프레임률 조절 트랜스코더의 개선된 움직임 벡터 합성 기법

(An Enhanced Motion Vector Composition Scheme of the Frame-Rate Control Transcoder)

이 승 원 † 박 성 호 †† 정 기 동 †††
(Seung Won Lee) (Seong Ho Park) (Ki Dong Chung)

요 약 네트워크 환경에 적응적인 비디오 스트리밍 서비스를 위하여 비디오의 트랜스코딩이 하나의 해결책으로 제시되었다. 시간당 프레임의 개수를 조절하는 기법은 이러한 비디오 트랜스코딩 기법들 중 하나이다. 이 기법에서는 삭제된 프레임을 참조하는 프레임의 움직임 벡터를 재생성 해야 되는데 이것은 비디오 트랜스코딩의 계산 복잡도를 높이는 주요 원인이다. 본 논문에서는 움직임 벡터의 재생성 시 요구되는 계산 복잡도를 줄이기 위해 삭제된 프레임의 움직임 벡터를 재사용하는 영역과 활동 상태 정보를 고려한 벡터 합성 기법을 제안한다. 이 기법은 각 매크로블록들의 활동 상태정보량과 중첩영역의 크기를 기반으로 축출한 가중치를 이용하여 각 움직임 벡터들을 합성하는 기법이다. 실험 결과 RABVC는 기존의 가중치 기반의 움직임 벡터 선택 기법들에 비해 비슷한 계산 복잡도에서 높은 PSNR값을 보였다.

키워드 : 움직임 벡터 재구성, 트랜스코딩, 계산 복잡도, PSNR

Abstract To provide adaptively video streaming services on network environment, video transcoding is introduced. The one of transcoding methods is the frame-rate conversion. It needs a re-estimation about a motion vector of the frame to refer a skipping frame. This re-estimation makes higher the computational complexity in video transcoding. To reduce the computational complexity of a motion vector refinement, this paper proposes a region & activity based motion vector composition scheme that refine the moving vector of a skipping frame. This scheme composes each motion vector from the weight based on the activity information of a macroblock and the size of the overlapped area. The experiment result shows that RABVC has a higher PSNR than the value of existing weight-based motion vector selection schemes though the computational complexity of our scheme is similar to that of other schemes.

Key words : Motion Vector refinement, Transcoding, Computational Complexity, PSNR

1. 서 론

최근 화상 전화, 원격 교육 시스템, VOD 등과 같이 네트워크로 지원되는 멀티미디어 서비스들이 각광을 받고 있다. 이러한 응용 서비스에서는 다양한 채널의 유효 대역폭에 적응성 있게 비디오의 비트율을 조정할 필요가 있다. 채널 대역폭 적응력은 다양한 이질적인 네트워크 환경에서 차등적인 품질의 서비스를 서로 다른 여러 사용자들에게 제공할 수 있게 해준다. 이를 위하여 일부 실시간 응용에서는 비디오 인코더의 흐름 제어 방법을 이용하여, 네트워크 대역폭 적응력을 지원한다. 그러나 VOD와 같이 이미 특정 비트율로 인코딩 되어 저장되어 있는 응용 서비스의 경우 비디오 인코더의 실시간 흐름 제어의 방법을 사용할 수 없다.

특정 비트율로 인코딩 되어 저장된 비디오 스트림을 트랜스코딩을 이용하여 낮은 비트율로 변환하는 기법은 다양한 네트워크 환경에 적합한 동적인 비트율 적응 기법을 제공한다[1-5]. 비디오 트랜스코딩을 이용하여 비트율을 변환하는 방법에는 크게 비디오 스트림의 포맷을 변경하지 않고 비트율을 조정하는 방식과 비디오 스트림의 포맷을 변경하여 비트율을 조정하는 방식이 있다[6]. 이 중에 스트림의 포맷을 변경하여 비트율을 변

† 정 회 원 : 부산대학교 전자계산학과
swlee@pusan.ac.kr
†† 정 회 원 : 부산대학교 정보전산원 교수
shpark@pusan.ac.kr
††† 종신회원 : 부산대학교 전자계산학과 교수
kdchung@pusan.ac.kr
논문접수 : 2004년 2월 3일
심사완료 : 2004년 10월 8일

경하는 방식은 본 논문의 범위를 벗어나므로 여기서는 언급하지 않겠다. 스트림의 포맷을 변경하지 않고 네트워크 환경 및 사용자의 장치 능력에 적당하도록 비트율을 낮추는 방법에는 양자화 스텝 사이즈를 조절하는 방법과 시간 당 요구 프레임의 개수를 낮추는 방법이 있으며, 추가적으로 화면 사이즈를 줄이는 방법이 있다.

양자화 스텝 사이즈를 조절하는 방법은 트랜스코더에 있는 인코더에서 재 양자화 할 때 양자화 스텝 사이즈를 늘려주는 방식으로서, 일반적으로 비트율을 감소시키는데 있어서 간단하면서도 많이 사용되고 있는 기술이다. 하지만 재 양자화 할 때 양자화 에러가 발생함으로 인해 화질 열화가 생긴다. 최근에는 이것을 보완해 주는 기술에 관련된 연구가 진행되어 왔다[7-10]. 시간 당 요구 프레임의 개수를 감소시키는 방법은 트랜스코더에서 일정 개수의 프레임들을 반복적으로 제거함으로써 비트율을 감소시키는 방법이다[11-16]. 이는 각 프레임들의 화질을 유지한 상태에서 사용자의 대역폭 환경에 적응적으로 서비스 할 경우에 사용하는 기법이다. 화면 사이즈를 줄이는 방법은 사용자의 장치적인 제약으로 인해 주로 사용되는 방식으로 네트워크 상황을 고려해서 화면비율을 조절한다[5,17].

본 논문에서는 비디오 타입을 변경하지 않고 비트율을 조정하는 방법 중에 시간 당 프레임의 개수를 조정하는 방법을 고려하였다. 이 방법은 앞서 언급했듯이 요구 비트율을 맞추기 위해 일정 간격마다 프레임을 제거하여 전송하는 방식이다. H.263[18]의 "IPPP" 프레임 타입과 같이 모든 프레임이 참조 프레임의 역할을 하는 비디오 타입에서는 프레임 제거는 제거된 프레임의 전후 프레임들 간의 새로운 움직임 벡터의 축출을 요구한다. 최적화 된 움직임 벡터를 full-scale 움직임 예측을 통해서 축출할 수 있지만 이것은 높은 계산 복잡도로 인해 적당하지 않다. 이러한 계산 복잡도에 대한 해결책으로 움직임 벡터의 재사용 기법들이 연구되고 있다[11-16].

벡터 재사용 기법은 트랜스코더에 입력된 비트 스트림으로부터 축출한 움직임 벡터를 재사용하게 된다. 이 방식은 삭제된 프레임의 움직임 벡터와 이 프레임을 참조하는 프레임의 움직임 벡터간의 단순한 재구성을 통해서 새로운 움직임 벡터를 축출하게 된다[11]. 그러나 이러한 단순한 움직임 벡터 재사용 기법은 여러 응용에서 현저한 화질 저하를 초래할 수 있다[2,11]. 따라서 입력된 스트림으로부터 축출한 움직임 벡터들에서 새로운 벡터를 재구성하는 여러 기법들이 연구 대상으로 관심을 받고 있다[15,16].

