논문 2010-47SD-7-10

효과적인 조기 중단 기법을 위한 변형된 3단계 탐색 움직임 추정 알고리즘

(Modified 3-step Search Motion Estimation Algorithm for Effective Early Termination)

양 현 철*, 이 성 수**

(Hyeon-Cheol Yang and Seongsoo Lee)

요 약

움직임 추정은 동영상 압축에서 가장 많은 연산량을 차지하는 부분으로 막대한 연산량을 줄이기 위한 많은 고속 탐색 기법이 제안되어 왔다. 움직임 추정에서 가장 많은 연산량을 차지하는 SAD (sum-of-absolute difference) 계산의 경우, 연산량을 줄이기 위해 SAD 계산 중간에 지금까지 계산된 중간값이 지금까지 찾아진 최소 SAD를 넘을 경우 더 이상의 SAD 계산을 중단하고 다음 탐색으로 넘어가는 조기 중단 기법이 많이 사용되고 있다. 본 논문에서는 대표적인 고속 탐색 기법인 3단계 탐색 기법을 변형하여 조기 중단이 자주 일어나도록 탐색 위치의 탐색 순서만을 적응적으로 재배열하는 움직임 추정 기법을 제안하였다. 모의 실험 결과, 제안하는 움직임 추정 기법은 추가 연산량이 거의 없이 기존의 3단계 탐색 기법에 비해 동일한 성능을 유지하면서 연산량을 17~30% 감소시켰다.

Abstract

Motion estimation occupies most of the required computation in video compression, and many fast search algorithms were propsoed to reduce huge computation. SAD (sum-of-absolute difference) calculation is the most computation-intensive process in the motion estimation. Early termination is widely used in SAD calculation, where SAD calculation is terminated and it proceeds to next search position if partial SAD during SAD calculation exceeds current minimum SAD. In this paper, we proposed a modified 3-step search algorithm for effective early termination where only search order of search positions are adaptive rearranged. Simulation results show that the proposed motion estimation algorithm reduces computation by 17~30% over conventional 3-step search algorithm without extra computation, while maintaining same performance.

Keywords: 움직임 추정, 3단계 탐색, 고속 탐색, 조기중단기법

I. 서 론

일반적으로 영상정보는 정보량이 큰 정보의 하나이므로 매체에 저장 또는 전송하기 위해서 정보량을 줄이는 과정이 필요하다. 이를 위해서 만들어진 동영상 압축 규약에는 MPEG-1, MPEG-2, MPEG-4, H.263, H.264[1]등이 있다. 영상정보의 양을 줄이기 위해서는 공간적인 유사성을 제거하기 위해 DCT (discrete cosine transform) 또는 IT (integer transfrom)등을 사용하거나, 정보의 상관관계를 이용하여 잉여 데이터를 줄이는 VLC (variable length coding)등의 방법을 사용하기도 한다. 또한, 시간이 다른 각각의 영상에 대해서 상관관계를 이용하여 정보량을 줄이는 움직임 추정 연산을 사용하게 된다.[2] 대부분의 방법에서 움직임 추정이라는

* 정회원, ** 평생회원, 숭실대학교 정보통신전자공학부
(School of Electronic Engineering, Soongsil University)

※ 이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2007-313-D00585)

접수일자: 2010년2월26일, 수정완료일: 2010년6월16일

연산의 경우 다른 연산에 비해 꽤 많은 연산량을 가지게 되며, 최근에 사용되고 있는 H.264/AVC의 경우 전체 부호화 연산의 약 30~40%정도를 차지할 정도로 많은 연산량을 가지는 것으로 이를 위한 여러 알고리즘들이 제안되고 있고, 또한 사용되고 있다.[3~5]

본 논문에서는 이러한 움직임 추정의 여러 가지 알고리즘 중 대표적인 알고리즘의 하나인 3단계 탐색 기법(3-step search)[6]에 대해 연산량을 줄일 수 있도록 조기 중단 기법의 효율을 높여줄 수 있는 움직임 추정 알고리즘을 제안하도록 한다.

II. 움직임 추정 방법 및 조기 중단 기법

1. 움직임 추정 알고리즘

움직임 추정에는 화소 단위로 구하는 화소 반복법과 블록 단위로 구하는 블록 정합법이 있으나 비용대비 성능이 좋은 블록 정합법 (block matching)을 주로 사용한다. 움직임 추정을 위한 유사도를 구하는 방법에는 SAD (sum of absolute difference), MSE (mean square error)등의 방법이 있다. U×V 크기를 갖는 블록에 대한 SAD를 구하는 식은 식 (1)과 같다.

