ActiveMovie에 기반한 디지털 영상 특수 효과

봉시종†, 한희일††, 이의택††, 문영식†
†한양대학교 전자계산학과, ††한국전자통신연구원

ActiveMovie-Based Special Effects for Digital Images

S. J. Bong†, H. I. Hahn††, E. T. Lee††, Y. S. Moon†
†Dept of CSE, Hanyang Univ.
††Electroincs and Telecommunications Research Institute
E-mail: sjbong@cse.hanyang.ac.kr

요 약

본 논문에서는 ActiveMovie와 같은 stream data 기반의 library에서 실시간으로 사용될 수 있는 영상 특수 효과 filter들을 소개하고, 알고리즘 개발 및 구현방법에 대해 기술한다. ActiveMovie는 기본적으로 mpeg decoder 및 Video for Windows(.avi file) decoder를 제공하여 프로그래밍을 수월하게 한다. 또한 각 module에 filter라는 개념을 도입하고 filter graph라는 구조를 이용하여 filter의 추가, 삭제를 용이하게 한다. 본 논문에서는, 디지털 영상 특수효과 중에서 Mosaic, Wind, Ghosting 등의 point processing filter들을 실시간으로 처리하기 위한 고속의 알고리즘을 제안한다. 제안하는 알고리즘은 픽셀의 포인터를 특정 위치로 이동시키지 않고 단순히 주소값을 하나씩 증가시키는 연산을 이용하여 실시간 특수효과를 얻을 수 있도록 한다. 또한 이와같은 특수효과 알고리즘들을 ActiveMovie환경에서 구현함으로써 제안된 기법에 의하여 실시간 동영상 특수효과 처리가 가능함을 입증한다.

I. 서론

개인용 컴퓨터의 성능 향상으로 인하여 과거 값비싼 특수 장비를 이용하여야 실현 가능했던 동영상 특수효과가 개인용 컴퓨터에서도 가능하게 되었다. 그 예로 Adobe Premier나 Media 100을 들 수 있다. 또한 멀티미디어 데이터 stream을 제어하고 처리하는 ActiveMovie SDK의 출현으로 PC환경에서 동영상 특수효과를 위한 프로그램 제작이 더욱 용이해졌다. 본 논문에서는 ActiveMovie와 같은 stream data 기반의 library에서 실시간으로 사용될 수 있는 영상 특수효과 filter들을 소개하고, 알고리즘 개발 및 구현방법에 대해 기술한다.

II. ActiveMovie의 구조

ActiveMovie의 구조는 filter graph라 불리는 구성요소내에서 단위 요소인 filter를 사용하여 time-stamped multimedia 데이터의 흐름을 어떻게 제어하고 처리하는가를 정의한다. 또한 ActiveMovie는 COM(Component Object Model) 구조를 기반으로 동작한다. 그림 1은 ActiveMovie와 응용프로그램과의 관계를 나타낸 그림이다. 그림에서 보는 것처럼 MCI를 사용하거나 OLE Control을 사용하거나 모두 COM interface를 거쳐서 응용프로그램과 통신을 한다.

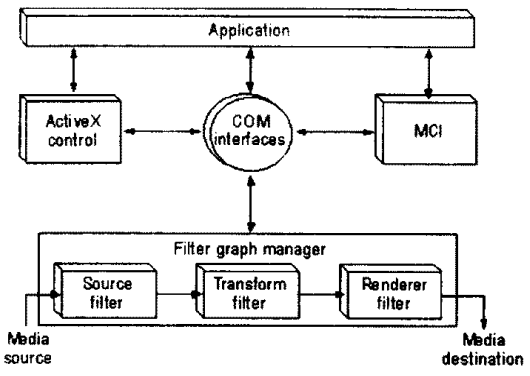


그림 1. ActiveMovie와 응용프로그램과의 관계

1. Filter graph와 filter

Filter graph는 서로 다른 종류의 필터의 배열이며 대부분의 filter는 대략 다음의 3가지중 하나로 구별지어진다.

· Source filter : 화일이나, 인터넷 서버와 같은 어떤 소스로부터 데이터를 읽는다.

· Transform filter : 전형적으로 데이터를 처리하는 필터이다.

· Rendering filter : 일반적으로 하드웨어 장치에 표현해주는 필터지만, 입력으로 받아들여지는 어떤 위치로도 표현가능하다.

예를 들어, MPEG의 경우 필터그래프의 구조는 그림 2와 같다.

