이웃 탐색점에서의 평균 절대치 오차 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬

정회원 정 원 식*, 이 법 기*, 한 찬 호*, 권 성 근*, 장 종 국**, 이 건 일*

A Fast Block Matching Algorithm Using Mean Absolute Error of Neighbor Search Point and Search Region Reduction

Won-Sik Cheong*, Bub-Ki Lee*, Chan-Ho Han*, Seong-Geun Kwon*, Jong-Kook Jang**, Kuhn-Il Lee* ***Regular Members***

요 약

본 논문에서는 이웃 탐색점에서의 평균 절대치 오차 (mean absolute error, MAE) 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬을 제안하였다. 이 알고리듬은 두 단계로 구성되어있다. 첫 번째 단계에서는 탐색영역을 3×3 크기의 영역으로 겹치지 않게 나눈 뒤, 각 영역의 중심 탐색점에 대하여 블록 정합을 행하여 MAE를 구하고, 이들 중 가장 작은 MAE를 기준 MAE로 정한다. 그리고, 두 번째 단계에서는 각 영역의 중심 탐색점에서의 MAE를 이용하여 각 3×3 영역의 나머지 탐색점에서의 MAE의 최소 범위를 구한 뒤, 최소 범위가 기준 MAE 보다 작은 탐색점에 대하여서만 블록 정합을 행한다. 이때, 최종 움직임 벡터는 첫 번째 단계에서 기준 MAE로 결정된 탐색점 근처에 존재할 가능성이 매우 큼을 이용하여 기준 MAE로 결정된 탐색점을 중심으로 탐색영역의 크기를 줄인 뒤, 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다. 모의 실험을 통하여 본 제안한 방법이 우수한 움직임 추정 성능을 유지하면서도 많은 계산량의 감소를 얻을 수 있음을 확인하였다.

ABSTRACT

In this paper, we propose a fast block matching algorithm using the mean absolute error (MAE) of neighbor search point and search region reduction. The proposed algorithm is composed of two stages. At the first stage, the search region is divided into nonoverlapped 3×3 areas and MAE of the center point of each area is calculated. The minimum MAE value of all the calculated MAE's is determined as reference MAE. At the second stage, because the possibility that final motion vector exist near the position of reference MAE is very high, we use smaller search region than first stage. And, using the MAE of center point of each area, the lower bound of rest search point of each area is calculated and block matching process is performed only at the search points that the lower bound is smaller than reference MAE. By doing so, we can significantly reduce the computational complexity while keep the increasement of motion estimation error small.

* 경북대학교 전자전기공학부(formula@palgong.knu.ac.kr),
** 영동대학교 전자공학부
논문번호 : 99410-1011, 접수일자 : 1999년 10월 11일

Ⅰ. 서 론

블록 정합 알고리듬 (block matching algorithm, BMA)은 H.261,[1] H.263,[2] 및 MPEG[3],[4] 등과 같은 많은 동영상 압축 시스템에서 사용되고 있으며, 시간적인 중복성을 제거하여 높은 압축율을 얻는데 핵심적인 역할을 담당하고 있다.

BMA의 목적은 현재 프레임을 임의의 작은 블록으로 나눈 뒤, 각 블록에 대하여 이전 프레임에 설정된 탐색영역에서 정합 척도가 최적인 블록을 찾는 것이다. 이때, 정합 척도로는 평균 자승 오차 (mean squared error, MSE)와 비슷한 성능을 가지면서도 계산량이 적은 MAE가 많이 사용된다.

BMA를 이용하여 움직임을 추정하는 경우에 가장 좋은 성능을 얻을 수 있는 방법은 이전 프레임에 설정된 탐색영역의 모든 후보 정합 블록 (candidate matching block)에 대하여 탐색을 행하는 전역 탐색 알고리듬 (full search algorithm, FSA)이다. 그러나, 이 방법은 움직임 추정 오차 측면에서 최적인 움직임 벡터를 얻을 수 있지만 계산량이 많은 단점이 있다.

이상에서와 같은 단점을 줄이기 위하여 움직임 벡터를 고속으로 추정할 수 있는 여러 가지 고속 블록 정합 방법들이 제안되었다. 고속 블록 정합 방법은 크게 FSA와 동일한 움직임 추정 성능을 유지하면서 계산량을 줄이는 방법[5]과 블록 정합을 행하는 탐색점 수를 줄이는 방법[6]-[12] 및 블록 정합 시에 사용되는 화소의 수를 줄이는 방법[13],[14]으로 나눌 수 있다.

Li 등[5]은 FSA와 동일한 성능을 유지하면서도 계산량을 줄일 수 있는 연속적 제거 근사화 (successive elimination approximation, SEA) 알고리듬을 제안하였다. 이 방법에서는 현재 블록의 평균값과 탐색영역의 후보 정합 블록의 평균값을 이용하여 블록 정합이 필요한 탐색점 수를 줄임으로써, 고속으로 움직임을 추정하였다. 그러나, 이 방법에서는 이전 프레임의 각 후보 정합 블록의 평균값을 저장하기 위하여 두 프레임 정도의 메모리가 추가로 필요하다는 단점과, 블록 정합 여부를 결정하는 기준 MAE를 이전 프레임의 움직임 벡터를 이용하여 결정하므로 입력 영상의 움직임 특성에 따라 성능이 저하될 수 있다는 단점을 가진다. 또한, 이 방법은 움직임 추정 오차 측면에서는 최적의 성능을 유지하지만 계산량의 감소는 별로 크지 않다.

Liu 등[14]은 블록의 화소들을 4:1로 부표본화 한 뒤, 이를 이용하여 블록 정합을 행하는 방법을 제안하였다. 이 방법에서는 블록 정합에 전체 화소의 일부분만 사용되므로 블록내의 화소들이 비슷한 값을 가지는 평탄한 블록의 경우에는 우수한 움직임 추정 성능을 나타내지만, 복잡한 블록의 경우에는 부표본화에 의한 해상도의 저하로 인하여 움직임 추정 오차가 커지는 단점을 가진다. 또한, 움직임 추정에 필요한 계산량이 FSA의 약 26%로서 블록 정합을 행하는 탐색점 수를 줄이는 방법에 비하여 많은 계산량을 가진다.

블록 정합을 행하는 탐색점 수를 줄임으로써 고속으로 움직임을 추정 방법으로는 간단하면서도 성능이 좋은 3 단계 탐색 (three step search, TSS) 알고리듬이 가장 많이 사용되며, MPEG-1 SM3 및 H.261 RM8[15]에 권고되었다. 그러나, 이 방법은 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 탐색영역의 일부분에 대하여서만 탐색을 행하므로, 계산량은 크게 줄일 수 있었지만 국부 최소에 빠질 수 있는 단점을 가진다. 또한, 이 방법은 탐색영역이 작은 경우에는 어느 정도의 움직임 추정 성능을 기대할 수 있지만, MPEG에서와 같이 탐색영역이 큰 경우에는 계산량은 크게 줄일 수 있지만 국부 최소에 빠질 확률이 매우 커지게 되어 움직임 추정의 정확성이 크게 떨어지게 된다.

Jong 등[12]은 TSS를 큰 탐색영역에 적용할 경우에 발생하는 성능 저하를 개선하기 위한 방법으로 탐색영역을 네 개로 나누어서 각 영역에 대하여 TSS를 적용하는 4TSS 방법과 탐색영역의 중심에 대하여 TSS를 한번 더 적용하는 5TSS 방법을 제안하였다. 그러나, 이 방법은 MPEG-1 SM3에서 권고하는 TSS의 단계 수를 늘이는 방법의 성능을 어느 정도 개선하였지만, FSA에 비하여서는 여전히 큰 움직임 추정 오차를 가진다. 그러므로 움직임 추정 오차 측면에서 최적인 FSA에 가까운 움직임 추정 성능을 유지하면서도 계산량을 줄일 수 있는 방법이 필요하다.

본 논문에서는 이웃 탐색점에서의 평균 절대치 오차 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬을 제안하였다. 제안한 방법에서는 현재 탐색점에서 블록 정합을 통하여 얻을 수 있는 MAE의 최소 범위를 이웃 탐색점에서의 MAE를 이용하여 구한 뒤, 이를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행하였다. 이때, 현재 탐색점에서의 MAE의 최소 범위를 구하기 위해서는 이웃 탐색점에서의 MAE가 필요하며, 블록 정합의 수행 여부를 결정하기 위해서는 MAE의 최소

범위와 비교할 수 있는 기준 MAE가 필요하다.

