프레임율 변환을 위한 개선된 트랜스코딩 기법

정회원 양 시 영*, 정 제 창*

Enhanced Transcoding Technique for Frame Rate Conversion

Siyoung Yang*, Jechang Jeong* *Regular Members*

요 약

네트워크에 의해서 필요한 비트율로 줄이거나 터미널의 제한된 한계를 만족하기 위해서, 비디오 비트 스트림의 시간적 해상도를 줄이는 것이 사용된다. 본 논문에서는 압축된 비디오 스트림의 감소된 해상도 트랜스코딩의 문제점에 대해서 논의하고, 시간적 해상도 변환을 위한 트랜스코딩 기법을 논의한다. 트랜스코딩의 속도를 증가시키기 위해서, 비디오 트랜스코더는 보통 입력된 비디오 스트림으로부터의 움직임 벡터를 재사용한다. 본 논문에서는 코딩된 프레임의 높은 화질을 유지하기 위해서 강화된 움직인 재추정 기법을 제안한다. 실험결과는 감소된 프레임율을 가진 비디오 트랜스코더를 위한 저 복잡도를 가지면서도 성능이 개선된 것을 보여준다.

Key Words : Transcoding, Frame-skipping, Motion re-estimation, E-FDVS, Transcoder

ABSTRACT

To reduce the bit-rate requirements imposed by a network or satisfy processing limitations imposed by a terminal, Conversion the temporal resolution of a video bit stream is a technique that may be used. This paper discusses the problem of reduced resolution transcoding of compressed video bit streams, and discussed the technique for temporal transcoding. To speed up this operation, a video transcoder usually reuses the coded motion vectors from the input video bit stream. In this paper we propose an enhanced motion re-estimation technique to maintain higher quality of coded frames. The performance of experimental results can be improved while maintaining low computational complexity for a reduced frame rate video transcoder.

Ⅰ. 서 론

일반적으로 트랜스코딩은 하나의 코딩된 신호를 다른 형식의 신호로 변환하는 것으로 정의된다. 형식으로는 그림 1과 같이 비트율, 프레임율, 공간적 해상도, 코딩 신택스(coding syntax), 그리고 컨텐츠 등이 있다. 트랜스코딩은 다른 종류의 접속 방식과 링크를 가진 인터넷 사용자들을 위한 범용 멀티미디어 접속 방법을 제공하는 핵심기술이기도 하다. 통신 네트워크와 네트워크 접속 터미널 사이의 호환을 맞추기 위해서는 압축된 비디오의 비트율 변환뿐만 아니라 시간적 해상도를 변환할 필요가 있다[1-2].

시간적 해상도 변환 트랜스코딩을 위해서 입력 비디오에는 존재하지 않는 새로운 움직임 벡터의 세트를 얻어야 한다. 움직임 추정을 위한 대표적인 블록 정합 알고리듬으로 전역탐색 기법이 있지만, 높은 성능에 비해 너무나 높은 복잡도를 가진다. 전역탐색 기법대신 비슷한 블록 왜곡을 가지면서 속도를 향상하기 위한 많은 알고리듬이 제안되었다. Three Step Search (TSS)[5], New Three Step Search (NTSS)[6], Four Step Search (4SS)[7],

※ 본 연구는 서울시 산학연협력사업으로 구축된 서울 미래형콘텐츠 컨버전스 클러스터 지원으로 수행되었습니다.
* 한양대학교 전자통신전파공학과 영상통신 및 신호처리 연구실 (wirbel@ece.hanyang.ac.kr)
논문번호 : KICS2006-10-450, 접수일자 : 2006년 10월 24일, 최종논문접수일자 : 2008년 7월 9일

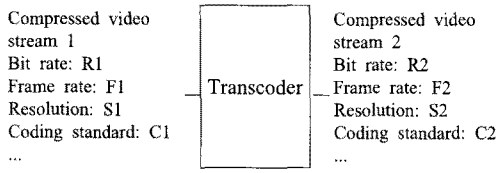

그림 1. 비디오 트랜스코더를 사용한 형식 변환
Fig. 1. Format conversion using a video transcoder.

