

게임 응용을 위한 가상 터치스크린 시스템

김 형 준
한세대학교 IT학부
harry@hansei.ac.kr

Virtual Touch Screen System for Game Applications

HyungJun Kim
Division of Information Technology, Hansei University

요약

본 논문에서는 일반적인 터치스크린의 단점을 보완하고, 새로운 형태의 입력장치로서 뿐만 아니라 게임 등에 이용할 수 있는 가상막을 이용한 가상 터치스크린을 구현했다. 기존의 터치스크린에서 요구되는 물리적인 막이 없어, 두 대의 웹캠만으로 가상막을 설치함으로써 크기의 제한이 없는 영상처리 기반의 가상 터치스크린이다. 본 논문에서 개발한 가상터치스크린은 가격이 저렴하고 쉽게 설치가 가능하여 새로운 형태의 게임을 구현하는데 사용할 수 있는 시스템이다.

Abstract

We have developed a virtual touch screen system which has improved on the standard functions of general touch screen systems, and it also can be used for game applications. It is based on an image processing system which does not require a physical screen like traditional touch-screen systems. Our system requires only two standard PC cameras, so that there is no limit of the size of the touch screen. The developed system can be installed easily and implemented with low cost, and can be applied to new kinds of games.

Keyword: Touch screen, virtual touch screen, image processing, game application

1. 서론

컴퓨터 사용 시 문자와 좌표 입력에 쓰이는 가장 기본이 되는 장치가 키보드와 마우스이다. 이러한 키보드와 마우스 없이도 입력을 손쉽게 할 수 있어 키오스크 등에서 널리 쓰이고 있는 장치로 터치스크린이 있다. 터치스크린은 실제 모니터 화면의 정보를 보고 화면속의 버튼이나 스크롤 등을 손가락이나 펜을 이용하여 조절함으로써 정보를 컴퓨터에 입력할 수 있는 장치이다. 터치스크린은 평소 키보드 자판에 익숙하지 않은 유아들이나 노인, 몸이 불편한 장애인들의 컴퓨터 이용에 편리성을 제공한다. 이와 같은 이유로 터치스크린을 이용한 어린이용 게임도 이미 출시되고

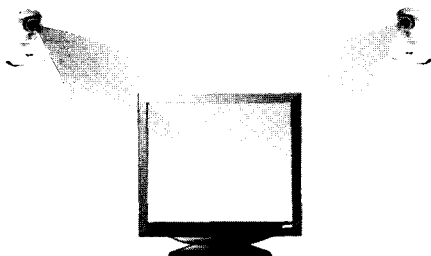
있고, 일반인들을 위한 단순한 게임도 시장을 형성하고 있다. 또한 험한 작업 환경에서도 충분히 활용이 가능하고, 모니터와의 통합으로 좁은 공간에도 쉽게 설치할 수 있다. 그러나 현재의 터치스크린은 물리적 성질을 이용한 것이 대부분이기 때문에 그 크기가 한정되어 있으며, 일반 빔 프로젝션용 스크린 같은 곳에 설치가 불가능하다. 다른 문제점으로 광투과율이 최대 86%에 그치고 있고, 진동에 대해 약점을 가지고 있으며, 가격이 비싸다는 단점을 갖고 있다.

본 연구에서는 이러한 일반적인 터치스크린의 단점을 보완하는 가상막을 이용한 가상 터치스크린을 구상하였다. 기존의 터치스크린에서 요구되는 물리적인 막이 없어, 두 대의 가상 카메라를 이용하여 모니터 화면이나 일반 빔 프

로젝션용 스크린에 가상막을 설치함으로써 크기의 제한이 없고, 쉽게 설치가 가능한 영상처리 기반의 저렴한 가격의 터치스크린을 구현했다. 입력 장치로서 뿐만 아니라 게임 등에 이용할 수 있고, 고급 지식의 요구 없이 직관적이면서 저렴한 가격으로 새로운 형태의 게임을 구현하는데 사용할 수 있는 시스템이다[1]. 본 논문의 구성은 2장에서는 가상 터치스크린 시스템의 구성을 설명하고, 3장에서는 캠의 위치 설정에 필요한 요건을 설명한다. 그리고 4장에서는 가상 터치스크린 시스템에 사용한 영상처리 알고리즘에 대하여 설명하고, 5장에서는 사용 예와 실험 결과 및 분석을 설명한다. 마지막으로 6장에서 결론 및 개선 방향을 제시한다.

