論文2003-40SP-4-6

H.264의 가변 블록 움직임 보상을 위한 고속 움직임 벡터 탐색 및 모드 결정법

(Fast motion estimation and mode decision for variable block sizes motion compensation in H.264)

李 帝 玧 *, 崔 雄 一 *, 全 炳 宇 **, 石 玟 秀 **

(Jeyun Lee, Woongil Choi, Byeung Woo Jeon, and Minsoo Suk)

요 약

최근 표준화가 완성된 H.264는 가변 크기 블록 움직임 보상, 복수 참조 영상, 그리고 1/4 화소 움직임 벡터 정확도를 지원하고 있다. 그러나 이러한 새로운 부호화 기술은 부호화 효율 향상의 주된 요인이 되긴 하지만 동시에 높은 복잡도의 요인이기도 하다. 따라서 H.264 비디오 표준의 실제 응용 확대를 위해서는 이러한 기술에 대한 속도 향상이 필요하다. 본 논문에서는 부호화 복잡도의 가장 큰 비중을 차지하는 움직임 벡터 추정을 고속화하기 위해 가변 크기 블록 움직임 보상에 적합한 고속 움직임 벡터 탐색법을 제안한다. 또한 H.264가 갖는 다양한 부호화 모두 중 최적치를 빠르게 결정할 수 있는 고속 결정법을 제안한다. 실험 결과, 고속 움직임 벡터 탐색법은 기존 MVFAST[8 10]에 비해 SAD 계산 수가 4.5배, SATD 계산 수는 2.6배 감소하였으며, 고속 모드 결정법의 사용으로 비트율-왜곡 계산치인 RDcost 계산 수는 45%가 감소되었다. 따라서 제안된 방법은 큰 부호화 손실 없이 계산량 감소에 있어서 매우 높은 효율을 제공함을 확인하였다.

Abstract

The new video coding standard H.264 employs variable block size motion compensation, multiple references, and quarter-pel motion vector accuracy. These techniques are key features to accomplish higher coding gain, however, at the same time main factors that increase overall computational complexity. Therefore, in order to apply H.264 to many applications, key techniques are requested to improve their speed. For this reason, we propose a fast motion estimation which is suited for variable block size motion compensation. In addition, we propose a fast mode decision method to choose the best mode at early stage. Experimental results show the reduction of the number of SAT, SATD calculations by a factor of 4.5 and 2.6 times respectively, when we compare the proposed fast motion estimation and the conventional MVFAST[8 10]. Besides, the number of RDcost computations is reduced by about 45%. Therefore, the proposed methods reduces significantly its computational complexity without noticeable coding loss.

Keyword : encoder optimization, fast motion estimation, diamond search, fast mode decision

* 正會員, ** 學生會員, 成均館大學校 情報通信工學部
(Dept. of Information and Computer Engineering, Sungkyunkwan University)

※ 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2002-041-D00405).
接受日字:2003年6月5日, 수정완료일:2003年6月26日

Ⅰ. 서 론

기존의 H.263 이나 MPEG-4 표준들과 같이, 최근 표준화가 마무리된 H.264 표준도 역시 블록단위 움직임

보상 구조를 채택하고 있다. 이러한 움직임 보상 기법은 프레임간에 존재하는 중복성을 제거하여 압축 효율을 높이는 기술이다. 특히 H.264의 움직임 보상 블록 크기는 <그림 1>과 같이 16×16에서부터 4×4에 이르기까지 다양하다. 이러한 다양한 크기와 모양의 가변 블록 움직임 보상 기법은 주관적 화질과 부호화 효율에 있어 많은 이득을 제공한다. 또한 복수개의 참조 영상 중 현재 블록과 유사도가 가장 높은 것을 참조 영상으로 택할 수 있어, 특정 영상이 반복될 경우에는 큰 부호화 이득을 얻을 수 있다. 그리고 H.264의 움직임 벡터의 정확도는 1/4 화소까지 지원하고 있기 때문에, 기존 표준들 보다 움직임 추정을 통해 발생되는 잉여 데이터의 양을 더욱 효율적으로 줄일 수 있다. 이런 기술 외에 움직임 보상을 위한 블록 크기 및 모양 선택, 참조 영상 선택, 움직임 벡터 선택 기법 역시 부호화 효율에 큰 영향을 미치는 요인인데, 이러한 다양한 부호화 조건 중 최적 조합을 결정하기 위해 H.264에서는 RD 최적화 ((Rate-Distortion Optimization) 기술을 사용하고 있다[1]. 이러한 H.264의 특징적인 부호화 기술로 인해, H.263이나 MPEG-4에 비해 동일한 비트율에서는 PSNR이 2dB정도 높고, 동일한 PSNR에서는 영상 압축률이 2배 더 좋다는 것이 실험을 통해 밝혀졌다[2, 3]. 그러나 복잡하게 상호 연관된 구조로 부호화하기 때문에 복잡도는 MPEG-4 SP(Simple Profile)보다 약 16배 높다는 것도 분석되었다[4].

H.264에서 움직임 보상을 위해 사용할 수 있는 블록의 종류는 <그림 1>과 같다. <그림 1>에서 SUB8×8 모드는 매크로 블록내의 4개의 8×8 블록들이 독립적으로 8×8, 8×4, 4×8, 4×4의 움직임 보상 구조를 가질 수 있는 모드를 뜻한다. 이런 다양한 구조의 움직임 보상은 영상의 특징과 영상 내부 물체의 움직임 특성에 잘 맞추어 부호화할 수 있는 효율적인 기술이다. 예를 들어 움직임이 적고, 배경이나 영상내의 물체 크기가 클 경우에는 16×16 블록이 많이 선택되고, 회전이나 화면 확대처럼 복잡한 움직임, 그리고 영상내 물체 크기가 작을 경우에는 작은 블록 단위로 움직임 보상이 이루어진다. 특히 SUB8×8 모드로 움직임 보상을 할 경우 객관적인 부호화 효율은 작지만, 주관적인 화질에서는 뛰어난 성능을 나타내고 있다.

H.264의 움직임 보상을 위한 다양한 블록 크기 중, 16×16, 16×8, 8×16, 8×8은 각각 복수 개의 참조 영

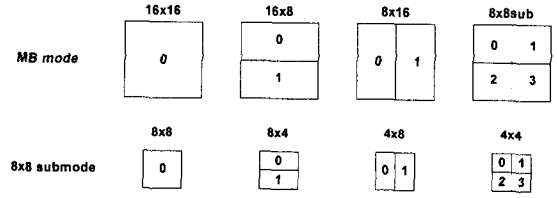
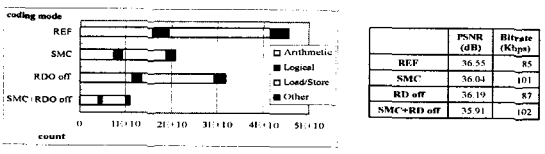


그림 1. 다양한 크기의 움직임 보상을 위한 매크로블록 분할

Fig. 1. Macroblock segmentation for variable block size motion compensation.

