# MPEG 비디오 트랜스코딩에서 프레임율 변환을 위한 효율적인 움직임 벡터 재추정 기법

*최영태, *양시영, *정제창
*한양대학교 전자통신전파공학과
point53@ihanyang.ac.kr

# An Efficient Motion Vector Re-estimation Algorithm for Frame Rate Conversion in MPEG Video Transcoding

*Youngtai Choi, *Siyoung Yang, and *Jechang Jeong
*Dept. of Electrical and Computer Engineering, Hanyang University.

## 요약

다양한 처리능력을 가진 단말기들은 복잡한 네트워크 환경의 호환성을 제공하기 위해서, 전송 네트워크 채널이 허용하는 범위내로 부호화된 비디오의 비트율을 적응적으로 맞춰 주어야 한다. 트랜스코더는 특정 비트율로 부호화 되어 있는 비디오를 원하는 비트율로 다시 변환하기 위해서 복호화한 후 다시 부호화의 과정을 거쳐야 하기 때문에 이에 따른 계산량의 증가와 더불어 전송시간에 문제가 발생한다. 이를 해결하기 위한 한 가지 방법으로 제안된 것이 프레임 건너뜀 기법, 즉 시간적 해상도 변환 트랜스코딩이다. 비디오를 부호화하는 과정에서 계산량을 가장 많이 차지하는 움직임 추정과정의 계산량을 줄임으로써 트랜스코딩을 수행하는데 소모되는 시간과 노력을 크게 줄이고, 건너뛰지 않고 남아있는 프레임에 더 많은 비트를 할당하여 요구되는 화질을 유지할 수 있다. 본 논문에서는 움직임 벡터의 방향성을 고려한 움직임 벡터 재추정 기법과 정제 기법을 제안한다. 제안된 기법인 E-FDVS (Efficient-Forward Dominant Vector Selection)는 움직임 벡터의 방향성을 고려하여 매크로블록과 움직임 벡터의 차이를 이용하여 움직임 벡터를 재추정한다. 그리고 제안한 기법인 DOS (Direction Oriented Search)를 사용하여 재추정된 움직임 벡터의 방향으로 강한 움직임 벡터 정제를 함으로써 기존의 기법에 비해 우수한 화질을 제공한다.

## Ⅰ. 서론

방송의 디지털화는 고화질 방송을 제공하여 방송 서비스의 품질을 한 단계 높이는 한편으로, 기존의 아날로그 방송과는 달리 이동 중에도 화면이 흔들리지 않는 방송을 가능하게 하였다. 디지털방송 기술에 기반을 두어 새로이 출현한 DMB (Digital Multimedia Broadcasting) 서비스는 최대 7인치 화면에서 이동 중 언제 어디서나 CD급 고품질의 라디오, TV 동영상 및 문자방송 수신이 가능한 서비스이다. 또한 디지털 방송을 보여주는 디지털 TV는 디지털 미디어 센터의 의미를 가지게 됨에 따라 다양한 포맷의 컨텐츠를 해석하고 이를 디스플레이 하거나 다른 기기들로 전송해줄 필요가 생긴다. 이에 따라 다양한 포맷의 미디어를 복호화하고 포맷간의 상호변화와 화질을 최대한 유지하면서 비트스트림상에서 고속으로 변환하기 위한 트랜스코딩 기술이 필요하다. 트랜스코딩은 비트율 변환, 해상도 변환, 신텍스 변환, 그리고 에러내성 삽입 등 크게 4가지로 분류할 수 있으며, 일반적인 트랜스코더의 구조는 그림 1과 같다 [1].

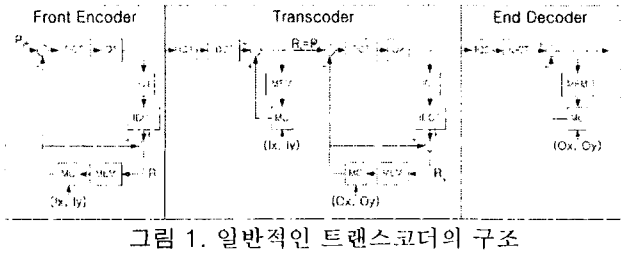


그림 1. 일반적인 트랜스코더의 구조

특정 비트율로 부호화 되어 있는 비디오를 원하는 비트율로 다시 변환하기 위해서는 복호화한 후 다시 부호화의 과정을 거쳐야 하기 때문에 이에 따른 계산량의 증가와 더불어 전송시간에 문제가 발생하게 된다. 이를 해결하기 위한 방법으로 프레임율을 변환시키는 프레임 건너뜀(frame skipping) 기법, 즉, 시간적 트랜스코딩이 있다. 이는 비디오를 부호화하는 과정에서 계산량을 가장 많이 차지하는 움직임 추정과정의 계산량을 줄임으로써 트랜스코딩을 수행하는 시간과 노력을 크게 줄이고, 건너뛰지 않고 남아있는 프레임에 더 많은 비트를 할당하여 요구되는 화질을 유지할 수 있다 [2].

