論文 89-26-3-11

속성문법을 이용한 필기체 한글 문서 내의 자모인식

(The Recognition of Vowels and Consonants in a Handwritten Hangul Text with Attributed Grammars)

柳 承 必*, 金 太 均**

(Sung Pil Lyu and Tae Kyun Kim)

要 約

글자의 간격과 크기가 일정하지 않으므로 전처리 과정에서 각 글자를 분리하기 어려운 필기체 한글 문서로 부터 자모들을 인식하는 방법을 제안한다.

본 방법은 세선화된 필기체 한글문서의 영상 내에 있는 모든 글자들을 스트로크들로 변환시키고, 이들 사이의 배열관계를 나타내는 속성을 추출한 다음, 이들 스트로크와 속성들에 대해 속성문법을 적용하여 자모들을 인식한다.

Abstract

This paper proposes a method to recognize vowels and consonants in a handwritten Hangul text, in which the sizes of chracters and the spaces between characters are not uniform.

In this method, all characters in the thinned image of a handwritten Hangul text are transformed into strokes, and the attributes which represent the relations between strokes are extracted from these strokes. The vowels and consonants are recognized by applying attributed grammars to the strokes and attributes.

I. 서 론

카메라, 핵시믿 등을 이용하여 서류등에 쓰여진 한글 영상을 컴퓨터에 입력시킨 후, 이들 글자들을 인식하는 방법을 정보의 입력수단으로 이용하기위해 많은 연구가 계속되어 오고 있다. 지금까지 많은 논문들이 주로 한 글자를 단위로 많은 연구[1,2,3]가 있어 왔으나, 한글 인식에 의한 다량의 문자정보를 처리하기 위해서는 한글 문서인식에 관한 연구가 필수적이라 할 수 있다.

지금까지 연구 되어온, 한 글자를 단독으로 인식하는 알고리듬을 이용하여 한글문서를 인식하려면, 인식 이전에 문서내의 글자들을 서로 분리해야 한다. 인쇄체 문서의 경우 글자 및 줄들 사이의 간격과 글자의 크기가 거의 일정하므로 필기체에 비해 글자들을 분리하기가 용이하므로, 각 글자를 분리하여 인식하는 연구가 현재 진행되고 있으나[4], 글자들 및 줄들 사이의 간격, 그리고 글자의 크기가 일정하지 않

*正會員, 韓國에너지研究所 計測制御研究室
(Dept. of I&C., Korea Advanced Energy Research Institute)

**正會員, 忠南大學校 電子計算機工學科
(Dept. of Computer Eng., Chungnam Univ.)

接受日字 : 1987年 3月 20日

(※본 연구는 1986년도 전반기 한국과학재단 차관 연구비 지원에 의해 수행된 연구입니다.)

은 필기체 문서의 경우, 아직 각 글자를 분리하여 인식하는 연구는 미미하다.

한글은 직선성분을 많이 포함한 스트로크(stroke)들로 구성되어 있고, 같은 줄 내의 글자들을 구성하는 스트로크들은 상대적으로 어느 정도의 범위 내에 존재하고 있다. 이 논문에서는 이를 이용하여, 전처리과정에서 글자들이 각각 분리되지 않은 필기체 한글 문서내의 글자를 인식하는 방법을 다음과 같이 제안한다.

1) 이차원적으로 배열되어 있는 문자들로 부터 스트로크들을 줄 단위로 추출하여 입력패턴을 일차원화 한다.

2) 위에서 추출된 스트로크들은 주로 직선성분으로 되어 있으므로, 같은 모양을 가진 스트로크들이 많이 존재하게 된다. 그러나 같은 종류의 스트로크들도 배열되는 형태에 따라 여러가지 자모가 될 수 있으므로 이들 배열관계는 인식에 중요한 정보가 된다. 따라서 이들 배열관계를 여러가지 속성으로 정의하고, 과정 1)에서 구해진 스트로크들 사이에 해당되는 속성들을 추출한다.

3) 속성문법을 이용하여 위에서 구해진 스트로크들과 속성들을 조합하면서 이들로 부터 자모를 추출해 낸다.

필기체는 쓰는 사람의 습관에 따라 자모가 여러가지 형태로 변형이 되므로 모든 경우의 글자를 인식하는 것은 어렵다. 현재까지 발표된 한 글자를 대상으로 하는 대부분의 인식법도 인쇄체나, 정서된 글자에 국한되어 있다. 특히 흘림체의 경우 자모 또는 글자와 글자가 서로 연결되어 이들을 분리하기가 매우 어려워지므로 이를 인식하기 위해서는 앞으로 많은 연구가 필요하다. 이 논문은 정서된 필기체문서를 대상으로, 글자의 평균높이가 대략 32가 되게 컴퓨터에 입력시킨 다음, 이를 세선화한 입력패턴에 대해 다음과 같은 조건을 만족하는 경우에 적용한다.

1) 같은 줄 내에서 이웃하는 글자들은 그 글자들을 포함하는 최소면적의 직사각형의 중심의 높이 차이가 16(글자의 세로 평균 길이의 절반)이내이다.

2) 같은 줄 내에 있는 각 글자의 최상위점의 높이 차이는 32 이내이다.

3) 상위에 있는 줄의 각 글자의 최상위점은, 하위에 있는 줄의 각 글자를 구성하는 어떤 점보다 상위에 있다.

조건 1)의 경우는 10여명의 필기자들로 부터 얻은 결과에 의해, 거의 모든 글자들이 만족하였고, 2), 3)의 경우는 줄이 기울어진 정도를 나타내는데, 필기된 문서들 자체는 거의 이 조건들을 만족을 하나, 입력시 문서자체가 기울어질 수도 있으므로, 이 때에 이들 조건을 만족하지 않는 경우가 생긴다. 이 논문에서는 이 경우 사람이 수동으로 문서를 바로잡는 것으로 하였다.

Ⅱ. 입력패턴의 스트로크열화

1. 특징점과 스트로크끝점

스트로크는 한개 이상의 특징점과 기본패턴을 포함한 연결된 선으로 되어 있다. 스트로크내에 포함되는 특징점의 종류는 그림 1과 같다.

