네트워크 상에서 동영상 검색을 위한 브라우징 시스템의 구현

하명환*, 나윤정, 이상길
한국방송공사 기술연구소

Implementation of Browsing System for Video Retrieval over Network

Myung-Hwan Ha*, Yoon-Jeong Na, Sang-Gil Lee
Technical Research Institute, Korean Broadcasting System(KBS)

* Email : mhha@tri.kbs.co.kr Tel. : +82-2-781-5967 Fax : +82-2-781-5999

요 약

최근에 윈도 NT를 탑재한 고성능 PC의 등장, ATM 카드, 실시간 MPEG 인코더/디코더 보드의 개발 등, 컴퓨터 업계의 눈부신 발전으로 영상 데이터를 네트워크에 분산 저장하고, 브라우저를 사용하여 필요한 자료를 검색, 활용, 편집할 수 있는 멀티미디어 방송 제작 환경 구축이 가능하게 되었다.

본 연구에서는 실시간 영상 전송을 위해 ATM 망을 구성하고, 네트워크상에서 효율적인 검색을 위한 클라이언트/서버 모델을 제시하며, 자동 인덱싱 기능을 가진 동영상 검색 브라우저 시스템을 구현해보고 앞으로의 연구방향을 검토하였다.

모든 사용자 인터페이스는 편리한 윈도 GUI 환경을 사용하므로 사용자는 프로그램을 쉽게 사용할 수 있으며, 프로그램 전체는 C++를 사용하여 클래스 단위로 제작되어 향후 시스템 개발이 용이하게 설계되었다.

I. 서 론

방송사에서는 많은 방송 자료를 테이프에 저장하고 관련된 키워드를 부가하여 테이프 단위로 관리하고 있다. 사용자는 원하는 테이프를 찾기 위해 먼저 키워드 검색을 통해 후보 테이프를 찾은 다음 이를 프레임단위로 검색하여 정확히 원하는 장면이 들어있는 테이프를 찾아서, 프로그램 제작에 활용하고 있다. 만일 원하는 장면이 없으면 위와 같은 과정을 여러 번 반복해야 하며, 테이프의 반복 사용으로 원본의 손상 우려가 있으며, 이미 대출된 테이프는 반납 때까지 사용할 수 없는 불편이 있다.

하지만, 최근에 고성능 PC와 ATM 카드 등, 컴퓨터 업계의 눈부신 발전으로 방대한 양의 영상 자료를 네트워크에 분산 저장하고, 브라우저를 사용하여 필요한 자료를 검색할 수 있는 멀티미디어 데이터베이스 구축이 가능하게 되었다. 이러한 시스템이 구축되면 개인 PC를 이용하여 방송 자료에 쉽게 접근이 가능하며, 프로그램 제작 면에서도 획기적인 발전이 있을 것이라 기대한다. 즉, 기획, 편집, 송출까지 프로그램 제작 작업 전체를 컴퓨터를 이용한 멀티 미디어 기술에 의해 지원 받는 새로운 프로그램 제작 기법이 미래 방송 제작 환경으로 예상된다. (그림 1)

본 논문에서는 ATM 네트워크 상에서 비디오 서버와 클라이언트를 구성하여 효율적으로 동영상을 검색, 활용할 수 있는 동영상 검색 시스템의 설계 및 구현에 관해 다루었다. 제 II장에서 동영상 검색 시스템의 구성과 프로그램 구조에 대해, 제 III장에서는 시스템의 구현 결과에 대해 각각 설명하고 제 IV장에서 결론을 맺고자 한다.

