효율적으로 계산 복잡도를 줄인 프레임 제거 트랜스코더 시스템 구조

김 성 민[†] · 김 현 희[††] · 박 시 용[†††] · 정 기 동[††††]

요 약

트랜스코딩은 한 가지 형태로 부호화된 멀티미디어 데이터를 서로 다른 재생 데이터율을 요구하는 이질적인 클라이언트에게 적응적으로 전달하기 위한 해결책이다. 따라서, 트랜스코딩 기법은 입력 스트림을 복호시켜 클라이언트가 요구한 출력 스트림으로 부호화하는 것이 필요하다. 일반적으로, 계산량을 줄이기 위해서 제안된 트랜스코딩 기법들은 비디오 화질의 열화를 발생시키고 그와 반대의 경우는 많은 계산량을 초래한다. 이와 같은 계산량과 화질 사이의 문제를 해결하기 위해서 여러 가지 기법들이 연구되었다. 하지만, 대부분의 연구가 트랜스코더 내부에 한정되어 있었고, 서버 측과의 상호작용을 통한 성능 향상에 대한 연구는 적었다. 멀티미디어 데이터를 전력과 성능이 낮은 단말기 또는 낮은 대역폭의 네트워크에 속한 이질적인 클라이언트로 서비스할 때, 트랜스코더 자체의 해결 방안에 서버 측에의 특정 작업을 추가할 경우 트랜스코더에서 실제 처리해야 하는 프레임의 개수를 줄일 수 있고 이를 통해서 서비스 효율의 향상을 기대할 수 있다. 따라서 본 논문에서는 효율적인 트랜스코더와 서버 측 기반의 알고리즘을 함께 고려하여 계산 처리 과정을 줄일 수 있는 프레임 제거 트랜스코더 시스템 구조를 제안한다.

키워드 : 트랜스코딩, 트랜스코더 구조, 계산 복잡도

An efficient and Low-Complexity Frame-Skipping Transcoder System Architecture

Sung-Min Kim[†] · Hyun-Hee Kim[††] · Si-Yong Park[†††] · Ki-Dong Chung[††††]

ABSTRACT

The transcoding is a solution which is able to adapt to heterogeneous clients of requesting a different playback rate of multimedia data. Thus, The transcoding needs decoding and encoding. In general, previous studies to reduce complexity have a problem, the degradation of visual quality. On the contrary, previous studies to reduce the degradation of visual quality lead to heavy computation. Thus, many researchers have studied a solution between the complexity and the degradation of visual quality. But until now, most researches of this region have dealt with the transcoder itself, such researches about a server's assistance to improve the performance of transcoder is rarely studied. In case of servicing multimedia data to heterogeneous clients which have low capabilities, the assistance of server side is able to reduce frames with processing in the transcoder and improve the performance of the transcoder. Thus in this paper, we propose the frame-skipping transcoder system architecture that takes into consideration transcoder and server side to reduce the complexity of the transcoder.

Key Words : Transcoding, Transcoder Architecture, Complexity

1. 서 론

인터넷을 통해 비디오, 오디오를 포함한 멀티미디어 데이터를 서비스 받는 것은 전자메일을 확인하거나, 텍스트로만 된 정보를 검색하는 것과 같이 일반화 되어 있다. 지나간 방송의 다시 보기나 신작영화의 예고편을 VoD(Video on Demand)를 통해 서비스 받는 것도 이런 모습의 한 예라 할 수 있다. 하지만, 동일한 멀티미디어 컨텐츠를 LAN, VDSL과 같은 높은 대역폭의 네트워크에 속한 클라이언트에 서비스할 경우와 무선 네트워크 또는 이동 통신 네트워크에 속한 클라이언트에 서비스할 경우에는 여러 가지 문제점이 발생될 수 있다. 특정한 네트워크의 클라이언트는 만족할 만한 서비스를 받지 못하거나 반대의 경우에는 클라이언트가 서비스를 전혀 받지 못하는 경우가 발생한다. 이를 위해서 동일한 멀티미디어 컨텐츠를 서로 다른 네트워크와 클라이언트 단말을 고려한 여러 가지의 버전으로 준비하는 것은

† 준 회 원 : 부산대학교 컴퓨터공학과 박사과정
†† 준 회 원 : 부산대학교 컴퓨터공학과 석사과정
††† 준 회 원 : 부산대학교 전자계산학과 박사과정
†††† 종신회원 : 부산대학교 전자계산학과 교수

논문접수 : 2004년 9월 17일, 심사완료 : 2005년 6월 20일

시스템의 비용을 높이게 된다.

트랜스코딩은 클라이언트가 포함된 접속 네트워크와 코어 네트워크의 경계 지점에서 해당 클라이언트에 맞는 형태의 버전으로 변환할 수 있는 적응성을 제공한다. 앞서 언급한 문제점들은 트랜스코딩을 통해서 쉽게 해소할 수 있다. 그러나 트랜스코딩은 멀티미디어 데이터의 변환을 위해 복호화와 부호화 과정이 동시에 필요하게 되는데, 이 과정은 많은 계산량을 필요로 한다. 또한, 복호화와 부호화를 통한 화질의 열화가 한 번 더 발생되는 특징을 가진다. 이에 따라, 지금까지 적은 계산량과 높은 비디오 화질을 위한 효율적인 트랜스코딩 기법과 구조가 많이 연구되었다. 과거의 주요한 연구 기법들은 하나의 시퀀스 내에서 프레임 사이의 부호화 연관성을 고려하여 움직임 벡터를 예측하거나 예측된 움직임 벡터를 보완하는 움직임 벡터 보정기법들이다[5,6,7,8]. 그리고 부호화 연관성의 특징을 고려하여 트랜스코더 시스템의 성능을 향상시키는 구조들에 대해서도 연구되었다[9,10]. 하지만, 이러한 기법들은 트랜스코더의 구조를 개선시키거나, 화질과 계산량의 향상을 위한 기법들을 트랜스코더 내부에 적용시킨 구조가 대부분이다. 그러나 이질적인 클라이언트를 지원하기 위한 트랜스코더에서 복호화 과정의 처리량을 최소화시키기 위해 서버 측에서의 특정 알고리즘을 함께 고려한다면 트랜스코더 시스템의 성능을 더욱 더 향상시킬 수 있다. 따라서 본 논문에서는 기존의 우수한 트랜스코더를 이용하고 서버 측의 추가 작업을 통해 효율적으로 계산량을 줄일 수 있는 프레임 제거 트랜스코더 시스템 구조를 제안한다.