$$SAD = \sum_{i=0}^{U-1}\sum_{j=0}^{V-1} |Cur_{i,j} - Ref_{i,j}| \quad (1)$$

이를 이용해 탐색하고자 하는 탐색 영역 전체에 대해서 SAD연산을 수행하는 전역 탐색 기법 (full search method)과 특정 표본 지점에 대해서만 연산을 하는 고속 탐색 기법으로 나누어진다. 고속 탐색 기법의 대표적인 방법은 N단계 탐색 기법으로 이 방법은 각 단계별로 n을 N부터 1까지 감소시켜가며 스텝 크기 (step size) $S=2^{n-1}$을 구하고, 원점과 이를 중심으로 ±S만큼 떨어진 9개의 지점에 대해서 유사도를 측정하여 가장 큰 유사도를 가지는 지점을 찾아낸다. 다음 단계에서는 이전 단계에서 찾은 지점을 기준으로 다시 다음 스텝 크기만큼 떨어진 9개의 지점에 대해 유사도를 측정하며, 최종적으로 S가 1이 될 때까지 반복하여 움직임 벡터를 구하게 된다.

N=3에 해당하는 3단계 탐색 기법에서는 그림 1과 같이 각 단계마다 9개의 탐색 위치에 대해 SAD를 계산하고 비교하며, 이전 단계에서 SAD 값이 가장 작았던 탐색 위치를 기본으로 탐색 범위를 1/2로 줄여서 다시 9개의 탐색 위치에 대해 SAD를 계산한다. 실제로는 9개

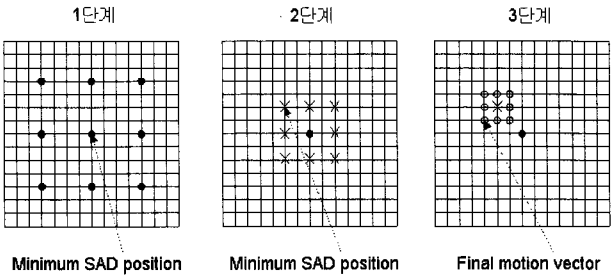


그림 1. 3단계 탐색 기법
Fig. 1. 3-step search.

의 탐색 위치 중에서 중심에 있는 탐색 위치는 이전 단계에서 이미 SAD 값이 계산된 바 있으므로 별도의 SAD 계산을 수행하지 않기 때문에 1단계에서는 9개의 탐색 위치, 2단계와 3단계에서는 8개의 탐색 위치, 총 25개의 탐색 위치에 대해 SAD 계산을 수행한다.

N단계 탐색 기법은 전역 탐색 방식에 비해 연산해야 할 탐색 지점이 크게 줄어들지만 성능 저하는 비교적 크지 않고 하드웨어 구현도 비교적 간단하기 때문에 고속 탐색 기법 중 가장 많이 사용되는 방법이다.

2. 조기 중단 기법

움직임 추정 연산은 기본적으로 가장 유사도가 높은 지점을 찾는 연산이다. 유사도가 가장 높은 지점은 다시말해 두 영상의 차분값인 SAD값이 가장 작은 지점을 의미한다. 다시 말해 움직임 추정연산은 최소 SAD 지점을 찾는 연산을 의미하는 것이다.

이를 이용하여 기존에 구하여진 최소 SAD의 값보다 현재 연산중인 SAD의 중간 연산값이 더 커질 경우 더 이상 연산을 수행하지 않고 다음 탐색 위치에 대해서 연산을 수행하는 방법을 조기 중단 기법 (early termination)[7~8]이라고 한다. 조기 중단 기법을 사용할

```
...
for(j=0;j<V;j++) {
   for(i=0;i<U;i++) {
      SAD += [sad operation]
         if ( minimum_SAD < SAD )
            [escape loop]
   }
}
...
```

그림 2. 조기 중단 기법의 가상코드
Fig. 2. Pseudo-code of early termination.

경우 필요없는 잉여 연산을 수행하지 않으므로 연산수를 크게 줄일 수 있다.

이러한 조기 중단 기법은 최종 움직임 벡터에 대한 SAD의 값이 작을수록 조기 중단이 자주 일어나서 연산을 빨리 종료할 수 있어 움직임 추정에 필요한 연산량이 작아진다.

Ⅲ. 제안하는 알고리즘

통계적으로 현재 블록의 SAD의 값은 주변블록의 SAD값과 비슷한 값을 가질 확률이 높다.[9] 마찬가지로 움직임 벡터는 주변 블록의 움직임 벡터와 비슷한 값을 가지고 있을 확률이 높아서 이들을 잘 활용하면 효과적으로 움직임 추정을 수행할 수 있다. H.264/AVC 등 많은 동영상 압축 기법에서는 이미 이러한 방식의 움직임 벡터의 예측을 사용하고 있다.

움직임 벡터의 예측 방법에는 여러 가지 형태의 방법들이 있으나 일반적으로는 그림 3과 같이 이전에 움직임 추정이 끝난 블록들인 블록 A, 블록 B, 블록 C의 움직임 벡터를 이용하여 현재 블록의 움직임 벡터를 예측하게 된다. H.264/AVC의 경우 부호화기 내에서 참고블록들의 움직임 벡터값의 메디안 (median) 값을 현재 블록 움직임 벡터의 예측값으로 결정하고 이 값을 부호화에 사용하고 있다.

