論文94-31B-9-18

자소간의 흘림을 허용하는 연속형 온라인 필기 인식 시스템의 구현

(Implementation of An On-Line Continuous Recognition System for Cursive Handwriting)

權 五 成*, 權 寧 彬*

(Oh-Sung Kwon and Young-Bin Kwon)

要 約

본 논문에서는 흘려쓴 한글 필기의 온라인 인식을 위한 정합과 구현 방법을 설명한다. 구현된 인식시스템에서 한글인식은 자소정합 결과의 결합으로 이루어지는데, 이를 위하여 새로운 스트링 정합을 제안한다. 이 정합과정은 스트링 편집을 이용하여 수행되며 두 자소간의 정합과정은 1개의 편집 경로로 표현되는 고속의 정합방식에 의해 수행된다. 정합 결과들은 heap구조에 저장되고 최종적인 인식 결과는 탐색과정을 거쳐 결정된다. 제안하는 온라인 한글 인식 시스템은 이러한 고속 정합에 의한 인식방법을 사용하므로써 인식과 필기의 병행적인 수행이 가능하다. 실험결과 50명이 쓴 21,076자에 대하여 86.36%의 인식률을 보였고, 글자당 평균 1.75초의 인식시간을 소요하므로써 제안하는 구현 방법이 자소간의 흘림을 허용하는 연속형 온라인 필기 인식에 적합하게 동작함을 알 수 있었다.

Abstract

In this paper, an implementation of on-line continuous recognizer for cursive Hangul handwriting is explained. For the Hangul recognition system, we propose a high speed string matching. The editing process in our proposed string matching is accomplished by single editing path. And the matching results are stored in a heap structure and we decide the recognition result by search of the heap structure. And our recognition system is able to give the user comfortibility of unceasing writing during recognition owing to the high speed matching. In the experimental result, a recognition rate of 86.36% at 1.75 second/character over 21,076 characters collected from 50 persons are obtained. And it is shown that the proposed recognition system is operated properly for the on-line recognition for cursive handwring between graphemes.

*正會員, 中央大學校 工科大學 컴퓨터工學科
컴퓨터비젼 硏究室/ 인공지능연구센터
(Computer Vision Laboratory, Department of Computer Engeneering, College of Engineering, Chung-Ang University and Center for Artificial Intelligence Research)

接受日字 : 1993年 12月 16日

Ⅰ. 서 론

본 논문에서는 격자로 구분된 형태의 화면위에 전자펜으로 한글을 필기하고, 필기된 한글을 인식하여 처리하는 온라인 인식 시스템의 구성을 제안하기로 한다. 이러한 온라인 인식 시스템은 사용의 편리성을 위하여 만족스런 인식률, 고속의 인식속도를 요구한다. 현재까지 온라인 한글 인식에 관한 연구는 주로 자소내 홀림 필기에 관한 것이었고, 아직까지 자소간 홀림 필기의 경우는 심한 필기 변형과 자소분할의 어려움때문에 인식률과 인식속도 면에서 많은 개선을 필요로 하고 있다.[1,7]

온라인 한글 인식에 관한 연구는 지속적으로 이루어져 왔으며, 대표적인 방법들을 살펴보면 다음과 같다. 우선, 획의 표준패턴을 정의하고 이를 조합하는 구조적인 방법을 사용하여 인식하고자 한 연구가 있었으며, 신경망을 이용한 경우의 연구로는 필기형태에 근거한 획(stroke)분류 테이블을 구성한 후 추출된 특징들을 획 인식 신경망의 입력으로 받아서 문자를 인식하도록 한 연구[4]가 있다. 이와 같은 방법들은 필기 방식이 다양하기 때문에 미리 설정하여 놓은 획의 형태만으로는 인식률을 높이는 것이 어렵다는 문제를 지니고 있다. DP(dynamic programing)방법은 온라인과 오프라인의 한글을 포함한 문자인식에서 자주 사용되는 인식방법으로서 다양한 연구 결과[3,6,13,14,16]가 보고되고 있다. 일반적으로 DP방법은 탐색 및 계산과정이 복잡하여 시간이 많이 걸린다는 단점이 있다. 그러므로, 탐색해야 할 공간을 줄이기 위하여 인식에 앞서 대분류 과정을 갖기도 한다. 통계적인 방법으로는 HMM(Hidden Markov Model)을 이용한 연구[5]가 있는데, 대부분 한글의 제자원리에 따라 필기 네트워크를 구성하고, 확률적으로 최적의 경로를 찾기 위한 탐색을 하는 방식으로 구성되고 있다. 이러한 통계적 방법은 홀림체 필기의 모델링에는 탁월한 효과를 갖지만, 계산상의 복잡도로 인한 인식속도의 저하라는 문제점을 갖고 있다. 다른 방법의 연구로서는 인식과정에서 발생되는 획의 불명확성을 퍼지 소속함수로 표현하여 해결하고자 한 경우[7]가 존재한다.

이들 대부분은 낱자 필기후 인식을 하도록 구성되어 있으며 자소내 홀림에 관련된 문제를 해결하는 데에 주력하고 있다. 그러므로 자소간 홀림도 허용하면서 연속 필기까지 가능하게 하는 인식시스템의 구현을 위해서는 고속의 인식속도를 갖는 인식방법이 요구되며, 인식중에도 계속적인 필기가 가능하도록 하여 사용자에게 불편함이 없도록 하여야 한다. 본 논문에서는 스트링 정합방법에 기반한 인식 방법을 사용하고 있는데, 이 정합방법은 스트링 편집과 정의된 비교순서규칙에 의해 고속의 인식이 가능한 장점을 갖는다. 또한, 문자 인식과 필기 데이타 수집이 동시에 수행될 수 있도록 구성하므로써 자유로운 필기 동작을 가능하도록 하고 있다. 생성된 후보문자들 중에서 인식 결과를 결정하기 위한 탐색은 A* 알고리즘에 의해 진행되도록 하여 탐색공간을 줄였으며, 인식과정 중에 계속적인 변동사항은 heap구조를 이용하여 손쉽게 처리되도록 하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 인식시스템의 전체적인 구성을 설명하고, Ⅲ장에서는 스트링 정합을 이용한 인식방법을 설명한다. Ⅳ장에서는 실험결과를 통하여 제안하는 인식 시스템의 효율성을 보이고, 결론을 제시하기로 한다.