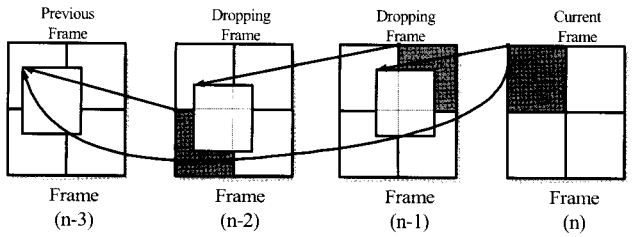
본 논문의 구성은 다음과 같다. 2장에서는 기존의 트랜스코딩 기법 및 트랜스코더에 관련된 연구들을 소개하고, 3장에서는 제안한 기법인 효율적인 프레임 제거를 위한 트랜스코더 시스템 구조에 대해서 설명한다. 마지막으로, 결론 및 향후 과제는 4장에서 제시한다.

2. 관련 연구

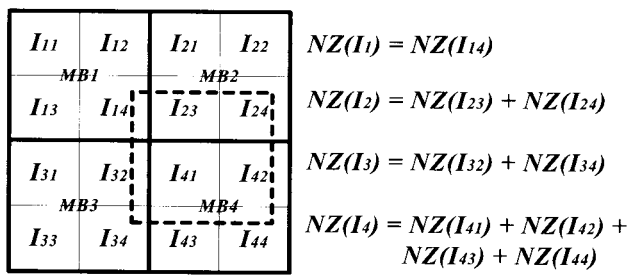
트랜스코딩은 일반적으로 비트율을 줄여서 서비스하기 위한 방법을 제공한다. 서버측에서는 높은 재생 데이터 율로 만든 컨텐츠를 불특정한 다수의 이질적인 클라이언트에게 서비스할 때, 클라이언트의 특성을 고려하지 않고 보유한 컨텐츠를 전송하게 되고 트랜스코더에서 해당 클라이언트에게 적절한 재생 데이터 율로 변환시켜서 서비스하게 된다.

비트율을 줄이기 위해서 가장 일반적인 방법은 복호기와 부호기를 직렬로 배치시켜 픽셀 영역까지 복호화시킨 후 부호화 하는 것이다[2]. 하지만, 복호화와 부호화의 진행에 따른 계산량이 너무 많으므로 실시간 응용에는 부적합하다. 따라서, 최근에는 픽셀 영역에서의 트랜스코딩이 아닌 DCT 영역에서의 트랜스코딩을 위한 기법들이 연구되었다. 양자화된 DCT 계수의 양자화 계수를 조정함으로써 비트율을 줄이거나, 양자화된 DCT 계수의 일부분을 복호화시켜 트랜스코딩하는 기법들이 제안되었다[3, 4, 9, 10]. 하지만, 이러한 기법들도 양자화 에러가 축적되면서 연이은 프레임에 영향을 미치게 되는 drift 문제가 발생하고, 이를 해결하기 위해서는 부가적인 계산량을 필요로 한다.

멀티미디어 데이터를 클라이언트에서 요구하는 재생 데이터 율로 변환하기 위해서 앞서 언급한 기법 중 재 양자화 기법만으로는 불가능한 경우가 발생한다. 이 때는 재생 프레임 율을 조절함으로써 클라이언트가 원하는 재생 데이터 율을 얻을 수 있게 된다. 프레임 율의 조절을 위해 중간 프레임을 클라이언트의 재생 데이터 율에 맞추어 삭제시킬 때, 클라이언트로 전송되는 프레임 간에는 연관성의 변화가 발생하게 된다. 프레임 율을 조절하는 트랜스코더에서는 이러한 연관성의 변화로 움직임 벡터의 정보 수정이 요구된다. 정확한 움직임 벡터를 얻기 위해서는 픽셀 영역에서의 움직임 예측과정을 거쳐야 한다. 하지만, 이 과정은 부호화 과정에서 가장 많은 계산량을 차지하는 움직임 예측과정을 되풀이하게 되므로 실시간에 적용하기 힘든 단점을 가진다. 따라서, 삭제되는 프레임이 가지는 정보들을 재사용하여 변화된 연관성을 발견하는 기법인 FDVS(Forward Dominant Vector Selection), ADVS(Activity Dominant Vector Selection)와 같은 기법들이 제안되었다[5, 6]. (그림 1)은 중첩되는 매크로 블록 사이에서 가장 큰 중첩영역을 가지는 매크로 블록의 움직임 벡터를 그대로 사용하는 FDVS에 대해서 설명하고 있고, (그림 2)는 중첩되는 영역의 움직임 정보의 양을 통해서 움직임 벡터를 선택하는 ADVS에 대해서 설명한다. (그림 2)에서 ADVS기법을 통해서 선택되는 움직임 벡터는 NZ(I1)~NZ(I4)중에서 가장 큰 값을 가지는 움직임 벡터이다.