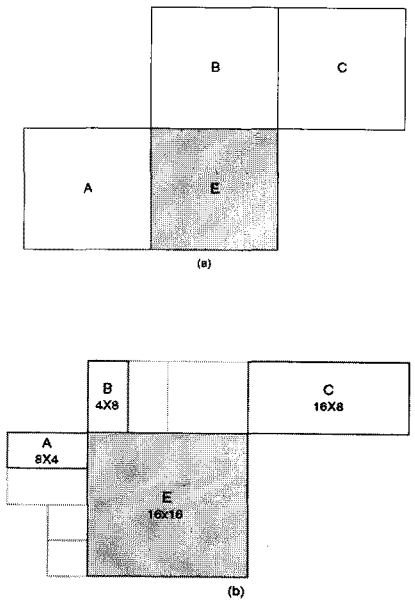


그림 3. 움직임 벡터 예측을 위한 참고블록
(a) 크기가 같은 블록인 경우
(b) 크기가 다른 블록인 경우

Fig. 3. Reference block for motion vector prediction.
(a) Case of same sized block
(b) Case of different sized block

조기 중단 기법에서는 기존에 구하여진 최소 SAD의 값보다 현재 연산중인 SAD의 중간 연산값이 더 커질 경우 더 이상 연산을 수행하지 않기 때문에, 조기 중단 기법의 연산량을 줄이기 위해서는 탐색해야 할 탐색 위치 중에서 SAD가 가장 작을 것으로 생각되는 탐색 위치부터 먼저 탐색하는 것이 바람직하다. 즉, 먼저 탐색한 탐색 위치의 SAD가 작을수록 나중에 탐색한 탐색 위치의 SAD가 쉽게 최소 SAD를 넘기 때문에 조기 중단이 더 자주, 더 빨리 일어남으로 연산량이 감소한다.

일반적인 3단계 탐색 기법의 경우, 1단계는 9개, 2단계 이후는 8개의 탐색 위치를 고정적인 레스터 순서 (raster order), 즉 좌상, 중상, 우상, 좌중, 우중, 좌하, 중하, 우하 순서로 탐색한다. 이에 반하여 본 논문에서 제안하는 알고리즘은 조기 중단 기법이 가장 효율적으로 적용될 수 있도록 현 단계에서 탐색해야 할 탐색 위치 중에서 움직임 벡터의 예측 값과 가장 가까운 순서, 즉 SAD가 최소가 될 것으로 예측되는 순서대로 탐색 순서를 적응적으로 재정렬함으로서 조기 중단이 일어날 확률을 높이고 이에 따라 연산량을 줄일 수 있다.

일반적으로 현재 블록 움직임 벡터의 예측값은 확률적으로 SAD가 최소가 될 확률이 가장 높으며, 이 예측값에 가까운 탐색 위치일수록 SAD가 최소가 될 가능성이 높으므로 본 논문에서는 3단계 탐색 기법에서 단계별로 탐색해야 할 탐색 위치를 현재 블록 움직임 벡터의 예측값에 가까운 순서대로 탐색하는 방법을 사용하였다. H.264/AVC 등 많은 동영상 압축 기법에서는 이미 부호화 내의 여러 부분에서 움직임 벡터 예측을 사용하고 있으므로 움직임 벡터의 예측값을 구하기 위하여 별도의 추가 연산이 필요하지는 않으며, 움직임 벡터 예측을 사용하지 않는 다른 동영상 압축 기법에서도 매우 간단한 연산을 통해서 움직임 벡터의 예측이 가능하다.

이론상으로는 탐색위치마다 움직임 벡터 예측값과의 공간적 거리를 구하여 이 순서대로 탐색 순서를 결정하는 것이 가장 효율적이나, 실제로는 공간적 거리를 구하기 위하여 곱셈 연산이 필요하고 공간적 거리 순서대로 탐색을 하기 위해서는 정렬 (sorting) 연산이 필요하다는 문제점이 있다. 따라서 본 논문에서는 각 단계별로 9개의 탐색 위치를 중심으로 탐색 영역을 그림 4와 같이 9개의 구역으로 나누고, 움직임 벡터 예측값이 속

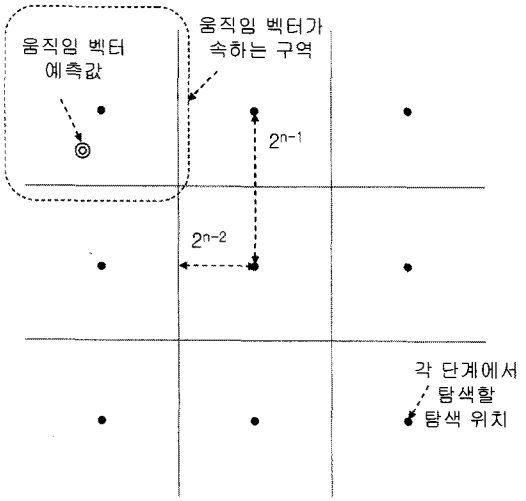


그림 4. 탐색 영역의 분할
Fig. 4. Division of search area.