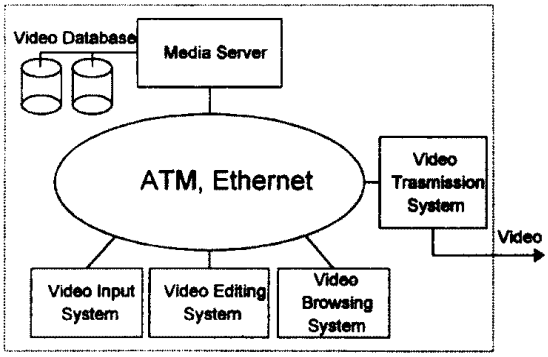


그림 1. 미래 멀티미디어 방송 제작 환경

II. 동영상 검색 시스템

1. 연구 배경 및 동기

동영상과 같은 방대한 양의 멀티미디어 데이터를 효율적으로 관리 및 검색하기 위한 비디오 라이브러리의 구축은 현재 세계적으로 큰 관심사이다. 국내에서도 통상산업부의 주관 하에 방송자료의 데이터베이스화를 위한 프로젝트를 추진한 바 있다. [1]

이러한 비디오 라이브러리는 네트워크, 데이터베이스, 저장 시스템 등 기반 기술 이외에도, 영상 처

리, 영상 및 음성 인식, 인간과 컴퓨터간의 인터페이스 기술 등을 복합시켜야 한다는 점에서 기존의 VOD(Video On Demand) 서비스와 구별된다.

KBS 기술연구소에서는 '96년에 비디오 라이브러리 구축의 핵심기술이 되는 X 윈도 상의 MPEG 동영상 검색 시스템을 연구한 바 있다. 이를 활용하여 압축된 동영상을 빠르게 검색/색인할 수 있어 검색 시스템 개발의 가능성을 제시하였다. 그러나, 이더넷으로 연결된 워크스테이션 환경하에서 개발된 이 시스템은 전송속도가 느리고, MPEG 비디오 스트림만을 처리할 수 있는 문제점이 있었다. [2]

ATM 망과 고성능 PC의 도입은 이 시스템을 보다 발전시킬 수 있는 계기가 되었으며, 윈도 NT와 고속 네트워크에 기초한 검색 시스템 개발이 가능하게 되었다.

2. 시스템 구성

그림 2에 검색 시스템의 기본 구성도를 나타내었다. 모두 윈도 NT를 탑재한 듀얼 펜티엄 프로급의 PC이며, 각각은 PCI타입의 ATM 카드를 통하여 ATM 스위치에 연결된다. 웹 서버의 경우는 인터넷 서비스를 위해 LAN 카드와 ATM 카드를 동시에 장착할 수 있다.

비디오 입력 시스템에서는 아날로그 테이프에 기록된 영상 자료를 실시간 MPEG 인코더/디코더 보드를 거쳐서, 디지털로 압축하여 비디오 서버의 하드디스크에 저장한다. 빠른 검색을 위해서 화질보다는

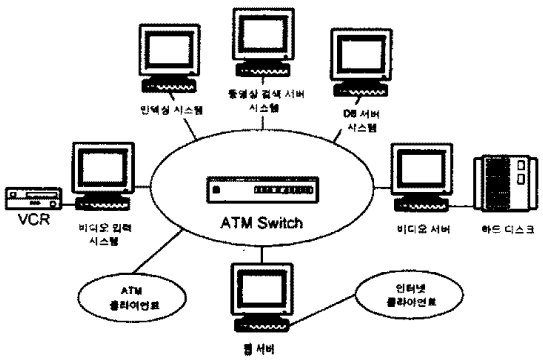


그림 2. 동영상 검색 시스템 구성도

검색 속도를 고려하여 MPEG-1으로 압축된다. 또한 MPEG-1의 경우 소프트웨어만으로 재생이 가능하므로, 클라이언트 PC에 전용 MPEG 디코드 보드를 사용하지 않아도 되고, 웹 브라우저에서도 Plug-In으로 재생이 가능하다. 향후 스튜디오 화질의 영상을 제공하기 위해서는 검색 시스템은 MPEG-2나 M-JPEG 영상 포맷으로 저장되어야 할 것이다.

DB 서버 시스템에는 비디오 서버에 저장된 비디오의 위치 정보와 인덱싱 정보가 저장된 데이터베이스가 구축되어 동영상 검색 서버 시스템의 요청에 따른 결과를 전송한다.