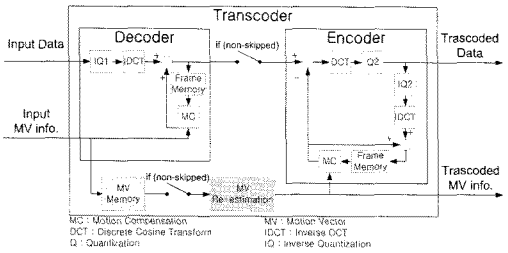

그림 2. 프레임율 감소를 위한 직렬 픽셀 영역 트랜스코딩 구조
Fig. 2. Illustration of cascaded pixel-domain transcoding architecture for frame-rate conversion

Cross-Diamond Search (CDS)[8]등이 대표적인 고속 움직임 추정 기법들이다. 그러나 고속 움직임 추정 기법들은 고속 알고리듬에도 불고하고, 트랜스코딩에 적용하기에는 높음 복잡도와 낮은 성능을 보인다. 시간적 해상도 변환인 프레임 스킵은 스킵되지 않고 남아있는 프레임에 남아있는 비트를 할당함으로써, 남아있는 프레임의 화질을 유지하는 효율적인 기술로서 사용된다[3-4]. 그림 2는 프레임율 변환을 위한 직렬 픽셀 영역 트랜스코딩 구조를 보여준다.

입력되는 비디오 비트 스트림이 프레임율이 변화된 비디오 비트 스트림으로 변환 될 때, 스킵된 프레임의 움직임 벡터는 트랜스코딩된 비트 스트림에는 더 이상 존재하지 않기 때문에, 스킵되지 않은 프레임의 움직임 벡터는 더 이상 사용할 수 없다. 따라서 새로운 움직임 벡터를 재추정할 필요가 있다. 속도를 증가시키고 성능을 향상하기 위해서, 쌍선형 보간법 (Bilinear Vector Interpolation, BVI)과 전방향 지배적인 벡터 선택 법 (Forward Dominant Vector Selection, FDVS)와 같은 기존의 프레임 스킵 기법은 입력 비디오 비트 스트림의 움직임 벡터를 사용한다.

본 논문에서는 개선된 프레임 스킵 기법을 제안한다. 본 논문의 순서는 다음과 같다. 기존의 프레임 스킵 기법을 Ⅱ장에서 소개하고, Ⅲ장에서는 프레임 스킵을 위한 제안된 움직임 벡터 합성 기법을 기술한다. Ⅳ장에서 제안된 방법의 실험 결과를 보이고 Ⅴ장에서 결론을 맺는다.

Ⅱ. 프레임율 변환

2.1 역방향 움직임 벡터 추적

입력된 프레임들이 프레임율 변환을 위해서 스킵될 때, 입력된 움직임 벡터는 사용불가능 하다. 왜냐하면 스킵된 프레임은 트랜스코딩된 프레임에 더 이상 존재하지 않기 때문이다. 그림 3에서 두 개의 프레임이 스킵되었을 때의 상황을 묘사하였다.

RB_l^{n-1}는 프레임 (n-1)에서의 MB_l^n의 참조블록으로 MB_l^n에 최적 정합 블록을 의미한다. RB_l^{n-2}는 프레임 (n-2)에서의 MB_l^n의 참조블록으로 MB_l^n에 최적 정합 블록을 의미한다. mv_l^n은 MB_l^n의 움직임 벡터를 의미하고, V_x^{n-1}은 RB_l^{n-1}의 움직임 벡터를 의미한다. 이 경우에 MB_l^n의 움직인 벡터는 식 (1)과 같이 벡터 V_x^{n-d}와 벡터 mv_x^n로 구해진다. 그러나 RB_l^{n-d}는 매크로블록이 아니기 때문에 V_x^{n-d}는 입력된 비트 스트림으로부터 구할 수 없다. 즉, MB_l^n의 움직임 벡터는 구할 수 없다.