한 구역에서 거리상으로 가까운 구역에 있는 탐색 위치부터 탐색을 수행한다.

$$MV_{dist}(x) = MV_{predicted}(x) - MV_{nstep\_center}(x) \quad (2)$$

$$MV_{dist}(y) = MV_{predicted}(y) - MV_{nstep\_center}(y) \quad (3)$$

식 (2),(3)과 같이 움직임 벡터 예측값 $MV_{predict}(x,y)$와 9개 탐색 위치의 중심값 $MV_{nstep_center}(x,y)$를 x, y 좌표 별로 비교하여 $MV_{dist}(x)$, $MV_{dist}(y)$값을 얻고, 이 값을 $\pm 2^{n-2}$와 비교하면 움직임 벡터 예측값이 속한 구역을 찾을 수 있다. 이 과정에서는 각 단계별로 6번의 뺄셈 연산이 필요한데, 단계별로 SAD 계산에 필요한 6,144번의 연산에 비하면 제안하는 알고리즘으로 인해 추가되는 연산은 극히 미미하다.

1단계는 9개, 2단계 이후는 8개의 탐색 위치에 대해 움직임 추정을 수행하는 순서는 움직임 벡터 예측값이 속한 구역이 그림 4의 9개 구역중 어느 것이냐에 따라 그림 5와 같이 정해진다.

그림 5의 순서는 1)-5)의 원칙에 의해 정해졌으며, 기본적으로 1)에서 움직임 벡터 예측값이 속한 구역으로부터 가까운 구역 순으로 탐색을 수행하되 같은 거리에 있는 구역끼리는 2)-5)순으로 탐색을 수행하도록 하였다.

1) 기하학적인 거리가 짧은 구역 순
2) y 방향 거리가 짧은 구역 순
3) x 방향 거리가 짧은 구역 순
4) 왼쪽에서 오른쪽 구역 순
5) 위쪽에서 아래쪽 구역 순

그림 5는 미리 정의되어 있는 순서이기 때문에, 실제로 움직임 추정을 수행할 때는 1)-5) 원칙에 따라 매번 탐색 순서를 계산하는 것이 아니라 해당되는 타입만 골라내면 된다. 따라서 이 과정에서도 제안하는 알고리즘으로 인해 추가되는 연산은 매우 미미하다.

기존 3단계 탐색 기법과 제안하는 알고리즘에서 탐

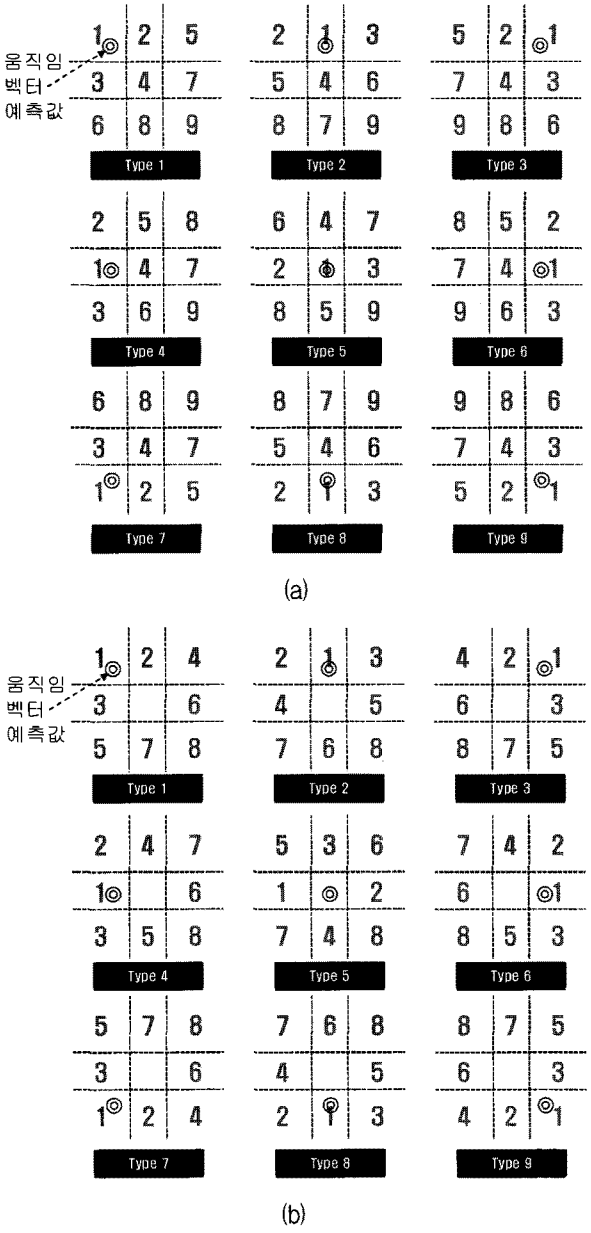


그림 5. 움직임 벡터 예측값에 따른 탐색 순서
(a) 1단계 (b) 2단계 이후
Fig. 5. Search order according to predicted motion vector.
(a) 1st step (b) 2nd step and so on

