論文95-32B-3-9

움직임 벡터 분할을 이용한 움직임 보상 성능 개선

(Performance Improvement of Motion Compensation using Motion Vector Segmentation)

蔡 鍾 吉 *, 郭 成 一 *, 黃 燦 植 *

(Jong Kil Chae, Seong Il Kwak, and Chan Sik Hwang)

요 약

블럭 정합 알고리즘에서는 한 블럭내의 모든 물체는 동일 방향의 움직임을 갖는다고 가정함으로써 한 블럭내에 다른 움직임을 갖는 물체가 존재할 때 움직임 벡터는 단지 한 물체에만 정합되거나 최악의 경우에는 어는 물체에도 정합되지 않게 된다. 이는 하드웨어의 복잡성과 계산량을 줄이는 효과를 갖는 고속 블럭 정합 알고리즘에서는 두드러지게 나타난다. 작은 크기의 블럭을 사용한 움직임 벡터의 추정은 정확성을 지니지만 움직임 벡터를 나타내기 위한 비트의 수를 증가 시킨다. 본 논문에서는 기존의 방법에 비하여 보다 작은 부가정보를 가지면서 하드웨어를 단순화 시킬 수 있는 움직임 벡터 분할을 제안하였다. 제안된 방법에서 움직임 벡터 분할을 위한 블럭으로 부터 하나의 움직임 벡터와 이웃한 4 블럭으로 부터 움직임 벡터를 선택하여 움직임 벡터쌍을 형성한다. 그리고 각 부블럭에 대하여 보다 정확한 움직임 벡터를 정하기 위하여 평균 자승 오차에 따라 움직임 벡터를 지정한다. 또한, DFD를 줄이기 위해서 창을 이용한 중첩 움직임 보상을 하였다.

Abstract

It is assumed in the block matching algorithm(BMA) that all the pels in a block have a same motion vector. Then, the motion vector of a block in the BMA is matched to only one or none of the objects in the worst case if objects in a block have different motion vectors. This is apparent in the motion estimation using the fast BMA which has the effect of reducing the computation time and hardware complexity, compared to the full search BMA. Although the motion vector in the motion estimation using small block size is accurate, the increased number of bits is required to represent motion vectors. In this paper, new motion vector segmentation with less additional information and hardware complexity than the conventional method is proposed. In the proposed method, a motion vector is derived from the block for motion vector segmentation and another motion vector is extracted from four neighboring blocks to consitute a motion vector pair. For the accurate motion vector of each subblock, the motion vector is assigned to each subblock by mean squared error measure. And the overlapped motion compensation using window is also applied to reduce displaced frame difference.

* 正會員, 慶北大學校 電子工學科

(Dept. of electronics, Kyungpook Nat'l Univ.)

接受日字 : 1993年 12月 1日

Ⅰ. 서 론

동영상은 대체로 움직임 보상 부호화(motion compensated coding : MCC)를 이용하여 높은 데이타 압축을 얻는다. MCC에 의한 데이타의 압축은 영상의 움직임을 추정, 보상하는 과정과 원영상과 움직임 벡터(motion vector)에 의해서 보상된 영상과의 차분신호(displaced frame difference : DFD)를 부호화하는 과정으로 구성된다 [1]. MCC에서 움직임을 추정하는 방법으로는 크게 화소 순환 알고리즘(pel recursive algorithm : PRA) [2-4]과 블럭 정합 알고리즘(block matching algorithm : BMA) [5-17]이 있다. PRA는 화소 단위로 움직임 벡터을 추정하는 것으로 한 화소의 움직임 벡터는 다음 화소의 움직임 벡터를 추정하는데 이용된다. PRA에서는 비교적 물체의 정확한 움직임을 추정할 수 있으며 움직임 벡터가 정확하게 추정되지 못한 경우에도 이로 인한 영향은 하나의 화소내지는 단 몇개의 화소에만 나타나게 되므로 예측 오차를 줄일 수 있으나 시간이 너무 많이 소요되며 병렬처리가 어려운 단점이 있다. BMA는 PRA와 달리 영상을 임의의 작은 블럭으로 나눈 후 한 블럭내의 모든 화소는 동일 방향의 움직임을 갖는다고 가정하고서 움직임 벡터를 추정하는 것으로 이전 프레임에서 설정된 탐색영역에서 어떤 블럭이 현재 프레임의 정해진 블럭으로 이동하였는가 찾는 것이다. BMA는 PRA에 비하여 움직임 벡터 추정에 소요되는 시간이 작으며 병렬처리가 용이한 장점이 있어서 MCC에 많이 적용되고 있다.

BMA에서는 한 블럭내의 모든 화소들이 동일 방향의 움직임을 갖는다는 것을 가정함으로 블럭내에서 서로 움직임이 다른 물체가 둘이상 있는 경우에는 각 물체의 변화에 따른 움직임을 효과적으로 반영하기가 어려워진다. 이와 같이 움직임이 다른 물체는 인접 블럭간의 움직임 벡터를 다르게 하기 때문에 움직임 보상 영상의 블럭 경계부분에 블럭화 현상(blocking artifact)를 일으킨다. BMA는 일정한 탐색범위에서 모든 블럭을 순차적으로 비교해 보는 전탐색(full search) BMA 를 기준으로 하지만, 탐색점의 수를 줄임으로써 움직임 벡터를 보다 고속으로 추정하며 하드웨어를 단순화 시킬 수 있는 방법이 연구되고 있다. 고속 BMA는 전탐색 BMA 보다 움직임 벡터의 추정이 부정확하게 되므로 물체의 보다 정확한 움직임을 표현하기 위한 방법의 도입이 필요하다. 한가지 방법으로 블럭의 크기를 보다 작게하여 움직임을 추정함으로써 물체의 움직임을 보다 정확하게 표현할 수 있으나 이는 움직임 벡터 표현을 위한 비트의 수를 증가 시키게 된다.

이러한 문제점을 해결하기 위한 방법으로 Liu [18]와 Orchard [19] 는 움직임 벡터 분할을 제안하였다. Liu 의 방법은 블럭을 동일 크기의 부블럭으로 나눈 후 각 부블럭의 이웃에 있는 다른 2 블럭 움직임 벡터와 자신의 움직임 벡터로 3 개의 벡터쌍을 형성하고 최소 왜곡이 되는 하나의 벡터쌍을 선택한다. 이 벡터쌍에서 또 다시 최소 왜곡이 되는 벡터를 선택함으로써 각 블럭에 대한 움직임 벡터 분할을 얻을 수 있다. Liu의 방법에서 블럭을 4 개의 부블럭으로 나눌 때 움직임 벡터를 나타내기 위한 부가정보는 블럭당 나타나는 81 가지의 벡터쌍으로 부터 부블럭의 벡터쌍 선택을 나타내기 위한 7 비트와 벡터쌍으로 부터 부블럭에 대한 움직임 벡터의 선택을 나타내기 위한 4 비트로 구성되므로 블럭당 11 비트가 필요하다. Orchard는 블럭내에 서로 다른 움직임을 갖는 물체에 따라 현재 부호화된 프레임의 역방향 움직임 벡터 추정인 이전 영상에서 움직임 벡터를 분할하는 방법을 제시하였다. 이때 각 영역의 움직임 벡터는 인접 블럭의 움직임중 가장 가까운 벡터로 표현되므로 움직임 벡터 분할에 대한 부가정보는 줄어든다. 그러나 이는 부,복호기에서 상당히 많은 계산을 필요로 하고, 블럭의 나누어진 영역들이 규칙적인 모양을 가지지 않기 때문에 하드웨어의 구현이 어려워 진다.