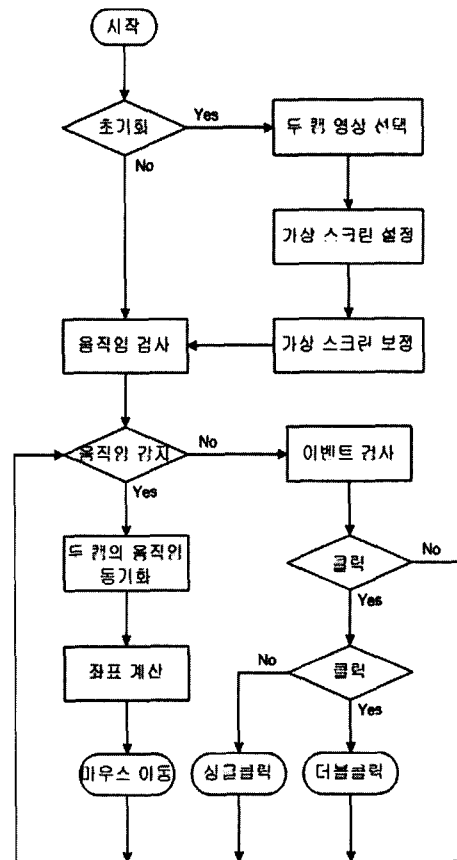
2. 가상 터치스크린 시스템 구성

가상 터치스크린이란 2대의 화상 카메라를 이용하여 일반 모니터를 터치스크린의 기능이 가능하도록 만든 시스템이다. 또한 일반 프로젝션용 스크린에 설치하면 스크린 상에서도 터치스크린 기능을 수행할 수 있다. 가상 터치스크린의 전체 시스템의 구조는 [그림1]과 같다. 가상 터치스크린으로 사용되어질 모니터 화면이나 일반 빔 프로젝션용 스크린의 좌우 모서리의 연장선상에 웹캠을 마주보게 설치한다. 이 두 웹캠이 서로 마주보는 영역에 가상 터치스크린이 생성된다. 여기서 주의할 사항은 컴퓨터 화면이나 일반 빔 프로젝션용 스크린의 크기가 커질수록 가상 스크린을 위한 웹캠과의 공간이 비례하여 필요하다는 점이다. 이는 두 웹캠의 화각 내에 터치스크린으로 사용할 영역이 포함되어야 함을 의미한다. 또한 가상 터치스크린은 카메라가 움직이는 물체를 포착하여 위치를 계산하는 시스템이므로 가상 스크린 상에는 위치를 정하는 손과 같은 물체 외에는 움직이는 물체가 없다고 가정한다.



[그림1] 전체 시스템의 구조

전체 시스템의 순서도는 [그림2]와 같다. 프로그램이 시작되면 초기화 여부를 결정한다. 만약 새로운 장소에 가상 터치스크린을 설치하는 것이라면 초기화 과정을 수행하고, 이미 설정되어 있는 가상 터치스크린을 계속 사용하는 경우에는 초기화 과정을 생략하고 사용할 수 있다. 초기화 과정은 먼저 모니터 화면 또는 빔 프로젝션용 스크린 앞에 가상막을 설정하는 두개의 부분 영상을 선택하는 과정이다. 또한 가상 스크린을 형성해 실제로 작동해야 하는 스크린 구역을 생성한다. 사용자별 차이를 고려하여 보정 작업을 거치면 초기화 작업은 종료된다.



[그림2] 가상 터치스크린 알고리즘

초기화 과정이 종료되면 모니터 화면이나 일반 빔 프로젝션용 스크린의 양쪽 모서리에 설치된 웹캠이 자동으로 가상막의 움직임을 감지하거나 움직임이 감지되지 않으면 클릭을 하는 이벤트를 확인하여 한 번의 클릭이나 두 번의 더블클릭을 검사한다. 이러한 과정 도중 어떤 움직임이 발생

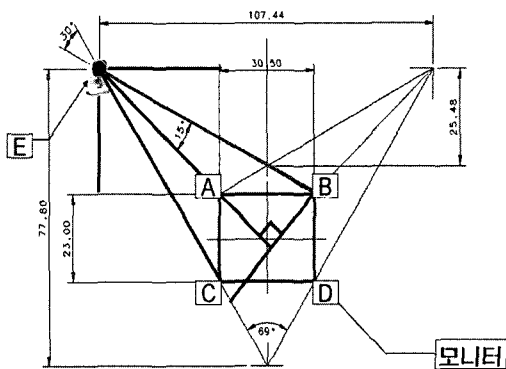
하면 두 화상 카메라의 동기화 작업을 수행하여 좌표를 산출한다. 그 후 해당 좌표로 마우스의 포인터를 이동시키고, 마우스의 이동, 드래그 등 특정 이벤트를 수행한다. 이벤트 처리 후 다시 움직임을 계속 감지하게 된다.

3. 캠의 위치 설정

이 장에서는 두 대의 웹캠으로부터 가상 스크린 상의 커서의 위치를 계산하기 위한 요건과 웹캠의 위치를 설정하는 방법에 대하여 설명한다.

3.1 캠의 위치 요건

두 대의 웹캠으로부터 입력되는 영상에서 커서의 위치 좌표를 구하기 위해서는 입력되는 영상으로부터 두 직선의 방정식을 구하고 그들의 교차점을 구해야 한다. 즉 두 웹캠의 위치 좌표들의 조건은 두 직선의 방정식을 구할 수 있는 조건을 만족해야 한다. [그림3]은 웹캠의 위치를 구하기 위한 요건의 예를 보여준다. 사각형 ABCD는 모니터 화면을 나타내며 하나의 웹캠은 E의 위치에 있다고 가정하고, 다른 하나의 웹캠은 모니터 화면의 중심선을 대칭으로 반대편에 있다고 가정한다.



