論文
4-12-17

계층적인 탐색점 추출을 이용한 고속 블록 정합 알고리즘 (A Fast Block Matching Algorithm Using Hierarchical Search Point Sampling)

정수목(Soo-Mok Jung) [1]

요 약

본 논문에서는 비디오코딩의 움직임 추정을 위한 빠른 움직임 추정알고리즘을 제안하였다. 제안된 알고리즘은 다단계 연속제거 알고리즘과 효율적인 다단계 연속제거 알고리즘에 기초하고 있다. 제안된 알고리즘은 계층적으로 탐색점을 추출하여 매우 많은 연산량을 필요로 하는 정합 연산량을 감소시키면서 최상의 움직임벡터를 얻을 수 있다. 실험을 통하여 제안된 알고리즘의 효율성을 확인하였다

ABSTRACT

In this paper, we present a fast motion estimation algorithm to reduce the computations of block matching algorithm for motion estimation in video coding. The proposed algorithm is based on Multi-level Successive Elimination Algorithm and Efficient Multi-level Successive Elimination Algorithms. The best estimate of the motion vectors can be obtained by hierarchical search point sampling and thus the proposed algorithm can decrease the number of matching evaluations that require very intensive computations. The efficiency of the proposed algorithm was verified by experimental results.

Key words: Multilevel Successive Elimination Algorithm, Motion estimation, Motion vector

1) 정회원 : 삼육대학교 컴퓨터학과 부교수

논문접수 : 2003. 12. 11.
심사완료 : 2003. 12. 22.

1. 서론

동영상의 데이터량은 매우 방대하기 때문에 저장하거나 전송하기 위하여 데이터를 압축하는 것이 필수적이다. 동영상에는 한 프레임 내에 공간적인 중복성(spatial redundancy)이 존재하고 연속적인 프레임사이에는 시간적인 중복성(temporal redundancy)이 존재한다. 이러한 중복성을 제거하면 동영상 데이터를 효과적으로 압축할 수 있다. 시간적인 중복성을 제거하기 위하여 사용되는 움직임 추정기법은 블록 정합 알고리즘(BMA: block matching algorithm)[1][2]과 화소 순환 알고리즘(PRA: pel-recursive algorithm)[3] 두 종류로 나누어진다. 블록 정합알고리즘의 구현이 간단하기 때문에 블록 정합 알고리즘이 CCITT H.261[4], ITU-T H.263[5] 그리고 MPEG[6] 등 여러 비디오 코딩 표준(video coding standards)으로 널리 채택되어 왔다.

블록 정합 알고리즘의 목적은 현재 프레임을 임의의 작은 블록으로 나눈 뒤, 각 블록에 대하여 이전 프레임에 설정된 탐색영역에서 정합척도가 최적인 블록을 찾는 데 있다. 각 블록의 움직임 벡터(motion vector)는 이전 프레임(previous frame) 상에 설정된 탐색 윈도우 내에서 정합 기준(matching criteria)에 따라 최적 정합 블록(the best matching block)을 찾은 후, 최적 정합 블록과의 상대적인 변위를 계산함으로 구해진다.

블록 정합 알고리즘 중 전역 탐색 알고리즘(FSA: Full Search Algorithm)은 이전 프레임 상에 정의된 탐색 윈도우 내의 모든 정합 블록 후보(candidate matching block)들 중에서 최적 정합 블록을 찾음으로 최적 움직임 벡터(global optimum motion vector)를 찾지만, 전역 탐색을 수행하기 위하여 매우 많은 연산량을 필요로 한다. 따라서 실제적인 응용에서는 전역 탐색 알고리즘이 사용되지 않는다.