동영상 검색 서버 시스템에서는 실제 ATM 클라이언트의 요청을 받아, 데이터 베이스에 질의한 후 필요한 인덱싱 정보와 동영상 데이터를 비디오 서버로부터 읽어 클라이언트에 전송한다.

인덱싱 시스템에서는 동영상을 재생하면서 동영상을 대표할만한 특성을 추출하여 이 인덱싱 정보만을 데이터베이스에 저장해 둔다.

웹 서버에는 인터넷 클라이언트의 요구를 받는 CGI(Common GateWay Interface) 프로그램이 클라이언트의 요청을 받아, 데이터 베이스를 통해서 인덱싱 정보와 동영상 데이터를 읽어 HTML 형식의 파일로 만들어 클라이언트에 전송한다.

인터넷 클라이언트는 인터넷에 연결된 어떠한 단말기도 가능하며 웹 브라우저를 통해서 원하는 자료를 검색/재생할 수 있다. 동영상의 소프트웨어 재생을 위해서 멀티미디어 데이터 스트림을 제어할 수 있는 ActiveMovie를 사용하였다. [3]

3. 동영상 검색 시스템의 구조

비디오를 촬영하는 기본적인 도구는 카메라로서 여러 개의 카메라 장면들을 편집하여 연속적으로 보여줌으로써 하나의 비디오를 구성하게 된다. 디지털 비디오에서 이러한 하나의 카메라에 의한 연속적인 장면을 샷(shot)이라 표현하고, 샷사이의 장면의 전환이 이루어지는 부분을 컷(cut)이라고 한다. 하나의 샷은 연속적인 비디오 프레임들로 구성되어 있으므로 이러한 샷을 비디오의 최소 단위로 규정할 수 있고, 샷 단위의 검색은 프레임 단위의 검색에 비해 정보의 양을 훨씬 줄일 수 있으므로 효율이 높다고 할 수 있다. 샷 단위의 검색을 위해서는 각 샷을 대표하는 대표 프레임만을 추출해 두는 인덱싱 과정이 필요하게 된다. [4]

동영상 검색 시스템의 핵심은 동영상 파일을 프레임단위로 검색하면서 자동으로 인덱싱 정보를 만드는 인덱싱 시스템이며, 이 정보는 데이터베이스에 저장된다. 본 연구에서는 장면 전환 지점만을 찾아, 대표 프레임을 추출하여 조각그림(thumbnail)으로 만들고, 이때의 프레임 번호와 키워드, 조각 그림을 인덱싱 정보로 사용하였다. 조각 그림은 정지 영상 압축 포맷인 JPEG(Joint Photographic Experts Group)으로 저장하고 프레임 번호와 키워드를 하나의 인덱스 파일로 만들어 하드디스크에 저장하였다.

인덱싱과정이 행해진 후, 클라이언트 쪽에서는 작은 크기의 인덱싱 정보만을 네트워크를 통해 전송받아 비디오를 대표하는 조각그림을 전체적으로 보면서 검색하므로 효율적으로 검색이 가능하다.

그림 3에 동영상 검색 시스템의 전체 구조를 보였

다.