본 논문에서는 시간당 프레임의 개수를 조정하여 비트율을 조정할 때 삭제된 프레임을 참조하는 프레임의 움직임 벡터들을 재구성하는 새로운 기법인 영역과 활동 상태 정보를 고려한 벡터 합성 기법(Region & Activity Based Vector Composition)을 제안하고, 기존 기법들과 비교하여 제안된 RABVC 기법의 성능을 비교 분석하였다. 기존 연구들은 주로 제거된 프레임에서 가중치를 고려하여 가장 유력한 움직임 벡터를 선택하는 기법이지만, 본 논문에서는 제거된 프레임의 여러 벡터들 간의 가중치를 고려하여 합성된 움직임 벡터를 적용하는 방법을 제안하였다. RABVC 기법과 기존 기법간의 성능평가를 위해 *H.263 Encoder/Decoder*[22]로 구현된 트랜스코더에서 움직임 벡터를 재사용 할 수 있게 각각의 기법들을 적용하였고, 성능 평가의 항목으로 출력된 스트림의 PSNR과 트랜스코딩 과정의 계산 복잡도를 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구들을 소개한다. 3 장에서는 본 논문에서 제안한 영역과 활동 상태 정보를 고려한 벡터 합성 기법(Region & Activity Based Vector Composition)을 소개하고, 4장에서 실험을 통하여 RABVC 기법과 기존 기법의 성능을 비교 분석하였다. 마지막으로 5장에서 결론 및 향후 과제를 제시한다.

2. 관련연구

움직임 벡터를 축출하는 과정은 시간 당 프레임의 개수를 조정하여 비트율을 조정하는 트랜스코더에서 계산 복잡도를 높이는 요인이다. 이러한 트랜스코더의 계산 복잡도를 줄이는 방법으로 개선된 움직임 예측 기법 외에도 입력 스트림의 움직임 벡터를 재사용하는 기법도 널리 채택되어 왔다. 반복적으로 프레임을 삭제하는 트랜스코딩 기법에서는, 삭제된 프레임을 참조하는 프레임을 위하여 새로운 움직임 벡터가 계산되어져야 한다. 그림 1은 한 개의 프레임이 삭제되었을 경우 각 프레임에서의 움직임 벡터들 간의 관계를 보여준다. 그림 1에서 MV′는 MV1과 MV2의 벡터 합으로 구해진다. 그러나 MV1의 경우 매크로블록의 움직임 벡터로부터 쉽게 축출할 수 있지만 MV2의 경우 매크로블록의 움직임 벡터가 아니기 때문에 축출하기가 쉽지 않다. MV2를 결정하는 가장 간단한 방법은 중첩영역이 존재하는 매크로블록 중 하나를 선택하여 그 블록의 움직임 벡터를 선택하는 것이다. 그러나 일반적으로 이런 방법으로는 최적의 MV2를 구할 수 없기 때문에 현저한 화질의 저하를 초래한다. 따라서 입력 스트림의 움직임 벡터에서 MV2를 축출하는 기법에 대한 연구가 필요하다. 본 장에서는 입력 스트림의 움직임 벡터들로부터 MV2를 구하여 MV′를 축출하는 기법 중에 대표적인 기법들인 Bilinear 기법, FDVS(Forward Dominant Vector Selec-

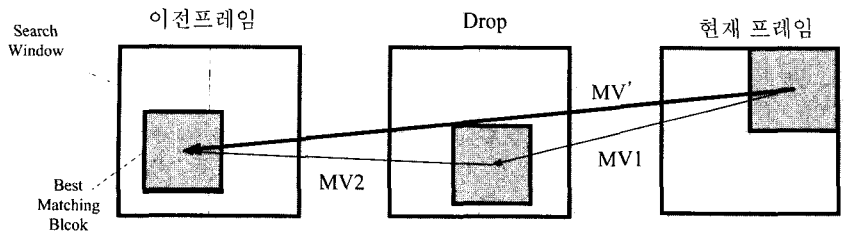


그림 1 움직임 벡터의 재구성

tion) 기법, ADVS(Activity Dominant Vector Selection) 기법을 소개한다[15-17].

2.1 Bilinear interpolation 기법

Bilinear interpolation은 제거되는 프레임의 모든 움직임 벡터를 저장하고 마지막에 제거되는 프레임에서 첫 번째로의 역방향 순서(backward order)로 움직임 벡터를 보간하는 방법이다[17]. 이 기법은 AMVR(Adaptive Motion Vector Resampling)[17] 기법에서 제시된 다음 식을 사용하여 각 매크로블록의 공간적 활동량(0이 아닌 AC 계수의 개수)을 합성하여 새로운 움직임 벡터를 도출한다. 아래의 식에서 mv_i와 A_i는 각각 i번째 블록의 움직임 벡터와 활동량 측정치를 나타낸다.

$$mv' = \frac{1}{2} \times \frac{\sum_{i=1}^{4} mv_i \times A_i}{\sum_{i=1}^{4} A_i}$$

그러나 제거되는 프레임마다 모든 움직임 벡터를 저장해야하는 메모리 문제, 합성된 움직임 벡터가 중첩되는 네 개의 매크로블록들이 차지하는 영역의 크기를 고려하지 못한 단점이 있다.

2.2 FDVS(Forward Dominant Vector Selection) 기법

FDVS(Forward Dominant Vector Selection) 기법은 그림 1의 MV2를 구할 때 가장 많은 영역을 차지하고 있는 매크로블록의 움직임 벡터를 선택하는 기법이다[16].

그림 2와 같은 입력 스트림 구조에서는 단순히 프레임(n-1) 하나만 제거 되었을 경우 MB_1의 움직임 벡터는 프레임(n)에서 MB_1의 움직임 벡터와 프레임(n-1)에서 $MB_{1'}$의 움직임 벡터를 단순히 더하여 재사용할 수 있다. 이러한 기법은 full-scale 움직임 예측을 하지 않음으로 상당한 계산 처리량을 줄일 수 있다. 하지만 그림 2에서 보듯이 프레임(n-1)에 있는 $MB_{1'}$이 네 개의 인접 매크로블록 영역을 일부분씩 차지하고 있을 때 이 중 어떤 매크로블록의 움직임 벡터를 사용할 것인가를 선택하는 것이 중요하게 된다.

FDVS 기법에서는 여러 매크로블록에 걸쳐 있는 블록에서 유력한 움직임 벡터를 선택하는 경우 공간적으로 가장 많은 영역을 차지하고 있는 매크로블록의 움직임 벡터를 선택한다. 예를 들어 그림 2에서 프레임(n-1)이 제거되었을 경우 $MB_{1'}$이 차지하는 공간에서 오른쪽 상단에 위치한 매크로블록의 움직임 벡터가 선택될 것이다. 이것은 프레임이 두 개 이상 제거될 경우에도 적용할 수 있는 기법으로서, 프레임(n-1)과 프레임(n-2)이 제거되었을 경우에 MB_1이 지닐 수 있는 움직임 벡터는 $I_1^{(n)} + I_2^{(n-1)} + I_3^{(n-2)}$으로 구성할 수 있다. FDVS기법은 움직임 벡터를 재사용하여 계산량을 줄일 수 있는 방법으로 한 개 또는 연속적인 여러 프레임들이 제거되었을 경우에 효율적으로 사용할 수 있는 기법이다.

2.3 ADVS(Activity Dominant Vector Selection) 기법

FDVS는 네 개의 인접한 매크로블록에서 가장 큰 중첩영역을 가지는 매크로블록의 움직임 벡터를 선택하게

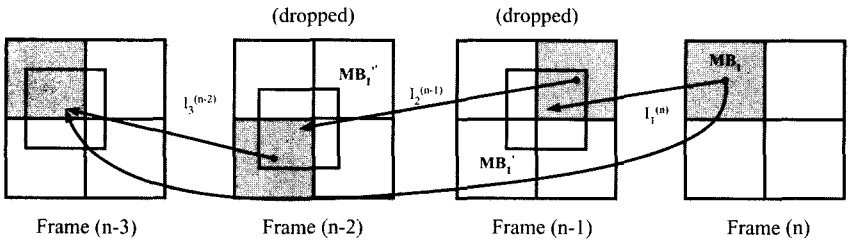
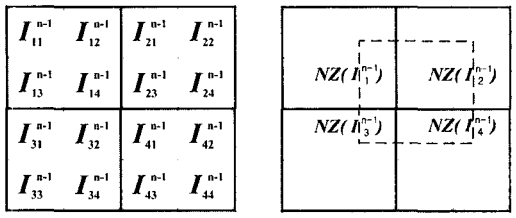


그림 2 주요한 영역을 선택하는 전방향 움직임 벡터 구성 기법

된다. 일반적으로 움직임 예측에서 많이 쓰이는 블록정합법(Block Matching Algorithm)은 객체의 경계면에서 더 큰 예측 오차를 생성하는 경향이 있다. 따라서 선택된 움직임 벡터가 더 큰 예측 오차를 가지는 움직임 벡터가 될 수 있다. 이를 개선하기 위해서는 예측 오차의 측정치가 필요한데, 압축된 비트 스트림으로부터 예측 오차의 측정치를 바로 얻어내기 위해서 일반적으로 DCT 에너지를 사용한다. 이러한 방법을 이용하여 FDVS의 단점을 개선한 ADVS[14] 기법은 DCT 에너지가 큰 매크로블록을 유력한 매크로블록으로 선택하여 그 블록의 움직임 벡터를 선택한다.