상을 이용하여 부호화 될 수 있다. 예를 들어 매크로블록이 8×8 단위로 움직임 보상된다면, 4개의 8×8블록의 참조 영상이 모두 다를 수 있다. 즉, 참조 영상의 정보를 최소 8×8단위까지 가질 수 있다. 특히 mobile, tempete와 같이 회전이나 반복적인 움직임이 많은 영상인 경우, 참조 영상을 바로 이전 영상만을 이용했을 때보다 PSNR이 최고 1dB까지 증가한다[5]. 그리고 가능한 최대 참조 영상 수에 대한 권고 사항은 프로파일과 레벨에 맞춰 명시되어 있다[6]. 또한 H.264의 움직임 벡터는 1/4 화소 정확도로 추정되어진다. 이를 위해 6-tap 필터를 이용하여 참조 영상을 1/2 화소 단위로 보간한 후, 1/2 위치의 화소 값들을 다시 선형 보간 법을 이용하여, 1/4 위치의 화소를 생성한다. 따라서 움직임 벡터 추정을 위하여 모든 참조 영상은 4배에 해당되는 화소 보간이 이루어 져야 한다.



	PSNR (dB)	Bitrate (Kbps)
REF	36.55	85
SMC	36.04	101
RD off	36.19	87
SMC+RD off	35.91	102

그림 2. H.264 부호 모드에 따른 성능 비교

Fig. 2. H.264 performance comparison with regard to encoding modes.

<그림 2>는 여러 부호화 모드의 복잡도를 나타내고 있다. 여기서 'REF (Reference)'로 표시된 부호화 모드는 H.264 표준의 모든 부호화 기술을 사용하여 부호화한 것이다. 그리고 'SMC (Simplified Motion Compensation)'로 표시된 모드는 바로 이전 참조 영상만을 이용하여 16×16 블록 크기로만 움직임 보상한 것을 의미한다. 'RDO(Rate-Distortion Optimization) off'로 표시된 모드는 REF 모드에서 RD 최적화 기술만을 제외한 것이다. 마지막으로 'SMC+RDO off'로 표시된 모드

는 'SMC' 모드와 'RDO off' 모드가 결합된 것으로, 움직임 보상은 16×16 블록 크기로만 하고, 바로 이전 영상만을 참조 영상으로 하여 부호화 한 것이다. 실험을 위해 사용된 부호기는 'JM(Joint Model) 6.1d'이고, foreman, QCIF 영상을 Qp(Quantization Parameter) 28로 부호화한 것이다. 복잡도 분석은 'iprof (instruction profile)'[7]를 사용하였는데, 이는 MPEG 그룹에서 영상 알고리듬 각각에 대한 복잡도 분석을 위해 사용한 적이 있는 프로그램으로, 연산 종류별 사용 빈도 수를 분석해준다.

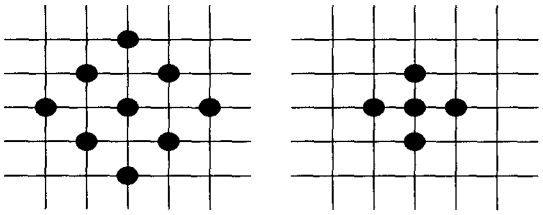
'SMC' 모드는 'REF' 모드에 비해 PSNR은 0.41dB 감소되었고, 비트율은 약 19%정도 증가하였다. 복잡도와 관계있는 연산량은 절반이상 감소하여, 가변 블록 움직임 보상과 복수 참조 영상이 부호화기의 대부분의 복잡도를 차지하고 있음을 알 수 있다. 그리고, 'REF' 모드에서 RD 최적화 기술을 사용하지 않았을 때는 PSNR이 0.36dB 감소하였고, 비트율은 2.35%정도 증가하였다. 이때 줄어든 연산량은 약 25%인데 , 이것으로 미루어 보아 RD 최적화 기술이 부호화기에서 차지하는 비율이 상당히 높다는 것을 알 수 있다. 마지막으로 'SMC+RDO off' 모드로 부호화하였을 때, 'REF' 모드에 비해 PSNR은 0.64dB 감소하였으며, 비트율은 20% 증가하였다. 이 경우 줄어든 연산량은 'REF' 모드에 비해 70%이상 감소하였다. 이렇듯 움직임 보상과, RD 최적화 기술은 부호화기 성능에 미치는 영향이 크지만, 그에 따른 복잡도 또한 매우 높다. 따라서 본 논문에서는 부호화기의 부호화 성능은 유지하면서 계산량을 최소화하기 위해, 가장 많은 계산량을 차지하는 가변 블록 움직임 보상 구조에 적용할 수 있는 고속 움직임 벡터 탐색과 RD 최적화 기반 모드 결정의 계산량을 줄이기 위한 고속 모드 결정 알고리듬을 제안한다.

Ⅱ. 고속 움직임 벡터 탐색법

제안한 고속 움직임 벡터 탐색법은 기존의 MVFAST(Motion Vector Field Adaptive Fast Motion Estimation)[8-10]로 알려진 알고리듬을 기초로 고안되어 졌다. MVFAST는 DS(Diamond Search)[11]를 개선한 방법으로서, DS에 비해 부호 효율과 움직임 벡터 탐색 속도에 있어 큰 성능 향상을 이루었다. 제안한 고속 움직임 벡터 탐색법은 특히 정수 화소 위치에서 움직임 벡터를 찾을 때 적용할 수 있는 것으로서, 주변 블록의 움직임 벡터 활성도를 낮음, 중간, 높음으로 구분하고, 각 구분에 따라 탐색 방법과 초기 탐색 위치를 다르게 한 것이다. 이에 대한 구체적인 방법은 다음과 같다. 최초, 왼쪽, 위쪽, 오른쪽 위 블록 각각의 움직임 벡터의 크기 l 을 구한다. 이 때 움직임 벡터 $v_i=(x_i, y_i)$의 크기는 $l_{v_i}=|x_i|+|y_i|$로 정의한다. 다음으로, 주변 블록의 움직임 벡터 크기 중 가장 큰 것을 두 개의 문턱치 (L_1, L_2)와 비교하여 움직임 활성도를 구한다. 움직임 활성도를 분류하는 구체적인 수식은 다음과 같다.