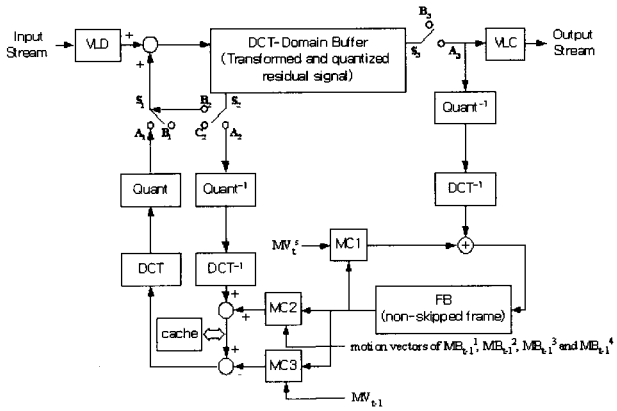


(그림 1) FDVS에서의 움직임 벡터 재구성



(그림 2) ADVS에서의 움직임 벡터 재구성

하지만, 이러한 기존의 정보를 이용하여 움직임 벡터를 선택하는 기법들은 실시간에 적합하고 원하는 비트율은 얻을 수 있지만, 부정확한 움직임 예측으로 인한 화질의 열화를 피할 수 없다. 발생되는 화질의 열화를 줄이기 위해서는



(그림 3) [10]에서 제안된 트랜스코더 구조

움직임 예측의 정확성을 높이기 위한 보완 기법들이 적용되어야 하는데, 이러한 기법들은 추가적인 작업을 요구하는 단점을 가진다[7, 8].

[9]-[10]에서는 프레임 율을 조절할 때 움직임 예측의 특징에 따라 프레임 전체를 복호화시키기 않고 부분적으로 복호화를 시켜서 선택적으로 프레임을 삭제할 수 있는 구조를 제안하였다. 이 기법에서는 [5]-[6]에서 제안된 움직임 벡터의 재사용 기법들을 프레임이 삭제할 경우 변화되는 연관성의 발견을 위해서 사용하고 있으며, 움직임 벡터값에 따라 비움직임 보상모드와 움직임 보상모드로 구분하여 처리 절차를 간소화시켰다.

이전 프레임을 통해서 움직임 벡터를 예측할 경우, 가장 유사한 블록의 위치를 선정해서 그 위치와의 상대적인 차이값을 찾아낸다. 이 때, 많은 비디오 시퀀스들은 움직임 벡터의 값이 (0,0)과 같이 이전 프레임의 동일한 위치를 참조하여 부호화되는 경우가 많다. [9]-[10]에서는 이러한 특징을 통해서 움직임 벡터가 (0,0)인 경우는 비움직임 보상모드로서 DCT 영역에서 DCT가 가지는 선형성을 이용하여 픽셀 영역까지 부호화하는 과정없이 처리한다. 삭제되는 프레임이 가지는 정보를 통해서 부호화되어야 할 프레임의 새로운 연관성을 DCT 영역에서 만들어낸다. 따라서, 특정 비디오 시퀀스가 가지는 비움직임 보상모드가 전체 매크로 블록중에서 차지하는 비율이 높아지면 높아질수록 처리율의 향상은 더욱 커진다. 또한, DCT 영역에서의 처리는 픽셀영역까지 복호화시킨 후의 재 부호화 과정에서 발생하는 추가적인 원본데이터와의 차이값을 줄이는 역할을 한다. (그림 3)은 [10]에서 제안된 트랜스코더 구조를 설명한다.

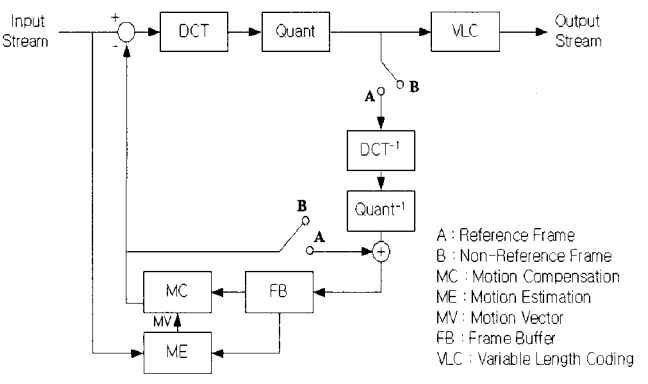
이 외에도 클라이언트 측의 VCR 기능을 지원하기 위한 기법도 소개되었는데, 스트리밍 서버가 효율적으로 비디오를 서비스하기 위해서는 클라이언트의 환경을 고려해야 한다. 서버 측 기반의 알고리즘의 예로, VOD 서버가 VCR 기능을 제공하기 위해서 여러 가지의 서로 다른 버전의 같은 비디오를 저장하여 서비스하는 기법이 있다. 제공되는 비디오는 속도, 방향에 구애받지 않고 서비스 될 수 있으므로, 클라이언트는 디코더의 성능에 제한받지 않고 다양한 서비스를 받을 수 있다. 그러나 서버의 저장 공간 문제를 해결하기 위해 두 가지 스트림만으로 Fast-Forward, Fast-Reverse 등의 기능을 제공할 수 있는 기법이 연구되었다[1].

