카메라 모션 벡터 추출기를 이용한 임베디드 기반 가상현실 시뮬레이터 제어기의 설계

Implementation of Embedded System Based Simulator Controller Using Camera Motion Parameter Extractor

이희만, 박상조
서원대학교 컴퓨터정보통신공학부

Hee-Man Lee(hlee@seowon.ac.kr), Sang-Jo Park(parks@seowon.ac.kr)

요약

예전의 영상처리 장비는 독립적으로 구현이 되었다고 하여도 단순히 디스플레이만 하는 정도이지만 현재 여러 가지 칩들의 발전으로 인한 그 응용에 있어 활용 범위가 다양해졌다. 본 연구에서는 아날로그 영상신호를 디지털로 변환하여 PC 없이 독립적으로 영상 데이터를 처리하는 시스템을 설계한다. 가상의 움직임에 따라 시뮬레이터를 제어하는 모션 벡터 추출기 및 시뮬레이터 제어기를 설계하고, 추출된 모션 벡터를 이용하여 가상현실 시뮬레이터를 구동한다.

■ 중심어 : | 임베디드 시스템 | 모션 벡터 | 영상신호처리 | 객체탐지 | 시뮬레이터 제어기 | 가상현실 |

Abstract

In the past, the image processing system is independently implemented and has a limit in its application to a degree of simple display. The scope of present image processing system is diversely extended in its application owing to the development of image processing IC chips. In this paper, we implement the image processing system operated independently without PC by converting analogue image signals into digital signals. In the proposed image processing system, we extract the motion parameters from analogue image signals and generate the virtual movement to Simulator and operate Simulator by extracting motion parameters.

■ keyword : | Embedded System | Motion Vector | Image Processing Technology | Object Extractor | Simulator Controller | Virtual Reality |

I. 서 론

최근 영상처리의 분야는 급속도로 발전하고 있다. 우리 생활 속의 비디오 신호는 크게 아날로그와 디지털로 분류되지만, 최근의 비디오 카메라나 비디오 편집기기 등의 내부에서는 디지털신호에 의한 처리가 실행되고 있다. 디지털 카메라, CCTV, DVD, PC용 그래픽 보드 등에서 비디오 신호는 디지털로 처리되는 것이 당연시 되고 있다. 또한 반도체 기술의 진보에 따른 고속 A/D 컨버터 및 메모리의 발전에 따라 영상 영상처리 장비 및 기술도 크게 변하고 있다[1]. 아날로그 영상신호를 디지털 신호로 변환하여 처리화하면 신호를 기록, 재생, 전송하는 것뿐만 아니라 가공, 조정 등 미세한 처리가 가능하므로 여러 가지 면에서 장점이 있다고 볼 수 있다.

접수번호 : #060118-001
접수일자 : 2006년 01월 18일
심사완료일 : 2006년 02월 23일
교신저자 : 박상조, e-mail : parks@seowon.ac.kr

아날로그 비디오 신호로부터 모션 벡터 추출하여 처리할 때 일반적으로 PC에 의존하고 있다[2].

본 논문에서는 임베디드 시스템을 이용하여 TV나 비디오, 카메라 영상 신호로부터 독립적으로 영상 데이터를 처리하는 시스템을 설계한다. 디지털 영상 신호로부터 모션 파라미터를 추출하여[3] 가상의 움직임에 따라 시뮬레이터를 제어하는 모션 벡터 추출기 및 시뮬레이터 제어기를 설계하고, 영상 제어 알고리즘에 대하여 분석한다. 그리고 추출된 모션벡터를 이용하여 가상현실 시뮬레이터를 구동한다.

II. 카메라 모션 추출을 위한 기본 이론

카메라의 이동방향은 [그림 1]과 같이 7가지로 분류할 수 있다. 고정된 위치에서 카메라의 이동방향으로는 Panning(수평방향 회전), Rolling(고정위치에서 회전), Zooming(초점거리 변경), Tilting(수직방향 회전)이 있으며, 가변적인 위치에서 카메라의 이동방향으로는 Booming(수직방향 횡단), Tracking(수평방향 횡단), Dolling(앞뒤로 이동)이 있다. 동영상에서 카메라의 7가지 이동방향을 정확히 분석한다는 것은 거의 불가능하므로 본 연구에서는 카메라가 특정한 위치에 고정된 상태에서 이동하는 방향을 카메라의 부분동작이라고 정의한다. 즉, Tilting은 Booming으로, Panning은 Tracking으로, Zooming은 Dolling으로 표현한다. 이러한 3가지 부분 동작(Booming, Tracking, Dolling)으로 구성되는 카메라 모션 파라미터와 이미 추출된 동영상 모션벡터를 이용하여 카메라 모션의 가속도를 구하며, 이를 3개의 축으로 구성된 가상현실 체험용 시뮬레이터의 구동에 이용한다.

3개의 카메라 모션 파라미터 값을 계산하는 과정과 추출된 모션 파라미터 값과 영상의 모션벡터를 이용하여 카메라 모션의 가속도를 구하는 과정을 설명하면 다음과 같다[4].

