

# 제스처 제안 시스템의 설계 및 구현에 관한 연구

문성현<sup>†</sup>, 윤태현<sup>\*\*</sup>, 황인성<sup>\*\*\*</sup>, 김석규<sup>\*\*\*\*</sup>, 박 준<sup>\*\*\*\*\*</sup>, 한상영<sup>\*\*\*\*\*</sup>

## 요 약

제스처는 빠르고 간편하게 명령을 수행 할 수 있어서 스마트폰과 태블릿PC, 웹 브라우저를 비롯한 수많은 어플리케이션에서 사용되고 있다. 어플리케이션에 제스처를 적용하기 위해서는 제스처를 디자인해야하는데 사용자와 시스템 두 가지 측면을 모두 고려하여 디자인해야 한다. 이러한 제스처 디자인을 도와주고자 몇몇 툴들이 개발되어왔다. 그럼에도 불구하고 제스처를 디자인하기 위해서는 다음과 같은 어려움이 남아있다. 첫째, 모든 제스처를 사람이 직접 디자인해야 한다. 둘째, 디자인한 제스처를 인식기가 올바르게 인식할 수 있도록 반복적인 트레이닝 작업을 해야 한다. 본 논문에서는 보다 간편한 제스처 디자인 환경을 제공해 주고자 자동화 트레이닝, 제스처 제안, 제스처 생성을 제안하였다. 이를 통해 제스처를 트레이닝 시킬 필요가 없어졌고 생성된 제스처와 수집된 제스처의 마할라노비스 거리를 계산하여 이 중 인식이 잘 될 가능성이 높은 순서대로 제스처들을 제안해 줌으로서 모든 제스처를 직접 디자인해야 하는 노력을 줄일 수 있게 되었다.

## A Study on Design and Implementation of Gesture Proposal System

Sung Hyun Moon<sup>†</sup>, Tae-Hyun Yoon<sup>\*\*</sup>, In-Sung Hwang<sup>\*\*\*</sup>,  
Seok-Kyoo Kim<sup>\*\*\*\*</sup>, Jun Park<sup>\*\*\*\*\*</sup>, Sang-Yong Han<sup>\*\*\*\*\*</sup>

## ABSTRACT

Gesture is applied in many applications such as smart-phone, tablet-PC, and web-browser since it is a fast and simple way to invoke commands. For gesture applications, a gesture designer needs to consider both user and system during designing gestures. In spite of development of gesture design tools, some difficulties for gesture design still remains as followings; first, a designer must design every gesture manually one by one, and, second, a designer must repeatedly train gestures. In this paper, we propose a gesture proposal system that automates gesture training and gesture generation to provide more simple gesture design environment. Using automation of gesture training, a designer does not need to manually train gestures. Proposed gesture proposal system would decrease difficulties of gesture design by suggesting gestures of high recognition possibility that are generated based on mahalanobis distance calculation among generated and pre-existing gestures.

**Key words:** Gesture Training(제스처 트레이닝), Gesture Design(제스처 디자인), Gesture Proposal(제스처 제안), Pen Gesture(펜 제스처)

※ 교신저자(Corresponding Author): 문성현, 주소: 서울특별시 관악구 서울대학교 공과대학 301동 419호 병렬처리 연구실(608-743), 전화: 02)880-6575, FAX: 02)000-0000, E-mail: shmoon@pplab.snu.ac.kr  
접수일: 2011년 3월 17일, 수정일: 2011년 7월 28일  
완료일: 2011년 9월 20일

<sup>†</sup> 정회원, 서울대학교 컴퓨터공학부

<sup>\*\*</sup> 준회원, 서울대학교 컴퓨터공학부

(E-mail: ytn9042@nate.com)

<sup>\*\*\*</sup> 준회원, 서울대학교 컴퓨터공학부

(E-mail: ishwan@pplab.snu.ac.kr)

<sup>\*\*\*\*</sup> 준회원, 한국과학기술정보연구원 슈퍼컴퓨팅본부

(E-mail: panemone@gmail.com)

<sup>\*\*\*\*\*</sup> 정회원, 홍익대학교 컴퓨터공학부

(E-mail: joseph.j.park@gmail.com)

<sup>\*\*\*\*\*</sup> 정회원, 서울대학교 컴퓨터공학부

(E-mail: syhan@pplab.snu.ac.kr)

## 1. 서 론

스마트폰과 태블릿PC[1,2]가 시장으로 대거 출시되고 있다. 이러한 휴대 장치는 키보드나 마우스보다는 휴대성이 뛰어난 펜이나 손 터치 중심의 인터페이스를 주로 사용한다. 이 인터페이스에서 제스처<sup>1)</sup>를 사용하는 것은 매우 편리하기 때문에 이를 시스템에 적용하여 사용하고 있다[3]. 또한 웹브라우저[4]와 웹브라우저가 제공해 주지 않는 추가적인 기능들을 제공해주는 프로그램인 웹툰바[5,6]에서도 제스처를 사용하고 있다.

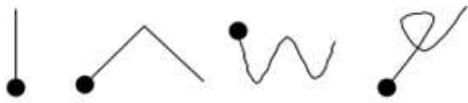


그림 1. 제스처

이렇게 컴퓨터 기술의 급격한 발전과 함께 고전적인 인터페이스인 키보드와 마우스 외에도 수많은 인터페이스들이 개발되었다[7]. 그 중에서 펜 제스처와 같은 멀티 모달 유저인터페이스, 손 제스처 기반의 인터페이스, 제스처를 단축키로 이용하는 데스크 탑 기반의 인터페이스에서 제스처를 사용자가 효과적으로 활용하게 하기 위해서는 사람이 기억하기 쉬우며 인식률이 높은 좋은 제스처들을 디자인해서 제공해 주어야 한다.

### 1.1 제스처 디자인 틀

좋은 제스처를 디자인하기 위해서는 사용자와 시스템의 성능 두 가지 측면을 고려해야 한다. 먼저, 사용자 측면에서는 배우기 쉽고 기억하기 쉬우며 혼동되지 않고 사용하기 쉬운 제스처를 디자인해야 한다. 배우기 쉽고 기억하기 쉽게 하기 위해서는 + 기호가 덧셈을 의미한다는 것과 같이 의미를 가지는 제스처를 디자인함으로써 해결 할 수 있다. 그리고 혼동되지 않는 제스처를 디자인하기 위해서는 제스처들 간에 유사한 모양을 가지지 않게 디자인해야 한다.

