디지털산업정보학회 논문지 제5권 제4호-2009년 12월

동영상의 블록내 지역성을 이용하는 효율적인 다단계 연속 제거알고리즘

정 수 목*

An Efficient Multi-level Successive Elimination Algorithm using the Locality in Block

Jung, Soo Mok

〈Abstract〉

In this paper, an efficient multi-level successive elimination algorithm using the locality in block was proposed for motion estimation. If SAD(sum of absolute difference) is calculated from large absolute difference values to small absolute difference values, SAD is increased rapidly. So, partial distortion elimination in SAD calculation can be done very early. Hence, the computations of SAD calculation can be reduced. In this paper, an efficient algorithm to calculate SAD from large absolute difference values to small absolute difference values by using the locality in block. Experimental results show that the proposed algorithm is an efficient algorithm with 100% motion estimation accuracy for the motion estimation of motion vectors.

Key Words : Motion estimation, Block matching algorithm, Motion vector, Multi-level successive elimination algorithm

Ⅰ. 서론

연속적인 프레임(frame)들로 이루어진 동영상(moving picture)의 데이터량은 매우 방대하다. 따라서 동영상 데이터를 저장하거나 전송하는데 적합하도록 동영상의 데이터를 압축하는 과정이 필수적이다. 특히 실시간 응용을 위하여 동영상 데이터 압축과 복원이 신속하게 이루어지도록 하는 고속 동영상 압축 및 복원기술이 중요하다.

동영상을 구성하는 각 프레임 내에는 공간적 중복성(spatial redundancy)이 존재하고 연속적인 프레임 간에는 시간적 중복성(temporal redundancy)이 존재하는데, 이러한 중복성을 제거함으로 동영상 데이터를 효과적으로 압축할 수 있다. 연속적인 프레임 간에 존재하는 시간적 중복성은 움직임 추정(motion estimation) 기법을 사용하여 효과적으로 제거할 수 있다.

움직임 추정 기법은 블록정합알고리즘(block matching algorithm, BMA)[1,2]과 화소순환알고리즘(pel-recursive algorithm, PRA)[3]의 두 그룹으로 나누어지는데, 블록정합알고리즘이 화소순환알고리즘에 비하여 구현하기가 간단하기 때문에 블록정합알고리즘이 H.261, H.263, H.264, MPEG 등 여러 비디오 코딩 표준(video coding standards)으로 채택되어 사용되고 있다[4-6].

* 삼육대학교 컴퓨터학부 교수

블록정합알고리즘에서는 현재 프레임을 일정한 크기의 정 사각 블록들로 나누고, 나누어진 각 목표블록(target block)에 대하여 이전 프레임(previous frame)상에 설정된 탐색윈도우(search window) 내의 탐색점(search point)들을 좌측 상단 점으로 하는 정합후보블록(matching candidate block)들에 대하여 정합 기준(matching criteria)을 사용하여 최적정합블록(the best matching block)을 찾은 후, 최적정합블록과의 상대적인 변위를 계산하여 움직임 벡터(motion vector)를 구한다. 현재 프레임의 각 목표블록들은 블록 정합기법에서 구해진 움직임 벡터와 최적정합블록과의 화소 차 값들로 변환된 후 동영상 데이터 압축절차를 따라 압축이 이루어진다.

블록정합알고리즘 중 최적 움직임 벡터를 찾는 전역탐색알고리즘(full search algorithm, FSA)은 이전 프레임 상에 설정된 탐색윈도우 내의 모든 정합후보블록들을 대상으로 최적정합블록을 찾음으로 최적의 움직임 벡터(global optimum motion vector)를 구할 수 있지만 전역탐색알고리즘을 수행하는데 필요한 연산량이 매우 많기 때문에 움직임 추정에 소요되는 시간이 길어진다. 따라서 실제적인 응용에서는 전역탐색알고리즘이 사용되지 않는다.

전역탐색알고리즘의 단점인 연산량을 감소시키기 위하여 2차원 로그탐색(two dimensional logarithmic search)[1], 3단계 탐색(three step search)[7], 직교탐색(orthogonal search)[8], 교차탐색(cross search)[9], 변형 3단계 탐색(variation of three-step search)[10,11], 일차원 전역탐색(one-dimensional full search)[12], Unrestricted center-biased diamond search[13] 등 많은 고속 블록정합 알고리즘들이 제안되었다. 그러나 이러한 고속 블록정합 알고리즘들은 움직임 보상 된 잔여 에러 면(motion compensated residual error surface)이 움직임 벡터의 변위에 대한 볼록 함수(convex function)임을 가정[14]하고 있는데 이러한 가정은 거의 대부분의 경우 실제적이지 않다[15]. 따라서 이러한 고속 블록정합 알고리즘들에서 구한 움직임 벡터는 국소적 최적치(local optimum)가 되는 한계를 가지며, 이러한 블록정합 알고리즘들은 움직임 추정의 정확도를 희생하여 움직임 추정 연산량을 감소시키는 알고리즘들이다.

