논문 02-27-9A-8

고속 블록 정합 움직임 추정을 위한 개선된 MSEA

준회원 안 태 경*, 정회원 문 용 호**

An Improved Multilevel Successive Elimination Algorithm for Fast Full-Search Block Matching

Tae-gyoung Ahn* *Associate Member*, Yong-ho Moon** *Regular Member*

요 약

비디오 영상 압축에서 움직임 추정은 많은 계산량을 필요로 한다고 알려져 있다. 최근에 적은 계산량으로 최적의 움직임 추정을 수행하는 MSEA가 제안되었다. MSEA는 레벨에 따라 분할된 부 블록들로부터 계산되어진 최소 범위를 이용하여 불필요한 정합 척도 계산을 순차적으로 제거함으로써 계산량을 감소시켰다. 본 논문에서는 첫 번째 레벨에서의 최소 범위를 이용하여 각 레벨에서의 최소 범위를 계산 할 수 있음을 이론적으로 보인다. 그리고 이와 같은 이론적 근거를 토대로 각 레벨에서의 최소 범위 계산에 소요되는 계산량을 감소시키는 개선된 고속 전역 탐색 블록 정합 알고리즘을 제안한다. 컴퓨터 모의 실험을 통해 제안 방식이 MSEA에 비해 적은 계산량으로 최적의 움직임 추정을 수행함을 확인하였다.

ABSTRACT

It is well known that a lot of computations are used for the motion estimation process in video coding. Recently, the MSEA was proposed to reduce the computation cost required for block matching in the motion estimation while keeping their accuracy. The MSEA evaluates lower bounds for the matching criteria for subdivided blocks in order to reduce the number of search positions. In this paper, we show that the lower bounds are calculated efficiently by using a lower bound that is already calculated for the first level. Based on this result, we propose a new fast full search motion estimation algorithm, in which the computation cost required for calculation of the lower bounds is reduced. The simulation results show that the proposed algorithm achieves computational saving as compared with the MSEA.

I. 서 론

다양한 비디오 영상압축 방식에서 사용되고 있는 움직임 추정은 비디오 신호를 압축하여 전송, 저장함에 있어 매우 중요한 역할을 한다. MPEG-1, MPEG-2, MPEG-4, H.261, H.263 등과 같은 비디오 압축 표준안에서 움직임 추정은 동영상에 존재하는 시간적 중복성을 제거하기 위해 사용된다. 현재 널리 사용되고 있는 움직임 추정 방식으로는 하드웨어 구현이 용이한 블록 정합 알고리즘 (block matching algorithm; BMA)이 있다[1]. 이때 블록 정합을 위한 정합 척도로는 *SAD* (sum of absolute difference)가 널리 사용된다[2].

BMA 중 전역 탐색 BMA는 탐색 점 (search position) 전체에 대하여 탐색을 수행한다. 따라서 이 방식은 최적의 움직임 벡터를 보장하지만 계산량이 많은 단점을 지닌다[3]. 이를 해결하기 위하여 여러 가지 고속 움직임 추정 기법들이 제안되어 왔다. 대표적인 고속 BMA로는 3단계 탐색 (three-step search; TSS)[4], 2차원 로그 탐색 (two

* 부산대학교 전자공학과 영상통신실험실(tgahn@pusan.ac.kr)
** 부산대학교 산업자동화 및 정보통신분야 인력양성 사업단(BK21)(yhmoon5@pusan.ac.kr)
논문번호 : 020289-0704, 접수일자 : 2002년 7월 4일

dimensional logarithm search; 2-D LOG)[5], 그리고 교차 탐색 (cross search; CS)[6] 등이 있다. 그러나 이 기법들은 계산량이 감소하는 이점에도 불구하고 탐색영역의 일부에서만 탐색을 수행하기 때문에 국부 최소 (local minimum)에 빠질 수 있는 문제점을 가진다.

이에 반하여 고속 전역 탐색 BMA는 적은 계산량으로 최적의 움직임 벡터를 얻기 위하여 연구되어 왔다. 대표적인 고속 전역 탐색 BMA로는 연속 제거 알고리즘 (successive elimination algorithm; SEA)[1], 다중레벨 연속 제거 알고리즘 (multilevel successive elimination algorithm; MSEA)[7], 그리고 확장 연속 제거 알고리즘 (extended successive elimination algorithm; ESAE)[2] 등이 있다. SEA는 압축할 블록과 후보 블록간의 실제 *SAD* 계산에 앞서 *SAD*의 최소 범위 (lower bound)를 먼저 계산한다. 만약 최소 범위가 현재까지 탐색된 최소 *SAD*인 SAD_{Min} 보다 클 경우, 실제 *SAD* 계산은 무의미하다. SEA는 이와 같은 불필요한 *SAD* 계산을 제거함으로써 계산량을 감소시켰다. MSEA는 SEA를 다중레벨로 확장시킨 방식이다. 정합 할 블록을 부 블록들로 분할할수록 여기서 얻어진 최소 범위는 점차 실제 *SAD*에 근접한다는 것이 이론적으로 알려져 있다[7]. 이와 같은 사실을 근거로 MSEA는 레벨에 따라 분할된 부 블록들로부터 계산되어진 최소 범위를 순차적으로 적용하여 불필요한 *SAD* 계산을 제거함으로써 수행속도를 증가시켰다. 한편 MSEA에서 제거되지 않은 *SAD* 계산은 기존 방식을 따라 수행해야 한다. ESEA에서는 이와 같은 경우 이미 계산된 첫 번째 레벨에서의 최소 범위를 이용하여 *SAD* 계산을 효율적으로 수행하였다.