본 논문에서는 한 블럭의 움직임 벡터 분할에 대해 Liu의 방법에 비하여 보다 작은 부가정보를 가지면서 Orchard의 방법에 비하여 구현하기에 용이한 새로운 움직임 벡터 분할 방법을 제안한다.

이는 먼저 자신의 블럭과 이웃해 있는 블럭의 움직임 벡터와 서로 상이한 것이 있을 때 블럭을 4 개의 부블럭으로 나눈다. 그리고 각 부블럭에 대해 이웃한 4 개의 블럭의 움직임 벡터를 가정했을 때 각 부블럭의 왜곡과 블럭 자체의 움직임 벡터에 의한 각 부블럭의 왜곡중에서 작은 것을 선택하여 더했을 때 최소를 나타내는 하나의 인접 블럭 움직임 벡터와 블럭 자신의 움직임 벡터로써 벡터쌍을 형성한다. 다시 이 벡터쌍으로 부터 각 부블럭에 대해 작은 왜곡을 갖는 벡터를 선택함으로써 블럭의 움직임 벡터 분할을 한다. 이때 부가정보는 벡터쌍을 형성하기 위한 2 비트와 벡터쌍으로 부터 부블럭의 움직임 벡터 선택을 나타내는 4 비트로 이루어지므로 6 비트가 된다. 그리고 움직임 벡터 분할에서의 움직임 보상의 성능을 개선하기 위하여 움직임 정합의 블럭을 단순히 이동시킴으로써 움직임 보상을 하는 기존의 방법과 달리 움직임 정합된 블럭에 이웃하는 화소들은 동일한 방향의 움직임을 가진다는 점을 고려하여 창을 이용한 중첩 움직임 보상 [

[20,21] 을 한다.

Ⅱ. BMA 를 이용한 움직임 벡터의 추정

영상 부호화에서 물체의 움직임을 추정하고 이를 보상하여 얻은 움직임 보상 영상과 원영상과의 시간적인 중복성을 제거한 차신호인 DFD를 부호화하는 방식을 MCC라 한다. 이때 DFD는 공간적인 중복성도 함께 제거하여 부호화를 함으로서 높은 데이타 압축 효과를 얻는다. 일반적으로 MCC에서는 병렬 처리를 통한 실시간 처리가 용이한 BMA에 의해서 움직임 벡터를 추정한다. 그림 1은 BMA를 이용한 움직임 벡터 추정을 나타낸 것이다.

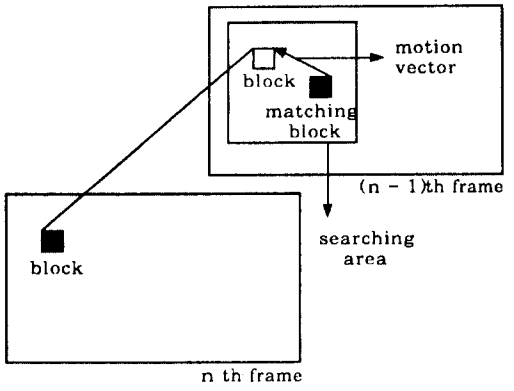


그림 1. BMA를 이용한 움직임 벡터 추정
Fig. 1. Motion vector estimation using BMA.

한 블럭내의 모든 화소들은 동일 방향의 움직임을 갖는다는 가정의 BMA에 의한 움직임 벡터의 추정은 현재 프레임에서 일정한 크기로 나누어진 블럭을 기준으로 하여 이전 프레임에 탐색영역을 설정하고 가장 유사한 블럭을 현재 프레임의 블럭과 정합시킴으로서 이루어진다. S×S 화소의 블럭을 고려할 때 프레임간 화소의 최대 움직임 거리를 P 화소라 가정하면 상하, 좌우로 최대 P 화소 만큼의 움직임이 발생할 수 있으므로 탐색영역의 크기는 (2P+S)×(2P+S)가 되고 탐색영역에서 정합이 가능한 탐색점의 수는 $(2P+1)^2$ 이 된다. 이때 고정된 부호화율로 하나의 움직임 벡터를 나타낼 때 비트수는 $\log_2(2P+1)^2$ 이다.

BMA에서 정합의 척도로는 NCCF(normalized cross correlation function), MSD(mean squared difference), MAD(mean absolute difference) 및 PDC(pel difference classification) [21] 등이 쓰이고 있으며 다음과 같다.

$$NCCF(i,j) = \frac{\sum_{k=1}^{S}\sum_{l=1}^{S} X_t(k,l)\cdot X_{t-1}(k+i,l+j)}{\left[\sum_{k=1}^{S}\sum_{l=1}^{S}(X_t(k,l))^2\right]^{1/2}\cdot\left[\sum_{k=1}^{S}\sum_{l=1}^{S}(X_{t-1}(k+i,l+j))^2\right]^{1/2}} \quad (1)$$

$$MAD(i,j) = \frac{1}{S^2}\sum_{k=1}^{S}\sum_{l=1}^{S}|X_t(k,l) - X_{t-1}(k+i,l+j)| \quad (2)$$

$$MSD(j,j) = \frac{1}{S^2}\sum_{k=1}^{S}\sum_{l=1}^{S}[X_t(k,l) - X_{t-1}(k=i,l+j)]^2 \quad (3)$$

$$PDC(i,k) = \frac{1}{S^2}\sum_{k=1}^{S}\sum_{l=1}^{S} T(k,l,i,j) \quad (4)$$

식 (4)에서

$$T(k,l,i,j) = \begin{cases} 0, & |X_t(k,l) - X_{t-1}(k+i,l+j)| < \text{Threshold} \\ 1, & \text{otherwise} \end{cases}$$

이다.

NCCF의 경우에는 탐색영역에서 $(2P+1)^2$ 개의 NCCF를 구하여 NCCF가 가장 큰 값을 갖는 블럭을 현재 블럭에 대한 움직임이 있기 전의 블럭으로 가정하여 (i,j)를 움직임 벡터로 정의한다. MAD 및 MSD의 경우에도 동일한 방법으로 탐색영역에서 $(2P+1)^2$ 개의 MAD 및 MSD를 구하고 가장 작은 값을 갖는 (i,j)를 움직임 벡터로 정의한다. 또한 적절한 문턱값을 사용함으로써 가장 빠른 정합을 이룰 수 있도록 제안된 PDC도 동일한 방법으로 움직임 벡터를 정의한다.