ADVS 기법에서 DCT의 에너지는 0이 아닌 DCT 계수의 개수를 이용하여 구한다. 양자화 과정을 거친 후 압축의 최소 단위인 8 × 8 블록의 DCT 계수들은 일부가 0이 된다. 일반적으로 한 개의 블록에서 0이 아닌 DCT 계수들이 많을수록 활동 상태 정보량은 많아진다. ADVS 기법에서는 DCT 계수 값의 이러한 특징을 이용하여, 인접한 매크로블록들에서 유력한 움직임 벡터를 선택할 때 각 매크로블록의 활동 상태 정보량을 측정하여 움직임 벡터를 선택하는 방법을 사용한다. 이 기법은 입력 스트림에서 0이 아닌 DCT 계수의 개수를 추출할 수 있기 때문에 추가적인 계산 복잡도 없이 움직임 벡터를 축출할 수 있다.



$$NZ(I_1^{n-1}) = NZ(I_{12}^{n-1}) + NZ(I_{14}^{n-1})$$

$$NZ(I_2^{n-1}) = NZ(I_{21}^{n-1}) + NZ(I_{22}^{n-1}) + NZ(I_{23}^{n-1}) + NZ(I_{24}^{n-1})$$

$$NZ(I_3^{n-1}) = NZ(I_{32}^{n-1})$$

$$NZ(I_4^{n-1}) = NZ(I_{41}^{n-1}) + NZ(I_{42}^{n-1})$$

$NZ(.)$: *number of nonzero quantized DCT coeffs*

그림 3 주요 활동 상태 정보량을 이용한 움직임 벡터 선택 기법

그림 3은 ADVS 기법에서 각 영역 별로 활동 상태 정보량을 축출하기 위해 DCT 계수의 개수를 구하는 방법을 보여준다. 한 예로 그림 3에서 보는 것처럼, 두 개의 블록으로 구성된 I_1^{n-1} 블록 영역의 DCT 계수의 개수가 네 개 블록 영역으로 구성된 I_2^{n-1}의 그것보다 클 수 있다. 즉 $NZ(I_1^{n-1}) > NZ(I_2^{n-1})$인 경우가 발생 할 수 있다. 이 경우 FDVS와 달리 비록 작은 영역을 차지하고 있더라도 ADVS 기법은 두 개 블록을 차지하고 있는 I_1^{n-1} 매크로블록의 움직임 벡터를 선택하게 된다.

이상으로 입력 스트림의 움직임 벡터를 활용하여 새로운 움직임 벡터를 축출하는 대표적인 두 가지 기법인 FDVS와 ADVS를 살펴보았다. 이 두 기법들은 움직임 벡터 축출에서의 가장 큰 문제인 계산 복잡도를 줄이는 것에는 기여하였으나 몇 가지 문제점을 안고 있다. 즉 인접한 매크로블록들 간에 중복 영역의 크기나 활동 상태 정보량이 비슷할 경우 어떤 매크로블록의 움직임 벡터를 선택할 것인가의 문제가 발생한다. 또한 움직임 벡터는 매크로블록 단위로 지정되어있는데 반해, ADVS 기법의 경우 매크로블록 전체가 아닌 8×8 블록의 DCT 계수 값만으로 움직임 벡터의 활동성을 판단하는 것이 문제이다. 따라서 이러한 문제점을 보완할 수 있는 방법에 관한 연구가 필요하다.

3. 영역과 활동 상태 정보를 고려한 벡터 합성 기법(Region & Activity Based Vector Composition)

FDVS와 ADVS 기법들은 그림 1의 MV2를 결정하기 위하여 중복 영역의 크기나 활동 상태 정보량 등의 가중치를 기반으로 유력한 매크로블록의 움직임 벡터를 선택하는 기법이다. 그러나 매크로블록 간의 가중치의 편차가 작을 경우 이 기법들에 의해 선택된 움직임 벡터는 의미가 없을 수 있다. 따라서 단순히 가중치를 기반으로 유력한 움직임 벡터를 선택하는 방법보다는 그림 1의 MV1이 지정하는 영역과 중첩되는 매크로블록들의 움직임 벡터들을 가중치를 기반으로 합성하는 것이 더 효율적일 수 있다. 본 논문의 이후 내용에서 MV1과 MV2는 그림 1의 움직임 벡터를 지칭한다. 또한 MV1이 지정하는 영역은 Area(MV1)으로 표기한다.

본 논문에서는 Area(MV1)과 중첩되는 매크로블록들에서 유력한 하나의 매크로블록을 선택하는 기존의 기법과는 달리 가중치를 기반으로 중첩되는 매크로블록들의 움직임 벡터들을 합성하는 기법인 영역과 활동 상태 정보를 고려한 벡터 합성 기법 (RABVC)을 제안한다. 그리고 기존의 기법에서 사용하던 가중치의 문제점을 개선한 새로운 가중치 축출 알고리즘을 제안하다.

3.1 가중치 축출

ADVS에서는 MV2를 선택하기 위해 가중치로서 활동 상태 정보량[1]을 사용하였다. 그러나 움직임 벡터의 단위인 매크로블록이 아닌 Area(MV1)과 중첩되는 8×8 내부 블록들의 활동 상태 정보량만을 사용한다. 그림 3에서 보듯이 $NZ(I_1^{n-1})$는 I_1^{n-1} 매크로블록 전체의 활동

1) 0이 아닌 DCT 계수의 개수

상태 정보량이 아닌 블록 I_{12}^{n-1}과 I_{14}^{n-1}의 활동 상태 정보량을 나타낸다. 움직임 벡터는 매크로블록 단위로 지정되기 때문에 매크로블록의 일부 블록들의 활동 상태 정보량만을 가중치로 사용하는 것은 잘못된 결과를 초래할 수 있다. 따라서 본 논문에서는 각 매크로블록 전체의 활동 상태 정보량과 중첩 영역의 크기를 고려한 가중치 축출 기법을 사용한다.

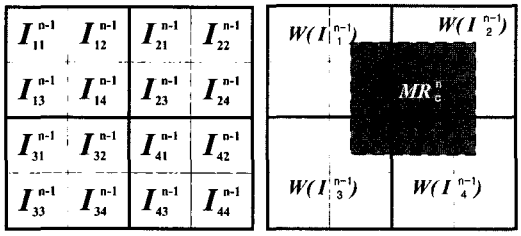


그림 4 네 개의 인접한 매크로블록에서의 가중치 설정의 예

그림 4는 네 개의 내부 블록을 가지는 인접한 네 개의 매크로블록들과 각 중첩영역의 가중치를 표시한다. 그림에서 I_a^{n-1}는 n-1번째 프레임의 a 매크로블록이고, I_{ax}^{n-1}는 I_a^{n-1}의 내부 매크로블록이다. MR_c^n(*Matching Region*)은 n번째 프레임의 c 매크로블록의 움직임 벡터가 지정하는 영역이다. 그림 4를 기준으로 각 중첩 영역의 가중치는 아래의 다섯 단계의 과정으로 구해진다.

