IM1 프레임워크 상에 MPEG-4 비디오 디코더 통합

민옥기[0] 정영우 이광의 김학영

한국전자통신연구원 인터넷서버연구팀

{ogmin, jungyw, kelee, hykim}@etri.re.kr

An Integration of Mpeg-4 Video Decoder and IM1 Decoder Framework

Okgee Min[0] Youngwoo Jung Kwangeue Lee Hakyoung Kim

Internet Server Research Team, ETRI

요 약

MPEG-4에서는 다양한 객체를 취급하기 위하여 시스템 부분(Part1)이 차지하는 비중이 MPEG-1이나 MPEG-2에서 보다 훨씬 높아졌다. 이러한 MPEG-4의 시스템 부분을 구현한 참조 모델을 IM1이라고 한다. IM1에는 다양한 오디오/비디오(A/V) 객체를 수용하기 위하여 디코더 프레임워크를 마련하고, 어떤 A/V 객체든 이 프레임워크에 맞추어 디코더를 구현하면 IM1 프리젠터에서 플레이가 가능토록 하고 있다. 현재 IM1 버전 3.8에서는 H.263 비디오, G.723 오디오, JPEG 이미지, AAC 오디오를 지원하고 있다. 이 논문에서는 MPEG-4 비디오 디코더를 IM1 디코더 프레임워크에 맞추어 설계, 수정한 내용을 기술하였다.

1. 개요

MPEG-4[7,8]는 MPEG-1,2가 오디오와 비디오만을 포함하는 동영상 처리에 역점을 둔 것과는 달리, 지오메트릭 객체, 오디오/비디오, 이미지 등과 같은 다양한 객체를 기반으로 컨텐츠를 구성할 수 있는 방법을 마련하였다. 이와 함께, 상호 운용성을 제공할 수 있는 메커니즘을 마련하였다. MPEG-4를 구조 상으로 보면 3개의 계층으로 나눌 수 있다. 제일 상위 계층은 MPEG-4의 미디어에 대한 인코딩과 이를 단위 스트림(Elementary Stream)으로 디코딩하는 기술을 포함하는 압축 계층이다. 두 번째 계층인 동기화 계층(Sync Layer)은 MPEG-4 장면을 구성할 단위 스트림들 간의 동기화와 그들 간의 구조를 결정한다. 마지막으로 전달 계층(Delivery Layer)은 전달 방법이나 프로토콜에 독립적으로 MPEG-4 컨텐츠를 투명하게 접근할 수 있도록 하는 계층이다.

MPEG-4 시스템 부분(part 1)[1]은 장면을 기술하기 위한 BIFS(BInary Format for Scene), A/V 객체를 설명하기 위한 OD(Object Descriptor), A/V 객체 등의 동기화 계층 등에 대한 정보를 기술하고 있다. IM1[6]은 MPEG 그룹 내에서 시스템 부분에 대한 참조 소프트웨어이다. IM1은 다양성을 제공하기 위하여 전송에 대한 프레임워크와 디코더에 대한 프레임워크를 제공하고 있다. 전송 프레임워크는 DMIF[3]에 대한 인터페이스를 제공하고 DMIF를 플러그 인할 수 있도록 되어 있으며, 디코더 프레임워크는 다양한 오디오/비디오 스트림에 대한 디코더들을 프레임워크 맞추어 구현하면 IM1에서 해당 스트림을 플레이할 수 있도록 되어 있다. 현재 IM1에는 H.263 비디오, G.723 오디오, AAC 오디오, JPEG 이미지에 대한 디코더들이 포함되어 있으나 정작 MPEG-4 비디오[2]에 대한 디코더는 포함되어 있지 않다. MPEG-4 비디오 그룹에서도 참조 소프트웨어를 제공하고 있는데, 제공되는 참조 소프트웨어 모두가 오프라인으로 MPEG-4 비디오 파일로 인코딩/디코딩 하도록 되어 있어서 IM1에서는 그대로 사용할 수 없다.

MPEG-4에서 정의하는 다양한 객체들과 MPEG-4 비디오를 한 장면으로 표현하기 위해서는 IM1 디코더 프레임워크에 맞게 MPEG-4 비디오 디코더를 구현하여 플러그 인 시켜 주어야 한다. 본 논문에서는 이를 구현한 내용을 중점적으로 기술하였다.

2장에서는 오프라인으로 동작하도록 되어 있는 MPEG-4 비디오 디코더에 관하여 간략히 설명하였다. 3장에서는 IM1에 전체 구성에 대한 설명과 IM1 디코더 프레임워크에 대한 내용을 기술하고, 4장에서는 MPEG-4 비디오 디코더를 IM1 디코더 프레임워크에 맞게 수정 구현한 내용을 설명하였다.