움직임 벡터를 재추정(re-estimation)하는 과정에서 계산량을 줄이기 위해 입력 비트스트림으로부터 추출해 낸 기서 움직임 벡터를 재사용한다. 대표적인 기법으로 선형 보간법 (Bilinear Interpolation), FDVS (Forward

Dominant Vector Selection), 그리고 ADVS (Active Dominant Vector Selection) 기법 등이 있다 [3, 4, 5]. 그러나 이 기법들은 움직임 벡터의 오추정 또는 전이가 발생하게 된다. 이를 피하기 위해 본 논문에서는 움직임 벡터의 방향성을 고려한 움직임 벡터 재추정 기법을 제안한다.

또한 움직임 재추정 기법을 통해 얻은 움직임 벡터는 최적의 움직임 벡터가 아니므로, 보다 더 양호한 화질을 얻으면서 비트율을 줄이기 위해 움직임 벡터 정제(refinement) 과정을 수행하게 된다. 본 논문에서는 움직임 재추정 기법을 통해 얻어진 기저 움직임 벡터의 방향을 고려한 움직임 벡터 정제기법도 제안한다.

본 논문의 Ⅱ장에서는 기존의 트랜스코딩 기법들에 대해 기술하며, Ⅲ장에서는 움직임 재추정 기법과 움직임 정제 기법을 제안한다. Ⅳ장에서는 실험결과를 제시하여 기존의 기법들과의 차이를 비교하고, Ⅴ장에서 결론을 맺는다.

## Ⅱ. 시간적 트랜스코딩

### 1. 프레임율 변환

기존의 현존하는 비디오 압축 표준들은 대부분 매크로블록 단위의 BMA (block matching motion estimation algorithm)을 기반으로 움직임 추정을 수행한다. 그림 2는 프레임 건너뜀 기법을 나타낸다.

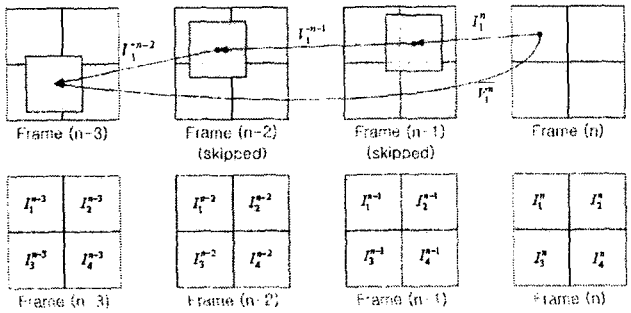



그림2. 프레임율 변환시의 움직임 벡터 추정 방법

그림 2에서 $I_l^n$은 n번째 프레임의 매크로블록의 실제 움직임 벡터를 말하고, $V_l^n$은 가상의 움직임 벡터를 말한다. 움직임 추정을 거치지 않고 건너뛴 프레임의 움직임 벡터를 찾는 방법은 벡터 I와 V를 합하는 방법을 생각할 수 있으나, 실제로 벡터 V는 유효하지 않으므로 더할 수 없다. 이를 해결하기 위해 제안된 선형 보간법은 움직임 벡터가 가리키는 매크로블록과 겹치는 참조 프레임의 이웃 매크로블록의 움직임 벡터에 가중치를 두어 계산하는 방법이다. 그러나 선형 보간법은 건너뛴 프레임의 매크로블록만큼의 움직임 벡터를 저장하기 위한 많은 메모리가 소요되고 계산이 복잡해지며, 하나의 움직임 벡터가 상당히 발산하면 잘못된 움직임 벡터를 찾는다는 단점이 있다 [4].

