

영상처리 기반의 레이저 포인터 인터랙션 시스템

김남우[†], 이승재^{**}, 이준재^{***}, 이병국^{****}

요 약

컴퓨터에서 포인팅 입력장치로써 마우스가 소개된 후로 그 발전 속도는 매우 느렸었다. 스타일러스나 터치스크린과 같은 몇 가지의 대안이 제시되긴 하였으나, 이러한 방법들은 모두 사용자가 컴퓨터와 가까운 영역에서 사용하도록 설계 되었다. 특히 프레젠테이션과 같은 특별한 상황에서 사용자가 원거리로부터 인터랙션을 명령할 경우 다양한 인터랙션에 대한 부가적인 기능이 요구된다. 따라서 본 논문에서는, 영상처리를 기반으로, 경제적이면서도 어디서나 쉽게 구입이 가능한 레이저 포인터만을 이용하여 컴퓨터로부터 멀리 떨어진 위치에서도 사용자가 마우스와 같은 인터랙션이 가능한 장비를 제안한다. 또한, 평면 스크린에서 뿐만 아니라, 가상현실 및 3차원 입체영상 구성을 위하여 오늘날 많이 사용되어지고 있는 타원형이나 휨이 있는 스크린에서도 자동으로 보정이 가능한 좌표 매핑 방법을 제안하고자 한다.

Laser Pointer Interaction System Based on Image Processing

Nam-Woo Kim[†], Seung-Jae Lee^{**}, Joon-Jae Lee^{***}, Byung-Gook Lee^{****}

ABSTRACT

The evolution of input device for computer has pretty much slowed down after the introduction of mouse as pointing input device. Even though stylus and touch screen were invented later on which provide some alternatives, all these methods were designed to have close range interaction with computer. There are not many options available for user to interact with computer from afar, which is especially needed during presentation. Therefore, in this paper, we try to fill the gap by proposing a laser pointer interaction system to allow user to give pointing input command to the computer from some distance away using only laser pointer, which is cheap and readily available. With the combination of image processing based software, we could provide mouse-like pointing interaction with computer. The proposed system works well not only in currently plane screen, but also in flexible screen by incorporating the feature of non-linear coordinate mapping algorithm in our system so that our system can support non-linear environment, such as curved and flexible wall.

Key words: Laser Pointer Interaction(레이저 포인터 인터랙션), Mapping Coordinate(좌표 매핑), Mouse Function(마우스 기능), Human Computer Interaction(휴먼 컴퓨터 인터랙션)

1. 서 론

Text 기반 운영체제에서는 인터랙션을 발생시키

기 위한 대표적 장비로 키보드가 있었으나, GUI기반의 운영체제로 변화되면서, 마우스와 같은 시각적 인터랙션의 제공이 필요하게 되었다. 하지만 이러한 장

※ 교신저자(Corresponding Author) : 이병국, 주소 : 부산광역시 사상구 주례 2동 산 69-1 (617-716), 전화 : 051) 320-1727, FAX : 051)327-8955, E-mail : lbg@dongseo.ac.kr

접수일 : 2007년 6월 28일, 완료일 : 2008년 1월 18일
[†] 준회원, 동서대학교 소프트웨어전문대학원 영상콘텐츠학과 수료(박사)
(E-mail : knw@dit.dongseo.ac.kr)

^{**} 동서대학교 소프트웨어전문대학원 영상콘텐츠학과 졸업(석사)

(E-mail : frog3147@dit.dongseo.ac.kr)

^{***} 종신회원, 계명대학교 게임모바일콘텐츠학과 부교수
(E-mail : joonlee@kmu.ac.kr)

^{****} 동서대학교 컴퓨터정보공학부 부교수

※ 본 연구는 산업자원부와 한국산업기술재단의 지역혁신인력양성사업으로 수행된 연구결과임

비들은 컴퓨터와 사용자 사이에 직접 연결되어 동작하는 지역적 제약을 안고 있다. 유비쿼터스란 용어의 일반화로 인하여 오늘날에는 컴퓨터 시스템으로부터 독립적이며, 지역적 한계를 탈피하기 위한 연구가 많이 이루어지고 있는데, Human Computer Interfaces (HCI) 및 Computer Vision(CV)등의 분야에서도 많은 관심이 증대되고 있으며 중심적 주제로 많이 다루어지고 있다. 그 결과, 전자칠판과 같은 장비들이 연구되어졌으나 외적 부피로 인한 이동의 불편함과 다양한 환경변화에 적응적으로 대처하지 못하는 또 다른 문제점이 발생하였다.

Kirstein and Muller등은 X-Window 기반에서 동작하는 레이저를 이용한 인터랙션 방법을 제안하였으나, 레이저를 인식하기 위한 클라이언트로부터 네트워크를 통하여 프레임을 전송받은 서버에서 좌표를 매핑하기 위한 연산을 수행해야 하는 시스템의 특성상 적은 수의 프레임으로 처리를 해야 하기 때문에 성능이 좋지 않다[1].

Olsen and Nielsen등에 의해 X_Web이라는 아키텍처가 구성되었으나 그림 1과 같은 커서 모델에 의한 패턴 분석으로 인터랙션이 동작한다. 이는 사용자가 인터랙션을 위하여 일정한 영역을 선택하면 커서 모델이 그려진 UI가 발생하게 되고 그 중에서 사용자가 레이저 포인터를 활용하여 원하는 UI에 위치시키면, 레이저 포인터의 위치를 카메라가 인식하여 해당 인터랙션을 발생시키는 방식으로 구성되었다. 이는 매번 사용자가 명령을 위하여 반복적으로 UI를 발생시킨 후 원하는 커서모델에 레이저 포인터를 위치시켜야 하며, 빠른 명령에 부적합한 문제점을 안고 있는 것이다[2].

Sukthankar, Stockton 및 Mullin에 의해, 위의 두 가지 방식과는 다르게 인터랙션과 영상처리가 한 대의 컴퓨터에서 모두 처리가 되는 방식이 도입되었으나, 사용자가 일반적으로 마우스와 같은 인터랙션 장비로 대체하여 사용하는 형태가 아니라, 프레젠테이션만을 위한 목적으로 구현되었다[3]. 그림 2와 같이 프레젠테이션을 위한 페이지의 모서리에 각각의



그림 1. 커서 모델

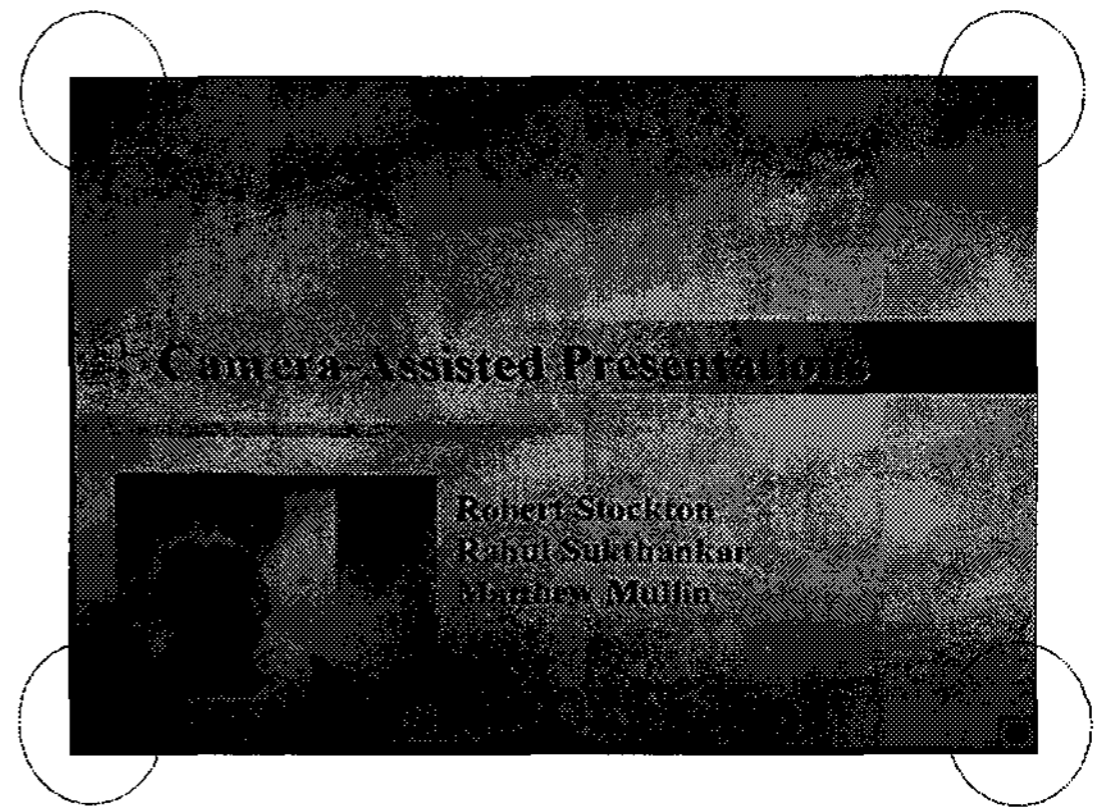


그림 2. 프레젠테이션 인터페이스

인터랙션을 위한 UI를 삽입한 후, 사용자의 레이저 포인터가 영역 내에서 확인될 경우 해당 인터랙션을 발생시키는 방법을 취한다. 이는 단순히 프레젠테이션 시에만 가능하며, 일반적인 어플리케이션에서 적용하기에는 부적합한 형태로 구성되어 있는 것이다.

그 후에도 많은 연구가 진행되어 왔으며, Rahul Sukthankar, Robert G. Stockton, Matthew D. Mullin등에 의해 본 시스템과 가장 유사한 형태로까지 연구가 되었다[4]. 하지만 이 또한, 프레젠테이션과 같은 특정 어플리케이션에서만 적용되는 형태의 가상버튼을 두어 동작하는 형태로 구성하였다. 기존의 연구와 차이점이라면 카메라와 비춰지는 영상간의 키스톤 조절이 자동으로 연산되어 정확한 값을 도출할 수 있었으나 선형 스크린에서만 가능하였고, 오늘날 가상환경 및 3차원 입체영상 구성을 위하여 사용빈도가 증가하고 있는 곡면 스크린에서의 적용은 불가능하다.

Buntarou Shizuki *et al.*은 위에서 언급한 논문들과는 다르게 다양한 어플리케이션에 적용 가능한 연구를 하였으나, 아래 그림 3과 같이, 어플리케이션을 구성하고 있는 네 개의 면에서 레이저 포인터가 내부에 존재를 하는지 또는 내부에서 인식이 불가능하게 되었는지를 판단함으로써 패턴에 의한 적응적 동작을 발생하도록 하였다[5].