[그림3] 캠의 위치 요건 예

직선의 방정식을 구하기 위해서는 [그림3]에서 웹캠 위치 E와 모니터 화면의 두 꼭짓점 B, C를 통과하는 삼각형이 이등변 삼각형이 되어야 하고, E점에서 밑변에 수직으로 직교하는 선분이 모니터 화면의 A점을 지나야 한다. 이는 두 캠의 입력각이 30도이며, 직선의 방정식으로부터 커서의 위

치를 계산 할 때에 부분 영상의 중심이 A점이 되고, A로부터의 상단 선분 AB와 하단 선분 AC의 길이를 인위적으로 같게 하여 위치좌표의 연산을 정확하게 하려는데 있다. A는 영역의 중심이 되고, 선분 EB와 EC로부터 밑변에 투영되는 선분의 영역이 웹캠에서 받아들이는 영상의 상단과 하단이 된다. [그림3]은 15인치 모니터 화면일 경우의 예를 나타낸 것으로 모니터의 두 꼭짓점 A와 B, 그리고 A와 C로부터 두 웹캠까지의 거리를 구하는 방법은 다음 절에서 설명한다.

3.2 캠의 위치 좌표

앞에서 설명한 웹캠을 위한 위치 좌표의 요건에 따라 [그림4]와 같이 A와 C를 한 원상의 점들로 설정하고 두 점 A, C와 원의 중심 사이의 각도가 30도인 원을 그리면, 중심까지의 수평거리를 (식1)을 이용하여 구할 수 있다.

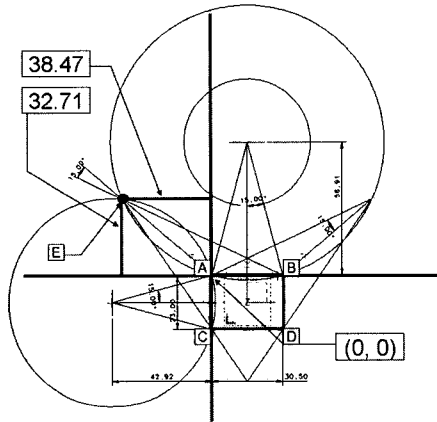
$$d = (l/2)\tan^{-1}(\pi/12) \quad (\text{식1})$$

여기서 d 는 모니터 화면의 한 면 AC로부터 원의 중심까지의 거리를 나타내고, l 은 모니터 화면의 높이를 나타낸다. 예를 들어, 15인치 모니터 화면의 경우 높이 l 은 23cm 이므로 모니터 화면의 좌측 42.92cm의 위치에 중심을 가지는 원을 구할 수 있다. 또한 같은 방법으로 A와 B를 한 원상의 점들로 설정하고 두 점 A, B와 원의 중심 사이의 각도가 30도인 원을 그리면 모니터 화면의 상단에 56.91cm의 위치에 중심을 가지는 또 하나의 원을 구할 수 있다. [그림4]와 같이 2차원 평면의 원점을 A라하고 모니터 화면이 4사분면에 위치한다고 가정하고, 이들 중심의 좌표들을 이용하여 반지름이 각각 r_1, r_2 인 두 원의 방정식을 구하면 (식2)와 같다.

$$\begin{aligned} (x_1 - r_1 \sin(\pi/12))^2 + (y_1 - r_1 \cos(\pi/12))^2 &= r_1^2 \\ (x_2 + r_2 \cos(\pi/12))^2 + (y_2 + r_2 \sin(\pi/12))^2 &= r_2^2 \end{aligned} \quad (\text{식2})$$

이 두 원의 방정식을 이용하여 원들의 두 교점을 구하면 원점 A와 다른 한 점을 구할 수 있다. 이 두 점들 중에 원점을 제외한 좌표 (-38.47, 32.71)가 한 웹캠의 위치가 된다. 왜냐하면, 이 교점은 밑변 AB를 갖고 원주 상에 있는 점이므로 같은 밑변 AB를 갖고 원의 중심을 꼭짓점으로 하는 삼각

형의 내각 30도의 반을 내각으로 가지므로, 한 변 AB를 포함하는 캠의 각도는 15도가 된다. 같은 방법으로 밑변을 AC로 하는 원주 상의 같은 교점의 각도도 15도 이므로 캠의 각도는 전체적으로 30도를 만족하고, 부분 영상의 윗부분과 아랫부분의 길이를 인위적으로 같게 하는 캠의 위치를 구할 수 있다. 그러므로 [그림4]에서 보는 바와 같이 $\angle AEB = \angle AEC = 15^\circ$ 가 되어 캠의 화각은 이등변 삼각형이 되어야 하는 전제 조건을 만족하는 것을 알 수 있다.



