論文2003-40SP-5-10

가변 크기 블록에서 정수단위 화소 움직임 벡터의 빠른 검색

(A Fast Motion Vector Search in Integer Pixel Unit for Variable Blocks Siz)

李 戎 基 *, 李 英 烈 **

(Yung-Gi Lee and Yung-Lyul Lee)

요 약

본 논문에서 정수단위 화소(integer pixel unit)로 움직임 예측(motion estimation)을 수행하는 빠른 움직임 예측(fast motion estimation) 알고리즘을 제안한다. 제안하는 방법은, sum norm을 사용하여 가장 좋은 움직임 벡터를 찾아내는 연속 제거 기법(SEA : Successive Elimination Algorithm)을 기반으로 16×16블록에서는 전체 영역에 대해 검색을 하고 16×8, 8×16, 8×8블록에서는 16×16블록의 움직임 벡터로부터 그 주위 8개의 위치에서 가장 좋은 벡터를 구하고, 8×4, 4×8, 4×4블록은 8×8블록의 움직임 벡터로부터 그 주위 8개의 위치에서 벡터를 검색하여 그 중에서 가장 좋은 움직임 벡터를 찾아내는 것이다. 이러한 움직임 검색(motion search) 방법을 가변 크기 블록(16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)으로 움직임 예측을 하는 H.264 부호기(encoder)에 적용하였다. 제안하는 검색 알고리즘을 계산 복잡도 측면에서 보면, 조기 종료가 적용 안 된 나선형으로 전체 영역을 검색(Spiral full search without early termination)하는 방법보다 23.8배가 빨라졌고, 4×4 블록들의 계층적 SAD(Sum of Absolute Difference)를 이용하는 빠른 움직임 예측 방식보다 4.6배의 속도증가를 보인다. 반면에 신호 대 잡음 비(PSNR : Peak Signal to Noise Ratio)는 0.1dB에서 0.4dB정도 떨어짐을 보인다.

Abstract

In this paper, a fast motion search algorithm that performs motion search for variable blocks in integer pixel unit is proposed. The proposed method is based on the successive elimination algorithm (SEA) using sum norms to find the best estimate of motion vector and obtains the best estimate of the motion vectors of blocks, including 16×8, 8×16, and 8×8, by searching eight pixels around the best motion vector of 16×16 block obtained from all candidates. And the motion vectors of blocks, including 8×4, 4×8, and 4×4, is obtained by searching eight pixels around the best motion vector of 8×8 block. The proposed motion search is applied to the H.264 encoder that performs variable blocks motion estimation (ME). In terms of computational complexity, the proposed search algorithm for motion estimation (ME) calculates motion vectors in about 23.8 times speed compared with the spiral full search without early termination and 4.6 times speed compared with the motion estimation method using hierarchical sum of absolute difference (SAD) of 4×4 blocks, while it shows 0.1dB~0.4dB peak signal-to-noise ratio (PSNR) drop in comparison to the spiral full search.

Keyword : SEA, motion estimation, spiral full search, fast full search, software

* 學生會員, 世宗大學校 컴퓨터工學科
(Department of Computer Engineering Sejong University)
* 正會員, 世宗大學校 인터넷工學科
(Department of Internet Engineering Sejong University)

※ 마지막으로 한국학술진흥재단(KRF - 2002 - 003 - D00339)의 후원에 감사를 표한다.
接受日字:2003年8月27日, 수정완료일:2003年9月15日

Ⅰ. 서 론

ITU-T H.263[1], MPEG-4[2]와 H.264[3] 등의 대부분의 영상 압축 표준은 블록기반 움직임 예측 및 움직임 보상(motion compensation)을 한다. 움직임 예측은 동영상 부호화에서 시간적 중복성(temporal redundancy)을 이용하여 비트-율(bit-rate)을 줄이는데 큰 역할을 하지만, 상당히 많은 연산이 필요하다. 특히, H.264(MPEG에서는 MPEG-4 AVC(Advanced Video Coding)이라 명명됨)는 가변 크기 블록기반 움직임 예측 및 보상을 하는 부분에서 많은 연산이 소요된다.

움직임 예측을 위한 여러 가지 제안되었던 알고리즘들은 부호기에서 연산 시간을 줄임으로 효과를 나타내었다. 움직임 검색의 속도를 향상시킨 것으로 잘 알려진 알고리즘으로는 Three-Step Search[4]와 2-D logarithm Search[5], OTS(One-At-a-Time Search)[6], New Diamond Search[7, 8], FES(Fast and Efficient Search)[9] 등이 있다. 이러한 방법은 검색하는 위치의 수를 줄임으로써 전체 영역 검색의 계산량의 대략 3%~5%정도만 사용하여 정수단위 화소의 움직임 벡터를 찾아낸다. 그 결과, 계산량의 감소로 압축시간의 향상은 있으나, 이렇게 찾는 방법은 국부적 최소점(local minimum)에 빠지기 쉬울 뿐 아니라, PSNR도 전체 영역 검색을 했을 때만큼 좋은 결과를 나타내지 못한다. 반면에 검색하는 위치를 줄이지 않고 모든 검색 영역을 검색하면서 계산량을 감소하여 속도를 높인 연속제거 기법(SEA)[10]은 전체 검색 영역에서 검색이 필요없는 부분은 계속해서 제거해나가기 때문에 전체 영역 탐색을 했을 때와 같은 성능을 보이면서 계산량은 전체 영역 검색의 13%정도만 소요하게 된다. 그리고 ESEA(Early termination SEA)[11]는 SEA를 향상시킨 것으로 최저 경계 SAD를 둠으로써 그 이상이 되는 SAD는 계산을 종료하여, PSNR은 그대로 유지하고 계산량은 줄였다.