볼록성 가정(convex assumption) 없이 전역탐색 알고리즘의 연산량을 감소시킨 연속 제거 알고리즘(successive elimination algorithm, SEA)[16]이 Li와 Salari에 의하여 제안되었고, 연속 제거 알고리즘의 연산량을 감소시키는 세 가지 알고리즘[17]을 Wang등이 제안하였다. 그 후, 연속 제거 알고리즘의 연산량을 효과적으로 감소시키는 다단계 연속 제거 알고리즘(multi-level successive elimination algorithm, MSEA)[18]이 Gao 등에 의하여 제안되었다. 그리고 다단계 연속 제거 알고리즘의 연산량을 효과적으로 줄일 수 있는 다양한 기법[19]들이 저자에 의하여 제안 되었다.

본 논문에서는 블록내의 지역성을 이용하여 다단계 연속 제거 알고리즘의 정합평가(matching evaluation) 과정에서의 연산량을 효과적으로 감소시키는 선택적 SAD(sum of absolute difference) 계산 기법을 제안하였다. 제안된 기법을 적용한 다단계 연속 제거 알고리즘은 움직임 추정의 정확도가 100%이면서 움직임 추정에 필요한 연산량을 효과적으로 감소시킨다.

Ⅱ. 다단계 연속 제거 알고리즘

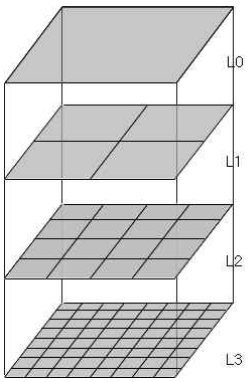
다단계 연속 제거 알고리즘에서는 현재 프레임상의 각 블록은 <그림 1>과 같이 계층별로 서로 다른 크기의 서브블록들로 분할된다. (N)x(N)크기의 목표블록(target block)은 레벨 0(level 0)에 해당되고, 레벨 1(level 1)에서는 (N)x(N)크기의 level 0의 블록이 ($N/2$)x($N/2$)크기의 4개의 서브블록으로 나누어진다. 레벨 2(level 2)에서는 ($N/2$)x($N/2$)크기의 level 1의 서브블록들이 ($N/4$)x($N/4$)크기의 서브블록 4개로 나누어진다. 이러한 절차가 서브블록의 크기가 2x2가 될 때까지 반복된다. 따라서 블록(target block)의 최대 분할 레벨은 $L_{max}=\log_2 N-1$이 된다. L-level 분할을 가지는 다단계 연속 제거 알고리즘을

L-level MSEA라고 하며 이때 L은 $0 \le L \le L_{max}$ 조건을 만족하는 값이다. 따라서 연속 제거 알고리즘(SEA)은 0-level MSEA에 해당된다. l번째 레벨에서의 서브블록의 전체 개수는 $(2^l)\times(2^l)=2^{2l}$이 되고, 서브블록의 크기는 $(N_l)\times(N_l)$이며 $N_l=N/2^l$ 이다.

현재 프레임과 이전 프레임의 좌표 (i, j)에서의 픽셀의 휘도(intensity)의 값을 각각 $f_c(i, j)$, $f_R(i, j)$로 나타내고 $(N)\times(N)$크기의 픽셀들로 이루어진 목표블록(target block)을 고려한다. 탐색윈도우 크기는 $(2M+1)\times(2M+1)$로 하고, 정합기준함수(matching criteria function)로 목표블록과 정합후보블록 간의 왜곡을 나타내는 절대오차의 합(sum of absolute difference, SAD)을 사용하며, 현재 프레임 상에서 좌측상단점이 (i, j)인 목표블록과 이전 프레임의 탐색윈도우 상에서 좌측상단점이 $(i+x, j+y)$인 정합후보블록을 각각 $B_c^{(i, j)}$, $B_p^{(i, j, x, y)}$로 나타내면 블록내 임의의 점 (m, n)에서 식 (1), (2)가 성립한다. 여기서 $B_c^{(i, j)}$는 움직임 벡터를 구하고자하는 현재 프레임 상의 목표블록이다.

$$B_c^{(i, j)}(m, n) = f_c(i+m, j+n) \quad (1)$$

$$B_p^{(i, j, x, y)}(m, n) = f_R(i-x+m, j-y+n) \quad (2)$$



<그림 1> 다단계 연속 제거 알고리즘에서의 블록 분할 방법

여기에서 x와 y는 움직임벡터 후보의 좌표를 나타내고 $-M \le (x, y) \le M$ 을 만족한다. 또한 (m, n)은 $0 \le (m, n) \le N-1$ 을 만족하는 값이다.

