구문분석과 패턴분류를 이용한 한글인식

강 현 철, 최 동 혁, 이 완 주, 박 규 태
연세대학교 전자공학과

Hangul Recognition using Syntax Analysis
and Pattern Classification

Hyunchul Kang, Dong Hyuk Choi, Wanjoo Lee, Kyu Tae Park
YOUNSEI UNIV.

요 약

한글에서 발생하는 자소의 접촉에 의한 오인식을 해결하기 위하여, 접촉점을 중심으로 원소를 변환하여, 확장 가능한 구조를 모두 검증하고, 수락된 후보패턴에 대하여 가장근접한 패턴 클래스로 할당하는 한글 인식방법을 제안한다.

프로그램드 배열문법을 이용하여 화소의 2차원 배열에서 입력패턴을 인식하고, PEACE(Primitive-Extraction and Attribute-Computation Embeded)파싱을 이용하여, 원소(*primitive*)의 추출과 속성(*attribute*)의 계산을 구문분석 과정에 통합하고, 전체 시스템이 동적인 구조를 갖게하여, 1차원 스트링으로의 변환에 따르는 패턴의 변형과 부가적인 노력을 억제한다.

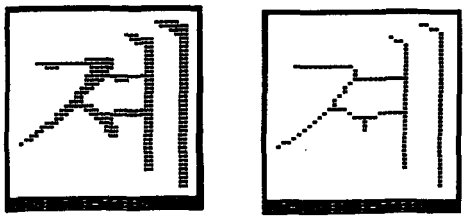
I. 서 론

한글은 24자의 기본 자소의 조합으로 간주할 수 있으며, 기본 자소는 획의 조합으로 생각할 수 있다[1]. 궁극적으로, 한글 패턴은 획의 조합으로 고려할 수 있으며, 복잡한 패턴의 자소는 간단한 자소로부터 획을 확장함으로써 생성시킬 수 있어, 복잡한 패턴은 간단한 패턴을 내포하게 된다.

따라서, 각 자소의 접촉은 획을 확장하여 다른 자소로 인식되거나, 다른 자소의 획을 침범하게 되어 오인식의 주요한 원인이 된다.

그림 1은 입력패턴과 획을 추출하기 위한 전처리를 수행한 후의 골격패턴이다. 입력패턴 '제'는 초성과 중성의 접촉에 의하여 'ㅈ' 이 'ㅊ'으로 확장하게 되어, '제' 또는 '계'로 인식될 수 있는 모호함이 발생하게 된다.

이러한 자소의 접촉에 의한 오인식을 해결하기 위하여서는 접촉점을 중심으로 확장 가능한 구조를 모두 검증하여야 할 필요가 있으며, 전체 시스템이 동적인 구조를 가져야 하며, 구조적 정보외에도 수치정보, 문맥적 의미를 같이 고려할 수 있어야 한다.



a) 입력 패턴 b) 골격 패턴

그림 1. 입력 패턴과 골격 패턴

본 연구에서는 원소변환(primitive transformation)을 이용하여 자소의 접촉을 해결하고, PEACE(Primitive-Extraction and Attribute-Computation Embeded) 파싱을 이용하여, 원소의 추출과 속성의 계산을 구문분석과정에 통합 하여, 전체 시스템이 동적인 구조를 갖게한다.

Ⅱ. 한글 자소의 표현

1. 프로그램드 배열문법

문자는 여러가지 변형이 존재하며, 패턴 클래스의 수가 방대하여, 구조적 인식방법(structural pattern recognition)을 이용한다[2,3]. 특히, 획의 조합으로 표현될 수 있는 한글은 획의 구조적 관계를 성형언어의 구문으로 분석하여 인식하는 구문론적 방법(syntactic approach)이 사용된다.