BMA에서 전탐색을 함으로써 나타나는 계산량을 줄이기 위한 대표적인 고속 탐색 BMA는 MVSA(menu vector search algorithm) [5], 2D-LOGS(two dimensional-logarithmic search) [6], OTS(one at a time search) [7-9], TSS(three step search) [10,11], PH-ODS(parallel hierarchical-one dimensional search) [12], CSA(cross search algorithm) [13], HSDE(hierarchically structured displacement estimate) [14,15] 와 이들을 개선, 결합한 형태의 방법 [16,17] 등이 있다. 이들은 근사 움직임을 찾은 후 미세 움직임을 단계적으로 찾는 것으로 탐색점의 수를 줄여 보다 효율적으로 움직임을 추정할 수 있도록 하며 하드웨어를 단순화시키지만 전탐색 BMA에 비하여 물체의 움직임을 정확하게 찾지 못한다.

Ⅲ. 움직임 벡터 분할과 창을 이용한 중첩 움직임 보상

BMA 에 의한 움직임 벡터의 추정은 한 블럭내의 모든 화소들이 동일 방향의 움직임을 갖는다고 가정함

으로 실제 영상에서는 동일 블럭내에서 움직임이 다른 물체가 하나 또는 그 이상 존재할 수 있다. 이러한 경우 주어진 움직임 벡터는 블럭내 움직임 물체 중 단지 하나에게만 타당하거나 최악의 경우에는 움직임 물체 어느 것에도 일치하지 않게 되어 움직임 보상시 아주 큰 예측 오차를 일으킨다. 이때 블럭의 크기를 작게하면 이러한 현상은 상당히 줄어 들지만 움직임 벡터를 표현하기 위해서는 많은 비트가 필요하게 되는 문제점을 일으킨다. 이러한 문제점을 해결하기 위하여 Liu [18] 와 Orchard[19] 는 움직임 벡터를 분할하는 방법을 제시하였다. Liu 는 각 블럭을 4 개의 부 블럭으로 나눈 후 각 부블럭에 이웃하는 움직임 벡터와 자신의 움직임 벡터를 이용하여 보다 나은 움직임 추정을 하였으나 블럭당 11 비트의 부가정보를 필요로 하는 단점이 있다. Orchard 는 현재 부호화된 프레임의 역방향 움직임 벡터 추정인 이전 영상에서 움직임 벡터를 분할함으로써 보다 나은 움직임을 추정하였으나 하드웨어로 구현하기에 복잡하다.

그래서 약간의 부가정보를 가지면서도 계산양을 줄여 구현하기에 용이한 움직임 벡터 분할 방법을 제안한다. 이는 자신의 블럭과 이웃해 있는 블럭의 움직임 벡터와 서로 상이한 것이 있을 때 블럭을 동일 크기의 4 개의 부블럭으로 나누고 자신과 인접하는 블럭의 움직임 벡터를 고려하여 움직임 벡터를 분할하는 것이다. 인접 블럭 움직임 벡터와 자신의 움직임 벡터로서 벡터쌍을 형성하고, 각 부블럭에 대해 작은 왜곡을 갖는 움직임 벡터를 벡터쌍으로 부터 찾아서 부블럭의 수정된 움직임 벡터로 하는 것이다.

그림 2의 (a)는 같은 움직임을 가지는 물체가 다른 움직임 벡터로 나타나는 두개의 블럭 b_1 및 b_2에 의해서 나뉘어져 있는 경우이다. 블럭 b_1은 이동 물체가 b_1 블럭 대부분을 차지하고 있기 때문에 정확한 움직임 추정이 가능하다. 그러나 블럭 b_2는 이동 물체가 주위 배경에 비해 상대적으로 작은 영역으로 되어 있어서 움직임 벡터는 그림 2의 (a)와 같이 나타날 수 있으나 O_2 영역에 대해서는 부정확한 움직임의 추정이 될 수 있다. 이러한 문제점을 해결하기 위한 하나의 방법은 블럭을 같은 크기의 4 개의 부블럭으로 나누어 자신의 블럭과 수직, 수평 방향으로 이웃하는 블럭의 움직임 벡터를 고려하여 예측 오차를 가장 작게 나타내는 것을 각 부블럭의 수정된 움직임 벡터로 선택하는 것이다. 한가지 방법으로 그림 2의 (b)에 나타낸 것과 같이 O_2 영역의 움직임 벡터 V_C로 V_L을 적용하면 움직임 벡터는 정확하게 되고 예측 오차는 줄어들게 된다.

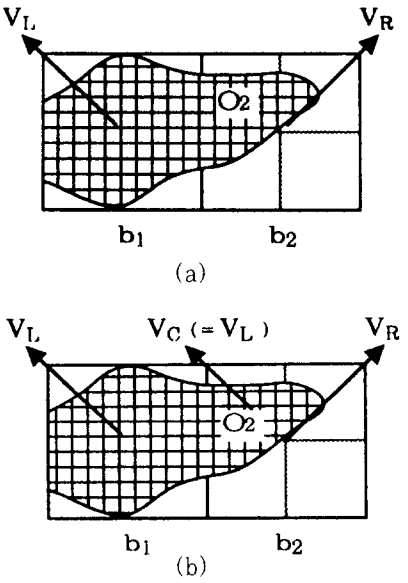


그림 2. 다른 움직임 벡터를 갖는 인접한 두 블럭
(a) 인접한 두 블럭에 걸쳐있는 이동 물체
(b) b_2 블럭에서 영역 O_2의 실제 움직임 벡터

Fig. 2. Two neighborning blocks having different motion vector.
(a) Moving object crossed over two neighboring blocks
(b) Real motion vector of region O_2 in the block b_2

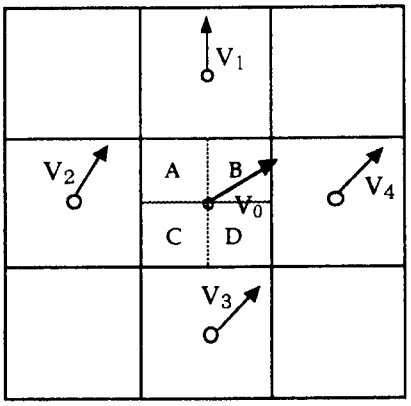


그림 3. 4개의 부 블럭 분할 및 인접 블럭의 움직임 벡터

Fig. 3. Four-subblock segmentations and motion vectors of neighbor blocks.

그림 3에서 나타낸 것과 같이 부블럭 A를 자신의 움직임 벡터 V_0와 인접한 두 블럭의 움직임 벡터 V_1, V_2 중 하나를 부블럭 A의 수정된 움직임 벡터로 선택하게 되면 이를 나타내기 위한 부가정보로 2 비트가

필요하다. 따라서 블럭당 모두 8 비트의 부가 정보가 필요하게 된다. 이는 움직임 추정 블럭의 크기를 8×8 할 때 화소당 0.125 비트가 되어서 최대 7 화소 움직임을 가정한 BMA에서 움직임 벡터을 나타내기 위한 비트율이 화소당 0.125 비트인 것을 고려하면 데이타 압축의 의미를 상실하게 된다.

또한, 그림 4와 같이 부블럭 B의 물체가 실제로는 블럭 b_1의 움직임 벡터를 가진다고 했을 때 위의 방식에서는 부블럭 B는 b_1 블럭의 움직임 벡터로 표현될 수 없기 때문에 최적 움직임 벡터의 선택이 불가능하다.