Ⅱ. 인식시스템의 전체적인 구성

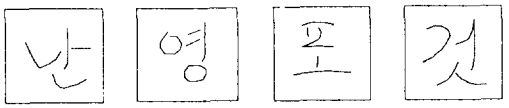
온라인 한글 필기의 유형은 표 1과 같이 구분될 수 있다. 표 1의 글자의 홀림 형태 중에서 자소간 홀림은 자소내 홀림을 허용하는 경우와는 달리 자소인식에 앞선 자소분할을 요구하며, 이러한 요구는 한글 인식을 어렵게 하는 중요한 요인으로 작용하고 있다.[1,3,5]

표 1. 온라인 한글 필기의 유형분류
Table 1. Types of on-line Hangul handwriting.

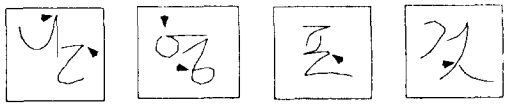
글자 홀림 형태에 따른 분류	글자 구분 방식에 따른 구분	인식수행 시점에 따른 구분
• 자소내 홀림 필기 • 자소간 홀림 필기 • 글자간 홀림 필기	• 격자구분 필기 • 공백구분 필기 • 시간구분 필기 • 무제한 필기	• 한 문자 필기 후 인식 • 문자열 필기 후 인식 • 필기와 병행하는 인식

이러한 이유로 기존의 온라인 한글인식 연구는 주로 자소내 홀림 허용 필기 인식에서 이루어져 왔으며, 최근들어 자소간 홀림의 연구도 활발히 수행되고 있다. 자소내와 자소간 홀림필기의 예는 그림 1과 같으며, 그림 1 (b)에서 화살표는 자소가 연결된 부분으로서 인식을 위하여 분할이 요구되는 부분을 나타낸다.

글자간 홀림의 경우는 일반적인 홀림의 형태가 아니기 때문에 문자인식 분야에서는 아직까지 고려하지 않는 필기형태라고 할 수 있다.



(a)



(b)

그림 1. 글자 흘림 형태의 예
(a) 자소내 흘림 필기
(b) 자소간 흘림 필기

Fig. 1. Examples of cursive handwriting.
(a) Cursive style within grapheme
(b) Cursive style between graphemes

글자구분 방식에서, 격자구분 필기의 경우에 사용자는 원고를 작성하듯이 필기를 해야하는 제약이 따른다. 반면에, 이러한 유형은 글자열에서 글자분할을 위한 별도의 과정을 요하지 않는 용이함을 갖는다. 공백구분에 의한 필기는 글자와 글자 사이에 공백을 두면서 필기하는 형태로서 글자열의 공백을 기준으로 글자를 분할하고 인식을 수행하며, 이러한 필기형태 역시 필기시 공백을 만드는 주의를 요한다. 또 다른 필기 유형으로는 필기시 아무런 제약을 요구하지 않는 무제한 필기를 들 수 있는데, 이러한 글자유형의 분할을 위해서는 보다 많은 연구가 필요한 상황이다.

인식기가 수행되는 시점에 따른 인식방법은 표 1과 같이 3가지 유형으로 구분될 수 있다. 이러한 구분중 한 문자 필기 후 인식은 한 문자의 필기를 마친 후, 인식기가 필기 종료 신호를 받아들여 인식을 수행하는 방식이다. 이 방식에서 사용되는 필기종료 신호는 주로 시간적인 정보 혹은 공간적인 정보로서 시간적으로 정의된 간격을 경과하거나 현재의 필기 공간을 벗어날 때 이러한 신호가 발생된다. 이러한 방법은 필기후 종료 신호가 발생되고 필기된 문자가 인식되기를 기다렸다가 다음 글자를 필기해야 하는 어려움이 따른다. 문자열 필기 후 인식 방법은 일정분량의 문자열 필기후에 인식을 수행하는 방법으로 사용자는 아무런 제약없이 일정 분량의 필기를 진행할 수 있으나 필기자는 필기 후 문자열이 인식되는 동안 기다려야 하는 불편을 갖는다. 필기와 병행하는 인식방법에서는 필기자가 인식기에 구애받지 않고 필기를 진행할 수 있다. 이러한 방법에서는 복수개의 프로세스가 병행적으로 수행되야 하기 때문에 신속한 인식시간을 갖는 인식방법을 요구한다. 이 방법에서, 연속적으로 입력되는 필기데이타의 글자분할은 필기 유형에 따라 다양한 방법이 적용될 수 있다.

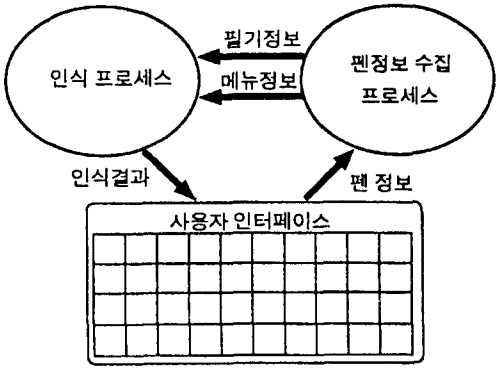


그림 2. 인식시스템의 전체적인 구성도

Fig. 2. Overall configulation of the recognition system.

본 논문에서는 설명하고자 하는 온라인 인식 시스템은 표 1에서 밑줄로 표시한 것처럼 격자내 자소간 흘림을 허용하고, 인식기의 수행되는 시점은 필기동작과 병행되는 구조를 갖도록 구성하였다. 입력된 필기 데이타로 부터 글자 분할은 시간적인 정보와 격자의 공간적인 정보를 이용하여 실행하였고, 인식은 분할된 글자 단위로 동작한다. 전체적인 시스템 구성도는 그림 2와 같다. 전체적인 시스템은 글자를 인식하는 프로세스, 전자평판(tablet)으로 부터 얻어진 필기데이타와 메뉴선택정보를 수집하여 글자 단위로 글자인식 프로세스로 전송하는 프로세스로 구성된다. 인식되어진 글자는 원고작성 인터페이스에 되돌려져 디스플레이된다.

인식프로세스에 입력된 필기는 인식에 영향을 미치는 필기시의 떨림과 삐침을 최소화하기 위하여 전처리 과정을 거친 후에 등간격의 선분들로 나뉘어진다. 이러한 방식으로 분할된 선분들은 선분의 특성에 따라 가상 선분, 모서리 선분, 일반 선분의 3가지 유형으로 분류된다. 이러한 선분들은 스트링으로 표현하기 위한 알파벳(alphabet)은 선분의 유형 분류와 선분들의 방향값에 따라서 정의되었다. 즉, 입력문자는 정의된 알파벳의 기호들을 사용하여 스트링으로 변환되도록 하였다. 이와같은 스트링의 생성과정은 Ⅲ장

의 1절에서 설명되어지며, 전체적인 스트링 정합과정은 그림 3과 같이 보여질 수 있다.