$$\text{움직임 벡터 활성도} = \begin{cases} \text{낮음}, & l \le L_1 \\ \text{중간}, & L_1 < l \le L_2 \\ \text{높음}, & l > L_2 \end{cases} \quad (1)$$

주변 블록의 움직임 활성도와 현재 블록의 움직임 활성도 사이에는 상관도가 높기 때문에, 움직임 벡터 활성도에 따라 초기 탐색 위치와 탐색 방법을 다르게 적용하는 것이 합리적일 것이다. 즉 주변 블록의 움직임 벡터 활성도가 낮다면, 현재 블록에 대한 움직임 벡터의 크기가 작을 개연성이 높기 때문에 초기 탐색 위치를 (0,0)으로 하고, <그림 3(b)>의 SDS(Small Diamond Search)를 적용한다. 그리고 움직임 벡터 활성도가 중간이라면, 초기 탐색 위치는 (0,0)으로 하고 <그림 3(a)>의 LDS (Large Diamond Search)를 적용한다. 마지막으로 움직임 벡터 활성도가 높다면 주변 블록의 움직임 벡터들이 지시하는 위치에서 SAD (Sum of Absolute Difference)를 구하고, 이중 가장 작은 SAD를 갖는 움직임 벡터를 찾아낸다. 그리고 이 곳을 초기 탐색 위치로 하여 SDS를 수행한다. 이때 움직임 벡터 탐색법은 기존의 DS와 같다. 움직임 활성도는 PMV(Predicted Motion Vector)로 쓰이는 주변 블록의 움직임 벡터를 이용했기 때문에, 동일한 매크로블록내



(a) LDSP (9 points) (b) SDSP (4 points)

그림 3. MVFAST에 대한 탐색 패턴
Fig. 3. Two search patterns for MVFAST.

의 하부 블록들일지라도 그 크기에 따라 움직임 활성도가 다를 수 있다. 즉 블록 크기마다 움직임 벡터 탐색 방법과 초기 탐색 위치가 다르게 적용될 수 있는 것이다. LDS와 SDS의 탐색 패턴은 <그림 3>과 같다.

LDS를 이용하여 움직임 벡터를 탐색할 경우, <그림 3(a)>의 9개 위치 각각에 대해 SAD를 구하고, 이중 가장 작은 SAD를 갖는 위치를 탐색 중심으로 하여, 다시 LDS를 적용한다. 이때 최소 SAD가 중앙에 위치한 화소에서 발견되었다면, LDS를 중지하고, 이 위치에서 다시 SDS를 수행한다. 마찬가지로 <그림 3(b)>의 SDS 또한 4개의 위치에서 SAD를 구하고, 최소 SAD를 갖는 점을 다시 탐색의 중심으로 하여 최소 SAD가 중앙에 위치한 화소에서 발견될 때까지 SDS를 반복적으로 수행한다. H.264에서, 16×16부터 4×4까지 움직임 벡터를 위한 SAD 계산은 16×16 블록 크기로만 움직임 벡터 탐색을 하는 H.263의 베이스라인에 비해 7배의 SAD 계산이 필요하다. 즉 {1×(MB16×16) + 2×(MB16×8) + 2×(MB8×16) + 4×(SUB8×8) = 7×(MB16×16)}인 것이다[5]. 따라서 하나의 매크로블록을 여러 크기의 블록으로 움직임 벡터 탐색을 수행하게 되면, 특정 위치에서 SAD 계산이 중복되어 수행된다. 또한 4×4 블록에 대한 16번의 SAD 계산과정은 16×16 블록의 SAD를 1번 계산하는 것과 계산량은 같을지라도 메모리 접근 회수가 16배 많기 때문에 데이터 전송에 추가적인 시간이 소요된다. 따라서 SAD를 계산할 때, 이미 탐색된 위치의 SAD를 저장하여, 다음에 다시 동일한 위치에서 SAD를 재사용 한다면, 속도 향상과 작은 블록의 움직임 벡터를 탐색할 때, 반복적인 메모리 접근 또한 피할 수 있을 것이다. 움직임 보상의 가장 작은 블록 크기가 4×4이기 때문에, SAD를 4×4단위로 저장한다. 예를 들어 설명하면, (0, 0)에서의 16×16 블록의 SAD를 16개의 4×4 단위의 SAD로 저장하여, 16×16 블록보다 더 작은 블록들이 이 4×4 단위의 SAD를 다시 이용할 수 있도록 하는 것이다. 만약 해당되는 위치에 SAD가 저장되어 있다면 이 값을 이용하고, 그렇지 않다면, SAD를 추가적으로 계산하여 다음 블록의 SAD 계산에서 이 값들을 이용할 수 있도록 한다.

현재 'JVT(Joint Video Team)'에서 제공하고 있는 H.264 부호화기인 'JM 6.1d'에서는 RD 최적화 기법을 이용하여 움직임 벡터를 결정하고 있다. 움직임 벡터를 부호화하는데 필요한 비트수와 이때의 SAD를 이용하여 움직임 벡터들에 대한 비트율-왜곡치인 RDcost를 계산하고 이 중 최소가 되는 것을 선택한다. 이에 대한 구체적인 식 (2)와 같다.

$$J(mv_c, \lambda_M) = SAD_{M\times N}(B_c, B_{REF}(m)) + \lambda_M R(mv_c - pmv) \quad (2)$$

식 (2)에서 $J(mv_c, \lambda_M)$는 RDcost를 나타내고, B_c는 현재 블록, B_{REF}는 참조영상에서 움직임 벡터가 가르치는 곳의 블록을 나타낸다. 그리고 λ_M은 Lagrangian 계수이다. 마지막으로 $R(mv_c - pmv)$는 PMV와 현재 블록의 움직임 벡터의 차이 값을 부호화하는데 필요한 비트수이다. 이런 RDcost는 탐색되는 모든 위치마다 수행된다.

