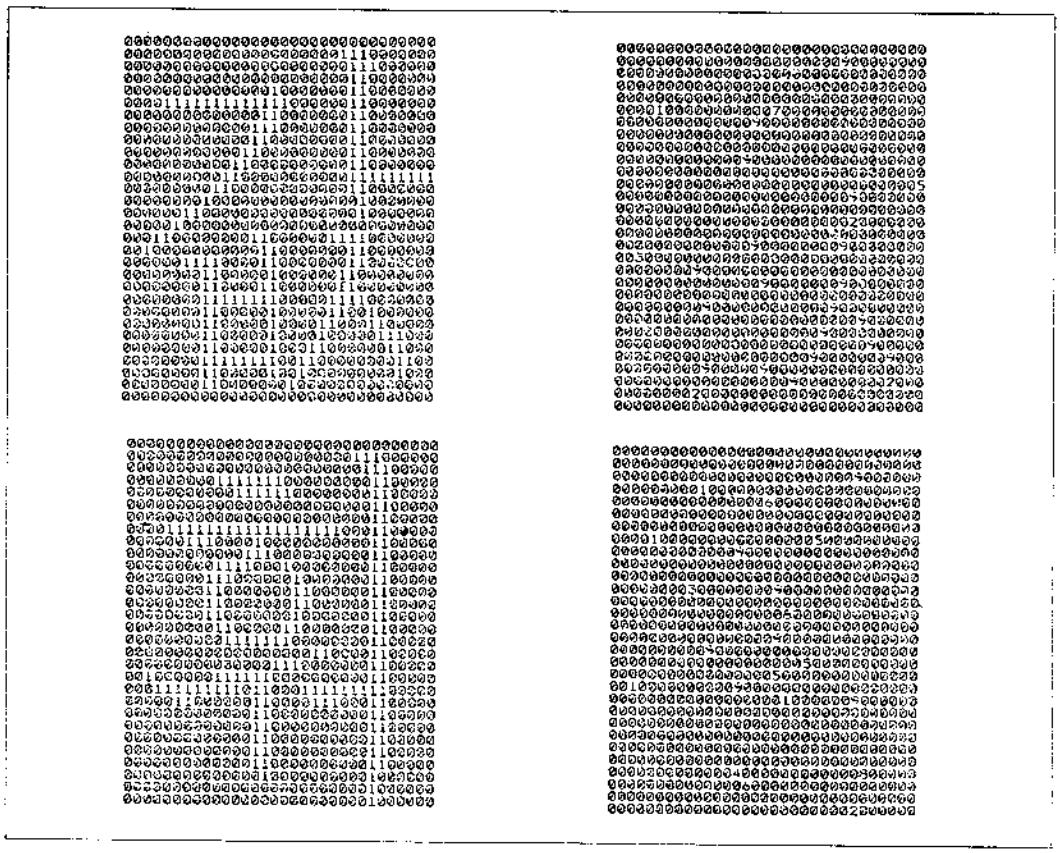


그림 6. 패턴 추출 예

길이의 비(Rate)를 통해 원의 존재 여부를 판단한다.

원의 특성 존재가 확인되면 고것은 'ㅎ'과 'ㅇ의 두 자소가 될 수 있는데 이것에 대한 판단은 자모 분리 과정에서 결정하게 되고, 원의 특성 패턴이 처음 발견된 부분은 초성과 종성 여부를 판별하는데 결정 요소가 된다. 초성에서의 원의 특성 존재는 'ㅎ'과 'ㅇ' 두가지의 자소가 될 수 있는 반면, 종성의 경우는 'ㅎ', 'ㅇ', 'ㄴㅎ', 'ㄹㅎ'이 될 수가 있으므로 자모 추출시에 이를 고려해야 한다.

