論文99-36S-5-6

인접 블록의 중첩된 탐색 영역을 고려한 고속 블록 정합 알고리즘

(Fast Block Matching Algorithm Considering Overlapped Search Region of Neighboring Block)

李 法 基 *, 李 京 桓 *, 鄭 元 植 *, 崔 正 鉉 *, 李 健 一 *, 金 德 奎 *

(Bub Ki Lee, Kyeong Hwan Lee, Won Sik Cheong, Jung Hyun Choi, Kuhn Il Lee, and Duk Gyoo Kim)

요 약

본 논문에서는 움직임 추정시 인접된 블록간의 탐색 영역이 중첩됨을 이용하여 전역 탐색 기법과 동일한 성능을 유지하면서 계산량을 현저히 감소시킬수 있는 고속 움직임 추정 기법을 제안하였다. 제안한 기법에서는 현재 움직임 추정을 행하고자 하는 블록에 대한 탐색점 중에서 인접된 블록과 중첩되는 탐색점에 대하여서는 먼저 평균 절대 오차 (mean absolute difference; MAD)를 계산할 필요가 있을지에 대한 판별을 행한 뒤 MAD 계산이 필요한 경우에 대하여서만 MAD를 구한다. MAD 계산 여부에 대한 판별에는 현재 블록과 인접 블록간의 MAD와 인접 블록의 각 탐색점에 대한 MAD를 이용한다. 여기에 사용된 현재 블록과 인접 블록간의 MAD는 각 블록에 대하여 한번만 계산하면 되고, 인접 블록의 각 탐색점에 대한 MAD는 이미 구해져 있기 때문에 한번의 MAD 계산을 추가함으로써 탐색점 수를 현저히 감소시킬 수 있었다. 컴퓨터 모의 실험 결과로부터 제안한 방법이 전역 탐색 알고리즘과 동일한 성능을 유지하면서 많은 계산량의 감소을 얻을 수 있음을 확인 할 수 있었다.

Abstract

In this paper, we propose a fast motion estimation method that reduces the computational complexity remarkably with the same performance as full search method using the overlapped search region between neighboring blocks. At first, we calculate the mean absolute difference (MAD) after determining whether the computation of MAD is necessary among the overlapped blocks. Then we determine the necessity of computing MAD using the MAD's of current block and neighboring block. The computational complexity can be reduced by calculation MAD just once because the MAD for neighboring block is already known. Experimental results show that we can reduce the computational complexity considerably without any degradation of picture quality.

I. 서 론

일반적으로 동영상은 공간 및 시간적으로 중복되는 성분을 많이 가지고 있기 때문에 공간적인 처리만으로는 높은 압축율을 얻을 수 없다. 따라서 압축 효율을 높이기 위하여 시간적인 중복성을 제거하기 위한 움직임 보상 부호화를 수행한다. 이를 이용한 데이터의 압축 방법에서는 이전 프레임을 이용하여 움직임 추정 및 보상을 행한 뒤, 추정된 움직임 벡터 (motion vector)에 의해 보상된 영상과 원 영상의 차영상을 부호화 한다.[1]-[4] 움직임 추정 방법으로는 크게 화소 순환 알고리즘 (pel recursive algorithm; PRA)

* 正會員, 慶北大學校 電子電氣工學部
(School of Electronic and Electrical Engineering, Kyoungpook National University)

接受日字:1998年11月19日, 수정완료일:1999年4月26日

과 블록 정합 알고리즘 (block matching algorithm; BMA)이 있다.[5]-[7] 이때 BMA는 PRA에 비하여 움직임 추정에 소요되는 시간이 적으며 하드웨어 구현이 용이한 장점이 있어 움직임 보상 부호화에 널리 사용된다.

BMA는 영상을 임의의 작은 블록으로 나눈 후 한 블록 내의 모든 화소는 동일한 방향의 움직임을 갖는다고 가정하여 이전 프레임에서 설정된 탐색영역에서 모든 블록을 순차적으로 비교해 보는 전역 탐색 블록 정합 알고리즘 (full search block matching algorithm; FSBMA)을 기준으로 한다. 그러나 이 방법은 움직임 추정 오차 측면에서는 최적이지만 계산량이 많은 단점이 있다. 이와 같은 많은 계산량을 줄이기 위하여 움직임 벡터를 고속으로 추정 할 수 있는 여러 가지 고속 움직임 추정 기법들이 제안되었다. 이러한 고속 움직임 추정 기법으로는 블록 정합을 행할 탐색점 수를 줄이는 방법과 부표본화(subsampling를 통하여 정합 척도의 계산에 사용되는 블록내 화소의 수를 줄이는 방법이 있다.[6]-[12] 먼저, 탐색점 수를 줄임으로써 계산량을 감축시키는 방법으로는 2차원 로그 탐색 (two dimensional logarithm search; 2-D LOG search), 3단계 탐색 (three step search; TSS) 및 방향 추적 탐색 (conjugate direction search; CDS) 방법 등이 있다.[8]-[10] 이러한 방법들은 고속 움직임 추정이 가능하다는 장점을 가지지만, 전역 탐색을 행하지 않고 움직임 보상 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 탐색 영역의 일부에 대해서만 탐색을 행하므로 국부 최소(local optimal)에 빠질 수 있는 단점이 있다. Liu 등[11]은 전역 탐색을 수행하면서 움직임 추정을 고속으로 행하기 위한 방법으로 블록의 화소들을 수직, 수평방향으로 부표본화한 뒤 이를 이용하여 블록 정합을 행하는 방법을 제안하였다. 그러나 이 방법에서도 블록의 화소의 일부분만이 블록 정합의 계산에 사용되므로 전역 탐색 블록 정합 알고리즘에 비하여 성능이 떨어지는 단점을 가진다.