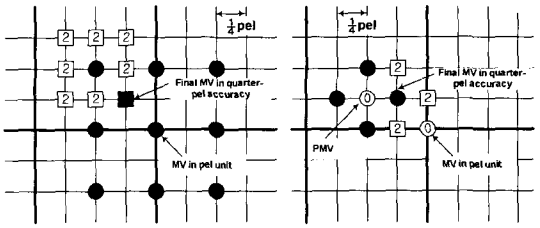


그림 4. 부 화소 위치 움직임 추정 (a) 기존의 2단계 탐색법 (b) SDS를 기초로한 제안한 탐색법

Fig. 4. Sub-pel motion search. (a) Conventional 2-step search algorithm (b) Proposed method based on SDS

H.264에서 움직임 벡터는 1/4 화소 정확도로 추정되어진다. <그림 4(a)>는 H.264에서 쓰이고 있는 부 화소 위치에서의 움직임 벡터 탐색법을 나타내고 있다. 정수 화소에서는 SAD를 이용하여 움직임 벡터를 찾았는데, 부 화소 위치에서의 움직임 벡터 추정은 SATD를 이용한다. SATD는 현재 블록의 화소 값과 참조 영상의 화소 값에 대한 차이를 4×4단위로 Hadamard 변환 후에 각 계수에 대한 절대값을 합한 것이다. 부 화소 위치에서 SAD대신에 SATD를 사용하는 이유는 최종적으로 비트를 생성하게 되는 잉여 데이터는 정수 위치가 아닌, 부 화소 위치에서의 잉여 값이기 때문에, 가장 간단한 변환 부호화를 취하여 가장 작은 비트를 생성하는 위치를 찾아내는 것이다. 실제 이러한 Hadamard 변환을 통해 약 6%의 비트 감소가 있음이 알려졌다[4]. 기존의 부 화소 위치에서의 움직임 벡터는 TSS(Three Step Search)[12]를 이용하여 추정된다. 정수 화소에서 찾은 움직임 벡터를 중심으로 8개의 1/2 부 화소 위치 중 가장 작은 SATD를 갖는 위치를 찾는

다. 그리고 1/2 부 화소 움직임 벡터를 중심으로 다시 주위 8개 1/4 화소 위치에서 SATD를 계산한 후, 그중 가장 작은 값을 갖는 위치를 선택한다. 이 TSS는 항상 17번의 SATD계산이 수행된다. 따라서 한 매크로블록당 $17\times\{16(MB16\times16)+8\times2(MB16\times8)+8\times2(MB8\times16)+4\times4\times4(SUB8\times8)\}=1094(Total)$번의 SATD가 수행되어 진다[5]. 또한 4×4 Hadamard 변환은 72번의 덧셈 연산과 64번의 뺄셈 연산이 필요하다. 이러한 계산량은 때때로 정수 화소에서 움직임 벡터 추정을 위한 계산량보다도 크다. 이러한 이유로 본 논문에서는 소수 위치에서 움직임 벡터 탐색에 대해 좀 더 효과적인 방법을 다음과 같이 제안한다.

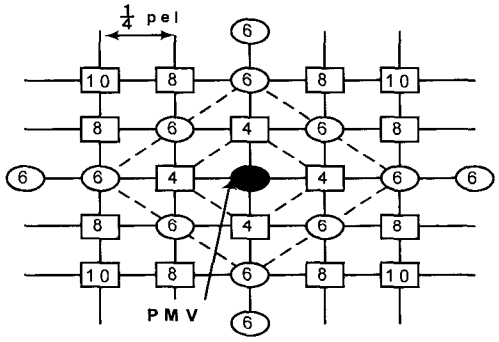


그림 5. DMV의 부호어 길이와 DMV에 대한 EG 부호 (a) DMV의 부호어 길이 (b) DMV에 대한 EG 부호

Fig. 5. Codeword length of DMV and EG code for DMV. (a) Codeword length for DMV (b) EG code for DMV

<그림 5(a)>는 DMV (Differential Motion Vector)를 부호화하는데 필요한 비트수를 각 화소 위치마다 나타낸 것이다. DMV는 PMV(Predictive Motion Vector)와 현 블록의 움직임 벡터에 차이 값을 나타낸다. <그림 5(b)>는 DMV를 부호화하기 위한 EG(Exponential Golomb) 부호[14] 이다. 만약 EG 부호가 DMV를 부호화하는데 최적이라면, 부호어의 길이로부터 DMV가 발생할 확률을 계산 할 수 있을 것이다. 일반적으로, 기하학적인 확률 모델과 잘 부합하는 EG 부호는 DMV를 부호화하는데 효율적이라는 것이 밝혀졌다[14]. 그리고 DMV에 대한 EG 부호의 효율이 95%이라는 것이 실험을 통하여 알려졌다[15, 16]. 따라서 EG 부호어 길이가 l인 DMV는 발생 확률이 2^{-l}에 매우 가깝다. 그리고 부 화소 위치에서 움직임 벡터를 탐색할 때, SDS로 움직임 벡터를 찾을 확률은 약 75%가 될 것이다. 그럼으로 부 화소에서 움직임 벡터에 대한 검색에 SDS를 사용한다면, 부호 효율에 대한 성능 저하는 최소로 하면서, 작은 연산량만으로도 고속 움직임 벡터 탐색이 가능할 것이다. 이에 대해 제안하는 구체적인 방법은 다음과 같다.

먼저 정수 화소 위치에서 움직임 벡터를 찾는다. 이 정수 화소 위치의 움직임 벡터를 중심으로 PMV가 부 화소 단위로 ±3 범위 내에 존재하는지 확인한다. 만약 ±3 범위 내에 PMV가 존재한다면, 정수 화소 움직임 벡터에서의 SAD 값과 PMV에서의 SAD값을 계산한다. 그리고 두 위치의 SAD 중에서 값이 작은 화소를 중심으로 SDS를 수행하며, 이를 통해 최종적으로 1/4 화소 단위 움직임 벡터를 찾는다. 이때 PMV와 현재 움직임 벡터의 상관도는 매우 높기 때문에 ±3범위 내에 PMV가 존재할 확률이 매우 높다. 만일 PMV가 정수 화소 움직임 벡터를 기준으로 ±3범위 내에 존재하지 않는다면, 정수 화소 움직임 벡터를 중심으로 SDS를 적용한다.

Ⅲ. 고속 모드 결정법

H.264의 모드 결정을 위한 RD 최적화 기법은 움직임 벡터의 선택이나 참조 영상의 선택보다 좀더 중요하고 복잡하다. 매크로블록 모드로는 SKIP, MB16×16, MB16×8, MB8×16, SUB8×8, INT4×4, INT16×16으로 총 7 가지가 있다. 여기서 SKIP은 움직임 벡터가 (0, 0)이거나 PMV와 같고, 정수 변환 후의 변환 계수가 모두 0인 16×16 블록으로 정의된다. 그리고 INT4×4와 INT16×16은 둘 다 공간 예측모드를 나타내는데, INT4×4는 4×4 블록 단위로 방향에 따른 9가지 모드가 있으며, INT16×16은 16×16 블록 단위로 수평, 수직, DC, 평면 (Plane) 예측 모드가 있다. 7가지 매크로블록 모드 중 최적의 모드를 결정하기 위하여 RD 최적화 기법(Rate-Distortion 이 사용되며[1], 이때 각 모드별로 비트량과 화질 열하를 고려하여 비트율-왜곡치인 RDcost를 계산하여 가장 작은 RDcost에 대한 모드를 선택하게 된다. 따라서 각 매크로블록 모드에 대한 RDcost를 구하기 위해서는 각 모드마다 정수 변환 부호화, 양자화, Entropy 부호화가 수행되어야 한다. Lagrangian 계수를 이용한 RDcost 계산식은 다음과 같다.