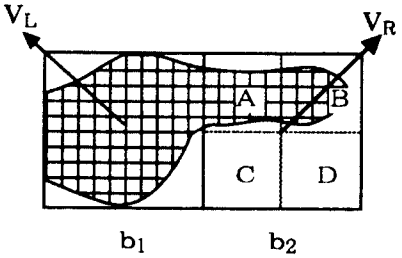


그림 4. 부 블럭의 인접 블럭을 이용한 움직임 벡터 분할의 부정합

Fig. 4. Mismatching of the motion vector segmentation using neighbor blocks of subblock.

따라서 화질 개선과 데이타 압축을 조화시키기 위하여 주변 4 개의 블럭을 모두 이용할 수 있으면서 위의 방법에 비해 더 적은 부가정보로 움직임 벡터를 분할하는 방법을 제안한다. 이를 위한 우선적인 가정은 한 블럭내에서는 움직임이 다른 물체가 단지 두개만이 존재할 수 있다는 것이다. 이는 BMA에서 블럭의 크기가 전체 영상에 비해 상당히 작기 때문에 충분히 가능한 조건이라 할 수 있다. 움직임 벡터를 분할하기 위하여 자신의 움직임 벡터 V_0와 인접 블럭의 움직임 벡터 V_1, V_2, V_3 및 V_4에 의한 각 부블럭의 평균 DFD 에너지를 아래의 식을 이용하여 구한다.

$$D_m(V_0) = \frac{4}{S^2} \sum_i \sum_j [\ (x_n(i,j) - X_{n-1}((j,j) - V_0)\ ]^2 \quad (5)$$

$$D_m(V_k) = \frac{4}{S^2} \sum_i \sum_j [\ (X_n(i,j) - X_{n-1}(i,j) - V_k\ ]^2 \quad (6)$$

여기서 V_k는 4 개의 인접 블럭의 움직임 벡터를 나타내고 i, j는 해당 부블럭내의 화소들을 가리킨다. 그리고 m은 4 개의 부블럭 A, B, C 및 D를 나타낸다. 각 부블럭에 대하여 자신의 움직임 벡터 V_0와 인접 블럭의 움직임 벡터 V_k에 의한 DFD 중에서 작은 왜곡을 갖는 것을 선택하여 4 개 부블럭에 대하여 더했을 때 최소로 하는 V_k를 다음과 같이 하여 벡터쌍 (V_0, V_k)를 구한다.

$$(V_0, V_k) = \arg \underset{k=1,2,3,4}{[} \sum \min \{D_m(V_0), D_m(V_k)\} \underset{m \in \{A,B,C,D\}}{]} \quad (7)$$

그리고 구해진 벡터쌍 (V_0, V_k)에 의한 m 번째 부블럭에 대한 왜곡 $D_m(V_0)$와 $D_m(V_k)$ 중에서 작은 것을 나타내는 V_0 또는 V_k를 m 번째 부블럭에 대한 수정된 움직임 벡터로 함으로써 움직임 벡터의 분할을 한다. 여기서 이를 움직임 벡터 분할-I (Motion vector segmentation-I : MVS-I)이라 한다. 이때 움직임 벡터 분할을 나타내기 위한 부가정보는 벡터쌍의 선택에 대한 2 비트, 각 부 블럭에 대하여 V_0 또는 V_k의 선택에 대한 1 비트가 되어 전체적으로 6 비트가 된다.

MVS-I은 성능면에서는 최적이나 모든 인접 블럭의 움직임 벡터에 의한 왜곡과 자신의 움직임 벡터에 대한 왜곡을 모두 계산하고 저장을 해서 비교를 하기 때문에 다소 복잡한 편이다. 따라서 새로운 방법으로 우선 다음식을 이용하여 각 부블럭당 평균 DFD를 최소로 하는 인접 블럭의 움직임 벡터 $V_k(m)$을 구한다.

$$V_k(m) = \underset{V_k \in \{V_1, V_2, V_3, V_4\}}{\text{agr Min}} \{D_m(V_k)\} \quad (8)$$

여기서 m은 부블럭 A, B, C 및 D를 나타낸다.

인접 블럭의 움직임 벡터를 이용하여 식 (8)에서 구한 각 부블럭에 대한 $V_k(m)$은 다시 식 (9)에 의해 선택된 $V_k(m)$과 자신의 움직임 벡터 V_0에 의해서 벡터쌍이 형성된다.

$$(V_0, V_k) = \underset{V_k(m)}{\arg \text{Max}} [\ D_m(V_0) - D_m(V_k(m))\ ] \quad (9)$$

식 (9)와 같이 V_0와 V_k로 만들어진 벡터쌍에서 각 부블럭에 대해 보다 작은 왜곡을 나타내는 V_0 또는 V_k를 선택하여 각 부블럭에 대한 수정된 움직임 벡터로 함으로써 움직임 벡터의 분할을 한다. 여기서 이를 움직임 벡터 분할-II (Motion vector segmentation-II : MVS-II)이라고 한다. 이때 움직임 벡터 분할을 나타내기 위한 부가정보는 MVS-I과 같이 벡터쌍의 선택을 위한 2 비트와 각 부블럭의 왜곡에 따른 V0와 Vk의 선택을 나타내기 위한 1 비트로 이루어지므로 전체적으로 6 비트가 된다.

따라서 움직임 벡터 분할을 나타내기 위한 부가정보

는 프레임당 고정된 부호화율을 고려할 때 블럭당 6 비트가 필요하게 되므로 블럭의 크기를 8×8 로 하여 움직임 벡터를 추정할 때 화소당 평균 0.09375 비트가 소요된다. 그러나 움직임 벡터 분할에서 인접 블럭의 움직임 벡터가 모두 자신의 움직임 벡터와 같을 때 움직임 벡터 분할에 대한 정보는 필요가 없으므로 프레임당 고정된 부호화율을 적용하지 않는다면 부가 정보량은 줄어든다.

기존의 BMA에서는 움직임 정합 블럭을 움직임 벡터만큼 이동시키는 형태로 현재 프레임에 대한 움직임 보상 영상을 얻는다. 그러나 인접 블럭간의 움직임 벡터가 다르면 현재 프레임에 대한 예측 영상인 움직임 보상 영상의 블럭 경계부분에서는 화소 밝기의 불연속으로 인한 블럭화가 일어나므로 이러한 현상을 줄이기 위해서 움직임 정합 블럭에 이웃한 화소들은 일반적으로 동일한 방향의 움직임을 갖는다는 점을 고려하여 창을 이용한 중첩 움직임 보상(overlapped motion compensation using window : OMCW) [20,21] 을 한다. 본 논문에서는 식 (10)의 Sinusoid 창을 이용한다. 이때 S 는 블럭 크기를 나타내며 움직임 벡터 분할을 한 경우의 S는 S/2가 된다.

$$W_{ij} = \frac{\sin^2\pi(i+0.5)}{2S} \sin^2\frac{\pi(j+0.5)}{2S} \qquad (10)$$

여기서 i, j 는 0, 1, ... 2S-1 이다.