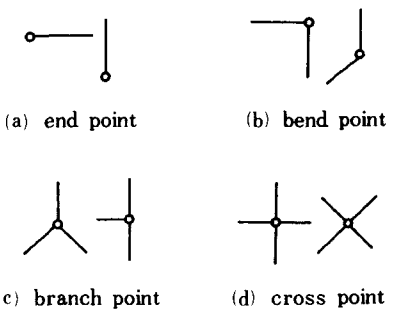


그림 1. 특징점
Fig. 1. Feature points.

만약 위와 같은 특징점을 만날 때마다 스트로크를 만들면, 자모들의 접촉 상태에 따라 많은 여러가지 스트로크 종류가 발생하며, 이에 따라 패턴을 표현하기 위한 문법의 수가 많아지고, 인식도 복잡해 진다. 따라서 이 논문에서는 불필요하게 스트로크들이 분리되는 것을 줄이기 위해서

1) 단점을 만나거나, 시작점과 끝점이 연결된 원 또는 네모의 경우 끝점을 만날 때,

2) 시작점이 단점인 선이 분기점을 만날을 때, 이 분기점까지의 직선의 방향이 분기점 이후로 연결되는 직선과 45도 이상의 각도를 이룰 때, (여기서 직선은 다음 절에 설명할 기본패턴을 이루는 선과 같다)

3) 분기점이나 굴곡점을 한개 이상 포함한 패턴이 교차점을 만날 때,

스트로크를 분리하도록 하고, 이때 이 특징점들을 스트로크 끝점이라 한다. 2)의 경우 시작점이 단점이 아닌 경우는 시작점이 원 또는 네모의 한 점이 되므로 분기점이나 교차점을 만나도 이를 무시하고 계속해서 1)과 같은 스트로크 끝점을 찾는다.

그림 2. 스트로크 끝점
Fig. 2. End points of stroke.

2. 스트로크내의 굴곡점 추출

굴곡점 외에는 참고문헌[3]의 방법으로 간단하게 추출되므로 여기서는 생략한다. 이 논문에서는 스트로크를 추출하는 과정에서 스트로크 내부의 굴곡점들을 순차적으로 추출하고, 이들을 이용하여 다음에 올 특징점이 스트로크 끝점인지 아닌지 결정한다. 만약 스트로크 내부 점들의 집합을 $\{P\}s$, 점 P_i와 P_{i+1} 사이의 진행방향을 C_i라할 때 이들 진행방향의 집합을 $\{C\}s$, C_i와 C_{i+1} 사이의 방향변화를 D_i라할 때 이들 집합을 $\{D\}s$라 하면,

$\{P\}s = \{P_1, P_2, \cdots, P_n\}$
$\{C\}s = \{C_1, C_2, \cdots, C_{n-1}\}$
$C_i \in \{0, 1, \cdots, 7\}$
$\{D\}s = \{D_2, D_3, \cdots, D_i, \cdots, D_{n-1}\}$
n; 스트로크를 구성하는 흑점의 총 수

여기서,

$$D_i = \begin{cases} 0; & C_i = C_{i-1}\text{일 때} \\ 1; & 0 < C_i - C_{i-1} \leqq 4 \text{ 또는} \\ & -7 \leqq C_i - C_{i-1} \leqq -4\text{일 때} \\ -1; & -4 < C_i - C_{i-1} < 0 \text{ 또는} \\ & 4 < C_i - C_{i-1} \leqq 7\text{일 때} \end{cases}$$

단, $2 \leqq i \leqq n-1$

이다.

그리고 스트로크 내의 굴곡점의 집합을 $\{Pb\}s$, 시작점 P1, 굴곡점들 그리고 스트로크 끝점 Pn 사이의 선들 집합을 $\{L\}s$, 각 선들의 첫 점에서 끝점으로 향하는 방향의 각도들의 집합을 $\{T\}s$라 하면,

$\{P_b\}s = \{P_{b1}, P_{b2}, \cdots, P_{bi}, \cdots, P_{bn_s}\}$
$\{L\}s = \{L_1, L_2, \cdots, L_i, \cdots, L_{n_s+1}\}$
$\{T\}s = \{T_1, T_2, \cdots, T_i, \cdots, T_{n_s+1}\}$
$\{P\}_{Li} = \{P_{i1}, P_{i2}, \cdots, P_{ini}\}$

여기서,

$P_{bi}, P_{ik} \in \{P\}s$
$0 \leqq Ti < 360$, Ti는 실수
n_s; 한 스트로크내의 굴곡점수
n_i; Li를 구성하는 흑점의 총 수

이다. 시작점(P_1)로 부터 한 점씩 진행하면서 각 점에 대해 진행방향과 변화방향을 구하고, 굴곡점은 이들을 이용하여 다음과 같은 방법으로 추출된다. (그림 3. 참조)

1) $\{P\}_{L1} = \{\ \}$라 둔다.

2) $k > 3$이고 $D_{K1} < \ > 0$을 만족하는 최초의 P_{K1} 또는 스트로크 끝점인 P_{K1}를 찾은 다음

$\{P\}_{L1} = \{P_{11}, \cdots, P_{n_1 1}\} = \{P_1, \cdots, P_{K1}\}$

로 한다. 여기서, $n_1 = k_1$이고, $P_j \in \{P\}s\ (1 \leqq j \leqq k_1)$이다.

로 한다.

3) 만약 P_{K1}이 스트로크 끝점이면 진행을 중지하고, 아니면 i=2로 한다. 여기서 i는 스트로크내에서 추출된(또는 앞으로 추출될) 선의 수를 의미한다.

4) $\{P\}_{Li} = \{\ \}$라 둔다.

