論文
6-5-3

인터랙티브 레이져 포인팅 마우스 시스템
(Interactive laser pointing mouse system)

박민순(Min-Sun Park)[1)]

요약

본 논문에서는 레이져 포인터 마우스를 이용한 윈도우 기반 인터랙티브 프리젠테이션 시스템을 구성하였다. 시스템은 발표자가 레이져 포인터를 사용하여 인터랙티브하게 프리젠테이션을 할 수 있도록 제공한다. 발표동안 디스플레이용 PC는 비트맵으로 포현되는 한 국부의 포인트 위치를 검출하고, 이 점은 프로젝터로 전달되어 디스플레이 된다.디스플레이된 프리젠테이션 자료는 또한, 스크린에 있는 레이져 스폿을 모니터링하며 제어된다. 프로세싱 단에서는 미리 알고 있는 공간상의 패턴과 레이져 스폿을 일치 시키게 되며, 이에 따라 디스플레이 명령이 세팅된다. 이때, 디스플레이에 대한 명령은 컴퓨터로 전달되며 마우스 기능과 같은 동작을 얻을 수 있게 된다.

Abstract

In this paper, we made a windows-based interactive presentation system using a laser pointer mouse. The system provides that a speaker controls the presentation interactively by means of a laser pointer. During the presentation, a display PC generates on its local display a bitmap corresponding to the presentation. This bitmap is then transmitted to the projector and the bitmap is then projected onto the screen. The display of the presentation is controlled by the monitoring of the laser spots that are also projected onto the screen. Laser spot control is achieved through a control system. When the processing section matches the successive laser spot positions with a pre-established spatial pattern, the corresponding display command is issued. The display command may be transmitted to the display computer which responds by an action like mouse function.

논문접수 : 2005. 11. 20.
심사완료 : 2005. 12. 22.

1) 정회원 : 광운대학교 교양학부 교수

본 논문은 2004년도 광운대학교 교내연구비에 의해 연구되었음.

1. 서 론

현재까지의 프리젠테이션 방식은 OHP 자료를 프로젝터를 사용하여 발표하는 방식이 주류를 이루고 있다. 최근에 LCD 프로젝터의 경우 LCD의 가격이 많이 하락하면서 액정프로젝터의 가격이 대폭 낮아져서 노트북 컴퓨터와 액정프로젝터를 사용하여 프리젠테이션 하는 방식이 거의 일반화 되고있는 추세이다. 이와 같은 기존 프리젠테이션의 가장 커다란 단점은 레이져 포인터를 사용하는 경우 본인이 발표위치와 컴퓨터 사이를 오가면서 발표를 하게되어 발표에 집중을 하지 못하는 단점을 지니게 된다. 다른 보조 요원에 의해 컴퓨터 마우스를 조작하는 경우 발표자와 보조요원 사이에 의사소통이 요구되는데, 이는 마찬가지로 발표에 전념해야하는 발표자의 원활한 진행을 방해하는 단점을 지니게 된다. 레이져 포인터를 사용하지 않더라도 문제점은 여전히 남아 있으며, 레이져포인터를 사용할 수 없어 생기는 단점 이외에도 발표자가 항상 컴퓨터 근처에 있어야 하므로 행동 반경의 제약을 받는다. 무선마우스를 사용하는 경우에도 불편함은 여전하다고 할 수 있다.

본 논문에서는 디지털카메라를 이용한 이미지 프로세싱을 통하여 레이져포인터가 단순한 포인팅 기능뿐만 아니라 마우스 기능을 수행할 수 있도록 함으로써, 상기의 문제점을 해결함과 동시에 링크 파일을 가진 하이퍼텍스트, 인터넷 웹페이지, 멀티미디어 파일등을 프리젠테이션할 수 있는 프리젠테이션 시스템을 제작하였다.

전체 시스템은 프로젝션된 영상을 캡쳐하는 카메라부, 카메라 영상으로부터 포인터 위치를 검출하여 위치정보를 빔프로젝터와 연결된 컴퓨터에 전송하는 하드웨어 모듈부, 포인팅 기능을 하는 버튼과 마우스 왼쪽 및 오른쪽 키 역할을 하는 두 개의 버튼, 합해서 세 개의 버튼을 가진 레이저포인터부, 커서가 절대 좌표에 따라 동작하도록 하고 흔들림 방지를 보완하는 컴퓨터 드라이버부로 구성되어 있다. 발표 자료 정보가 빔 프로젝터에 의해서 스크린에 디스플레이 되었을 때 발표자가 레이저포인터로 관심 부분을 포인팅 하면 카메라는 스크린 영상을 캡쳐하고 이를 하드웨어 모듈로 전송한다. 하드웨어 모듈의 신호 처리부는 포인터의 위치를 검출하고 이를 시리얼 포트를 통해서 컴퓨터에 전송한다. 컴퓨터의 드라이버 프로그램은 커서가 입력받은 절대 좌표에 따라 이동하게 한다. 이 과정은 초당 30번 정도 반복되도록 하여 커서가 항상 레이저포인터와 같은 위치에 있도록 한다. 만약 링크되는 파일을 가진 하이퍼텍스트를 발표할 때 링크되는 페이지로 이동하고 싶으면 레이저포인터를 링크위치로 이동시켜 커서가 같은 위치에 놓이게 한 후 레이저포인터의 마우스키 역할을 하는 버튼을 누른다. 이때 레이저포인터에서는 RF 송신 신호를 하드웨어 모듈로 전송하고 하드웨어 모듈은 이를 수신해서 버튼 입력을 컴퓨터로 전송한다. 이때 드라이버 프로그램은 윈도우 OS에 마우스 키 버튼 입력을 전달하여 연속적으로 동작이 이루어질 수 있게된다. 이와같은 방식으로 마우스와 같은 역할을 하는 인터랙티브 레이져 포인터를 구성하였다.

2. 프리젠테이션 시스템을 위한 영상처리

2-1. 시스템 개요

본 논문에서 제안하는 시스템은 액정프로젝터를 통해 출력되는 화면과 레이저 포인터를 액정프로젝터 위에 설치된 카메라와 PCI 슬롯에 설치된 영상입력카드를 사용하여 입력받고, 입력받은 영상에서 레이저 포인터의 위치를 추적하여 PC의 마우스 커서를 레이저 포인터의 위치로 이동시킴으로서 레이저 포인터를 이용하여 마우스의 기능을 구현하는 것으로서 제안된 시스템의 전체적인 개요를 그림1 에 보여준다. 전체 시스템은 프로젝션된 영상을 캡쳐하는 카메라부, 카메라 영상으로부터 포인터 위치를

검출하여 위치정보를 빔프로젝터와 연결된 컴퓨터에 전송하는 하드웨어 모듈부, 포인팅 기능을 하는 버튼과 마우스 왼쪽 및 오른쪽 키 역할을 하는 두 개의 버튼, 합해서 세 개의 버튼을 가진 레이저포인터부, 커서가 절대 좌표에 따라 동작하도록 하고 흔들림 방지를 보완하는 컴퓨터 드라이버부로 구성되어 있다. 시스템의 동작을 살펴보면 발표 자료 정보가 빔프로젝터에 의해서 스크린에 디스플레이 되었을 때 발표자가 레이저포인터로 관심 부분을 포인팅 하면 카메라는 스크린 영상을 캡쳐하고 이를 하드웨어 모듈로 전송한다. 하드웨어 모듈의 신호 처리부는 포인터의 위치를 검출하고 이를 시리얼 포트를 통해서 컴퓨터에 전송한다. 컴퓨터의 드라이버 프로그램은 커서가 입력받은 절대 좌표에 따라 이동하게 한다.