Step 1. Area(MV1)과 중첩되는 매크로블록들의 활동 상태 정보량을 구한다.

$$AI(I_b^a) = NZ(I_{b1}^a) + NZ(I_{b2}^a) + NZ(I_{b3}^a) + NZ(I_{b4}^a) \quad (1)$$

ADVS와 달리 움직임 벡터의 정확한 활동 상태 정보량을 구하기 위해서 매크로블록 단위로 0이 아닌 DCT 계수들의 개수를 구한다. $AI(I_b^a)$는 a 프레임내의 매크로블록 b의 활동 상태 정보량을 나타낸다.

Step 2. 각 매크로블록에서 Area(MV1)과 중첩되는 부분의 크기를 픽셀 단위의 비율로 구한다.

$$RW(I_b^a) = \frac{PixelCount(OA(I_b^a))}{PixelCount(MacroBlock)} \quad (2)$$

수식 1에 의해서 구해진 활동 상태 정보량은 중첩영역의 크기가 반영되지 않아서 가중치로 바로 사용할 수 없다. 중첩영역의 크기를 반영하기 위해서 Step 2에서는 픽셀 개수를 이용하여 중첩영역의 비율을 구한다. $OA(I_b^a)$는 매크로블록 I_b^a에 존재하는 중첩영역을 나타낸다.

Step 3. 매크로블록의 중첩영역의 활동 상태 정보량을 Step 1과 Step 2의 수식으로 구한다.

$$RA(I_b^a) = AI(I_b^a) \times RW(I_b^a) \quad (3)$$

수식 1에서 구한 활동 상태 정보량은 매크로블록 전체의 값이다. 실제 적용될 값은 중첩영역의 크기를 고려해야 하기 때문에, 수식 2에서 구한 중첩영역 크기 비율 값을 이용하여 각 매크로블록에 존재하는 중첩영역의 활동 상태 정보량을 구하였다.

Step 4. Area(MV1)과 중첩되는 매크로블록들에서 중첩영역의 활동 상태 정보량의 합을 구한다.

$$SA(I_c^{a+1}) = \sum RA(I_b^a) \quad (4)$$

$MR(I_c^{a+1})$의 활동 상태 정보량을 구하기 위하여 수식 3을 이용하여 인접한 매크로블록들 내의 중첩영역 모두의 활동 상태정보량을 합하였다.

Step 5. 수식 3과 4를 이용하여 매크로블록의 가중치를 구한다.

$$W(I_b^a) = \frac{RA(I_b^a)}{SA(I_c^{a+1})} \quad (5)$$

수식 5와 같이 매크로블록 I_b^a의 움직임 벡터의 가중치는 $MR(I_c^{a+1})$의 활동 상태 정보량에 대한 I_b^a의 활동 상태 정보량을 비율로서 구한다. 다섯 단계의 과정을 통하여 구해진 가중치는 ADVS와 같은 이전의 기법의 가중치와 달리 활동 상태 정보량과 영역의 크기를 모두 정확히 고려한 가중치이다. 본 논문의 이후 내용에서는 이 가중치를 사용하는 벡터 합성 방법 (RABVC)과 벡터 선택 방법 (RADVS)을 제시한다. 벡터 선택 방법을 제시한 이유는 같은 가중치를 선택하였을 때 벡터 합성 방법의 우수성을 보이기 위한 것이다.

3.2 벡터 합성 기법(Vector Composition)

그림 5는 Area(MV1)과 중첩되는 매크로블록들에서 움직임 벡터를 보여준다. 본 논문에서 제안하는 영역과 활동 상태 정보를 고려한 벡터 합성 기법 (Region & Activity Based Vector Composition)은 앞서 구한 가중치를 이용하여 중첩 매크로블록들의 움직임 벡터들을 합성한다.

n-1 번째 프레임이 삭제되었을 때 n 번째 프레임의 a 매크로블록의 움직임 벡터는 앞서 다섯 단계의 과정을

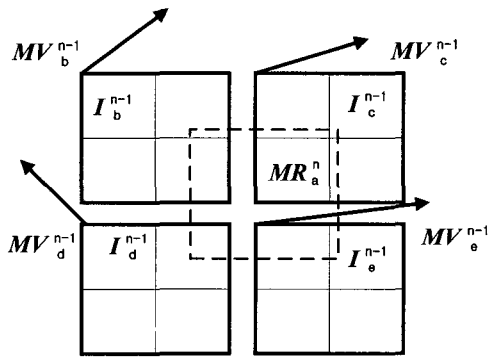


그림 5 인접 매크로블록들의 움직임 벡터

통하여 얻은 가중치를 이용하여 여섯 번째 단계에서 수식 6에 의해 재구성된다. 즉 수식에서 MV1이 MV_a^n이고 MV2는 수식의 나머지 부분이다.

Step 6. n번째 프레임 내의 a 매크로블록의 움직임 벡터를 재구성한다.

$$RefineMV_a^n = MV_a^n + \sum_{I}^{b,c,d,e}(MV_I^{n-1} \times W(I_I^{n-1})) \quad (6)$$

위의 수식에서는 그림 5에서와 같이 삭제된 $n-1$ 번째 프레임에서 MR_a^n과 중첩되는 인접 매크로블록을 각각 b, c, d, e라고 가정한다. 그림 6은 Step 1에서 Step 6까지의 모든 수식들을 알고리즘으로 정리하여 보여준다.

```
if(encoding_frame == skip_frame) {
   for(y = 0 ; y < mb_height ; y++) {
      for(x = 0 ; x < mb_width ; x++) {
         // 제거 프레임의 모든 움직임 벡터를 저장
         skip_frame_mv[x][y] = get_mv(encoding_frame, x, y);
         //제거 프레임의 모든 매크로블록의 non-zero DCT계수
         의 개수를 저장
         nzc[x][y] = get_nzdct(x,y);
      }
   }
   encoding_frame++;    // 다음 프레임으로 넘어간다.
   for(y = 0 ; y < mb_height ; y++) {
      for(x = 0 ; x < mb_width ; x++) {
         // 매크로블록마다 움직임 벡터를 축출
         cur_frame_mv = get_mv(encoding_frame, x, y);
         // 축출한 움직임 벡터가 가르키는 매칭 영역과
         // 중첩되는 매크로블록들의 움직임 벡터들을 축출한다.
         neighbor_mvs = get_nb_mvs(cur_frmae_mv);
         // 중첩영역 전체의 활동 상태 정보량을 구한다.
         sum_activity = get_sum_activity();
         // 중첩된 각 매크로블록의 가중치를 구한다.
         for(i = 0 ; i < neighbor_cnt ; i++) {
            weight[i] = get_region_activity(i) / sum_activity;
         }
         // 재구성되는 움직임 벡터를 먼저 현재의 움직임 벡터
         로 설정한다.
         refine_mv = cur_frame_mv;
         // 여기에 가중치를 적용하여 구한 움직임 벡터를 이용하여
         // 벡터합을 구한다. 최종적으로 구해지는 움직임 벡터가
         // 재구성된 움직임 벡터이다.
         for(i = 0 ; i < neighbor_cnt ; i++) {
            refine_mv = refine_mv + weight[i] * neighbor_mvs[i];
         }
      }
   }
}
```

그림 6 벡터 재구성 기법(RABVC) 알고리즘

3.3 벡터 선택 기법(Vector Selection)