2. Mpeg-4 비디오 디코더

MPEG-4 비디오 파트(Part 2) 에서 제공하고 있는 참조 소프트웨어로는 두 가지가 있다. 하나는 C 로 구현되어 있는 Momusys 에서 만든 것이고, 또 하나는 MicroSoft 사가 만든 C++로 구현되어 있는 버전이다. 본 장에서는 IM1 에 접목시킨 MicroSoft 사에서 만든 비디오 디코더에 대하여 간략히 설명하고자 한다. 사용한 소스 코드는 MicroSoft 의 Video version1-FDIS-V1.0-99.8.12 디코더 소스를 사용하였다. 이 소스코드는 오프라인으로 실행되며 입력으로는 MPEG-4 로 압축된 비디오 파일이, 출력으로는 YUV 포맷의 로우 데이터를 출력한다. 디코더 실행은

```
% decoder [입력화일] [출력화일] [가로] [세로]
```

와 같이 이루어진다. 예를 들어,

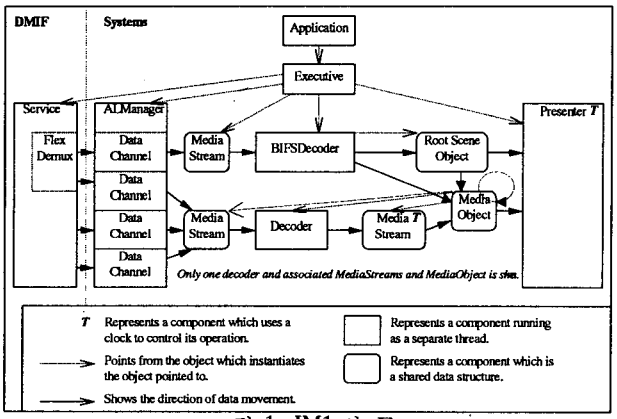
```
% decoder fish.cmp fishout 176 144
```

와 같이 실행하면, 176x144 크기의 fish.cmp 라는 MPEG-4 비디오 파일을 입력으로 디코딩하여 fishout.yuv 파일이 결과 파일로 생성된다.

3. Im1 과 Im1 디코더 프레임워크

3.1 Im1

Im1 은 <그림 1>과 같이 구성되어 있다. Im1 은 크게 3 부분으로 나누어 볼 수 있는데, 장면을 화면에 표현하는 일을 담당하는 프리젠터, 동기화를 맞추기 위한 Sync 계층, 그리고 디코더 부분이다.



<그림 1> IM1 의 구조

프리젠터는 BIFS 디코더가 해석한 장면을 화면 상에 그려주면서 필요한 경우에 A/V 스트림에 대한 버퍼를 참조하면서 화면을 그려 준다. Sync 계층에서는 A/V 및 각 스트림의 접근 단위(Access Unit)로 표시되어 있는 시간을 제어한다. Sync 계층에 있는 각 데이터 채널들은 DMIF 를 통해서 올라온 각각의 ES(Elementary Stream)을 디코더의 입력 버퍼에 써 주는 일을 담당한다. 디코더는 각각 쓰레드로 표현되며, 입력과 출력 부분에 버퍼를 관리하는 객체가 있다. 입력 버퍼는 앞서 말한 바와 같이 데이터 채널이 써 주게 되며, 출력 버퍼로 나간 데이터는 프리젠터에서 사용한다. MPEG-4 비디오 디코더를 IM1 에 플러그 인하기 위해서는 디코더 프레임워크와 디코더의 입출력 버퍼를 핸들링하고 있는 MediaStream 클래스의 운용 방법을 알아야 한다.

3.2 Im1 디코더 프레임워크

Im1 디코더 프레임워크가 되는 클래스의 구조는 다음과 같다.

```
class DecoderImp : public Decoder {
    virtual void Start ();
    virtual void Stop ();
    virtual bool Setup () {return TRUE;}
    virtual bool
 SetFormat(Capabilities,DecoderParams);
    virtual void SetInputStream (int, MediaStream)
    virtual void SetOutputStream (int, MediaStream)
    virtual bool Init () {return TRUE;};
    void DecoderThread ();
    virtual BOOL Decode ()
protected:
    virtual void Run ();
    virtual void Terminate () {}
public:
    DecoderImp ();
    virtual ~DecoderImp ();
};
```

setup()함수는 OD 로부터 얻은 프레임 사이즈 등과 같은 정보를 셋업하며, setFormat() 함수는 디코딩 결과의 Format 등을 결정한다. Run() 함수는 쓰레드로 동작하여 입력 버퍼로부터 프레임 단위로 받아 디코딩 후 출력 버퍼에 써주는 일련의 작업을 반복한다.