본 논문에서 H.264 JM(Joint Model) 원시 부호(source code)[14]에 있는 조기 종료가 적용 안 된 나선형 전체 영역 검색(Spiral full searchwithout early termination), 조기 종료가 적용된 나선형 전체 영역 검색(Spiral full searchwith early termination), 빠른 전체 영역 검색(Fast full search with early termination) 방식을 제안한 알고리즘과 비교하였다. 조기 종료가 적용 안 된 나선형 전체 영역 검색과 조기 종료가 적용된 나선형 전체 영역 검색은 공통적으로 주위의 블록을 이용하여 Median-Predictor로 얻은 움직임 벡터를 중심으로 나선형 순서로 16의 영역으로 움직임 예측을 하는데, 조기 종료가 적용된 것은 현재 검색점(search point)의 계산중인 SAD가 이전에 구한 최소 SAD보다 더 크면 더 이상 현재 검색점에 대한 SAD계산을 하지 않는 조기 종료의 조건을 포함한다. 빠른 전체 영역 검색은 하나의 매크로블록에서 ±16영역으로 16개의 4×4 블록의 SAD계산을 수행하고, 그 SAD값을 이용하여 계층적으로 더 큰 블록들의 SAD값을 만들어서 움직임 예측을 한다. 이렇게 계층적으로 SAD값을 만들어 나가기 때문에 가변 크기 블록의 움직임 벡터 검색 시 많은 연산을 줄여 속도 향상을 가져왔다.

본 논문은 가변 크기 블록으로 움직임 예측을 하는 H.264에 SEA의 개념을 적용하여 정수단위 화소의 움직임 벡터를 빠르게 검색하는 소프트웨어 기반에 유리한 알고리즘을 제안한다. 제안하는 기법은 H.264의 율-왜곡 최적화(rate-distortion optimization)에 계층적 움직임 예측 방법을 적용한다.

다음 장에서는 제안하는 기법을 자세히 살펴보고, Ⅲ장에서는 실험 결과를 바탕으로 제안하는 기법을 평가한다. 마지막으로 Ⅳ장에서는 본 논문의 결론을 기술한다.

Ⅱ. 정수단위 화소에서의 움직임 검색 기법

1. 정수단위 화소에서의 움직임 검색

H.264는 하나의 16×16매크로블록을 16×16, 16×8, 8×16, 8×8로 나누고 다시 8×8을 8×4, 4×8, 4×4의 블록 모양으로 나누어 가변 크기 블록으로 움직임 예측을 수행한다. 이러한 가변 크기 블록을 이용한 계층적 움직임 예측은 각 블록들의 SAD와 움직임 벡터 비트 수에 따라 최적의 움직임 벡터를 구한다. 본 논문에서는 빠른 움직임 검색을 위해 먼저 참조 프레임의 sum norm을 계산한다.

현재의 시간 t에서의 프레임 (i,j)위치의 화소값을 현재 프레임 $f(i,j,t)$라고 하고, t-1에서의 프레임 (i,j)위치의 화소 값을 참조 프레임 $f(i,j,t-1)$이라 하며, 프레임 크기는 H×W이고, 가변 크기 블록(S×T : 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)으로 움직임 예측을 한다고 가정하자. 그러면 SEA에서 사용하였던 부등식[10, 13]으

로부터 움직임 벡터가 (x, y)인 다음 부등식을 유도 할 수 있다.

$$|\sum_{i=1}^{S}\sum_{j=1}^{T}|f(i-x,j-y,t-1)|-\sum_{i=1}^{S}\sum_{j=1}^{T}|f(i,j,t)||\le\sum_{i=1}^{S}\sum_{j=1}^{T}|$$

$$f(i,j,t)-f(i-x,j-y,t-1)| \quad (1)$$

식 (1) 좌변의 첫 번째 합은 움직임 벡터 (x,y)를 가진 참조 프레임에서 S×T 후보블록의 sum norm(이후 $SN_{S\times T}(x,y)$로 표기)이고, 두 번째 합은 현재 프레임의 S×T 블록의 sum norm(이후 $R_{S\times T}$로 표기)이다. 우변의 합은 현재 S×T 블록과 움직임 예측된 S×T 블록의 SAD(이후 $SAD_{S\times T}(x,y)$로 표기)이다. 그러면 식 (1)을 다음과 같이 나타낼 수 있다.

$$SN_{S\times T}(x,y)-R_{S\times T}\le SAD_{S\times T}(x,y)$$
$$R_{S\times T}-SN_{S\times T}(x,y)\le SAD_{S\times T}(x,y) \quad (2)$$

그리고 초기 움직임 벡터(predicted motion vector) (m,n)에서 $SAD_{S\times T}(m,n)$를 가지고 있을 때, 더 좋은 움직임 벡터 (x,y)가 있다면 다음 식 (3)을 만족한다.

$$SAD_{S\times T}(x,y)\le SAD_{S\times T}(m,n) \quad (3)$$

그러므로 식 (2)와 식 (3)으로 다음 식 (4)을 유추할 수 있다.

$$SN_{S\times T}(x,y)-R_{S\times T}\le SAD_{S\times T}(m,n)$$
$$R_{S\times T}-SN_{S\times T}(x,y)\le SAD_{S\times T}(m,n) \quad (4)$$

또한 이 식들을 하나의 부등식으로 나타내면 다음 식 (5)과 같이 표현 할 수 있다.

$$R_{S\times T}-SAD_{S\times T}(m,n)\le SN_{S\times T}(x,y)\le R_{S\times T}$$
$$+SAD_{S\times T}(m,n) \quad (5)$$