$$CMV_x^n = \sum_{d=1}^{N} V_x^{n-d} + mv_x^n \tag{1}$$

단, N: 스킵된 프레임 개수
n: 현재 프레임 번호
x: 프레임에서의 매크로블록 번호

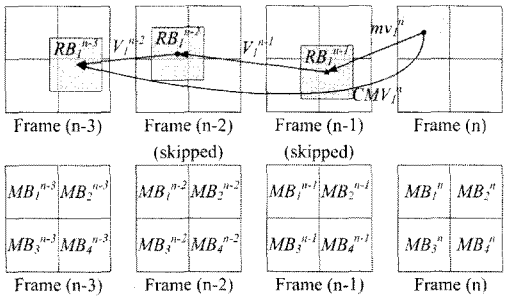

그림 3. 두 개의 프레임이 스킵시의 역방향 움직임 벡터 추적 방법
Fig. 3. Backward motion vector tracking, when two frames skipped

2.2 쌍선형 벡터 보간법

움직임 추정 없이 움직임 벡터를 구하는 한가지 방법은 압축된 비디오 비트 스트림으로부터 입력된 벡터를 사용하는 방법이다. 참조블록은 최대 4개의 매크로블록에 걸치게 된다. 그림 4에서 보여지듯이,

쌍선형 벡터 보간법 (Bilinear Vector Interpolation)은 참조 블록이 위치한 최대 4개 매크로블록의 움직임 벡터로부터 식 (2)와 같이 쌍선형 벡터 보간법을 사용하여 움직임 벡터를 구한다[3]. 그러나 몇 개의 움직임 벡터가 엉뚱한 방향으로 발산하면 쌍선형 벡터 보간으로 구해진 움직임 벡터도 엉뚱한 방향으로 발산 할 수 있다.

$$\begin{aligned} CMV = \{&(MB\_size-\alpha)(MB\_size-\beta)mv_1 \\ &+(\alpha)(MB\_size-\beta)mv_2 \\ &+(MB\_size-\alpha)(\beta)mv_3 \\ &+(\alpha)(\beta)mv_4\}/MB\_size \end{aligned} \quad (2)$$

단, MB_size: 매크로블록의 크기

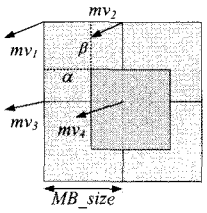

그림 4. 쌍선형 벡터 보간법
Fig. 4. Bilinear vector interpolation

2.3 지배적인 벡터 선택법

지배적인 벡터 선택법은 최대 4개의 인접한 매크로 블록으로부터 하나의 지배적인 움직임 벡터를 선택한다. 전방향 지배적인 벡터 선택법 (Forward Dominant Vector Selection, FDVS) 방법은 최대 4개의 인접한 매크로 블록중 하나의 지배적인 움직임 벡터를 선택한다. 지배적인 매크로블록이란 입력된 움직임 벡터가 가리킨 참조 블록이 가장 크게 겹쳐진 매크로블록으로 정의되고, 지배적인 움직임 벡터는 지배적인 매크로블록의 움직임 벡터로 정의된다[4]. 그림 5는 두 개의 프레임이 스킵 되었을 때의 FDVS 기법을 묘사하였다.

이 경우, MB_1^n의 지배적인 블록으로 MB4n-1를 선택한다. MB_1^n의 움직임 벡터는 mv_4^{n-1}가 된다. 최종적으로 합성된 움직임 벡터 CMV_1^n는 식 (3)와 같이 움직임 벡터의 합으로 구해진다.

$$CMV_1^n = mv_4^{n-2} + mv_4^{n-1} + mv_1^n \quad (3)$$

그러나 FDVS 기법은 참조 블록의 움직임 벡터를 구하는 것이 아니기 때문에 후보블록은 정확하지 않다. 예를 들어서, 지배거인 움직임 벡터 mv_4^{n-1}는 MB_4^{n-1}의 움직임 벡터이지 프레임 (n-1)에서의 MB_1^n의 움직임 벡터는 아니다. 즉, 지배적인 움직인 벡터의 보상이 필요하다.