위의 모든 방법을 통해서 얻어진 특성점들은 가능한 모든 경우에 대해 추출하는 것이기 때문에, 이 가운데는 불필요한 점(Redundancy) 까지도 찾아내는 경우가 많은데 이는 문자 이미지가 좋지 않을수록, 획의 두께가 두꺼워 질수록 심해진다. 이러한 점들은 이후 알고리즘 적용시에 오류를 유발시킬 수 가 있는데, 가능한 한 이런 오류를 줄이기 위해 불필요한 점을 제거하여야 한다.

그림 6은 '값'자와 '훼'자의 경우에 대한 원래의 이미지 데이타와 이 데이타로부터 패턴을 추출한 결과를 보여준다.

각 우측의 그림에서 ①-⑧의 의미는 앞서의 그림 6에서 표시한 것과 같고, ⑨의 표시는 굴곡점 또는 분기점의 가능성이 있는 점을 의미한다.

2-3. 추출된 패턴으로부터 획(Strock)의 추출 :

추출된 패턴으로부터 획(Strock)을 결정하기 위하여는 그림 6에서와 같은 추출점을 이용하여야 하는데 여기서는 특성점과 점을 링크시키는 방법을 취하였다. 링크시에 방향 심볼은 8-방향 Primitive와 4-방향 Primitive로 하였으나 주로 8-방향 심볼을 사용하였고, 필요시에는 4-방향 심볼도 사용하였다.

그림 6의 패턴 추출 과정에서 보듯이 모든 가능한 변화점을 찾았기 때문에 그러한 점을 링크시키는

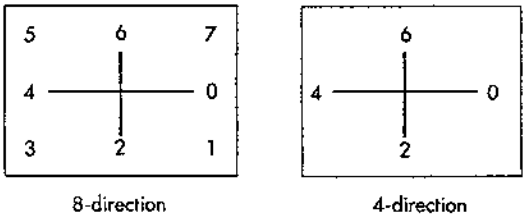
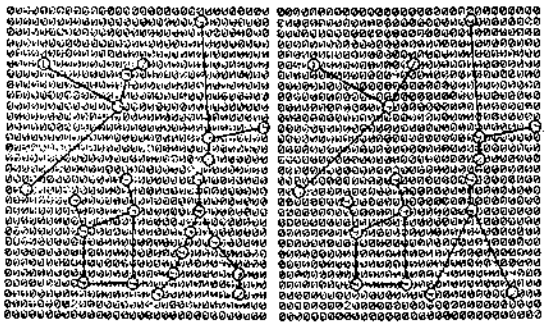


그림 7. 방향 심볼

것만으로는 획을 쉽게 끌어낼 수 없으므로 이에 대한 처리작업을 병행하여야 한다. 첫번째 점들을 링크하는 과정에서는 같은 시점을 같은 점들끼리, 방향이 같은 경우 가까운 점 우선으로, 가능한 각 방향의 연결된 점들을 링크시켰다. 이때에 고려할 점은 0방향으로 링크가 이루어진 경우 7방향의 링크는 이루어 지지 않으며 2방향의 링크가 이루어지면 3 또는 1방향의 링크는 불가하다(이외에 많은 경우가 존재함). 이것은 한글의 경우 그러한 분기점은 존재하지 않기 때문이다. 이때 연결된 각 점들은 상대점의 위치 정보를 가지게 되고, 이 정보는 자모 분리시 패스 추적에 이용된다. 이와같은 여러 제약들을 두면서 모든 특성점들을 링크시켰을 때, 이 결과로부터는 직접 획을 뽑아낼 수 있는 상태가 되지 못하며, 2차에 거쳐서 90도 내로 구부러진 연결점들을 가능하다고 판단되면 펴는 작업을 실시하였다. 왜냐하면 각 자소의 고유 경로는 가던 방향에서 90도 내로 구부러지는 선분이 존재하지 않기 때문에 이 작업을 하여주면 오류를 감소시켜 준다. 1차시기에는 선분을 펴는 작업을, 2차시기에서는 일정길이 미만의 선분을 제거하면서 동시에 1차에서와 같은 작업을 실시하였다. 이때 필요하다고 판단되면 4-방향 심볼을 적용시켜 링크 방향의 재 계산 작업도 수행하였다. 그림 8은 2차에 걸친 펴는 작업전의 링크 결과와 작업후의 결과를 보여주고 있다.