식 (5)가 검색 영역 내의 모든 후보 벡터 중에 임의의 움직임 벡터(x,y)에 만족한다면, 움직임 벡터 (x,y)에서의 SAD인 $SAD_{S\times T}(x,y)$가 계산되고 계산된 $SAD_{S\times T}(x,y)$가 $SAD_{S\times T}(m,n)$보다 작다면 $SAD_{S\times T}(m,n)$의 값은 $SAD_{S\times T}(x,y)$으로 대체된다. 그러므로 검색 영역 ±M(본 논문에서는 ±16)에서의 S×T 블록의 최적의 벡터를 구하기 위한 검색은 식 (5)를 만족하는 sum norm을 가진 블록에 대해서만 수행된다. 식 (5)를 만족하는 블록의 수가 많지 않기 때문에 검색 영역 내의 일부의 움직임 벡터 후보점 만을 계산함으로서 계산량 감소를 얻는다. 이 방법의 효과는 얼마나 빠르게 sum norm을 계산하는가, 그리고 초기 움직임 예측 값, 즉 $SAD_{S\times T}(m,n)$가 얼마나 전역 최소점(global minimum)에 가까운가에 따라 좌우될 수 있다. 현재 블록의 초기 움직임 예측 방식으로 Median-Predictor에 의해 예측되는 움직임 벡터를 사용한다. Median-Predictor란, <그림 1>에서 처럼 현재 블록 E에서, 상단의 블록 B와 우측 상단의 블록 C와 좌측의 블록 A의 움직임 벡터를 각각 MV_B, MV_C, MV_A로 나타내었을 때 다음과 같은 식 (6)에 의해 계산된다.

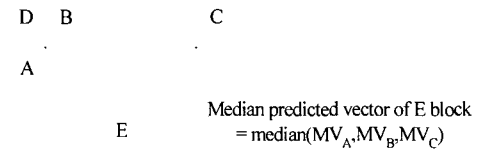


그림 1. 현재블록의 움직임 벡터 예측(E는 현재 블록, A, B, C, D는 주위의 블록)

Fig. 1. The neighboring blocks for motion vector prediction of current block. E is current block and A, B, C, and D are neighboring blocks.

$$Median\ PMV = median(MV_A, MV_B, MV_C) \quad (6)$$

식 (6)에서 PMV는 현재 E블록의 Median-Predictor에 의해 예측된 움직임 벡터이다.

그러나, 식 (5)에 의해서 계산 복잡도를 줄이긴 했으나, 하나의 매크로블록에서 작은 블록들만 선택되었을 경우, 그 작은 블록에 의해 많은 움직임 벡터가 발생하고 이로 인해 비트-율은 증가하게 된다. 특히 움직임 벡터가 하나의 매크로블록에 대해 모두 4×4 블록들이 선택된다면 움직임 벡터는 16개가 되고 16개의 벡터를 복호기에 보내주어야 된다. 따라서 움직임 벡터의 비용(이후 $MVCost(m,n)$이라고 표기)도 포함하여 식 (5)를 수정하였다.

$$R_{S\times T}-(SAD_{S\times T}(m,n)+MVCost(m,n))\le SN_{S\times T}(x,y)$$

$$+ MVCost(x,y) \le R_{S\times T} + SAD_{S\times T}(m,n) + MVCost(m,n)$$

$$MVCost(m,n) = \lambda \times MVbits(PMV - MV)$$

$$where\ \lambda = \text{int}[2^{\lfloor QP-12 \rfloor / 6} + 0.5] \text{ for } QP \ge 12,$$

$$\lambda = 1\ for\ QP < 12 \qquad (7)$$

식 (7)에서 PMV는 Median-Predictor에 의해 예측된 움직임 벡터이고, MV는 현재 블록의 움직임 벡터이다. $MVbits(PMV-MV)$는 UVLC (Universal Variable Length Coding)된 비트 수를 나타내고, QP는 H.264[13, 14]에서의 양자화 값(Quantization Parameter)이다. 그러므로 최적의 움직임 벡터를 구하기 위하여 sum norm과 움직임 벡터의 비용($MVCost$)을 포함한 식 (7)은 본 논문에서 제안한 계층적 SEA(Hierarchical SEA)이다.

2. 가변 크기 블록 sum norm의 계산

sum norm의 계산은 우선 가장 작은 블록(4×4)에 대하여 계산된다. <그림 2>에서 보여진 것과 같이 4의 길이로 하나의 열(column)을 형성하는데 첫 번째 줄을 $C(1,1)$이라 하고 이것은 참조 프레임의 $f(1,1)$, $f(2,1)$, $f(3,1)$, $f(4,1)$의 4개의 화소 값이 더해져서 만들어진다. 이러한 4의 길이를 가진 각 행의 열의 합, 즉 $C(i,j)$, i=1,2,...,H-3, j=1,2 ,...,W,는 다음과 같이 계산된다.

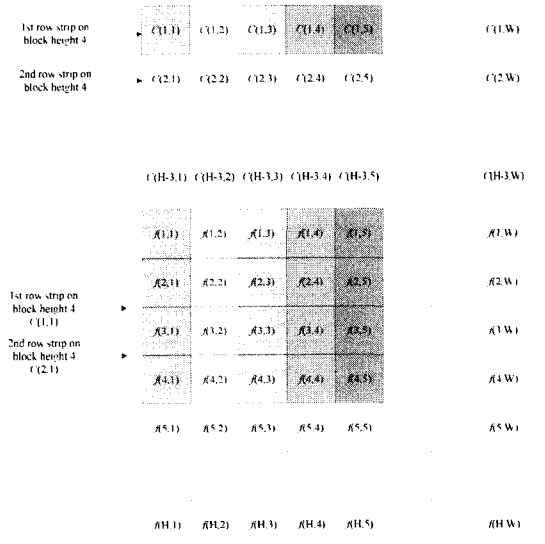


그림 2. 가장 작은 블록(4×4)의 sum norm 계산
Fig. 2. The sum norm calculation of the smallest block (S=T=4) in which T is the block height of each row strip.