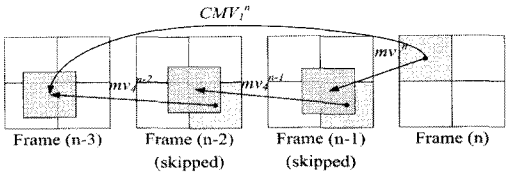

그림 5. FDVS 기법
Fig. 5. Illustration of the FDVS method.

Ⅲ. 제안된 움직임 벡터 합성 기법

본 논문에서, 개선된 전방향 지배적인 벡터 선택 (enhanced-forward dominant vector selection, E-FDVS) 기법은 수정된 지배적인 벡터 선택을 사용한다. E-FDVS 기법은 그림 6과 같이 진행되고 다음과 같이 정리하였다.

초기화: 프레임 (n)에서의 보상벡터 cv^n는 식 (4)과 같이 초기화 된다. 최종 움직임 벡터 lmv는 식 (5)와 같이 초기화 된다.

$$cv^n = (cv^n[0], cv^n[1]) = (0,0) \quad (4)$$

$$lmv = (lmv[0], lmv[1]) = (0,0) \quad (5)$$

1 단계: mv^k는 프레임 (k)에서의 지배적인 매크로블록의 움직임 벡터이다. lmv는 식 (6)와 같이 lmv에 mv^k를 더함으로써 갱신된다. k는 프레임 번호이고 x는 프레임에서의 매크로블록 번호이다.

$$lmv = lmv + mv_x^k \quad (6)$$

2 단계: 프레임 (k)에서의 지배적인 매크로블록의 전방향 벡터인 fvk는 식 (7)과 같이 구해진다. 전방향 벡터는 보상벡터로 보상된다. 보상 벡터가 E-FDVS 기법의 핵심되는 부분이다.

$$fv^k = cv^k + mv_x^k \quad (7)$$

3 단계: 프레임 (k-1)에서의 다음 지배적인 매크로블록은 fv^k에 의해서 선택된다.

4 단계: 보상벡터 cv^{k-1}는 프레임 (k-1)에서의 지배적인 매크로블록과 프레임 (k)에서의 지배적인 움직임 벡터의 차이로, 식 (8)과 식 (9)로 구해진다.

$$cv^{k-1}[i] = mv_x^k[i] \quad (8)$$

$$
\begin{aligned}
&while(|cv^{k-1}[i]|>8)\{\\
&\quad cv^{k-1}[i]=\begin{cases} cv^{k-1}[i]-16 & if\ cv^{k-1}[i]>8 \\ cv^{k-1}[i]+16 & if\ cv^{k-1}[i]<-8 \end{cases}\\
&\}
\end{aligned} \tag{9}
$$

where, k : *frame number*

i : *vector index*, $i = 0, 1$

x : *macroblock number in a frame*

5 단계: 스킵된 프레임을 모두 통과할 때까지 2 단계부터 4단계까지 반복한다.

6 단계: E-FDVS 기법은 종료되고, 합성된 움직임 벡터는 최종 움직임 벡터 lmv이다.

프레임 (n-1)이 스킵되었을 때, FDVS와 E-FDVS 모두 MB_1^n의 지배적인 움직임 벡터는 MB_4^{n-1}가 되고, 최종 움직임 벡터는 그림 6 (a)처럼 $mv_1^n+mv_4^{n-1}$가 된다. 프레임 (n-2)가 스킵되었을 때 그림 6 (b)에서 보여지는 것과 같이, FDVS는 MB_1^n의 지배적인 매크로블록으로 MB_4^{n-2}를 선택하는 반면, E-FDVS는 식 (7)과 같이 cv로 보상받기 때문에 MB_1^{n-2}를 선택한다. mv_4^{n-1}는 MB_4^{n-2},의 움직임 벡터이지, MB_1^n를 위한 참조 블록의 움직임 벡터는 아니다. 즉, FDVS와 E-FDVS의 차이는 E-FDVS가 보상벡터를 사용한다는 점이다. 그 결과 FDVS의 최종 움직임 벡터는 $mv_1^n+mv_4^{n-1}+mv_4^{n-2}$,가 되지만, E-FDVS의 최종 움직임 벡터는 그림 6 (c)와 같이 $mv_1^n+mv_4^{n-1}+mv_1^{n-2}$가 된다. 그림 7에서 E-FDVS 기법이 역방향 움직임 벡터 추적과 비슷함을 보인다.

