產業科學技術硏究所 論文集 第二卷 第二號
(J. of Ind. Sci. and Tech. Institute)
Vol.2, No.2, 1988. 10.

구문적 패턴을 이용한 필기체 숫자의 인식에 관한 연구

全 炳 敏 · 延 承 鎬 · 潘 陽 綠
忠北大學校 工科大學 電子計算機工學科

A Study on Recognition of Handwritten Numerals Using Syntactic Patterns

Byoung-Min Jun, Seung-Ho Youn, Yang-Rok Ban

Dept. of Computer Engineering, Chungbook National University, Cheongju 360-763, Korea

Abstract

In this thesis, a syntactic pattern recognition method, which recongnizes the handwritten numerals using context-free grammars, is proposed.

After thinning numeral image of binary pattern, the thinned binary pattern is converted into the string which consists of Freeman's chain code, string operator and terminal symbol. The noise is eliminated from the string, and the string is manipulated for input of the parser.

As a result of experiment, the correct recognition rates are 92.7%, the erroneous recognition rates, 0.91% and the rejection rates 6.36%.

I. 서 론

패턴 분류기법은 표현방법과 판별방법에 따라 원형비교방법, 통계적 방법, 구문적 방법등으로 분류할 수 있으며, 구문적 방법은 패턴을 구성하고 있는 기본 원소들의 스트링 형태나 원소들과 입력패턴 사이의 관계를 나타내는 트리나 그래프 형태로 표현한다. 입력패턴이 주어졌을때 먼저 그 패턴을 구성하고 있는 기본 원소들을 인식한 다음에 패턴 그래프의 자료구조를 생성한다.

인식대상숫자를 카메라로 입력하여, 입력된 영상을 32×32의 이진 영상으로 변환하고, Deutch의 알고리즘[3]을 이용하여 중심 골

격선을 찾는 세선화를 수행하였다. 세선화된 숫자영상을 스트링으로 변환하는데 있어서 복잡한 문제가 발생되는 굴곡점과 교점을 제거하는 과정을 추가하고, 스트링을 추출하기 위해서 2차원 영상의 하단 최좌측 점에서 우측으로 한 라인 단위로 상단으로 주사하여, 처음 만난 점을 출발점으로 하였다. 각 화소에 대해서 Freeman[2]의 방향코드를 기본 생성원 (a, b, c, d, e, f, g, h) 으로 하여, 방향 우선순위에 따라 입력버퍼에 입력하였다. 입력 스트링을 그대로 사용할 경우, 문법이 복잡하고 파서의 구성이 방대해지므로 인식시간의 증가와 인식률이 저하되는 원인이 되기 때문에 잡음을 제거하고, 숫자의 특징을 잃지 않는 범위 내에서 스트링을 간략하게 하였다. 처리된 스트링은 context-free 문법을 이용하여 구성한 파서에 의해 파싱을 수행하고, 성공적으로 수락하면 해당숫자를 출력하였다.

Ⅱ. 스트링 언어

기본 생성원은 단위 성분의 구조적인 정보를 포함하는 패턴의 기본적인 성분이다. 패턴은 기본 생성원으로 나타낼 수 있으며, 입력패턴은 Freeman의 방향코드를 Fig.2-1과 같이 소문자 알파벳 (a, b, c, d, e, f, g, h)으로 나타내고 이를 기본 생성원으로 한다. 인식 대상 숫자 영상을 파서의 입력으로 사용하기 위해서 기본 생성원을 이용하여 각 입력 숫자 영상을 스트링으로 변환하여야 하고, $m \times m$배열로 이루어진 숫자 영상의 화소중 하단 최좌측 점을 스트링의 출발점으로 선택한다. 입력 숫자 패턴을 스트링으로 표현하기 위해서 각 기본 생성원에

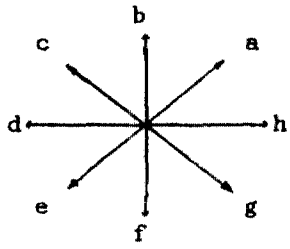
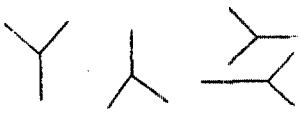