FSA의 연산량을 감소시키기 위하여 3단계탐색[7], 2-D 로그탐색[2], 직교탐색[8], 교차탐색[9], 일차원전역탐색(one-dimensional full search)[10], 변형 3단계 탐색(variation of three-step search)[11][12], unrestricted center-biased diamond search[13] 등 여러 알고리즘들이 개발되어 왔다. 이러한 고속 블록 정합 알고리즘들은 움직임 보상된 잔여 에러면(motion compensated residual error surface)이 움직임 벡터의 변위에 대한 볼록함수(convex function)라는 것을 가정하고 있다[14]. 그러나 이러한 가정은 거의 실제적이지 않다[15]. 그러므로 이러한 고속 블록 정합 알고리즘들을 사용하여 얻어진 움직임 벡터는 근본적으로 국소적 최적치(local optimum)가 된다. 다르게 표현하면 대부분의 고속 블록 정합 알고리즘들은 움직임 추정의 정확도를 희생하여 연산량을 감소시킨다.

Li와 Salari에 의해서 제안된 연속 제거 알고리즘(SEA: successive elimination algorithm)[16]은 이러한 볼록성 가정(convexity assumption) 없이 전역 탐색 알고리즘의 연산량을 감소시켰다. Wang등이 SEA의 연산량을 감소시키기 위한 알고리즘들 제안하였다.[17]. 또한 X. Q. Gao 등이 SEA의 연산량을 감소시키기 위하여 다단계 연속제거 알고리즘(MSEA)[19]을 제안하였고 본 연구팀에 의하여 MSEA의 성능을 개선한 방안들[20]이 제안되었다.

본 논문에서는 [20]에서 제안된 방안을 계층적인 기법으로 확장하여 MSEA의 성능을 개선하였다. 본 논문에서 제안된 고속 움직임 추정 알고리즘은 MSEA의 움직임 추정 정확도와 같이 정확도가 거의 100% 유지되면서 움직임 추정의 연산량을 효율적으로 감소시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 다단계 연속제거 알고리즘에 대하여 간략히 기술하였고, 3장에서 본 논문에서 제안하는 기법을 기술하였다. 제4장에서는 제안된 알고리즘에 대하여 실험한 결과를 기술하였다. 마지막으로 4장에서 결론을 맺었다.

II. 다단계 연속 제거 알고리즘

다단계 연속 제거 알고리즘에서, 각 블록은 그림 1과 같이 여러 개의 서브 블록

(sub-block)들로 나누어진다. 먼저 블록은 크기가 (N/2)x(N/2)인 4개의 서브 블록으로 나누어진다. 그 후 각 서브 블록들은 크기가 (N/4)x(N/4)인 4개의 서브 블록으로 다시 나누어진다. 이러한 절차가 서브 블록의 크기가 2x2가 될 때까지 반복된다. 블록의 크기가 NxN인 경우, 최대 분할 레벨은 L_{max}=log$_2N$-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 SEA는 0-level MSEA에 대응된다. L-level MSEA의 l번째 레벨에서 서브 블록들의 수는 $S_l = 2^{2l}$ 이고 각 서브 블록의 크기는 (N_l)x(N_l)이며 N_l=N/2^l)x(N/2^l) 이다.

$f_c(i,j)$와 $f_p(i,j)$는 현재 프레임과 이전 프레임 상에서 좌표가 (i,j)인 화소(pixel)의 휘도(intensity) 값을 각각 나타내고, 블록(H.263에서 매크로블록의 Y성분)의 크기를 (N)x(N), 탐색 윈도우 크기를 (2M+1)x(2M+1), 정합 기준 함수로 두 블록 사이의 왜곡을 나타내는 절대 오차의 합(SAD: sum of absolute difference)을 사용하는 것으로 한다. $\boldsymbol{B}_c^{(i,j)}$와 $\boldsymbol{B}_p^{(i,j,x,y)}$를 각각 현재 프레임과 이전 프레임 상에서 좌측 상단 점이 (i,j)와 $(i-x, j-y)$인 참조 블록과 탐색 윈도우 내에 있는 정합 블록 후보라고 둔다. $\boldsymbol{B}_c^{(i,j)}$는 움직임 벡터를 구하고자 하는 참조 블록이다. 이때 블록내 임의의 점 (m,n)에 대하여 식 (1), (2)가 성립한다.

$$\boldsymbol{B}_c^{(i,j)}(m,n) = f_c(i+m,\ j+n) \qquad (1)$$

$$\boldsymbol{B}_p^{(i,j,x,y)}(m,n) = f_p(i-x+m,\ j-y+n) \quad (2)$$

여기서 x와 y는 움직임 벡터 후보(candidate motion vector)의 좌표를 나타내고 $-M \le (x,y) \le M$ 조건을 만족한다. (m,n)은 $0 \le (m,n) \le N-1$ 조건을 만족하는 값이다.