한글인식에 구조해석을 이용한 접근으로는 세선화된 입력패턴을 트리로 표현하여 인식하는 방법[4], 입력패턴의 원소에 속성을 부여하여 문맥정보를 이용하는 방법[5]이 제안되었으나, 입력패턴을 1차원적인 스트링으로 변환하여, 이를 다시 2차원적 구조로 해석하여, 스트링의 변환에서 발생하는 오류가 구문분석에 영향을 주게되고, 스트링의 변환에 필요한 부가적인 노력이 필요하다.

본 연구에서는 프로그램드 배열문법(PAG, Programmed Array Grammar)을 이용하여, 표준패턴의 2차원적인 구조를 성형언어의 구문으로 표현하고, 이를 토대로 입력패턴의 구조를 분석한다.

PAG는 배열문법에 프로그램드 문법을 도입한 개념으로, 배열문법은 문장과 그림이 같이 존재하는 문서를 하나의 통합된 문법으로 처리하고자 하는 개념에서 제안되었으며[6], 성형문법을 2차원으로 확장하여 정의한다[7].

$$G_A = (V_n, V_t, P, S, \#)$$

여기에서, V_n은 비종단기호(nonterminal symbol)의 유한집합이고, V_t은 종단기호(terminal symbol)의 유한집합이다.

#는 배경기호 또는 공백기호로, $\# \notin (V_n \cup V_t)$이다. 배경기호는 오토마타이론에서의 공백기호와 유사한 개념으로 비공백기호의 확장을 가능하게 하는 동시에 무제한 확장을 방지하는 기능을 갖는다. S는 시작기호로 $S \in V_n$이다.

생성규칙 P는 $\alpha \rightarrow \beta$의 형태를 갖는다. 여기서, α, β는 배열이다. $\alpha \rightarrow \beta$의 생성규칙이 존재하고, α가 ζ의 부배열이고, β가 ξ의 부배열이면, ξ는 배열 β의 대응기호만 α로 대치한 ζ와 같다. 이때, ζ는 ξ를 생성시킨다고 하고, $\zeta \rightarrow \xi$로 표시한다.

배열문법에서의 생성규칙은 성형문법에서와 동일하게 4가지 유형을 갖는다.특히 α와 β가 기하학적으로 동일한 형태의 배열을 갖게 될 때, 동형 배열 문법(isotonic array grammar) 이라고 하며, 실제 응용에 많이 사용되는 문법이다. 본 연구에서는 2차원 배열상에서 원소의 2차원 배열을 발생시킬 수 있는 ICFAG(Isotonic Context-Free Array Grammar)를 이용하여 한글자소의 문법을 구성한다.

프로그램드 문법은 Rosenkrantz에 의하여 제안되었으며[8], 생성규칙에 적용순서를 부여하여 구문을 분석하며, 종래의 programming기법과 비슷한 과정으로 인식의 과정을 진행시킬 수 있으며, 문법의 추론을 쉽게 하며, 생성규칙은 문법보다 넓은 범위의 언어를 처리할 수 있기때문에, 보다 강력하고, 편리한 도구를 제공한다[9].

본 연구에서는 배열문법의 생성규칙에 순서를 부여한 PAG를 이용하여 표준패턴의 구조를 표현하고, 이를 이용하여 입력패턴의 구조를 분석한다. PAG G는 다음과 같이 정의될 수 있다.

$$G = (V_n, V_t, P, J, S, \#)$$

여기서, V_n, V_t, S는 배열문법에서의 정의와 같고, J는 생성규칙의 label의 유한 집합으로, 각 생성규칙은 하나의 label $r \in J$를 가지며,

(r) $\alpha \rightarrow \beta$ S(V) F(W)

의 형태를 갖는다. 생성규칙은 배열문법의 4가지 형태 중 하나를 취하며, 각 생성규칙은 두개의 분기필드 S(V)와 F(W)를 갖는다. S(V)는 성공분기 필드(success go-to field)이고, F(W)는 실패분기 필드 (failure go-to field)이며, 각 분기필드는 label의 집합이다.