$$J_{Mode} = D + \lambda_{Mode} R \quad (3)$$

식 (3)에서 D는 현재 영상과 참조 영상사이의 SAD 이며, λ_{Mode}는 Lagrangian 상수로서 $\lambda_{Mode} = 0.85 \times 2^{Qp/3}$과 같고 Qp는 양자화 매개 변수이다. 마지막으로 R은 주어진 모드로 부호화해서 생성되는 비트량이다. 위에서 살펴보았듯이 RDcost를 구하는 과정은 매우 복잡하며 많은 계산량을 필요로 한다. 따라서, 조기에 효과적으로 모드 결정을 할 수 있다면, 불필요한 움직임 벡터 탐색과 RDcost 계산 과정을 생략할 수 있어, 부호화기의 복잡도와 계산량을 줄 일 수 있을 것이다. 본고에서는 이를 위해 Inter 매크로블록 모드 결정법과, 공간 예측 부호화 과정을 생략할 수 있는 방법을 제시한다.

첫째로, Inter 매크로블록 모드를 초기에 효과적으로 결정하기 위하여, Inter 매크로블록 모드를 Class16={SKIP, MB16×16, MB16×8, MB8×16}과 Class8={SUB8×8}로 분류하였다. 이렇게 분류한 이유는 SUB8×8 모드는 4개의 8×8 블록이 독립적으로 4개의 모드 중 하나로 움직임 보상되는데, 이 때의 모드 결정도 8×8 단위로 RDcost를 계산하여 하부 모드를 결정함으로, 결과적으로 SUB8×8 전체의 RDcost 계산량이 Class16 전체에 대한 계산량과 같기 때문이다. 본고에서 제안한 Inter 모드 결정법은 초기에 MB16×16의 RDcost16과 MB8x8의 RDcost8을 구하여, 최소 값을 갖는 Class에서 현 매크로블록의 모드가 결정되도록 하는 방법이다. 즉 RDcost16이 RDcost8보다 작다면, 현 매크로블록의 최적 모드는 Class16에서 결정되도록 하며, 반대인 경우에는 Class8에서 최적의 매크로블록 모드가 결정되도록 하는 것이다. 그러나 이런 흑백 논리적인 모드 결정법은 잘못된 모드로 귀착될 위험성이 크기 때문에, 좀더 효과적인 Class 분류 기준이 필요하다. 이를 위해 최소의 계산으로 얻을 수 있도록 측정치 r 을 식 (4)와 같이 정의하였다. <그림 6>은 r 과 모드 결정 에러에 대한 그래프이다.

$$r = \frac{RDcost8 - RDcost16}{RDcost16} \times 100 \quad (4)$$

<그림 6>에서 양수 r 에 해당되는 그래프는, Class16에 최적 모드가 있다고 결정했지만, 실제 최적 모드는 Class8에 있는 경우이다. 반대로 음수 r 에 해당되는 그래프는 최적 모드가 Class8에 있다고 결정했

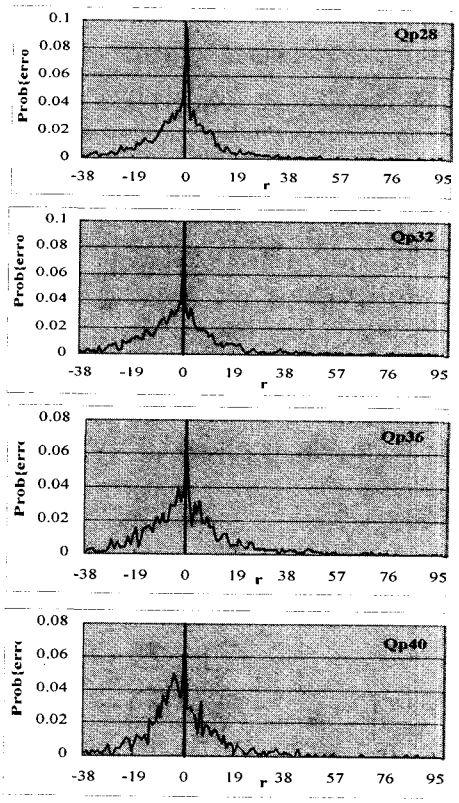


그림 6. Qp에 대한 RDcost와 모드 결정 오류에 대한 확률 분포(영상 : foreman)

Fig. 6. Probability distribution of decision error with RDcost with regard to quantization parameter(foreman sequence).

는데, 실제 최적 모드는 Class16에 있는 경우이다. <그림 6>에서 r 이 0에 가까워질수록 모드 결정 에러의 확률이 높아짐을 관찰 할 수 있다. 따라서 에러를 최소화하기 위해서는 에러가 많은 부분에 대해서는 기존의 방법대로 부호화하고, 상대적으로 에러가 적은 부분에 대해서는 Class16 또는 Class8로 부호화하는 것이다. 즉 r 이 'th_low'보다 작을 경우 Class8 내에서만 모드를 결정하고, r 이 'th_high' 보다 클 경우에는 Class16 내에서만 모드를 결정한다. 그 외의 th_low ≤ r < th_high에 해당되는 경우에는 Class를 구분하지 않고 기존 방법대로 모드를 결정한다. 'th_low'와 'th_high'에 대한 식은 다음과 같다.

$$th\_low = mean - \frac{std}{2},\ th\_high = mean + \frac{std}{2} \quad (5)$$

식 (5)에서 mean과 std는 주어진 Qp에 따른 r 의 평균값과 표준편차를 의미한다. Chebyshev's inequa-

lity[17]에 따르면 평균값을 중심으로 하여 표준편차 값에 해당되는 범위 안에 정규 분포 Gaussian 확률 곡선을 갖는 랜덤 변수가 포함될 확률이 0.7에 가깝다. 그런데 <그림 6>의 에러에 대한 확률 분포 곡선은 Gaussian인 경우 보다 평균값에 많은 값들이 밀집되어 있음으로 제안한 모드 결정법을 사용하더라도 실제 모드로 결정할 확률이 70% 이상 된다. 즉 식 (4)를 식 (5)의 경계치와 비교하여 최적 모드 결정을 위해 RDcost 값을 계산해 보아야 하는 경우 수를 상당수 줄여 부호화 속도를 높일 수 있는 것이다.