두 블록사이의 SAD는 식 (3)과 같이 정의된다.

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

움직임 추정의 목표는 식 (4)와 같이 최소 SAD를 갖는 최적의 (x,y) 값을 찾는 것이다.

$$d = \min_{x,y} \mathrm{SAD}(x,y) \quad (4)$$

수학적인 부등식 $|\,\|\boldsymbol{X}\|_1 - \|\boldsymbol{Y}\|_1\,| \le \|\boldsymbol{X} - \boldsymbol{Y}\|_1$ [18]을 $\boldsymbol{X}$=$\boldsymbol{B}_c^{(i,j)}$와 $\boldsymbol{Y}$=$\boldsymbol{B}_p^{(i,j,x,y)}$에 대하여 적용하면 식 (5)를 얻을 수 있다.

$$|R - M(x,y)| \le \mathrm{SAD}(x,y) \qquad (5)$$

여기서

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

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

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

R과 $M(x,y)$는 각각 참조 블록과 정합 블록 후보의 sum norm 들이고 이 값들은 [9]에 기술된 효과적인 방법으로 미리 계산된다.

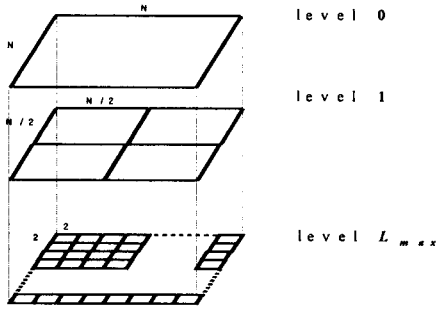

그림 1. 다단계 연속제거 알고리즘에서 블록의 분할

l 레벨에서 $R_l^{(u,v)}$, $M_l^{(u,v)}(x,y)$, $\mathrm{SAD}_l^{(u,v)}(x,y)$, $\mathrm{SAD_SB}_l(x,y)$를 아래와 같이 정의한다.

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

$$M_l^{(u,v)}(x,y) = \sum_{m=0}^{N_l-1}\sum_{n=0}^{N_l-1} \boldsymbol{B}_p^{(i,j,x,y)}(m+uN_l, n+vN_l) \quad (7)$$

$$\mathrm{SAD}_l^{(u,v)}(x,y) = \sum_{m=0}^{N_l-1}\sum_{n=0}^{N_l-1} \left| \boldsymbol{B}_c^{(i,j)}(m+uN_l, n+vN_l) - \boldsymbol{B}_p^{(i,j,x,y)}(m+uN_l, n+vN_l) \right| \quad (8)$$

$$\mathrm{SAD\_SB}_l(x,y) = \sum_{u=0}^{2^l-1}\sum_{v=0}^{2^l-1} \left| R_l^{(u,v)} - M_l^{(u,v)}(x,y) \right| \quad (9)$$

여기서 (u,v)는 서브 블록의 첨자(index) 이고 l은 $0 \le l \le L_{max}$ 조건을 만족하는 값이다. $0 \le (u,v) \le 2^l-1$ 조건을 만족하는 l번째 레벨 분할에서 다음의 관계식이 성립함을 쉽게 알 수 있다.

$$\mathrm{SAD}(x,y) = \sum_{u=0}^{2^l-1}\sum_{v=0}^{2^l-1} \mathrm{SAD}_l^{(u,v)}(x,y) \quad (10)$$

그리고 $\mathrm{SAD\_SB}_l(x,y)$ 은 다음과 같이 분류될 수 있다.