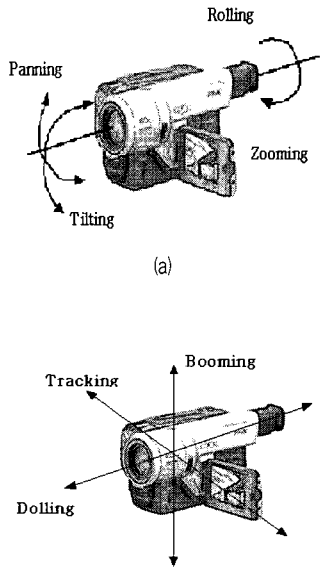

그림 1. 카메라의 모션

[그림 2]에서 COP를 중심으로 좌표 $r(x,y,z)$ 까지의 벡터를 $\vec{r}$ 이라 하고, 이에 대응되는 수직 단위 벡터를 i,j,k 로 표현하면 $\vec{r}$ 은 식(1)과 같다.

$$\vec{r} = xi + yj + zk \tag{1}$$

좌표 $r(x,y,z)$ 이 평면에 투영되는 좌표 $p(u,v)$ 를 이미지 평면상의 중심점 o' 에서 벡터로 표현하면 다음과 같다.

$$\overrightarrow{op} = ui + vj \tag{2}$$

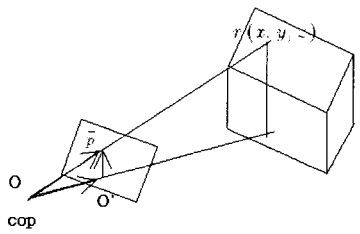

그림 2. 피사체의 투영

조건으로, $\overrightarrow{oo'} = b\boldsymbol{k}$ 인 경우 벡터 $\overrightarrow{op}$ 와 $\vec{r}$ 의 관계를 표현하면 다음과 같다.

$$\overrightarrow{op} = \frac{\vec{r}}{z}|\overrightarrow{oo'}| = b\frac{\vec{r}}{z} = \overrightarrow{oo'} + \overrightarrow{o'p} \quad (3)$$

그러므로 이미지 평면상의 벡터 $\overrightarrow{o'p}$는 벡터 $\overrightarrow{op}$와 벡터 $\overrightarrow{oo'}$ 의 차이 값에 해당되므로 식(4)가 성립된다.

$$\overrightarrow{o'p} = \overrightarrow{op} - \overrightarrow{oo'} = b\frac{\vec{r}}{z} - b\boldsymbol{k} \quad (4)$$

비례식을 이용하여 다음 식을 구한다.

$$b : z = |\overrightarrow{op}| : |\vec{r}| \quad (5)$$

따라서 다음 식이 얻어 진다.

$$\overrightarrow{op} = \frac{b}{z}\vec{r} = \frac{b}{z}x\boldsymbol{i} + \frac{b}{z}y\boldsymbol{j} + \frac{b}{z}z\boldsymbol{k}$$

$$= \frac{b}{z}x\boldsymbol{i} + \frac{b}{z}y\boldsymbol{j} + b\boldsymbol{k} \quad (6)$$

$$\overrightarrow{o'p} = \frac{b}{z}x\boldsymbol{i} + \frac{b}{z}y\boldsymbol{j} \quad (7)$$

한편, $\overrightarrow{op} = u\boldsymbol{i} + v\boldsymbol{j}$ 이므로 식(7)으로부터 이미지 평면의 좌표 $p(u, v)$ 와 공간 좌표 $r(x, y, z)$ 의 관계를 식(8)과 같이 얻을 수 있다.

$$u = \frac{b}{z}x \ , \quad v = \frac{b}{z}y \quad (8)$$

모션벡터를 추출하기 위하여 동영상에 포함된 카메라의 이동방향을 역추적 한다. 따라서 이미지 평면에서 모션벡터의 변화로부터 카메라의 움직임을 계산해야 한다. 이를 위해 공간상의 속도벡터 $\frac{d\vec{r}}{dt}$와 이미지 평면상의 모션벡터 변화율 $\frac{du}{dt}$, $\frac{dv}{dt}$와의 관계식을 유도해야 한다. 벡터 $\vec{r}$ 의 변화율 $\dot{\vec{r}}$ 와 벡터 $\overrightarrow{o'p}$ 의 변화율($\dot{\overrightarrow{o'p}}$) 은 각각 식(9), (10)과 같다.

$$\frac{d\vec{r}}{dt} = \dot{\vec{r}} = \dot{x}\boldsymbol{i} + \dot{y}\boldsymbol{j} + \dot{z}\boldsymbol{k} \quad (9)$$

$$\frac{d}{dt}(\overrightarrow{o'p}) = \frac{d}{dt}(\frac{b}{z}\dot{\vec{r}} - b\boldsymbol{k})$$

$$= \frac{d}{dt}(\frac{b}{z}x\boldsymbol{i} + \frac{b}{z}y\boldsymbol{j})$$

$$= (b\frac{\dot{x}}{z} - \frac{bx\dot{z}}{z^2})\boldsymbol{i} + (\frac{b\dot{y}}{z} - \frac{by\dot{z}}{z^2})\boldsymbol{j}$$

$$= \dot{u}\boldsymbol{i} + \dot{v}\boldsymbol{j} \quad (10)$$

여기서 $\frac{b}{z}\dot{x}$ 는 수평으로 이동하는 성분이며, $\frac{bx\dot{z}}{z^2}$ 는 이동속도의 성분을 나타내고,

$$\frac{du}{dt} = \dot{u} = b\frac{\dot{x}}{z} - \frac{bx\dot{z}}{z^2} \quad (11)$$

$$\frac{dv}{dt} = \dot{v} = \frac{b\dot{y}}{z} - \frac{by\dot{z}}{z^2} \quad (12)$$

