시간 연속적인 스크린 이미지와 오디오의 스트리밍을 위한 멀티미디어 시스템

황 기 태†

요 약

본 논문에서는 컴퓨터를 이용한 원격 강의, 원격 학습, 데모 화면 제작 등의 응용들에 필요한 동영상 멀티미디어 시스템을 제안한다. 이들 멀티미디어 응용들을 지원하기 위해서는 실세계 비디오를 다루는 동영상 시스템과는 달리, 시간적으로 변하는 컴퓨터 스크린과 오디오로 구성되는 동영상을 저작하고 재생하는 멀티미디어 시스템이 요구된다. 연속적으로 변하는 컴퓨터 스크린 이미지는 실세계 비디오의 크기와 영상 특성에 있어 차이점을 가지므로 기존의 MPEG 등과 같은 압축 알고리즘이 부적합하며 따라서 본 논문에서는 새로운 압축 알고리즘을 제안하고 멀티미디어 시스템을 설계 구현한 내용을 설명한다. 또한 본 논문에서 제안된 압축 알고리즘의 성능과 시스템 전체의 성능을 평가한 결과를 보인다.

Multimedia System for Streaming Time-Continuous Screen Images and Audio

Kitae Hwang†

ABSTRACT

This paper proposes a motion-video multimedia system needed for computer applications like remote lecturing, distance learning, product demonstrations, and so on. The applications need a multimedia system which can author and play a motion-video that is composed with computer screen images and audio continuously varing as time flows, not with real motion videos. Since the computer screen images are not like the real world video images in several repects, MPEG is not competent as a compression algorithm for computer screen images varing continuously. In this paper a new compression algorithm has been proposed, and a multimedia system that authors and plays a motion-video file which contains computer screen images and audio has been implemented. Also this paper shows the result of performance evaluation of both the compression algorithm and the multimedia system implemented in the paper.

키워드 : 멀티미디어 시스템(multimedia system), 멀티미디어 저작(multimedia authoring), 컴퓨터 스크린 이미지(computer screen image), 압축 알고리즘(compression algorithm)

1. 서 론

동영상 기술이란 멀티미디어를 다루는 기술로서 시간적으로 변하는 사진이나 영상 혹은 컴퓨터로 생성한 동적 이미지등을 오디오나 텍스트 등과 함께 저장하고, 다시 재생하는 기술을 의미한다. 동영상 기술은VOD(video on demand), 원격 교육, 원격 진단, 게임, 오락, 시뮬레이션, 화상 통신 등 다양한 응용을 가능하게 하였으며 점점 더 그 응용 분야를 확대하고 있다[10, 13, 16]. 동영상 기술의 핵심은 각종 미디어의 압축 성능과 미디어들 사이의 동기화 정확성을 이루는데 있다[10, 11]. 압축 성능은 원래의 소스 데이터에 근접한 질을 가지면서 가능한 한 용량이 적은 멀티미디어 데이터로 생성하는 압축 알고리즘에 의해 좌우되며 동기화 기술은 멀티미디어 동영상이 재생될 때 각 미디어 들 사이에 시간적으로 차이가 나지 않도록 멀티미디어 데이터를 저장하고 재생하는 기법에 달려 있다[10, 11, 16].

대부분의 동영상 기술은 비디오 카메라 등에 의해 다루어지는 실세계의 영상에 적용되어 왔다. 그러나 시간적으로 변하는 컴퓨터 스크린에 대해서도 동영상 기술이 적용된다면 효과적일 수 있다. 특정 프로그램의 사용 설명, 프로그램의 실행 과정이나 실행 결과를 동적으로 저장하거나, 소프트웨어 패키지의 사용 방법 등은 보통 문서 형식으로 작성된다. 이러한 문서는 사용자가 보고 학습하기에 많은 시간과 노력이 요구되며 때로는 이해하기 어려운 면이 있다. 이 문서들을 동영상으로 만들어 배포한다면 매우 효과적일 것이다. 또한 컴퓨터를 이용한 원격 강의, 원격 수강, 강의 노트 제작, 데모 화면 제작 등 많은 응용 분야가 존재한다.

※ 본 논문은 2001년도 한성대학교 교내 연구비 지원 과제임.
† 정 회 원 : 한성대학교 컴퓨터공학과 교수
논문접수 : 2001년 9월 28일, 심사완료 : 2002년 1월 25일

한편 시간적으로 변하는 컴퓨터 스크린의 이미지는 실세계 비디오 이미지와는 다른 몇 가지 차이점을 가지고 있다. 첫째 모니터 기술의 발전과 가격 저렴화로 인해 대부분의 응용 소프트웨어는 24비트 칼라와 1024*768이상의 해상도로 동작된다. 24비트 칼라에 1024*768의 해상도를 가지는 한 화면은 약 2.25MB의 데이터를 차지한다. 이는 MPEG1 화면의 최대 크기인 360*240에 비해 9배나 큰 데이터 량이다[17]. 그러므로 컴퓨터 스크린 전체를 동영상으로 구성하는 경우 데이터 량은 매우 커지게 된다. 두 번째 특징은 실세계 영상과는 달리 소프트웨어가 실행되고 있는 컴퓨터 화면이 변하는 속도는 실세계 비디오보다는 매우 작다는 특성을 보인다는 점이다. 실세계 영상은 화면 전반에 걸쳐 적은 양이지만 지속적으로 화면이 변한다. 그러므로 실세계 영상이 자연스러운 연속적인 화면이 되려면 초당 15프레임이상 샘플링되어야 한다[10]. 그러나 응용 프로그램이 실행되고 있는 컴퓨터 화면은 본 논문의 실험 결과 초당 2번 정도의 샘플링이면 충분한 것으로 평가된다. 세 번째 특징은 컴퓨터 화면의 변하는 부분이 국소적이라는 특징이 있다. 예를 들면 워드 프로그램 사용시 주메뉴를 선택하면 해당하는 메뉴 부분이 사각형의 영역에서 변한다. 그리고 워드 입력시 주로 마우스가 있는 곳의 내용만 변한다. 그러나 실세계 화면은 전체 화면이 크게 변하지 않더라도 화면 전반에 걸쳐 화소들이 계속 변하는 등 변화가 화면 전체에 걸쳐 나타난다.

이와 같은 이유로 실세계의 동영상을 압축하는 주된 기법인 MPEG[10, 12, 14, 15, 17]은 시간적으로 변하는 컴퓨터 스크린 이미지를 동영상화하는데 적합하지 않다. 멀티미디어 코딩에 대한 연구가 실세계를 주로 동영상화하는 MPEG 연구에 초점이 맞추어 온 반면, 시간적으로 변하는 컴퓨터 스크린 이미지의 동영상화에 대한 연구는 극히 미미한 편이다[5, 6]. 따라서 본 논문은 시간적으로 변하는 컴퓨터 스크린 이미지와 컴퓨터에서 발생하는 오디오 스트림를 동시에 녹화하여 동영상으로 제작하고 이를 다시 재생하는 멀티미디어 시스템을 구성하는 방법론을 보이며 실제 구현한 시스템을 보인다. 본 논문에서는 이 시스템을 CAM이라고 부른다.

동영상 멀티미디어 데이터가 생성되거나 재생되기 위해서는 두 개의 스트리밍 시스템이 필요하다. 하나는 각 미디어 소스로부터 생성되는 미디어들을 압축하여 동기화가 이루어질 수 있도록 멀티미디어 스트림을 파일로 저장하는 저작 스트림이며, 나머지 하나는 멀티미디어 파일에서 스트림을 읽고 압축을 복원하여 재생 장치에 출력하는 재생 스트리밍이다. 본 논문은 이러한 두 스트리밍 엔진의 설계 방법론을 제안하며 시간적으로 변화는 컴퓨터 스크린 이미지의 압축 알고리즘을 제안한다. 또한 멀티미디어의 동기화를 위해 설계된 파일, 버퍼, 그리고 멀티스레드 구성을 보인다. 본 논문에서 제안되는 압축 방법 및 스트리밍 엔진은 마이크로소프트 사의 DirectShow[4]를 이용하여 개발되었다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문에서 설계 구현한 CAM 시스템의 구성을 소개하고, 3장에서는 CAM 시스템의 스트리밍 엔진을 보이며, 4장에서는 CAM 시스템에서 사용한 압축 알고리즘을 보인다. 5장에서는 CAM 시스템의 압축 알고리즘 및 시스템의 성능을 평가하고 6장에서 결론을 맺는다.