영역과 활동 상태 정보가 우세한 벡터 선택 기법(Region & Activity Dominant Vector Selection)은 Step 1~Step 5까지의 과정을 통해서 얻어진 가중치들 중 가장 큰 값을 가지는 매크로블록으로부터 유력한 움직임 벡터를 선택하는 기법이다. FDVS 및 ADVS와 같은 기법으로 가중치의 크기 비교만으로 간단하게 구현된다. 그림 7은 본 논문에서 제안한 가중치를 가지고 벡터를 선택하여 움직임 벡터를 재구성하는 알고리즘이다.

```
if(encoding_frame == skip_frame) {
   for(y = 0 ; y < mb_height ; y++) {
      for(x = 0 ; x < mb_width ; x++) {
         // 제거 프레임의 모든 움직임 벡터를 저장
         skip_frame_mv[x][y] = get_mv(encoding_frame, x, y);
         //제거 프레임의 모든 매크로블록의 non-zero
         DCT계수의 개수를 저장
         nzc[x][y] = get_nzdct(x,y);
      }
   }
   encoding_frame++;    // 다음 프레임으로 넘어간다.
   for(y = 0 ; y < mb_height ; y++) {
      for(x = 0 ; x < mb_width ; x++) {
         // 매크로블록마다 움직임 벡터를 축출
         cur_frame_mv = get_mv(encoding_frame, x, y);
         // 축출한 움직임 벡터가 가르키는 매칭 영역과
         // 중첩되는 매크로블록들의 움직임 벡터들을 축출한다.
         neighbor_mvs = get_nb_mvs(cur_frmae_mv);
         // 중첩영역 전체의 활동 상태 정보량을 구한다.
         sum_activity = get_sum_activity();
         // 중첩된 각 매크로블록의 가중치를 구한다.
         for(i = 0 ; i < neighbor_cnt ; i++) {
            weight[i] = get_region_activity(i) / sum_activity;
         }
         // 재구성되는 움직임 벡터를 먼저 현재의 움직
         임 벡터로 설정한다.
         refine_mv = cur_frame_mv;
         // 여기에 가장 큰 가중치를 가지는 매크로블록
         의 움직임 벡터를
         // 이용하여 벡터합을 구한다. 최종적으로 구해
         지는 움직임 벡터가
         // 재구성된 움직임 벡터이다.
         refine_mv = refine_mv + get_max_weight_mv(weight,
         neighbor_mvs);
      }
   }
}
```

그림 7 벡터 선택 기법(RADVS) 알고리즘

4. 실험 및 성능 평가

본 장에서는 여러 가지 측정 인자를 이용하여 본 논문에서 제안한 RABVC(Region & Activity Based Vector Composition)의 성능을 비교 실험하였다. RABVC의

성능을 비교 실험하기 위해서 본 논문에서는 RABVC와 같은 가중치를 사용하는 RADVS (Region & Activity Dominant Vector Selection)기법을 제안하였다. 이와 함께 개선된 움직임 예측 기법인 PHODS(Parallel Hierarchical One Dimension Search)[20], EPZS (Enhanced Predictive Zonal Search)[21]와 기존의 벡터 재사용 기법들인 Bilinear, FDVS, ADVS도 비교 대상으로 사용하였다.

실험을 위하여 세 개의 QCIF 포맷의 H.263 비디오 파일들과 하나의 CIF 포맷의 H.263 비디오 파일을 사용하였다. 세 개의 QCIF 포맷의 비디오 파일은 움직임의 정도를 세 가지로 분류하여 선정하였다. "Football" 비디오는 움직임 가장 많은 비디오이고, "Carphone"과 "Suzie"의 경우 각각 움직임이 다소 있는 비디오와 작은 비디오이다. 유일하게 CIF 포맷인 "Tempete" 비디오는 높은 해상도와 줌 아웃이 있는 예외적인 비디오라는 이유로 실험에 사용되었다. 비디오 파일들은 첫 번째 프레임은 intraframe(I-frame)으로 코딩되었으며, 나머지 프레임들은 interframe(P-frame)으로 코딩되었다. 이것은 트랜스코딩으로 출력되는 비디오 스트림에서도 동일하다. 트랜스코더와 비디오 코덱은 ffmpeg[22]의 H.263 코덱을 수정하여 구현하였다. 표 1과 표 2는 각각 입력 비디오 스트림과 실험 환경을 보여준다.

표 1 입력 비디오 스트림

비디오 이름	양자화 파라미터	평균 비트율(Kbps)	Video Type	프레임률 (fps)	프레임 개수
Football	3	256	QCIF	30	258
Carphone		584			381
Suzie		512			150
Tempete		4746	CIF		256

표 2 실험 환경

실험 시스템	Pentium IV 1.2 Ghz CPU	CPU
	256M	메모리
	Redhat Linux 9.0	운영체제
비디오 포맷	H.263	"IPPP.." 로 구성
비디오 코덱	ffmpeg	version 0.4.8
출력 비디오 프레임률	15	절반의 프레임을 삭제
출력 비디오 양자화 파라미터	5, 15, 25	
탐색 윈도우 크기	± 1	픽셀 단위

RABVC와 비교하는 기법들은 움직임 예측 기법으로 Full ME, PHODS, EPZS의 세 가지 기법을 사용하였고 움직임 벡터 재사용 기법으로 Bilinear, FDVS, ADVS, RADVS의 네 가지 기법을 사용하였다. 이 중 RADVS는 본 논문에서 제안한 가중치 축출 기법을 사용하는 유력 벡터 선택기법이다. MV1과 MV2의 벡터 합으로 얻어진 MV'를 기준으로 최적의 움직임 벡터를 검색하기 위해서 움직임 벡터 재사용 기법들은 정해진 탐색 윈도우 내에서 Full Search를 사용하여 검색한다. 탐색 윈도우의 크기는 표 2에서 제시한 +1이다. 모든 움직임 벡터 재사용 기법을 같은 벡터 탐색 기법을 이용하여 실험한 것은 일단 본 논문에서 제안한 벡터 재구성 기법의 성능을 비교 평가하기 위한 것이다. RABVC와 연동되는 벡터 탐색 기법에 대한 연구와 이 기법을 [15,16]의 HAVS와 VSS와 비교 실험하는 것은 향후 연구과제로 남긴다.

이어지는 내용에서 그림 8, 9, 10, 11은 움직임 예측 기법들 중 가장 높은 PSNR값을 가지는 Full ME와 다섯 가지 움직임 벡터 재사용 기법들의 프레임 당 PSNR 값을 비교하여 보여준다. 그림 12, 13, 14, 15는 움직임 예측 기법들 중 가장 작은 처리 시간을 보여주는 EPZS와 움직임 벡터 재사용 기법들 중 작은 처리시간을 보여주는 두 가지 기법(FDVS, ADVS)과 논문에서 제안한 RABVC 기법의 프레임 처리 시간을 비교하