### 2. FDVS 기법

트랜스코더는 복호화부터 부호화 순으로 수행되며, 복호기는 전탐색 움직임 추정을 거친 움직임 벡터가 비트스트림으로부터 입력되며, 부호기는 입력 비트스트림을 원하는 비트율의 출력 비트스트림으로 재부호화한다. 트랜스코더에서의 전역탐색 움직임 추정은 높은 계산 복잡도를 갖는다. 계산 복잡도를 감소시키기 위해서 움직임 벡터를 재사용하는 움직임 추정 방법을 사용한다.

트랜스코딩에서 입력 프레임들의 프레임율이 감소될 때 출력 비트스트림들은 새로운 움직임 벡터로 구성된다. 대표적인 기법인 선형 보간법, FDVS 기법, 그리고 ADVS 기법은 네 개의 인접한 움직임 벡터들로부터 한 개의 움직임 벡터를 구성한다. 그림 3은 두 프레임이 건너뛰어졌을 때의 FDVS 기법을 나타낸 것이다. FDVS는 그림 3에서 보는 것과 같이, 각각의 건너뛴 프레임에서의 가상의 벡터 V들에 가장 지배적인(가장 많이 겹쳐진) 움직임 벡터를 찾아 해당 프레임 매크로블록의 움직임 벡터로 결정한다. 즉, FDVS 기법을 통해 얻어진 실제 벡터 I들을 합함으로써 (n) 프레임의 (n-3) 프레임으로의 움직임 벡터를 구할 수 있다. FDVS 기법은 많은 계산량과 메모리를 요구하는 선형 보간법보다 우수한 결과를 얻을 수 있다 [4, 5].

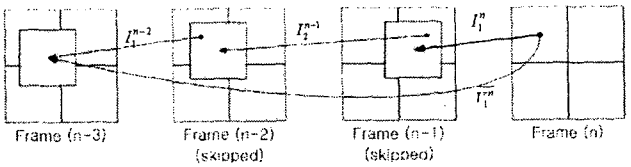



그림 3. 두 프레임을 건너뛰었을 때의 FDVS 기법

## Ⅲ. 제안하는 알고리즘

### 1. 제안하는 움직임 벡터 재추정 기법

FDVS 기법은 인접하는 매크로블록으로부터 가장 많이 겹쳐지는 지배적인 매크로블록을 선택함으로써 움직임 벡터를 재추정한다. 그러나 FDVS 기법은 프레임 건너뜀이 많아질수록 움직임 벡터의 오추정이 전파된다. 오추정이 누적되면, 재추정된 움직임 벡터는 실제의 원영상과 전혀 다른 매크로블록을 현재 매크로블록의 움직임 벡터로 오인될 수 있다. 그리고 BMA 방법의 특성상 블록 경계에서의 움직임 추정치가 많은 차이를 나타낸다.

이를 보완하기 위해서 본 논문에서는 매크로블록의 위치정보와 움직임 벡터를 이용하여 새로운 움직임 벡터를 재추정하는 E-FDVS (Efficient-FDVS) 기법을 제안한다. 만약 현재의 프레임 (n)에서 움직임 벡터 (mx, my)가 매크로블록 경계에 위치해 있다면, 더 이상의 움직임 재추정 과정이 필요 없게 된다. 그러나 대부분의 움직임 벡터는 이전 참조 프레임의 주변 2~4개의 매크로블록에 겹쳐지게 된다. 주변 매크로블록 중에서 가장 지배적인 매크로블록의 현재 프레임의 움직임 벡터로 결정하게 되면, 새로운 움직임 벡터와 원래의 움직임 벡터간의 오차가 발생하게 되고, 프레임 건너뜀 수가 많아질수록 움직임 벡터가 오추정될 확률이 높아진다. 따라서 본 논문에서는 원래의 움직임 벡터와 새로 재추정한 움직임 벡터의 오차를 이용하여 움직임 벡터를 재추정함으로써 계산상의 복잡도는 기존의 움직임 재추정기법과 같으면서, 그 성능은 향상된다.

그림 4는 제안하는 움직임 벡터 재추정 기법을 나타낸다. 식 (1)에서 기저 움직임 벡터 (mx, my) 의 방향과 크기에 따라 움직임 벡터 오차 (rx, ry)를 구하고, 식 (2)에서 (rx, ry)와 이전 프레임의 (mx, my)를 합하여 구해진 (vx, vy)의 방향과 크기에 따라 지배적인 움직임 벡터를 재추정한다.