2. CAM 시스템 구성

2.1 시스템 개요

본 논문에서는 제안하는 시스템은 시간적으로 변하는 컴퓨터 스크린 스트림과 오디오 스트림으로 구성되는 동영상 멀티미디어 시스템이다. 사용자가 컴퓨터에서 작업을 하든지 아니면 자동으로 어떤 작업이 실행되고 있든지 간에 시간적으로 변하는 컴퓨터 스크린 이미지를 연속적으로 캡쳐한다. 또한 컴퓨터에 연결된 마이크를 통하여 발생하는 소리나 기타 컴퓨터의 스피커를 통해 나오는 소리, 혹은 사운드 카드를 통해 나오는 소리 등 컴퓨터에서 최종적으로 발생되는 모든 소리를 캡쳐한다. 그리고 이 두 연속적인 미디어 스트림을 압축하고 멀티미디어 파일로 저장한다. 또한 본 시스템은 멀티미디어 파일에 저장된 두 스트림이 시간적인 일치성을 가지고 컴퓨터 스크린과 오디오가 연속적으로 재생되게 한다. 본 논문에서 구현한 스트리밍 시스템은 멀티미디어 파일로의 저작, 저작 중 일시 중단, 재생, 재생 중 중단, 재생 중 일시 중단 등의 많은 기능을 가지지만 본 논문의 범위를 벗어나므로 더 이상 다루지 않는다.

2.2 시스템 설계 고려 사항

2.2.1 시간 연속적인 스트리밍

본 논문에서 제안된 시스템은 컴퓨터 스크린 및 오디오를 시간적으로 연속성을 가지도록 캡쳐하여야 한다. 소스 미디어의 캡쳐시 두 캡쳐 사이의 시간 간격이 너무 길면 재생시 자연스러운 사용자의 스크린 이미지와 소리를 기대할 수 없다. 또한 캡쳐한 시간 간격에 맞추어 스크린 이미지와 오디오를 적절한 시점에 재생하지 못하면 역시 스크린의 움직임 및 소리가 부자연스럽게 된다. 그러나 소스 미디어의 캡쳐의 시간 간격을 너무 좁히면 필요 이상으로 저장 데이터의 크기가 늘어나게 되는 단점이 있다.

2.2.2 스크린과 오디오 스트림의 동기화

멀티미디어를 구성하는 스크린 이미지와 오디오는 시간적인 관점에서 서로 밀접하게 연관되어 있다. 원격 강의의 예를 들면, 강의가 진행되는 스크린과 강사의 목소리가 일치하여야 한다. 예를 들어 워드프로세서 사용법을 설명하는 강의에 있어서 강사가 키보드로 문자 A, B, C 등을 입력하면서 목소리로 "에이", "비", "씨"라고 말할 때 이들이 서로 일치하여 출력되어야 한다. 멀티미디어의 동기화를 위해서는 미디어 소스로부터 동시에 발생한 미디어들이 멀티미디어 파일에 적절한 구조로 기록되어야 하며, 재생시에는 멀

티미디어 파일에서부터 재생 장치로 각 미디어 데이터가 적절한 시점에 전달되도록 제어되어야 한다.

2.2.3 데이터의 소량화

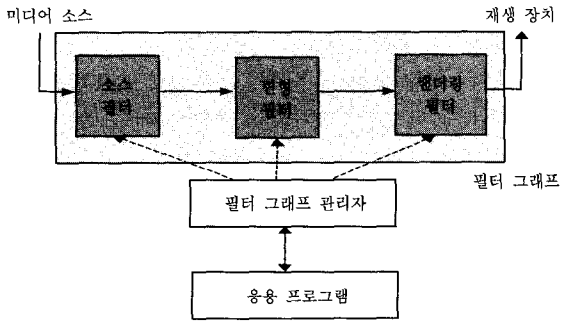
1024*768의 픽셀 해상도에 3바이트로 색상이 표현되는 경우, 압축하지 않을 때 한 프레임의 크기는 약 2.25MB가 된다. 1초에 15 프레임을 가정하면 1초 동안의 동영상은 약 34MB 정도로서 매우 크다. 만일 100분의 1로 압축한다고 하더라도 1분에 20MB나 된다. 데이터의 크기가 클수록 디스크 입출력 시간이 길어지게 되며, 네트워크의 통신량을 늘리게 되는 악영향을 미치게 된다. 가능한한 압축율을 높여 멀티미디어 데이터를 소량화 하여야 한다.

2.2.4 마우스의 원활한 캡쳐

마우스의 움직임을 특별히 다룰 필요가 있다. 컴퓨터를 사용하는 많은 경우에 스크린이 변하지 않은 채 마우스만을 움직이고 있는 경우가 대부분이다. 이때마다 스크린 전체를 캡쳐하는 것은 바람직하지 못하며 마우스의 움직임을 독립적으로 캡쳐하여 효율을 올릴 필요가 있다. 본 논문에서는 스크린 캡쳐보다 마우스 움직임을 빈번히 캡쳐함으로써 재생시 마우스의 움직임이 원활하게 처리한다.

2.2 시스템 구성

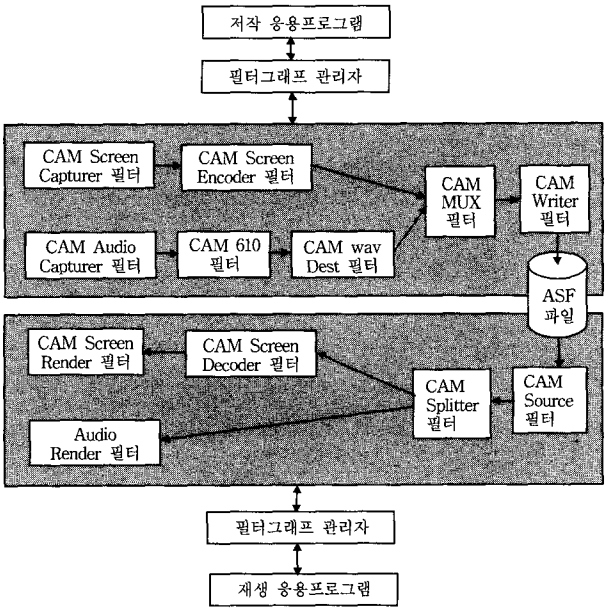
본 논문에서 설계 구현한 CAM 시스템은 윈도우 운영체제를 탑재한 개인용 컴퓨터를 대상으로 하며 마이크로소프트에서 제공하는 툴인 DirectShow[4]를 이용하였다. DirectShow는 멀티미디어 스트림 처리를 위한 기본 골격 소프트웨어를 제공하며 DirectShow를 이용하는 경우 스트리밍 시스템의 전형적인 모델은 (그림 2-1)과 같다. 필터란 DirectShow에서 지원되는 프레임워크에 의해 만들어진 COM[9] 프로그램 모듈이다.