3. 효율적으로 계산 복잡도를 줄인 프레임 제거 트랜스코더 시스템 구조

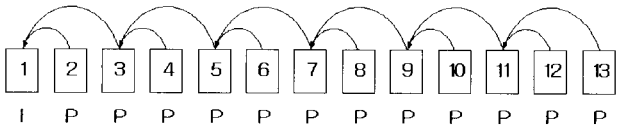
일반적으로, 프레임을 제거하는 트랜스코더에서 가장 큰 이슈는 시간적 연관성을 가지는 프레임들 사이의 예측 관계의 재구성이다. 예를 들어, 하나의 시퀀스가 I-프레임만으로 구성되었다면 이웃한 프레임간의 참조 관계는 불필요할 것이고, 프레임 제거 과정만을 통해서 원하는 비트율을 얻을 수 있을 것이다. 그러나 가장 인기 있는 비디오 압축 표준인 MPEG-1/2/4 또는 H.26x로 부호화된 시퀀스의 I-프레임뿐만 아니라 P-/B-프레임들은 보다 높은 압축을 위해 시간적인 예측 관계를 가지게 된다. P-/B-프레임의 경우, 현재 프레임은 이전 프레임과의 차이 값만을 가지고 부호화 되므로 I-프레임에 비해 크기가 훨씬 작다. 프레임 제거 트랜스코더에서 삭제되는 프레임은 연이어 오는 프레임(삭제되거나 혹은 출력될)에 의해 참조되므로, 트랜스코더 내부에서는 처리되는 프레임에 해당된다. 그러므로 대부분의 삭제되는 프레임은 그것이 포함하고 있는 중요한 정보가 출력될 프레임에 활용되기 위해 단순히 제거되지 않고 부가적인 처리 과정을 거친다.

본 논문에서는, "예측주기"를 사용하여 트랜스코더에서 삭제될 프레임의 부가적인 처리를 줄일 수 있는 방법에 대해서 논의한다. 여기서, 예측주기는 참조 프레임이 나타나는 프레임 간격을 말한다. 예측주기의 시작 프레임은 예측주기 내의 모든 프레임이 참조하고 있으므로, 구간 내에서 여러 개의 프레임이 삭제되더라도 참조되는 프레임이 버퍼에 저장되어 반복적으로 재사용할 수 있다.

기존의 기법들이 트랜스코더 내부에서의 특별한 절차를 통해서 계산량을 줄이거나 화질의 개선을 연구한 반면, 본 논문에서는 서버 측에서 예측 주기를 통해 코딩을 한 후 전송되는 멀티미디어 스트림을 트랜스코더에서 효율적으로 프레임을 제거하도록 한다. (그림 4)는 트랜스코더에 입력 스



(그림 4) 서버의 부호기



(그림 5) 예측주기가 2인 프레임 부호화 구조

트림을 제공하는 비디오 서버의 구조이고, 부호기의 내부에 포함된 두 개의 스위치를 통해서 (그림 5)에서 설명하는 시퀀스 구조를 출력한다. (그림 5)는 예측주기가 2일 경우 생성된 프레임의 구조를 나타낸다.

예측되는 프레임으로 구성된 대부분의 시퀀스에서 참조되는 프레임간의 거리가 멀수록 참조 프레임을 통해 예측한 차이값이 커진다. 이 점을 고려하여 본 논문에서는 프레임간의 예측주기 가 2~4일 경우의 연속적으로 프레임을 제거하는 트랜스코더 시스템 구조를 제안한다. 그리고 본 논문에서는 단방향 예측만을 고려하고 있으므로, B-프레임을 포함하지 않고 I-/P-프레임으로 구성된 시퀀스에 대해서만 적용 가능하다.

[10]에서 제안한 기법은 앞서 언급했듯이 움직임 벡터의 특징에 따라 기존의 불필요한 계산량을 줄임으로써, 보다 효율적인 트랜스코딩을 수행한다. 본 논문에서는 [10]에서 제안한 기법의 움직임 재사용 기법을 개선시켜 적용한 MMAT(Motion-compensation Mode Adaptive Transcoder)를 사용한다.

3.1 활동정보 및 중첩영역을 고려한 움직임 합성 기법

관련연구에서 소개한 움직임 벡터를 재사용하는 FDVS, ADVS 기법은 제거되는 프레임의 특정 움직임 벡터를 선택함으로써 움직임 예측의 복잡한 처리과정을 제거했으나, 움직임 벡터의 선택이 가지는 정밀함의 한계는 사용자 측의 화질 저하를 초래했다. 따라서 활동성 정보를 지니고 있는 0이 아닌 DCT 계수의 개수 합과 중첩되는 매크로 블록의 영역비율을 모두 고려하여, 움직임 벡터의 정밀함을 높인 AAVC(Activity and Area based Vector Composition) 기법을 MMAT에 적용하였다. <표 1>과 <표 2>는 MMAT에 적용한 움직임 합성 기법 및 FDVS, ADVS 기법의 움직임 벡터 정밀성을 비교하기 위한 입력 시퀀스와 실험 환경을 보인다.

움직임이 많은 경우(Football)와 상대적으로 움직임 적은 경우(Suzie)를 비교하였으며, 배경의 움직임이 많은 경우(Carphone)와 움직임이 특별한 형태의 시퀀스(Tempete)에 대해서도 비교하였다.