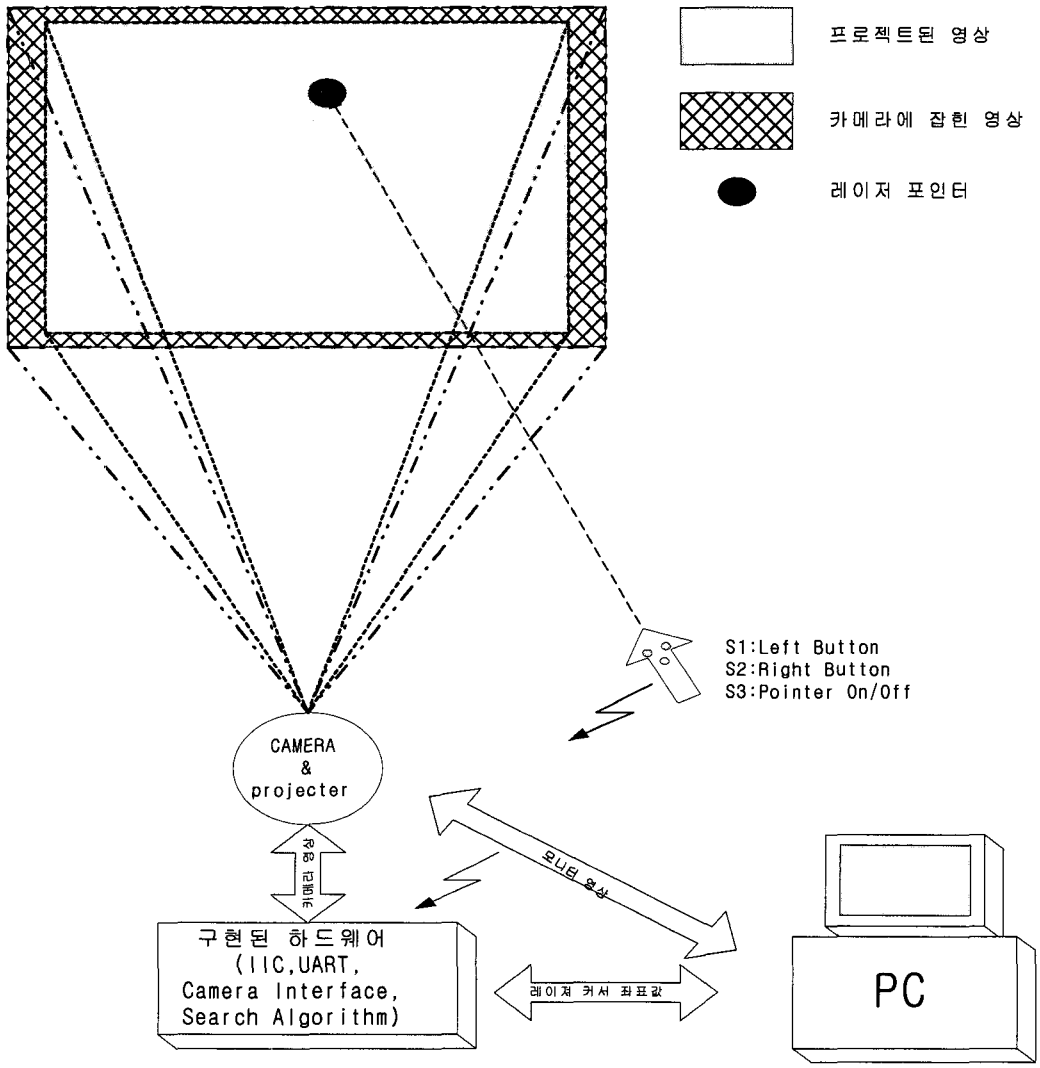
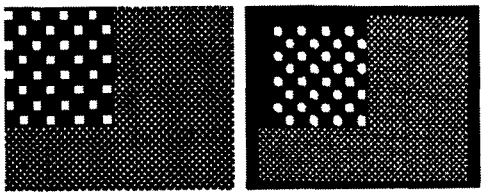


그림 22 시스템 블록도

이 과정은 초당 30번 정도 반복되도록 하여 커서가 항상 레이저포인터와 같은 위치에 있도록 한다. 만약 링크되는 파일을 가진 하이퍼텍스트를 발표할 때 링크되는 페이지로 이동하고 싶으면 레이저포인터를 링크위치로 이동시켜 커서가 같은 위치에 놓이게 한 후 레이저포인터의 마우스키 역할을 하는 버튼을 누른다. 이때 레이저포인터에서는 RF 송신신호를 하드웨어 모듈로 전송하고 하드웨어 모듈은 이를 수신해서 버튼 입력을 컴퓨터로 전송한다. 이때 드라이버 프로그램은 윈도우 OS에 마우스 키 버튼 입력을 전달한다. 실험을 위해 액정프로젝터에 의해 프로젝션 되는 영상을 출력영상이라 하고, 카메라를 통해 입력되는 영상을 입력영상이라 하자. 출력영상의 해상도는 액정프로젝터로 프로젝션 되는 PC의 화면 해상도와 동일하며 입력영상은 사용하게 되는 카메라의 해상도와 동일하다. 그림 2의 (a)와 같이 특수한 무늬를 가지는 영상을 전체화면으로 프로젝션 시키고, 이를 NTSC 출력을 가지는 CCD 카메라와 영상 캡쳐보드를 사용하여 640x480의 해상도를 가지는 영상을 초당 30프레임(60필드)의 속도로 입력받아 그림 2(b)와 같이 입력영상으로 사용하였다.



(a) (b)

그림 2. (a) 출력영상 (b) 입력영상

Dword	Byte3	Byte2	Byte1	Byte0
DW0	Y1	Cr0	Y0	Cb0
DW1	Y3	Cr4	Y2	Cb4
DW2	Y7	Y6	Y5	Y4

그림 3. BTYUV(YUV 4:1:1) 데이터 포맷

영상입력보드를 이용하여 입력받은 영상은 그림 3과 같이 BTYUV 포맷의 칼라영상으로 입력되며 PCI버스 마스터링을 통해 시스템 메모리 상에 배열의 형태로 저장된다. 그림 2에서 보는 바와 같이 출력영상과 입력영상은 많은 차이를 가지게 된다. 입력영상은 출력영상을 카메라로 입력받아 출력영상과 주변의 이미지를 포함하고 있기 때문에 입력영상에서 레이저포인터의 위치는 실제 윈도우의 마우스 커서 위치와 차이가 난다. 그러므로 제일 먼저 처리해야 할 부분은 입력영상에서 출력영상의 위치를 찾아내는 작업이다. 본 논문에서는 먼저 640x480의 해상도에 32x48개의 격자를 가지는 특수한 그림 2(a) 영상을 프로젝션 시키고, 카메라를 통해 입력된 영상에서 격자들을 찾아낸 후, 각 격자의 중심점에 미리 정의된 좌표값을 맵핑(mapping)하는 방법으로 입력영상에서 윈도우의 좌표를 찾았으며, 이 윈도우의 좌표를 이용하여 입력영상에서 레이저 포인터의 좌표를 윈도우의 마우스 커서 좌표로 변환시키는 변환 테이블을 마들어 놓고, 이후의 과정에서는 입력영상에서 레이저 포인터의 위치를 찾아 변환 테이블을 사용하여 레이저 포인터가 지시하는 윈도우의 마우스 커서 좌표를 얻을 수 있었다.

2-2. 출력 영상의 특성

변환테이블 생성을 위해서는 입력영상에서 액정프로젝터의 의해 출력된 PC의 윈도우의 위치를 정확히 추출해내야 한다. 윈도우의 위치는 카메라의 위치나 상황에 따라서 항상 변하기 때문에 입력영상에서 윈도우의 위치를 정확히 찾아내기 힘들다. 본 논문에서는 이러한 문제점의 해결을 위해 그림 4와 같은 5x5 화소 크기의 사각형 32x48개가 지그재그로 배열된 특정한 패턴을 지니는 출력영상을 액정프로젝터로 출력하고, 이 영상을 카메라로 입력받은 입력영상에서 각각의 사각형의 위치를 추출하여 입력영상과 윈도우의 좌표를 일치시키는 변

환테이블을 생성하였다.