두 번째로 공간 예측 부호화 과정을 생략하는 제안 방법은 다음과 같다. 먼저 최적의 Inter 매크로블록 모드를 위의 방법대로 결정한 후에, 이때의 RDcost를 'th_intra' 값과 비교하여, RDcost가 더 작다면 공간 예측 부호화 과정을 생략한다. 'th_intra' 값은 최적의 Inter 매크로블록 모드의 RDcost 분포와 최적의 공간 예측 모드의 RDcost 분포를 구하여, 이 둘의 평균값들을 다시 평균한 것이다. <표 1>은 고속 모드 결정을 위해 필요한 문턱치들을 나타낸 것이다. 이 값들은 foreman 영상을 통하여 실험적으로 얻은 것인데, foreman 영상은 다양하고 복잡한 움직임을 갖고 있기 때문에, foreman 영상을 통해 얻어진 문턱치들은 다른 영상에도 비교적 잘 적용될 것이다.

표 1. Qp값에 따른 고속 모드 결정법의 문턱치
Table 1. Thresholds for fast mode decision with regard to quantization parameter values.

Threshold \ Qp	28	32	36	40
th_low	-8.28	-8.6	-7.92	-8.39
th_high	11.66	12.17	10.74	6.18
th_intra	8643	15463	25089	32209

<그림 7>은 고속 모드 결정법에 대한 흐름도를 나타낸 것이다. 여기서 ClassINT는 공간 예측 모드의 Class로서 INT4×4와 INT16×16이 있다. 그리고 RDcost Class8, RDcostClaa16, RDcostINT는 각각 Class8, Class16 그리고 ClassINT의 모드 별 RDcost 중에서 가장 작은 값을 의미하는 것으로, 이 값을 최대 값으로 설정하면 마지막 최적 모드 결정 단계에서 절대로 해당 Class에서 최적 모드가 결정되지 않게 된다. 또한 *RDcostInter*는 인터 모드의 RDcost 중 가장 작은 값으로, RDcostClass8 또는 RDcostClass16 중 하나와 같은 값을 갖게 된다.

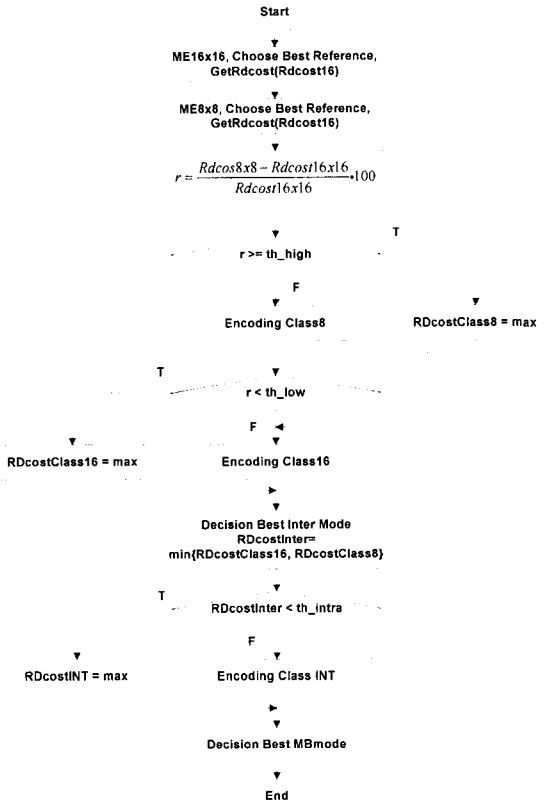


그림 7. 제안한 고속 모드 결정법에 대한 흐름도
Fig. 7. Flow chart of proposed fast mode decision method.

Ⅳ. 실험 결과

실험 조건은 H.264 표준화 그룹에서 권고하고 있는 공통 실험 조건[18]을 따라 <표 2>와 같이 하였다. 실험에 사용된 부호화기는 JM6.1d이며, 실험은 B 슬라이스와 CABAC을 사용하지 않는 H.264의 베이스라인 프로파일을 사용하였다.

<표 3>은 고속 움직임 벡터 탐색에 대한 결과이다. 성능 비교를 위하여 전역 탐색을 기준으로 하여 DS, MVFAST, 그리고 제안한 고속 움직임 벡터 탐색을 비교하였다. 이때 부호화 성능을 평가하기 위한 객관적 지표로, BDBR(Bjonteggard Delta BitRate)과 BDPSNR (Bjonteggard Delta PSNR)[19]을 사용하였는데, 이들은 각각 비교하고자 하는 두 방법 간의 비트율과 PSNR 차이의 평균을 의미한다. 그리고 BDBR의 (+)부호와

표 2. 실험 조건
Table 2. Simulation condition.

	news (qcif)	container (qcif)	foreman (qcif)	silent (qcif)	paris (cif)	mobile (cif)	tempete (cif)
Frame rate(Hz)	10	10	10	15	15	30	30
Total frames	100	100	100	150	150	300	300
Qp	28, 32, 36, 40						
Coding options used	RD optimization, Hadamard transform, IPPP structure, CAVLC, no error tools						
Codec	JM 6.1d encoder						

표 3. 제안한 방법과 DS, MVFAST와의 성능 비교
Table 3. Performance comparison of proposed fast ME with DS, MVFAST.

(a) 탐색 범위 = ±16
(a) Search range = ±16

	DS				MVFAST				Proposed FME			
	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}
container	0.33	-0.02	81.6	1.0	0.87	-0.04	171.8	1.0	0.73	-0.04	901.0	3.1
foreman	14.56	-0.76	48.9	1.0	6.03	-0.33	101.5	1.0	6.71	-0.36	305.9	2.2
news	0.55	0.03	76.6	1.0	1.94	-0.10	157.0	1.0	0.87	-0.05	665.6	2.8
mobile	0.68	-0.03	65.8	1.0	0.66	-0.03	146.8	1.0	1.20	-0.05	733.3	2.2
silent	2.17	-0.12	73.1	1.0	2.25	-0.12	144.1	1.0	2.30	-0.12	570.0	2.6
paris	2.88	-0.15	74.8	1.0	2.78	-0.14	155.4	1.0	3.01	-0.15	678.7	2.8
tempete	0.65	-0.03	70.2	1.0	0.87	-0.04	144.0	1.0	0.61	-0.03	648.4	2.2
average	3.11	-0.16	70.1	1.0	2.20	-0.11	145.8	1.0	2.20	-0.11	646.1	2.6