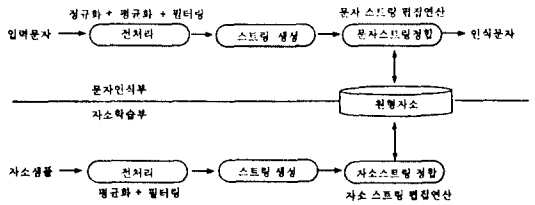


그림 3. 스트링 정합과정
Fig. 3. String matching process.

그림 3에 나타나 있듯이 본 논문의 인식방법은 입력문자를 자소들로 분할한 뒤, 각각의 자소를 원형(prototype)자소와 정합하는 과정을 거쳐서 인식하도록 하고 있다. 인식을 위한 원형자소의 생성을 위해서는 학습샘플(learning sample)이 요구되며, 학습은 기존의 원형자소들을 기준으로 하는 자소 스트링 정합에 의하여 새로운 원형자소의 생성여부를 결정하는 과정이다. 학습부에서의 전처리는 문자 인식부와는 다르게 정규화 과정을 필요로 하지 않는데, 그 이유는 문자 샘플들을 정규화한 상태에서 자소 샘플을 수집했기 때문이다.

Ⅲ. 고속정합에 의한 입력글자의 인식

1. 스트링 생성

인식시스템의 입력장치로 사용되는 전자평판은 필기자의 필기를 이루는 점들의 x,y좌표값과 사용자의 pen-up과 pen-down 정보를 제공한다. 1개의 획은 pen-down과 pen-up사이의 좌표값들에 의해서 구성된다. 일반적으로 입력된 필기문자는 tablet과 전자펜이 갖는 재질에 관련된 떨림과 삐침을 포함하고 있다. 본 논문에서는 이러한 현상들이 인식에 미치는 영향을 최소화하기 위한 인식에 앞선 전처리과정을 갖는다. 전처리는 평활화(smoothing), 공간필터링(spatial filtering)과 정규화(normalization) 과정으로 구성된다. 평활화는 이웃하는 점들과의 평균화를 통하여 필기시의 떨림을 흡수한다.[8,14] 공간필터링은 필기자의 불규칙한 필기속도에 기인하여 연이은 점들의 간격이 필요없이 조밀하거나 동떨어진 경우를 고르게 조정한다. 정규화는 입력되는 문자의 크기를 특정한 사각형(100×120)안으로 높이와 폭을 대응시킴으로써 입력문자로 부터 생성되는 스트링이 문자의 크기에 무관하도록 한다.

이러한 전처리 과정을 거친 입력문자들은 등간격의 선분들로 나뉘어지며, 획과 획을 연결하는 가상 획도 등간격의 선분들로 나뉘어 표현된다. 나뉘어진 선분은 표 2와 같이 3가지 유형으로 분류한다.

표 2. 선분의 유형분류
Table 2. Classification of line-segment.

선분의 유형	유형의 정의
가상선분	가상획으로 부터 생성된 선분
모서리선분	자신이 갖는 방향성과 시간적으로 바로 다음의 선분의 방향성의 차이가 급격한 차이를 보이는 선분
일반선분	가상 선분과 모서리 선분에 해당하지 않는 선분

가상선분은 가상 획의 등간격 분할의 결과로 생성된 선분으로서, 가상획은 입력된 필기데이타 중에서 획과 획을 잇기 위하여 펜이 pen-up하여 다음 획의 pen-down까지의 이동으로 발생하는 펜의 동작을 의미한다.

모서리 선분을 검출하기 위해서는 일반적인 모서리(corner) 검출 방법이 적용될 수 있으며, 일반적으로 영상 정합의 형태분석시 모서리는 매우 유용한 특징으로 알려져 있다.[11,12] 3장에서 기술할 스트링 정합 방법에서는 동일한 유형의 선분사이에서만 정합이 일어나도록 제한함으로써 모서리 특징이 스트링 정합의 복잡도를 효과적으로 줄일 수 있도록 하였다. 본 논문에서는 모서리 선분의 검출을 위하여 Rosenfeld-Johnston 알고리즘을 적용하였다.[12] 이 방법은 디지탈 곡선상에서 의미있는 곡률의 최대치를 발견하기 위하여, 가변적 크기의 벡터를 생성하고 벡터내적 계산에 의해 구하여진 cosine 값들의 열에서 국소적인 최소값이나 최대값을 검출하여 모서리로 정의하는 방법이다.

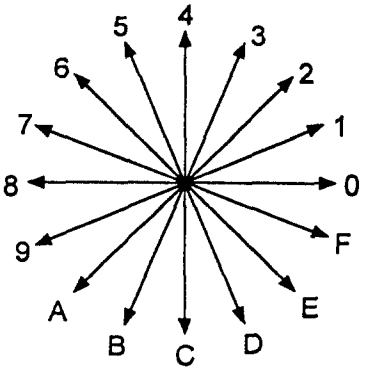


그림 4. 16방향 코드
Fig. 4. 16-directional code.

입력문자와 원형자소(prototype grapheme)는 방향값에 따라 정의한 기호들로 구성된 스트링으로 표현된다. 이러한 기호의 배정은 생성된 선분들마다 1개의 기호씩 이루어지며, 선분의 유형이 다르면 동일한 방향값일지라도 다른 기호들을 사용하여 선분의 유형을 구별할 수 있도록 하였다. 선분의 3가지 분류유형 중에서 일반 선분의 방향성을 표현하기 위한 기호들의 집합 V_{NL}은 다음과 같으며, 각 기호가 의미하는 방향성은 그림 4와 같이 정의된다.

$$V_{NL} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F \}$$

가상 선분의 방향성 표현을 위한 기호들의 집합 V_{VL}은 다음과 같으며, 각각의 기호가 의미하는 방향값은 각 기호에서 '*'을 제거한 V_{NL}의 기호와 동일하다.

$$V_{VL} = \{ 0^*, 1^*, 2^*, 3^*, 4^*, 5^*, 6^*, 7^*, 8^*, 9^*, A^*, B^*, C^*, D^*, E^*, F^* \}$$

모서리 선분의 방향성 표현을 위한 기호들의 집합 V_{BL}은 다음과 같으며 각각의 기호가 의미하는 방향값은 각 기호에서 '′'을 제거한 V_{NL}의 기호와 동일하다.

$$V_{BL} = \{ 0', 1', 2', 3', 4', 5', 6', 7', 8', 9', A', B', C', D', E', F' \}$$

그러므로 어떤 문자의 스트링표현을 위하여 정의된 알파벳(alphabet) V는 다음과 같다. 기호 '△'은 입력문자의 끝을 의미하기 위해 사용한다.

$$V = V_{NL} \cup V_{VL} \cup V_{BL} \cup \{\triangle\}$$

2. 제안하는 스트링 정합 알고리즘

Ⅲ.1의 방법으로 생성된 두 스트링의 비교를 위해서 표 3과 같은 4 가지 스트링 편집 연산이 정의되어졌다. 이 표에서 A와 B는 비교되어질 스트링을 의미하며, a와 b는 스트링의 속한 원소로서 $a \in A$, $b \in B$이고, λ은 공백스트링(null string)을 나타낸다. 이러한 스트링 편집 연산은 비교되어지는 스트링상의 기호 유형에 따라 적용여부가 결정되며, 이러한 편집연산자의 적용조건은 표 3과 같다.