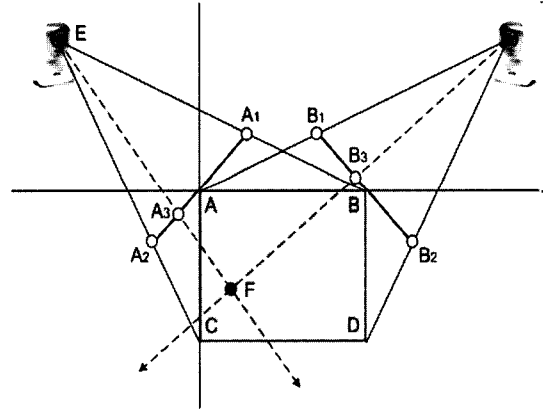
[그림4] 캠의 위치 좌표 구하기

3.3 가상 스크린 상의 좌표 사상

웹캠이 받아들이는 영상의 어느 부분에서 물체를 인식할 것인지 표시해주는 부분이 가상의 스크린이다. 가상의 스크린을 결정한 후 이 영역을 영상에 표시해주면 웹캠 영상에 빨간색 줄이 표시된다. 왼쪽과 오른쪽 웹캠 모두 가상의 스크린을 초기화하는 작업이 반드시 필요하다. [그림4]를 보면 웹캠의 위치를 구할 때 모니터 화면의 좌상단 모서리를 (0,0)으로 두고 가상의 좌표를 정하게 된다. [그림5]에서 모니터의 가상 스크린이 ABCD라 가정하면, 선분 A_1A_2 가 왼쪽 웹캠의 좌표 계산을 위한 부분 영상이 되고, 선분 B_1B_2 가 오른쪽 웹캠의 좌표 계산을 위한 부분 영상이 된다. 그러므로 왼쪽 웹캠의 부분 영상의 중심에 A점이 위치하게 되고, 최상단에는 A_1 이, 최하단에는 A_2 가 위치하게 된다. 같은 개념으로 오른쪽 웹캠의 부분 영상의 중심에 B점이 위치하게 되고, 최상단에는 B_1 이, 최하단에는 B_2 가 위치하게 된다. 예를 들어, [그림5]에서 가상 스크린 상의 한 점 F의 좌표는 왼쪽 웹캠으로부터는 부분 영상의 한 점 A_3 로 인식되고, 오른쪽 웹캠으로부터는 부분 영상의 한 점 B_3 로 인식된

다.

두 웹캠의 위치와 기울기를 알면 두 직선의 방정식을 알 수 있으며 교점의 위치를 계산할 수 있다. 이러한 과정을 통하여 가상 스크린 상의 모든 점들을 선분 A_1A_2 와 선분 B_1B_2 의 부분 영상 좌표로 사상시킬 수 있다. 이를 미리 계산하여 사상 테이블을 사용하면 가상 스크린을 활성화시킬 때 연산량을 줄일 수 있어서 반응 속도를 빠르게 할 수 있다. 예를 들어 좌측 웹캠에서는 110번째, 우측 웹캠에서는 60번째 화소에서 움직임이 관찰됐다면, mapping[110][60]에는 실제 가상 스크린 상의 위치 좌표 (34,93)와 같이 마우스가 이동해야 할 위치의 좌표값이 들어 있다. 따라서 두 부분 영상에서 물체가 감지되었을 경우 양쪽 웹캠의 위치를 저장된 값과 비교하여 가상 스크린 상의 커서 위치를 결정하게 된다.



[그림5] 가상 스크린의 위치 좌표 사상

4. 영상 처리

이 장에서는 가상 터치스크린을 게임에 적용하기 위한 전체적인 영상 처리 알고리즘을 설명한다. 인텔에서 만든 강력한 영상처리 라이브러리인 OpenCV(Open Source Computer Vision)는 기초 영상처리부터 고급 수준의 영상 처리까지 다양한 알고리즘들이 함수로 구현되어 있다[2]. OpenCV는 오픈소스로서 스펙만 맞추면 자신의 알고리즘도 라이브러리에 등록시킬 수 있다는 장점이 있기 때문에 영상처리 응용 프로그램에 많이 사용된다[3,4]. 이와 같은 이유로 본 시스템 구성의 툴로 OpenCV를 사용한다.

우선 배경화면을 저장하여 움직임 검출의 참조 영상으로

사용하고, 입력되는 영상과 비교하여 움직임 유무를 확인한다. 저장된 배경화면과 입력된 영상과의 차를 구해서 임계값 이상일 경우 움직인 화소로 인식하게 된다. 이 방법은 움직임 검출 방법 중 가장 빠르게 처리할 수 있는 방법이다. 일반적으로 웹캠이 초당 30프레임을 처리하기 때문에 차영상은 움직임을 판독할 수 있는 충분한 정보를 제공한다. 먼저 배경 영상 $B_n(i, j)$ 을 저장하고 다음 영상이 입력되면 (식3)과 같이 배경 영상 프레임과 새로 입력된 영상 프레임 $I_n(i, j)$ 의 차를 계산하여 움직임이 발생한 영역을 찾는다.

$$|I_n(i, j) - B_n(i, j)| > T_n(i, j) \quad (식3)$$

즉, 배경 영상에서 현재 영상의 화소값을 뺀 차가 임계값보다 크면 움직임이 있는 영역이다. 이 방법은 움직임이 있는 모든 화소들을 찾을 수 있는 장점이 있는 반면 배경의 변화도 움직임으로 인식하는 단점도 있을 수 있다. 차영상을 구한 후에 바로 2진화를 시켜서 어떤 화소들이 움직였는지 확인하게 된다. 2진 영상 데이터는 칼라 영상의 임계값을 이용하여 흑백의 두 가지 색으로 데이터를 변환하는 것이다. 이러한 2진 영상 데이터는 처리가 단순하여 실용적인 시스템에 널리 이용되고 있다. 여기서 임계값 처리는 입력 영상의 각 화소에 대해서 명도가 있는 임계값 이상의 경우에, 대응하는 출력 영상의 화소값을 1로서, 그 외의 경우는 0으로 하는 (식4)와 같다.

