멀티미디어 데이터의 Flickering에 관한 연구

김인환
선문대학교 전자계산학과
e-mail:hdcc6th@hanmail.net

A Study on Flickering of Multimedia Data

In-Hwan Kim
Dept of Computer Science, SunMoon University

요약

컴퓨터에 있어서 멀티미디어는 발전가능성이 풍부한 분야이다. 이러한 멀티미디어는 하드웨어 및 소프트웨어, 네트워크 환경의 발달로 인하여 더욱 더 크고 많은 데이터가 전송될 것이다. 이에 더 좋은 데이터를 더 빠르고 손실 없이 전달하는 것은 중요한 문제이다. 이에 멀티미디어 데이터의 전송 및 재생에 있어서 하나의 문제로 떠오를 수 있는 깜빡임 현상(Flickering)에 대해서 연구하고자 한다.

1. 서론

인터넷은 전 세계를 24 시간 언제, 어디서나(장소 제한 없이) 모두를 연결할 수 있는 가히 상상을 초월하는 폭발력으로 우리에게 다가왔다. 그야말로 세계의 정보와 경제 생활의 모든 것이 인터넷이라는 사이버 공간에서 새롭게 창조되고 빠른 속도로 변화되어 가고 있다. 컴퓨터를 통한 멀티미디어 처리기술과 고속 네트워크의 발달로 컴퓨터 통신망을 이용한 다양한 분산 멀티미디어 응용들 즉, 멀티미디어 원격강의 시스템, 멀티미디어 화상회의 시스템, 멀티미디어 정보검색 시스템, 인터넷 방송 등에 대한 많은 연구가 진행되고 있다. 이러한 멀티미디어 응용들을 구현하기 위해서는 해결되어야 하는 여러 가지 문제들이 있는데 그 중 하나가 멀티미디어 데이터의 구현 기술이다.

멀티미디어 데이터 중에서 동영상에 대한 비중이 점차 높아지고 있는데, 동영상을 전송받아서 재생하고, 필요에 따라서는 저장을 하고, 정지시켜 출력을 하는 등 동영상의 제어 및 활용 필요성이 생기게 된다. 동영상의 재생 중간에 잠시 정지시켜 재생을 멈추거나 정지화상이 필요할 경우 화면이 깜빡이는 Flickering 현상이 있을 수 있다. 이는 많은 양의 데이터를 짧은 시간에 화면에 표현하는 과정에서 생기게 되는데, 매끄러운 진행이나 원하는 화면에서 정지하고자 할 때 문제가 생길 수 있다.

그래서, 본 논문에서는 이러한 Flickering 현상을 줄여 동영상의 재생 및 제어시 매끄럽고 원활하게 하고자 한다.

본 연구 논문에서 다루고 있는 내용을 구체적으로 나열하면 다음과 같다.

(1) 동영상의 개요 및 처리 과정
(2) 구현 원리
(3) 처리 툴 설계 및 구현
(4) 결론

2. 컴퓨터의 개요 및 처리 과정

컴퓨터 문화의 급속한 신장과 함께 영상 산업도 피할 수 없는 기로에 접어들었다. 기계, 즉 아날로그 방식의 집기들이 컴퓨터의 도움으로 디지털화 되고 자동화되었다. 고품질 유지와 작업 시간의 단축, 편집의 편리함은 곧 사업의 흥망성쇠를 좌우하기에 이르렀다. 또한 가정의 사진 문화를 영상 문화로 탈바꿈시키고 있다.

기존의 보편적인 영상 기록 매체인 필름과 비디오는 현실적인 움직임을 낱장 단위의 이미지 상태로 기록한다. 필름은 광화학적인 방법으로 일반적으로 초당 24프레임의 이미지를 기록하며, 비디오는 전자적인 방법으로 초당 30프레임(유럽에서는 25프레임)의 이미지를 기록하고 재생하여 움직임의 환상을 만들어낸다.

디지털 영상작업에서는 흐름 속에 있는 각각의 이미지들을 전자적 방법을 사용하여 낱장 단위로 기록하고, 각각의 이미지에 낱장 단위로 효과를 가하거나 합성작업을 한 후 다시 이를 묶어 연속적으로 제시함으로써 일반적인 실시간 특수효과에서는 달성할 수 없었던 고도의 합성, 효과 첨가 등의 조작을 수행할 수 있다.
이러한 동영상의 포맷(format)에는 다음과 같은 것들이 있다.