(그림 2-1) DirectShow를 이용한 전형적인 스트리밍 모델

소스 필터는 미디어 소스로부터 적절한 샘플링이나 캡쳐를 통해 데이터를 입력하는 기능을 수행하고, 변형 필터는 데이터를 압축하든지 변환하든지 등 입력된 데이터를 변형하는 기능을 수행하며 렌더 필터는 멀티미디어 데이타를 출력하는 기능을 수행한다. 미디어 데이터는 미디어 소스로부터 소스필터, 변형 필터를 거쳐 렌더링 필터로 이동되며 재생된다. 시스템 개발자는 원하는 기능에 따라 여러 개의 필터를 작성하고 이들을 서로 (그림 2-1)과 유사한 형태로 연결하여 필터그래프를 구성한다. 각 필터를 연결하고 응용프로그램의 요청에 따라 필터를 제어하는 필터그래프 관리자는 DirectShow에 의해 지원된다. (그림 2-1)에서 응용 프로그램이란 사용자 인터페이스를 제공하고 사용자의 조작에 따라 필터 그래프에 재생, 중단, 저작 등의 명령을 전달하는 것으로 대표적인 예로 마이크로소프트사의 윈도우 미디어 플레이어를 들 수 있다.

본 논문에서 구현된 CAM 시스템은 (그림 2-2)와 같은 구성을 가진다. 본 시스템은 총 12개의 필터로 구성되며, 마이크로소프트에서 제공되는 오디오 압축/인코딩 필터인 GSM 610 필터와 Audio Render필터를 제외한 나머지 필터는 실제 구현하였다. 각 필터의 기능은 <표 2-1>과 같다.



(그림 2-2) CAM 시스템 구성

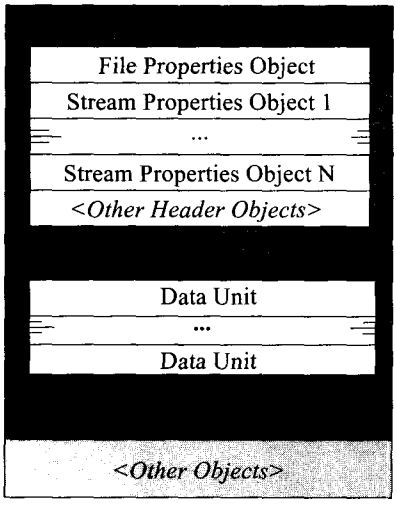
<표 2-1> 필터의 기능 요약

필터이름	기 능
CAM Screen Capturer	스크린을 캡쳐하고 이전 스크린에서 변경된 부분 추출
CAM Screen Encoder	변경된 스크린 이미지를 압축
CAM Mux	오디오와 스크린 스트림을 멀티플렉싱
CAM Writer	압축된 오디오/스크린 스트림을 ASF 포맷으로 저장
CAM Audio Capturer	모든 오디오 소스로부터 샘플링
GSM 610	샘플링된 오디오 스트림을 GSM포맷으로 인코딩
CAM Wav Dest	GSM 포맷의 오디오 스트림을 wav 포맷으로 변환
CAM Source	ASF 파일로부터 스트림 읽기
CAM Splitter	오디오인지 스크린 스트림인지 분별
CAM Screen Decoder	압축된 스크린 스트림을 복원
CAM Screen Render	복원된 스크린 스트림을 화면에 출력
Audio Render	인코딩된 상태의 오디오를 디코딩과 동시에 재생

2.3 CAM 시스템의 멀티미디어 파일 포맷

2.3.1 ASF(Advanced Streaming Format) 파일 포맷

멀티미디어 시스템을 구성하는데 매우 중요한 요소가 멀티미디어 파일 포맷이다. 스트림 데이터들이 서로 동기화가 용이하도록 배치되어야 하며, 순차적인 재생과 임의적인 위치에서의 재생이 용이하도록 설계되어야 한다. ASF[1, 2] 파일 포맷은 널리 사용되는 멀티미디어 파일 포맷으로 많은 멀티미디어 파일들이 ASF 파일 포맷으로 구현되었다. 마이크로소프트사는 최근에 ASF를 발전시킨 윈도우 미디어 파일 포맷[3]을 개발하여 사용하고 있다. 윈도우 미디어 파일 포맷은 ASF 파일 포맷과 거의 동일하나, 미디어 데이터가 반드시 MPEG4로 인코딩 되어야 한다는 특징을 가진다. 서론에서도 밝혔듯이 시간적으로 변하는 스크린 이미지의 특성상 MPEG 코딩이 적합하지 않기 때문에 새로운 압축 방식으로 멀티미디어 파일을 생성하기 위해서는 윈도우 미디어 파일 포맷 대신 ASF파일 포맷을 사용한다. ASF 파일 포맷의 입출력을 위해서는 마이크로소프트사의 ASF PDK[2] 라이브러리를 사용하였다.



(그림 2-3) ASF 파일 포맷

ASF 파일 포맷은 (그림 2-3)과 같은 구조를 가진다. 헤더 부분은 스트림 개수, 총 재생 시간 등 파일 전체에 대한 속성과, 각 스트림에 대해 데이터 율, 스트림 종류 등 스트림 속성, 그리고 특정 위치에서 재생이 가능하도록 마킹한 정보 등을 가진다. 헤더 밑에는 여러 스트림의 데이터들이 데이터 유닛이라는 단위로 저장되며, 파일에 읽고 쓰는 단위는 항상 데이터 유닛 단위이다. 파일의 끝에는 임의의 위치에서 재생이 가능하도록 인덱스 정보를 둘 수 있으며 인덱스는 데이터 유닛에 대한 위치 정보를 가진다.

2.3.2 멀티미디어 파일 설계

본 논문에서는 다음과 같은 원칙으로 멀티미디어 데이터를 저작한다.

(1) 시간적으로 변하는 스크린 스트림과 오디오 스트림의 두 스트림만을 다룬다.

(2) 하나의 데이터 유닛은 오직 한 종류의 스트림 만을 저장한다.

(3) 각 데이터 유닛은 동기화를 위해 재생 시작 시간에 대한 상대적 시간을 가진다.

(4) 스트림에 관계없이 모든 데이터 유닛은 시간 순서로 정렬되어 저장된다.

이 원칙이 위배되면 재생 시 두 스트림 사이에 동기화가 이루어지지 않는 문제가 발생할 수 있다. 오디오와 스크린이 각각 다른 스레드에 의해 처리된다고 가정하면(이것은 매우 일반적인 가정이다), 오디오와 스크린이 동시에 캡쳐되었지만 오디오 스트림은 압축 시간이 스크린에 비해 매우 짧기 때문에 시간적으로 후에 캡쳐된 오디오 데이터가 먼저 캡쳐된 스크린 데이터보다 ASF 파일의 앞부분에 기록되게 되는 현상이 발생한다. 이런식으로 ASF 파일 내에서 모든 데이터 유닛이 시간순으로 정렬되지 않은 현상이 벌어지게 되면, 재생시 나중에 재생되어도 되는 오디어 데이터 유닛를 먼저 처리하느라고 현재 재생되어야 하는 스크린 데이터 유닛을 늦게 처리하는 현상이 일어나게 되고 스크린의 재생이 부자연스럽고 불연속적으로 될 가능성이 있다.

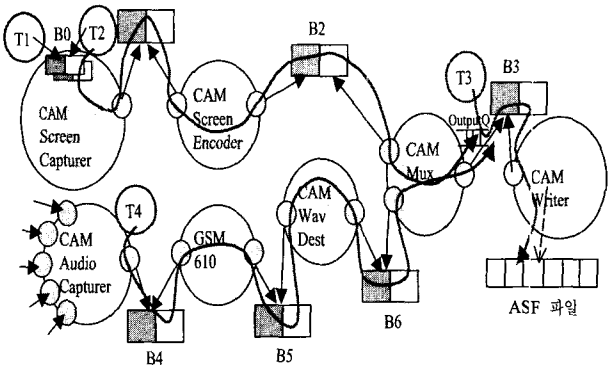
(5) 하나의 오디오 데이터 유닛은 여러 번에 걸쳐 캡쳐한 오디오 정보를 합하여 저장한다.