Ⅳ. 실험 결과 및 검토

제안한 알고리즘의 성능을 평가하기 위하여 256 밝기 레벨을 갖는 480×704크기를 갖는 영상을 수평, 수직 방향으로 2 : 1 간축에 의해서 얻은 240×352 크기의 복잡한 움직임을 갖는 "Football", "Calendar", "Flower Garden" 영상과 288×352 크기의 단순한 움직임을 갖는 "Salesman" 영상 16 프레임에 대해서 실험을 하였다. BMA에서 움직임 벡터의 추정을 위한 블럭의 크기는 8×8 을 기준으로 하였으며 왜곡 척도는 간단하면서도 MSD와 성능의 차이가 거의 없는 MAD를 사용하였다. 그리고 제안한 움직임 벡터 분할 및 Sinusoid 창을 이용한 중첩 움직임 보상인 OMCW를 한 것이 기존의 BMA에 의한 것과 성능의 차이를 식 (11)의 PSNR과 식 (12)의 엔트로피(entropy)를 이용하여 평가를 하였다.

$$PSNR = 10\log(255\times255/\sigma_e^2) \quad [\ dB\ ] \qquad (11)$$

$$Entropy = -P_i\sum_i \log P_i \quad [\ bits\ ] \qquad (12)$$

여기서 σ_e^2 은 재생영상의 평균 잡음 전력 이며 Pi는 화소 밝기의 확율이다.

표 1은 최대 7 화소 움직임의 BMA에서 제안된 움직임 벡터 분할과 중첩 움직임 보상인 OMCW된 15 프레임의 움직임 보상 영상에 대한 평균 PSNR을 나타낸 것이다. 또한 단순히 전탐색 BMA 및 움직임 벡터 분할을 나타낸 것은 기존의 움직임 보상 방법을 적용한 것이다.

표 1. 움직임 벡터 분할과 OMCW에 의한 영상의 PSNR [dB]

Table 1. PSNR [dB] of image by motion vector segmentation and OMCW.

Algorithm \ Image	Football	Calendar	Flower Garden	Salesman
Full Search(8×8)	23.72	21.94	23.39	35.71
OMCW	24.12	22.17	23.78	36.20
MVS-Ⅱ	24.06	21.95	23.42	35.88
MVS-Ⅰ	24.06	21.96	23.44	35.89
MVS-Ⅱ, OMCW	24.42	22.24	23.95	36.30
MVS-Ⅰ, OMCW	24.43	22.25	23.96	36.40
Full search(4×4)	25.57	22.93	24.10	36.91

표 1를 살펴보면 블럭의 크기를 8×8 로 한 전탐색 BMA에 기존의 움직임 보상을 한 것보다 중첩 움직임 보상인 OMCW를 한 경우에는 영상에 따라 약 0.2~0.5 [dB] 의 PSNR을 개선을 나타낸다. 그리고 부가정보 6 비트를 사용하여 움직임 벡터 분할을 한 것은 중첩 움직임 보상인 OMCW를 한 것에 비하여 전체적으로 약간의 성능은 떨어지지만 전탐색 BMA에 기존의 움직임 보상을 한 것 보다는 나은 성능을 나타낸다. 그리고 움직임 벡터 분할 MVS-Ⅰ에 의한 것 보다는 움직임 벡터 분할 MVS-Ⅱ에 의한 것이 나은 성능을 보인다. 부가정보 6 비트를 사용하여 움직임 벡터 분할을 한 후 중첩 움직임 보상인 OMCW를 한 것은 블럭의 크기를 8×8 로 한 전탐색 BMA 에 기존의 움직임 보상을 한 것보다 영상에 따라 약 0.8~1.1 [dB] 의 PSNR 개선을 나타낸다. 따라서 표 2을 종합해보면 약간의 부가정보를 사용하여 움직임 벡터 분할을 한 후 중첩 움직임 보상인 OMCW를 함으로서 효율적으로 움직임 보상 영상의 PSNR을 증가시킬 수 있음을 알 수 있다. 이는 DFD의 부호화를 쉽게 할 수 있을 뿐아니라 양호한 화질을 갖는 최종 재생영상을 얻을 수 있음을 의미한다.

표 2는 "Football" 영상에 대하여 블럭의 크기를 8×8 로 한 고속 BMA에서 움직임 벡터의 추정 방법에 따라 OMCW, MVS-Ⅰ 및 MVS-Ⅱ를 적용함으로써

나타나는 15 프레임의 움직임 보상 영상에 대한 평균 PSNR 이다.

표 2. 고속 BMA에 의한 움직임 보상된 "Football" 의 PSNR [dB]

Table 2. PSNR [dB] of motion compensated "Football" by fast BMA.

Type / Algorithm	Conventional motion compensation	OMCW	MVS-II	MVS-I
2D-LOGS	23.03	23.57	24.15	24.18
PH-ODS	20.81	21.56	22.64	22.71
C S A	20.21	20.38	20.65	20.65
T S S	22.03	22.44	22.76	22.78
O T S	21.58	22.02	22.62	22.65

표 3은 "Calendar" 영상에 대하여 표 2와 동일한 방법에 의하여 얻은 PSNR을 나타낸 것이다.

표 3. 고속 BMA에 의한 움직임 보상된 "Calendar" 의 PSNR [dB]

Table 3. PSNR [dB] of motion compensated "Calendar" by fast BMA.

Type / Algorithm	Conventional motion compensation	OMCW	MVS-II	MVS-I
2D-LOGS	20.53	21.27	21.74	21.78
PH-ODS	18.79	19.77	21.34	21.40
C S A	21.09	21.70	21.74	21.76
T S S	17.71	18.24	18.58	18.59
O T S	19.92	20.79	21.61	21.66

표 4는 "Flower Garden" 영상에 대하여 표 2와 동일한 방법에 의하여 얻은 PSNR을 나타낸 것이다.

표 4. 고속 BMA에 의한 움직임 보상된 "Flower Garden" 의 PSNR [dB]

Table 4. PSNR [dB] of motion compensated "Flower Garden" by fast BMA.

Type / Algorithm	Conventional motion compensation	OMCW	MVS-II	MVS-I
2D-LOGS	22.64	23.32	23.46	23.46
PH-ODS	18.91	20.01	22.19	22.25
C S A	19.78	20.28	21.12	21.14
T S S	20.35	21.02	21.04	21.06
O T S	20.29	21.22	22.69	22.73

표 5는 "Salesman" 영상에 대하여 표 2와 동일한 방법에 의하여 얻은 PSNR을 나타낸 것이다.

표 5. 고속 BMA에 의한 움직임 보상된 "Salesman" 의 PSNR [dB]

Table 5. PSNR [dB] of motion compensated "Salesman" by fast BMA.

Type / Algorithm	Conventional motion compensation	OMCW	MVS-II	MVS-I
2D-LOGS	35.35	35.90	35.97	35.99
PH-ODS	33.52	34.28	34.77	34.96
C S A	33.75	33.94	34.10	34.11
T S S	34.45	34.94	34.95	34.97
O T S	34.93	35.45	35.46	35.46

표 2~5에서 2D-LOGS, PH-ODS 및 OTS에서는 최대 움직임을 7 화소로 하였다. 그리고 CSA에서는 최대 움직임을 8 화소로 하였으며, TSS의 최대 움직임은 6 화소로 하였다. CSA에서 평균 MAD가 문턱값 9 보다 작거나 같으면 움직임 벡터 추정을 위한 탐색을 중단하고 그렇지 않으면 탐색을 계속하도록 하였다.