여기에서 $\dot{x} = 0$, $\dot{y} = 0$인 경우 다음 식을 유도할 수 있다.

$$\dot{u} = -\frac{bx\dot{z}}{z^2} \quad (13)$$

$$\dot{v} = -\frac{by\dot{z}}{z^2} \quad (14)$$

따라서 $\dot{z} > 0$ 인 경우는 물체가 카메라로부터 멀어지거나, 카메라가 물체로부터 멀어지는 경우로 모션벡터는 식(13), (14)로부터 [그림 3]과 같이 구할 수 있다.

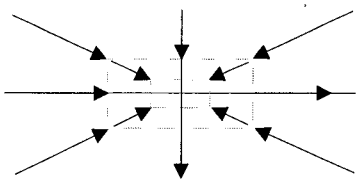

그림 3. 카메라가 물체에서 멀어지는 경우의 모션벡터

또한 만일 $\dot{z} = 0, \dot{y} = 0, \dot{x} > 0$인 경우 식(11), (12)에서 $\dot{u} = b\frac{\dot{x}}{z}$, $\dot{v} = 0$이 된다. 즉, 카메라가 x 축으로 이동하는 경우 모션벡터의 방향은 [그림 4]와 같다.

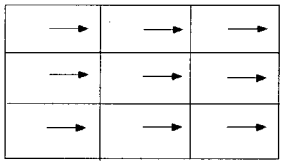

그림 4. 카메라가 x축으로 이동하는 경우의 모션벡터 방향

III. 카메라 모션 벡터 추출기의 설계

카메라 모션 벡터 추출기는 아날로그 영상신호를 디지털로 바꿔주는 A/D 변환기와 디지털로 변환된 영상정보를 처리하는 CPU 및 주변기기를 사용하여 설계한다. 카메라 모션 벡터 추출기의 구성은 [그림 5]와 같다. 제일 먼저 결정할 하드웨어 소자는 A/D 컨버터이다. A/D전용의 칩으로 SAA7111A를 이용하였다[5].

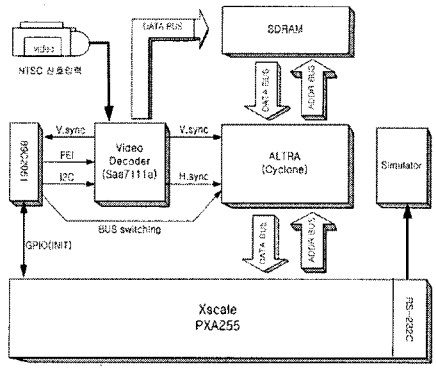

그림 5. 카메라 모션 벡터 추출기의 구성도

1. A/D 변환기

BT계열의 칩들은 PCI(Peripheral Component Interconnect)컨트롤러가 내장되어 PC 슬롯에만 사용가능하므로 제외시켰다. SAA7111A를 제어하기 위하여 레지스터를 설정해야 한다. 이 레지스터는 IIC (Inter-IC)버스 방식을 이용하여 설정하도록 되어있다. IIC 버스는 필립스사가 제안한 통신방식으로 2라인을 사용하여 병렬로 많은 수의 칩을 제어할 수 있는데 구성은 [그림 6]과 같다.

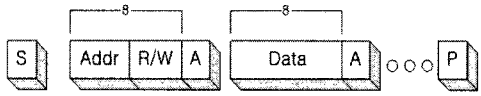

그림 6. IIC버스 데이터의 구성

개발에 사용한 Xscale 프로세서는 기본적으로 IIC 설정을 위한 인터페이스 SCL, SDA을 제공하지만 프로세서는 영상데이터 처리에 관련된 동작만 수행 하고자 별도의 컨트롤러를 사용하였다. 컨트롤러는 ATMEL사의 89C2051을 사용하였고 A/D 컨버터 레지스터 설정 및 인터럽트 루틴을 이용한 스위칭 기능을 하도록 [그림 7]과 같이 설계하였다.

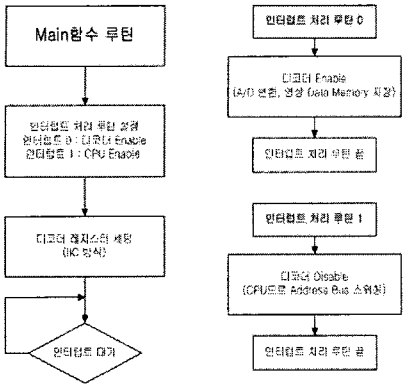

그림 7. IIC를 이용한 A/D 컨버터 셋팅

2. 메모리 및 CPU

A/D컨버터를 통해서 디지털화된 영상 정보를 메모리에 저장을 해야 하는데 SAA7111A에는 어드레스 발생기가 없으므로 일반 SRAM에 저장하기 위하여 FPGA를 이용하였다[6]. 영상처리용 메모리로 FIFO도 많이 사용되지만 더욱 실시간적인 처리를 위하여 일반메모리를 사용하였고 어드레스 발생기부분에서 일반 TTL 소자를 이용할 수도 있지만 칩 스피드와 회로설계 용이성을 위하여 [그림 8]과 같이 FPGA를 이용하였다. 또한 프로세서와 디코더가 메모리를 공유 하도록 bus switch 부분을 구현하였다.