제안한 방법의 첫 번째 단계에서는 탐색영역을 3×3 크기의 영역으로 겹치지 않게 나눈 뒤, 각 영역의 중심 탐색점에 대하여 블록 정합을 행하여 MAE를 구하고, 이들 중 가장 작은 MAE를 기준 MAE로 정한다. 또한, 두 번째 단계에서는 각 영역의 중심 탐색점에서의 MAE를 이용하여 각 3×3 영역의 나머지 탐색점에서의 MAE의 최소 범위를 구한 뒤, 최소 범위가 기준 MAE 보다 작은 탐색점에 대하여서만 블록 정합을 행한다. 이때, 최종 움직임 벡터는 첫 번째 단계에서 기준 MAE로 결정된 탐색점 근처에 존재할 확률이 매우 크다. 즉, TSS 방법 등에서 사용된 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정은 실제 영상에 대하여 잘 맞지 않지만 실제 움직임 근처의 작은 영역에서는 어느 정도 성립하므로,[16] 실제 움직임 벡터는 첫 번째 단계에서 각 3×3 영역의 중심점에 대하여 구한 MAE 중 최소 MAE를 가지는 탐색점 근처에 많이 분포하게 된다. 그러므로 제안한 방법의 두 번째 단계에서는 기준 MAE로 결정된 탐색점을 중심으로 탐색영역의 크기를 줄인 뒤, 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다.

제안한 방법의 성능을 평가하기 위한 여러 가지 동영상에 대한 컴퓨터 모의 실험을 통하여, 제안한 방법이 움직임 추정에 필요한 계산량을 현저히 줄이면서도 FSA에 가까운 움직임 추정 성능을 가짐을 확인하였다.

Ⅱ. 블록 MAE의 최소 및 최대 범위

본 논문에서는 이웃 탐색점에서의 MAE 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬을 제안한다. 제안한 방법에서는 탐색영역을 3×3 크기의 영역으로 겹치지 않게 나눈 뒤, 각 영역의 중심 탐색점에 대하여 블록 정합을 행하여 MAE를 구하고, 이들 중 최소 값을 갖는 MAE를 기준 MAE로 정한 후, 기준 MAE를 갖는 탐색점을 중심으로 탐색영역을 줄여서 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행한다. 본 장에서는 먼저, 현재 탐색점에서의 MAE를 이웃 탐색점에서의 블록 정합 결과로부터 구할 수 있음을 보인 뒤, 여기에 삼각 부등식 (triangular inequality)을 적용하여 현재 탐색점에서의 MAE의 최소 및 최대 범위를 구하는 방법을 제시한다. 그리고, 다음 장에서 최종 움직임 벡터가 기준 MAE를 갖는 탐색점 근처에 분포할 확률이 매우 높음을 보인 뒤, MAE의 최소 범위 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬을 제안한다.

1. 이웃 탐색점에서의 오차 블록을 이용한 현재 탐색점에서의 MAE 계산

수평 및 수직 방향의 크기가 N인 현재 블록 C와 탐색영역의 (x,y) 위치에서의 후보 정합 블록 (candidate matching block) $\mathbf{M}(x,y)$의 화소들의 휘도 값을 $N\times N$ 행렬로 나타내면 각각

$$\mathbf{C} = [\ \mathbf{C}_0\ \cdots\ \mathbf{C}_{N-1}\ ] \quad (1)$$
$$where,\ \mathbf{C}_k = [\ c_{0,k}\ \cdots\ c_{N-1,k}\ ]^T$$

$$\mathbf{M}(x,y) = [\ \mathbf{M}_0(x,y)\ \cdots\ \mathbf{M}_{N-1}(x,y)\ ] \quad (2)$$

$$where,\ \mathbf{M}_k(x,y) = [\ m_{y,x+k}\ \cdots\ m_{y+N-1,x+k}\ ]^T$$

와 같고, 이들의 오차 블록 $\mathbf{D}(x,y)=\mathbf{M}(x,y)-\mathbf{C}$는

$$\mathbf{D}(x,y) = [\ \mathbf{M}_0(x,y)-\mathbf{C}_0\ \cdots\ \mathbf{M}_{N-1}(x,y)-\mathbf{C}_{N-1}\ ] \quad (3)$$

와 같다. 그리고, 탐색점 (x,y)에서의 MAE는

$$MAE(x,y)=\frac{1}{N^2}\|\mathbf{M}(x,y)-\mathbf{C}\|=\frac{1}{N^2}\|\mathbf{D}(x,y)\| \quad (4)$$

와 같이 구할 수 있다.

또한, 블록 정합을 행한 탐색점 (x,y)의 수평 방향의 이웃 탐색점 $(x+1,y)$에서의 오차 블록은

$$\begin{aligned}&\mathbf{D}(x+1,y)\\&=[\ \mathbf{M}_0(x+1,y)-\mathbf{C}_0\\&\quad\cdots\ \mathbf{M}_{N-2}(x+1,y)-\mathbf{C}_{N-2}\ \mathbf{M}_{N-1}(x+1,y)-\mathbf{C}_{N-1}\ ]\\&=[\ \mathbf{M}_1(x,y)-\mathbf{C}_0\\&\quad\cdots\ \mathbf{M}_{N-1}(x,y)-\mathbf{C}_{N-2}\ \mathbf{M}_N(x,y)-\mathbf{C}_{N-1}\ ]\end{aligned} \quad (5)$$

와 같다. 여기서, 식 (2)의 $\mathbf{M}_k(x,y)$의 정의로부터 $\mathbf{M}_k(x+1,y)=\mathbf{M}_{k+1}(x,y)$의 관계가 성립함을 알 수 있다. 또한, 수평 방향의 이웃하는 탐색점인 (x,y)와

$(x+1, y)$의 블록 정합에 사용되는 탐색영역의 화소들은 그림 1에서 보는바와 같이 하나의 열을 제외한 나머지는 동일함을 알 수 있다. 이 사실을 이용하면 탐색점 (x, y)에서의 오차 블록 $\mathbf{D}(x, y)$를 이용하여 탐색점 $(x+1, y)$에서의 오차 블록 $\mathbf{D}(x+1, y)$를 구할 수 있다. 이를 위하여 현재 블록 내의 화소들의 이웃 화소간의 수평 방향으로의 화소값의 차를

$$\mathbf{C}_{DH+} = [\ \mathbf{C}_0 - \mathbf{C}_{N-1}\quad \mathbf{C}_1 - \mathbf{C}_0 \quad \cdots\quad \mathbf{C}_{N-1} - \mathbf{C}_{N-2}\ ] \tag{6}$$

와 같이 나타내면, $\mathbf{D}(x, y)$와 $\mathbf{C}_{DH+}$의 합은

$$\mathbf{D}(x, y) + \mathbf{C}_{DH+} = [\ \mathbf{M}_0(x, y) - \mathbf{C}_{N-1}\quad \mathbf{M}_1(x, y) - \mathbf{C}_0 \quad \cdots\quad \mathbf{M}_{N-1}(x, y) - \mathbf{C}_{N-2}\ ] \tag{7}$$