표 2~5를 살펴보면 전탐색 BMA에 비하여 근사한 움직임 벡터를 찾고 세밀한 움직임 벡터를 단계적으로 찾기 위한 방법으로 탐색점의 수를 줄임으로써 전탐색 BMA에 비하여 부정확한 움직임을 찾지만 보다 고속으로 움직임 벡터를 추정하기 위한 2D-LOGS, PH-ODS, OTS, CSA 및 TSS에서는 움직임 벡터를 추정하고 기존의 움직임 보상을 하는 것보다 중첩 움직임 보상인 OMCW를 하는 것이 보다 높은 PSNR을 가지는 움직임 보상 영상을 나타낸다. 그리고 탐색점의 수를 줄임으로써 전탐색 BMA에 비하여 부정확한 움직임 벡터의 추정이 이루어지는 이들 알고리즘에 대해서는 중첩 움직임 보상인 OMCW에 의한 것보다 전체적으로 부가정보 6 비트를 사용하여 움직임 벡터의 분할을 한 후 기존의 움직임 보상을 한 것이 보다 높은 PSNR의 움직임 보상 영상을 나타낸다.

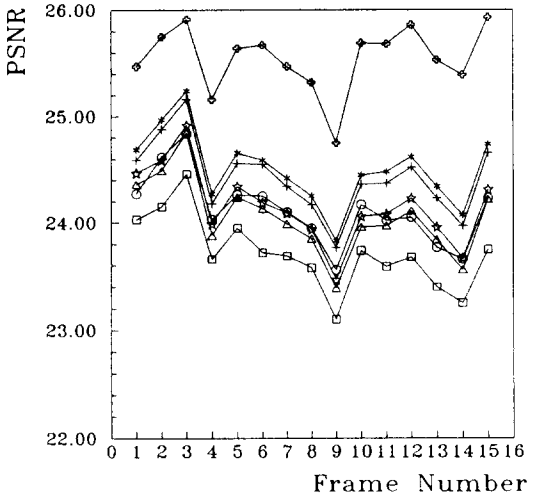
표 2~5에서 전체적인 PSNR을 살펴보면 탐색점의 수를 줄여 고속 BMA를 실현하기 위하여 본 논문에서 고려된 방법들 중에서 PH-ODS에 움직임 벡터를 분할을 한 움직임 보상 영상이 가장 큰 PSNR개선을 보인다. 그리고 2D-LOGS, OTS, TSS, CSA의 순으로 PSNR 개선을 나타낸다. 특히 CSA에서는 영상에 따라 PSNR 개선의 정도가 크게 차이가 나는데 그 이유는 CSA에서 움직임 벡터의 추정을 계속할 것인지 중단할 것인지를 나타내기 위해서 선택된 문턱값에 따른 특성이다.

그리고 영상에 따라 살펴보면 "Football", "Calendar", "Flower Garden" 영상과 같이 움직임이 크고 배경 부분이 복잡한 영상에서는 움직임 벡터 분할

을 함으로써 얻어지는 움직임 보상 영상은 기존의 움직임 보상에 의한 것보다 PSNR의 개선이 크게 나타난다. 그러나 "Salesman" 영상과 같이 움직임이 작고 배경 부분이 단순한 영상에서는 움직임 벡터 분할을 함으로서 얻어지는 움직임 보상 영상은 기존 움직임 보상에 의하여 얻어지는 것보다 PSNR의 개선은 크게 드러나지는 않지만 중첩 움직임 보상인 OMCW에 의한 것보다는 PSNR이 높게 나타난다.

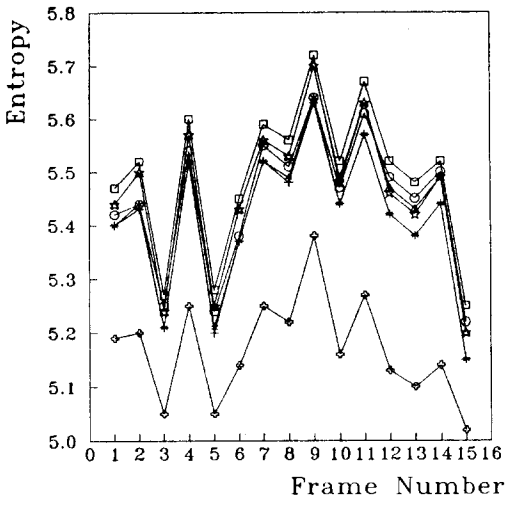
따라서 BMA에서 탐색점의 수를 줄임으로써 고속탐색을 하는 알고리즘에 부가정보 6 비트를 사용하여 움직임 벡터 분할을 하면 효율적으로 움직임 보상 영상의 PSNR을 개선할 수 있다. 그리고 고속 탐색 BMA에 움직임 벡터 분할 및 중첩 움직임 보상인 OMCW를 같이 적용하면 더욱 효율적일 것이다.

그림 5와 6은 최대 움직임을 7 화소로 가정하고 블럭의 크기를 8×8 한 전탐색 BMA에 OMCW와 움직임 벡터 분할을 "Football" 영상에 적용함으로써 나타나는 프레임에 따른 움직임 보상 영상의 PSNR과 엔트로피 이다.



□ : Full search BMA(8×8) & conventional motion compensation
● : Full search BMA(8×8) & OMCW
☆ : Full search BMA(8×8), MVS-I & conventional motion compensation
△ : Full search BMA(8×8), MVS-II & conventional motion compensation
* : Full search BMA(8×8), MVS-I & OMCW
+ : Full search BMA(8×8), MVS-II & OMCW
+ : Full search BMA(4×4) & conventional motion compensation

그림 5. 프레임에 따른 움직임 보상 영상의 PSNR [dB]
Fig. 5. PSNR [dB] motion compensated image according to frame.



□ : Full search BMA(8×8) & conventional motion compensation
● : Full search BMA(8×8) & OMCW
☆ : Full search BMA(8×8), MVS-I & conventional motion compensation
△ : Full search BMA(8×8), MVS-II & conventional motion compensation
* : Full search BMA(8×8), MVS-I & OMCW
+ : Full search BMA(8×8), MVS-II & OMCW
+ : Full search BMA(4×4) & conventional motion compensation

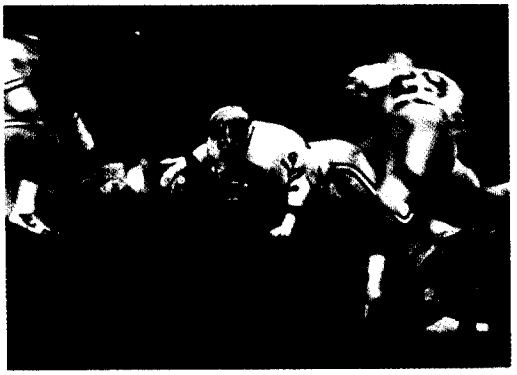
그림 6. 프레임에 따른 DFD 의 엔트로피 [bit]
Fig. 6. Entropy [bit] of DFD according to frame.

