# 실시간 고압축 MPEG-4 비디오 코딩을 위한 전처리 시스템

## (Preprocessing System for Real-time and High Compression MPEG-4 Video Coding)

김 준 기 † 홍 성 수 †† 이 호 석 †††
(Junki Kim) (Sung Soo Hong) (Ho Suk Lee)

**요 약** 본 논문에서는 MPEG-4 비디오 부호화와 복호화 시스템의 실용성과 고효율의 압축을 위한 새로운 알고리즘을 개발하였다. MPEG-4 비디오 그룹에서는 실험과 경험을 통하여 비디오 검증 모델인 VM(Verification Model)을 개발하였다. 또한 MPEG-4 표준화 과정을 통하여 ISO/IEC 14496-2 표준 문서와 VM에 기반하여 다양한 참조 소프트웨어가 개발되었다. MS-FDAM은 MPEG-4 참조 소프트웨어로서 표준 부호화와 복호화로 개발되었으나 고효율의 압축과 실용성에 제한이 있다. 이에 본 논문은 기본 MS-FDAM모델에 내용 기반 비디오 코딩의 핵심인 VOP 추출 알고리즘, 실시간 입력 시스템, 압축율을 높일 수 있는 움직임 감지 알고리즘을 추가하여 최대 180:1의 압축율을 보여주는 실시간 고압축 MPEG-4 시스템을 개발하였다.

**키워드** : MPEG-4, VOP 추출, 실시간 전처리, 고압축

***Abstract*** In this paper, we developed a new and robust algorithm for a practical and very efficient MPEG-4 video coding. The MPEG-4 video group has developed the video Verification Model(VM) which evolved through time by means of core experiments. And in the standardization process, MS-FDAM was developed based on the standard document of ISO/IEC 14496-2 and VM as a reference MPEG-4 coding system. But MS-FDAM has drawbacks in practical MPEG-4 coding and it does not have the VOP extraction functionality. In this research, we implemented a preprocessing system for a real-time input and the VOP extraction for a practical content-based MPEG-4 video coding and also implemented the motion detection to achieve the high compression rate of 180:1.

**Key words** : MPEG-4, VOP extraction, real-time preprocessing, high compression

## 1. 서 론

최근 고성능 멀티미디어 기기들의 활용과 무선 디지털 통신, 데이타베이스 등의 급속한 신장으로 기존 영상 기술의 핵심인 압축 부호화뿐만 아니라 고도의 새로운 기능들(영상분할, 내용인식, 3차원 편집)을 지원하는 내용 기반 압축 방식의 필요성이 대두되었다. MPEG (Moving Picture Experts Group) 그룹에서는 멀티미디어 산업의 빠른 변화를 예상하여 멀티미디어 환경에서 동영상 코딩을 위한 새로운 국제 표준인 MPEG-4를 제안하였다.

MPEG-4 비디오 표준은 오디오 비디오 데이타의 이식성과 범용 접근성, 고비율 압축, 비디오 데이타의 상호작용을 고려하여 멀티미디어 응용에서 비디오 코딩을 위한 표준으로 설계되었다. 따라서 MPEG-4 비디오는 영상 데이타의 조작과 전송 그리고 높은 압축을 제공하는 핵심기술을 목표로 하였다.

MPEG-4 비디오 그룹에서는 실험과 경험을 통하여 비디오 검증 모델인 VM(Verification Model)을 개발하였다. VM은 표준 형태로 부호화와 복호화를 정의하였고, 여러 기능을 알고리즘과 도구로서 소개하였다. 다음 표 1은 VM 버전 1과 2에서 지원하는 알고리즘이다.

또한 VM 개발에서는 MPEG-4 국제 표준화 문서인

· 본 연구는 호서대학교 2003년도 특별학술연구비 지원 및 2003년도 IT 학과 교과과정 개편지원사업 특별연구비 지원에 의해 이루어졌음
† 비 회 원 : 호서대학교 컴퓨터공학부
kjk73@hanmail.net
†† 종신회원 : 호서대학교 컴퓨터공학부 교수
sshong@office.hoseo.ac.kr
††† 정 회 원 : 호서대학교 컴퓨터공학부 교수
hslee@office.hoseo.ac.kr
논문접수 : 2002년 10월 9일
심사완료 : 2003년 6월 23일

표 1 VM-1,2에서 지원하는 알고리즘

| VM-1,2에서 지원하는 알고리즘 | 버전 | 설명 |
|---|---|---|
| 기본<br>(I-VOP, P-VOP, AC/DC 예측,<br>4MV(Motion Vector), 비제한 MV | 1 | 지원 |
| B-VOP<br>(* B-VOP 비트율 제어부분 지원 없음) | 1 | 지원 |
| 양자화(Quantization) | 1 | 지원 |
| 에러 내성(Error Resilience)<br>(* 에러 복구 지원 없음) | 1 | 지원 |
| 이진 형상(Binary Shape) | 1 | 지원 |
| 그레이 형상(Grayscale Shape) | 1 | 지원 |
| Temporal Scalability | 1 | 지원 |
| Spatial Scalability | 1 | 지원 |
| Sprite | 1 | 지원 |
| Still Texture | 1 | 지원 |
| Global Motion Compensation | 2 | 지원 |
| Quarter-pel Motion Compensation | 2 | 지원 |
| SA-DCT | 2 | 지원 |
| Error Resilience for Still Texture Coding | 2 | 지원 |
| Object Based Spatial Scalability | 2 | 지원 |
| Wavelet | 2 | 지원 |
| FGS(Fine Granularity Scalability) | 2 | 지원 |
| Scalable Arbitrary Shape for Still Texture Coding | 2 | 지원 |
| Complexity Estimation Support | 2 | 지원 |

ISO/IEC 14496-2 비디오 부분이 반영되었으며, MPEG-4 표준화가 진행되는 동안 다양한 MPEG-4 참조 소프트웨어가 개발되었다. 대표적인 참조 소프트웨어로는 MS-FDAM이 있다[1]. MS-FDAM은 기본적인 부호화와 복호화 처리를 하였으나 실용적인 소프트웨어 개발을 위한 전처리 시스템 및 실시간 비디오 처리, 압축효율에 많은 제한이 있다. 다음 리스트는 MS-FDAM에서 개발되지 못한 부분이다[2,3].

- VOP코딩은 다중 VOP 부호화로 설계되었으나 부호화 처리는 오직 단일 VOP로 부호화 된다.
- 완벽한 프레임 처리를 위한 VOP 혼합 알고리즘이 존재하지 않는다.
- 프레임처리에 있어 오직 4:2:0의 비디오 포맷만을 지원한다.
- 후처리 필터(filter) 부분이 없다.
- 전처리 단계의 핵심 기술인 비디오 객체 분할을 위한 자동 알고리즘이 구현되어 있지 않다.
- 화상카메라를 이용하여 실시간으로 데이타를 입력받는 부분이 없다.
- 비트율 제어부분이 불안정하다.
- B-VOP에 대한 MPEG-4 비트율 제어부분이 없다.
- 부호화와 복호화를 위한 불명확한 요구사항이 많다.

이에 본 논문에서는 VM, MS-FDAM, ISO/IEC 14496-2 비디오 문서를 기반으로 기존에 개발되지 못한 MS-FDAM 기능중에서 실용적이고 더욱 높은 압축율을 나타낼 수 있는 기능을 구현하여 MPEG-4 부호화와 복호화 시스템을 개발하였다.

첫째, 기본적인 MS-FDAM에 화상카메라를 이용한 실시간 동영상 입력을 위한 병렬 쓰레드 모듈을 개발하였다. 따라서 개발된 MPEG-4 시스템은 실제 동영상을 처리할 수 있으며 더욱 정밀한 프레임 처리가 가능하였다.

