프레임제거 기반의 비디오 트랜스코딩에서의 화질 개선을 위한 효율적인 움직임 벡터 산출 방법☆

An Efficient Motion Vector Estimation For Improving Picture Quality Of Frame Skipping Based Video Transcoding

박 용 대* 노 병 희**
Yong-Dae Park Byeong-hee Roh

요 약

본 논문에서는 프레임 제거시에 화질 개선을 위한 새로운 트랜스코딩 방식을 제안한다. 제안 방식은 연속된 프레임들간의 움직임 벡터들의 상관 관계를 활용하고 있으며, 트랜스코딩시 한 매크로블록당 네 개의 움직임 벡터를 고려한다. 제안 방법은 트랜스코딩시 부호화된 비트스트림으로부터 영상을 복호화할 필요가 없으므로, 움직임 벡터를 산출하는 복잡한 연산이 요구되지 않는다. 실험 결과는 트랜스코딩된 프레임이 I-픽쳐에서 멀리 떨어질수록, 움직임이 큰 영상일수록 기존 방식에 비하여 더 나은 화질 개선 효과를 나타냄을 보여준다.

Abstract

In this paper, we propose a new transcoding method for improving picture's visual quality when frames are skipped. The proposed method utilizes the relationships of motion vectors between successive frames, and 4 motion vectors a macroblock are considered in transcoding. Since the proposed method does not require any decoding process of encoded bit stream, the computational complexity for estimating motion vectors can be eliminated. Experimental results illustrate that as transcoded frames are getting far from I pictures and as the degree of motions in video sequences are getting higher, the picture quality by using our proposed method shows better performances than existing schemes.

☞ Keyword : Transcoding, H.263, Motion Vector, Frame Skipping

1. 서 론

많은 응용들에 있어서, 부호화된 비디오 비트스트림을 대상으로 하여 비트율을 제어할 필요가 있다[1]. 이에 대한 예로서는, 큰 전송율을 갖는 네트워크로부터 전달되어온 영상 비트스트림을 작은 전송율을 갖는 네트워크로 전달하는 경우를 들 수 있다. 이와 같이, 직접적으로 영상을 대상으로 하지 않고, 부호화된 비트스트림에 조작을 가하여 새로운 비트스트림을 만들어 내는 방법을 트랜스코딩(transcoding)이라 한다[2].

트랜스코딩을 통하여 비트율을 조절하기 위하여 부호화(encoding)된 모든 프레임을 복호화(decoding)하여 이를 다시 원하는 형태로 부호화하는 방법들이 제안되었다[3-6]. 이들 방법들은 복호화와 부호화를 함께 수행하여야 하므로 많은 연산을 필요로 하는 단점이 있다. 연산량을 줄이기 위하여, 모든 프레임을 복호화한 후 다시 부호화되는 프레임을 선택적으로 결정하는 프레임 제거(frame skipping)에 의한 트랜스코딩 방식이 제안되었다[7]. 이 방법은 모든 프레임을 부호화하지 않으므로 전체적인 연산량은 줄일수 있으나, 제거되지 않는 프

* 정 회 원 : 디지털스트림테크 주임연구원
neverdaii@hanmail.net(제 1저자)
** 정 회 원 : 아주대학교 정보통신전문대학원 부교수
bhroh@ajou.ac.kr(공동저자)
[2004/11/30 투고 - 2004/12/28 심사 - 2005/05/17 심사완료]
☆ 본 논문은 과학기술부 목적기초연구(R05-2002-000-00829-0) 지원으로 수행되었음.

레임에 대한 복호화와 부호화에 대한 연산 요구량은 달라지지 않게 된다. 이것은 복호화후에 영상을 다시 부호화시에 움직임 벡터(MV, Motion Vector)를 찾는데 소요되는 연산량이 매우 많이 요구되기 때문이다. [8]에서는 제거되는 프레임과 전후 프레임간의 움직임 벡터들간의 상관관계를 활용하여 움직임 벡터를 검색할 영역을 줄임으로써 움직임 벡터를 찾는데 소요되는 연산량을 감소시켜 전체적인 연산량을 줄이기 위한 방법을 제안하고 있다. 그러나, 이 방법도 부호화된 비트스트림을 픽셀 수준까지 복호화하여야 하는 부담을 내포하고 있다. 이와 같은 비트스트림의 복호화없이 움직임 벡터간의 상관 관계만을 활용하여 프레임 제거를 수행함으로써 연산량을 크게 줄일 수 있는 방식이 제안되었다[9]. 이 방법은 움직임이 매우 적은 경우에는 효율적인 성능을 보여주나, 움직임이 큰 경우에는 계산되어지는 움직임 벡터들이 가리키는 영역과 매크로블록 (macroblock) 과 일치하는 경우가 매우 드물게 되어 화질이 저하될 수 있다.

본 논문에서는 복호화 없이 움직임 벡터를 활용한 프레임 제거 방식[9]에서의 화질 저하 성능을 개선하기 위한 새로운 트랜스코딩 방식을 제안한다. 이를 위하여, H.263[10]에 정의된 하나의 매크로블록에 네 개의 움직임 벡터를 적용한 4MV 방식을 트랜스코딩에 적용한다. 제안 방식은 일반적인 하나의 매크로블록에 한 개의 움직임 벡터가 적용된 원래의 부호화된 비트스트림에 대하여 트랜스코딩을 적용한다. 그러나, 트랜스코딩시 제거된 프레임으로부터 산출되는 움직임 벡터가 지시하는 매크로블록과 일치하는 영역이 크게 불일치가 발생하는 경우 이를 4MV로 확장하여 재산출한다. 실험 결과는 본 논문에서 제안하는 방식은 기존의 움직임 벡터 방식들에 비하여 나은 성능 향상을 보여주며, 특히 움직임이 큰 비디오의 경우에 더 나은 화질 성능 향상을 보여준다.

본 논문의 제2장에서는 [9]에서 제안하는 트랜스코딩 방식과 이에 대한 문제점을 설명하고, 제3장에서는 본 논문에서 제안하는 방법을 기술한다. 제4장에서는 실험결과를 보이고, 마지막으로 제5장에서는 결론을 맺는다.

