주변 블록의 움직임 벡터를 고려한 수정된 3 단계 탐색 방법

이성호[01] 박일우[1] 조용국[1] 오승준[1] 안창범[2]
1: 광운대학교 전자공학과, 2: 광운대학교 전기공학과
(shlee[0], lrain, aurum, sjoh)@media.gwu.ac.kr and cbahn@daisy.gwu.ac.kr

A Modified Three-Step Search Method Using Motion Vectors at Adjacent Macro Blocks

Sung-Ho Lee[01], Il-Woo Park[1], Yong-Kuk Cho[1], Seoung-Jun Oh[1], and Chang Beom Ahn[2]
1: Dept. of Electronics Engineering, Kwangwoon University, 2: Dept. of Electrical Engineering, Kwangwoon University

요 약

영상압축 인코더에서 가장 많은 계산량을 차지하는 부분이 움직임 벡터를 추정하는 부분이다. 일반적인 3 단계 탐색 (Three-Step Search: TSS) 방법은 한 개의 매크로 블록에 대해 25 개의 탐색점에 대해서 탐색을 한다. 본 논문에서는 주변 움직임 벡터로부터 예측된 움직임 벡터를 또 하나의 탐색점으로 추가하는 수정된 3 단계 탐색 방법을 제안한다. 제안한 방법은 기존 방법 보다 일반적으로 속도를 향상시키면서 최대 15% 정도 비트율을 감소시킬 수 있다.

1. 서 론

움직임 벡터 추정은 동영상 데이터 압축을 위한 핵심 기술이다. 따라서 움직임 벡터 (Motion Vector: MV) 추정에 관한 많은 연구가 진행되어 왔다. 블록 기반 움직임 벡터 예측에서 전역탐색(Full Search: FS) 방법은 가장 정확하게 움직임 벡터를 찾는 반면, 계산량이 많기 때문에 빠른 인코딩을 요구하는 시스템에 사용하는데 문제가 있다. 따라서 고속 알고리즘이 많이 사용되는데 TSS 방법[1] 은 가장 일반적인 고속 움직임 벡터 추정 알고리즘이다. 3 단계 탐색 (Three-Step Search: TSS) 방법은 한 개의 매크로 블록에 대해 총 25 개의 탐색점에 대해 SAD(Sum of Absolute Difference) 값을 구하여 그 중에서 가장 작은 값을 가지는 방향을 움직임 벡터로 정한다.

2. TSS 방법

TSS 방법은 3 단계로 나누어 움직임 벡터를 탐색한다.

1 단계 : 현재 매크로 블록 (MB)과 탐색 영역에서 8 방향으로 최 외곽에 위치한 MB 등 9 개의 탐색점에 대해서 최소 SAD 값을 갖는 탐색점을 구한다.
2 단계 : 스텝 크기를 반으로 줄인 다음 1 단계에서 구한 탐색점을 중심으로 8 개의 탐색점에 대해서 최소 SAD 값을 갖는 탐색점을 구한다.
3 단계 : 스텝 크기를 반으로 줄인 다음 2 단계에서 구한 탐색점을 중심으로 8 개의 탐색점에 대해서 최소 SAD 값을 갖는 탐색점을 구한다.

1 단계부터 3 단계를 통하여 각 단계마다 최소의 SAD 값을 갖는 탐색점을 구하고 마지막 단계에서 구한 탐색점을 움직임 벡터로 정한다.

3. 주변 움직임 벡터의 이용

H.263+와 MPEG4 코덱에서는 움직임 벡터를 인코딩하는 경우 주변 움직임 벡터의 중간값을 사용하여 그 차를 인코딩한다. 이것은 현재 MB 에 대하여 주변 움직임 벡터가 미치는 영향을 반영한 것이다. 이러한 예측된 움직임 벡터 (Predicted Motion Vector: PMV)를 움직임 벡터 추정에 이용하는 알고리즘에는 Nearest Neighbor 방법이 있다[2]. Nearest Neighbor 방법은 속도와 PSNR 측면에서 모두 우수한 성능을 나타낸다[2].

본 논문에서는 PMV 를 3 단계 탐색 방법에 적용하여 보다 정확한 MV 를 구함으로써 비트율을 줄이고자 한다 3 단계 탐색 방법은 주변 움직임 벡터에 상관없이 항상 고정된 위치의 25 개의 탐색점에 대해서 탐색을 수행한다. 1 단계에서의 9 개의 탐색점에 PMV 를 새로운 탐색점으로 추가한다. 표 1 은 첫번째 단계에서 PMV 가 선택된 평균횟수를 CIF 시퀀스 90 프레임 (frame)에 대해 측정한 결과이다. 백분율 %는 평균횟수를 프레임 당 MB 개수로 나눈 백분율이다.