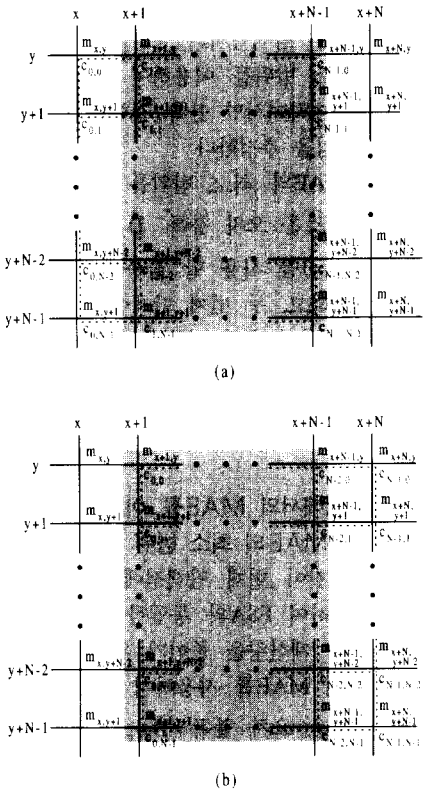

그림 1. (a) 탐색점 (x, y) 및 (b) 탐색점 $(x+1, y)$에서의 블록 정합

와 같으며, 이 식을 식 (5)와 비교해 보면, 식 (5)의 마지막 열을 제외한 나머지 부분과 이 식의 첫 번째 열을 제외한 나머지 원소들이 동일함을 알 수 있다. 즉, 그림 1에서 회색으로 표시된 탐색영역의 화소들이 탐색점 (x, y) 및 탐색점 $(x+1, y)$의 블록 정합에 공통으로 사용되고 있으므로, 이 부분에 대하여서는 각 탐색점의 오차 블록 계산에 사용되는 현재 블록의 화소들만 다르다. 그러므로, $\mathbf{D}(x+1, y)$ 중에서 이 영역에 해당하는 부분은 $\mathbf{D}(x, y)$를 이용하여 $\mathbf{D}(x, y)$ 계산에 사용된 현재 블록의 화소들을 제거하고, $\mathbf{D}(x+1, y)$의 계산에 사용되는 현재 블록의 화소들을 사용함으로써 계산할 수 있다. 이러한 과정은 식 (6)에서와 같이 현재 블록의 이웃 화소간의 차를 구한 뒤, 식 (7)에서와 같이 $\mathbf{D}(x, y)$와 더함으로써 구현된다.

또한, 그림 1을 살펴보면, 탐색점 (x, y)에서의 수평 좌표 x에 해당하는 부분과 탐색점 $(x+1, y)$에서의 수평 좌표 $x+N$에 해당하는 부분이 서로 다름을 알 수 있다. 이러한 차이로 인하여 식 (7)과 식 (5)의 차이가 발생된다. 그러므로 $\mathbf{D}(x, y)$의 첫 번째 열 $\mathbf{M}_0(x, y) - \mathbf{C}_0$를 $\mathbf{M}_N(x, y) - \mathbf{C}_0$로 대치시킨 새로운 오차 블록 $\hat{\mathbf{D}}_{H+}(x, y)$를

$$\hat{\mathbf{D}}_{H+}(x, y) = [\ \mathbf{M}_N(x, y) - \mathbf{C}_0\quad \mathbf{M}_1(x, y) - \mathbf{C}_1 \quad \cdots\quad \mathbf{M}_{N-1}(x, y) - \mathbf{C}_{N-1}\ ] \tag{8}$$

와 같이 구한 뒤, 이를 $\mathbf{C}_{DH+}$와 더하면

$$\hat{\mathbf{D}}_{H+}(x, y) + \mathbf{C}_{DH+} = [\ \mathbf{M}_N(x, y) - \mathbf{C}_{N-1}\quad \mathbf{M}_1(x, y) - \mathbf{C}_0 \cdots \mathbf{M}_{N-1}(x, y) - \mathbf{C}_{N-2}\ ] \tag{9}$$

와 같다. 이를 식 (5)의 $\mathbf{D}(x+1, y)$와 비교해 보면, 각 열이 오른쪽으로 순환 이동 (circular shift)된 형태로서 동일한 원소로 구성되어 있음을 알 수 있다. 그러므로 탐색점 $(x+1, y)$에서의 MAE는

$$MAE(x+1, y) = \frac{1}{N^2} \| \mathbf{M}(x+1, y) - \mathbf{C} \| = \frac{1}{N^2} \| \hat{\mathbf{D}}_{H+}(x, y) + \mathbf{C}_{DH+} \| \tag{10}$$

와 같이 구할 수 있다. 이 식으로부터 $MAE(x+1, y)$는 미리 구해진 탐색점 (x, y)에서의 오차 블록으로부터 구할 수 있음을 알 수 있다.

2. MAE의 최소 및 최대 범위

앞 절에서 살펴본 바와 같이 각 탐색점에서의 MAE는 이웃 탐색점에서의 오차 블록과 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차를 이용하여 식 (10)에서와 같이 구할 수 있다. 그러나 이와 같은 방법으로 $MAE(x+1,y)$를 구한다면 $\mathbf{C}_{DH+}$를 구하는 과정과 $\hat{\mathbf{D}}_{H+}(x,y)$의 첫 번째 열을 대치시키는 과정으로 인하여 계산량의 이득을 얻을 수 없다. 그러므로 본 논문에서는 삼각 부등식을 이용하여 $MAE(x+1,y)$의 최소 범위를 구한 뒤 이를 이용하여 움직임 추정을 위한 계산량을 줄인다.

임의의 $N\times N$ 행렬 $\mathbf{A}$의 합의 놈 (sum norm)은

$$\| \mathbf{A} \| = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | a_{ij} | \tag{11}$$

와 같이 정의된다.[5] 그리고, 삼각 부등식은

$$| \| \mathbf{A}_1 \| - \| \mathbf{A}_2 \| | \le \| \mathbf{A}_1 + \mathbf{A}_2 \| \le \| \mathbf{A}_1 \| + \| \mathbf{A}_2 \| \tag{12}$$

와 같다. 이를 식 (10)의 $\| \hat{\mathbf{D}}_{H+}(x,y) \|$와 $\| \mathbf{C}_{DH+} \|$에 적용하고, $\| \hat{\mathbf{D}}_{H+}(x,y) + \mathbf{C}_{DH+} \| = \| \mathbf{M}(x+1,y) - \mathbf{C} \|$ 이므로 이를 대입한 뒤, 양변을 N^2으로 나누면

$$\left| \frac{1}{N^2} \| \hat{\mathbf{D}}_{H+}(x,y) \| - \frac{1}{N^2} \| \mathbf{C}_{DH+} \| \right| \le MAE(x+1,y) \le \frac{1}{N^2} \| \hat{\mathbf{D}}_{H+}(x,y) \| + \frac{1}{N^2} \| \mathbf{C}_{DH+} \| \tag{13}$$

와 같은 수식이 얻어진다. 이 식을 살펴보면 탐색점 $(x+1,y)$에서의 MAE의 최소 및 최대 값을 현재 블록 내의 화소들의 이웃 화소간의 화소값의 차 $\mathbf{C}_{DH+}$와 탐색점 (x,y)에서의 오차 블록 $\mathbf{D}(x,y)$의 첫 번째 열을 대치한 블록 $\hat{\mathbf{D}}_{H+}(x,y)$ 로부터 구할 수 있음을 알 수 있다. 이때 식 (13)에서 $\| \mathbf{C}_{DH+} \|$는 전체 탐색영역에 대하여 한번만 계산하면 되고, $\| \hat{\mathbf{D}}_{H+}(x,y) \|$는 N개의 열 중에서 첫 번째 열을 대치시키는 과정이 필요하므로, 블록 정합을 행하는 경우에 비하여 매우 적은 계산량으로 MAE의 최소 범위를 구할 수 있다. 그러므로 이를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 정합을 행한다면 계산량을 크게 줄일 수 있다. 그리고 이것은 수직 및 대각선 방향의 탐색점에 대하여서도 동일하게 적용할 수 있다.

Ⅲ. 제안한 고속 블록 정합 알고리듬

앞 장에서 살펴본 바와 같이, 현재 탐색점에서의 MAE의 최소 및 최대 범위는 이웃 탐색점에서의 MAE를 이용하여 식 (13)과 같이 구할 수 있다. 그리고, 이를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행한다면, 움직임 추정 오차 측면에서 최적인 FSA와 동일한 성능을 얻으면서도 움직임 추정에 필요한 계산량을 줄일 수 있다. 그러나, 식 (13)의 최소 및 최대 범위를 구하기 위해서는 식 (3)으로 표현되는 이웃 탐색점에서의 오차 블록을 사용하지 않고, 식 (8)에서 정의된 첫 번째 열을 대치시킨 새로운 오차 블록을 구하여야하므로 이에 따른 부가 계산량이 필요하다. 그러므로, 본 논문에서는 식 (3)에서 정의된 이웃 탐색점에서의 오차 블록 $\mathbf{D}(x,y)$를 사용하여 현재 탐색점에서의 MAE의 근사 최소 범위를 구하여 이용한다. 또한, 두 단계로 나누어 수행되는 제안한 방법에서는 두 번째 단계에서 탐색영역의 크기를 줄인 뒤, MAE의 근사 최소 범위를 이용하여 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정한다.