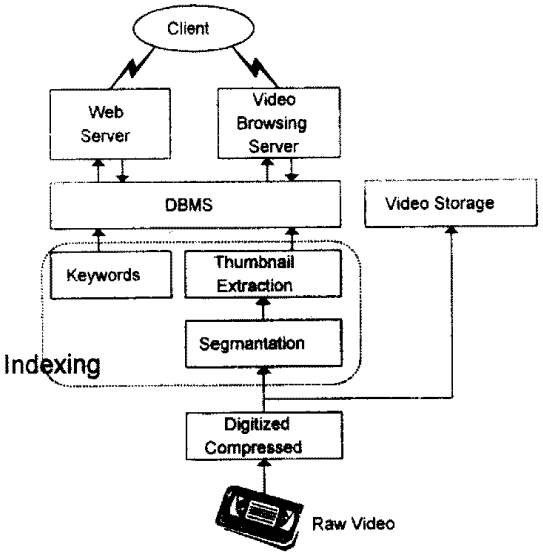


그림 3. 동영상 검색 시스템 전체 구조

3. 서버/클라이언트간 통신

동영상 검색 서버는 클라이언트의 요구에 따라 필요한 정보를 네트워크로 전송하며 서버/클라이언트간의 통신 프로토콜을 그림 4에 설명하였다. 먼저 서버는 클라이언트의 접속 요구에 응답하면서 검색 가능한 동영상 파일 리스트를 전송한다. 클라이언트가 원하는 동영상을 선택하면 그의 인덱싱 정보를 전송한다. 사용자가 전체 비디오의 정상 재생을 원하는 경우 비로소 전체 비디오를 빠른 ATM망을 이용하여 전송하므로 효율적인 네트워크 사용이 가능하다.

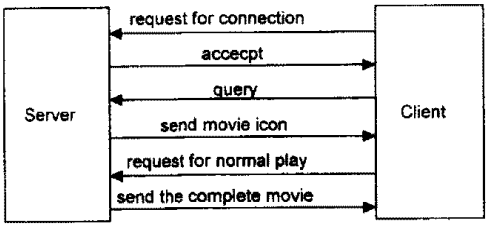


그림 4. 서버/클라이언트간 통신

4. 장면 전환 검출 알고리즘

비디오 인덱스를 위한 첫 단계로 연속되는 비디오 열을 샷 단위로 경계를 구분 지어 인덱스의 단위를 만들어 주는 컷 검출(cut detection) 과정이 필요하다. 컷 검출 과정은 같은 샷 내에서는 프레임간의 유사도가 높다는 이론에 근거한다. 본 논문에서는 인접한 프레임간의 밝기차(luminance difference)와 히스토그램의 차이를 동시에 고려하여 장면 전환 지점을 검출하였다. [5]

4.1 컷 검출

화소 비교법은 연속 프레임에서 대응하는 화소의 세기를 비교하여 얼마나 많은 변화가 일어났는지 측정하는 방식으로 식 (1)에 나타내었다.

$$DP_i = \sum_{k=0}^{M}\sum_{l=0}^{N} | Y_i(k,l) - Y_{i+1}(k,l) | \quad (1)$$

DP_i : luminance difference at the *ith* frame

$Y_i(k,l)$: the luminance value at pixel (k,l)

M : the total number of vertical pixel

N : the total number of horizontal pixel

DP_i가 특정 임계값보다 크면 장면 전환 지점으로 판단하며, 이 방법은 카메라의 움직임에 민감한 단점이 있어, 움직임이 많은 비디오에서는 컷 검출 오류를 일으킨다.

히스토그램 비교법은 화소의 세기(Y 성분)나 색상(Cb, Cr 성분)을 히스토그램으로 표현하여 식 (2)처럼 히스토그램 차이로 두 인접 프레임간의 유사도를 측정한다. 이 방법은 화소비교법에 비해, 프레임내의 공간적인 변화를 무시하므로 카메라나 물체의 움직임에 덜 민감한 장점이 있다. 그러나, 같은 물체를 다른 각도에서 촬영한 영상의 경우처럼 내용은 전혀 다르지만 비슷한 히스토그램을 가진 인접한 영상의 경우에는 장면 검출이 불가능하다.

$$SD_i = \sum_{j=0}^{G} | H_i(j) - H_{i+1}(j) | \quad (2)$$

SD_i : histogram difference at the *ith* frame

G : the number of possible grey or color levels

$H_i(j)$: the histogram value for the *ith* frame

j : one of the G possible levels

본 논문에서 사용한 방법은 먼저 히스토그램 비교법으로 컷 검출을 하였으며 히스토그램 차이가 작은 경우에는 화소 비교법을 한번 더 적용해 히스토그램 비교법의 단점을 보완해 만족한 결과를 얻었다.