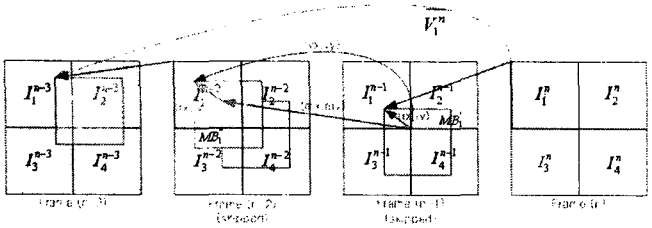


그림 4. 제안하는 움직임 벡터 재추정 과정 (E-FDVS 기법)

$$(rx, ry) = \begin{cases} mv+16, & if\ mv \le -8 \\ -mv, & if\ -8 < mv < 0 \\ mv, & if\ 0 \le mv < 8 \\ mv-16, & if\ mv \ge 8 \\ 0, & otherwise \end{cases} \quad \cdots\cdots (1)$$

$$where,\ search\ range \le 16$$

$$(vx, vy) = (rx, ry) + (mx, my) \quad \cdots\cdots\cdots (2)$$

### 2. 제안하는 움직임 벡터 정제 기법

움직임 재추정 과정을 거쳐 얻어진 움직임 벡터들은 원래의 움직임 벡터를 근사화한 값이어서 최적의 움직임 벡터라 할 수 없으므로 더 나은 성능을 얻기 위해서 움직임 벡터 정제 과정이 필요하다. 움직임 벡터 정제를 위해 전탐색 기법을 사용한다면, 재추정된 움직임 벡터 주변의 작은 탐색영역을 모두 탐색하게 되므로 계산량이 많이 소요된다. 따라서 본 논문에서는 탐색 포인트를 줄이면서 원하는 성능을 얻기 위해서 방향성을 고려한 DOS (Direction Oriented Search) 기법을 제안한다. DOS 기법은 기존의 VSS (Variable Step-size Search) 기법 [3]과 마찬가지로 재추정된 움직임 벡터의 크기가 크면 스텝 크기가 커지게 된다. 탐색 포인트의 수는 VSS 기법의 최대 9개보다 2개적은 최대 7개로 감소된다. 전체적인 과정은 다음과 같다.

단계 1) 식 (3)을 이용하여 스텝 크기 결정

$$|S_x| = \frac{|MV_x|}{2} + 1, \quad |S_y| = \frac{|MV_y|}{2} + 1 \quad \cdots\cdots\cdots (3)$$

단계 2) 원점을 포함한 기저 움직임 벡터 방향 3개의 탐색 포인트 검색 (스텝 크기 적용)

단계 3) SAD (Sum of Absolute Difference)가 가장 작은 점의 가로방향 2개의 탐색 포인트 검색

단계 4) SAD가 가장 작은 점의 세로방향 2개의 탐색 포인트 검색

단계 5) SAD가 가장 작은 점을 움직임 벡터로 선택

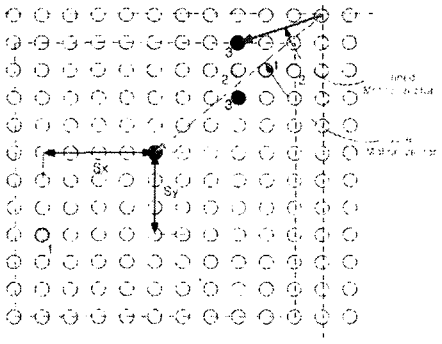


그림 5. 제안하는 움직임 벡터 정제 과정 (DOS 기법)

## IV. 실험결과

제안한 알고리즘을 검증하기 위하여 하드웨어 환경은 Pentium 4 CPU 1.50GHz, 384MB RAM, Microsoft Windows XP에서 구현 하였고, 실험은 MPEG-2 TM5 환경에서 프레임율 변환에 근거하여 FDVS 기법과 함께 제안한 기법을 비교하여 화질 개선의 평가 기준을 두었다. 참조 소프트웨어의 파라미터는 다음과 같이 설정하였다.

- Input Sequence : CIF (352*288, YUV 4:2:0)
- 탐색 영역 : 15 (Integer Pixel)
- 양자화 파라미터 (QP) : 1
- 참조 입력 프레임 : 100 프레임 (I-프레임, P-프레임)
- 참조 출력 프레임 : 34 프레임 (3 프레임당 1프레임)

시퀀스는 foreman, bus, football, stefan 을 이용해서 제안된 알고리듬을 실험하였다. 프레임 구조는 첫 번째 프레임만 I 프레임으로 하고 그 이후 프레임은 P 프레임으로 부호화했다. 실험 결과는 표 1과 표 2. 그리고 그림 6~그림9에 나타내었다.