본 장에서는 MAE의 최소 범위를 구할 때, 식 (8)에서 정의된 새로운 오차 블록 $\hat{\mathbf{D}}_{H+}(x,y)$를 사용하지 않고 $\mathbf{D}(x,y)$를 사용 할 경우에 발생할 수 있는 오차를 해석하고, 두 번째 단계에서 탐색영역을 줄이는 것이 움직임 추정 성능에 미치는 영향을 살펴본 뒤, 이웃 탐색점에서의 MAE 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬을 제안한다.

1. 이웃 탐색점에서의 MAE를 이용한 현재 탐색점에서의 MAE의 최소 범위 계산

식 (13)에서와 같이 현재 탐색점에서의 MAE의 최소 범위를 이용하여 FSA와 동일한 움직임 추정 성능을 유지하면서 계산량을 줄이기 위해서는 이웃 탐색점 (x,y)에서의 MAE를 사용하지 않고, 식 (8)에서 정의된 새로운 오차 블록 $\hat{\mathbf{D}}_{H+}(x,y)$를 이용하여야 한다. 또한, 우리가 이웃 탐색점 (x,y)에서 블록 정합을 통하여 구하는 것은 오차 블록 $\mathbf{D}(x,y)$가 아니라, 탐색점 (x,y)에서의 MAE인 $\| \mathbf{D}(x,y) \|$이다. 그러므로, 이로부터 식 (13)의 최소 범위를 구하는데 필요한 $\| \hat{\mathbf{D}}_{H+}(x,y) \|$를 구하기 위해서는

$$\| \hat{\mathbf{D}}_{H+}(x,y) \| = \| \mathbf{D}(x,y) \| - \| \mathbf{M}_0(x,y) - \mathbf{C}_0 \| + \| \mathbf{M}_N(x,y) - \mathbf{C}_0 \| \quad (14)$$

$$where, \| \mathbf{M}_0(x,y) - \mathbf{C}_0 \| = \sum_{i=0}^{N-1} | m_{y+i,x} - c_{i,0} |$$

$$\| \mathbf{M}_N(x,y) - \mathbf{C}_0 \| = \sum_{i=0}^{N-1} | m_{y+i,x+N} - c_{i,0} |$$

와 같이 미리 계산되어 진 $\| \mathbf{D}(x,y) \|$의 첫 번째 열에 해당하는 양을 뺀 후, 대치시키는 열에 해당하는 양을 더하는 과정이 필요하다. 이를 위해서는 $\| \mathbf{M}_0(x,y) - \mathbf{C}_0 \|$를 구하기 위한 계산량과 $\| \mathbf{M}_N(x,y) - \mathbf{C}_0 \|$를 구하기 위한 계산량이 부가적으로 필요하다. 그러므로, 본 논문에서는 $\| \hat{\mathbf{D}}_{H+}(x,y) \|$를 사용하지 않고, $\| \mathbf{D}(x,y) \|$를 이용하여 현재 탐색점에서의 MAE의 최소 범위의 근사 값을 구한 뒤, 이를 이용하여 고속으로 움직임을 추정한다.

2. 블록 MAE의 근사 최소 범위의 오차

수평 방향의 이웃 탐색점에서의 오차 블록을 이용하여 구한 현재 탐색점에서의 MAE의 실제 최소 범위는 식 (13)에서 구한 것과 같이

$$LB_{act}(x+1,y) = | \frac{1}{N^2} \| \hat{\mathbf{D}}_{H+}(x,y) \| - \frac{1}{N^2} \| \mathbf{C}_{DH+} \| | \quad (15)$$

와 같이 나타낼 수 있다. 그러나 이를 구하기 위해서는 식 (14)와 같은 과정이 필요하며, 이에 따른 부가 계산량이 필요하다. 그러므로, 본 논문에서는 이러한 부가 계산량을 줄이고, 알고리듬을 간소화하기 위하여 MAE의 최소 범위의 근사 값, 즉 MAE의 근사 최소 범위를

$$LB_{app}(x+1,y) = | \frac{1}{N^2} \| \mathbf{D}(x,y) \| - \frac{1}{N^2} \| \mathbf{C}_{DH+} \| | \quad (16)$$

와 같이 구하여 사용한다. 이 식을 살펴보면, 식 (15)에서 사용된 $\| \hat{\mathbf{D}}_{H+}(x,y) \|$ 대신 $\| \mathbf{D}(x,y) \|$를 사용하여 $LB_{app}(x+1,y)$를 구한다. 그러므로, 식 (15)에서와 같이 $\| \hat{\mathbf{D}}_{H+}(x,y) \|$를 구하기 위해서 필요한 부가 계산량이 필요 없다. 그러나, 실제 MAE의 최소 범위인 $LB_{act}(x+1,y)$를 사용하지 않고, 이의 근사 값인 $LB_{app}(x+1,y)$를 사용하기 때문에 이로 인한 오차가 발생할 수 있다.

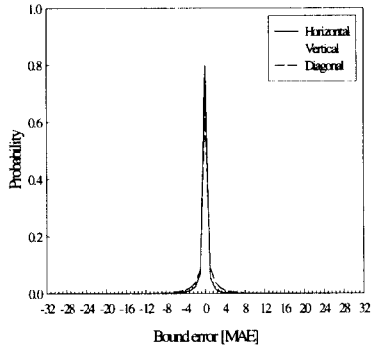

그림 2. MAE의 최소 범위 오차의 확률 분포

이러한 MAE의 최소 범위의 오차는 움직임 추정에 사용되는 블록의 크기가 $N \times N$라 하면, 수평 및 수직 방향의 이웃 탐색점의 경우에는 MAE의 최소 범위는 $-N$에서 N사이의 오차를 가질 수 있다. 그리고, 대각 방향의 이웃 탐색점의 경우에는 행과 열이 하나의 화소에서 겹치기 때문에 $-(2N-1)$에서 $2N-1$사이의 오차를 가질 수 있다. 실제 영상에 대한 MAE 최소 범위의 오차의 확률 분포를 그림 2에 나타내었다. 여기서, 오차는 $LB_{act}(x+m,y+n) - LB_{app}(x+m,y+n)$을 나타낸다. 이때, m과 n은 -1과 1사이의 값을 가질 수 있으며, $LB_{act}(x\pm1,y) - LB_{app}(x\pm1,y)$는 수평, $LB_{act}(x,y\pm1) - LB_{app}(x,y\pm1)$는 수직, $LB_{act}(x\pm1,y\pm1) - LB_{app}(x\pm1,y\pm1)$는 대각 방향을 나타내고, 복호의 순서는 같다. 이 그림을 살펴보면, MAE의 최소 범위를 구하는데 $\| \mathbf{D}(x,y) \|$를 사용하더라도 식 (14)와 같이 행 및 열을 대치시키는 과정을 통하여 구한 실제 최소 범위와 같을 확률이 매우 높음을 알 수 있으며, 오차가 커짐에 따라 오차가 발생할 확률이 급격히 떨어짐을 알 수 있다. 이는 오차가 발생하더라도 작은 값을 가짐을 나타낸다.

또한, 최소 범위의 오차가 움직임 추정 성능에 영향을 미치는 경우는 현재 탐색점에서의 MAE값이 탐색영역 내에서 최소 값을 가짐에도 불구하고, 최소 범위의 오차로 인하여 블록 정합을 행하지 않는 경우이다. 그러나, 앞에서 살펴본 바와 같이 MAE의 최소 범위의 오차는 작은 값에 집중되어 나타나므로, 최소 범위의 오차로 인하여 현재 탐색점에서의 MAE값이 탐색영역 내에서 최소 값을 가짐에도 불구하고, 블록 정합을 행하지 않기 때문에 발생하

는 움직임 추정 성능의 저하는 거의 없을 것으로 예상할 수 있다. 그러므로 본 논문에서는, 실제 최소 범위 $LB_{act}(x+1,y)$를 사용하지 않고, 최소 범위의 근사 값 $LB_{app}(x+1,y)$를 사용하여 고속으로 움직임을 추정한다.

3. 고속 블록 정합 알고리듬

본 논문에서는 현재 탐색점에서의 MAE의 근사 최소 범위를 이웃 탐색점에서의 MAE를 이용하여 구한 뒤, 이를 이용하여 블록 정합이 필요한 탐색점의 수를 줄임으로써 고속으로 움직임을 추정할 수 있는 방법을 제안한다. 제안한 방법에서는 움직임 추정 과정을 두 단계로 나누어 수행한다. 또한, 제안한 방법의 두 번째 단계에서는 첫 번째 단계에서 기준 MAE로 결정된 MAE를 가지는 탐색점을 중심으로 탐색영역의 크기를 줄여서 블록 정합이 필요한 탐색점에 대하여서만 블록 정합을 행한다.