표 3에서 $VType(k)$는 기호 k의 선분유형을 표 2의 3가지 선분유형으로 분류하는 함수이다. 표 4의 비교순서 규칙은 다음에 비교되어질 스트링의 기호를 결정하며 각 연산자의 편집특성에 근거하여 정의되었다. 제안하는 스트링 편집은 입력을 기준으로 하여 원형을 편집하는 것으로 가정하였는데, 예를들어 표 4의 삽입편집은 비교되는 원형의 기호위치는 변화되지 않고 입력의 위치만 1이 증가되게 되므로 원형에 기호가 삽입되는 동작을 나타내고 있다.

표 3. 편집연산의 적용 조건 정의

Table 3. Definition of application condition for editing operation.

편집연산	적용 조건
교환 ($a \rightarrow b$)	$VType(A<i>) \neq VType(B<j>)$
삽입-a ($\lambda \rightarrow b$)	$VType(A<i>) \neq VType(A<i-1>)$ and $VType(B<j>) = VType(B<j-1>)$
삽입-b 혹은 자소분할 ($\lambda \rightarrow b$)	$VType(A<i>) = $ 가상유형 and $B<j>$ = "Δ"
삭제 ($a \rightarrow \lambda$)	$VType(A<i>) = VType(A<i-1>)$ and $VType(B<j>) \neq VType(B<j-1>)$

표 4. 기호의 비교 규칙 정의

Table 4. Definition of rules for symbol comparion.

편집연산	비교순서 (현재) → (다음)
교환	$(i, j) \rightarrow (i+1, j+1)$
삽입-a	$(i, j) \rightarrow (i+1, j)$
삽입-b (자소분할)	$(i, j) \rightarrow (i+1, j)$
삭제	$(i, j) \rightarrow (i, j+1)$

흘림체 한글 필기로 부터 자소를 구분하는 일은 자소구분의 특징을 정의하는 것이 매우 어렵기 때문에 인식과정 중에 후보 자소분할점을 생성하고 타당성을 검증하는 내부분할 과정을 일반적으로 사용한다.[1,14] 본 인식방법에서도 이와같은 내부분할 과정을 이용하여 자소를 분할하는데, 표 3의 삽입-b 연산은 이러한 목적을 위하여 정의되었다.

표 3의 적용조건에서 보듯이 삽입-a와 삭제 연산은 적용 조건이 상반되므로 구별되고, 나머지 연산의 적용조건은 경우에 따라서 동시에 만족될 수 있다. 본 논문에서는 매 편집 단계마다 적용 가능한 편집 연산의 결정이 유일하게 이루어질 수 있도록 표 5와 같은 적용 우선 순위를 정의하였다.

표 5. 연산자 적용의 우선 순위에 관한 정의
Table 5. Definition of priority in operator application.

교환	순위높음
삽입-b	↕
삽입-a 혹은 삭제	순위낮음

그러므로 표 2의 적용조건과 표 5의 우선 순위의 정의에 의하여 매 편집단계마다 유일한 편집연산자가 결정되고, 전체적으로 1개의 편집경로에 의한 스트링 편집이 가능하다.

각 편집 연산자의 소요 비용 계산 중에서 교환 연산의 비용 $\gamma(a \rightarrow b)$은 다음과 같이 정의된다. 아래 식에서 $O_n(j)$는 기호 j가 의미하는 방향값을 나타낸다.

$$\gamma(a \rightarrow b) = |O_n(a) - O_n(b)| \ / \ MD$$

위식에서 MD는 최대의 특징값(방향값) 차이를 의미하는데 본 정합방법은 16방향에 의해 선분의 방향성이 결정되기 때문에 최대의 특징값 차이는 8이 되고, 예를들자면 방향값이 0과8 혹은 15와 7일때 해당한다. 그러므로 $\gamma(a \rightarrow b)$는 0과 1사이의 값을 갖게 된다. 삽입 혹은 삭제 연산의 경우는 비교 대상인 한 쪽 스트링의 기호가 존재하지 않는 경우의 연산이므로 실험적으로 구한 편집비용값으로 0.5를 사용하였다.

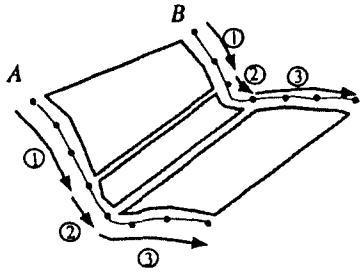


그림 5. 편집연산에 따른 편집경로
Fig. 5. editing path according to editing operation.

제안하는 편집 방법은 표 2에서 정의한 3 가지의 선분유형을 고려하므로, 유형을 고려하지 않은 편집연산에 비하여 보다 적은 계산량을 갖게 된다. 즉, 비교하고자 하는 기호의 유형이 동일한 경우에만 교환연산이 발생하고 그렇지 않을 경우는 삽입 혹은 삭제 연산이 발생하도록 정의되었다. 예를들어 그림 5는 어떤 두 스트링 A와 B사이의 편집경로를 나타내고, 이 그림에서 스트링의 ①과 ③부분은 일반선분들이 모여있는 유형이 동일한 부분이고 ②는 모서리 선분 1개로 이루어진 부분이라고 가정하자.

이때, 유형 동일부의 특징은 동일부를 구성하는 선분들이 동일부내의 어떤 위치에 존재하는가 보다는 동일부 전체의 평균적인 특징값과 동일부의 길이(선분의 갯수)에 의해서 결정된다. 왜냐하면 선분의 특징값이 급격히 변화하는 부분은 이미 분류되어 동일 유형부 내에는 서로 유사한 특징값을 갖는 선분만이 모여있기 때문이다.