$$\begin{cases} C(1,1) = f(1,1)+f(2,1)+f(3,1)+f(4,1) \\ C(1,2) = f(1,2)+f(2,2)+f(3,2)+f(4,2) \\ \cdots \\ C(1,W) = f(1,W)+f(2,W)+f(3,W)+f(4,W) \end{cases}$$

$$\begin{cases} C(2,1) = C(1,1)-f(1,1)+f(5,1) \\ C(2,2) = C(1,2)-f(1,2)+f(5,2) \\ \cdots \\ C(2,W) = C(1,W)-f(1,W)+f(5,W) \end{cases}$$

...

$$\begin{cases} C(H-3,1) = C(H-4,1)-f(H-4,1)+f(H,1) \\ C(H-3,2) = C(H-4,2)-f(H-4,2)+f(H,2) \\ \cdots \\ C(H-3,W) = C(H-4,W)-f(H-4,W)+f(H,W) \end{cases}$$

그리고 $C(i,j)$로 부터 $SN_{4\times4}(x,y)$, $x=1,2,...,H-3$, $y=1,2,...,W-3$,가 유도된다.

$$\begin{cases} SN_{4\times4}(1,1) = C(1,1)+C(1,2)+C(1,3)+C(1,4) \\ SN_{4\times4}(1,2) = SN_{4\times4}(1,1)-C(1,1)+C(1,5) \\ \cdots \\ SN_{4\times4}(1,W-3) = SN_{4\times4}(1,W-4)-C(1,W-4)+C(1,W) \end{cases}$$

$$\begin{cases} SN_{4\times4}(2,1) = C(2,1)+C(2,2)+C(2,3)+C(2,4) \\ SN_{4\times4}(2,2) = SN_{4\times4}(2,1)-C(2,1)+C(2,5) \\ \cdots \\ SN_{4\times4}(2,W-3) = SN_{4\times4}(2,W-4)-C(2,W-4)+C(2,W) \end{cases}$$

...

$$\begin{cases} SN_{4\times4}(H-3,1) = C(H-3,1)+C(H-3,2)+C(H-3,3)+C(H-3,4) \\ SN_{4\times4}(H-3,2) = SN_{4\times4}(H-3,1)-C(H-3,1)+C(H-3,5) \\ \cdots \\ SN_{4\times4}(H-3,W-3) = SN_{4\times4}(H-3,W-4)-C(H-3,W-4)+C(H-3,W \end{cases}$$

$SN_{4\times8}(i,j)$와 $SN_{8\times4}(i,j)$는 $SN_{4\times4}(i,j)$를 더하면서 (*e.g.* $SN_{1\times8}(1,1) = SN_{4\times4}(1,1) + SN_{4\times4}(5,1)$)만들 수 있고, $SN_{8x8}(i,j)$도 같은 방법으로 $SN_{4\times8}(i,j)$나 $SN_{8\times4}(i,j)$로부터 만들 수 있다. $SN_{8\times16}(i,j)$과 $SN_{16\times8}(i,j)$은 $SN_{8\times8}(i,j)$로 만들 수 있으며, $SN_{16\times16}(i,j)$은 $SN_{8\times16}(i,j)$이나 $SN_{16\times8}(i,j)$로 만들 수 있다. 이처럼 각 각의 가변 크기 블록(16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)sum norm들은 기존에 만들어놓은 작은 블록들의 sum norm들을 더하여 움직임

예측 이전에 미리 구성할 수 있다. 위의 각 가변 크기 블록에 대한 sum norm 계산이 제안된 알고리즘의 부하(overhead) 계산량이다.

3. 가변 크기 블록에서의 정수단위 화소 움직임 예측

<그림 3>에 가변 크기 블록의 검색 순서를 나타내었다. 움직임 예측의 순서는 16×16을 시작으로 4×4로 진행하면서 각각의 블록별로 움직임 예측을 한다. 먼저 일곱 개의 sum norm 테이블, 즉 $SN_{4\times4}(i,j)$, $SN_{4\times8}(i,j)$, $SN_{8\times4}(i,j)$, $SN_{8\times8}(i,j)$, $SN_{8\times16}(i,j)$, $SN_{16\times8}(i,j)$, $SN_{16\times16}(i,j)$를 구한 후, 식 (7)의 제안된 조건을 이용하여 움직임 예측을 수행하게 된다. 식 (7)은 전체영역 탐색 범위(±16) 내에서 불필요한 후보 위치들은 제거하면서도, 전체 영역 탐색에서와 같은 결과의 움직임 벡터를 찾아내 준다. <그림 4>에서 정수단위 화소의 움직임 예측을 나타내었는데, 제안된 방식은 16×16에서 찾은 움직임 벡터에 대하여 16×8, 8×16, 8×8 블록에 대하여 ±16 움직임 영역에 대하여 움직임 예측을 하고, 8×8블록들에 대해 찾은 움직임 벡터에 대하여 8×4, 4×8, 4×4 블록들에 대해 ±16 움직임 영역에 대하여 움직임 예측을 한다. 이 방법을 적용하면 나선형 전체 영역 검색(Spiral full search)과 같은 율-왜곡 곡선을 얻을 수 있다. 좀 더 계산량을 줄이기 위하여 다음과 같은 제한된 움직임 영역을 갖는 방식을 사용하였다. <그림 4(a)>는 현재 움직임 예측을 하려는 16×16 매크로블록이고, <그림 4(b)>는 참조 프레임에서 현재 매크로블록 <그림 4(a)>에 대한 16×16 크기 블록의 최적 움직임 벡터 (l,m)을 나타내었다. <그림 4(c), (d), (e)>는 16×8, 8×16, 8×8 크기의 블록에서의 움직임 예측인데, 이는 앞서 찾은 16×16 크기 블록의 움직임 벡터(l,m)과 주위 8개의 화소(8-neighbor pixels)만을 검색한다. 그 중에서 16×8블록의 움직임벡터 검색을 보면, <그림 4(c)>와 같이 우선 위쪽 16×8