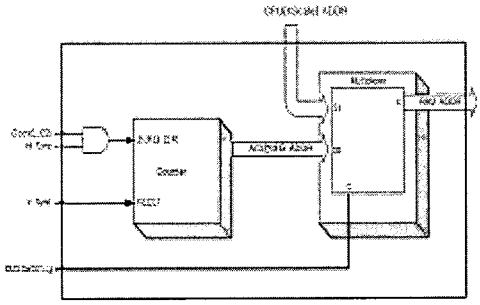

그림 8. FPGA 내부 블록도

A/D 컨버터와 FPGA를 이용하여 영상 데이터를 메모리에 저장을 하였지만 아직은 처리는 되지 않은 상태이므로 프로세서를 이용하여 데이터를 응용할 수 있다. 영상 처리 프로세서를 선택할 때는 메모리 확장과 처리 속도 면에서 고려를 해야 할 것이다. 영상 데이터 처리를 위한 프로세서는 주로 DSP를 많이 이용하였지만, 임베디드급 프로세서의 발전으로 인하여 고속의 연산, 데이터 처리뿐만 아니라 OS의 운영이 가능하고 다양한 기능을 활용할 수 있는 Xscale PXA255를 프로세서로 이용하였다[7]. A/D 컨버터를 통하여 디지털로 변환되어 메모리에 저장된 영상 데이터는 PC에 의존하지 않고 독립적으로 영상을 처리 하여 여러 방면에 이용할 수 있을 것이다.

IV. 모션 파라미터 처리기법

영상처리 알고리즘들을 분류하는 기본적인 분류방법은 4가지가 있는데, 포인트 처리, 영역처리, 기하학적 처리, 그리고 프레임 처리이다. 포인트 처리는 화소의 원래 값이나 위치에 기반한 화소값을 변경한다. 영역처리는 화소의 원래 값과 이웃하는 화소의 값을 기반으로 하여 화소값을 변경한다. 기하학적 처리는 화소들의 위치나 배열을 변화시킨다. 프레임 처리는 두 개 이상의 영상들에 대한 연산을 기반으로 하여 화소값들을 생성한다[1][8-11].

1. 모션 파라미터 검출을 위한 전처리

일반적으로 움직이는 물체를 분석하기 위해서는 두 프레임간의 차분 영상(difference image)을 통하여 움직임 영역을 검출할 수 있다. 본 연구에서는 배경이 고정되었다는 가정아래 이동 객체의 밝기의 시변값에 대해서 적절한 임계값을 적용하여 움직임 여부를 판단하여 모션 파라미터를 검출한다.

1.1 객체의 탐지

연속되는 영상 속에서 이동 객체의 움직임을 탐지하고 이동 객체의 추출을 위하여 연속 프레임간의 차영상을 [그림 9]와 같이 추출한다. 두 영상 f1(x,y)와 f2(x,y)의 차는 f1과 f2의 해당하는 화소끼리의 차를 계산함으로써 구할 수 있다.

$$g(x,y)= f1(x,y)-f2(x,y) \quad (15)$$

그림 9. 프레임간 차(差)의 영상

영상으로부터 객체의 움직임을 탐지하는 경우, 조명의 변화와 그 외의 여러 요인으로 인하여 두 영상 간에 별로 중요하지 않은 차이가 생길 수 있기 때문에 영상인 차 영상에 임계값을 처리를 하여 실제 변화만 추출하였다. 차가 임계값 보다 작으면 영으로 놓고, 크면 화소가 가질 수 있는 최대값으로 설정하여 변화를 뚜렷하게 보이도록 한다.

1.2 객체의 영역탐지

임계값을 처리하여 탐지된 객체를 최소 사각형으로 나타내어야 한다. 객체의 수평 및 수직의 최소 및 최대 값을 [그림 10]과 같이 이용하여 구할 수 있다.

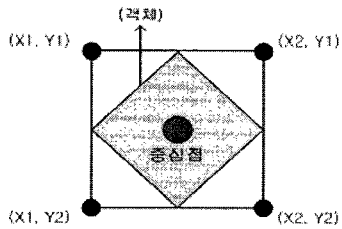

그림 10. 최소사각형의 중심점

여기서 x1=min(수평), x2=max(수평), y1=min(수직), y2=max(수직)이 된다. 탐지된 객체의 정확한 모션 파라미터를 구하기 위해 최소 사각형으로부터 중심점을 구한다.

$$(x',\ y') = \left(\frac{\sum_{x1}^{x2}\sum_{y1}^{y2} Px,}{\sum_{x1}^{x2}\sum_{y1}^{y2} dIx(x,y)}, \frac{\sum_{x1}^{x2}\sum_{y1}^{y2} Py,}{\sum_{x1}^{x2}\sum_{y1}^{y2} dIx(x,y)} \right)$$

$$\begin{cases} Px = xx,\ Py = y & \text{if} \quad dI(x,y) = 1 \\ Px = 0,\ Py = 0 & otherwise \end{cases} \qquad (16)$$

1.3 객체간의 모션 파라미터 검출

차 영상에 의해서 구해진 객체를 최소 사각형으로 중심점을 구하면 현재 프레임의 객체중심점과 다음 프레임의 객체중심점 간의 이동영역을 검출하고 객체들의 영상 차이 위치와 거리의 값에 따라서 [그림 11]과 같이 모션 파라미터를 추출한다.