〈표 1〉 입력 시퀀스

시퀀스	양자화 파라미터	평균 비트율(Kbps)	Video Format	프레임율 (fps)	프레임 개수
Football	3	256	QCIF	30	258
Carphone		584			381
Suzie		512			150
Tempete		1186			256

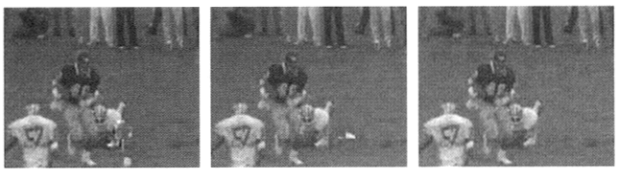
〈표 2〉 실험 환경

시스템 환경	CPU & 메모리	Pentium IV 1.2 Ghz CPU, 256M
	운영체제	Redhat Linux 9.0
포맷	H.263	"IPPP.." 시퀀스
코덱	FFMPEG[11]	Version 0.4.8
출력 시퀀스 프레임율	15	프레임율을 1/2로 조절
출력 시퀀스 양자화 파라미터	5, 15, 25	

〈표 3〉 시퀀스별 평균 PSNR

시퀀스	양자화 파라미터	FDVS	ADVS	AAVC
Football	5	30.23	30.28	33.97
	15	25.7	26.65	32.15
	25	24.48	25.5	29.93
Carphone	5	32.1	33.7	37
	15	28.11	28.9	30.56
	25	26.13	26.76	27.86
Suzie	5	37.75	38.38	38.59
	15	32.89	32.98	33
	25	30.63	30.75	30.8
Tempete	5	27.4	27.65	35.47
	15	24.97	25.05	28.31
	25	23.53	23.63	25.5

<표 3>을 통해서 확인할 수 있듯이 FDVS, ADVS를 적용하여 움직임 벡터를 선택하는 것보다 본 논문에서 제안하는 움직임 합성 기법이 객관적으로 우수한 움직임 벡터의 정밀함을 제공한다. (그림 6) ~ (그림 9)는 각 기법들을 통해 만들어진 실제 이미지를 보여주고 있다. 주관적으로 보여지는 화질도 움직임 합성 기법이 나은 성능을 보임을 확인할 수 있다.

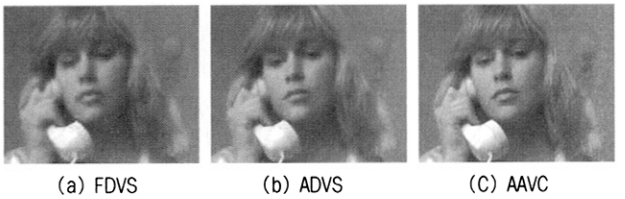


(a) FDVS (b) ADVS (C) AAVC

(그림 6) Football - 87번째 프레임

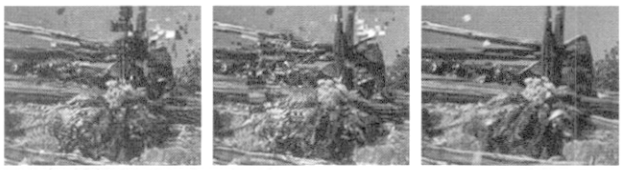
(a) FDVS (b) ADVS (C) AAVC

(그림 7) Carphone - 5번째 프레임



(a) FDVS (b) ADVS (C) AAVC

(그림 8) Suzie - 30번째 프레임



(a) FDVS (b) ADVS (C) AAVC

(그림 9) Tempete - 98번째 프레임

PSNR 및 화질의 비교를 통해서도 알 수 있듯이 기존의 기법들이 선택의 한계를 드러내는 반면, 본 논문에서 제안하는 활동정보 및 중첩영역 기반의 움직임 벡터 합성 기법은 제거되는 프레임에 의해 손실되는 움직임 벡터 정보의 복원력이 뛰어나다.

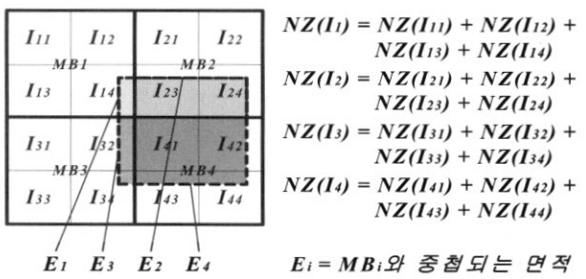
움직임 벡터의 합성을 위한 단계는 다음과 같다.

step 1. 움직임 벡터 합성을 위한 매크로 블록 별 가중치 계산 : 가중치는 중첩되는 영역의 매크로 블록의 0이 아닌 DCT 계수의 개수 합과 중첩되는 영역의 좌표크기 면적을 통해서 구한다. (그림 10)을 통해서 얻어지는 각 매크로 블록의 가중치 Wi는 (식 1)과 같이 구한다.

$$Wi = (Ei * NZ(Ii)) / \sum_{i=1}^{4} Ei * NZ(Ii) \qquad (식\ 1)$$

step 2. 각 매크로 블록 별 가중치를 통해 새로운 움직임 벡터의 합성 : step 1을 통해서 얻어진 각 매크로 블록의 가중치와 해당 매크로 블록의 움직임 벡터의 곱을 통해서 새로운 움직임 벡터를 합성한다. 합성식은 (식 2)와 같다.