본 논문에서는 움직임 추정시 인접된 블록간의 탐색영역이 중첩됨을 이용하여 전역 탐색 알고리즘과 동일한 성능을 유지하면서 계산량을 현저히 감소시킬 수 있는 고속 움직임 추정 기법을 제안하였다. 먼저, 제안한 기법에서는 인접 블록과 움직임 탐색 영역이 중첩된 영역에 대하여 현재 블록에 대한 각 탐색점에서의 MAD의 최소 범위 및 최대 범위를 인접 블록의 MAD 및 현재 블록과 인접 블록의 차의 MAD를 이용하여 구하는 방법을 제시하였다. 그리고, 이렇게 구한 현재 블록의 평균 절대 오차의 최소 범위를 이용하여 인접 블록의 움직임 탐색 영역과 중첩되는 영역에서의 탐색점 수를 줄임으로써 계산량을 현저히 감소시켰다.

반대로 현재 블록의 움직임 탐색 영역이 인접 블록의 움직임 탐색 영역과 중첩되지 않는 모든 영역은 움직임 탐색을 수행하였다. 즉 먼저, 제안한 방법에서는 현재 블록과 인접 블록의 움직임 탐색 영역이 중첩되지 않은 영역에서 움직임 탐색을 수행한 후 왜곡이 최소가 되는 탐색점에서의 평균 절대 오차를 기준 MAD로 구한다. 또한 현재 블록과 인접 블록의 움직임 탐색 영역이 중첩되는 영역에 대하여서는 이 영역에 속한 탐색점에서 가질 수 있는 현재 블록의 MAD의 최소값을 현재 블록과 인접 블록간의 MAD와 인접 블록의 각 탐색점에 대한 MAD를 이용하여 구한다. 이렇게 구한 현재 블록의 MAD의 최소값이 앞에서 구한 기준 MAD 보다 크다면 중첩된 영역에 대하여서는 움직임 추정을 수행하지 않았다. 그러므로 전역 탐색 블록 정합 알고리즘과 같은 성능을 유지하면서 움직임 벡터의 추정을 위한 계산량을 줄일 수 있었다

제안한 방법의 성능을 평가하기 위하여 여러 가지의 동영상에 대하여 컴퓨터 모의실험을 수행하였다. 그 결과로부터 제안한 방법이 전역 탐색 알고리즘과 동일한 성능을 유지하면서 많은 계산량의 감소를 얻을 수 있음을 확인하였다.

Ⅱ. 블록 정합 알고리즘을 이용한 움직임 벡터 추정

블록 정합 알고리즘은 현재 프레임을 일정한 크기의 블록으로 나눈 후, 이전 프레임에서 설정된 탐색 영역에서 현재 프레임의 블록이 정합 척도가 최적인 위치를 찾는 것이다.

이때 사용되는 정합 척도로는 평균 자승 오차 (mean squared difference; MSD)와 비슷한 성능을 유지하면서도 계산량이 적고 하드웨어 구현이 용이한 MAD가 널리 이용되고 있다.[1] [4] 이때 MAD는 다음과 같다.

$$MAD_{(k,l)}(x,y) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | I_t(k+i, l+j) - I_{t-1}(k+x+i, l+y+j) | \quad (1)$$

여기서 t는 현재 프레임, $I_t(i,j)$는 i,j 좌표에서 화소의 휘도 값, (k,l)은 $N\times N$ 블록의 위치 좌표, 그리고 (x,y)는 탐색영역에서의 탐색점의 위치를 나타낸다.

