논문 99-24-8B-13 한국통신학회논문지 '99-8 Vol.24 No.8B

이웃 블록과 중첩된 탐색영역에서의 MAD 분포 및 부표본화를 이용한 고속 블록 정합

정회원 이 법 기*, 정 원 식*, 이 경 환*, 최 정 현*, 김 경 규*, 김 덕 규*

Fast Block Matching Algorithm Using The Distribution of Mean Absolute Difference at The Search Region Overlapped with Neighbor Blocks and Subsampling

Bub-ki Lee*, Won-Sik Cheong*, Kyeong-Hwan Lee*, Jung-Hyun Choi*, Kyoung-Kyoo Kim*, and Duk-Gyoo Kim* *Regular Members*

요 약

본 논문에서는 이웃 블록과 중첩된 움직임 탐색영역에서의 MAD 분포 및 블록내 화소의 부표본화를 이용한 두 가지 고속 움직임 추정 방법을 제안하였다. 제안한 두 가지 방법에서는 이웃 블록과 중첩된 영역에서 현재 블록 MAD의 최소 및 최대 범위를 구하여 이용하였으며, 이때 사용되는 MAD의 최소 및 최대 범위는 이웃 블록의 MAD와 현재 블록과 이웃 블록간의 MAD를 이용하여 구하였다. 또한, 블록 정합은 블록내 화소를 부표본화한 뒤 행하였다. 제안한 첫 번째 방법에서는 블록 MAD의 최소 범위를 이용하여 이웃 블록과 움직임이 중첩되지 않는 영역에서 구한 기준 MAD가 블록 MAD의 최소 범위보다 큰 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다. 또한, 본 논문에서는 계산량을 크게 줄이기 위한 방법으로 현재 블록 MAD의 분포 특성을 이용한 고속 움직임 추정 방법을 제안하였다. 이 방법에서는 이웃 블록과 중첩된 각 탐색점에서의 실제 MAD는 MAD의 최소 및 최대 범위 사이에 존재하게된다는 것을 이용하여, 실제 MAD 분포의 확률적인 모델을 이용하여 고속으로 움직임을 추정하였다. 제안한 방법의 성능을 평가하기 위한 컴퓨터 모의 실험결과로부터 제안한 방법이 움직임 추정 오차 측면에서 우수한 성능을 유지하면서도 계산량을 현저히 줄일 수 있음을 확인할 수 있었다.

ABSTRACT

In this paper, we propose two fast block matching algorithm using the distribution of mean absolute difference (MAD) at the search region overlapped with neighbor blocks and pixel subsapmling. The proposed methods use the lower and upper bound of MAD at the overlapped search region which is calculated from the MAD of neighbor block at that search position and MAD between the current block and neighbor block. In the first algorithm, we can reduce the computational complexity by executing the block matching operation at the only necessary search points. That points are selected using the lower bound of MAD. In the second algorithm, we use the statictical distribution of actual MAD which exists between the lower bound and upper bound of MAD. By using the statistical distribution of actual MAD, we can significantly reduce the computational complexity for motion estimation. after striking space key 2 times.

* 경북대학교 전자전기공학부(khlee@palgong.kyungpook.ac.kr)
논문번호 : 99071-0225, 접수일자 : 1999년 2월 25일

I. 서 론

동영상 압축 (video compression)은 중요한 디지털 기술의 하나로서, 고선명 TV (high definition TV; HDTV), 주문형 비디오 (video on demand; VOD), 영상 회의 (video conferencing) 등의 다양한 응용 분야에 사용된다. 특히, 최근에는 초 저속 (low bit rate) 동영상 전송을 위한 움직임 보상 동영상 압축 (motion compensated video compression)에 대하여 많은 연구가 진행되고 있다. 여기서 움직임 추정 및 보상은 연속된 프레임간의 시간적인 중복성을 제거함으로써 초 저속 전송을 위한 높은 압축율을 얻는데 핵심적인 역할을 담당하고 있으며, 동영상 부호화를 위한 표준인 H.261[1], H.263[2], MPEG[3],[4] 등에 사용되고있다.

움직임 추정 및 보상 기법으로는 수행시간이 비교적 적게 소요되고, 하드웨어 구현이 용이한[5] 블록 정합 알고리듬 (block matching algorithm; BMA)[6]이 많이 사용되고 있다. 이 방법에서는 입력 영상을 임의의 작은 블록으로 나눈 뒤, 블록내의 모든 화소들은 같은 방향으로 평행 이동한다는 것을 가정하여 이전 프레임의 탐색영역에서 정합 척도가 최적인 블록을 찾는다. BMA를 이용하여 움직임을 추정하는 경우에 가장 좋은 성능을 얻을 수 있는 방법은 이전 프레임의 탐색영역의 모든 탐색점에 대하여 탐색을 행하는 전역 탐색 블록 정합 알고리듬 (full search block matching algorithm; FSBMA)이다. 이 방법에서는 모든 탐색점에 대하여 탐색을 행하여 정합 척도가 최적인 블록을 찾기 때문에 움직임 추정 오차 측면에서 최적인 움직임 벡터를 얻을 수 있지만 계산량이 많은 단점이 있다.

이와 같은 단점을 줄이기 위하여 움직임 벡터를 고속으로 추정할 수 있는 여러 가지 고속 움직임 추정 기법들이 제안되었다. 대표적인 고속 BMA 방법으로는 2차원 로그 탐색 (two dimensional logarithm search; 2-D LOG), 3단계 탐색 (three step search; TSS), 교차 탐색 (cross search) 알고리듬 등이 있으며[6]-[10],[13], 이 방법들 중 간단하면서도 성능이 좋은 3단계 탐색 알고리듬이 가장 많이 사용된다. 그러나, 이 방법들은 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 탐색영역의 일부분에 대하여서만 탐색을 행하므로, 계산량은 줄일 수 있었지만 국부 최소 (local minimum)에 빠질 수 있는 단점을 가진다. 즉, 이 방법들은 움직임 추정의 정확성을 어느 정도 희생함으로써 계산량의 감소를 얻는다.

이러한 문제점을 해결하기 위하여 부표본화 (subsampling)를 통하여 블록내의 화소수를 줄임으로써, 모든 탐색점에 대하여 탐색을 행하면서도 고속으로 움직임 추정을 행할 수 있는 방법이 제안되었다[11],[12]. Liu 등[11]이 제안한 부표본화를 이용한 방법에서는 모든 탐색점에 대하여 탐색을 행하므로 TSS 알고리듬에 비하여 적은 움직임 추정 오차를 가진다. 그러나 이 방법은 TSS 알고리듬에 비하여 두 배 이상의 많은 계산량을 필요로하는 단점이 있다. 그러므로 움직임 추정 오차 측면에서 최적인 전역 탐색 블록 정합 알고리듬에 가까운 성능을 유지하면서도 계산량을 크게 줄일 수 있는 고속 움직임 추정 방법이 필요하다.