그림 5를 살펴보면 블럭의 크기를 8×8 로 한 전탐색 BMA에 기존의 움직임 보상을 한 것보다 중첩 움직임 보상인 OMCW를 한 것이 영상에 따라 약 0.2~0.4 [dB] 의 PSNR 을 개선을 갖는 움직임 보상 영상을 나타낸다. 그리고 부가정보 6 비트를 사용하여 움직임 벡터 분할을 한 후 기존의 움직임 보상을 한 것도 보다 높은 PSNR을 갖는 움직임 보상 영상을 나타내지만 OMCW를 한 것보다는 약간 못하다. 부가정보 6 비트를 사용한 움직임 벡터 분할 및 OMCW를 한 것은 블럭의 크기를 8×8 로 한 전탐색 BMA에 기존의 움직임 보상을 한 것보다 약 0.8~1.1 [dB] 의 PSNR 개선을 갖는 움직임 보상 영상을 나타낸다.

그림 6를 살펴보면 블럭의 크기를 8×8 로 한 전탐색 BMA에 기존의 움직임 보상을 한 것보다 중첩 움직임 보상인 OMCW를 한 것이 영상에 따라 약 0.01~0.06 비트 낮은 엔트로피를 갖는 DFD를 나타낸다. 그리고 부가정보 6 비트를 사용하여 움직임 벡터 분할을 한 후 기존의 움직임 보상을 한 것도 보다 낮은 엔트로피를 갖는 DFD를 나타내지만 OMCW를 한 경우

의 DFD 보다는 높은 엔트로피를 나타낸다. 부가정보 6 비트를 사용한 움직임 벡터 분할을 한 후 OMCW를 한 것은 블럭의 크기를 8×8 로 한 전탐색 BMA에 기존의 움직임 보상을 한 것보다 약 0.05~0.08 비트 낮은 엔트로피를 갖는 DFD를 나타낸다. 그림 5와 6에서 블럭의 크기를 8×8 로 한 전탐색 BMA에 부가정보 6 비트를 사용하여 움직임 벡터 분할을 한 후 OMCW한 것은 전탐색 BMA에 기존의 움직임 보상을 한 것보다 높은 PSNR의 움직임 보상 영상을 나타낸다. 또한, DFD의 엔트로피도 PSNR과 마찬가지로 낮게 나타난다.

따라서 제안된 움직임 벡터 분할을 한 후 OMCW를 하는 경우에는 움직임 벡터 분할을 나타내기 위한 부가정보 6 비트를 사용하여 움직임 보상 영상의 PSNR을 개선하고 DFD의 엔트로피를 효율적으로 감소시킨다. 이는 블럭화가 없는 최종 재생영상을 얻을 수 있도



(a)

(b)

(c)

(d)

(e)

(f)

(g) (h)

그림 7. 원영상 및 움직임 보상 영상
(a) 8 번째 프레임 원영상 (b) 기존 움직임 보상에 의한 보상 영상
(c) 전탐색 BMA 및 OMCV (d) 전탐색 BMA(4×4)
(e) 전탐색 BMA 및 MVS-II (f) 전탐색 BMA 및 MVS-I
(g) 전탐색 BMA, MVS-II 및 OMCV (h) 전탐색 BMA, MVS-I 및 OMCV

Fig. 7. Original image and motion compensated image.
(a) the 8 th frame of original image (b) Compensated image by conventional motion compensation
(c) Full search BMA and OMCV (d) Full search BMA(4×4)
(e) Full search BMA and MVS-II (f) Full search BMA and MVS-I
(g) Full search BMA, MVS-II and OMCV (h) Full search BMA, MVS-I and OMCV

록 하며 DFD의 부호화를 쉽도록 한다.

그림 7는 전탐색 BMA에 움직임 벡터의 분할 방법 MVS-I 또는 MVS-II 및 중첩 움직임 보상인 OMCW를 적용함으로써 얻어지는 움직임 보상 영상을 나타낸 것이다. 이때 움직임의 정도는 최대 7 화소인 것으로 하였으며 왜곡의 척도는 MAD를 사용하였다. 그리고 창은 Sinusoid를 이용하였다. 전탐색 BMA에서 블럭 크기를 4×4 로 한 것은 괄호안에 4×4 로 표시하였으며 그렇지 않은 경우의 전탐색 BMA는 블럭 크기를 8×8 로 한 것이다.

그림 7을 살펴보면 블럭의 크기를 8×8 로 한 전탐색 BMA에 의한 기존의 움직임 보상 방법에 의해서 움직임 보상된 영상에서는 블럭화가 나타나 보이다. 그러나 중첩 움직임 보상인 OMCW를 한 것에서는 블럭화가 많이 줄어듬을 볼 수 있다. 그리고 약간의 부가정보를 사용한 움직임 분할 방법 MVS-I 또는 MVS-II를 한 후 기존 움직임 보상 방법을 한 움직임 보상된 영상에서는 블럭의 크기를 8×8 한 전탐색 BMA에 기존 움직임 보상 방법을 한 것보다 보다 작은 블럭의 형태로 블럭화가 나타나 보인다. 이는 블럭의 크기를 4×4 로 한 전탐색 BMA에 기존의 움직임 보상을 한 움직임 보상된 영상에 나타나는 블럭화 현상과 비슷한 모양을 보이나 블럭의 크기를 4×4 로 한 전탐색 BMA에 기존 움직임 보상 방법을 한 것보다는 블럭의 모양이 큰 것이 많이 나타남을 볼 수 있다. 부가정보 6 비트를 사용하여 움직임 벡터를 분할을 한 후 중첩 움직임 보상인 OMCW를 한 움직임 보상 영상은 블럭의 크기를 8×8 한 전탐색 BMA에 중첩 움직임 보상인 OMCW를 한 것보다 양호한 화질을 갖는다. 블럭의 크기를 8×8 한 전탐색 BMA에 움직임 벡터를 분할을 한 후 중첩 움직임 보상인 OMCW를 한 영상에서는 블럭화가 거의 나타나지 않으나 블럭의 크기를 4×4 로 한 전탐색 BMA에 기존 움직임 보상 방법을 한 것에서는 블럭화가 있음을 알 수 있다. 이는 DFD를 부호화하여 더하더라도 완전한 제거가 이루어지지 않으므로 최종 재생영상에 화소 밝기의 불연속인 블럭화를 일으킨다.

따라서 약간의 부가정보를 사용한 움직임 벡터를 분할을 한 후 중첩 움직임 보상인 OMCW를 하는 것은 높은 PSNR의 블럭화가 없는 움직임 보상 영상을 얻는 유용한 방법이다. 그리고 움직임 보상 영상의 블럭화는 최종 재생영상에 영향을 미치므로 움직임 벡터를 분할을 한 후 OMCW를 하는 것은 DFD의 부호화를 보다 쉽게 할 수 있게 할 뿐만아니라 보다 양호한 화

질의 재생영상을 얻기 위한 효율적인 방법이다.