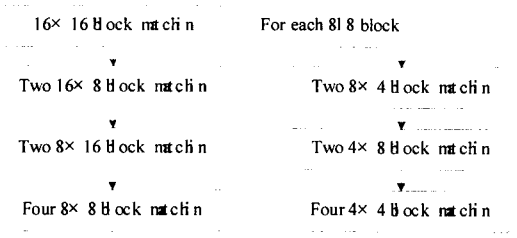


그림 3. 가변 크기 블록의 예측 순서
Fig. 3. The order for the variable blocks matching.

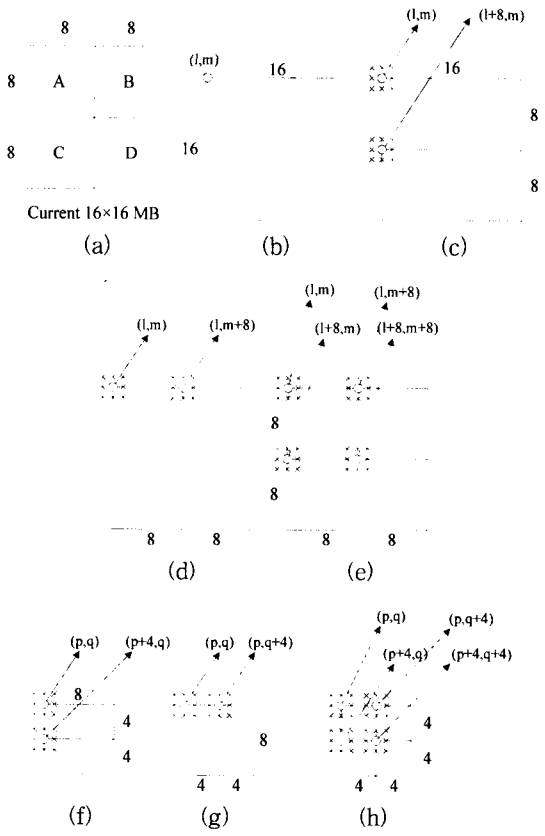


그림 4. 움직임 예측 시 검색 영역 (a) 현재 프레임에서 16×16매크로블록 (b) 16×16 블록의 참조 프레임에서 움직임 벡터 위치 (c) 두 개의 16×8 블록의 검색 위치 (d) 두 개의 8×16 블록의 검색 위치 (e) 블록 A, B, C, D 각 8×8 블록의 검색 위치 (f), (g), (h) 8×4, 4×8, 4×4 블록의 검색 위치

Fig. 4. The search positions for each variable block matching: (a) A current 16×16 MB in a current frame, (b) 16×16 matching block position in the reference frame of the current 16×16 MB, (c) search positions in the reference frame of the current two 16×8 blocks in the current MB, (d) search positions in the reference frame of the current two 8×16 blocks in the current MB, (e) search positions in the reference frame of the current A,B,C, and D 8×8 blocks in the current MB, (f) search positions in the reference frame of the current two 8×4 blocks in A,B,C, or D block of the current MB, (g), (h) are similarly analogized.

(<그림 4(a)>에서 A+B)은 16×16의 움직임벡터 (l,m)이 초기 벡터가 되고 인접한 8개의 화소를 식 (7)의 부

등식에 따라 최적 움직임 벡터를 찾아내고, 아래쪽 16×8(<그림 4(a)>에서 C+D)은 앞에 것과 유사하게 $(l+8,m)$ 과 그 주위의 8개의 화소만이 검색 범위가 된다. 8×4, 4×8, 4×4의 움직임 예측은 <그림 4(f)>에서(h)까지 나타내었는데, 이 역시 앞의 방법과 유사하게 먼저 8×8 블록 A내의 것은 A블록의 움직임벡터인 (p,q)를 사용하게 되며, (p,q)는 $l-\leq p \leq l+1$, $m-1 \leq q \leq m+1$을 만족하게 된다. <그림 4(f)>에 나타난 것처럼, A의 위쪽 8×4 블록의 움직임 검색은 16×8 블록의 검색과 비슷하게 (p,q)와 그 주위 8개 화소만, A의 아래쪽 8×4블록의 움직임 탐색은 $(p+4,q)$와 그 주위 8개 화소만을 대상으로 하게 된다. B, C, D의 8×4 탐색 역시 각각이 속한 8×8의 (p,q)를 참조하여, 유사하게 진행된다. 나머지 4×8과 4×4도 8×4와 같은 방법으로 움직임 예측을 하여 움직임 벡터를 구해낸다. 여기서 조금 더 속도를 높이기 위해 SAD 계산을 조기 종료하는 개념을 식 (7)에 적용할 수 있다. $SAD_{S\times T}(x,y)$를 계산하는 동안 지금까지 구해진 최소 SAD인 $SAD_{S\times T}(m,n)$과 비교해서, 더 크다면 계산을 멈추고, 그렇지 않다면, 식 (7)의 계산을 계속하게 된다. 이로 인해 약간의 계산량을 줄일 수 있게 된다.