5) $D_{K1} \neq 0$이고 $3 \leqq k_i - k_{i-1}$을 만족하는 최초의 점 또는 스트로크 끝점인 P를 찾은 다음,

$\{P\}_{Li} = \{P_{i1}, \cdots, P_{n_i i}\} = \{P_{K i-1}, \cdots, P_{Ki}\}$

로 한다. 여기서, $n_i = k_i - k_{i-1} + 1$, $P_j \in \{P\}s\ (k_{i-1} \leqq j \leqq k)$이고, $P_{K i-1}$은 L_{i-1}의 마지막 점인 동시에 L_i의 첫점이며, k_{i-1}부터 k_i까지는 연속된 정수이다.

6) T_{i-1}과 T_i의 각도차이를 t라 할 때, 만약 $t > 45$이고 $k_i - k_{i-1} \geqq 3$이면

$P_{bi-1} = P_{K i-1}$; (굴곡점)
$i = i + 1$

로 하고 아니면,

$\{P\}_{Li-1} = \{P\}_{Li-1} \cup \{P\}_{Li}$

로 한다.

7) 마지막점이 스트로크 끝점이 아니면 과정 4)를 계속한다.

위의 과정이 모두 끝나면 하나의 스트로크가 추출된다.

3. 기본패턴(primitive)과 부분패턴(subpattern)

기본패턴은 원과 8방향(그림 3 a)으로 양자화 한 코드로 구성되며, 부분패턴은 이들 기본패턴이 하나이상 모여서 이루어지는 패턴을 말한다.

원의 경우, 입력패턴이 양자화 및 세선화를 거치면, 방향성분이 다른 여러개의 작은 직선이 생기고, 이들 직선은 대부분이 같은 쪽으로 방향변화가 일어난다. 네모의 경우는 주로 굴곡점 근처에서 방향변화가 일어나므로 방향변화(D) 합의 절대값이 각 모서리 근처에서 1씩 증가하여 4이하가 되는데, 원의

경우는 5이상이 된다. 따라서,

$$\left|\sum_{i=2}^{n-1} Di\right| \geq 5 \quad (n;\text{스트로크내 흑점의 총 수})$$

이면 그 스트로크를 원으로 간주한다.

만약 어떤 스트로크가 원이 아니라면 시작점, 굴곡점들 및 스트로크 끝점 사이의 선분(L_i, i=1, 2, ··)의 첫점으로 부터 끝점으로 향하는 방향(T_i, i=1, 2, ···)을 8방향으로 양자화한 코드가 된다.

일반적으로 양자화 화면의 특성 및 세선화 과정에서 발생하는 찌그러짐에 의하여 길이가 긴 선내에는 불필요한 방향성분을 가진 작은 선들이 다수 포함되어 있다. 특히 굴곡점 근처에서 이러한 작은 선들이 존재하는 경우가 많은데, 만약 이런 성분들이 모두 기본패턴으로 된다면 이를 위한 문법설계가 복잡해지고 이에 따라 인식도 어려워지게 된다. 따라서 이 논문에서는 이런 불필요한 성분들이 기본패턴이 되는 것을 가급적 피하기 위해서, 2개이상의 기본패턴이 모여 스트로크가 되는 경우, 위의 과정 1)과 7)에서 처럼 기본패턴이 되는 선의 길이를 최소 3이상으로 하였다.

그림 3(b)는 두개의 스트로크로 부터 기본패턴을 추출하는 예를 보인다. 첫번째 스트로크는 굴곡점 하나, 선이 두개 생기고, 이 선들의 각도를 각각 양자화하면 그림 3(a)에 의해서 두개의 기본패턴 0과 6이 생긴다.

또 두번째의 경우 $\left|\sum_{i=2}^{20} D_i\right| = 7$ 이므로, 원으로 간주되어 이 스트로크는 기본패턴 8만 존재한다.

4. 박스(box)와 블록(block)

[정의 1] 각 변이 X, Y축에 평행하는 임의의 직사각형을 B 또는 B(Xl, Yu, Xr, Yd)라 하고, 어떤 패턴 A가 있을 때, 이를 포함하는 최소 면적의 직사각형을 B_A 또는 B_A(XL, YU, XR, YD)라 표현하고, 이를 박스라고 한다. 여기서,

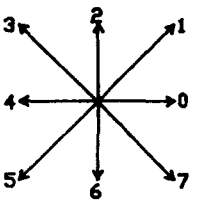
(Xl, Yu), (XL, YU) ; 직사각형 및 박스의 최좌측 상단점들

(Xr, Yd), (XR, YD) ; 직사각형 및 박스의 최우측 하단점들

[정의 2] 박스중에서 Y축에 평행한 변이 어떤 스트로크도 절단하지 않을 때 이 박스를 블럭이라 한다.

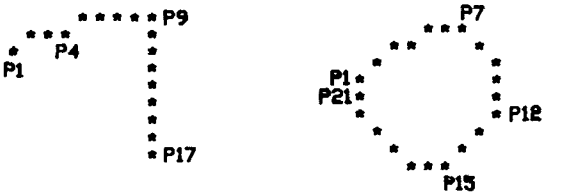
5. 입력패턴으로 부터 스트로크들의 추출

만약 입력패턴이 서론에서 기술한 조건을 만족한다면, 입력패턴으로 부터 블럭들과 블럭들 내의 스트로크들을 추출할 수 있다. 다음은 임의의 k번째 블럭내의 스트로크들을 추출하는 방법이다(그림 5.참조).



8 ; circle

(a) Primitives (0—8)



{P}s = {P_1, ···, P_{17}}
{C}s = {1, 0, 0, 1, 0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6}
{D}s = {−1, 0, 1, −1, 0, 0, 0, −1, 0, 0, 0, 0, 0, 0, 0, }
{P_o}s = {P_9}
{T}s = {$\tan^{-1}(2/8)$, $\tan^{-1}(-8/0)$}
{R}s = {0, 6}

{P}s = {P_1, ···, P_{11}}
{C}s = {1, 1, 0, 1, 0, 0, 7, 7, 6, 6, 6, 5, 5, 5, 5, 4, 4, 3, 3, 3, 2}
{D}s = {0, −1, 1, −1, 0, −1, 0, −1, 0 0, −1, 0, 0, −1, 0, −1, 0, −1}
{P_o}s = {P_7, P_{12}, P_{15}}
{T}s = {$\tan^{-1}(3/6)$, $\tan^{-1}(-2/5)$, $\tan^{-1}(-3/(-3))$, $\tan^{-1}(4/(-5))$}
{R}s = {8}

{R}s ; set of primitives for a stroke

(b) bend points and primitives extracted from strokes

그림 3. 기본패턴 추출의예
Fig. 3. Examples of primitives extracted from strokes.