그런데 MSEA에서 각 레벨에서의 최소 범위와 *SAD* 사이에는 유사성이 있다. 먼저, 각 레벨에서의 최소 범위 계산과 *SAD* 계산은 본질적으로 동일한 과정이다. 각 레벨에 속하는 부 블록들간의 *SAD*가 해당 레벨에서의 최소 범위가 된다. 또한, MSEA에서 각 레벨에 속한 부 블록들의 절대치 합을 모두 더한 총합과 전체 블록의 총합이 같다. 즉, 최소 범위 계산에 사용되는 부 블록들의 절대치 합들을 모두 더하면 실제 *SAD* 계산에 사용되는 전체 블록의 총합과 동일하다. 그런데 ESEA에서 첫 번째 레벨에서의 최소 범위는 전체 블록의 총합을 이용하여 계산되어진다. 다시 말해서, 부 블록과 전체 블록의 총합이 동일하고 ESEA에서는 이 총합을 이용하여 효율적인 *SAD* 계산을 수행한다. 그런데 앞서 언급한 바와 같이 최소 범위와 *SAD* 계산이 동일하므로 최소 범위 계산에도 전체 블록의 총합을 이용할 수 있음을 짐작할 수 있다. 이상의 사실들로부터 첫 번째 레벨에서의 최소 범위는 *SAD* 계산뿐만 아니라 각 레벨에서의 최소 범위 계산에도 이용될 수 있음을 짐작할 수 있다. 본 논문에서는 첫 번째 레벨에서의 최소 범위를 이용하여 각 레벨에서의 최소 범위를 계산 할 수 있음을 이론적으로 보인다. 그리고 이와 같은 이론적 근거를 토대로 각 레벨에서의 최소 범위 계산에 소요되는 계산량을 감소시키는 개선된 고속 전역 탐색 BMA를 제안한다.

제안 알고리즘의 절차는 다음과 같다. 먼저 첫 번째 레벨에서의 최소 범위를 계산한다. 이 최소 범위가 SAD_{Min}보다 큰 경우 실제 *SAD* 계산을 수행하지 않고 다음 탐색 점으로 이동한다. 반대로 첫 번째 레벨에서의 최소 범위가 SAD_{Min}보다 작은 경우 두 번째 레벨에서의 최소 범위를 계산하고 이를 SAD_{Min}과 비교한다. 두 번째 레벨에서의 최소 범위는 전체 블록에서 분할된 부 블록들로부터 계산된다. 이때, 첫 번째 레벨에서 이미 계산된 최소 범위를 이용하여 두 번째 레벨에서의 최소 범위를 효율적으로 계산한다. 이상의 과정을 나머지 레벨에 대해 반복한다. 제안 알고리즘은 첫 번째 레벨을 제외한 나머지 레벨에서의 최소 범위 계산을 효율적으로 수행함으로써 계산량을 절감한다.

본 논문은 다음과 같이 구성된다. 먼저 II장에서 기존의 MSEA와 ESEA에 대하여 보다 자세히 설명하고 이들의 장단점을 살펴본다. 그리고 III장에서는 제안 알고리즘에 대하여 설명한다. IV장에서는 모의실험을 통하여 제안 알고리즘의 우수성을 객관적으로 검증한다. 끝으로 V장에서 결론을 맺는다.

II. 기존의 고속 전역 탐색 BMA

움직임 추정을 위한 블록 정합 방식은 현재 프레임에서 압축할 매크로 블록과 가장 유사한 블록의 위치를 이전 프레임의 탐색 영역 (search region)내에서 찾는 것이다. 식 (1)은 (m, n) 위치에 존재하는 탐색 점에 대한 *SAD*를 나타낸다.

$$SAD(m,n)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|f(i,j,t)-f(i-m,j-n,t-1)| \quad (1)$$

여기서 $f(i,j,t)$와 $f(i,j,t-1)$는 각각 현재 프레임과 이전 프레임에서 (i,j) 위치의 화소 밝기를 나타낸다. 매크로 블록의 크기를 $N\times N$, 탐색 영역의 크기를 $(2S+N)\times(2S+N)$이라 할 때 전역 탐색 BMA는 탐색 영역 내의 모든 후보 블록들에 대하여 최소 SAD를 가지는 블록의 위치를 찾는 것이다. 그림 1은 이러한 과정을 보여준다. 가장 유사한 블록의 위치, 즉 움직임 벡터 (x,y)는 식 (2)와 같이 결정된다.

$$(x,y)=\arg\min_{(m,n)} SAD(m,n) \tag{2}$$

이러한 전역 탐색 BMA는 최적의 움직임 추정을 보장하지만 계산량이 많은 문제점을 지니고 있다.