(k,l)번째 블록의 움직임 벡터는 다음과 같이 구한다.

$$v(k,l) = \arg\min_{(x,y)} MAD_{(k,l)}(x,y) \quad (2)$$

그러므로 블록 정합 알고리즘은 정합 될 수 있는 모든 화소에 대해 MAD를 구하고 그 중 가장 작은 MAD값을 갖는 탐색 점 (x,y)의 값을 움직임 벡터로 결정하는 방법이다. 이때 $N\times N$ 화소의 블록을 고려할 때 탐색 영역의 크기는 움직임 벡터의 상하, 좌우의 최대 변위가 w 인 경우 $(2w+N)\times(2w+N)$가 되고, 탐색 영역에서 정합이 가능한 탐색점 수는 $(2w+1)^2$이 된다. 따라서 블록 정합 알고리즘은 탐색 영역 내에서 최적인 움직임 벡터를 추정할 수 있지만, 탐색점 수가 너무 많으므로 많은 계산량을 필요로 하는 단점이 있다. 이와 같은 단점을 줄이기 위해 탐색점을 줄이는 방법과 블록의 화소들을 표본화하는 방법 등이 연구되었다.[8]-[11] 이들 방법들은 간단하면서도 많은 계산량을 줄일 수 있다. 그러나 탐색 영역에서 탐색점을 줄이는 방법의 경우에는 전역 탐색을 행하지 않고 움직임 보상 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 탐색 영역의 일부에 대해서만 탐색을 행하므로 국부 최소에 빠질 수가 있고, 블록내의 화소들 중 표본화 된 화소들만 정합에 이용하는 방법의 경우에는 화소의 일부분만이 블록 정합의 계산에 사용되므로 전역 탐색 블록 정합 알고리즘에 비해 정확한 움직임 벡터를 추정할 수 없는 단점이 있다. 그러므로 추정 오차 측면에서 최적인 전역 탐색 블록 정합 알고리듬의 성능을 유지하면서 계산량을 줄이는 방법이 필요하다.

Ⅲ. 중첩된 탐색 영역을 고려한 고속 추정 알고리즘

본 논문에서는 인접 블록과 움직임 탐색 영역이 중첩된 영역에서 현재 블록 평균 절대 오차의 최소 및 최대 범위를 인접 블록의 평균 절대 오차 및 현재 블록과 인접 블록의 차의 평균 절대 오차를 이용하여 구한 뒤 이를 이용하여 탐색점 수를 줄임으로써 고속으로 움직임을 추정할 수 있는 방법을 제안하였다.

1. 평균 절대 오차의 최소 및 최대 범위

일반적으로 블록 정합 알고리즘에서의 탐색 영역의 크기는 블록 크기보다 크다. MPEG-2 TM(test model)에서는[13] CCIR601 표준 영상에서 블록의 크기가 16×16이면 탐색 영역의 범위는 수직과 수평 방향 각각 -16~+15를 권고하고 있다. 그러므로, 그림 1과 같이 현재 프레임에서 인접된 블록사이의 탐색 영역은 중첩되는 영역이 나타남을 알 수 있다. 그림1에서 블록 A는 현재 프레임의 (k,l)번째 $N\times N$ 블록, 블록 B는 블록 A와 현재 프레임의 수직방향의 인접 블록인 $(k-N,l)$번째 블록, R_1과 R_2는 각각 블록 A와 블록 B의 움직임 탐색 영역, 그리고 R_3은 블록 A와 블록 B의 움직임 탐색 영역이 중첩된 영역을 나타낸다. 본 논문에서는 블록 A의 MAD의 최소 및 최대 범위를 인접된 블록 B와 움직임 탐색 영역이 중첩되는 성질을 이용하여 구하는 방법을 제안하였다. 블록 A의 MAD의 최소 및 최대 범위는 다음과 같이 구하였다.

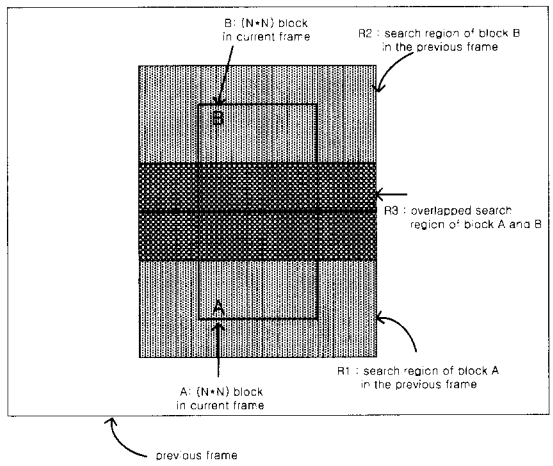


그림 1. 중첩된 움직임 탐색 영역
Fig. 1. Overlapped motion search region.