2. 배 경

여기에서는 기존의 움직임 벡터를 활용한 프레임 제거 방식들인 [8]과 [9]에 대하여 간략히 설명하고, 이에 대한 문제점에 대하여 고찰한다.

프레임 제거 방식은 프레임을 제거함으로써 전송 비트율을 낮추기 위한 방식으로서, 그림 1에 보인 바와 같이 n-번째 프레임이 제거되는 경우, (n+1)-번째 프레임은 복호화시 (n-1)-번째 프레임을 참조하도록 움직임 벡터(MV)값을 수정하여야 한다. 이때, MV는 방향성을 갖는 벡터값이므로, 수정되는 MV값은 다음과 같이 나타낼 수 있다.

$$MV(new) = MV(now) + MV(skip) \quad (1)$$

여기에서 MV(now)는 현재 프레임의 해당 매크로블록의 움직임 벡터고, MV(skip)은 MV(now)에 의하여 지정되는 제거되는 프레임의 매크로블록이 가리키는 움직임 벡터다. MV(now)는 현재 프레임의 매크로블록이 참조하게 될 제거된 프레임 이전의 블록 위치를 나타내도록 수정된 움직임 벡터값이다.

일반적으로 움직임 벡터의 산출은 지정된 검색 영역 (H.263의 경우, -15에서 15사이)내의 모든 블록들을 비교하여 가장 작은 차이를 갖는 블록을 찾아내는데 이것은 많은 연산을 요구한다. 그러나, 프레임 제거시 움직임 벡터를 산출하기 위하여 검색할 영역은 식 (1)에서 지정되는 MV 값의 근처에 있을 확률이 아주 높기 때문에, 검색 영역은 상당히 축소될 수 있다. [8]에서는 식 (1)을 사용하여 구한 MV 위치를 기준으로 -2에서 2 사이를 검색 영역으로 하여 MV를 찾아냄으로써 연산량을 줄이고 있다. 그러나, 이 방법은 새로운 MV를 산출하기 위하여 제거되지 않는 프레임들을 복호

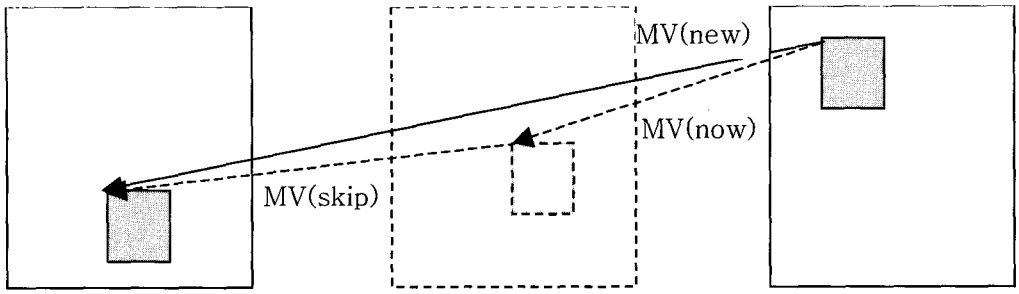


그림 1. 프레임 제거에 따른 새로운 MV의 지정 (그림이 제대로 나오지 않았음. 정상적인 그림으로 대체 요망)

화 해야만 한다.

[9]에서는 프레임들을 복호화하지 않고, 수정된 MV는 식 (1)에 의하여 구해지는 MV(new)를 그대로 적용하고, MV(new)에 의하여 지정되는 이전 프레임 블록과 현재 프레임 블록들의 TCOEFF (Transform Coefficients) 값들을 DCT(Discrete Cosine Transform) 영역이나 VLD (Variable Length Decoding) 영역까지만 복호화를 하여 구해진 값들을 더한 후 이에 대한 새로운 TCOEFF 값을 구하고 있다. 이와 같이 함으로써 픽셀 수준까지 복호화하는 연산과 움직임 벡터를 찾아내는 연산량을 줄일수 있게 된다. 이 방식은 움직임이 적은 비디오 비트스트림일 경우는 아주 좋은 성능을 나타내나, 움직임이 많은 비디오의 경우에는 화질이 많이 떨어지는 단점이 있다.

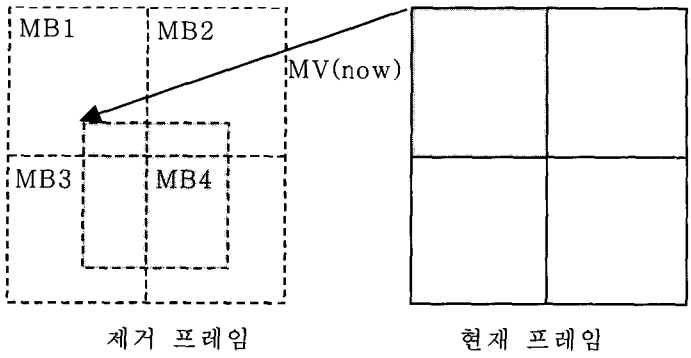
[8]과 [9]에 의한 방법들이 움직임이 많은 경우에 화질이 떨어지는 이유는 다음과 같다. 움직임 벡터 정보들은 매크로블록(MB) 단위로 유지된다. 따라서, 그림 2에서와 같이 현재 프레임의 MV가 가리키고 있는 제거 프레임의 영역과 제거 프레임 상의 MB의 위치가 일치하지 않고, 네 부분의 MB에 걸쳐있게 될 가능성이 많다. 식 (1)의 MV (new)를 계산하기 위하여 요구되는 MV(skip)값은 이들 네 개의 MB들중의 하나에 의하여 정해지므로 이들중 적절한 하나를 선택하여야 한다. [8]과 [9]에서는 네 개의 MB들중에서 가장 많이 겹치는 MB를 선택하고 있다. 이때, 선택되지 않는 MB와 MV가 가리키는 영역이 겹치는 부분이 커질수록, 즉 선택된 MB와 선택되지 않은 MB의 MV가 많이 틀릴수록 화질은 더 큰 폭으로 떨어진다. 이것은 움직임이 많은 영상일수록 이러한 경우가 더 많이 발생하게 될 가능성이 높고, 이에 따라 더 큰 폭으로 화질이 떨어지게 된다.