다음으로, 시스템의 성능측면에서는 인식률과 제스처셋<sup>2)</sup>의 확장성을 고려하여 디자인해야 한다[7].

위의 조건들을 만족하는 제스처셋을 디자인하기 위해서는 제스처를 디자인 하는 사람이 제스처가 인식기에 의해서 어떻게 인식되는지 어떻게 제스처를 디자인하는 것이 기억하기 쉽고 사용하기 쉽게 하는 것인지 등에 대한 지식을 가지고 있어야 된다[8].

이 지식들을 제공해 주고자 Long이 처음 연구한 제스처 디자인 틀은 Gesture Design Tool(GDT)이다[8]. 이 틀은 디자인한 제스처가 올바르게 인식되지 않는 문제점을 발견하는데 도움을 주기 위해 distance matrix 테이블을 제공해 주었다. 이 테이블에는 인식기의 특징 공간에서의 제스처들 간의 유사한 정도를 측정하기 위한 용도로 사용되는 마할라노비스 거리(MD: Mahalanobis Distance)[9]값이 들어가 있다. 이 도구는 인간과 컴퓨터의 상호작용을 연구하는 분야에서도 인식기 측면에서 제스처들 간의 유사한 정도를 측정하기 위한 용도로도 사용되어 왔다[7,8,10,11,12]. 이 값이 작은 제스처들은 인식기가 해당 제스처들을 혼동하게 된다는 것을 의미하기 때문에 이 제스처들을 같이 사용하게 되면 제스처셋의 인식률 저하가 발생한다. 이 틀을 사용함으로써 자신이 디자인 한 제스처들의 인식률을 91.4%에서 95.4%까지 4%를 향상시킬 수 있었다. 그러나 실험참가자들은 이 틀을 사용하면서 자신이 디자인 한 제스처가 올바르게 인식되지 않는 문제점을 찾고 수정하는 것이 어렵다고 지적하였다[8].

이 문제점을 해결하고자 인식기와 사람측면에서 디자인한 제스처를 분석하여 제스처의 문제점과 이를 어떻게 수정해야 하는지에 대한 정보를 능동적으로 상세하게 제공해주는 지능적인 툴인 킴을 개발하였다[10]. 이를 통해서 제스처에 어떠한 문제점이 있는지 어떻게 고쳐야 하는지 사람이 직접 찾아낼 필요가 없어졌다. 하지만 제공되는 피드백 정보에 수학적 용어를 사용하기 때문에 이 용어들에 익숙하지 않은 사람들은 이 정보를 이해하기 어렵다.

이 문제가 해결되더라도 디자이너가 제스처를 특정 어플리케이션에 적용하기 위해서는 모든 제스처를 직접 디자인해야 한다. 그리고 제스처를 디자인한 뒤에는 인식기가 디자인된 제스처를 올바르게 인식

1) 제스처란 그림 1에 있는 펜 제스처나 마우스 제스처를 말하는데 특정 도형이나 문자 등을 확으로 그림으로서 그에 대응하는 명령을 수행시키는 것을 말한다[3]. 그림 1에서 굵은 점은 제스처의 시작점을 나타낸다.

2) 제스처셋이란 특정 어플리케이션에서 사용되는 제스처들의 집합을 말한다.

할 수 있도록 트레이닝<sup>3)</sup> 작업을 수행해야한다. 이러한 트레이닝 작업을 거친 뒤에는 테스트 과정을 거쳐서 디자인한 제스처의 인식률이 적절한지를 확인하게 된다. 테스트 과정에서 인식률을 저하시키는 제스처가 있는 경우에는 해당 제스처를 수정해야 하는데 이때 수정한 제스처를 인식기가 올바르게 인식할 수 있도록 다시 15번의 트레이닝 작업을 해야 한다. 이렇게 계속되는 디자인의 변경과 트레이닝 작업은 제스처를 특정 어플리케이션에 적용하기까지 많은 시간을 소모하게 만든다.

본 논문에서는 제스처 디자인에 드는 노력을 줄여 주고자 이를 도와줄 수 있는 방법에 대한 연구를 수행하였다. 이 연구를 통해서 자동화 트레이닝, MD를 이용한 제스처 제안 기능, 제스처의 생성기능을 제안하였다.

### 1.2 인지모델링

Long은 제스처 디자인 틀을 만들면서 사람들이 비슷한 제스처를 혼동하게 되고 그로 인해 제스처를 배우거나 기억하기 어려워 한다는 것을 알게 되었다. 이를 해결하고자 사람들이 어떤 특징들에 의해 제스처들을 유사하게 보는가에 대한 연구를 하였고 부산물로 제스처들 간의 유사함을 계산할 수 있는 모델을 만들었다[13]. 이 모델의 경우 모든 제스처들 간의 유사성을 구별 할 수 있는 것은 아니다. 따라서 보다 완벽한 모델을 만들기 위해서는 이를 보완할 특성들을 찾아내는 추가적인 연구가 필요하다.

### 1.3 제스처 디자인 가이드라인

Tian은 20가지 제스처들을 선택하여 사람들을 대상으로 실험을 하였다[14]. 실험을 통해 좋은 제스처가 가져야 할 특성들을 아래와 같이 알게 되었다. 이 중 상징성을 가지는 제스처가 왜 기억하기 쉬운가를 Dual Coding Theory[15]로 설명하였다.

1) 상징성(Iconicness): 90%이상의 참가자들이 명령어와 관련된 시각적인 의미를 갖는 제스처가 기억

3) 트레이닝이란 인식기가 제스처를 올바르게 인식할 수 있도록 하나의 제스처를 여러 번 그리는 작업을 말하는데 가장 적은 트레이닝 데이터가 필요한 Rubine알고리즘[11]의 경우에도 최소 15번의 트레이닝 작업을 거쳐야 96%정도의 인식률이 나온다.

하기 쉽고 배우기 쉽다고 하였다.

2) 사용하기 쉬움(Operating easily): 사용하기 쉬운 제스처들은 다음의 특징을 가져야 한다. 첫째, 편리하게 동작되어야 한다. 둘째, 하나의 제스처는 한 획(one stroke)으로 그려야 된다. 셋째, 곡선(curve)으로 디자인 되어야 한다.

3) 보기 좋은 제스처(Good-looking on figures): 몇몇 참가자들은 편안해 보이고 보기 좋은 제스처들을 선호하였다.