표 3의 스트링 편집연산은 교환, 삽입-a, 삽입-b, 삭제의 4 가지로 이루어지는데, 교환연산은 선분의 특징값 비교, 나머지 3 가지 연산자들은 스트링의 길이 비교와 자소분할의 목적으로 사용되었다. 제안하는 정합방법에 따르면 그림 5의 경우에 A의 ①과 B의 ①, A의 ②과 B의 ②, A의 ③과 B의 ③부분 사이에서만 정합이 이루어진다. 왜냐하면 선분유형이 동일하지 않을 경우에는 표 3의 적용조건에 의해서 삽입과 삭제연산을 수행하며 동일 선분유형이 발견되기를 기다리기 때문이다. 그림 5에서의 정합의 진행은 A의 ②는 B의 ②의 교환연산이 수행되어야만 A의 ③과 B의 ③사이의 정합으로 넘어간다.

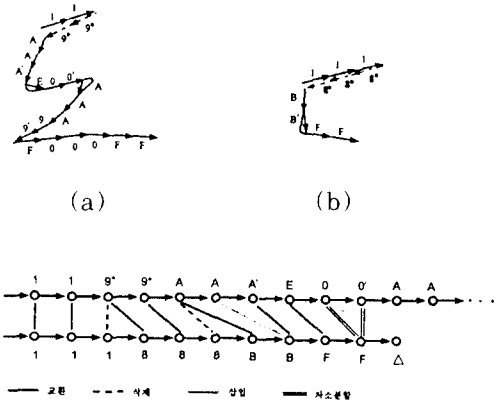


그림 6. 스트링 A와 스트링 B의 정합 과정
(a) 입력문자의 스트링 A (b) 원형자소의 스트링 B (c) 정합과정
Fig. 6. Matching process between string A and B.
(a) input string A (b) prototype string B (c) matching process.

예를들어 두 스트링의 정합과정 예를 보이면 그림 6과 같다. 그림 6의 (a)와 (b)는 각각 입력과 원형 스트링을 의미하고, (c)는 이 두 스트링 사이의 정합 과정을 보여준다.

그림 6(c)의 정합과정에 따라 그림 7과 같은 초성 자소 후보들이 발생한다. 그림 7에서 ○ 은 생성된 후보 자소에 따른 입력 문자의 자소 분할점을 나타내며, 또한 그림 6의 삽입-b(자소분할)연산이 적용된 스트링 A의 기호 위치를 의미하기도 한다. 그림 6에서 삽입-b연산이 두번 적용되었기 때문에 그림 7은 두 개의 후보자소를 나타내고 있다. 그림 7에서 초성 자소의 분할 후에 남겨진 입력 스트링 부분은 중성 자소 원형과 정합을 수행하게 되며, 중성 자소와의 정합 후에도 남겨진 부분이 있으면 종성 자소 원형과 계속적인 정합을 시도하게 된다.

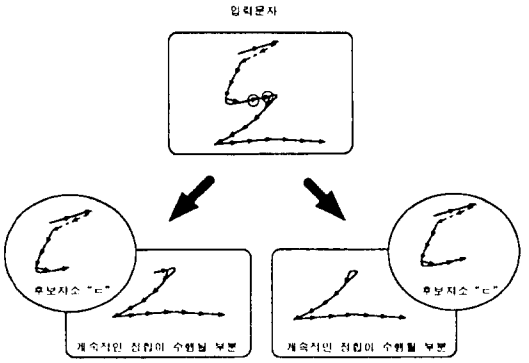
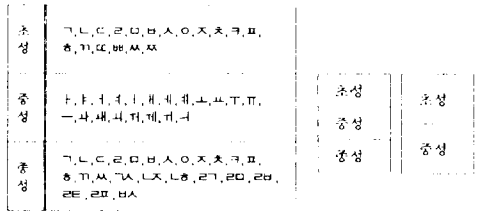


그림 7. 가능한 초성자소의 생성
Fig. 7. Generation of initial consonants.



(a) (b)

그림 8. 한글의 2방체구조
(a) 자소의 분류
(b) 가능한 2가지의 글자구조
Fig. 8. Two-type structure of Hangul.
(a) Classification of graphemes
(b) Two kinds of character structure.

생성된 자소들로 부터 글자를 생성하는 방식은 그림 8과 같은 한글의 2방체 구조를 사용하였다. 이와 같은 방식은 Ⅵ장의 후보문자공간의 트리(tree)구조를 간략화시킬 수 있다. 즉, 초성자음 다음에는 중성 모음만이 이어질 수 있고, 중성 모음 다음에는 종성자음이 올 수 있으므로 후보문자 공간의 깊이 3을 갖는 트리로 표현된다.

3. 원형자소의 생성

한글인식의 경우에 각 문자를 독립적인 개개의 클래스(class)로 정의하고 인식을 시도하기에는 분류해야할 클래스의 수가 너무 많고 클래스간의 유사성때문에 효과적인 적용 방법을 찾기 힘들다. 그런 이유로 한글인식(특히 흘림체 한글)의 문제는 한 문자를 이루는 자소들로 구분하고 그것들을 결합하므로써 문자를 인식하는 방법이 일반적으로 사용되어 왔다.[1-7] 그러므로 문자인식은 입력문자를 학습에 의해 구성된 원형자소와 정합하고, 자소 정합 결과의 조합에 의해 이루어진다. 원형자소의 생성은 자신이 속한 클래스 정보를 갖는 자소 샘플들에 의해 이루어진다. 학습은 이 자소 샘플 데이타베이스에서 1 샘플씩 순서대로 행해진다. 학습과정 중 입력된 자소가 해당 클래스의 원형자소군에 더해지는 경우는 다음과 같다.

(1) 입력자소가 갖고 있는 해당 클래스로 인식되지 않는 경우

(2) 정인식되었으나 1후보 자소가 2후보 자소와 극미한 차이를 갖는 경우

위의 (1)과(2) 경우가 아닐 때는 원형자소군에 변화없이 다음 샘플로 넘어간다. 이러한 학습과정은 학습자소 샘플들을 정인식하기 위해 더 이상의 원형자소 생성이 필요하지 않을 때까지 반복하여 수행된다. 학습과정 중에 자소를 분류하는 과정은 인식에서의 분류 방법과 동일하게 이루어진다.