3. 제안하는 네 개의 움직임 벡터를 활용한 프레임 제거 방식

3.1. 제안 알고리즘

본 절에서는 본 논문에서 제안하는 프레임 제거 방식에 대하여 설명한다. 본 논문에서 제안하는 방식의 기본 사상은 MV에 의하여 지정되는 영역에 의하여 지정되는 네 개의 MB들 중, 선택되지 않는 영역의 크기를 최소화하도록 하는 것이다. 이를 위하여, H.263의 확장 모드에 정의된 한 MB 당 네 개의 MV를 사용하는 advanced prediction mode[10]를 적용한다. 본 논문에서의 트랜스코딩을 위하여 네 개의 MV를 적용하지만, 트랜스코딩의 대상이 되는 원본 압축 스트림은 한 개의 MV를 통하여 부호화가 되어 있음에 주의한다. 즉, 제안하는 방법은 기존의 트랜스코딩 방법들에서 적용하는 동일한 원본 압축 스트림을 사용한다.

본 논문에서 제안하는 방법을 설명하기 위하여 다음과 같은 변수들을 정의한다. 이들 사용된 변



〈그림 2〉 움직임 벡터가 가리키는 영역과 매크로블록과의 불일치

수들에 설명을 그림 3에 도식화하여 나타내었다.

MB_n(now) 트랜스코딩할 현재 프레임의 n-번째 매크로블록

MB_{nk}(now) MB_n(now)를 8x8로 4등분한 네 개의 서브블록들중 k-번째의 서브블록 (k=1,2,3,4)

MB^n(skip) MB_n(now)에 대응되는 제거 프레임내의 16x16 블록으로서 MB_n(now)의 움직임 벡터에 의하여 결정됨

MB^n_m(skip) MB^n(skip)와 겹쳐지게 되는 제거 프레임의 네 개의 매크로블록들 중 m-번째 매크로블록 (m=1,2,3,4)

MB^n_{ml}(skip) MB^n_m(skip)을 8x8로 4등분한 네 개의 서브블록들중 l-번째의 서브블록 (l=1,2,3,4)

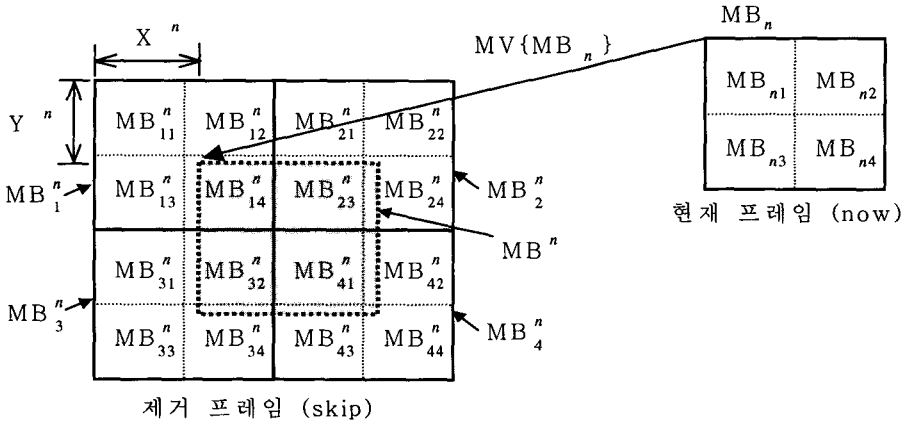
3 X^n(skip) MB^n(skip) 첫번째 픽셀의 수평측 위치와 MB^n_1(skip)의 첫번째 픽셀의 수평측 위치와의 차이값

Y^n(skip) MB^n(skip) 첫번째 픽셀의 수직측 위치와 MB^n_1(skip)의 첫번째 픽셀의 수직측 위치와의 차이값

MV_n(new) 프레임 제거후 계산된 n.3 MB_n(now)에 대한 수정된 움직임 벡터

MV_{nk}(new) 프레임 제거후 계산된 MB_{nk}(now)에 대한 수정된 움직임 벡터 (k=1,2,3,4)

2장에서 설명한 바와 같이, MB_n(now)에 대응되는 제거할 프레임내에서의 16x16 블록 Equation.3



〈그림 3〉 논문에서 사용된 변수들

$MB^n(skip)$는 결정되나, $MB^n(skip)$은 자신의 앞 프레임내에서 대응시킬 16x16 블록에 대한 움직임 벡터 정보를 보유하고 있지 않으므로, [8]과 [9]에서는 $MB^n(skip)$와 겹쳐서 나타나게 되는 네개의 매크로블록들 $MB_m^n(skip)$ 중 가장 많이 겹치는 매크로블록을 선택하여 이 매크로블록의 움직임 벡터를 사용하여, $MB_n(now)$에 대한 새로운 움직임 정보를 갱신하고 있다. 이때, 선택되지 않는 MB와 MV가 가리키는 영역이 겹치는 부분이 커질수록, 화질은 더 큰 폭으로 떨어진다. 본 논문에서는 이러한 겹치는 영역을 최소화하기 위하여 다음과 같은 방법을 제안한다.