본 논문에서는, 경제적이면서 주변에서 흔히 접할 수 있는 레이저 포인터와 USB 2.0 카메라를 이용하여 기존의 장비를 대체할 수 있는 원격제어의 마우스 인터랙션 시스템을 개발하고자 한다. 제안한 시스템은, 기존의 앞에서 언급된 장비들과는 차별화될 수

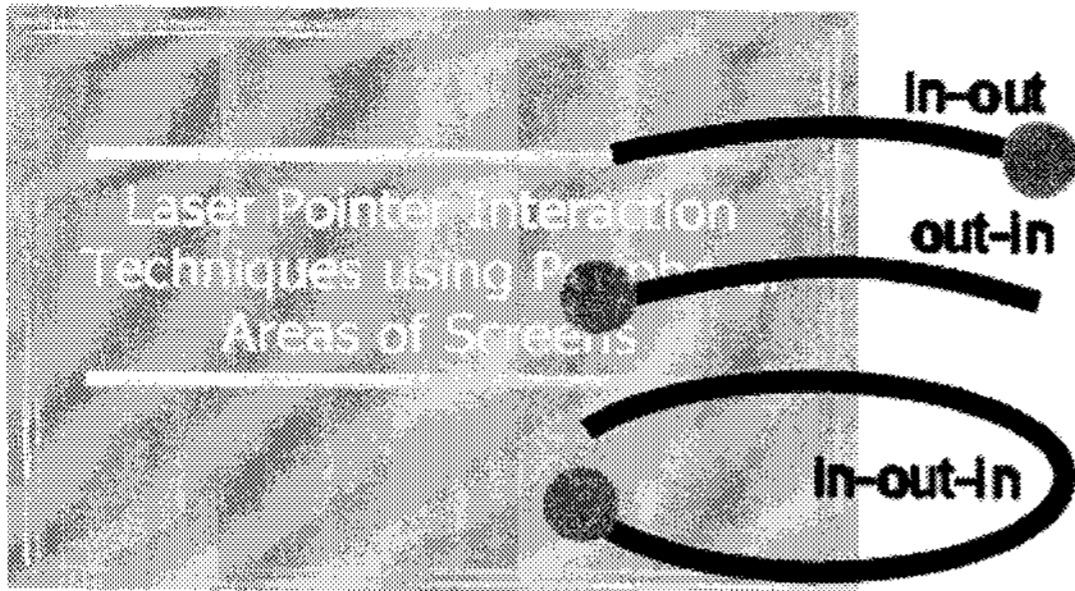


그림 3. 궤적을 이용한 인터랙션

있도록 가상버튼 및 커서 모델의 패턴 등을 지양하고, 레이저 포인터만으로도 마우스와 동일한 방법으로 동작할 수 있도록 최대한 용이하게 구성하였다.

또한, Rahul Sukthankar *et al.*이 제안한 일반적인 평면 스크린 뿐만 아니라, 오늘날 가상현실 및 3차원 입체영상에서 자주 사용되는 곡면이나 휨이 있는 스크린에서도 자동으로 캘리브레이션이 가능한 기법이 적용되어 스크린의 일그러짐을 자동으로 보정해 줌으로서 사용자가 지시하는 위치를 더욱 정확하게 인식할 수 있고, 실시간적인 사용자의 인터랙션 명령 요구에도 부응할 수 있도록 설계하였다[6].

2. 시스템의 구성

본 연구에서 구현한 시스템은 그림 4와 같이, 컴퓨터에 빔 프로젝터와 PC 카메라가 연결이 되어있고, 빔 프로젝터에 의해 스크린에 투영된 전체 영상을 획득할 수 있는 충분한 거리에 카메라를 위치시킨다. 사용자의 컴퓨터와 빔 프로젝터에 의해 스크린 상에 영상이 투영되면, 사용자가 원하는 마우스 인터랙션을 발생시키기 위하여 원거리에서 레이저 포인터를

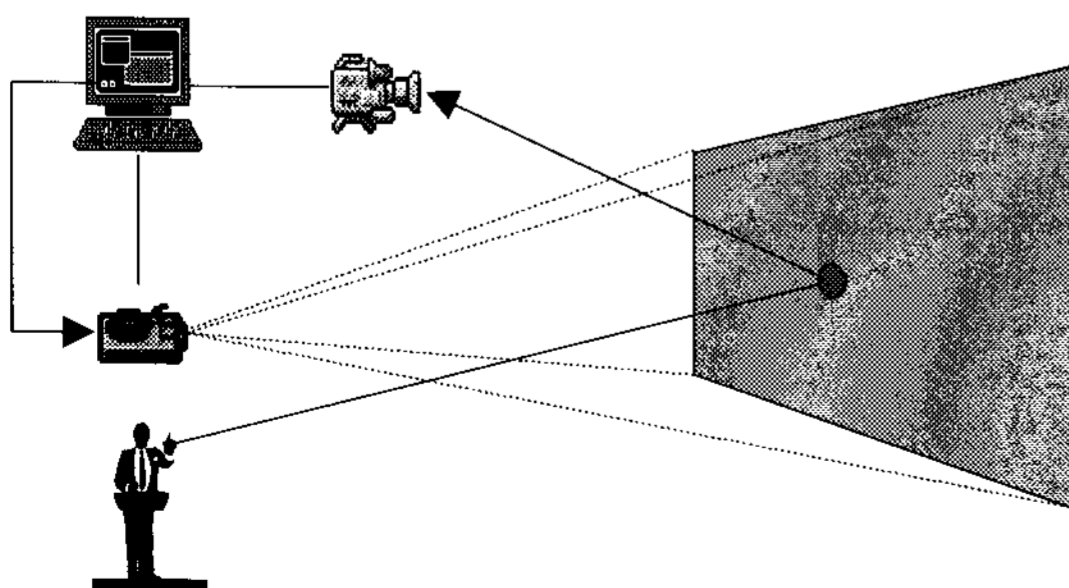


그림 4. 마우스 인터랙션 시스템의 구성

이용하여 스크린에 지시를 하게 되는데, 이때 컴퓨터에 연결되어 있는 PC 카메라가 획득한 영상을 활용하여 영상을 처리함으로써 레이저 포인터의 위치 및 사용자가 요구하는 다양한 인터랙션을 구분하게 된다.

마우스 인터랙션 시스템은 그림 5, 그림 7과 같이, 전체 시스템 구성을 위한 하드웨어 부분과 영상처리를 위한 소프트웨어 부분으로 나뉜다.

그림 5에서와 같이, 하드웨어 시스템은 컴퓨터상의 영상을 스크린으로 투영하기 위한 빔 프로젝터, 스크린 상에 투영되어진 영상위에서 사용자가 인터랙션을 발생시킬 레이저 포인터, 스크린 상에 투영된 영상을 획득하기 위한 PC 카메라와 레이저 필터, 전체 프로세싱을 담당할 컴퓨터 등으로 구성된다. 레이저 포인터는, 강의나 프레젠테이션 시 흔히 사용하는 가장 일반적인 모델을 사용하였고, 카메라는 USB 2.0 PC카메라로 구성되었으며, 30(f/s)의 속도로 동작을 하며, 640×480의 해상도를 가지는 모델이다. 빔이 투영되는 스크린은 곡면 스크린을 사용하였고, 좌표 매핑 또한 자동으로 연산되어 진다. 그림 6은 본 실험에서 사용된 USB 2.0 카메라와 레이저 포인터이다.

그림 6에서 보이고 있는 카메라와 레이저 포인터의 제원은 아래 표1과 같다. USB 2.0 카메라는 최대 1280×960의 해상도를 가지지만, 해상도와 프레임 속

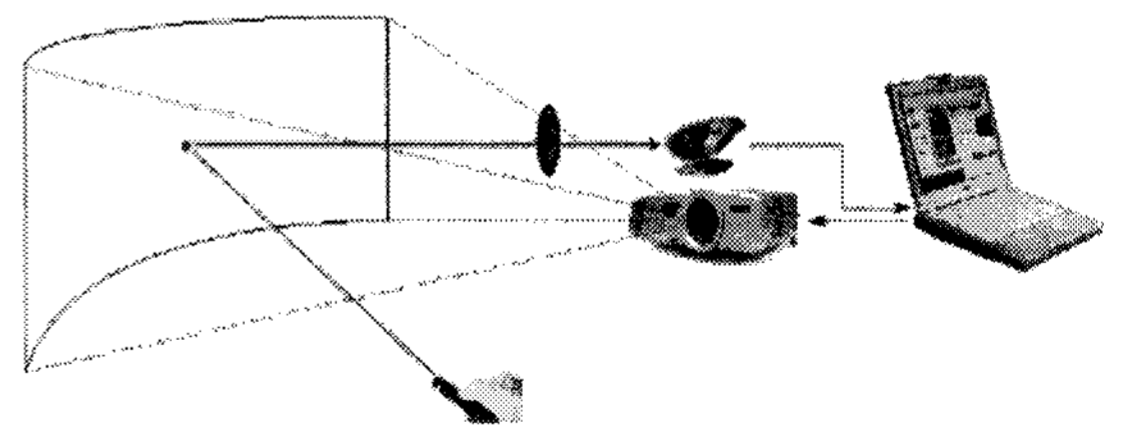
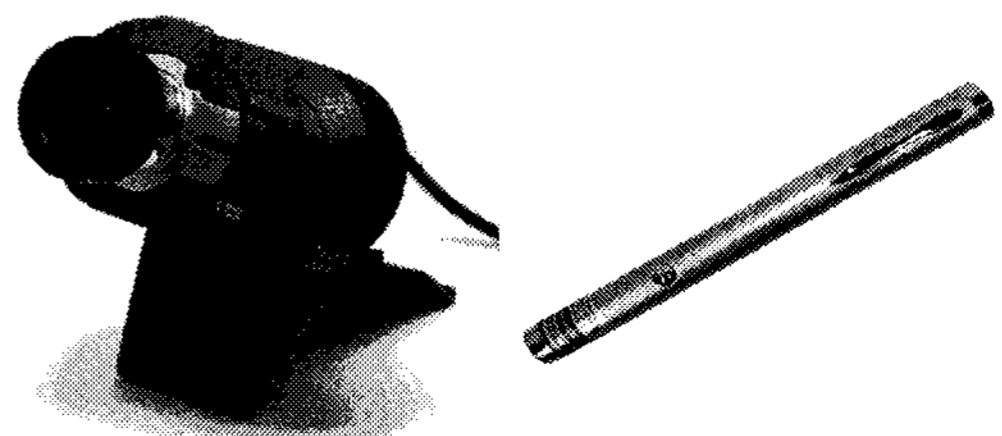


그림 5. 하드웨어 구성



(a) USB 2.0 카메라

(b) 레이저 포인터

그림 6. 카메라와 레이저 포인터

표 1. 카메라와 레이저 포인터의 제원

PC 카메라	레이저 포인터
<ul style="list-style-type: none"> • USB 2.0 • 1280 × 960 (Max) • 24 Bit, True color • 30 f/s (Max) • 4.3mm ~ ∞ 	<ul style="list-style-type: none"> • Red : 650 ~ 670nm • Green : 515 ~ 535nm

도가 반비례하는 문제점을 감안하여, 본 연구에서는 최대 프레임인 30(f/s)의 속도를 유지하면서 자연스러운 레이저 포인터 동작이 가능한 640×480 해상도에서 실험하였다.

레이저 포인터는 일반적으로 사용되어지고 있는 적색과 녹색에 대하여 실험하였고, 각각의 파장 영역의 차이로 인하여 칼라별 실험이 가능하였다.

소프트웨어 부분은 입력된 영상으로부터 영상처리를 이용하여 레이저 포인터 검출 모듈, 마우스 인터랙션을 발생시키기 위한 모듈 및 정확한 포인터의 위치를 파악하기 위한 캘리브레이션 모듈로 구성된다. 소프트웨어에 의한 프로그램의 순서는 시스템 흐름을 나타내고 있는 아래 그림 7과 같이 빔에 의해 투영된 영상을 PC 카메라를 이용하여 영상을 획득하고 획득된 영상으로 영상처리를 거쳐 레이저 패턴에 의하여 인터랙션을 구분지은 후, 카메라 캘리브레이션에 적용함으로써 사용자가 원하는 위치에 인터랙션 명령을 실행하는 단계로 구성이 되어있다.