그림 1 에서 블록 A 및 블록 B 화소들의 휘도 값을 $N\times N$ 행렬로 나타내면 다음과 같이 표현 할 수 있다

$$A=\begin{bmatrix} I_t(k,l), & \cdots & I_t(k,l+N-1) \\ \vdots & & \vdots \\ I_t(k+N-1,l), & \cdots & I_t(k+N-1,l+N-1) \end{bmatrix} \quad (3)$$

$$B=\begin{bmatrix} I_t(k-N,l), & \cdots & ,I_t(k-N,l+N-1) \\ \vdots & & \vdots \\ I_t(k-1,l), & \cdots & I_t(k-1,l+N-1) \end{bmatrix} \quad (4)$$

그리고, 이전 프레임의 중첩된 영역 R_3의 탐색 점 (x,y)에서 정합 될 블록 $S(x,y)$, 즉 이전 프레임에서의 $(k+x,l+y)$번째 블록을 $N\times N$ 행렬로 나타내면 다음과 같다.

$$S(x,y)=\begin{bmatrix} I_{t-1}(k+x,l+y), & \cdots & ,I_{t-1}(k+x,l+y+N-1) \\ \vdots & & \vdots \\ I_{t-1}(k+x+N-1,l), & \cdots & I_{t-1}(k+x+N-1,l+y+N-1) \end{bmatrix} \quad (5)$$

이때, $(x,y)\in R_3$ 이다.

블록 A와 블록 $S(x,y)$의 정합 및 블록 B와 블록 $S(x,y)$의 정합으로 생긴 오차 블록은 다음과 같이 표현 할 수 있다.

$$U(x,y)=A-S(x,y) \quad (6)$$

$$V(x,y)=B-S(x,y) \quad (7)$$

그리고, 인접 블록 A, B간 차의 오차 블록은 다음과 같이 나타낼 수 있다.

$$W=A-B \quad (8)$$

이때, 식 (6), 식(7), 그리고 식(8)의 관계에서 아래와 같은 식을 구할 수 있다.

$$U(x,y)=W+V(x,y) \quad (9)$$

식(9)식을 보면 블록 A와 블록 S의 오차 블록 $U(x,y)$는 블록 B와 블록 $S(x,y)$의 오차 블록 $V(x,y)$와 인접 블록 A, B간 오차 블록 W의 합으로 표현됨을 볼 수 있다. 그러나 우리가 최종적으로 구하고자하는 것은 블록 A와 블록 $S(x,y)$간의 MAD이므로 식 (8)을 이용하여 MAD를 구한다면 식 (10)의 과정으로 인하여 계산량의 이득을 얻을 수 없다. 그러므로 움직임 추정을 위한 계산량을 줄이기 위하여 블록 A의 최소값 MAD와 최대값 MAD을 다음과 같이 구하였다.

임의의 $N\times N$ 행렬 M의 L_1 놈 (norm)은 다음과 같이 정의된다.

$$\| M \| = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | m_{ij} | \quad (10)$$

그리고, 삼각부등식을 적용하면 다음과 같이 표현할 수 있다.

$$| \| M_1 \| - \| M_2 \| | \le \| M_1+M_2 \| \le \| M_1 \| + \| M_2 \| \quad (11)$$

이를 식(9)의 우변에 적용하면 다음과 같다

$$| \| W \| - \| V(x,y) \| | \le \| W+V(x,y) \| \le \| W \| + \| V(x,y) \| \quad (12)$$

이때, $\| U(x,y) \| = \| W+V(x,y) \|$ 이므로 식 (11)은 다음과 같이 표현 할 수가 있다.

$$| \| W \| - \| V(x,y) \| | \le \| U(x,y) \| \le \| W \| + \| V(x,y) \| \quad (13)$$

여기에 N의 제곱으로 양변을 나누면 식 (12)는 다음과 같다.

$$| \frac{1}{N^2} \| W \| - \frac{1}{N^2} \| V(x,y) \| | \le \frac{1}{N^2} \| U(x,y) \| \le \frac{1}{N^2} \| W \| + \frac{1}{N^2} | V(x,y) | \quad (14)$$

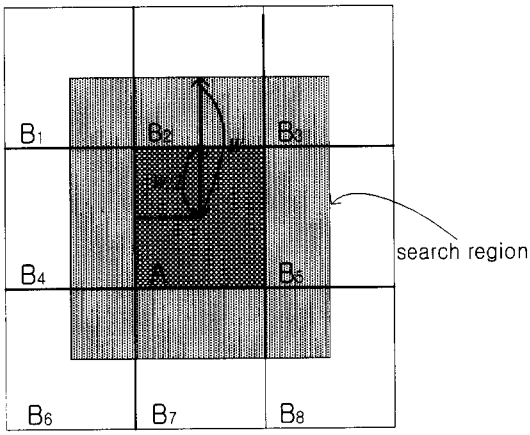
식 (14)식을 보면 $\frac{1}{N^2}\| U(x,y) \|$ 는 블록 A의 MAD, $\frac{1}{N^2}\| W \|$ 는 인접 블록 차의 MAD, 그리고 $\frac{1}{N^2}\| V(x,y) \|$ 는 블록 B의 MAD를 의미한다.