Fig.2-1 Freeman's chain code

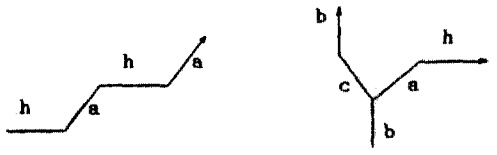
h,a, b, c, d, e, f, g의 순서로 우선순위를 부여한다. 입력 패턴을 단점 (end point), 굴곡점 (right angle point), 분기점 (branch point), 교점 (cross point)으로 구성되지만, 굴곡점과 교점은 스트링 표현에 부적당하기 때문에 단점과 분기점만으로 구성된 입력 패턴을 스트링으로 나타낸다. 단점은 주위에 1개의 화소만 존재하는 점이고 분기점은 이웃에 3개의 화소가 존재하는 점으로, 입력 패턴을 스트링으로 나타낼 때 단점은 스트링 연산자 *를 삽입하고, 분기점은 스트링 연산자 +를 삽입하여 표현한다. Fig.2-2에서 (a)와 (b)는 각각 단점과 분기점을 나타내고, (c)는 스트링 연산자 *, +를 이용하여 스트링으로 변환하는 예를 나타낸다.

스트링 연산자를 삽입하여 나타낸 스트링

(a)



(b)



(c)

Fig.2-2 (a) End point
(b) Branch point
(c) String operator (* , +)

은 최종으로 입력버퍼에 입력되는 스트링으로 모든 처리 결과의 끝에 나타나는 스트링 연산자 *를 스트링의 끝을 나타내기 위해서 종단기호 $로 대치한다. 따라서 숫자는 8개의 기본 생성원, 단점과 분기점을 표현하는 스트링 연산자 *, +, 그리고 입력 버퍼와 마지막에 삽입되는 기호, $를 스트링으로 표현한다. Fig.2-3은 숫자 3이 처리되는 순서를 나타내며, 처리 순서에 따라 표현하면 스트링 habc + abcd * d * c $을 얻는다.

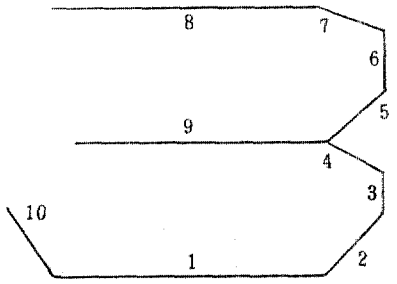


Fig.2-3 Pattern description of numeral 3

Ⅲ. Context-free Grammar와 파서

인쇄체 숫자의 경우 잡음이 적고, 같은 필체인 경우 처리된 스트링이 같기 때문에 문법작성이 용이하나, 필기체 숫자의 경우 필자의 습성에 따라 다양한 형태의 스트링을 형성하기 때문에 가장 공통적인 특성을 가지며, 가능한 한 많은 스트링을 생성하는 Context-free 문법을 적용한다.

정의 1 Context-free Grammar G는

$$G=(N,T,P,S)$$

로 정의하며, 여기에서

N : 논터미널 기호의 유한집합
T : 터미널 기호의 유한집합
$N \cap T=\phi$, $N \cup T=V$
S : $S \in N$인 생성의 시작기호
P : 생성규칙의 유한집합으로
$\alpha \rightarrow \beta$, $\alpha \in V^{+}$, $\beta \in V^{*}$

와 같이 표현하며, 단 α는 N을 적어도 한 개 포함한다.

생성은 $\phi A \psi \rightarrow \phi W \psi$에서 ϕ, ψ는 공백이 되어 생성은

$$A \rightarrow W \quad \begin{cases} A \subset N \ \{S\} \\ W \subset (N \cup T)^{*} - \{\lambda\} \end{cases}$$

로 표현되며, $S \rightarrow \lambda$를 문법이 포함하기도 한다. 문법 G가 Context-free grammar 이면 L(G)는 Context-free langnage가 된다. 문법 G에 의하여 생성된 Context - free langnage L(G)는

$$L(G)=\{W \mid W \subset T^{*} \text{ and } S \Rightarrow W^{*}\}$$

로 정의되며 $S \Rightarrow W$의 W는 문법 G에 의하여 S에서 유도됨을 나타낸다.