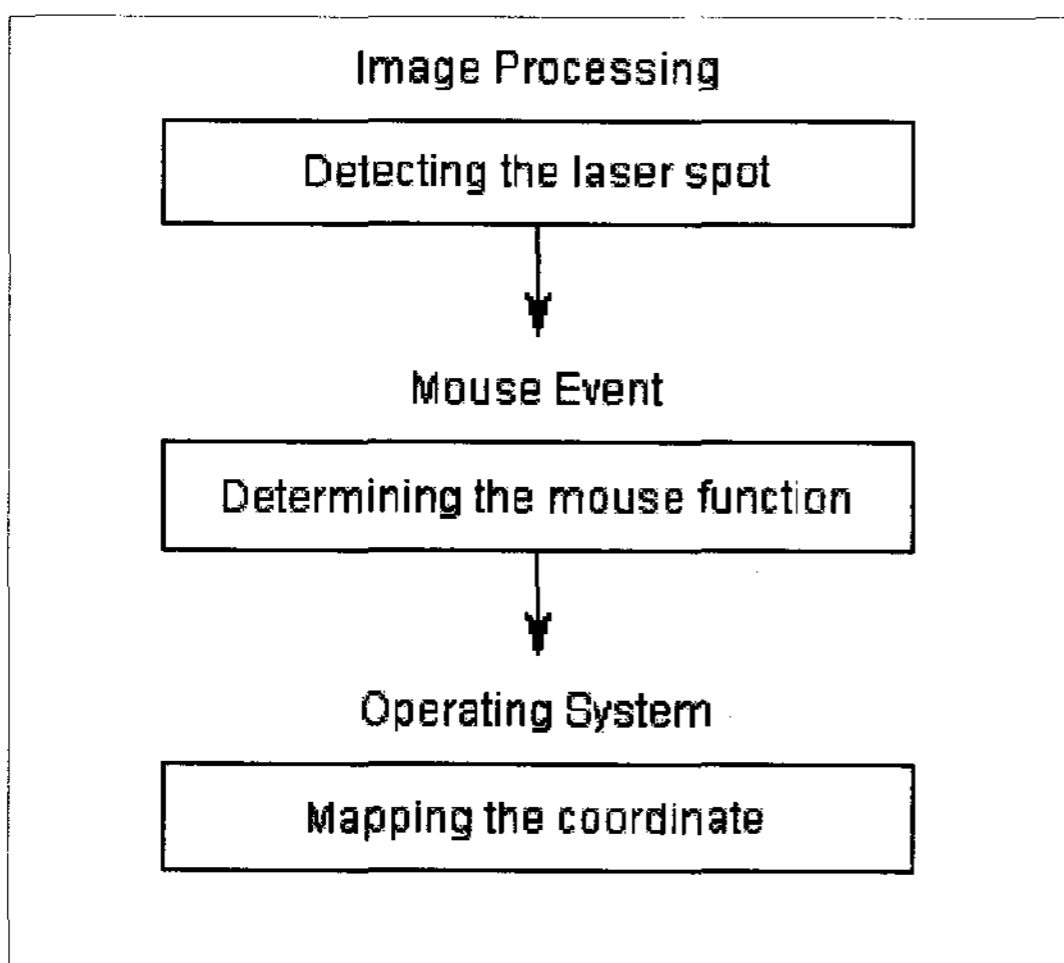


그림 7. 소프트웨어 구성

3. 레이저 포인터 인식 기술

레이저는 한 가지의 파장만으로 구성된 빛을 일컫는데, 빛의 퍼짐 또한 일반적인 빛에 비해 매우 작다. 따라서 선택된 파장만을 통과시키거나 단일 색만을 추출할 수 있다면 우리가 원하는 레이저 빛의 추출은 가능하게 된다. 일반적으로 사용하는 적색의 레이저 포인터는 650~670nm의 파장에 속하고, 녹색의 레이저 포인터는 515~535nm의 파장에 속하며, 밝은 적색 또는 녹색의 가시광선 영역에 해당하는 단색성의 특징을 가진다. 따라서 레이저 포인터를 인식하기 위한 방법으로, 칼라기반 영상처리를 이용하는 방법과 파장을 이용하여 레이저 포인터를 인식 하는 두 가지 방법으로 구분할 수 있다.

빔 프로젝트에 의해 스크린 상에 투영된 영상은 USB2.0 PC카메라에 의해 640×480 사이즈, 30(f/s)속도로 영상을 획득된다. 영상획득 시 외부환경에 카메라를 자동노출조절이 되면 설정된 값과의 오차로 인하여 레이저 포인터가 잡음으로 감지된다. 따라서 자동노출을 배제시킴으로서 레이저 포인터의 값을 일정하게 유지할 수 있어야만 하는데, 본 연구에서 사용되어진 PC 카메라를 사용한 이유도 자동노출의 동작 여부를 선택할 수 있는 특성이 있었기 때문이다. 영상에 맺힌 레이저 포인터를 인식하기 위하여 잡음 및 기타 물체와의 분리과정을 거쳐야 하는데, 획득된 영상을 구성하고 있는 각 픽셀의 칼라 값, 즉 각 픽셀의 R, G, B 컴포넌트를 추출하고, 임계값을 이용한 영상 이진화 단계를 거치게 된다. 이렇게 인식된 물체 중에서 영역기반 알고리즘을 통하여 그 중심 값을 확보하고, 이로써 포인터의 정확한 좌표 추출을 가능하게 하였으며, 외부환경이나 기타요인에 민감하게 반응하는 카메라의 특성상 변화하는 환경에서도 레이저 포인터를 안정적으로 추출하기 위한 방안으로 레이저 포인터가 가지는 파장의 영역만을 투과시키는 필터를 사용하였다.

3.1 칼라 컴포넌트를 이용한 인식

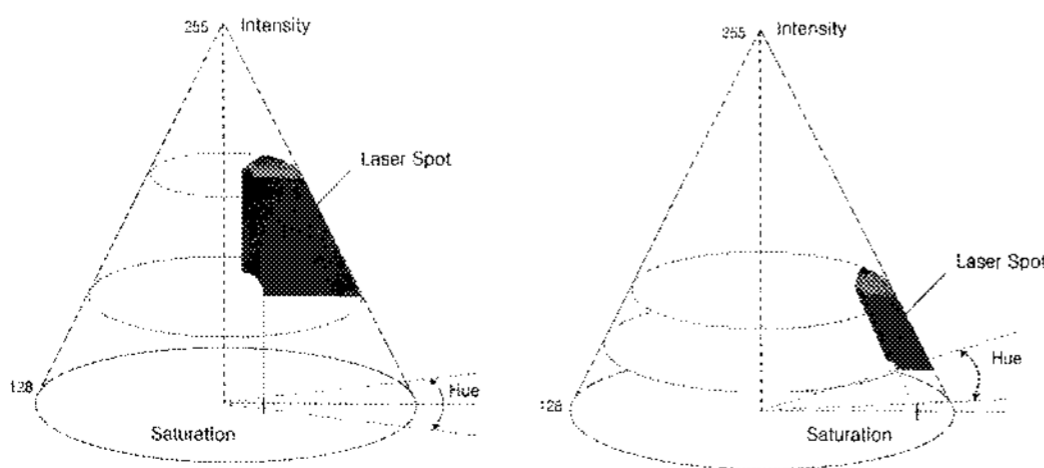
레이저 포인터가 가지는 단색성의 특성을 이용하여 영상처리를 활용함으로써 레이저 포인터에 해당하는 단일색 만을 추출할 수 있다면 우리가 원하는 레이저 빛의 추출이 가능하게 된다. 칼라색상을 표준적으로 표현하기 위하여 RGB, YIQ, CMY, CMYK등

과 같은 다양한 칼라모델이 있는데, 흔히 가장 간단히 영상을 처리하기 위하여서는 R, G, B 칼라모델을 사용하지만, 인간이 눈을 이용하여 색을 인지하는 방법과 가장 유사한 방법으로는 H, S, I 모델을 들 수 있다. 이는 R, G, B 모델이 빛의 기본 3색인 적색, 녹색, 청색으로 구성된 원리에 착안한데 반해, 인간이 눈으로 색을 구분하기 위하여 사용하는 색상(Hue), 채도(Saturation), 명도(Intensity)에 의한 컴포넌트로 구성되어 있다. 이 모델을 사용하면, 어떤 구체적인 컬러를 만들기 위해 색을 조합할 필요가 없을 뿐만 아니라 Hue 좌표축 자체가 색상을 표현하고 있어 특정 칼라를 추출하기가 용이하다. 동일 색 중에서도 색의 농도를 결정하기 위해서는 또 다른 좌표축 중의 하나인 Saturation을 조절하면 되고, 밝기를 바꾸기 위해서는 Intensity값을 조절한다. 이와 같이, 사용자가 원하는 특정 영역의 색을 정의하기 위한 표현이 쉬워지게 되는 것이다. R, G, B 모델에서 H, S, I 모델로 변환하기 위해서는 다음 식(1)과 같은 칼라 값의 변환 과정이 필요하다.

$$\begin{aligned}
 I &= \frac{1}{3}(R+G+B) \\
 S &= 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \\
 H &= \cos^{-1} \left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{((R-G)^2 + (R-B)(G-B))}} \right] \\
 \text{if}(B > G) \quad H &= 360^\circ - H
 \end{aligned}
 \tag{1}$$

R, G, B 모델에서 H, S, I 모델로 변환 하였을 경우에 레이저 포인터로 인식되는 H, S, I 칼라모델에서의 영역을 아래 그림 8에서 잘 표현하고 있다.

그림 8(a)에서의 영역은 레이저 필터를 사용하지 않고 레이저 포인터의 칼라 값을 카메라로 인식하였



(a) 레이저 필터 미적용 (b) 레이저 필터 적용

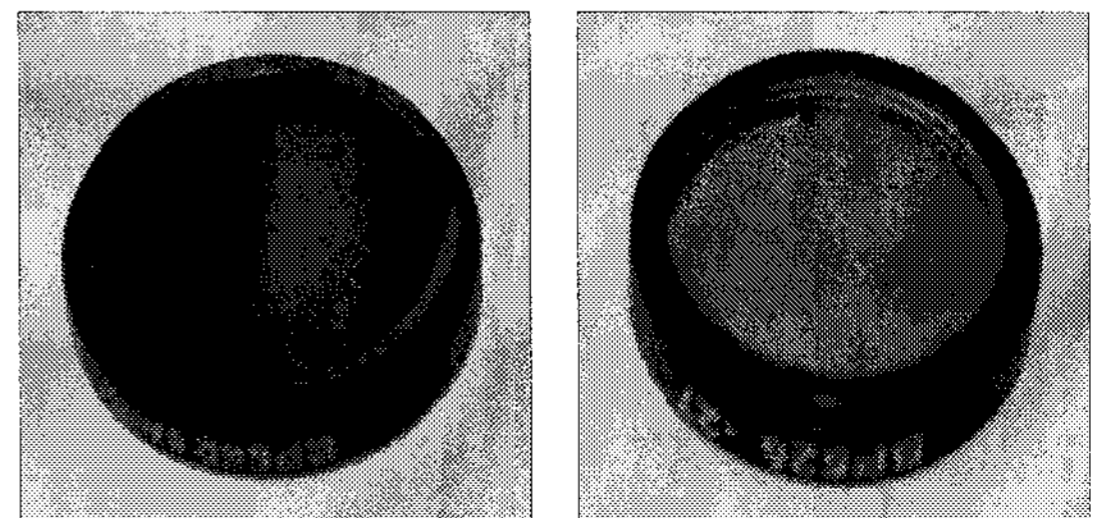
그림 8. H, S, I 칼라모델에서의 레이저 포인터 영역

을 경우의 영역이고, 그림 8(b)에서의 영역은 레이저 필터를 사용하였을 경우의 영역을 표현하고 있다. 위 그림에서 보는 바와 같이, 그림 8(a)에서의 영역보다 그림 8(b)에서의 영역이 훨씬 좁음을 쉽게 알 수 있다. 이것은 레이저 필터를 장착하였을 경우는 그렇지 못한 경우보다 레이저 포인터의 영역이 훨씬 축소됨으로써 다른 물체와의 구분 또한 용이해지는 것을 표현한다.

3.2 대역통과 광학 필터(Optical Band Pass Filter)를 이용한 인식

칼라 컴포넌트만을 이용하여 레이저 포인터를 인식하기 위해서는 H, S, I 칼라모델에서 레이저 포인터라고 정의된 영역에 존재하는 칼라를 추출하게 되는데, 화면을 구성하고 있는 바탕의 그림이나 내용이 가지고 있는 색이 레이저 포인터와 비슷한 색상이나 밝기를 가진 경우는 구별하기가 쉽지 않다. 뿐만 아니라, 영상획득 과정에서 카메라의 자동노출 등의 문제로 인하여 전체 영상의 칼라 값이 변질될 우려도 있다. 따라서 이러한 문제를 해결하기 위하여 레이저 필터를 이용함으로써 레이저 포인터의 추출이 용이하게 하였다.