### 1.4 인식 알고리즘

제스처 인식 알고리즘은 크게 은닉 마르코프 모델(HMM: Hidden Markov Models)[16-19], 신경망(Neural Networks)[20], 특징 기반의 통계적 분류기[11,21], 다이내믹 프로그래밍[22,23], 그리고 애드-혹 휴리스틱 인식기[24,25]로 나눌 수 있다.

이 중 HMM과 신경망, 통계적 분류기의 경우에는 높은 수준의 인식률을 얻기 위해서 수많은 트레이닝 데이터가 필요하고 알고리즘을 프로그래밍 하기 어렵기 때문에 실제 시스템에 적용하기에는 어려움이 있다. 그리고 다이내믹 프로그래밍 방식은 계산해야 할 데이터가 너무 많기 때문에 알고리즘의 수행속도가 느리다[26]. 마지막으로 애드-혹 휴리스틱 인식기는 사용자가 새로운 제스처를 정의(definition)하거나 채택하는 것(adaptation)을 허용하지 않기 때문에 제스처셋의 확장성(expandability)이 떨어진다.

본 논문에서 사용한 Rubine 알고리즘은 여러 논문에서 사용되고 있고[7,8,10,27] 프로그래밍 하기 쉬우며 제스처마다 15개 정도의 트레이닝 데이터만 있으면 96%정도의 인식률을 얻을 수 있다[11].

## 2. 제스처 제안 시스템

본 장에서는 제스처 제안 시스템의 구조와 제스처를 제안하기 위해 사용한 제스처 데이터, 제스처 제안 시스템의 새로운 기능들, 기존 툴과의 비교를 통해 제스처 제안 시스템에 대해 상세하게 알아보도록 하겠다.

### 2.1 제스처 제안 시스템의 구조

제스처 제안 시스템의 구조는 그림 2와 같다. 이

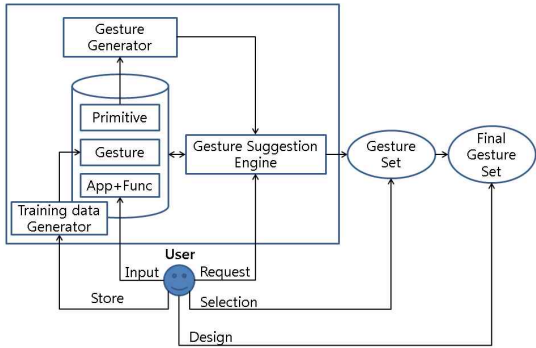


그림 2. 제스처 제안 시스템의 구조

그림에 있는 각각의 용어들에 대한 정의는 다음과 같다. 데이터베이스 안에는 제스처를 제안하기 위해 사용되는 제스처 데이터인 Primitive와 Gesture가 들어가 있다. 그리고 사용자가 정의한 어플리케이션(App)과 그 어플리케이션에서 사용할 기능(Func)들이 정의되어 있다. Primitive는 제스처를 새로이 생성할 때 조합되는 가장 기본적인 유닛 제스처들을 나타내고 Gesture는 제스처를 제안할 때 사용되는 제스처들을 나타낸다.

Gesture Generator는 제스처 프리미티브를 이용하여 제스처들을 생성하는 제스처 생성기를 말한다. 그리고 Gesture Suggestion Engine은 MD를 이용하여 사용자가 정의한 기능에 대응하는 제스처들을 데이터베이스 안에서 검색하여 인식이 잘 될 가능성이 가장 높은 제스처들부터 순서대로 제안해주는 엔진을 말한다. Training data generator는 하나의 트레이닝 데이터만 입력하면 자동으로 15개의 트레이닝 데이터를 만들어주는 트레이닝 데이터 생성기를 말한다. Gesture Set은 제스처 제안 엔진을 통해 제안된 제스처셋을 나타내고 Final Gesture Set은 제안된 제스처셋 중에서 디자이너가 자신이 정의한 기능들과 가장 잘 어울린다고 생각하여 선택한 제스처들과 직접 디자인한 제스처들로 구성된 집합을 말한다. 좀 더 자세한 작동방식에 대해서는 아래에서 자세하게 설명하겠다.

데이터베이스에 저장되어 있는 제스처들은 다양한 어플리케이션과 논문[3,28-37]에서 수집한 제스처들이 저장되어 있는데 구조는 표 1과 같다. 각각의 제스처는 x, y 좌표들과 time값을 저장하였고 태그는 수집한 제스처에 대응하는 기능들을 저장시켜놓았다.

이러한 구조로 구축되어 있는 시스템에 디자이너

표 1. 데이터베이스의 구조

제스처	태그 리스트
●	up, next, on
●	down, pre, off

가 제스처를 적용할 어플리케이션과 그 어플리케이션에서 사용할 기능들을 정의한다. 그 뒤에 자신이 정의한 각각의 기능에 대응하는 제스처의 제안을 시스템에 요청하게 된다. 시스템에 제스처 제안 요청이 들어오게 되면 제스처 제안 엔진이 제스처를 제안 받은 기능에 대응하는 태그를 가진 제스처들을 데이터베이스에서 검색한다. 검색된 제스처가 존재하는 경우 기존에 제안되어 있는 제스처와의 MD차이가 가장 큰 제스처들부터 차례대로 보여준다. 즉 인식률의 저하가 가장 일어나지 않을 가능성이 큰 제스처들부터 순서대로 보여준다. 그 다음 디자이너가 자신의 직관력과 MD를 고려하여 정의한 기능과 가장 잘 어울린다고 생각하는 제스처를 선택하게 된다. 만약 이 중에서 마음에 드는 제스처가 없는 경우에는 시스템에 제스처 생성을 요청하게 된다. 그러면 제스처 생성기가 8개의 간단한 제스처 프리미티브들을 이용하여 58개의 제스처들을 생성하게 된다. 이 중에서도 마음에 드는 제스처가 없는 경우에는 최종적으로 자신이 직접 제스처를 디자인 하게 된다. 이렇게 제스처를 직접 디자인한 경우에는 디자인한 제스처를 추후에 사용할 수 있도록 데이터베이스 안에 저장을 한다.