1. MSEA

전역 탐색 BMA에서 계산량이 많은 문제점에 대한 해결책으로 MSEA가 제안되었다. MSEA는 각 레벨별로 순차적으로 SAD 계산 여부를 검사함으로써 기존의 불필요한 SAD 계산을 제거하는 방식이다. 그림 2는 MSEA에서 레벨 0과 레벨 1에 대한 압축할 블록과 후보 블록을 표시한 것이다. 그림 2에서 레벨 0의 R_0과 $M_0(m,n)$은 각각 압축할 블록과 후보 블록에 대한 전체 블록의 절대치 합(sum norm)을 나타낸다. 블록의 절대치 합은 최소 범위 계산에 이용되며 그 정의 식은 다음과 같다.

$$R_0=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|f(i,j,t)| \tag{3}$$

$$M_0(m,n)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|f(i-m,j-n,t-1)| \tag{4}$$

그림 2에서 $M_0(m,n)$은 서술의 편의를 위하여 M_0으로 표기한 것이다. 레벨 1은 전체 블록을 4개로 분할하여 각 부 블록들에 대한 절대치 합을 표시한 것이다.

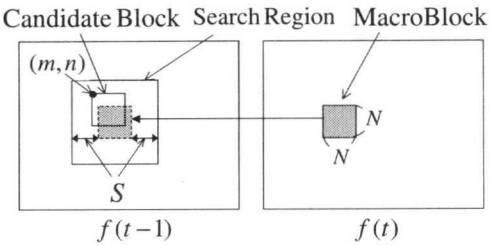


그림 1. 블록 기반 움직임 추정

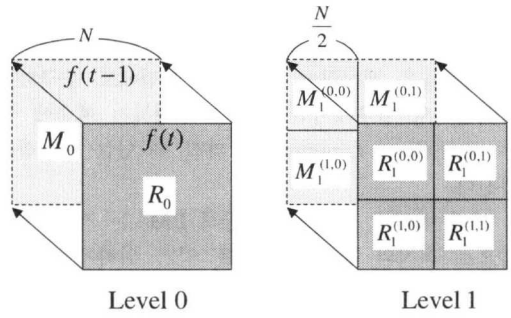


그림 2. 레벨에 따른 부 블록의 절대치 합들

$$R_1^{(u,v)}=\sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{\frac{N}{2}-1}\left|f(i+u\frac{N}{2},j+v\frac{N}{2},t)\right| \tag{5}$$

$$M_1^{(u,v)}(m,n)=\sum_{i=0}^{\frac{N}{2}-1}\sum_{j=0}^{\frac{N}{2}-1}\left|f(i+u\frac{N}{2}-m,j+v\frac{N}{2}-n,t-1)\right| \tag{6}$$

여기서 $0\le u,v\le 1$이다. 이상의 정의로부터 레벨 1은 부 블록의 절대치 합을 그 밝기 값으로 하는 4개의 화소로 구성된 2×2 블록으로 간주할 수 있다. 동일한 관점에서 볼 때 레벨 0은 1×1 블록으로 간주할 수 있다. 이들 블록에 대한 SAD는 다음과 같이 나타내어진다.

$$SSAD_0=|R_0-M_0| \tag{7}$$

$$SSAD_1=\sum_{u=0}^{1}\sum_{v=0}^{1}|R_1^{(u,v)}-M_1^{(u,v)}| \tag{8}$$

한편 $SSAD_0$, $SSAD_1$, 그리고 $SAD(m,n)$ 사이에 수학적 부등식을 적용하면 식 (9)와 같은 관계가 성립한다[7][8].

$$SSAD_0\le SSAD_1\le SAD(m,n) \tag{9}$$

식 (9)는 레벨 0에서 $SAD(m,n)$의 최소 범위가 $SSAD_0$임을 보여준다. 또한 레벨 1에서는 $SSAD_1$이 $SAD(m,n)$의 최소 범위임을 보여준다. 여기서, $SSAD_1$이 $SSAD_0$보다 $SAD(m,n)$에 근접하는 것을 알 수 있다. 따라서 레벨이 커질수록 최소 범위가 실제 $SAD(m,n)$에 가까워짐을 알 수 있다. 이와 같은 사실을 바탕으로 다음의 두 경우를 살펴보자.

