DirectShow를 이용한 비디오 특수 효과 개발

*남기현, **장덕호, *문영식
*한양대학교, **한국전자통신연구원

Development of DirectShow-Based Special Effects for Digital Video

*K. H. Nam, **D. H. Chang, *Y. S. Moon
*Hanyang Univ., **ETRI

요 약

본 논문에서는 다양한 멀티미디어 스트림(multimedia stream)을 재생할 수 있는 DirectShow SDK를 이용하여, 비디오의 특수효과를 구현하기 위한 변환 필터(Transform filter) 제작 방법에 대해 소개한다 DirectShow는 여러 가지 포맷(MPEG, AVI, MOV, WAV)으로 부호화된 digital movie나 sound를 재생할 수 있는 run-time(.ocx)과 dynamic-link library (DLLs)를 제공하여 필터의 제작을 용이하게 한다. 본 논문에서는 디지털 영상 특수효과 중에서 Clip이라는 특수효과를 예로하여 변환 필터를 제작하는 방법을 소개한다. 또한 DirectShow Documentation에서 제공된 변환방법을 수정하여 변환필터의 입력 pin에서 출력 pin으로의 불필요한 복사를 줄임으로써 수행 속도를 개선하는 특수효과 구현 방법을 제시한다

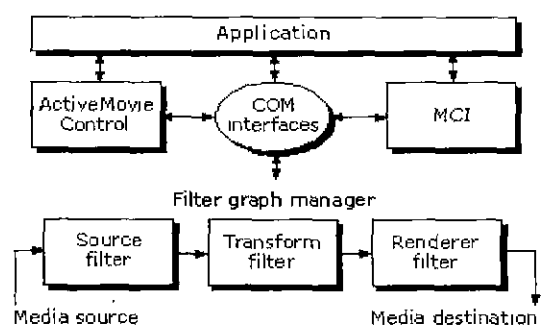
1. 서론

HDTV나 DVD등이 나오면서 디지털 영상의 편집에도 점점 관심이 집중되고 있는 시점에서, 많은 비디오 편집 및 디지털 영상 특수효과를 위한 도구가 개발되었다 그리고, 이러한 도구를 개인용 컴퓨터에서 별도의 장비 없이도 실행하는 것이 가능해 졌다 그 예로서 외국의 Adobe Premiere, Media Studio Pro, Final Cut 등을 들 수 있다.[1] 현재 이 분야의 기술은 몇몇 기업들이 독점하고 있는 상태이고, 국내의 경우 이러한 도구기 거의 전무하며, 이 분야에 대한 기술의 축적 또한 부족한 상태이다. 이에, 본 논문에서는 다양한 멀티미디어 스트림을 재생할 수 있는 DirectShow SDK를 이용하여, 비디오의 특수효과를 구현하기 위한 변환 필터 제작 방법에 대해 소개한다.

2. DirectShow의 구조

마이크로소프트사는 비디오의 재생이나 처리과정 개발을 위한 SDK(Software Development Kit)로 DirectShow SDK를 제공하고 있다.[2] DirectShow는 여러 가지 포맷으로 부호화된 디지털 영상을 재생할 수 있는 run-time(.ocx)과 dynamic-link library (DLLs)를 제공하여 필터의 제작을 용이하게 한다.

DirectShow의 구조는 [그림 1]과 같으며, 필터라는 modular component를 사용하여 어떻게 멀티미디어 스트림을 제어하고 처리할 수 있는지 정의한다.



[그림 1] DirectShow 구조

필터는 입력 또는 출력 pin을 가지고(혹은 둘다 가지고), 필터 그래프(filter graph)라는 구성으로 연결된다.

Application은 필터 그래프 매니저(filter graph manager)라 불

※ 본 연구는 전자 통신 연구원의 지원으로 수행되었음

리는 object를 사용하여 필터 그래프(filter graph)를 생성하고, 데이터가 필터 그래프를 통하여 움직이게 한다.

[그림 1]에서와 같이 DirectShow의 구조는 Application이 COM Interface, ActiveMovie Control 또는 Media Control Interface (MCI)를 통하여 DirectShow를 access 할 수 있도록 되어 있다.

필터 그래프는 다양한 종류의 필터들의 집합으로 구성되며, 대부분의 필터들은 다음과 같이 3가지 종류로 구분할 수 있다.

- 소스 필터(Source filter) : 디스크, 인공위성, 인터넷 서버, 또는 VCR에서 소스를 얻어오고, 필터 그래프에 알리는 역할을 한다
- 변환 필터(Transform filter) : 데이터를 받아서, 원하는 처리를 해주고, 바뀐 데이터를 내보낸다.
- 렌더링 필터(Rendering filter) : 하드웨어 장치에 렌더링을 해준다. 메모리나 디스크 파일에도 렌더링 할 수 있다

이 논문에서는 변환 필터의 하나인 effect 필터(데이터의 type은 변경시키지 않고, 변환효과만을 추가)의 제작 방법에 대해 설명한다.