표 1. 프레임율 변환시 움직임 재추정 결과 비교 (단위 : dB)

| method | foreman | bus | football | stefan |
|---|---|---|---|---|
| FS | 30.86 | 21.20 | 20.24 | 21.69 |
| Bilinear | 26.30 | 19.08 | 18.23 | 20.38 |
| FDVS | 28.00 | 19.82 | 19.31 | 20.96 |
| E-FDVS | **28.17** | **19.91** | **19.38** | **21.07** |

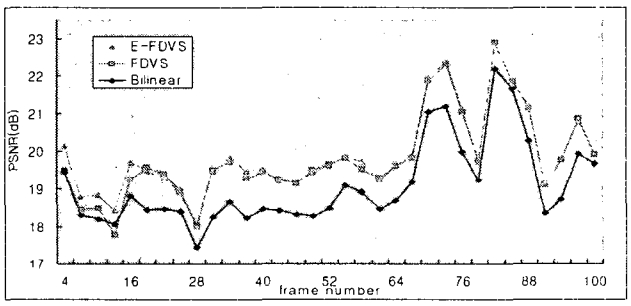



그림 6. bus 시퀀스의 움직임 벡터 재추정 기법 비교

(a) original (b) bilinear (19.4871 dB)

(c) FDVS (19.4874 dB) (d) E-FDVS (20.1241 dB)

그림 7. bus 시퀀스의 4번째 프레임의 움직임 벡터 재추정 기법 비교

표 2. 움직임 벡터 정제 기법의 PSNR, 탐색 포인트 비교
단위 : PSNR(dB), Search Point(search point/frame)

| method | | | foreman | bus | football | stefan |
|---|---|---|---|---|---|---|
| FDVS | FSS | PSNR | 29.64 | 22.62 | 21.20 | 23.02 |
| | | (SP) | (97061) | (306162) | (161382) | (268938) |
| | VSS | PSNR | 28.64 | 21.10 | 20.16 | 21.80 |
| | | (SP) | (2441) | (3176) | (2150) | (2451) |
| | DOS | PSNR | 29.05 | 21.79 | 20.32 | 22.50 |
| | | (SP) | (2480) | (2568) | (2143) | (2100) |
| E-FDVS | FSS | PSNR | 29.64 | 22.66 | 21.21 | 23.07 |
| | | (SP) | (96760) | (309890) | (161087) | (272591) |
| | VSS | PSNR | 28.70 | 21.16 | 20.20 | 21.88 |
| | | (SP) | (2443) | (3183) | (2150) | (2445) |
| | DOS | PSNR | **29.10** | **21.87** | **20.35** | **22.61** |
| | | (SP) | **(2481)** | **(2568)** | **(2141)** | **(2094)** |

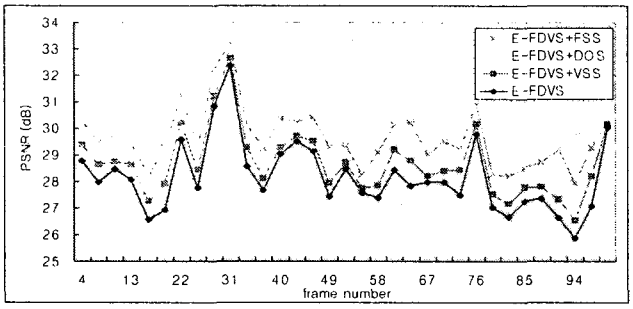
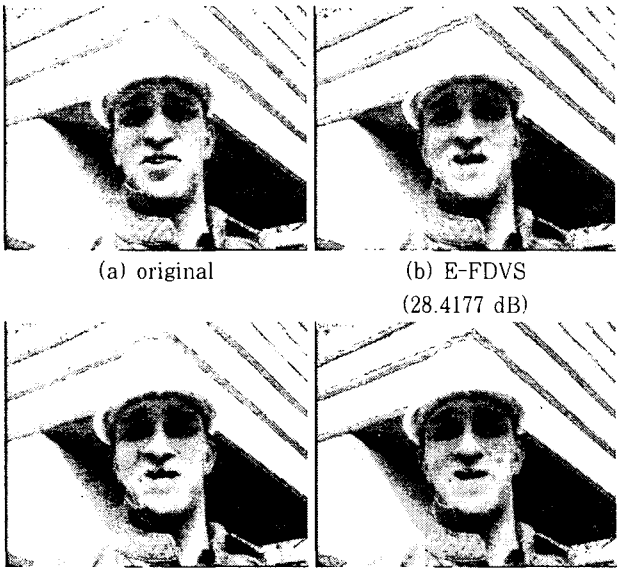