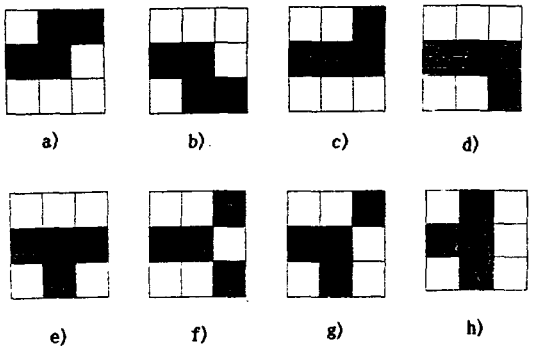
문법 G에서의 언어의 유도과정은 시작기호에 첫번째 생성규칙을 먼저 적용한다. 현재의 문장 γ이 하나 이상의 부배열 α을 포함하고 있을 때, 가장 왼쪽의 α에 생성규칙을 적용하고 다음에 적용할 생성규칙을 성공분기필드에서 선택한다. 문장이 α를 포함하고있지 않을 때는 실패분기필드에서 생성규칙을 선택한다. 더이상 적용할 분기필드가 없을 때, 유도는 멈추게 된다.

2. 기본자소의 문법

한글은 초성, 중성, 종성의 조합 문자로 파악할 수 있으며, 가능한 조합의 수는 14,000여 가지에 이르게 된다. 또, 인쇄체의 경우, 활자체에 따라 여러가지 다양한 형태의 변형을 가지며, 필기체의 경우, 각 개인의 필기의 습관에 따라 변형이 초래되게 되어, 영문자의 경우처럼 각 문자별로 표준문법을 작성하는 것은 불가능하게 된다. 따라서, 한글의 경우, 각 자소별로 문법을 작성하여, 자소의 조합으로 문자를 인식하여야 한다.

각 자소는 획의 2차원적인 구조로 간주할 수 있기 때문에 2차원적 구조를 표현할 수 있는 원소를 설정하여야 하며, 본 연구에서는 3*3 부배열에서 원소를 정의하였다. 원소는 획의 방향성분을 나타내는 원소와 획의 방향의 변화를 표현하는 원소로 구분하여 설정한다. 방향성분을 나타내는 원소로는 수평선, 좌대각선, 수직선, 우대각선 성분을 나타내는 배열로, 대응하는 종단기호는 a, b, c, d로 정의한다. 방향의 변화를 나타내는 원소로는 시작점, 끝점, 굴곡점(fraction), 분기점(bifurcation), 교차점(cross), 끝점이 있으며, 속성값으로 방향값을 가지며, 대응종단기호는 각각 stx, end, fr, bf, cr로 정의한다.

그림 2에 원소들의 배열중 후보화소가 들인 경우(진행방향이 0)를 나타내었다.



a), b), c), d) : 수평원소, e), f), g), h) : 분기점.

그림 2. 진행방향이 0일 때 원소의 정의
(후보화소가 들인 경우)

한글의 기본자소는 이러한 원소의 구조적 관계로 표현할 수 있으며, 그림 3에 자소 'ㅈ'패턴의 구조와 문법을 나타낸다. 시작점(종단기호는 stx)에서 수평원소(a)를 탐색하다 0방향과 6방향으로 갈라지는 분기점(bf, 속성값 006)을 만나게 되면 'ㅈ'의 머리부분의 수평 획을 찾고, 분기점을 복구하여 수직원소(c)를 찾는다. 6방향으로 진행하다, 5방향과 7방향의 분기점(속성값 657)을 만나면, 5방향의 좌대각선 선소(d)를 찾고('ㅈ'의 오른쪽 다리), 다시 분기점을 복귀하여 끝점을 만날 때까지 우대각 선소(b)를 찾는다('ㅈ'의 왼쪽 다리). 입력패턴이 이와 같은 원소들로 이루어지며, 원소들의 관계가 이와 같은 구조를 만족하면 'ㅈ'의 패턴이라고 인식한다. 이 때, 방향은 Freeman의 code값으로 정의한다. 'ㅈ'의 패턴은 위치에 따라 그 모습이 다르고, 문자체에 따라 변형이 초래하게 되지만, 그림 3의 표준패턴을 기준으로 변형을 흡수하게 하여, 'ㅈ'의 패턴을 인식한다. 24자의 기본 자소의 문법을 작성하여 입력패턴의 구조를 분석하고, 구문분석시 접촉점을 중심으로 원소변환을 수행하여 확장 가능한 구조를 모두 검증한다.