일반적으로 사용하는 적색 레이저 포인터인 경우, 650~670nm이고, 녹색일 경우에는 515~535nm의 파장을 가진다. 레이저 포인터의 파장을 이용하여 추출하기 위하여, 그림 9와 같이 적색 레이저만을 투과시키는 필터와 녹색 레이저만을 투과시키는 필터를 사용하면 레이저 포인터와 유사한 칼라나 밝기 외에도 외부적인 변화에 반응하지 않는 시스템의 구성이 가능하다. 본 시스템에 해당되는 파장대의 레이저 필터를 영상을 획득하는 USB2.0 PC 카메라의 전면부에



(a) 적색 필터(650~670nm) (b) 녹색 필터(515~535nm)

그림 9. 대역통과 광학 필터

장착하고 영상을 획득하면, 투영영상에 다양한 파장의 빛이 존재를 하더라도 우리가 원하는 레이저 포인터에 해당되는 빛만 획득된다.

레이저 필터를 사용하였을 경우에는 필터에 의해 정해진 파장대의 칼라만이 통과 되므로, 필터를 통과하여 인식되는 영상을 레이저 포인터로 정의하여 바로 추출이 가능하나, 필터를 사용하지 않을 경우는 각기 다른 다양한 형태의 파장이 인식되므로, 특정 값에 한하여 레이저 포인터라고 정의를 하여야 한다. PC 카메라가 획득한 영상의 값은 버퍼에 R, G, B 칼라모델로 저장이 되므로, 먼저, H, S, I 칼라모델로 변환한 후, 임계값을 만족하는 것을 선택하여야 하는데, R, G, B 칼라모델은 H, S, I 칼라모델에 정확히 대응되므로, 이를 수식으로 표현하면 아래 식(2)와 같다.

$$\begin{aligned}
 LS &= \{(H, S, I) \mid \theta_1 < H < \theta_2, \alpha < S, \beta < I\} \\
 f_{(x,y)} &= \{R_{(x,y)}, G_{(x,y)}, B_{(x,y)}\} \\
 &= \{H_{(x,y)}, S_{(x,y)}, I_{(x,y)}\}
 \end{aligned}
 \tag{2}$$

전처리 과정에서 칼라모델에 의해 변화된 영상은 이진영상으로 구분하게 되는데, 임계값 내에 존재하는 칼라 값은 레이저 포인터로 추정하여 “1”로 설정을 하고, 그 외의 나머지 값은 “0”으로 정의하여, R, G, B 모델에서 “1” 값이 존재하는 부분은 각각의 R, G, B 컴포넌트에 “255, 0, 0”의 값을 부과하면 레이저 포인터가 위치한 부분은 적색으로 표시되고, 그 외의 아닌 부분은 모두 “0, 0, 0” 값을 부과하여 흑색으로 표현함으로써 레이저 포인터의 정확한 추출여부를 가늠할 수 있게 된다. 다음 식 (3)은 이를 잘 표현해 주고 있다.

$$B(x,y) = \begin{cases} 1, & f(x,y) \in LS \\ 0, & otherwise \end{cases}
 \tag{3}$$

4. 카메라 캘리브레이션

빔 프로젝트에 의해 투영된 영상은 다양한 외부 환경에 의해서 일그러짐이 발생을 한다. 빔이 투영되는 스크린의 일그러짐이나 투영 영상 획득과정에서의 오류 등이 그 주요 요인으로 해석될 수 있다. 따라서 사용자가 원하는 정확한 위치에 마우스 이벤트를 발생시키기 위해서는 빔에 의해 투영된 영상과 카메

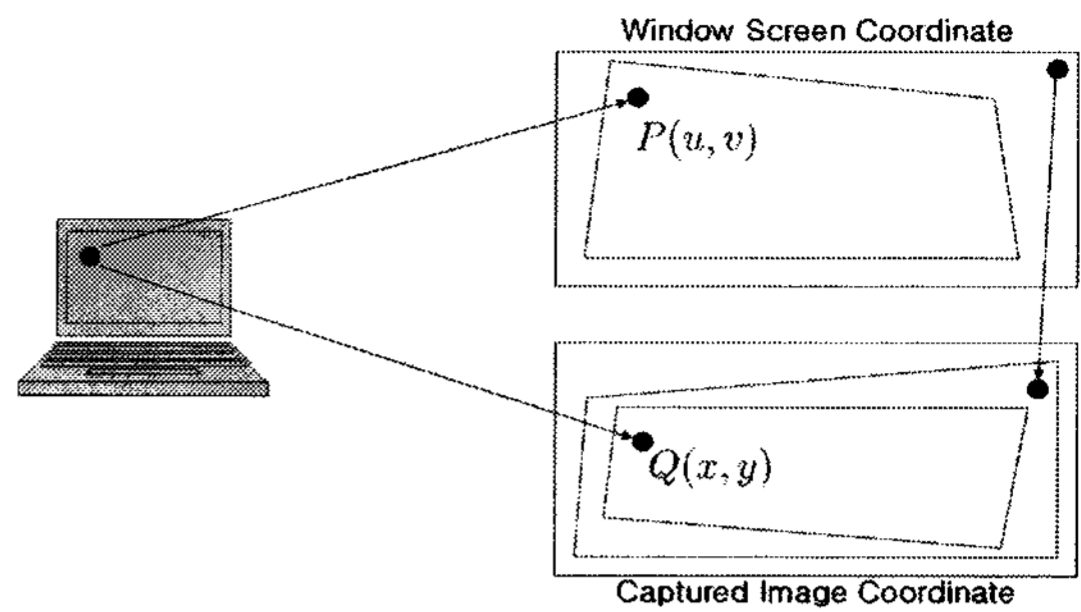


그림 10. 윈도우 좌표와 카메라 획득 영상 좌표의 매핑

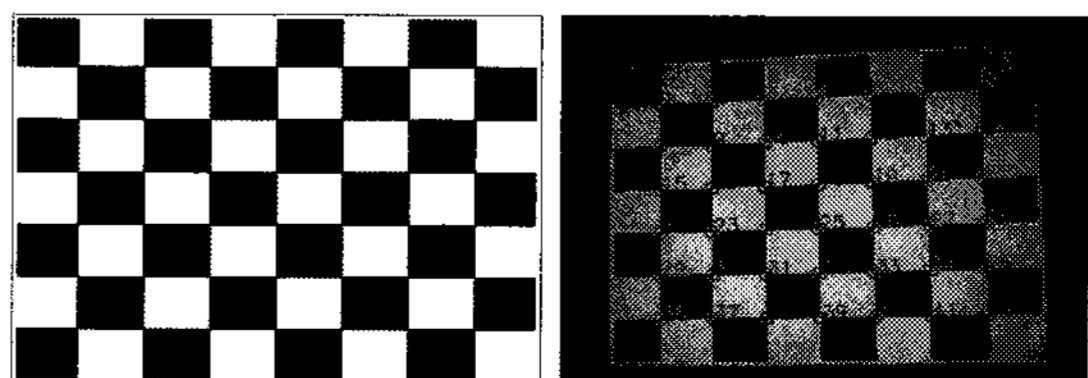
라에 의해 획득된 영상간의 관계를 파악하여야만 한다. 그림 10은 이와 같은 관계를 잘 표현해 주고 있다.

이를 위하여, 본 연구에서는 최초 어플리케이션 동작 시, 0.3초의 짧은 시간동안 아래 그림 11(a)와 같은 체스보드 이미지를 스크린 상에 프로젝션 하고, 그 동안 카메라는 본 영상을 그림 11(b)와 같이 획득하여 경계선 검출 알고리즘을 이용함으로써 각 체스보드 이미지를 구성하고 있는 코너를 인식하게 되고, 자동으로 인덱스 번호를 부여하여 동일 인덱스에 속하는 좌표간의 차를 연산하게 된다.

위 그림 10에서 나타내듯이 스크린 상의 윈도우 좌표와 카메라에 의해 획득된 영상의 좌표 간에는 오류가 존재한다. 따라서 카메라가 획득한 영상 내에서의 한 점을 $Q(x,y)$ 라 하고 윈도우 영역 내의 한 점을 $P(u,v)$ 라하면 $Q(x,y)$ 에서 $P(u,v)$ 로 변환할 수 있는 함수를 찾아내야 한다.

윈도우 좌표계는 사용자의 컴퓨터에서 사용되는 해상도를 일컫는다. 일반적으로 가장 많이 사용되는 해상도는 1024×768이고, 카메라 해상도는 640×480이다. 코너가 인식되고 나면, 두 좌표체계 간의 관계는 다음 식(4)에 의해 표현된다.

$$(u,v) = \left(\frac{P_1x + P_2y + P_3}{P_7x + P_8y + P_9}, \frac{P_4x + P_5y + P_6}{P_7x + P_8y + P_9} \right)
 \tag{4}$$



(a) 체스보드 이미지 (b) 코너 인식 및 인덱스 영상

그림 11. 캘리브레이션을 위한 체스보드 이미지

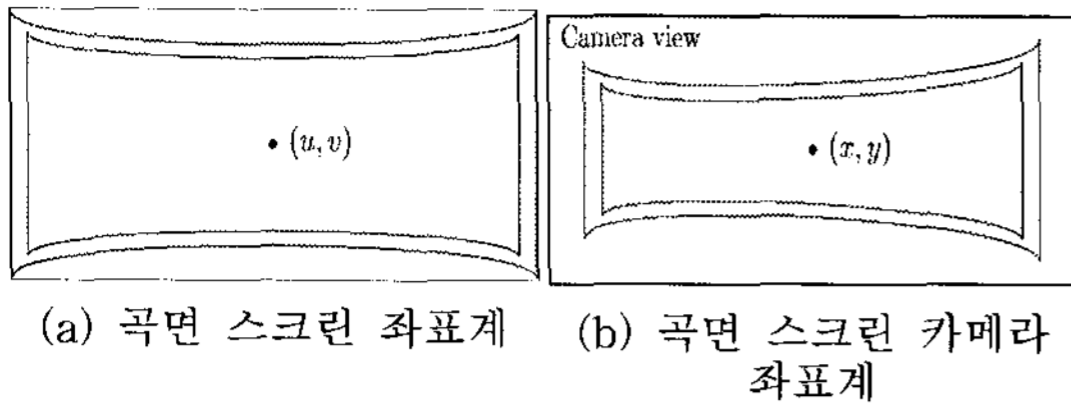


그림 12. 곡면 스크린에서 P와 q사이의 관계 $P = TQ$

위의 식(4)에서 찾고자 하는 각각의 P값을 구하기 위하여, 이미 알고 있는 값과 미지수 값으로 각각 구분하여 재 표현하면 다음과 같은 행렬 식(5)를 도출할 수 있다.

$$\begin{pmatrix} uw \\ vw \\ w \end{pmatrix} = \begin{pmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (5)$$

위 식에 의해서 $AT=0$ 라는 조건을 만족하는 T를 구하여 위 식에 대입했을 때 그 결과 값으로 나오는 (u, v) 를 찾아내면 정확하게 캘리브레이션이 이루어졌다고 할 수 있을 것이다. 하지만 현실적으로 $AT=0$ 이 되는 T를 찾는 것은 불가능하기 때문에 AT가 0에 가장 가까운 값을 가지는 T를 찾아 위 식에 대입하면 가장 정확한 캘리브레이션 결과를 얻어 낼 수 있다. 따라서 위의 식은 다음 식(6)에 의해,

$$A^T A = X \quad (6)$$

아래 식(7)과 같이 표현이 되고,

$$A[2n][9] = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1 x_1 & -u_1 y_1 & -u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1 x_1 & -v_1 y_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -u_n x_n & -u_n y_n & -u_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -v_n x_n & -v_n y_n & -v_n \end{pmatrix} \quad (7)$$

AT가 0에 가장 가까운 T를 찾아내기 위해서 위 식을 만족하는 정방행 행렬 X를 구하고 행렬 X의 고유 값과 고유벡터를 찾아낸다. 그 결과로 나온 9개의 고유 값들 중에서 가장 작은 고유 값에 대응되는 고유벡터를 T로 사용한다.