$$SAD_{Min}\le SSAD_0\le SSAD_1\le SAD(m,n) \tag{10}$$

$$SSAD_0 \le SAD_{Min} \le SSAD_1 \le SAD(m,n) \quad (11)$$

식 (10)과 식 (11)의 경우에는 (m,n)이 가장 유사한 블록의 위치 (x,y)가 될 수 없음을 알 수 있다. 그리고 이것은 $SAD(m,n)$을 계산하지 않고도 SAD_{Min}을 $SSAD_0$, $SSAD_1$과 비교함으로써 쉽게 판단할 수 있다. 이러한 사실에 기초하여 각 레벨에서의 최소 범위를 SAD_{Min}과 순차적으로 비교함으로써 불필요한 $SAD(m,n)$ 계산을 제거할 수 있다. 이상에서 설명한 방식을 임의의 다중레벨로 일반화 한 것이 MSEA이다. 식 (12)는 식 (9)를 다중레벨로 일반화 한 것이다.

$$SSAD_0 \le SSAD_1 \le \cdots \le SSAD_l \le \cdots \le SSAD_L \le SAD(m,n) \quad (12)$$

여기서 l은 다중레벨 중 임의의 한 레벨이며, L은 분할 가능한 최대 레벨을 나타낸다. MSEA는 계산량이 적은 낮은 레벨에서부터 계산량이 많은 높은 레벨 순으로 순차적으로 검사를 수행함으로써 계산량을 절감한다. 그러나 만약 $SSAD_L$보다 SAD_{Min}이 클 경우 이들 사이의 대소 관계는 알 수 없다. 따라서 이 경우에는 기존의 $SAD(m,n)$을 계산하여 새로운 SAD_{Min}을 구하여야 한다. SEA는 MSEA에서 $L=0$인 경우에 해당한다.

그런데 MSEA는 다중레벨에 대한 절대치 합의 계산에 필요한 부가 계산량을 최소로 하기 위해 전체 영상 한 장에 대해 각 레벨 별로 절대치 합들을 미리 계산하여 저장해 둔다. MSEA에서 절대치 합 계산에 필요한 부가 계산량은 전체 계산량의 약 10%정도라고 알려져 있다[7]. 그리고 절대치 합을 저장해 놓기 위한 메모리의 크기는 영상 한 장 크기의 약 3~5배 정도가 된다.

2. ESEA

MSEA에서 $SSAD_L$보다 SAD_{Min}이 클 경우 식 (1)과 같이 기존의 $SAD(m,n)$을 계산해야만 한다. 이와 같은 경우 이미 계산된 R_0-M_0을 이용하여 효율적으로 $SAD(m,n)$ 계산을 할 수 있는 방법을 ESEA에서 제시하였다. 레벨 0에서 압축할 블록과 후보 블록 사이의 절대치 합의 차는 식 (13)과 같이 전체 블록 내 화소 값의 차로 나타낼 수 있다.

$$R_0 - M_0 = \sum_{k=0}^{N^2-1} d(k) \quad (13)$$

여기서 k는 수식 전개의 편의를 위해 (i,j)를 1차원으로 변환한 것으로 $k=j\times N+i$로 정의된다. 따라서 $d(k)$는 $d(k)=f(k,t)-f(k,t-1)$로 나타내어지며 두 블록내의 k번째 화소 간의 차를 의미한다. 구하고자 하는 $SAD(m,n)$과 $|R_0-M_0|$ 간에는 다음과 같은 관계가 성립된다. 먼저 $R_0-M_0<0$인 경우에는

$$\begin{aligned} SAD(m,n) - |R_0 - M_0| &= \sum_{k=0}^{N^2-1} |d(k)| + \sum_{k=0}^{N^2-1} d(k) \\ &= \sum_{d(k)\ge 0} d(k) - \sum_{d(k)<0} d(k) \\ &\quad + \sum_{d(k)\ge 0} d(k) + \sum_{d(k)<0} d(k) \\ &= 2 \sum_{d(k)\ge 0} d(k), \end{aligned} \quad (14)$$

반대로 $R_0-M_0\ge 0$인 경우에는

$$SAD(m,n) - |R_0 - M_0| = -2 \sum_{d(k)<0} d(k) \quad (15)$$

와 같다. 식 (14)와 식 (15)는 R_0-M_0과 부호가 반대인 $d(k)$들을 $|R_0-M_0|$에 더함으로써 $SAD(m,n)$을 구할 수 있음을 보여준다.