$$\mathrm{SAD\_SB}_l(x,y) = \begin{cases} |R - M(x,y)| & l = 0 \\ \sum_{u=0}^{2^l-1}\sum_{v=0}^{2^l-1} \left| R_l^{(u,v)} - M_l^{(u,v)}(x,y) \right| & 1 \le l \le L \\ \mathrm{SAD}(x,y) & l = L+1 \end{cases} \quad (11)$$

식 (6) ~ 식 (11)을 이용하여 식 (5)가 식 (12)로 표현될 수 있음을 Gao 등이 [9]에서 증명하였다.

$$\mathrm{SAD\_SB}_l(x,y) \le \mathrm{SAD\_SB}_{l+1}(x,y) \quad (12)$$

여기서 l은 $0 \le l \le L_{max}$ 조건을 만족하는 값이다.

식 (12)는 블록을 서브 블록으로 분할하여 만들어지는 각 서브 블록간 sum norm의 차의 절대치의 합인 $\mathrm{SAD\_SB}_l$[식 (11)]이 가지는 수학적인 성질을 나타낸 것으로, l이 $0 \le l \le L_{max}$ 인 경우 $\mathrm{SAD_SB}_l$은 항상 SAD보다 작거나 같으며 l이 작아질수록 $\mathrm{SAD\_SB}_l$ 값이 작아지는 수학적인 성질을 나타낸다. 이 수식을 아래와 같이 정합 블록후보를 제거하는데 사용하면 연산량을 매우 효과적으로 줄일 수 있다.

움직임 추정과정에서 현재까지의 최적 정합 블록보다 더 정합이 잘되는 새로운 최적 정합 블록이 있는지 찾기 위해서는 정합 블록 후보에서 SAD를 계산하여 현재까지의 최적 정합 블록이 가지는 SAD(*curr_min_SAD*)와 비교하는 정합 평가(matching evaluation: MSEA의 알고리즘 중 step 14, 15를 수행하여야한다. 이때 정합블록후보에서 구해진 SAD가 *curr_min_SAD*보다 작은 값을 가질 때 정합 블록 후보는 새로운 최적 정합 블록이 되지만, 정합 블록 후보에서 구해진 SAD가 *curr_min_SAD*보다 크거나 같게 되면 정합 블록 후보는 새로운 최적 정합 블록이 될 수 없어 제거된다. 이때 정합 블록 후보에서 계산되는 SAD연산은 계산 집중적인 연산임으로 매우 많은 연산량을 필요로 한다. 따라서 정합 블록 후보 제거 시, 매우 많은 연산량을 필요로 하는 SAD를 사용하지 않고 정합 블록 후보를 제거할 수 있다면 연산량을 감소시킬 수 있다.

MSEA에서는 식 (12)의 성질을 기초로 연산량이 작은 $\mathrm{SAD_SB}_l$을 사용하여 정합블록후보가 새로운 최적정합블록이 될 수 없음을 SAD 계산이전에 알 수 있어 연산량을 줄일 수 있다. $\mathrm{SAD_SB}_l$의 계산은 [9]에서 제안된 효과적인 방법으로 구해지고 SAD보다 훨씬 적은 연산량이 소요된다. 또한 l의 값이 작을수록 $\mathrm{SAD_SB}_l$계산에 필요한 연산량은 작아진다.

따라서 수식 (12)에 기초한 MSEA에서는 정합블록후보를 제거하기 위하여, 가장 연산량이 작은 $\mathrm{SAD_SB}_0$를 계산 후 *curr_min_SAD*와 비교한다. 이때 *curr_min_SAD*가 $\mathrm{SAD\_SB}_0$ 보다 작거나 같으면 식 (12)에 따라 *curr_min_SAD*는 SAD 보다 반드시 작거나