하지만 가상현실 및 3D 입체영상 등으로 인해 점차 일반화되고 있는 휼이 있거나 타원 형태로 구성된 곡면 스크린에서 이와 같은 방법만으로는 정확한 좌표매핑을 할 수가 없다. 따라서 본 논문에서는 곡면 스크린에서도 연산이 가능한 캘리브레이션 방법을 연구하였다. 곡면 스크린에서는 위 그림 12와 같이

(u, v) 와 (x, y) 의 좌표 매핑 형태를 평면 스크린에서의 캘리브레이션 방법과 동일하게 (u, v) 좌표를 다음 식(8)과 같이 대응시킨다.

$$u = \frac{p_1 x^2 + p_2 xy + p_3 y^2 + p_4 x + p_5 y + p_6}{p_{13} x^2 + p_{14} xy + p_{15} y^2 + p_{16} x + p_{17} y + p_{18}} \quad (8)$$

$$v = \frac{p_7 x^2 + p_8 xy + p_9 y^2 + p_{10} x + p_{11} y + p_{12}}{p_{13} x^2 + p_{14} xy + p_{15} y^2 + p_{16} x + p_{17} y + p_{18}}$$

식(8)을 이용하여 행렬식으로 표현하면 다음 식(9)와 같이 표현되며,

$$\begin{pmatrix} uw \\ vw \\ w \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} & p_{17} & p_{18} \end{pmatrix} \begin{pmatrix} x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{pmatrix} \quad (9)$$

고유 값과 고유벡터를 연산하기 위한 행렬식은 다음과 같이 $[2n \times 18]$ 의 행렬 형태를 갖추어 식(10)과 같이 표현 된다.

$$\begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -x_1^2 u_1 & -x_1 y_1 u_1 & -y_1^2 u_1 & -x_1 u_1 & -y_1 u_1 & -u_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 & -x_1^2 v_1 & -x_1 y_1 v_1 & -y_1^2 v_1 & -x_1 v_1 & -y_1 v_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -x_n^2 u_n & -x_n y_n u_n & -y_n^2 u_n & -x_n u_n & -y_n u_n & -u_n \\ 0 & 0 & 0 & 0 & 0 & 0 & x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 & -x_n^2 v_n & -x_n y_n v_n & -y_n^2 v_n & -x_n v_n & -y_n v_n & -v_n \end{pmatrix} \quad (10)$$

이를 이용하여 평면스크린에서 적용하는 방법과 동일하게 고유 값이 가장 작은 값을 연산하여 적용시키게 되는 것이다.

5. 마우스 기능 알고리즘

본 논문에서 원격 마우스 인터랙션을 위하여 새롭게 제안하는 알고리즘은 오늘날 흔히 사용하는 PDA에서 스타일러스를 사용할 경우 이용되는 방법에 착안하여, 스타일러스를 대체한 레이저 포인터를 인지함으로써 인터랙션을 동작토록 구성하였으며, 아래에서 설명되어질 표2에서 다양한 인터랙션의 구분을 확인할 수 있다.

5.1 Mouse Move

PC 카메라가 획득한 영상에서 레이저 포인터로

판단된 픽셀들은 평균값을 취하여 위 문단에서 설명한 캘리브레이션 계수 값을 적용함으로써 사용자가 지시한 정확한 위치 (u, v) 좌표에 마우스 커서를 위치시킬 수 있게 되는 것이다. 하지만 일반 사용자의 경우 손 떨림이나, 카메라의 특성 및 외부환경 등의 요인에 의해서도 매 프레임마다 레이저 포인터로 인식되는 픽셀이 변하여 커서가 떨리는 현상이 발생하게 된다. 따라서 다음 식(11)과 같이 임계값을 부여하여, 프레임 간 레이저 포인터의 평균 좌표간의 거리를 연산하고 거리 값이 부여된 임계값 이상일 경우에만 움직임으로 인식토록 함으로써 더욱 자연스러운 커서의 움직임을 확인할 수가 있다.

$$\|P_{t-1} - P_t\| > T \tag{11}$$

5.2 Mouse Left Button Click

Mouse Move와 같은 인터랙션 동작 중에 레이저 포인터를 한번 끌 경우, 꺼짐과 동시에 시스템은 좌표를 저장하게 된다. 이로써 1초 이내에 레이저 포인터의 재 감지가 이루어지지 않으면 저장된 좌표는 무효화 처리를 하게 되고 시간 내에 재인식이 되면, 마우스의 왼쪽 버튼 클릭으로 동작하게 된다. 이 부분에서는 Mouse Left Button Double Click과 구분을 위하여 꺼짐과 동시에 타이머가 따로 동작을 하게 되는데, 꺼지면서부터 0.5초 동안 대기하였다가 재소멸이 발생하지 않을 경우에만 Mouse Left Button Click으로 동작한다.

5.3 Mouse Left Button Double Click

레이저 포인터의 소멸이 반복적으로 두 번 발생하면 첫 번째 소멸 시에 저장해둔 좌표를 활용하게 된다. 레이저를 이용한 시스템은 원하는 위치를 정확히 지정하기가 무척 까다롭다. 따라서 사용자는 레이저가 켜져 있을 경우에 지시를 하고, 재 점등 시에는 같은 위치를 지정할 수 없으므로 소멸된 좌표를 활용하여 Mouse Left Button Double Click을 발생하는

것이 효율적이다.

5.4 Mouse Right Button Click

사용자의 레이저 포인터가 일정 범위 안에 지속적으로 속해 있기는 쉽지 않다. 대부분의 동작패턴을 분석하면 지속적으로 움직이며 인터랙션을 발생시킨다. 따라서 일정영역 내에서 포인터가 머무른 채 1초 이상 감지되면 Mouse Right Button Click을 발생한 것으로 간주하게 된다.

위에서 설명한 마우스 인터랙션들을 구분하기 위하여 레이저 포인터의 점/소등과 이전 프레임과의 위치 등을 이용하여 정리하면 표2와 같다.

6. 실험결과

본 연구에서는, 레이저 포인터만을 이용하여 마우스와 동일하게 동작하는 원격제어 인터랙션 시스템을 구성하고자 하였다.

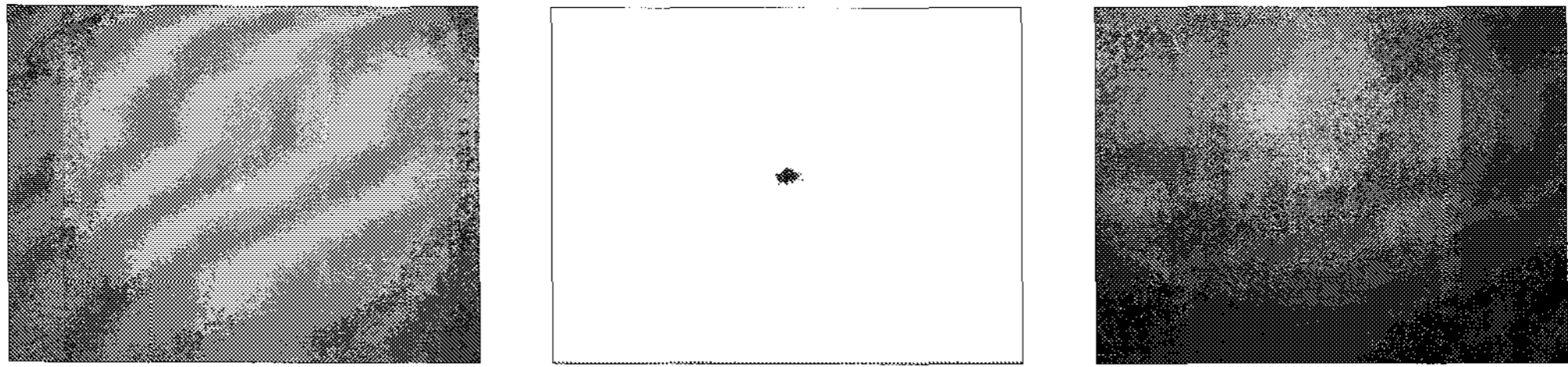
먼저, 그림 13은 칼라모델에 의하여 레이저 포인터를 추출한 결과와 레이저 필터를 이용하여 포인터를 추출한 결과를 잘 보여주고 있다. 칼라모델에 의한 레이저 포인터의 추출에 비하여 레이저 필터에 의하여 추출된 포인터의 크기가 훨씬 정교함을 확인할 수 있다.

레이저 포인터의 경우는 단파장으로서의 성질로 인하여 적색 또는 녹색 중에서도 가장 진한 채도와 가장 밝은 명도 값을 가진다. R, G, B 칼라모델은 빛의 기본 3원색을 조합하여 칼라를 표현하지만 H, S, I 모델에서는 H가 칼라를 표현하는 값이므로 H, S, I 모델에서 H에 해당되는 값만으로도 모든 R, G, B 칼라모델에 해당되는 값의 표현이 가능하다. 또한 R, G, B 칼라모델보다 S와 I의 파라미터가 더 포함되어 있어서 S와 I 값을 이용하면 R, G, B 칼라모델에 비해 H, S, I 칼라모델이 더욱 정밀하게 레이저 포인터를 추출할 수 있게 된다. 그림 14는 이와 같은 특성을 이용하여, 레이저 포인터와 유사한 칼라 값을

표 2. 마우스 인터랙션

○ 점등 ● 소등

	Mouse Move	Left Button	Left Button Double Click	Right Button
레이저 인식	○	○●○●	○●○●○●	○
이전 프레임과의 위치	≠	≠, =	≠, =	=



(a) 레이저 포인터 영상 (a) 영상 처리 영상 (a) 레이저 필터 영상

그림 13. 레이저 포인터의 인식

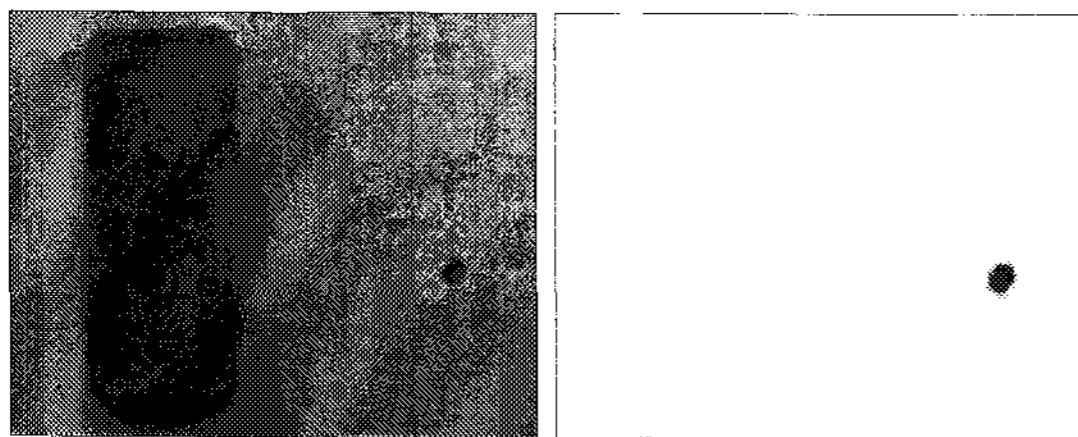
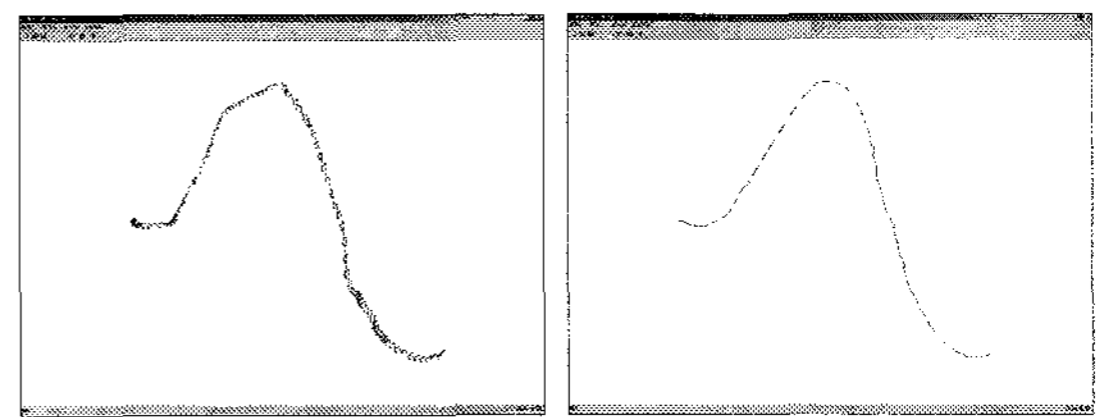