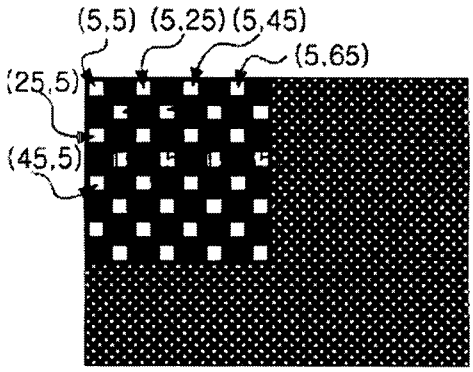


그림 4. 출력 패턴

그림 4에서 보는 바와 같이 전체 영상의 크기가 640x480일 경우 각각의 사각형은 5x5 화소 크기의 정사각형 모양이며, 각 사각형의 중심점은 미리 정의된 고유의 윈도우 좌표를 가지고 있다. 예를 들어 좌측 최 상단에 위치한 사각형의 경우 윈도우에서의 좌표는 X=5, Y=5의 고유 좌표값을 가지게 된다. 이 사각형이 입력영상의 어느 부위에 위치하던 간에 레이저 포인터가 이 사각형의 위치를 지시하게 되면 마우스의 커서를 (5,5)의 위치로 이동시켜 레이저 포인터와 윈도우의 좌표를 일치시킬 수 있다. 또한 액정프로젝터를 이용하여 프로젝션 할 경우 스크린과 액정프로젝터가 이루는 각도에 의해서 어느 정도의 화면 왜곡이 생기게 된다. 예를 들어 스크린보다 프로젝터의 위치가 낮을 경우 프로젝션 된 화면의 상단보다 하단이 더 좁게 나타나게 된다. 그러나 이러한 그림 4와 같은 패턴을 이용하여 부분구역으로 나누어 처리하게 되면 이러한 문제점을 해결할 수 있다.

2-3. 임계값 처리를 이용한 영역추출

칼라영상을 가지고 처리를 할 경우에 많은 연산시간을 필요로 하므로 이 영상을 바로 사용하지 않고 Y 값만을 사용하여 8bit의 그레이 영상을 얻는다. 본 논문에서는 배경을 제거하고 흰색의 사각형만을 추출하여야 하는데 가장 단순한 방법이 임계값(Threshold) 처리라고 부르는 것이다. 그림 5에서와 같이 문자를 포함한 영상에서 문자부분만 추출해보자. 임계값 처리는 입력영상의 각 화소에 대하여 밝기가 어떤 일정한 값(임계값) 이상인 경우에는 대응하는 출력화상의 화소값을 1로 하고 그 이외의 경우에는 0으로 하는 것이다.

임계값을 설정하는 방법에는 모드법, P 타일법, 판별분석법, 가변 임계값법 등이 있다. 모드법은 히스토그램의 골짜기를 사용하여 임계값을 정하는 방법이며, P 타일법은 전체 화상에서 차지하는 물체의 비율(예를 들어 P%)을 알고 있을 때 히스토그램 상에서 P% 부분을 임계값으로 하는 것이다. 판별분석법은 물체와 배경의 두 집단으로 나누었을 때 두 집단의 통계량이 다르도록 임계값을 결정하는 방법이며, 가변임계값은 배경의 밝기가 다른 경우에 부분별로 임계값을 변화시키는 방법이다.

본 논문에서는 히스토그램의 골짜기를 찾아 임계값을 정하는 모드를 사용하여 임계값을 결정하였으며, 그 예를 그림 6에 나타내었다.

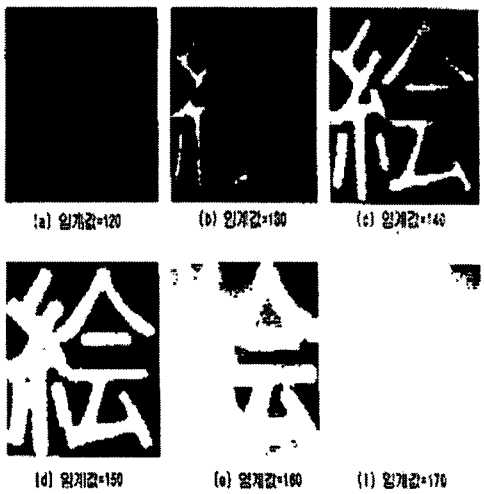


그림 5. 임계값 처리의 예

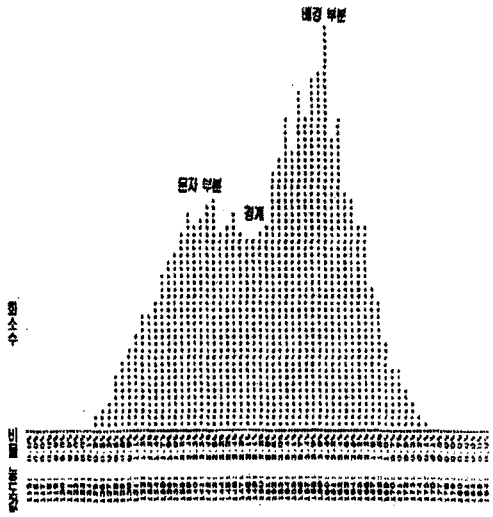


그림 6. 히스토 그램의 예

2-4. 2진 영상의 잡은 제거

임계값 처리를 이용한 영역추출에서 결과영상은 2진 영상으로 그림 7과 같이 Salt-and-Pepper Noise라 불리는 잡음을 가지고 있다. 이 잡음은 2진 영상이라는 점을 이용하여 팽창, 수축이라는 처리로 제거할 수 있다. 팽창이란 어떤 화소부근에 1이 하나라도 있으면 그 화소를 1로 그 외는 0으로 처리한다. 수축이란 어떤 화소의 부근에 하나라도 0이 있으면 그 화소를 0으로 그 외는 1로 하는 처리이다. 그림 7에서 보는 바와 같이 이 처리를 팽창 후 수축으로 작용시키면 결과의 화상은 팽창으로 굵어지고 수축으로 가늘어져 결과적으로 변화가 없지만, 검은 고립된 잡음이 팽창될 때 제거된다. 반대로 수축 후 팽창으로 작용시키면 흰 고립된 잡음이 수축일 때에 제거된다. 본 논문에서는 팽창 후 수축처리 후에 다시 수축 후 팽창 처리를 수행하여 깨소금 잡음을 제거하였다.

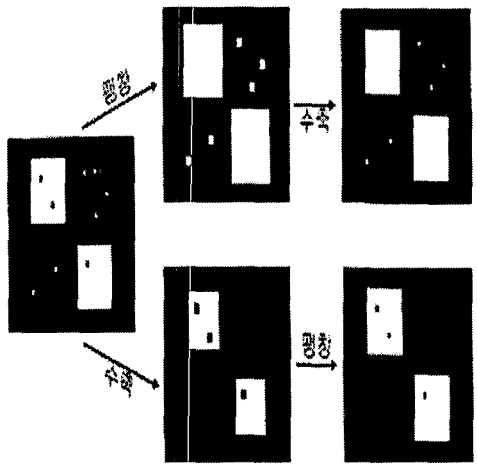


그림 7. 팽창, 수축 처리

2-5. 라벨링

연결되어 있는 모든 화소(연결 성분)에 동일 번호(라벨)를 붙이고 다른 연결 성분에는 다른 번호를 붙이는 처리를 라벨링(Labeling)이라 하며, 이 처리로 개개의 연결성분으로 분리할 수 있으며 각 연결 성분의 특징을 조사할 수 있다. 라벨링을 하는 방법은 영상을 순차적으로 주사하여 라벨이 붙지 않은 화소 P를 찾아내어 새로운 라벨을 붙이고 화소 P에 연결되어 있는 화소에 같은 라벨을 붙인다. 또한 지금 라벨을 붙인 화소와 연결되어 있는 모든 화소에 동일 화소를 붙이며, 이러한 과정을 라벨을 붙여야 할 화소가 없어질 때까지 반복하고 처음으로 돌아가서 새로운 라벨을 붙여야할 화소를 찾아 이상의 과정을 반복하여 영상 전체의 주사가 끝났을 때 모든 처리가 종료된다. 위에서 설명한 라벨링 과정을 그림 8에 순서대로 나열하였다.