2-4. 추출된 획으로부터 자모의 분리 :



수정작업전 수정작업후

그림 8. 패턴점들의 링크 결과

이미지로부터 자모를 트레이스(Trace)하여 추출하기 위하여는 비교 기준이 있어야 하는데 본 연구에서는 다음의 방법으로 하였다. 모든 자음과 모음 그리고 쌍자음과 복모음, 복자음에 대하여 8-방향 Primitive를 이용한 고유의 패스를 형성하였다. 각 자소에 대해 형성한 패스를 초성, 중성, 종성별로 구분하여 트리(Tree)를 만들면 자소간 상호 배타적인 의사 결정 나무(Decision-Tree)가 형성되며 자모 추출시 이것을 이용할 경우 빠르고 쉽게 사용될 수 있다.

단, 중성의 경우 'ㅓ'와 'ㅢ'는 같은 패스를 형성하는데 중성에 앞서 초성이 반드시 먼저 추출되므로 추출된 초성의 위치를 이용해 이를 구별할 수 있다. 그림 9는 각 자소의 고유 패스를 형성하는 방법과 초성, 중성, 종성에 대한 의사 결정 나무를 보였다.

보통의 자소에 대하여는 위의 방법을 취하고 다음 3가지의 경우는 추가적인 원칙이 적용되었다.

1) 'ㅎ'과 'ㅇ' 자소의 인식.

제 2절에서 파악한 원의 존재 여부를 통해 'ㅎ'과 'ㅇ'의 판단을 한다. 'ㅎ'의 경우는 'ㅇ'과 다르게 바로 위에 0방향의 선분이 존재한다. 즉 이러한 특징을 통해서 원의 존재로부터 'ㅇ'과 'ㅎ'을 구분한다.

2) 'ㅈ' 자소의 인식.

◎ 8-DIRECTIONAL PRIMITIVE에 의한 PATH 결정 RULE :

PRIORITY 1 : 가던 방향의 선분.

PRIORITY 2 : 가던 방향의 가장 왼쪽 선분.

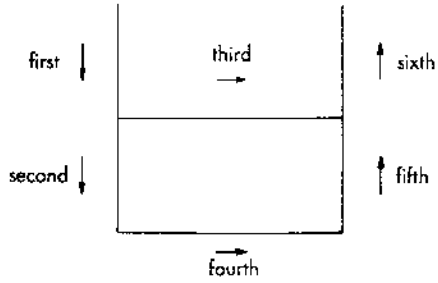


그림 9. 패스 형성 예

위의 트레이스순에 따라 8-방향 Primitive를 적용시켜 패스를 표현하면 다음과 같다.

② → ② → ◎ → ◎ → ⑤ → ⑥

first second third fourth fifth sixth

'ㅈ'의 경우 'ㅈ'의 경우와는 다른점이 하나 있는데 바로 위에 0방향의 선분이 존재한다는 것이다. 이것을 이용해 초성 또는 종성에 'ㅈ'의 존재가 확인되면 'ㅈ'의 존재 가능성이 있는 것으로 보고 이를 확인하였다.

3) 복모음, 복자음의 인식.

모든 획이 붙어있는 경우는 연속적인 하나의 패스가 형성되지만 떨어진 경우에 한해서 ⑨라는 Primitive를 첨가하여 패스를 연결하여 나갔다. 그림 10의 예는 그러한 경우를 보여준다.