2. 모션 파라미터 추출

이미지 평면을 일정 구획으로 나누어 각 구역마다 [그림 12]와 같이 모션벡터를 구하고, 이들의 평균 값 $\bar{u}$ $\bar{v}$ 으로부터 모션 파라미터를 추출한다.

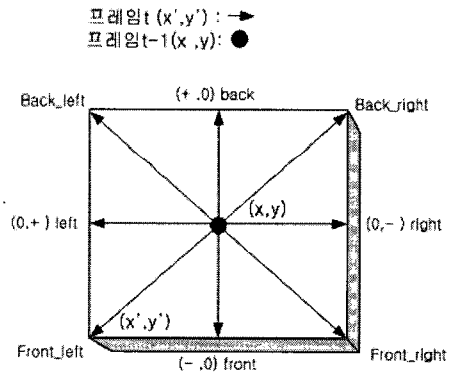

그림 11. 객체간의 위치 파라미터

3	2	1
4	9	8
5	6	7

그림 12. 영상 구획번호

식(11), (12)에서 평면모션벡터의 성분을 계산하면 다음과 같다.

$$\bar{u} = \sum_{i=1}^{N} u_i = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{b}{z_i}\bar{x}_i - \frac{b x_i z_i}{z_i^2}\right) \qquad (17)$$

$$\bar{v} = \sum_{i=1}^{N} v_i = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{b}{z_i}\bar{y}_i - \frac{b y_i z_i}{z_i^2}\right) \qquad (18)$$

식(17), (18)에서와 같이 모션벡터의 평균속도는 카메라의 x축 또는 y축에 대한 이동성분과 z축에 대한 이동성분의 조합으로 이루어진다. 이미지 평면상의 $\bar{u}$와 $\bar{v}$는 이미지 평면상의 여러 블록으로부터 각각의 모션벡터를 추출하여 이들의 평균 값을 계산하여 구할 수 있다. 이미지 평면상의 블록 중심점을 기준으로 대칭시켜 대칭되는 블록들 중에서 작은 블록을 선택하면 복수개의 블록이 이미지 평면상에서 동일한 피사체의 영상이 되므로 $z_i = z_0 = const$가 성립된다. 이미지 평면을 [그림 12]와 같이 9개의 영역 블록으로 나눈다면 블록 9는 이미지 평면상의 중심점에 해당된다.

식(17), (18)에서 작은 블록을 선택한다고 가정을 하면 다음 식이 얻어진다.

$$\dot{x}_i = \dot{x} \quad \dot{y}_i = \dot{y} \quad \dot{z}_i = \dot{z} \tag{19}$$

대칭적인 블록 좌표로부터 다음 식을 얻을 수 있다.

$$\begin{aligned} z_1 &= z_2 = z_3 \cdots = z_0 \\ x_1 &= x_7 = x_8 = x_0 \\ x_3 &= x_4 = x_5 = -x_0 \\ x_2 &= x_9 = x_6 = 0 \end{aligned} \tag{20}$$

따라서 식(13)으로부터 다음 식을 구할 수 있다.

$$\begin{aligned} \dot{u}_1 &= \frac{b}{z_0}\dot{x} - \frac{b}{z_0^2}x_0\dot{z} \\ \dot{u}_2 &= \frac{b}{z_0}\dot{x} \\ \dot{u}_3 &= \frac{b}{z_0}\dot{x} + \frac{b}{z_0^2}x_0\dot{z} \\ \dot{u}_4 &= \frac{b}{z_0}\dot{x} + \frac{b}{z_0^2}x_0\dot{z} \\ \dot{u}_5 &= \frac{b}{z_0}\dot{x} + \frac{b}{z_0^2}x_0\dot{z} \\ \dot{u}_6 &= \frac{b}{z_0}\dot{x} \\ \dot{u}_7 &= \frac{b}{z_0}\dot{x} - \frac{b}{z_0^2}x_0\dot{z} \\ \dot{u}_8 &= \frac{b}{z_0}\dot{x} - \frac{b}{z_0^2}x_0\dot{z} \\ \dot{u}_9 &= \frac{b}{z_0}\dot{x} \end{aligned} \tag{21}$$

따라서 다음과 같은 카메라 모션 파라미터를 추출한다.