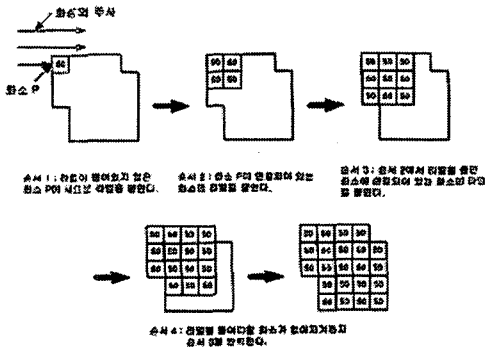


그림 8. 라벨링 방법

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	2	2	2	0
0	1	1	1	0	0	0	0	0	2	2	2	0
0	1	1	1	0	0	0	0	0	2	2	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	3	3	0	0	0	0	0
0	0	0	0	0	3	3	3	0	0	0	0	0
0	0	0	0	0	3	3	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	4	4	0	0	0	0	0	5	5	5	0
0	4	4	4	0	0	0	0	0	5	5	5	0
0	4	4	4	0	0	0	0	0	5	5	5	0
0	0	0	0	0	0	0	0	0	0	0	0	0

그림 9. 라벨링의 예

그림 9는 13x13 화소의 크기를 가지는 입력영상에 대하여 라벨링 과정을 수행한 경우의 라벨링 결과를 보여주는 그림이다. 라벨링을 수행함으로서 그림 9와 같이 각각의 사각형들을 분리해 낼 수 있다. 상기의 라벨링 과정을 본 논문에서 정의한 입력화면을 대상으로 수행하였을 경우, 1536(32x48)번까지 라벨링 작업을 수행하게 되며, 만약 라벨링 과정 후 최후의 라벨이 1536이 아닐 경우는 임계값을 잘못 설정하여 생긴 잡음에 의한 오차이거나 혹은 카메라에 프로젝션된 전체 윈도우가 포함되지 않은 경우이다. 2진 영상에서 흑과 백의 비율이 맞지 않는다면 임계값을 조정하고, 전체 화면이 카메라에 잡히지 않았다면 전체 화면이 잡힐 수 있도록 카메라 조작 후에 이전의 과정을 재 수행하게 되면 1536개의 사각형을 모두 분리해 낼 수 있다.

2-6. 중심점 추출 및 정렬

라벨링 후에는 분리된 각각의 사각형들의 중심점을 찾아내야 한다. 중심점의 X좌표는 같은 라벨을 가진 모든 화소의 평균이 되며, Y좌표는 같은 라벨을 가지 모든 화소의 평균이 된다. 여러 화소들의 평균을 취하게 되므로 중심점은 실수 값을 가지게 되나 중심점을 실수 값으로 처리하게 될 경우 많은 양의 부하가 예상되므로 실수 값으로 처리를 하지 않고 반올림 처리하여 정수 값을 사용하였다. 이러한 처리 과정에서 640x480의 해상도에서 약∓0.5 화소의 오차가 발생하게 되나, 이 정도는 프리젠테이션에서 활용시 무시할 만한 오차이다.

i 라는 라벨을 가진 사각형의 중심점은 다음과 같은 식으로 나타낼 수 있다. 이때 L은 라벨링의 결과이며 n은 i 라는 라벨을 가진 총 화소수를 나타낸다.

$$C_{i,x} = \sum_{j=1}^{n} L_{i,j,x} \quad (1)$$

$$C_{i,y} = \sum_{j=1}^{n} L_{i,j,y} \quad (2)$$

중심점 추출의 결과는 1536 길이의 배열에 저장되며, 여기 저장된 결과는 영상을 주사하며 가장 먼저 나타나는 사각형 순서로 라벨링한 결과대로 배열하였기 때문에 어느 정도의 규칙성을 가지고 있긴 하지만, 정확한 규칙성은 찾아보기 힘들다. 가장 먼저 라벨이 붙여지는 사각형이 좌측 상단에 있는 사각형이라는 보장이 없기 때문이다. 따라서 좌측부터 우측, 상단부터 하단의 순서로 중심점을 정렬할 필요가 있다.

중심점 정렬을 위해서 좌측 상단의 중심점을 기준으로 잡고, 그 기준으로부터 일정 위치에 있는 다른 중심점을 찾아가는 방식으로 중심점을 정렬하였다. 그림 10에서와 같이 출력영상에서 1번과 2번 중심점의 Y좌표는 동일하며, X좌표는 20 화소의 차이가 있다. 또한 1번과 65번 중심점의 Y좌표는 동일하며, X좌표는 20의 차이가 있다. 1번과 33번 중심점은 X좌표, Y좌표 모두 10화소의 차이가 있다. 따라서 이러한 영상을 전체 화면으로 입력받은 경우 비슷한 상관관계가 입력영상에도 나타날 수밖에 없다. 이러한 상관 관계를 이용하여 중심점 정렬을 수행하였다. 식 1과 2를 이용해 찾아진 중심점 중에서 기준이 되는 중심점(그림 10의 1번 중심점)을 먼저 찾고, 그 기준 중심점을 기준으로 수평 방향의 기준점(1, 2, 3, ... 32)들을 찾는다.

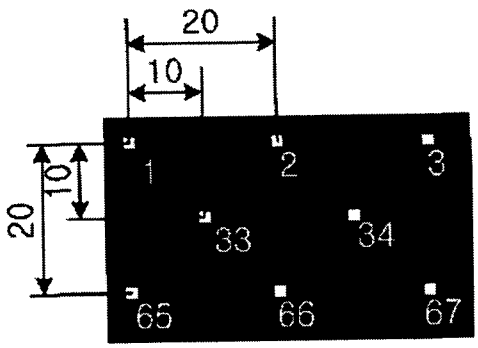


그림 10. 중심점의특징

1536개의 중심점 중에서 X좌표와 Y좌표의 합이 가장 작은 중심점이 기준 중심점(1번 중심점)이 된다. 기준 중심점이 찾아졌으면 기준 중심점의 X좌표에 10화소만큼 더한 위치의 근처에 있는 중심점(2번 중심점)을 찾는다. 2번 중심점을 기준점으로 3번 중심점을 찾고. 3번 중심점을 기준으로 4번을 찾는 식으로 상기 과정을 반복하면 32번 중심점까지 정렬시킬 수 있다. 똑같은 방법을 이용해서 수직방향의 중심점(33, 65, ...)을 찾고 이렇게 찾아진 수직 방향의 중심점을 기준점으로 하여 상기와 같이 수평방향의 중심점들을 찾아나가면 1번부터 1536번까지의 중심점들을 좌측 상단에서 우측 하단의 순서로 정렬시킬 수 있다.

2-7. 변환 테이블 생성

중심점 정렬이 끝나면 중심점과 중심점을 잇는 사각형을 만들어 각각의 부분구역으로 나누어 변화테이블을 작성하게 된다. 그림 11에서와 같이 1번과 33번 중심점을 대각 꼭지점으로 갖는 사각형, 33번과 2번 중심점을 대각 꼭지점으로 갖는 사각형을 만들어 각각의 부분구역으로 나누고 그 사이의 화소에는 윈도우 좌표를 맵핑함으로써 레이저 포인터가 가리키는 곳의 윈도우 좌표를 얻어낼 수 있다. 그림 11은 입력화면의 일부를 나타낸 그림이며, 흰색 배경부분은 입력영상에서 윈도우가 차지하고 있는 영역이다. 그러나 지금까지 구한 중심점만을 연결하는 사각형을 만들어 부분구역을 나누었을 경우, 전체 윈도우의 영역을 포함하지 못한다. 그래서 지금까지 구한 중심점을 기준으로 가상 중심점(그림 11의 A, B)을 만들어서 가상 중심점과 실제 중심점을 대각 꼭지점으로 갖는 부분구역(점선으로 된 사각형)을 만들어 주어야 한다. 그림 11의 점선 부분이 가상 중심점을 추가하여 윈도우 영역을 모두 포함할 수 있게 부분구역을 나눈 그림이다. 가상 중심점의 좌표 설정은 가상 중심점이 위치할 지점의 수평 수직 방향에 위치한 중심점들을 고려하여 설정하였다. 예를 들어 그림 11의 A지점의 가상 중심점의 X좌표는 33번 중심점의 X좌표와 동일하며, Y좌표는 33번 중심점과 97번 중심점의 Y좌표의 차이를 33번 중심점에서 뺀 값이다. 또한 B지점의 가상 중심점의 X좌표는 34번 중심점의 X좌표와 33번 중심점의 X좌표의 차이를 33번 중심점의 X좌표에서 뺀 값이며, Y좌표는 33번 중심점의 Y좌표와 동일하다.

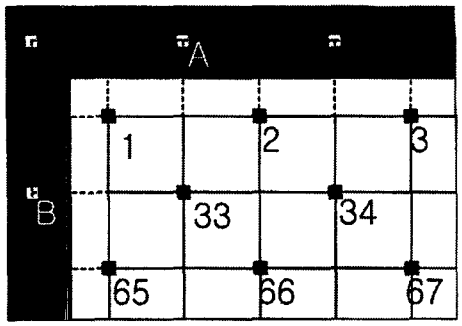


그림 11. 가상 중심점의 생성

가상 중심점 생성 후에는 가상 중심점과 실제 중심점을 대각 꼭지점으로 하는 사각형을 만들고 양 꼭지점의 고유의 좌표 값을 가지고 사이의 화소에 알맞은 윈도우 좌표 값을 인터폴레이션 한다.