움직임이 적은 시퀀스에서는 PMV 가 (0,0)으로 대부분 선택될 것이므로 TSS 의 첫단계에서 중앙 탐색점인 (0,0)이 PMV 와 중복될 것이다. 이러한 경우 TSS 와 동일해진다. 그러므로 표 1 에서 PMV 가 9 개의 탐색점과 중복이 되는 경우는 제외하였다. 첫 단계에서 10 개의 탐색점이 후보로 선택되는 경우 중에 PMV 가 첫 단계에서 가장 작은 SAD 값을 가지는 경우가 전체 검색에서 차지하는 비중을 표 1 에 정리하였다. 움직임이 많은 시퀀스일수록 제안한 방법의 효과가 커서 많은 성능 향상을 얻을 수 있다는 것을 보여준다.

그림 1 은 참조할 주위의 매크로 블록을 나타내고 있다. PMV 는 MB0, MB1, MB2 의 중간값이며 MB0, MB1,

MB2 가 움직임 벡터를 가지지 않는 경우에 대해서는 H.263+ 방법을 따른다.

표 1. 첫번째 단계에서 PMV 가 선택된 횟수

구분	평균횟수	%
Stefan	129	32.5
Coastguard	165	41.6
Foreman	82	20.6
Container	0	0.0
Akiyo	3	0.7

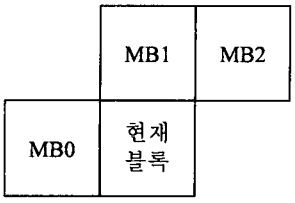

그림 1. 참조할 주위 블록

4. 수정된 3 단계 탐색 방법

그림 2 는 ±7 의 탐색영역에 대해서 수정된 TSS 방법의 1 단계에서의 탐색점들을 나타내고 있다. 기존의 9 개의 탐색점에 PMV 를 새로운 탐색점으로 추가한다. 탐색 순서는 중앙에서 시작하여 PMV 를 탐색하고 난후 왼쪽 상단 탐색점에서부터 시계방향으로 탐색한다.

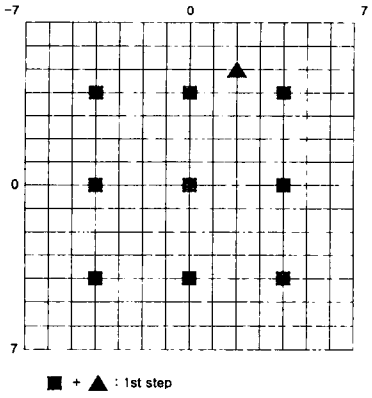

그림 2. 1 단계에서의 탐색점들

SAD 값을 계산할 때에는 Partial Distortion Elimination (PDE) [3] 방법을 사용하여 SAD 계산 과정에서 필요 없는 계산을 줄이는 효과를 볼 수 있어 속도 향상을 가져올 수 있다. 이 경우에 초반의 탐색점에서 작은 SAD 값을 가질수록 더욱 효과가 있기 때문에 중앙점을 탐색한 후 바로 PMV 를 탐색하도록 한다.

5. 실험 결과

H.263+ 코덱의 움직임 예측 모듈을 FS, TSS, 본 고에서 제안한 수정된 TSS 에 대하여 각 방법들의 성능을 비교한다. 제안한 수정된 TSS 를 TSS+P 로 표기한다. 사용한 비디오 시퀀스는 H.263 과 MPEG4 에서 성능 검사를 위해서 가장 보편적으로 사용하는 스테판 (Stefan), 아키오 (Akiyo), 코스트가드 (Coastguard), 컨테이너 쉽 (Container Ship), 포맨(Foreman) 비디오 시퀀스 등이다. 비디오 시퀀스의 시간적인 화질과 공간적인 화질을 살펴보기 위해서 비트율을 고정한 경우와 프레임율을 고정한 경우로 분류하여 실험한다. 각 비디오 시퀀스에 대하여 CIF 크기를 사용하고 양자화 매개 변수 QP 는 13 을 사용한다. 3 가지 방법 모두 SAD 값을 계산할 때에는 PDE 방법을 사용한다.

먼저 프레임율을 고정한 실험에서는 프레임율을 30fps 로 고정하고, 목표 비트율을 정하지 않고 인코딩한다. 따라서 각 방법의 성능에 따른 결과는 비트율에 따라 판단할 수 있다. 표 2-6 은 각 비디오 시퀀스에 대하여 프레임율을 고정한 경우에 대한 성능 평가 결과이다.

표 2. 스테판 (CIF 크기 90 프레임, 30 fps)

구분	FS	TSS	TSS+P
비트율*1	761.29	1022.06	870.11
TIME*2	262.04	32.94	31.26
PSNR(Y) *3	29.78	29.68	29.67

주) *1 비트율 단위: kbps, *2 TIME: 움직임 모듈 시간 (ms/frame), *3 PSNR 단위: dB

표 3. 코스트가드 (CIF 크기 90 프레임, 30 fps)

구분	FS	TSS	TSS+P
비트율	531.78	590.19	556.11
TIME	324.44	32.27	32.86
PSNR(Y)	29.86	29.82	29.83

표 4. 포맨 (CIF 크기 90 프레임, 30 fps)

구분	FS	TSS	TSS+P
비트율	255.64	303.00	292.60
TIME	277.17	31.27	31.92
PSNR(Y)	32.46	32.09	32.12

표 5. 콘테이너 쉽 (CIF 크기 90 프레임, 30 fps)