목표블록 $B_c(i, j)$와 정합후보블록 $B_R(i, j, x, y)$사이의 SAD 값은 다음과 같이 정의된다.

$$\mathrm{SAD}(x, y) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left|\boldsymbol{B}_c^{(i,j)}(m,n) - \boldsymbol{B}_P^{(i,j,x,y)}(m,n)\right| \quad (3)$$

움직임추정 기법에서는 식 (3)에서 구하여 진 SAD값이 최소가 되는 최적의 (x, y)를 구하여 이를 움직임 벡터 값으로 한다.

$X=B_c^{(i, j)}$와 $Y=B_p^{(i, j, x, y)}$에 대하여 수학적인 부등식 $||X||_1 - ||Y||_1 | \le ||X - Y||_1$ [20]을 적용하면 식 (4)가 유도된다.

$$|R-M(x, y)| \le \mathrm{SAD}(x, y) \quad (4)$$

여기서 R, $M(x, y)$, $\mathrm{SAD}(x, y)$는 다음과 같다.

$$R = \left\|\boldsymbol{B}_c^{(i,j)}\right\|_1 = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\boldsymbol{B}_c^{(i,j)}(m,n)$$

$$M(x, y) = \left\|\boldsymbol{B}_p^{(i,j,x,y)}\right\|_1 = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\boldsymbol{B}_p^{(i,j,x,y)}(m,n)$$

$$\mathrm{SAD}(x, y) = \left\|\boldsymbol{B}_c^{(i,j)} - \boldsymbol{B}_p^{(i,j,x,y)}\right\|_1 = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left|\boldsymbol{B}_c^{(i,j)}(m,n) - \boldsymbol{B}_P^{(i,j,x,y)}(m,n)\right|$$

이 때, R과 $M(x, y)$는 목표블록과 정합후보블록의 sum norm이며 이들은 [18]에 기술된 효과적인 절차에 따라 미리 계산된다.

L-level MSEA의 l번째 레벨에서 $R_l^{(u, v)}$, $M_l^{(u, v)}(x, y)$, $\mathrm{SAD}_l^{(u, v)}(x, y)$, $\mathrm{SAD_SB}_l(x, y)$은 아래와 같이 정의 된다.

$$R_l^{(u,v)} = \sum_{m=0}^{N_l-1}\sum_{n=0}^{N_l-1}\boldsymbol{B}_c^{(i,j)}(m+uN_l, n+vN_l) \quad (5)$$

$$M_l^{(u,v)}(x,y) = \sum_{m=0}^{N_l-1}\sum_{n=0}^{N_l-1} \boldsymbol{B}_p^{(i,j,x,y)}(m+uN_l, n+vN_l) \quad (6)$$

$$\mathrm{SAD}_l^{(u,v)}(x,y) = \sum_{m=0}^{N_l-1}\sum_{n=0}^{N_l-1} \left| \boldsymbol{B}_c^{(i,j)}(m+uN_l, n+vN_l) - \boldsymbol{B}_P^{(i,j,x,y)}(m+uN_l, n+vN_l) \right| \quad (7)$$

$$\mathrm{SAD\_SB}_l(x,y) = \sum_{u=0}^{2^l-1}\sum_{v=0}^{2^l-1} \left| R_l^{(u,v)} - M_l^{(u,v)}(x,y) \right| \quad (8)$$

(u, v)는 서브블록의 첨자(index) 이고 $0 \le (u, v) \le 2^l-1$ 조건을 만족하며 l은 $0 \le l \le L_{max}$ 조건을 만족하는 값이다. 0번째 레벨에서는 u, v가 모두 0이며, $R_l^{(u,v)}$, $M_l^{(u,v)}(x, y)$, $\mathrm{SAD}_l^{(u,v)}(x, y)$는 식 (4)에 정의된 R, $M(x, y)$, $\mathrm{SAD}(x, y)$와 같다. 따라서 L-level MSEA에서, 식 (8)의 $\mathrm{SAD_SB}_l(x, y)$은 조건에 따라 다음과 같이 식 (9)로 표현된다.