4.2 점진적 장면 전환 검출

컷으로 바뀌는 장면 전환의 검출은 비교적 간단하나, 특수효과(디졸브, 페이드 인, 페이드 아웃, 와이프)에 의한 장면 전환의 경우 검출을 위한 다른 알고리즘이 필요하다. 점진적 장면 전환의 경우 프레임간의 차이값이 비교적 작아 단일 임계치만으로는 정확한 검출이 어려운 문제가 있었다. 이중 비교법(twin-comparison method)은 Zhang[5]에 의해 제안된 방법으로 두 개의 임계치를 두어 컷 검출과 점진적 장면 전환 검출을 하였다. 두 개의 임계치는 각각

다음과 같다.

Tb : 컷 검출을 위한 높은 임계값

Ts : 특수 효과 검출을 위한 낮은 임계값

일단 히스토그램 차이값이 Ts보다 크면 점진적인 변화의 잠재적인 시작(Fs)이라고 간주한다. 영상이 점진적으로 변하는 동안 연속되어지는 프레임과 잠재적인 시작점과의 히스토그램 차이값을 계산한다. 일반적으로 이 값은 시간적으로 점점 증가하므로 누적 차이값(accumulated difference)이라 부른다. 만약 계산된 값이 Tb를 초과하면 장면 전환 지점으로 결정한다. 이 값이 Tb를 초과하기 전에 인접한 프레임 간의 차이값이 Ts보다 작아지면 그 잠재적인 시작점을 제거하고 다른 점진적 장면 전환 지점을 찾는다. 그림 5에 일반적인 영상의 경우 히스토그램 차이값과 누적 차이값의 그래프를 나타내었다.

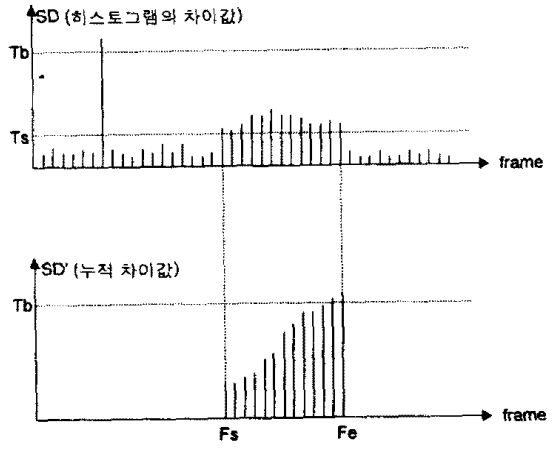


그림 5. 이중비교법 설명도

이중 비교법의 핵심은 두 개의 분명한 임계치 상태가 동시에 만족되어야 한다는 것이다. 이 방법의 단점은 연속적인 차이가 Ts보다 작은 값을 가지는 동안에도 점진적인 장면 전환이 일어날 수 있다는 점이다. 이 문제를 해결하기 위한 방법으로 낮은 차이 값을 가진 연속적인 프레임의 수가 사용자가 허용하는 허용치내에만 속한다면, 잠재적인 시작점을 제거하지 않음으로써 장면 전환 검출 오류를 방지한다.

5. 검색 프로그램 설계 및 구현

모든 프로그램 개발은 Visual C++ 5.0 환경하에서 개발하였으며 동영상 재생을 위해서는 Activemovie SDK를 사용하였다. 사용자 인터페이스는 윈도우 GUI 환경을 사용하므로 쉽게 프로그램을 사용할 수 있으며, 프로그램 전체는 안정된 구조인 C++의 클래스 단위로 제작되어 향후 시스템 개발이 용이하게 설계되었다.