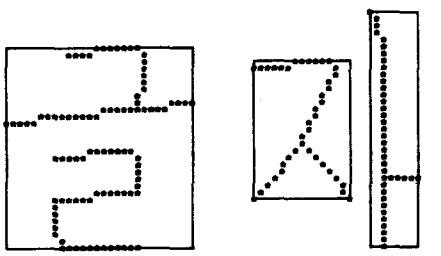


그림 4. 블럭
Fig. 4. Block.

1) 입력패턴으로 부터 최상단에 있는 흑점중에 최좌측점(X_{hk}, Y_{hk})을 찾는다.

2) 입력패턴내에 흑점이 존재하면 k번째 블럭 $B_{bk}(Xl_k, Yu_k, Xr_k, Yd_k)$를 초기화하고, 흑점이 존재하지 않으면 모든 과정을 끝낸다.

3) 직사각형 $B_{hk}(Xc, Y_{hk}, X_{hk}, Y_{hk}+31)$을 구성하고, B_{hk}내의 최좌측에 있는 점 중에서 최상단에 있는 흑점 Ps를 찾는다. 여기서,

$$Xc=\begin{cases} Xr_{k-1}+1 ; i>1\text{이고 } Xl_k<X_{hk}\text{일 때} \\ 1 \quad ; \text{그외} \end{cases}$$

Xr_{k-1} ; k−1번째 블럭의 최우측 X좌표

Xl_{k-1} ; k−1번째 블럭의 최좌측 X좌표

i ; k번째 블럭내에서 추출된 스트로크 수

4) Ps_i와 연결된 단점들을 비교하여 가장 좌상위에 있는 점(X좌표와 Y의 좌표의 합을 구해서 가장 작은 값을 가지는 점)을 찾는다. 이 점이 스트로크의 시작점이 된다.

5) 이 점으로 부터 스트로크 끝점을 만날때 까지 스트로크 S_i를 찾는다. 스트로크를 구성하는 흑점은 스트로크 추출중에 입력패턴으로 부터 하나씩 제거된다.

6) 스트로크 S_i를 포함하는 박스 Bs_i를 구한다. 그리고 Bb_k와 Bs_i를 포함하는 직사각형 B'_{bk}를 만든다음, B'_{bk}의 중심이 B_{bk-1}의 중심보다 16이상 아래에 위치하고 $B'_{bk}\neq Bs$이면, 스트로크 S_i를 k번째 블럭에서 제외시킨다. 만약 중심의 위치 차이가 16이하이면 $Bb_k(Xl_k, Yu_k, Xr_k, Yd_k)=B'_{bk}$로 한다.

7) 직사각형 $Bu(Xl_{k-1}+1, Y_{hk}, Xr_k, Yd_k)$ 내에서, 이 직사각형 내에 흑점이 없으면, $Ye=MAX(Yd_k, Yu_k+31)$라 할 때, $Bd(Xl_{k-1}+1, Yd_k, Xr_k, Ye)$내에서, 최좌측에 있는 흑점중 최상단점을 찾는다. 만약 있으면 i를 하나 증가(i=i+1)시키고, 이점을 Psi로 한다음, 과정 4)를 수행한다.

8) Bu와 Bd내에 흑점이 없으면, 과정 6)에서 발생된, 블럭내에서 제외될 스트로크가 있으면 이들을 흑점으로 복원한다. 이상으로 k번째 블럭내의 스트로크 추출을 모두 끝낸다.

그림 5.는 블럭을 추출하는 예를 보인다. 입력패턴의 최상위좌측점(X_{h1}, Y_{h1})을 구하고, 과정 3)으로 부터 B_{h1}을 구한다. 이 박스내의 최좌측에 있는 최상단점은 Ps_1이 되고 이점으로 부터 스트로크 '_'를 구할 수 있다. 이 스트로크를 포함하는 박스를 중심으로 Bu를 구성하면 Ps_2를 구할 수 있고 이로부터 스트로크 'ㄱ'을 구할 수 있다. 스트로크 '_'와 'ㄱ'을 포함하는 박스를 기준으로 Bu를 구성하면 이 박스내에 흑점이 존재하지 않으므로 Bd를 형성하여 Ps_3를 찾을 수 있다. 이로부터 스트로크 'ㄹ'을 찾을 수 있고, 더 이상 Bu와 Bd내에 흑점이 없으므로 첫번째 블럭의 추출이 끝난다. 그리고 다시 과정 1)을 수행하면 최상단좌측점 $(X_{h1}, Y_{h1})=(X_{h2}, Y_{h2})$을 구할 수 있고, 이로부터 점 Ps_4를 찾아내어 스트로크 'ㄱ'을 구할 수 있고, 계속해서 Ps_5를 찾고 스트로크 '\'을 구하여, 이로부터 Bd를 형성하면 아래줄 글자 '의'스트로크 중 'ㅣ'의 일부가 포함된다. 만약 'ㅣ'를 'ㅈ'의 일부 스트로크로 간주하면 'ㄱ', '\', 'ㅣ' 모두를 포함하는 박스의 중심이 '글'의 블럭 중심보다 16보다 더 아래에 위치하게 되므로 과정 6)에 의해서 스트로크 'ㅣ'는 두번째 블럭에서 제외된다. 이와같이하여 입력패턴내의 흑점이 모두 없어질 때까지 모든 블럭내의 스트로크를 추출하면, 이로서 입력패턴내의 모든 스트로크의 추출이 끝난다. 과정 3)에서 $Xl_{k-1}>X_{hk}$이면 입력패턴 조건 3)에 의해 (X_{hk}, Y_{hk})는 다음 줄 글자의 최상위점이 되므로 이 점 이후로 다음 줄의 글자로 부터 스트로크를 추출하게 되므로 스트로크들을 줄단위로 분리할 수 있다.