그림 12는 (5, 5)의 윈도우 좌표 값을 가진 1번 중심점과 (15, 15)의 윈도우 좌표 값을 가진 33번 중심점 사이의 화소들에 대하여 알맞은 값들로 인터폴레이션 한 예이다.

5,5	6,5	7,5	9,5	10,5	11,5	13,5	14,5	15,5
5,6	6,6	7,6	9,6	10,6	11,6	13,6	14,6	15,6
5,7	6,7	7,7	9,7	10,7	11,7	13,7	14,7	15,7
5,9	6,9	7,9	9,9	10,9	11,9	13,9	14,9	15,9
5,10	6,10	7,10	9,10	10,10	11,10	13,10	14,10	15,10
5,11	6,11	7,11	9,11	10,11	11,11	13,11	14,11	15,11
5,13	6,13	7,13	9,13	10,13	11,13	13,13	14,13	15,13
5,14	6,14	7,14	9,14	10,14	11,14	13,14	14,14	15,14
5,15	6,15	7,15	9,15	10,15	11,15	13,15	14,15	15,15

그림 12. 부분 변환 테이블의 예

변환테이블을 생성할 때 입력영상을 그림 13과 같이 크게 9개 영역으로 나눌 수 있다. E 영역은 윈도우 영역으로 액정 프로젝터에 의해서 프로젝션 된 윈도우가 위치한 영역이다. 레이저 포인터가 이 영역을 가리키게 되면 윈도우의 변환테이블의 출력 값은 그림 13과 같이 X좌표는 0에서 640, Y좌표는 0에서 480까지 변화한다

A X=0 Y=0	B X=0~640 Y=480	C X=640 Y=0
D X=0 Y=0~480	E X=0~640 Y=0~480	F X=640 Y=0~480
G X=0 Y=480	H X=0~640 Y=480	I X=640 Y=480

그림 13. 변환 테이블의 구성

E 영역을 제외한 다른 영역은 프로젝션 된 윈도우의 영역을 벗어난 영역으로 레이저 포인터가 지시하는 영역이 A 영역이면 (0, 0), C 영역이면 (640, 0), G 영역이면 (0, 480), I 영역이면 (640, 480)으로 변환되어 윈도우의 각 꼭지점을 지시하도록 하였으며 B, D, F, H 영역은 윈도우의 모서리를 지시하도록 하였으며 지면 관계상 변환테이블 전체를 도시화하지는 못하였으며, 윈도우 영역(E 영역)의 좌측 상단의 변환테이블을 입력영상에서의 좌표와 윈도우에서의 좌표를 비교해서 그림 14에 나타내었다.

마우스	0.0	0.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
영상	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
마우스	0.0	0.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
영상	0.1	1.1	2.1	3.1	4.1	5.1	6.1	7.1	8.1	9.1
마우스	0.0	0.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
영상	0.2	1.2	2.2	3.2	4.2	5.2	6.2	7.2	8.2	9.2
마우스	0.0	0.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0	8.0
영상	0.3	1.3	2.3	3.3	4.3	5.3	6.3	7.3	8.3	9.3
마우스	0.1	0.1	0.1	1.1	2.1	3.1	4.1	5.1	6.1	8.1
영상	0.4	1.4	2.4	3.4	4.4	5.4	6.4	7.4	8.4	9.4
마우스	0.2	0.2	0.2	1.2	2.2	3.2	4.2	5.2	6.2	8.2
영상	0.5	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
마우스	0.3	0.3	0.3	1.3	2.3	3.3	4.3	5.3	7.3	8.3
영상	0.6	1.6	2.6	3.6	4.6	5.6	6.6	7.6	8.6	9.6
마우스	0.4	0.4	0.4	1.4	2.4	3.4	4.4	5.5	7.5	8.5
영상	0.7	1.7	2.7	3.7	4.7	5.7	6.7	7.7	8.7	9.7
마우스	0.5	0.5	0.5	1.5	2.5	3.5	4.5	5.6	7.5	8.6
영상	0.8	1.8	2.8	3.8	4.8	5.8	6.8	7.8	8.8	9.8
마우스	0.6	0.6	0.6	1.6	2.7	3.7	4.7	5.7	7.7	8.7
영상	0.9	1.9	2.9	3.9	4.9	5.9	6.9	7.9	8.9	9.9
마우스	0.8	0.8	0.8	1.8	2.8	3.8	4.8	6.8	7.8	8.8
영상	0.10	1.10	2.10	3.10	4.10	5.10	6.10	7.10	8.10	9.10

그림 14. 변환 테이블의 예

지금까지 앞에서 설명한 변환테이블 과정을 그림 15에 순서대로 나타내었으며, 그림 16에는 구현된 변환테이블 생성 프로그램의 실행 화면을 나타내었다. 변환테이블 생성 프로그램이 실행되면 배경은 입력화면에서 윈도우 영역을 추출하기 위한 사각형의 배경무늬로 채워지면, 그 위에 현재의 입력화면과 히스토그램을 보여주는 창이 생성된다.

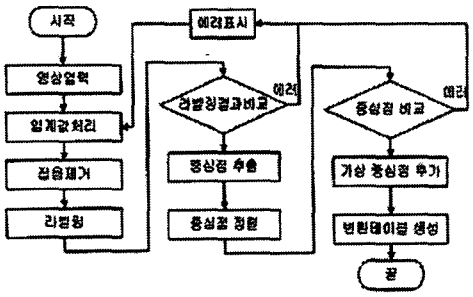


그림 15. 변환테이블 생성 순서도

이 화면을 기준으로 카메라 각도와 영상을 조절한 후 변환테이블 생성버튼을 누르면 영상 조절 창은 없어지고, 전체화면에 격자 무늬만이 출력되어 그림 15의 순서도에 나타난 과정을 수행하여 변환테이블을 생성하게 된다.

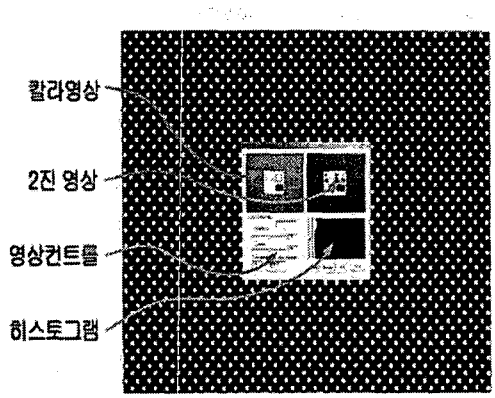


그림 16. 변환테이블 생성 프로그램

3. 마우스 기능의 구현

3-1. 레이저 포인터의 위치 추적

레이저 포인터를 이용하여 마우스의 역할을 수행하기 위해서 가장 기본적인 것은 당연히 레이저 포인터의 위치를 찾아내는 것이다. 그림 17은 액정프로젝터를 이용하여 PC의 화면을 스크린에 주사하고 레이저 포인터를 이용하여 특정 지점을 지시한 화면을 카메라를 이용하여 캡처한 모습이다. 그림 17의 입력영상에서 흰색 사각형 안의 흰 점이 레이저 포인터가 지시하는 부분이다. 이 부분을 확대한 그림을 보면 레이저 포인터가 배경화면에 비해 밝게 나타남을 알 수 있다.

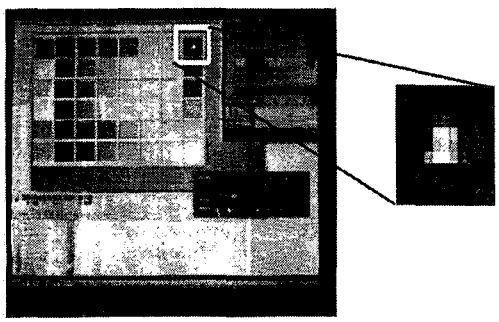


그림 17. 레이저 포인터 입력 영상

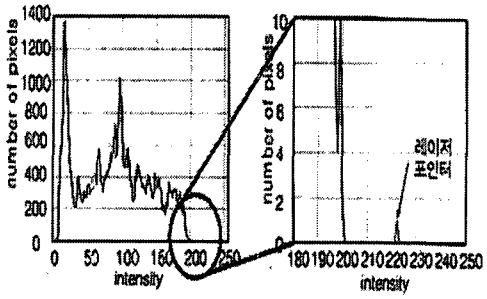


그림 18. 입력 영상의 히스토그램

그림 18에 나타난 입력화면의 히스토그램을 살펴보면 더 자세히 알 수 있다. 그림 18의 히스토그램에서 농도 200까지는 배경화면 부분이며, 화면에 비해서 밝게 나타나는 특성을 이용하여 입력된 화면에서 레이저 포인터의 위치를 찾아내었으며, 레이저 포인터가 없는데도 화면상의 가장 밝은 점을 레이저 포인터로 인식하는 것을 방지하기 위하여 레이저 포인터의 ON/OFF 스위치의 상태를 RF로 마우스 버튼의 상태와 함께 PC로 전송하여 레이저 포인터가 ON 상태일 때만 마우스 커서를 이동시키도록 프로그램 하였다.