본 논문에서는 이웃 블록과 중첩된 움직임 탐색영역에서의 MAD 분포 및 블록내 화소의 부표본화를 이용한 두가지 고속 움직임 추정 방법을 제안하였다. 제안한 두 가지 방법에서는 이웃 블록과 중첩된 영역에서 현재 블록 MAD의 최소 및 최대 범위를 구하여 이용하였으며, 이때 사용되는 MAD의 최소 및 최대 범위는 이웃 블록의 MAD와 현재 블록과 이웃 블록간의 MAD를 이용하여 구하였다. 또한, 블록 정합은 블록내 화소를 부표본화한 뒤 행하였다.

본 논문에서 제안한 첫 번째 방법에서는 블록 MAD의 최소 범위를 이용하여 이웃 블록과 움직임이 중첩되지 않는 영역에서 구한 기준 MAD가 블록 MAD의 최소 범위보다 큰 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다. 이 방법에서는 움직임 추정 오차 측면에서 Liu 등이 제안한 방법과 동일한 성능을 가지면서도 계산량을 현저하게 줄일 수 있었다. 그리고, TSS 알고리듬에 비하여서는 움직임 추정 오차 측면에서는 월등한 성능을 나타내지만 계산량은 약간 많았다. 그러므로 본 논문에서는 계산량을 현저히 줄일 수 있는 방법으로 현재 블록 MAD의 분포 특성을 이용한 고속 움직임 추정 방법을 제안하였다. 이웃 블록과 중첩된 각 탐색점에서의 실제 MAD는 MAD의 최소 및 최대 범위 사이에 존재하게되므로, 본 논문에서는 실제 MAD가 MAD의 최소 및 최대 범위 사이에 어떻게 분포하는지를 조사한 뒤, 이 분포를 이용하여 고속으로 움직임을 추정하였다. 이 방법에서는 TSS 알고리듬에 비하여 월등히 적은 움직임 추정 오차를 얻을 수 있었을 뿐만 아니라 계

산량 역시 현저히 줄일 수 있었다.

제안한 방법의 성능을 평가하기 위한 컴퓨터 모의 실험결과로부터 제안한 방법이 움직임 추정 오차 측면에서 우수한 성능을 유지하면서도 계산량을 현저히 줄일 수 있음을 확인할 수 있었다.

Ⅱ. 블록 정합 알고리듬을 이용한 움직임 추정

BMA는 현재 입력 프레임을 일정한 크기의 블록으로 나눈 후, 이전 프레임에서 설정된 탐색영역에서 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. 이때, MAD는

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

와 같다. 여기서, N은 블록의 수직 및 수평 방향의 크기, $I_t(i,j)$는 현재 프레임의 (i,j) 좌표에서 화소의 휘도 값, $I_{t-1}(i,j)$는 이전 프레임의 (i,j) 좌표에서 화소의 휘도 값, (k,l)은 블록의 위치 좌표, 그리고 (x,y)는 탐색영역에서의 탐색점의 위치를 나타낸다. 이를 이용하여 (k,l)번째 블럭의 움직임 벡터는

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

와 같이 구한다.

BMA를 이용하여 움직임 추정을 행하는 경우에 가장 좋은 성능을 얻을 수 있는 방법은 전역 탐색 블록 정합 알고리듬으로써, 이 방법은 정합 될 수 있는 모든 화소에 대해 MAD를 구하고 그 중 가장 작은 MAD 값을 갖는 탐색점 (x,y)의 값을 움직임 벡터로 결정하는 것이다. 이때 $N \times N$ 화소 크기의 블록의 움직임 벡터의 수직 및 수평 방향의 최대 범위가 w 인 경우에 탐색영역에서 정합을 행하여야 할 탐색점수는 $(2w+1)^2$이고, 한 프레임에 대하여 FSBMA를 위한 계산량은

$$C_{FSBMA} = 2T(2w+1)^2N^2 \quad (3)$$

와 같다. 여기서, T는 한 프레임의 블록의 개수이다. 따라서 전역 탐색 블록 정합 알고리즘은 탐색영역 내에서 움직임 추정 오차 측면에서 최적인 움직임 벡터를 추정할 수 있지만, 탐색점 수가 너무 많으므로 계산량이 많은 단점이 있다.

이와 같은 단점을 줄이기 위하여 탐색점을 줄이는 방법과 블록의 화소들을 부표본화한 뒤 블록 정합을 행하는 방법 등이 연구되었다. 그러나 탐색영역에서 탐색점을 줄이는 방법의 경우에는 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 전역 탐색을 행하지 않고 탐색영역의 일부에 대해서만 탐색을 행하므로, 계산량은 크게 줄일 수 있었지만 국부 최소에 빠질 수 있는 단점을 가진다. 즉, 움직임 추정의 정확성을 어느 정도 희생함으로써 계산량의 감소를 얻기 때문에 움직임 추정 오차가 커지며 특히, 이 방법에서 사용된 가정이 잘 맞지 않는 경우에는 움직임 추정 오차가 매우 커지는 단점이 있다.

그리고 Liu 등이 제안한 블록내의 화소들 중 부표본화 된 화소들만을 정합에 이용하는 방법의 경우에는 모든 탐색점에 대하여 탐색을 행하므로 대표적인 탐색점을 줄이는 방법의 하나인 TSS 알고리듬에 비하여 적은 움직임 추정 오차를 가지며, 블록내의 픽셀 값들이 비슷한 평탄한 블록의 경우에는 우수한 움직임 추정 성능을 나타낸다. 그러나 에지를 포함하고 있는 블록이나, 복잡한 블록의 경우에는 부표본화에 의한 해상도의 저하로 인하여 움직임 추정 오차가 커질 수 있으며, 특히 움직임 추정에 필요한 계산량은

$$C_L = 2T(2w+1)^2(N^2/4) = T(2w+1)^2N^2/2 \quad (4)$$

로서 부표본화로 인하여 FSBMA에 비하여서는 1/4의 계산량을 가지지만 TSS 알고리듬에 비하여서는 두 배 이상의 많은 계산량을 필요로 한다.

그러므로 움직임 추정 오차 측면에서 최적인 FSBMA에 가까운 성능을 유지하면서도 계산량을 줄일 수 있는 고속 움직임 추정 방법이 필요하다.

Ⅲ. 블록 MAD의 최소 및 최대 범위를 이용한 탐색점 줄임

본 논문에서는 움직임 탐색영역이 이웃 블록과 중첩되는 영역에서 현재 블록 MAD의 최소 및 최대 범위를 이웃 블록의 각 탐색점에 대한 MAD 및 현재 블록과 이웃 블록간의 MAD를 이용하여 구한

뒤, 이를 이용하여 탐색점 수를 줄임으로써 고속으로 움직임 추정을 행한다. 여기서, 현재 블록 MAD의 최소 및 최대 범위는 삼각부등식 (triangular inequality)을 이용하여 구한다.