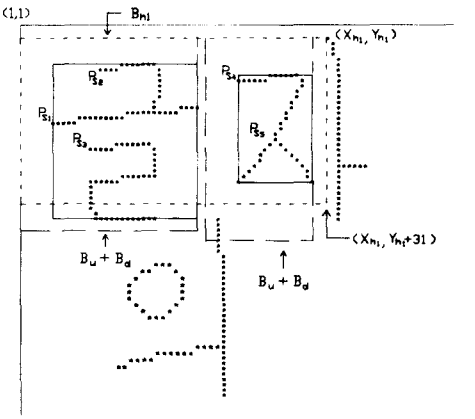


그림 5. 입력패턴으로 부터 블럭 추출
Fig. 5. Extraction of blocks from input pattern.

6. 블럭내의 스트로크의 Sorting

5절의 방법으로 스트로크를 추출하면 '논'의 경우 스트로크열은 '_', 'ㄴ', 'ㅣ', 'ㄴ'순으로 추출된다. String 문법의 경우 이들을 순차적으로 받아들여 인식하므로 위와같이 배열된 스트로크순으로 문자를 인식하는 것은 어렵다. 따라서 이들 스트로크들을 가능한 쓰여지는 순서대로 배열할 필요가 있다. 한글

의 자모들은 좌측 또는 상단순으로 쓰여지므로 다음에 정의하는 상위, 좌위, 그리고 좌상위(그림 6. 참조)우선으로 스트로크를 재배열한다.

임의의 두박스 $B_1(Xl_1, Yu_1, Xr_1, Yd_1)$, $B_2(Xl_2, Yu_2, Xr_2, Yd_2)$가 있다고 할 때,

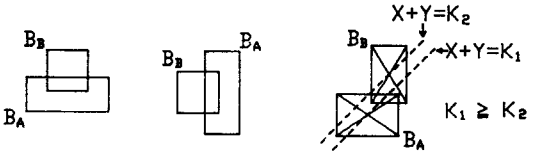
[정의 3] $Xr_2-Xl_2 \leqq Xr_1-Xl_1$이고, $Xl_1 \leqq Xl_2 \leqq Xr_2 \leqq Xr_1$이거나, $Xr_2-Xl_2 \geqq Xr_1-Xl_1$이고, $Xl_2 \leqq Xl_1 \leqq Xr_1 \leqq Xr_2$이면 두박스는 상하관계라 하고, 상하관계의 B_1, B_2가 $Yd_2 \leqq Yd_1$이고, $Yu_2 < Yu_1$이면 B_2는 B_1의 상위에 있고, $Yd_1 \leqq Yd_2$이고, $Yu_1 < Yu_2$이면 B_2는 B_1의 하위에 있다고 한다.

[정의 4] $Yd_2-Yu_2 \leqq Yd_1-Yu_1$이고, $Yu_1 \leqq Yu_2 \leqq Yd_2 \leqq Yd_1$이거나 $Yd_2-Yu_2 \geqq Yd_1-Yu_1$이고, $Yu_2 \leqq Yu_1 \leqq Yd_1 \leqq Yd_2$이면 두 박스는 좌우관계라 하고, 좌우관계의 B_1, B_2가 $Xr_2 \leqq Xr_1$이고, $Xl_2 < Xl_1$이면 B는 B의 좌위에 있고, $Xr_1 \leqq Xr_2$이고, $Xl_1 < Xl_2$이면 B_2는 B_1의 우위에 있다고 한다.

[정의 5] $(Xl_1+Yu_1+Xr_1+Yd_1)/2 > (Xl_2+Yu_2+Xr_2+Yd_2)/2$이거나, $(Xl_1+Yu_1+Xr_1+Yd_1)/2 = (Xl_2+Yu_2+Xr_2+Yd_2)/2$이고, $(Xl_1+Xr_1) > (Xl_2+Xr_2)$이면 B_2은 B_1의 좌상위에 있다고 한다.

앞의 예 '논'의 경우 4개의 스트로크들을 상위, 좌위 및 좌상위 순으로 sorting들을 하면 'ㄴ', 'ㅣ', 'ㅡ', 'ㄴ'순으로 된다.

입력패턴으로 부터 블럭단위로 스트로크들을 추출 및 sorting을 하고, sorting이 된 스트로크들을 블럭 순서대로 나열하면, 이것으로 입력패턴으로 부터 스트로크열로의 변환이 끝난다.



(a) up (b) left (c) left-up

그림 6. 박스 B_A에 대한 박스 B_B의 위치
Fig. 6. The relative position of box B_B to box B_A.

Ⅲ. 속성 및 속성생성 함수

이 논문에서는 사용하는 속성들은 스트로크와 스트로크 또는 패턴과 패턴간의 방향, 위치, 접촉, 포함 및 거리관계를 나타내는 5가지 종류의 속성과 이미 인식된 패턴의 정보를 이용하여, 다른 패턴을 인식하는데 이용되는 속성 1가지를 합해 모두 6가지로 되어있다.

앞장에서 서술한 바와같이 입력패턴은 스트로크열로 변환되고, 스트로크열과 이들 사이의 속성들이 인식과정에서 함께 조합되어 부분패턴의 열이 되며, 다시 이들 부분패턴들은 그들의 속성과 함께 조합되어 보다 복잡한 부분패턴열이 된다. 이렇게 부분패턴들이 보다 복잡한 부분패턴의 열로 변환되는 것이 반복되어, 최종적으로 자모의 열이 생성된다.

이들 부분패턴 사이의 위치관계와 부분패턴들이 갖고 있는 각각의 의미를 나타내는 속성들을 생성하는 함수는 다음과 같이 정의된다.