이를 위하여, MV{A}는 해당 매크로블록 또는 서브블록 A의 움직임 벡터값을 나타내는 함수로서 나타내기로 한다. 현재 프레임과 제거될 프레임의 매크로블록들을 8x8 크기를 갖는 네개의 서브블록들로 구분하고, 각 구분된 서브블록들에 대한 움직임 벡터값들은 이들을 포함하는 매크로블록에 대한 움직임 벡터값들로 설정한다. 즉,

$$MV\{MB_{nk}(now)\} = MV\{MB_n(now)\}, \; k=1,2,3,4 \quad (2)$$

$$MV\{MB_{ml}^n(skip)\} = MV\{MB_m^n(skip)\}, \; m=1,2,3,4, \; l=1,2,3,4 \quad (3)$$

이로부터, 각 서브블록들에 대한 움직임 벡터값은 다음과 같이 갱신된다.

$$MV_{nk}(new) = MV\{MB_{nk}(now)\} + f_{MV}\{X^n(skip), Y^n(skip)\}, \; k,m,l=1,2,3,4 \quad (4)$$

여기에서 $f_{MV}\{X^n(skip), Y^n(skip)\}$는 Equation.3 $X^n(skip)$과 $Y^n(skip)$에 의하여 결정되는 제거 프레임내의 서브블록의 움직임 벡터값을 나타낸다. 제거 프레임내의 서브블록의 결정은 $MV\{MB_n(now)\}$에 의하여 지정되는 제거 프레임내에서의 매크로블록 영역을 8x8 크기의 네 개 서브블록들로 구분하였을때, 각 서브블록과 $MB_{ml}^n(skip)$ (m=1,2,3,4, l=1,2,3,4)들 중에서 가장 많이 영역이 겹치는 서브블록이 되고, 이 결정된 서브블록의 움직임 벡터값이 식 (4)의 $MV_{nk}(new)$ 계산에 적용된다.

프레임이 연속해서 skip되는 경우에 대한 처리는 한 개 skip된 경우를 연속해서 적용하면 된다. 현재 i-번째 프레임이전의 h개의 프레임들 (i-1, i-2, …, i-h 번째 프레임)을 제거하여야 하는 경우에 대하여 i-번째 프레임의 n-번째 매크로블록의 k-번째 서브블록의 움직임 벡터를 갱신하는 과정은 다음과 같다.

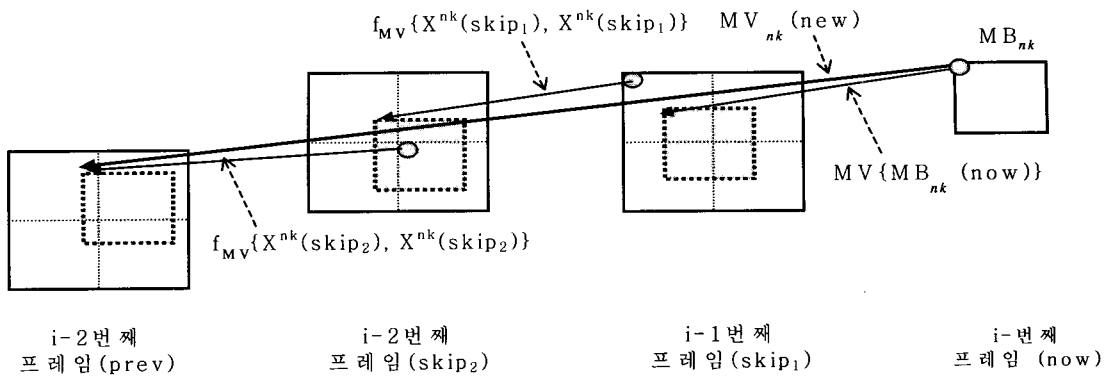
$MV_{nk}(new_j)$를 $MB_{nk}(now)$가 j-번째로 제거될 (i-j)-번째 프레임내에서 매칭되는 서브블록을 가리키도록 설정되는 움직임 벡터값이라 하고, $MV\{MB_{nk}(new_{j-1})\}$를 이 서브블록에 저장된 움직임 벡터값이라 하기로 한다. 그리고, $f_{MV}\{X^{nk}(skip_j), Y^{nk}(skip_j)\}$를 $MV\{MB_{nk}(new_{j-1})\}$에 의하여 지정되는 j-번째로 제거될 프레임내의 서브블록 위치와 가장 일치하는 움직임 벡터 정보를 유지하는 서브블록이 보유한 움직임 벡터로 정의한다. 이로부터, 다음과 같은 연속된 제거 프레임을 갖는 경우에 대한 새로운 움직임 벡터값을 구할 수 있다.

$$MV_{nk}(new) = \sum_{j=1}^{h} MV_{nk}(new_j) \quad (5)$$

$$MV_{nk}(new_j) = \begin{cases} MV\{MB_{nk}(now)\} + f_{MV}\{X^{nk}(skip_1), Y^{nk}(skip_1)\} & , j=1 \\ MV\{MB_{nk}(new_{j-1})\} + f_{MV}\{X^{nk}(skip_j), Y^{nk}(skip_j)\} & , j>1 \end{cases} \quad (6)$$

여기에서 h는 연속해서 제거될 프레임의 개수를 나타낸다. 그림 4에는 연속해서 제거될 프레임이 두개, 즉 h=2인 경우에 대한 식 (5),(6)의 과정을 도식적으로 나타내었다.

움직임 벡터값이 갱신된 후에는 TCOEFF 값도 갱신을 하여야 한다. 이를 위하여, $E_{nk}(now)$를 서브블록 $MB_{nk}(now)$의 TCOEFF값에 대한 디코딩



〈그림 4〉 연속된 프레임의 제거 (h=2)

된 값으로 정의하기로 한다. 마찬가지로, $E_{nk}(new_j)$를 $f_{MV}\{X^{nk}(skip_j), Y^{nk}(skip_j)\}$를 보유한 서브블록의 TCOEFF값에 대한 디코딩된 값으로 정의하기로 하면, 다음과 같은 $E_{nk}(new)$를 구하게 된다.

$$E_{nk}(new) = \sum_{j=1}^{h} E_{nk}(new_j) \quad (7)$$

이로부터, 다음과 같은 갱신된 TCOEFF값을 구할 수 있다.

$$TCOEFF_{nk}(new) = VLC(Q(DCT(E_{nk}(new)))) \quad (8)$$

여기에서, DCT(X)는 X에 대한 DCT 연산을, Q(X)는 양자화 연산을, 그리고 VLC(X)는 VLC (Variable Length Coding) 연산을 나타낸다.