◎ 'ㅟ'의 경우 'ㅜ'와 'ㅣ'가 떨어져 있을 때 :

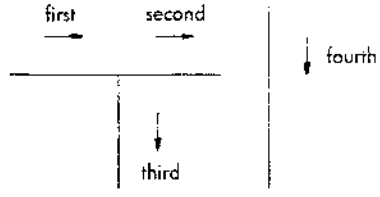


그림 10. 패스 형성 예

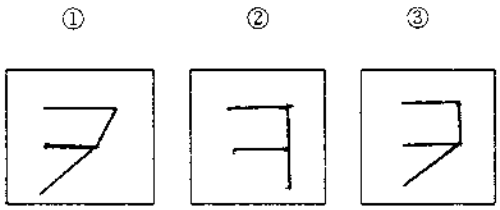
위의 트레이스순에 따라 8-방향 Primitive와 ⑨

Primitive를 적용시켜 패스를 표현하면 다음과 같다.

◎ → ◎ → ② → ⑨ → ②
first second third fourth

그림 11은 위와같은 방법으로 형성한 초성에 대한 의사 결정 나무 예이다. 현재 이 나무에는 각 자모의 기본 모양을 크게 벗어나지 않은 형태에 대한 패스를 넣었다. 그러나, 이 나무 구조에는 필요시 좀더 많은 정보를 쉽게 넣을 수 있다. 예를 들어 'ㅋ'의 경우를 보자.

현재에는 1과 2의 경우의 정보만을 나무에 수록하였으나 3과 같은 정보도 쉽게 넣을 수 있다.



패스 ① : ◎ → ③ → ③ → ④
② : ◎ → ② → ② → ④
③ : ◎ → ② → ③ → ④

그림 11. 패스 첨가

다음의 자소의 판별과정에서 적용되는 몇가지 기본적인 원칙을 나타내었다.

1) 초성, 중성, 종성의 시점(START POINT) 찾는 방법.

분리 방식은 그림 12의 의사 결정 나무를 따라 내려가면서 찾아진 획과 비교 추출 방식을 통해 각 자소를 분리해 낸다.

각 자소중 초성을 가장 먼저 찾고, 종성이 존재할 경우에는 수평 모음이 존재하지 않을 경우에 한해 종성이 먼저 찾아지고, 그외의 경우에는 중성이 먼저 찾아지게 된다. 먼저 찾아진 자소의 크기 및 위치는 아직 찾아지지 않은 자소의 위치를 추적하는데 기준이 되며, 이 방식은 문자가 일반적인 글자 구성 형식에 크게 어긋난 모양이 아니라면 자형에 관계없이 적용될 수 있는 장점을 가지고 있다. 찾아진 각 자소에는 고유의 식별 번호가 부여된다.

2) 붙어있는 자소의 분리

중성 중에서는 초성 또는 종성과 자소간 붙어있는 경우가 많이 발생한다. 'ㅐ', 'ㅒ', 'ㅗ'와 'ㅛ'는 초성과, 'ㅜ'와 'ㅠ'는 초성과 종성에 연결되어 있는 경우가 자주 발생하고, 이밖에 수직 모음 중에서도 종성과 연결되어 있는 경우가 존재한다. 그러므로 연결되어 있는 자소의 분리가 요구되어 지고 있다. 연결되어 자소가 있는 문자들 중 수평 모음이 존재하는 경우 자세히 관찰해 보면 수평 모음의 2와 6 방향의 획과 다른 자소가 대부분 연결되어 있음을 알 수 있고, 이는 수평 모음의 경로 추적중 2방향의 선분과 연결되어 있는 선분을 끊어주면 올바른 분리를 할 수 있다는 것을 의미한다. 수직 모음과 초성과의 연결되어 있는 경우를 살펴보면, 초성이 중성보다 반드시 먼저 찾아지므로 수평 모음과는 다르게 초성의 경로 추적시에 이를 분리 시켜야 한다. 연결 자소를 가지고 있는 문자중에서, 초성은 수직 모음의 0방향의 선분과 대부분 연결되어 있는데 초성은 2, 3, 6 방향의 선분 추적시 0방향의 선분을 만나면 이를 끊어버리면 분리가 가능해 진다.

