실시간 고압축 MPEG-4 부호화를 위한 비디오 객체 분할과 프레임 전처리

김 준 기*, 정회원 이 호 석**

Video object segmentation and frame preprocessing for real-time and high compression MPEG-4 encoding

Jun-Ki Kim*, Ho-Suk Lee** Regular member

요 약

비디오 객체 분할(Video Object Segmentation)은 MPEG-4 부호화의 핵심기술로 실시간 요구사항을 위해 빠르고 정확하여야 한다. 그러나 대부분의 존재하는 알고리즘은 계산량이 많으며 실시간 응용을 위해 적합하지 않다. 또한 이전 MPEG-4 VM(Verification Model) 기본 모델은 MPEG-4 부호화 처리를 위한 기본 알고리즘을 제공하였으나 실시간 요구사항을 위한 카메라 입력 시스템, 실용적인 소프트웨어 개발, 비디오 객체 분할 그리고 압축 효율에 많은 제한이 있다. 이에 본 논문은 기본 MPEG-4 VM모델에 내용 기반 비디오 코딩의 핵심인 VOP 추출 알고리즘, 실시간 카메라 입력 시스템, 압축율을 높일 수 있는 움직임 감지 알고리즘을 추가하여 최대 180:1의 압축율을 보여주는 실시간 고압축 MPEG-4 전처리 시스템을 개발하였다.

주요어 : 비디오 객체 분할, MPEG-4, 실시간, 고압축, 전처리 시스템

ABSTRACT

Video object segmentation is one of the core technologies for content-based real-time MPEG-4 encoding system. For real-time requirement, the segmentation algorithm should be fast and accurate but almost all existing algorithms are computationally intensive and not suitable for real-time applications. The MPEG-4 VM(Verification Model) has provided basic algorithms for MPEG-4 encoding but it has many limitations in practical software development, real-time camera input system and compression efficiency. In this paper, we implemented the preprocessing system for real-time camera input and VOP extraction for content-based video coding and also implemented motion detection to achieve the 180:1 compression rate for real-time and high compression MPEG-4 encoding.

Keyword : video object segmentation, MPEG-4, real-time, high compression, preprocessing

Ⅰ. 서 론

MPEG-4 비디오 표준은 오디오 비디오 데이터의 이식성과 범용 접근성, 고비율 압축, 비디오 데이터의 상호작용을 고려하여 멀티미디어 응용에서 비디오 코딩을 위한 표준으로 설계되었다. 이것은 비디오 회의나 비디오 폰 같은 많은 실시간 비디오 시스템에 적용되었다.

MPEG-4의 중요 기능은 내용 기반 부호화(content based coding) 내용이다[1][3]. 기반 부호

* 호서대학교 컴퓨터공학과 멀티미디어 연구실(kjk73@hanmail.net), ** 호서대학교 컴퓨터공학과(hslee@office.hoseo.ac.kr)
논문번호 : 020446-1014, 접수일자 : 2002년 10월 14일
※본 연구는 2002년도 호서대학교 학술연구조성비에 의하여 연구되었습니다.

화는 형상(shape), 움직임(motion), 텍스쳐(texture)와 같은 비디오 객체(Video Object)의 혼합된 장면으로 나타낸다. 이러한 작업에서는 객체와 같은 형상 정보가 필요해지며 이 형상 정보는 비디오 시퀀스의 알파 평면(alpha plane)으로 저장된다. 즉 비디오 객체는 비디오 시퀀스로부터 비디오 객체 형상을 발생하는 중요한 방법이 되어진다.

자동 객체 분할은 MPEG-4 실시간 카메라 시스템에서 매우 중요한 역할을 한다. 만약 형상 정보를 즉시 발생하는 자동 분할 알고리즘이 없다면 그 시스템은 입력되는 시퀀스에 기반한 객체 기반 비디오 부호화를 적용할 수가 없다. 그리고 대부분의 효과와 기능을 MPEG-4가 잃어버린다.

객체 분할 처리는 움직임 추정, 움직임 보상, 텍스쳐 코딩 전에 끝난다. 그러므로 자동 객체 분할을 위한 두 가지 요구사항은 먼저 분할은 매우 빨라야 한다. 적어도 객체 분할 처리는 초당 10프레임정도는 처리되어야 하며 두 번째로 이러한 알고리즘은 좋은 결과를 보여주어야 한다. 나쁜 결과는 MPEG-4 비디오의 질을 떨어지게 한다[7][8].

MPEG-4 비디오 그룹에서는 실험을 통하여 비디오 검증 모델인 VM(Verification Model)을 개발하였다. VM은 부호화뿐만 아니라 복호화 알고리즘을 정의하였다. VM 버전 1에서는 내용 기반 객체 부호화, 시간적 스케일러빌러티, 공간적 스케일러빌러티, 에러내성, 압축효율등 다양한 기능을 소개하였다. 더욱더 VM 버전2에서는 기존의 알고리즘을 향상 시켰으며, 인터넷 비디오 스트리밍 서비스를 위한 새로운 코딩방법을 추가하였다. MPEG-4 비디오 VM 모델의 전체적인 특징은 향상된 압축효율, 오류내성, 형상(shape)과 알파맵(alpha map) 코딩, 임의 형상 텍스쳐(texture) 코딩, 다양한 부호화를 지원하는 다기능 코딩툴과 알고리즘 그리고 FGS(Fine Granularity Scalability) 코딩 등이 있다. 위와 같은 특징에도 불구하고 MPEG-4 VM은 실용적인 소프트웨어 개발과 압축효율에 많은 제한이 있다. 우선 VM은 MPEG-4의 전처리 단계의 핵심 기술인 비디오 객체 분할 알고리즘이 구현 되어있지 않다. 또한 VM에서는 모든 기능을 배제하고 단지 비디오 처리부분만 나타내었다. VM에서는 상업적 구현을 위한 실용적인 소프트웨어 기능을 배제하고 있다. 프레임처리에 있어 오직 4:2:0의 비디오 포맷만을 지원하며 후처리 필터(filter) 부분이 없다. 특히 부호화, 복호화를 위한 불명확한 요구사항이 많이 있다[2][3][4][5].