1. 블록 MAD의 최소 및 최대 범위

MPEG[3],[4]에서는 CCIR601 표준 영상에서 움직임 추정에 사용된 블록의 크기가 16×16인 경우에 탐색영역의 크기를 수직 및 수평 방향으로 -16~15를 권고하고 있으며, 본 논문에서는 SIF 영상에 대하여 블록 크기를 8×8, 탐색영역의 크기를 -8~7로 하여 움직임 추정을 행하였다. 이 경우에 움직임 탐색영역이 이웃 블록과 겹치게 되는데, 이를 그림 1에 나타내었다. 이 그림에서 각 격자는 탐색점을 나타내고, 검은 점으로 표시된 부분이 이웃 블록과 중첩되는 탐색점을 나타내며, 흰점으로 나타낸 부분이 MAD를 계산하여 탐색하여야 할 탐색점을 나타낸다.

(k,l) 위치의 N×N 크기의 현재 블록을 블록 A라 하고, 수직 방향의 이웃 블록을 블록 B라 하면, 중첩된 탐색영역에서 블록 A의 MAD의 최소 및 최대 범위를 블록 B를 이용하여 구할 수 있다. 즉, 블록 A와 블록 B의 각 화소의 휘도값들을 N×N 행렬로 나타내면

$$A=\begin{bmatrix} I_t(k,l) & \cdots & I_t(k,l+N-1) \\ \vdots & & \vdots \\ I_t(k+N-1,l) & \cdots & I_t(k+N-1,l+N-1) \end{bmatrix} \quad (5)$$

$$B=\begin{bmatrix} I_t(k-N,l) & \cdots & I_t(k-N,l+N-1) \\ \vdots & & \vdots \\ I_t(k-1,l) & \cdots & I_t(k-1,l+N-1) \end{bmatrix} \quad (6)$$

와 같고, 이전 프레임에서 중첩된 탐색점 (x,y)에서

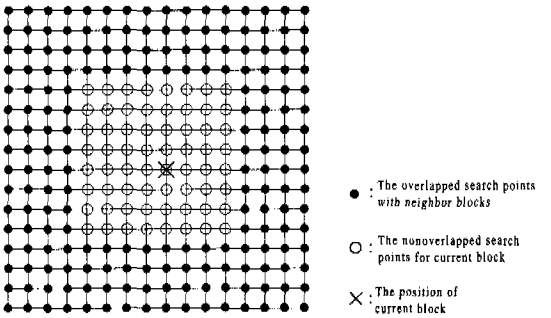


그림 1. 이웃 블록과의 탐색영역 중첩

정합될 블록 S(x,y), 즉 이전 프레임에서 (k+x,l+y)번째 블록을 N×N 행렬로 나타내면

$$S(x,y)=\begin{bmatrix} I_{t-1}(k+x,l+y) & \cdots & I_{t-1}(k+x,l+y+N-1) \\ \vdots & & \vdots \\ I_{t-1}(k+x+N-1,l+y) & \cdots & I_{t-1}(k+x+N-1,l+y+N-1) \end{bmatrix} \quad (7)$$

와 같다. 이때, 블록 A와 블록 S(x,y) 및 블록 B와 블록 S(x,y)의 정합에 의한 오차 블록은

$$U(x,y) = A - S(x,y) \quad (8)$$

$$V(x,y) = B - S(x,y) \quad (9)$$

와 같고, 블록 A와 블록 B사이의 차를 나타내는 오차 블록은

$$W = A - B \quad (10)$$

와 같다. 여기서 식 (8), (9), 및 (10)의 관계로부터

$$U(x,y) = W + V(x,y) \quad (11)$$

를 구할 수 있다. 이 식을 살펴보면, 블록 A와 블록 S(x,y)의 오차 블록 U(x,y)는 블록 B와 블록 S(x,y)의 오차 블록 V(x,y)와 이웃한 블록 A와 블록 B의 오차 블록 W의 합으로 표현될 수 있음을 알 수 있다. 그러나 우리가 최종적으로 구하고자하는 것은 블록 A와 블록 S(x,y)간의 MAD이므로 식 (11)을 이용하여 MAD를 구한다면 식 (10)의 과정으로 인하여 계산량의 이득을 얻을 수 없다. 그러므로 본 논문에서는 위의 관계식에 삼각부등식을 적용하여 블록 A의 최소 및 최대 범위를 구하여 이를 이용하여 탐색점의 수를 줄인다.

임의의 N×N 행렬 M의 L_1 놈 (norm)은

$$\| \mathbf{M} \| = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} | m_{ij} | \quad (12)$$

와 같이 정의되고, 삼각부등식은

$$\begin{aligned} | \| \mathbf{M}_1 \| - \| \mathbf{M}_2 \| | &\le \| \mathbf{M}_1 + \mathbf{M}_2 \| \\ &\le | \| \mathbf{M}_1 \| + \| \mathbf{M}_2 \| | \end{aligned} \quad (13)$$

와 같이 표현된다. 이를 식 (11)의 우변에 적용하면

$$\begin{aligned} |\,\|\mathbf{W}\|-\|\mathbf{V}(\mathbf{x},\mathbf{y})\|\,| &\le \|\mathbf{W}+\mathbf{V}(\mathbf{x},\mathbf{y})\| \\ &\le |\,\|\mathbf{W}\|+\|\mathbf{V}(\mathbf{x},\mathbf{y})\|\,| \end{aligned} \tag{14}$$

와 같다. 이때, 식 (11)로부터 $\|\mathbf{U}(\mathbf{x},\mathbf{y})\| = \|\mathbf{W}+\mathbf{V}(\mathbf{x},\mathbf{y})\|$ 이므로 식 (14)는

$$\begin{aligned} |\,\|\mathbf{W}\|-\|\mathbf{V}(\mathbf{x},\mathbf{y})\|\,| &\le \|\mathbf{U}(\mathbf{x},\mathbf{y})\| \\ &\le |\,\|\mathbf{W}\|+\|\mathbf{V}(\mathbf{x},\mathbf{y})\|\,| \end{aligned} \tag{15}$$

와 같이 표현 할 수 있다. 이 식의 양변을 N2으로 나누면

$$\begin{aligned} &|\,\tfrac{1}{N^2}\|\mathbf{W}\| - \tfrac{1}{N^2}\|\mathbf{V}(\mathbf{x},\mathbf{y})\|\,| \\ &\le \tfrac{1}{N^2}\|\mathbf{U}(\mathbf{x},\mathbf{y})\| \\ &\le |\,\tfrac{1}{N^2}\|\mathbf{W}\| + \tfrac{1}{N^2}\|\mathbf{V}(\mathbf{x},\mathbf{y})\|\,| \end{aligned} \tag{16}$$

와 같다. 이 식에서 $\frac{1}{N^2}\|\mathbf{U}(\mathbf{x},\mathbf{y})\|$는 블록 A의 MAD, $\frac{1}{N^2}\|\mathbf{V}(\mathbf{x},\mathbf{y})\|$는 블록 B의 MAD, 그리고 $\frac{1}{N^2}\|\mathbf{W}\|$는 이웃한 블록 A와 블록 B의 MAD를 의미한다. 즉, 블록 A와 블록 B가 중첩된 탐색점 (x,y)에서 블록 A의 MAD의 최소 범위는 블록 B의 MAD와 블록 A, B간의 MAD의 차의 절대값으로부터 구할 수 있고, 최대 범위는 블록 B의 MAD와 블록 A, B간의 MAD의 합으로 구할 수 있음을 알 수 있다. 이것은 수평, 수직 및 대각선의 각 방향 인접 블록에 대하여 동일하게 적용 될 수 있다.