4. 설계 및 구현

4.1 설계

Im1 디코더 프레임워크에 맞는 MPEG-4 비디오 디코더의 클래스는 다음과 같이 정의하였다.

```
Class MPEG4Decoder : public DecoderImp {
    setup(SpecificInfo);
    setFormat(Capabilities, DecoderParams)
    GetOptimizedOutputSize()
    Run()
    AnalysisVOHeader() //추가
    DumpFrame(YUV class, YUV stream,...) //추가
    YUVtoRGB() //추가
}
```

AnalysisVOHeader() 함수는 VO(Visual Object) 와 VOL(Video Object Layer) 의 헤더 정보를 분석하여, 프레임 크기, 임의 형상(arbitrary shape) 인지의 여부등에 대한 정보를 얻는다. DumprFrame() 함수는 디코딩된 프레임을 출력 버퍼에 써주는 일을 하며, YUVtoRGB() 함수는 기존의 디코더가 YUV 로 결과를 출력하던 것을 Im1 프레임워크에 맞게 RGB 로 바꾸어 주는 일을 한다. Run() 함수에서는 프레임 단위로 입력 버퍼에 쓰여진 비디오 스트림을 가져 온 후 일련의 디코딩 작업 후, 출력 버퍼에

쓰는 일까지 전체적인 흐름을 제어한다.
MPEG-4 디코더를 Im1 디코더 프레임워크에 플러그 인하기 위해 고려할 사항들을 요약하면 다음과 같다.

- 파일을 오픈하여 배치 처리하던 기존의 디코더의 입출력을 버퍼 입출력으로 변경해야 한다.
- VOP(Video Object Plane) 단위로 출력되는 YUV 데이터를 RGB 형식으로 변경하여 프리젠터에게 넘겨 주어야 한다.
- 프레임의 크기 정보를 헤더에서 추출할 수 있어야 한다.
- 임의형상(arbitrary shape)의 경우는 shape 을 나타내는 코드와 매스킹을 해주어야 한다.

4.2 구현 및 결과

구현 환경은 다음과 같다.

- OS : Windows 98/2000/NT
- Lang. : Visual-C++ 6.0
- Thread : ZTL threads

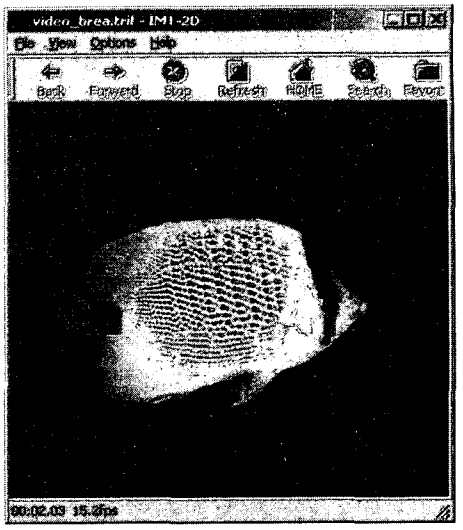
구현의 기본 소스는 다음과 같은 2 개의 소스 코드를 사용하였다.

- IM1-2D Version 3.8 소스 코드
- MS, Video Version1-FDIS-V1.0-99.8.12 디코더

구현은 종료되었으며, 결과 파일은 DLL 파일로 만들어지고, DLL 파일을 레지스트리에 등록하여 사용한다. <그림 2>는 일반적인 사각 형상을 Im1 플레이어에서 플레이하고 있는 모습이다.

<그림 2> MPEG-4 Video 의 플레이 모습

<그림 3>은 임의 형상의 비디오를 매스킹하여 플레이하고 있는 그림이다.



<그림 3> 임의 형상의 플레이 모습

5. 맺음말

본 논문에서는 IM1 에서 지원되고 있지 않은 MPEG-4 비디오 디코더를 IM1 에 플러그 인하기 위한 방법과 구현 결과를 기술하였다. IM1 디코더 프레임워크를 사용하여 구현된 MPEG-4 비디오 디코더는 MicroSoft 사의 참조 소프트웨어를 사용, 수정하여 구현 완료하였으며, 현재 사용되고 있다.
아직까지 미비한 점으로 MPEG-4 비디오는 다양한 클라이언트를 위하여 확장성(scalability)을 제공하고 있는데, 이 경우에는 입력 버퍼를 두개 필요로 하는데, IM1 프레임워크의 경우는 디코더의 입력 버퍼를 하나만 쓰도록 되어 있어서 IM1 프레임워크 자체를 바꿔야 하는 문제점을 가지고 있다.

6. 참고 문헌

[1] ISO/IEC FDIS 14496-1, Information technology Generic Coding of Audio-Visual Object Part 1: Systems, ISO/IEC JTC1/SC29/WG11, 1998.5
[2] ISO/IEC FDIS 14496-2, Information technology Generic Coding of Audio-Visual Object Part 2: Visual, ISO/IEC JTC1/SC29/WG11, 1998,10
[3] ISO/IEC FDIS 14496-6, Information technology Generic Coding of Audio-Visual Object Part 6: Delivery Multimedia Integration Framework, ISO/IEC JTC1/SC29/WG11, 1998.5
[4] Erich Gamma, et al, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-wesly, 1995
[5] Bjarne Stroustrup, The C++ Programming Language, Second Ed., AT&T, 1991
[6] Zvi Lifshitz, "APIs for Systems Software Implementation," AHG on MPEG-4 Systems Software Implementation, 1997.11
[7] 김종옥, *MPEG-4 의 세계*, 영풍문고, 1999
[8] MPEG HomePage, http://drogo.cselt.it/mpeg/