이에 본 논문에서는 기본적인 MPEG-4 VM 모델에 실시간 처리가 가능한 화상카메라 입력과 파일 입출력 인터페이스를 새롭게 구현하였으며, 더욱 고효율 압축을 위해 움직임이 적은 프레임에서는 부호화를 수행하지 않는 움직임 감지 알고리즘을 추가하였다. 또한 MPEG-4 부호화의 핵심 기술인 객체 기반 부호화를 위해 반 자동, 자동 비디오 객체 분할 알고리즘을 추가하였다. 입출력 인터페이스는 사용자의 선택에 따라 화상카메라가 지원하는 표준규격인 IYUV, YUV2, RGB 24를 지원하며, 프레임 사이즈는 176*144의 QCIF 포맷을 지원한다. 또한 다른 멀티미디어 기기에서 저장된 동영상 파일을 입출력 할 수 있도록 개선하였다. 움직임 감지 알고리즘은 현재 프레임과 이전프레임 사이의 픽셀 값의 차이(difference)를 이용하여 일정한 감지점 이하를 취하면 부호화를 처리 하지 않도록 구성하였다. 비디오 객체 분할은 시간적 알고리즘으로 두 영상의 차이를 쉽고 빠르고 정확하게 추출할 수 있는 프레임 차이를 사용하였고, 공간적 알고리즘으로는 영상 자체의 구분을 위한 Canny 에지 검출 알고리즘을 사용하였다. 이후 두개의 알고리즘에서 추출된 객체를 조합하여 VOP를 추출 하였다.

본 논문에서는 일반적인 MPEG-4 실험 비디오 시퀀스와 화상카메라를 이용하여 실시간으로 입력 받은 시퀀스를 사용하여 실험하였다. 개발된 객체 분할은 정확한 객체의 경계를 보여주었고, 객체를 사용한 MPEG-4 객체 기반 부호화는 최대 180:1의 고효율 압축률을 나타내었다.

본 논문의 구조는 다음과 같다. 2장에서는 관련 연구분야와 MPEG-4 VM 알고리즘을 소개하며 3장에서는 제안한 알고리즘, 실시간 카메라 입력 모듈, 움직임 감지 알고리즘, 적용된 비디오 객체 분할구조를 소개하였다. 4장에서는 실험 결과와 사용자 인터페이스를 제시하고 결론을 맺었다

Ⅱ. 관련 연구 및 MPEG-4 VM 알고리즘

1. VOP 추출 알고리즘

현재 존재하는 비디오 객체 분할 알고리즘은 4가지 범위로 나누어진다. 수리형태학 분수계 분할 알고리즘 (morphological watershed), 칼라 분할(color segmentation), 변환 검출 마스크(change detection mask) 그리고 하이브리드(hybrid) 알고리즘이다. 수리형태학 분수계 알고리즘은 이미지 분할을 위한 툴로 유용하다. 이것은 또한 비디오 분할에 널리 사용되어지고 있다. 분수계 알고리즘 적용 이후는 분할된 이미지를 동일한 지역으로 모으는 과정이 필요하다. 이 지역은 확장된 움직임 정보로 병합되어진다. 이것은 일반적으로 움직임 측정이나 광학적 흐름(optical flow)에 의해 발생되어진다. 유사하게 칼라 분할은 많은 칼라 지역으로 나누어진 이미지 공간을 칼라맵또는 칼라 공간의 특징을 분석하는데 이용되어진다. 다음단계는 움직임 측정으로 움직임 정보를 이용하여 지역을 병합하는 것이다. 움직임 추정에 의한 알고리즘은 실시간 응용에서 많은 계산량이 필요해진다. 비디오 분할의 세 번째 종류는 변환 검출에 기반한 것이다. 이것은 변화검출 마스크에 의해서 발생되는 연속적인 프레임의 차이이다. 또한 저 사양 컴퓨터만으로도 변환 검출을 수행할 수 있다. 그러나 이것은 몇몇 상황에 적당하지 않다. 처음으로 변환 검출 마스크는 노이즈에 민감하다. 두 번째로 변환검출은 객체의 그림자나 빛 변화에 민감하다. 객체가 움직였을 때 변환검출 마스크는 겹치지 않는 지역으로 맞물린다. 이것은 실질적으로 배경지역의 부분이다. 객체가 정지하고 있을 때 다른 한편으로 변환 검출 마스크는 대부분의 객체를 잃어버린다는 것이다. 몇몇 방법은 이러한 문제를 다루는 방법을 개발하였다. 그러나 그것은 복잡한 연산없이 그림자나 빛 변화를 다루기 힘들다. 비디오 분할의 마지막 종류는 하이브리드 알고리즘에 기반한다. Mieier and Ngan's 알고리즘 또는 M.Kim's 의 두 알고리즘은[8][10] 계산량의 증가와 복잡하기 때문에 실시간 MPEG-4 시스템에 통합하기가 어렵다.

본 논문에서는 새로운 효과적인 알고리즘을 제안하였다. 이 알고리즘 기초적인 변환 검출을 이용한다. 본 알고리즘은 움직임 추정이나 공간분석 특징과 같은 계산적인 증가량이 없다. 변환검출과 배경 저장, 수리 형태학 알고리즘을 이용한 노이즈 지역 제거는 빠르고 강력하고 정확한 결과를 실시간 응용에서 보여줄 수 있다.

2. MPEG-4

MPEG-4의 기본적인 목표는 고효율 고압축 부호화의 추구이다. MPEG-4 비디오 부호화에서는 부호화 효율 개선을 위하여 8x8 블록 움직임 보상, 직접(direct)예측, AC/DC 계수의 예측등 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 데이터 전송중 외부 노이즈에 대한 데이터 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, 데이터 파티션닝(data partitioning), 스터핑 바이트(stuffing byte)의 개선등이 행해졌다[1][3][5].

마지막으로 MPEG-4 특징은 멀티미디어 대응이다. 이것은 종래의 자연 비디오, 오디오등의 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.

3. 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-scale 과 같은 3가지 종류로 나누어진다.

MPEG-4 VM 움직임 보상은 임의의 형상 VOP를 이용하기 때문에 VOP 경계에 걸치는 매

크로블록이 존재한다. 이를 해결하기 위하여 새롭게 패딩(padding)과 다각형 매칭(polygon matching) 기술을 이용한 움직임 보상이 수행된다. 또한 움직임 벡터의 탐색에는 제한없는 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[1][3][11].