$$g(i, j) = \begin{cases} 1 & \text{if } f(i, j) \geq t \\ 0 & \text{if } f(i, j) < t \end{cases} \quad (식4)$$

여기에서 $g(i, j)$, $f(i, j)$ 는 각각 영상처리 후의 위치 (i, j) 에 있는 화소의 농담값을, t 는 임계값을 나타낸다. 차영상을 2진화 시킨 후 모폴로지 연산인 침식, 팽창(열림)을 해서 잡음을 제거하게 한다. 모폴로지 연산(morphological operation)은 어떤 영상의 형태적인 면을 조작하는 과정이다[5]. 경계, 골격, 블록과 같은 영역 형태를 표현하거나 서술하는 데 있어서 유용한 영상 요소들을 추출하기 위한 도구로 수학적 형태론의 개념을 사용한다. 주로 영상 전처리 작업이나 초기 객체 분류 또는 이러한 처리 뒤에 이어서 물체의 내재된 구조를 명확히 하는데 사용한다[6]. 이것은 최외각선의 한 화소를 빼거나 더하는 작업을 통해 물체의 외

곽선이나 골격선을 간단하게 하는 과정이다. 모폴로지 기법 중에서 본 논문에서 사용한 연산은 침식(erosion), 팽창(dilation), 열림(opening), 닫힘(closing)으로 (식5)와 같이 각각 정의한다.

$$\begin{aligned} A \ominus B &= \{x : B_x \subseteq A\} \\ A \oplus B &= \bigcup_{x \in B} A_x \\ A \circ B &= (A \ominus B) \oplus B \\ A \bullet B &= (A \oplus B) \ominus B \end{aligned} \quad (식5)$$

차영상에서 임계값 이상의 영역이 발생하면 움직임 검출 과정을 통해 어느 부분에 물체가 입력되었는지 검사한다. 그러나 임계값 이하의 영역이 발생하면, 다음 프레임의 영상을 받아들이게 된다. 잡음의 임계값은 가상막 두께 +2로 기본값이 되어 있고, 사용자가 두꺼운 펜을 사용할 경우나 더 얇은 펜을 사용할 경우에는 옵션에서 수정해 주면 된다.

2진 영상을 이용하지 않고 색상변화를 이용하는 방법을 적용할 수도 있다. 영상에서 얻을 수 있는 정보 중에 가장 쉽고 빠르게 얻을 수 있는 정보가 바로 색상 정보이다. 영역 검색을 위하여 다른 알고리즘으로 (식6)과 같은 $Y C_b C_r$ 을 이용할 수도 있다. 광도를 분리해 낼 수 있기 때문에 컬러가 조명에 민감한 면을 보완할 수 있다. (식6)과 같이 RGB 컬러 영상값에 각각의 가중치를 곱해서 C_b 값과 C_r 값을 추출할 수 있다. Y 는 광도값을 C_b 값은 푸른색영역을 나타내며 C_r 은 붉은색 영역을 나타낸다.

$$\begin{aligned} Y &= (0.299 \times R) + (0.587 \times G) + (0.144 \times B) \\ C_b &= (-0.168 \times R) + (-0.331 \times G) + (0.5 \times B) \\ C_r &= (0.5 \times R) + (-0.418 \times G) + (-0.081 \times B) \end{aligned} \quad (식6)$$

사람의 살색을 이용하여 빛의 간섭을 줄일 수 있는 이 방법은 다른 물체는 인식 되지 않고 손가락만을 인식하므로, 포인터가 손가락인 경우에는 이 방법을 사용할 수 있지만, 다른 경우에는 사용이 불가능하다는 단점이 있어서 옵션으로 사용한다. 칼라 영상으로부터 흑백 영상을 만들기 위해 RGB 모두에 동일 가중치를 곱해서 흑백 영상을 나타낼 수 있으나, NTSC의 제안으로 (식6)의 Y 와 같이 RGB에서 서로 다른 가중치를 곱하는 방법을 많이 사용한다.

가상 스크린을 동작하기 위해서 물체의 움직임을 인식할

경우에 그림자가 물체로 잘못 인식되는 경우가 발생할 수 있다. 칼라 영상에서의 물체 그림자는 그림자 영역과 배경 영역간의 화소의 채도(chromaticity)변화량은 적지만 밝기 변화량은 크다는 특성을 이용하여 그림자의 오인식을 줄일 수 있다. 기존 RGB 모델에서 채도 성분을 추출하기 위해 정규화된 RGB(NRGB, normalized RGB)를 (식7)과 같이 구할 수 있다.

$$\begin{aligned} r &= R/(R+G+B) \\ g &= G/(R+G+B) \\ b &= B/(R+G+B) \end{aligned} \quad (\text{식7})$$

