고속 움직임 예측을 위한 움직임 적응적 손실성 엄격 다단계 연속 제거 알고리즘

*이경준 **Ng Teck Sheng ***유종상 ****정제창

한양대학교 전자통신컴퓨터공학부

*kjlee888@naver.com **justsheng@yahoo.com ***kar_pondier@hotmail.com

****jjeong@ece.hanyang.ac.kr

Motion Adaptive Lossy Strict Multi-level Successive Elimination Algorithm for Fast Motion Estimation

*Lee, Kyung-Jun **Ng, Teck Sheng ***Yoo, Jong-Sang ****Jeong, Je-Chang

Electronic and Computer Eng. Dept., Hanyang University

요약

본 논문에서는 고속 움직임 예측(Fast Motion Estimation)방법의 일종인 다단계 연속 제거 알고리즘(MSEA : Multi-level Successive Elimination Algorithm)에 움직임의 역동성 정도를 고려하여 적응적인 가중치를 적용하는 방안에 대해 제안하였다. 움직임을 예측하는 과정에서 영상의 화질 손상이 발생하는 방식(Lossy Motion Estimation Algorithm)에서 모든 단위 블록(Macro Block)에 고정된 가중치만을 적용하는 기존의 방식과 달리 주위 블록의 움직임 벡터(Motion Vector)를 통해 움직임의 정도를 가정하여 적응적인 가중치를 적용함으로써 화질 손상을 줄이는 것이 목적이다. 제안하는 알고리즘으로 설계한 실험으로부터 MSEA에 적응적 가중치를 사용할 경우의 효율성을 확인하였다.

1. 서론

연속적인 비디오 프레임에는 현재 프레임과 이전 프레임 간에 시간적 중복성(temporal redundancy)이 존재한다. 비디오 압축의 관점에서 이러한 프레임간의 시간적 중복성을 제거하기 위해 움직임 추정(motion estimation)과 움직임 보상(motion compensation) 기법을 사용한다. 움직임 추정의 방법으로는 하드웨어적 설계가 용이한 이유로 블록 정합 알고리즘(BMA : Block Matching Algorithm)이 여러 비디오 코딩 표준으로서 사용되고 있다.

블록 정합 알고리즘의 하나로 사용되는 전역 탐색 알고리즘(full search algorithm)은 고정된 탐색 영역(search range) 내의 모든 위치에서 SAD(Sum of Absolute Difference)값을 비교하여 최소 SAD값을 가지는 위치를 최적 정합 블록의 움직임 벡터(MV : Motion Vector)로 정한다. 그러나 이러한 전역 탐색 알고리즘은 매우 많은 연산량으로 인하여 실제적으로 사용되지는 않는다.

이러한 문제점으로 인해 전역 탐색 알고리즘의 PSNR을 유지하면서 연산량을 줄이는 고속 움직임 추정 방법으로 연속 제거 알고리즘(SEA : Successive Elimination Algorithm)이 제안되었다. 연속 제거 알고리즘은 탐색 영역 내의 후보 블록들의 SAD를 계산하기 전에 미리 계산되어 있는 평균 성분(sum norm)의 값을 이용하여 후보의 개수를 줄임으로써 연산량을 줄이게 된다. 다단계 연속 제거 알고리즘(MSEA : Multi-level Successive Elimination Algorithm)은 연속 제거 알고리즘에서 탐색지점에 대한 조건을 다단계로 나누어 비교하게 된다. 단계가 내려갈수록 강화된 조건을 이용하기 때문에 탐색지점의 수를 연속 제거 알고리즘보다 더 줄일 수 있게 된다.

손실성 엄격 다단계 연속 제거 알고리즘(Lossy Strict MSEA)은 다단계 연속 제거 알고리즘의 각 레벨에 가중치를 부여하는 방법이다. 이 알고리즘은 PSNR의 손해를 보더라도 최소 SAD 값을 가지지 못하는 블록이라 예상되는 경우를 과감하게 생략하고 다음 탐색 위치로 넘어감으로써 연산량을 줄이게 된다.