Ⅲ. 실험 결과

우리가 실험에 사용한 H.264의 기본 조건으로 Exp-Golomb Code와 가변 크기 블록(16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)의 움직임 예측 및 보상, ±16 의 검색 영역, 4×4 정수 DCT(Discrete Cosine Transform), 율-왜곡 최적화를 사용했다. 300프레임을 갖는 각 시퀀스들에 대하여 실험을 했으며, 첫 번째 프레임만 Intra 프레임이고 나머지는 모두 Predictive 프레임으로 압축하였다. 제안된 방법들은 JM(Joint Model)4.2 원시 부호(source code)[14]에 구현되어 있는 조기 종료가 적용 안 된 나선형 전체 영역 탐색(Spiral full search without early termination)과 조기 종료가 적용된 나선형 전체 영역 탐색(Spiral full search with early termination) 그리고 빠른 전체 영역 탐색(Fast full searchwith early termination)의 PSNR과 움직임 예측에만 소요되는 계산 시간을 비교하였다. JM 원시 부호에서 빠른 전체 영역 탐색은 먼저 4×4 블록에 대해 모든 SAD를 구한 뒤, 계층적으로 큰 블록의 SAD를 구하는 방법을 사용하고 앞서 말한 율-왜곡 최적화

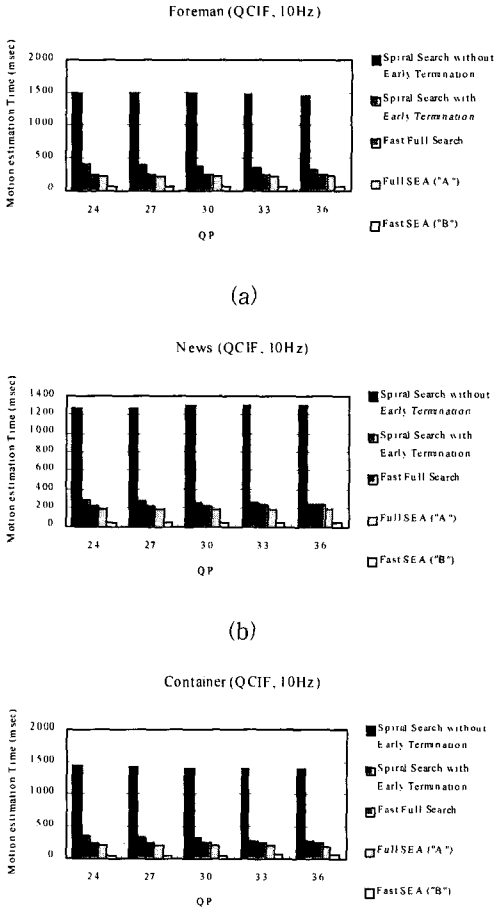
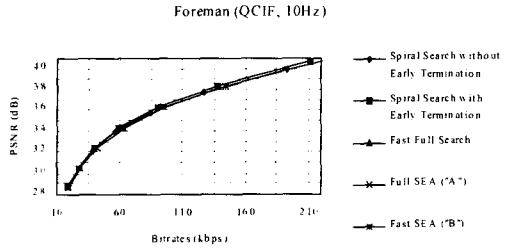
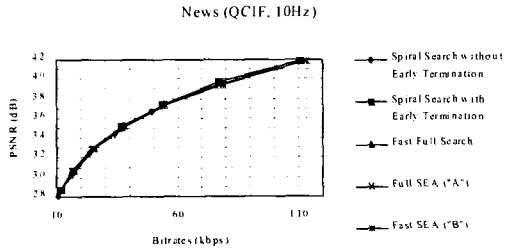


그림 5. 조기 종료가 적용 안 된 나선형 전체 영역 검색과 조기 종료가 적용된 나선형 전체 영역검색, 빠른 전체 영역 검색, "A", "B" 다섯 가지 방법의 계산량 비교 (a) 프레임율 10Hz, QCIF 크기의 "Foreman" (b) 프레임율 10Hz, QCIF 크기의 "News" (c) 프레임율 10Hz, QCIF 크기의 "Container"

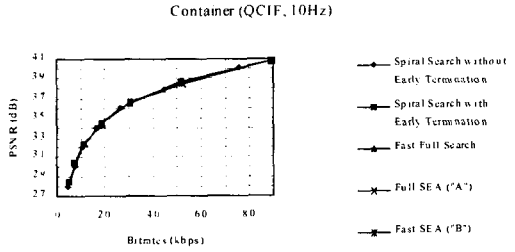
Fig. 5. The computation time comparison among the spiral full search without early termination, the spiral full search with early termination, the fast full search, the Hierarchical SEA method using ±16 search ranges for all variable blocks denoted by "A", Hierarchical SEA method with limited search range denoted by "B". Only 1st frame is Intra-coded: (a) "Foreman" sequence with a frame rate of 10Hz and QCIF resolution, (b) "News" sequence with a frame rate of 10Hz and QCIF resolution, (c) "Container" sequence with a frame rate of 10Hz and QCIF resolution.



(a)



(b)

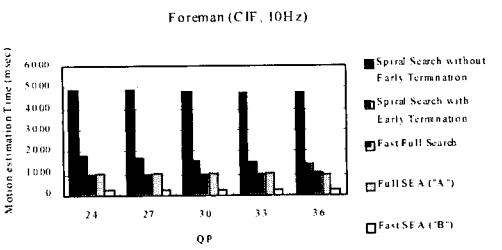


(c)

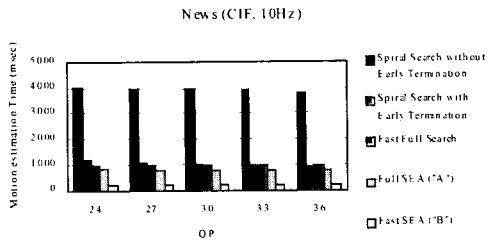
그림 6. 율-왜곡 곡선 : (a) "Foreman" (b) "News" (c) "Container"

Fig. 6. Rate-distortion plots: (a) "Foreman" rate-distortion plot corresponding to Fig. 5(a), (b) "News" rate-distortion plot corresponding to Fig. 5(b), (c) "Container" rate-distortion plot corresponding to Fig. 5(c).