오디오 샘플의 크기를 사실상 매우 작기 때문에 캡쳐시마다 하나의 데이터 유닛을 두게 되면 빈번한 파일 입출력으로 성능을 나쁘게 만드는 요인이 된다.

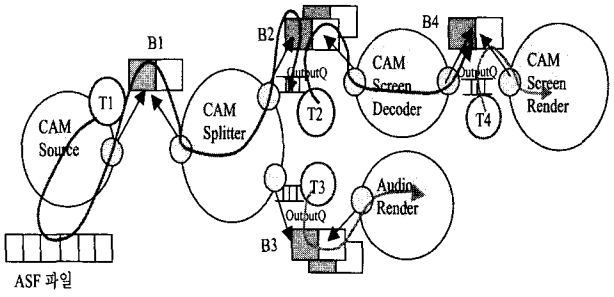
3. 스트리밍 엔진 구현

3.1 엔진 구성

(그림 2-2)에 묘사된 CAM 시스템 구성도에서 필터 그래프를 구성하는 스트리밍 엔진은 구체적으로 (그림 3-1)과 (그림 3-2)와 같다. (그림 3-1), (그림 3-2)에서 T는 스레드를 의미하며, B는 버퍼를 의미한다. 필터와 필터 사이에는 버퍼가 존재하며 한 필터에서 다운 스트림 필터로 데이터를 넘길 때 이용된다.



(그림 3-1) 저작 스트리밍 엔진



(그림 3-2) 재생 스트리밍 엔진

3.2 스레드 구성 및 동기화 처리

(그림 3-1), (그림 3-2)의 엔진들은 멀티스레드로 구성된다. DirectShow의 필터는 수동적인 코드이다. 필터들마다 스레드가 존재하는 것은 아니며 하나의 스레드가 여러 필터의 코드를 실행하기도 한다.

3.2.1 저작 스트리밍을 위한 스레드들

우선 저작 스트리밍 엔진을 구성하는 스레드들을 설명한다. T1, T2 스레드는 CAM Screen Capturer 필터에 존재하는 스레드로서 T1 스레드는 주기적으로 스크린 이미지를 캡쳐하여 버퍼 B0에 저장한다. T2 스레드는 B0에 캡쳐된 스크린 이미지를 이전 스크린 이미지와 비교하여 변경된 부분을 B1 버퍼로 전송하고 CAM Screen Encoder 필터의 코드를 실행하여 B1의 내용을 압축하여 다시 B2 버퍼에 저장한다. 그리고 나서 CAM Mux의 코드를 실행하여 압축된 이미지를 B2에서 B3로 전송하며 OutputQ인 큐에 요청을 삽입하고 리턴한다. T2 스레드가 이전 스크린에 비해 캡쳐한 현재 스크린 이미지의 변경된 부분을 찾는 방법에 대해서는 4장의 압축 알고리즘에서 논한다. T2의 스레드의 작업은 T1에 의해 캡쳐된 스크린이 있는 한 반복적으로 수행한다.

T4 스레드는 CAM Audio Capturer 필터에 존재하는 스레드로서 오디오 소스로부터 오디오를 캡쳐하고 B4 버퍼에 저장하고 다시 GSM610 필터의 코드를 실행하여 압축/인코딩한 후 B5 버퍼에 저장한다. 그리고 나서 CAM Wav Dest 필터의 코드를 실행하여 오디오 스트림을 웨이브(wav) 포맷으로 변환한 후 B6 버퍼에 삽입한다. 다시 CAM Mux 필터의 코드를 실행하여 B3에 전송하며 OutputQ 큐에 요청을 삽입하고 리턴한다. CAM Mux의 코드는 T2와 T4 스레드에 의해 동시에 진입이 가능하므로 임계 영역(critical section)이 설정되었다.

T3 스레드는 압축/인코딩된 스크린 이미지와 오디오 스트림을 ASF 파일 포맷으로 변환하여 저장하는 일을 수행한다. T3 스레드는 CAM Mux필터의 내부에 구현된 스레드로서 OutputQ 큐에 요청이 있으면 깨어나서 CAM Writer 필터의 코드를 실행하여 B3 버퍼의 내용을 ASF 포맷으로 변환하고 이를 파일에 저장한다.

3.2.2 재생 스트리밍을 위한 스레드들

ASF 파일의 재생을 위해서 재생 스트리밍 엔진은 4개의 스레드로 구성된다. T1 스레드는 ASF 파일로부터 스트림 데이터를 읽어 B1 버퍼에 계속적으로 공급하는 일을 실행한다. B1 버퍼가 비어있는 한 이 작업은 반복적으로 계속된다. T1 스레드는 일단 하나의 스트림 데이터 유닛을 ASF 파일로부터 버퍼 B1으로 읽어들이고 CAM Splitter 필터의 코드를 실행하여 어떤 스트림인지 판별하고 스크린 스트림이면 B2 버퍼에, 오디오 스트림이면 B3 버퍼에 삽입한다. 그리고 나서 요청을 두 OutputQ 큐 중의 하나에 삽입한다.

T2 스레드는 자신의 OutputQ 큐에 요청이 존재한다면 B2 버퍼로부터 스트림 데이터를 읽고 압축 복원을 실행하기 위해 CAM Screen Decoder 필터의 코드를 실행한다. 압축 복원된 스크린 이미지를 B4에 삽입하고 역시 요청을 해당하는 OutputQ 큐에 삽입한다.

T3 스레드는 자신의 OutputQ큐에 요청이 존재하는 경우에 동작하며 B3 버퍼로부터 오디오 데이터를 디코딩하여 재생하기 위해 Audio Render 필터의 코드를 실행한다. T2와 T3 스레드는 CAM Splitter 필터에 의해 생성되며 자신의 OutputQ 큐에 요청이 존재하는 경우 반복적으로 일을 수행하며 큐에 요청이 없으면 잠을 잔다.

T4 스레드는 자신의 OutputQ에 요청이 존재하면 동작하며, B4 버퍼로부터 압축복원된 스크린 이미지를 스크린에 출력하기 위해 CAM Screen Render 필터의 코드를 실행한다. 압축 복원된 스크린 이미지는 현재 스크린에 출력되어 있는 이미지에서 변경된 부분만을 가지므로 T4 스레드는 B4 버퍼에 있는 스크린 이미지의 재생 시작 시간이 될 때까지 기다린 후 스크린에 출력한다.

3.2.3 멀티스레드 설계 구현

멀티스레드를 설계하는 기본 원칙은 각 스트림의 동기화와 처리 시간의 최소화에 있다.

다음은 저작 스크리밍을 위한 멀티스레드 설계의 주요 내용을 설명한다.

(1) 주기적인 스크린의 캡처를 원활히 하기 위해 스크린 캡처와 압축은 분리되어 처리되어야 한다.

스크린을 캡처하는 작업은 주기적으로 이루어져야 하는 작업이며 압축은 스크린 데이터의 내용에 따라 처리 시간에 차이가 난다. 그러므로 이 두 작업이 하나의 스레드로 이루어지면 캡처를 주기적으로 할 수 없게 되는 문제가 발생한다.

(2) 오디오 캡처 및 인코딩은 하나의 스레드로 처리 가능하다.

오디오 인코딩 처리에 따른 시간은 매우 작으며 wav 형식으로 변환하는 작업 역시 매우 짧은 시간밖에 요하지 않기 때문에 하나의 스레드로 처리하여도 문제가 없다. 인코딩의 과정동안 기본적으로 윈도우에 설치된 오디오 드라이

버는 발생하는 오디오 데이터를 일정량 저장하기 때문에 오디오 데이터를 잃어버릴 염려가 없다. 그러므로 T4 스레드 하나로 오디오 캡쳐 및 인코딩, wav 포맷으로의 변환 등의 작업을 모두 실행하도록 설계한다.

(3) 디스크 입출력을 수행하는 작업에 하나의 스레드를 따로 할당한다.