그림 14. 유사 칼라와 레이저 포인터의 구분



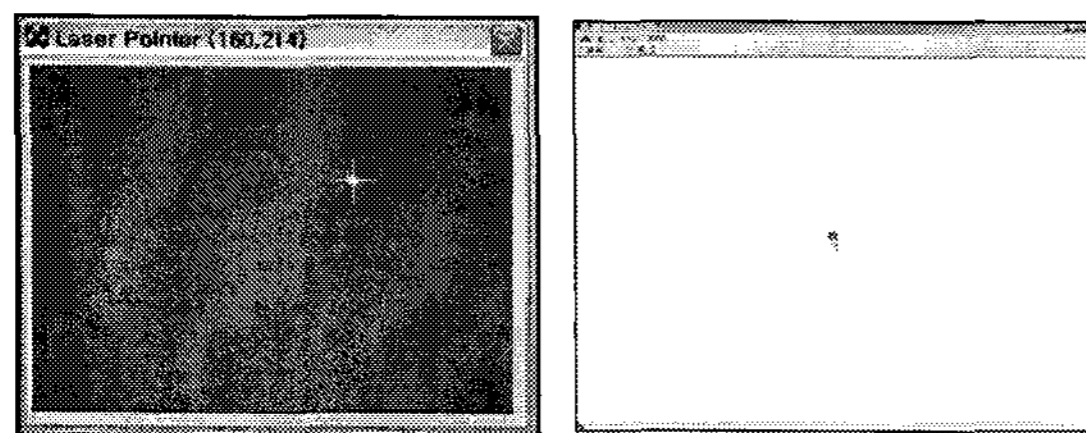
(a) 임계값 미적용 시 (b) 임계값($T=5$) 적용 시

그림 16. 레이저 포인터의 임계값 적용 유무

가지고 있는 물체가 존재하는 경우라 하더라도 레이저 포인터만을 추출할 수 있음을 잘 표현해 주고 있다.

칼라모델이나 레이저 필터를 이용하여 포인터를 추출하고 나면, 포인터로 인식된 각 픽셀들의 평균 좌표에 의하여 가운데 위치를 마우스 커서가 지시하는 위치로 정의하고, 운영체제로 하여금 커서를 이동시킨다. 아래 그림 15는 레이저 포인터를 인식한 후, 마우스 커서를 이동하여 정확히 위치시킨 결과를 보여주고 있다.

본 시스템에서, 실험을 위하여 사용한 USB 2.0 카메라의 특성이나 외부환경 및 손 떨림 등의 요인들은 레이저 포인터로 인식된 평균 픽셀의 위치를 변화시켜 커서가 떨리는 현상을 발생하게 되고, 임계값을 부여함으로써 커서의 떨림을 방지할 수 있었다. 그림 16과 같이, 왼쪽 그림 16(a)는 레이저 포인터의 평균

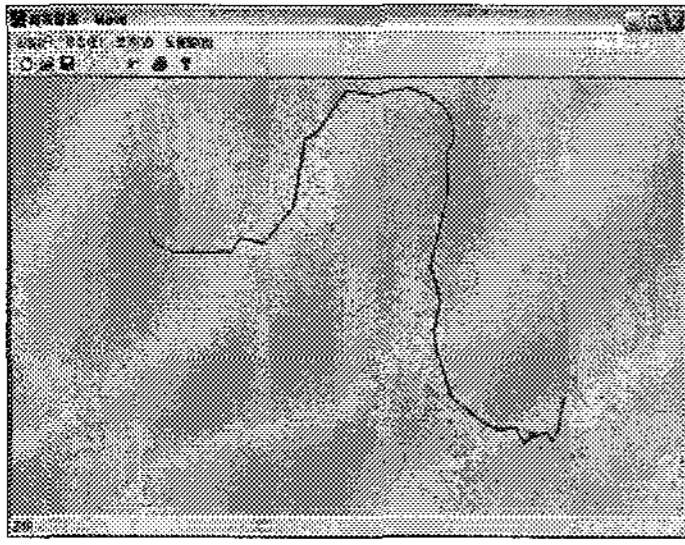


(a) 포인터 위치인식 (b) 마우스 커서 적용

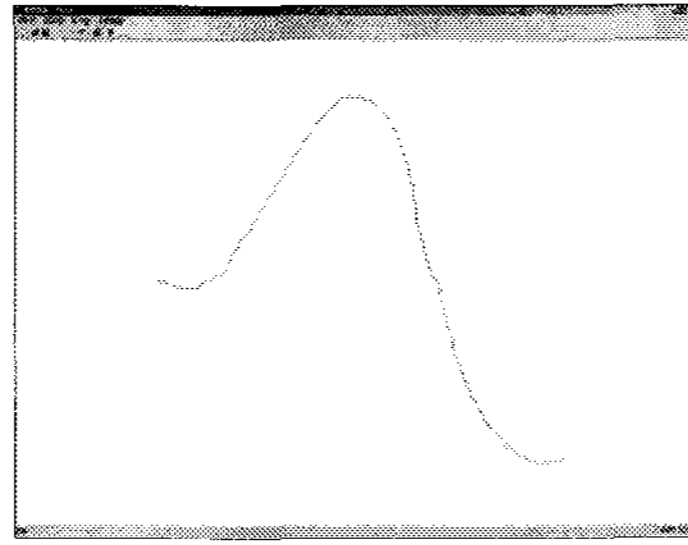
그림 15. 마우스 커서의 이동

간의 거리를 이용한 임계값을 적용하지 않아 외부환경에 의해 레이저 포인터로 인식되는 픽셀이 오인식되어 커서가 떨리는 경우이고, 오른쪽은 임계값($T=5$)를 적용하였을 경우의 실험결과이다. 아래 그림 16(b)에서 보듯이 임계값($T=5$)를 적용하였을 경우가 그렇지 못한 경우에 비해 마우스 커서의 떨림을 방지함으로써 훨씬 부드럽고 자연스러운 모습을 잘 표현해 주고 있다.

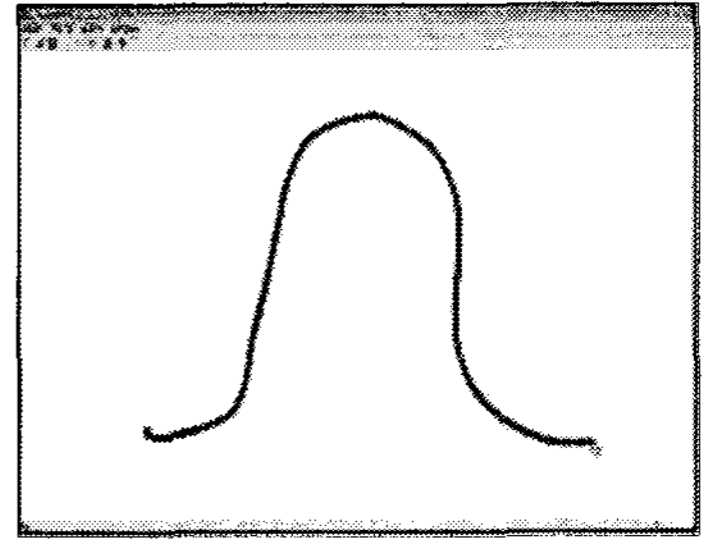
마우스가 동작하기 시작하면, 마우스의 위치를 인지하여 커서가 동작하게 되는데, 동작의 자연스러움은 USB 2.0 카메라의 프레임 수에 의해 크게 영향을 받게 된다. 따라서 카메라의 프레임에 의하여 레이저가 얼마나 부드럽게 동작하는지에 대한 실험 결과가 아래 그림 17에서 그 차이를 명확하게 확인할 수 있다. 그림 17(a)에서는 15(f/s)의 카메라를 이용하여 동작시킨 결과이고, 그림 17(b)에서는 30(f/s)의 카메라를 이용하여 동작시킨 결과이다. 또한, 마우스 동작을 아무런 가공 없이 적용하여 드로잉 기능 등에 이용하려면 카메라가 레이저 포인터를 아무리 정교하게 추출한다고 해도 그림 17(b)에서와 같이 약간의 떨림은 존재를 하게 된다. 이와 같이 하드웨어의 특성상의 오류로 인하여 포인터의 위치가 변질될 수 있으므로, 어플리케이션에 적용할 경우에는 추출된 레이저 포인터의 위치 값을 컨트롤 포인터로 이용



(a) 15(f/s)에서의 마우스 동작



(b) 30(f/s)에서의 마우스 동작



(c) 마우스 동작보정

그림 17. 카메라 프레임에 의한 마우스 동작

한 B-Spline 등의 알고리즘에 적용해 훨씬 부드럽고 향상된 움직임으로 보정한 후, 그 결과를 어플리케이션에 적용하여야 한다. 그림 17(c)는 포인터의 위치 값을 이용하여 보정한 후의 결과화면이다.

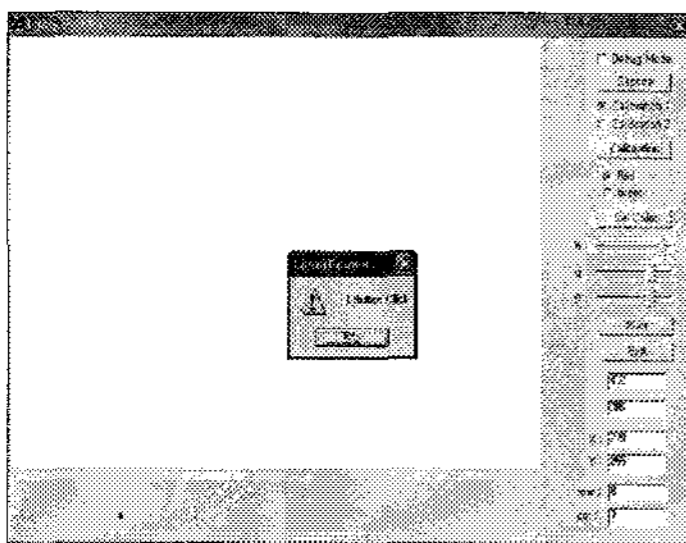
마우스의 위치 추출이 가능하다면, 사용자가 원하는 위치에서 정확한 인터랙션이 발생 가능하여야 한다. 이는 레이저 포인터의 점/소등과 소등 시에 저장해 두었던 위치를 이용하여 그 위치에서 사용자가 어떠한 인터랙션을 발생시키기를 원하는지 분석하여야 한다. 모든 인터랙션은 마우스의 버튼과 동일하게 레이저 포인터의 On/OFF 동작으로 인지하게 되며, 각각의 인터랙션이 바르게 동작하는 결과가 아래 그림 18에서 잘 표현하고 있다. 그림 18(a)는 1회 소등 시, 왼쪽 버튼의 클릭 메시지, 그림 18(b)는 1초 동안 2회 소등 시, 왼쪽 버튼더블 클릭 메시지, 그림 18(c)는 레이저 포인터가 일정 영역 내에서 소등이 되지 않으면서 1초 이상을 머무를 경우 오른쪽 버튼의 클릭으로 인지하는 결과를 보여주고 있다.

