제스쳐 허용 전자 잉크 에디터의 개발

조미경[†] · 오암석[††]

요 약

전자 잉크 데이터는 스타일러스 펜을 주된 입력 도구로 사용하는 PDA 등과 같은 펜 기반 컴퓨터의 개발로 출현한 멀티미디어 데이터이다. 최근 들어 펜 기반 모바일 컴퓨터의 발전과 보급은 전자 잉크 데이터 처리 기술에 대한 필요성을 증가시키고 있다. 본 논문에서는 펜 제스쳐(pen gesture)를 허용하는 전자 잉크 에디터 개발에 필요한 기술들을 연구하였다. 제스쳐와 잉크 데이터는 펜 기반 사용자 인터페이스의 가장 큰 특징 중 하나이지만 아직 충분한 연구가 되지 않았다. 본 논문에서는 펜 제스쳐 구분을 위한 새로운 제스쳐 인식 알고리즘과 제스쳐 명령을 수행하기 위한 잉크 데이터의 분할 방법이 제안되었으며 제안된 방법들을 이용하여 제스쳐를 허용하는 전자 잉크 에디터 *GesEdit*를 개발하였다. 제스쳐 인식 알고리즘은 입력된 획의 여덟 가지 특징에 기반하고 있으며 전자 잉크 데이터를 *GC*(Gesture Components) 단위로 분할하는 방법은 볼록 껍질(convex hull)과 입력 시간을 사용하였다. 열 명의 피실험자에 의해 수행된 다양한 실험 결과 아홉 가지 제스쳐들은 평균 99.6%의 인식률을 보여 주었다.

Development of Gesture-allowed Electronic Ink Editor

Mi-Gyung Cho[†] and Am-Suk Oh[††]

ABSTRACT

Electronic ink is multimedia data that have emerged from the development of pen-based computers such as PDAs whose major input device is a stylus pen. Recently with the development and supply of pen-based mobile computers, the necessity of data processing techniques of electronic ink has increased. Techniques to develop a gesture-allowed text editor in electronic ink domain were studied in this paper. Gesture and electronic ink data are a promising feature of pen-based user interface, but they have not yet been fully exploited. A new gesture recognition algorithm to identify pen gestures and a segmentation method for electronic ink to execute gesture commands were proposed. An electronic ink editor, called *GesEdit* was developed using proposed algorithms. The gesture recognition algorithm is based on eight features of input strokes. Convex hull and input time have been used to segment electronic ink data into GC(Gesture Components) unit. A variety of experiments by ten people showed that the average recognition rate reached 99.6% for nine gestures.

Key words: 펜 기반 컴퓨터, 전자잉크 데이터, 펜 제스쳐, 제스쳐 인식, 잉크 데이터 분할

1. 서 론

스타일러스 펜을 주요 입력 도구로 사용하는 PDA(Personal Digital Assistants)나 펜 기반 컴퓨터의 발전과 보급으로 펜 컴퓨팅(pen computing)이 사용자들에게 친숙해지고 있다. 펜 기반 컴퓨터에서 가장 이상적인 데이터 입력은 종이에 연필을 가지고 쓰듯이 글자나 숫자, 표, 그림 등의 데이터를 스크린 위에 스타일러스 펜으로 자유롭게 입력하는 방식이다. 이처럼 펜으로 입력한 모든 형태의 데이터를 전자 잉크 데이터(electronic ink data)라고 한다[1,2]. 그리고 펜으로 입력한 데이터를 문자 인식기를 이용하여 아스키코드로 변환하지 않고 전자 잉크 데이터

본 연구는 2003년도 동명정보대학교 학술 연구비 지원으로 수행되었음.
접수일 : 2003년 2월 20일, 완료일 : 2003년 4월 9일
[†] 정회원, 동명정보대학교 멀티미디어공학과 전임강사
[††] 정회원, 동명정보대학교 멀티미디어공학과 부교수

형태로 처리하는 방식을 전자 잉크 데이터 영역에서의 데이터 처리라고 한다.

전자 잉크 데이터 영역에서의 데이터 처리가 사용자에게 주는 가장 큰 장점은 입력할 데이터 형태에 제한을 주지 않는다는 것이다. 사용자는 단순한 문자에서 수식, 그림 등에 이르기까지 표현하고자하는 어떤 형태의 정보든지 펜으로 자유롭게 표현할 수 있다. 이것은 문자 인식기가 인식할 수 있는 제한된 문자, 숫자만을 입력할 수 있도록 하는 전통적인 시스템과 비교할 때 매우 큰 장점이다. 하지만 전자 잉크 데이터를 사용하기 위해서는 시스템에서 전자 잉크 데이터의 편집, 검색 기능이 제공되어야 하고 다양한 플랫폼에서 전자 잉크 데이터를 사용하기 위한 표준 교환 포맷이 개발되어야 한다[1-3].