3-2. 마우스 커서의 위치 보상

입력영상의 특성을 이용해서 찾아진 레이저 포인터의 위치는 시간에 따라서 계속 변하고 있다. 일반 마우스가 책상 위에 한쪽 면이 고정되어 있는 것과는 달리 레이저 포인터는 손에 들고 사용하기 때문에 특정 위치에 고정되어 있을 수 없으므로 손의 흔들림에 따라서 고정되어 있지 못하고 계속 흔들리게 된다. 그러므로 이러한 흔들림을 보정할 수 있어야 마우스의 역할을 해낼 수 있다. 제안된 시스템에서는 이러한 흔들림의 방지를 위하여 흔들림에 의한 위치오차를 보정할 수 있는 템포랄 필터(Temporal Filter)를 사용하였다. 흔들림 보정에 사용한 템포랄 필터의 전달함수는 다음의 식과 같다.

$$\frac{Y(z)}{X(z)} = \frac{Cz^{-1}}{1-(1-C)z^{-1}} \quad (3)$$

여기서 X(z)는 입력영상의 특성을 이용해서 찾아낸 레이저 포인터의 좌표, Y(z)는 임시적인 필터의 출력 값, 그리고 C는 템포랄 필터계수(Filter Coefficient)이다. 필터링은 한 필드만 레이저 포인터의 X좌표와 Y좌표에 대하여 각각 1회씩 진행된다.

그림 19는 필터 계수 C값에 따른 필터의 스텝

응답(Step Response)을 나타내는 그림으로서 필터 계수가 1에 가까워질수록 최종 입력 값이 출력에 반영되는 비율이 많아지며 필터 계수가 작아질수록 이전의 입력 값들이 반영되는 비율이 많아지게 된다.

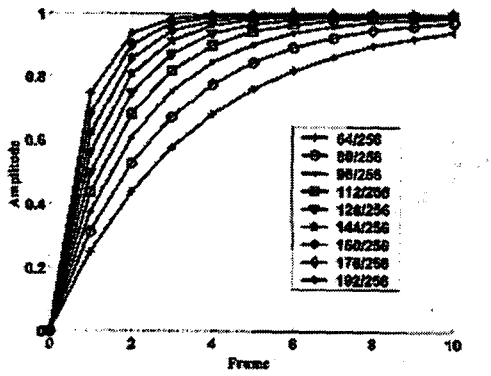


그림 19. Temporal 필터의 스텝 응답

필터 계수를 작게 하면 작게 할수록 레이저 포인터의 흔들림은 줄어들지만, 그와 비례해서 레이저 포인터를 다른 곳으로 이동시킬 때 나타나는 지연시간이 늘어나게 되고 필터 계수가 1에 가까울수록 레이저 포인터 이동시의 지연시간은 줄어들지만 레이저 포인터의 흔들림도 늘어나게 된다. 따라서, 흔들림과 지연시간 사이에서 적절한 타협점을 찾아 필터 계수를 정해야 한다.

3-3. 버튼의 입력

레이저 포인터로 마우스 기능을 구현하기 위해서는 커서의 위치와 함께 버튼도 입력받아야 한다. PC에서 버튼의 상태를 입력받는 방법으로는 크게 유선을 통한 방법과 무선을 통한 방법이 있다. 본 시스템의 특성상 유선을 통한 입력은 제외하였으며, 311MHz 대역의 RF 시그널을 사용한 무선 입력 방식을 사용하였다. 그림 20의 송신부는 마우스의 좌측버튼, 마우스의 우측버튼, 그리고 레이저 포인터 ON/OFF 버튼 3개의 버튼과 함호화부, RF 송신부로 구성되어 있으며 버튼이 눌려 있는 동안 계속해서 RF 시그널을 전송하게 된다.
수신부는 RF 시그널을 수신하는 수신부, 데이터를 해독하는 해독부, 마이크로 컨트롤러로 구성되어 있으며, RF 신호를 해독하고 어느 버튼이 눌렸는지를 판별하여 PC로 전송한다. PC로의 데이터 전송은 속도 9600bps, 데이터 8bit, 패리티 0bit, 정지 1bit의 RS-232 포맷을 사용하였다.

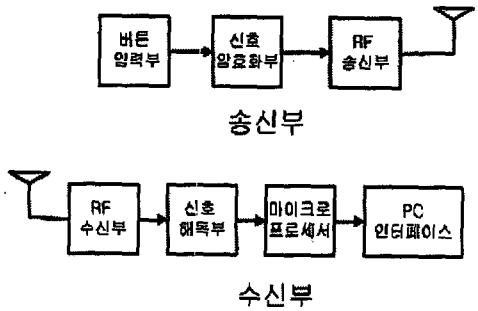


그림 20. 송수신부의 블록도

3-4. 마우스 기능의 구현

변환테이블을 통해 최종적으로 얻어진 마우스 커서의 좌표는 640x480의 해상도를 가지며, 윈도우의 해상도는 프로그램이 실행되는 PC마다 개개의 설정을 가지고 있지만, 윈도우에서는 마우스 커서의 좌표를 해상도에 관계없이 전체 화면을 65536x65536으로 관리한다. 다시 말해서 윈도우의 해상도에 상관없이 윈도우 전체 화면상의 왼쪽 상단의 좌표는 (0, 0)이 되고, 우측 하단은 (65536x65536)이 된다. 그러므로 640x480의 좌표계를 65536x65536의 좌표계로 변환하여야 한다. 최종적인 마우스 좌표는, X좌표는 현재 마우스의 X좌표를 640으로 나눈 값에 65536을 곱하고, Y좌표는 480으로 나눈 값에 65536을 곱하여 얻은 값에 필터를 통과시켜 얻는다.

```
VOID mouse_event(
  DWORD dwFlags, //motion and click options
  DWORD dx,  //horizontal position or change
  DWORD dy, //vertical position or change
  DWORD dwData  //wheel movement
  ULONG_PTR dwExtraInfo
          //application_defined information
);
```

그림 21. 마우스 이벤트 함수

그림 21은 마이크로소프트 비주얼 C++에서 제공하는 마우스 이벤트 함수를 나타내었다. 첫 번째 변수인 dwFlags는 함수의 동작을 정의하는 부분으로서 마우스 커서의 위치이동과 관련된 동작은 MOUSEEVENTF_LEFTDOWN, 좌측 버튼을 눌렀다 떼었을 경우 MOUSEEVENTF_LEFTUP 등과 같이 해당 버튼의 상태를 써주면 된다.

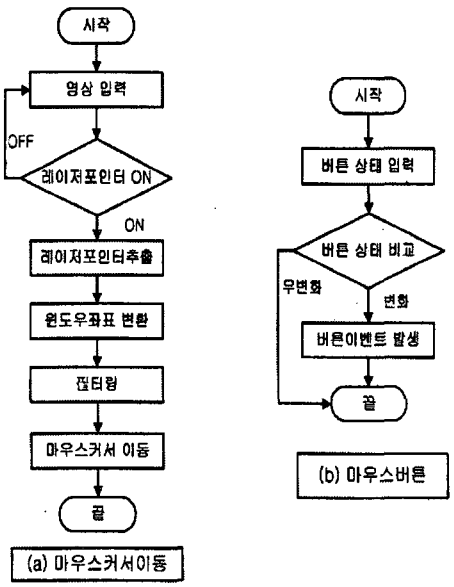


그림 22. 마우스 기능 구현 순서도

프로그램이 실행되고 있는 동안 레이저 포인터의 새로운 위치는 한 필드마다 갱신되므로 초당 60번의 속도로 마우스 이벤트 함수가 호출되어 마우스의 위치를 변경하며, 시리얼 포트를 통해 들어오는 버튼의 상태가 변경될 때마다 마우스 이벤트 함수가 호출된다.
레이저 포인터를 이용하여 마우스 기능을 하기 위한 순서도를 그림 22에 나타내었으며, 그림 22의 (a)는 마우스 커서의 이동과정을 순서도로 나타내었고, (b)는 마우스 버튼의 동작 과정을 나타내었다.