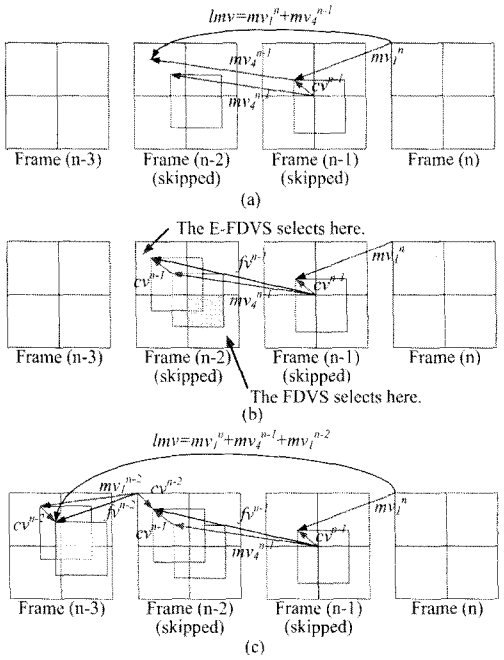

그림 6. E-FDVS 기법의 진행: (a) 2nd 1단계, (b) 2nd 2&3 단계, 그리고 (c) 2nd 4단계 & 3rd 1단계
Fig. 6. Progress of the E-FDVS method: (a) 2nd step 1, (b) 2nd step 2&3, and (c) 2nd step 4 & 3rd step 1

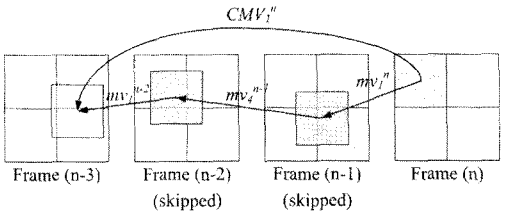

그림 7. E-FDVS 기법
Fig. 7. Illustration of the E-FDVS method.

움직임 벡터 합성 과정 중에 인트라 매크로블록이 스킵된 프레임에 존재할지도 모른다. 인트라 매크로블록은 움직임 벡터를 나르지 않기 때문에 우리는 이 매크로블록에서는 제로 움직임 벡터로 가정하고 움직임 벡터 합성 과정을 진행한다.

트랜스코딩된 비디오의 화질을 개선하기 위해서 우리는 움직임 벡터 합성과정을 통해서 구한 움직임 벡터를 정제한다. 대개 지배적인 움직임 벡터는 추정된 움직임 벡터이므로 합성된 움직임 벡터는 최적화 되진 않는다. 합성된 움직임 벡터는 성능을 향상시키기 위해서 정제과정을 거칠 필요가 있다. 본 논문에서는 우리는 몇가지 움직임 벡터 정제 기법을 사용한다. 전방향 크기 탐색 (Full Scale Search, FSS)는 일정한 탐색범위의 모든 탐색점을 탐색하는 방법이고, 수평 수직 탐색 (Horizontal And Vertical Search, HAVS) [4]은 고정된 탐색범위 안에서 가로방향으로 탐색을 시작한 후에 세로방향으로 탐색을 하는 방법이다. 가변 스텝 크기 탐색 (Variable Step-size Search, VSS) [9]는 합성된 움직임 벡터의 크기에 따라 가변적으로 탐색범위를 설정하고 탐색범위의 중간점부터 탐색을 시작하는 기법이다.