식 (5) ~ 식 (9)를 이용하여 식 (4)를 식 (10)으로 표현할 수 있음을 Gao등이 [18]에서 증명하였다.

$$\mathrm{SAD\_SB}_l(x,y) = \begin{cases} |R - M(x,y)| & l = 0 \\ \sum_{u=0}^{2^l-1}\sum_{v=0}^{2^l-1} \left| R_l^{(u,v)} - M_l^{(u,v)}(x,y) \right| & 1 \le l \le L \\ \mathrm{SAD}(x,y) & l = L+1 \end{cases} \quad (9)$$

$$\mathrm{SAD\_SB}_l(x, y) \le \mathrm{SAD\_SB}_{l+1}(x, y) \quad (10)$$

이 때 l의 값은 $0 \le l \le L_{max}$ 조건을 만족한다. 이 식은 블록을 서브블록으로 분할하는 경우, 목표블록 및 정합후보블록의 서브블록 간 sum norm의 차의 절대치 합인 $\mathrm{SAD_SB}_l$의 수학적인 특성을 나타내고 있다. 즉, $0 \le l \le L_{max}$ 조건을 만족하는 l-level 에서 $\mathrm{SAD_SB}_l$은 반드시 SAD보다 작거나 같고 l의 값이 작을수록 $\mathrm{SAD_SB}_l$의 값이 작아지는 특성을 나타낸다. 다단계 연속제거 알고리즘(MSEA)에서는 식 (10)을 정합후보블록을 제거하는데 사용하여 연산량을 효과적으로 줄인다.

움직임 추정과정에서 현재까지의 최적정합블록보다 더 정합이 잘되는 최적 정합블록이 있는지 찾기 위해서는 정합후보블록에서 SAD를 구하여 현재까지의 최적정합블록이 가지는 SAD(curr_min_SAD)와 비교하는 정합평가(matching evaluation)를 수행하는데, 정합후보블록에서 구한 SAD값이 현재까지의 최적정합블록에서의 SAD(curr_min_SAD)보다 작은 경우에는 정합후보블록은 새로운 최적정합블록이 되고 그렇지 않은 경우에는 최적정합블록이 아니기 때문에 제거된다. 그런데 정합후보블록에서의 SAD 연산은 계산 집중적인 연산임으로 매우 많은 연산량을 필요로 한다. 따라서 정합후보블록이 새로운 최적정합블록이 될 수 없어 제거될 수 있는지 판단하는데 SAD를 사용하지 않고 판단 할 수 있다면 움직임 추정 연산량을 감소시킬 수 있다.

다단계 연속 제거 알고리즘에서는 정합후보블록이 제거될 수 있는지 판단 시, SAD계산 이전에 SAD의 연산량보다 연산량이 적은 $\mathrm{SAD_SB}_l$을 사용하여 수행함으로 연산량을 줄일 수 있다. $\mathrm{SAD_SB}_l$의 계산은 [18]에서 제안된 효과적인 방법으로 구해지고 l의 값이 적을수록 $\mathrm{SAD_SB}_l$연산에 보다 적은 연산량이 소요된다. 다단계 연속 제거 알고리즘에서는 정합후보블록이 새로운 최적정합블록이 될 수 없어 제거될 수 있는지 판단 할 때 먼저 연산량이 가장 적은 $\mathrm{SAD_SB}_0$을 계산한 후 curr_min_SAD와 비교한다. curr_min_SAD가 $\mathrm{SAD_SB}_0$보다 작거나 같다면 식 (11)이 성립되어 정합후보블록은 새로운 최적정합블록이 될 수가 없어 제거된다. 이때 정합후보블록은 0-level 에서 제거된다고 한다.

$$\text{curr\_min\_SAD} \le \mathrm{SAD\_SB}_0(x, y) \le \mathrm{SAD} \quad (11)$$

그러나 curr_min_SAD가 $\mathrm{SAD\_SB}_0$ 보다 크게 되면 l을 증가시키면서 정합후보블록이 제거될 수 있는지 판단하는 과정을 반복한다. MSEA에서는 정합후보블록을 제거하는 과정이 0-level에서 부터 최고 L_{max}-level까지 수행될 수 있으며 L_{max}-level까지 수행 후에도 정합후보블록이 제거되지 않으면 SAD를 계산하여 curr_min_SAD와 비교한다. 이 때, curr_min_SAD가 SAD보다 작거나 같

으면 정합후보블록은 제거되나 그렇지 않은 경우에는 정합후보블록은 새로운 최적정합블록이 되고 curr_min_SAD는 SAD값으로 대치된다.