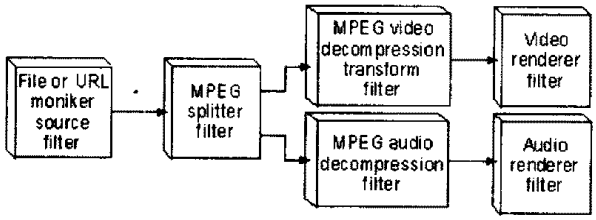


그림 2. MPEG 디코더의 필터 그래프 모양

현재 ActiveMovie는 DirectShow라는 이름으로 변경되어 사용되고 있다.

III. 실시간 특수효과 알고리즘

ActiveMovie SDK는 동영상을 디코딩한후 스트림데이터 형태로 다음 필터에 넘겨준다. 그러므로 특수효과 필터 제작자는 이 스트림 데이터를 처리함으로써 화면에 다양한 효과를 줄 수 있다. 특수효과 필터 처리를 위한 전체적인 흐름도는 그림 3과 같다.

1. Mosaic

모자이크의 기본 원리는 모듈로 연산을 통해 해당 픽셀을 원하는 수 만큼 주위에 복사시키는 것이다. 하지만, 본 논문에서는 실시간으로 데이터를 처리하는 것에 중점을 두었으므로, 3개의 포인터를 이용하여 스트림 영상에서 좀더 빠른 모자이크를 구현하고자 하였다. 즉, 원래 영상의 처리할 부분을 가르키는 포인터(prgb)외에 2개의 포인터를 더 도입하여 첫 번째 포인터(prgb2)는 처리할 행의 첫 열을 가르키고 있고 두 번째 포인터(prgb3)는 모자이크 블록이 바뀔 때 마다 새로운

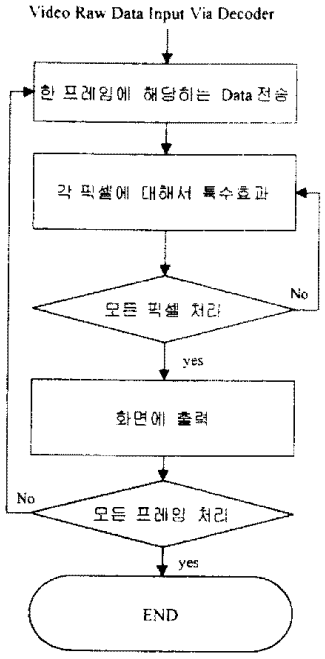


그림 3. 특수효과 필터의 흐름도

블록의 첫 번째 열을 가르키고 있는다. 그림 4는 이를 나타내고 있다.

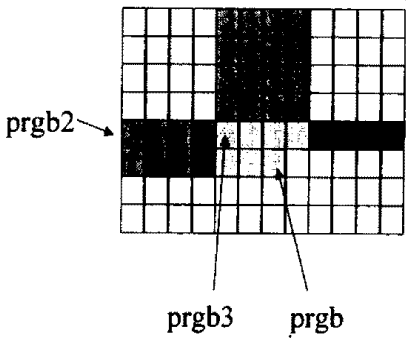


그림 4. 모자이크 효과

세 개의 포인터를 이용한 mosaic 효과의 알고리즘 가상 코드는 아래와 같다.

- input : *blocksize*(모자이크를 수행했을때 블록의 크기), *pInputData*(처리할 영상)

```
PerformMosaic() {
 trgb, *prgb2, *prgb3 : image type;
 prgb ← (image_type*) pInputData;
 prgb3←prgb2←prgb;
 For y ← 0 to height {
  For x ← 0 to width {
   prgb++, prgb2++;
   if (x%blocksize) = 0 then trgb ← *prgb2;
   *prgb ← trgb;
```

```
  } /* end of x-axis*/
  prgb2 ← prgb3;
   /* 한 라인의 처리가 끝난후 prgb2포인터는
    여전히 전 라인을 가르키고 있게 한다. */
  if (y%blocksize) = 0 then
    /* 세로축으로 볼때 다음 블록이 나온 경우 */
    prgb2←prgb3←prgb;
  }
}
```

2. Wind

다른 특수효과와 마찬가지로 스트림데이터를 입력으로 받고 포인터를 하나씩 증가시키며 연산을 수행한다. Wind의 원리는 바람에 흩날린 픽셀이 그 위치에 있던 이전 픽셀보다 더 밝다는 사실과 흩날린 픽셀은 점점 어두워 진다는 사실, 그리고 바람의 방향은 수평이라는 사실을 이용하는 것이다. 입력되는 스트림 데이터를 실시간으로 처리하는데 중점을 두었다.