앞에서 말한 MD에 따른 제스처 제안의 이해를 돕기 위해 표 2의 예를 들어 살펴보자. 이 표에 있는 값들은 MD값들을 나타내고 Del과 Sel은 이미 디자이너가 특정 제스처를 선택하거나 디자인해놓은 제스처들이 대응되어 있는 기능들이다. 그리고 디자이너가 Paste에 대한 제스처를 제안 받으려고 할 때 데이터베이스에 Paste에 대응하는 태그를 가진 제스처가 두 개 있다고 하자. 이 경우 후보1 제스처와 Del과 Sel에 대응하는 제스처들과의 MD합이 19+20 =

표 2. 인식률을 고려한 제스처 제안 시 마할라노비스 거리

기능	Paste(후보1)	Paste(후보2)
Del	19	35
Sel	20	9

39이고, 후보2의 MD합은  $35+9 = 44$ 가 된다. 이 경우 후보2의 제스처를 먼저 보여주고 후보1의 제스처를 보여주는 식으로 MD합이 높은 순으로 제스처를 보여준다. 하지만 MD합이 높은 것만을 고려해서 제스처 제안의 우선순위를 정할 경우에는 특정 제스처와의 MD가 낮더라도 우선순위를 높게 해서 제안해 주게 된다. 이 제스처를 선택하게 되는 경우 제안된 제스처 셋의 인식률이 떨어지기 때문에 경험적으로 MD 8.5이하인 제스처가 포함 된 경우에는 제스처 제안 순위를 뒤로 미루었다.

## 2.2 제스처 데이터

### 2.2.1 수집한 제스처 데이터

이 시스템에서는 제스처 디자인의 드는 노력을 줄여보고자 여러 어플리케이션과 논문에서 제스처들을 수집하여 데이터베이스에 입력해 두었다. 사용자

가 자신이 제스처를 직접 디자인 하는 것보다 데이터베이스 안에 있는 제스처들을 선택해서 사용한다면 제스처 디자인에 드는 노력을 상당부분 줄일 수 있다. 다음 장에 있는 표 3과 표 4는 수집한 36개의 제스처들이다.

표를 살펴보면 제스처마다 태그가 여러 개 있는 것을 볼 수 있는데 그 이유는 동일한 제스처라도 어플리케이션마다 적용된 기능이 다르기 때문이다. 즉 어떤 곳에서는 delete에 a라는 제스처를 사용하고 있었고 다른 곳에서는 b라는 제스처를 사용하고 있었다.

제스처를 수집하면서 기능과 전혀 연관성이 없어 보이는 표 5와 같은 제스처들은 사용자에게 제공해 주기에는 효용성이 없기 때문에 수집 대상에서 제외시켰다.

또한 동일한 기능들도 어플리케이션에 따라서 서로 다른 단어로 쓰여 있었는데 이 단어들도 하나의 단어로 일반화 시켰다. 예를 들어, delete의 기능을

표 3. 비선형 제스처

제스처	태그 리스트	제스처	태그 리스트
	insert, paste		paste
	copy		delete
	undo		redo
	cut, delete		delete, forward, copy
	play		rewind
	redo		undo
	delete		delete
	delete		un-group
	cancel		group
	undo		previous
	zoom-in		next
	select, zoom-out, record		swap, transpose
	Select Left		Select Right
	New Line		Down Right

표 4. 선형 제스처













제스처	태그 리스트	제스처	태그 리스트
	up		down
	forward, right		backspace, back
	on		off
	forward-slash		back-slash

표 5. 기능과 연관성이 없어 보이는 제스

제스처	태그	제스처	태그
	New window		New tab
	Refresh		Close window

하는 erase라고 쓰여 있는 제스처의 경우에는 태그를 delete로 통일하여 저장시켰다.

그리고 제스처들마다 디자인된 방식이 달랐다. 먼저, 기능의 첫 번째 글자를 따서 디자인한 제스처가 있었고 예를 들어 Copy의 알파벳 C를 따서 만든 그림 3의 (a)와 같은 제스처가 있었고 기능의 시각적인 모양을 따서 디자인한 제스처인 가위모양으로 만든 제스처 그림 3의 (b)와 같은 제스처 Cut이 있었다. 제스처를 사용하는 사용자 입장에서 두 가지 방식을 혼용하여 디자인된 제스처셋을 사용하는 경우에는 제스처를 기억하는데 혼동이 발생할 수 있다[14].



그림 3. 디자인 방식에 따른 제스처의 형태

2.2.2 제스처 데이터의 분류

제스처 제안 시스템에서 사용되는 제스처들은 제스처 프리미티브와 일반적인 제스처 두 가지로 분류된다. 제스처 프리미티브는 제스처 제안과 생성 두 가지 용도로 사용되는 제스처들로서 그림 4에 있는 가장 기본적인 제스처들만을 사용하였다. 그리고 일

반적인 제스처들은 제스처 제안 시에만 사용되는 제스처들이다.

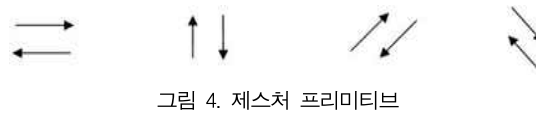


그림 4. 제스처 프리미티브

2.3 제스처 제안 시스템만의 새로운 기능들

기존의 제스처 디자인을 도와주기 위해 제공되는 툴들은 제스처를 디자인하기 위해 제스처 디자인, 트레이닝, 테스트의 과정을 거쳤다. 또한 디자인한 제스처에 문제가 있는 경우에 이를 다시 수정하고 트레이닝 시키고 테스트를 하는 방식이었다. 제스처 제안 시스템은 이러한 기존의 툴보다 편안한 제스처 디자인 환경을 제공해주고자 다음과 같은 기능들을 추가하였다.

먼저, 하나의 제스처만 그리면 자동으로 15개의 트레이닝 데이터들을 만들어 주는 트레이닝 자동화 기능을 추가하였다. 이 기능을 제공해줌으로서 제스처 제안 시스템은 기존 툴들의 제스처 디자인 방식인 그림 5에서 제스처 트레이닝 절차가 없어지면서 그림 6과 같은 제스처 디자인 방식을 가지게 되었다.