$$\frac{\dot{u}_1 + \dot{u}_2 + \cdots \dot{u}_9}{9} \quad \frac{b}{z_0}\dot{x} = \frac{1}{m}\dot{x} \tag{22}$$

$$\frac{(\dot{u}_3 + \dot{u}_4 + \dot{u}_5) - (\dot{u}_1 + \dot{u}_7 + \dot{u}_8)}{6}$$

$$= \frac{b}{z_0^2}x_0\dot{z} = \frac{1}{m}\frac{x_0}{z_0}\dot{z}$$

$$= \frac{1}{m}\tan\left(\frac{FOV_x}{2}\right)\dot{z} \tag{23}$$

여기서 $\frac{b}{z_0}$는 카메라 렌즈 배율 m의 역수에 해당한다. 또한 $\frac{x_0}{z_0}$는 카메라의 x축에 대한 FOV(Field of View)의 각도이다. 한편 식(12)으로부터 v에 대해 위와 유사한 방법으로 식(24), (25)를 구할 수 있다.

$$\frac{\dot{v}_1 + \dot{v}_2 + \cdots \dot{v}_9}{9} = \frac{b}{z_0}\dot{y} = \frac{1}{m}\dot{y} \tag{24}$$

$$\frac{(\dot{v}_3 + \dot{v}_4 + \dot{v}_5) - (\dot{v}_1 + \dot{v}_7 + \dot{v}_8)}{6}$$

$$= \frac{b}{z_0^2}y_0\dot{z} = \frac{1}{m}\frac{y_0}{z_0}\dot{z} = \frac{1}{m}\tan\left(\frac{FOV_y}{2}\right)\dot{z} \tag{25}$$

즉, 이미지 구획의 모션벡터$\{(\dot{u}_1, \dot{v}_1) \cdots (\dot{u}_9, \dot{v}_9)\}$를 구하고 식(22), (23), (24)를 이용하여 카메라의 모션 파

라미터인 $\dot{x}$, $\dot{y}$, $\dot{z}$ 를 구한다.

시뮬레이터는 [그림 13]과 같이 모션 프레임은 3개로 구성된다. 하단 프레임, 상단 프레임, 상단과 하단을 연결하는 3개의 전자기계적 구동기로 구성된다. 카메라 모션 벡터 추출기에 추출된 모션 벡터를 RS232 포트를 사용하여 전송시켜 시뮬레이터로 하여금 기본적인 동작을 구동하였다.

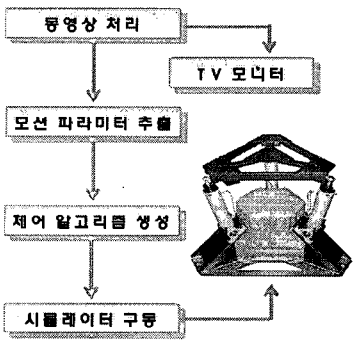

그림 13. 시뮬레이터 제어 구성도

3. 제어 알고리즘의 생성

카메라 모션의 가속도 값 $\dot{x}$, $\dot{y}$, $\dot{z}$로부터 시뮬레이터를 구동시킬 수 있는 제어 명령어를 생성한다. 본 논문에서 사용하는 시뮬레이터는 3개의 축으로 구성되어 있으므로 카메라 모션의 가속도 값에 따라 3개의 축을 적절히 제어할 수 있어야 한다. 가속도 값과 시뮬레이터 축의 위치이동 관계는 선형이 아닌 비선형 관계이므로 뉴럴 네트워크의 한 종류인 역전파망을 이용한다. [그림 14]와 같이 뉴럴 네트워크는 카메라 모션의 가속도 값 $\dot{x}$, $\dot{y}$, $\dot{z}$와 시뮬레이터 3개 축의 현재 위치 값 L_1,L_2,L_3를 입력벡터로 하여 시뮬레이터 축의 다음 이동위치 값인 L_1',L_2',L_3'를 출력한다. 시뮬레이터의 현재 위치 값과 이동위치 값의 차이를 계산하여 시뮬레이터의 축을 제어할 수 있는 명령어를 생성할 수 있다. 그림 14에서와 같이 역전파망은 총 21개의 뉴럴로 구성되며, 입력층은 6, 중간층은 12개, 출력층은 3개의 뉴럴로 구성된다. 역전파망을 학습시키기 위해서는 훈련 데이터를 이용해야 한다. 시뮬레이터 제어 알고리즘의 구성요소는 [그림 15]와 같으며, 가속도 값 $\dot{x}$, $\dot{y}$, $\dot{z}$와 정규화 되어 나온 결과값 $\widehat{L_1}$, $\widehat{L_2}$, $\widehat{L_3}$을 입력으로 하여 뉴럴 네트워크에서 훈련을 하여 새로운 시뮬레이터의 위치값인 L_1',L_2',L_3'이 출력된다.