(1) JPEG (Joint Photographic Experts Group) :
사진과 같은 정지화상 정보를 통신하기 위하여 압축하는 기술의 표준이지만 동영상 압축에서도 쓰이기도 한다. 프레임안에서 일정한 계산 기준을 정하여 압축시킨 파일 형식이다.

(2) M-JPEG (Motion JPEG) :
JPEG과 거의 같은 의미로 움직이는 비디오 파일의 압축에서도 쓰인다.

(3) MPEG (Moving Picture Experts Group) :
1988년 설립된 MPEG(엠펙)은 ISO산하의 동영상의 압축기법 표준을 정하는 단체이다. MPEG는 200대 1의 압축률을 이루기 위해 다수의 압축 기법을 사용한다. MPEG는 정지 화상 압축과 프

레임간 압축의 압축 방식을 모두 사용하고 있어, 압축효율이 높으며 프레임을 압축할 때 압축률을 지정할 수가 있어 화질의 상태를 선택할 수 있다는 장점이 있다. MPEG의 또 다른 장점은 플랫폼의 제한을 받지 않는다는 것이다. MPEG는 PC나 워크스테이션, 매킨토시와 같은 모든 기종에서 사용할 수 있도록 만들어졌기 때문에 이식성이 높다. 이식성이 높다는 것은 MPEG 파일을 만들 수 있는 장비와 편집할 수 있는 프로그램이 풍부하다는 뜻도 된다. MPEG 동영상 파일은 mpg 확장자를 가지고 있으며, 비디오 CD에 사용될 때는 dat 확장자를 가진다.

MPEG 이미지 해상도는 각 프레임 내의 픽셀들의 양을 감소시켜 320*240까지 감소되어진다. 그 신호는 휘도와 채도 형식으로 표현되어집니다. 눈이 칼라보다 밝기에 더 민감하므로 다소의 칼라 정보를 무시할 수 있다. 한 프레임에서 이산 코사인 변환이 수행되고, 다음에 그 변환의 높은 빈도 구성 요소의 폐기가 수행된 후 양자화가 수행된다. 결국, 그 결과 정보는 Huffman 코딩을 사용하여 다시 압축되어진다. MPEG는 이렇게 압축되어진 프레임들을 각각 따로 압축하는 것이 아니라, 연계해서 압축하는 방식을 사용한다. 먼저 현재 프레임을 압축하고, 그런 다음 앞으로 실행될 4번째 프레임을 현재 프레임과 비교해서 차이점을 압축 저장한다. 그리고 중간 프레임은 앞 프레임과 뒤 프레임을 모두 비교하고 그 차이점을 저장하는 모션 평가 알고리즘 방식을 사용한다. 따라서 MPEG로 데이터를 저장한 경우 깨끗한 화질을 유지하면서도 압축률을 최대로 높일 수는 있지만, 압축시 여러 화면이 연결되어 있기 때문에 프레임당 편집이 불가능하다다. 배경화면보다는 움직이는 물체에서 그 변화가 크게 나타난다.

① MPEG1:1991년 ISO(국제표준화기구) 11172로 규격화된 영상압축기술이다. 초창기 MPEG-1의 출생목적은 600MB의 저장용량을 가진 CD(Compact Disk)에 약 1시간 분량의 영화(동영상)를 삽입하기 위해 노력하였다. CD-ROM과 같은 디지털 저장매체에 VHS 테이프 수준의 동영상과 음향을 최대 1.5Mbps로 압축·저장할 수 있다. MPEG-1을 자세히 보면 해상도는 SIF(352x240)형식이며 초당 30Frame의 구조를 갖고, 동영상을 약 200:1까지 압축할 수 있다. 이 규격으로 상품화된 것이 비디오와 CD와 CD-I/FMV 이다.

② MPEG2:1994년 ISO 13818로 규격화된 영상압축기술이다. 디지털 TV, 대화형 TV, DVD 등은 높은 화질과 음질을 필요로 하는 분야로 높은 전송속도 처리가 필요한데, 영상 및 음향을 압축하기 위해 MPEG1을 개선한 것이다. 현재 DVD 등의 컴퓨터 멀티미디어 서비스, 직접위성방송·유선방송·고화질 TV 등의 방송서비스, 영화나 광고편집 등에서 널리 쓰인다.