Ⅲ. 제안 알고리즘

제안된 MPEG-4 비디오 전처리시스템의 전체 구조와 비디오 객체 분할 모듈은 <그림 1>과 같다.

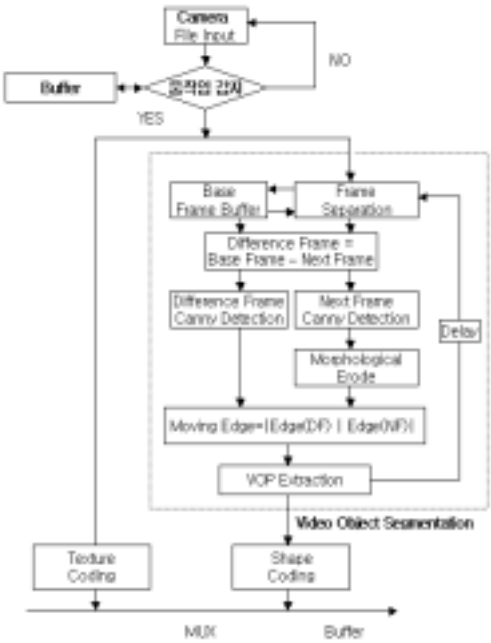

그림 1. MPEG-4 전처리 시스템 및 비디오 객체 분할 모듈

본 논문은 MPEG-4 부호기의 실용적인 소프트웨어 개발과 고효율 고압축율 상승에 초점을 맞추었다. 또한 시간-공간영역의 복잡도를 줄이고 빠르고 적응성이 강한 자동, 반자동 비디오 객체 분할 알고리즘을 적용하였다. 이전 MPEG-4 VM 기본 모델은 윈도우 환경의 GUI(Graphic User Interface)가 아니라 콘솔 모드(console mode)로 영상 데이터의 입력과 출력 및 부호화를 수행하였다. 부호화 과정은 영상 데이터를 입력하는 부분부터 부호화에 필요한 모든 파라미터 설정을 텍스트 파일을 사용하여 처리하였다. 때문에 영상 데이터를 부호화함에 있어 사용자에게 요구되는 부담이 가중되었다. 또한 실험에 필요한 영상은 다른 영상기기에서 만든 고정된 데이터를 사용하였다. 이에 실용적인 소프트웨어로 사용할 수 있는 실시간 카메라 시스템과 같은 적용범위가 제한되어 있다. 더욱 VM모델 자체에 많은 메모리 누수가 있어 운영체제를 다운 시키는 현상도 발생하였다. 본 논문은 이러한 단점을 개선하고 실용적인 소프트웨어를 위하여 모든 영상 데이터 처리는 GUI환경에서 동작하며 프로그램상에 메모리 누수를 방지하여 안정된 영상 데이터 처리가 가능하였다. 실험 영상은 적용범위를 늘려 실용적인 소프트웨어를 위한 실시간 입출력 모듈을 적용하였다. 실시간 입출력 모듈은 화상 카메라 입력 영상뿐만 아니라 다른 멀티미디어 기기에서 만든 영상과 파일을 처리 할 수 있도록 하였다.

고효율 고압축율 상승을 위한 방법으로는 움직임 감지 기법을 사용하였다. 즉 동영상 데이터의 유사성을 고려하여 연속적인 동영상 프레임 사이에서 움직임 변화를 검출하였다. 이전 MPEG-4 VM 기본 모델은 유사한 모든 영상을 부호화하기 때문에 압축 데이터양과 부호화 시간이 증가하였다. 본 논문에서는 움직임이 아주 미세한 부분을 감지하여 불필요한 영상데이터는 부호화를 수행하지 않았다. 이에 이전 VM 보다 영상 데이터의 처리 시간과 고효율 고압축율의 효과를 가져왔다. 또한 MPEG-4 핵심기술인 이진 형상기반 부호화와 그레이 스케일 형상 기반 부호화의 효율성을 고려하여 전처리 단계인 자동 객체 분할 알고리즘을 적용하였다.

실시간 입출력은 <그림 1>과 같이 사용자 정의에 따라 화상 카메라 입력과 파일입력으로 분할하였다. 화상 카메라 입력은 실시간 처리의 기본으로 입력되는 영상과 부호화 영상을 따로

처리하는 병렬 쓰레드 모듈로 구현하였다. 각 영상의 입력 포맷은 기존 VM 모델이 오프라인 형태의4:2:0 파일 포맷만을 지원하는데 반에 본 논문에서는 화상카메라의 입력 형태에 따라 4:2:0, 4:2:2, 그리고 4:4:4 포맷을 지원하였다. 또한 오프라인 형태의 파일 입출력을 통하여 MPEG-4의 기본 입출력 모드인 합성 영상 및 자연 영상을 모두 처리 할 수 있도록 하였다.

움직임 감지 모듈은 화상 카메라와 파일을 통해 연속적으로 입력되는 프레임사이의 픽셀을 비교하였다. <그림 1>에서와 같이 버퍼에 저장된 영상과 입력되는 영상을 비교하여 입력되는 영상과 이전 영상과의 차이를 비교하여 일정한 감지점 이하의 값이 산출될 경우 부호화 과정을 스킵(skip)하도록 하였다. 더욱더 효율적인 압축을 위하여 민감도와 감지점을 입력하여 유동적으로 압축률을 조절하였다. 즉 이전 프레임과 현재 프레임을 비교하여 설정된 민감도와 감지점이하의 픽셀 차이를 보이면 현재 프레임을 스킵하는 형태로 중복되는 부호화 과정을 줄였다.

객체 분할 알고리즘은 실시간 처리에 적용할 수 있도록 구현 하였다. 즉 계산량을 더욱 줄여 부호화 처리의 지연 시간을 향상 시켰다. 이전에 언급 하였듯이 MPEG-4의 효율적인 부호화 과정은 일반적인 프레임 압축뿐만 아니라 객체를 이용하는 형상 부호화 알고리즘에 초점을 맞추었다. MPEG-4 VM 기본 모델에서는 이러한 형상 부호화를 수행하기 위해서 두 가지 작업이 필요하다. 우선 부호화하고자 하는 영상을 준비해야 하며, 다음으로 알파맵을 만드는 영상 분할 과정을 수동으로 만들어주어야 한다. 즉 부호화하고자 하는 영상과 알파맵 영상을 통하여 이진 형상 부호화 및 그레이 스케일 형상 부호화를 수행하였다. 이에 본 논문에서는 이러한 과정을 자동으로 변경하였으며 부호화하고자 하는 하나의 입력 영상으로 내부적으로 자동 객체 분할을 통하여 알파맵을 작성하고 이에 따라 실시간으로 형상 부호화를 수행하여 효율적인 부호화 처리가 가능하게 하였다.