S(V)

1. S# -> stxA 2
2. A# -> aA 2,3
3. A# -> bfC 4
   # B
4. C# -> aC 4,5
5. C# -> aend 6
6. B -> c 6,7
   # B
7. B -> bf 8
   # # D E
8. D -> b 9
   # D
9. D -> b 10
   # end
10. E -> d 10,11
    # E
11. E -> d
    # end

그림 3. 'ㅈ'의 표준 패턴과 문법.

Ⅲ. 한글 인식

1. PEACE Parsing

구문분석(syntax analysis)은 성형문법을 이용하여 입력패턴의 구조를 분석하는 과정으로, 작성된 문법으로 입력패턴을 수용할 수 있으면 구조적으로는 수락되었다고 본다.

본 연구에서는 PEACE parsing기법을 도입하여 구문분석과 동시에 원소를 추출하고 속성을 계산하여, 전처리부에서 독립적으로 수행되는 원소추출, 속성계산을 한번에 처리할 수 있게 하며, 시스템이 동적 구조를 가질 수 있게 한다. parser에 속성문법(attributed grammar)을 도입하여, 정파싱거리, 위치정보와 같은 속성과 원소의 종단기호를 linked list의 형태로 보관하여, 패턴의 분류에 사용한다. 구문분석의 결과는 list로 보관하며, 각 list는 수락된 각 자소(초성, 수직중성, 수평중성, 종성)의 코드와 속성 list로 구성한다.

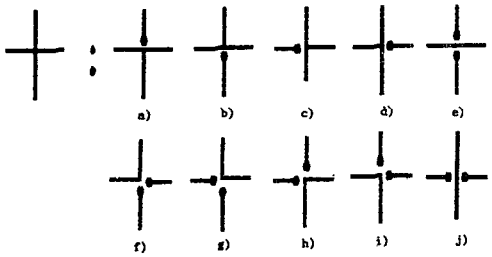
2. 원소변환

한글 인식에서 발생하게 되는 각 자소간의 접촉(contact)은 다른 자소의 패턴의 일부분을 침범하거나, 획의 확장으로 다른 패턴으로 인식하게 된다. 이를 해결하기 위하여서는 가능한 경우의 패턴을 모두 검증하여야 하며, 자소의 접촉부를 가능한 패턴의 원소로 변환하여야 한다. 각 자소의 접촉부의 패턴을 관찰하면, 접촉점은 주로 분기점, 교차점에서 발생함을 알 수 있다.

분기점은 그림 4와 같이, 하나의 끝점과 하나의 굴곡점, 또는 하나의 끝점과 원소로 변환할 수 있다. 교차점은 그림 5와 같이 하나의 끝점과 분기점, 또는 하나의 굴곡점과 끝점 들, 또는 두 개의 끝점과 원소, 또는 두개의 굴곡점으로 변환할 수 있으나, 한글의 경우 후자의 경우는 거의 발생하지 않는다.

a) 원소와 하나의 끝점, b), c) : 굴곡점과 하나의 끝점.

그림 4. 분기점에서의 원소 변환.



a), b), c), d) : 분기점과 하나의 끝점.

f), g), h), i) : 굴곡점과 두개의 끝점.

e), j) : 원소와 두개의 끝점.