이 식을 이용하여 블록 A의 MAD의 최소 범위 및 최대 범위를 구하는 경우의 계산량을 살펴보면, $\mathbf{V}(\mathbf{x},\mathbf{y})$는 블록 B의 탐색시에 계산된 MAD를 이용하면 되므로 별도의 계산량은 필요 없고, $\mathbf{W}$는 중첩된 탐색영역 전체에 대하여 한번만 계산하면 되므로 매우 적은 계산량으로 구할 수 있다. 이를 이용하여 정합이 필요한 탐색점에 대하여서만 MAD를 구하여 정합을 행한다면 움직임 추정 오차의 증가 없이 FSBMA와 같은 성능을 유지하면서도 고속으로 움직임을 추정할 수 있다.

2. 중첩된 탐색영역에서의 탐색점 줄임

이웃 블록과 중첩된 탐색영역에서는 식 (16)에서와 같이 현재 블록의 MAD의 최소 및 최대 범위를 이용하여 정합이 필요한 탐색점에 대하여서만 정합을 행한다. 이를 위하여 먼저, 이웃 블록과 중첩되지 않는 탐색점 (i,j)에서 정합을 행하여 이들 중 최소 MAD를 기준 평균 절대 오차로 한다. 즉, 기준 MAD를

$$MAD_{Ref.} = \min MAD(i,j) \tag{17}$$

where, (i,j) : Nonoverlapped search point with neighbor blocks.

와 같이 구한 뒤, 이웃 블록과 중첩된 탐색영역에 대하여서는 현재 블록과 다음 블록과의 MAD인 $\|\mathbf{W}_i\| = \|\mathbf{A}-\mathbf{B}_i\|$를 구하고, 이웃 블록의 MAD인 $\|\mathbf{V}_i\| = \|\mathbf{B}_i-\mathbf{S}(\mathbf{x},\mathbf{y})\|$를 이용하여

$$\begin{aligned} &IF\ (\ MAD_{Ref.} < |\,\tfrac{1}{N^2}(|\ \mathbf{V}_i(\mathbf{x},\mathbf{y})|\ ) \\ &\qquad -\tfrac{1}{N^2}(|\ \mathbf{W}_i|\ )\,|\ ) \\ &\qquad No\ block\ matching \end{aligned} \tag{18}$$

$$\begin{aligned} &ELSE\ \{ \\ &\qquad Block\ matching \\ &\qquad IF(MAD(x,y) < MAD_{Ref.}) \\ &\qquad\qquad MAD_{Ref.} = MAD(x,y) \\ &\} \end{aligned}$$

where, (x,y) : Ovrlapped search point with neighbor blocks.

와 같이 정합이 필요한 부분에 대하여서만 블록 정합을 행한다. 이와 같은 방법으로 탐색점을 줄임으로써 FSBMA와 같은 성능을 유지하면서도 계산량을 줄일 수 있다.

Ⅳ. 제안한 고속 블록 정합 알고리듬

이웃 블록과 탐색영역이 중첩된 영역에서의 MAD의 최소 및 최대 범위를 이용하여 탐색점을 줄이는 방법의 경우에는 FSBMA와 동일한 성능을 유지하면서도 계산량을 줄일 수 있다. 그러나 이 방법은 기존의 고속 움직임 추정 방법에 비하여 계산량이 많으므로 좀더 고속의 움직임 추정 방법이 필요하다.

본 논문에서는 이웃 블록과 중첩된 움직임 탐색영역에서의 MAD 분포 및 블록내 화소의 부표본화를 이용한 두가지 고속 움직임 추정 방법을 제안한다. 본 논문에서 제안한 첫 번째 고속 움직임 추정 방법에서는 앞 절에서 살펴본 MAD의 최소 및 최

대 범위를 이용한 탐색점 줄임과 블록내 화소의 부표본화를 이용하였다. 또한, 본 논문에서 제안한 두 번째 고속 움직임 추정 방법은 블록내 화소의 부표본화와 실제 MAD값의 분포를 이용하여 추정 오차의 큰 증가 없이 계산량을 현저히 줄일 수 있는 방법이다.

1. 블록 MAD의 최소 및 최대 범위와 부표본화를 이용한 고속 움직임 추정

블록 MAD의 최소 및 최대 범위를 이용한 탐색점 줄임 방법은 움직임 추정 오차 측면에서 FSBMA와 동일한 성능을 얻으면서도 계산량을 줄일 수 있는 효과적인 움직임 추정 방법이다. 그러나 기존의 고속 움직임 추정 방법에 비하여 계산량이 많은 단점이 있다. 그러므로 본 논문에서는 MAD의 최소 및 최대 범위와 부표본화를 이용하여 고속으로 움직임 추정을 행할 수 있는 방법을 제안한다.

제안한 방법에서는 각 블록내의 화소를 4:1로 부표본화한 뒤, 이웃 블록과 중첩된 탐색 영역에서는 MAD의 최소 및 최대 범위를 이용하여 식 (18)에서와 같이 MAD의 최소 범위가 기준 MAD 보다 작은 경우에만 블록 정합을 행한다. 이때 부표본화 방법은 Liu 등이 제안한 방법과 같은 방법을 사용하였다.

제안한 방법의 계산량을 살펴보면, 이웃 블록과 중첩되지 않은 탐색영역의 크기가 전체 탐색영역의 1/4에 해당하고, 블록의 화소들에 대하여 4:1로 부표본를 행하였으므로 이 영역에서의 블록 정합을 위하여 $2T((2w+1)^2/4)(N^2/4) = T(2w+1)^2N^2/8$ 의 계산량이 필요한데 이는 Liu 등이 제안한 방법의 계산량의 1/4에 해당하는 양이다. 또한 수직 및 수평 방향의 이웃 블록과의 MAD를 계산하기 위한 $8TN^2$ 및 기준 MAD가 MAD의 최소 범위 보다 큰 탐색점에 대한 MAD 계산을 위한 $2P(N^2/4) = PN^2/2$ 의 부가적인 계산량이 필요하므로 제안한 방법의 총 계산량은

$$C_{pl} = T(2w+1)^2N^2/8 + 8TN^2 + PN^2/2 \qquad (19)$$

과 같다. 이 식에서 보는 바와 같이 제안한 방법의 계산량은 기준 MAD가 MAD의 최소 범위 보다 큰 탐색점의 한 프레임 총 개수 P에 의존하므로 입력 영상의 특성에 따라 약간의 차이는 나타낼 수 있지만, Liu 등이 제안한 방법에 비하여 동일한 움직임 추정 오차를 가지면서도 많은 계산량의 감소를 얻을 수 있다.