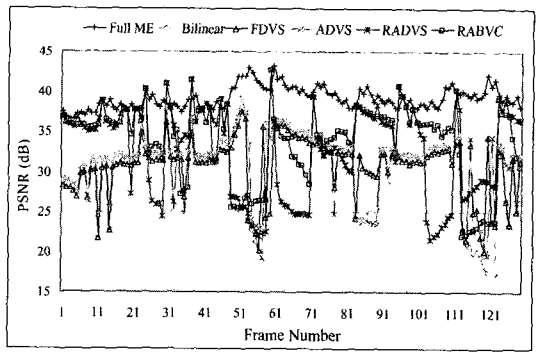


그림 8 Football 비디오의 PSNR 비교

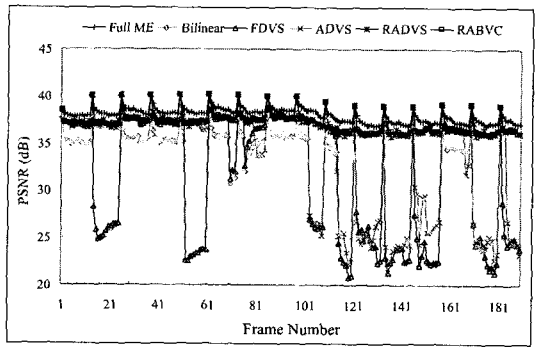


그림 9 Carphone 비디오의 PSNR 비교

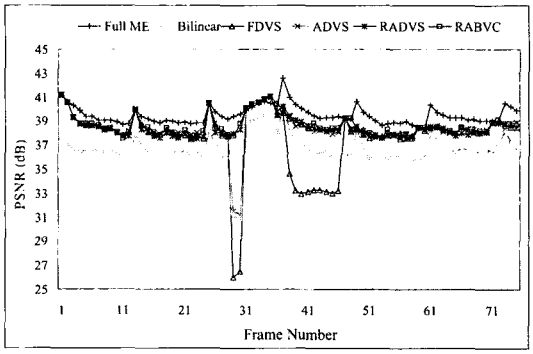


그림 10 Suzie 비디오의 PSNR 비교

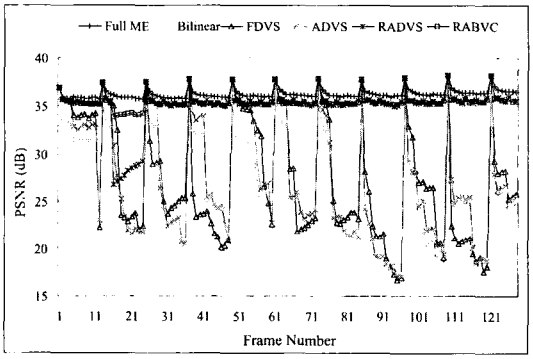


그림 11 Tempete 비디오의 PSNR 비교

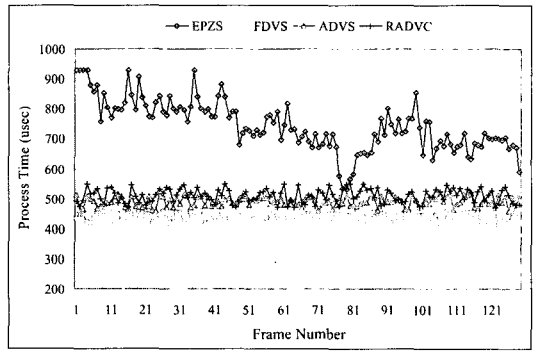


그림 12 Football 비디오의 프레임 당 처리 시간 비교

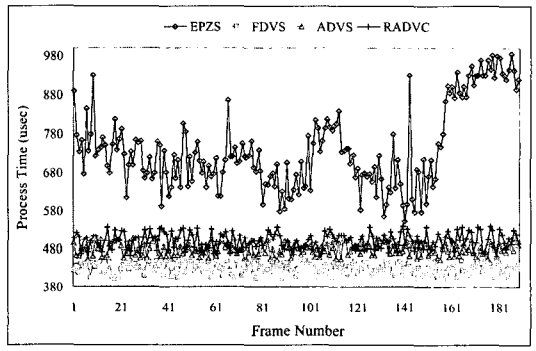


그림 13 Carphone 비디오의 프레임 당 처리 시간 비교

여 보여준다. 그림 8~15에서의 모든 출력 스트림은 양자화 파라미터 5로 했을 때의 결과이다. 모든 기법들과 세 가지 양자화 파라미터에 대한 처리시간 및 PSNR 값의 비교는 표 3, 4, 5, 6에서 보여준다.

그림 8~11에서 Full ME의 경우 움직임의 정도에 상관없이 높은 품질의 PSNR을 작은 편차로 유지하는 것을 알 수 있다. 본 논문에서 제안한 RABVC의 경우 움직임이 많은 "Fooball" 비디오에서 Full ME에 비해 다소 편차가 클 뿐 나머지 비디오에서는 Full ME에 근접하게 높은 품질의 PSNR을 작은 편차로 제공한다.

그림 8~11에서 Bilinear, FDVS, ADVS와 같은 기존의 움직임 벡터 재사용 기법들은 움직임이 활발히 있는 부분에서는 PSNR 값이 급격하게 떨어진다. 이것은 기존의 움직임 벡터 재사용 기법들이 움직임 벡터들의 변화가 큰 부분에서는 벡터의 재구성에 적합하지 않다는 근거이다. 반면 본 논문에서 제안한 RABVC의 경우 "Footbal"과 같이 움직임 매우 큰 비디오에서 다소 편차가 있을 뿐 대부분의 비디오에서 작은 편차의 PSNR을 제공한다.

본 논문에서 제안한 가중치를 사용한 RADVS와 RABVC의 경우, 벡터 합성 기법을 사용한 RABVC가 더 좋은 PSNR과 작은 편차를 보여준다. 이것은 유력한 하나의 벡터를 선택하는 것보다는 활동 상태 정보량과 같은 여러 인자를 고려하여 중첩되는 모든 매크로블록의 움직임 벡터를 합성하는 것이 더 좋은 결과를 보인다는 것을 나타낸다. 또한 RADVS의 경우 FDVS, ADVS와 같은 기존의 벡터 선택 기법보다 근소한 차이로 좋은 결과를 보여준다. 이 결과로부터 본 논문에서 제시한 방법으로 축출한 가중치가 더 정확한 활동 상태 정보량을 구해줌을 알 수 있다.

그림 12, 13, 14, 15에서 보이듯이 움직임 예측 기법들 중 가장 작은 처리 시간을 요구하는 EPZS의 경우 움직임 벡터 재사용 기법에 비해 2-3배 정도 높은 처리 시간을 요구하며 비디오의 움직임 여부에 따라 처리시간의 편차도 매우 높다. 반면 기존의 움직임 벡터를 재사용하는 기법들은 프레임들의 움직임 변화의 대소 여부에 관계없이 작은 편차의 처리 시간을 요구한다. 본 논문에서 제안한 RABVC의 경우 FDVS, ADVS에 비해 고려하는 가중치 인자들이 많기 때문에 다소 높은 처리 시간을 보여준다.

그림 8~15를 살펴보면 본 논문에서 제안한 RABVC가 기존의 여러 움직임 벡터 재사용 기법과 비슷한 계

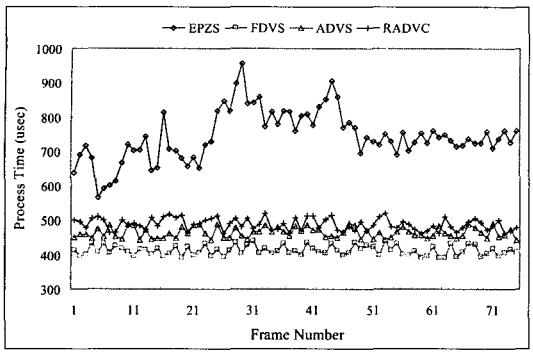


그림 14 Suzie 비디오의 프레임 당 처리 시간 비교

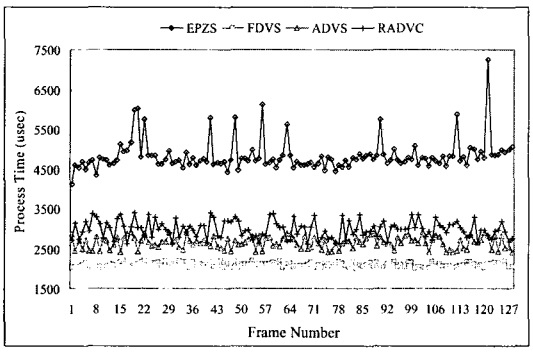


그림 15 Tempete 비디오의 프레임 당 처리 시간 비교

산 복잡도에서 더 좋은 화질(PSNR)을 가진다는 것을 알 수 있다. 또한 RABVC가 기존의 개선된 여러 움직임 예측 기법에 비해 5%~60%의 계산 복잡도로서 만족할 만한 화질을 제공할 수 있음을 보였다. 이것은 표 3, 4, 5, 6의 각 비디오 파일에 대한 평균 처리시간과 PSNR에서 좀 더 분명하게 확인할 수 있다.