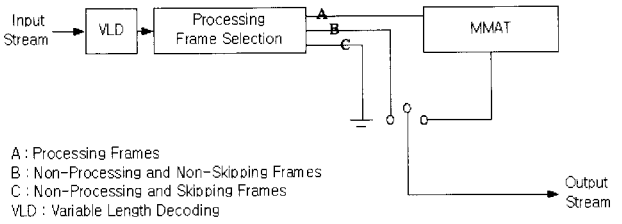
$$\text{합성된 움직임 벡터} : \sum_{i=1}^{4} MVi * Wi \qquad (식\ 2)$$



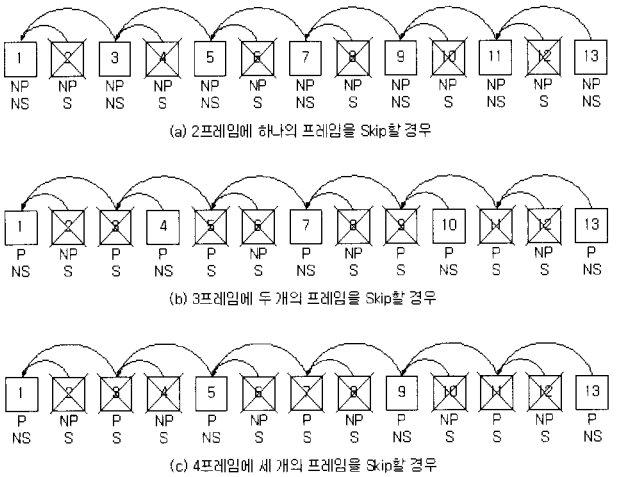
(그림 10) 활동정보 및 중첩영역을 통한 움직임 벡터 재구성

3.2 트랜스코더 구조

(그림 4)에서 출력되는 스트림은 네트워크를 통해 (그림 11)의 트랜스코더에 전달된다. (그림 11)은 입력된 스트림의 프레임이 NP, P 혹은 NS, S 프레임인지에 따라 선택적으로 처리하는 트랜스코더의 구조를 설명하고 있다. 각각의 프레임이 가지게 되는 속성으로 NP는 Non-Processing, P는 Processing, NS는 Non-Skipping, 그리고 S는 Skipping을 의미한다. 트랜스코더에서 트랜스코딩의 처리를 거치는 프레임은 Processing 프레임이고, 바로 클라이언트에 전달되거나 삭제되는 프레임은 Non-Processing 프레임이다. 또한, 트랜스코더에서 입력되는 스트림의 데이터 율과 출력해야 할 스트림의 데이터 율을 고려하여 프레임을 삭제하는 경우, Skipping 프레임은 삭제가 되는 프레임이고 Non-Skipping 프레임은 클라이언트에 전달될 프레임이 된다.



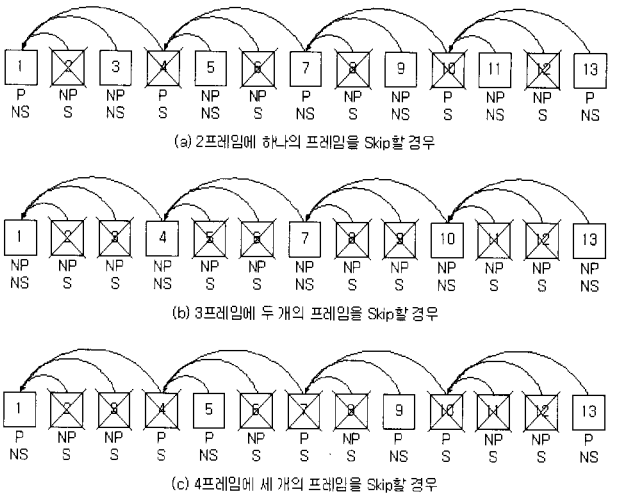
(그림 11) 트랜스코더 구조



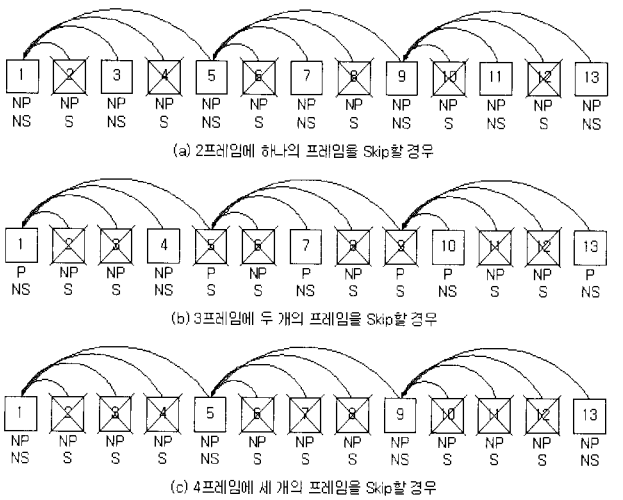
(그림 12) 프레임 제거의 예 (예측주기가 2인 경우)

(그림 11)의 트랜스코더가 제거되는 프레임 개수의 동적인 특성을 제공하기 위해서 본 논문에서는 제거되는 프레임의 개수가 서로 다른 세 가지의 경우를 (그림 12) ~ (그림 14)에서 설명하고 있다.

(그림 12) (a)의 경우는 삭제되는 프레임이 출력될 프레임에 참조되지 않으므로, 부가적인 처리 과정이 전혀 필요하지 않다. (그림 12) (b)에서는 2번 프레임이 삭제되어도 이후의 어떤 프레임도 그것을 참조하지 않으므로 특별한 처리를 요구하지 않는다. 하지만, 3번 프레임의 경우에는 출력되어야 할 4번 프레임이 참조하고 있으므로, 삭제되는 프레임이지만 처리가 필요하게 된다. 그리고, 1번 프레임의 경우는 4번 프레임이 3번 프레임을 통해서 움직임 정보 및 부호화 정보를 변환할 때 참조되므로 처리가 필요한 프레임이다. (그림 12) (c)의 경우도 동일한 과정으로 설명된다. 기존의 트랜스코더는 모든 프레임에 대해서 처리를 해야 하지만, 예측 주기를 통한 부호화에 따라 트랜스코딩 과정 중에 처리해야 할 프레임의 개수를 상당량 줄일 수가 있다. (그림 13)과 (그림 14)는 예측 주기가 3과 4일 때의 경우를 설명하고 있다. (그림 12)의 경우와 마찬가지로 설명되고, 각각의 경우에 따른 처리가 필요한 프레임의 개수는 <표 4>와 같다. 이 때, 대상되는 프레임의 개수는 2,3,4를 가지는 예측



(그림 13) 프레임 제거의 예 (예측주기가 3인 경우)



(그림 14) 프레임 제거의 예 (예측주기가 4인 경우)

주기와 2,3,4를 가지는 삭제 주기의 최소공배수인 12프레임을 대상으로 한다.