파서는 파싱 테이블을 이용하여 정유한자동으로 인식을 수행한다. 파싱 테이블은 파

서의 구동 프로그램 내에 위치하며, Shift, Reduce, Go To의 동작을 정해준다. 파싱 테이블 작성기법은 SLR (Simple LR), CLR (Cannonical LR), LALR(Lookahead LR)로 구분하고, 작성하기가 가장 간단한 SLR파싱 테이블을 적용한다.[4]

파싱은 주어진 스트링이 정의된 문법에 의해 생성될 수 있는지를 결정하는 것으로 올바른 스트링에 대해서는 스트링의 구조를 나타내는 파스트리를 구성하여 인식한 숫자에 해당하는 숫자 코드를 출력으로 나타낸다.본 논문에서는 Context-free 문법으로 작성한 숫자들의 문법에 대하여 구성이 용이하며,파싱 방법도 가장 보편성을 띄고, 오차의 색출을 입력의 검조에서 쉽게 할 수 있는 LR 파서를 적용한다.[4] LR파싱은 입력 스트링의 검조를 왼쪽에서 오른쪽으로 읽어가며 우단유도의 역을 이용하는 것으로 상태번호와 handle이 발견될 때까지 입력포인터가 지시하는 기호들을 순차적으로 스택에 저장하며, 입력 버퍼에는 아직 읽지 않은 기호들이 존재하게 된다. 스택에 있는 기호와 아직 읽지 않은 기호는

스택 : $S_0X_1S_1X_2 \cdots X_nS_m$
(S_i : 상태, $X_i \in V$)

버퍼 : $a_1a_2a_3 \cdots a_n\$$ (a_i : 읽지 않은 입력)

으로 나타낸다. Fig.3-1은 입력버퍼, 스택, 파싱 테이블을 갖는 구문 분석기로 LR파서를 구성하고, Algorithm 3-1은 파싱하는 과정을 나타낸다.

Algorithm 3-1 LR Parsing

1. 만일 Action $[S_m, a_i]$ = Shift S이면 파서는 shift를 실행하며 스택과 입력은

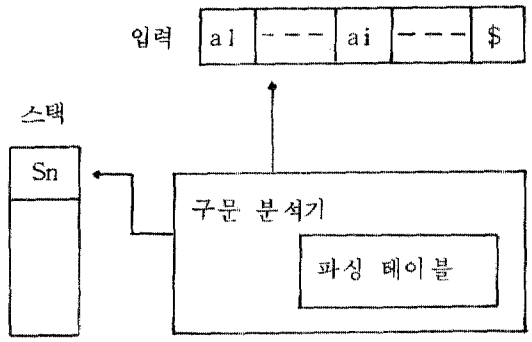


Fig.3-1 LR 파서

($S_0X_1S_1X_2S_2 \cdots X_mS_m\ a_iS,\ a_{i+1} \cdots a_n\$$)으로 변하며 a_iS가 스택에 Shift되며, a_{i+1}이 새로운 현재 입력이 된다.

2. 만일 Action $[S_m, a_i]$ = reduce $\alpha \rightarrow \beta$이면 파서는 reduce를 실행하고, 스택과 입력은

($S_0X_1S_1X_2S_2 \cdots X_{m-n}S_{m-n}AS a_i\ a_{i+1} \cdots a_n\$$) 으로 변하며, 여기서 S = GO TO $[S_{m-n}, A]$이며 $n = |\beta|$이다.

3. 만일 Action $[S_m, a_i]$ = accept이면 파싱은 완료되고

4. 만일 Action $[S_m, a_i]$ = error 이면 인식불가의 결과를 가져온다.

Ⅳ. 실험및 결과고찰

2진영상을 획득하여 문자열을 추출하고, Context-free에 의한 문법을 작성하여, 파서에 의해 인식을 수행하였다.