그리고 제스처 디자인에 드는 노력을 줄여주고자

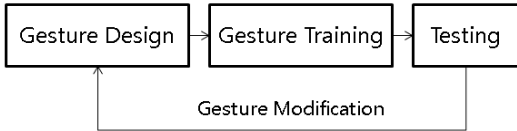


그림 5. 기존 틀들의 제스처 디자인 방식

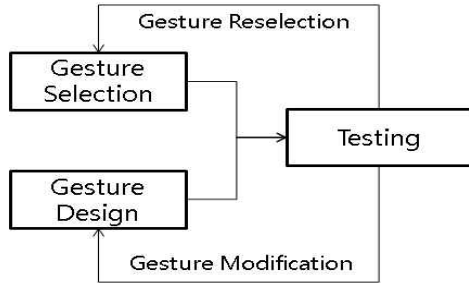


그림 6. 본 시스템의 제스처 디자인 방식

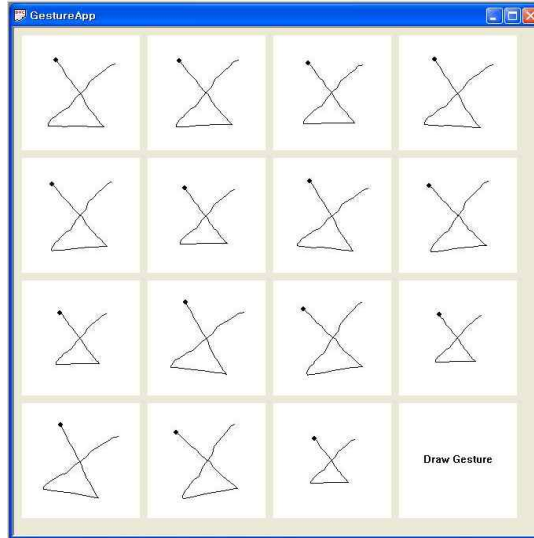


그림 7. 자동 생성된 트레이닝 데이터

여러 어플리케이션과 논문들에서 제스처들을 수집하여 태그를 대응시켜 데이터베이스에 저장시켜둔 뒤 이를 이용하여 제스처를 제안해주는 기능을 추가하였다. 이 기능은 디자이너가 정의한 기능에 대응하는 태그를 가진 제스처들을 데이터베이스에서 검색하여 인식이 가장 잘 될 가능성이 높은 순서대로 정렬하여 제안해주는 기능이다. 디자이너는 제안 받은 제스처들 중 마음에 드는 제스처가 있는 경우 제스처를 선택할 수 있다. 만약 정의한 기능에 대응하는 태그를 가진 제스처가 없거나 제안된 제스처들 중에 마음에 드는 제스처가 없는 경우에는 제스처를 생성하여 제안해주거나 디자이너가 직접 제스처를 디자인 할 수 있다. 이 기능을 통해 자신이 정의한 기능에 대응하는 제스처들을 모두 디자인하는 것이 아니라 제안을 통해 제공받을 수 있게 되었다. 이에 따라 디자인 방식도 그림 5의 기존 방식에는 없었던 제스처 선택이라는 절차가 추가되면서 그림 6처럼 되었다.

2.3.1 제스처 트레이닝의 자동화

제스처의 트레이닝이란 하나의 제스처를 변이( Variation)을 주어 여러 번 그림(Drawing)으로서 인식기가 제스처를 올바르게 인식하도록 하는 작업이다. 본 논문에서 사용한 Rubine 알고리즘의 경우 이 알고리즘을 만든 Rubine의 연구에 따르면 크기와 방향의 변이( Variation)를 고려한 15개의 트레이닝 데이터가 적절하다고 하였다[11]. 다른 알고리즘과 비교하여 상당히 적은 수의 트레이닝 데이터를 입력시킴

에도 불구하고 이러한 작업은 매우 고된 작업이기 때문에 제스처를 어플리케이션에 적용하기 어렵게 만들 것이다[32].

이러한 트레이닝 작업을 없애기 위해서 하나의 제스처만 입력하면 그 제스처를 회전, 축소하여 트레이닝 데이터를 만드는 기능을 추가하였다. 이와 통해 하나의 트레이닝 데이터만 입력하면 그림 7처럼 15개의 트레이닝 데이터를 만들어낸다. 이 기능을 통하여 더 이상 제스처를 인식기가 인식할 수 있도록 사람이 직접 트레이닝 데이터를 만들지 않아도 된다.

2.3.2 제스처 트레이닝 자동화의 성능테스트

제스처의 자동트레이닝 성능이 얼마나 되는지를 테스트하기 위해 실험을 해 보았다. 먼저 테스트를 위하여 그림 8과 같이 8개의 제스처로 이루어진 테스트셋을 만들었다. 자동트레이닝과 어느 정도 동일한 조건으로 테스트를 하기위해 수동 트레이닝 시 방향과 크기를 고려하여 총 15개의 트레이닝 데이터를 만들었다. 수동 트레이닝 데이터의 경우 사람이 직접 입력하는 것이기 때문에 자동트레이닝과 100% 동일한 조건으로 트레이닝 데이터를 만들기는 어려웠다. 이렇게 15개씩 트레이닝 데이터를 만든 뒤에 각각의 제스처마다 10번씩 총 80번의 테스트를 하였다.

그 결과 자동 트레이닝의 경우 79/80 = 98.75%의 인식률이 나오고 수동 트레이닝의 경우 80/80 = 100%의 인식률이 나오는 것을 확인할 수 있었다. 앞

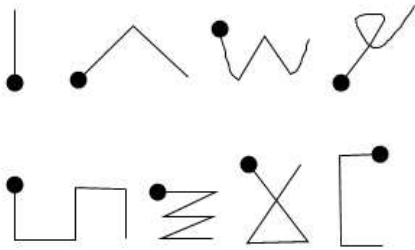


그림 8. 테스트에 사용한 제스처셋

에서 자동트레이닝 테스트에서 발생한 오 인식은 제스처의 트레이닝으로 인해서도 영향을 받지만 테스트 시에 테스트 제스처를 잘못그림으로 인해서도 오 인식이 발생하게 된다. 이러한 고려를 해 보았을 때 자동으로 트레이닝 시키더라도 수동트레이닝과 비교하여 인식률에 큰 차이가 없는 것을 알 수 있다.

### 2.3.3 제스처 제안

디자이너가 제스처를 직접 디자인하는 부분을 줄여보고자 본 시스템에서는 데이터베이스 안에 여러 어플리케이션과 논문에서 수집한 제스처들을 태그와 대응시켜 입력해두고 그림 9처럼 제안해준다. 이

그림은 delete에 대응하는 제스처들이 제안된 것을 보여주고 있다. 이전에 먼저 copy에 대응 하는 제스처가 선택 된 상태이고 이 제스처와의 MD를 계산하여 가장 차이가 많이 나는 순서대로 리스트가 나열된 것이다.