둘째, MPEG-4 부호화의 핵심 기술인 내용 기반 부호화 코딩을 위해 반 자동, 자동 비디오 객체 분할 알고리즘을 개발하였다. 비디오 객체 분할은 시간적 분할로 두 영상의 차이를 빠르고 정확하게 추출할 수 있는 프레임 차이(frame difference)를 사용하였고, 공간적 분할로는 영상 자체의 구분을 위한 Canny 에지 검출 알고리즘을 사용하였다. 이후 두 개의 알고리즘에서 추출된 객체를 조합하여 VOP를 추출하였다. 추출된 VOP를 이용한 내용 기반 부호화는 최대 180:1의 높은 압축을 보여주었다.

셋째, 더욱 높은 효율 압축을 위해 움직임이 없는 프레임에서는 부호화를 수행하지 않는 효과적인 움직임 감지 알고리즘을 개발하였다. 움직임 감지 알고리즘은 현재 프레임과 이전프레임 사이의 픽셀 값 차이(difference)를 이용하여 일정한 감지점 이하가 나타나면 부호화 하지 않도록 프레임을 드롭(drop)하였다. 즉 영상에서 반복되는 고정된 프레임을 드롭하여 더욱 높은 압축율을 이룩하였다. 또한 복호화 부분에서는 드롭된 프레임의 복구를 위해 드롭된 프레임의 번호 정보를 이용하여 원영상과 거의 차이가 없는 영상으로 프레임을 복구할 수 있도록 하였다.

본 논문은 이전 MPEG-4 VM과 MS-FDAM을 참고하여 효과적인 알고리즘을 적용하여 실용적이고 고효율의 MPEG-4 부호화와 복호화 시스템을 개발하였다. 새롭게 개발된 MPEG-4 시스템은 일반적인 MPEG-4 실험 영상과 화상카메라를 이용하여 실시간으로 입력 받은 실제 동영상을 사용하여 MS-FDAM의 압축결과와 비교하여 실험하였다. 실험결과 개발된 객체 분할은 정확한 객체의 경계를 보여주었고, 객체를 사용한 MPEG-4 내용 기반 부호화 코딩은 최대 180:1의 고효율, 고압축율의 향상된 결과를 보여주었다.

본 논문의 구조는 다음과 같다. 2장에서는 관련 연구분야와 MPEG-4 VM의 알고리즘을 소개하며 3장에서는 제안된 알고리즘을 소개하였고, 4장에서는 시스템 구현과 실험 결과 및 인터페이스를, 마지막으로 5장에서는 결론을 맺었다.

## 2. 관련 연구

MPEG-4의 기본적인 목표는 고효율, 고압축 부호화의 추구이다. MPEG-4 비디오 부호화의 특징은 부호화 효율 개선을 위하여 8x8 블록 움직임 보상, 직접(direct)예측, AC/DC 계수의 예측 등 몇 개의 툴이 포함되었다. 또 스프라이트(sprite)로 불리는 완전히 새로운 배경 합성의 기술도 고려되었다. 또한 용도를 한정(비디오 또는 음성)해서 한층 더 고효율, 고압축의 부호화를 달성하였다.

또 다른 MPEG-4 특징은 에러 내성의 강화였다. 에러 내성은 데이타 전송중 외부 노이즈에 대한 데이타 결손 보호에 따른 기법으로 에러 수정 부호의 접근이 아니라 에러 은폐 기술이었다. 그것은 비트스트림에 에러가 혼합되도 외형으로는 그 영향을 모르게 하는 기술로 생성 비트 길이(bit length)를 기준으로한 패킷 분할, 데이타 파티션닝(data partitioning), 스터핑 바이트(stuffing byte)의 개선 등이 행해졌다.

마지막으로 MPEG-4 특징은 멀티미디어 대응이다. 이것은 종래의 자연 비디오, 오디오 등의 단일의 미디어만을 취급하는 것이 아니라, MPEG-4에서는 그것들에 추가적으로 합성 비디오, 오디오, 음성 등의 여러가지 미디어를 하나의 표준안에서 동등하게 취급하였다. 또한 MPEG-4에서는 장면을 구성하는 비디오나 오디오등의 각 객체(Audio Visual)에 주목해, 각 AV객체는 독립해 부호화 될 수 있도록 최적의 부호화 방법을 선택할 수 있다.

### 2.1 MPEG-4 VM 부호화 알고리즘

MPEG-4 VM 비디오 부호화 과정은 다양한 형태로 제공된다. 우선 형상 부호화는 VOP의 형상 정보를 사용해서 각 픽셀의 투과도를 나타내는 알파(alpha) 값으로 주어진다. 이러한 알파값을 이용하여 VOP의 영역을 분할하고 복수의 VOP를 표시할 수 있다. VOP 형상 정보는 Rectangle, Binary, Gray-scale과 같은 3가지 종류로 나누어지며 각 형상정보에 따라 부호와 된다.

MPEG-4 VM 움직임 보상은 임의의 형상 VOP를 이용하기 때문에 VOP 경계에 걸치는 매크로블록이 존재한다. 이를 해결하기 위하여 새롭게 패딩(padding)과 다각형 매칭(polygon matching) 기술을 이용한 움직임 보상이 수행된다. 또한 움직임 벡터의 탐색에는 비제한 움직임 탐색 기법이 도입되었다. 이전 움직임 보상의 참조 블록은 완전하게 참조 화상 내부에 있는 것 만 선택하였다. 그러나 MPEG-4는 화상 경계에 걸치는 곳의 움직임 보상도 가능하다. 화상 경계의 블록은 패딩처리를 수행한다. 이 확장된 참조 화상을 기초로 움직임 보상을 하기 때문에 비제한 움직임 탐색 기법은 화상 경계에 있는 움직임 정밀도가 올라가며, 화질, 부호량이 개선된다. 또한 직접(Direct) 예측 모드를 적용해 하나의 움직임 벡터 값으로 양방향 효과를 보여준다[4,5].

### 2.2 VOP 추출 알고리즘

현재 존재하는 객체 분할 방법은 다음과 같은 방법을 포함한다. 컬러 동질성을 이용한 지역 기반 접근 방법, 움직임 표준을 이용한 객체 기반 접근 방법, 객체 추적을 이용한 접근 방법 등의 많은 방법이 있다. 영상에서 의미가 있는 객체는 많은 움직임과 컬러에 있어서 공통적인 값을 포함한다. 그래서 단일 컬러나 기본 움직임은 객체 분할에 있어 만족한 결과를 이끌 수 없다. 이에 최근 VOP 추출 방법에 대하여 많은 제안이 발표되었다[6-9].

MPEG-4 표준 버전에서 권장하는 객체 분할 구조는 시간적 분할[10], 공간적 분할, 시간-공간적 분할 결합의 세 부분으로 나누어져 있다. 시간적 분할이라는 것은 영상안에 있는 객체의 움직인 위치를 추출하는 작업이고 공간적 분할은 주어진 기준에 따라서 정확한 객체의 경계와 영상안에있는 의미있는 객체의 지역을 추출하는 작업이다. 이 두 가지의 작업을 통합하여 마지막으로 영상으로부터 각 객체를 추출하는 것이다. 움직임 객체를 추출하는 단계에서 가장 중요한 단계는 시간-공간의 조합이다. 적절한 방법을 통하여 두 객체를 조합에 따라서 정확한 움직임 객체를 구별할 수 있다.

## 3. 제안된 알고리즘

MPEG-4 비디오 시스템 및 부호기 전체 구조는 그림 1과 같다.