한편 기존의 $SAD(m,n)$ 계산 방식과 ESEA에서의 $SAD(m,n)$ 계산 방식의 계산량을 비교해 보면 ESEA에서 계산량이 적게 소요됨을 추정할 수 있다. 기존 방식에서는 모든 화소에 대해 뺄셈, 절대치, 덧셈 연산을 각각 수행해야 한다. 그러나 ESEA에서는 $d(k)$ 계산에 앞서 R_0와 M_0의 대소 관계를 비교하여 $d(k)$의 부호를 예측한다. 그리고 예측된 $d(k)$의 부호와 R_0-M_0의 부호가 반대인 경우에 대해서만 $d(k)$를 계산한다. R_0-M_0과 부호가 반대인 $d(k)$의 개수는 전체 $d(k)$ 개수, 즉 N^2보다 항상 적을 것이다. 그러므로 ESEA에서는 일부 $d(k)$에 대해서만 연산을 수행하기 때문에 기존 $SAD(m,n)$ 계산 방식에 비하여 보다 효과적으로 $SAD(m,n)$을 얻는다.

Ⅲ. 제안 알고리즘

앞에서 설명한 바와 같이 ESEA는 MSEA에서 $SAD(m,n)$을 계산하여야 할 경우 이미 계산된 R_0-M_0과 일부 $d(k)$만을 이용함으로써 계산량을 감소시켰다. 그런데 $SAD(m,n)$과 $SSAD_l$ 사이에 존재하는 유사성은 $SSAD_l$의 계산에도 R_0-M_0을 이용할 수 있음을 짐작하게 해준다.

MSEA에서 정의된 R_0과 M_0은 식 (16)과 식 (17)로 재정의 될 수 있다.

$$R_0 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|f(i,j,t)| = \sum_{u=0}^{1}\sum_{v=0}^{1}|R_1^{(u,v)}| \quad (16)$$

$$M_0 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|f(i,j,t-1)| = \sum_{u=0}^{1}\sum_{v=0}^{1}|M_1^{(u,v)}| \quad (17)$$

$R_1^{(u,v)}$와 $M_1^{(u,v)}$가 2×2 블록이라고 간주하면 최소 범위 계산식과 정합 척도 계산식이 같음을 알 수 있다. 이러한 사실은 $SSAD_1$ 계산에도 R_0-M_0을 이용할 수 있음을 예상할 수 있게 해준다.

$SSAD_1$의 계산에 R_0-M_0을 이용하기 위하여 $SSAD_1$과 R_0-M_0 사이의 차를 계산해 보자. 먼저, 레벨 1에서의 부 블록에 대한 $D_1(k)$를 다음과 같이 정의하자.

$$D_1(k) = R_1^k - M_1^k \quad (18)$$

이때 $k=v\times2+u$가 된다. 식 (16), 식 (17), 그리고 식 (18)로부터 식 (13)은 다음과 같이 변환될 수 있다.

$$R_0 - M_0 = \sum_{k=0}^{3}(R_1^k - M_1^k) = \sum_{k=0}^{3}(D_1(k)) \quad (19)$$

$R_0-M_0<0$인 경우, 식 (14)에서와 같은 과정을 적용할 경우 식 (20)이 유도된다.

$$\begin{aligned} SSAD_1 - |R_0-M_0| &= \sum_{k=0}^{3}|D_1(k)| + \sum_{k=0}^{3}D_1(k) \\ &= \sum_{D_1(k)\geq0}D_1(k) - \sum_{D_1(k)<0}D_1(k) \\ &\quad + \sum_{D_1(k)\geq0}D_1(k) + \sum_{D_1(k)<0}D_1(k) \\ &= 2\sum_{D_1(k)\geq0}D_1(k) \end{aligned} \quad (20)$$

또한 $R_0-M_0\geq0$인 경우에는 식 (21)이 유도된다.

$$SSAD_1 - |R_0-M_0| = -2\sum_{D_1(k)<0}D_1(k) \quad (21)$$

식 (20)과 식 (21)은 $SSAD_1$ 역시 R_0-M_0를 이용하여 구할 수 있음을 보여준다. 이러한 사실을 다중레벨에 대해 일반화하면 식 (22)와 같다.

$$SSAD_l = \begin{cases} |R_0-M_0| + 2\sum_{D_l(k)\geq0}D_l(k) & : R_0-M_0<0 \\ |R_0-M_0| - 2\sum_{D_l(k)<0}D_l(k) & : R_0-M_0\geq0 \end{cases} \quad (22)$$

식 (22)는 모든 레벨에 대한 $SSAD_l$들이 R_0-M_0을 이용하여 효율적으로 계산될 수 있음을 보여준다.