제스처를 제안 받기 위해 디자이너는 제스처를 제안 받을 기능을 선택하게 된다. 그러면 시스템이 해당하는 기능에 대응하는 제스처들이 데이터베이스 안에 존재하는지 검색을 하게 된다. 대응하는 제스처들이 존재하는 경우에는 기존에 제안된 제스처들과의 MD를 계산해서 차이가 가장 큰 순서대로, 즉 인식기측면에서 기존에 제안된 제스처셋과 가장 구별되어 잘 인식될 수 있는 제스처들을 순서대로 보여준다. 디자이너는 제안된 제스처들 중에서 MD와 제스처의 디자인이 해당 기능과 직관적으로 잘 어울리는지를 고려하여 마음에 드는 제스처를 선택하게 된다. 만약에 제안된 제스처들 중에서 마음에 드는 제스처가 없는 경우에는 제스처 프리미티브들을 이용하여 생성된 제스처의 리스트들을 제공받을 수 있다. 이 중에서도 마음에 드는 제스처가 없는 경우에는 사용자가 직접 제스처를 디자인하게 된다. 제스처 제안

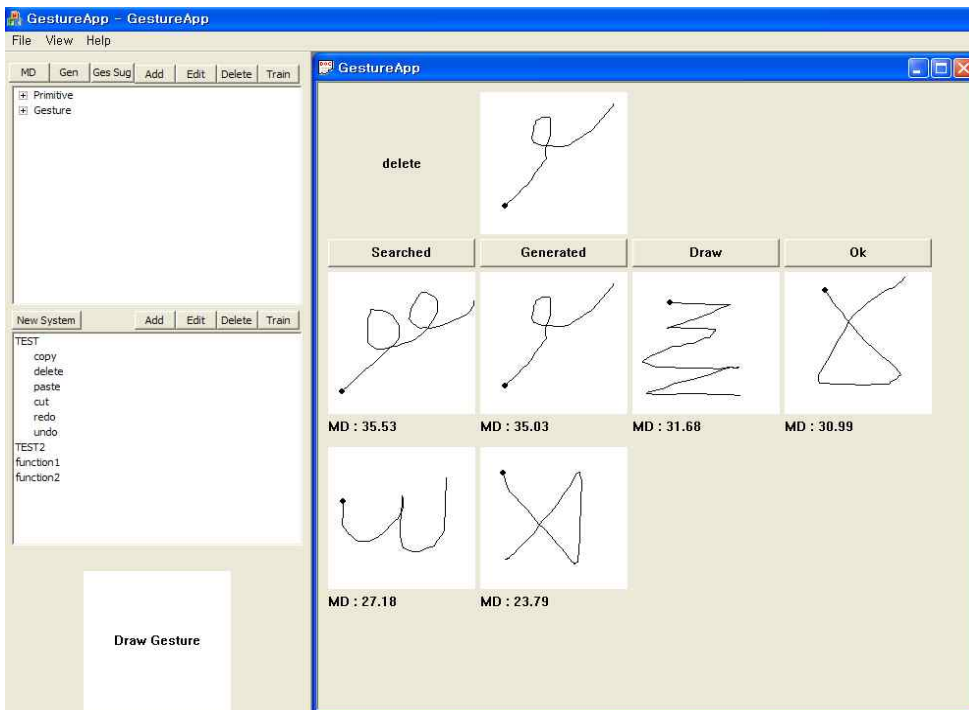


그림 9. 구현된 제스처 제안 시스템



기능을 통해서 디자이너는 모든 제스처를 직접 디자인하는 수고를 덜 수 있게 되었다.

2.3.4 제스처 생성

제스처 제안 시스템에서 저장되어있는 제스처들로 제안된 제스처들을 만족하지 못하여 더 많은 제스처들을 제안 받고 싶은 경우에는 제스처 프리미티브를 이용하여 제스처를 생성해서 제안해준다. 제스처를 생성할 때 그림 10과 같이 (a)의 두 개의 제스처 프리미티브들을 서로 연결하여 (b)를 생성하고 추가로 연결된 제스처를 회전하여 (c)를 생성하였다. 연결은 첫 번째 프리미티브의 끝점에 두 번째 프리미티브를 연결하는 방식을 사용하였고 회전은 앞에서 연결을 통해 생성된 제스처를 45도씩 회전하여 135도까지 3번 회전하는 방식을 사용하였다. 이와 같은 조합을 통해서 제스처 프리미티브 8개를 중복되지 않게 순차적으로 연결하여 28개의 제스처를 생성하였고 각각의 제스처를 3회씩 회전하여 중복된 제스처를 제외 30개의 제스처를 생성하여 총58개의 제스처가 표 6과 같이 생성된다.

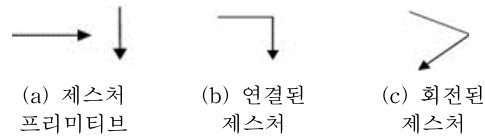


그림 10. 제스처 생성

2.4 기존 툴과 제스처 제안 시스템의 비교

표 7은 기존 툴과 제스처 제안 시스템을 비교한 것이다. GDT는 MD값을 보여주는 테이블과 특정 제스처가 다른 제스처로 인식되는 횟수를 기록한 테이블을 제공해준다[8].

그리고 킴은 GDT에서 테이블로 제공되던 추상적이고 수동적이던 정보를 능동적으로 제공해주었다. 이를 통해 사용자는 제스처에 어떤 문제점이 있고 어떻게 고쳐야 하는지에 대해 상세하게 알게 되었다. 또한 디자인한 제스처의 인식률 테스트의 어려움을 덜어주고자 테스트 할 제스처들을 저장해 놓는 기능을 추가하였다. 이 기능을 통해 제스처를 테스트 할 때마다 다시 그릴필요 없이 저장해놓은 제스처를 이용해서 테스트 할 수 있게 하였다[10].

본 논문에서 제안한 시스템은 기존툴에서 트레이닝 데이터를 일일이 입력하던 작업을 없애기 위해 하나의 트레이닝 데이터만 입력하면 그 데이터를 회전, 축소하여 15개의 트레이닝 데이터를 자동으로 만들어주는 기능을 추가하였다. 또한 제스처 디자인의 어려움을 덜어주고자 인식이 가장 잘 될 가능성이 높은 순서대로 제스처들을 정렬하여 제안해주는 기능을 추가하였다.