4. 후보문자의 생성과 인식 문자의 결정

입력 문자는 Ⅲ장에서 설명한 자소 정합 방식에 의하여 인식된다. 인식과정 중의 정합은 자소를 단위로 이루어지며, 정합의 결과로서 편집비용과 후보 자소가 생성된다. 얻어진 후보 자소들은 별도의 장소에 저장되었다가 후에 선택되어 그 후보자소에 연결되는 자소를 결정하기 위한 계속적인 정합을 수행한다. 이러한 자소 정합의 반복은 후보자소열을 생성하고 완전한 문자 인식 결과로 변환된다. 이러한 자소정합에 의한 인식과정은 트리구조로 쉽게 표현될 수 있는데, 트리의 각 노드는 생성된 후보 자소와 그에 따른 정합 비용을 저장하게 된다. 원형과의 정합순서는 초·중·종성의 순서로 자소가 필기되고 입력되기 때문에

정합되는 자소 원형도 초성의 원형들이 먼저 정합하고, 이 단계에서 계산된 결과에 따라 중성 원형, 그리고 종성 원형의 순으로 정합이 수행된다. 그러므로 후보 자소 공간을 표현하는 트리는 그림 9와 같이 높이 3의 트리구조로 표현된다.

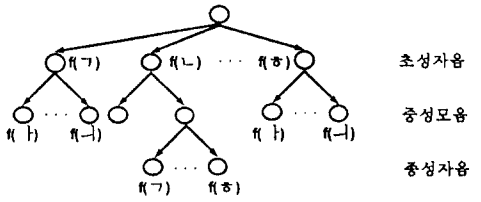


그림 9. 인식과정 중의 트리생성
Fig. 9. Tree generation in recognition process.

확장되어질 후보 노드의 선택은 트리의 단말 노드 중에서 최소의 편집비용을 갖는 것이 해당된다. 그리고 선택되어진 노드가 저장하고 있는 기존의 정합 결과를 바탕으로 입력문자의 나머지 부분을 원형과 정합하는 과정을 수행하게 된다. 이에따라 새로운 정합 결과가 얻어지며 이 결과는 트리구조의 확장을 가져온다. 이러한 과정의 반복으로 생성된 후보자소열로부터 후보문자를 생성하게 된다. 만약 현재까지 n개의 후보문자가 생성되었고 후보문자 i의 현재까지의 정합비용이 $Cost(i)$라고 표기할 때, 다음과 같은 절단값(pruning value) α 를 얻을 수 있게 된다.

$$\alpha = \min\ [Cost(1),\ Cost(2),\ \ldots,\ Cost(n\text{-}1),\ Cost(n)]$$

트리의 각각의 단말노드는 지금까지의 정합비용이 α 를 초과하지 않는 한 계속적인 정합을 수행할 수 있고, 그렇지 못한 경우는 해당 노드의 확장은 이루어지지 않는다. 트리 확장의 종료는 더 이상의 트리 구조의 확장이 불가능한 시점에서 발생되며 현재까지 생성된 후보 문자들을 정렬하여 인식결과로 출력하게 된다. 또한, 확장될 노드의 선택시 평가기준은 root 노드로 부터 현재 노드까지의 비용에만 의존하도록 하였고, 절단값 의 결정이 최적의 인식결과를 보장하도록 하였기 때문에 본 연구의 트리 확장에 의한 인식 문자의 탐색은 A*의 특성을 만족하면서 탐색의 효율을 갖는다.[9]

자소간 연결필기는 한글의 필기 특성에 근거할 때 가능한 경우와 그렇지 않은 경우를 정의할 수 있다. 탐색과정의 복잡도를 줄이기 위하여 본 연구에서는 자소간 연결필기가 불가능한 경우는 후보생성이 이루어지지 않도록 하였다. 그림 10은 초성과 중성에서의 자소간 연결 필기시에 가능한 연결 형태를 나타낸다.

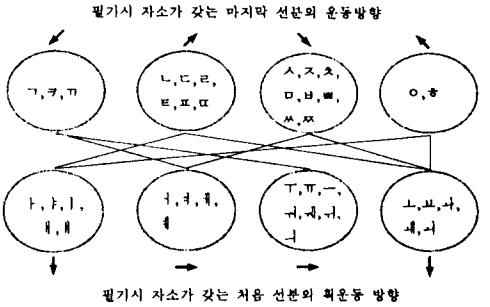
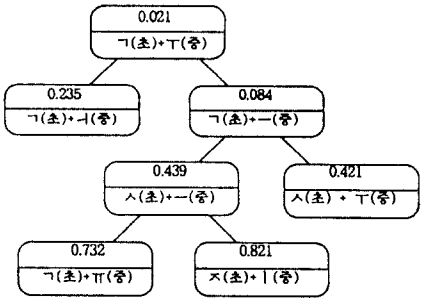


그림 10. 자소간 연결필기가 가능한 초성과 중성의 연결
Fig. 10. Possible concatenation between initial consonant and vowel.

트리 구조는 단말 노드들만을 표현하며 heap구조로 표현했다. heap구조 표현은 노드의 삽입과 삭제가 용이하며, 정해진 key 값에 의해 정렬된 형태를 유지하므로 후보문자 공간의 효율적인 표현을 제공받을 수 있다.[9,10] 그림 11은 heap 구조에 의한 탐색공간 표현의 예를 보여준다.



(a)

1	2	3	4	5	6	7
0.021	0.235	0.084	0.439	0.421	0.732	0.821

(b)

그림 11. Heap 구조에 의한 탐색공간의 표현
(a) Heap 의 이진트리 표현
(b) Heap 의 배열 표현
Fig. 11. Representation of search space using heap structure.
(a) Binary-tree representation of heap
(b) Array representation of heap.

Ⅳ. 실험 결과

본 인식시스템은 SUN Workstation IPC(10 mips)상에서 구현하였고 Tablet은 WACOM SD-422A 모델을 사용하였다. 구현된 인식시스템은 그림 12와 같다. 프로세스간의 통신은 2개의 pipe를 사용하여 이루어지도록 하였다. 사용자가 선택할 수 있는 메뉴는 흘림정도의 구분, 학습, 새화면 등으로 이루어져 있으며, 사용자는 학습버튼을 선택하고 초·중·종성의 자소구분과 필기문자에서 자소 구분 영역을 설정하여 즉석에서 학습을 수행할 수 있는 특징을 갖도록 구현되었다.

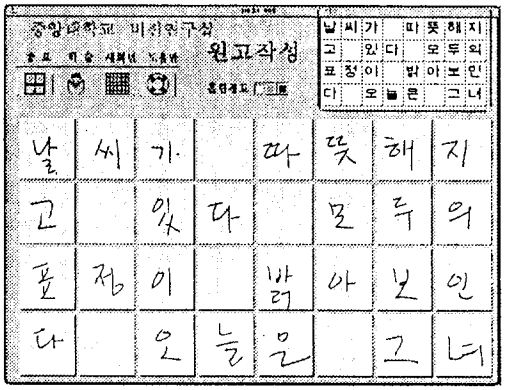


그림 12. 구현된 인식 시스템의 화면
Fig. 12. User interface of implemented recognition system.