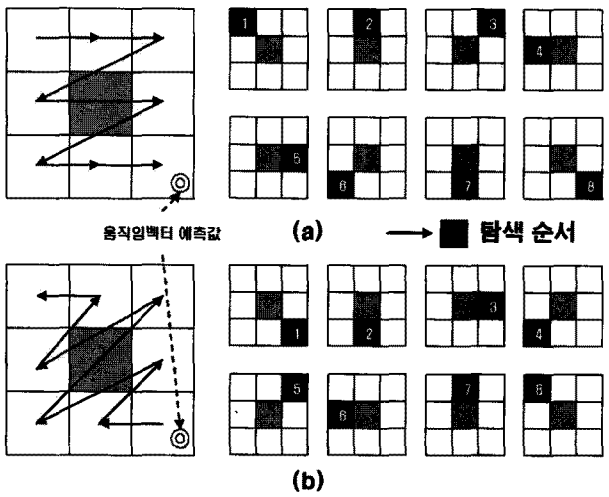


그림 6. 기존 알고리즘과의 비교
(a) 기존 알고리즘 (고정적 순서)
(b) 제안하는 알고리즘 (적응적 순서)

Fig. 6. Comparison with conventional algorithm.
(a) Conventional algorithm (fixed order)
(b) Proposed algorithm (adaptive order)

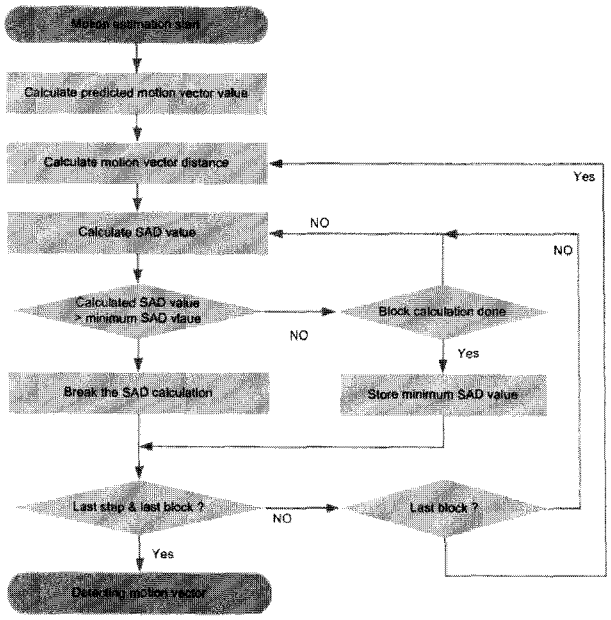


그림 7. 제안하는 알고리즘의 순서도

Fig. 7. Flowchart of proposed algorithm.

색 순서의 차이는 그림 6과 같다. 기존 3단계 탐색 기법의 경우에는 움직임 벡터 예측값에 관계없이 탐색 순서가 고정적이나 제안하는 알고리즘에서는 움직임 벡터 예측값에 따라 탐색 순서가 적응적으로 재정렬되므로 조기 중단 기법을 효과적으로 적용할 수 있다.

본 논문에서 제안하는 움직임 추정 알고리즘의 순서도는 그림 7과 같다.

Ⅲ. 모의 실험 결과

제안하는 알고리즘을 검증하고 성능을 평가하기 위해, 현재 사용되는 여러가지 동영상 부호화 기법 중 가장 많이 사용되는 H.264/AVC를 선택하였으며, 부호화 모델로 JM15.0[10]을 기반으로 모의 실험을 실시하였다. 움직임 추정 알고리즘은 다음과 같은 알고리즘을 비교하였으며, 조기 중단 기법은 1)~4) 모두에 적용되었다.

1) TSS: 기존의 3단계 탐색 기법
2) UMHEX: Hexagonal search[11]를 이용한 기법
3) EPSZ: Efficient Predictive Zonal search[12]를 이용한 기법
4) Proposed: 본 논문에서 제안하는 변형된 3단계 탐색 기법

부호화 조건은 H.264/AVC baseline profile을 사용하고 QP (quality parameter)값은 28로 고정하였으며, 움

표 1. 모의실험 결과
(a) QCIF 영상 (b) CIF 영상 (c) D1 영상

Table 1. Simulation result.
(a) QCIF image (b) CIF image (c) D1 image

(a)