PDA와 같은 펜 기반 모바일 컴퓨터에서 현재 많이 사용하고 있는 데이터 입력 방식은 소프트 키보드 방식과 온라인 문자 인식기를 이용하는 방식이다. 소프트 키보드 방식은 화면에서 보여주는 소프트 키보드에서 원하는 문자를 펜으로 태핑하므로 데이터를 입력하기 때문에 펜의 기능을 살리지 못한다. 온라인 문자 인식기의 인식률은 많이 개선되었지만 여전히 인식 문제가 남아 있으며 입력할 수 있는 문자의 종류도 매우 제한적이다. 또한 인식 오류가 발생하면 잘못 인식된 문자를 지우고 다시 입력해야 되는 불편함이 있다. 그럼에도 불구하고 소프트 키보드와 문자 인식기를 많이 사용하는 이유는 입력 데이터를 아스키코드로 변환하면 일반적인 컴퓨터에서와 동일하게 데이터를 처리할 수 있기 때문이다.

최근 들어 PDA와 같은 모바일 컴퓨터의 발전으로 펜 기반 사용자 인터페이스 개발에 대한 요구가 증가하면서 전자 잉크 데이터의 검색 및 편집 등 잉크 데이터 처리에 대한 다양한 연구들이 진행되고 있다[1-3]. 또한 전자 잉크 데이터를 사용할 수 있는 응용 프로그램들도 나오고 있다[4,5]. 하지만 잉크 데이터로 구성된 문서를 편집하도록 해 주는 에디터는 개발되지 않았다. 한 예로 포켓 PC의 "notes"라는 프로그램은 잉크 데이터 문서를 작성하도록 해 주지만 편집 기능은 제공하지 않는다.

본 논문의 목적은 펜 기반 컴퓨터에서 가장 자연스러운 데이터 형태인 전자 잉크 데이터를 편집하는 에디터를 개발하는 것이다. 특별히 PDA에서 다양한 편집 명령을 제스쳐(gesture) 형태로 제공해 주는 전자 잉크 에디터를 개발하고자 한다. 최근 PDA와 같은 펜 기반 휴대용 컴퓨터의 등장으로 펜 기반 사용자 인터페이스에 대한 연구가 활발해지면서 제스쳐에 대한 연구도 많이 이루어지고 있다[5-9].

제스쳐를 허용하는 전자 잉크 에디터를 구현하기 위해서는 두 가지 기술이 개발되어야 한다. 첫째는 펜 제스쳐를 잉크 데이터와 구별하여 인식할 수 있는 제스쳐 인식 알고리즘의 개발이고 둘째는 제스쳐 명령을 수행하기 위해 전자 잉크 데이터를 제스쳐를 수행할 수 있는 논리적인 단위로 분할하는 방법을 개발하는 것이다. 본 논문에서는 획의 여덟 가지 특징을 이용한 새로운 제스쳐 인식 알고리즘과 볼록 껍질(convex hull)과 획의 입력 시간을 이용하여 전자 잉크 데이터를 제스쳐 요소 단위(GC: *Gesture Components*)로 분할하는 방법을 제안하였다. 또한 제안한 방법들을 이용하여 PDA용 전자 잉크 에디터 *GesEdit*를 개발하였다.

논문의 구성은 다음과 같다. 2장에서는 PDA에서 사용되고 있는 제스쳐의 현황과 제스쳐 설계 및 인식 알고리즘에 관련된 최근 연구들을 살펴볼 것이다. 3장에서는 본 논문에서 제안한 제스쳐의 종류들과 새로운 제스쳐 인식 알고리즘에 대해 설명하고 4장에서는 전자 잉크 데이터를 제스쳐 수행 단위인 *GC* 단위로 분할하는 방법을 제안할 것이다. 5장에서는 구현 및 실험 결과를 살펴보고 마지막으로 6장에서 결론을 맺을 것이다.

2. 제스쳐 관련 연구

펜 제스쳐는 하나의 명령을 수행시킬 수 있는 표시(marks)로 단일 획(single stroke) 혹은 다중 획(multiple strokes)으로 정의할 수 있다[5,6]. Christan Long의 연구 조사에 의하면 PDA 사용자들은 많은 제스쳐들이 개발되어 다양한 애플리케이션에서 사용되기를 원하는 것으로 발표되었다[5,6]. 제스쳐 기능을 제공하는 Pilot과 Newton 사용자들을 대상으로 한 실험에서 대부분의 사람들이 제스쳐가 매우 강력한 기능이라고 대답했다. Newton 사용자들의 경우 70% 이상의 사람들이 매우 자주 사용하는 것으로 보고되었다[6].

이제까지 개발된 제스쳐 지원 시스템들을 살펴보면 영어권의 경우 GRANDA[8], Quill[6], Newton,

Pilot 등이 있고 우리나라의 경우 온라인 한글 인식기인 디오펜[9]에서 제공하는 제스쳐 등이 있다. 표 1은 현재까지 개발된 시스템들과 본 연구에서 개발한 *GesEdit*가 제공하는 제스쳐의 종류와 특성들을 비교해서 보여준다. Quill과 GRANDA는 제스쳐 설계 툴로 설계한 제스쳐의 인식률을 측정하면서 사용자가 사용하기 쉽고 오래 기억할 수 있는 제스쳐를 설계하도록 돕는 시스템이다. Newton은 Newton PDA에서 사용되는 제스쳐들을 디오펜은 국내에서 가장 많이 사용되는 온라인 한글 인식기인 디오펜에서 제공하는 제스쳐들을 보여준다.