1. 인식률측정

구현된 인식 시스템의 인식률 측정을 위하여 구성한 실험자료는 50명의 필기자로 부터 얻은 총 21,076 자을 사용하였다. 원형자소 생성을 위한 자소샘플 데이타베이스를 구성하기 위해서 본 인식시스템에서는 10명의 필기자료를 모으고 그 중에서 잘못 필기된 문자를 제외한 문자를 수작업으로 초·중·종성의 자소로 분류하여 총 12,722 자소를 얻어 사용했다. 얻어진 인식률은 표 6과 같다. 이 인식률은 흘림정도를 최대로 하여 절단값에 의한 탐색과정 축소를 최소로 한 경우의 인식률이다. 각 필기자들의 인식률 분포 상황을 나타내면 그림 13과 같다.

표 6. 21,076자에 대한 정인식률
Table 6. Recognition rate to 21,076 characters.

1후보 정인식률	3후보 정인식률
86.36 %	89.23 %

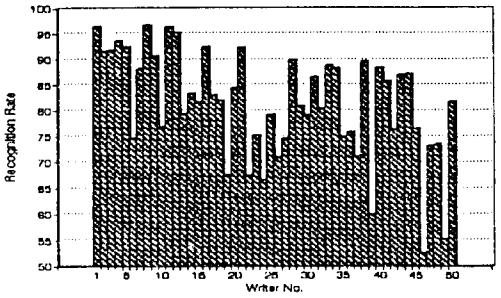


그림 13. 필기자들의 인식률 분포
Fig. 13. Distribution of recognition rate.

또한, 흘림정도 버튼 선택에 따른 인식률의 변화와 인식에 소요되는 시간을 나타내면 표 7과 같다. 흘림정도 선택 버튼은 인식과정 중에 정합되는 원형 자소의 수효에 관계가 있으며, 심함을 선택하면 약함의 경우에 비하여 보다 많은 원형이 정합에 참여하도록 구현하였다. 이와 같은 흘림정도의 구분은 처음 인식기를 대하는 필기자와 온라인 한글 필기에 능숙한 사용자 모두에게 편리함을 제공하기 위해서 만들어졌다. 표 7의 인식시간은 필기자의 한글 필기 속도를 고려할 때, 펜정보 수집 프로세스로 부터 제공받은 필기 정보를 인식 프로세스가 충분히 인식할 수 있는 시간임을 실험적으로 확인하였다. 그러나 복잡한 구조의 글자를 인식하는 데에는 보다 긴 인식시간을 필요로 하기 때문에 필기자의 필기시간 동안에 인식결과를 제공하기 어렵다. 예를들면 '밝', '빨', '뚫' 등과 같은 글자를 들 수 있다. 그러한 경우에 본 시스템은 필기자가 현재의 복잡한 글자에 연이어 다음 글자를 필기하는 동안에 인식결과를 출력하도록 하고 있다. 그러므로 필기자는 인식기의 수행과정에 신경쓰지 않고 계속적인 필기를 진행할 수 있다.

표 7. 흘림정도 구분에 따른 인식률
Table 7. Recognition rate according to the button of cursive style.

	약 함	중 간	심 함
인식률	82.45 %	86.36 %	88.96 %
인식시간	0.93 초	1.75 초	3.32 초

2. 다른 방법과의 비교

우리가 구현한 인식 시스템은 스트링으로 문자를 표현하고 서로 다른 길이의 스트링상의 편집연산에 의해 가장 유사한 원형문자를 선택한다. 이와 같은 기호적 스트링 정합 방법으로 자주 사용되는 알고리

즘으로 Wagner와 Fisher의 방법을 들 수 있다.[15] 이 알고리즘을 간략한 설명은 다음과 같다.어떤 스트링 A 를 B 로 변환하기 위하여 스트링의 기호들상에서 이루어 지는 기본적 연산은 교환($a \rightarrow b$), 삽입($\lambda \rightarrow b$), 삭제($a \rightarrow \lambda$) 연산이며, 이때 $a \in A$, $b \in B$ 이고 은 null string이다. A와 B사이의 거리는 다음과 같이 DP(dynamic programming)에 의한 누적거리로 정의하였다.

$$D(i,j) = \min \begin{bmatrix} D(i-1,j-1) + cost(a \rightarrow b) \\ D(i-1,j) + cost(a \rightarrow \lambda) \\ D(i,j-1) + cost(\lambda \rightarrow b) \end{bmatrix}$$
$$1 \le i \le |A|, \quad 1 \le j \le |B|$$

DP에 의한 탐색은 A*알고리즘에 의해 진행되며 절단값에 의해 전망없는 경로의 탐색은 제거하도록 하였다.[9,10] 여기서 사용되는 비용함수는 우리의 인식 시스템과 동일하게 구현되었다. 원형자소 생성을 위한 학습은 각 자소마다 30개의 샘플들을 모아서 구성한 데이타베이스를 만들어 수행하였고 생성된 원형자소의 갯수는 13개를 넘지 않도록 하였다. 인식 실험은 10,712 자소를 대상으로 자소인식률을 측정하였으며, 인식실험 비교는 표 8과 같다.

표　8. 자소인식률 비교
Table 8. Comparison of recognition rate.

제안하는 인식방법		DP를 사용한 인식방법	
1후보 성인식률	인식소요시간	1후보 성인식률	인식소요시간
92.54	0.03	87.14	0.54

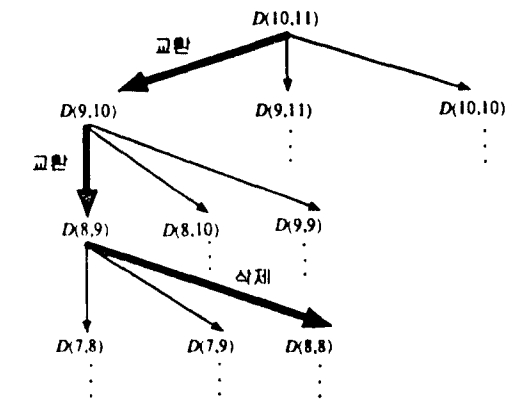


그림 14. 제안하는 방법과 DP와의 편집과정 비교
Fig. 14. Comparison of editing process between the proposed and DP method.

제안하는 정합방법은 선분유형에 근거하여 편집연산자를 정하므로 보다 적은 탐색시간을 갖는다. 예를 들어 그림 6의 두 스트링 사이의 편집과정을 DP를 사용하여 나타내면 그림 14와 같다. 그림 14에서 진한 선으로 나타낸 것은 제안하는 방법을 적용할 때, 편집과정에 따른 편집 경로를 의미한다.