3. 변환 필터의 제작

변환 필터는 미디어 입력을 받게 되고, 그 입력을 변형시킨다 변환 필터를 제작하기 위해서는 CTransformFilter, CTransInPlaceFilter, CVideoTransform 또는 좀더 일반적인 CBaseFilter로부터 새로운 필터 클래스를 유도하여야 한다 본 논문에서는 CTransformFilter로부터 필터를 유도하고, CTransformFilter의 멤버함수(member function)인 CheckInputType과 Transform을 오버라이딩(overriding)해서 Clip이라는 특수효과를 만드는 방법을 소개한다 Clip 필터는 영상의 상, 하, 좌, 우를 자르고 원하는 색으로 채워 넣는 효과를 가진다

3.1 CheckInputType 함수

주어진 입력이 필터에 적당한 것인지 아닌지 결정해 주기 위해 오버라이딩 된 CheckInputType 함수는 아래와 같다

```
HRESULT CClipFilter CheckInputType(const CMediaType *pmt)
{
    if (pmt->majortype != MEDIATYPE_Video) {
        return S_FALSE;
    } else {
        return S_OK;
    }
}
```

3.2 Transform 함수

Transform 함수에서는 영상변환을 위한 코드를 구현하면 된다. 본 논문에서는 여러 가지 디지털 영상 처리 기법을[3][4] 적용해 다양한 필터를 개발하였다

Clip 필터의 경우 Copy 함수에서 입력 인터페이스(pIn)를 출력 인터페이스(pOut)에 복사하고, Transform(pOut) 함수에서 출력 버퍼만을 변경하여 Clip 효과를 주면 된다 오버라이딩 된 Transform 함수는 아래와 같다.

```
HRESULT CClipFilter Transform(IMediaSample *pIn, IMediaSample *pOut)
{
    HRESULT hr = Copy(pIn, pOut); // 미디어 정보와 데이터를 복사
    if (FAILED(hr)) { return hr; }
    return Transform(pOut);
}

HRESULT CClipFilter Transform(IMediaSample *pOut)
{
    AM_MEDIA_TYPE* pType=&m_pInput->CurrentMediaType();
    VIDEOINFOHEADER *pvi =
                    (VIDEOINFOHEADER *) pType->pbFormat;

    BYTE *pDst;                         // 실제 이미지 버퍼의 주소
    RGBTRIPLE *prgb;
    pOut->GetPointer(&pDst);
    int cxImage = pvi->bmiHeader.biWidth;
    int cyImage = pvi->bmiHeader.biHeight

    prgb = (RGBTRIPLE *)pDst;
    for(int y = 0 ; y < cyImage ; y++)
        for(int x = 0 ; x < cxImage ; x++){
            if (y <= bottom || x <= left || y >= top || x >= right){
              prgb->rgbtRed = r; prgb->rgbtGreen = g; prgb->rgbtBlue = b;
            }
          prgb++;
        }
    return NOERROR;
}
```

위의 방법은 DirectShow Documentation에서 제공한 ezrgb24라는 필터에서 사용한 방법인데[2], Copy 함수를 통해서 입력 버퍼를 출력 버퍼에 모두 복사한 다음, 다시 출력 버퍼를 변경하는 방법이기 때문에 출력 버퍼를 두 번 변경하게 되는 부분이 생긴다 이러한 증복되는 복사를 피하기 위해 Copy 함수에서는 입력 인터페이스의 타입만을 복사하고 실제 데이터의 복사는 DoTransform 함수에서 한번만 이루어지게 함으로써, 변환 속도를 향상시킬 수 있다 수정된 함수는 아래와 같다.

```
HRESULT CClipFilter::Transform(IMediaSample *pIn, IMediaSample *pOut)
{
    HRESULT hr = Copy(pIn, pOut); // 실제 데이터는 복사하지 않고,
                                  // 타입만은 복사.
    if (FAILED(hr)) { return hr; }
    return DoTransform(pIn, pOut);
}

HRESULT CClipFilter DoTransform(IMediaSample *pIn, IMediaSample *pOut)
{
    AM_MEDIA_TYPE* pType=&m_pInput->CurrentMediaType();
    VIDEOINFOHEADER *pvi =
```

```
                (VIDEOINFOHEADER *) pType->pbFormat;

    BYTE *pSrc, *pDst;          // 실제 이미지 데이터의 주소
    RGBTRIPLE *prgbSrc, *prgbDst;
    pIn->GetPointer(&pSrc); pOut->GetPointer(&pDst);
    int cxImage = pvi->bmiHeader.biWidth;
    int cyImage = pvi->bmiHeader.biHeight;
    prgbSrc = (RGBTRIPLE *)pSrc;
    prgbDst = (RGBTRIPLE *)pDst;
    for(int y = 0 ; y < cyImage ; y++)
        for(int x = 0   x < cxImage ; x++){
            if (y <= bottom || x <= left || y >= top || x >= right){
              prgb->rgbtRed = r;
              prgb->rgbtGreen = g;
              prgb->rgbtBlue = b;
            }else{
              prgbSrc->rgbtRed = prgbDst->rgbtRed;
              prgbSrc->rgbtGreen = prgbDst->rgbtGreen;
              prgbSrc->rgbtBlue = prgbDst->rgbtBlue;
            prgbSrc++; prgbDst++;
        }
    return NOERROR;
}
```