즉 R3의 탐색점 (x,y)에서 블록 A의 MAD의 최소값은 블록 B의 MAD와 블록 A, B간의 MAD의 차의 절대값으로부터 구할 수 있다. 또한 블록 A의 MAD의 최대값은 블록 B의 MAD와 블록 A, B간의 MAD의 합으로 구할 수 있음을 알 수 있다. 이때 식 (14)에서 R_3의 모든 탐색점 (x,y)에서 블록 A의 MAD의 최소값 및 최대값은 인접 블록 탐색시에 계산된 MAD를 이용하면 $V(x,y)$ 에 대한 계산량은 필요 없고, W는 한번의 인접 블록 차의 MAD를 계산하면 되므로 매우 적은 계산량으로 구할 수 있다. R_3에서 이전 프레임 블록과 현재 블록의 정합 여부는 식 (14)를 이용하면 MAD 최소 및 최대값을 알 수 있으므로 최소값이 낮은 탐색점에 대해서만 판단하면 될

것이다. 그리고 이것은 수평, 수직 및 대각선의 각 방향 인접 블록에 대해서도 동일하게 적용 할 수 있다.

2. 중첩된 탐색영역에서의 탐색점 줄임

인접블록간의 움직임 탐색 영역이 중첩된 영역에서 현재 블록의 MAD의 최소값은 인접 블록간의 차와 인접 블록의 MAD로 알 수 있다. 그러므로 제안한 방법에서는 현재 블록에 대해 움직임 탐색 영역이 중첩되지 않은 모든 탐색 점에서 블록 정합을 수행하고, 반대로 움직임 탐색 영역이 중첩되는 탐색 점에서는 인접 블록의 탐색 영역의 탐색 점에서의 MAD와 인접 블록간의 차의 MAD를 이용하여 현재 블록의 MAD의 최소값을 구하여 블록의 정합 여부를 판단하였다.



A : current block
Bi: neighbor block

그림 2. 제안한 알고리즘을 위한 인접 블록
Fig. 2. The neighbor blocks for proposed algorithm.

이와 같은 인접 블록의 탐색 영역의 탐색 점에서의 MAD와 인접 블록간의 차의 MAD를 이용하는 제안한 방법에서는 그림 2에서와 같이 현재 프레임의 (k, l) 번째 블록 A는 수직, 수평방향으로 움직임 탐색 영역을 $-w/2 \le x, y \le w/2$ 로 줄여 모든 탐색 점에서 대해서 블록정합을 계산하여 최소의 왜곡이 되는 기준 평균 절대 오차 $MAD^{Ref.}_{(k,l)}$를 다음과 같이 구한다.

$$MAD^{Ref.}_{(k,l)} = \arg\min_{(x,y)} MAD_{(k,l)}(x,y) \quad (15)$$
$$where, \quad -w/2 \le x, y \le w/2$$

그리고, 인접 블록과 겹치는 움직임 탐색 대상인 $-w < x, y < w$ 중에서 $|w/2| < x, y < |w|$ 부분은 인접 블록 B_i 의 줄여진 움직임 탐색 영역과 중복된 부분이 생긴다, 그러므로 이 부분에서는 블록 B_i의 움직임 탐색시에 구한 MAD와 블록 A, B_i간의 차의 MAD를 계산하여 블록 A의 최소값을 갖는 MAD를 계산한 후 아래와 같이 다음과 같이 비교한다.

$$IF \left(MAD^{Ref.}_{(k,l)} < \left| \frac{1}{N^2} \| V_i(x,y) \| - \frac{1}{N^2} \| W_i \| \right| \right.$$
$$Don't\ perform\ block\ matching. \quad (16)$$
$$ELSE \{$$
$$Find\ MAD(x,y)$$
$$IF(MAD(x,y) < MAD^{Ref.}_{(k,l)})$$
$$MAD^{Ref.}_{(k,l)} = MAD(x,y)$$
$$\}$$
$$where, \quad |w/2| < x, y < |w|$$

이때, $W_i = A - B_i$, $V_i(x,y) = B_i - S(x,y)$이고, $MAD(x,y)$는 현재 탐색점에서 MAD이다.

식 (16)을 보면 $\frac{1}{N^2} \| V_i(x,y) \|$ 는 인접 블록 B_i의 탐색점에서의 MAD이고, $\frac{1}{N^2} \| W_i \|$ 는 블록 A, B_i 차의 MAD를 의미하므로 움직임 탐색 영역이 중첩된 영역에서는 블록 A의 최소값의 MAD를 알 수 있다. 그러므로 제안한 방법은 식(16)에서와 같이 움직임 탐색 영역이 중첩된 영역의 탐색 점에서는 블록 A의 최소값 MAD가 기준 평균 절대 오차 $MAD^{Ref.}_{(k,l)}$보다 크면 블록 정합을 수행할 필요가 없고, 블록 A 의 최소값 MAD가 기준 평균 절대 오차 $MAD^{Ref.}_{(k,l)}$보다 작으면 블록 정합을 수행하여 $MAD(x,y)$를 구하여 이와 비교한다. 이때 $MAD(x,y)$가 $MAD^{Ref.}_{(k,l)}$보다 작으면 $MAD^{Ref.}_{(k,l)}$는 $MAD(x,y)$갱신되고, 이것의 최종값을 지닌 탐색점이 최적 움직임 벡터로 결정된다. 이와 같이, 현재 블록의 움직임 탐색 영역의 모든 탐색 점에서 가장 작은 MAD를 갖는 탐색 점 (x,y)의 값을 결정함으로써 제안한 방법은 전역 탐색 방법과 동일한 성능을 유지하면서 계산량을 줄일 수 있다.