1. 실시간 MPEG-4 입력 모듈

실시간 입력 모듈을 위한 병렬 쓰레드 구성은 <그림 2> 와 같다.

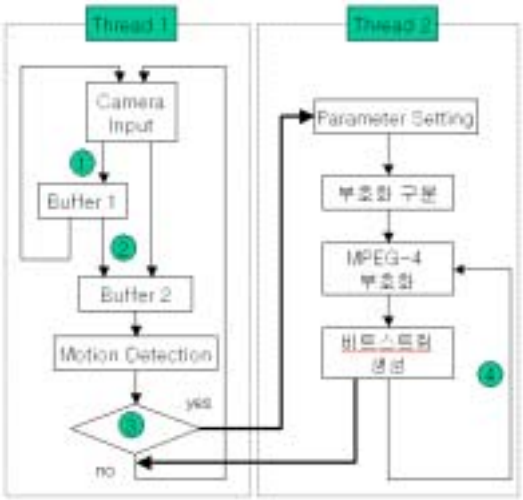

그림 2. 병렬 쓰레드 구성 및 입력 모듈

동영상 관련 데이터는 최근 하드웨어의 급속한 발전에도 불구하고 영상 데이터 처리에 많은 계산을 필요로 해 높은 CPU 점유율을 차지한다. 이전 VM 모델의 입력 시스템은 단일 처리 형태로 구성되어 있다. 때문에 영상 데이터 입력하여 출력 결과를 얻기까지 많은 시간이 소비되며 사용자의 부담이 가중하였다. 또한 이러한 단일 구성은 소프트웨어형태로 처리하기에는 많은 제한이 있다. 더욱 실시간 입출력 부분의 시간적인 지연은 전체적으로 시스템 성능을 저하시키는 결과를 낳았다. 이에 본 논문에서는 입력과 데이터 부호화 모듈을 병렬 쓰레드로 구성하여 전체적인 속도 향상과 소프트웨어만으로도 효율적인 데이터 처리가 가능하도록 하였다.

병렬 쓰레드 구성은 <그림 2>와 같이 쓰레드-1과 쓰레드-2로 구성된다. 쓰레드-1의 모듈은 화상 카메라 입력 포맷을 설정하고 연속해서 영상을 카메라로부터 입력 받아 버퍼에 저장하는 기능 및 저장된 영상을 통하여 움직임을 감지하는 기능으로 구성된다. 쓰레드-2에서는 움직임이 발생된 영상 데이터를 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>의 쓰레드-1 모듈에서 1번 과정은 부호화를 수행했을 때 처음으로 영상 데이터를

버퍼에 입력하는 루틴이다. 이렇게 하는 이유는 움직임 감지 및 객체 분할 모듈를 위해 우선 이전 영상 데이터가 필요하기 때문이다. 이후 다시 두 번째 영상 입력을 위해 2번 과정을 수행한다. 1번과 2번의 수행이후 두개의 버퍼는 이전 영상과 현재 영상의 데이터를 갖고 있다. 이 데이터를 이용하여 <그림 2>와 같이 움직임 감지 계산을 수행한다. 여기서 3번은 움직임을 감지하여 움직임이 발생하면 쓰레드-2의 모듈로 영상데이터를 보내고 그렇지 않으면 최근에 입력 받은 영상을 버퍼에 저장하고 다음 영상을 입력받아 계속해서 움직임 감지를 수행한다. 움직임 감지가 발생하면 쓰레드-2번 부분의 4번 과정인 부호화를 수행한다. 부호화가 끝나면 다시 3번으로 제어가 옮겨져 부호화가 끝날 때까지 계속해서 반복한다.

2. 고압축을 위한 움직임 감지 처리

움직임 감지는 영상 데이터를 고효율로 압축할 수 있는 핵심 기술이다. 이전 VM에서는 MPEG-4 실험 영상인 "Mother & daughter", "Akiyo", "Hall Monitor"등의 모든 프레임을 압축하였다. 이는 움직임이 없는 부분도 압축을 수행함으로서 압축 데이터의양을 증가시켰다. 본 논문에서 적용된 움직임 감지 기법은 사용자 정의에 따라 민감도와 감지점을 입력받아 부호화하고자 하는 영상의 차이를 측정하여 일정한 감지점 이하의 값이 발생하면 부호화 과정을 스킵하고 그 이상의 값이 발생하면 부호화 과정을 수행하여 부호화되는 영상의 압축율을 높였다. 움직임 감지를 위한 핵심적인 수식은 다음과 같다.

[식1-1]
if (CIP PIP > threshold)
PDC = PDC + 1 otherwise PIP = CIP

[식1-2]
if (PIP CIP > threshold)
PDC = PDC + 1; otherwise PIP = CIP

[식1-3]
if(PDC > theta)
Flag = 1 otherwise Flag = 0

PDC : Pixel Difference Count
CIP : Current Image Pixel
PIP : Previous Image Pixel
threshold : 민감도 , *theta :* 감지점

움직임 감지란 두 영상 사이에서 차이를 발견하는 것이다. 움직임 감지를 위한 방법으로는 변화 검출 마스크(change detection mask), 유사 움직임 검출(affine motion detection), 광학적 흐름 알고리즘(optical flow algorithm), 전 탐색 블록 매칭 알고리즘(full search block matching algorithm)등이 있다. 변환 검출 마스크는 쉽게 구현되고 빠르게 움직임 차이를 찾을 수 있는 장점에 비하여 아주 미세한 움직임 차이에 대해서는 정확한 측정이 불가능한 단점이 있다. 유사 움직임 검출은 각 파라미터(6파라미터, 8파라미터, 12파라미터)에 따라 전체적인 카메라 움직임을 측정하는 것으로 미세한 움직임 차이를 찾는데 유리하지만 다른 움직임 검출 알고리즘의 도움 없이는 정확한 움직임 차이를 찾을 수 없는 단점과 각 픽셀에 대한 벡터를 찾기 때문에 많은 시간적 낭비를 갖는다. 또한 광학적 흐름 알고리즘과 전 탐색 블록 매칭 알고리즘의 가장 큰 단점은 시간적 낭비이다. 이에 본 논문에서는 실시간 처리를 고려하여 시간적 낭비를 줄이고 유동적으로 움직임 차이를 취할 수 있도록 [식 1-1], [식 1-2], [식 1-3]과 같이 임계값을 지정하는 형식을 적용하였다.