그림 14에서 보듯이 DP정합방법은 매 단계에서 교환, 삽입, 삭제 연산을 모두 적용하여 3가지의 새로운 정합 단계를 생성한다. 이러한 과정은 더 이상의 편집이 필요하지 않을 때까지 재귀적으로 반복된다. 반면에, 제안하는 방법은 Ⅲ.2 절의 정의에 따라 어떤 정합 단계에서 이루어질 편집연산을 선택적으로 행하며 전체적으로 1 개의 편집경로를 생성한다. 표 8의 결과는 이러한 방법론적인 차이에 근거하여 설명될 수 있다. 제안하는 방법은 인식과정 중에 거의 일정한 분량의 계산량을 요구하는 특성을 갖는데, 이러한 결과는 표 8과 그림 14를 통하여 설명될 수 있다. DP정합의 사용시에 보다 신속한 인식을 위해서 두가지 보완 방법을 생각할 수 있다. 첫째는 인식에 앞서 대분류 과정을 통해 후보를 줄여 약간의 인식률 저하를 감수하는 방법이다. 둘째로는 DP정합을 위한 H/W를 디자인하여 해결하는 경우가 해당한다.[13,14,16,17]

Ⅴ. 결 론

본 논문에서는 홀림체 한글 필기의 온라인 인식시스템을 설명하였다. 홀림체 한글 필기의 인식은 글자 형태의 심한 변형으로 인하여 인식과정이 복잡하여지고, 그로 인한 고속의 인식속도이 어려운 것으로 알려져 왔다. 본 논문에서는 이러한 어려움을 해결하는 방법으로서 스트링 정합방법에 기반한 새로운 정합방법을 제안하였으며, 이 방법에서는 스트링 편집과 비교순서규칙에 의한 고속 인식이 가능하다. 그리고 구현되어진 인식시스템은 사용자 편의를 위하여 연속적인 필기와 인식이 가능하도록 구성하였다. 실험결과, 제안하여 구현되어진 인식시스템은 한글 홀림체의 연속적인 필기의 인식에 적합함을 알 수 있었다. 제안하는 인식방법은 연속적인 온라인 필기와 고속 인식이 가능하므로 폭넓은 응용이 기대된다. 계속적인 연구가 진행되어야 할 부분으로서는 기울어진 한글 필기의 경우에 기울어짐을 흡수할 수 있는 기법과 한글뿐만 아니라 영어와 숫자도 함께 인식할 수 있는 방법론의 연구가 필요하다.

參 考 文 獻

[1] 권오성, 권영빈, "선분정합에 의한 홀림체 온

라인 한글 인식", 인지과학, vol.3, No.2, pp.271-289, 1993

[2] 김태균, 이은주, "한글에 적합한 획 해석에 의한 연속 필기 한글의 on-line 인식에 관한 연구", 한국 정보과학회 논문지, Vol.15, No. 3, pp.171-181, 1988

[3] 박재성, 김성신, 김태균, "변형된 DP매칭과 구조해석적 방법을 이용한 흘려 쓴 한글 인식", 제 1회 문자인식 워크샤 논문집, pp. 181-184, 1993

[4] 성태진, 방승양, "문자조합 규칙 학습에 의한 한글 온라인 필기 인식기의 설계", '91 년 가을 학술발표논문집, 한국정보과학회, pp.223-226, 1991

[5] 신봉기, 김진형, "은닉 마르코프 모델을 이용한 온라인 한글 인식", 제1회 문자인식 워크샤 논문집, pp.189-194, 1993

[6] 이성환, 박희선, "한글 인식의 사례 연구(최근 5년 동안의 연구결과를 중심으로)", 제1회 문자인식 워크샤 논문집, pp.3-46, 1993

[7] 전병환, 구본석, 김성훈, 김재희, "퍼지이론을 응용한 펜 컴퓨터에서의 On-Line 획 인식 기법", 91년 춘계학술발표논문집, 한국퍼지시스템연구회, pp.168-176, 1991

[8] S.-T. Bow, *Pattern Recognition and Image Preprocessing*, Marcel Dekker, New York, 1992.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, 1992.

[10] S. Edelman, et al., "Reading Cursive Handwriting by Alignment of Letter Prototypes", *Int. J. of Computer Vision*, Vol.5, No.3, pp.303-331, 1990.

[11] C. A. Higgins and D. M. Ford, "A New Segmentation Method for Cursive Script Recognition", *Int. Workshop on Frontiers in Handwriting Recognition, Chateau de Bonas*, France. Sep.,1991.

[12] H.-C. Liu and M. D Srinath, "Corner Detection from Chain-code", *Pattern Recognition*, Vol. 23, No. 1/2, pp.51-68, 1990.

[13] C. C. Tappert and H. H. Jeanty, *A Study of Several Accuracy-Improvement methods for Hanwriting Recognition Syatem*, IBM Research Report RC 15373, Nov. 1990.

[14] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition", *IEEE Trans. on PAMI*, Vol.12, No.8, pp. 787-803, Aug. 1990.

[15] R. A. Wagner and M. J. Fischer, "The string to string correction problem", *J. ACM*, Vol. 21, pp. 168-173, 1974.

[16] T. Wakahara, Hiroshi Murase, Kazumi Odaka, "On-Line Handwriting Recognition", *Proc. of the IEEE*, Vol.80, No.7, pp.1181-1194, Jur. 1992.

[17] F. Nouboud and R. Plamondon, "A Structural Approach to On-Line Character Recognition: Systam and Application," in character & Handwriting Recognition (editor: P.S.P. Wang), World Scientific, Singapore, pp123-137, 1991.

著 者 紹 介

權 五 成(正會員)

1987年 한남대학교 전자계산학과 졸업(공학사). 1989年 중앙대학교 대학원 전자계산학과 졸업(이학석사). 1994年 8月 중앙대학교 대학원 전자계산학과 졸업(공학박사). 현재 중앙대학교 컴퓨터공학과 시간 강사. 주관심 분야는 컴퓨터비젼, 패턴인식(문자인식), 영상처리 등임.

權 寧 彬(終身會員)

1978年 아주대학교 전자공학과 전교수석졸업(공학사). 1981年 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 1986年 프랑스 파리 ENST 졸업(공학박사). 1986年 ~ 현재 중앙대학교 컴퓨터공학과 부교수로 재직중. 주관심 분야는 컴퓨터비젼, 영상처리, 패턴인식, 한글 처리 등임.