2. MAD의 분포 특성 및 부표본화를 이용한 고속 움직임 추정

앞에서 설명한 블록 MAD의 최소 및 최대 범위와 부표본화를 이용한 고속 움직임 추정방법은 우수한 움직임 추정 성능을 가지면서도 계산량이 적은 방법이지만, TSS 방법에 비하여 계산량이 조금 많다. 본 논문에서는 TSS 알고리듬에 비하여 움직임 추정 오차 측면에서 월등한 성능을 가지면서도 계산량을 현저히 줄일 수 있는 고속 움직임 추정 방법을 제안한다. 제안한 방법에서는 MAD의 분포 특성과 부표본화를 이용하였다.

식 (16)은 블록 정합을 통하여 얻을 수 있는 최소 MAD가 존재할 수 있는 범위를 나타낸 것으로서, 실제 MAD는 MAD의 최소 및 최대 범위 사이의 값을 가지게되며, 일반적으로 실제 MAD값은 MAD의 최소 범위보다 큰 값을 가지게 된다. 그러므로 본 논문에서는 실제 MAD값이 MAD의 최소 및 최대 범위 사이에 어떻게 분포하는지를 구한 뒤, 이로부터 블록 정합의 실행 여부를 결정하는 문턱값을 구하여

$$\begin{aligned} &IF\ (\ MAD_{Ref.} < TH\ ) \\ &\qquad No\ block\ matching \qquad (20) \\ &ELSE\ \{ \\ &\qquad Block\ matching \\ &\qquad IF(MAD(x,y) < MAD_{Ref.}) \\ &\qquad\qquad MAD_{Ref.} = MAD(x,y) \\ &\} \end{aligned}$$

where, (x,y) : Overlapped search point with neighbor blocks.

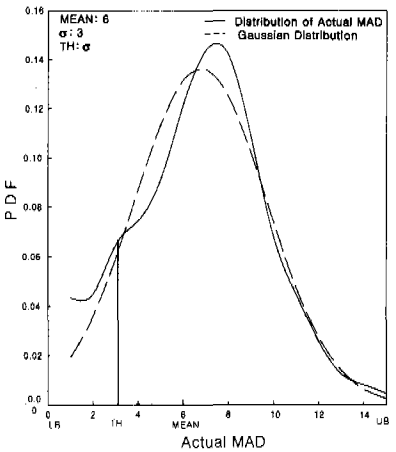
와 같이 필요한 경우에 대하여서만 블록 정합을 행하였다. 여기서 TH는 문턱값이다.

본 논문에서 제안한 방법을 구체적으로 살펴보면 먼저, 실제 MAD의 분포는 식 (11)과 (16)으로부터 유추할 수 있다. 식 (11)을 살펴보면 현재 블록의 오차 블록 U(x,y)를 현재 블록과 이웃 블록과의 오차 블록 W와 이웃 블록의 현재 탐색점에서의 오차 블록 V(x,y)로부터 구하였다. 그러나 우리가 구하고자하는 것은 현재 블록의 MAD이므로 U(x,y)의 놈을 구하는 과정에서 W와 V(x,y)의 놈으로부터 직접 구하지 못하고 최소 범위 및 최대 범위로 나타나게된다. 여기서 식 (11)과 (16)을 연관시켜 살펴보면, 식 (16)에서 최소 범위는 식 (11)의 W와 V(x,y)의 부호가 모두 반대인 경우의 $\|W+V(x,y)\|$

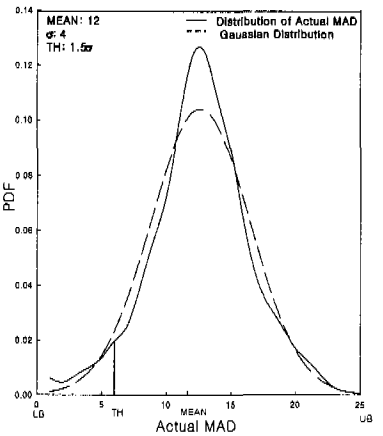
와 같고, 최대 범위는 식 (11)의 W와 V(x,y)의 부호가 모두 같은 경우의 ‖W+V(x,y)‖와 같다. 이로부터 W와 V(x,y)의 부호가 모두 같거나 모두 반대인 경우는 매우 적고, 부호가 혼재하는 경우가 많음을 알 수 있다. 즉, 실제 MAD값의 분포는 MAD의 최소 및 최대 범위와 같을 확률은 매우 적고, MAD의 최소 범위 와 최소 범위의 평균값 근처에 많이 분포함을 유추할 수 있다. 그림 2에는 실제 MAD의 분포를 블록 정합을 통해 구하여 나타내었다. 이 분포는 최소 범위를 0으로 정규화 (normalization)시킨 뒤, 최대와 최소 범위의 차가 같은 것들에 대하여 구한 것으로서 대표적인 몇 가지 경우에 대하여서만 그림으로 나타내었다.

이 그림에서 볼 수 있는 바와 같이 실제 MAD값은 MAD의 최소 범위와 최대 범위의 평균값으로 나타날 확률이 가장 높고, 최소 및 최대 범위에 가까워질수록 확률이 줄어드는 가우스적 분포를 가짐을 확인할 수 있으며, 이를 확인하기 위하여 같은 평균값 및 분산값을 가지는 가우스적 분포를 함께 나타내었다. 이러한 실제 MAD의 분포 특성을 이용하여 문턱값을 결정하여 사용하면 움직임 추정 오차의 큰 증가 없이 움직임 추정에 필요한 계산량을 줄일 수 있다.

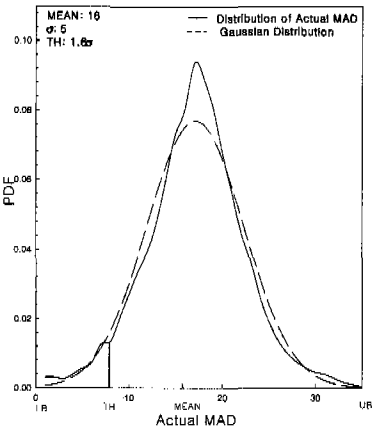
본 논문에서 사용한 방법을 그림 2의 (c)에 나타나있는 최소 범위와 최대 범위의 차가 35인 경우를 예를 들어 설명하면, 이 분포의 경우에 평균값은 16이 되고 표준 편차를 구해보면 5가 되며, 그림에 표시된 것처럼 최소 범위와 최대 범위의 차의 1/4 지점을 문턱값으로 사용하였다면 이 값은 8로서 평균값으로부터 표준 편차의 1.6배 떨어진 위치이다.