Ⅳ. 실험결과

4.1 실험 환경 구축

이 장에서 제안된 E-FDVS 기법의 실험 결과를 정리했다. 입력 비디오 시퀀스는 CIF (352x288)로 IPPP 구조 (인트라 길이=50)을 사용했고, 1Mbps 전송률 및 30fps 프레임율이 되도록 MPEG-2 Test

Model 5 (TM5) [10]로 부호화 했다. 그리고 이 시퀀스 들은 300kbps 전송률 및 10fps로 트랜스코딩 하였다. PSNR (Peak Signal to Noise Ratio)은 프레임당 평균값을 구했으며, SU (Speed Up)은 전역 탐색법에 대한 상대적인 빠르기를 나타낸다. 본 실험 결과들은 1Mbps 비트율 및 30fps 프레임율의 입력 시퀀스를 300kbps 비트율 및 10fps 프레임율로 트랜스코딩 하였다.

4.2 실험 성능 평가

먼저 움직임 합성 기법들의 성능을 비교하기 위해서 전역 탐색 움직임 추정 (full search motion estimation, FSME) 뿐만 아니라, Three Step Search (TSS), New Three Step Search (NTSS), Four Step Search (4SS), 그리고 Cross Diamond Search (CDS)과 같은 고속 탐색 알고리듬과도 비교하였다. 이때 블록 왜곡 측정은 SAD (Sum of Absolute Error)를 이용하였고 탐색 범위는 ±15이다. 표 1에서 제안된 E-FDVS 기법은 BVI 기법과 FDVS 기법과 비교하였다.

표 1. 움직임 벡터 합성 기법들 및 고속 탐색 알고리듬의 비교. [PSNR: dB, SU: Speed Up]
Table 1. Performance obtained using the motion vector composition methods and the fast motion estimation methods. [PSNR: dB, SU: Speed Up]

Test Sequence	Measure	FSME	Fast Motion Estimation				Motion Vector Composition		
			TSS	NTSS	4SS	CDS	BVI	FDVS	E-FDVS
Bus	PSNR	19.184	17.932	17.745	17.727	17.161	17.284	18.599	19.047
	SU	1	29.12	30.23	29.37	40.83	-	-	-
Coastguard	PSNR	25.317	24.938	24.493	24.977	24.11	21.475	24.916	24.916
	SU	1	29.12	33.66	33.07	48.19	-	-	-
Football	PSNR	19.736	19.19	19.072	18.973	18.495	16.790	18.620	18.779
	SU	1	29.12	35.09	31.52	44.47	-	-	-
Foreman	PSNR	28.873	27.693	27.637	28.012	28.048	21.112	28.166	28.282
	SU	1	29.12	35.77	30.78	43.99	-	-	-
Mother & Daughter	PSNR	37.481	37.101	37.132	37.269	37.236	23.143	36.351	36.352
	SU	1	29.12	49.75	36.79	86.29	-	-	-
Stefan	PSNR	21.162	20.256	20.238	20.339	20.019	18.463	20.898	21.045
	SU	1	29.12	36.85	32.27	50.82	-	-	-

표 2. 움직임 벡터 정제기법을 가진 FDVS와 E-FDVS의 비교 [PSNR: dB, SU: Speed Up]
Table 2. Performance obtained using motion the vector refinement method with the FDVS and the E-FDVS. [PSNR: dB, SU: Speed Up]

Test Sequence	Measure	FSME	FDVS				E-FDVS			
			None	FSS	HAVS	VSS	None	FSS	HAVS	VSS
Bus	PSNR	19.184	18.599	19.169	19.104	19.233	19.047	19.602	19.534	19.582
	SU	1	-	65.94	226.25	164.96	-	66.37	229.53	166.98
Coastguard	PSNR	25.317	24.917	25.183	25.156	25.049	24.916	25.183	25.156	25.049
	SU	1	-	41.66	164.39	114.63	-	41.66	164.39	114.63
Football	PSNR	19.736	18.620	18.961	18.872	19.131	18.779	19.119	19.026	19.281
	SU	1	-	50.77	189.43	135.38	-	50.62	188.86	135.43
Foreman	PSNR	28.873	28.166	28.673	28.537	28.566	28.282	28.787	28.658	28.619
	SU	1	-	42.78	167.28	117.64	-	42.79	167.42	117.75
Mother & Daughter	PSNR	37.481	36.351	37.435	37.312	37.061	36.352	37.435	37.312	37.061
	SU	1	-	41.64	163.19	114.11	-	41.64	163.19	114.11
Stefan	PSNR	21.162	20.898	21.493	21.426	21.371	21.045	21.63	21.564	21.499
	SU	1	-	51.03	192.10	134.33	-	51.13	192.30	134.81