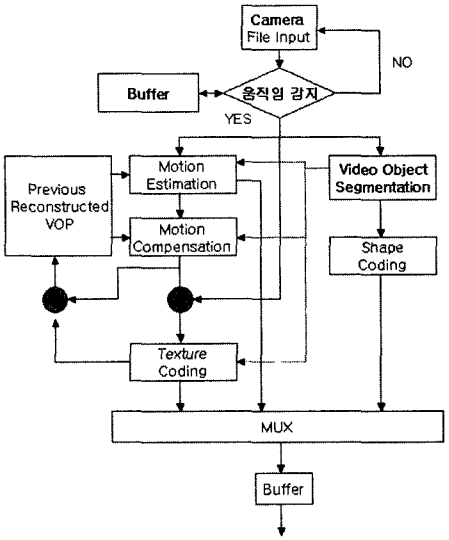



그림 1 MPEG-4 시스템 및 부호기 구조

본 논문의 실험 영상은 실시간 입출력 모듈을 적용하였다. 실시간 입출력 모듈은 파일 입출력을 통한 동영상 데이타의 처리가 아니라 화상 카메라 입력을 통한 실제 동영상 데이타의 처리이다. 즉 실시간이란 이전 MS-FDAM과 같이 오프라인의 파일 입출력이 아니라 온라인상에서 화상카메라를 이용한 실제 동영상을 입력 받아 처리하는 형태이다. 실시간 처리를 위한 시간적인 제약은 화상카메라로 입력되는 초당 프레임 수에 제한된다. 화상카메라 입력 영상은 초기에 초당 30프레임으로 설정하였고 사용자 정의에 따라 화상 카메라의 시간 제약과 프레임 수는 표 2와 같이 설정하였다.

표 2 초당 프레임 수

| 설정 모드 | 초당 프레임 수(s/f) |
|---|---|
| 1 | 1/30 |
| 2 | 1/15 |
| 3 | 1/10 |
| 4 | 1/5 |
| 5 | 1/1 |

움직임 감지 기법은 움직임이 아주 미세한 부분을 감지하여 객체의 움직임이 없는 영상 데이타는 부호화를 수행하지 않도록 프레임을 드롭(drop)하였다. 복호기에서는 프레임 드롭에 따른 동영상 화질(프레임수)를 복원하기 위하여 감지된 프레임의 번호 정보를 이용하여 같은 크기의 영상 파일로 복원하였다. 이에 이전 MS-FDAM보다 영상 데이타의 고효율, 고압축율의 효과를 가져왔다.

자동 객체 분할은 이전에 개발된 알고리즘을 이용하여[6,7], 실시간 처리에 적용할 수 있도록 새롭게 개발하였다. 즉 계산량을 더욱 줄여 부호화 처리의 지연 시간을 향상 시켰다. VOP 추출의 처리 시간은 4장 시험 결과에 제시하였다. 이전에 언급 하였듯이 MPEG-4의 효율적인 부호화 과정은 일반적인 프레임 압축뿐만 아니라 객체를 이용하는 형상 부호화 알고리즘에 초점을 맞추었다.

### 3.1 실시간 MPEG-4 입력 모듈

실시간 입력 모듈을 위한 병렬 쓰레드 구성은 그림 2와 같다.

동영상 관련 데이타는 최근 하드웨어의 급속한 발전에도 불구하고 영상 데이타 처리에 많은 계산을 필요로 해 높은 CPU 점유율을 보여준다. 때문에 영상 데이타를 입력하여 출력 결과를 얻기까지 많은 시간이 소요된다. 이에 본 논문에서는 동영상 데이타 입력과 부호화 모듈을 병렬 쓰레드로 구현하여 순차 쓰레드를 사용할 경우보다 평균 11% 이상의 효율 향상을 이룩하였다.

병렬 쓰레드 구성은 그림 2와 같이 쓰레드-1과 쓰레드-2로 구성된다. 쓰레드-1의 기능은 화상 카메라 입력 포맷을 설정한다. 그리고 연속해서 영상을 카메라로부터 입력 받아 버퍼에 저장하는 기능 및 저장된 영상을 통하여 움직임을 감지하는 기능으로 구성된다. 쓰레드-2에서는 움직임이 발생된 영상 데이타를 입력 받아 파라미터를 설정하고 이후 부호화 형태(프레임 부호화, 이진 형상 부호화, 다중 형상 부호화, 스프라이트 코딩)를 구분하여 MPEG-4 부호화 과정을 수행한다.

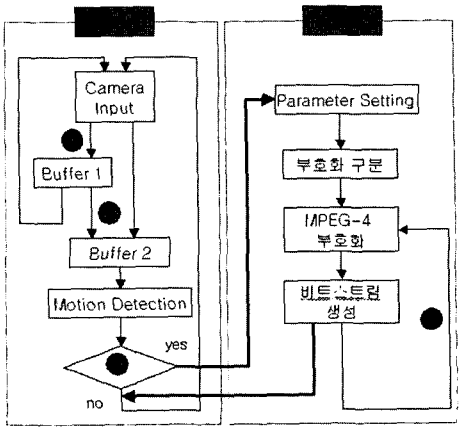
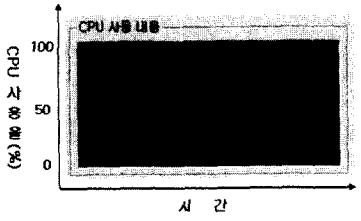



그림 2 병렬 쓰레드 구성 및 입력 모듈

다음은 순차 쓰레드 방식과 병렬 쓰레드 방식의 입력 모듈에 대한 CPU 사용율의 실험 결과를 그래프를 이용하여 좀 더 자세하게 비교하여 나타낸 것이다.

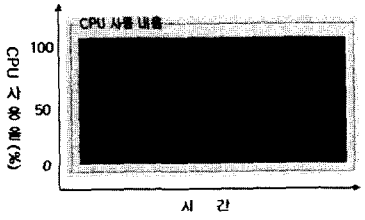
[예 1-1] 순차 쓰레드 방식

1) 움직임이 많은 QCIF(176 x 144) 포맷 입력 영상 2000 프레임 (예: Hall monitor)




이 경우에 CPU의 사용율은 평균 76%였다.

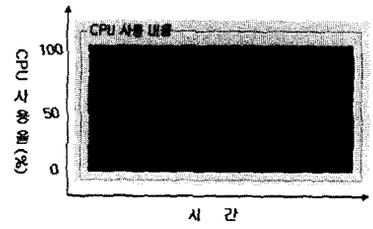
2) 움직임이 적은 QCIF(176 x 144) 포맷 입력 영상 2000 프레임 (예: Akiyo)



이 경우에 CPU의 사용율은 평균 74%였다.

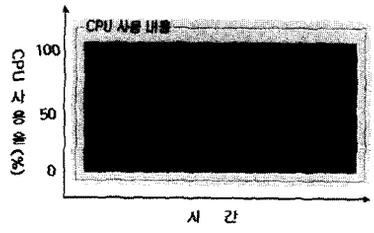
[예 1-2] 병렬 쓰레드 방식

1) 움직임이 많은 QCIF(176 x 144) 포맷 입력 영상 2000 프레임 (예: Hall monitor)




실험 결과 CPU의 사용율은 평균 65%였다.

2) 움직임이 적은 QCIF(176 x 144) 포맷 입력 영상 2000 프레임 (예: Akiyo)




실험 결과 CPU의 사용율은 평균 61%였다.

이 그래프들로부터, 본 논문에서 구현하여 사용한 병렬 쓰레드 방식의 입력 모듈은 순차 쓰레드 방식의 입력 모듈보다 평균 11% 이상의 효율 향상을 나타내는 것을 알 수 있다.

### 3.2 고압축을 위한 움직임 감지 처리

본 논문에 적용된 움직임 감지 기법은 사용자 정의에 따라 민감도와 감지점을 입력받아 부호화하고자 하는 영상의 차이를 측정하였다. 측정된 영상이 일정한 감지점 이하의 값이 발생하면 부호화 과정을 스킵하고 그 이상의 값이 발생하면 부호화 과정을 처리하였다.

움직임 감지를 수행하는 방법은 다음과 같다.

1. 민감도 및 감지도 입력
2. 화상 카메라로 처음 입력 영상을 버퍼에 저장
3. 두번째 영상을 화상 카메라로 입력
4. 저장된 영상과 입력 영상의 픽셀 차이값 계산
5. 차이 픽셀이 감지점 이하이면 부호화 과정을 스킵(skip), 부호화 플래그 0으로 설정하고 시간 정보 입력
6. 차이 픽셀이 감지점 이상이면 부호화 과정을 수행, 부호화 플래그 1로 설정하고 시간 정보 입력
7. 2~6번과정 반복

위의 처리방법 중 핵심 부분인 4,5,6번을 간결하게 수식으로 표현하면 다음과 같다.