③ MPEG3:MPEG2를 완성한 후 후속작업으로 고화질 TV 품질에 해당하는 고선명도의 화질을 얻기 위해 개발한 영상압축기술이다. 그러나 이후에 MPEG2에 흡수·통합되어 규격으로는 존재하지 않는다.

④ MPEG4:멀티미디어 통신을 전제로 만들고 있는 영상압축기술로 1998년 완성되었다. 낮은 전송률로 동화상을 보내고자 개발된 데이터 압축과 복원 기술에 대한 새로운 표준을 말한다. 매초 64kb, 19.2kb의 저속 전송으로 동화상을 구현할 수 있다. 인터넷 유선망과 이동통신망 등 무선망에서 멀티미디어 통신·화상회의 시스템·컴퓨터·방송·영화·교육·오락·원격감시 등의 분야에서 널리 쓰인다.

(4) QuickTime :

Macintosh의 운영체제에서 제공되는 동영상 규약으로, QuickTime으로 만들 어진 동영상 파일이 MOV 혹은 QT 파일이다. QuickTime 개발사인 Apple사 에서는 재생기인 QuickTime Player를 함께 발표해왔기 때문에 컴퓨터에서도 어렵지 않게 MOV 동영상 파일을 만들거나 볼 수 있다. 동영상 파일 형식 중 가장 먼저 실용화되기 시작한 QuickTime은 Apple가 가진 멀티미디어 관련 기술의 열매라고 할 수 있는 것으로 초기에 Macintosh 용으로만 나왔지만, 그 뛰어난 성능으로 많은 사람들의 관심을 끌자 'QuickTime for Windows'라는 이름으로 Windows용이 나오게 되었다.

QuickTime 규약의 특징은 압축 방식의 다양함에 있다. Apple사는 QuickTime 규약을 설계 할 때 다른 회사들이 마음 놓고 새로운 압축/해제 방식을 추가할 수 있도록 하였다. 이 때문에 같은 MOV 확장자를 가진 QuickTime 동영상 파일이라고 해도 실제로는 다른 압축/해제 방식을 사용하는 파일일 수 있다.

퀵타임은 현재 나와 있는 대부분의 파일 포맷들을 지원하며, 퀵타임 플러그인은 멀티미디어 기능이 접목된 수백만의 웹사이트를 지원하고 있다. 웹에서 제공되는 대부분의 멀티미디어 기능은 브라우저용으로 제공되는 단 하나의 퀵타임 플러그인만 있으면 된다. 퀵타임은 수많은 멀티미디어 기술들을 통합하고 있기 때문에, 당신의 Mac이나 PC에 퀵타임을 설치하면 수많은 플러그인 들을 일일이 설치할 필요없이 이용할 수 있다. 즉 .rm파일과 .asf 파일을 제외한 거의 모든 방식의 미디에서 MP3까지. 3D 파일에서 VR(가상현실)까지 퀵타임은 지원하여 준다.

(5) RM (Real Media) :

인터넷상에서 실시간으로 media 파일을 전송하는 streaming 기술 중의 하나로 빠른 전송률과 고압축률을 특징으로 한다. 원래는 사운드만 지원하는 Real Audio와 동영상만을 지원하는 Real Video가 따로 있었는데, 최근에는 Real Media라는 형식이 등장해 사운드와 동영상을 모두 표현할 수 있도록 통일되는 추세이다. 이 파일 형식은 인코딩을 할 때 서버의 부하를 최대한 줄이면서 빠른 속도로 인코딩 할 수 있는 구조로 개발되었기 때문에 다른 파일 형식에 비해서는 상대적으로 음질과 화질이 많이 떨어진다. 예를 들어 오디오 RA파일은 압축률은 뛰어나지만 음질이 MP3나 SWA에 비해 떨어지는 단점이 있다. 그러므로 본인 생각에는 RM 파일 형식은 전송 기술이 아직 덜 발달된 현시점에서 과도기적인 형태로 느린 모뎀 환경에서도 실행할 수 있는 형태라는 이유로 인기를 누리고 있지 않은가 생각한다. 그러나 폭 넓은 사용자를 확보하고 있다는 대중성을 토대로 최고의 인기를 구가하고 있으며 1999년에는 G2라 불리는 새로운 기술을 채택하여 음질과 전송 속도를 대폭 향상하였다. 나날이 발전이 기대될 뿐 아니라 streaming 기술에서 당분간 최고의 우위를 점할 것이라 예상된다.