2진영상을 획득하기 위해 숫자영상을 CCD 비디오 카메라로 입력하여 NTSC복합 영상신호를 디지타이저에 의해 512×512의 해상도를 갖고 256 계조도를 갖는 디지틀 영상을 필기체 숫자의 특성이 상실되지 않

도록 스레쉬홀딩 (thresholding)기법에 의해 2진영상으로 변환한 후, 2진 보간법을 이용하여 해상도 32×32의 정규화된 2진영상을 얻었다. 그림 4-1의 (a)는 실험에 사용된 해상도 32×32의 영상표본을 나타내며, (b)는 영상(a)를 세선화 알고리즘[3] 으로 스트링 처리과정이 간단해지도록 처리한 후의 영상이다. (c)는 숫자영상을 스트링으로 만들기 위해서 세선화된 영상의 32×32배열의 좌단 최하측 점에서 좌에서 우로 한 라인씩 상

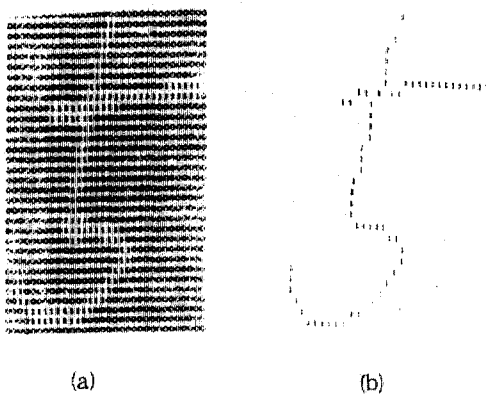


Fig.4-1

단으로 주사하면서 처음 만난화소를 출발점으로 정하고 숫자를 구성하는 각 화소를 Freeman의 방향코드인 기본생성원을 이용하여 얻은 결과이다. 그림(d)는 2차원 영상에서 1차원 문자열을 추출한 후 숫자의 특성이 없어지지 않는 범위내에서 줄여놓은 문자열이다. 이 문자열을 Context-free문법에 의해 문법을 작성하여

0	S5 → A	1	A → *h*B
2	A → B + F	3	A → B + *k*
4	A → B + G	5	B → *h*C
6	B → *h*D	7	B → aC
8	C → a D	9	C → bD
10	C → c L	11	D → d E
12	D → d L	13	D → c D
14	D → c J	15	E → b
16	F → *h*	17	G → F * H
18	G → F * E	19	H → E * I
20	I → c	21	J → a
22	K → bL	23	L → a F

을 얻는다. 파서는 각 숫자에 대한 문법을 이용하여 파싱테이블을 얻은 후 이 파싱테이블을 Fig.4-2와 같은 숫자의 순서로 인식

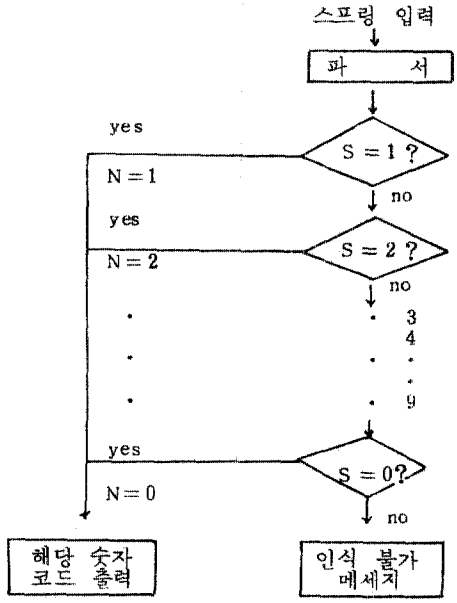


Fig.4-2 Flowchart of Parsing

을 수행하였다.