같게 됨으로 새로운 최적 정합 블록이 될 수 없어 정합 블록 후보는 제거된다. 이때 정합 블록 후보가 0-level에서 제거된다고 한다. 그러나 *curr_min_SAD*가 SAD_SB_0 보다 크게 되면 정합 블록 후보를 제거하기 위하여 l을 증가시키면서 정합 블록 후보를 제거하는 과정을 반복한다(step 6-12). L-level MSEA에서는 정합 블록 후보를 제거하기 위한 과정을 최고 L-level까지 수행할 수 있다. 만약 L-level MSEA의 경우 0-level에서부터 L-level까지 정합 블록 후보를 제거하기 위한 과정을 반복하였어도 제거되지 않았다면 SAD를 계산하여 *curr_min_SAD*와 비교한다. 이때 *curr_min_SAD*가 SAD보다 작거나 같으면 정합 블록 후보는 제거되나, SAD가 작게되면 정합 블록 후보는 새로운 최적 정합 블록이 되고 *currr_min_SAD*는 SAD 값으로 대치된다(step 15).

MSEA를 적용하는 경우 0-level에서 많은 정합블록후보들이 제거되고, 특히 L_{max}-level까지 수행하면 대부분의 정합 블록 후보들이 제거되기 때문에, MSEA알고리즘의 step 14의 SAD계산 전에, 작은 연산량이 소요되는 SAD_SB_l만을 이용하여 step 6~12에서 정합 블록 후보들을 제거할 수 있어 연산량을 획기적으로 줄일 수 있다.

이러한 과정이 탐색 윈도우 내에 있는 모든 탐색점에 대하여 수행된 후의 최종 *curr_min_SAD*가 최소 SAD가 되며 최소 SAD를 가지는 블록이 최적 정합 블록이 된다. 이 때 최적 정합 블록의 좌표에서 참조 블록의 좌표를 뺀 상대적인 변위가 참조 블록의 움직임 벡터가 된다.

수식 (12)에 기초한 L-level MSEA의 알고리즘이 그림 2에 나타나 있다.

Algorithm: L-level MSEA
step 1 select an initial search point within the search window in the previous frame
step 2 calculate SAD at the selected search point and let the SAD as current minimum SAD (*curr_min_SAD*=SAD)
step 3 while(all the search points in the search window are tested?){
step 4 delete_flag=0;
step 5 select another search point within the search window
step 6 for(i=0; $i \le L$; i++){
step 7 calculate SAD_SB_i at the selected search point
step 8 if(*curr_min_SAD* $\le SAD\_SB_i$){
step 9 delete_flag=1;
step 10 break;
step 11 }
step 12 }
step 13 if(delete_flag==0){
step 14 calculate SAD at the selected search point
step 15 if(*curr_min_SAD*〉 SAD) *currr_min_SAD*=SAD;
step 16 }
step 17 the minimum SAD= *curr_min_SAD*
step 18 calculate Motion Vector(***MV***) at the search point which has the minimum SAD
step 19 the optimum motion vector= ***MV***

그림 2. L-level MSEA알고리즘

III. A Fast Block Matching Algorithm Using Hierarchical Search Point Sampling

다단계 연속제거 알고리즘과 거의 유사한 정확도를 유지하면서 블록 정합 연산량을 감소시키기 위하여 그림 3과 같이 계층적으로 표본 추출된 탐색점들에 대하여 다단계 연속제거 알고리즘을 적용하여 블록 정합을 수행한다. 제안된 알고리즘의 절차는 그림 4과 같다.

Level 0에서는 spiral search pattern에 따라 검은색으로 표시된 탐색점들에 대하여 다단계 연속제거

알고리즘을 적용한 블록 정합을 순차적으로 수행한다. Level 0에서 spiral search pattern을 따라 검은 탐색점들에 대하여 블록 정합을 수행할 때, 계산된 SAD값이 현재까지의 최소 SAD값보다 적을 경우에는 현재의 탐색점의 좌표를 저장한다. 이러한 과정을 level 0의 모든 탐색점들에 대하여 수행한다.