T3 스레드는 스트림 데이터를 ASF 파일 포맷으로 변환하고 디스크 쓰기를 실행한다. 디스크 쓰기는 운영체제의 의해 물리적으로 처리되며 디스크에 대한 작업 부하, 디스크나 파일 시스템의 상태, 버퍼 캐쉬의 상태에 따라 쓰기 시간이 달라지므로 디스크 쓰기를 실행하는 스레드는 항상 균일한 처리 시간을 보장받을 수 없으며, 어떤 경우는 많은 시간을 기다리게 된다. 그러므로 이를 별도 스레드로 생성하여 처리하는 것이 효과적이다. 따라서 T3 스레드를 별도로 생성하여 ASF 파일 쓰기를 전담하게 한다.

재생 스트리밍을 위한 멀티스레드 설계의 주요 내용을 설명한다.

(4) ASF 파일로부터 스트림 읽기와 압축복원은 별도의 스레드로 분리한다.

T1 스레드는 ASF 파일로부터 스트림 데이터를 계속적으로 읽어 내어 버퍼 B2에 저장하는 일을 수행한다. 만일 T1 스레드가 스트림 데이터를 파일로부터 읽은 후 스크린 스트림의 압축복원까지 실행한다면 그 동안 오디오 스트림의 공급이 일시 중단되므로 소리가 끊기게 되는 현상이 일어나게 된다. 혹은 오디오와 스크린의 동기가 일치하지 않게 된다. 이를 해결하기 위해 T2와 T3 스레드를 T1 스레드와 별도의 스레드로 구성하였다.

(5) 스크린 스트림의 압축복원과 재생을 서로 분리된 스레드로 처리한다.

각 스트림의 데이터 유닛은 재생 시간에 대한 정보를 가지고 있으며 스트림 데이터가 재생되기 전에 재생 시작 시간까지 기다려야 한다. 만일 T2 스레드가 스트림 데이터의 디코딩 후 렌더링(재생)까지 실행한다면, T2 스레드는 스트림 데이터에 포함한 재생 시작 시간까지 기다리게 되므로 현재 B2 버퍼에서 압축 복원을 기다리는 다음 스트린 이미지를 늦게 복원하게 된다. 즉, B2에 존재하는 다음 스크린 이미지는 자신의 재생 시작시간 전까지 복원이 이루어지지 못하게 되는 문제가 발생하게 된다. 이렇게 되면 재생 시작 시간 이후에 재생되는 스크린 이미지가 존재하게 되며, 이때 오디오 스트림과 동기가 일치하지 않게 되는 문제를 야기시킨다. 그러므로 T4 스레드를 독립적으로 생성하여 렌더링만을 담당하게 한다. 오디오의 경우는 윈도우에서 기본적으로 지원되는 Audio Render 필터에 의해 처리되므로 본 논문에서는 다루지 않는다.

3.3 버퍼 구현

본 논문에서 구현하는 멀티미디어 스트리밍 엔진의 버퍼 사용은 DirectShow에 의해 제약받는다. 초기에 필터들이 로드되어 (그림 3-1), (그림 3-2)와 같이 연결될 때 두 필터 사이의 버퍼의 크기 및 개수는 정적으로 결정된다. <표 3-2>는 본 논문에서 사용하는 버퍼의 크기와 개수를 보이고 있다.

〈표 3-2〉 버퍼

버퍼 이름	크기(바이트)	개수	데이터 타입
저작 B1	1024 * 768 * 3	1	1024*768 해상도에 24비트 칼라 비트맵
저작 B2	1024 * 768 * 3	1	압축된 스크린 스트림(최악 경우 가정)
저작 B3	1024 * 768 * 3	2	압축된 오디오/ 스크린 이미지
저작 B4	알수없음	알수없음	샘플링된 오디오 스트림
저작 B5	알수없음	알수없음	GSM 포맷
저작 B6	B5와동일	B5와동일	wav 포맷
재생 B1	파일에서 가장 큰 데이터 유닛	1	데이터 유닛 포맷
재생 B2	파일에서 가장 큰 스크린 데이터 유닛	2	압축된 형태의 스크린 이미지(변경된 부분)
재생 B3	파일에서 가장 큰 오디오 데이터 유닛	5	wav 포맷의 오디오
재생 B4	1024 * 768 * 3	1	DIB 포맷의 비트맵

저작 스트리밍의 경우, B1과 B2 버퍼는 각각 하나로 충분하다. B1버퍼의 크기는 하나의 스크린 이미지를 모두 저장할 수 있는 정도로 설정된다. 1024*768 해상도에 24비트 칼라를 사용하는 경우 약 3MB(1024*768*3바이트)를 할당하면 된다. B3의 경우 두개의 버퍼가 필요하다. T2나 T4 스레드와 T3 스레드가 동시에 실행되게 하기 위함이다. B3 버퍼의 크기는 초기에 B2와 B6 중 큰 것으로 설정된다. B4와 B5의 경우는 본 논문의 관심사항이 아니다. B6 버퍼는 초기에 필터들이 연결될 때 B5의 크기 및 개수를 알아내어 동일하게 설정한다.

재생 스트리밍의 경우, B1 버퍼의 크기는 ASF 파일의 헤더 부분에 기록된 가장 큰 데이터 유닛의 크기로 설정된다. 이는 T1 스레드가 ASF 파일로부터 데이터 유닛 단위로 읽고 이를 처리하기 때문이다. B2 버퍼의 크기는 ASF 파일 헤더에 기록된 스크린 스트림 중 가장 큰 데이터 유닛의 크기로 설정하며 B3 버퍼의 크기는 역시 ASF 파일 헤더에 기록된 오디오 스트림 중 가장 큰 데이터 유닛의 크기로 설정한다. 이 정보들은 ASF 파일의 헤더에 존재한다. B4 버퍼의 개수는 더블버퍼링을 위해 2개로 설정하나 B3 버퍼의 개수는 5개로 설정하는데 그 이유는 오디오 샘플 개수가 스크린 스트림의 샘플 개수보다 많아 ASF 파일로부터 B4 버퍼로 자주 공급되기 때문에 적당히 5개로 설정하였다. 사실 B4 버퍼의 크기는 몇 KB 정도이므로 메모리를 많이 차지하지는 않는다.

3.4 마우스의 움직임 캡쳐

사용자가 컴퓨터를 사용함에 있어 마우스 사용의 빈도수는 대단히 높다. 스크린에 변화가 없어도 마우스의 움직임은 결국 스크린이 변한 것으로 판단하게 하므로 마우스의 움직임을 따로 추적하여 관리하면 압축되는 스크린 스트림의 양을 줄일 수 있다. 마우스의 위치를 추적하는 스레드를 따로 두고 화면 캡쳐보다 높은 빈도수로 마우스의 위치를 계속 추적한다. 마우스 모양은 응용 프로그램, 클릭, 드래깅 등에 따라 계속 변하기 때문에 마우스의 위치 추적과 마우스의 모양 정보를 함께 알아낸다. 마우스에 관한 이 두 정보는 스크린 스트림화하여 필터 그래프를 따라 ASF 파일로 보내져 스크린 스크림의 일종으로 저장된다. 마우스에 관한 정보는 매우 작기 때문에 ASF 파일의 크기에 거의 영향을 미치지 않는다.