실험결과는 Table.4-1에서와 같이 110개의 숫자중 1개의 오인식과 7개의 인식불가이었으며, 92.7%의 인식률을 얻었다. 숫자 1의 경우, 필기체의 긴 선분이 a의 방향을 갖고, e방향 선분이 처리과정에서 d의 방향으로 처리되었기 때문에 7로 오인식 되었다. 인식불가된 숫자 7개는 첫째, 1의 중심 특징선이 ab이고 윗 부분은 e방향에서 d방향으로 처리되어 스트링 abd$의 결과를 얻으며 둘째, 숫자2의 좌측하단 부분이 호의 형태를 갖는 것으로 h+babc*h*$의 결과를 얻고 셋째, 숫자4의 중심 골격선 윗부분과 횡선 좌측 부분이 잡음 제거시 삭제되어 b*da$의 결과를 얻으며 다섯째, 숫자6에서 호와 윗선분이 만나는점에서 연결되지 않아 hd*ca$의 결과를 얻고, 여섯째 숫자9에서 호의 좌측 부분이 세선화시에 혹으로 남아 스트링 연산자+가 삽입되어 a+ad+*h*$의 결과를 얻고 일곱째, 숫자0에서 호의 윗부분이 연결되지 않아 hacd*cba$의 결과를 얻는다. 따라서 인식불가는 숫자의 특징이 되지 않는 선분이 존재할 경우, 꼭 필요한 특징이 잡음 제거시 삭제될 경우 숫자의 모양을 갖추지 못하여 특징이 상실될 경우, 연결되어야 할 선분이 연결되어 있지 않을 경우, 이외에도 영상입력시 불필요한 잡음이 세선화 과정에서 처리되지 않아 숫자와 분리하여 영상의 아랫부분에 존재할 경우에 발생하였다.

V. 결 론

본 연구에서는 필기체 숫자영상을 세선화하고, 세선화된 패턴을 스트링으로 변환한 후, 스트링 처리과정을 거쳐 입력 스트링을 추출하였다. 추출된 스트링을 각 숫자에 대하여 Context-free 문법을 이용하여 구성한 인식파서에 입력하고, 파싱이 성공적으로 수락되면 인식결과에 대한 해당 코드를 출력하였다. 임의 선정한 4명의 학생이 필기한 110개의 숫자를 대상으로 인식 실험한 결과는 정인식이 92.7%, 오인식이 0.91%, 인식불가가 6.36%이었다. 스트링 처리과정에서 특징이 변형되어 오인식이 발생하였으로, 숫자의 특징이 되지 않는 선분이 존재하거나, 필요한 특징이 잡음제거시 삭제되거나 숫자의 모양을 갖추지 못하여 특징이 상실될 경우, 또한 연결되어야 할 선분이 연결되지 않을 경우에 인식이 불가능 하였다. 인식률을 향상시키기 위해서는 스트링 추출 방법을 더 효율적으로 개선하고, 융통성있는 문법의 작성이 요구되며, 오류 수정 루틴을 적용하여 재인식하는 방법이 요망된다. 또한 인식 불가된 숫자를 인식하기 위

Table.4-2 Recognition results

분류숫자	1	2	3	4	5	6	7	8	9	0	계
정 인 식	9	10	11	10	11	9	11	11	10	10	102
오 인 식	1	0	0	0	0	0	0	0	0	0	1
인식불가	1	1	0	1	0	2	0	0	1	1	7
인식률(%)	81.8	90.9	100	90.9	100	81.8	100	100	90.9	90.9	92.7

해서 문법의 생성규칙을 추가하면 인식파서가 복잡하고 인식시간이 지연되므로 효율적인 파서의 구성이 요구된다.

참 고 문 헌

1. J.T. Tou, R.C. Conzalez, Pattern Recognition Principles, p.316-360, Addison-Wesley, 1974.
2. K.S.Fu, Syntactic Pattern Recognition and Application, p.192, p.246-257, p.334, Prentice-Hall, 1982.
3. E.S.Deutsch, Thinning Algorithm On Rectangular Hexagonal and Trianguglar Arrays. Communication of the ACM Vol.15, p.827-837, 1972.
4. A.Aho, J.Ullman, Principle of Compiler Design. p.188-189, p.197-224, Addison-Wesley, 1977.