3. 탐색점 줄임에 의한 계산량 감소

전역 탐색 블록 정합 알고리즘의 계산량은 이전 프레임의 각 탐색 점에서 MAD를 계산하므로 식(1)를 보면 $2N^2$번의 덧셈, 뺄셈에 대한 계산량이 필요하다.

그러므로 현재 프레임의 블록의 전체 개수가 T이고, 움직임 탐색 영역이 상하, 좌우의 변위가 최대 u이면 전역 탐색 블록 정합 알고리즘 위한 총 계산량은 다음 식과 같다.

$$C^{FSBMA} = 2T(2w+1)^2N^2 \quad (17)$$

제안한 방법의 계산량은 식 (15)의 계산량과 식 (16)의 계산량의 합으로 볼 수 있다. 식 (15)에서와 같이 현재 프레임의 블록들은 움직임 탐색 영역을 $-w/2 \le x, y \le w/2$로 줄여서 블록 정합을 수행하므로 전역 블록 정합 알고리즘에 비해 계산량은 1/4이 된다. 식 (16)에서 현재 블록의 최소값 MAD의 계산량은 인접 블록의 움직임 탐색 영역의 탐색 점에서 블록 정합으로 구한 MAD을 이용하므로 $V_i(x,y)$의 계산량은 필요 없고, 인접 블록 간의 차의 MAD인 W_i의 부가적인 계산량이 필요하다. 또한 움직임이 중첩된 탐색 영역들에서 현재 프레임 블록들의 최소값 MAD가 $MAD^{Ref.}_{(k,l)}$보다 작은 탐색 점에서 블록 정합을 위한 계산량이 필요하다. 그러므로 제안한 방법의 계산량은 다음과 같이 표현할 수 있다.

$$C^{proposed} = 2T((2w+1)^2/4)N^2 + 2PN^2 + 8TN^2 \quad (18)$$

여기서 P는 식(16)에서 움직임이 중첩된 탐색 영역들에서 현재 프레임 블록들 MAD의 최소값이 $MAD^{Ref.}_{(k,l)}$보다 작은 탐색 점의 총 개수이다.

즉 (18)에서 $2T((2w+1)^2/4)N^2$은 현재 프레임 전체 블록들에 대한 식 (15)의 계산량, $2PN^2$은 움직임이 중첩된 탐색 영역들에서 현재 프레임 전체 블록들 MAD 의 최소값이 $MAD^{Ref.}_{(k,l)}$보다 작은 모든 탐색 점에서 MAD를 구하기 위한 계산량, 그리고 $8TN^2$는 현재프레임 인접 블록들 차의 MAD를 구하기 위한 부가적인 계산량을 의미한다. 그러나 $8TN^2$의 부가적인 계산량은 $2T((2w+1)^2/4)N^2 \gg 8TN^2$ 이므로 매우 작음을 알 수 있다. 그리고 식(16)의 뺄셈 및 비교 연산자는 이에 비해 너무 적은량이기 때문에 식(18)에서 무시하였다.

식(17) 및 식 (18)에서 전역 탐색 알고리즘 비해 제안한 알고리즘의 계산량은 움직임이 중첩된 탐색 영역에서 현재 블록의 최소값이 기준 평균 절대 오차 $MAD^{Ref.}_{(k,l)}$보다 큰 탐색점 수가 많을수록 감소됨을 알 수 있다.

Ⅳ. 실험결과 및 고찰

본 논문에서 제안한 움직임 추정 기법의 성능을 평가하기 위하여 컴퓨터 모의 실험을 수행하였다. 본 실험에서는 FOOTBALL, FLOWER GARDEN, MOBILE 영상을 각각 30프레임을 사용하였다. 이들 영상은 720×480 크기를 가지며 8비트로 양자화 되어 있는 것이다. 제안한 방법의 실험에서는 블록의 크기는 16×16, 탐색 영역의 크기는 수직과 수평방향으로 -16~15, -20~19 및 -24~23에서 성능을 비교하였다. 여기에 사용한 FOOTBALL은 운동장의 배경이 전체적으로 복잡하고, 물체 및 카메라의 움직임이 매우 빠른 영상이며, FLOWER GARDEN은 꽃밭과 같은 복잡한 부분이 많이 존재하고, 정지된 물체에 카메라가 빠르게 움직이는 영상이며, 그리고 MOBILE은 고정된 배경에 기차가 움직이는 영상으로써 움직임이 거의 없는 배경과 물체의 움직임이 있는 영역이 혼재 하는 영상으로 볼 수 있다. 그리고 정합의 척도로는 움직임 벡터를 찾기 위해서 계산량이 빠른 평균 절대 오차를 사용하였다.