V. 결 론

본 논문에서, 트랜스코더의 복잡도를 줄이고, 출력 비디오의 화질을 개선하기 위해서, 입력된 비디오 비트 스트림의 정보를 사용하는 프레임 스킵 기법을 제공하였다. 또한 프레임율의 변환시 움직임 벡터 합성 기법에 대해서 논의 하였다. 제안된 방법이 다른 방법보다 성능이 우수함을 실험 결과를 통해서 보였고, 특정 시퀀스의 경우에는 제안된 방법이 움직임 벡터 정제 기법의 조합이 전역 탐색 움직임 기법보다 성능이 우수함을 보인다.

참 고 문 헌

[1] A. Vetro, C. Christopoulos, H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, Vol.20, No.2, pp.18-29 March 2003.d

[2] I. Ahmad, X. Wei, Y. Sun, Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Trans. Multimedia*, Vol.7, No.5, pp.793-804, Oct. 2005.

[3] J.-N.Hwang, T.-D. Wu, C.-W. Lin, "Dynamic frame-skipping in video transcoding," *1998*

IEEE Second Workshop on Multimedia Signal Processing, pp.616-621, 7-9 Dec. 1998.

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

[5] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, "Motion compensated interframe coding for video conferencing," *in Proc. of National Telecommunication Conf. 1981*, pp.G5.3.1-5, New Orleans, LA, Dec. 1981.

[6] R.Li, B. Zeng, M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits and Syst. Video Technol.*, Vol.4, pp.438-442, Aug. 1994.

[7] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, Vol.6, Issue 3, pp.313-317, June 1996.

[8] C.-H. Cheung and L.-M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, Vol.12, pp.1168-1177, Dec. 2002.

[9] M.-J. Chen, M.-C. Chu, C.-W. Pan, "Efficient motion-estimation algorithm for reduced frame-rate video transcoder," *IEEE Trans. Circuits Syst. Video Technol.*, Vol.12, No.4, pp.269-275, April 2002.

[10] ISO/IEC, "Information technology - Generic coding of moving pictures and associated audio information: Video (MPEG-2 video)," ISO/IEC 13818-2, 2nd ed., 2000.

양 시 영 (Siyoung Yang) 정회원

2001년 2월한양대학교 전자 전기공학부 졸업 (공학사)
2003년 2월 한양대학교 전자통신전파공학과 졸업 (공학석사)
2003년 3월~현재 한양대학교 전자통신전파공학과 박사과정
<관심분야> 영상처리 및 영상압축, 의료영상

정 제 창 (Jechang Jeong) 정회원

1980년 2월 서울대학교 전자공학과 졸업 (공학사)
1982년 2월 KAIST 전기전자공학과 졸업 (공학석사)
1990년 3월 미국 Univ. of Michigan 전기공학과 졸업 (공학박사)
1982년 2월~1986년 7월 KBS 기술연구소 연구원 (디지털 TV 및 뉴미디어 연구)
1990년 8월~1991년 1월 미국 Univ. of Michigan 전기공학과 연구교수
1991년 1월~1995년 2월 삼성전자 멀티미디어 연구소 (MPEG, HDTV 및 멀티미디어 연구)
1995년 3월~현재 한양대학교 전자전기컴퓨터공학부 교수 (영상통신 및 신호처리 연구실)
<관심분야> 영상처리 및 영상압축