[식1-1]은 현재 영상 픽셀에서 이전 영상 픽셀을 빼는 과정으로 그 결과값이 민감도 보다 크다면 전체 픽셀 차이 수를 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 커지며 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.

본 논문에 적용된 움직임 감지 구조는 <그림 3> 과 같다.

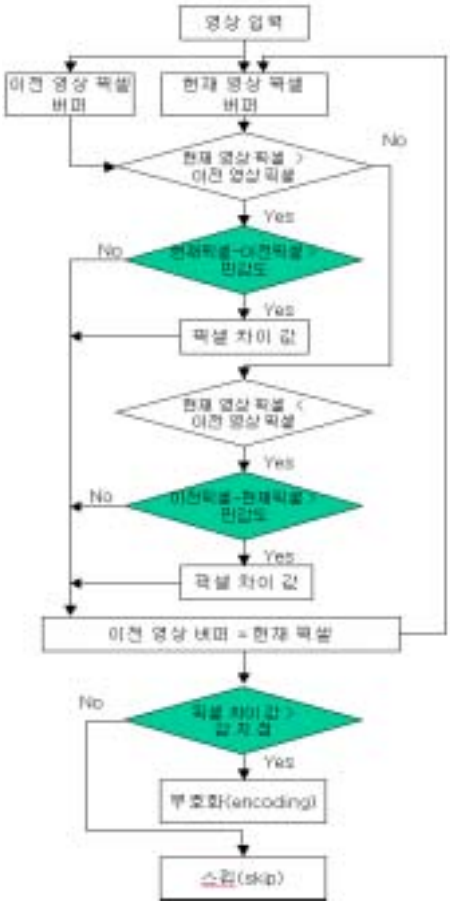

그림 3.움직임 감지 구조

<그림 3>과 같이 파일이나 화상 카메라로 입력되는 영상은 이전 영상 픽셀을 갖는 버퍼에 저장된다. 이후 현재 영상을 입력 받아 현재 영상 픽셀과 이전 영상 픽셀을 비교한다. 만일 현재 영상 픽셀 값이 이전영상 픽셀 값보다 크면 다음 단계인 현재 픽셀에서 이전 픽셀 값의 뺄셈을 통하여 나온 결과 값과 사용자가 입력한 민감도 값과 비교한다. 만일 민감도 값에 비해 두 영상에서 픽셀 차이가 크다면 두 영상의 차이가 존재한다는 픽셀 차이 값을 증가한다. 픽셀 차이 값이란 두 영상을 비교하여 차이가 나는 모든 픽셀의 수이다. 그렇지 않으면 현재 픽셀 값을 이전 영상 버퍼에 저장한다. 즉 이전 영상 버퍼는 연속적인 움직임 감지 영상의 기초 영상으로 사용되어진다. 다음으로 입력되는 새로운 영상은 업데이트된 이전 영상 버퍼에 있는 데이터를 이용하여 새롭게 움직임 감지를 수행한다. 영상 비교는 현재 영상 픽셀이 이전 영상 픽셀 보다 큰지를 비교하거나 반대로 이전영상 픽셀이 현재영상 픽셀보다 크다는 비교를 하였다. 이렇게 현재 영상 픽셀과 이전 영상 픽셀을 반대로 비교하는 이유는 영상 픽셀에서의 음수 값을 제한하고자 하기 때문이다. 민감도란 두 영상에서의 픽셀 차이크기를 보여준다. 만일 민감도 값을 아주 큰 값으로 설정하였다면 일반적인 프레임처리와 같이 수행되어진다. 즉 모든 프레임이 부호화 되어진다. 그와 반대로 민감도의 값을 작게 설정하면 많은 양의 프레임을 버리는(drop) 효과가 발생한다. 감지점이란 그림에서와 같이 마지막으로 움직임을 판단하는 곳이다. 즉 민감도는 두 영상의 전체 픽셀 차이를 얻을 수 있고 이를 이용하여 감지점 값을 설정하여 전체 픽셀의 수와 감지점의 값이 비교된다. 만일 감지점의 값이 전체 픽셀 수보다 작다면 부호화를 스킵하고 그렇지 않으면 MPEG-4 부호화과정을 수행한다.

3. 움직임 감지 처리 실험 결과

움직임 감지 알고리즘에 가장 중요한 부분은 민감도와 감지점에대한 임계값을 적용하는 과정이다. 민감도 값의 초기 설정은 0.0에서 100.0으로 하였다. 이러한 값을 설정한 이유는 동영상 부호화는 한 장의 영상을 부호화하는 것이 아니라 연속적인 영상 데이터의 상관 관계를 통하여 부호화 한다. 때문에 이전 영상과 현재 영상과의 관계는 유사한 형태를 보인다. 영상에서 하나의 픽셀이 갖는 값은 RGB일 경우 0부터 255까지의 값을 갖는다. 즉 이전영상의 픽셀 값과 현재영상의 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 나타낸 실험 결과를 <그림 4-3>에 나타내었다. 즉 사용자가 화상 카메라로 입력되는 영상의 특성에 따라 유동적으로 민감도를 낮추면 두 영상의 전체 차이 픽셀수가 작아지며, 민감도를 높이면 두 영상의 전체 차이 픽셀수가 커진다. 감지점은 민감도에 따라 적절한 범위의 임계값을 적

용하였다. 움직임 감지를 위한 실험 영상은 화상 카메라에서 적용될 수 있는 형태로 움직임이 많은 영상과 움직임이 적은 영상으로 실험하였다.