4. 구현 결과 및 분석

다음 [표 1]은 352×240 크기의 영상 한 장에 대하여, 기존의 알고리즘과 제안된 알고리즘으로 각각 10번씩 수행하였을 때의 시간을 비교하여 보여주고 있다

[표 1] 수행시간 비교 (352×240)

특수효과의 종류	기존 방법 (msec)	수정한 방법 (msec)	차이 (msec)	개선정도 (%)
Clip	1430	1357	73	5
Extract	495	440	55	11
Trace Contour	820	770	50	6
Camera Blur	6860	6808	52	1

540*368 크기의 영상에 대한 수행 결과는 [표 2]와 같다

[표 2] 수행시간 비교 (540*368)

특수효과의 종류	기존 방법 (msec)	수정한 방법 (msec)	차이 (msec)	개선정도 (%)
Clip	3322	3202	120	4
Extract	1210	1100	110	9
Trace Contour	1897	1790	107	6
Camera Blur	16133	16053	80	1

실험 결과를 요약하면, 제안된 방법에 의해서 수행 속도가 약 1% ~ 11% 정도 개선됨을 알 수 있다 또한, 기존의 방법과 본 논문에서 수정한 방법간의 수행 시간 차이는 영상의 크기에 따라 변하고 특수효과의 종류에는 거의 무관함을 볼 수 있다. 즉, 352×240 크기의 영상인 경우 평균 57.5 msec, 540×368인 경우엔 평균 10425 msec 정도 차이가 나타난 것이다. 이는 본 논문이 영상 변환 알고리즘을 수정한 것이 아니고, 영상을 전달하기 위한 방법을 수정한 것이기 때문이다.

Camera Blur 필터의 경우는 다른 필터에 비해 많은 수행 시간이 필요하므로 제안된 방법에 의한 속도 개선 효과가 상대적으로 감소되며, 이 경우엔 더욱 빠른 영상 변환 알고리즘을 개발하는 것과 그 알고리즘을 최적화 하는 것이 중요할 것이다.

지금까지 개발된 필터들은 Camera Blur, Camera View, Clip, Crop, Diffuse, Extract, Gamma Correction, Ghosting, Image Pan, Radial Blur, Solarize, Tint, Trace Countour, Wind 등 약 14가지의 특수 효과가 있다

5. 결론

많은 디지털 장비들이 출시됨에 따라 디지털 영상과 비디오에 대한 처리 기술에 관심이 모아지고 있다. 실제로 디지털 영상 처리를 위한 Adobe Premiere나 Media Studio Pro와 같은 도구가 만들어져 세계적으로 널리 쓰이고 있지만, 국내에는 이러한 기술의 축적이 부족한 상태이다 이러한 흐름과 취지에 따라, 본 논문에서는 디지털 영상 편집에 필요한 특수효과를 개발하기 위해, DirectShow SDK를 사용하여 필터를 제작하는 방법에 대해 설명하였고, 알고리즘에 따른 각 필터의 수행속도 차이를 몇 개의 예를 들어 보여주었다. 지금까지 개발한 필터들은 독자적으로 개발한 것들도 있지만, Adobe Premiere에서 제공하는 필터들을 모방한 것이 많다 앞으로는 이미 개발된 필터들의 알고리즘의 최적화와 독자적인 필터 개발을 위한 연구가 더욱 많이 이루어져야 할 것이다

참고문헌

[1] 김성욱, "Adobe Premiere 4.2", 도서출판 혜지원, 1996

[2] http://www.microsoft.com/directx/dxm/help/ds/default.htm, Microsoft DirectShow SDK Documentaion.

[3] A. K Jain, "Fundamentals of Digital Image Processing", Prentice Hall. 1989

[4] Randy Crane, "A simplified approach to Image Processing", Prentice Hall, 1997.