움직임추정에 MSEA를 적용하는 경우, 많은 정합후보블록들이 0-level에서 제거되며 L_{max}-level까지 수행하면 매우 많은 정합후보블록들이 제거된다. 따라서 MSEA에서는 계산집중적인 SAD를 계산하지 않고도 적은 연산량이 소요되는 SAD_SB만을 계산하여 많은 정합후보블록들을 제거하여 움직임 추정의 연산량을 크게 줄일 수 있다. 이러한 과정이 탐색윈도우내의 모든 탐색점에 대하여 수행된 후에 최종 curr_min_SAD를 가지는 블록이 최적정합블록이 된다. 이 때 최적정합블록의 좌표에서 목표블록의 좌표를 뺀 상대적인 변위가 목표블록의 움직임 벡터가 된다.

L-level MSEA의 procedure는 다음과 같다.

단계 1 이전 프레임의 탐색윈도우내에서 초기 탐색점을 선택

단계 2 선택된 탐색점에서의 움직임벡터(MV)와 SAD 계산.
계산된 움직임 벡터와 SAD는 각각 현재 임시 움직임 벡터(curr_ temp_**MV**)와 현재까지의 최소 SAD(curr_min_SAD)값이 됨.

단계 3 탐색윈도우내의 나머지 탐색점들중에서 다른 한 탐색점을 선택

단계 4.0 선택된 탐색점에서 SAD_SB_0 계산
if(curr_min_SAD≤SAD_SB_0) goto 7

단계 4.1 선택된 탐색점에서 SAD_SB_1 계산
if(curr_min_SAD≤SAD_SB_1) goto 7

•
•

단계 4.L 선택된 탐색점에서 SAD_SB_L 계산
if(curr_min_SAD≤SAD_SB_L) goto 7

단계 5 선택된 탐색점에서 SAD 계산
if (curr_min_SAD≤SAD) goto 7

단계 6 선택된 탐색점을 좌측 상단 점으로하는 정합후보블록이 현재까지의 최적정합블록임으로 SAD를 현재까지의 최소 SAD값으로 함 (curr_ min_SAD = SAD)
움직임 벡터를 구하여 현재 임시 움직임 벡터로 함(curr_temp_ **MV=MV**).

단계 7 if (탐색윈도우내에 탐색해야 할 탐색점이 남아 있는가?) goto 3

단계 8 현재까지의 최소 SAD(curr_min_ SAD)를 갖는 탐색점을 좌측 상단점으로 하는 정합후보블록이 최적정합블록임
global minimum SAD=
curr_min_SAD
optimum motion vector=
curr_temp_**MV**

Ⅲ. 제안 알고리즘

다단계 연속 제거 알고리즘에서는 0-level부터 연속적으로 L_{max}-level까지 SAD_SB을 사용하여 정합후보블록을 제거하지만(단계 4.0 ~ 단계 4.L), L_{max}-level 에서도 정합후보블록이 제거되지 아니하면 정합후보블록에 대한 SAD를 계산하여 정합후보블록이 새로운 최적정합블록이 되는지 판단하게 된다. 이 때, $(N)x(N)$ 크기의 정합후보블록내의 모든 점에서 SAD를 계산하는 연산은 식(3)과 같이 계산집중적인 연산임으로 연산량이 매우 크다. 단계 5에서 SAD를 효율적으로 계산하고, 정합후보블록을 신속히 제거할 수 있으면 단계 5에서의 연산량을 줄일 수 있다.

블록에서 SAD계산 시, 누적 부분 SAD (cumulative_ partial_SAD)값이 curr_min_SAD 값보다 크거나 같으면 정합후보블록이 곧바로 제거될 수 있는 기법(partial distortion elimination, PDE)[21]이 사용될 수 있다.

(N)x(N) 크기의 블록에서 SAD계산에 PDE기법을 적용하면 각 행에 해당하는 N개의 픽셀에대한 SAD를 누적 계산한 값(cumulative _partial_SAD)이 curr_min_SAD 값보다 크거나 같으면 부분왜곡제거가 발생하여 정합후보블록이 곧바로 제거된다. 따라서 PDE기법을 적용하는 경우, 단계 5는 다음과 같이 단계 5.1~ 단계 5.3으로 표현된다.