표 6. 생성된 제스처 리스트


표 7. 기존 툴들과 제스처 제안시스템의 비교

	GDT	킴	본 시스템
제스처 제안	X	X	O
제스처 생성	X	X	O
제스처에 관한 정보제공	MD행렬, 인식행렬	제스처의 문제점, 고치는 방법	MD 높은 순으로 정렬
제스처 자동화 트레이닝	X	X	O
테스트 데이터의 재사용	X	O	X

### 3. 결론과 향후 과제

본 장에서는 결론을 통해 이 논문에서 연구한 내용을 다시 살펴보고 향후 과제를 통해서 어떠한 연구를 수행해야 하는지에 대해서 알아본다.

#### 3.1 결론

본 논문에서는 제스처 디자인을 도와줄 수 있는 방법에 대한 연구를 수행함으로써 자동화 트레이닝, MD를 이용한 제스처 제안 기능, 제스처의 생성기능을 제안하였다.

자동화 트레이닝은 하나의 제스처만 입력하면 크기와 방향을 고려하여 15개의 트레이닝 데이터를 자동으로 만들어 주는 기능이다. 이 기능을 제공해 줌으로써 제스처를 인식기가 인식할 수 있도록 별도의 트레이닝 작업을 할 필요가 없어졌다.

그리고 MD를 이용한 제스처 제안을 통해서 디자이너는 자신이 정의한 기능에 대응하는 제스처를 제안 받을 수 있게 되었다. 제안 시에는 인식이 가장 잘 될 가능성이 높은 순서대로 제스처를 제안 해 줌으로써 디자이너는 어떤 제스처가 좀 더 인식률을 높일 수 있는지를 알 수 있게 되었다. 또한 데이터베이스에 쌓여있는 제스처들과 생성기를 통해 생성된 제스처들 중에서 자신이 정의한 기능에 대응하는 제스처를 직접 선택할 수 있게 함으로써 어플리케이션에서 사용할 모든 제스처들을 직접 디자인해야 되는 수고를 덜 수 있게 되었다.

#### 3.2 향후 과제

이 시스템을 연구하면서 시스템이 보다 유용해지고 널리 사용되기 위해서는 다음과 같은 연구들이 필요하다고 생각하였다.

##### 3.2.1 제스처 생성방식의 강화

현재의 시스템은 단순히 제스처를 생성할 때 프리미티브만을 이용하여 생성한다. 생성된 제스처들 중에서 마음에 드는 제스처가 없는 경우에는 디자이너가 제스처를 직접 디자인해야 한다. 이러한 제스처 디자인 작업을 완전하게 없애고 제스처 디자인에 관한 지식이 없는 비전문가들도 제스처를 손쉽게 자신의 어플리케이션에서 활용할 수 있게 하기 위해서는 사람이 직접 디자인한 것과 같은 형태의 제스처들을

생성할 수 있는 방법에 관한 연구가 필요하다. 이러한 제스처들의 생성이 가능해진다면 제스처를 직접 디자인할 필요가 없게 될 것이고 이에 따라서 비전문가도 제스처를 손쉽게 자신이 개발한 어플리케이션에 적용할 수 있을 것이다.

##### 3.2.2 명확하게 구별되는 제스처 제안

제스처는 메뉴기반의 인터페이스와는 다르게 특정 기능을 수행하기 위해서는 그 기능에 대응하는 제스처를 기억하고 있어야 한다. 제스처를 기억하기 쉽게 하기 위해서는 사람이 볼 때 제스처들의 모양이 확연하게 구별되어야 한다. 따라서 기능들에 대응하는 제스처들이 제안되어 있는 경우 그 다음기능들에 대해서는 앞에서 제안된 제스처들과는 명확하게 구별되는 제스처들을 제안해 주어야 한다.

##### 3.2.3 제스처 사용의 어려움을 판단하는 모델

VCR on, VCR off와 같은 민감한 기능들은 사용자가 어플리케이션을 사용하면서 실수로 제스처를 그려서 기능들이 수행되지 않게 해야 한다. 따라서 이러한 기능들에 대해서는 사용자가 쉽게 수행할 수 없는 어려운 제스처를 제안해 주어야 한다. 이를 위해서는 사람이 볼 때 두 개의 제스처가 유사한가를 컴퓨터가 판단하게 해 주기 위한 모델을 만들었던 것과 같이 제스처 사용의 어려움을 컴퓨터가 판단할 수 있는 모델에 대한 연구가 필요하다.

### 참 고 문 헌

- [1] Apple, [www.apple.com](http://www.apple.com)
- [2] Samsung Electronics, <http://www.samsung.com/sec/>
- [3] A. Long, A. James, and A. Lawrence, "PDA and Gesture User in Practice: Insights for Designers of Pen-based User Interfaces," *Technical Report UCB//CSD-97-976*, 1997.
- [4] Google Chrome Plus, <http://www.chrome-plus.org/>
- [5] Altoolbar, [http://www.altools.co.kr/Product/ALToolbar\\_Function.aspx](http://www.altools.co.kr/Product/ALToolbar_Function.aspx)
- [6] Naver WebToolbar, <http://toolbar.naver.com/intro/index.nhn?menuId=15>