(b) 탐색 범위 = ±32
(b) Search range = ±32

	DS				MVFAST				Proposed FME			
	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}
container	0.98	-0.05	316.5	1.0	1.52	-0.08	666.7	1.0	1.37	-0.07	3495.8	3.1
foreman	-4.50	-0.75	189.9	1.0	5.98	-0.32	398.7	1.0	6.66	-0.36	1186.8	2.2
news	2.03	-0.11	297.1	1.0	3.43	-0.18	609.0	1.0	2.36	-0.13	2660.1	2.8
mobile	0.75	-0.03	255.2	1.0	0.73	-0.03	568.5	1.0	1.27	-0.06	2844.9	2.2
silent	1.97	-0.11	283.5	1.0	2.05	-0.11	559.0	1.0	2.10	-0.11	2211.6	2.6
paris	3.49	-0.18	290.3	1.0	3.39	-0.17	602.9	1.0	3.39	-0.17	2633.0	2.8
tempete	0.92	-0.04	272.4	1.0	1.14	-0.05	558.7	1.0	0.89	-0.04	2515.5	2.2
average	3.52	-0.18	272.1	1.0	2.61	-0.14	565.6	1.0	2.58	-0.13	2506.8	2.6

BDPSNR의 (−)부호는 부호화 성능의 손실을 나타낸다. 식 (6)을 통해 얻어진 S_{SAD}와 S_{SATD}는 전역 탐색에 비해 SAD와 SATD의 줄어든 계산 수를 비율로 나타낸 것으로, 값이 클수록 계산 수가 많이 줄어든 것을 의미한다.

$$S_{SAD}=\frac{\#\,SAD_{4\times4}\text{ for full search}}{\#\,SAD_{4\times4}\text{ in given method}},\quad S_{SATD}=\frac{\#\,SA(T)D_{4\times4}\text{ for full search}}{\#\,SA(T)D_{4\times4}\text{ in given method}} \tag{6}$$

검색 범위가 ±16인 경우, 정수 화소 움직임 벡터 탐색에 대해 DS 방법이 약 3%의 비트 손실이 있었지만, 전역 검색에 비해 SAD 계산 수가 70배 줄어들었다. 그리고 MVFAST 방법은 비트 손실이 2% 발생되었지만, DS 보다 계산 수가 2배 더 줄어든 것을 알 수 있다. 그러나 MVFAST의 손실이 특정 영상에서 DS보다도 많은데, 이것은 MVFAST 방법에 쓰인 문턱치 L1, L2가 움직임이 큰 영상에 비교적 잘 맞기 때문이다. 제안한 정수 화소 움직임 벡터 탐색법은 MVFAST의 문턱치 L1, L2를 동일하게 이용하였다. 제안한 방법은 전역 검색에 비해 SAD 계산수가 646배 줄었으며, 검색 범위가 ±32인 경우에는 PSNR이 0.11dB 감소하였지만, SAD 계산 수는 무려 2056배나 줄어들었다. 결과적으로 제안한 고속 움직임 벡터 탐색법은 MVFAST에 비해 동일한 부호 효율에서 SAD계산수가 약 4.5배 정도 줄어들었으며, 추가적으로 SATD에 대한 계산량 또한 반으로 줄어들었다.

표 4. 제안한 고속 모드 결정법에 대한 결과
Table 4. Experimental results of proposed fast mode decision(FMD).

	FMD					FME with FMD (SR=±16)			
	BDBR [%]	BDPSNR [dB]	R_{inter} [%]	R_{intra} [%]	R_{total} [%]	BDBR [%]	BDPSNR [dB]	S_{SAD}	S_{SATD}
container	3.73	-0.19	28.14	67.32	50.58	5.06	-0.25	932.1	5.2
foreman	3.53	-0.19	27.77	59.69	46.06	10.72	-0.57	355.0	3.6
news	2.86	-0.17	29.49	71.07	53.31	4.42	-0.26	787.3	4.9
mobile	2.56	-0.11	20.35	14.65	17.07	3.71	-0.16	785.1	3.1
silent	2.55	-0.13	26.55	69.04	50.86	4.32	-0.21	656.9	4.4
paris	3.44	-0.17	25.84	79.26	64.83	6.54	-0.33	769.8	4.6
tempete	2.42	-0.10	25.07	38.16	32.61	3.62	-0.15	708.8	3.4
average	3.01	-0.15	26.17	57.03	45.05	5.49	-0.27	713.6	4.2

<표 4>는 제안한 고속 모드 결정법에 대한 결과이다. R_{inter}, R_{intra}, R_{total} 은 줄어든 RDcost 계산 수에 대한 백분율을 나타낸 것이다. 이에 대한 수식은 다음과 같다.

$$R_{mode} = \frac{\#RDcost_{mode}\{REF\} - \#RDcost_{mode}\{proposed\}}{\#RDcost_{mode}\{REF\}} \times 100\ [\%] \quad (7)$$

식 (7)에서 REF는 기존의 모드 결정법을 사용하고, 움직임 벡터를 전역 검색으로 찾는 부호화기이다. <표 4>에서 감소한 Inter 모드의 RDcost 계산 수는 약 26%, 공간 예측 모드인 경우 57%의 계산량 감소가 있었다. 그래서 전체 줄어든 RDcost 계산 수는 약 45%이고, 반면 BDBR의 손실은 약 3%정도다. 그런데 mobile과 tempete 영상의 RDcost 계산 수 감소 비율은 다른 영상에 비해 비교적 낮은데 그 이유는 생략되는 프레임 없이 부호화되었기 때문에, 프레임간 유사도가 매우 높은 까닭이다. 이런 이유로 측정치 r 이 th_low와 th_high의 사이에 놓이게 되는 경우가 많아져, 결과적으로 제안한 고속 모드 결정법의 적용 빈도수가 낮아진 것이다. 또한 tempete 영상은 빠르고 회전하는 움직임을 갖는 나뭇잎이 많이 존재하고, mobile 영상은 회전, 상하로 반복되는 움직임, 카메라 회전과 같은 복잡한 움직임이 많기 때문에, 이 두 영상에 대해서는 제안한 공간 예측 모드에 대한 생략법이 적용될 가능성이 낮은 것이다. 그리고 <표 4>에는, 추가적으로 제안한 고속 움직임 벡터 탐색과 고속 모드 결정법을 결합한 결과도 나타내었다. 그 결과 SAD 계산 수의 감소는 전역 검색에 비해 713배나 되었으며, SATD에 대한 계산 수 감소도 4.2배나 되었지만, BDBR의 손실은 약 5.5%가 되었다.