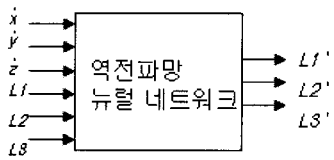

그림 14. 역전파망의 구조

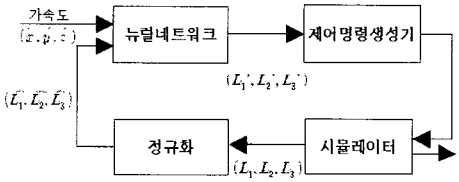

그림 15. 시뮬레이터 제어

제어명령생성기에서는 새로운 시뮬레이터의 위치값에 따라서 시뮬레이터를 가동하기 위한 명령어를 생성하여서 시뮬레이터에 RS232를 통하여 보내게 되면 시뮬레이터는 명령어에 따라서 구동하게 되며, 현재 위치값인 L_1,L_2,L_3이 정규화처리 되어 나온 결과값인 $\widehat{L_1}$, $\widehat{L_2}$, $\widehat{L_3}$이 다시 뉴럴 네트워크의 입력으로 보내지게 된다. 이와 같은 반복적인 작업으로 시뮬레이터를 제어할 수 있다.

4. 실험결과

시뮬레이터는 모션 프레임은 3개로 구성된다. 하단 프레임, 상단 프레임, 상단과 하단을 연결하는 3개의 전자기계적 구동기로 구성된다. RS232 포트를 이용하여 영상 데이터의 모션 파라미터 값을 전송하여 시뮬레이터를 구동한다. 시뮬레이터는 영상 데이터의 동적 변화를 재현함으로써 가상의 움직임을 현실처럼 느낄 수 있도록 하는 것이다.

[그림 16]은 물체의 좌우 이동에 따라 모션 파라미터

를 검출하고 시뮬레이터가 구동하는 모습들을 보여준다.

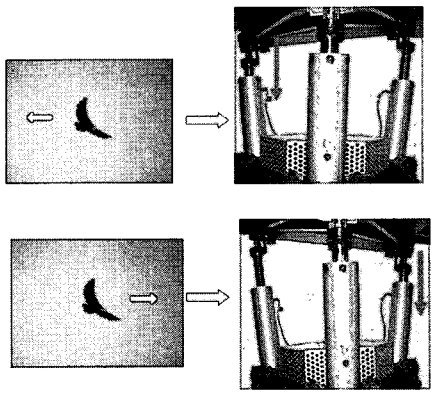

그림 16. 시뮬레이터의 구동

[표 1]은 CPU의 1 프레임의 처리 시간에 따른 좌우로 물체가 이동시 에러율을 나타내고 있다. 프레임 처리 시간이 길수록 A/D 처리시간에 물체가 이동하므로 에러율이 상승하게 된다.

표 1. CPU의 프레임 처리시간에 따른 에러율

1 프레임 처리시간	좌로 이동	우로 이동
10 mS	8%	5%
20 mS	9%	6%
30 mS	19%	16%
40 mS	25%	27%

[표 2]에서 훈련용 실험 데이터는 가속도 센서를 부착하여 시뮬레이터 3개의 축을 직접 움직여 가면서 측정한 가속도 값과 3개 축의 위치 측정값으로 생성한다. 표에서 3개축의 위치값은 실제 사용될 시뮬레이터 3개 축의 위치값을 의미한다. 시뮬레이터의 3개 축의 위치값은 -14000~14000 값을 사용한다.

[그림 17]은 가속 x, y, z 값과 시뮬레이터 3개 축의 움직임 값과 3축의 합성 움직임 값에 대한 그래프를 나타내고 있다. 견본 데이터를 이용한 실험을 하여 생성되는 훈련용 데이터는 [표 3]에서 보여 주며, 입력벡터 6개, 출력벡터 3개로 구성한다. 실험결과에서 뉴럴 네트워크를 이용하여 구하여진 축들의 값은 표에서 보여주듯이 0~1 사이로 정규화된 것이다.

[그림 18]은 실험 데이터를 그래프로 나타낸 것이다. 데이터는 0~1로 정규화시킨 것이다. 그림에서의 값들은 가속도 x, y, z와 시뮬레이터 3개 축의 위치값 L_1, L_2, L_3와 출력으로 새로운 시뮬레이터 3개 축의 위치값 L_1', L_2', L_3'이다.

표 2. 역전파망 훈련용 실험 데이터

x축	y축	z축	1축 위치값	2축 위치값	3축 위치값
0.1	1	9.9	5015	11821	13806
4.4	1.2	10.3	4545	11449	13905
8.5	6.2	13.5	4073	11044	13970
4.1	0.9	9.1	3631	10629	13998
-1.2	-1.3	9.8	3205	10197	13991
2.9	1.5	11.2	2798	9748	13950
4.5	-0.4	10.4	2413	9286	13873
4.8	2.8	13.2	2040	8797	13758
5.7	3.1	11.2	1703	8314	13611
4.5	2.5	14.5	1392	7824	13432