[식 1-1]

*if ( CIP - PIP > threshold )*

*PDC = PDC + 1 otherwise PIP = CIP*

[식 1-2]

*if ( PIP - CIP > threshold )*

*PDC = PDC + 1; otherwise PIP = CIP*

[식 1-3]

*if(PDC > theta )*

*Flag = 1 otherwise Flag = 0*

*PDC : Pixel Difference Count*

*CIP : Current Image Pixel*

*PIP : Previous Image Pixel*

*threshold : 민감도, theta : 감지점*

본 논문에서는 실시간 처리를 고려하여 시간적 낭비를 줄이고 유동적으로 움직임 차이를 빠르게 얻을 수 있도록 [식 1-1], [식 1-2], [식 1-3]과 같이 프레임 차이 픽셀값을 이용하여 임계값을 비교하였다.

[식 1-1]은 현재 영상 픽셀에서 이전 영상 픽셀을 빼는 과정으로 그 결과값이 민감도 보다 크다면 전체 픽셀 차이수를 하나 증가한다. 그렇지 않으면 이전 영상 픽셀을 현재 영상 픽셀로 변환한다. [식 1-2]는 영상에 음수값을 배제하기 위하여 [식 1-1]의 과정을 역으로 적용하였다. [식 1-1]과 [식 1-2]에서와 같이 만약 민감도 값이 크면 두 영상의 전체 픽셀 차이수가 작아지며 그 반대이면 전체 픽셀 차이수가 커진다. [식 1-3]은 전체 픽셀 차이 값과 임의의 감지점 값을 적용하여 유동적으로 화상 카메라에 입력되는 영상에 따라 효율적인 움직임 감지를 처리할 수 있다.

### 3.3 민감도와 감지점 설정

움직임 감지를 처리하는 소스 코드는 그림 3과 같다.

움직임 감지 알고리즘에서 가장 중요한 부분은 민감도와 감지점에 대한 임계값을 적용하는 과정이다. 그림 3의 sw1과 sw2와 같이 민감도란 두 영상에서의 픽셀 차이크기를 보여준다. 만일 민감도 값을 아주 큰 값으로 설정하였다면 차이 픽셀 수가 작아지며 반대로 민감도 값을 아주 작은 값으로 설정하였다면 차이 픽셀 수가 커진다. sw3와 같은 감지점이란 그림 3에서와 같이 마지막으로 민감도에 의해서 발생한 차이 픽셀 수에 따른 프레임의 드롭(drop)을 판단할 수 있다. 즉 민감도는 두 영상의 전체 픽셀 차이를 얻을 수 있고 이를 이용하여 감지점 값을 설정하면 전체 픽셀의 수와 감지점의 값을 비교하여 부호화 유무를 결정할 수 있다.

민감도 값의 초기 설정은 0.0에서 100.0사이로 하였다. 이러한 값을 설정한 이유는 동영상 부호화는 한 장의 영상을 부호화하는 것이 아니라 연속적인 영상 데이타의 상관 관계를 통하여 부호화를 하기때문에 이전 영

```
//움직임 감지
for (픽셀 위치 = 0 ; 픽셀 위치 <
                          픽셀 저장 버퍼; 픽셀 위치++)
{
  현재 픽셀값 = *(현재 영상 데이타 + 픽셀 위치);
  if (현재 픽셀값 > *(이전 영상 버퍼 + 픽셀 위치)) {
        if(현재 픽셀값 -
          *(이전 영상 버퍼 + 픽셀 위치) > sw2) {
            차이 픽셀 수 ++;
        }
  }
  if (현재 픽셀값 < *(이전 영상 버퍼 + 픽셀 위치)) {
        if(*(이전 영상 버퍼 + 픽셀 위치) -
           현재 픽셀값  > sw2) {
            차이 픽셀 수 ++;
        }
  }
  *(이전 영상 버퍼 + 픽셀 위치) = 현재 픽셀 값;
}
감지점 정의 값 = sw3;
if(차이 픽셀 수  > 감지점 정의 값){
      부호화 수행; 시간정보 설정;
}
else{
      부호화 스킵; 시간정보 설정;
}
```

• sw1 : 민감도, sw2 : 민감도, sw3 : 감지점

그림 3 움직임 감지 추출 소스

상과 현재 영상과의 관계는 유사한 형태를 보인다. 영상에서 하나의 픽셀이 갖는 값은 RGB일 경우 0부터 255까지의 값을 갖는다. 즉 이전 영상의 픽셀 값과 현재 영상의 픽셀 값의 최대 차이 값은 255이다. 일반적으로 이전 영상과 현재 영상의 차이가 255 나오는 경우는 사용자가 임의로 만든 영상이외에는 발생확률이 적다. 때문에 본 논문에서는 여러 영상을 실험하여 0.0부터 100.0까지의 민감도 값을 설정하였을 경우 두 영상의 모든 차이값을 효율적 나타내었다. 즉 사용자가 화상 카메라로 입력되는 실제 동영상의 특성에 따라 유동적으로 민감도를 낮추면 두 영상의 전체 차이 픽셀수가 커지며, 민감도를 높이면 두 영상의 전체 차이 픽셀수가 작아진다. 감지점값은 사용자의 선택에 따라 프레임 드럽을 많이 하려면 감지점 값을 높게 설정하고 그 반대이면 낮게 설정하도록 하였다. 즉 감지점 값은 민감도에 따라 유동적으로 적절한 범위의 임계값을 적용하였다.

### 3.4 움직임 감지 알고리즘 실험

움직임 감지를 위한 실험 영상은 화상 카메라에 적용될 수 있는 세 종류 형태의 실제 동영상으로 움직임이 많은 영상과 움직임이 적은 영상으로 실험하였다. 실험 영상은 각각 RGB24 비트 포맷의 영상으로 320*240사이즈 영상을 사용하였으며 전체 프레임 수는 2000프레임을 사용하였다. 민감도의 범위는 0.0부터 100.0까지

그림 4 100 프레임에 대한 차이 픽셀 수와 프레임 수

| 시퀀스 (RGB24) 320*240 | 민감도/감지점 | 전체 프레임 수 | 움직임 감지 수 | 드롭 프레임 수 |
|---|---|---|---|---|
| 움직임이 많은 영상 | 20/25000 | 2000 | 1968 | 32 |
| | 40/25000 | 2000 | 1884 | 116 |
| | 60/25000 | 2000 | 1643 | 357 |
| | 80/25000 | 2000 | 1547 | 453 |
| | 100/25000 | 2000 | 1268 | 732 |
| 움직임이 적은 영상 | 20/25000 | 2000 | 1802 | 198 |
| | 40/25000 | 2000 | 1675 | 325 |
| | 60/25000 | 2000 | 1461 | 539 |
| | 80/25000 | 2000 | 1245 | 755 |
| | 100/25000 | 2000 | 1032 | 968 |

그림 5 화상카메라 고정, 객체 움직임

설정하였고, 감지점의 범위는 0부터 100000.0까지를 설정하여 유동적으로 실험하였다. 그림 4에 보여지는 그래프는 가로 방향으로 100개의 프레임 수와 세로 방향으로 0.0부터 100000.0 범위의 차이 픽셀 수를 보여준다. 그림 4는 민감도를 20.0, 감지점을 25000.0으로 적용한 실험 결과이다. 즉 차이 픽셀 수가 25000.0의 값보다 크면 움직임이 검출되고 그 이하의 값이면 움직임을 스킵한다.

감지점은 그림 4에서 이중 실선으로 표시하였다. 그림 4는 0부터 100 프레임까지 두 영상 사이의 차이 픽셀수의 결과를 그래프로 보여준다. 그림 4에서와 같이 움직임이 검출된 100프레임까지의 전체 차이 픽셀 수는 평균 66914.0의 높은 수치를 보였다. 그림 5는 2000프레임의 실험 영상에 대하여 여러 민감도로 설정하였을 경우 전체 실험 결과를 보여준다.