4. 프리젠테이션 시스템의 성능

4-1. 테스트 프로그램

본 논문에서 제안한 프리젠테이션 시스템의 성능 테스트를 위하여 1024x768의 해상도를 갖는 윈도우에서 40x40 화소 크기의 버튼을 랜덤한 위치에 나타내고 버튼을 누르면 다른 위치에 다시 버튼을 나타내도록 하고, 버튼을 누르는데 소요되는 시간과 좌표 오차를 측정할 수 있는 간단한 성능 측정 프로그램을 MFC를 사용하여 제작하였다. 측정방법은 동일 인물에게 컴퓨터 마우스와 본 논문에서 제안한 프리젠테이션 시스템을 사용하게 하여 각각의 결과를 측정하고 비교하였다. 그림 15에 제작한 테스트 프로그램의 외형을 나타내었으며, 프로그램을 실행 후 더블 클릭 되는 순간부터 30개의 버튼이 순차적으로 랜덤한 위치에 나타나게 하였다.

그림 23. 테스트 프로그램

버튼을 누르게 되면 버튼이 나타난 순간부터 버튼이 눌러지기 전까지의 시간과 버튼의 중심으로부터 버튼이 눌린 지점까지의 위치 오차를 측정하여 파일로 기록하였다.

위의 테스트 프로그램을 사용하여 두 가지 방식으로 나누어 테스트를 수행하였다. 한가지 방식은 최대한 빠르게 버튼을 누르도록 하여 속도를 빠르게 했을 때의 각각의 버튼이 눌리는 시간과 오차를 측정하였고, 다른 한가지는 최대한 버튼의 중앙을 누르도록 요청하고 각각의 버튼이 눌리는 시간과 거리 오차를 측정하였다.

4-2. 정밀도 테스트

이 실험은 정밀도 위주로 제안된 프리젠테이션 시스템과 마우스를 사용하여 동일 작업을 하였을 때의 시간차와 거리오차를 측정하기 위한 실험으로, 실험을 하기 전에 실험자에게 불규칙하게 나타나는 버튼을 누를 때 가급적 버튼의 중앙을 누르도록 요청하고, 1회당 30번의 버튼을 누르도록 하고 4회 동안 측정하여 총 120번의 버튼을 누르는 데 소요된 시간을 측정한 결과를 그래프로 나타내었다. 그림 24는 마우스를 이용하여 정밀도 위주의 실험을 하였을 때의 속도를 측정한 결과를 나타내고, 그림 25는 구현된 시스템을 이용하여 정밀도 위주의 실험을 하였을 때의 속도를 측정한 결과이다. 그림 26은 마우스를 이용하여 정밀도 위주의 측정을 하였을 때 나타난 레이저 포인터와 마우스의 거리 오차를 나타낸 그래프이며, 그림 27은 구현된 시스템을 이용하여 동일한 실험을 하였을 때의 거리오차를 나타낸 그래프이다.

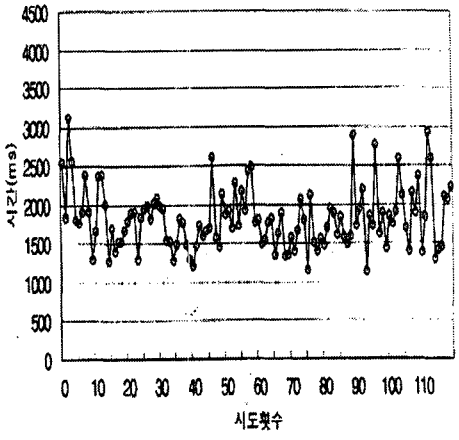


그림 24. 마우스를 이용한 속도 그래프(정밀도 우선)

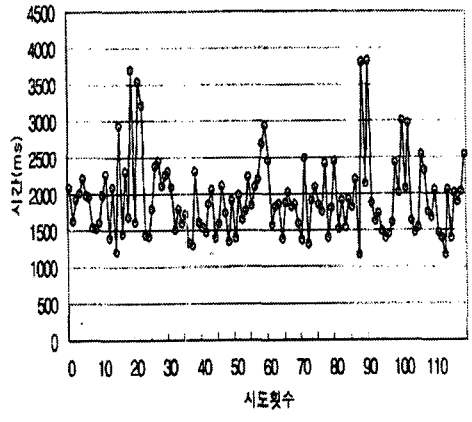


그림 25. 레이져 포인터를 이용한 속도 그래프(정밀도 우선)

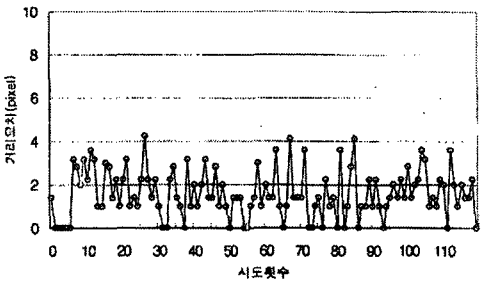


그림 26. 마우스를 이용한 오차 그래프 (정밀도 우선)

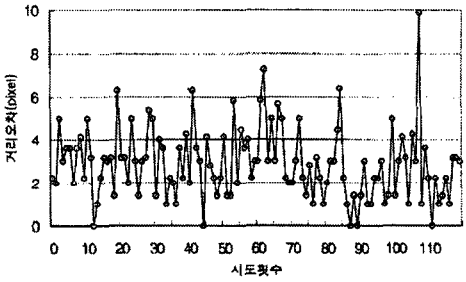


그림 27. 레이져 포인터를 이용한 오차 그래프(정밀도 우선)

4-3. 속도 테스트

이 실험은 속도 위주로 제안된 프리젠테이션 시스템과 마우스를 사용하여 동일 작업을 하였을 때의 시간차와 거리 오차를 측정하기 위한 실험으로, 실험을 하기 전에 실험자에게 불규칙하게 나타나는 버튼을 최대한 빠르게 누르도록 요청하고, 1회당 30번의 버튼을 누르도록 하고 4회 동안 측정하여 총 120번의 버튼을 누르는데 소요된 시간과 거리 오차를 측정한 결과를 그래프로 나타내었다. 그림 28은 마우스를 이용하여 속도 위주의 실험을 하였을 때 속도를 측정한 결과이며, 그림 29는 구현된 시스템을 이용하였을 때의 속도를 측정한 결과를 보여주고 있다. 그림 30은 마우스를 이용하여 속도 위주의 실험을 하였을 때의 거리 오차를 나타내고 그림 31은 구현된 시스템을 사용하여 측정한 거리 오차를 보여준다.

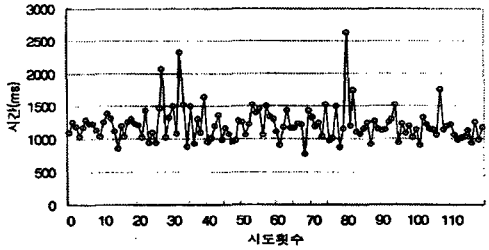


그림 28. 마우스를 이용한 속도 그래프 (속도 우선)

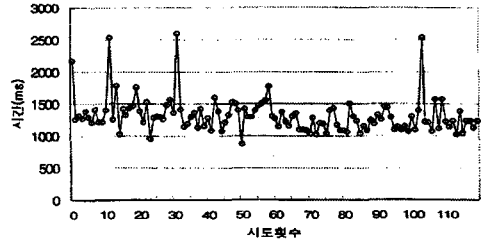


그림 29. 레이져 포인터를 이용한 속도 그래프(속도 우선)

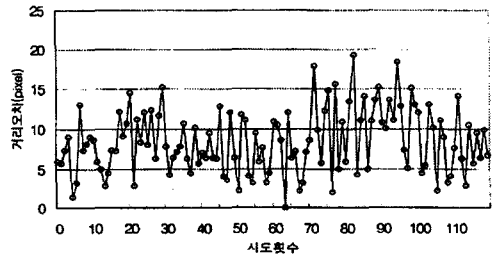


그림 30. 마우스를 이용한 오차 그래프 (속도 우선)