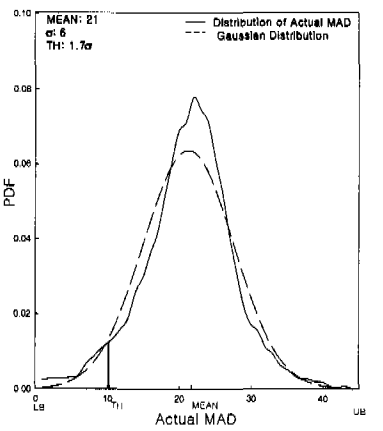
(a)



(b)



(c)



(d)

그림 2. MAD의 최소 및 최대 범위 사이에 존재하는 실제 MAD의 분포; (a) 최대 및 최소 범위의 차가 15인 경우, (b) 25인 경우, (c) 35인 경우 및 (d) 45인 경우

이 경우에 최소 MAD가 본 논문에서 설정한 문턱값 보다 작은 값을 가질 확률은 5.48%로 매우 작은 값을 가지기 때문에 문턱값을 8로 하더라도 움직임 추정 오차가 거의 발생하지 않음을 알 수 있다. 또한, 최소 MAD가 문턱값 보다 작은 경우가 발생하더라도 이때의 움직임 추정 오차는 7이하의 작은 값을 가지게 되므로 움직임 추정 오차의 증가가 적음을 알 수 있다.

제안한 방법의 계산량을 살펴보면, 이웃 블록과 중첩되지 않은 탐색영역이 전체 탐색영역의 1/4을 차지하고, 블록 정합시에 블록내 화소의 부표본화를 사용하므로, 이 영역에서 MAD를 계산하는데 $2T((2w+1)^2/4)(N^2/4) = T(2w+1)^2N^2/8$ 의 계산량이 필요하다. 또한 수직 및 수평 방향의 이웃 블록과의 MAD를 구하기 위하여 $8T(N^2/4) = 2TN^2$ 의 부가적인 계산량이 필요하며, 기준 MAD가 TH 보다 큰 탐색점의 한 프레임 총 개수가 P라면 $2P(N^2/4) = PN^2/2$의 추가 계산량이 필요하므로 본 논문에서 제안한 방법의 총 계산량은

$$C_{pl} = T(2w+1)^2N^2/8 + 2TN^2 + PN^2/2 \qquad (21)$$

와 같다. 그러므로 본 논문에서 제안한 방법의 계산량은 P에 따라 영향을 받게되는데, 문턱값이 커짐에 따라 P의 값이 작아지게되지만 움직임 추정 오차는 커지게 되므로 이를 고려하여 문턱값을 결정하여야 한다.

V. 실험결과 및 고찰

본 논문에서 제안한 방법의 성능 평가를 위하여 컴퓨터 모의 실험을 행하였다. 본 실험에서는 SIF 영상인 FLOWER GARDEN, MOBILE, 그리고 TABLE TENNIS 영상 각각 40 프레임을 사용하였다. 움직임 추정에 사용된 블록의 크기는 8×8이며 탐색 범위는 -7~7을 사용하였다. 본 논문에서 제안한 방법들 중 블록 MAD의 최소 및 최대 범위와 부표본화를 이용한 고속 움직임 추정방법을 제안방법1, MAD의 분포 특성 및 부표본화를 이용한 고속 움직임 추정방법을 제안방법2로 부르기로 한다.

제안방법1은 블록 MAD의 최소 및 최대 범위를 이용하여 필요한 경우에만 블록내 화소를 부표본화 한 뒤 블록 정합을 행하는 방법이므로 Liu 등이 제안한 방법과 동일한 PSNR을 얻으면서도 계산량을 현저히 줄일 수 있는 방법이다. 이 방법의 실험영상 각 40 프레임에 대한 평균 계산량을 표 1에 나타내었다. 여기서 계산량은 FSBMA를 기준으로 하여 상대적으로 나타내었다.

표 1. 제안방법1의 평균 계산량

Sequences	FSBMA	Liu's method	TSS	Proposed method1
FLOWER GARDEN	100.00	25.00	11.11	17.27
MOBILE	100.00	25.00	11.11	15.73
TABLE TENNIS	100.00	25.00	11.11	19.57

이 표로부터 제안방법1의 계산량이 Liu 등이 제안한 방법에 비하여 훨씬 적음을 볼 수 있으며, 특히 MOBILE 영상의 경우에는 TSS 알고리듬에 가까운 계산량을 가짐을 확인할 수 있다. 이를 자세히 살펴보기 위하여 FLOWER GARDEN 및 MOBILE 영상 40 프레임에 대한 계산량을 그림 3에 나타내었다. 그림의 세로축의 최대값 25는 Liu 등이 제안한 방법의 계산량을 나타내고, 최소값 11은 TSS 알고리듬의 계산량과 거의 같은 값을 나타내고 있다. 이 그림으로부터 제안방법1의 계산량이 Liu 등이 제안한 방법보다 훨씬 적고, TSS 알고리듬의 계산량에 가까움을 확인할 수 있다.

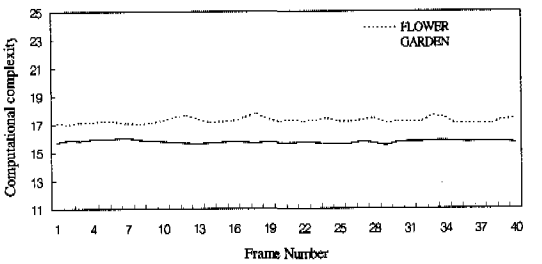


그림 3. FLOWER GARDEN 및 MOBILE 영상에 대한 제안방법1의 계산량

또한, 제안방법1의 계산량은 식 (19)에 나타난 것과 같이 기준 MAD가 MAD의 최소 범위 보다 큰 탐색점의 개수에 따라 달라지지만, 이 그림으로부터 계산량의 변화량이 평균값을 기준으로 매우 적음을 알 수 있다. 그리고, 이 표와 그림으로부터 제안방법1의 계산량은 TSS 알고리듬에 비하여 다소 많음을 볼 수 있다. 그러나, 움직임 추정 오차 측면에서는 제안방법1이 TSS 알고리듬에 비하여 월등히 좋은 성능을 나타낸다. 이를 확인하기 위하여 실험영상 각 40 프레임에 대한 평균 PSNR을 표 2에 나

타내었다. 이 표를 살펴보면 제안방법1이 TSS 알고리듬에 비하여 약 3.5[dB]에서 1.5[dB] 정도의 PSNR 향상을 얻을 수 있음을 알 수 있다. 그림 4에는 제안방법1의 FLOWER GARDEN 및 MOBILE 영상 40 프레임에 대한 PSNR을 FSBMA와 TSS 알고리듬과 비교하여 나타내었다. FLOWER GARDEN 영상의 경우에는 꽃으로 구성된 복잡한 부분이 많기 때문에 TSS 알고리듬에 사용된 가정인 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정이 잘 맞지 않아서 움직임 추정시에 전역 최소 (global minimum)를 찾지 못하고 국부 최소에 빠지는 현상을 보이지만, 제안방법1의 경우에는 모든 탐색점에 대하여 탐색을 행하는 효과를 얻기 때문에 움직임 추정 오차 측면에서 최적인 FSBMA에 가까운 우수한 움직임 추정 성능을 나타내고 있음을 이 그림으로부터 알 수 있다.