3.1. 1단계 - 각 영역의 중심 탐색점에서의 MAE 및 기준 MAE 계산

제안한 방법에서는 이웃 탐색점에서의 MAE를 이용하여 현재 탐색점에서의 MAE의 근사 최소 범위를 구한다. 이때, 하나의 탐색점에 대하여 블록 정합을 행하여 MAE를 구하였다면, 주위 8개 탐색점에서의 MAE의 근사 최소 범위를 구할 수 있다. 그러므로, 제안한 방법의 첫 번째 단계에서는 이전 프레임에 설정된 탐색영역을 그림 3에서와 같이 3×3 크기의 영역으로 겹치지 않게 나눈 뒤, 그림에서 검은 점으로 표시된 각 영역의 중심 탐색점에 대하여 블록 정합을 행하여 MAE를 구한다. 이렇게

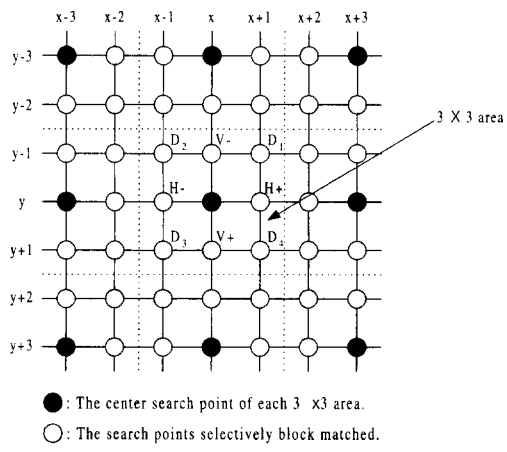

그림 3. 제안한 방법에서 사용되는 탐색 구조

구한 각 영역의 중심 탐색점에서의 MAE는 2단계에서 주위 8개 탐색점에 대한 근사 최소 범위를 구할 때 사용된다.

또한, 각 탐색점에서의 MAE의 근사 최소 범위를 이용하여 블록 정합의 수행 여부를 결정하기 위해서는 MAE의 근사 최소 범위와 비교할 수 있는 기준 MAE가 필요하다. 그러므로, 두 번째 단계에서 블록 정합을 행할 것인지를 결정하기 위하여 사용되는 기준 평균 절대치 오차 MAE_{ref}를

$$MAE_{ref} = \min MAE(x,y) \quad (17)$$

where, (x,y): *The center search point of each* 3×3 *area.*

와 같이 각 3×3 영역의 중심 탐색점에서 구한 MAE들 중 최소 MAE로 정한다.

3.1.1. 기준 MAE를 가지는 탐색점과 최종 움직임 벡터의 상관성

제안한 방법의 첫 번째 단계에서는 그림 3에서와 같이 탐색영역의 탐색점 아홉 개마다 하나의 탐색점에 대하여 블록 정합을 행하여 이들 중 최소 MAE를 기준 MAE로 결정하였다. 이 경우, 현재 블록의 최종 움직임 벡터가 기준 MAE로 결정된 탐색점 근처에 존재할 확률이 매우 커지게 된다. 즉, TSS 방법 등에 사용되는 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 살펴보면, 실제 영상에서 움직임 추정 오차는 탐색영역 내에 몇 개의 국부 최소를 가지기 때문에 움직임 방향으로 단조 감소한다는 것은 사실이 아니지만 전역 최소 근처의 작은 영역에서는 움직임 추정 오차들이 작은 값을 가지며, 움직임 추정 오차가 전역 최소 방향으로 단조 감소하게된다.[16] 그러므로, 제안한 방법에서와 같이 전체 탐색영역에 대하여 탐색점 아홉 개마다 하나의 탐색점에서 블록 정합을 행하여 얻은 최소 MAE 위치 근처에 최종 움직임 벡터가 존재할 가능성이 많다.

기준 MAE를 가지는 탐색점과 최종 움직임 벡터의 상관성을 확인하기 위하여 실제 영상에 대하여 구한 기준 MAE를 가지는 기준 탐색점 MP_{ref}와 최종 움직임 벡터 MV의 차의 분포를 그림 4에 나타내었다. 이 그림에서 $MP_{ref}(x)$와 $MP_{ref}(y)$는 각각 기준 탐색점의 수평 및 수직 좌표를 나타내고, $MV(x)$ 와 $MV(y)$는 각각 최종 움직임 벡터의 수평 및 수직 좌표를 나타낸다. 이 그림을 살펴보면, 최종 움직임 벡터의 대부분이 첫 번째 단계에서 구한 기준 MAE를 가지는 탐색점 근처에 분포함을

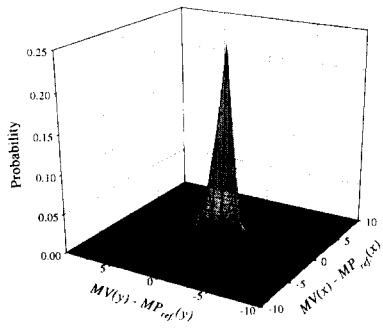

그림 4. 최종 움직임 벡터와 기준 MAE를 가지는 탐색점 위치의 차의 확률 분포

확인할 수 있다. 그러므로 제안한 방법의 두 번째 단계에서는 탐색영역 전체에 대하여 움직임 추정을 행하지 않고, 기준 MAE를 가지는 탐색점의 위치를 기준으로 탐색영역의 크기를 줄여서 최종 움직임 벡터를 구한다.

3.2. 2단계 - 근사 MAE의 최소 범위 및 탐색영역 줄임을 이용한 계산량 줄임

제안한 방법의 두 번째 단계에서는 기준 MAE를 갖는 탐색점을 중심으로 탐색영역의 크기를 줄인 뒤, 각 3×3 영역의 중심 탐색점에서 블록 정합을 통하여 구한 $MAE(x,y)$, 즉 $\frac{1}{N^2}\| \mathbf{D}(x,y) \|$를 이용하여, 탐색점 (x,y) 주위의 8개 이웃 탐색점 $(x\pm1,y)$, $(x,y\pm1)$ 및 $(x\pm1,y\pm1)$에서의 MAE의 근사 최소 범위 $|\frac{1}{N^2}\| \mathbf{D}(x,y) \| - \frac{1}{N^2}\| \mathbf{C}_{\mathbf{Dk}} \||$를 구한다. 여기서, k는 그림 3에 나타낸 것과 같이 주위 8개의 탐색점들 각각의 위치를 나타낸다. 제안한 방법에서는 이렇게 구한 MAE의 근사 최소 범위를 이용하여

$$
\begin{aligned}
&IF\ (\ MAE_{ref} < |\ \tfrac{1}{N^2}\| \mathbf{D}(x,y) \| - \tfrac{1}{N^2}\| \mathbf{C}_{\mathbf{Dk}} \|\ |\ ) \\
&\qquad No\ block\ matching \\
&ELSE\ \{ \qquad\qquad\qquad\qquad\qquad\qquad (18) \\
&\qquad Block\ matching \\
&\qquad IF(MAE(i,j) < MAE_{ref,}) \\
&\qquad\qquad MAE_{ref} = MAE(i,j)\ \}
\end{aligned}
$$

와 같이 MAE의 근사 최소 범위가 기준 MAE인 MAE_{ref}보다 큰 탐색점에 대하여서만 블록 정합을 행한다. 여기서 (i,j)는 각 영역의 중심 탐색점을 제외한 나머지 여덟 개의 탐색점을 나타낸다. 즉, 제안한 방법에서는 첫 번째 단계에서 블록 정합을 행하였던 각 3×3 영역의 중심 탐색점을 제외한 나머지 탐색점에 대하여 그 탐색점에서 가질 수 있는 MAE의 근사 최소 범위를 이웃 탐색점에서의 MAE를 이용하여 구한 뒤, 이 값이 기준 평균 절대치 오차 MAE_{ref}보다 큰 경우에는 블록 정합을 수행하지 않는다. 또한, MAE의 근사 최소 범위가 MAE_{ref}보다 작은 탐색점의 경우에는 이 탐색점에서의 MAE가 MAE_{ref}보다 작을 가능성이 있으므로, 이 탐색점에 대하여서는 블록 정합을 행하여 MAE를 구한 뒤 MAE_{ref}와 비교한다. 이때, 이 탐색점에서의 MAE인 $MAE(i,j)$가 기준 MAE인 MAE_{ref}보다 작으면 MAE_{ref}는 $MAE(i,j)$로 갱신되고, 이것의 최종 값을 지닌 탐색점이 최종 움직임 벡터로 결정된다. 이와 같이 제안한 방법에서는 현재 블록의 움직임 탐색영역의 모든 탐색점에 대하여 탐색을 행하지 않고, 적은 탐색점에 대하여서만 블록 정합을 행하면서도, 우수한 움직임 추정 성능을 얻을 수 있었다.