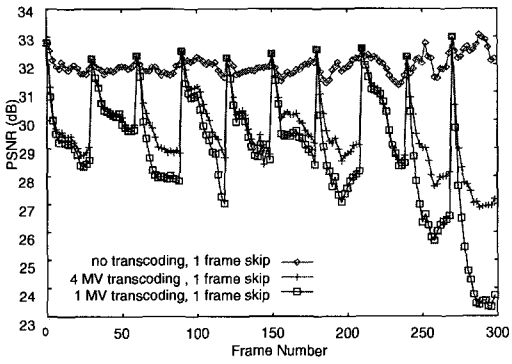
이와 같이, 본 논문에서 제안하는 방식은 네 개의 MV를 사용하는 방법을 확장 적용하고는 있으나, 트랜스코딩에 적용되는 원래의 부호화된 비트열이 네 개의 MV를 사용하여야 하는 제한은 없다. 즉, 원래의 부호화된 비트열은 일반적인 하나의 매크로블록당 하나의 움직임 벡터를 사용하는 구조를 기본적인 대상으로 한다.

4. 실험 결과

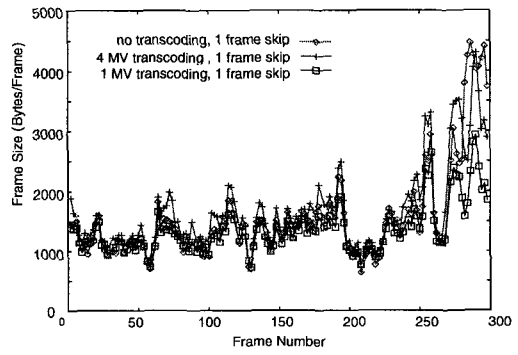
실험을 위하여, 본 논문에서 제안된 방식을 구현하여 H.263 TMN 코더에 적용하였다. 실험에는 CIF (Common Intermediate Format) 크기의 Foreman과 Hall_monitor 비디오 시퀀스를 사용하였다. 트랜스코딩 적용을 위한 실험 환경은 다음과 같다. 우선 총 300장의 비디오 프레임에 대하여 정상적인 방법으로 부호화를 하였다. 이때, 양자화 계수는 모두 13으로 하였으며, 부호화되는 I-픽쳐들간의 프레임 간격은 30으로 하였다. 트랜스코딩은 이와 같이 정상적으로 부호화된 비트스트림에 대하여 적용하였다. 이때, 트랜스코딩을 위하여 프레임 제거시에 I-픽쳐로 부호화된 프레임은 제거시키지 않고 유지를 시켰다.

그림 5는 한 프레임씩 걸러서 제거하였을 때의 성능을 보여준다. 한 프레임씩 걸렀다는 것의 의미는 프레임 번호가 0, 1, 2, 3, 4, 5, … 와 같다고 할 때, 0, 2, 4, 6, … 의 프레임들만 부호화하였음을 의미한다. 다음의 방식들을 대상으로 성능을 비교하였다.

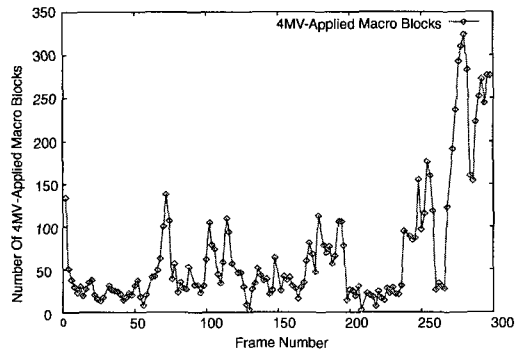
- no transcoding (ORIG) : 한 프레임씩 건너뛴 프레임들에 대하여 정상적인 H.263 부호화를 수행한 방식
- 1MV transcoding (1MV) : 모든 프레임들(건너뛰지 않은 경우)을 정상적으로 H.263 부호화한 비트스트림을 대상으로 1MV 방식 [9]을 사용하여 해당 프레임들을 제거하여 트랜스코딩한 방식



(a) PSNR



(b) 프레임 크기



(c) 4MV 적용 매크로블록 수

〈그림 5〉 한 프레임씩 제거한 경우 (Foreman)

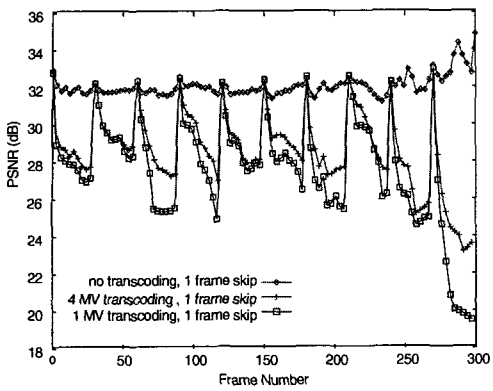
- 4MV transcoding (4MV) : 모든 프레임들(건너뛰지 않은 경우)을 정상적으로 H.263 부호화한 비트스트림을 대상으로 제안된 4MV 방식을 사용하여 해당 프레임들을 제거하여 트랜스코딩한 제안하는 방식

그림 5 (a)는 PSNR (Peak Signal to Nose Ratio) 성능을, 그림 5 (b)는 프레임당 부호화된 바이트 크기를 보여준다. 그림 5 (b)에는 I-픽쳐에 해당하는 프레임의 크기는 나타내지 않았다. 이것은 I-픽쳐 프레임 크기는 P-픽쳐들에 비하여 매우 크므로, 이를 표현하게 될 경우, 트랜스코딩이 적용되는 P-픽쳐 프레임들이 상대적으로 잘 나타나지 않기 때문이다. 전체적으로 I-픽쳐에서 멀어질수록 트랜스코딩된 시퀀스들의 PSNR값이 점차적으로 저하됨을 볼수 있으나, 제안된 4MV를 사용한 방식의 PSNR이 1MV를 사용한 방식들에 비하여 PSNR의 저하가 덜 심함을 알 수 있다. 반면에, 부호화된 크기를 보면 4MV를 사용한 방식이 1MV를 사용한 방식에 비하여 더 크게됨을 볼 수 있다. 이것은 그림 5 (c)의 4MV 방식이 적용되는 매크로블록의 수에 보인바와 같이, 4MV가 적용되는 매크로블록의 수가 많아질수록 움직임 벡터를 나타내는 비트수가 많아지기 때문이다.