(6) FLC/FLI :

애니메이션 파일에서 공용으로 쓰이는 파일 형식에는 마크로미디어 (Macromedia)의 MMM과 오토데스크(Autodesk)의 FLI, FLC가 있다. FLC는 오토데스크의 애니메이터 프로(Animator Pro) 프로그램의 2차원 애니메이션 파일 포맷이다. 256색 이하의 팔레트를 가지고 있고 매우 우수한 색 재현력을 가지고 있기는 하지만 압축을 거의 하지 않아 용량이 매우 크다.

(7) ASF (Active Stream Format) :

오디오, 비디오, 슬라이드 쇼, 그리고 동기화된 이벤트 등을 지원하는 마이크로 소프트의 스트리밍 미디어 형식의 멀티미디어 컨텐트의 송수신 데이터 포맷이다. 인터넷을 통해 오디오, 비디오 및 생방송을 수신하는 유틸리티인 마이크로소프트의 NetShow에서 사용된다. 동영상 데이터 등을 분할하고, 그것을 포함한 패킷의 사양을 규정하고 있다고 생각하면 된다. 하지만 ASF는 동영상 압축 등의 포맷을 정한 것이 아니라 AVI나 MOV (QuickTime),

MPG(MPEG)라는 데이터를 주고받기 위한 구조이다. 리얼비디오도 ASF에 포함된 형태로 송수신되는 것이다. 여기에는 이 파일 형식과 관련된 두 가지 파일 형태가 있다. 확장자가 .asx인 파일은 웹브라우저에게 윈도우 미디어 플레이어를 호출하고, 스트리밍 콘텐츠가 담겨있는 .asf 파일을 로드하도록 신호를 보내는데 사용된다.

(8) AVI (Audio Video Interleaved) :

Microsoft Video For Windows applications을 위한 industry용 standard file format. Apple사가 Macintosh에서 실행되는 동영상에 대한 획기적인 제안인 QuickTime을 발표하자 뒤늦게 Microsoft사에서 Window용으로 내놓은 것이 바로 VFW (Video For Windows)이다. 이때 사용된 파일 형식이 AVI이다. Macintosh용 멀티미디어 프로그램들이 QuickTime을 지원하는 것처럼 Windows용 멀티미디어 프로그램들은 대부분 AVI를 기본으로 지원하는 등 특정한 운영체제와 밀접한 관계를 가지고 있다는 것은 장점인 동시에 단점이기도 하다. 운영체제와 아무런 관련이 없는 MPEG처럼 동영상 파일 형식의 표준의 자리에 올라서지는 못했다.

여기서 참고로 Streaming에 대해서 알아본다면 Streaming은 인터넷 방송 등에서 많이 사용되는 기술로 전송되는 데이터를 마치 끊임없고 지속적으로 물이 흐르는 것처럼 처리할 수 있는 기술을 의미한다. 스트리밍 기술은 인터넷의 성장과 함께 더욱더 중요해지고 있는데, 그 이유는 대부분의 사용자들이 대용량 멀티미디어 파일들을 즉시 다운로드할 만큼 빠른 접속회선을 가지고 있지 못하기 때문이다. 스트리밍 기술을 이용하면, 파일이 모두 전송되기 전에라도 클라이언트 브라우저 또는 플러그인이 데이터의 표현을 시작할 수 있다. 스트리밍이 동작하려면, 데이터를 수신하고 있는 클라이언트 측은 데이터를 모으고, 그 데이터를 처리하여 사운드나 그림으로 변환해주는 응용프로그램에 마치 끊임없는 흐름처럼 보내줄 수 있어야 한다. 이것은 스트리밍 클라이언트가 필요이상으로 더 빠르게 데이터를 수신한다면, 여분의 데이터를 버퍼에 저장할 필요가 있다는 것을 의미한다. 그러나 만약 데이터가 충분히 빠르게 들어오지 못하면, 데이터의 표현은 매끄럽지 못하게 된다.