결과의 신뢰도를 높이기 위해 표 3~6에서는 여러 움직임 예측 기법들과 움직임 벡터 재사용 기법들에서 세 가지 양자화 스텝 별 평균 처리 시간, PSNR, 출력 비트율을 측정하여 보여준다. 표에 나타난 평균 처리 시간, PSNR, 출력 비트율을 통해 RABVC를 평가하면 다음과 같다. 첫째, RABVC는 여러 개선된 움직임 예측 기법들에 비해 낮은 계산 복잡도를 가진다. 둘째, RABVC는 여러 움직임 벡터 재사용 기법에 비해 높은 화질(PSNR)을 제공한다. 셋째, RABVC는 여러 개선된 움직임 예측 기법들과 비교하여 작은 계산 복잡도에 20%정도 높은 출력 비트율을 제공한다. 이 세 가지 평가를 종합하면 논문에서 제안한 RABVC 기법을 적용한 트랜스 코더는 작은 계산 복잡도를 통해 많은 스트림을 지원할 수 있으며, 높은 PSNR을 통해 사용자에게 적당한 화질의 서비스를 제공할 수 있다. 또한, 기존의 벡터 재사용 기법들에 비해 개선된 출력 비트율은 네트워크에 적응성 있는 트랜스코더의 구현을 도울 것이다.

표 3 Football의 평균 처리 시간, PSNR, 출력 비트율

비교 항목	양자화 파라미터	Full ME	PHODS	EPZS	Bilinear	FDVS	ADVS	RADVS	RABVC
계산 시간 (usec)	5	12195	797	764	485	436	490	503	511
	15	11224	951	893	476	438	475	499	505
	25	11943	852	779	481	456	501	515	520
PSNR(dB)	5	39.14	35.23	36.67	29.81	30.23	30.28	33.2	33.97
	15	34.63	33.21	34.12	25.21	25.7	26.65	31.89	32.15
	25	31.8	30.2	31.52	23.2	24.48	25.5	29.9	29.93
비트율(kbps)	5	292	325	301	416	414	414	407	392
	15	98	110	96	172	167	166	141	125
	25	59	72	62	126	121	121	119	101

표 4 Carphone의 평균 처리 시간, PSNR, 출력 비트율

비교 항목	양자화 파라미터	Full ME	PHODS	EPZS	Bilinear	FDVS	ADVS	RADVS	RABVC
계산 시간 (usec)	5	12167	1073	752	476	425	481	492	497
	15	12198	903	806	483	435	498	501	513
	25	11672	1084	793	469	424	480	492	485
PSNR(dB)	5	37.98	37.23	37.72	30.86	32.1	33.7	37.1	37
	15	31	30.78	30.77	27.52	28.11	28.9	30.39	30.56
	25	28.26	27.87	28.16	25.41	26.13	26.76	27.52	27.86
비트율 (kbps)	5	202	217	198	283	281	281	263	235
	15	55	58	52	78	77	77	71	67
	25	34	34	29	43	43	43	42	38

표 5 Suzie의 평균 처리 시간, PSNR, 출력 비트율

비교 항목	양자화 파라미터	Full ME	PHODS	EPZS	Bilinear	FDVS	ADVS	RADVS	RABVC
계산 시간 (usec)	5	12015	924	744	472	412	466	489	492
	15	10993	1321	812	469	430	475	493	511
	25	13616	880	718	481	425	481	501	506
PSNR(dB)	5	39.47	39.12	39.44	36.72	37.75	38.38	38.53	38.59
	15	33.22	33.02	33.16	32.42	32.89	32.98	32.99	33
	25	30.88	30.8	30.8	30.44	30.63	30.75	30.76	30.8
비트율 (kbps)	5	102	113	98	145	135	134	132	126
	15	32	34	29	45	44	44	43	37
	25	24	24	20	33	33	33	32	28

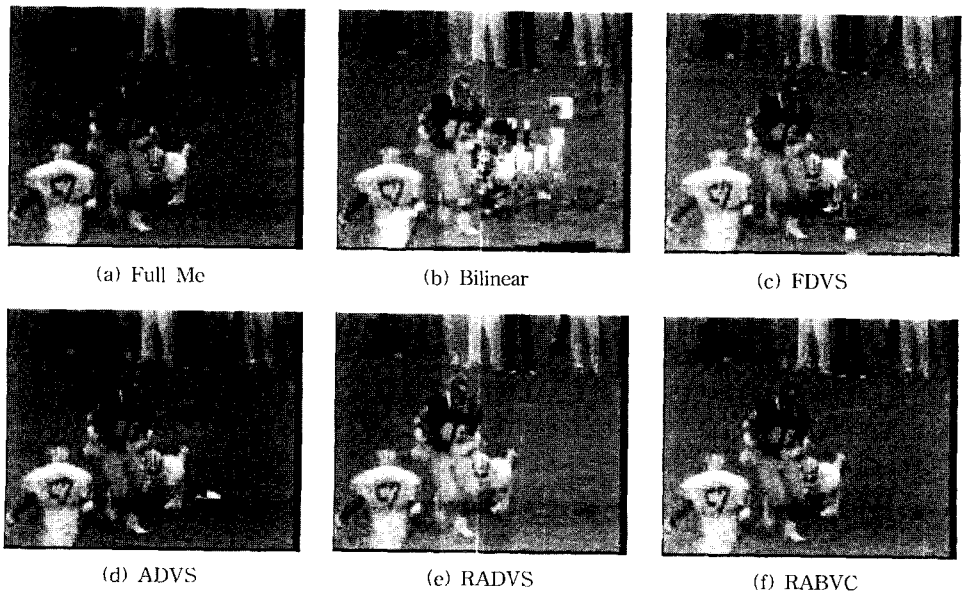
표 6 Tempete의 평균 처리 시간, PSNR, 출력 비트율

비교 항목	양자화 파라미터	Full ME	PHODS	EPZS	Bilinear	FDVS	ADVS	RADVS	RABVC
계산 시간 (usec)	5	53103	5558	4849	2573	2093	2639	2813	2969
	15	51301	5610	4736	2625	1968	2722	2777	3011
	25	51057	5613	5235	2611	2133	2691	2915	3127
PSNR(dB)	5	36.33	35.66	36.12	26.74	27.4	27.65	35.19	35.47
	15	28.64	28.12	28.3	24.55	24.97	25.05	28.21	28.31
	25	25.95	25.6	25.76	23.43	23.53	23.63	25.35	25.5
비트율 (kbps)	5	1581	1706	1649	1935	1829	1816	1775	1754
	15	386	427	395	495	486	486	472	451
	25	193	209	201	247	246	244	239	225

그림 16과 17은 "Football" 비디오에서 87번째 프레임과 "Tempete" 비디오의 96번째 프레임의 실제 이미지를 보여주고 있다. 각 비디오들의 움직임이 많은 부분에서 기존의 움직임 벡터 재사용 기법들에 비해 RABVC와 RADVS는 좋은 화질을 보여준다.