시스템들을 살펴보면 Quill과 GRANDA는 각각 15개와 11개의 제스쳐를 제공하는데 표 1에는 그 중 대표적인 것들을 보여준다. 제스쳐의 인식률은 각 각 95.4%, 100%로 알려져 있다[7,8]. 이 시스템들은 제스쳐 설계 툴이기 때문에 실제 제스쳐 기능은 구현되어 있지 않다. 반면 제스쳐 명령이 구현되어 있는 Newton과 디오펜에서 제공하는 제스쳐의 인식률은 공식적으로 발표된 자료가 없다(△로 표기). 하지만 항상 정확한 인식을 해 주는 것은 아니라고 알려져 있다[6]. 디오펜의 경우 11개의 제스쳐를 제공하지만 표에서 보는 바와 같이 비슷한 모양의 제스쳐들이 전혀 연관이 없는 명령을 수행하므로 사용자에게 혼란을 줄 수 있다.

기존의 시스템들과 *GesEdit*의 가장 큰 차이점은 제스쳐 수행 환경이다. 제스쳐 기능이 구현되어 있는 기존의 시스템들은 모두 펜으로 입력한 데이터들을 아스키코드로 변환한 상태에서 제스쳐 명령을 수행한다. 현재까지 전자 잉크 데이터 영역에서 제스쳐를 허용하면서 잉크 데이터를 편집할 수 있도록 도와주는 응용 프로그램은 없다. 하지만 본 논문에서 개발한 *GesEdit*의 경우 전자 잉크 데이터 환경에서 제스쳐를 사용할 수 있도록 개발되었으며 제스쳐 인식률도 99.6%로 매우 실용적이다.

전자 잉크 환경에서 제스쳐 명령을 수행하는 것은 아스키 문자 상태에서 제스쳐 명령을 수행하는 것보다 어려운 문제이다. 왜냐하면 제스쳐 명령을 수행하기 위해 전자 잉크 데이터를 분할하는 기술이 요구되기 때문이다. 4장에서 제스쳐 명령을 수행하기 위해 전자 잉크 데이터를 분할하는 방법에 대해 다룰 것이다.

제스쳐 인식에 대한 연구는 1990년대 초반부터 최근까지 이루어지고 있으며 몇 가지 연구 결과들이 발표되었다[5-8]. 제스쳐 인식 알고리즘은 뉴럴 네트워크 기반 알고리즘과 특징 기반 알고리즘으로 분류된다. 뉴럴 네트워크를 이용한 방법이 일반적으로 더 높은 인식률을 주지만 수백 혹은 수천의 훈련 샘플을 요구한다. 특징 기반 제스쳐 인식 방법으로 대표적인 것은 Rubine이 제시한 알고리즘으로 획의 초기 방향, 길이, 바운딩 박스(bounding box)의 크기 등 13가지의 특징들과 통계적 분석에 의해 제스쳐를 인식하는 방법으로 이후에 개발된 다른 제스쳐 인식기의

표 1. 제스쳐 제공 시스템들의 비교

	특성 / 시스템	대표적인 제스쳐 종류들						개수	구현 여부	사용 환경	평균 인식률
기존의 시스템	*Newton*	Delete	W	Line Insert		Cursor Move	•	5	○	아스키 문자	△
		Select	○	Words Insert							
	GRANDA	Delete	—	Swap		Space		11	×	×	100%
		Insert		Join		Move					
	Quill	Delete		Zoom out		Zoom in		15	×	×	95.4%
		Select		Undo		Redo					
	디오펜	Space	—	Tab		Help		11	○	아스키 문자	△
		Back-space	—	Enter		R Button					
제안된 시스템	*GesEdit*	Insert		Delete		Join		9	○	잉크 데이터	99.6%
		Swap		Line Insert		New Line					

기본 모델로 사용되었다[8].

PDA는 데스크탑 컴퓨터에 비해 하드웨어적인 제약조건이 있기 때문에 제스쳐 인식 알고리즘이 간단해야 한다. 뉴럴 네트워크 인식 알고리즘에 비해 Rubine이 제시한 제스쳐 인식 알고리즘이 단순하지만, 제스쳐에 대한 수십 개의 샘플들에 대한 특징 값을 계산한 후 통계적 분석을 이용해야 하기 때문에 하드웨어 제약이 있는 PDA용으로 구현하기에는 비효율적이다.

3. 제안된 제스쳐 설계 및 인식 알고리즘

3.1 제스쳐 설계

제스쳐 설계에 있어 중요한 고려 사항은 첫째 제스쳐들을 정확하게 인식할 수 있는가와 둘째 사용자가 쉽게 기억할 수 있는가 이다. 제스쳐는 정의된 일련의 명령을 수행하므로 잘못 인식된 제스쳐는 예기치 못했던 결과를 초래할 수 있다. 따라서 제스쳐의 정확한 인식은 매우 중요한 문제이다. 제스쳐를 얼마나 쉽게 기억할 수 있는 가도 문제가 된다.