구분	FS	TSS	TSS+P
비트율	131.87	131.01	131.01
TIME	330.70	31.81	31.68
PSNR(Y)	31.85	31.82	31.82

표 6. 아키오 (CIF 크기 90 프레임, 30 fps)

구분	FS	TSS	TSS+P
비트율	65.48	66.07	65.70
TIME	211.51	29.49	29.71
PSNR(Y)	35.71	35.67	35.66

비트율면에서 TSS+P 는 기존 TSS 방법에 비해 많게는 15%의 비트율 감소를 나타냈다. 움직임이 많은 비디오 시퀀스일수록 더욱 효과가 좋았다. PSNR 면에서는 비트량 증감의 영향으로 인해 모든 방법들이 유사한 PSNR 수치를 나타냈다. 속도 측면에서는 TSS+P 방법이 TSS 방법과 유사한 속도를 나타냈다. 그림 3 은 스테판 CIF 비디오 시퀀스에 대해 30fps 로 인코딩한 결과에 대한 각 프레임별 비트량을 나타낸 것이다.

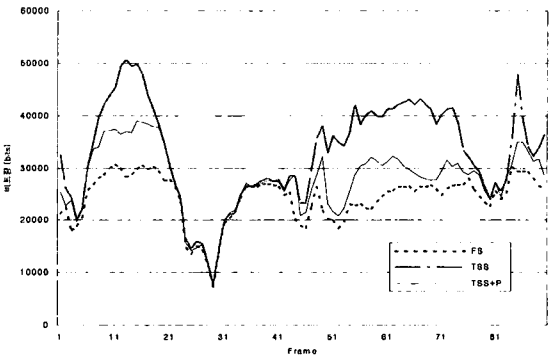

그림 3. 인코딩된 CIF 크기 스테판의 비트/프레임 비교

비트율을 고정한 실험에서는 목표 비트율을 시퀀스에 따라 300kbps, 100kbps, 48kbps 로 한다. 각 방법의 성능은 프레임율 (FRAME), 인코더의 움직임 예측 모듈에서 소요하는 시간 (TIME), PSNR 등을 측정하여 비교한다. 표 7-11 은 실험 시퀀스들에 대하여 비트율을 고정하여 인코딩한 경우에 대한 성능 평가이다.

TSS+P 방법은 TSS 방법에 비해 스테판 시퀀스의 경우 프레임율면에서 2.22 fps 증가했고 PSNR 은 0.2dB 증가했다. 즉, 시간적인 화질과 공간적 화질이 모두 증가하였다. 또한, 프레임율을 고정한 실험과 마찬가지로 움직임이 많은 시퀀스에서 더욱 효과가 좋았다.

표 7. 스테판 (CIF 크기 90 프레임, 300 kbps)

구분	FS	TSS	TSS+P
TIME	294.27	32.19	31.62
FRAME*4	25.71	22.83	25.05
PSNR(Y)	25.67	25.00	25.20

주) *4 FRAME: 프레임율(fps)

표 8. 코스트가드 (CIF 크기 90 프레임, 300 kbps)

구분	FS	TSS	TSS+P
TIME	353.58	33.25	34.02
FRAME	28.02	27.69	28.02
PSNR(Y)	28.00	27.66	27.81

표 9. 포맨 (CIF 크기 90 프레임, 300 kbps)

구분	FS	TSS	TSS+P
TIME	269.79	32.03	31.58
FRAME	28.35	28.35	28.35
PSNR(Y)	33.08	32.02	32.19

표 10. 콘테이너 쉽 (CIF 크기 90 프레임, 100 kbps)

구분	FS	TSS	TSS+P
TIME	330.90	31.74	31.87
FRAME	23.41	23.08	23.41
PSNR(Y)	31.16	31.03	31.09

표 11. 아키오 (CIF 크기 90 프레임, 48 kbps)

구분	FS	TSS	TSS+P
TIME	214.78	29.26	29.40
FRAME	21.43	21.43	21.43
PSNR(Y)	34.97	34.85	34.86

6. 결 론

본 논문에서는 PMV 를 TSS 방법에 적용한 TSS+P 방법을 제안하였다. TSS+P 방법은 기존의 TSS 방법과 유사한 속도를 유지하면서 시퀀스에 따라 다르지만, 최고 15% 정도의 비트량을 감소시킬 수 있었다. 그러므로 TSS+P 방법은 기존의 TSS 방법에 비해 이동 통신이나 인터넷 등과 같은 저대역폭 통신 환경에서 비디오 데이터 전송에 유용할 것이다.

감사의 글

본 논문은 2001 년도 광운대학교 연구년에 의하여 연구되었습니다.

참고 문헌

[1] B.Firtht, J.Greeberg and R.Westwater, *Motion Estimation Algorithms for Video Compression*, Florida Atlantic Univ.

[2] M. Gallant, G. Côté, and F. Kossentini, "An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding," IEEE Tr. on Image Processing , vol 8 , no 12, December, 1999.

[3] J.N. Kim and T.S. Choi, "Computational reduction using UESA, adaptive partial sum form gradient magnitude for fast motion estimation", in Proc. PCS, pp.107-111, 1999.