그림 5. 교차점에서의 원소 변환

3. 패턴 분류

입력패턴에서 자소의 접촉이 있으면 구문분석시 원소 변환을 수행하여, 여러 후보를 발생시키게 된다. 이를 최종적으로 하나의 패턴으로 분류하기 위하여, 유사성을 계산하여 입력패턴과 가장 근접한 패턴으로 분류한다. 유사성의 측도는 구문분석시 계산된 속성값과 처리되지않은 화소의 수를 거리로 환산하여 사용하며, 각 자소의 구조적 관계를 묘사할 수 있는 서술논리를 도입하여 한글 자소의 조합 규칙, 한글 자소의 위치에 따른 heuristics를 이용한다.

처리되지 않은 화소의 수는 유사성 측도의 중요한 수단이 되며, 파싱에 참여하지 않은 화소의 수가 많다는 것은 입력패턴의 구문을 잘못 분석한 결과이므로, 이러한 후보패턴은 분류의 과정에서 제외시킨다.

한글의 조합에는 여러 가지 규칙이 존재한다. 중성의 경우, 양성모음과 음성모음의 조합은 허용되지 않으며, 구문 분석의 과정에서 이러한 조합은 배제하며, 종성의 복자음도 허용되는 조합을 구분할 수 있다. 이러한 조합규칙을 위배하는 후보 패턴은 분류의 과정에서 제외시킨다.

한글 패턴은 각 자소가 존재하는 위치에 따라, 각 획은 일정한 관계를 유지하게 된다. '고'자의 경우 'ㅗ'의 수직 성분은 초성 영역 아래에만 존재한다. 이와 같은 위치정보에 따른 heuristics를 이용하기 위하여 서술논리의 개념을 도입한다. 서술논리(predicate)는 진위를 판정하는 논리함수의 형태로 사용하며, 획이나 원소의 전후, 좌우, 상하, 내포등의 위치적 관계나, 파싱거리의 비교와 같은 의미정보(semantics)[10]를 검증할 때 사용한다.

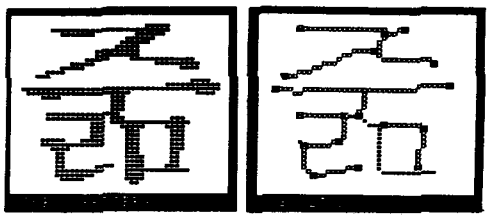
그러나, 이러한 heuristics는 글자체에 따라 다소 다르게 나타나기 때문에 적용에는 세심한 고려가 필요하다.

Ⅳ. 실험 및 결과 고찰

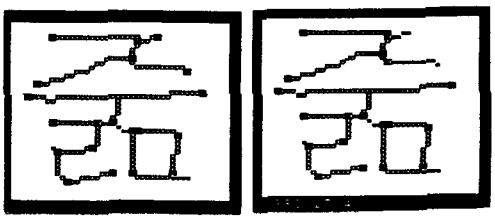
1. 구문분석 과정

본 연구에서는 입력 문자로부터 획을 추출하기 위하여, 세선화 알고리듬을 수행하여, 64*64 입력 문자패턴에서 폭이 1화소인 골격패턴을 추출한다.

그림 6은 간단한 패턴의 구조를 포함하고 있는 보다 복잡한 패턴의 구문분석 결과를 보인다. 패턴 '즒'의 초성 'ㅈ'은 'ㄱ'과 같은 패턴을 내포하며, 종성의 'ㅍ'도 'ㄱ'이나 'ㄴ'을 내포한다.



a) 입력 패턴. b) '즒'으로 인식.



c) '즒'으로 인식. d) '쿪'으로 인식.

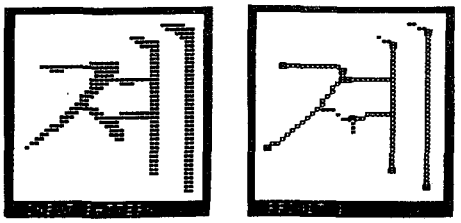
그림 6. 다른 패턴을 내재하는 패턴.