그런데 식 (22)를 $SSAD_l$ 계산에 직접 적용하는 경우 R_0-M_0과 부호가 반대인 $D_l(k)$들을 모두 더한 결과에 2를 곱하여야 한다. 이와 같은 경우 $SAD(m,n)$이 계산 과정에서 SAD_{Min}보다 커지면 계산을 중단하는 방식인 PDE (partial distortion elimination)[9]를 적용할 수 없다. 따라서 $SAD(m,n)$ 계산이 항상 끝까지 수행되어야 하기 때문에 효율이 떨어진다. 이를 해결할 한가지 방법은 $D_l(k)$에 2를 먼저 곱한 후 더하는 것이다. 그러나 이 경우에는 2의 곱셈을 위한 연산이 R_0-M_0과 부호가 반대인 $D_l(k)$의 개수만큼 더 필요하게 되는 문제점을 가진다. ESEA에서는 $2|D_l(k)|$을 위한 참조 테이블을 사용하여 이를 해결하였다[2]. 그런데 식 (22) 방식에 참조 테이블을 적용하는 것이 쉽지 않다. 상위 레벨로 갈수록, 즉 부 블록의 수가 적어질수록 부 블록에 대한 절대치 합은 점점 커지게 된다. 예를 들어 전체 블록에서 화소간의 차 $|d(k)|$의 최대값은 255이다. 그러나 레벨 1에서 부 블록의 절대치 합 $R_1^{(u,v)}$의 최대값은 $N=16$인 경우 16,384가 된다. 따라서 $2|D_l(k)|$을 위한 참조 테이블은 $2|d(k)|$을 위한 참조 테이블에 비하여 훨씬 커져야만 한다. 이와 같이 레벨이 많아질수록 참조 테이블이 급격히 커짐으로 인하여 많은 메모리가 필요하게 되는 문제점이 발생한다.

본 논문에서는 식 (22)를 식 (23)과 같은 형태로 변환함으로써 참조 테이블을 사용하지 않고도 효율적인 계산이 가능하게 하였다.

$$\begin{cases} \dfrac{SSAD_l - |R_0-M_0|}{2} = \sum_{D_l(k)\geq0}D_l(k) & : R_0-M_0<0 \\ \dfrac{|R_0-M_0| - SSAD_l}{2} = \sum_{D_l(k)<0}D_l(k) & : R_0-M_0\geq0 \end{cases} \quad (23)$$

식 (23)은 2를 곱하지 않고 R_0-M_0과 부호가 반대인 $D_l(k)$만을 더해가면서 조건을 비교할 수 있음을 보여준다. 식 (23)을 사용함으로써 참조 테이블을 사용하지 않고 효율적인 계산이 가능함을 알 수 있다. 그런데 참조 테이블을 사용하지 않는 대신 식 (23)의 왼쪽 항 계산을 위한 부가적인 계산량이 필요하다. 그러나 식 (23)의 왼쪽 항은 각 레벨에서

한번씩만 계산하기 때문에 참조 테이블 대신 소요되는 부가 계산량은 매우 적다. 이상의 사실로부터 실제 SAD_{Min}과의 비교를 위한 조건을 구해보면 다음의 식 (24)와 같다.

$$\begin{cases} \dfrac{SAD_{Min}-|R_0-M_0|}{2} < \sum_{D_i(k)\geq 0} D_i(k) & : R_0-M_0<0 \\ \dfrac{|R_0-M_0|-SAD_{Min}}{2} < \sum_{D_i(k)<0} D_i(k) & : R_0-M_0\geq 0 \end{cases} \quad (24)$$

이와 같은 이론적 근거를 토대로 본 논문에서는 모든 $SSAD_i$의 계산에 식 (24)를 적용함으로써 보다 효율적으로 정합 블록을 탐색할 수 있는 방식을 제안한다.

그림 3은 제안 알고리즘의 전체적인 흐름도를 나타낸다. 먼저 전체 블록의 총합을 이용하여 첫 번째 레벨에서의 최소 범위 $SSAD_0$을 계산한다. $SSAD_0$이 SAD_{Min}보다 큰 경우 실제 SAD 계산을 수행하지 않고 다음 탐색 점으로 이동한다. 반대로 $SSAD_0$이 SAD_{Min}보다 크지 않은 경우에는 두 번째 레벨에서의 최소 범위 $SSAD_1$을 계산하고 이를 SAD_{Min}과 비교한다. $SSAD_1$은 전체 블록에서 분할된 부 블록들로부터 계산된다. 이때, 첫 번째 레벨에서 이미 계산된 $SSAD_0$을 이용하여 두 번째 레벨에서의 최소 범위를 효율적으로 계산한다. 마찬가지로 세 번째 레벨에서의 최소 범위 $SSAD_2$도 $SSAD_0$을 이용하여 효율적으로 계산한다. 이상의 과정을 나머지 레벨에 대해 반복한다. 제안 알고리즘은 첫 번째 레벨을 제외한 나머지 레벨에서의 최소 범위 계산을 효율적으로 수행함으로써 계산량을 절감한다.

Ⅳ. 실험 결과

제안 알고리즘의 성능을 평가하기 위해 SIF(352×240) 형식의 5가지 비디오 실험 영상을 MPEG4 부호화기로 부호화하였다. 비디오 실험 영상은 모두 30Hz인 것을 사용하였고, 각각 100프레임씩 실험하여 평균값을 취하였다. 각 매크로 블록의 크기는 16×16으로 고정시켰고, 탐색영역은 ±7 화소로 하였으며, QP는 8로 고정시켰다. MSEA의 최대 레벨은 3으로 하였으며, 16화소마다 SAD_{Min}과 비교하여 계산의 중단 여부를 검사하였다. 초기 SAD_{Min}을 구하기 위하여 이전 프레임의 움직임 벡터를 사용하였으며[1] 탐색 순서는 왼쪽 상단부터 오른쪽 하단으로 순서대로 탐색하였다.