V. 결 론

본 논문에서는 정수 화소에서 MVFAST를 수정한 형태의 고속움직임 벡터 탐색과 새로운 1/4 부 화소에서의 고속 움직임 벡터 탐색법을 제안하였다. 그리고 모드 결정 초기 단계에서 효율적으로 모드를 결정하여 RDcost를 구하는 과정을 생략 할 수 있는 고속 모드 결정법도 제안하였다. 복잡한 RD 최적화 기법과 가변 블록 움직임 보상 기법은, H.264 표준에서 부호화기 전체의 복잡도에 많은 부분을 차지하고 있기 때문에 제안한 고속 움직임 베터 탐색과 모드 결정법은 부호화기의 복잡도를 낮출 뿐만 아니라 고속 부호화기의 구현을 가능하게 한다. 실험 결과 고속 움직임 벡터 탐색법은 MVFAST에 비해 SAD 계산 수가 4.5배, SATD는 2.6배 감소하였으며, 고속 모드 결정법의 사용으로 줄어든 RDcost 계산 수는 45%가 되었다. 그러나 제안한 고속 움직임 벡터 탐색은 많은 메모리의 사용을 필요로 하기 때문에, 한정된 메모리를 갖는 응용에서는 특정 범위내의 SAD 값만을 저장하여, 메모리 사용량을 줄여야 할 것이다. 그러나 SAD 재사용의 빈도수가 줄어들게 되어, 움직임 벡터 탐색 시간이 다소 늘어나게 될 것이다. 그럼으로 메모리 용량과 움직임 벡터 탐색 시간을 잘 고려하여 본 고에서 제시한 고속 움직임 벡터 탐색을 사용해야 할 것이다. 그리고 고속 모드 결정을 위해 제안한 측정치 r 은 매우 간단한 방법으로 얻어 졌기 때문에 이에 따른 계산량 증가는 매우 미미하지만, 모드 결정에 필요한 상당한 복잡도를 줄일 수 있는 장점이 있다. 그러나 모드 결정에 대한 정확도를 더욱 높이기 위해, 계산량이 허락한다면 좀 더 정밀한 새로운 측정치 r 을 찾아내는 것도 본 연구 결과의 확장으로 생각 할 수 있다.

참 고 문 헌

[1] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," IEEE Signal Processing Magazine, Vol.15, pp. 74~90, Nov. 1998.

[2] P. Topiwala, G. Sullivan, A. Joch, F. Kossentini, "Overview and Performance Evaluation of the Draft ITU-T H.26L Video Coding Standard," Proc. SPIE, Appl. Dig. Im. Proc, Aug. 2001.

[3] ITU-T SG16 Q6, "Performance Evaluation of H.26L, TML 8 vs. H.263++ and MPEG-4," VCEG-N18, Sep. 2001.

[4] M. Zhou, "Evaluation and Simplification of H.26L Baseline Coding Tools," JVT-B030, Jan, 2002.

[5] W. Choi and B. Jeon, "Selective fast motion estimation with variable motion block sizes," International Workshop on Advanced Image Technology (IWAIT' 03), pp. 101~106, Nagasaki, 21-22, Jan. 2003.

[6] Joint Video Team (JVT) of ISO/IEC MPEG &

ITU-T VCEG, "Study of Final Committee Draft of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," Doc. JVT-F100, Dec. 2002.

[7] P. Kuhn and W. Stechele, "Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation," Visual Communications and Image Processing (VCIP' 98), vol. SPIE 3309, pp. 498~509, San Jose, Jan. 1998.

[8] P. Hosur and K. Ma, "Motion vector field adaptive fast motion estimation," Second International Conference on Information, Communications and Signal Processing (ICICS '99), Singapore, 7-10, Dec. 1999.

[9] A. Tourapis, O. Au, and M. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, pp. 934~947, Oct. 2000.

[10] K. Ma, and P. Hosur, "Performance report of Motion Vector Field Adaptive Search Technique(MVFAST)," in ISO/IEC JTC1/SC29/WG11 MPEG99/m5851, Noordwijkerhout, NL, Mar, 2000.

[11] S. Zhu and K. Ma, "A new diamond search algorithm for fast block matching," IEEE Transactions on Image Processing, Vol. 9, No. 2, pp. 287~290. Feb. 2000.

[12] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, "Motion-compensated interframe coding for video conferencing," Proc. NTC'81 (IEEE), p G.5.3.1 - G.5.3.4, 1981.

[13] J. Teuhola, "A Compression method for clustered bit-vectors," Information Processing Letters, vol.7, pp. 308~311, Oct. 1978.

[14] Y. Itoh, "Bi-directional motion vector coding using universal vlc," Signal processing: Image communication 14, pp. 541~557, 1999.

[15] B. Jeon, "Entropy coding efficiency of H.26L," ITU-T Q.6/16, Doc. #Q15-J57, May, 2000.

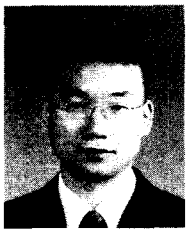
[16] 최웅일, 전병우, 유국렬, 천강욱, "고정 재배정 테이블 기반 동적 UVLC 부호화 방법," 전자공학회 논문지 제 39권 SP편 제2호, pp. 56~68, 2002, 3월

[17] R. Ziemer, Element of engineering probability and statistics, Prentice Hall, NJ, 1997.

[18] G. Sullivan and G. Bjontegaard, "Recommended simulation common conditions for H.26L coding efficiency experiments on low-resolution progressive-scan source material," ITU-T Q.6/16, Doc. #VCEG-N81, Sep. 2001.

[19] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ITU-T Q.6/16, Doc. #VCEG-M33, Mar. 2001.

저 자 소 개



李 帝 琉(學生會員)

2002년 2월 : 성균관대학교 전자공학과 졸업(공학사). 2002년 3월~현재 : 성균관대학교 정보통신공학부 석사과정 <주관심분야 : 영상압축, 멀티미디어 응용>

崔 雄 一(學生會員)

2000년 2월 : 성균관대학교 정보통신공학부 졸업(공학사). 2002년 2월 : 성균관대학교 정보통신공학부 졸업(공학석사). 2002년 3월~현재 : 성균관대학교 정보통신공학부 박사과정. <주관심분야 : 영상압축, 멀티미디어 응용>

全 炳 宇(正會員)

1985년 2월 : 서울대학교 전자공학과 졸업(공학사). 1987년 2월 : 서울대학교 전자 공학과 졸업(공학석사). 1992년 12월 : Purdue Univ, School of Elec. 졸업(공학박사). 1993년~1997년 8월 : 삼성전자 신호처리연구소 수석연구원. 1997년 9월~현재 : 성균관대학교 정보통신공학부 부교수. <주관심분야 : 멀티미디어, 영상압축, 영상인식, 신호처리>

石 玟 秀(正會員)

1969년 : 서울대학교 공과대학 전자공학과(수료). 1970년 : University of California, Davis(학사) 1972년 : University of California, Davis(석사). 1974년 : University of California, Davis(박사). 1979년~1982년 : 한국과학기술원 교수. 1983년~1994년 : Syracuse University 교수. 2001년~현재 : 성균관대학교 교수. <주관심분야 : 패턴인식, 신호처리 및 통신>