초성의 분기점에서 확장 가능한 구조는 'ㄱ', 'ㄲ', 'ㅈ' 이다. 종성 'ㅍ'의 경우, 'ㄱ'으로 인식하다 만나게 되는 분기점에서 원소를 변환하여 'ㅍ'의 구조까지 확장한다. 결과로 '즒'의 경우, 파싱에 참여하지 않은 화소의 수가 10이고, '쿪'의 경우 24, '즒'의 경우 24가 되어, '즒으로 분류된다.

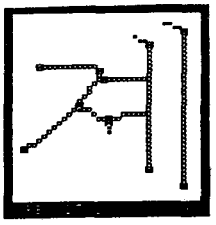
그림 7은 초성과 중성이 접촉된 경우이며, 구문 해석의 결과, '제' 또는 '계'로 인식된다. 초성에서 만나게 되는 두번째 분기점에서 자소의 접촉이라 생각하고 수직원소와 끝점으로 원소변환하여 'ㄱ'의 구조를 탐색하여 '계'라고 인식하고, 분기점을 회복하여 가능한 구조인 'ㅈ'의 구조로 해석하여 '제'의 구문을 검증한다. '제'의 경우 거리가 10이고, '계'의 경우는 15가 되어, 입력 패턴은 '제'로 분류된다.

그림 8은 중성과 종성이 접촉된 경우이며, 중성의 구문분석시 만나는 접촉점에서 먼저 'ㅜ'라는 구문을 분석한다. 수평모음에서 아래 쪽의 수직 획('ㅜ' 또는 'ㅠ')이 존재할 경우, 윗 쪽의 수직 획은 더 이상 탐색하지 않아 '루'라고 해석한다. 다음 분기점을 회복하고, 종성의 접촉이라 생각하여 수평원소와 끝점으로 원소를 변환하여, '른'의 구문을 인식한다. 거리 계산의

결과, 후보 패턴 '루'와 '룬'중에서 '룬'으로 인식하게 된다.

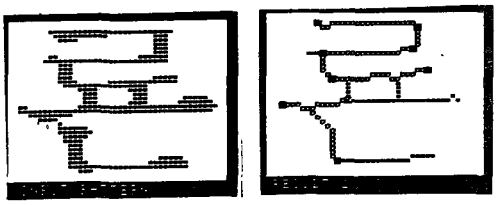


a) 입력 패턴. b) '계'로 인식.

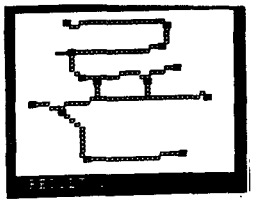


c) '계'로 인식.

그림 7. 초성과 중성의 접촉이 있는 패턴.



a) 입력 패턴. b) '루'로 인식.



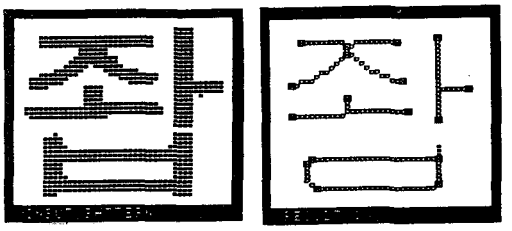
c) '룬'으로 인식.

그림 8. 중성과 종성의 접촉이 있는 패턴.

2 오인식의 고찰

오인식은 주로 세선화 과정에서의 오류와 정의하기가 힘든 자소의 접촉, 또는 획의 형태의 자소의 접촉에 의하여 발생한다.

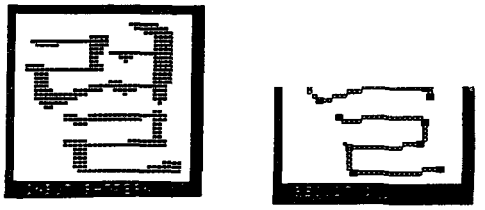
그림 9는 세선화 과정에서의 오류에 의한 오인식으로 종성 'ㅂ'의 꼭지 부분의 두께가 길이에 비해 작아 획이 없어져 버려, '잡'이 '잠'으로 인식된다.



a) 입력 패턴. b) '잠'으로 인식.

그림 9. 세선화에서의 획의 손실로 인한 오인식.