(그림 11)에서 Processing Frame Selection에서 실제 처리가 필요한 프레임과 그렇지 않은 프레임을 (그림 12), (그림 13), (그림 14)에서와 같이 선택해서 처리한다. 우선은 삭제 주기를 먼저 구한 후, 삭제 주기에 따라서 처리가 필요한 프레임과 그렇지 않은 프레임으로 분리해서 처리한다. (그림 15)는 Processing Frame Selection에서 처리하는 알고리즘을 나타낸다.

〈표 4〉 삭제 주기 및 예측 주기에 따른 처리가 필요한 프레임 율 비교

삭제 주기 \ 예측 주기	2	3	4
2(2프레임당 하나의 프레임 삭제)	0%	33%	0%
3(3프레임당 두 개의 프레임 삭제)	67%	0%	42%
4(4프레임당 세 개의 프레임 삭제)	50%	50%	0%

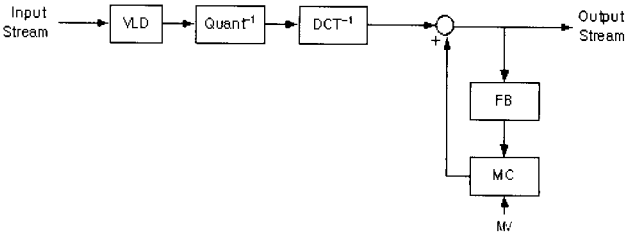
```
Procedure Processing_Frame_Selection()
Begin Procedure
//입력 비트 스트림의 데이터 율과 출력할 스트림의 데이터 율을 통
  해서 삭제 주기를 결정
nsp = 삭제 주기
pp = 예측 주기
If (pp == nsp * alpha) // alpha = 1,2,3,...
        Exit Procedure  //어떠한 프레임도 처리가 필요하지 않음
End If
//pp와 nsp의 최소공배수를 구함
lcm = 최소공배수
last_pp_skip=1 //트랜스코더에서 마지막으로 처리된 주기 프레임의
출력 여부(삭제 여부)
                //1 : 삭제된 프레임, 0 : 삭제되지 않은 프레임
k = 1 //프레임의 번호
For ( k <= lcm ) //프레임의 속성은 lcm 단위로 반복
        If((k-1) % pp == 0)
                //처리가 필요한 프레임
        Else
                If(pp <nsp) //삭제 주기가 예측 주기보다 클 경우
                  If((k-1) % nsp == 0)
                                //처리가 필요한 프레임
                    End If
                Else //삭제 주기가 예측 주기보다 작을 경우
                  If(last_pp_skip == 1 && (k-1) % nsp == 0)
                                //처리가 필요한 프레임
                    End If
                End If
        End If

        If((k-1) % pp == 0 && (k-1) % nsp == 0)
            last_pp_skip = 0 //가장 최근의 주기 프레임이 삭제
            프레임이 아님
        End If
        k++; //다음 프레임 처리
End For
End Procedure
```

(그림 15) Processing Frame Selection의 알고리즘

<표 4>를 통해 알 수 있듯이, 예측주기와 삭제주기를 이용한 모든 경우에 [10]에서 제안한 트랜스코더가 100%의 프레임을 처리해야 하는데 반해서, 많은 양의 프레임 처리를 줄인 것을 확인할 수 있다.

또한, 삭제 주기가 2 ~ 4인 경우에 예측 주기가 2인 경우보다는 3과 4인 경우에 트랜스코더에서 처리해야 할 프레임의 개수를 줄일 수 있다. 특히 예측 주기를 4로 했을 경우는 2로 했을 경우의 1/3, 3으로 했을 경우의 1/2 수준으로 나타난다. 앞서도 언급했듯이 예측주기가 길어지면 길어질수록 코딩 효율은 떨어지게 된다. 따라서 트랜스코더 시스템의 성능과 네트워크의 상태에 따라서 적절한 예측 주기를 선택해서 서비스할 수 있다.

(그림 16)은 일반적인 클라이언트의 복호기로서 본 논문에서는 어떠한 수정없이 적용이 가능하다. 이미 서버 측 기반의 프레임 참조 관계 재 정의와 제안된 트랜스코더 구조를 통해서 변환된 데이터는 일반적인 복호기로서 복호가 가능하다.



(그림 16) 클라이언트의 복호기

4. 결론 및 향후 과제

전통적인 복호기-부호기의 형태를 가지는 트랜스코더 구조는 너무 많은 계산량으로 인해 압축된 형태의 DCT-도메인에서 부분적인 DCT 계수만을 복호화하여 이용하는 기법들이 제안되었다. 그러나 에러 축적의 문제점 때문에 연구된 재 양자화 기법도 원하는 비트율을 얻기에는 부족했다. 그래서 이전 프레임과의 연관성이 가장 높은 매크로블록의 움직임 벡터를 재사용하거나 보정하여 사용하는 기법이 제안되었다. 그러나 이 기법들도 원하는 비트율은 얻을 수 있으나, 부정확한 시간적 연관성으로 화질의 열화가 발생하는 한계를 가진다.