물체로 추출된 영역에서 물체의 화소값과 그에 해당하는 배경 화소값에 대한 NRGB를 비교하여 서로 유사한 성분을 가지고 있을 경우에 그 화소를 그림자 화소로 판단하여 제거한다. 임계값 이상의 잡음이 발생하면 영역 검색으로 어느 부분에 물체가 들어왔는지 검사하게 된다. 가로 방향을 검색한 후 세로 방향으로 검색하고 배열을 NRGB 방식으로 검사하여 그림자 영역이 있는 곳은 제거한다. 검색된 부분 영상의 NRGB 정보를 각각 r, g, b 라고 정의하고, 검색된 배경화면 영상의 NRGB 정보를 각각 r_b, g_b, b_b 라 정의하면, NRGB 임계값 T_{rgb} 는 실험 결과 0.055로써 R, G, B 색상의 차가 (식8)과 같이 모두 임계값보다 작다면 그림자 영역으로 판단한다. 여기서 웹캠은 빛의 영향을 많이 받기 때문에 임계값은 해당 장소에 따라 변할 수 있다. 그림자 영역을 제거 후에 배열 안에 남아있는 가장 높은 위치를 가상 스크린상의 포인터 위치값으로 정한다.

$$\text{object} = \begin{cases} 1 & \text{if } |r-r_b| < T_{rgb} \cap |g-g_b| < T_{rgb} \cap |b-b_b| < T_{rgb} \\ 0 & \text{otherwise} \end{cases} \quad (\text{식8})$$

한 영상 프레임만 받아들이는 경우에 움직임이 발생해서 바로 움직임으로 결정하면 오인식을 일으킬 수 있다. 화상카메라는 매초 30 프레임을 처리하기 때문에 순간적으로 잡음이 급격히 증가할 수도 있기 때문이다[7]. 따라서 본 연구에서는 실험적으로 연속적인 3 프레임의 영상에서 비슷한 부분에 같은 움직임 이벤트가 발생해야 최종적인 움직임으로 인식하도록 설정하여 연속성을 확인할 수 있도록 한다. 마우스 움직임 이벤트는 물체(손가락, 펜)가 설정된 가상막 영역의 내부로 들어와 움직임이 감지되었을 경우, 그에 알

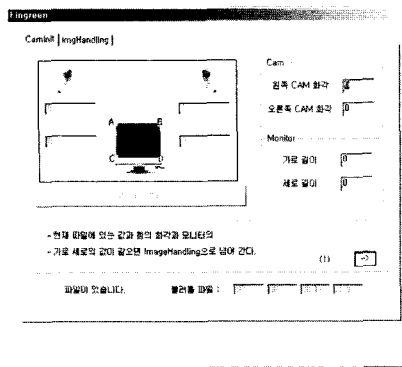
맞은 이벤트를 실행하는 부분이다. 이벤트는 이동(move), 클릭(click), 더블 클릭(double click), 드래그(drag)의 일반적인 마우스의 기능을 구현 해 놓아 일반적인 마우스 대용으로 사용이 가능할 뿐만 아니라 게임의 이벤트로도 활용이 가능하다. 각 이벤트의 정의는 다음과 같다.

- (1) 이동(move) : 물체가 설정된 가상막 영역의 내부로 들어 왔을 경우
- (2) 클릭(click) : 물체가 설정된 가상막 영역의 내부로 들어 왔다가 나간 후, 정해진 프레임 안에는 다시 들어오지 않을 경우
- (3) 더블 클릭(double click) : 물체가 설정된 가상막 영역의 내부로 들어 왔다가 나간 후, 정해진 프레임 안에 나갔던 위치에 다시 들어 왔을 경우
- (4) 드래그(drag) : 물체가 설정된 가상막 영역의 내부로 들어와 있는 상태에서 일정 시간을 위치 이동했을 경우

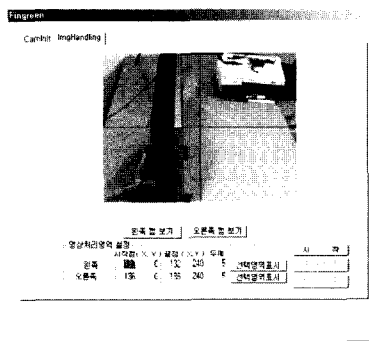
5. 실험 결과 및 분석

이 장에서는 전체 시스템의 구성 요건, 캠의 위치 설정을 하는 실용 예, 가상막 설정의 예를 설명하고 유사 시스템과의 차이점을 분석한다. 게임을 위한 가상 터치스크린 시스템을 구성하기 위해서는 일반적인 저가의 웹캠, 마이크로소프트 윈도우를 OS로 하는 컴퓨터, 인텔사의 영상처리 라이브러리인 OpenCV를 활용할 수 있는 C-컴파일러가 필요하다. [그림6]은 웹캠의 위치를 설정하는 사용자 인터페이스 화면이다. 양쪽 웹캠의 화각과 컴퓨터 모니터나 일반 빔 프로젝션용 스크린의 가로 세로 길이를 입력한 뒤 계산을 누르면 누르게 되면 웹캠의 위치 값이 자동으로 등록된다. 만약 이전 파일이 있으면 기본값을 다시 이용하고 다음 단계로 넘어간다.