5. 결론 및 향후 연구과제

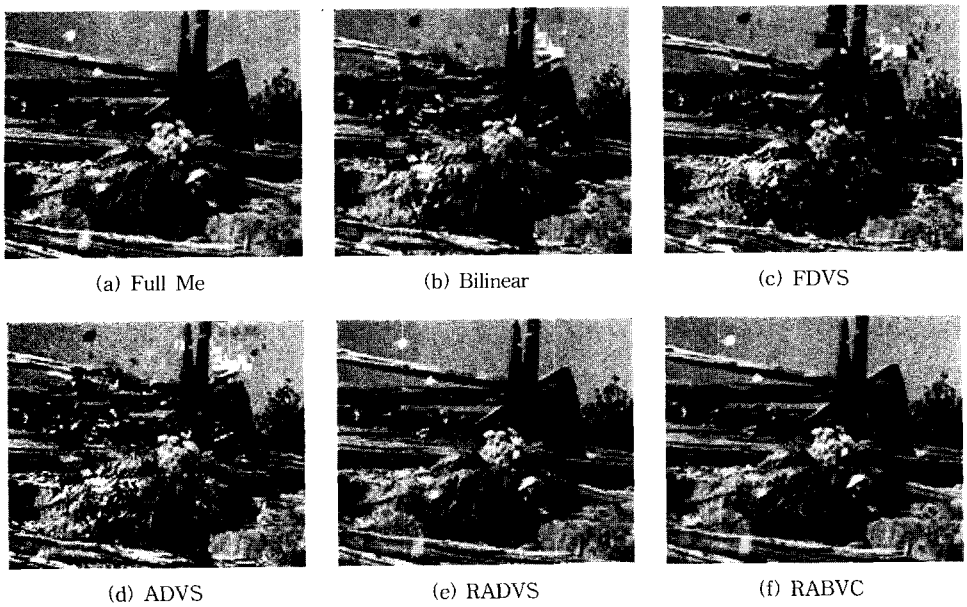
본 논문에서는 시간당 프레임의 개수를 조절하는 트



(a) Full Me (b) Bilinear (c) FDVS

(d) ADVS (e) RADVS (f) RABVC

그림 16 Football 출력 스트림 중 87번째 프레임의 화질 비교



(a) Full Me (b) Bilinear (c) FDVS

(d) ADVS (e) RADVS (f) RABVC

그림 17 Tempete 출력 스트림 중 96번째 프레임의 화질 비교

랜스코딩 기법에서 움직임 벡터를 재 정의할 때, 삭제되는 프레임의 움직임 벡터를 재사용해서 움직임 벡터를 재구성하는 기법인 영역과 활동 상태 정보를 고려한 벡터 합성 기법(RABVC)을 제안하였다. RABVC에서는 움직임 벡터의 정확한 재사용을 위하여 활동 상태 정보량과 중첩 영역의 크기를 고려한 새로운 가중치 축출 기법을 만들었다. RABVC는 이 가중치를 이용하여 중첩되는 매크로블록들의 움직임 벡터를 합성하여 새로운 움직임 벡터를 재구성하는 기법이다.

실험 결과 제안된 RABVC는 가중치를 기반으로 우세한 움직임 벡터를 선택하여 움직임 벡터를 재 정의하는 기법인 ADVS 등과 비슷한 계산 복잡도에서 full-scale 움직임 예측과 비슷한 수준의 PSNR값을 보였다. 이것은 실제 트랜스코딩 된 이미지에서 시각적으로 인지할 수 있을 정도로 분명한 것이었다. 또한 동일한 가중치를 사용한 벡터 선택 기법(RADVS)에 비해 벡터 합성 기법(RABVC)의 성능이 더 높게 나왔다. 따라서 벡터 선택 기법에 비해 벡터 합성 기법이 벡터 재사용 기법에 적당함을 알 수 있었다.

본 논문에서는 최적의 움직임 벡터를 축출하기 위한 벡터 탐색 기법으로 이미 많이 사용되고 있는 탐색윈도우 내의 full search 기법을 사용하였다. 이것은 본 논문의 큰 약점일 수 있다. 따라서 앞장에서도 언급했듯이 [15]에서 사용하는 HAVS와 VSS와 같은 벡터 탐색 기법을 RABVC와 연동되게 연구하는 것을 향후 연구과제로 남긴다.

참 고 문 헌

[1] 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.

[2] N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," IEEE Trans. Consumer Electron., vol. 44, pp. 88-98, Feb. 1998.

[3] T. Warabino, S. Ota, D. Morikawa, M. Ohashi, H. Nakamura, H. Iwashita and F. Watanabe, "Video Transcoding Proxy for 3Gwireless Mobile Internet Access," IEEE Communications Magazine, 38, (10), pp. 66-71, Oct. 2000.

[4] N. Bjork and C. Christopoulos, "Video Transcoding for Universal multimedia Access," ACM Workshop on Multimedia Standards, Interoperability and Practice(MM2000 Workshop), Proceedings of ACM Multimedia 2000, pp. 75-79, November 4, 2000.

[5] T. shanableh and M. Ghanbari, "Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats," IEEE Trans. Multimedia, vol. 2, no. 2, pp. 101-110, June 2000.

[6] Y. Nakajima and M. Sugano, "MPEG bit rate and format conversions for heteroheneous network/storage applications," IEICE Trans. Electron., E85-C, (3), pp. 492-503, Mar. 2002.

[7] M. Emad Modirzadeh, "Bit-Rate Reduction of MPEG Compressed Video," IEEE Canadian Con-

ference on Electrical and Computer Engineering, 2002.
[8] R.J. Safranek, C. Kalmanek, and R. Garg, "Methods for matching compressed video to ATM networks," in Proc. Int. Conf. Image, Washington, DC, Oct. 1995.
[9] J. Youn, M.T. Sun, and J. Xin, "Video Transcoder Architectures for Bit Rate Scaling of H.263 Bit Streams," Proceedings of ACM Multimedia'99, pp. 243-250, Nov., 1999.
[10] H.Sun, W. Kwok, and J. W. Zdepski, "Architecture for MPEG compressed bitstream scaling," IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 191-199. Apr. 1996. J. Youn and M.T.
[11] Sun, and C. W. Lin, "Motion Estimation for High Performance Transcoding," IEEE Trans. Consumer Electron., vol. 44, pp. 649-658, Aug. 1998.
[12] 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.
[13] J. N. Hwang, T. D. Wu and C. W. Lin, "Dynamic Frame-skipping in Video Transcoding," IEEE 2nd Workshop on Multimedia Signal Processing, Dec. 1998.
[14] K. T. Fung, Y. L. Chan and W. C. Siu, "New architecture for dynamic frame-skipping transcoder," in Proc. IEEE Workshop Multimedia signal Processing, Redondo Beach, CA, pp. 616-621, Dec. 1998.
[15] 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.
[16] 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.
[17] B. Shen, I. K. Sethi and B. Vasudev, "Adaptive motion vector resampling for compressed video downscaling," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 929-936, Sept. 1999.
[18] "Video coding for low bitrate communication," International Telecommunications Union, Geneva, Switzerland, ITU-T Recommendation H.263, 1998.
[19] J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe coding," IEEE Transactions on Communications, vol. COM-29, no. 12, December 1981, pp. 1799-1808.
[20] L. G. Chen, W. T. Chen, Y. Jehng, and T. Chiueh, "An efficient parallel motion estimation algorithm for digital image processing," IEEE Trans. on Circuits and Systems for Video Technology, vol. 1, pp. 378-85, Dec. 1991.
[21] "Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation," Alexis M. Tourapis1, XiWave plc Bath, BA1 2PH, United Kingdom.
[22] http://www.ffmpeg.org

이 승 원

1997년 부산대학교 전자계산학과 졸업(학사). 1999년 부산대학교 대학원 전자계산학과 졸업(이학석사). 2004년 부산대학교 대학원 전자계산학과 졸업(이학박사). 관심분야는 트랜스코더, 유비쿼터스 컴퓨팅, 멀티미디어 스트리밍

박 성 호

1996년 부산대학교 전자계산학과 졸업(학사). 1998년 부산대학교 대학원 전자계산학과 졸업(이학석사). 2002년 부산대학교 대학원 전자계산학과 졸업(이학박사). 2002년 9월~현재 부산대학교 정보전산원 조교수. 관심분야는 트랜스코딩, 멀티미디어 캐싱, 이동통신

정 기 동

1973년 서울대학교 졸업(학사). 1975년 서울대학교 대학원 졸업(석사). 1986년 서울대학교 대학원 계산통계학과 졸업(이학박사). 1990년~1991년 MIT, South Carolina 대학 교환 교수. 1995년~1997년 부산대학교 전자계산소 소장. 1978년~현재 부산대학교 전자계산학과 교수. 관심분야는 병렬처리, 멀티미디어 시스템, 멀티미디어 통신