본 논문에서는 시스템의 관점에서 효율적인 프레임-제거 트랜스코딩의 경량화된 구조를 제안한다. 서버의 부호기에서는 스위치를 통해서 미리 프레임간의 참조 관계를 제안된 기법에 따라 재구성하고, 트랜스코더에서는 처리 과정이 필요한 프레임과 그렇지 않은 프레임을 구분한다. 이후 처리 과정이 필요한 프레임에 대해서 트랜스코딩을 하게 되므로 계산 과정을 효율적으로 감소시킨다. 클라이언트의 복호기는 낮은 비트율의 트랜스코딩된 비디오를 복호화하고 재생하므로, 부가적인 처리 과정이 필요하지 않다. 따라서 클라이언트의 환경적인 요소에 의한 간섭을 최소화할 수 있으며 대부분의 단말기에서 지원될 수 있다.

제안된 시스템 구조는 서버측이 트랜스코더의 경유 없이는 직접적으로 클라이언트에 서비스를 할 수 없다. 따라서 앞으로 서버와 클라이언트 간의 직접적인 서비스가 가능한 트랜스코더와 클라이언트의 구조에 대한 연구가 필요하다. 본 논문에서 제안한 기법은 I,P프레임으로 구성된 시퀀스에만 적용이 가능하다. 따라서, 양방향 예측인 B프레임을 포함할 경우에도 적용 가능하도록 확장할 예정이다. 또한, 예측 주기에 따라 생성된 스트림의 크기와 트랜스코더 내부의 처리 효율 사이의 관계를 다양한 시퀀스를 통해 비교할 것이다.

참 고 문 헌

[1] S. Huang, "Improved Techniques for Dual-Bitstream MPEG Video Streaming with VCR Functionalities," *IEEE Trans. on Consumer Electronics*, Vol.49, No.4, NOVEMBER, 2003.

[2] A. Vetro and C. Christopoulos and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," *IEEE signal Processing Magazine*, ISSN : 1053-5888, Vol.20, Issue 2, pp.18-29, March, 2003.

[3] H. Sun, W. Kwok, and J. W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. Circuits, Syst., Video Technol.*, vol.6, pp.191-199, Apr., 1996. 900 IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL.11, NO.8, AUGUST, 2002.

[4] Y. Nakajima, H. Hori, and T. Kanoh, "Rate conversion of MPEG coded video by re-quantization process," in *Proc. IEEE Int. Conf. Image Processing'95*, vol.3, Washington, DC, Oct., 1995, ICIP95, pp.408-411.

[5] J. Youn, M. T. Sun and C. W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Trans. Multimedia*, vol.1, pp.30-40, Mar., 1999.

[6] Mei-Juan Chen; Ming-Chung Chu; Chih-Wei Pan, "Efficient motion-estimation algorithm for reduced frame-rate video transcoder," *Circuits and Systems for Video Technology*, IEEE Transactions on , Vol.12, Issue : 4, pp.269-275, Apr., 2002.

[7] J. Youn, M. T. Sun and C. W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Trans. Multimedia*, vol.1, pp.30-40, Mar., 1999.

[8] M. J. Chen, M. C. Chu and C. W. Pan, "Efficient motion estimation algorithm for reduced frame-rate video transcoder," *IEEE Trans. Circuits syst. Video Technol.*, vol.12, pp.269-275, Apr., 2002.

[9] Kai-Tat Fung and Wan-Chi Siu, "DCT-based Video Frame-skipping Transcoder," Proceedings, *IEEE International Symposium on Circuits and Systems* (ISCAS'03), Vol.II pp.656-659, Bangkok Tailand, May, 2003.

[10] K. Fung, Y. Chan, and W. Siu "Low-Complexity and High-Quality Frame-Skipping Transcoder for Continuous Presence Multipoint Video Conferencing," *IEEE Trans. Multimedia*, vol.6, no.1, February, 2004.

[11] http://www.ffmpeg.org

김 성 민

e-mail : morethannow@pusan.ac.kr

2001년 부산대학교 전자계산학과(학사)

2003년 부산대학교 전자계산학과(이학석사)

2003년~현재 부산대학교 컴퓨터공학과 박사과정

관심분야 : 트랜스코딩, 멀티미디어 스트리밍, 유비쿼터스 컴퓨팅

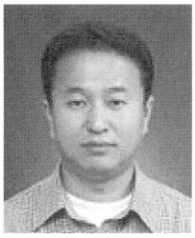
김 현 희

e-mail : h2k@pusan.ac.kr

2003년 신라대학교 멀티미디어공학과(학사)

2003년~현재 부산대학교 컴퓨터공학과 석사과정

관심분야 : 트랜스코딩, 유비쿼터스 컴퓨팅, 멀티미디어 스트리밍



박 시 용

e-mail : sypark@melon.cs.pusan.ac.kr
1997년 경성대학교 전자계산학과(학사)
2001년 부산대학교 멀티미디어과(이학석사)
2001년~현재 부산대학교 전자계산학과 박사과정
관심분야 : 유비쿼터스 컴퓨팅, 멀티미디어 스트리밍, 이동 통신

정 기 동

e-mail : kdchung@melon.cs.pusan.ac.kr
1973년 서울대학교 응용수학과(학사)
1975년 서울대학교 전자계산학과(석사)
1986년 서울대학교 계산통계학과(이학박사)
1990년~1991년 MIT, South Carolina 대학 교환교수
1995년~1997년 부산대학교 전자계산소 소장
1978년~현재 부산대학교 전자계산학과 교수
관심분야 : 병렬처리, 멀티미디어 시스템, 멀티미디어 통신