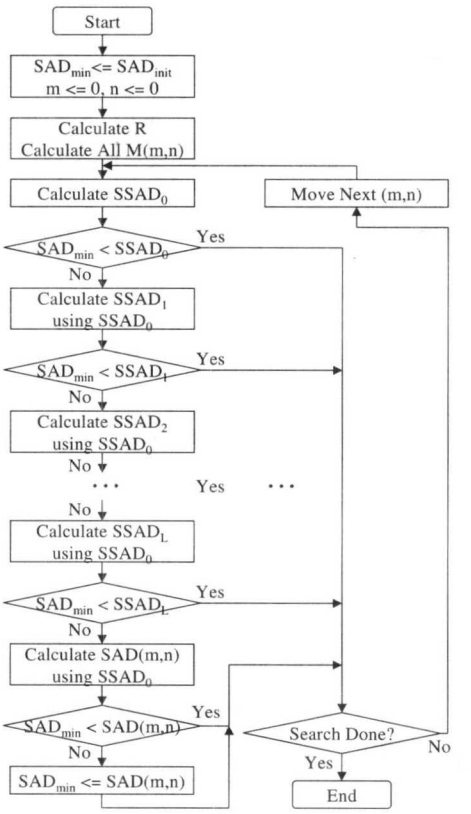


그림 3. 제안 알고리즘의 흐름도

계산량은 덧셈 (ADD), 뺄셈 (SUB), 절대치 (ABS), 비교 (CMP), 읽기 (LUT), 그리고 이동 (SHIFT)과 같은 6가지 연산에 대해 측정하였다. 읽기는 참조 테이블에서 값을 읽는 것을 의미한다. 제안하는 방식에서는 참조 테이블을 사용하지 않으므로 읽기가 없는 대신 SHIFT 연산이 추가되었다. 표 1~표 5는 각 실험 영상에 대한 실험 결과를 보여준다. 각 표에 표시된 값은 100프레임에 대한 실험에서 얻어진 총 계산량을 매크로블록의 수로 나눈 것이다. 즉, 움직임 벡터 한 개당 소요되는 계산량을 나타낸다.

표에 나타난 결과로부터 MSEA에 비하여 비교는 많이 증가하지만 덧셈과 뺄셈은 절반 정도에 불과함을 알 수 있다. 제안 알고리즘에서는 ESEA에서 사용되는 참조 테이블을 없애기 위하여 SHIFT와 뺄셈 연산이 부가적으로 소요된다. 위의 표들로부터 SHIFT가 제안 알고리즘 전체 계산량의 약 1%정도를 차지함을 알 수 있다. 따라서 ESEA에서 참조

표 1. Flower에 대한 계산량 비교

	ADD	SUB	ABS	CMP	LUT	SFT	Total
MSEA	7894	6004	6004	964	0	0	20866
ESEA	6453	3850	3850	2922	703	0	17778
Proposed Algorithm	4842	2867	967	5652	0	166	14494

표 2. Football에 대한 계산량 비교

	ADD	SUB	ABS	CMP	LUT	SFT	Total
MSEA	9047	7171	7171	1116	0	0	24505
ESEA	7699	4885	4885	3291	928	0	21688
Proposed Algorithm	5570	3606	941	7045	0	204	17366

표 3. Mobile에 대한 계산량 비교

	ADD	SUB	ABS	CMP	LUT	SFT	Total
MSEA	6950	5061	5061	917	0	0	17989
ESEA	6354	4032	4032	1890	427	0	16735
Proposed Algorithm	4725	2755	976	4806	0	163	13425

표 4. Table Tennis에 대한 계산량 비교

	ADD	SUB	ABS	CMP	LUT	SFT	Total
MSEA	23120	21412	21412	2390	0	0	68334
ESEA	15899	9505	9505	13524	4629	0	53062
Proposed Algorithm	11198	9387	1031	21942	0	456	44014

표 5. Tennis에 대한 계산량 비교

	ADD	SUB	ABS	CMP	LUT	SFT	Total
MSEA	10872	8996	8996	1196	0	0	30060
ESEA	7922	4555	4555	5299	1472	0	23803
Proposed Algorithm	5858	3912	945	8784	0	207	19706

표 6. 5가지 실험영상에 대한 평균 계산량의 비율(단위:%)

	Flower	Football	Mobile	Table Tennis	Tennis	Average
MSEA	100.0	100.0	100.0	100.0	100.0	100.0
ESEA	85.2	88.5	93.0	77.6	79.1	84.6
Proposed Algorithm	69.4	70.8	74.6	64.4	65.5	68.9

테이블을 없앰으로써 발생하는 부가 계산량은 뺄셈까지 고려하더라도 전체 계산량의 약 2%에 불과함을 알 수 있다. 또한 표들은 Table Tennis와 같이 빠른 움직임이 많은 실험 영상에서는 더 높은 계산량 감소가 이루어짐을 보이고 있다. 표 6은 5가지 실험 영상에 대한 실험 결과를 평균한 것이다.