1) 화상 카메라는 고정되어 있고 객체가 움직이는 경우

A.움직임이 많은 영상
(예:"Hall monitor"영상)
B.움직임이 적은 영상
(예:"Akiyo"영상)

실험 영상은 RGB24 비트 포맷의 화상카메라 입력 영상으로 320*240사이즈 영상을 사용하였으며 전체 프레임 수는 2000프레임을 사용하였다. 민감도의 범위는 0.0부터 100.0까지 설정하였고, 감지점의 범위는 0부터 100000.0까지를 설정하여 유동적으로 실험하였다. <그림 4-1>에 보여지는 그래프는 가로 방향으로 100개의 프레임 수와 세로 방향으로 0.0 부터 100000.0 범위의 차이 픽셀 수를 보여준다. <그림 4-1>은 민감도를 20.0, 감지점을 25000.0으로 적용된 실험 결과를 보여준다. 즉 차이 픽셀 수가 25000.0의 값보다 크면 움직임이 검출되고 그 이하의 값이면 움직임을 스킵한다. 본 실험은 각각 2000프레임의 영상을 움직임 감지하였을 경우 차이 픽셀 수, 전체 프레임 영상 및 움직임 감지된 프레임 영상 수, 움직임 감지가 스킵된 드롭(drop)프레임 영상 수를 보여주었다.

1) 화상 카메라 고정되어 있고 객체가 움직이는 경우.

A. 움직임이 많은 영상

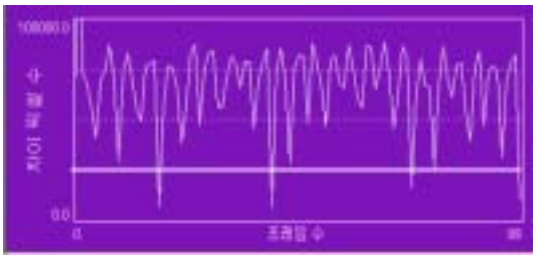

그림 4-1. 100 프레임에 대한 차이 픽셀 수와 프레임 수

감지점은 <그림 4-1>에서 이중 실선으로 표시하였다. <그림 4-1>는 0부터 100 프레임까지 두 영상 사이의 차이 픽셀수의 결과를 그래프로 보여준다. <그림 4-1>에서와 같이 움직임이 검출된 100프레임까지의 전체 차이 픽셀 수는 평균 66914.0의 높은 수치를 보였다.

B. 움직임이 적은 영상

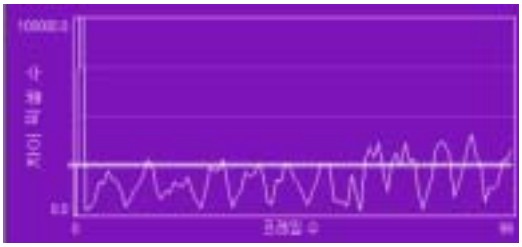

그림 4-2. 100 프레임에 대한 차이 픽셀 수와 프레임 수

<그림 4-2>은 움직임이 적은 영상에 대하여 민감도 20.0과 감지점 25000.0으로 설정하여 실험하였다. <그림 4-2>에서와 같이 대부분의 프레임이 감지점 이하로 발생하여 드롭되는 형태를 보여준다. <그림 4-2>에서와 같이 움직임이 검출된100프레임까지의 전체 차이 픽셀 수는 평균 10477.0의 낮은 수치를 보였다. <그림 4-3>는 2000프레임의 실험 영상에 대하여 민감도를 다르게 설정하였을 경우 전체 실험 결과를 보여준다.

시퀀스 (RGB24) 320*240	민감도/감지점	전체 프레임 수	움직임 감지 수	드롭 프레임 수
움직임이 많은 영상	20/25000	2000	1988	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 4-3. 실험 결과

그림 4. 화상카메라 고정, 객체 움직임

실험 결과 <그림 4-3>에서와 같이 움직임이 적은 영상에서 민감도 100.0과 감지점 25000.0의 설정은 최대 절반정도의 움직임 감지 스킵인 프레임 드롭을 확인 할 수 있었다.

4. 적용된 비디오 객체 분할 모듈

MPEG-4 비디오의 핵심은 높은 압축 효율이다. MPEG-4 그룹에서는 현존하는 알고리즘의 압축효율을 향상시키기 위해 다양한 방법을 제시하였다. 이전 압축기술은 영상에 담긴 내용과는 무관하게 프레임 단위로 압축하는 방식이다. 영상에 담긴 객체의 내용에 대한 이해와 구별 없이 프레임 단위로 처리하는 방법은 각 객체의 비트스트림에 대한 접근, 객체 기반 비트스트림의 조작, 객체와의 다양한 상호작용, 장면조합에 의한 영상 재사용등의 기능상 제한이 있다. 이처럼 다양한 요구를 충족시키기 위해 MPEG-4에서는 영상의 객체를 바탕으로 부호화하는 내용 기반 부호화를 통하여 효율적인 압축을 제공한다. MPEG-4표준의 내용기반 부호화는 매우 낮은 비트율에서 더욱 정확한 비디오 표현을 가능하게 한다. 내용 기반 부호화 기능을 제공하기 위해 MPEG-4에서는 우선적으로 비디오 시퀀스를 의미있는 객체 또는 VOP로 분할하여야 한다. 이후 영상의 분할 정보를 알파맵(alpha map)으로 저장하여 배경과 객체를 분리하여 부호화한다. 본 논문에서 적용한 비디오 영상 입력 모듈과 객체 분할 모듈은 <그림 5>와 같다.

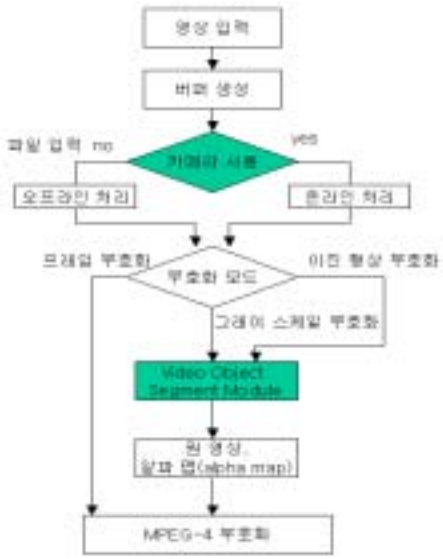

그림 5. 객체 분할 모듈 적용

영상 입력 모듈은 먼저 영상 데이터를 위한 버퍼를 설정한다. 이후 사용자 모드에 따라 오프라인 처리와 온라인 처리를 수행한다. 오프라인 처리는 파일 입출력을 위한 과정이며 온라인 처리는 실시간 처리를 위한 화상카메라 입력 과정이다. 이전의 VM 모델에서는 비디오 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.