- [7] C. Henrique and Q. Forster, "Design of Gesture Vocabularies through Analysis of Recognizer Performance in Gesture Space," *Intelligent Systems Design and Applications*, pp. 641-646, 2007.
- [8] A. Long, A. James, and A. Lawrence, "Implications for a Gesture Design Tool," *Proc. CHI, ACM Press*, pp. 40-47, 1999.
- [9] W. Krzanowski, Principles of Multivariate Analysis: A Users's Perspective, *Oxford Statistical Science Series*, Vol.3, Oxford University Press, 1988.
- [10] A. Long, A. James, and A. Lawrence, "Quill: A Gesture Design Tool for Pen-based User Interfaces," [Ph.D. Thesis] Berkeley: University of California, 2001.
- [11] D. Rubine, "Specifying Gestures by Examples," *Proc. SIGGRAPH*, ACM Press, pp. 329-337, 1991.
- [12] W. Dai and G. Shi, "Gesture-based Chemical Formula Editing System," *IEEE Computer Society*, 2009.
- [13] A. Long, A. James, and A. Lawrence, "Visual Similarity of Pen Gestures," *Proc. CHI*, ACM Press, pp. 360-367, 2000.
- [14] F. Tian, T. Cheng, H. Wang, and G. Dai, "Research on User-Centered Design and Recognition Pen Gestures," *Proc. CGI*, LNCS 4035, pp. 312-323, 2006.
- [15] A. Paivio, *Mental representations: A dual coding approach*. Oxford University Press, 1986.
- [16] D. Anderson, C. Bailey, and M. Skubic, "Hidden Markov Model symbol recognition for sketch-based interfaces," *AAAI Fall Symposium*, AAAI Press, pp. 15-21, 2006.
- [17] X. Cao and R. Balakrishnan, "Evaluation of An On-Line Adaptive Gesture Interface With Command Prediction," *Proc. Graphics Interface*, CHCCS Press, pp. 187-194, 2005.
- [18] T. Sezgin and R. Davis, "HMM-Based Efficient Sketch Recognition," *Proc. IUI*, ACM Press, pp. 281-283, 2005.
- [19] 김중호, 윤요섭, 김태영, and 임철수, "은닉 마르코프 모델 기반 동작 인식 방법," 한국멀티미디어학회논문지, v.12, no.4, pp.521-529, 2009.
- [20] J. Pittman, "Recognizing Handwritten Text," *Proc. CHI*, ACM Press, pp. 271-275, 1991.
- [21] M. Cho, "A New Gesture Recognition Algorithm and Segmentation Method of Korean Scripts for Gesture Allowed Ink Editor," *Information Sciences*, ESI Press, 176(9), pp. 1290-1303, 2006.
- [22] C. Myers and L. Rabiner, "A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected Word Recognition," *The Bell System Technical J*, 60(7), pp. 1389-1409, 1981.
- [23] C. Tappert, "Cursive Script Recognition by Elastic Matching," *IBM J. of Research & Development*, 26(6), pp. 765-771, 1982.
- [24] M. Notowidigdo, and B. Miller, "Off-Line Sketch Interpretation," *AAAI Fall Symposium*, AAAI Press, pp. 120-126, 2004.
- [25] A. Wilson and S. Shafer, "XWand: UI for Intelligent Spaces," *Proc CHI*, ACM Press, pp. 545-552, 2003.
- [26] S. Zhai and P. Kristensson, "Shorthand Writing on Stylus Keyboard," *Proc. CHI*, ACM Press, pp. 97-104, 2003.
- [27] I. Jason and A. James, "SATIN: A Toolkit for Informal Ink-Based Applications," *Proc. UIST*, pp. 63-72, 2000.
- [28] Jumping Mind, <http://jeroenarendsen.nl/2006/04/pen-gestures/>
- [29] MindView, [http://www.matchware.com/en/products/mindview/webhelp\\_mv3/user\\_guide/pen\\_mode/idhelp\\_working\\_in\\_pen\\_mode.htm](http://www.matchware.com/en/products/mindview/webhelp_mv3/user_guide/pen_mode/idhelp_working_in_pen_mode.htm)
- [30] Windows XP Tablet PC Edition, <http://www.microsoft.com/korea/WindowsXP/tablet/default.msp>
- [31] Motion4, <http://documentation.apple.com/en/motion/>
- [32] J. Kela, P. Korpipaa, J. Mantyjarvi, S. Kallio,

G. Savino, L. Jozzo, and S. Marca, "Accelerometer-Based Gesture Control for A Design Environment," *PUC*, SV Press, 10(5), pp. 285-299, 2006.

- [33] M. Nicholson and P. Vickers, "Pen-Based Gesture: An Approach to Reducing Screen Clutter in Mobile Computing," *S. Brewster and M. Dunlop (Eds.): MobileHCI*, LNCS 3160, pp. 320-324, 2004.
- [34] J. Landay and B. Myers, "Sketching Interfaces: Toward More Human Interface Design," *IEEE Computer*, Vol.34, No.3, pp. 56-64, 2001.
- [35] S. Chatty and P. Lecoanet, "Pen Computing for Air Traffic Control," *Proc. CHI*. ACM Press, pp. 87-94, 1996.
- [36] O. Bimber, L. Encarnac, and A. Stork, "A Multi-Layered Architecture for Sketch-Based Interaction Within Virtual Environments," *Computer & Graphics 24*, pp. 851-867, 2000.
- [37] J. Landay, and B. Myers, "Interactive Sketching for the Early Stages of User Interface Design," *Proc. CHI*, pp. 45-50, 1995.



**문 성 현**

2002년 University of Maryland, Information System Management 학사  
 2007년 서울대학교 컴퓨터공학부 석사  
 2007년~현재 서울대학교 컴퓨터공학부 박사과정

관심분야 : HCI, 증강현실, 컴퓨터비전, 디지털 스토리텔링



**윤 태 현**

2009년 한국산업기술대학교 컴퓨터공학 학사  
 2011년~현재 서울대학교 컴퓨터공학부 석사  
 관심분야 : HCI, 영상처리, IP Network Camera



**황 인 성**

2010년 KAIST 전산학과 학사  
 2010년~현재 서울대학교 컴퓨터공학부 석사과정  
 관심분야 : HCI, 증강현실, 컴퓨터비전



**김 석 규**

1988년 3월~1992년 2월 서울대학교 계산통계학과 이학사  
 1992년 3월~1994년 2월 서울대학교 계산통계학과 전산학 석사  
 1994년 1월~1996년 7월 현대전자소프트웨어연구소 연구원

1996년 8월~1999년 6월 현대정보기술 연구원  
 1999년 7월~2003년 6월 엔씨소프트 팀장  
 2003년 9월~2010년 2월 서울대학교 컴퓨터공학 박사  
 2010년 3월~2010년 12월 서울대학교 정보기술사업단 박사후연구원  
 2011년 1월~현재 한국과학기술정보연구원 선임연구원  
 관심분야 : 디지털 스토리텔링, 컴퓨터 게임, HCI, 증강현실, 그리드 컴퓨팅, 클라우드 컴퓨팅



**박 준**

1993년 서울대학교 계산통계학과 학사  
 1996년 University of Southern California 전산학 석사  
 2001년 University of Southern California 전산학 박사

2002년~현재 홍익대학교 컴퓨터공학과 부교수  
 관심분야 : 증강현실, HCI, 의료영상, 컴퓨터비전



**한 상 영**

1972년 서울대학교 응용수학과 학사  
 1977년 서울대학교 전산학 석사  
 1983년 University of Texas at Austin 전산학 박사  
 1977년~1978년 울산대학교 공과대학전임강사

1984년~2000년 서울대학교 계산통계학과 조교수, 부교수, 교수  
 2000년~현재 서울대학교 컴퓨터공학부 교수  
 관심분야 : 병렬처리, 보안, HCI