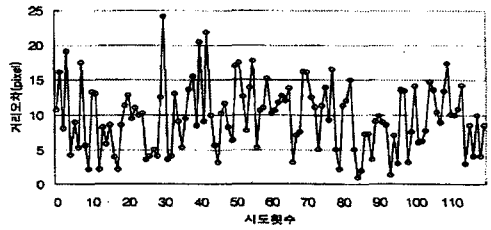


그림 31. 레이져 포인터를 이용한 오차 그래프(속도 우선)

4-4 실험 결과 분석

이상의 실험에서 나타난 결과를 그림 32에 종합하여 나타내었으며, 본 논문에서 제안한 레이저 포인터 시스템을 처음 사용하는 경우임에도 불구하고 익숙한 마우스를 사용하여 측정한 데이터 값과 속도 면에서 거의 차이가 나지 않아 실생활에서의 활용에 문제가 없음을 알 수 있었다. 레이저 포인터와 마우스 커서간의 오차도 마우스를 사용한 실험과 별 차이를 보이지 않았다. 속도 우선의 실험에서는 오히려 마우스를 사용한 실험보다 거리 오차가 더 작게 나타났는데, 이 실험 결과로 마우스를 사용하는 것보다 레이저 포인터를 사용하는 것이 사용자에게 있어서 더 직관적으로 움직이는 것을 알 수 있다. 속도 우선이나 정밀도 우선이나 마우스 사용보다는 속도가 떨어지는 결과를 얻었지만 실제 사용자의 입장에서는 레이저 포인터를 사용하는 것이 체감속도 면에서 더 빠르게 느껴졌다. 이번 실험에서는 마우스 커서가 느리게 반응하는 것을 볼 수 있었는데, 이는 영상입력에 소요되는 지연시간과 레이저 포인터의 흔들림을 보정하기 위하여 필터링 과정을 거치면서 발생하는 커서 이동의 지연시간의 영향으로 생각되며, 이 지연시간을 줄이게 되면 오히려 마우스를 사용한 경우보다 더 빠르게 입력할 수 있을 것이다. 거리 오차 면에서도 정밀도 우선으로 버튼을 클릭했을 경우의 거리 오차가 1024x768 화소의 해상도에서 약 3화소 정도의 차이를 보였다. 이는 이번 실험에서 범용의 레이저 포인터를 사용하여 측정하였기 때문에 레이저 포인터의 전원 스위치를 계속 누르고 있어 손에 힘이 들어간 상태에서 사용하여 레이저 포인터의 떨림이 심하였기 때문으로 생각되며, 원터치 방식의 전원 스위치를 갖는 레이저 포인터를 사용하여 손에 힘을 가하지 않고 사용할 수 있다면 오차를 좀더 줄일 수 있을 것으로 기대된다. 또한 컴퓨터에서 사용하는 폰트의 크기가 대부분 8픽셀 이상이라는 점을 감안할 때 3화소 정도의 오차가 발생한다 하더라도 글자를 지적하는데 무리가 없을 것으로 생각된다.

	마우스속도위주		레이저속도위주		마우스정밀도위주		레이저정밀도위주	
	속도	오차	속도	오차	속도	오차	속도	오차
1회	1204ms	8.6	1411ms	8.1	1905ms	1.8	2052ms	3.1
2회	1261ms	11.4	1375ms	6.9	1786ms	1.3	1839ms	2.8
3회	1229ms	9.4	1192ms	9.4	1673ms	1.5	1900ms	2.9
4회	1155ms	9.1	1274ms	9.0	1911ms	1.6	1945ms	2.5
평균	1212ms	9.6	1313ms	8.39	1819ms	1.59	1934ms	2.86

그림 32. 테스트 결과

5. 하드웨어 구현

레이저마우스 시스템의 입력부분은 입력으로 폐쇄회로 CCD 카메라를 사용하기 때문에 CVBS(CCIR-601 포맷) 입력이 가능한 필립스사의 SAA7111을 채택하였다. 이는 프로그램제어로(I2C-마이크로프로세서 이용) RGB(Red Green Blue) 또는 Y/C(휘도/색성분)를 선택해서 받을 수 있고 디지털 영상을 720*480 화소로 받을 수 있다. 레이저마우스 시스템의 화면 저장과 통신부의 메모리 용량은 한 개의 화면을 저장할 수 있고 레이저의 좌표 및 영상은 시리얼 통신(RS-232)을 이용하여 컴퓨터로 전송된다 마이크로프로세서와 버튼 제어에서 마이크로프로세서(CPU)는 입력부(SAA7111) 를 I2C(Inter-Integrated Circuit) 버스 방식으로 제어하고 제어버튼의 값을 읽어 시리얼 신호로 변환하여 컴퓨터로 전송하는 역할을 한다. CPLD(Complex Plogramable Logic Device)

구성을 위한 부품은 LATTICE사의 iSPLSI3160 을 사용했으며, 일반 TTL은 35㎒의 속도로 동작하는 반면 이는 100㎒의 속도로 동작하고 각 논리 회로간의 시간지연이(3ns) 일정하다. CPLD는 레이저마우스 시스템에서 가장 중요한 역할을 하며 다음과 같은 함수(function)를 수행한다.

- 영상 저장을 위한 메모리 어드레스 발생.
- 영상의 수직. 수평 카운트.
- 레이저 포인트의 최대값 추적
- 좌표 변환(1 Byte -> 2 Byte)
- 레이저 포인트의 좌표 전송

6. 결 론

본 논문에서 제안한 시스템의 중요한 응용분야는 프리젠테이션 분야가 될 수 있으며 스크린에 위치한 포인터의 위치정보를 이용해서 컴퓨터 커서를 이동시키므로 레이저 포인터가 포인팅 기능 외에도 마우스 역할을 동시에 수행하도록 하는 시스템을 제작하였다. 그러므로 레이저 포인터를 사용하면서도 보조요원이 필요치 않고 또한 발표자의 행동반경을 제약받지 않으면서 원활한 프리젠테이션을 진행할 수 있으리라 기대된다.

현재 구현된 시스템은 영상이 입력되는데 소비되는 지연시간을 가지고 있기 때문에 레이저 포인터의 위치로 커서가 이동하는데 지연시간을 가지게 되며, 또한 PC의 자원을 사용하므로 동영상 플레이와 같은 자원을 많이 차지하는 응용분야에는 아직 한계점을 가지고 있다. 향후의 연구에서는 이러한 문제점을 해결하기 위하여 영상 입력부와 레이저 포인터의 위치를 찾는 알고리즘을 HDL을 이용하여 하나의 집적회로에 구현하여 높은 성능과 저가격화를 이루어야 한다.

참고문헌

[1] D. Chen and A. C. Bovic, "Visual pattern image coding", IEEE Trans. Commun. Vol. 38, pp. 2134-2146, 1990

[2] R. Jain, D. Milter, and H. H. Nagel, "Separating Non-Stationary from Atatioonary Scene Components in a Sequence of Real World TV-Images", Proc. 5th Joint Conf. Artificial Inteligence, pp. 612-619, 1997

[3] B. Wohlberg and G. de Jager, "A Review of the Fractal Image Coding Literature", IEEE Trans. on Image Processing, Vol. 8, No. 12, pp. 1716-1729, 1999

[4] NHK 방송기술 연구소 화상연구부, C언어에 의한 화상처리 실무, 국제 테크노 정보연구소, 1994

[5] B. Jones, "Software Serial Port Implemented with the PCA:, Inter, 1988.

[6] A. Chan, "PC Mouse Implementation using COP800". National Semiconductor, 1990.

[7] Dallas Semiconductor, Fundamentals of RS 232 Serial Communications, 1998.

[8] Inter Corporation, MCS@51 Micro Controller Familly User's Manual, 1993.

[9] Conexant Corporation, Fusion 878A PCI Video Decoder, 1999.

[10] W. Oney, System Programming for Windows 95, Microsoft Press, 1996.

박민순
육군사관학교, 이학사
(1968)
고려대학교, EDP, 석사
(1974)
Univ. of North Texas,
BCIS, 석사(1985)
Univ. of North Texas, 정보과학, 박사(1991)
현재 광운대학교 교양학부 교수
관심분야 : 하이퍼텍스트 엔지니어링, 실감미디어