5. 객체 분할 알고리즘과 VOP 추출

본 논문에서는 추가적인 알고리즘의 필요 없이 시간적, 공간적으로 분할된 객체의 에지에 수리형태학 알고리즘을 적용하여 정확한 객체의 경계를 찾았다. 본 알고리즘의 구조는 매우 간단하며 쉽게 객체의 정확한 경계를 찾는다. 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 객체사이의 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 적요하여 에지 크기를 조절하여 두 프레임 사이의 움직임이 있는 에지를 정확하게 찾는다. 다섯번째 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 추출 할 수 있는 장점이 있다.

VOP 분할 알고리즘의 가장 중요한 점은 어떻게 영상에서 의미있는 객체를 검출할 수 있는지에 대한 방법의 문제이다. 즉 움직임은 물리적 객체를 식별하는 가장 널리 사용되는 핵심 요소이다. 그러므로 VOP 분할을 위한 중요한

문제는 정확한 움직임 측정을 수행하는 것이다. 우리의 분할 알고리즘은 움직임 정보를 기반으로 배경으로부터 객체를 분리하여 수행되어진다. 각 단계의 자세한 내용은 다음과 같다.

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 설명하였듯이 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 2]을 사용하였다.

[식 2]

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

baseframe :기준 프레임

nframe : 다음프레임

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 3]이다.

[식 3]

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

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 4]은 움직임 에지 추출 방법에 대한 수식이다.

[식 4]

$$ME(MovingEdge) = (G*|F(baseFrame)| \; |((morphology) \; G*|Fn|)$$

[식 4]은 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 6>는 검출된 움직임 에지와 전체 VOP 추출 과정을 보여준다.

[기본 프레임]

[분할될 프레임]

[차이 프레임]

[움직임 에지 영상]

[분할된 영상]

그림 6. VOP 추출 결과

Ⅳ. 실험 및 결과

본 논문에서 제안된 알고리즘은 MPEG-4 실험 영상과 화상카메라 입력영상으로 테스트 하였다. MPEG-4 실험 영상과 화상 카메라 영상은 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. 영상은 프레임 부호화를 통한 압축율과 이진 형상 부호화 통한 압축율을 나타내었다.

실험은 다음과 같은 형태로 구분하여 수행하였다.

1.이미지 종류.
   1) 176*144 사이즈의 움직임이 많은 2000프레임 영상
   2) 176*144 사이즈의 움직임이 적은 2000 프레임 영상
2. 부호화 방법.
   1) rectangle frame 부호화
   2) 이진 형상(binary shape)부호화
3. 적용된 알고리즘.
   1) 기본MPEG-4 VM 모델
   2) 제안된 알고리즘
4. 적용된 민감도, 감지점 파라미터.
   1) 1/1
   2) 5/5000
   3) 10/5000
   4) 20/5000
5. 측정된 결과.
   1) 입력 데이터 비트 수
   2) 압축 데이터 비트 수
   3) 압축율

Rectangle frame 부호화의 실험 결과는 다음과 같다.

표 1. Rectangle frame 부호화

이미지 종류	VM / Proposed	민감도 / 감지점	입력 데이터	압축 데이터	압축률
Rectangle 176 × 144 2000 프레임 움직임이 많은 영상	VM	1/1	72.5MB	3.00MB	약 24.16 : 1
	Proposed	5/5000	72.5MB	1.89MB	약 38.35 : 1
	Proposed	10/5000	72.5MB	1.47MB	약 49.31 : 1
	Proposed	20/5000	72.5MB	1.12MB	약 64.73 : 1
Rectangle 176 × 144 2000 프레임 움직임이 적은 영상	VM	1/1	72.5MB	1.70MB	약 42.64 : 1
	Proposed	5/5000	72.5MB	1.36MB	약 53.30 : 1
	Proposed	10/5000	72.5MB	1.10MB	약 65.90 : 1
	Proposed	20/5000	72.5MB	1.02MB	약 71.07 : 1

<표 1>과 같이 움직임이 많은 영상은 전체적으로 많은 압축 비트가 발생하였다. 일반적인 사각형 프레임 부호화 실험 결과는 VM모델에서는 최대 40:1 정도의 압축율을 보였고, 제안된 알고리즘은 민감도의 조절에 따라 최대 70:1 정도의 압축율을 보여준다.

Binary frame 부호화의 실험 결과는 다음과 같다.

표 2. 이진 형상(binary shape) 부호화

이미지 종류	VM / Proposed	민감도 / 감지점	입력 데이터	압축 데이터	압축률
Binary 176 × 144 2000 프레임 움직임이 많은 영상	VM	1/1	72.5MB	1.98MB	약 36.98 : 1
	Proposed	5/5000	72.5MB	1.42MB	약 51.05 : 1
	Proposed	10/5000	72.5MB	1.02MB	약 71.07 : 1
	Proposed	20/5000	72.5MB	925.0KB	약 80.25 : 1
Binary 176 × 144 2000 프레임 움직임이 적은 영상	VM	1/1	72.5MB	1.34MB	약 54.10 : 1
	Proposed	5/5000	72.5MB	550.0KB	약 134.98 : 1
	Proposed	10/5000	72.5MB	463.7KB	약 160.10 : 1
	Proposed	20/5000	72.5MB	412.3KB	약 180.06 : 1

<표 2>의 이진 형상 부호화는 비디오 객체 분할 모듈에 의해 추출된 VOP 객체를 부호화하기 때문에 <표 1>의 프레임단위 부호화 보다 상대적으로 고압축률을 보여준다. VM모델에서는 최대 50:1 정도의 압축율을 보이고 있으나 제안된 알고리즘은 최대 180:1 정도의 압축율을 보여준다.

본 논문에 적용된 실험 결과는 화상 카메라 입력을 통하여 머리와 어깨만 움직이는 "Akiyo", "Mother & daughter"와 같은 움직임이 작은 영상에서부터 감시 카메라에 적용되는 객체 전체가 움직이는 "Hall Monitor"와 같은 움직임이 많은 영상을 실험하였다. 특히 감시 카메라와 같은 영상에서는 움직임이 없는 부분의 프레임을 드롭시켜 더욱 효율적인 압축율을 보였다.