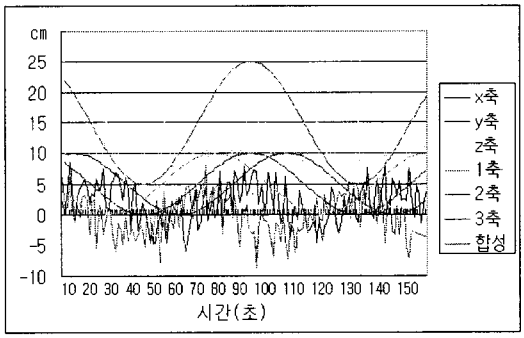

그림 17. 뉴런의 그래프

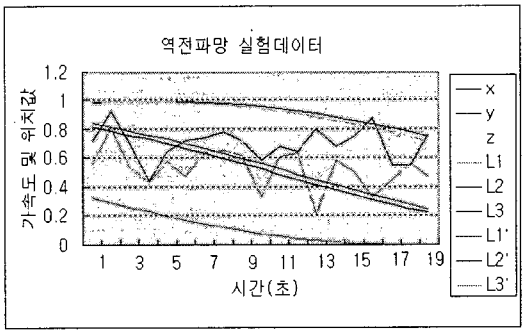

그림 18. 역전파망 실험 데이터 그래프

V. 결 론

예전의 영상처리 장비는 독립적으로 구현이 되었다고 하여도 단순히 디스플레이만 하는 정도였지만 현재 여러 가지 반도체 칩들의 발전으로 인한 그 응용에 있어 활용 범위가 다양해졌다.

본 논문에서는 아날로그 영상신호를 디지털로 변환하여 임베디드 시스템을 이용하여 독립적으로 영상 데이터를 처리하였다. 디지털 영상 신호로부터 모션 파라미터를 추출하여 가상의 움직임에 따라 시뮬레이터를 제어하는 모션 벡터 추출기 및 시뮬레이터 제어기를 설계하고, 영상 제어 알고리즘에 대하여 분석하였다. 그리고 추출된 모션벡터를 이용하여 가상현실 시뮬레이터를 구동하였다.

향후 계획으로는 시뮬레이터를 보다 현실감이 높은 제어가 되도록 하는 것이며 또한 교육용 콘텐츠와 연동이 되는 가상현실 e-Learning 시스템을 개발하고자 한다.

표 3. 역전파망 훈련 데이터

입력벡터						출력벡터		
$\bar{x}$	$\bar{y}$	$\bar{z}$	L_1	L_2	L_3	L'_1	L'_2	L'_3
0.72	0.59	0.515	0.358198	0.844333	0.988135	0.324628	0.817818	0.99319
0.925	0.81	0.675	0.324628	0.817818	0.99319	0.290962	0.788835	0.997873
0.705	0.545	0.455	0.290962	0.788835	0.997873	0.259349	0.759234	0.999885
0.44	0.435	0.49	0.259349	0.759234	0.999885	0.228941	0.728333	0.999391
0.645	0.575	0.56	0.228941	0.728333	0.999391	0.199893	0.696287	0.996394
0.725	0.48	0.52	0.199893	0.696287	0.996394	0.172349	0.663257	0.990909
0.74	0.64	0.66	0.172349	0.663257	0.990909	0.145689	0.628373	0.982684
0.785	0.655	0.56	0.145689	0.628373	0.982684	0.121615	0.593859	0.972245
0.725	0.625	0.725	0.121615	0.593859	0.972245	0.099437	0.558875	0.958438
0.59	0.33	0.43	0.099437	0.558875	0.958438	0.079268	0.523596	0.944328

[감사의 글]

저자들은 모션 벡터 추출기의 설계를 담당하였던 최용호군에게 감사드린다.

참 고 문 헌

[1] 최형일, *영상처리 이론과 실제*, 홍릉과학출판사, pp.175-178, 1997.

[2] 장석우외, "적응적 가중치 함수를 이용한 모션 벡터의 필터링", 한국정보과학회논문지, 제31권, 제11호, pp.1474-1482, 2004.

[3] 최용호, 이희만, 박상조, "임베디드 시스템을 이용한 모션 벡터 추출 및 시뮤레이터 제어기의 설계", 한국콘텐츠학회 추계종합학술대회, pp.181-184, 2003.

[4] 서정만, "모션벡터를 이용한 가상현실 체험 시스템의 구현", 박사학위논문, 2003.

[5] SAA7111A Enhanced Video Input Processor, PHILIPS, 1997.

[6] *XScale Microarchitecture for the PXA255 Processor Manual*, Intel, 2003.

[7] 이준성, *Xilinux Foundation을 이용한 디지털 시스템 설계*, 복두출판사, 2001.

[8] *비디오 신호처리시스템 설계*, 월간 전자기술, 제7호, 2001.

[9] 김충원, *MATLAB을 이용한 디지털 영상처리*, 홍릉과학출판사, 2003.

[10] I. Patas, *Digital Image Processing Algorithm*, Prentice Hall, 1993.

[11] 이문호, *영상 통신의 신호 처리*, 대영사, 2000.

저 자 소 개

이 희 만(Hee-Man Lee) 정회원

- 1984년 2월 : 고려대학교 전자공학과(공학사)
- 1986년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
- 1994년 2월 : Texas A&M E.E . (공학박사)
- 1996년 3월~현재 : 서원대학교 컴퓨터정보통신공학부 교수

<관심분야> : 컴퓨터 비전, 멀티미디어, 가상현실

박 상 조(Sang-Jo Park) 정회원

- 1983년 2월 : 서울대학교 제어계측공학과(공학사)
- 1985년 2월 : 서울대학교 대학원 제어계측공학과(공학석사)
- 1999년 3월 : 일본 오사카대학 통신공학과(공학박사)
- 2000년 3월~현재 : 서원대학교 컴퓨터정보통신공학부 교수

<관심분야> : 광무선 액세스 네트워크, 이동통신, 무선 LAN