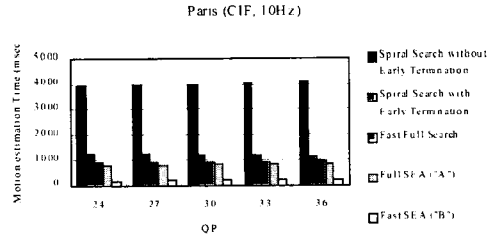
기술[12, 14] 또한 공정한 비교를 위해 사용하였다. 모든 실험은 펜티엄 IV-1.7GHz에서 H.264[14]를 이용해 QCIF와 CIF영상에 대해 행해졌다. 세 가지의 QCIF영상에 대한 다섯 가지 방법의 움직임 예측 계산 시간은 <그림 5(a), (b), (c)>에서 보여지고, 가로축은 H.264의 양자화값을 나타낸다. <그림 5(a), (b)>와 <그림 5(c)>의 그래프에서 제안된 "A"방법은 모든 가변 크기 블록이 16의 검색 영역을 가지는 계층적(Hierarchical) SEA이며 나선형 전체 영역 검색과 같은 PSNR을 보여준다. "A"방법은 계층적 SEA를 사용했을 때와 같은 PSNR을 보여주는 반면에 <그림 4>에서 보인 16×16 미만의 가변 크기 블록에 대하여 제한된 검색 영역을 갖는 제안된 "B"방법에 비해 많은 계산량을 보인다. <그림 5(a), (b)>와 <그림 5(c)>는 QCIF의 "Foreman"과 "News", 그리고 "Container" 영상일 경우, "A"방법이 사용되면 조기 종료가 적용 안 된 나선형 전체 영역 탐색보다 대략 6.9배 빠르고 빠른 전체 영역 탐색과는 비슷하지만 대략 1.2배 빠른 것을 알 수 있고, "B"방법이 사용되면 조기 종료가 적용 안 된 나선형 전체 영역 탐색보다 대략 27.7배 빠르고 빠른 전체 영역 탐색보다 대략 4.8배 빠르다는 것을 보여준다. <그림 6(a), (b), (c)>는 "Foreman"과 "News", 그리고 "Container" QCIF 영상에 대해 10Hz의 프레임율에서의 다양한 비트-율에 대한 율-왜곡 곡선을 보였다. 율-왜곡 곡선에서 "B"는 나선형 전체 영역 탐색이나 빠른 전체 영역 탐색에 비해 다소 떨어지는 결과를 보인다. <그림 7>, <그림 8(a), (b), (c)>에서는 각각 10Hz의 프레임율을 가지는 CIF 영상인 "Foreman"과 "News", "Paris" 영상에 대한 다섯 가지 방법의 움직임 예측 계산 시간과 율-왜곡 곡선을 비교한다. 그림에서와 같이 CIF 영상에서도 제안하는 방법은 QICF영상의 결과와 비슷한 결과를 보였다. 실험 결과를 전체적으로 살펴 보면, 제안된 "A"방법은 프레임 크기에는 상관없이 움직임이 적은 영상(News, Container, Paris)에서 나선형 전체 영역 탐색과 거의 비슷한 PSNR을 보이면서 움직임 예측 계산 시간은 최고 7.15배의 속도 향상이 있고, 움직임이 큰 영상(Foreman)에서는 PSNR의 측면에서는 거의 비슷하나 움직임 예측 계산 시간의 감소는 보이지 못하였다. 검색 영역을 1로 줄인 제안된 "B"방법은 프레임 크기에는 상관없이 대체로 움직임이 작은 영상(News, Container, Paris)에서 나선형 전체 영역 탐색보다 PSNR이 최대 0.2dB의 저하를 보였고, 움직임 예측 계산 시간은 최대 28.35배의 향상이 있었다. 움직임이 큰 영상(Foreman)에서는 나선형 전체 영역 탐색에 비해 최대0.4dB의 PSNR저하가 있었고, 움직임 예측 계산 시간은 최대 26.85배의 속도 향상을 보였다. 그래서 제안된 "A", "B" 모두 대체로 움직임이 적은 영상에서 더 좋은 결과를 나타내었다. 결과적으로 영상의 화질이 계산시간보다 더 중요한 경우는 PSNR감소나 비트-율의 증가가 없는 "A"방법을 이용하여 빠른 움직임 벡터를 구할 수 있다.



(a)



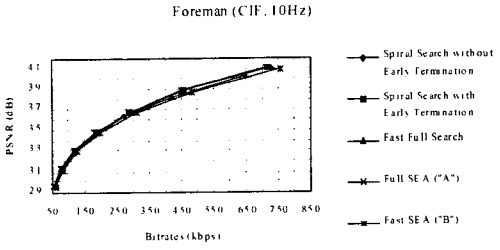
(b)



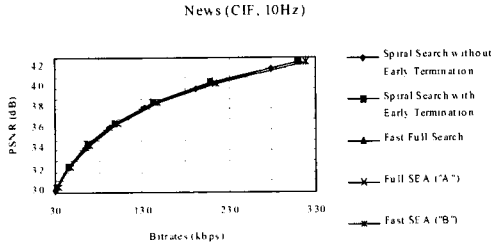
(c)