[정의 6] 두개의 패턴 X, Y가 있을 때, 이들의 Bx의 중심으로 부터 By의 중심을 직선으로 연결하는 방향을 8방향으로 양자화한 값을 나타내는 함수 f1은

$$f1(Bx, By) \in \{a0, a1, \cdots, a7\}$$

가 된다. 여기서,

a0, ⋯, a7 ; 8방향으로 양자화된 코드

[정의 7] 두개의 패턴 X, Y가 있을 때 이들의 좌우 또는 상하를 나타내는 함수 f2는

$$f2(Bx, By) = \begin{cases} b1 ; By가\ Bx대해\ 상위에\ 있을\ 때 \\ b2 ; By가\ Bx대해\ 하위에\ 있을\ 때 \\ b3 ; By가\ Bx대해\ 좌위에\ 있을\ 때 \\ b4 ; By가\ Bx대해\ 우위에\ 있을\ 때 \\ b0 ; 그\ 외 \end{cases}$$

가 된다.

[정의 8] 두개의 패턴 X, Y가 있을 때, 이들의 접촉관계를 나타내는 함수 f3는

$$f3(Bx, By) = \begin{cases} c1 ; 접촉 \\ c0 ; 그\ 외 \end{cases}$$

이며, 두 박스가 공유하는 영역이 있으면 접촉관계가 있다고 한다.

[정의 9] 두 패턴 X, Y가 있을 때, 이들의 포함관계를 나타내는 함수 f4는

$$f4(Bx, By) = \begin{cases} d1 ; 포함 \\ d0 ; 그\ 외 \end{cases}$$

로 되며, 하나의 박스가 다른 박스를 완전히 포함하면 포함관계에 있다고 한다.

[정의10] 두 패턴 X, Y가 있을 때, 두 박스 중심간의 거리가 d, 두 박스의 대각선 길이를 각각 $1_1, 1_2$라 할 때, $d \leqq (l_1+l_2)/2$이면 근접하다고 하고, 함수 f5는

$$f5(Bx, By) = \begin{cases} e1 ; d \leqq (l_1+l_2)/2 \\ e0 ; d > (l_1+l_2)/2 \end{cases}$$

이다.

[정의11] 어떤 패턴 X가 자음인지 모음인지를 판정하는 함수 f6은

$$f6(X) = \begin{cases} h1 ; X가\ 자음일\ 때 \\ h2 ; X가\ 모음일\ 때 \\ h0 ; 그\ 외 \end{cases}$$

이다.

Ⅳ. 속성문법

1. 속성문법 정의

다음은 문맥자유속성문법(context free attributed grammar)을 정의[5, 6]하고, 이들을 이용하여 한글 자모를 표현한 예를 보인다.

[정의12] 문맥자유속성문법 G는 4개의 항으로 G = {V_N, V_T, P, S}이다. 여기서,

V_N ; 비종단 기호의 유한집합
V_T ; 종단 기호의 유한집합
P = (Psy, Pse) ; 생성규칙의 유한집합으로 구문규칙(Psy)과 의미규칙(Pse)으로 구성
S ; 출발기호로 $S \in V_N$

이다. 비종단기호는 부분패턴을 나타내는 기호이고, 종단기호는 기본패턴을 나타내는 기호이다. 생성규칙중, 구문규칙은 부분패턴을 구성하는 패턴의 종류를 규정하는 것이고. 의미규칙은 이 부분패턴을 구성하는 패턴 사이에 존재하는 위치관계와 이 부분패턴을 구성하는 패턴 이외의 다른 부분패턴의 의미와 위치관계를 규정하는 규칙이다. 구문규칙의 형태는 $X \rightarrow X_1, X_2, \cdots, X_i, \cdots, X_m$으로, $X \in V_N$, $X_i \in V_N \cup V_T$ 이다. 의미규칙의 형태는 $A(X) = \{A(Xp), A(X_1), A(X_2), \cdots, A(X_i), \cdots, A(X_m)\}$으로 $A(X_i)$는 X_i와 X_{i+1} 사이에, A(Xp)는 패턴 X_1 이전에 처리된 부분패턴 Xp와 X_1 사이에, $A(X_m)$은 X_m과 X_m 다음에 오는 부분패턴 X_{m+1} 사이에, 앞장에서 정의한 함수로 구해지는 속성들의 집합이고, A(X)는 부분패턴 Xp, X_1, $\cdots$, X_{m+1} 사이에 A(Xp), $A(X_1), \cdots, A(X_m)$과 같은 연관관계를 가지고, m개 부분패턴 $X_1, \cdots, X_m$를 조합하여 생성되는 부분패턴 X의 합성속성(synthesized attribute)의 집합을 나타낸다.

다음은 위의 속성문법을 이용하여 자모에 대한 문법을 설계한 예이다.

예 1) 'ㅊ'의 경우를 속성문법으로 표현하면,

V_N = {S1, A, B, C, D}
V_T = {0, 5, 7}
P : 구문규칙　　의미규칙
　S1→ABC　　A(S1) = {A(A), A(B)}
　　A(A) = {b2}
　　A(B) = {a7, e1}
　A→0
　B→0D
　D→5
　C→7

이 문법의 구문규칙은 'ㅊ'이 'ㅡ', 'ㄱ', '丶'으로 구성되어 있는 것을 나타내며, 의미규칙은 'ㄱ'은 'ㅡ'의 하위에 있고, '丶'은 'ㄱ'의 7방향으로(그림 3. 참조) 근접해 있다는 것을 나타낸다.

예 2) 'ㄷ'을 표현하는 속성문법중의 하나의 예를 보면,

V_N = {S2, E, F, H}
V_T = {0, 6}
P : 구문규칙　　의미규칙
　S2→EF　　A(S2) = {A(X), A(E)}
　　A(X) = {b2, h2}
　　A(E) = {b2}
　　(여기서 X는 E의 직전에 인식된 패턴)
　E→0
　F→6H
　H→0

위의 예 2)에서 구문규칙은 'ㄷ'이 'ㅡ' 형태의 패턴과 'ㄴ' 형태의 패턴이 합성되어서 만들어 진다는 의미이며, 의미규칙은 모음하위에 부분패턴 'ㅡ'가 있고, 부분패턴 'ㅡ'의 하위에 'ㄴ'이 있다는 의미가 된다.