[그림7]은 가상막 설정 창의 예를 보여주고 있다. 왼쪽 웹캠 보기를 누른 후 가상막을 어디에 설정할 것인지 시작점과 끝점 높이와 두께를 입력하면, 입력한 위치에 현재 설정된 가상막의 위치가 파란색 선으로 보여 지고, 선택영역표시 확인을 누르면 영상에 빨간색으로 나타난다. 같은 방법으로 오른쪽 웹캠도 가상막을 설정한다. 설정을 마친 후 시작 버튼을 누르면 터치스크린 프로그램이 실행된다. 본 연

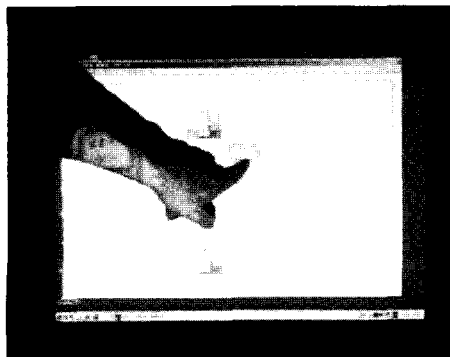


[그림6] 캠 위치 설정을 위한 사용자 인터페이스



[그림7] 가상 터치스크린 설정 예

구의 실험을 위해서 촬영 범위가 5cm에서 무한대인 저가형 VGA CMOS형 웹캠을 사용하였다. 이 웹캠은 640X480 크기의 영상을 VGA의 경우 15fps로 처리하고 CIF의 경우 30 fps로 처리할 수 있다. 그러므로 식별해상도는 세로축의 480 화소를 15인치 모니터의 대각선에 맵핑하는 경우에 해당되어, 화소당 0.03인치의 정확도를 갖는다. 간단한 게임은 물론 교육용으로도 충분히 활용이 가능한 정확도이지만 고해상도의 웹캠을 이용하면 정확도를 더욱 높일 수 있다.



[그림8] 가상 터치스크린을 이용한 게임 예

[그림8]은 일반 빔 프로젝션용 스크린에 가상 터치스크린을 적용하여 짝 맞추기 게임을 하는 게임 적용 예의 한 장면을 보여준다. 실제 터치스크린이 아닌 빔 프로젝션용 스크린에서도 일반 사용자가 마우스 없이 손가락으로 가상막상에서 직접 클릭 이벤트를 발생시켜 게임을 할 수 있음을 보여준다. 이와 같은 단순한 게임 이외에 정확한 위치 좌표를 요구하는 학습이나 고난이도 게임 등의 응용에도 본 연구에서 제안한 시스템을 적용할 수 있다.

이미 시장에는 PDA 형태의 터치스크린 게임은 많이 개발이 되어 있지만 크기의 제한성 때문에 다양한 형태의 게임으로 발전하지는 못하고 있다. 일반 빔 프로젝션용 스크린 정면에 스크린을 모니터링하는 카메라를 장치하여 터치스크린을 모방하는 제품이 있었지만, 설치의 용이성이 떨어지고 식별해상도가 낮아서 실용화되지 못하였다. 해외의 예로 일본 소니사에서 개발한 터치스크린은 손이 닿는 부분의 위치와 손바닥 면적을 인식해 이를 영상 이미지로 보여준다. 기본 개념은 이미 닌텐도SD 게임기에서 실현된 상태로 소니사는 이를 테이블 형태의 터치스크린으로 구성하였다. 테이블 주위에 많은 센서를 장착하여 사용자의 움직임을 감지한다. 일본의 미쯔비시 일렉트론사에서 개발한 다이아몬드터치(DiamondTouch)는 안테나 배열을 임베디드한 테이블을 이용한다[8,9,10]. 각각의 사용자 몸에 흐르는 전기신호를 안테나를 통하여 감지하여 여러 사용자가 함께 게임을 할 수 있다. 현재 32인치와 42인치의 시제품이 개발되었다. 두 경우 모두 특별한 센서나 안테나를 장착한 터치스크린 테이블을 사용하기 때문에 상당히 고가의 장비이다. 최근에 미국 애플사의 지원을 받고 뉴욕대에서 개발한 FTIR(Frustrated Total Internal Reflection) 기반 다중 터치 센싱 스크린은 기존과는 다른 새로운 개념의 터치스크린 인터페이스를 보여준다[1,11,12]. 시연에 사용된 스크린은 36" x 27" (50Hz)짜리 후면 투사 화면이고, 0.1인치 이하의 센싱이 가능하다. 이들 시스템들은 여러 가지 센서를 부착한 형태로 개발되어 다중 터치 센싱이 가능하다는 장점이 있으나 상당히 고가이기 때문에 시장성은 의문시된다. 본 연구에서 개발한 저가의 가상 터치스크린은 특별한 장치가 필요 없고 모니터나 빔 프로젝션용 스크린에도 설치가 용이하며, 상당히 높은 정확도를 갖추어서 게임 및 학습 시장에서 충분한 가능성이 있다.