본 연구에서의 자소의 접촉의 형태는 주로, 분기점이나 교차점을 중심으로 고찰하였다. 그러나, 그림 10처럼 분기점이나, 교차점으로 나타나지 않고, 획의 형태로 접촉되는 경우, 획이 침범당하게 되어 오인식을 하게된다. 이러한 경우는 수직중성과 종성의 'ㅂ' 이나 'ㅅ', 또는 초성의 'ㄴ'이나 'ㄹ'과 같이 마지막 획의 성분이 수평 원소이고, 'ㅓ'나 'ㅕ'와 같은 수평성분을 포함하고 있는 수직중성에서 발생하며, 입력패턴 '럴'은 '럳'로 오인식 된다.



a) 입력 패턴. b) '럳'로 인식.

그림 10. 획의 형태의 접촉에 의한 오인식.

3 인식률 고찰

입력 패턴은 조합 가능한 문자 14364자 중 실제 사용에 나타날 수 있는 패턴1980자를 사식으로 인쇄한 글자체중 11 point의 크기의 패턴을 image scanner로 입력한다. 인식률은 명조체 92.0%, 고딕체 96.2%를 나타내었다.

'ㄱ'의 경우, '갈'의 종성 'ㄹ'의 장식이 'ㅈ'의 획으로 인식 되며, 'ㄴ'이나 'ㄷ', 'ㄹ'과 같이 수평 획을 포함하는 경우, 수직모음('ㅓ'또는 'ㅕ')의 수평성분을 침범하게 된다. 'ㅎ'의 경우 머리부분과 'ㅇ'성분의 접촉은 정의하기가 어려운 형태가 되어, 오인식이 많아지게 된다. 'ㅊ'이나 'ㅋ'은 세선화의 영향으로 획이 손실되어, 변형이 많고, 'ㅂ'이나 'ㅍ'은 복잡한 형태의 접촉을 발생시켜, 오인식을 하게 된다.

중성의 오인식의 주된 원인은 'ㅜ'나 'ㅠ'와 같은 아랫쪽의 수직 성분을 포함하는 수평모음의 수직 획이나, 수직모음의 수직성분이 'ㅂ'이나, 'ㅅ'과 같은 종성의 수직성분을 침범하여 오인식을 하게 된다.

한글의 구조는 초성, 수평모음, 수직모음, 종성의 조합의 형태에 따라 6형식으로 분류하며[11], 패턴의 복잡한 정도를 의미한다. 1, 2형식은 가장 간단한 형태로 초성과 수평 모음, 수직모음의 조합만으로 이루어 지는 문자이고, 3형식은 한 문자내에 수평모음과 수직모음이 같이 존재하는 패턴이다. 4,5,6형식은 1,2,3 형식에 종성이 존재하는 경우로, 6형식은 초성, 수평모음, 수직모음, 종성이 한 문자내에 모두 존재하는, 가장 복잡한 형태의 패턴이다.

표 1에 형식별 인식률을 나타내었다.

표 1. 형식별 인식 결과.

대상자소	고딕체			명조체		
	실험자소	인식자소	인식율(%)	실험자소	인식자소	인식율(%)
F1	85	84	98.6	85	83	97.6
F2	138	134	98.8	138	135	97.8
F3	98	94	95.9	98	92	93.9
F4	480	466	97.0	447	420	93.9
F5	930	902	96.9	911	842	92.4
F6	247	221	89.4	247	200	81.0
계	1978	1903	96.2	1926	1772	92.0

패턴이 비교적 간단한 1,2,3 형식이 복잡한 4,5,6 형식보다 인식률이 높으며, 특히 종성의 수직성분이 많이 존재하는 6형식의 경우 인식률이 많이 떨어진다. 전반적으로 자소의 접촉이 덜한 고딕체가 명조체보다 나은 인식률을 기록 하였으며, 문자의 조합이 복잡할수록 인식률은 떨어짐을 보이지만, 이는 자소의 접촉에서 정의되지 않은 접촉점, 즉 분기점이나 교차점으로 접촉하지 않은 경우에서 대부분 발생함을 알 수 있다. 따라서, 두께 정보를 이용하여, 획의 형태로 접촉하는 자소의 분리를 수행할 수 있는 세선화가 필요하다.