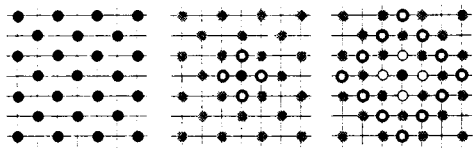

그림 3 Hierarchical search point sampling

Level 0에서의 블록 정합이 완료되면 저장 탐색들에 대하여 순서대로 Level 1에서의 블록 정합을 수행한다. 그림 3의 (b)에서 보는 바와 같이 Level 1에서는 반지름이 1인 주위의 4개의 탐색점들에 대하여 블록 정합을 수행한다. Level 2에서는 반지름이 3인 주위의 13개의 탐색점들에 대하여 블록 정합을 수행하고 level n에서는 반지름이 $2n$+1인 주위의 점들에 대하여 블록 정합을 수행한다. Level 0부터 level k까지 계층적으로 수행하는 알고리즘을 $FMSEA_k$로 표기한다. 탐색윈도우가 31x31(±15)일 때 k의 최대 값은 7이 되고 이 때 제안된 알고리즘은 MSEA알고리즘과 동일하게 된다. k값을 변화시킴으로 움직임 추정의 정확도와 연산량을 조절할 수 있다.

step 1 select an initial black colored search point within the search window in level 0 of the previous frame
step 2 calculate SAD at the selected search point and let the SAD as current minimum SAD (*curr_min_SAD*=SAD)
step 3 while(all the black colored search points in the level 0 are tested?){
step 4 delete_flag=0;
step 5 select another black colored search point in level 0
step 6 for(i=0; $i \le L$; i++){
step 7 calculate SAD_SB_i at the selected search point
step 8 if(*curr_min_SAD*≤SAD_SB_i){
step 9 delete_flag=1;
step 10 break;
step 11 }
step 12 }
step 13 if(delete_flag==0){
step 14 calculate SAD at the selected search point
step 15 if(*curr_min_SAD*〉 SAD) {
currr_min_SAD=SAD;
push the coordinate of the search point into a queue)
step 16 }
step 17 while(all the search points in the queue are tested?){
step 18 while(all the neighboring black circled search points with radius 1 to $2k$+1 in level 1 to k are tested?){
step 19 select an black circled search point which is not tested
step 20 for(i=0; $i \le L$; i++){
step 21 calculate SAD_SB_i at the selected search point
step 22 if(*curr_min_SAD*≤SAD_SB_i){
step 23 delete_flag=1;
step 24 break;
step 25 }
step 26 }
step 27 if(delete_flag==0){
step 28 calculate SAD at the selected search point
step 29 if(*curr_min_SAD*〉 SAD)
currr_min_SAD=SAD;
step 30 }
step 31}
step 32 the minimum SAD= *curr_min_SAD*
step 33 calculate Motion Vector(***MV***) at the search point which has the

minimum SAD
step 33 the optimum motion vector= ***MV***

그림 4. L-level MSEA에 적용된 제안된 알고리즘(k계층)

IV. 실험결과

본 연구의 실험에서 사용한 영상은 176x144 pixel 크기를 가지는 miss_america.qcif, trevor.qcif의 100 프레임(frame)에 대하여 실험을 수행하였다. 프레임에서 움직임 벡터를 구하는 블록의 크기는 16x16 pixel로 하였고 탐색 윈도우의 크기는 31x31 pixel이고 움직임 벡터는 정수 값만을 고려하였다. 이러한 실험환경은 [19]에서의 실험환경과 동일하다.

표에서 (in row)는 16x16 블록의 1행에 대한 sum norm을 구하는데 필요한 연산량을 기본 단위로 표시하였음을 나타낸다. $MSEA_3$는 3-level MSEA를 나타낸다. 움직임 추정 정확도는 9,900개의 움직임 벡터를 찾는 경우, 정확하게 찾아진 움직임 벡터의 개수를 나타낸다.

다단계 연속 제거 레벨 알고리즘을 이용하여 블록 정합을 수행할 때 가장 정합이 잘될 것으로 예상되는 블록을 먼저 테스트하면 효율적이게 된다. 본 연구에서는 이러한 특성을 잘 만족시키는 것으로 알려진 나선형 탐색(spiral search)기법을 사용하였다.