표 6에서 알 수 있듯이 제안 알고리즘이 MSEA의 약 69%의 계산량만으로 최적의 움직임 추정을 수행할 수 있다. 또한 ESEA와의 비교에서도 약 81%의 계산량만이 소요됨을 알 수 있다.

V. 결 론

본 논문에서는 MSEA에서 각 레벨에서의 최소 범위 계산을 효율적으로 수행하는 개선된 고속 전역 탐색 BMA를 제안하였다. $SAD(m,n)$과 $SSAD_l$ 계산 모두에 첫 번째 레벨에서의 최소 범위를 이용함으로써 적은 계산량만으로 최적의 움직임 벡터를 구할 수 있었다. 또한 계산 순서를 바꿈으로써 많은 메모리를 필요로 하는 참조 테이블을 사용하지 않고도 효율적인 계산을 수행할 수 있었다.

컴퓨터 모의 실험을 통해 제안 알고리즘이 MSEA의 약 69%의 계산량만으로도 최적의 움직임 벡터를 구할 수 있음을 확인하였다. 즉, 약 31%의 계산량 이득이 있음을 알 수 있었다. 그리고 MSEA를 기반으로 하는 다른 알고리즘들과 마찬가지로 제안 알고리즘도 TSS와 같은 고속 BMA들과 결합이 가능하다[7].

그러나 MSEA를 기반으로 하는 방법들에는 공통적으로 다중레벨에 대한 절대치 합을 저장하기 위한 메모리가 많이 필요한 단점을 가진다. 제안 알고리즘 또한 MSEA를 기반으로 하기 때문에 많은 메모리가 필요한 단점을 가진다. 이와 같은 문제를 해결하기 위해 메모리를 절약할 수 있는 방식에 대한 연구가 요구된다.

참 고 문 헌

[1] W. Li, E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. on Image Processing*, vol. 4, pp. 105-107, Jan. 1995.

[2] M. Brünig, W. Niehsen, "Fast full-search block matching," *IEEE Trans. on Circuits and System for Video Technology*, vol. 11, pp. 241-247,

Feb. 2001.

[3] F. Dufaux, F. Moscheni, "Motion estimation techniques for digital TV:A review and a new contribution,"*Proceedings of IEEE*, vol. 83, pp. 858-876, Jun. 1995.

[4] T. Koya, K. Iinuma, A. Hirano, Y. Iyima, T. Ishi-guro, "Motion-compensated inter-frame coding for video conferencing," *Proceedings of NTC81*, New Orleans, LA, pp. C9.6.1-C9.6.5, Nov. 1981

[5] J. R. Jain, A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Communications*, vol. 29, pp. 1799-1801, Dec. 1981.

[6] H. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. on Communications*, vol. 38, pp. 950-953, Jul. 1990.

[7] X. Q. Gao, C. J. Duanmu, C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. on Image Processing*, vol. 9, pp. 501-504, Mar. 2000.

[8] Erwin Kreyszig, *Introductory functional analysis with applications*, John Wiley & Sons, New York, 1996.

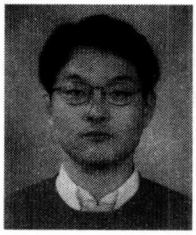
[9] H. S. Wang, R. M. Mersereau, "Fast algorithms for the estimation of motion vectors," *IEEE Trans. on Image Processing*, vol. 8, pp. 435-438, Mar. 1999.

안 태 경(Tae-gyoung Ahn) 준회원

1997년 2월 : 인제대학교 물리학과 졸업
1999년 2월 : 부산대학교 메카트로닉스과정 석사
1999년 9월~현재 : 부산대학교 전자공학과 박사과정

<주관심 분야> 영상처리, 동영상 압축, 컴퓨터 비전, FPGA/ASIC설계

문 용 호(Yong-ho Moon) 정회원



1992년 2월 : 부산대학교 전자공학과 졸업
1994년 2월 : 부산대학교 전자공학과 석사
1998년 8월 : 부산대학교 전자공학과 박사

1998년 9월~2001년 8월 : 삼성전자 중앙 연구소 Digital Media Lab. 책임연구원
2001년 9월~현재 : 부산대학교 산업자동화 및 정보통신분야 인력양성 사업단(BK21) 기금교수

<주관심 분야> 영상처리(정지영상 및 동영상 압축), VLSI, 신호처리, 신경회로망