이외에 연결 자소의 자연 분리도 의사 결정 나무로부터 가능하다. 패스 추적시 연결점에 도달했을 때, 의사 결정 나무에서 더 이상의 추적이 가능하지 않거나, 연결된 자소의 패스가 존재하지 않은다면 자연 분리가 된다.

3) 모호성에 대한 처리

문자의 이미지 데이터의 처리 과정에서, 또는 원 데이타상에서 불량으로 인해 짧은 선분의 획이 무시되는 경우가 자주 발생한다. 예를 들어 'ㅏ'의 경우 2방향의 선분이 짧아 'ㄴ'과 같은 꼴로 나타날 수도 있고, 'ㅂ'의 경우 처음의 2방향의 선분이

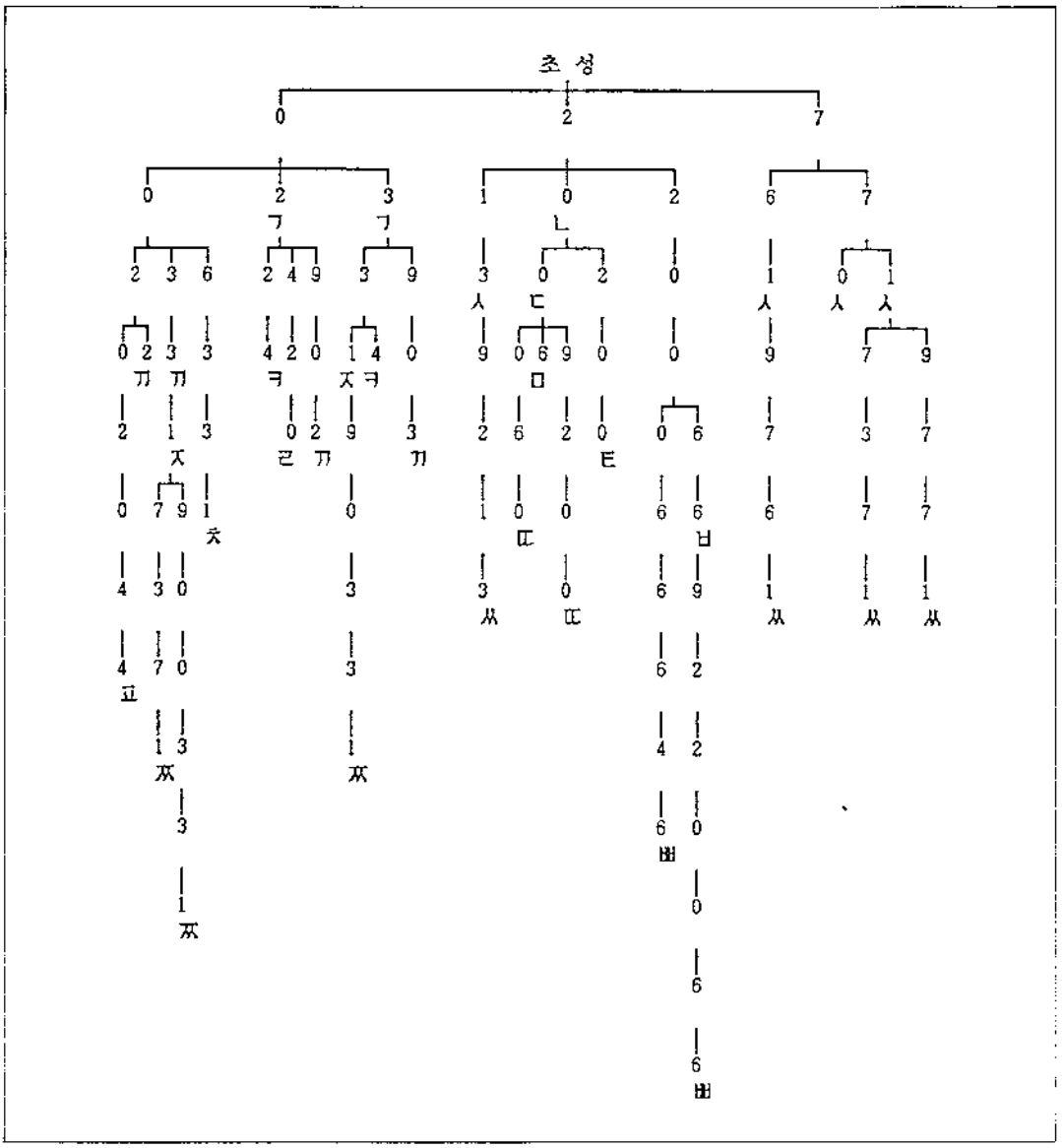


그림 12. 초성에 대한 의사 결정 나무

짧아 'ㅁ'과 같은 꼴로 나타내어지는 경우가 발생한다. 이와같이 모호성이 자주 나타나는 자소에 대해서는 의사 결정 나무에 새로운 꼴에 대한 패스를 첨가하면 이러한 문제를 해결할 수 있다.

다음 그림 14은 이러한 문제에 대한 예를 보이고 있다.

4) 불완전 자소의 처리

일반적으로 패턴점들의 링크시 올바른 처리를 하지 못하거나 원 데이터의 불량으로 인해서 불완전 획을 뽑아냈을시에 의사 결정 나무상의 어느정도 패스만 따라가면 가장 유사한 형태의 자소로 결정지을 수 있다.