레이저 포인터를 이용하여 마우스의 모든 기능을 대체 가능하다면, 사용자가 원하는 정확한 위치를 판별하여야 한다. 이는 스크린에 투영된 영상과 카메라

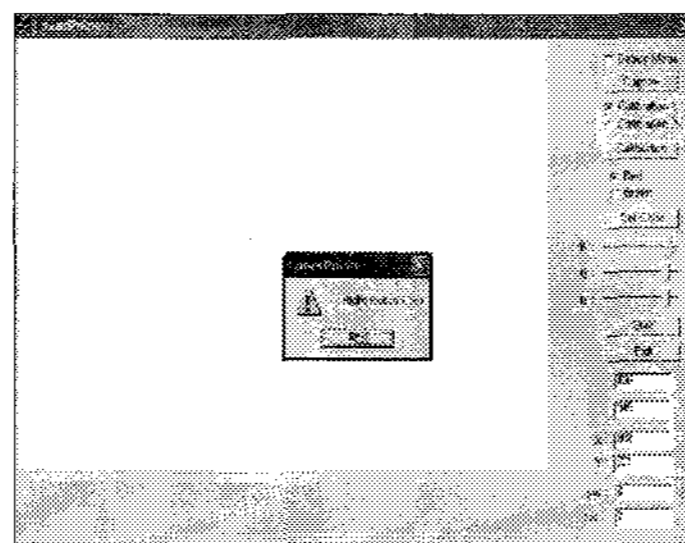
가 획득한 영상간의 오차가 발생하게 됨으로서 사용자가 지시하는 위치와 커서의 위치 사이에도 오차가 발생되기 때문이다. 따라서 정확한 좌표를 매핑하기 위한 연산이 필요한데, 이는 기존에 평면 스크린에서 적용한 기법을 사용함으로써 보정이 가능하나, 오늘날 새롭게 사용되어지는 휨이나 곡면 스크린 등의 등장으로 인하여 적용이 불가능하게 되었다. 따라서 본 실험에서는 이러한 환경에서도 가능한 캘리브레이션 기법을 적용시킴으로서 새로운 캘리브레이션 기법이 적용 가능함을 확인할 수 있었다.

캘리브레이션은 고유값의 선택에 따라 다양한 결과가 도출된다. 따라서 그림 19(a)와 같이 본 논문에서 제시한 휨 스크린 캘리브레이션 방법을 적용했을 경우 정확한 매핑이 가능하였고, 이를 활용하여 그림 19(b)와 같이 휨 스크린에서도 정확한 매핑을 확인할 수 있었다.

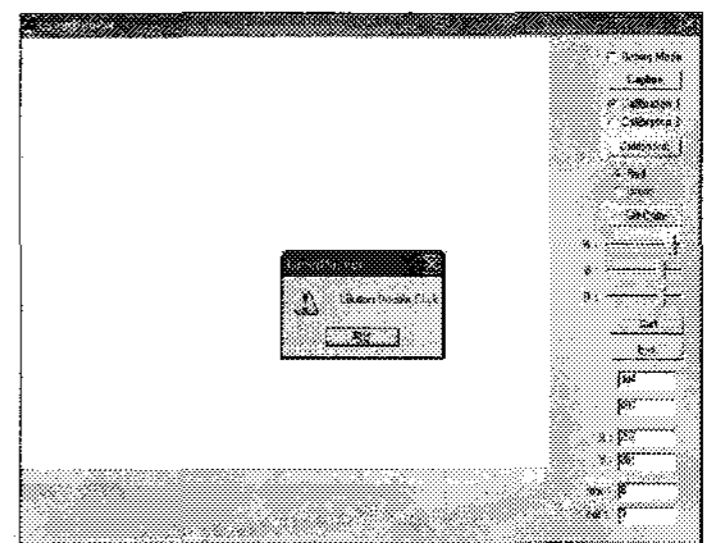
그림 20은 위에서 설명한 카메라 캘리브레이션 방법을 적용하여, 사용자가 레이저 포인터를 이용하여 지시한 위치에 마우스 커서가 정확하게 위치하는지에 대한 실험을 하였다. 그림 20(a)에서는 캘리브레이션을 적용하지 않아 레이저 포인터와 커서 간에