표 Ⅰ은 w가 -16~15인 경우에 움직임이 중첩된 영역에서 전역 탐색 방법과 제안한 방법의 블록 정합될 평균 탐색점 수의 결과를 나타내었다. 표 Ⅰ에서 알 수 있듯이, 제안한 방법은 전역 탐색 알고리즘에 비해 MOBILE 영상의 경우 68.8%, FLOWER GARDEN 영상의 경우 62.9%, 그리고 FOOTBALL 영상의 경우 47.2%의 정도로 탐색점 수의 감소를 얻을 수 있다. 그림 3은 w가 -16~15인 경우에 경우 실험 영상 프레임에 따른 움직임이 중첩된 영역에서 제안한 방법이 전역 탐색 방법에 비해 탐색점 수의 감소의 결과를 나타내었다. 그림 3을 보면 움직임과 복잡한 부분이 상대적으로 적은 MOBILE영상이 탐색점 수의 감소가 가장 큼을 알 수 있다. 그리고 영상의 복잡한 부분은 비슷하지만 FLOWER GARDEN영상보다 움직임이 빠른 FOOTBALL영상이 탐색점 감소가 가장 작음을 확인 할 수 있었다.

제안한 방법이 전역 탐색 알고리즘에 비해 가지는 계산량의 감소 정도 F는

$$Reduction\ rate = \left(1 - \frac{C^{proposed}}{C^{FSBMA}}\right) \times 100\ \% \quad (19)$$

와 같이 표현 될 수 있다.

표 1. 중첩된 움직임 영역에서 탐색점 수 비교

Table 1. The comparison of search point number in overlapped motion search region.

Methods \ Sequences	MOBILE	FLOWER GARDEN	FOOTBALL
FSBMA	768	768	768
Proposed	255	285	406

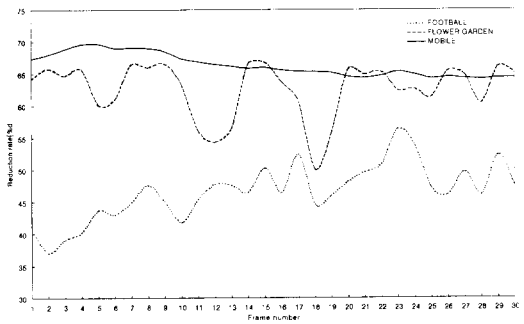


그림 3. 각 프레임에 대한 탐색점 수의 감소율

Fig. 3. The reduction rate of search point in each frame.

MOBILE, FLOWER GARDEN 및 FOOTBALL 영상에 대한 30 프레임에 대한 평균 계산량 감소를 표 II에 나타내었다. 표 II의 실험 결과를 보면, w가 -16~15인 경우에 전역 탐색 알고리즘에 비해 제안한 방법은 MOBILE 영상의 경우 50.0%, FLOWER GARDEN 영상의 경우 47.1%, 그리고 FOOTBALL 영상의 경우 36.2%의 정도의 계산량의 감소를 얻을 수 있다. 움직임이 적고 복잡한 부분이 적은 영상일수록 계산량의 감소가 많음을 확인 할 수 있다. 이는 움직임이 적고 단순한 영상일수록 현재 블록의 기준 평균 절대 오차 $MAD^{Ref.}_{(k,l)}$가 작아질 확률이 높기 때문에 식(16)에서 보면 블록 정합을 수행할 확률이 적어짐을 알 수 있다. 또한 표 II에서 알수 있듯이, 탐색 영역의 크기가 클수록 계산량의 감소가 많음을 확인할 수 있다. 이것은 제안한 방법 탐색 영역의 크기가 클수록 움직임 탐색 영역이 중첩되는 부분이 중첩되지 않는 영역보다 많기 때문이다. 그림 4에서는 w가 -16~15인 경우에 실험 영상 프레임 따른 계산량의 감소를 나타내었다. 그림 4에서 보면 MOBILE 및 FLOWER GARDEN 영상에 대해 각각 프레임에 따라 48.6~52.4% 및 38.0~50.1%의 계산량의 감소를 얻을 수 있다. 또한 FOOTBALL의 경우에는 프레임에 따라 24.2~46.0%정도로 계산량의 감소를 얻을 수 있었다.

표 2. 제안한 방법의 계산량 감소율

Table 2. The reduction rate of computation by proposed method.