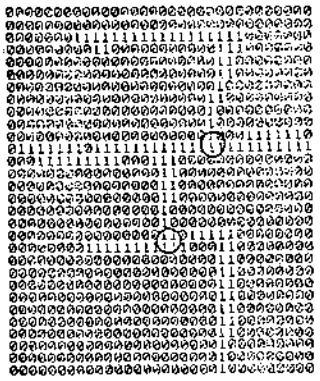


그림 13. 자소의 분리 예

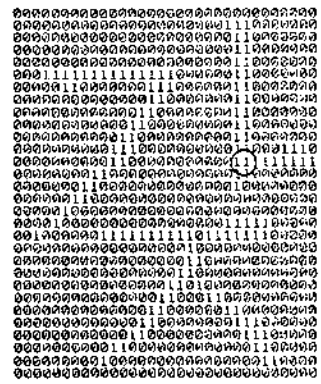


그림 14. 모호성에 대한 처리 예

5) 문자의 판별

최종 결과인 문자 판별은 앞서의 결과로 나타난 각 자소의 식별 번호를 통해서, 입력된 문자의 인식이 올바로 되어 있는가를 판단하였다.

3. 실험 및 결과 분석

다음은 본 인식 알고리즘을 실험한 결과에 대하여 나타 내었다. 실험은 삼성 386-DX(27.3 Mhz) 상에서, 실험 대상은 실험용 한글 데이터 베이스를 이용하여 656자를 실험 하였는데 그 결과는 인식률 85%, 처리속도 0.22초로 나타났다.

오 인식률이 이와같이 높게 나타난 이유는 잘못 인식된 문자의 90% 이상이 Point-Interlink 과정에서의 오류로 인해 발생 하였다. 그러므로 이 부분에 대한 개선된 알고리즘이 요구되며, 완전한 처리가 될 경우 수치상에서 본다면 실용화 수준에 근접하는 인식률 증가를 이룰 수 있다고 본다. 또한 패턴 추출 과정에서도 불필요하게 많은 특성점들이 추출되어 차후의 처리과정에서의 복잡성과 오류를 유발시켰다. 따라서 꼭 필요한 패턴점 만을 추출할 수 있도록 하는 연구가 요구된다.