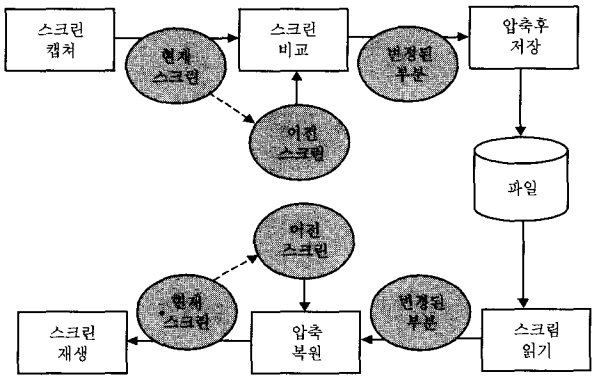
4 스크린 스트림의 압축 및 복원

4.1 기본 아이디어

시간적으로 변하는 스크린 스트림을 저장하고 재생하는데 있어 그 성능은 스크린 이미지의 압축 및 복원 성능에 달려 있다. 압축율을 높여 데이터량을 최소화하면서 재생시 스크린이 연속적으로 재생되도록 적절히 압축되어야 한다. 데이터 량을 최소화하기 위한 가장 간단한 방법은 스크린 스트림의 캡쳐 주기를 크게 설정하는 것이다. 그러나 이는 재생시 스크린이 불연속적으로 재생되는 결과를 낳게 된다. 다른 방법으로는 캡쳐된 이전 스크린과 현재 캡쳐한 스크린의 변화된 부분만을 가려내고 이를 압축하는 방법이다. 이 방법은 MPEG 등에서 사용되는 기본 방법으로서 본 논문은 이 방법으로 기본으로 하여 변형된 방법을 사용한다. 이 후자의 방법은 스크린이 시간에 따라 부분적으로 변할 때 높은 압축율을 보장할 수 있다. 스크린이 시간에 따라 급격히 변한다면 현재 스크린 이미지는 이전 스크린 이미지와 대부분 달라 현재 스크린 이미지의 대부분을 압축하게 되므로 방법 자체가 데이터 량을 줄이는데 도움을 주지 못한다. 그러나 컴퓨터 스크린은 대체로 시간에 따라 국소적으로 변하고 자주 변하지 않는 특성을 가진 것으로 판단되며 후자의 방법이 압축되는 데이터 량을 줄일 수 있을 것으로 예측된다.

4.2 스크린 스트림의 압축/복원 엔진

스크린 스트림의 압축/복원 엔진은 논리적으로 (그림 4-1)과 같이 표현된다. 스크린을 압축하는 엔진은 크게 스크린 캡쳐, 스크린 비교, 압축 후 저장하는 부분으로 구성되며, 현재 캡쳐된 스크린과 이전에 캡쳐된 스크린을 비교하여 변경된 부분만을 알아내어 이를 압축하고 파일에 저장한다. 현재의 스크린은 다시 이전 스크린이 된다. 스크린 캡쳐 부분은 일정한 주기로 스크린을 캡쳐하고 상기와 같은 일련의 작업을 진행한다. 맨 처음 스크린은 비교없이 바로 압축되어 저장된다. 압축을 복원하는 엔진은 파일로부터 변경된 스크린 부분에 대한 압축 데이터를 읽고, 현재 스크린에 변경된 부분만을 갱신한다. 스크린 스트림의 압축 및 복원은 CAM Screen Encoder 필터와 CAM Screen Decoder 필터에서 각각 구현된다.



(그림 4-1) 스크린 스트림의 압축/복원 엔진의 구성도

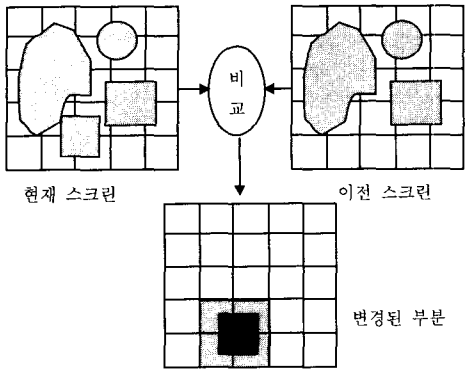
4.2.1 스크린 캡쳐

스크린 캡쳐시 스크린 이미지는 윈도우에서 기본적으로 제공하는 DIB(Device Independent Bitmap)[7, 8]의 포맷을 이용한다. 이 포맷은 그래픽 장치 독립적인 스크린 이미지의 표현 방법으로 모든 스크린 이미지는 DIB 포맷으로 처리된다.

스크린 캡쳐의 주기는 저작 응용프로그램에서 선택할 수 있으며 스크린 스트림의 연속적인 재생 성능과 압축된 데이터의 량의 이해득실(trade-off)에 따라 선택된다. 구현을 통해 실험한 결과 대체로 1초에 두 번 스크린을 캡쳐하는 정도가 적합한 것으로 판단되었다.

4.2.2 압축 알고리즘

스크린 스트림을 저장하기 위해서는 캡쳐된 이전의 스크린과 현재 캡쳐한 스크린을 비교하여 변경된 부분만을 압축 저장한다. 만일 두 스크린을 픽셀대 픽셀로 비교하여 변경된 픽셀만을 저장한다면 픽셀의 위치정보도 함께 저장하여야 하며 항상 모든 픽셀을 비교하여야 하므로 데이터량과 처리시간에 있어 비효율적인 면이 있다. 본 논문에서는 변경된 부분을 찾기 위해서는 DIB 포맷을 캡쳐된 이전 스크린 이미지와 현재 스크린 이미지를 (그림 4-2)와 같이 가로와 세로로 $N \times M$ 등분하여 작은 사각형 영역으로 분할하고 각 사각형 영역 단위로 비교한다. 과거와 현재의 스크린에서 비교되는 두 사각형 영역에 한 픽셀이라도 변경되었다면 그 사각형 영역이 통째로 변경된 것으로 판단한다. 변경되었다고 판단된 사각형 영역을 JPEG[7, 10] 알고리즘을 이용하여 압축한다. JPEG 알고리즘은 인접한 픽셀들 사이의 값이 다르지 않는 경우에 매우 효과적인 것으로 알려져 있으며[10], 따라서 컴퓨터 화면의 사각형 영역을 압축하는데 매우 효과적일 것으로 판단된다.



(그림 4-2) 압축 알고리즘의 적용 예(5×5로 스크린을 등분하여 비교)

물론 변경되었다고 판단된 사각형 영역에는 변경되지 않는 픽셀들도 더불어 존재하게 된다. 만일 스크린을 더 작은 사각형으로 등분하여 비교할수록 실제 변경된 데이터량에 근접하여 갈 것이다. 그러나 너무 세분하게 되면 한 스크린에서 저장하여야 하는 사각형의 수가 많아지게 되고 따라서 JPEG으로 압축하여야 하는 사각형의 개수가 많아지게 되어 오히려 전체적으로 시간과 압축된 데이터 량이 증가하게 되는 결과를 낳게 된다. 본 논문에서는 가장 적절한 스크린 등분비에 대해서는 논하지 않는다. 본 논문에서의 스크린 스트림의 압축 알고리즘은 다음과 같이 기술할 수 있다.

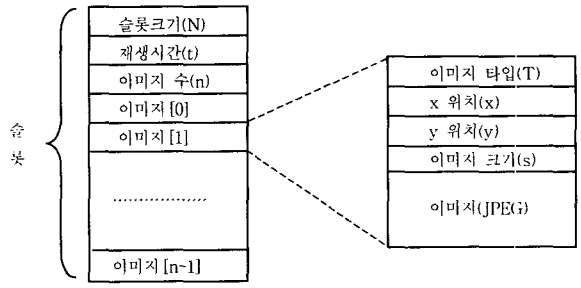
1. 스크린 캡처하여 DIB 포맷으로 변환한다.
2. 캡처한 현재 스크린을 N×M으로 분할하고 이전에 캡처한 스크린과 분할된 영역별로 각각 비교하여 이전 스크린에 비해 변경된 사각형 영역들을 결정한다.
3. 현재 캡처한 스크린 이미지를 이전 스크린 이미지로 지정한다.
4. 각 사각형 영역을 JPEG 알고리즘을 이용하여 압축한다.