실험 결과 그림 5에서와 같이 움직임이 적은 영상에서 민감도 100.0과 감지점 25000.0의 설정은 최대 절반 정도의 프레임 드롭을 확인할 수 있었다.

다음 그림 6은 화상카메라가 움직임이 있고 객체는 고정되어있는 경우 민감도는 20.0이고 감지점은 25000.0을 적용한 실험 결과이다.

그림 6의 실험 결과는 그림 5의 실험 결과보다 프레임 드롭 수가 낮았다. 프레임 드롭 수가 낮은 이유는 그림 5의 경우는 전체 화면에 객체만 움직이기 때문에 주변 배경 픽셀은 거의 동일하다. 그러므로 움직이는 객체

| 시퀀스 (RGB24) 320*240 | 민감도/감지점 | 전체 프레임 수 | 움직임 감지 수 | 드롭 프레임 수 |
|---|---|---|---|---|
| 움직임이 많은 영상 | 20/25000 | 2000 | 1992 | 23 |
| | 40/25000 | 2000 | 1958 | 83 |
| | 60/25000 | 2000 | 1923 | 152 |
| | 80/25000 | 2000 | 1908 | 243 |
| | 100/25000 | 2000 | 1897 | 453 |
| 움직임이 적은 영상 | 20/25000 | 2000 | 1948 | 52 |
| | 40/25000 | 2000 | 1867 | 133 |
| | 60/25000 | 2000 | 1766 | 234 |
| | 80/25000 | 2000 | 1647 | 353 |
| | 100/25000 | 2000 | 1424 | 576 |

그림 6 화상카메라 움직임, 객체 고정

| 시퀀스 (RGB24) 320*240 | 민감도/감지점 | 전체 프레임 수 | 움직임 감지 수 | 드롭 프레임 수 |
|---|---|---|---|---|
| 움직임이 많은 영상 | 20/25000 | 2000 | 1991 | 9 |
| | 40/25000 | 2000 | 1978 | 22 |
| | 60/25000 | 2000 | 1955 | 45 |
| | 80/25000 | 2000 | 1933 | 67 |
| | 100/25000 | 2000 | 1911 | 89 |
| 움직임이 적은 영상 | 20/25000 | 2000 | 1988 | 12 |
| | 40/25000 | 2000 | 1965 | 35 |
| | 60/25000 | 2000 | 1933 | 67 |
| | 80/25000 | 2000 | 1909 | 91 |
| | 100/25000 | 2000 | 1855 | 145 |

그림 7 화상 카메라 움직임, 객체 움직임

의 주변 픽셀 차이만 계산하기 때문에 전체 차이 픽셀 수가 낮아진다. 여기에 감지점 값의 범위를 높여주면 더욱 크게 프레임 드롭 수가 발생한다. 하지만 그림 6에서는 화상카메라가 연속적으로 다른 영상을 입력하기 때문에 전체 차이 픽셀 수가 높아지며, 또한 움직임의 변화의 크기에 따라 차이 픽셀 수가 크게 발생한다. 이에 그림 6의 실험 결과는 프레임 드롭수가 낮은 결과를 얻었다.

마지막으로 그림 7은 화상카메라와 객체 모두가 움직이는 영상으로 민감도는 20.0이고 감지점은 25000.0을 적용한 결과이다.

전체 실험 결과 움직임이 작은 영상은 20~30.0정도의 민감도와 25000.0정도의 감지점을 설정하였을 경우 효율적인 움직임 검출이 수행되었으며, 움직임이 많은 영상은 40~50.0 정도의 민감도와 25000.0정도의 감지점 설정이 효율적인 움직임 검출을 수행하였다. 그림 7은 카메라와 객체 모두가 움직이는 영상이기 때문에 이전 실험 결과인 그림 6보다 전체적으로 프레임 드롭수가 낮은 결과를 보였다.

### 3.5 움직임 감지 복원

적용된 움직임 감지 알고리즘은 부호화하는 전체 프레임에서 민감도와 감지점의 설정에 따라 움직임이 없거나 움직임이 적은 프레임을 드롭하였다. 이러한 결과는 움직임이 없는 고정된 프레임을 드롭하여 높은 압축율을 보여 주었다. 또한 압축된 영상을 복원할 경우 움직임이 있는 동영상의 중요한 정보는 모두 확인할 수 있었다. 즉 이것은 동영상의 화질이 떨어지는 것이 아니라 프레임 수가 줄어드는 것이다.

프레임 복원을 위한방법으로는 프레임의 번호 정보를 사용하였다. 프레임 번호는 프레임의 드롭 유무를 판단할 수 있는 값이다. 실험은 드럽된 영상과 복원된 영상 사이의 화질 차이를 비교하기 위하여 PSNR을 구하였다. 두 영상의 PSNR(Peak Signal-to-Noise Ratio)를 구하기 위한 공식은 [식 2]와 같다.

[식 2-1]

$$PSNR = 10\log_{10}\left(\frac{(peak-pixel-value)^2}{MSE}\right)dB$$

[식 2-2]

$$MSE = \frac{\sum[f(i,j)-F(i,j)]^2}{N^2}$$

여기서 MSE는 원 영상과 복원 영상간의 평균 제곱 오차를 나타낸다. 만일 MSE값이 0이면 원영상임을 가만하여 기준 설정인 50dB의 결과를 얻기 위해 MSE 값을 0.65025로 설정하였다.

다음 그림 8은 초당 5프레임이상의 입력으로 영상 포맷은 176*144사이즈, RGB24비트이다.

그림 8의 원영상을 민감도 20, 감지점 2500으로 설정하여 움직임 감지를 적용한 결과는 그림 9와 같다.

그림 9와 같이 초당 프레임수가 1이상인 경우는 1초에 한 프레임이 감지되거나 드롭되지 않는다. 즉 초당


그림 8 원영상

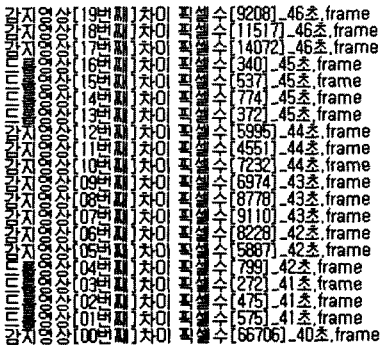



그림 9 감지결과와 프레임 번호 정보

그림 10 초당 5프레임 이상의 움직임 감지 영상

그림 11 복구 영상

프레임 수가 증가하면 같은 시간에 여러 프레임을 감지하기도 하며, 다른 시간에 걸쳐 영상의 감지와 드롭이 결정된다. 따라서 초당 1프레임 이상의 입력 영상은 프레임 간격이 좁아지므로 프레임과 프레임사이에 미세한 변화를 모두 얻을 수 있다. 이러한 결과는 앞 영상과 대치 영상간의 차이가 작기 때문에 표 3과 같은 높은 PSNR 결과를 얻을 수 있다. 다음 그림 10은 감지된 영상을 보여준다.

표 3 움직임이 발생한 부분에서의 PSNR

| 프레임 번호 | PSNR | 비고 |
|---|---|---|
| 원영상00번/<br>복원영상 00번 | 50.0000dB | 차이 없음 |
| 원영상01번/<br>복원영상 01번 | 36.5511dB | 13.4489dB 차이 |
| 원영상02번/<br>복원영상 02번 | 37.0987dB | 12.0913dB 차이 |
| 원영상03번/<br>복원영상 03번 | 36.3840dB | 13.616dB 차이 |
| 원영상04번/<br>복원영상 04번 | 33.0652dB | 16.9348dB차이 |
| 원영상12번/<br>복원영상 12번 | 50.0000dB | 차이 없음 |
| 원영상13번/<br>복원영상 13번 | 37.8303dB | 12.1697dB 차이 |
| 원영상14번/<br>복원영상 14번 | 34.5188dB | 15.4812dB 차이 |
| 원영상15번/<br>복원영상 15번 | 36.7430dB | 13.257dB 차이 |
| 원영상16번/<br>복원영상 16번 | 37.4523dB | 12.5477dB차이 |

그림 10은 감지 영상의 프레임 번호를 고려하여 좌측 상단의 첫번째 감지 영상인 00영상과 두번째 감지 영상인 05 사이에 01, 02, 03, 04번의 빠진 프레임 번호를 00번으로 대치하였다. 또한 12번 영상과 17번 영상 사이에 12번 영상을 13, 14, 15, 16으로 대치하였다. 이러한 과정으로 복원된 영상은 그림 11이다.