4. 제안한 기법의 계산량

FSA는 이전 프레임에 설정된 모든 탐색점에 대하여 MAE를 계산하며, 각 탐색점에서 MAE를 계산하기 위해서는 $2N^2$번의 덧셈과 뺄셈의 계산이 필요하다. 그러므로 현재 프레임의 블록의 전체 개수가 T이고, 움직임의 상하, 좌우의 최대 변위가 w이면, 한 프레임에 대한 FSA의 총 계산량은

$$C_{FSA} = 2T(2w+1)^2N^2 \qquad (19)$$

와 같다.

반면, 제안한 방법의 계산량은 첫 번째 단계에서 각 3×3 영역의 중심점에서의 MAE를 구하기 위한 계산량과 두 번째 단계에서 탐색영역을 줄인 뒤, 식 (18)에서와 같이 블록 정합이 필요한 탐색점에 대하여 블록 정합을 행하기 위한 계산량의 합으로 볼 수 있다. 먼저 첫 번째 단계에서 각 3×3 영역의 중심점에서의 MAE를 구하기 위한 계산량을 살펴보면, 전체 탐색점의 1/9에 해당하는 탐색점에 대하여 블록 정합을 행하여야하므로 FSA의 계산량의 1/9에 해당하는 $2T(2w+1)^2N^2/9$의 계산량이 필요하다. 식 (18)의 계산에서는 각 탐색점에 대하여 MAE의 근사 최소 범위를 구하기 위한 계산량과 MAE의 근사 최소 범위가 MAE_{ref}보다 작은 탐색점에서의 블록 정합을 위한 계산량이 필요하다. 여기서, MAE의 최소 범위를 구하기 위해서는 수평, 수직 및 대

각선 방향에 대응되는 $\| C_{Dk} \|$를 구하여야 한다. $\| C_{Dk} \|$의 계산을 위해서는 전체 탐색영역에 대하여 각 방향에 $2N^2$번의 덧셈과 뺄셈의 계산이 필요하지만, 탐색점의 위치가 각 3×3 영역의 중심점에 대하여 180° 의 차이를 갖는 경우에 대하여서는 동일한 값을 갖는다. 즉, $\| C_{DH+} \|$와 $\| C_{DH-} \|$는 동일한 값이다. 그러므로 $\| C_{Dk} \|$를 구하기 위한 계산량은 한 블록 전체에 대하여 $8N^2$번의 덧셈과 뺄셈의 계산이 필요하다. 또한, MAE의 최소 범위가 MAE_{rej}보다 작은 탐색점에 대하여 블록 정합을 행하기 위한 계산량은 탐색점당 $2N^2$이다. 그러므로 제안한 방법의 계산량은

$$C_{proposed} = \frac{2}{9} TN^2(2w+1)^2 + 2PN^2 + 8TN^2 \quad (20)$$

와 같다. 여기서 F는 식 (18)에서 기준 MAE의 위치를 중심으로 한 작은 탐색영역에서 MAE의 최소 범위가 MAE_{rej}보다 작은 탐색 점의 총 개수이다. 이 식으로부터 제안한 방법의 계산량은 MAE의 최소 범위가 기준 평균 절대 오차 MAE_{rej}보다 큰 탐색점 수가 많을수록 FSA의 계산량에 비하여 많이 감소됨을 알 수 있다.

Ⅳ. 실험 결과 및 고찰

본 논문에서 제안한 방법의 성능을 평가하기 위하여 컴퓨터 모의실험을 행하였다. 본 실험에서는 720×480의 공간해상도를 가지는 FLOWER GARDEN, MOBILE 및 SUSIE 영상 각 40프레임을 사용하였으며, 블록 정합에 사용된 블록의 크기는 16×16, 탐색영역의 크기는 수직과 수평방향으로 -16~+15로 MPEG의 권고안과 동일하게 하였다. 본 실험에서 사용한 FLOWER GARDEN 영상은 꽃밭과 같은 복잡한 부분이 많이 존재하고, 정지된 물체에 카메라가 빠르게 움직이는 영상이고 MOBILE 영상은 작은 움직임을 가지면서 에지가 많이 존재하는 배경에 기차가 움직이는 영상으로서 움직임이 작은 배경 영역과 물체의 움직임이 빠른 영역이 혼재 하는 영상이다. 그리고, SUSIE 영상은 배경이 매우 단순한 영상으로 큰 움직임이 없는 영상이다. 그리고 정합 척도로는 MAE를 사용하였다.

본 실험에서는 제안한 MAE의 근사 최소 범위의 오차가 움직임 추정 성능에 미치는 영향을 알아보고, 실험 영상에 대한 기준 MAE를 가지는 탐색점과 최종 움직임 벡터의 상관성을 알아본 뒤, 제안한 고속 블록 정합 알고리듬에 대한 결과를 계산량 및 움직임 보상된 영상과 원 영상간의 PSNR을 이용하여 평가하였다.

1. MAE의 근사 최소 범위의 오차가 움직임 추정 성능에 미치는 영향

제안한 방법에서는 현재 탐색점의 MAE의 근사 최소 범위를 이용하여 블록 정합이 필요한 탐색점 수를 줄임으로써 고속으로 움직임을 추정한다. 이때, 현재 탐색점의 MAE의 최소 범위를 사용하지 않고, 근사값을 사용하기 때문에 이에 의한 오차가 발생할 수 있다. 이러한 오차로 인하여 움직임 추정 오차가 증가할 수 있는 경우는 실제 최소 범위 LB_{act}는 MAE_{rej}보다 작았지만, 근사 최소 범위 LB_{app}는 MAE_{rej}보다 커지는 경우이다. 이 경우는 LB_{act}를 사용한다면 블록 정합을 행하여야 하지만, LB_{app}를 사용하기 때문에 블록 정합을 행하지 않는 경우이다. 그러나, 이러한 경우가 모두 움직임 추정 오차의 증가를 일으키는 것은 아니며, 움직임 추정 오차가 증가하기 위해서는 실제 움직임 벡터 위치에서 LB_{app}의사용으로 인한 오차 때문에 블록 정합을 행하지 않음으로서 움직임 벡터를 잘못 찾는 경우이다.

이러한 경우가 움직임 추정 오차에 미치는 영향을 알아보기 위하여 최소 범위 오차로 인하여 움직임 벡터를 잘못 찾을 확률 및 움직임 벡터를 잘못 찾은 경우에 최적의 움직임 벡터와의 MAE 차의 분포를 표 1 및 표 2에 각각 나타내었다. 표 2에서 MAE 차를 8까지만 나타낸 것은 본 실험에서는 MAE 차가 8이상인 경우가 없었기 때문이다. 표 1을 살펴보면, 영상에 따라 약간의 차이는 있지만 제안한 방법에서 찾은 움직임 벡터가 FSA로 찾은 움직임 벡터와 다를 확률이 약 2%로 매우 낮음을 알 수 있고, 표 2로부터 제안한 방법이 최적의 움직임 벡터를 찾지 못하더라도, 이로 인하여 발생하는 오차의 증가는 매우 적음을 알 수 있다. 즉, 제안한 방법이 움직임 벡터를 잘못 찾은 경우에, 최적의 움직임 벡터에 의한 MAE와 제안한 방법으로 찾은 움직임 벡터에 의한 MAE의 차는 1인 경우가 약 74%로 많은 부분을 차지하고 있고, 차가 3이하인 경우가 약 98%로 대부분을 차지함을 표 2로부터 확인할 수 있다. 이로부터 제안한 방법이 최적의 움직임 벡터를 찾지 못할 확률은 매우 낮으며, 최적의 움직임 벡터를 찾지 못하는 경우에도 최적의 움직임 벡터와 MAE 차가 매우 작음을 확인할 수 있다.