(a) LButton Click

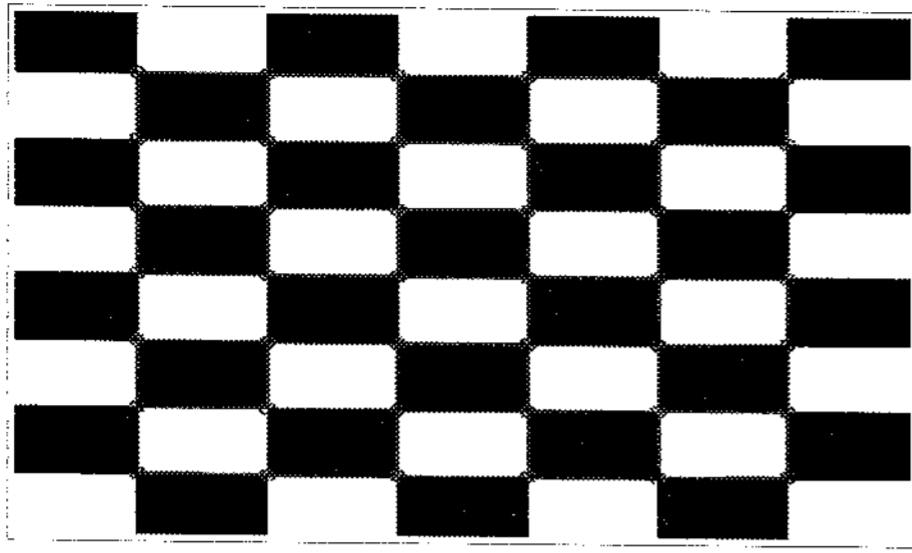


(b) LButton Double Click

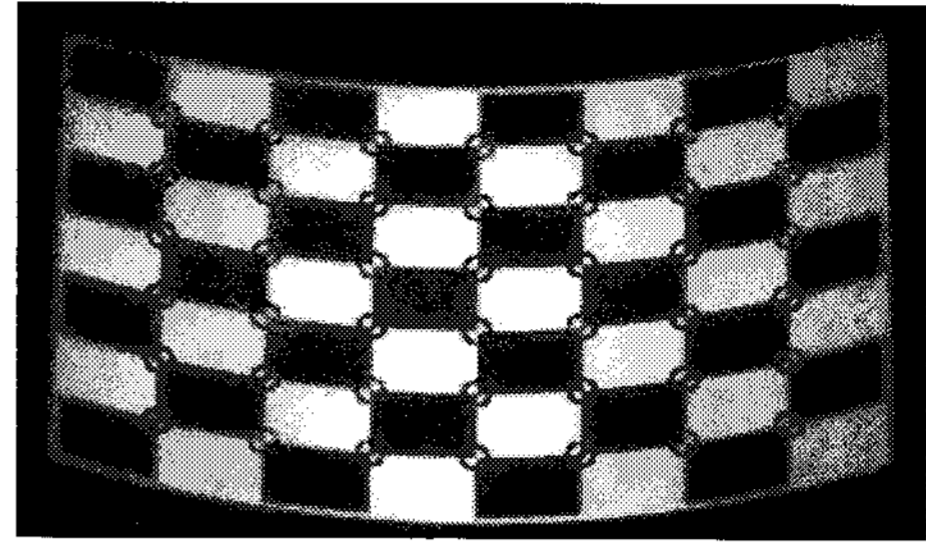


(c) RButton Click

그림 18. 마우스 인터랙션

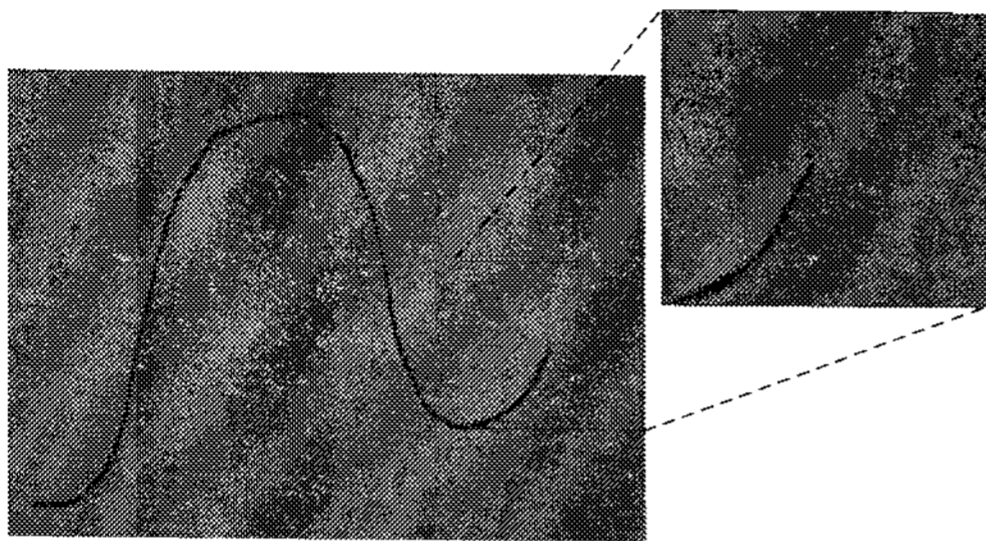


(a) 휨 스크린 캘리브레이션

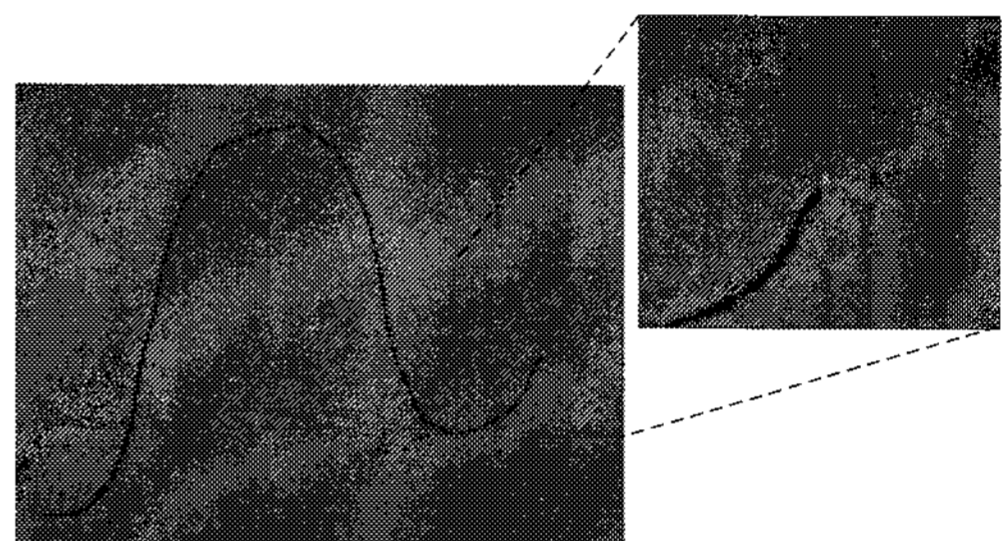


(b) 휨 스크린 캘리브레이션

그림 19. 휨 스크린 캘리브레이션



(a) 캘리브레이션 미적용



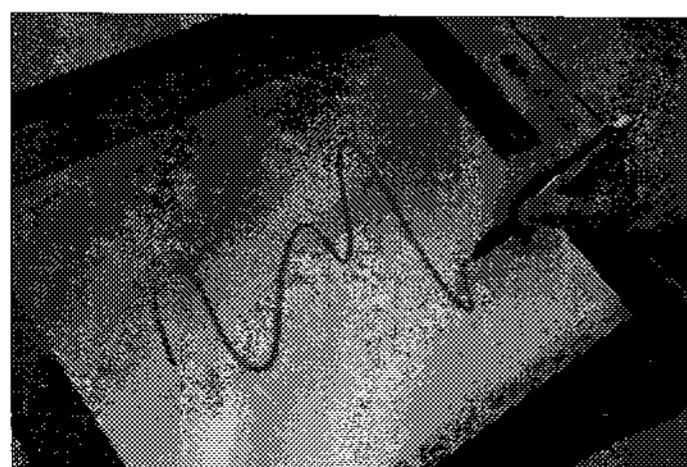
(b) 캘리브레이션 적용

그림 20. 레이저 포인터와 마우스 커서의 위치

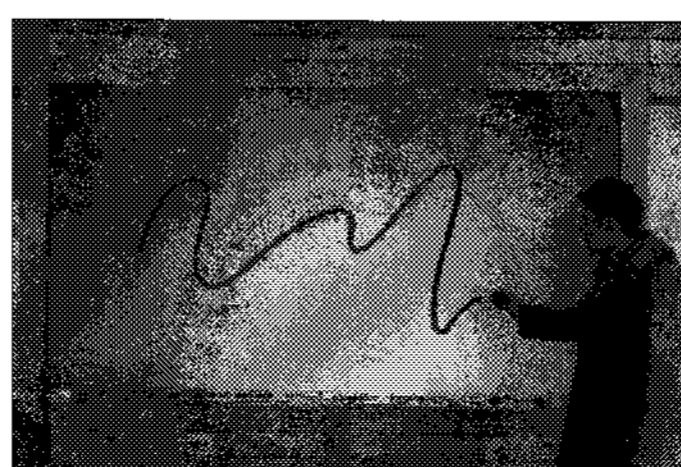
간격이 발생하였고, 그림 20(b)에서는 캘리브레이션을 적용함으로써 사용자가 원하는 위치와 커서의 위치가 정확하게 일치함을 확인할 수 있다. 그림에서 보이고 있는 곡선은 각 부분에서 레이저 포인터의 위치로 인식된 궤적을 표시하고 있다.

아래 그림 21에서는 프레젠테이션과 같은 일반적인 어플리케이션에 본 시스템의 적용 가능 여부를 실험 하였다. 본 실험에서는 (주)이즈커뮤니케이션즈의 i-Pro 시스템에 접목하여 원격 실험을 하였는데, 기존의 i-Pro 시스템은 그림 21(a)와 같이 터치펜을 이용하여 위치를 인식한 후, 동작하는 어플리케이션으로 설계되어 있으며, 또한 전자칠판 시스템에 연결되어, 그림 21(b)와 같이 기존 흑판을 대신 할 수 있는 디지털 판서시스템으로 제공되고 있다.

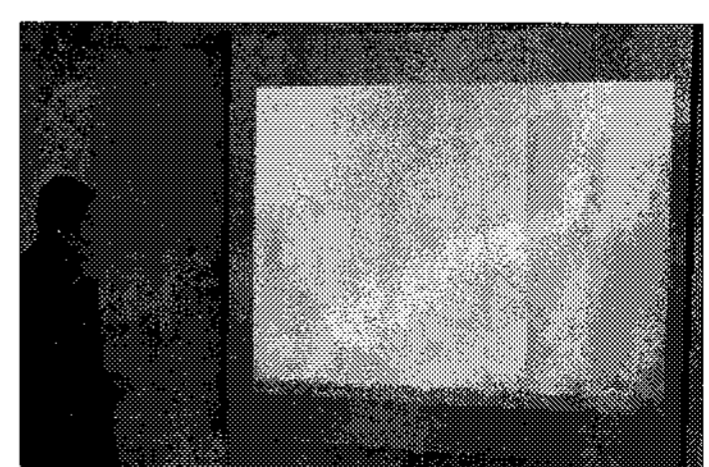
(주)이즈커뮤니케이션즈의 i-Pro 시스템은 터치펜을 이용하여 사용자가 요구하는 인터랙션의 위치를 인식한 후 동작하는 어플리케이션이 이미 구성되어 있으므로, 터치펜 대신에 본 시스템의 레이저 포인터 인터랙션 시스템으로 대체함으로써 원거리에서의 마우스 인터랙션이 가능하도록 구성하였고, 21(c)는 이를 활용하여 원거리에서 사용자가 직접 인터랙션을 동작시키는 모습을 보여주고 있다.



(a) i-Pro의 터치펜 기능



(b) 전자칠판 시스템



(c) 원거리 인터랙션

그림 21. i-Pro의 연동

7. 결론 및 향후계획

본 시스템에서는, 일반적으로 흔히 사용하는 레이저 포인터와 저가의 USB2.0 PC 카메라를 이용하여 원격으로 마우스 인터랙션이 가능한 시스템의 구성에 관하여 연구를 하였다. 초기 연구된 장비들은 서버 / 클라이언트를 활용한 통신 시스템으로 구성이 되었으나 특정 장비를 이용하여 입력하는 불편함이 있었다. 그 후 프레젠테이션과 같은 상황에서 가상 버튼을 이용하여 부분적인 제어가 가능한 시스템이 연구되었고, 커서 모델 및 레이저 포인터의 패턴 등을 이용한 인터랙션 방법도 제안되었으나, 이는 오늘날 사용자들에게 가장 친숙한 인터랙션 장비인 마우스와는 사용 방법상에 차이가 있어 사용자로 하여금 사전 학습을 요구하는 번거로움이 있다.

본 연구는 이와는 차별화 될 수 있도록 패턴에 의한 인터랙션 방식을 지양하고, 레이저 포인터만으로도 일반적으로 사용하는 마우스와 동일한 방법으로 사용할 수 있으며, 원거리에서 또한 제어가 가능할 수 있도록 시스템의 구성을 최대한 용이하게 하였을 뿐만 아니라, 기존에 구현하였던 포인팅 시스템에 비하여 동작 속도 및 사용자 중심의 환경 적응적 실험을 통하여 향상된 결과를 확인할 수 있었다.

또한 제안한 시스템에서는, 오늘날 가상현실 및 3차원 입체영상의 구성을 위하여 많이 활용되고 있는 타원형이나 휨이 있는 비선형 스크린에서도 자동으로 캘리브레이션이 가능한 기법이 적용되어 사용자가 지시하는 위치를 보다 정확하게 인식할 수 있도록 새로운 캘리브레이션 기법이 연구되었다.

위에서 연구되어진 결과들을 토대로 어플리케이션과의 연동 실험을 통하여 원거리에서 사용자가 원하는 정확한 위치에서 인터랙션이 동작할 수 있도록 시스템을 구성하였으며, 기존의 연구에 비하여 경제적이면서도, 구성이 용이할 뿐만 아니라, 컴퓨터 시스템으로부터 독립적이며, 지역적 한계를 탈피한 성공적인 실험 결과를 확인할 수 있었다.

하지만, 유비쿼터스 환경을 구성하기 위하여 오늘날 요구되는 사항은 멀티유저에 의한 동시 인터랙션도 가능해야 하며, 서로 다른 유저가 원격에서 동시 작업을 위한 가상환경 구성 등의 조건도 필요로 한다. 이번 실험에서는 이와 같은 문제에 대한 해결방안은 제시되지 않았다. 따라서 다양한 칼라를 이용하

여 동시에 동작시키는 멀티유저의 인터랙션이 가능한 방법을 모색하여야 할 것이며, 원격에서도 동시에 작업이 가능한 환경에 대한 연구가 계속적으로 수행되어야 한다.

참고 문헌

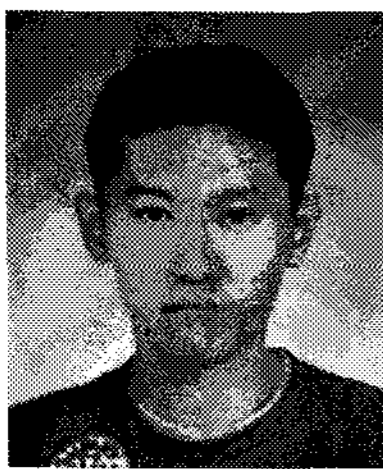
- [1] Carsten Kirstein, Heinrich Muller, "Interaction with a Projection Screen Using a Camera-Tracked Laser Pointer," *Proceedings of the International Conference on Multimedia Modeling (MMM 98)*, pp. 191-192, 1998.
- [2] Dan R. Olsen Jr., and Travis Nielsen, "Laser pointer interaction," *SIGCHI conference on Human factors in computing systems*, pp. 17-22, 2001.
- [3] Rahul Sukthankar, Robert G.Stockton, and Matthew D.Mullin. "Self-calibrating camera-assisted presentation interface," *International Conference on Control, Automation, Robotics and Vision*, (2000), (invited talk), 2000.
- [4] Rahul Sukthankar, Robert G.Stockton, and Matthew D.Mullin, "Smarter presentations: exploiting homography in camera-projector systems," *International Conference on Computer Vision*, Vol.1. pp.247-253, 2001.
- [5] Buntarou Shizuki, Takaomi Hisamatsu, Shin Takahashi, and Jiro Tanaka, "Laser pointer interaction techniques using peripheral areas of screens," *working conference on Advanced visual interfaces (AVI'06)*, pp. 95-98, 2006.
- [6] Rahul Sukthankar, Robert G.Stockton, and Matthew D.Mullin, "Automatic keystone correction for camera-assisted presentation interfaces," *International Conference on Multimedia Interfaces*, Vol.1948/2000. pp. 607-614, 2000.



김 남 우

2002년 동서대학교 컴퓨터공학과 졸업(학사)
2004년 동서대학교 소프트웨어 전문대학원 소프트웨어학과 졸업(석사)
2007년 동서대학교 소프트웨어 전문대학원 영상콘텐츠학과 수료(박사)

관심분야: 컴퓨터그래픽스, CAGD, 영상처리, 컴퓨터 비전



이 승 재

2006년 동서대학교 컴퓨터공학과 졸업(학사)
2008년 동서대학교 소프트웨어 전문대학원 영상콘텐츠학과 졸업(석사)

관심분야: 컴퓨터그래픽스, CAGD, 영상처리, 컴퓨터 비전

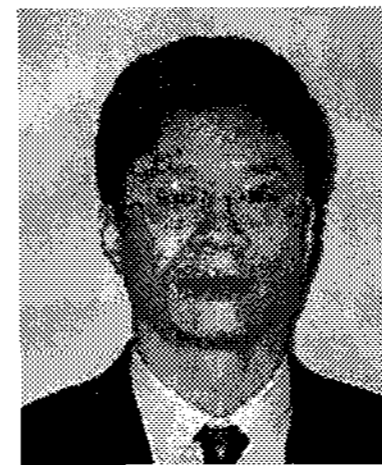


이 준 재

1986년 경북대학교 전자공학과 졸업(학사)
1990년 경북대학교 전자공학과 졸업(석사)
1994년 경북대학교 전자공학과 졸업(박사)
2000년~2001년 (주)파미 연구소장

1998년~1999년 Georgia Institute of Technology 객원 교수

1994년~2007년 동서대학교 컴퓨터정보공학부 부교수
2007년~현재 계명대학교 게임모바일콘텐츠학과 부교수
관심분야: 영상처리, 3차원 컴퓨터 비전, 지문분석



이 병 국

1987년 연세대학교 수학과 졸업(학사)
1989년 KAIST 응용수학과 졸업(석사)
1993년 KAIST 수학과 졸업(박사)
1993년~1995년 데이콤 종합연구소 선임연구원

1995년~현재 동서대학교 컴퓨터정보공학부 부교수
관심분야: 컴퓨터그래픽스, CAGD