그림 5는 Wind 효과를 적용했을때의 밝기값(점선으로 표현)을 영상 단면으로 표현한 것이다.

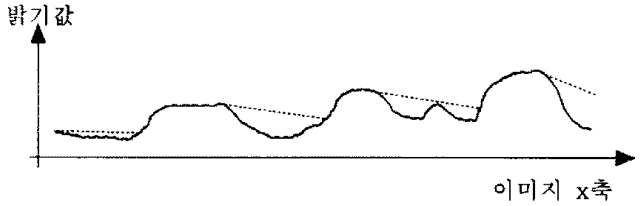


그림 5. Wind 효과 단면도

Wind 효과를 위한 알고리즘 가상 코드는 아래와 같다.

```
  /* direction of wind */
#define LEFT        0
#define RIGHT       1
  /* type of wind */
#define WIND        10
#define BLAST       11
#define STAGGER     12
```

- input : *direction*(바람의 방향), *method*(바람의 형태), *pInputData*(처리할 영상)

```
PerformWind() {
 For all images {
  if direction = RIGHT then {
   /* 바람의 방향이 Right인 경우 */
  prgb ← (image_type*)pInputData;
  For y ← 0 to height {
   For x ← 0 to width {
     /* obtain current value of pixel */
    Rcurr ← prgb->rgbtRed;
    Gcurr ← prgb->rgbtGreen;
    Bcurr ← prgb->rgbtBlue;

    switch(method) {
     case WIND:
      num ← (rand()%10 + 93) / 100.0;
      break;
     case BLAST:
      num ← (rand()%2 + 99) / 100.0;
      break;
     case STAGGER:
      num ← (rand()%10 + 95) / 100.0;
      break;
    }

    Rrand ← Rcurr * num;
    Grand ← Gcurr * num;
    Brand ← Bcurr * num;
    if Grand > 255.0 then Grand ← 255.0;
    if Rrand > 255.0 then Rrand ← 255.0;
    if Brand > 255.0 then Brand ← 255.0;
    /* Random함수에 의해 생성된 점의 밝기값을
     계산한후 다음 픽셀과의 밝기값을 비교하여 더
     밝은 것으로 대체한다. */
    Avgrand = ( Rrand + Grand + Brand ) / 3.0;
    prgb++;
    Avg ← ( prgb->rgbtRed +
    prgb->rgbtGreen + prgb->rgbtBlue) / 3.0;

    if j≠0 and Avgrand > Avg then {
     prgb->rgbtRed ← Rrand;
     prgb->rgbtGreen ← Grand;
     prgb->rgbtBlue ← Brand;
    }
   }
  }

 if direction = LEFT then {
 /* RIGHT인 경우와 같으며 다만 prgb가 감소
 하면서 계산을 한다.*/
 }
```

}

3. Ghosting

이 특수효과는 영상에 잔영을 주는 필터이다. 이 효과는 전 영상과의 더하기 연산을 통해 이루어진다. 먼저번 영상과 더한 영상의 픽셀값을 또다시 2로 나누어 다시 더하는 과정을 반복수행한다. 즉, 그림 6처럼 처음 영상의 잔영이 시간이 지남에 따라 감소하게 된다. 따라서 이 필터는 전번 영상과 현재 프레임의 연산을 위해 버퍼를 가지고 있다.

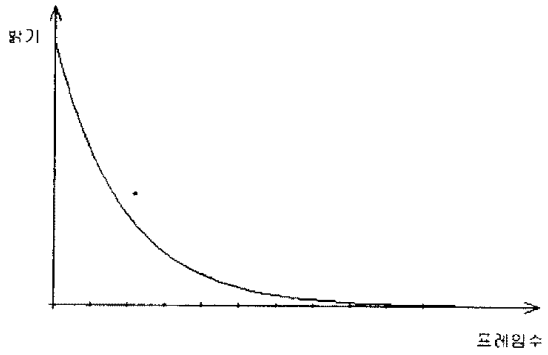


그림 6. Ghosting 효과 적용시 임의의 프레임 밝기 변화

Ghosting 효과를 위한 알고리즘 가상 코드는 아래와 같다.