표준영상	알고리즘	SAD연산수	PSNR(db)	연산비
akiyo	TSS	422271088	38.13	1.00
	UMHEX	179890698	38.10	0.43
	EPSZ	263247972	37.75	0.62
	Proposed	316839430	38.13	0.75
container	TSS	530342964	36.25	1.00
	UMHEX	218716781	36.25	0.41
	EPSZ	301519938	36.23	0.57
	Proposed	424629928	36.25	0.80
foreman	TSS	569400516	36.12	1.00
	UMHEX	452671985	36.11	0.79
	EPSZ	411415426	36.09	0.72
	Proposed	463907860	36.12	0.81
hall	TSS	411796300	37.80	1.00
	UMHEX	179977719	37.77	0.44
	EPSZ	241260379	37.81	0.59
	Proposed	306060408	37.80	0.74
average	TSS	483452717	37.08	1.00
	UMHEX	257814296	37.06	0.52
	EPSZ	304360929	36.97	0.63
	Proposed	377859407	37.08	0.78
maximum	TSS	569400516	N/A	1.00
	UMHEX	452671985	N/A	0.79
	EPSZ	411415426	N/A	0.72
	Proposed	463907860	N/A	0.81

(b)

표준영상	알고리즘	SAD연산수	PSNR(db)	연산비
akiyo	TSS	1813062445	39.72	1.00
	UMHEX	723640644	39.72	0.40
	EPSZ	1007382717	39.69	0.56
	Proposed	1390975960	39.72	0.77
container	TSS	2302820300	36.29	1.00
	UMHEX	1161406232	36.28	0.50
	EPSZ	1258071258	36.29	0.55
	Proposed	1879444935	36.28	0.82
foreman	TSS	2469885208	36.79	1.00
	UMHEX	2133927647	36.74	0.86
	EPSZ	1922946270	36.74	0.78
	Proposed	2049716344	36.79	0.83
hall	TSS	2040487900	38.13	1.00
	UMHEX	1062115861	38.11	0.52
	EPSZ	1106211349	38.10	0.54
	Proposed	1619351866	38.13	0.79
average	TSS	2156563963	37.73	1.00
	UMHEX	1270272596	37.71	0.57
	EPSZ	1323652899	37.70	0.61
	Proposed	1734872276	37.73	0.80
maximum	TSS	2469885208	N/A	1.00
	UMHEX	2133927647	N/A	0.86
	EPSZ	1922946270	N/A	0.78
	Proposed	2049716344	N/A	0.83

(c)

표준영상	알고리즘	SAD연산수	PSNR(db)	연산비
mobcal	TSS	7995642249	36.55	1.00
	UMHEX	8373802682	36.52	1.05
	EPSZ	7151431426	36.52	0.89
	Proposed	6536130487	36.55	0.82
parkrun	TSS	7944748144	34.44	1.00
	UMHEX	11218414568	34.43	1.41
	EPSZ	8225301727	34.43	1.04
	Proposed	6494284067	34.43	0.82
shields	TSS	8002844783	35.70	1.00
	UMHEX	7831610350	35.68	0.98
	EPSZ	7612528665	35.68	0.95
	Proposed	6554278145	35.70	0.82
stockholm	TSS	7847124932	35.82	1.00
	UMHEX	6247134066	35.80	0.80
	EPSZ	6176077032	35.79	0.79
	Proposed	6389100453	35.82	0.81
average	TSS	7947590027	35.63	1.00
	UMHEX	8417740417	35.61	1.06
	EPSZ	7291334713	35.60	0.92
	Proposed	6493448288	35.63	0.82
maximum	TSS	8002844783	N/A	1.00
	UMHEX	11218414568	N/A	1.40
	EPSZ	8225301727	N/A	1.03
	Proposed	6554278145	N/A	0.82

직임 추정은 정수 화소 정밀도 (integer-pel precision)만을 사용하였다. H.264/AVC의 경우 부호화 기법 내에서 이미 움직임 벡터 예측값을 구해서 부호화에 사용하기 때문에 제안하는 알고리즘이 움직임 벡터 예측을 구하는데 추가 연산량이 필요하지 않다.

영상의 크기에 따라 어느 정도 성능 차이를 보이는지를 확인하기 위하여 QCIF (176 × 144), CIF (352 × 288), D1 (720 × 480) 세 가지 크기의 표준 영상을 사용하였다.

표준 영상마다 300 프레임씩 움직임 추정을 수행하였을 때 PSNR과 SAD 연산수는 표 1과 같다. 연산비는 기존의 N단계 탐색 기법과 조기 중단 기법을 적용한 NSS(N-step search)를 1.00으로 보고 이를 기준으로 정규화한 값이다. NSS와 제안하는 방법은 탐색 순서만 다르기 때문에 이론상으로는 항상 동일한 움직임 벡터를 찾아내고 PSNR도 항상 동일하여야 한다. 그러나 실제로는 타이 브레이크 (tie-break), 즉, 두 개 이상의 탐색 위치가 동일한 최소 SAD를 가질 때 어느 탐색 위치를 다음 단계의 기준점으로 삼을지가 탐색 순서에 의해 결정되기 때문에 PSNR이 아주 미세한 차이를 보일 수 있다. 표에서 확인한 바와 같이, 제안하는 알고리즘은 기존의 3단계 탐색 기법과 조기 중단 기법에서 단지 탐색 순서만을 바꿈으로서 추가 연산량이나 PSNR 성능 저하 없이 SAD 연산수를 17~30%, 평균적으로 21% 가량 줄일 수 있었다. 또한 제안하는 알고리즘의 SAD 연산량 감소는 사용된 표준 영상의 크기나 종류에 비교적 무관하게 이루어지는 것을 알 수 있다.