제안된 제스쳐들의 모양은 오래 기억할 수 있도록 하기 위해 원고지 교정 부호에 기초하여 설계하였다. 원고지 교정 부호는 종이에 문서를 작성할 때 일반적으로 많이 사용하는 기호들이므로 사용자들에게 친숙하다는 점에 착안하였다. 비슷한 모양의 제스쳐들에 대한 인식률을 높이기 위해 획의 시작 위치를 달리하였다. 예를 들어 삽입과 삭제 제스쳐의 경우 모양에 대한 유사도가 높기 때문에 제스쳐가 시작되는 위치를 반대로 지정하였다. 표 2는 본 논문에서 제안하는 아홉 가지 제스쳐 종류와 기능들을 보여준다. 제스쳐의 종류는 문서 편집을 할 때 많이 사용되는 명령들을 중심으로 선정하였다. 표 2의 제스쳐 모양에서 검은 점은 획의 시작 위치를 의미한다.

3.2 제스쳐 인식 알고리즘

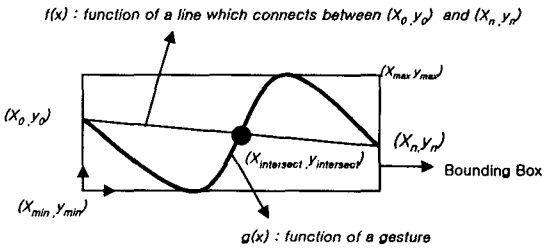
제스쳐 인식 알고리즘을 설명하기 전에 논문에서 개발한 전자 잉크 에디터 *GesEdit*를 살펴보면, 데이터 입력 도중 언제든지 제스쳐 명령을 입력할 수 있도록 하였다. 따라서 입력한 데이터가 잉크 데이터인지 아니면 제스쳐인지를 구별할 수 있어야 하는데 이를 위해 에디터는 베이스라인(baseline)을 가진다.

표 2. 제안된 펜 제스쳐들의 종류와 기능

제스쳐 종류	Insert	Delete	Join	Swap	Space	Line Insert	New Line	Line Up	Undo
제스쳐 모양									
제스쳐 기능	글자 (단어) 삽입	글자 (단어) 지우기	띄워진 글자 붙이기	글자앞뒤 순서 변경	공백 삽입	줄 사이의 새로운 줄 생성	줄 바꿈	줄 결합	수행한 명령 되돌리기

잉크 데이터는 두 개의 베이스라인 사이에 작성하도록 허용하는 반면 제스쳐는 베이스라인과 교차하여 입력하도록 하였다. 따라서 펜으로 입력된 획이 베이스라인과 만나는 교차점이 생기면 입력된 획이 제스쳐인지 아닌지를 검사하기 위해 제스쳐 인식 알고리즘이 수행된다.

제안한 제스쳐 인식 알고리즘은 제스쳐들을 구별할 수 있는 아홉 가지 특징들에 기반하고 있다. 먼저 제스쳐 인식을 위해 사용된 특징들을 살펴보면 그림 1과 같다.



$$f_1 = num(intersection(f(x), g(x)))$$

$$f_2 = sign(\int_{x_0}^{x_n}(f(x) - g(x))dx)$$

$$f_3 = (\int_{x_0}^{x_n}(f(x) - g(x))dx / ((x_{\max} - x_{\min}(\max(g(x)) - \min(g(x))))$$

$$f_4 = \sum_{i=0}^{n-1}(x_i - x_n)$$

$$f_5 = y_{median} - y_0$$

$$f_6 = y_n - y_0$$

$$f_7 = sign(f(x) \mid_{x_{intersect+\alpha}} - y_{intersect}), x_{intersect} < \alpha < x_n$$

$$f_8 = \max(g^{-1}(g(x))) - \max(f^{-1}(f(x)))$$

그림 1. 제스쳐 인식을 위해 사용된 특징들

특징 f_1은 제스쳐 $g(x)$와 제스쳐의 시작점과 끝점을 연결한 직선 $f(x)$사이에 생성된 교차점의 개수를 f_2는 $f(x)$와 $g(x)$의 면적 차의 부호(sign) 값을 의미한다. 그리고 f_3은 f_2에서 구한 면적을 바운딩 박스(Bounding Box) 면적으로 나눈 값을 f_4는 획을 구성하는 모든 점들의 x 좌표 값에서 마지막 점의 x 좌표 값을 뺀 값들을 더한 값을 나타낸다. 특징 f_5는 획의 중간 점의 y 좌표 값에서 시작점의 y 좌표 값을 뺀 값을 의미하고 f_6는 제스쳐를 구성하는 끝점의 y 좌표 값에서 시작점의 y 좌표 값을 뺀 값을, f_7는 교차점과 끝점 사이에 존재하는 점들의 y 좌표 값이 교차점의 y 좌표 값보다 큰지 작은지를 조사한다. 마지막으로 f_8는 $g(x)$의 최대 x 값에서 $f(x)$의 최대 x 값을 뺀 값을 의미한다.