위와 같은 포맷의 동영상은 압축의 과정을 거쳐 전송되게 되는데, 동영상은 정지영상의 반대 개념으로 궁극적으로는 정지된 각 비트맵 이미지들의 모임이다. 그러나 단순히 동영상을 이렇게 정의하는 데는 문제가 있다. 동영상이 시간적 변화가 있는 정지된 여러 이미지들을 필요에 따라 10, 15, 30을 1초에 모아서 보여주는 것은 사실이지만 이런식의 개념으로 동영상을 만든다면 어마어마한 용량을 감당하기 어렵다. 예를 들어 320x240 해상도의 JPEG 파일의 크기를 50KB라고 했을때 15FPS(초당 프레임수)짜리 동영상을 20초짜리를 만든다면 총 용량은 15MB이다. 만약 동영상을 아무런 압축없이 캡처, 저장, 재생하려면 엄청난 하드디스크, RAM이 필요하고 고속의 CPU가 필요하다. 또 시스템 전체에 엄청난 부하를 가하기 때문에 실제로 사용하기에는 부적합하다.

그래서 모든 동영상에서 가장 중요한 것이 압축기술이다. 동영상을 압축하는 데는 전체 화면을 16x16, 8x8, 6x6 픽셀(Pixel) 크기 순으로 블록화하고 움직임이 있는 부위만을 모자이크 모양으로 부호화하는 방법이 쓰이는데 다행히도 사람의 시각은 색의 변화보다는 명암차이에 민감하게 반응하기 때문에 이 원리를 이용하면 압축 후에 압축된 블록의 이질감을 줄일수 있다. 또 일정시간 동안 변화가 없는 부분은 같은 정보를 계속 이용하고 변화가 있는 부위만을 새로운 정보로 대체하여 꼭 필요한 정보만 표시하는 것도 압축과정의 중요한 부분이다. 하지만 이론상 그렇다는 것이며 실제로는 캡처나 압축시 여러 요인에 영향을 받고 움직임이 없는 화면 이라도 화면의 노이즈등 사소한 것에도 영향을 받는다.

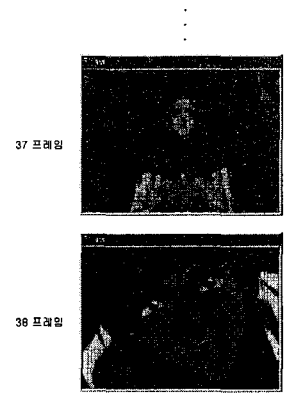
일반적으로 코덱은 비디오 파일을 압축하는 방식을 말하는데, 압축코덱들은 어떠한 품질로 어떠한 압축률을 적용하느냐에 따라 화질이 달라진다. 그러나 지금까지 대부분의 동영상 압축 방식은 화질을 중요시한 것이 아니라 파일의 용량을 중요시했다. 최근 들어 화질을 중요시하는 경우가 빈번해지고 있는데, 이는 저장 매체의 발달과 매우 관계가 깊다.

3. 구현 원리

동영상의 재생시 Flickering 현상이 생기게 되는데, 이는 동영상 재생시 그래픽 장치는 display buffer memory에 저장되어 있는 display list를 읽어 형광물질이 칠해진 화면의 해당 위치에 전자빔을 주사시켜 발생하는 형광현상을 이용하는데 잔상이 화면에 남아 있게 하기 위하여 1/30초에 refresh가 이루어져야 하는데 그래픽 정보가 복잡하면 한번의 refresh에 1/30초 이상 소요됨으로써 화면이 껌벅거리는 문제가 생기는데 이를 flickering 현상이라고 한다.

1초에 30프레임이 재생된다고 가정하고, 프레임의 재생 도중에 사용자가 '정지' 버튼을 눌러 동영상 재생 중간에 잠시 멈추고자 할 경우가 있다. 순간적으로 많은 양의 데이터가 재생되는 중간이라면, Flickering 현상이 생기기 때문에 1초의 시간 안에 재생되는 30프레임들 중에서 영상신호의 변화가 최대한 적은 프레임 사이에서 화면을 정지시켜 Flickering 현상을 방지한다.

다음 그림에서는 31에서 60프레임까지 재생 중간에 '정지' 버튼을 눌렀을 경우 데이터 변화량이 큰 37프레임과 38프레임 사이가 정지되는 부분이 된다[그림 1].



[그림 1] 데이터량의 변화

4. 처리 툴 설계 및 구현

Windows98 운영체제 하의 Pentium 컴퓨터에서 Visual C++로 코딩, 실행하였다. 소스 코드의 일부를 아래에 설명한다.