기존의 3단계 탐색외에 레퍼런스 소프트웨어에서 기본적으로 사용하는 고속 탐색 방법인 hexagonal search(UMHEX) 또는 predictive zonal search(EPSZ)와의 비교에서는 영상의 크기에 따라 다소 차이를 보인다. 기존의 UMHEX와 EPSZ 기법에 비해 제안하는 알고리즘은 QCIF 영상과 CIF 영상에서 화질은 약간 우수하나 연산량은 20% 정도 많다. 이에 반하여 D1 영상에서는 화질은 약간 우수하면서 연산량도 10~20% 정도 적다.

여기서 주목해야할 점은 기존의 UMHEX와 EPSZ 기법은 제안하는 기법에 비해 데이터 흐름이 복잡하기 때문에 하드웨어 구현도 복잡해지고, 다음 단계에서 읽어들여야 할 탐색 위치가 제안하는 기법에 비해 불규칙적이기 때문에 이미 읽어들인 화소 데이터의 재사용이 어려워서 내부 버퍼의 크기가 증가하며, 제안하는 기법

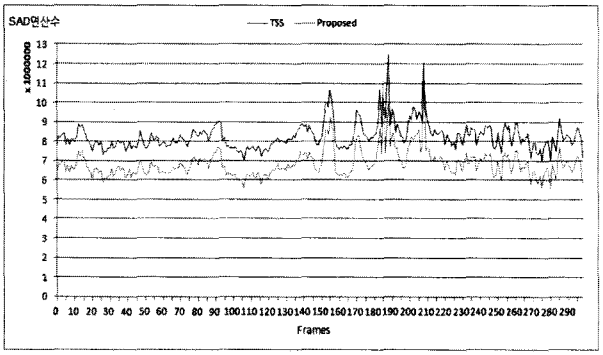


그림 8. foreman 시퀀스의 SAD 연산수 비교
Fig. 8. Comparation of SAD calculation in foreman sequence.

과는 달리 기존에 제안된 수많은 TSS 기반 움직임 추정기 하드웨어를 재활용하기 어렵다는 점이다.

또한, 일반적으로 움직임 추정기는 실시간 동작이 중요하기 때문에 대부분 평균 연산량보다는 최대 연산량이 하드웨어 크기를 결정하게 되는데, 제안하는 기법의 최대 연산량은 기존의 UMHEX와 EPSZ 기법과 비교할 때 QCIF 영상과 CIF 영상에서는 거의 동일하며 D1 영상에서는 확연히 우수함을 알 수 있다. 이러한 점을 종합해볼 때, 제안하는 기법은 기존의 UMHEX와 EPSZ 기법에 비해 하드웨어 구현 측면에서 여러 가지로 우수함을 알 수 있다.

그림 8은 QCIF 크기의 foreman 표준 영상에 대해서 300프레임을 부호화하였을 때의 각 프레임에서의 SAD 연산수를 나타낸 그림인데, 제안하는 알고리즘은 각 프레임마다 비교적 고르게 연산량을 감소시키는 것을 알 수 있다.

이와 같이 제안하는 알고리즘은 영상의 크기, 종류, 프레임 순서에 비교적 관계없이 고르게 연산량을 감소시킬 수 있으므로, 휴대폰부터 방송용 카메라까지 다양한 영상 어플리케이션에 폭넓게 사용이 가능하고 H.264/AVC 뿐만 아니라 움직임 추정을 사용하는 MPEG-2, MPEG-4등의 다른 동영상 부호화 기법에도 큰 문제없이 적용시킬 수 있다. 또한 제안하는 알고리즘은 움직임 추정 기법 자체는 큰 수정 없이 탐색 위치의 탐색 순서만을 재정렬하기 때문에 기존에 설계된 움직임 추정기에도 하드웨어의 수정을 최소화하면서 적용이 가능하고, CPU, DSP, GPU와 같은 프로세서에서 소프트웨어 형태로 수행되는 경우에도 소프트웨어의 수정을 최소화하면서 적응이 가능하다.

Ⅳ. 결 론

본 논문에서는 움직임 추정 기법에서 일반적으로 많이 사용되고 있는 조기 중단 기법의 효율성을 높이기 위해 탐색 위치의 탐색 순서를 적응적으로 재정렬함으로서 SAD가 가장 작을 것으로 생각되는 탐색 위치부터 먼저 탐색하는 기법을 제안하였다.