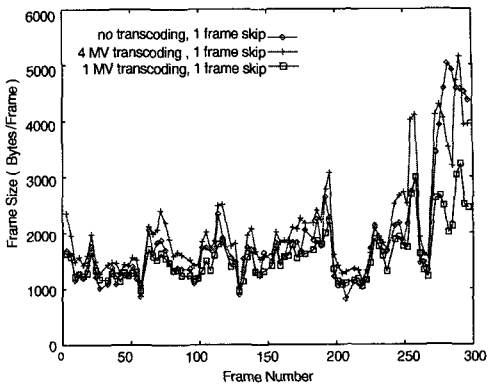
그림 6 과 그림 7에는 각각 두 프레임씩 연속하여 제거한 경우와 세 프레임씩 연속하여 제거한 경우에 대한 PSNR과 프레임 크기 성능을 비교하였다. 연속되어 제거되는 프레임의 수가 많아질수록, 트랜스코딩하지 않은 경우에 비하여 PSNR 저하가 I-픽쳐에서 멀어질수록 더 심하나, 4MV를 적용한 방식이 1MV를 적용한 방식에 비하여는 PSNR 성능이 매우 뛰어남을 알수 있다. 반면에, 부호화되는 프레임 크기 측면에서는 4MV를 사용하는 방법이 더 많은 비트를 사용함을 보여준다.

전체적으로 볼 때, I-픽쳐에서 멀어질수록 트랜스코딩에 의한 PSNR 성능 저하가 더 심해지면서, 1MV와 4MV를 사용한 방식들간의 상대적인 PSNR 차이도 커지고 있다. 이에 대한 성능을 비교하기 위하여 그림 8에는 I-픽쳐들로부터의 거리에 따라 1MV 방식에 대한 제안된 4MV 방식의 PSNR 차이와 프레임크기 비율이 어떻게 변화하는지를 나타내었다.

그림 8 (a)에는 4MV 방식의 PSNR값과 1MV

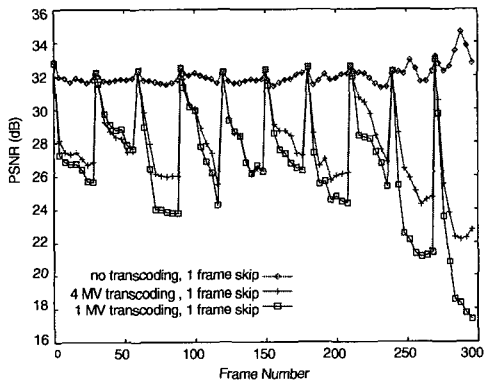


(a) PSNR

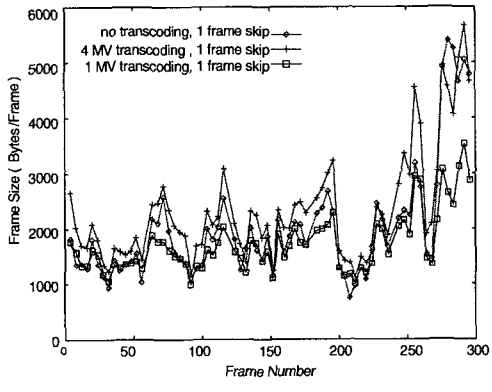


(b) 프레임 크기

〈그림 6〉 두 프레임씩 연속하여 제거 시킨 경우 (Foreman)

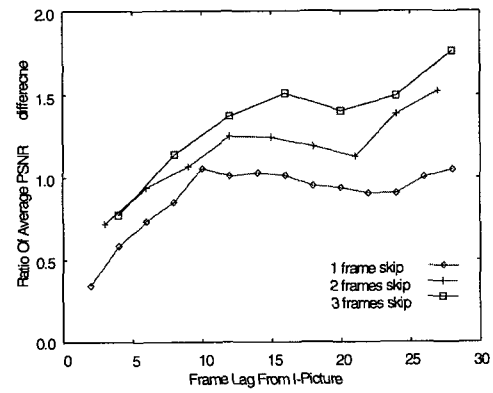


(a) PSNR

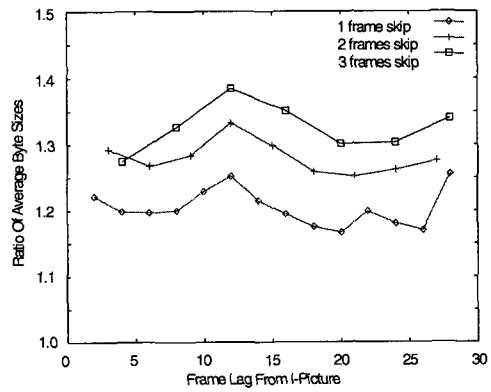


(b) 프레임 크기

〈그림 7〉 세 프레임씩 연속하여 제거한 경우 (Foreman)

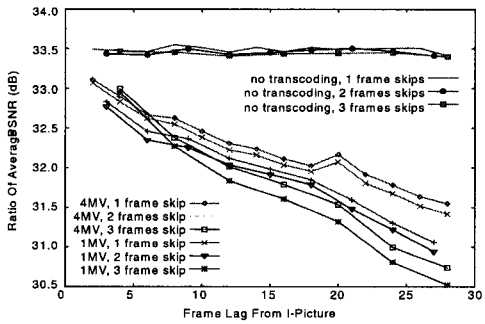


(a) 평균 PSNR 차이

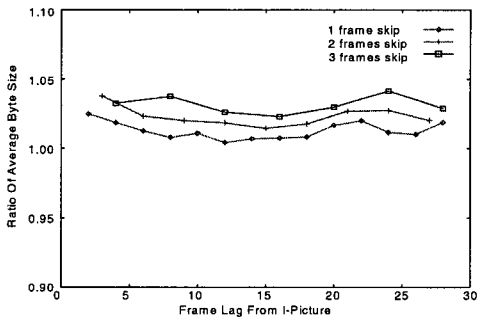


(b) 평균 프레임 크기 비율

〈그림 8〉 I-픽쳐로부터의 거리에 따른 방식간 성능 비교 (Foreman)