단계 5.1 for(i=0 ; i 〈N ; i++) {
단계 5.2 N개의 픽셀에 대하여 누적 부분 SAD(cumulative_partial_SAD)계산
단계 5.3 if (cumulative_partial_SAD≥ curr_min_SAD) goto 7 }

단계 5에 PDE 기법을 적용하여 단계 5가 단계 5.1 ~ 단계 5.3으로 되는 경우, 단계 5.3에서의 부분왜곡 제거가 신속히 발생할 수 있도록 SAD를 효율적으로 계산하는 기법을 본 논문에서 제안하였다. 누적 부분 SAD (cumulative_partial_SAD)를 계산 할 때 정합후보블록내의 지역성을 활용하여 식 (3)의 픽셀값 간의 차의 절대치인 절대오차(absolute difference)가 큰 부분에 대하여 SAD값을 먼저 계산하고, 절대오차가 작아지는 영역의 순으로 SAD를 구하면 누적 부분 SAD(cumulative_partial_SAD)값이 신속히 증가하여 단계 5.3에서 정합후보블록이 보다 빨리 제거될 수 있어 움직임 추정의 연산량을 줄 일 수 있다.

16x16(N=16) 블록의 경우, 각 블록마다 256개의 절대오차 값들이 존재하고 이 값들은 0에서부터 픽셀의 최대 휘도 값(maximum pixel intensity value)에 이르는 값을 갖는다. 큰 절대오차 값을 가지는 픽셀 주변의 절대오차 값들은 큰 값을 가지는 경향이 있다. 즉, 서로 인접한 픽셀의 절대오차의 값은 서로 유사한 값을 가지는 경향이 있다. 이는 동영상의 각 프레임 상에 있는 블록 내에서 인접한 화소는 매우 연관성이 높고 연관성이 높은 현재 프레임내의 목표블록과 이전 프레임 내의 정합후보블록에 대한 픽셀값 간의 차의 절대치인 절대오차들도 서로 높은 상관성을 가지기 때문이다[19]. 따라서 SAD계산 시, 절대오차가 큰 부분부터 작은 부분 순으로 계산을 수행하면 누적 부분 SAD값(cumulative_partial_SAD)이 신속하게 증가하기 때문에 단계 5.3에서의 부분왜곡 제거가 매우 빨리 발생하여 SAD계산에 필요한 연산량이 감소하게 된다.

블록에서 SAD를 효과적으로 구하기 위하여 블록을 8x8 크기를 갖는 4개의 서브블록으로 분할하였다. 4개의 서브블록 중에서 절대오차의 값이 큰 영역과 적은 영역의 순서를 예측하기 위하여 상위레벨에서 계산되어진 SAD_SB$_i$값들을 사용한다. 이 때 i의 값은 SAD값이 계산되어지는 L_{max}+1레벨에 가까울수록 SAD계산에 사용되는 절대오차의 값의 성질을 충실히 내포하기 때문에 L_{max}레벨에서의 값을 사용하는 것이 바람직하다. 따라서 절대오차 값의 성질을 가장 충실히 포함하고 있는 L_{max} 레벨에서의 SAD_SB$_i$ 값들을 각 서브블록 별로 더하고 이를 크기순으로 정렬(sorting)하여 서브블록들의 SAD 크기에 따른 순서를 예측한다. SAD크기가 클 것으로 예측된 서브블록부터 SAD를 계산하고 그 후 예측된 서브블록의 순으로 SAD를 계산하면 누적 부분 SAD (cumulative_partial_SAD)값이 신속히 증가하여 부분왜곡제거가 매우 빨리 발생함으로 계산집중적인 연산인 SAD 연산량을 줄일 수 있다.

Ⅳ. 실험결과

본 논문에서는 영상의 크기가 176x144 pixel 크기를 갖는 stefan. qcif와 foreman. qcif 영상 100 frame에 대하여 실험을 수행하였다. 다단계 연속 제거 알고리즘에서의 실험환경과 동일하게 블록의 크기는 16x16(N=16)로 하였고 탐색윈도우의 크기는 31x31(M=15)로 하였으며 움직임벡터는 정수 값만을 고려하였다. <표 3>의 실험결과에 표시된 m. e. 는 SAD계산을 필요로 하는 정합평가(matching evaluation)를 나타내고, avg. # of m. e.

/frame은 프레임당 정합평가(matching evaluation)가 수행되는 평균 횟수를 나타내는 것으로서 정합평가 횟수는 SAD를 계산하여 현재까지의 최소 SAD(curr_ min_SAD) 값과 비교되는 횟수가 된다. avg. # of rows/m. e. 는 정합평가시 부분 왜곡 제거가 일어나기 전까지 계산된 16개 픽셀 기준의 행들의 평균수를 나타낸다. (in rows)는 16x16 pixel 크기의 블록에서 1x16 pixel로 구성되는 한 행에 대한 SAD를 계산하는데 필요한 연산량을 기본 단위로 하여 표시한 연산량을 나타낸다.