2. 속성문법의 자료화

문법은 어떻게 자료화하는가에 따라서 인식방법이 달라질 수 있다. 이 논문에서는 입력패턴이 스트로크열로 표현되고, 이들로부터 스트로크 및 속성을 조합하여 패턴을 인식하므로, Bottom-up 방식으로 인식하는 것이 유리하다.[6] 따라서 구조가 간단한 부분패턴에서 점점 복잡한 부분패턴으로 조합해 나갈 수 있도록 문법을 자료화하는 것이 필요하다.

[정의13] [정의12]에 의해서 어떤 부분패턴 X가 부분패턴 $X_1, \cdots, X_m$로 구성되면 이들 사이의 속성집

합을 $A(X)=\{A(Xp), A(X_1), A(X_2), \cdots, A(X_m)\}$로 표시된다. 이들 속성 및 부분패턴을 다음과 같이 의미-구문규칙열로 만들고,

$$Ap_1, \cdots, Ap_{n_P}, X_1, A_{11}, \cdots, A_{n_1 1}, \cdots, X_m, A_{1m}, \cdots, An_m m$$

이를 부분패턴 X에 대한 의미-구문규칙열이라 한다. 여기서, $A_{ij} \in A(X_j)$ $(1 \leqq i \leqq n_j, 1 \leqq j \leqq m)$이고, $Ap_k \in A(Xp)$ $(1 \leqq k \leqq n_P)$이며, A_{ij} 및 Ap_k는 속성함수 f1, …, f6에 의해서 생성될 수 있는 6가지 속성중의 하나로, $n_j, n_P \leqq 6$이 된다. 만약 위의 의미-구문규칙열에서 X_j가 스트로크일 때는 이들 스트로크를 기본패턴의 열로 치환시켜 이를 의미-구문규칙열로 할 수 있다.

이 논문에서는 자모에 대한 모든 문법의 부분패턴을 의미-구문규칙열로 표현하고, 의미-구문규칙열을 하나의 path로, 의미-구문규칙열내의 속성과 부분패턴을 각각 node로 하는 tree를 구성한 다음, 이 tree를 문법의 데이터 베이스(data base)로 한다. 만약 어떤 node까지 이르는 하나의 path가 있을 때, 이 path를 구성하는 node들이 각각 의미하는 부분패턴들이 조합되어 보다 복잡한 새로운 부분패턴을 나타내는데, 이 새로운 부분패턴에 대한 정보는 이 path의 마지막 node가 갖고 있다. 따라서 각 node는 자신이 의미하는 부분패턴의 정보와 root로 부터 자신까지 이르는 path내의 node들이 조합되어 구성되는 보다 복잡한 부분패턴의 정보를 갖고 있다.

V. 인 식

입력패턴으로 부터 맨처음 스트로크와 이들사이에 속성을 구할 수 있고, 이들을 앞에서 정의한 의미-구문규칙열 tree를 이용하여 부분패턴을 인식할 수 있고, 다시 이들 부분패턴들 사이에 속성을 찾아낸 다음, 이들에 다른 의미-구문규칙열을 적용하여 부분패턴을 찾아낸다. 새로운 부분패턴들에 대해 이러한 방법을 계속 적용하여 최종적으로 자모들을 인식해 낼 수 있다.

만약 부분패턴의 열 $Y_1, \cdots, Y_t, Y_{t+1}, Y_{t+2}, \cdots, Y_{t+m}, Y_{t+m+1}, \cdots$(여기서 $Y_1, \cdots, Y_t$는 이미 인식 처리된 보다 복잡한 부분패턴들의 열이고, Y_{t+1} 이후의 패턴은 해당되는 의미-구문규칙열을 적용하여 보다 복잡한 부분패턴으로 되어야할 부분패턴열이다)이 있고 이중에 부분패턴 $Y_{t+1}, \cdots, Y_{t+m}$열이 [정의15]에 있는 의미-구문규칙열 $Ap_1, \cdots, Ap_{n_P}, X_1, A_{11}, \cdots, A_{n_1 1}, \cdots, X_m, A_{1m}, \cdots, An_m m$(여기서 $A_{ij} \in A(X_j)$ $(1 \leqq i \leqq n_j, 1 \leqq j \leqq m)$ 이고, $Ap_k \in A(Xp)$ $(1 \leqq k \leqq n_P)$이다.)에 의해 인식되려면, 다음과 같은 조건을 만족해야 한다.

1) $Y_{t+j} = X_j$
2) $A(Y_{t+j}) \geqq A(X_j)$이고, $A(Y_t) \geqq A(Xp)$
   단, $1 \leqq j \leqq m$이다.

어떤 부분패턴의 열로 부터 보다 복잡한 부분패턴을 추출하기 위해서 의미-구문규칙열 tree로 부터 부분패턴의 열과 위의 조건이 일치하는 path중 가장 긴 것을 적용한다. 만약 일치하는 path가 없으면 적용한 부분패턴의 열중 첫번째 부분패턴을 리젝트(reject)하고, 다음 부분패턴부터 해당하는 의미-구문규칙열을 찾는다.

그림 7.은 '글자'라는 입력패턴으로 부터 추출된 기본패턴의 열로 부터 보다 복잡한 부분패턴열을 조합해 가는 과정을 보인다.

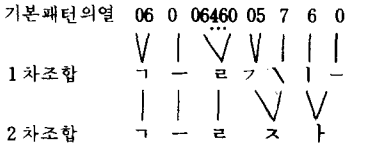


그림 7. 부분패턴열의 조합
Fig. 7. Combinations of subpattern string.

위 그림 7.의 예는 기본패턴열로 부터 부분패턴의 조합을 2번 시도하여 최종자모가 인식된 예이다.