5.1 ActiveMovie

멀티미디어 데이터를 제어하고 처리하는 구조인 ActiveMovie는 사용자가 다양한 방법으로 코딩된 음성 정보와 영상 정보를 실시간으로 재생할 수 있는 기능을 제공한다. 이는 다양한 미디어 형태(e.g. MPEG 오디오, WAV 오디오, MPEG 비디오, AVI 비디오, Apple QuickTime video)를 지원하며 Microsoft에서는 시스템 개발자를 위해서 ActiveMovie가 제공하는 서비스를 구현할 수 있는 SDK를 갖추고 있다. [3][6]

응용 프로그램에서 ActiveMovie를 사용하기 위해서는 그림 6에 보인 것처럼 COM(Component Object Model) 인터페이스, Active X control 또는 MCI(Media Control Interface)를 통하여 접근해야만 한다.

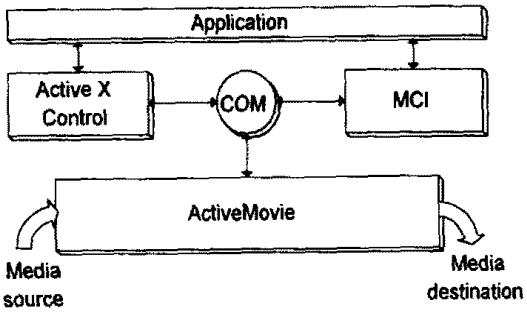


그림 6. ActiveMovie 개요도

본 연구에서는 COM 인터페이스를 사용하며, 동영상을 재생/처리할 수 있는 클래스 CAmovie를 제작하여 프로그램 제작에 활용하였다.

5.2 윈도 그래픽 시스템과 화면 캡쳐

윈도용 프로그램은 그래픽 디스플레이 장치인 화면과 프린터 등의 하드웨어에 직접 접근하지 않는다. 대신에 윈도는 그래픽 및 형식화된 텍스트를 쉽게 나타낼 수 있는 그래픽 인터페이스 프로그래밍 언어인 GDI(Graphic Device Inferface)를 가지고 있어, 디스플레이 하드웨어를 가상화시킨다. 윈도우의 이러한 기능은 두가지 중요한 윈도 구성 요소에 의하여 지원되는데, 하나는 GDI32.DLL이고 다른 하나는 비디오나 프린터 드라이버 파일이다.

CAmovie 클래스에서 VGA에 동영상을 재생할 때, 화면상의 각 픽셀에 대해 RGB값을 얻어 내려면 디스플레이 하드웨어의 비디오램에 저장된 값을 읽어내야 하지만, 윈도 그래픽 시스템의 GDI 객체를 사용하면 RGB값을 쉽게 구할 수있다.

즉, GDI 객체의 일부인 bitmap을 만들고, 화면상

의 특정 영역에서 생성한 bitmap으로 비트 전송 함수를 사용하여 픽셀값을 얻어낸다. 65536 컬러의 경우 1 WORD 길이의 픽셀값이 얻어지며, 내부의 RGB의 비트 할당량은 그림 7과 같다.

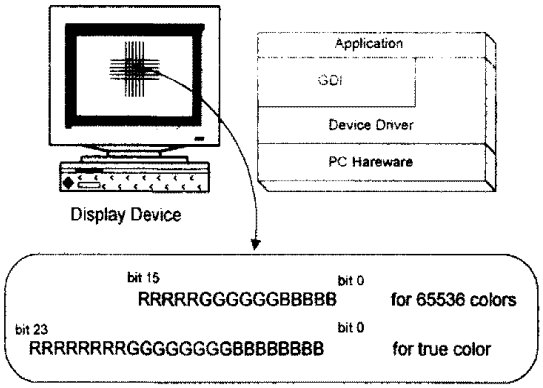


그림 7. 윈도 그래픽 시스템

본 연구에서는 위에서 설명한 윈도 VGA 화면을 이용한 픽셀값 추출 방법을 이용하여 영상의 재생을 위해 어떤 인터페이스를 사용하느냐에 무관한 동영상 검색 프로그램 설계가 가능하였다. 따라서, 초당 24프레임의 재생 능력만 가진 ActiveMovie 인터페이스를 사용하지 않고, 전용 MPEG 디코더 보드와 오버레이 기능을 가진 VGA 카드를 사용한다면 검색 프로그램의 변경없이 실시간 재생이 가능할 것으로 기대한다.