그림 11은 움직임 감지 영상을 프레임 번호 정보를 이용하여 복구한 영상이다.

다음으로 표 3은 그림 8의 원영상과 그림 11의 복구 영상에 대한 PSNR의 결과이다.

결론적으로 표 3의 움직임이 발생한 부분에서의 PSNR은 높은 PSNR을 나타내었다. 즉 초당 5프레임 이상의 입력으로 복원된 영상은 더욱 효율적으로 움직임 복원이 가능하며, 원영상과 복원된 영상의 화질차이는 시각적으로 거의 차이가 없었다. 그러므로 3.5절과 같은 움직임 감지 복원 방법은 드롭 영상과 복원 영상 간의 시각적 차이가 낮게 발생하므로 고효율, 고압축을 의 MPEG-4 부호화가 가능하였다.

### 3.6 비디오 객체분할 모듈 적용

본 논문에서 적용한 비디오 영상 입력 모듈과 객체 분할 모듈은 그림 12와 같다.

영상 입력 모듈은 먼저 영상 데이타를 위한 버퍼를 설정한다. 이후 사용자 모드에 따라 오프라인 처리와 온라인 처리를 수행한다. 오프라인 처리는 파일 입출력을 위한 과정이며 온라인 처리는 실시간 처리를 위한 화상 카메라 입력 과정이다. 이전의 MS-FDAM에서는 비디

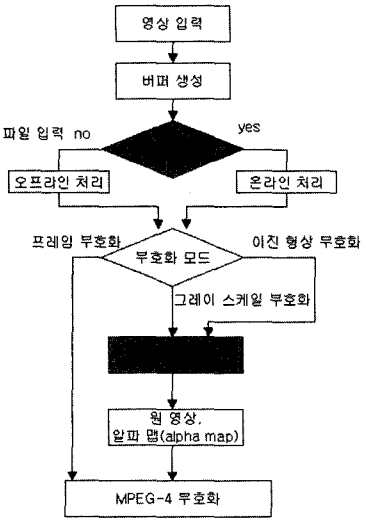



그림 12 객체 분할 모듈 적용

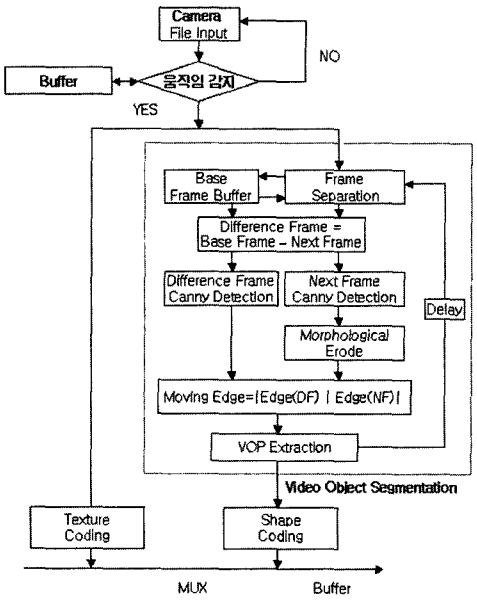



그림 13 비디오 객체 분할 모듈

오 객체 분할 모듈이 구현되어 있지 않다. 때문에 이진 형상 부호화나 다중 영상 부호화 처리를 위해서는 원 영상과 수동으로 제작된 알파맵 영상을 입력해야 하는 사용자 부담이 가중되었다. 하지만 본 논문에서 개발된 전처리 시스템은 초기 사용자의 입력 모드에 따라 자동으로 영상을 분할하는 객체 분할 모듈을 추가하였다. 이후 분할된 영상은 알파맵을 통하여 객체 기반 부호화를 수행한다.

### 3.7 VOP 추출 알고리즘

제안된 MPEG-4 비디오 전처리 시스템의 전체 구조와 비디오 객체 분할 모듈은 그림 13과 같다.

참고문헌[7,8]의 M.Kim's의 알고리즘[6] 또는 Meier and Ngan's 알고리즘[7]은 객체 분할을 위한 계산량 증가와 복잡한 구조 때문에 실시간 MPEG-4 시스템에 통합하기가 어렵다. 이에 논문에서는 새로운 알고리즘을 개발하였다. 이 알고리즘은 기초적인 변환 검출을 이용하기 때문에 움직임 추정이나 공간 분석 특징과 같은 계산적인 증가량이 없다. 개발된 알고리즘은 변환검출과 배경 저장, 수리 형태학 알고리즘을 이용하여 노이즈 지역의 제거가 빠르고 정확한 결과를 실시간 응용에서 보여주었다. 본 논문에서는 추가적인 알고리즘의 필요 없이 시간적, 공간적으로 분할된 객체의 에지(edge)에 수리형태학 알고리즘을 적용하여 정확한 객체의 경계를 찾았다. 본 알고리즘의 구조는 매우 간단하며 쉽게 객체의 정확한 경계를 찾았다. 알고리즘 구성은 5단계로 이루어져있다. 처음 단계는 분할을 위한 기준 프레임의 설정이다. 두 번째 단계는 현 프레임과 다음 프레임 사이의 차이값을 찾는다. 즉 두 프레임 사이의 차이값을 검출하여 두 객체 사이의 움직임 위치를 검출하는 것이다. 세 번째 단계는 객체 검출이 이루어지는 프레임의 에지 연산과 차이영상에 에지 연산을 수행한다. 네 번째 단계는 차이영상과 검출 영상에 수리형태학 녹임 연산을 적용하여 에지 크기를 조절하여 두 프레임 사이의 움직임이 있는 에지를 정확하게 찾는다. 다섯 번째 단계는 추출된 움직임 에지 영상을 통하여 VOP를 추출한다. 위와 같은 알고리즘을 사용하면 우선 알고리즘의 복잡도를 줄일 수 있으며, 복잡한 계산을 필요하지 않고 간단하게 정확한 객체 경계를 찾을 수 있다. 또한 거친 경계면이나 다중분할(over-segmentation)을 막을 수 있으며 영상에서 움직임이 있는 곳만을 빠르고 정확하게 추출할 수 있는 장점이 있다.

처음 단계는 프레임 분리와 기준 프레임의 설정이다. 즉 이것은 기준 프레임을 설정하는 과정이다. 왜냐하면 영상에서는 배경과 객체의 조합에 따라 최소한 3가지 타입의 영상 형태를 고려해야 하기 때문이다. MPEG-4 분할 과정에서 가장 널리 사용하는 영상 구성은 첫째 감시 카메라에서와 같이 배경은 고정되어 있고 전경에 움직임이 있는 영상이다. 이러한 영상은 본 논문에서 제한된 방법으로는 아주 정확한 움직임 경계를 얻을 수 있다. 또한 배경이 움직이는 경우와 전경이 고정되어 있는 경우는 전자에서 설명한 과정의 반대 형태로 분할이 가능하다. 또한 전경과 배경이 움직이는 형태의 영상은 아직까지 많은 연구가 진행되고 있으며 차후 다른 알고리즘을 사용하여 분할을 수행할 것이다. 즉 위에서 설명

하였듯이 첫 번째 과정은 움직임 검출에서 가장 중요한 단계로 움직임의 기준이 되는 프레임을 분리하여 기준 영상을 프레임 버퍼에 저장하는 과정이다. 일반적으로 영상 시퀀스에서 첫 번째 영상을 기준 프레임으로 설정한다.