표 2. 제안방법1의 평균 PSNR

Sequences	FSBMA	TSS	Proposed method1	Improved PSNR
FLOWER GARDEN	25.98	22.44	25.92	3.48
MOBILE	24.35	22.66	24.15	1.49
TABLE TENNIS	30.62	28.83	30.48	1.65

이상의 결과로부터 제안방법1은 움직임 추정 오차 측면에서 TSS 알고리듬에 비하여 월등히 우수한 성능을 나타냄을 확인할 수 있으며, 계산량 측면에서는 TSS 알고리듬에 비하여 약간 많은 계산량을 가짐을 확인할 수 있다.

제안방법2는 블록 MAD의 최소 및 최대 범위사이에 존재하는 실제 MAD의 분포의 확률적 모델을 이용하여 TSS 알고리듬에 비하여 적은 계산량을 가지면서도 우수한 움직임 추정 성능을 가지는 방법이다. 제안방법2의 문턱값에 따른 평균 계산량과 PSNR을 표 3에 나타내었다. 여기서 계산량은 FSBMA를 기준으로 하여 상대적으로 나타내었으며, 문턱값은 그림 2에서 설명한 바와 같이 MAD의 최대 범위와 최소 범위의 차에 대하여 최소 범위로부터의 상대적인 위치로 나타내었다.

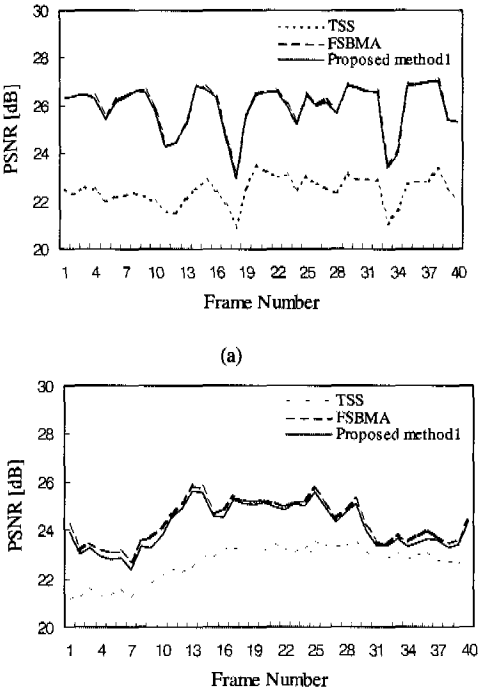


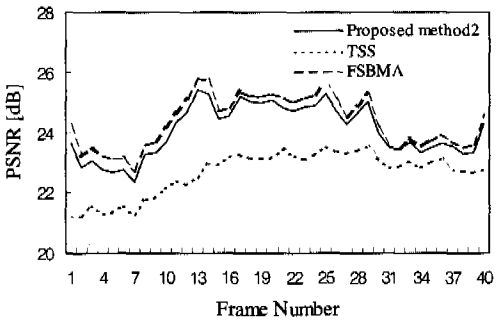
그림 4. 제안방법1의 (a) FLOWER GARDEN 및 (b) MOBILE 영상에 대한 PSNR 비교

이 표를 살펴보면 제안 방법2의 계산량이 문턱값의 증가에 따라 크게 줄어들다가 3/8 이상의 문턱값에서는 큰 변화가 없음을 알 수 있으며, 계산량의

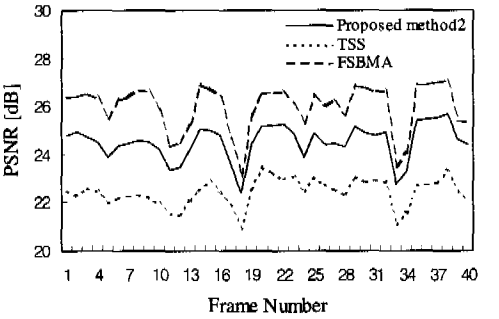
표 3. 제안방법2의 평균 계산량 및 PSNR

Performance	Sequences	FSBMA	TSS	Proposed method1			
				TH = 1/8	TH = 1/4	TH = 3/8	TH = 1/2
Computational Complexity	MOBILE	100.00	11.11	14.06	11.24	9.63	9.19
	FLOWER GARDEN	100.00	11.11	15.99	12.62	10.14	9.42
	TABLE TENNIS	100.00	11.11	19.23	15.68	11.12	9.48
PSNR [dB]	MOBILE	24.35	22.66	24.06	24.00	23.96	23.91
	FLOWER GARDEN	25.98	22.44	25.09	24.89	24.52	24.32
	TABLE TENNIS	30.62	28.83	30.43	30.37	30.29	30.22

감소에 비하여 PSNR은 상대적으로 느리게 감소함을 알 수 있다. FLOWER GARDEN 영상은 복잡한 꽃 부분과 나무가 빠르게 움직이고, 복잡하지 않은 배경 부분이 느리게 움직이는 영상이다. 이 영상의 경우에는 빠른 움직임을 가지는 복잡한 부분이 많기 때문에 탐색영역의 일부분만을 탐색하는 TSS 알고리듬의 움직임 추정 성능이 FSBMA에 비하여 크게 떨어짐을 이 표로부터 알 수 있다. 이에 비하여 이 영상에 대한 제안방법2의 움직임 추정 성능을 살펴보면, 복잡한 부분의 빠른 움직임으로 인하여 이웃 블록과 중첩된 부분에서의 블록 정합 여부를 결정하는 문턱값이 증가함에 따라 어느 정도의 PSNR 감소를 보이고 있으나 TSS 알고리듬에 비하여 현저히 우수한 성능을 나타냄을 알 수 있다. 또한, MOBILE 영상은 기차부분이 빠르게 움직이고, 나머지 배경 부분은 느리게 움직이는 영상이다. 이 영상의 경우에는 영상의 많은 부분을 차지하는 배경 부분이 느리게 움직이고 있기 때문에 많은 움직임 벡터가 전체 탐색영역의 1/4을 차지하는 이웃 블록과 중첩되지 않는 부분에 존재하게 된다. 그러므로 문턱값을 어느 정도 크게 하더라도 움직임 추정 오차가 적음을 알 수 있다. TSS 알고리듬에 비하여 작거나 비슷한 계산량을 가지는 3/8의 문턱값을 사용한 경우의 FLOWER GARDEN 영상과 MOBILE 영상 40 프레임에 대한 PSNR을 그림 5에 나타내었다. 이 그림으로부터 MOBILE 영상의 경우에 제안방법2가 FSBMA에근접한 우수한 움직임 추정 성능을 나타냄을 확인할 수 있으며, FLOWER GARDEN 영상의 경우에는 FSBMA에 비해서는 움직임 추정 성능이 조금 떨어지지만 제안방법2보다 많은 계산량을 가지는 TSS 알고리듬에 비하여 우수한 움직임 추정 성능을 나타냄을 알 수 있다. 또한 제안방법2의 계산량을 그림 6에 나타내었다.