각 제스쳐에 대해 계산된 특징 값들은 표 3과 같다. 표에서 '△' 표시는 관심이 없는 (don't care) 값을 의미하고 '-' 표시는 식이 정의되지 않으므로 계산할 수 없는 값을 의미한다. 각 제스쳐에 대한 계산된 특징 값들을 살펴보면 다른 제스쳐들과 구별되는 특징을 가지고 있음을 알 수 있다. 이러한 특징 값들을 조합하여 제스쳐 결정 함수 $F_i(1 \le i \le 9)$을 정의하였다. 제스쳐 인식 알고리즘은 다음과 같이 세 단계로 수행된다.

[알고리즘] 제스쳐 인식 알고리즘

입력: 베이스라인과 교차점이 생성된 획(stroke)

출력: 제스쳐이면 제스쳐의 종류를, 제스쳐가 아니면 NO_GESTURE 값을 리턴한다.

1. 입력된 획에 대해 특징 값 $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8$을 계산한다.
2. 특징 값들을 이용하여 아홉 가지 제스쳐들에 대한 제스쳐 결정 함수 $F_i(1 \le i \le 9)$을 계산한다.
3. 제스쳐 결정 함수 $F_i(1 \le i \le 9)$의 값이 true인 i를 리턴한다. True 값을 가지는 결정함수가 존재하지 않으면 NO_GESTURE를 리턴한다.

입력된 획이 베이스라인과 교차점을 생성하면서 함수 F_i 의 값이 참이면 제스쳐 인식 알고리즘은 입력된 획을 제스쳐 i로 인식한다. 예를 들어 입력된 획의 특징 값이 <1, △, △, 음의 값, △, 음의 값, 양의 값, 0 >이면 F_4의 이 참이 되므로 입력된 획은 Swap 제스쳐로 인식된다. 만약 함수 F_i의 값이 참이 되는 제스쳐 i가 없으면 입력된 획은 제스쳐가 아닌 데이터로 처리하게 된다.

제스쳐 명령을 수행하기 위해서는 전자 잉크 데이터들을 논리적인 데이터 집합으로 분할하는 과정이 필요하다. 이것은 제스쳐 명령이 최소한 한 문자 단위로 수행되어야 하는데 잉크 데이터는 사용자가 시간 순으로 입력한 획의 집합으로 구성되어 있어 문자에 대한 정보가 포함되어 있지 않기 때문이다.

4. 전자 잉크 데이터의 분할 방법

전자 잉크 데이터들을 분할하여 제스쳐 명령을 수행하기 위한 논리적인 단위로 만든 것을 본 논문에서

표 3. 각 제스쳐에 대한 특징 값들

특징 값 / 제스쳐 종류	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	제스쳐 결정 조건
1. Insert	0	<0	<1/2	<0	<0	△	-	0	$F_1=(f_1==0)\cap(f_2<0)\cap(f_3<1/2)\cap(f_4<0)\cap(f_8==0)$
2. Delete	0	<0	>3/5	>0	<0	△	-	0	$F_2=(f_1==0)\cap(f_2<0)\cap(f_3>3/5)\cap(f_4>0)\cap(f_8==0)$
3. Join	0	>0	△	<0	>0	△	-	0	$F_3=(f_1==0)\cap(f_2>0)\cap(f_4<0)\cap(f_5>0)\cap(f_8==0)$
4. Swap	1	△	△	<0	△	<0	<0	0	$F_4=(f_1==1)\cap(f_4<0)\cap(f_6<0)\cap(f_7<0)$
5. Space	>1	△	△	△	△	<0	△	△	$F_5=(f_1>1)\cap(f_6<0)$
6. Line Insert	0	△	△	>0	<0	<0	-	>0	$F_6=(f_1==0)\cap(f_4>0)\cap(f_5<0)\cap(f_6<0)\cap(f_8>0)$
7. New Line	1	△	△	<0	>0	>0	>0	0	$F_7=(f_1==1)\cap(f_4<0)\cap(f_6>0)\cap(f_7>0)$
8. Line Up	1	△	△	>0	<0	<0	<0	0	$F_8=(f_1==1)\cap(f_4>0)\cap(f_5<0)\cap(f_6<0)\cap(f_7>0)$
9. Undo	0	>0	△	>0	<0	<0	-	0	$F_9=(f_1==0)\cap(f_2>0)\cap(f_4>0)\cap(f_5<0)\cap(f_8==0)$

는 제스쳐 요소, *GC*(Gesture Components)라고 정의하였다. 전자 잉크 데이터를 *GC* 단위로 분할하기 위해서는 먼저 잉크 데이터를 *OC*(Overlapped Components) 단위로 분할하여야 한다. *OC*는 획을 구성하는 점들의 x좌표의 값을 수직으로 투영시켰을 때 서로 겹쳐지는 영역이 발생되는 획들을 하나로 묶은 단위이다[10]. 종성이 있는 한글의 경우 대부분 초성과 중성에 해당되는 획과 종성에 해당되는 획이 겹치게 된다. 그림 2의 잉크 데이터는 여덟 개의 획들(S^1, S^2, S^3, ..., S^8)로 구성되어 있지만 각 획의 x 좌표 값을 수직으로 투영시켰을 때 서로 겹치는 영역이 발생하므로 하나의 *OC*가 생성되는 예이다. 그림에서 색칠된 사각형 영역은 첫 번째 획(S^1)과 마지막 획(S^8)이 겹쳐지는 부분을 보여 준다.