(a) 평균 PSNR



(b) 평균 프레임 크기 비율

〈그림 9〉 I-픽쳐로부터의 거리에 따른 방식간 성능 비교 (Hall_monitor)

(a) 원래 비트스트림 영상

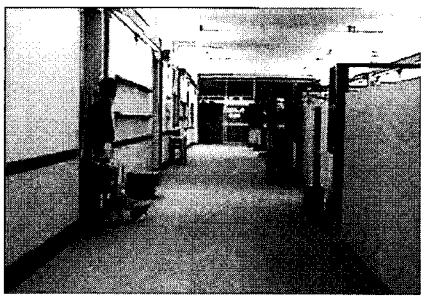
(b) 1MV 방식

(c) 제안된 4MV 방식

〈그림 10〉 부호화된 영상 비교 (Foreman 72번째 영상)

(a) 원래 비트스트림 영상

(b) 1MV 방식



(c) 제안된 4MV 방식

〈그림 11〉 부호화된 영상 비교 (Hall_monitor)

〈표 1〉 Foreman 영상의 평균 PSNR 및 Frame Size 비교

skip frame 수 / methods	1		2		3	
	PSNR (dB)	Frame Size (Bytes)	PSNR (dB)	Frame Size (Bytes)	PSNR (dB)	Frame Size (Bytes)
1MV	28.40	1754.8	27.24	2137.2	26.71	2403.2
4MV	29.34	2050.5	28.30	2589.9	27.79	2949.0

〈표 2〉 Hall_monitor 영상의 평균 PSNR 및 Frame Size 비교

skip frame 수 / methods	1		2		3	
	PSNR (dB)	Frame Size (Bytes)	PSNR (dB)	Frame Size (Bytes)	PSNR (dB)	Frame Size (Bytes)
1MV	32.32	1051.7	32.07	1397.6	31.96	1668.5
4MV	32.37	1056.4	32.20	1416.9	32.13	1691.1

방식의 PSNR 값과의 차이를 I-픽쳐와의 거리에 대하여 나타내었다. I-픽쳐와의 거리가 멀어질수록 제안된 4MV 방식의 PSNR값이 1MV 방식의 PSNR값에 비하여 점차적으로 커지고 있으며, 이 차이는 연속되어 제거되는 프레임의 크기가 커질수록 더 커지고 있다. 반면에 그림 8 (b)에 나타낸 바와 같이, 프레임 크기의 비율은 I-픽쳐 거리의 변화에 크게 변화하지 않고 일정하게 유지됨을 알 수 있다.

그림 9에는 Hall_monitor 비디오 시퀀스에 대하여 적용한 결과를 보여준다. 그림 9 (a)는 I-픽쳐와의 거리에 따른 방식별 평균 PSNR 값들을 보여준다. 트랜스코딩을 하지 않은 프레임 제거 방식들(no transcoding)은 PSNR 값에 크게 변동이 없으나, 트랜스코딩을 적용한 방식들은 Foreman 비디오 시퀀스의 결과와 마찬가지로, I-픽쳐와의 거리가 멀어질수록 PSNR이 저하된다. PSNR 저하 정도는 제안된 4MV 방식이 1MV 방식에 비하여 낮으나 그 차이는 그리 크지 않다. 또한, 그림 9 (b)에 보인 바와 같은 1MV 방식에 대한 4MV 방식의 프레임크기 비율은 I-픽쳐와의 거리에 무관하게 거의 유사한 결과를 나타내 주고 있다. 이것은 Hall_monitor 비디오 시퀀스는 움직임이 매우 적은 영상들로 이루어지고 있어, 95%이상의 많은 매크로블록들이 값이 0인 움직임 벡터를 갖고 있기 때문에 4MV가 적용되는 매크로블록의 수가 매우 적기 때문에서 기인한다.

그림 10과 그림 11에는 각 방식들에 대하여 부호화된 영상을 나타내었다. 그림 10의 Foreman 영상의 경우를 보면, 제안된 4MV 방식이 전체적인 화면에서 1MV 방식보다 더 나은 화질을 보여줌을 알 수 있다. 특히, Foreman 영상에서 움직임이 많이 발생하는 얼굴과 주변에서 큰 화질 차이가 발생하는 것을 볼수 있다. 또한, 그림 11의 Hall_monitor 영상의 경우는 다른 부분에서는 큰 차이가 없으나, 움직임이 발생한 사람부분에서 1MV 방식의 왜곡 정도가 4MV 방식에 비하여 더 큼을 볼수 있다.

표 1과 표 2에는 각각 Foreman 영상과 Hall_monitor 영상의 앞의 결과들에 대하여 구한 평균 PSNR과 Frame Size를 비교하였다. Skip 되는 프레임의 개수가 많을수록 PSNR은 감소하나, 제안된 4MV는 1MV 방식에 비하여 더 나은 평균 PSNR 성능을 보여준다. 그러나, 4MV 방식이 움직임이 큰 경우에는 4개의 움직임 벡터를 사용하므로, 이의 표현을 위한 정보량이 증가하여 평균 Frame Size는 증가하고 있다. 일반적으로 트랜스코딩에서 프레임을 제거하면 전송율이 급격히 감소하게 되나, 그만큼의 화질 저하가 발생한다. 그러나, 제안된 방법은 데이터량의 감축에 있어서 1MV 방식에 비하여 다소의 정보량 증가는 있으나 화질의 개선효과가 뚜렷하게 나타나므로, 화질

을 우선시 하는 전송율 제어 응용에 적합하다.