표 1. 최소 범위 오차로 인하여 움직임 벡터를 잘못 찾을 확률

	FLOWER GARDEN	MOBILE	SUSIE	Average
Probability of finding wrong motion vector	1.48%	1.21%	3.32%	2.00%

표 2. 움직임 벡터를 잘못 찾은 경우의 최적의 움직임 벡터와의 MAE 차의 분포

Difference of MAE	FLOWER GARDEN	MOBILE	SUSIE	Average
1	61.20%	76.34%	84.06%	73.86%
2	23.03%	18.78%	13.21%	18.34%
3	9.64%	4.27%	2.29%	5.40%
4	3.88%	0.61%	0.45%	1.64%
5	1.50%	0.00%	0.00%	0.50%
6	0.63%	0.00%	0.00%	0.21%
7	0.13%	0.00%	0.00%	0.04%
8	0.00%	0.00%	0.00%	0.00%

실험 영상에 대하여 움직임 추정 오차를 평가하기 위하여 FSA와 Li 등이 제안한 SEA 및 본 논문에서 제안한 근사 최소 범위를 이용한 블록 정합 알고리듬 (BMA using approximated lower bound; ALBMA) 즉, 제안한 알고리듬의 2 단계에서 탐색영역의 크기를 줄이지 않고, 근사 최소 범위를 이용하여 -16~+15 크기의 탐색영역에 대하여 움직임 추정을 행하는 경우의 실험영상 각 40 프레임에 대한 평균 PSNR 및 계산량을 표 3에 나타내었다. 본 실험에서 SEA의 초기 움직임 벡터는 이전 프레임의 움직임 벡터를 사용하였다.[5] 이 표에서 계산량은 FSA의 계산량에 대하여 상대적으로 나타내었다. 이 표를 살펴보면 ALBMA의 경우에 근사 최소 범위의 사용으로 인한 움직임 추정 오차의 증가가 0.03 dB에서 0.17 dB로 매우 적음을 확인할 수 있다. 또한 움직임 추정에 필요한 계산량을 살펴보면 ALBMA의 계산량이 FSA의 약 29%에서 33% 정도로 FSA의 약 40%에서 51%의 계산량을 필요로 하는 SEA에 비하여 평균적으로 약 14% 정도의 계산량 감소를 얻을 수 있음을 확인할 수 있다.

이상의 결과로부터 근사 최소 범위를 이용하여 MAE의 최소 범위를 구하여 사용하는 경우에 발생하는 오차는 매우 적으며, FSA와 거의 같은 움직임 추정 성능을 유지하면서도 많은 계산량의 감소를 얻을 수 있음을 확인할 수 있었다.

표 3. FSA, SEA 및 ALBMA의 실험 영상에 대한 평균 PSNR 및 계산량

Performance		FLOWER GARDEN	MOBILE	SUSIE
PSNR	FSA	27.42 dB	32.21 dB	36.60 dB
	SEA	27.42 dB	32.21 dB	36.60 dB
	ALBMA	27.39 dB	32.16 dB	36.43 dB
Computational complexity	FSA	100.0%	100.0%	100.0%
	SEA	51.4%	44.8%	40.2%
	ALBMA	28.8%	33.1%	32.2%

2. 제안한 고속 블록 정합 알고리듬의 움직임 추정 성능

본 절에서는 제안한 방법의 움직임 추정 성능을 움직임 보상된 영상과 원 영상간의 PSNR과 움직임 추정에 필요한 계산량을 이용하여 평가하였다. 제안한 방법의 최악의 상황 (worst case)에서의 계산량은 작은 탐색영역 전체에 대하여 블록 정합을 행하는 경우이므로 두 번째 단계에서 적용되는 작은 탐색영역의 수평 및 수직 방향의 최대 변위가 w_r이라면,

$$C_{worst} = \frac{2}{9} TN^2(2w+1)^2 + 2TN^2(2w_r+1)^2 - \frac{2}{9} TN^2(2w_r+1)^2 + 8TN^2 \quad (21)$$

와 같다. 이 경우의 두 번째 단계에서의 탐색영역 크기에 따른 계산량은 표 4에 나타내었다. 이 표를 살펴보면, 최악의 경우의 계산량은 두 번째 단계에서의 탐색영역의 크기에 따라 크게 증가함을 알 수 있다. 그러므로, 두 번째 단계에서의 탐색영역의 크기를 적절히 선택하여야 한다.

두 번째 단계에서의 탐색영역의 크기를 결정하기 위하여, 최종 움직임 벡터와 기준 MAE를 가지는 탐색점의 좌표의 차가 주어진 값 사이에 존재할 확률을 표 5에 나타내었다. 최종 움직임 벡터와 기준 MAE를 가지는 탐색점의 좌표의 차는 -31에서 31 사이의 값을 가질 수 있지만, 그 중 많은 분포를 가지는 일부분을 나타내었다. 이 표를 살펴보면 FLOWER GARDEN 영상 및 MOBILE 영상의 경우에는 최종 움직임 벡터와 기준 MAE를 가지는

표 4. 두 번째 단계에서의 탐색영역 크기에 따른 최악의 경우의 계산량

Search area size of second path	3×3	9×9	15×15	21×21	27×27	32×32
Computational complexity	12.6%	18.8%	31.3%	50.1%	75.1%	100%

표 5. 최종 움직임 벡터와 기준 MAE를 가지는 탐색점의 좌표의 차가 주어진 값 사이에 존재할 확률

Given area		-1~+1	-4~+4	-7~+7	-10~+10	-13~+13
Probability	FLOWER GARDEN	67%	81%	86%	89%	91%
	MOBILE	67%	86%	89%	91%	93%
	SUSIE	34%	60%	66%	69%	72%
	Average	56%	76%	80%	83%	85%

탐색점의 좌표의 차가 -7에서 +7 사이에 있을 확률이 86% 및 89%로서 많은 부분이 여기에 존재한다. SUSIE 영상의 경우에는 매우 평탄한 배경을 가지고 있으므로, 각 탐색점에서의 MAE가 별로 차이를 나타내지 않기 때문에 최종 움직임 벡터가 넓게 분포함을 알 수 있다. 표 4와 표 5를 종합적으로 살펴보면 탐색영역의 크기의 증가에 따라 최악의 경우의 계산량의 증가량은 더욱 커지고 있고, 최종 움직임 벡터가 탐색영역 내에 존재할 확률의 증가량은 점점 작아지는 것을 알 수 있다. 그러므로, 제안한 방법에서는 계산량의 증가량과 최종 움직임 벡터가 탐색영역 내에 존재할 확률 고려하여 두 번째 단계에서의 탐색영역의 크기를 15×15로 결정하였다.

또한, 제안한 방법에서는 이웃 탐색점에서의 MAE를 이용하여 현재 탐색점에서의 MAE의 최소 범위를 구한 뒤, 블록 정합이 필요한 탐색점에서만 블록 정합을 행하므로 식 (20)에서와 같이 MAE의 최소 범위가 MAE_{ref}보다 작은 탐색 점의 총 개수 F에 따라 달라지며, 표 4에 나타난 최악의 경우의 31.3%에 비하여 적은 계산량을 필요로 한다. 제안한 방법의 실험 영상 각 40 프레임에 대한 평균 계산량 및 PSNR을 MPEG-1 SM3에 권고 되었던 TSS방법과 Jong 등이 제안한 4TSS 및 5TSS 방법과 비교하여 표 6에 나타내었다.