V. 결 론

MCC에서 BMA는 한 블럭내의 모든 화소들은 동일 방향의 움직임을 갖는다고 가정함으로써 한 블럭내에 움직임이 서로 다른 물체가 있는 경우에는 물체의 움직임을 효과적으로 반영하지 못하게 되어 움직임 보상 영상에 블럭화를 일으킨다. 특히, 근사 움직임을 찾은 후 미세 움직임을 단계적으로 찾기 위하여 탐색점의 수를 줄임으로써 하드웨어를 단순화시키며 움직임 벡터를 보다 빨리 탐색하도록 하는 고속 BMA는 전탐색 BMA에 비하여 부정확한 움직임 벡터를 추정하기 때문에 움직임 보상 영상에 더 많은 블럭화를 일으킨다. 본 논문에서는 이러한 문제점을 해결하기 위하여 기존의 방법에 비하여 작은 부가정보를 가지면서 구현하기 쉬운 새로운 움직임 벡터 분할을 제안하였다. 이는 먼저 각 부블럭에 대해 이웃한 4 개 블럭의 움직임 벡터를 가정했을 때 각 부블럭의 왜곡과 블럭 자체의 움직임 벡터에 의한 각 부블럭의 왜곡중에서 작은 것을 선택하여 더했을 때 최소를 나타내는 하나의 인접 블럭 움직임 벡터와 자신의 움직임 벡터로써 벡터쌍을 형성한다. 그리고 이 벡터쌍으로 부터 각 부블럭에 대하여 작은 왜곡을 나타내는 움직임 벡터를 찾아 부블럭의 새로운 움직임 벡터로 하는 것이다. 또한, 움직임 벡터 분할의 성능을 향상시키기 위하여 창을 이용한 중첩 움직임 보상을 적용하였다.

본 논문에서 제안된 움직임 벡터 분할만을 한 것은 창을 이용한 중첩 움직임 보상을 한 것보다 움직임 보상 영상의 PSNR이 약간 낮게 나타나고 DFD의 엔트로피도 약간 높게 나타난다. 그러나 움직임 벡터 분할과 창을 이용한 중첩 움직임 보상을 함께 한 것은 그렇지 않은 것에 비하여 움직임 보상 영상의 PSNR을 개선시킨다. 특히 하드웨어를 단순화시키며 움직임 벡터를 보다 빨리 탐색하는 고속 BMA에서는 약간의 부가정보를 사용하여 움직임 벡터 분할을 함으로써 움직임 보상 영상의 PSNR을 전탐색 BMA에서 얻을 수 있었던 것보다 개선할 수 있었다. 또한 DFD의 엔트로피도 더 많이 감소시킬 수 있었다. 그리고 보다 양호한 화질을 갖는 움직임 보상 영상을 얻을 수 있었다.

참 고 문 헌

[1] H. G. Musmann, P. Pirch and H. J. Grallert, "Advances in pcicture coding", *Proc. of IEEE*, Vol. 73, No. 4, pp. 523-548, Apr. 1985.

[2] A. N. Netravali and D. Robbins, "Motion compenated television coding : Part I," *Bell Syst. Tech. J.*, Vol. 58, No. 3, pp. 631-670, Mar. 1979.

[3] C. Cafforia and C. Rocca, "Methods for measuring small displacements of television images," *IEEE Trans. on Information Theory*, Vol. IT-22, No. 5, pp. 573-579, Sep. 1976.

[4] D. R. Walker and K. R. Rao, "Motion compensated coder," *IEEE Trans. on Commun.*, Vol. COM-35, No. 10, pp. 1171-1178, Nov. 1987.

[5] J. R. Jain and A K. Jain, Displacement measurement and its application in interframe image coding," *IEEE Trans. on Commun.*, Vol. COM-29, No. 12, pp. 1799-1808, Dec. 1981.

[6] Y. Ninomiya and Y. Ohtsuka, "A motion compensated interframe coding scheme for television pictures," *IEEE Trans. on Commun.*, Vol. COM-30, No. 1, pp. 201-211, Jan. 1982.

[7] S. Kappagantula and K. R. Rao, "Motion compensated interframe image coding," *IEEE Trans. on Commun.*, Vol. COM-33, No. 9, pp. 1011-1015, Sep. 1985.

[8] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. on Commun.*, Vol. COM-33, No. 8, pp. 888-896, Aug. 1985.

[9] R. Srinivasan and K. R. Rao, "Motion compensated coder for video conferencing," *IEEE Trans. on Commun.*, Vol. COM-35, No. 3, pp. 297-304, Mar. 1987.

[10] T. Koga, K. Iinuma, A. Hirano and T. Ishguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat'l Telecommun. Conf.*, New Orleans, LA, pp. G.5.3.1.-5.3.5, Nov. 1981.

[11] A. Furukawa ,T. Koga, K. Iinuma,, "Motion adaptive interpolation for video conference picture," *in Proc. ICC '84,* Amsterdam, The Netherlands, pp. 707-701, May. 1984.

[12] M. Ghanbari, "The cross search algorithm for motion estimation," *IEEE Trans. on Commun.*, Vol. COM-38, No. 7, pp. 950-953, Jul. 1990.

[13] L. G. Chen, W. T. Chen, Y. S. Jehng and T. D. Chiueh, "An efficient parallel motion estimation algorithm for digital image processing," *IEEE Trans. on Circuit and System for Video Tech.*, Vol. 1, No. 4, pp. 378-385, Dec. 1991.

[14] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimation," *Signal Processing,* Vol. 11, No. 4, pp. 387-404, Dec. 1986.

[15] M. Bierling, "Displacement estimation by hierarchical block matching," *Proc. of SPIE Conf. on Visual Commun. and Image Processing,* Vol. 1001, pp. 942-951, Nov. 1988.

[16] B. Liu A. Zaccarin, "New fast algorithm for estimation for the estimation of block motion vectors," *IEEE Trans. on Circuit and System for Video Tech.*, Vol. 3, No. 2, pp. 148-157, Apr. 1993.

[17] H. Gharavi and M. Mills, "Block matching motion estimation algorithm - New results," *IEEE Trans. on Circuit and System.*, Vol. 37, No. 5, pp. 649-651, May. 1990.

[18] B. Liu, K. W. Chow and A. Zaccarin, "A simple method to segment motion field for video coding", *Proc. of SPIE conf. on Visual Commun.. and Image Processing '92,* vol. 1818, pp. 542-551, 1992.

[19] M. T. Orchard, "Predictive motion field segmentation for image sequence coding," *IEEE Trans. on Circuit and System for Video Tech.*, Vol. 3, No. 1, pp. 54-70, Feb. 1993.

[20] C. Auyeung and J. Kosmach, "Overlapped block motion compensation," *Proc. of SPIE Conf., on Visual Commun.. and Image Processing,* Vol. 1818, pp.561-572, Nov. 1992.

[21] H. Watnabe and S. Singhal, "Windowed motion compensation," *Proc. of SPIE Conf., on Visual Commun. and Image Processing,* Vol. 1605, pp. 582-589, Nov. 1991.

저 자 소 개

郭 成 一(正會員)

1969년 4월 19일생. 1991년 2월 경북대학교 전자공학과 졸업(공학사). 1994년 2월 경북대학교 대학원 전자공학과 졸업(공학석사). 1994년 ~ 현재 삼성전자 근무중. 주관심분야는 영상신호처리 등임.

黃 燦 植(正會員) 第 32卷 B編 第 2號 參照.
현재 경북대학교 전자공학과 부교수.

蔡 鍾 吉(正會員) 第 32卷 B編 第 2號 參照.
현재 통신사업부 특허청 근무중