5.3 프로그램의 클래스 구조

이 절에서는 프로그램을 클래스 단위로 설명한다.

CAmovie : 동영상의 정상/빠른/느린 재생 기능, 정지, 프레임 찾기 기능

CImageProcess : bitmap을 RGB 메모리 버퍼에 저장, RGB -> YUV 포맷 변경 기능, 화소/히스토그램 비교법 수행, 장면전환 검출 파라메터 조정 기능

CDocument : 인덱스 파일 저장/로드 기능

CView : 조각 그림 디스플레이 기능, JPEG 저장 기능, 화면 캡쳐 기능, 조각 그림 bitmap 생성 기능

CSocket : 클라이언트 요구 송신 기능, 인덱스 리스트/인덱스 파일/동영상 전송 기능

III. 결과 및 검토

1. 결과

사용한 동영상 화면 크기는 SIF(352x240) 포맷이며, 구현된 검색 프로그램의 사용 예를 그림 8에 나타내었다.

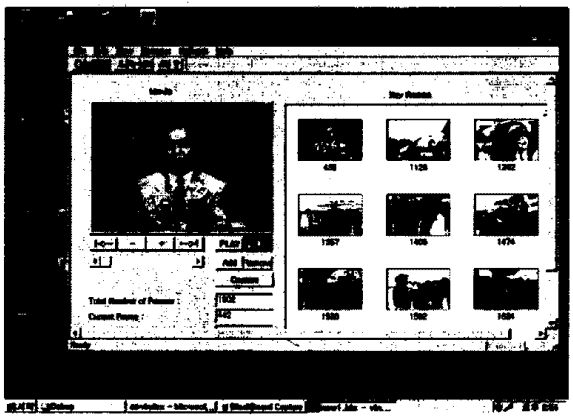


그림 8. 동영상 검색 프로그램 사용 예

사용자가 선택한 MPEG 파일은 자동 인덱싱 과정을 거쳐 오른쪽 표시창에 조각 그림으로 표시되며 불완전한 부분은 수동으로 첨가, 삭제할 수도 있다. 인덱싱이 끝난후에는 원하는 장면을 마우스로 클릭하면서 원하는 지점부터 재생할 수 있다. 생성된 인덱싱 결과가 만족스러우면 다음에 활용할 수 있게 파일로 저장한다. 인덱스 파일은 프레임번호와 조각 그림 이름을 순서대로 저장해 둔다. (그림 9) 뉴스 영상물의 경우 비교적 만족할 만한 결과를 얻었다.

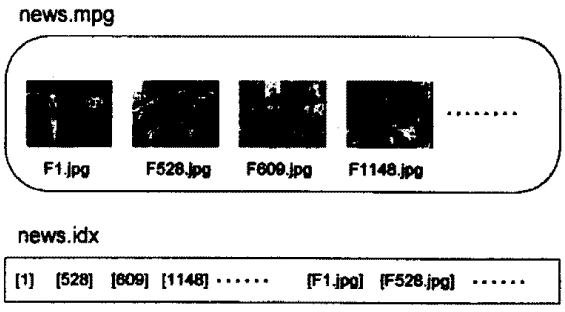
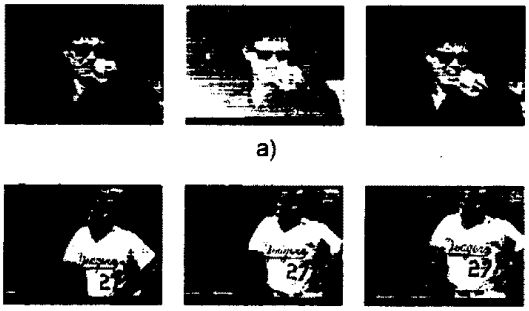


그림 9. 비디오 인덱스 파일

자동 인덱싱 과정에 필요한 파라메터는 뉴스 영상을 기준으로 맞추었으며 같은 파라메터에 대해서 쇼나 스포츠 프로그램 같은 비교적 움직임이 많은 영상에 대해 알고리즘의 성능을 평가하였다. 쇼 프로그램은 현란한 조명 때문에, 그리고 스포츠의 경우 움직임 많은 경우 검출 오류가 발생하였다. (그림 10)



a)

b)

그림 10. 검출 오류의 예
a) 조명효과에 의한 경우
b) 움직임이 많은 경우

또, 그림 11에는 웹 브라우저를 사용한 검색 예를 보였다.