두 번째 단계는 영상 시퀀스의 움직임을 검출하는 단계로 프레임 차이를(frame difference) 이용하여 움직임을 검출한다. 본 논문에서 차이값을 측정하는 방법으로는 다음과 같은 [식 3]을 사용하였다.

[식 3]

DFn = | F(baseframe)- Fn(nframe) |

*baseframe* : 기준 프레임, *nframe* : 다음 프레임

차이 영상 *DFn*(*Difference Frame*)은 원 영상의 기본 프레임(*base frame*)에서 다음 프레임 *Fn*(*next frame*)의 차이값을 계산하여 구한다. 차이값을 구하면 차이 영상으로부터 움직임 변화를 측정할 수 있다. 즉 처음 영상을 기본 프레임으로 버퍼에 저장하고 다음 영상과의 차이를 구한다.

세 번째 단계는 차이 영상에 대한 에지와 다음 프레임에 대한 에지를 추출하는 과정으로 각 영상에 대하여 Canny 에지 검출 알고리즘을 사용하였다. 다음은 [식 4]이다.

[식 4]

CannyEdge[Fn] = $\Phi(\nabla G * | fn |)$

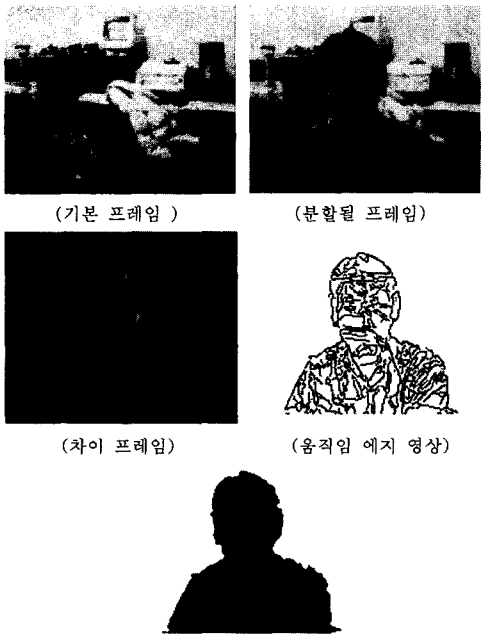



그림 14 VOP 추출 결과

Canny 에지 검출 방법은 기울기 연산 ▽에 가우시한 함수 G*fn을 수행하는 것이다.

네 번째 단계는 움직임 에지(moving edge) 추출 단계이다. 움직임 에지는 기준 프레임 에지와 다음 프레임 에지, 즉 두 영상에서 움직임이 있는 객체의 에지를 말한다. 움직임 에지 추출방법은 세 번째 단계에서 추출된 차이 영상의 에지와 다음 프레임 영상의 에지를 비트단위 논리합 알고리즘을 사용하여 추출한다. [식 5]는 움직임 에지 추출 방법에 대한 수식이다.

[식 5]

ME(MovingEdge)=(▽G*|F(baseFrame)|

|((morphology) ▽G*|Fn|)

[식 5]는 기준 프레임의 에지 영상과 다음 프레임의 에지 영상에 비트단위 논리곱을 수행하는 것을 보여준다. 여기서 움직임 에지를 정확하게 찾기 위하여 다음 프레임에 수리 형태 불림 연산을 적용하였다.

마지막 단계는 추출된 움직임 에지로부터 VOP(Video Object Plane)를 생성하는 단계이다. 추출된 에지를 수평과 수직으로 각각 에지 영역안에 픽셀 값을 채우고 두 수평과 수직에 대하여 논리곱 연산을 수행하였다. 여기서 정확한 객체를 찾기 위해 VOP 추출 전에 수리 형태 녹임과 불림 연산을 적용하였다. 그림 14는 검출된 움직임 에지와 전체 VOP 추출 과정을 보여준다.

## 4. 실험 결과

본 논문에서의 실험은 압축율과 VOP 추출의 실행 시간 분석으로 나누었다.

먼저 압축율 실험은 화상카메라로 입력받은 실제 동영상으로 QCIF 포맷의 176 * 144 사이즈이고 총 2000 프레임을 부호화 하였다. 또한 입력되는 영상은 크게 움직임이 많은 영상과 움직임이 적은 영상으로 구분하여 압축율을 나타내었다. 입력 영상은 사이즈에 따라 차이 픽셀의 발생이 적으므로 움직임 감지에 입력되는 민감도는 20.0을 설정하였고, 감지점은 5000.0으로 설정하였다.

실험 결과는 프레임 부호화를 통한 압축율과 이진 형상 부호화 통한 압축율을 나타내었다.

Rectangle frame 부호화의 실험 결과는 다음과 같다.

표 4와 같이 움직임이 많은 영상은 전체적으로 많은 압축 비트가 발생하였다. 일반적인 사각형 프레임 부호화 실험 결과는 MS-FDAM모델에서는 최대 40:1 정도의 압축율을 보였고, 제안된 알고리즘은 민감도의 조절에 따라 최대 71:1 정도의 압축율을 보여주었다.

Binary frame 부호화의 실험 결과는 다음과 같다.

표 5의 이진 형상 부호화는 비디오 객체 분할 모듈에

표 4 Rectangle frame 부호화

| Sequence | MS-FDAM /Proposed | Sensitive Level / Detection Level | Input Data | Compression Data | Compression Ratio |
|---|---|---|---|---|---|
| Rectangle 176 * 144 2000 frame Active Movement Image | MS-FDAM | NO | 72.5MB | 3.00MB | 24 : 1 |
| | Proposed | 5/5000 | 72.5MB | 1.89MB | 38 : 1 |
| | Proposed | 10/5000 | 72.5MB | 1.47MB | 49 : 1 |
| | Proposed | 20/5000 | 72.5MB | 1.12MB | 64 : 1 |
| Rectangle 176 * 144 2000 frame Small Movement Image | MS-FDAM | NO | 72.5MB | 1.70MB | 42 : 1 |
| | Proposed | 5/5000 | 72.5MB | 1.36MB | 53 : 1 |
| | Proposed | 10/5000 | 72.5MB | 1.10MB | 65 : 1 |
| | Proposed | 20/5000 | 72.5MB | 1.02MB | 71 : 1 |

표 5 이진 형상(binary shape) 부호화

| Sequence | MS-FDAM /Proposed | Sensitive Level / Detection Level | Input Data | Compression Data | Compression Ratio |
|---|---|---|---|---|---|
| Binary 176 * 144 2000 frame Active Movement Image | MS-FDAM | NO | 72.5MB | 1.96MB | 37 : 1 |
| | Proposed | 5/5000 | 72.5MB | 1.42MB | 51 : 1 |
| | Proposed | 10/5000 | 72.5MB | 1.02MB | 71 : 1 |
| | Proposed | 20/5000 | 72.5MB | 925.0KB | 80 : 1 |
| Binary 176 * 144 2000 frame Small Movement Image | MS-FDAM | NO | 72.5MB | 1.34MB | 54 : 1 |
| | Proposed | 5/5000 | 72.5MB | 550.0KB | 134 : 1 |
| | Proposed | 10/5000 | 72.5MB | 463.7KB | 160 : 1 |
| | Proposed | 20/5000 | 72.5MB | 412.3KB | 180 : 1 |

표 6 알고리즘 수행시간

| 알고리즘 | 실험 환경 | Sequence | 실행 시간 (m sec) | 프레임 수 | 초당 프레임 수 |
|---|---|---|---|---|---|
| 개발된 알고리즘 | 펜티엄4 2.0 | 움직임이 많은 영상 (176*144) | 28968 | 101 | 3.48 |
| | | 움직임이 적은 영상 (176*144) | 27734 | 101 | 3.79 |

의해 추출된 VOP 객체를 부호화하기 때문에 표 4의 프레임단위 부호화 보다 상대적으로 높은 압축률을 보여준다. MS-FDAM에서는 최대 54:1 정도의 압축율을 보이고 있으나 개발된 알고리즘은 최대 180:1 정도의 고압축을 보여주었다.