이상의 결과들을 종합하여 보면, 제안된 4MV 방식은 1MV 방식에 비하여 매우 개선된 PSNR 성능을 보여준다. 이러한 성능 개선 효과는 I-픽쳐에서 멀어질수록, 연속해서 제거되는 프레임의 수가 많아질수록 더 크게 나타난다. 반면에, 부호화되는 프레임크기는 4MV 방식이 1MV 방식에 비하여 다소 큰 값을 보여준다. 또한, 움직임이 큰 비디오 시퀀스가 움직임이 작은 비디오 시퀀스에 비하여 PSNR과 프레임크기에 대한 차이는 더 커진다. 연산의 복잡성 측면에서 제안된 4MV 방식은 각 매크로블록에 대하여 네 개의 움직임 벡터를 고려하는 경우가 발생하므로, 1MV 방식에 비하여 더 많은 연산이 요구된다. 그러나, 제안된 4MV 방식이나 1MV 방식들은 모두 이미 부호화되어 구해진 움직임 벡터값을 사용한 가감 연산만 수행하므로, 이러한 연산의 증가량은 무시할 수 있다. 일반적으로 영상의 부호화에서 처리 시간의 90%이상은 움직임 벡터값을 찾아내는데 걸리는 시간이고, 나머지 5% 이상은 DCT 연산에 소요되는데, 제안 방법이나 1MV 방식 모두 움직임 벡터 산출이 필요없고 동일한 DCT 연산량이 요구되므로, 이러한 증가량은 전체 처리 시간 차원에서 그리 크게 나타나지 않을 것이다.

5. 결 론

본 논문에서는 프레임간의 움직임 벡터의 상관관계를 활용한 프레임 제거 방식에서의 화질 저하 성능을 개선하기 위한 새로운 트랜스코딩 방식을 제안하였다. 제안 방법은 네 개의 움직임 벡터를 적용하여 화질 저하 성능을 개선시켰으며, 트랜스코딩시 부호화된 비트스트림으로부터 영상을 복호화할 필요가 없으며, 따라서 움직임 벡터를 산출하는 복잡한 연산이 요구되지 않는다. 본 논문에서의 제안 방법이 트랜스코딩시에 네 개의 움직임 벡터를 고려하지만 원래의 부호화된 비트스트림은 매크로블록당 한 개의 움직임 벡터를 갖는 일반적인 부호화 방식을 대상으로 하고 있다.

본 논문에서 제안된 방법은 부호화된 비디오 비트스트림을 사용 빈도에 따라 변화하는 인터넷의 가용 대역폭에 비트율을 적응적으로 맞추어 실시간으로 서비스하는데 있어서 매우 적합한 방법들중의 하나가 될 것으로 생각된다. 이러한 이유는, 인터넷의 가용 대역폭이 줄어들었을 때, 프레임 제거 트랜스코딩 방식을 사용하여 비트율을 낮추게 되면, 화질도 함께 떨어지게 되므로, 비트율을 낮추는 효과는 이룩할수 있지만 화질 측면에서의 불이익을 감수하여야 한다. 그러나, 제안된 방법은 기존의 방식에 비하여 화질의 현저한 개선 효과를 보여주면서, 비트율의 감소를 이룩하므로, 인터넷과 같이 가변하는 네트워크에 맞추어 적용하는데 더 적합하다. 실제적으로 제안된 방법을 가변하는 인터넷 대역폭에 맞추어 프레임 제거율과 비트율을 조절하도록 하기 위하여는 더 연구가 될 필요가 있다.

참 고 문 헌

[1] D. Wu, Y. T. Hou, and Y. Zhang, "Transporting Real time Video over the Internet: Challenges and Approaches," Proc. IEEE, Vol. 88, No. 12, December 2000, pp.1855 1875

[2] A. H. Sadka, Compressed Video Communications, John Wiley & Sons, Inc., 2002

[3] A. Vetro, C. Christopoulos, and H. Sun, 'Video Transcoding Architectures and Techniques: an Overview', IEEE Signal Processing Magazine, Vol.20, No.2, March 2003.

[4] A. Eleftheriadis, "Dynamic Rate Shaping of Compressed Digital Video," Ph D. dissertation, Dept. Elec. Eng., Columbia Univ., New York, June 1995.

[5] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG Compressed Bitstream Scaling," IEEE Trans. Circuits Syst. Video Technol., Vol. 6, No. 2, pp. 191 199, Apr. 1996.

[6] P. Yin, M. Wu, and B Lui, 'Video Transcoding by Reducing Spatial Resolution', IEEE Int. Conf. Image Processing, Vancouver, BC, Canada, Vol. 1, Oct. 2000, pp. 972 975

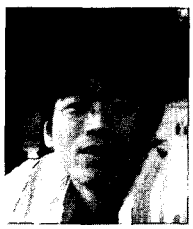
[7] K. Fung, Y. Chan, and W. Siu, "New Architecture for Dynamic Frame Skipping Transcoder," IEEE Tr. Image Processing, Vol. 11, No 8, Aug. 2002

[8] J. Youn, M. Sun, and C. Lin, "Motion Vector Refinement for High Performance Transcoding," IEEE Tr. Multimedia, Vol. 1, No. 1, pp. 30 40, Mar. 1999

[9] J. Youn, M. Sun, and C. Lin, "Motion Estimation for High Performance Transcoding," IEEE Tr. Consumer Electron., Vol. 44, No.3, pp. 649 658, Aug. 1998.

[10] ITU T Recommend. H.263, Video coding for low bitrate communication, May 1997

◐ 저 자 소 개 ◑



박 용 대 (Yong-Dae Park)
2002년 아주대학교 정보및컴퓨터공학부 졸업(학사)
2004년 아주대학교 정보통신전문대학원 졸업(석사)
2004년 ~ 현재 디지털스트림테크놀로지 주임연구원
관심분야 : 유/무선 인터넷 멀티미디어 통신
E-mail : neverdaii@hanmail.net

노 병 희 (Byeong-hee Roh)
1987년 한양대학교 전자공학과 졸업(학사)
1989년 한국과학기술원 전기및전자공학과 졸업(석사)
1998년 한국과학기술원 전기및전자공학과 졸업(박사)
1989년 ~ 1994년 한국통신 통신망연구소
1998년 ~ 2000년 삼성전자
2000년 ~ 현재 아주대학교 정보통신전문대학원 부교수
관심분야 : 유/무선 인터넷 멀티미디어 통신및 응용, 트래픽 제어, 유비쿼터스 네트워킹, RFID 네트워킹, 인터넷보안
E-mail : bhroh@ajou.ac.kr