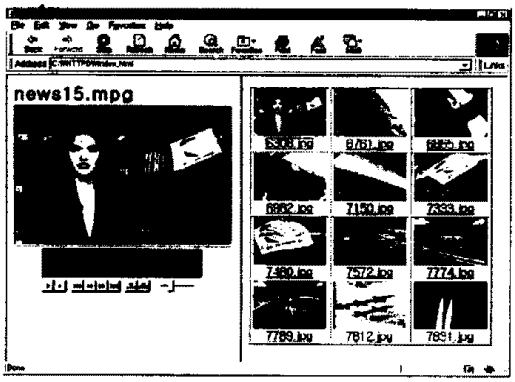


그림 11. 웹 브라우저를 사용한 검색 예

2. 검토 및 향후 연구

구현된 시스템은 멀티미디어 데이터베이스 구축을 위한 많은 가능성을 제시해주었지만 아직 몇가지 문제점을 안고 있다. 우선 자동 인덱싱에 많은 시간이 소요되며 이것은 주로 화면을 캡처하는데 걸리는 시간이다. 시공간적인 표본화 작업을 통해서 인덱싱 시간은 단축될 것으로 기대된다. 다음으로, 생성된 인덱스 파일 단위로만 검색이 가능하므로 다양한 검색 기능이 부족하여, 실제로는 보다 완벽한 데이터베이스와 검색 엔진의 구축이 필요하다. 자동 인덱싱 알고리즘의 경우, 카메라 움직임 분석기법 및 물체의 움직임 추정 연구를 통해 앞서 보인 검출 오류의 개선을 연구 중이다.

V. 결 론

본 논문에서는 멀티미디어 데이터베이스 구축과 효과적인 인터페이스 개발을 위한 기반 단계로 비디오 검색과 저작 편집에 주로 이용될 수 있는 샷 단위의 동영상 검색 시스템을 네트워크 상에서 구현하였다. 이를 활용하여 원거리에 위치한, 압축된 동영상을 효과적으로 검색/재생할 수 있다. KBS에서는 검색 뿐만 아니라 스튜디오 화질의 영상 포맷을 사용하며, 비선형편집 기능을 가진 네트워크에 기반한 프로그램 제작 환경 구축에 관한 연구를 계속 추진할 계획이며, 본 논문에서 제시한 동영상 검색 시스템은 이러한 연구의 기반 기술로서 차세대 방송 환경 구축에 크게 기여할 것으로 기대된다.

참 고 문 헌

[1] 김종태, 조창익, "방송 자료의 데이터베이스와 검색시스템 구축" 방송공학회지 제 1권 4호, pp.12-20, 1996년 12월

[2] 한국방송공사 기술연구소, 영상 정보 처리 시스템 연구보고서, 1996년 12월

[3] Microsoft Corporation, ActiveMovie 1.0 SDK Developer's Guide, 1997

[4] Stephen W. Smoliar and HongJiang Zhang, "Content-Based Video Indexing and Retrieval", IEEE Multimedia, Vol. 1, No. 2, pp. 62-72, 1994

[5] HongJiang Zhang, Atreyi Kankanhalli, Stephen W. Smoliar, "Automatic Partitioning of Full-motion Video", Proc. ACM Multimedia System, Vol. 1, No. 1, pp. 10-28, 1993

[6] Alian Legault, Janet Matey, "Professional Video Under 32-bit Windows Operation Systems", SMPTE Journal Technical Paper, pp 760-767, December 1996