또한 명조체의 경우, 장식이 주요한 오인식의 원인이 되며, 획의 길이를 이용하여 장식임을 판단할 수 있는 heuristics를 고려할 수 있어야 한다.

V 결 론

본 연구에서는 자소의 접촉과, 한 패턴이 다른 패턴의 구조를 내포하는 데에서 기인하는 한글 인식의 오인식 문제를 해결하기 위하여 원소변환과 PEACE parsing을 이용한 한글인식 방법을 제안하였다.

접촉점에서 원소변환을 수행하여, 가능한 구조를 모두 검증하여, 한글 패턴에서 발생하는 자소의 접촉 문제를 해결하였고, PEACE 파싱 개념을 도입하여 구문분석과 동시에 원소를 추출하고, 속성을 계산함으로써, 종래에는 독립적으로 수행되던 원소추출과 속성계산 과정을 통합하여, 1차원 스트링으로의 변환에서 발생하는 패턴의 왜곡, 정보의 손실을 방지하고, 시스템이 동적인 구조(dynamic structure)를 갖도록 하였다.

오인식은 세선화 과정에서 발생하는 획의 손실, 규정하기 어려운 접촉점의 발생, 직선의 형태의 자소의 접촉 때문이었으며, 이를 보완하기 위하여서는 획의 두께 정보를 이용하여, 하나의 획처럼 표현될 수 있는 자소간의 접촉을 분리할 수 있는 세선화 처리가 필요하며, 현재까지 진행된 결과와 한글 패턴에서 관찰되는 의미정보(semantics)를 더 많이 고려할 수 있어야 한다.

또한, 생성규칙의 적용과 패턴 분류에 확률의 개념을 도입하고, 인식의 결과를 축적하여, 학습의 능력을 가질 수 있는 learning automata에 관한 연구가 필요하다.

참고 문헌

1. 이 주근, " 한글 문자의 인식에 관한 연구 (IV), " 대한전자 공학회 논문지, 제 9 권 제 4 호, 1972, pp. 197-204.

2. K. S. Fu, Syntactic Pattern Recognition and Appli., Prentice -Hall, 1982.

3. T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, 1977.

4. T. Agui, M. Nakajima, T. K. Kim, and E. T. Takahashi, " A Method of Recognition and Representsation of Korean Characters by Tree Grammars, " IEEE Trans. Pattern Analy. Machine Intell. Vol. PAMI-1, No. 3, 1979, pp. 245-251.

5. K. H. Lee, K. B. Eom and R. L. Kashyap, " Character Recognition using Attributed Grammar," IEEE Cmputer Soceity Conf. on Computer Vision and Pattern Recognition, June, Ann Arbor, 1988 , pp.418-423.

6. R. A. Kirsch, " Computer Interpretation of English Text and Picture Patterns," IEEE Trans. Elec. Comput. Vol.EC-14, 1964, pp.363-376.

7. A. Rosenfeld, Picture Languages, Academic Press, 1979.

8. D. J. Rosenkrantz, "Programmed Grammars and Classes of Formal Languages," J. ACM. Vol.16, No.1, 1969, pp.107 - 131.

9. P. H. Swain and K. S. Fu,"Stochastic Programmed Grammars for Syntactic Pattern Recognition," Pattern Recogniton, Vol.4, 1972, pp. 83-100.

10. M. L. Baird and M. D. Kelly, " A Paradigm for Semantic Picture Recognition," Pattern Recogntion, Vol.6, 1974, pp.61-74.

11. 이 병래, 군일 비용 검색방식을 이용한 인쇄체 한글의 인식에 관한 연구, 연세대학교 대학원, 석사학위 논문, 1986, 12.