TSS 방법은 작은 탐색영역에 적합하도록 만들어진 방법으로서 탐색영역이 커짐에 따라 첫 번째 단계에서의 탐색 변위가 커지기 때문에 계산량은 크게 줄일 수 있었지만 움직임 추정 오차도 매우 커지게 된다. 이를 표 6으로부터 확인할 수 있는데, 이 표로부터 TSS 방법은 3.7%의 매우 적은 계산량을 가지지만 움직임 추정 오차의 증가가 약 3.2 dB에서 1.8 dB로 매우 큼을 확인 할 수 있다. 또한, TSS 방법의 성능을 개선하기 위하여 제안된 4TSS 방법과 5TSS 방법을 살펴보면, 먼저 4TSS 방법의 움직임 추정 오차의 증가는 6.8 dB에서 1.8 dB로 TSS 방법에 비하여 움직임추정 성능이 오히려 떨어짐을 알 수 있다. 이는 4TSS 방법의 경우에는 큰 탐색영역에 대하여 움직임 추정을 행하기 위하여 탐색영역을 중첩되지 않은 4개의 영역으로 나눈 뒤, 각 영역에 TSS를 적용하므로 탐색영역의 중심부에 움직임 벡터가 존재하는 경우, 즉 움직임이 크지 않은 경우에 움직임 벡터를 제대로 찾지 못하기 때문이다. 이러한 작은 움직임을 가지는 블록에 대하여 보다 정확한 움직임 추정을 위하여, 탐색영역의 중심부에 TSS를 한번 더 적용하는 5TSS 방법에 대하여 살펴보면, TSS 방법이나 4TSS 방법의 움직임 추정 성능을 어느 정도 개선하였으나 움직임 추정 오차의 증가가 약 0.53 dB에서 1.47 dB 정도의 큰 움직임 추정 오차를 가짐을 알 수 있다. 특히, SUSIE 영상과 같은 단순한 영상에 대하여서는 어느 정도 좋은 움직임 추정 성능을 나타내지만, FLOWER GARDEN 영상과 같이 복잡한 영상에 대하여서는 약 1.5 dB 정도의 큰 움직임 추정 오차

표 6. 평균 PSNR 및 계산량 비교

		FSA	Proposed method	5TSS	4TSS	TSS
PSNR	FLOWER GARDEN	27.42 dB	27.15 dB	25.94 dB	20.65 dB	24.19 dB
	MOBILE	32.22 dB	32.15 dB	31.44 dB	28.30 dB	30.02 dB
	SUSIE	36.56 dB	36.18 dB	36.03 dB	34.77 dB	34.74 dB
Computational complexity	FLOWER GARDEN	100%	16.9%	13.9%	11.1%	3.7%
	MOBILE	100%	19.3%	13.9%	11.1%	3.7%
	SUSIE	100%	17.7%	13.9%	11.1%	3.7%
	Average	100%	18.0%	13.9%	11.1%	3.7%

의 증가가 발생함을 알 수 있다.

이에 비하여 제안한 방법은 FSA의 약 17%에서 19%정도의 계산량을 가지면서도 움직임 추정 오차의 증가량이 약 0.07 dB에서 0.38 dB 정도이다. 즉, 계산량은 5TSS보다 조금 많은 계산량을 가지지만, 움직임 추정 성능은 월등히 좋음을 확인할 수 있다. 또한, 이 표로부터 SUSIE 영상과 같이 단순한 영상에 대한 제안한 방법의 움직임 추정 오차가 FLOWER GARDEN 영상과 같은 복잡한 영상이나 MOBILE 영상처럼 에지가 많이 존재하는 영상보다 SUSIE 영상과 같이 단순한 영상에서의 움직임 추정 오차의 증가가 더 큰 것을 확인할 수 있다. 이는 표 5에서 알아본 바와 같이 SUSIE 영상의 경우에는 매우 평탄한 배경을 가지고 있으므로, 각 탐색점에서의 MAE가 큰 차이를 나타내지 않기 때문에 최종 움직임 벡터가 넓게 분포하므로 탐색영역의 크기를 줄이는 경우에 작은 오차들이 누적되기 때문인 것으로 판단된다.

또한, 제안한 방법의 계산량은 식 (20)에서와 같이 MAE의 최소 범위가 MAE_{req}보다 작은 탐색점의 총 개수 F에 따라 달라질 수 있다. 이를 살펴보기 위하여 각 실험 영상의 프레임별 계산량을 그림 5에 나타내었다. 이 그림에서 계산량 축의 최대 값과 최소 값은 제안한 방법의 최악의 경우의 계산량과 5TSS 방법의 계산량의 근사값으로 하였다. 이 그림을 살펴보면 제안한 방법의 계산량이 각 프레임에 따라 달라질 수 있음을 나타내고 있으나, SUSIE 영상과 MOBILE 영상의 경우에는 계산량의 최대 값과 최소 값의 차가 2% 이내 이고, FLOWER GARDEN 영상의 경우에는 3% 이내로서 입력 영상에 따른 계산량의 변화가 별로 없는

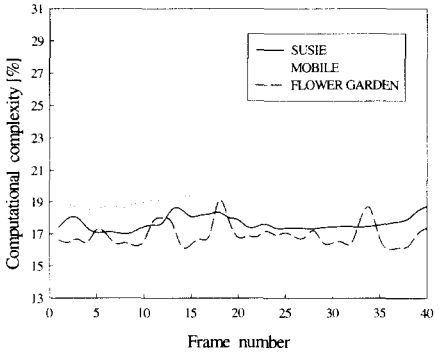

그림 5. 제안한 방법의 실험영상 각 프레임에 따른 계산량

매우 안정된 성능을 보임을 알 수 있다.

이상의 결과로부터 제안한 방법이 적은 계산량을 가지면서도 FSA와 비슷한 움직임 추정 성능을 나타냄을 확인할 수 있었다.

V. 결 론

본 논문에서는 이웃 탐색점에서의 평균 절대치 오차 및 탐색영역 줄임을 이용한 고속 블록 정합 알고리듬을 제안하였다. 제안한 방법은 두 단계로 구성되어있다. 제안한 방법의 첫 번째 단계에서는 탐색영역을 3×3 크기의 영역으로 겹치지 않게 나눈 뒤, 각 영역의 중심 탐색점에 대하여 블록 정합을 행하여 MAE를 구하고, 이들 중 가장 작은 MAE를 기준 MAE로 정하였다. 또한, 두 번째 단계에서는 기준 MAE로 결정된 탐색점을 중심으로 탐색영역의 크기를 줄인 후, 각 3×3 영역의 중심 탐색점에서의 MAE를 이용하여 각 나머지 탐색점에서의 MAE의 최소 범위를 구한 뒤, 최소 범위가 기준 MAE 보다 작은 탐색점에 대하여서만 블록 정합을 행함으로써 고속으로 움직임을 추정하였다. 제안한 방법의 성능 평가를 위한 모의 실험을 통하여 제안한 방법은 FSA의 약 17%에서 19%정도의 적은 계산량을 가지면서도 움직임 추정 오차의 증가량이 약 0.07 dB에서 0.38 dB 정도로 우수한 성능을 나타냄을 확인할 수 있었다.

참 고 문 헌

[1] *ITU-T Recommendation H.*261, "Video codec for audiovisual services at p×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] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing,* vol. 4, no. 1. pp. 105-107, Jan. 1995.

[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] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 7, no. 2, pp. 429-433, Apr. 1997.

[12] H.-M. Jong, L.-G. Chen, and T.-D. Chiueh, "Accuracy Improvement and Cost Reduction of 3-Step Search Block Matching Algorithm for Video Coding," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 4, no. 1, pp. 88-90, Feb. 1994.

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

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

[15] *CCITT SGXV Working Party* XV/4, "Description of Reference Model 8 (RM8) CCITT SGXV Doc. #525," 1989.

[16] K. H.-K. Chow and M. L. Liou, "Genetic Motion Search Algorithm for Video Compression," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 3, no. 6, pp. 440-445, Dec. 1993.

정 원 식(Won-Sik Cheong) 정회원
통신학회논문지 제 24권 1B호 참조

이 법 기(Bub-Ki Lee) 정회원
통신학회논문지 제 24권 1B호 참조

한 찬 호(Chan-Ho Han) 정회원
1990년 2월 : 경북대학교 전자공학과 졸업 (공학사)
1992년 2월 : 경북대학교 대학원 전자공학과 졸업 (공학석사)
1992년~1997년 8월 : 현대전자 미디어 연구소 연구원
1997년 8월~현재 : 경북대학교 대학원 전자공학과 박사과정 재학중
<주관심 분야> 영상신호처리, 비디오 공학, 음향 공학

권 성 근(Seong-Geun Kwon) 정회원
1996년 2월 : 경북대학교 전자공학과 졸업 (공학사)
1998년 2월 : 경북대학교 대학원 전자공학과 졸업 (공학석사)
1998년 3월~현재 : 경북대학교 대학원 전자공학과 박사과정 재학중
<주관심 분야> 영상신호처리 및 시스템 설계

장 종 국(Jong-Kook Jang) 정회원
통신학회논문지 제 25권 1B호 참조

이 건 일(Kuhn-Il Lee) 정회원
통신학회논문지 제 24권 1B호 참조