Performance	Sequences	FSBMA	Proposed method		
			w= -16~15	w= -20~19	w= -24~23
Reduction rate	MOBILE	100	50.0	59.30	62.97
	FLOWER GARDEN	100	47.1	57.67	61.95
	FOOTBALL	100	36.2	42.31	47.63

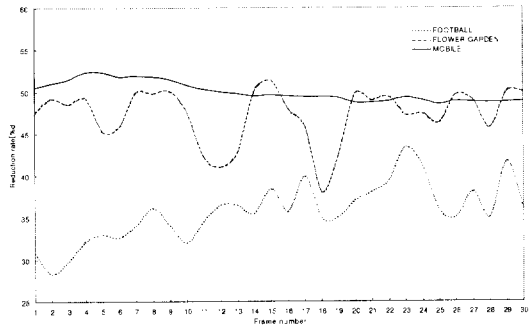


그림 4. 각 프레임에 대한 계산량 감소율(w=-16~15)

Fig. 4. The reduction rate of computation in each frame.(w= -16~15).

이상의 결과로부터 제안한 블록 정합 알고리즘은 전역 탐색 블록 정합 알고리즘과 동일한 성능을 유지하면서 많은 계산량의 감소를 얻을 수 있으며, 특히 움직임이 적고 복잡한 부분이 적은 영상일수록 성능이 우수함을 알 수 있다.

V. 결 론

본 논문에서는 인접된 블록간의 움직임 탐색 영역의 중첩을 이용하여 전역 탐색 블록 정합 알고리즘의 동일한 성능을 유지하면서 계산량을 감소 시킬 수 있는 움직임 추정 기법을 제안하였다. 제안한 방법에서는 인접 블록과 움직임 탐색 영역이 중첩된 영역에서 현재 블록 평균 절대 오차의 최소값 및 최대값을 인접 블록의 평균 절대 오차 및 현재 블록과 인접 블록의 차의 평균 절대 오차를 이용하여 구하는 방법을 제시

하였다. 제안한 방법은 현재 블록과 인접 블록의 움직임 탐색 영역이 중첩되지 않은 영역에서 움직임 탐색을 수행한 후 왜곡이 최소가 되는 기준 평균 절대 오차를 구한다. 반대로, 현재 블록과 인접 블록의 움직임 탐색 영역이 중첩되는 영역에서는 현재 블록의 평균 절대 오차의 최소값이 기준 평균 절대 오차 보다 크면 움직임 탐색을 수행하지 않아 계산량을 감소시킬 수 있었다.

제안한 방법의 성능을 평가하기 위하여 컴퓨터 모의실험을 수행하였다. 이들 결과로부터 제안한 방법은 전역 탐색 알고리즘에 비해 *PSNR* 측면에서는 동일한 성능을 유지하면서 많은 계산량을 줄일 수 있었다.

참 고 문 헌

[1] A. K. Jain, "Image data compression: A review," *Proc. of IEEE*, vol. 69, no. 3, pp. 349-389, March 1981.

[2] A. N. Netravali and J. O. Limb, "Picture coding: A review," *Proc. of IEEE*, vol. 68, no. 3, pp. 366-406, March 1980.

[3] Motion Picture Experts Group, " MPEG Committee Draft," *ISO-IEC JTC1/SC29/WG11/602*, Nov. 1993.

[4] H. G. Musmann, P. Pirch, and H. J. Grallert, "Advances in picture coding," *Proc of IEEE*, vol. 73, no. 4, pp. 523-548, April 1985.

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

[6] S. Zafar, Y. Q. Zhang, and J. S. Baras, "Predictive block-matching motion estimation for TV coding-Part I : Inter-block prediction," *IEEE Trans. on Broadcasting*, vol. 37, no. 3, pp. 97-101, Sep. 1991.

[7] A. N. Netravali and J. D. Robbins, "Motion compensated television coding : Part I," *Bell Syst. Tech. J.*, vol. 58, no. 3, pp. 361-670, March 1979.

[8] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing," *in Proc. Nat. Telecommun. Conf*, LA, Nov. 29_Dec. 3, pp. G.5.3.1-5.3.5, 1981.

[9] J. R. Jain and A. K. Jain, "Displacement measurement and it's application in interframe coding," *IEEE Trans. Commun.* vol. COM-29, pp. 1799-1808, July 1981.

[10] R. Strinivasan and K. R. Rao, "Predictive Coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. COM-33, pp. 1011-1014, Sept.1985.

[11] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vector," *IEEE Trans. Circuits Syst. Video Technol.*, vol 3, no. 2, Apr. 1993.

[12] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. on Image Processing*, vol. 4. No. 1, Jan. 1995

[13] Motion Picture Experts Group, "MPEG Test model 5 Draft revision 2," *ISO-IEC JTC1/SC29/WG11/N0400*, April 1993.

저 자 소 개

李 法 基(正會員) 第 35卷 第 5號 參照

崔 正 鉉(正會員) 第 35卷 第 5號 參照

李 京 桓(正會員) 第 35卷 第 5號 參照

李 健 一(正會員) 第 34卷 第 9號 參照

鄭 元 植(正會員) 第 34卷 第 9號 參照

金 德 奎(正會員) 第 35卷 第 5號 參照