그림 7. 조기 종료가 적용 안 된 나선형 전체 영역 검색과 조기 종료가 적용된 나선형 전체 영역검색, 빠른 전체 영역 검색, "A", "B" 다섯 가지 방법의 계산량 비교 (a) 프레임율 10Hz, CIF 크기의 "Foreman" (b) 프레임율 10Hz, CIF 크기의 "News" (c) 프레임율 10Hz, CIF 크기의 "Paris"

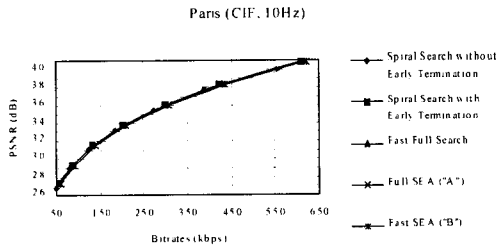
Fig. 7. The computation time comparison among the spiral full search without early termination, the spiral full search with early termination, the fast full search, the Hierarchical SEA method using ±16 search ranges for all variable blocks denoted by "A", Hierarchical SEA method with limited search range denoted by "B". Only 1st frame is Intra-coded: (a) "Foreman" sequence with a frame rate of 10Hz and CIF resolution, (b) "News" sequence with a frame rate of 10Hz and CIF resolution, (c) "Paris" sequence with a frame rate of 10Hz and CIF resolution.



(a)



(b)



(c)

그림 8. 왜곡 곡선 : (a) "Foreman" (b) "News" (c) "Paris"

Fig. 8. Rate-distortion plots: (a) "Foreman" rate-distortion plot corresponding to Fig. 7(a), (b) "News" rate-distortion plot corresponding to Fig. 7(b), (c) "Paris" rate-distortion plot corresponding to Fig. 7(c).

Ⅳ. 결 론

제안된 방법은 가변 크기 블록의 움직임 검색을 위해 계층적 sum norm을 이용한 움직임 예측에 심각한 영상의 화질 저하나 비트-율의 증가 없이 쉽게 적용 가능하다. 계층적으로 sum norm을 계산하는 방법은 전체 계산 복잡도를 감소시키는데 기여한다. H.264는 7가지 형태의 가변 크기 블록과 1/4 단위 화소 움직임 예측 및 보상을 채택하였다. 제안된 제한된 움직임 영역

검색을 하는 계층적 SEA("B")는 검색 과정에서 아주 작은 PSNR의 손실은 있지만 계산 복잡도 감소를 줄일 수 있기 때문에 H.264의 움직임 예측 방식으로 적용시킬 수 있다. 그리고 제안하는 방법은 심각한 화질 저하 없이 다중 참조 프레임에서도 사용될 수 있고 소프트웨어 기반에서 효과적이다. 만약 영상의 화질이 속도보다 중요한 경우는 모든 가변 크기 블록들에 대하여 전체 영역을 검색하는 방식 "A"가 사용되면 효과적일 수 있다.

참 고 문 헌

[1] ITU Telecom. Standardization Sector, "Video Codec Test Model Near-Term, Version 10 (TMN10) Draft 1," H.263 Ad Hoc Group, April 1998.

[2] "Information Technology - Coding of Audio-Visual Objects Part2: Visual Amendment 1: Visual Extensions", ISO/IEC JTC1/SC29/WG11 N3056, Dec. 1999.

[3] "Draft ITU-T Recommendation and Final Draft International Standard of JointVideo Specification(ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", Joint Vide Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Mar. 2003.

[4] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," Proc. Nat. Telecommunications Conf., pp. G.5.3.1-G.5.3.5., Nov. 1981.

[5] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. Commun., vol. 29, pp. 1799-1808, Dec. 1981.

[6] R. Srinivansan and K. Rao, "Predictive coding based on efficient motion estimation," Proc. IEEE IVV'84, pp. 521-526, 1984.

[7] S. Zhu and K-K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Trans. Image Processing, vol. 9, no. 2, pp. 287-290, Feb. 2000.

[8] 윤효순, 손남례, 이귀상, "A Modified Diamond Search Algorithm for Fast Block Matching Motion Estimation", 대한전자공학회 01 제14회 신호처리 합동 학술대회 논문집 2001,9 v.2001, pp. 393-396

[9] S. G. Kim, T. H. Lee, T. Y. Jung,and D.G. Kim, "Fast and Efficient Search Algorithm of Block Motion Estimation", 대한전자공학회 00 ITC-CSCC (2) 2000,7 v.2000,pp.885-888.

[10] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation," IEEE Trans. Image Processing, vol. 9, no. 1, pp.105-107, Jan. 1995.

[11] M. Brunig and W. Niehsen, "Fast Full-Search Block Matching," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 2, pp. 241-247, Feb. 2001.

[12] G. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," IEEE Signal Processing Magazine, pp. 74-90, Nov. 1998.

[13] D. M. Young and R. T. Gregory, A Survey of Numerical Mathematics, New York: Dover, vol. 2, pp. 759-762, 1988.

[14] http://ftp.imtc-files.org/jevt-experts/reference-software/jm42.zip

저 자 소 개

李 戎 基(學生會員)

2002년 2월 : 세종대학교 전산과학과 졸업. 2002년 3월~현재 : 세종대학교 컴퓨터공학과 석사과정. <주관심분야 : 영상처리, 멀티미디어 응용>

李 英 烈(正會員)

1985년 2월 : 서강대학교 전자공학과 졸업. 1987년 2월 : 서강대학교 전자공학과 석사. 1999년 2월 : 한국 과학 기술원 전기 및 전자공학과 박사. 2001년 8월 : 삼성전자 중앙연구소 DMS Lab. 수석연구원. 2001년 9월~현재 : 세종대학교 인터넷학과 조교수. <주관심분야 : 영상처리(압축,복원), 영상전송, 멀티미디어 시스템, Video Transcoding>