종성이 없는 한글 데이터인 경우 대부분 초성과 중성이 각각 하나의 *OC*가 된다. *OC* 단위로 제스쳐 명령을 수행하면 제스쳐 명령 수행에 의해 초성과 중성에 해당되는 획들이 서로 분리될 수 있다. 그림 3는 이와 같은 현상을 설명해 준다. *OC* 단위로 잉크 데이터가 분할되어 있다고 가정할 때 사용자가 Insert 제스쳐를 입력한 경우 예상되는 결과이다. 그림에서 "이"는 "ㅇ"와 "ㅣ"가 각각 하나의 *OC* 단위이므로 삽입 제스쳐 수행 결과 두 획이 서로 분리되어 "ㅣ" 이하의 획들이 밑의 줄로 이동되었다. 이러한 현상을 막기 위해서는 초성과 중성에 해당되는 획들을 가능한 한 단위가 되도록 잉크 데이터를 분할하는 작업이 필요하다.

온라인이나 오프라인의 필기 데이터를 단어 단위로 분할하기 위한 다양한 방법들이 제안되었는데 가장 좋은 결과를 주는 것은 볼록 껍질(convex hull)을 이용하는 방법으로 알려져 있다[10]. 이 방법은 *OC* 단위의 획 집합에 대해 볼록 껍질을 구하여 이것의 중심점과 이웃한 볼록 껍질의 중심점을 연결할 때 생기는 두 교차점사이의 거리를 계산한다. 그리고 임계값 이하의 거리값을 가지는 경우 이웃한 두 개의 *OC*를 하나로 묶어서 하나의 단위로 취급하였다.

그림 4는 *OC* 단위의 잉크 데이터를 *GC* 단위로 만드는 방법을 설명해 준다. 먼저 각 $OC^i(1 \le i \le 4)$에 대한 볼록 껍질 $CH^i(1 \le i \le 4)$을 구하여 이것의 중심점을 구한다. 그리고 이웃한 볼록 껍질의 중심점들을 서로 연결하면 볼록 껍질과 만나는 교차점들이 생성되는데 그림 4의 점 q_1, ..., q_6 들이다. 이제 교차점 사이의 거리값 $d(CH^i, CH^j)$을 계산하여 이 값을 이웃한 두 *OC* 사이의 거리값으로 지정한다.

*GC*를 생성하기 위한 두 번째 조건은 *OC*를 구성하는 획들의 입력 시간을 이용한다. 사용자가 잉크 데이터를 입력할 때 각 획이 입력되는 시간을 분석해 보면 동일 문자에 속하는 획들에 대한 입력 시간의 차와 다른 문자에 속하는 획들 사이의 입력 시간의 차가 다르게 나타났다. 예를 들어 "모바일"이라는 문자를 입력할 때 대부분의 사용자들은 "모"를 구성하는 획들을 머뭇거리지 않고 입력한 후 다음 문자의 첫 번째 획을 입력하기까지 시간적인 여유를 둔다는 것이다. 획의 입력 시간은 마이크로 초까지 구별 가능하다. 각 OC^i의 첫 번째 획을 구성하는 시작점의 입력 시간을 $t_{start}{}^i$라고 하고 OC^i의 마지막 번째 획의 마지막 점의 입력 시간을 $t_{end}{}^i$ 라고 하자. 이때 두 *OC* 사이의 시간적인 거리 $t(CH^i, CH^j)$는 $t_{start}{}^j$에서 $t_{end}{}^i$을 뺀 시간으로 계산한다.

잉크 데이터를 *GC* 단위로 만들기 위해 먼저 이웃

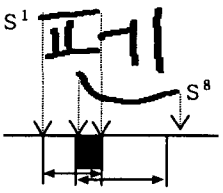
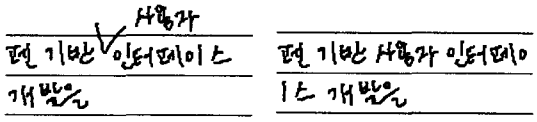
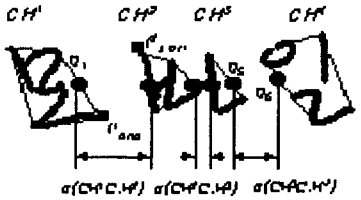