(압축 알고리즘)

4.2.3 스크린 스트림 저장 포맷

스크린 스트림의 압축된 최종 결과는 여러 개의JPEG 데이터 모듈 들이다. 한 번의 캡처에 의해 압축된 JPEG 데이터 모듈들은 재생시 동시에 필요하므로 이들은 ASF 파일에 저장될 때 하나의 데이터 유닛 속에 저장되고 함께 관리되어야 한다. 이렇게 동시에 관리되어야 하는 데이터의 집합을 슬롯이라고 명명한다. 슬롯의 구조는 (그림 4-3)과 같다. 슬롯에는 바이트 단위로 표현된 슬롯 전체의 크기(N), 재생 시작 시간(t : msec 단위), 압축된 사각형 영역의 개수(n), 그리고 JPEG으로 압축된 각 사각형 영역의 이미지들이 순차적으로 존재한다. 압축된 사각형 영역은 이미지 타입 정보(T), 사각형 영역의 위치(x, y), 이미지 크기(s), 실제 압축된 JPEG 데이터 등의 정보를 가진다. 이미지 타입은 마우스 이미지와 스크린 이미지로 구분된다. 마우스 이미지는 JPEG으로 압축할 수도 있지만 하지 않는 것이 보다 효과적인 것으로 평가되었다. 그 크기가 기껏 32*32 픽셀 정도이므로 ASF 파일 크기에 거의 영향을 미치지 않으며 JPEG으로 압축하면 압축 및 복원에 따른 실행 시간만 허비하게 되기 때문이다.

스크린 캡쳐 및 변경 영역의 찾기는 CAM Screen Capturer 필터에 의해 구현되며 압축은 CAM Screen Encoder 필터에 의해 구현되고, 이를 하나의 슬롯으로 포맷하는 기능은 CAM Screen Encoder 필터에 의해 구현된다.



(그림 4-3) 압축된 스크린 이미지를 저장하는 슬롯 구조

4.2.4 압축 복원

복원 과정은 보다 단순하다. ASF 파일로부터 스크린 스트림이 읽혀지면 데이터 유닛에서 슬롯을 끊어내고 각 JPEG 데이터들을 분할하여 낸다. 그리고 JPEG 디코딩을 통해 DIB 포맷으로 변환한다. 이 작업은 CAM Screen Decoder 필터에서 구현된다.

4.2.5 스크린 스트림의 레더링

CAM Screen Render로 전달된 스크린 스트림은 현재 스크린에 출력된 이미지에 대해 변한 영역의 이미지만을 가지고 있다. 그러므로 CAM Screen Render의 T4 스레드는 자신의 큐에 도착한 데이터 유닛에 대해 재생 시작 시간이 되기까지 기다린 다음, 슬롯에 존재하는 DIB 비트맵들을 해당하는 스크린 위치에 단순 복사한다.

5. 성능 평가

5.1 압축 알고리즘의 성능 평가

우선 본 논문에서 제안한 압축 알고리즘의 성능 평가의 결과를 보인다. 성능 평가를 위해 1024*768 해상도에 16 비트 칼라로 고정하였으며 1초에 2번 캡처하여 100초 동안 실험하였다. 압축하지 않고 비트맵을 그대로 저장하였을 때 314.5MB 정도되었다. 압축율은 스크린에 출력되는 내용에 따라 달라지게 되므로 3개의 서로 다른 응용 프로그램이 실행되는 상황에서 압축율을 측정하여 평가하였다. 스크린의 변화가 매우 빈번히 일어나는 상황을 평가하기 위해 AVI 동영상이 출력되는 응용을, 화면이 자주 변하지는 않지만 변하는 영역이 큰 상황을 평가하기 위해 FLASH 응용을, 그리고 화면의 변화가 상대적으로 작고 국소적인 경우를 평가하기 위해 한글 워드를 그 대상 응용으로 설정하였다. 그

리고 스크린을 5×5로 분할하였다.

압축율은 압축 없이 저장한 ASF 파일의 경우에 대해 본 논문의 알고리즘을 이용하였을 경우의 ASF 파일의 크기로 계산하였다.

압축율 = (압축없는 스크린 - 압축된 스크린)/
압축없는 스크린×100 (%)

〈표 5-1〉 압축 알고리즘의 성능 평가(압축율 %)

JPEG 압축질	10%	30%	75%
AVI 응용	99.37	99.31	99.17
FLASH 응용	99.62	99.59	99.52
한글워드	99.70	99.69	99.52

<표 5-1>은 세 응용프로그램이 실행되는 동안 본 논문에서 제안하는 압축 알고리즘의 성능인 압축율의 결과를 보이고 있다. 본 실험에서 JPEG 압축질을 변화시키면서 본 논문의 압축 알고리즘의 성능을 측정하였다. JPEG는 손실 압축(loss compression) 알고리즘으로서 압축시 원이미지의 정보를 손실하면서 압축하기 때문에 압축 후 복원하면 본래의 이미지로 완벽히 복원할 수 없는 알고리즘이다. 이때 JPEG 압축질이란 JPEG 알고리즘의 실행시 압축 정도를 말하는 것으로 압축질이 작으면 작을수록 압축시 원이미지에 대한 손실(loss) 정도가 커진다. 즉 압축한 이미지를 화면에 출력하면 원 이미지와의 차이가 크게 나타난다. JPEG의 압축질을 낮추면 당연한 결과로서 본 논문의 압축율이 향상된다. 그러나 압축된 스크린을 복원하여 재생하면 화질이 떨어지게 된다. 본 논문에서는 JPEG 압축질을 75%로 설정하는 것이 화질 및 압축율이 대체로 좋은 것으로 평가되었다.

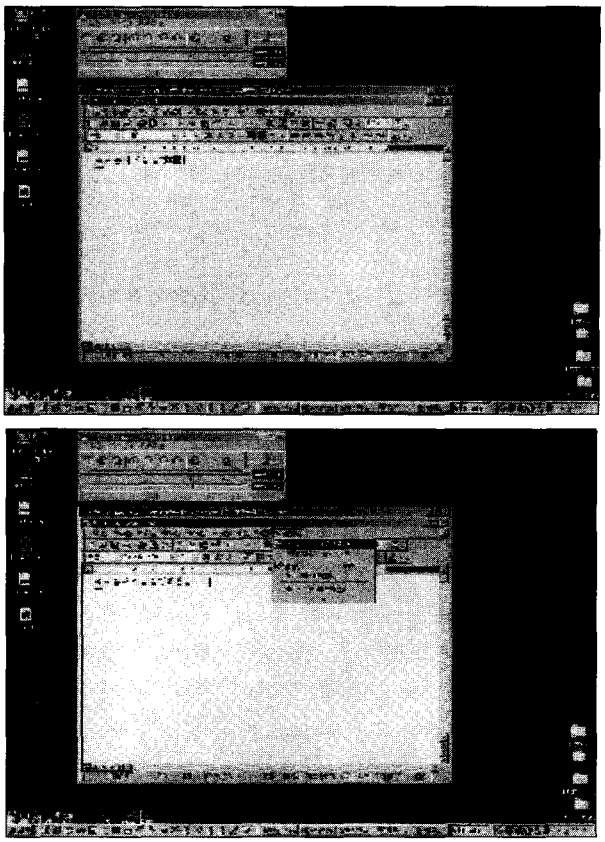
<표 5-1>에서 볼 수 있듯이 본 논문의 알고리즘은 대단히 우수한 압축율을 보이고 있다. 실험에서 AVI는 계속적으로 화면이 변하며, 또한 셋 중 가장 빈번히 화면이 변하므로 압축율이 셋 중에서 가장 낮다. 한글 워드는 사용자의 입력이 상대적으로 느리며 변화하는 화면의 영역이 매우 국소적이므로 압축율이 제일 좋게 평가되었다. FLASH의 경우 화면의 변화되는 영역은 상대적으로 크나 화면이 변화하는 시간 간격이 작지 않다.