부호화 과정에서는 화상 카메라로 입력 되는 영상을 움직임 감지하여 고압축 비트스트림을 만들어 낸다. 복호화는 부호기에서 만든 고압축 비트스트림을 이용하여 영상 데이터를 화면에 출력 한다. 복호화 과정은 다음의 인터페이스 설명부분에 추가하여 소개하였다.

본 논문에서 개발된 MPEG-4 부호화의 프로그램 수행 과정과 인터페이스는 다음 <그림 7>과 같다.

1. 초기화면

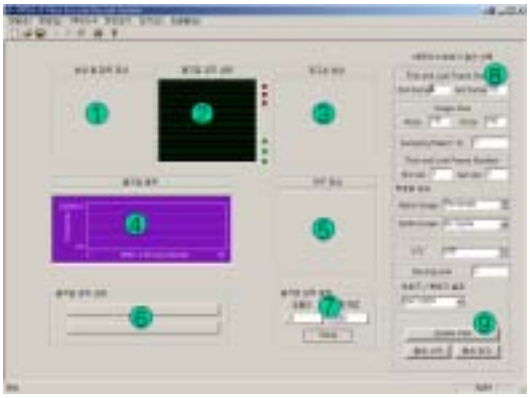

그림 7-1. 초기화면

<그림 7-1>의 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, 영상 분할을 설정한다. 9번 버튼은 초기입력 데이터를 설정하는 버튼이다.

2. 카메라 설정 화면

그림 7-2. 카메라 설정 화면

<그림 7-2>는 영상 입력 모드에서 카메라 입력을 선택했을 경우 보여주는 박스이다. 이곳에서 IYUV, YUV2, RGB24 비트를 선택한다.

3. 움직임 감지 설정 및 Texture 코딩-1

그림 7-3. 움직임 감지 설정 및 텍스쳐 코딩

<그림 7-3>은 움직임 감지를 위해 민감도와 감지점을 입력하는 부분이다.

4. 움직임 감지 설정 및 Texture 코딩-2

그림 7-4. 움직임 감지 설정 및 텍스쳐 코딩

<그림 7-4>는 민감도와 감지점설정 후 사각형 프레임 부호화를 처리하는 형태를 보여준다

5. 객체 분할과 이진 형상 부호화-1

그림 7-5. 객체 분할과 이진 형상 부호화

<그림 7-5>와 <그림 7-6>은 이진 형상 부호화를 보여준다. <그림 7-1>의 초기화면과 같이 <그림 7-5>의 1번은 화상카메라 입력 3번은 이진 형상 부호화 후의 재구성 영상 5번은 VOP 형상을 보여준다.

6. 객체 분할과 이진 형상 부호화-2

그림 7-6. 객체 분할과 이진 형상 부호화

그림 7. 구현된 인터페이스

7. 복호화 과정

그림 8-1. 압축 파일 열기

<그림 8-1>은 저장된 파일을 오픈하는 과정이다

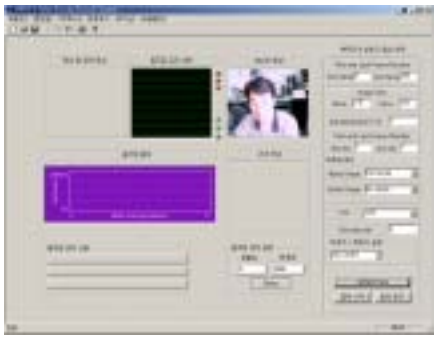

그림 8-2. 영상 복호화
그림 8. 영상 데이터 복호화 과정

<그림 8-2>는 압축된 데이터의 복호화 수행 과정을 보여준다.

본 논문에서는 실시간 고압축 MPEG-4를 위한 전처리 시스템과 비디오 객체 분할 알고리즘을 개발하였다. 개발된MPEG-4 전처리 시스템은 실용적인 소프트웨어 사용을 위해 실시간 카메라 시스템을 적용하였고, 고효율 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 고효율 압축율은 멀티미디어 환경에서 효율적인 비디오 처리를 가능하게 할 것이다. 즉, 내용기반 부호화에 의하여 높은 압축율을 적용하더라도 영상의 손상을 최소화할 수 있다. 개발된 MPEG-4 비디오 전처리 시스템은 인터넷 디지털 컨텐츠, 인터넷 TV, 영상 통신, 영상 회의, 감시 카메라등 뉴미디어 관련 분야에 광범위하게 적용할 수 있을 것이다.

향후 계획으로는 본 논문에서 개발된 MPEG-4 전처리 시스템을 MPEG-4부호기와 복호기에 적용하여 실용적인 MPEG-4 시스템을 개발하는 것이다.

참고문헌

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

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

[3] ISO/IEC/JTC1/SC29/WG11, MPEG/N3908, "MPEG-4 Video Verification Model version 18.0,"Jan. 2001/Pisa. [4] ISO/IEC 14496 Version 1 Part 2 Visual.

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

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

[7] B.K.P. Horn and B.G. Schunck, "Determining optical flow," Artif. Intell., vol. 17, pp. 185-203, 1981.

[8] 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.

[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.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.

[11] 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

김 준 기(Jun-Ki Kim)

1998년 2월 : 호서대학교 전자계산학과 이학사 졸업
2000년 2월 : 호서대학교 컴퓨터 공학과 석사 졸업
2003년 2월 : 호서대학교 컴퓨터 공학과 박사 수료
2003년 -현재 : 호서대학교 컴퓨터 공학과 시간강사

<관심분야> 영상처리, 영상분할, 인터넷 응용, MPEG-4, MPEG-7, MPEG-21

이 호 석(Ho-Suk Lee) 정회원

1983년 2월 : 서울대학교 전자계산기 공학과 공학사 졸업
1985년 2월 : 서울대학교 컴퓨터 공학과 석사 졸업
1985년 2월 : 서울대학교 컴퓨터 공학과 박사 졸업
1994년 - 현재 : 호서대학교 컴퓨터공학부 근무

<관심분야> 영상처리, 영상압축, 웨이블릿, MPEG-4, MPEG-7, MPEG-21