그림 8. foreman 시퀀스의 움직임 벡터의 정제기법 비교


(a) original (b) E-FDVS (28.4177 dB)

(c) FDVS + VSS (29.1867 dB) (d) E-FDVS + DOS (29.5861 dB)

그림 9. foreman 시퀀스의 61번째 프레임의 움직임 벡터 정제 기법 비교

표 1은 움직임 벡터 정제 과정이 생략된 상태에서의 성능을 비교한 것이고, 그림 6은 bus 시퀀스의 움직임 벡터 재추정 과정을 비교하였고, 그림 7은 bus 시퀀스의 4번째 프레임의 움직임 벡터 재추정 결과 영상을 비교한 것이다. 표 1을 살펴보면 제안하는 E-FDVS 기법이 FDVS와 같은 계산량으로 0.07~0.18dB 정도 향상된 성능을 나타냈다.

표 2와 그림 8은 움직임 벡터의 정제과정에 대한 실험 결과이다. 표 2에서는 FDVS와 E-FDVS에 대한 FSS (Full Scale Search), VSS, 그리고 DOS의 조합의 결과를 나타냈고, PSNR의 단위는 dB이며, SP (Search Point)의 단위는 한 프레임당 평균 탐색 포인트 개수이다. FSS는 식 (3)의 DOS와 같은 탐색 범위를 갖으면서 탐색 범위 안의 모든 포인트에서 검색한다. 그림 9는 foreman 시퀀스에 대해서 움직임 정제 기법을 비교한 것이고, 그림 foreman 시퀀스의 61번 째 프레임의 움직임 벡터 정제 결과 영상을 비교한 것이다. 재추정된 움직임 벡터에 대해 방향성을 고려하여 움직임 벡터를 정제함으로써 VSS 기법에 비해 foreman 시퀀스가 평균 약 40포인트 정도 증가하는 것을 제외하고, 프레임당 약 10~600포인트 감소했다. FDVS 기법과 E-FDVS 기법 모두가 탐색 포인트가 감소했다는 것은 그만큼 속도가 향상된다는 것을 뜻한다. PSNR은 약 0.15~0.73dB 정도 더 나은 성능을 보였다. 제안하는 움직임 벡터 정제 기법을 사용하면 VSS 기법에 비해 약 25%의 계산량을 감소시킴으로써 성능이 향상되었다.

## V. 결론

유무선 네트워크 환경에서 제공되는 대부분의 비디오 서비스와 멀티미디어 응용서비스를 사용자가 이용할 때 다양한 채널의 대역폭에 따라 적절한 비트율로 변환해 주어야만 정해진 채널상황 속에서 최대한의 비디오 품질을 보장할 수 있다. 본 논문에서는 움직임 벡터의 방향성을 이용한 움직임 재추정 기법과, 방향성을 고려함으로써 보다 더 정확한 움직임 벡터를 찾는 정제 기법을 제안하였다. 움직임 벡터 추정시 움직임 벡터의 차이 값과 방향성을 이용함으로써 움직임 벡터의 오추정율을 감소시키고, 움직임 벡터 정제 기법에서는 방향성을 강조하여 기존의 기법보다 불필요한 탐색 포인트에 대한 계산량을 줄이면서 보다 우수한 성능을 보였다.

## 참고문헌


[1] A. Vetro, C. Christopoulos, and Sun Huifang, "Video transcoding architectures and techniques: an overview," IEEE Signal Processing Magazine, Vol.20, Issue 2, pp.18-29, March 2003.

[2] Jenq-Neng Hwang, Tzong-Der Wu, and Chia-Wen Lin, "Dynamic frame-skipping in video transcoding," 1998 IEEE Second Workshop on Multimedia Signal Processing, pp.616-621, 7-9 Dec. 1998.

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

[4] Jeongnam Youn, Ming-Ting Sun, and Chia-Wen Lin, "Motion vector refinement for high-performance transcoding," IEEE Transactions on Multimedia, Vol.1, Issue 1, pp.30-40, March 1999.

[5] Jeongnam Youn and Ming-Ting Sun, "fast motion vector composition method for temporal transcoding," Proceedings of the 1999 IEEE International Symposium on Circuits and Systems 1999. Vol.4, pp.243-246, 30 May-2 June 1999.