4. 결 론

한글 문자의 조합성과 그 모양의 다양성이 컴퓨터를 통한 이미지 자동 처리와 자형에 구애 받지 않고, 크기에 민감하지 않은 인식 알고리즘의 개발을 어렵게 한다. 본 연구에서는 한글의 이런 문제점을 해결하는 방향의 연구를 하였다. 일반적으로 많은 한글 인식 알고리즘에서 가정한 특정 자형, 특정 크기에 관한 인식은 실용화 하기에는 많은 제약성을 가지며, 다른 자형, 다른 크기에 대해 인식률이 매우 민감하게 변한다. 또한 여러 자형에 대한 인식을 할 수 있도록 하기 위하여 각 자형의 특징 정보를 가지고 처리를 한 알고리즘의 경우 그 정보량의 방대함이 처리 속도를 느리게 할 뿐만이 아니라, 컴퓨터가 수십가지의 다양한 자형의 정보를 갖고 처리하도록 한다는 것은 거의 불가능하다. 각 자형의 평균 특성을 자모 분리 기준으로 한 알고리즘의 경우 처리 속도에서는 좋으나, 자형이 다른 문자의 인식에 있어서 오류를 범할 가능성이 매우 크다.

한글에 있어서 자형이나 크기에 관계없이 변화하지 않는 특성이 있다. 그것은 'ㅎ'과 'ㅇ'을 제외한 전 자소는 직선 성분으로 구성되어 있다는 것이다. 본 연구에서의 가장 큰 의의라고 한다면 한글의 이런 특성들을 이용, 자모 추출 기준으로 보다 빠르게 처리할 수 있도록 한 의사 결정 나무(Decision-Tree)를 만든 것이라고 할 수 있다. 이

나무의 처리 정보량은 7 Kilobyte 정도 소요하는데, 이는 알고리즘 수행시 주기억 장치(Ram)에 직접 실어서 사용할 수 있는 크기이기 때문에 디스크에서 데이타 베이스를 찾는 시간을 감소 시킨다. 실제로 문자 인식 전 시간의 1/6만이 자모 분리시에서 판별에 이르는 과정에 소요되었다.

본 알고리즘의 인식율이나 처리 속도는 아주 좋다고는 할 수 없으나 그것은 패턴 추출 과정이나, 패턴점들의 링크 과정에서 보다 많은 연구를 하고, 세선화 처리를 한 데이타를 이용한다면 실용화 수준에 접근할 것으로 생각되며 그 가능성이 매우 크다고 생각한다.

참고문헌

[1] 김명원, 이광노, "Pattern 認識을 위한 Neural Network," 전자통신, 제 11 권, 1호, pp. 42-58, 1989.

[2] 김태균, A. Takeshi, "Syntactic 법에 의한 한글의 패턴 認識에 關한 硏究," 전자공학회지, 제 14 권, 제 5 호, pp. 15-21, 1977.

[3] 비트 방식의 한글 문자 표준화에 관한 연구, 공업진흥청.

[4] 이광호, "다중 활자체 한글 인식을 위한 자모의 분리," 한국과학기술원, 전산학과, MCS-87 276, 1989.

[5] 이주근, "한글文字의 認識에 관한 硏究 (Ⅳ)," 전자공학회지, 제 9 권, 제 4 호, pp. 25-32, 1972.

[6] 이주근 외, "한글 패턴에서 Subpattern 분리와 인식에 관한 연구," 대한전자공학회지, 제 18 권, 제 3 호, pp. 1-8, 1981.

[7] 최병욱, T. Ichikawa, H. Fujita, "한글 認識에 있어서의 子素抽出," 전자공학회지, 제 18 권, 제 2 호, pp. 36-43, 1981.

[8] Simon Kahan, Theo Pavlidis, "On the Recognition of Printed Charac ters of Any Font and Size," *IEE Transactions on Pattern Analysis and Machine Intelligence*, Vol Pami-9, No. 2, pp. 274-287, March, 1987.