다음 표 6은 VOP 추출 알고리즘의 수행시간 실험결과이다.

실험 결과 표 6에서와 같이 객체 분할은 펜티엄-IV 2.0에서 총101 프레임을 약 27초에 처리하였다. 즉 이전 MS-FDAM과 같은 수동으로 영상분할과정을 수행하는 알고리즘보다 부호화 처리 지연 시간 없이 효율적인 형상 부호화를 수행하였다.

본 논문에서 개발된 MPEG-4 부호화의 프로그램 수행 과정과 인터페이스는 다음 그림 15와 같다.

1. 초기화면

그림 15의 화면은 개발된 프로그램의 전체 구조이다. 1번 사각형은 화상 카메라나 파일 입력을 보여준다. 2번 가운데 검은색 사각형은 움직임 감시 상태를 보여준다. 3번 사각형은 부호화 된 이후 재구성된 영상을 보여준다. 4번 부분은 부호화되는 프레임수와 두 영상간의 차이 픽셀 수를 보여준다. 5번 부분은 VOP 추출 영상을 보여준다. 6번 부분은 전체 프레임 수와 현재 감지된 영상 수를 보여준다. 7번은 민감도와 감지점을 입력하는 곳이다. 8번 부분은 영상의 사이즈 입력과 샘플링율, 부호화 모드 및 부호기와 복호기, 영상분할을 설정한다. 9번 버튼은 초기입력 데이타를 설정하는 버튼이다.

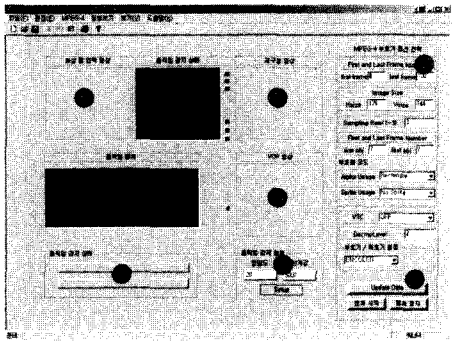


그림 15 초기화면

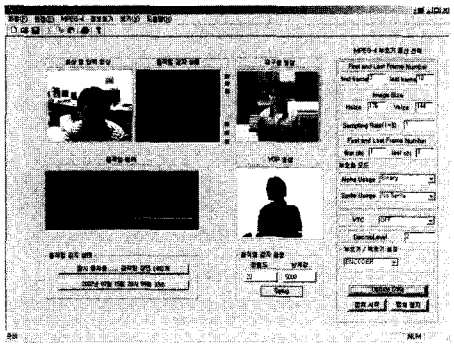


그림 16 객체 분할과 이진 형상 부호화

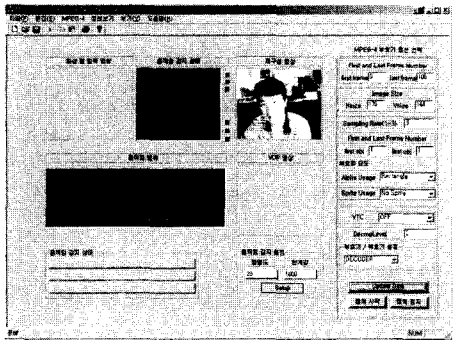


그림 17 영상 데이타 복호화 과정

2. 객체 분할과 이진 형상 부호화

그림 16은 이진 형상 부호화를 보여준다. 그림 15의 초기화면과 같이 그림 16의 1번은 화상카메라 입력 3번

은 이진 형상 부호화 후의 재구성 영상 5번은 VOP 형상을 보여준다.

3. 복호화 과정

그림 17은 압축된 데이타의 복호화 수행 과정을 보여준다.

## 5. 결론 및 향후 계획

본 논문에서는 실시간 고압축 MPEG-4부호화와 복호화 시스템을 개발하였다. 개발된 MPEG-4 시스템은 실용적인 소프트웨어 사용을 위해 실시간 카메라 시스템을 적용하였고, 고효율의 압축을 위해 움직임 감지 알고리즘을 적용하였다. 또한 MPEG-4의 핵심 기술인 이진 형상 부호화를 자동으로 처리하기 위해 비디오 객체 분할 모듈을 추가하였다.

본 연구에 의해 개발된 MPEG-4 전처리 시스템은 최대 180:1의 압축률을 보였다. 이러한 높은 압축율은 멀티미디어 환경에서 효율적인 비디오 처리를 가능하게 할 것이다. 즉, 내용기반 부호화에 의하여 높은 압축율을 적용하더라도 영상의 손상을 최소화할 수 있다. 이러한 장점으로 인하여 고성능 멀티미디어 통신 서비스가 가능하다. 개발된 MPEG-4 비디오 전처리 시스템은 인터넷 디지털 컨텐츠, 인터넷 TV, 영상 통신, 영상 회의, 감시 카메라 등 뉴미디어 관련 분야에 광범위하게 적용할 수 있을 것이다.

## 참 고 문 헌


[1] ISO/IEC 14496(MPEG-4) Video Reference Software, Version: Microsoft-FDAM1-2.3-001213.

[2] ISO/IEC/JTC1/SC29/WG11, MPEG/N3908, "MPEG-4 Video Verification Model version 18.0," Jan. 2001/Pisa.

[3] ISO/IEC 14496 Version 1 Part 2 Visual.

[4] ISO/IEC 13818-2, "Information technology-Generic coding of moving pictures and associated audio information-Video," 1994.

[5] T.Sikora, "The MPEG-4 video standard verification model," IEEE Trans. Circuits Syst. Video Technology, vol.7, pp.19-31, Feb. 1998.

[6] J.G.Choi, M.Kim, M.H.Lee, and C.Ahn, "Automatic segmentation based on spatio-temporal information," IEEE Trans. Circuits Syst. Video Technol., vol.8, pp.525-538, Sept. 1998.

[7] T.Meier and K.N.Ngan, "Automatic segmentation of moving objects for video object plane generation," IEEE Trans. Pattern Anal. Machine Intell., vol.15, pp.525-538, Setp, 1998.

[8] Ju Guo et al., "Fast and accurate moving object extraction technique for MPEG-4 object-based video coding," SPIE, vol.3653, pp.1210-1221, January, 1999.

[9] P.Bouthemy and E.Francois, "Motion segmentation and qualitative dynamic scene analysis from a image sequence," Int. J. Computer Vision, vol,10, no.2, pp.157-182, 1993.

[10] J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Machine Intell., vol.PAMI-8, pp.679-698, Nov. 1986.



김 준 기

1992년 3월~1998년 2월 호서대학교 전자계산학과 이학사 졸업. 1998년 3월~2000년 2월 호서대학교 컴퓨터공학부 대학원 공학석사 졸업. 2000년 3월~2001년 2월 호서대학교 컴퓨터공학부 대학원 공학박사 수료. 2002년 3월~현재 호서대학교 컴퓨터공학부 강사. 관심분야는 영상처리, 영상분할 MPEG-4, MPEG-7

홍 성 수

1974년 3월~1979년 2월 광운대학교 전자계산학과 공학사 졸업. 1979년 3월~1981년 2월 쌍용양회 기획실 근무. 1982년 3월~1984년 2월 광운대학교 전자계산학과 대학원 공학석사 졸업. 1984년 3월~1990년 8월 광운대학교 전자계산학과 대학원 공학박사 졸업. 1990년 9월~현재 호서대학교 컴퓨터공학부 교수. 관심분야는 영상에니메이션, 영상처리, 멀티미디어 컨텐츠 개발

이 호 석

1979년 3월~1983년 2월 서울대학교 전자계산기공학과 공학사 졸업. 1983년 3월~1985년 2월 서울대학교 컴퓨터공학과 대학원 공학석사 졸업. 1989년 3월~1993년 8월 서울대학교 컴퓨터공학부 대학원 공학박사 졸업. 1994년 3월~현재 호서대학교 컴퓨터공학부 교수. 관심분야는 영상처리, 웨이블릿, MPEG-4, MPEG-7, 프로그래밍 언어