제안하는 알고리즘을 3단계 탐색 기법에 적용한 결과, 추가 연산량이나 PSNR 성능 저하 없이 SAD 연산수를 1/5가량 감소시킬 수 있었다. 또한 3단계 탐색방법이 다른 방법들에 비해 정형적임을 고려할 때 하드웨어로 구현시 크기가 작고 효과적인 움직임 추정기를 제작할 수 있도록 도움이 될 것으로 보인다. 특히, 제안하고자 하는 알고리즘은 3단계 탐색 방법에 국한된 것이 아닌 조기중단기법을 사용하는 다른 고속 탐색 방법에서도 사용이 가능한 알고리즘으로 다른 알고리즘에서도 연산수 감소의 효과를 보일 것으로 예상된다.

이러한 연산량 감소는 영상의 크기, 종류, 프레임 순서에 비교적 관계없이 고르게 일어나기 때문에 다양한 영상, 어플리케이션, 부호화 기법에 폭넓게 적용이 가능하다. 또한 제안하는 알고리즘은 탐색 위치의 탐색 순서만을 재정렬하기 때문에 기존에 개발된 하드웨어 또는 소프트웨어에도 수정을 최소하면서 적용이 가능하다. 제안하는 알고리즘은 멀티미디어 프로세서나 아플리케이션에서 성능을 높이고 하드웨어 크기, 시스템 자원, 전력 소모를 줄이는데 큰 도움이 될 것으로 생각된다.

참 고 문 헌

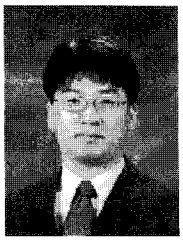
[1] Joint Video Team, Draft. ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A., "Overview of the H.264/AVC video coding standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.

[3] W. Choi and B. Jeon, "Hierarchical motion search for H.264 variable block-size motion compensation," SPIE Opt. Eng., vol. 45, no. 1, pp. 1-9, Jan. 2006.

[4] Y. W. Huang, S. Y. Chien, B. Y. Hsieh, and L. G. Chen, "Global elimination algorithm and architecture design for fast block matching motion estimation" IEEE Trans. Circuits Syst. Video Technology, vol. 14, no. 6, pp. 898-907, Jun. 2004.
[5] S. Y. Huang, C. Y. Cho, and J. S. Wang, "Adaptive fast block-matching algorithm by switching search patterns for sequences with wide-range motion content," IEEE Trans. Circuits System Video Technology, vol. 15, no. 11, pp. 1373-1384, Nov. 2005.
[6] T. Koga, K. Iimuma, A. Hirano, Y. Ijima, T.Ishiguro, "Motion compensated interframe coding for video conferencing", pp. C9.6.1-9.6.5, in Porc. NTC 81.
[7] Guevorkian D.,Launianen, A., Liuha P., Lappalainen V., "Architecture for the sum of absolute differences operation", Signal Processing Systems, 2002(SIPS '02), pp. 57-62, 16-18 Oct. 2002.
[8] C.-Y. Su and S.-L. Chang, "Adaptive early termination for fast H.264 Video Coding," in Proc. 9th IEEE Int. Symp. Multimedia 2007, , pp. 72-76. Taichung, Taiwan.
[9] Z.-B. Chen, J.-F. Xu, Y. He, and J.-L. Zheng, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC," J. Visual Commun. Image Representation, vol. 17, no. 2, pp. 264-290, Apr. 2006.
[10] L. Winger, H.264/AVC reference model version JM15.0, http://iphome.hhi.de/suehring/tml, Jan, 2009.
[11] C. Zhu, X. Lin, and L. P. Chau, "Hexagon-Based search pattern for fast block motion estimation", IEEE Trans. Circuits Syst. Video Technol.,vol. 12, no. 5, pp. 349-355, May 2002.
[12] A.M. Tourapis, "Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation", Proc. Visual Comm. and Image Processing 2002, pp. 1069-79, Janm 2002.

저 자 소 개



양 현 철(정회원)
2004년 숭실대학교 정보통신전자공학부 학사 졸업.
2006년 숭실대학교 전자공학과 석사 졸업
2008년 숭실대학교 전자공학과 박사 수료
<주관심분야 : SoC, H.264/AVC 설계, 저전력 설계, Embedded system>

이 성 수(평생회원)
1991년 서울대학교 전자공학과 학사 졸업
1993년 서울대학교 전자공학과 석사 졸업
1998년 서울대학교 전자공학과 박사 졸업
1998년~2000년 University of Tokyo Research Associate
2000년~2002년 이화여자대학교 정보통신학과 연구교수
2002년~현재 숭실대학교 정보통신전자공학부 부교수
<주관심분야 : 바이오칩 설계, H.264설계, 저전력 설계, SiP설계>