<표 1>은 QCIF 영상의 100개 프레임에서 7,734,000개의 블록에 대하여 최대 SAD와 최소 SAD의 비(maximum to minimum SAD ratio)를 조사한 결과이다. SAD계산 이전에 L_{max}-레벨에서 미리 계산되어진 SAD_SB을 사용하여 서브블록의 SAD 크기 순서를 예측한 정확도는 <표 2>와 같다. <표 2>에서 Hit n($1 \le n \le 4$)은 SAD값이 n번째 클 것으로 추정된 서브블록이 실제 n번째 크기의 SAD를 갖는 서브블록과 일치하는 비율이다. Hit 1/2는 최고 큰 SAD와 두 번째로 큰 SAD값을 가질 것으로 예측된 서브블록이 실제 서브블록과 일치하는 경우를 나타낸다. Hit 1/2/3/4는 예측된 서브블록의 SAD 크기순이 실제 크기순과 완전히 일치하는 경우이다. 인접한 픽셀들의 휘도 값은 서로 비슷한 값을 가지는 특성인 지역성(locality)을 갖기 때문에 L_{max}-레벨에서 미리 계산되어진 SAD_SB을 사용하여 서브블록의 SAD값의 크기 순서를 예측하는 정확도가 높다. 예측된 각 서브블록의 SAD 크기순으로 SAD 값을 계산함으로 SAD값이 신속하게 증가하게 되어 단계 5.3에서 부분왜곡제거가 빨리 발생하게 된다.

본 논문에서 각 서브블록의 특성에 따라 SAD값을 선택적으로 계산 할 수 있도록 제안된 기법을 MSEA에 적용하였을 경우, avg. # of rows/frame 값이 줄어 움직임 추정에 필요한 연산량이 감소함을 <표 3>에서 알 수 있다. 이는 제안된 기법을 다단계 연속 제거 알고리즘에 적용하였을 경우 누적 부분 SAD값이 신속히 증가하여 단계 5.3에서의 부분왜곡 제거가 보다 빨리 발생하기 때문에 SAD 계산 수행이 보다 적은 탐색점들에서 수행되었기 때문이다.

<표 3>에서 보는 바와 같이 제안된 기법을 다단계 연속 제거 알고리즘(3-level MSEA)에 적용하여 stefan. qcif와 foreman. qcif에 대하여 움직임 추정을 수행할 때 다단계 연속 제거 알고리즘의 연산량이 2.4%, 1.9% 감소하였다. 제안된 알고리즘은 움직임 추정의 정확도를 100%로 유지하면서 다단계 연속제거 알고리즘의 연산량을 감소시키는 효율적인 알고리즘이다.

<표 1> (서브블록의 최대 SAD/서브블록의 최소 SAD) 비에 따른 블록의 분포(%)

Video	1 ≤	1.2 ≤	1.4 ≤	1.6 ≤	1.8 ≤	2 ≤	4 ≤	6 ≤	8 ≤	10 ≤
stefan	4.8	14.4	15.3	12.1	8.6	26.5	7.7	4.3	2.7	3.6
foreman	0.9	3.7	5.7	6.7	6.9	38.9	12.9	7.1	4.7	12.5

<표 2> SAD_SBI을 이용하여 서브블록의 실제 SAD 크기 순서를 예측한 정확도(%)

Video	Hit 1	Hit 1/2	Hit 1/2/3	Hit 1/2/3/4	Hit 2	Hit 3	Hit 4
stefan	71.1	50.9	37.9	37.9	53.4	54.2	70.8
foreman	85.5	70.2	58.3	58.3	71.4	69.1	82.5

<표 3> 제안된 기법을 3-level MSEA에 적용한 경우의 연산량 감소

Algorithm	Video	Avg. # of m.e./frame	Avg. # of rows/m.e.	Overhead (in rows)	Total (in rows)	Computation reduction
3-level MSEA	stefan	1,241.9	14.17	52,284.0	69,881.7	
	foreman	748.6	14.62	24,967.2	35,911.7	
3--level MSEA +제안된 기법	stefan	1,241.9	12.74	52,397.7	68,219.5	2.4%
	foreman	748.6	13.61	25031.9	35,220.3	1.9%