그림 2. 여덟 개의 획들로 구성된 하나의 *OC*



(a) Insert 제스쳐 입력 (b) 제스쳐 수행 결과

그림 3. *OC* 단위로 제스쳐 수행할 경우의 문제점



$$d(CH^i, CH^j) = \sqrt{(q_j - q_i)^2}$$

$$t(CH^i, CH^j) = (t^j_{start} - t^i_{end})ms$$

그림 4. 획의 볼록 껍질과 입력 시간을 이용한 거리값 계산

한 각 OC 사이의 $d(CH^i, CH^j)$와 $t(CH^i, CH^j)$ 값을 계산한다. 두 값이 모두 지정된 임계값 이하를 가질 때 두 OC를 묶어서 하나의 GC를 생성한다. 임계값은 PDA에서 1000개 이상의 단어를 입력하여 구한 두 종류의 거리값을 분석하여 결정하였다. 두 값 중 하나라도 지정된 임계값 이상일 경우 하나의 OC가 그대로 하나의 GC가 된다. 그림 5의 경우 OC^2와 OC^3를 하나의 GC로 묶고 나머지 OC^1과 OC^4는 각각 하나의 GC가 된다.

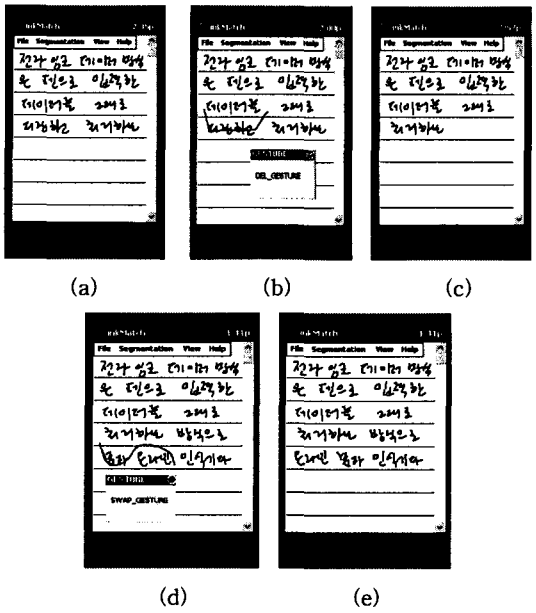
제스쳐 입력에서 수행까지의 과정은 정리하면 다음과 같다. 먼저 베이스라인과 교차점이 생기는 획이 입력되면 제스쳐인지 아닌지를 조사한다. 만약 입력된 획이 제스쳐이면 동일 베이스라인에 있는 잉크 데이터들을 OC 단위로 분할한다. 다음 단계로 OC들에 대한 볼록 껍질을 구하여 거리값 $d(CH^i, CH^j)$와 시간 거리값 $t(CH^i, CH^j)$을 구한다. 마지막으로 계산된 두 종류의 값을 이용하여 잉크 데이터를 GC 단위로 분할한 후 최종적으로 GC 단위의 잉크 데이터에 대해 제스쳐 명령을 수행한다.

5. 구현 및 실험 결과

본 논문에서 제안한 제스쳐 인식 알고리즘과 잉크 데이터의 분할 방법을 이용하여 제스쳐를 허용하는 전자 잉크 에디터 *GesEdit*을 구현하였다. 그림 5은 Pocket PC에서 구현된 *GesEdit*의 수행 화면을 보여준다. 데이터는 베이스라인 사이에 입력된 반면 제스쳐는 베이스라인과 교차하여 입력하였음을 볼 수 있다. 그림 5의 (b)는 Delete 제스쳐가 입력된 것을 보여주고 (c)는 제스쳐의 수행으로 제스쳐 내에 포함되는 획들은 삭제되고 삭제된 획들에 이어서 나오는 잉크 데이터들은 앞으로 이동되었음을 보여 준다. 그림 (d)은 새로운 잉크 데이터의 입력과 Swap 제스쳐의 입력을 보여주며 (e)는 Swap 제스쳐 수행 결과를 보여준다.

표 4는 각 제스쳐에 대한 평균 인식률을 보여준다. 실험은 열 명의 피실험자들에게 각 제스쳐를 50회 이상 입력하도록 한 후 평균 인식률을 계산하였다. 표에서 보는바와 같이 Space와 Line Insert 제스쳐를 제외한 제스쳐들의 평균 인식률은 100%이고 Space와 Line Insert 제스쳐의 인식률은 98%였다.

전체 제스쳐들에 대한 평균 인식률은 99.6% 였고



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

그림 5. 전자 잉크 데이터 편집과 제스쳐 수행 결과 화면: (a) 잉크 데이터 입력 (b) Delete 제스쳐 입력 (c) Delete 제스쳐 수행 결과 (d) Swap 제스쳐 입력 (e) Swap 제스쳐 수행 결과

표 4. 각 제스쳐에 대한 평균 인식률

제스쳐 종류	Insert	Delete	Join	Swap	Space	Line Insert	New Line	Line Up	Undo
사람 수	10	10	10	10	10	10	10	10	10
시도 횟수	50	50	50	50	50	50	50	50	50
평균 인식률 (%)	100	100	100	100	98	98	100	100	100

제스쳐의 인식 및 수행은 모두 1초 이내에 이루어졌다. 본 논문에서 제안한 제스쳐 인식 알고리즘은 빠르면서도 100%에 가까운 인식률을 제공해 주므로 PDA용으로 매우 실용적임을 입증해 주었다.