제안한 알고리즘을 비디오 시퀀스에 대하여 실험한 결과는 표1과 같다

실험결과에서 보는 바와 같이 제안한 알고리즘에서 계층의 깊이를 깊게 할수록 정확도는 증가함을 알 수 있다. 제안한 알고리즘은 계층적인 탐색점 추출의 레벨이 클수록 움직임 추정 정확도가 증가하나 연산량은 커지게 된다. 제안한 알고리즘에서 움직임 추정 정확도가 MSEA와 같게 유지되는 경우, miss_america와 trevor에 대하여 MSEA의 연산량이 최대 13.0% 13.6% 감소하였다.

<표 1> $FMSEA_k$의 연산량과 움직임 추정 정확도

video sequence	Algorithm	평균 정합 평가 회수 /frame	평균 row /정합 평가	오버헤드 (in rows)	총 연산량	연산량 감소율 (%)	움직임 추정 정확도
miss_am	$MSEA_3$	358.6	14.89	28,280.7	33,620.3		9900
	$FBMA_1$	260.4	14.97	15,699.0	195,97.2	41.7	9874
	$FBMA_2$	282.5	14.69	17,082.5	21,232.4	36.8	9887
	$FBMA_3$	302.9	14.18	18,644.0	22,939.1	31.8	9894
	$FBMA_4$	319.6	13.47	20,527.2	24,832.2	26.1	9898
	$FBMA_5$	332.3	12.34	22,791.1	26,891.7	20.0	9899
	$FBMA_6$	342.8	10.73	25,584.3	29,262.5	13.0	9900
	$FBMA_7$	350.4	8.93	29,097.2	32,226.3	4.1	9900
trevor	$MSEA_3$	1,037.2	14.12	29,767.9	44,413.2		9900
	$FBMA_1$	970.1	14.23	21,565.8	35,370.3	20.4	9888
	$FBMA_2$	985.4	13.38	22,776.1	35,960.8	19.0	9897
	$FBMA_3$	997.5	12.89	24,268.5	37,126.3	16.4	9899
	$FBMA_4$	1,007.8	12.19	26,090.6	38,375.7	13.6	9900
	$FBMA_5$	1,016.7	1109	28,226.3	39,501.5	11.1	9900
	$FBMA_6$	1,024.0	9.51	30,871.5	40,609.7	8.6	9900
	$FBMA_7$	1,029.2	7.72	34,473.2	42,418.6	4.5	9900

V. 결론

FSA의 연산량을 매우 효율적으로 감소시키는 움직임 추정 알고리즘인 MSEA의 연산량을 줄이는 제안된 알고리즘은 탐색점을 계층적으로 표본 추출하여 MSEA와 거의 동일한 정확도를 유지하면서 움직임 추정 연산량을 감소시킨 효율적인 움직임 추정알고리즘이다. 제안된 알고리즘에서 $FMSEA_k$의 k값을 적절히 조절하여 움직임 추정정확도와 움직임 추정 연산량을 조절 할 수 있다. 제안된 알고리즘에서 움직임 추정 정확도가 MSEA와 같게 유지되는 경우, miss_america와 trevor 각각에 대하여 MSEA의 연산량이 최대 13.0%, 13.6% 감소하였다.

References

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

[2] 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

[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. G5.3.1- 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] 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

[11] 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

[12] 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

[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] C. H. Lee, and L. H. Chen, "A Fast

Motion Estimation Algorithm Based on the Block Sum Pyramid," IEEE Trans. Image Processing, Vol. 6, No.11, pp. 1587-1591, Nov. 1997.

[19] 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

[20] 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.

정수목

1984 경북대학교
전자공학과(학사)
1986 경북대 대학원
전자공학과(석사)
1986~1991 LG 정보 통신
연구소 연구원
2002 고려대 대학원
컴퓨터학과(박사)
1998~현재 삼육대학교 컴퓨터학과 부교수
관심분야: 멀티미디어, 컴퓨터시스템