(a)



(b)

그림 5. 제안방법2의 (a) MOBILE 및 (b) FLOWER GARDEN 영상에 대한 PSNR 비교

이 그림에서 보는바와 같이 제안방법2는 TSS 알고리듬보다 적은 계산량을 보이고 있음을 확인할 수 있으며, 제안방법2의 계산량은 식 (21)에서와 같이 기준 MAD의 값이 문턱값 보다 큰 탐색점의 개수에 따라 달라지지만, 거의 일정한 계산량을 유지하고 있음을 알수 있다.

이상의 결과로부터 본 논문에서 제안한 두 가지 고속 움직임 추정 방법이 계산량이 적으면서도 우수한 움직임 추정 성능을 나타냄을 확인할 수 있었다.

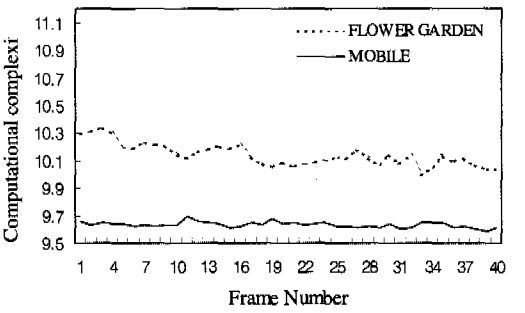


그림 6. FLOWER GARDEN 및 MOBILE 영상에 대한 제안방법2의 계산량

Ⅵ. 결 론

본 논문에서는 이웃 블록과 중첩된 움직임 탐색영역에서의 MAD 분포 및 블록내 화소의 부표본화를 이용한 두가지 고속 움직임 추정 방법을 제안하였다. 제안한 두 가지 방법에서는 이웃 블록과 중첩된 영역에서 현재 블록 MAD의 최소 및 최대 범위를 구하여 이용하였으며, 이때 사용되는 MAD의 최소 및 최대 범위는 이웃 블록의 MAD와 현재 블

록과 이웃 블록간의 MAD를 이용하여 구하였다. 또한, 블록 정합은 블록내 화소를 부표본화한 뒤 행하였다. 제안한 첫 번째 방법에서는 블록 MAD의 최소 범위를 이용하여 이웃 블록과 움직임이 중첩되지 않는 영역에서 구한 기준 MAD가 블록 MAD의 최소 범위보다 큰 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다. 이 방법은 모든 탐색점에 대하여 탐색을 행하는 것과 같은 효과를 가지는 방법으로서 움직임 추정 오차 측면에서 최적인 FSBMA에 가까운 우수한 움직임 추정 성능을 나타내었다. 또한, 본 논문에서는 계산량을 크게 줄이기 위한 방법으로 현재 블록 MAD의 분포 특성을 이용한 고속 움직임 추정 방법을 제안하였다. 제안한 방법에서는 이웃 블록과 중첩된 각 탐색점에서의 실제 MAD는 MAD의 최소 및 최대 범위 사이에 존재하게된다는 것을 이용하여, 실제 MAD 분포의 확률적인 모델을 이용하여 고속으로 움직임을 추정하였다. 이 방법은 대표적인 고속 움직임 추정 방법의 하나인 TSS 알고리듬에 비하여 적은 계산량을 가지면서도 뛰어난 움직임 추정 성능을 나타내었다. 제안한 방법의 성능을 평가하기 위한 컴퓨터 모의 실험결과로부터 제안한 방법이 움직임 추정 오차 측면에서 우수한 성능을 유지하면서도 계산량을 현저히 줄일 수 있음을 확인할 수 있었다.

참 고 문 헌

[1] *ITU-T Recommendation H.261,* "Video codec for audiovisual services at $p \times 64$ kbits/s."

[2] *ITU-T Recommendation H.263,* "Video coding for low bit rate communication."

[3] *ISO/IEC 11172-2,* "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbits/s: Video."

[4] *ISO/IEC 13818-2,* "Information technology - Generic Coding of Moving Pictures and Associated Audio Information: Video."

[5] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI design for the motion compensation block-matching algorithm," *IEEE Trans. Circuit and Systems,* vol. 36, no. 10, pp. 1309-1316, 1989.

[6] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.,* vol. 29, no. 12, pp. 1799-1801, Dec., 1981.

[7] T. Koga, K. Iinuma, A Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. Nat. Telecommun. Conf.*, pp. G5.3.1-G5.3.5, Nov./Dec. 1981.

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

[9] M. Ganbari, "The cross search algorithm for motion estimation," *IEEE Trans. Commun.,* vol. COM-38, no. 7, pp. 950-953, July 1990.

[10] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuit and Systems for Video Technology,* vol. 4, no. 4, pp. 438-442, Aug. 1994.

[11] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuit and Systems for Vidoe Technology*, vol. 3, no. 2, pp. 148-157, Apr. 1993.

[12] Y. L. Chan and W. C. Siu, "New adaptive pixel decimation for block motion vector estimation," *IEEE Trans. Circuit and Systems for Vidoe Technology*, vol. 6, no. 1, pp. 113-118, Feb. 1996.

[13] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuit and Systems for Vidoe Technology,* vol. 6, no. 3, pp. 313-317, June 1996.

이 법 기(Bub-Ki Lee) 정회원
통신학회논문지 제24권 제1B호 참조

정 원 식(Won-Sik Cheong) 정회원
통신학회논문지 제24권 제1B호 참조

이 경 환(kyeong-Hwan Lee) 정회원
통신학회논문지 제24권 제1B호 참조

최 정 현(Jung-Hyun Choi) 정회원
통신학회논문지 제24권 제1B호 참조

김 경 규(Kyoung-Kyoo Kim) 정회원
통신학회논문지 제24권 제1B호 참조

김 덕 규(Duk-Gyoo Kim) 정회원
통신학회논문지 제24권 제1B호 참조