5.2 CAM 시스템 성능 평가

5.2.1 CAM 시스템의 실행 예

본 논문에서 설계 구현한 소프트웨어 중 재생 응용프로그램이 실행되는 한 예를 (그림 5-1)에 보였다.

(그림 5-1)의 상단에는 재생 응용프로그램 사용자 인터페이스가 보이며 그 밑에는 저작된 동영상이 재생되고 있다. 저작된 동영상은 강사가 총 46초 동안 워드 프로그램을 사용하면서 강의를 하는 내용이다. 첫 번째 그림은 11초에 재생되는 화면이며 두 번째 그림은 30초에 재생되는 화면이다.



(그림 5-1) CAM 시스템으로 저작된 동영상이 재생되는 예

5.2.2 성능 평가 지표

이 장에서는 본 논문에서 실제 구현한 시스템의 성능을 측정하고 평가한 결과를 보인다. 성능 평가에서 멀티미디어 데이터 율, 오디오와 스크린 이미지의 질, 동기화 질 등의 항목을 평가하여야 하지만, 실제 오디오와 스크린 이미지의 질, 동기화 질은 객관적인 수치화가 어려운 점이 있다. 그러므로 본 논문에서는 멀티미디어 데이터 율로 성능을 평가한다.

5.2.3 작업 부하 및 성능 평가 결과

실제로 본 논문에서 구현한 시스템은 오디오나 스크린 스트림의 압축 및 음질/화질을 선택하도록 구현되었으나 본 실험에서는 압축질만 변화시키면서 실험하였다. 실험에 사

〈표 5-2〉 성능 평가 결과(오디오는 44.1KHz, 16비트, 스테레오)

JPEG압축질	측정시간	작업부하	데이터 량	데이터율 (MB/min)
10%	1분 46초	flash	1.2MB	0.68
	16초	avi	315KB	1.15
	38초	한글워드	462KB	0.71
30%	1분 46초	flash	1.3MB	0.73
	16초	avi	349KB	1.27
	38초	한글워드	368KB	0.57
75%	1분 46초	flash	1.5MB	0.85
	16초	avi	419KB	1.53
	38초	한글워드	478KB	0.74

용된 컴퓨터는 윈도우 NT 운영체제가 탑재된 펜티움 II 300 MHz, 메인 메모리 64MB의 일반적인 PC이다. 작업 부하는 5.1절에서 이용한 3가지의 응용을 그대로 사용하였다.

<표 5-2>는 3가지 작업 부하에 대해 CAM 시스템을 사용하여 저작한 멀티미디어 데이터 량(ASF 파일 크기)을 평가한 결과를 보여준다. 성능 평가의 결과, 본 논문에서 제안 구현한 CAM 시스템은 응용에 따라 다르지만 분당 1MB 내외의 압축 성능을 보인다. 본 논문에서는 구체적으로 밝히지 않지만 Lotus의 Screen CAM을 이용하여 멀티미디어 파일을 저작하는 경우에 분당 10MB 정도가 되는 것으로 측정되었다. 본 시스템은 이에 비해 월등한 성능 향상을 보여 준다. 또한 현재의 실험 결과는 최고의 오디오의 질로 설정하고 실험한 결과이므로 음악을 듣는 경우가 아니고 일반적인 원격 강의가 이루어지는 응용에서는, 모노, 8비트, 그리고 더 낮은 샘플링비를 사용하여도 무방하므로 멀티미디어 파일의 크기가 더 줄어들 것이다. 분당 1MB의 압축 성능은 다시 말해 0.13Mb/sec의 데이터 율로서 10Mb/sec의 LAN 환경에서 실시간으로 저작 및 재생이 가능함을 의미한다. 그리고 본 실험은 1024*768의 전체 화면을 캡쳐한 경우이므로 저작하고자 하는 영역이 스크린의 일부인 경우에는 더 낮은 데이터 율을 얻을 수 있다.

6. 결 론

본 논문에서는 컴퓨터를 이용한 원격 강의, 강의 노트 제작, 데모 화면 제작, 소프트웨어 패키지 사용 방법 등의 응용들에 필요한 동영상 멀티미디어 시스템을 제안하였다. 이들 응용들은 실세계 비디오와는 달리, 시간적으로 변하는 컴퓨터 스크린 스트림과 오디오 스트림으로 구성되는 동영상 시스템을 필요로 한다. 본 논문에서는 이러한 멀티미디어 시스템의 스트리밍 엔진과 스크린 이미지의 압축/복원 알고리즘을 제안하고 시스템을 실제 구현하여 성능을 평가하였다. 본 논문에서 제안된 압축 알고리즘은 압축하지 않는 경우에 비해 90%이상의 압축율을 보이며 약 0.13Mb/sec내외의 데이터 율을 보이는 것으로 평가되어 압축율 및 데이터 율의 성능이 좋은 것으로 평가되며 현실적으로 실용 가능한 것으로 평가된다.

추후 다음과 같은 연구가 계속되어야 할 것으로 본다. 첫째, 압축 알고리즘에서 스크린을 N×M으로 분할할 때 가장 최적의 N, M 값에 대한 연구와 본 논문의 알고리즘을 변형된 휴리스틱 알고리즘을 개발하여야 할 것으로 본다. 두 번째, 원격 강의 등에서 강사의 강의가 수강자의 원격 컴퓨터 상에 실시간으로 재생될 수 있도록 본 논문의 시스템에 실시간 처리를 추가하는 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] Microsoft, An Universal Contaner File Format for Synchronized Media, 1998.

[2] Microsoft, Advanced Streaming Format PDK, 1999.

[3] Microsoft, Windows Media Format SDK, 1999.

[4] Microsoft, Microsoft DirectShow SDK, 1999.

[5] Lotus, "ScreenCAM," http : //www.lotus.co.kr.

[6] 미리온 시스템즈, Wincam 2000, http : //www.wincam.net.

[7] C. Wayne Brown and Barry J. Shepherd, Graphics File Formts : Reference and Guide, Manning, 1995.

[8] John Miano, Compressed Image File Formats, Addison Wesley, 1999.

[9] Jonathan Bates, Creating Lightweight Components with ATL, SAMS.

[10] R. Steinmetz and K. Nahrstedt, Multimedia : Computing, Communications and Applications, Prentice Hall, 1995.

[11] D. P. Anderson and G. Homsy, "A Continuous Media I/O Server and its Synchronization Mechanism," IEEE Computer Special Issue on Multimedia Information System, pp.51-7, Oct. 1991.

[12] P. Panda and M. E. Zarki, "MPEG Coding for Variable Bit Rate Video Transmission," IEEE Computer, Vol.32, No.5, May, 1994.

[13] T. D. C. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand," IEEE Multimedia, Vol.1, No.3, 1994.

[14] D. L. Gall, "MPEG : A Video Compression Standard for Multimedia Applications," Comm. of the ACM, Vol.34, No.4, pp.46-58, 1991.

[15] F. Back, A. T. Lindsay, "Everything you wanted to know about MPEG-7," IEEE Multimedia, pp.65-77, 1999.

[16] D. D. Kandlur, M. S. Chen, and Z. Y. Shae, "Design of a Multimedia Storage Server," IBM Rsearch Report, Jun. 1993.

[17] 정제창, "최신 MPEG", 교보문고, 1995.

황 기 태

e-mail : calafk@hansung.ac.kr

1986년 서울대학교 컴퓨터공학과 학사

1988년 서울대학교 컴퓨터공학과 석사

1994년 서울대학교 컴퓨터공학과 박사

1990년~1993년 비트컴퓨터 기술연구소 실장

1993년 IBM Watson Research Center 방문연구원

2000년~2001년 미국 University of California, Irvine대학의 방문교수

1994년~현재 한성대학교 컴퓨터공학과 교수

관심분야 : 멀티미디어 시스템, 고성능 입출력 시스템, 내장형 실시간 시스템