6. 결론 및 향후 과제

본 논문에서는 펜 기반 모바일 컴퓨터에서 사용할 수 있는 제스쳐를 허용하는 전자 잉크 에디터 개발을 위해 필요한 기술들을 개발하였다. 개발된 기술로는 새로운 제스쳐 인식 알고리즘과 전자 잉크 데이터를 제스쳐 요소 단위로 분할하는 방법이다. 제안한 방법들을 이용하여 전자 잉크 에디터 *GesEdit*를 개발하였으며 특징을 살펴보면 다음과 같다. 첫째, PDA 등

펜 기반 모바일 컴퓨터에서 개발된 최초의 전자 잉크 에디터이다. 둘째, 편집 명령을 제스쳐 형태로 제공하여 문서를 작성하도록 허용하는 시스템이다. 현재 PDA용 응용 프로그램 중 펜 제스쳐를 사용하며 전자 잉크 데이터를 편집하게 해 주는 것은 없으므로 *GesEdit*는 최초의 제스쳐 허용 전자 잉크 에디터라고 할 수 있다. 셋째, 제스쳐의 인식률이 99.6%이므로 매우 실용적이다. 넷째, 제스쳐 인식 및 명령 수행이 1초 이내에 이루어지므로 매우 효율적이다. 다섯째, 제스쳐의 모양이 원고지 교정 부호를 참조하여 설계하였기 때문에 사용자들이 쉽게 익힐 수 있고 오래 기억할 수 있다.

향후 연구 과제로는 첫째 Space와 Line Insert 제스쳐의 인식률을 100%로 향상시키는 것이다. 둘째 다중 획을 이용하여 제스쳐를 설계하는 것이다. 본 논문에서 사용한 제스쳐는 단일 획으로 획의 개수를 제한하였다. 단일 획으로 표현할 수 있는 제스쳐의 종류는 다양하지 못하기 때문에 다중 획으로 제스쳐를 정의하여 다양한 제스쳐를 설계할 수 있도록 그 기능을 확장할 계획이다.

참 고 문 헌

[1] Walid Aref and baniel Barbara, "Supporting Electronic Ink Database," Information Systems, Vol. 24, No. 4, pp. 303-326, 1999.

[2] W. Aref, Ibrahim Kamel and Daniel P. Lopresti, "On Handling Electronic Ink," ACM Computing Surveys, Vol. 27, No 4. pp. 564-567, 1995.

[3] D. Frohlich and R. Hull, "The usability of scribble matching," ACM CHI'96, pp. 189-190, 1996.

[4] T. Yoshino, J. Munemori and K. Yunokuchi, "Development of a PDAs based Idea Collecting System and Its Application to an Idea Generation Consistent Support Systems," IEEE pp.737-742, 2001.

[5] Allan C. Long, "Improving Gestures and Interaction Techniques for Pen-Based User Interfaces," ACM CHI 98, pp.58-59, 1998.

[6] Allan Christian, "Quill: A Gesture Design Tool for Pen-Based User Interfaces," Ph.D Thesis, University of California, Berkeley, 2001.

[7] D. Avraham etc, "Guided Gesture Support in the Paper PDAs," ACM UIST'01 pp.197-198, 2001.

[8] Rubine, D. "Specifying gestures by example," ACM SIGGRAPH Vol. 25, No. 4, pp.329-337, 1991.

[9] http://www.diotek.co.kr

[10] S.H. Kim, S.Jeong, G.S Lee and C.Y.Suen, "Gap metrics for handwritten Korean word segmentation," IEEE Electronics Letters, Vol. 37, No. 14, pp.892-893, 2001.

조 미 경

1990년 부산대학교 전자계산학과 이학사
1992년 부산대학교 전자계산학과 이학석사
1998년 부산대학교 전자계산학과 이학박사
1999년 3월~2000년 8월 신라대학교 강의전담교수
2000년 9월~2002년 8월 부산대학교 컴퓨터및정보통신연구소 기금교수
2002년 9월~현재 동명정보대학교 멀티미디어공학과 전임강사
관심분야 : 그래픽스, 모바일 컴퓨팅
E-mail : mgcho@tit.ac.kr

오 암 석

1984년 부산대학교 전자계산학과 이학사
1986년 중앙대학교 컴퓨터공학과 공학석사
1997년 부산대학교 컴퓨터공학과 공학박사
1987년~1990년 LG연구소 연구원
1990년~1998년 울산과학대학 전자계산과 부교수
1998년~현재 동명정보대학교 멀티미디어공학과 부교수
관심분야 : 데이터베이스, XML, CBD
E-mail : asoh@tmic.tit.ac.kr

교신저자

조 미 경 608-711 부산광역시 남구 용당동 535 동명 정보대학교 멀티미디어공학과