(1) 동영상 재생을 위한 클래스 작성

```
#ifndef __VIDEO_H__
#define __VIDEO_H__

#include <mmsystem.h>
```

```
class CVideo : public CObject
{
public:
        void Close();           // 오픈된 디바이스를 닫는다.
        BOOL Open(CString strFileName, CWnd* pWnd, CRect rc);
        void SetPrevFrame(UINT pos);  // 이전 프레임으로 이동
        void SetNextFrame(UINT pos);  // 다음 프레임으로 이동
        void EndPos();          // 마지막 프레임으로 이동
        void StartPos();        // 처음 프레임으로 이동
        DWORD GetCurFrame();// 현재 프레임 수
        DWORD GetTotalFrame();  // 총 프레임 수
        BOOL Play();    // 재생
        void Pause();   // 일시 정지
        void DisplayError(DWORD error); // 에러 메시지 출력

        CVideo();
        ~CVideo();
protected:
        MCI_PLAY_PARMS          mciPlay;
        MCI_ANIM_WINDOW_PARMS   mciWindow;
        MCIDEVICEID dwDeviceID;
};

#endif // _VIDEO_H_
```

(2) 프레임의 이동

```
void CVideo::StartPos() //처음 프레임으로 이동
{
        mciSendCommand(dwDeviceID,                  MCI_SEEK,
MCI_SEEK_TO_START,
                (DWORD)(LPVOID)NULL);

}

void CVideo::EndPos() //마지막 프레임으로 이동
{
        mciSendCommand(dwDeviceID,                  MCI_SEEK,
MCI_SEEK_TO_END,
                (DWORD)(LPVOID)NULL);
}
```

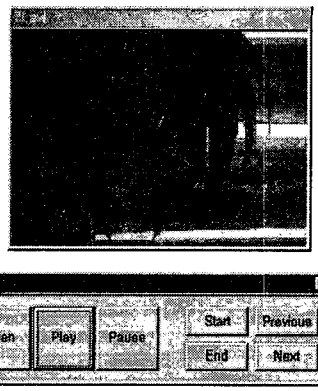
(3) 프레임의 위치

```
DWORD CVideo::GetTotalFrame() // 총 프레임 수 얻음
{
        MCI_STATUS_PARMS    mciStatus;
        mciStatus.dwItem = MCI_STATUS_LENGTH; // 전체 프레임
길이
        mciSendCommand(dwDeviceID,              MCI_STATUS,
MCI_STATUS_ITEM,
                (DWORD)(LPVOID)&mciStatus);
   return mciStatus.dwReturn;
}

DWORD CVideo::GetCurFrame() // 현재 프레임 위치 얻음
{
        MCI_STATUS_PARMS    mciStatus;
        mciStatus.dwItem = MCI_STATUS_POSITION;
        mciSendCommand(dwDeviceID,              MCI_STATUS,
MCI_STATUS_ITEM|MCI_NOTIFY,
                (DWORD)(LPVOID)&mciStatus);
        return mciStatus.dwReturn;
}
```

(4) 프레임의 재생

```
BOOL CVideo::Play()
{
        DWORD result = 0; // 에러 체크
   mciPlay.dwCallback = (DWORD)mciWindow.hWnd;
        result  =  mciSendCommand(dwDeviceID,  MCI_PLAY,
MCI_NOTIFY,
                        (DWORD)(LPVOID)&mciPlay);
        DisplayError(result);

        result = mciSendCommand(dwDeviceID, MCI_REALIZE,
                        MCI_ANIM_REALIZE_NORM, NULL);
        DisplayError(result);
        return TRUE;
}
```

(5) 프로그램 실행 결과



5. 결론

실시간 네트워크를 통한 재생방식이 아닌, 로컬 드라이브의 데이터 재생에 있어서는 Flickering 현상이 쉽게 일어나는 현상은 아니다. 그러나 앞으로 전송되는 데이터가 더욱 커지고 많아지는 등 갑작스런 많은 양의 데이터가 재생될 경우 생기게 되는데, 그런 경우에는 효율적인 해결 방안이 될 수 있겠다. 이를 바탕으로 인터넷 방송, VOD 서비스, 원격 강의 시스템, 병원 의료정보 시스템 등의 분야에 활용할 수 있을 것이다.