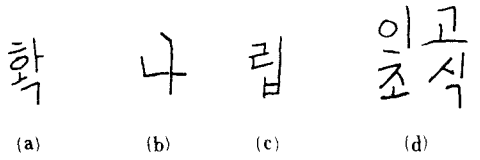
VI. 실험 및 결과

10명으로 부터 앞장에서 정의한 6가지 속성을 포함하는 20자 내외의 글자를 3줄로 정자로 쓰게하고, 글자사이의 간격은 자유스럽게 하였으나 글자를 간격없이 연결하여 쓴 글은 제외하였다. 그리고 글자 높이는 5mm에서 10mm정도 내로하고, 세로 크기에 제한을 두지 않고 쓰는이의 평소 습관대로 종이위로 쓰도록 한다음 1cm글자 높이가 40개의 화소로 나타나도록 컴퓨터에 입력시켰다. 입력패턴으로 부터 스트로크를 추출하기전에 세선화 처리를 수행하였다. 10인이 쓴 총 218자에 대해 수행한 결과를 표 1.에 나타낸다. 인식 처리중에 발생한 오인식 또는 리젝트된 경우를 보면,

1) 스트로크의 sorting이 잘못된 경우 (그림 8 a)
2) 스트로크 끝점 판정이 잘못된 경우 (그림 8 b)
3) 자모의 스트로크가 연결된 경우 (그림 8 c)

4) 블럭의 추출이 잘못된 경우 (그림 8 d)
5) 세선화 처리후 패턴에 변형이 생긴경우

그림 8(a)의 경우는 '확'에서 'ㅗ'와 'ㅏ'의 순서가 바뀌어 결과가 'ㅎ', 'ㅏ', 'ㅗ', 'ㄱ'가 되어 모음 '나'가 오인식 되었으며, 그림 8(b)의 경우 모음이 교차점에서 분리되어 리젝트가 되었고, 그림 8(c)의 경우는 자음의 일부가 모음의 일부로 처리되어 받침 자음이 리젝트 되었다. 그리고 그림 8(d)의 경우는 아래줄의 '초'의 맨위의 스트로크가 윗줄의 '이'의 일부로 인식되어 두글자의 모음과 자음이 둘다 오인식 되었다. 세선화에 의해서 발생하는 오인식중 가장 많은 경우는 'ㅇ'과 'ㅁ'으로 'ㅇ'의 경우 세선화 과정을 거치면서 방향변화를 일으키는 작은 선분이 무시되어 직선성분으로 나타나 'ㅁ'으로 오인식 되고, 'ㅁ'의 경우는 굴곡점 근처의 작은 직선성분이 강조되어 'ㅇ'으로 처리되는 경우가 있었다.



(a) (b) (c) (d)

그림 8. 오인식 또는 리젝트되는 경우의 예
Fig. 8. Examples of ill-recognized or rejected characters.

표 1. 인식실험의 결과
Table 1. Result of experimentation.

정 인 식	513	94%
리 젝 트	21	4%
오 인 식	12	2%
자 모 총 수	546	100%
글 자 수	218	

Ⅶ. 결 론

한 글자단위로 인식하는 알고리듬의 경우는 인식 이전에 각 글자들을 분리해야 하므로 글자의 간격과 크기가 일정치 않은 필기체 문서에 적용하는 것은 어렵다. 이 논문은 이러한 필기체 한글 문서로 부터 각 글자들을 분리하지 않고 줄 단위로 스트로크를 분리하여, 이들 사이의 배열관계에 대한 속성을 추출하고, 다시 이들을 조합하여 문서내의 자모를 인식하는 방법을 제안하였다.

문서로 부터 자모의 인식이 끝나면, 이들 자모의 위치 및 의미를 알 수 있으므로 정인식된 부분 뿐만 아니라, 오인식 및 리젝트된 부분에 대해서도 좌우 자모의 배열 및 의미 등을 이용하여 대부분 분리가 가능하다. 따라서 리젝트된 부분(앞장의 오인식 및 리젝트되는 경우중 1), 2), 3)의 경우)에 대해서 이미 개발된 한 글자단위로 인식하는 알고리듬을 이용하여 다시 처리하면 인식율도 많이 향상될 것으로 생각된다.

參 考 文 獻

[1] T. Agui, M. Nakajima, T.K. Kim, and E.T. Takahashi: "A method of recognition and representation of Korean characters by tree garmmars," *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 245-25, 1979.

[2] 김태균, 安居院猛, 中嶋正之 : "Stroke 조합에 의한 필기체 한글의 표현과 인식," 전자공학회 논문지, 제 5 권, 제 1 호, pp. 18-26, 1988.

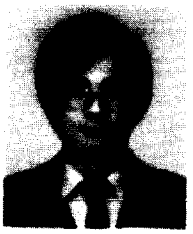
[3] 이주근, "패턴인식 시스템에 관한 동향," 정보과학회지, 제 3 권, 제 2 호, pp. 22-33, 1985.

[4] 이성환, 조창제, 김진영 : "실용적 한글 문서 자동 인식 시스템 개발의 문제점 및 개선방향," 한국정보과학회 봄 학술발표논문집, vol. 15, no. 1, pp. 127-130, 1988.

[5] W.H. Tsai, K.S. Fu: "Attributed grammar-a tool for combining syntactic and statistical approaches to pattern recognition," *IEEE. Trans. on Systems, Man, and Cybernetics*, vol. SMC-10, no. 12, pp. 873-875, 1980.

[6] K.S. Fu: Syntactic Pattern Recognition and Applications, Prentice Hall, pp. 116-0123, 1982. *

著 者 紹 介



柳 承 必(正會員)
1955年 6月 16日生. 1979年 서울대학교 전자공학과 졸업. 1987年 충남대학교 대학원 전자공학과 졸업 석사학위 취득. 1987年～현재 충남대학교 대학원 전자공학과 박사과정. 1980年～현재 한국에너지연구소 계측제어연구실 선임연구원. 주관심분야는 Computer, Vision, 패턴인식, 인공지능 등임.

金 太 均 (正會員) 第26卷 第 1號 參照
현재 충남대학교 전자계산기공학과 부교수