Ⅳ. 결론

본 논문에서는 비디오코딩의 움직임추정을 위한 다단계 연속 제거 알고리즘의 연산량을 감소시키기 위하여 SAD 값이 클 것으로 예상되는 부분을 우선적으로 선택하여 계산할 수 있는 알고리즘을 제안하였다. 제안된 알고리즘에서는 L_{max}-레벨에서 이미 계산되어졌고 서브블록의 SAD값들의 분포 특성을 잘 반영하는 SAD_SB을 이용하여 서브블록들의 SAD크기에 따른 순서를 예측하기 때문에 SAD_SB을 계산하기 위한 추가적인 연산량 없이 서브블록의 특성에 따라 SAD값이 클 것으로 예측되는 서브블록부터 순차적으로 계산할 수 있다. 제안된 알고리즘을 3-level MSEA에 적용하여 stefan. qcif와 foreman. qcif에 대하여 성능평가를 수행한 결과 움직임 추정 연산량이 2.4%, 1.9% 각각 감소하였다. 이는 서브블록내 절대오차(absolute difference)가 큰 부분에 대하여 먼저 SAD값을 계산함으로 누적 부분 SAD(cumulative_partial_SAD) 값이 신속하게 증가하여 단계 5.3에서의 부분왜곡제거가 빨리 발생하기 때문이다. 제안된 알고리즘은 움직임 추정의 정확도를 100% 유지하면서 움직임 추정에 필요한 연산량을 감소시키는 효율적인 알고리즘이다.

참고문헌

[1] J. R. Jain, and A. K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. Commun., Dec. 1981, vol. COMM-29, pp. 1799-1808.

[2] H. G. Musmann, P. Pirsh, and H. J. Gilbert, "Advances in picture coding," Proc. IEEE, Apr. 1985, vol. 73, pp. 523-548.

[3] A. N. Netravali, and J. D. Robbins, "Motion compensated television coding: Part I," Bell Syst. Tech. J., Mar. 1979, vol. 58, pp. 631-670.

[4] CCITT Standard H.261, "Video codec for audiovisual services at px64 kbit/s," ITU, 1990.

[5] ITU-T DRAFT Standard H.263, "Video coding for narrow telecommunication channel at (below) 64kbit/s," ITU, Apr. 1995.

[6] ISO-IEC JTC1/SC2/WG11, "Preliminary text for MPEG video coding standard," ISO, Aug. 1990.

[7] T. Koga, K. Iinuma, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," Proc. Nat. Telecommunications Conf., Nov. 1981. pp. G.5.3.1- G.5.3.5.

[8] A. Puri, H. -M. Hang, and D. L. Schilling, "An efficient block matching algorithm for motion compensated coding," Proc. Int. Conf. Acoust., Speech, Signal Processing, 1987, pp. 25.4.1-25.4.4.

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

[10] 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. Circuits Syst. Video Technol., Feb. 1994, vol. 4, pp. 88-91.

[11] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., Aug. 1994, vol. 4, pp. 438-442.

[12] M. J. Chen, L. G. Chen and T. D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," IEEE Trans. Circuits Syst. Video Technol., Oct. 1994, vol. 4, pp. 504-509.

[13] J. Y. Tham, S. Ranganath, M. Ranganath, and A. Ali Kassim, "A novel unrestricted center-biased

diamond search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., Aug. 1998, vol. 8, no. 4, pp. 369-377.

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

[15] K. H. K. Chow and M. L. Liou, "Genetic motion search algorithm for video compression," IEEE Trans. Circuits Syst. Video Technol., Dec. 1993, vol. 3, pp. 440-446.

[16] W. Li, and E. Salari, "Successive Elimination Algorithm for Motion Estimation," IEEE Trans. Image Processing, Jan. 1995, vol. 4, No. 1, pp. 105-107.

[17] H. S. Wang and R. M. Mersereau, "Fast Algorithms for the Estimation of Motion Vectors," IEEE Trans. Image Processing, Mar. 1999, vol. 8, No. 3, pp. 435-438.

[18] X. Q. Gao, C. J. Duanmu, and C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," IEEE Trans. Image Processing, Mar. 2000, Vol. 9, No. 3, pp. 501-504.

[19] S. M. Jung, S. C. Shin, H. Baik and M. S. Park, "Efficient multilvel successive elimination algorithms for block matching motion estimation," IEE Proc. -Vis. Image Signal Process., Vol. 149, No. 2, April 2002.

[20] T. M. Apostol, "Mathematical Analysis," Addison-Wesley, MA, 1975.

[21] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression," Boston, MA, Kluwer, 1991.

■ 저자소개 ■

정 수 목
Jung, Soo Mok

1992년 3월~현재 삼육대학교 컴퓨터학부 교수
2002년 2월 고려대학교 컴퓨터학과(이학박사)
1986년 2월 경북대학교 전자공학과(공학석사)
1984년 2월 경북대학교 전자공학과(공학사)

관심분야 : Multimedia, Video coding
E-mail : jungsm@syu.ac.kr

논문접수일 : 2009년 10월 15일
수 정 일 : 2009년 11월 17일
게재확정일 : 2009년 11월 23일