- Input : *pInputData*(처리할 영상),
pPrevData(전번 영상)

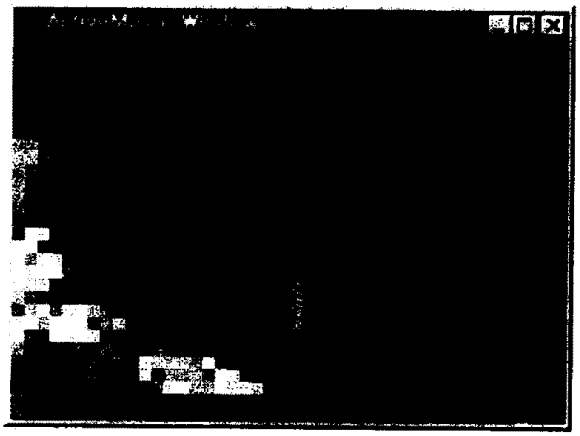
```
PerformGhost() {
 For every pixel p such that p ∈ pInputData {
  pInputData(p) ←
  (pInputData(p) + pPrevData(p)) / 2;
 }
 copy pInputData to pPrevData;
  /* 다음번 덧셈연산을 위해 현재의 프레임을 전
    프레임 버퍼에 복사한다. */
}
```

IV. 구현 결과 및 분석

본 논문의 개발환경은 Pentium 120, 32M RAM, Win95이다. 입력영상은 352x240 mpeg 영상이다.

1. Mosaic 결과

블록사이즈를 8로 했을 때 mosaic효과의 구현 결과는 아래와 같다.



제안한 방법에 의한 구현 결과를 단순히 영역을 복사한 후 픽셀을 분산시키는 방법과 수행시간 면에서 비교한 결과는 표 1과 같다.

표 1. 수행 결과 비교표

수행 회수	고전적 방법 (displayed frames / Average rate)	제안한 방법 (displayed frames / Average rate)
1	127 / 6.25	144 / 7.15
2	127 / 6.25	144 / 7.13
3	125 / 6.14	144 / 7.14
4	127 / 6.23	144 / 7.14
5	126 / 6.22	144 / 7.15

표에서 볼 수 있듯이 제안한 방법에 의하여 수행 속도가 약 15%정도 개선됨을 알 수 있다.

2. Wind 결과

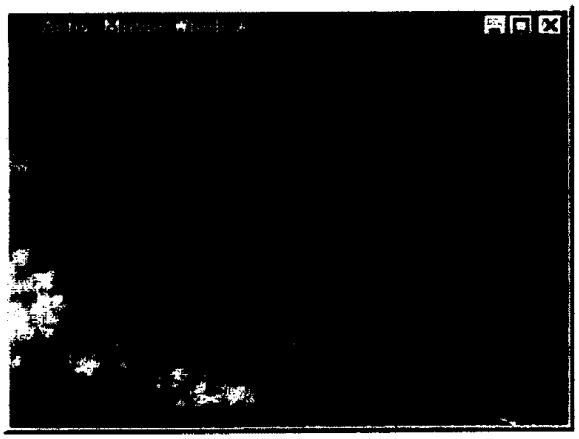
바람의 방향이 오른쪽이고, 바람의 종류가 blast

일때의 wind 효과는 아래와 같다.

즉, 픽셀들이 마치 바람에 흩날리는 듯한 효과를 볼 수 있다.

3. Ghosting 결과

Ghosting 효과의 구현 결과는 아래와 같으며, 전 번영상의 잔영이 남아있는 효과를 볼 수 있다.



V. 결론

본 논문은 ActiveMovie와 같은 스트림 영상 데이터를 제공하는 library에 적합한 실시간 point processing 알고리즘을 제안하였으며, 이의 구현을 통하여 제안된 기법의 효율성을 입증하였다. Mosaic, wind, ghosting 효과에 대한 알고리즘을 제시하였으며, 향후 convolution형태의 필터에 대한 효율적 알고리즘을 개발할 필요가 있다.

참고문헌

[1] Dale Rogerson, "Inside COM," Microsoft Press

[2] 김성욱, Adobe Premiere 4.2, 도서출판 혜지원, 1996

[3] http://www.microsoft.com/devonly/tech/amov1doc, Microsoft

[4] http://www.microsoft.com/msdn/sdk/inetsdk/help/dxm/ds/default.htm, Microsoft