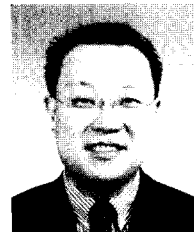
6. 결론 및 개선 방향

본 논문에서 제안한 가상 터치스크린은 사용자들의 환경에 따라 다양하게 응용할 수 있다. 일반적으로 사용하는 컴퓨터 모니터나 일반 빔 프로젝션용 스크린을 터치스크린으로 변형하여 게임을 즐기는 것도 가능하다. 현재의 터치스크린은 상당히 고가로, 기능에 비하여 많이 활용되지 못하고 있기 때문에 터치스크린을 이용한 게임은 시장성이 낮은 실정이다. 본 논문은 두 대의 웹캠만으로 언제 어디서든 일반 모니터뿐만 아니라 빔 프로젝션용 스크린을 터치스크린화 할 수 있다. 저가로 이용할 수 있고, 설치가 용이하여 새로운 형태의 게임에 응용할 수 있을 뿐만 아니라 정확도가 상당히 높아서 많은 활용분야를 가질 수 있다. 또한 키보드나 마우스를 쓸 수 없는 환경에도 적용이 가능하고, 기존의 보조 입력장치를 쓸 수 없는 사람도 이용이 가능하다. 특히 유아 교육에 적용시켜 학습 효율을 증가시키고, 손이 불편한 장애인에게는 컴퓨터 이용에 많은 도움을 줄 수 있다. 또한 기술적 특징으로는 두 웹캠으로 만들어지는 화각이 완전한 이등변 삼각형을 이루기 때문에 모니터 화면상의 모든 좌표를 정확히 수학적으로 사상시킬 수 있다는 장점이 있어서, 간단한 게임에서부터 매우 정교한 좌표를 요하는 게임이나 학습에 적용이 가능하다. 그리고 모니터 화면 앞 가상막의 위치와 두께를 설정할 수 있기 때문에 다양한 게임 형태로 발전할 수 있고, 응용 프로그램에 따라 마우스 모드의 변경이 가능하다. 그러나 두 웹캠의 교점을 이용하여 가상 터치스크린상의 좌표를 계산하기 때문에 다중 센싱에는 오류가 발생할 수도 있다. 일반적으로 구할 수 있는 웹캠은 화각이 30도 내외이기 때문에 웹캠의 거리가 15인치 모니터의 경우 50cm 정도 확보되어야 한다는 문제점도 있다. 따라서 광각 렌즈 등을 사용하여 웹캠의 화각을 넓힌다면 모니터와 웹캠의 위치가 현재의 시스템보다 상당히 좁혀질 수 있고, 광각 렌즈를 사용하여 모니터 화면이나 빔 프로젝션용 스크린의 양쪽 모서리에 웹캠을 부착한 형태로도 시스템 구성이 가능하다. 광각 렌즈를 이용한 가상 터치스크린은 계속 연구가 진행되고 있다. 본 연구에서 개발한 가상터치스크린은 가격이 저렴하고 설치가 용이하기 때문에 새로운 형태의 게임을 구현하는데 쉽게 사용할 수 있는 시스템이라 할 수 있다.

참고 문헌

- [1] <http://mrl.nyu.edu/~jhan/firtouch/>
- [2] <http://www.intel.com/technology/computing/opencv/index.htm>
- [3] 이혼진, 황미란, "OpenCV 라이브러리를 이용한 영상 처리1: 쉽고 빠른 OpenCV 영상처리의 첫걸음," 마이크로 소프트웨어, pp. 274-283, 2004년 12월.
- [4] 황미란, 이혼진, "OpenCV 라이브러리를 이용한 영상 처리2: OpenCV의 고급 영상처리 알고리즘의 활용," 마이크로 소프트웨어, pp. 272-282, 2005년 1월.
- [5] A. McAndrew, Introduction To Digital Image Processing with Matlab, Thomson Course Technology, 2004.
- [6] J. Bangham, S. Marshall, "Image and signal processing with mathematical morphology," Electronics & communication engineering journal, pp. 117-128, June 1998.
- [7] R. Cucchiara, C. Granan, M. Piccardi, A. Prati, "Detecting moving object, ghosts, and shadows in video streams," IEEE Trans. PAMI, vol. 25, pp. 1337-1342, 2003.
- [8] P. H. Dietz, D. L. Leigh, "DiamondTouch: A Multi-User Touch Technology," ACM Symposium on User Interface Software and Technology, pp. 219-226, November 2001.
- [9] E. Tse, S. Greenberg, C. Shen, C. Forlines, "Multimodal Multiplayer Tabletop Gaming," International Workshop on Pervasive Gaming Applications (PerGames), May 2006.
- [10] M. Wu, C. Shen, K. Ryall, C. Forlines, R. Balakrishnan, "Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces," IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop), pp. 185-192, January 2006.
- [11] J. Y. Han, "Low-cost multi-touch sensing through frustrated total internal reflection," in proceedings of 18th annual ACM symposium on User interface

- software and technology, pp. 115-118, 2005.
- [12] H. Benko , A. D. Wilson , P. Baudisch, "Precise selection techniques for multi-touch screens," Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 1263-1272, 2006.



김형준

1985년 2월 서강대학교 전자공학사
1987년 2월 서강대학교 전자공학석사
1996년 9월 Univ. of Pittsburgh 전자공학박사
1999년 12월 Univ. of California, Riverside 전임강사
2002년 2월 서강대학교 영상대학원 연구교수
2004년 2월 고려대학교 정보보호대학원 조교수
2005년 8월 고려대학교 정보보호연구센터 연구교수
2005년 9월~현재 한세대학교 IT학부 전임강사
관심분야: 영상신호처리, 멀티미디어, HCI, 게임시스템

논문투고일 - 2006년 6월 12일

심사완료일 - 2006년 9월 6일