본 논문에서는 기존의 손실성 엄격 다단계 연속 제거 알고리즘에서 각 단계마다 고정되어 있는 가중치 값을 블록의 움직임정도를 가정하여 적응적으로 바꾸어 적용한다. 이를 통해 영상의 움직임 정도에 따라 PSNR과 움직임 벡터의 탐색 속도를 적절히 조정하게 된다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 탐색 알고리즘에 대하여 간략히 소개하고 3장에서는 제안하는 알고리즘을 설명한다. 4장에서는 제안하는 알고리즘에 기반 한 실험결과를 분석하고 5장에서는 결론을 맺는다.

2. 기존 알고리즘

가. 연속 제거 알고리즘(SEA)

$N \times N$크기의 블록에 대하여 두 블록간의 SAD는 다음과 같다.

$$SAD(x,y) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} |B_c(m,n) - B_p(m+x, n+y)| \quad (1)$$

여기서 $B_c(m,n)$과 $B_p(m,n)$은 각각 현재 프레임과 이전 프레임에서 SAD를 계산하고자 하는 블록을 나타낸다. (x,y)는 탐색 영역 내에서의 블록의 위치를 의미한다. 움직임 벡터(v_x,v_y)는 최소 $SAD(x,y)$ 값을 가지는 블록의 (x,y) 값이다.

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

현재 블록과 이전 블록의 평균 성분(sum norm)은 다음과 같이 정의한다.

$$R=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}|B_c(m,n)| \quad (3)$$

$$M(x,y)=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}|B_p(m+x,n+y)| \quad (4)$$

R과 $M(x,y)$ 사이에는 다음과 같은 부등식이 성립한다.[1]

$$|R-M(x,y)|\le SAD(x,y) \quad (5)$$

탐색 영역 내에서 구해진 최소 SAD를 $SAD_{\min}$이라 할 때, 식 (5)을 통해 최적 정합 블록의 후보를 제거해 나가면서 연산량을 감소시킨다.

나. 다단계 연속 제거 알고리즘(MSEA)

다단계 연속 제거 알고리즘은 기존의 연속 제거 알고리즘을 단계별로 나누어 더 많은 조건을 통하여 후보를 제거하여 최적의 SAD를 구하는 효율을 높였다. 기존의 SEA 과정을 MSEA의 0-level이라고 할 때, l-level에서는 먼저 $N\times N$ 크기의 블록을 $N/2^l\times N/2^l$ 크기의 네 개의 서브 블록으로 나눈다. $N_l=N/2^l$이라 할 때 level은 서브 블록이 모두 2×2가 될 때까지 반복 된다. l-level에서의 현재 블록과 이전 블록의 평균 성분과 SAD는 다음과 같이 나타내어진다.

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

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

$$SAD_l^{(u,v)}=(x,y)=\sum_{m=0}^{N_l-1}\sum_{n=0}^{N_l-1}|B_c(m+uN_l,n+vN_l)-B_p(m+uN_l,n+vN_l)| \quad (8)$$

(u,v)는 서브 블록에서의 픽셀 좌표이다. 식 (6)을 통해 서브 블록의 평균 성분과 SAD 사이에도 다음을 만족함을 알 수 있다.

$$|R_l^{(u,v)}-M_l^{(u,v)}(x,y)|\le SAD_l^{(u,v)}(x,y) \quad (9)$$

만약 서브블록의 SAD를 SSAD(sub-block SAD)로 두고 다음과 같이 정의 한다면

$$SSAD_l=\sum_{u=0}^{2^l-1}\sum_{v=0}^{2^l-1}|R_l^{(u,v)}-M_l^{(u,v)}(x,y)| \quad (10)$$

각 level의 SSAD 사이에는 다음과 같은 부등식이 성립한다.

$$SSAD_l\le SSAD_{l+1}\le\cdots\le SAD(x,y) \quad (11)$$

식 (12)를 통해 레벨을 증가시키며 $SSAD_l$과 SAD를 비교함으로써 SEA를 수행할 때 보다 더 많은 후보 블록들을 제거할 수 있다. 하지만 제거될 가능성이 있는 후보들에 대하여 여러 레벨에 거쳐 SAD와 비교하는 연산을 수행하기 때문에 연산량이 증가하는 단점이 있다.

다. 손실성 엄격 다단계 연속 제거 알고리즘(Lossy strict MSEA)

연속 제거 알고리즘과 다단계 연속 제거 알고리즘은 전역 탐색 알고리즘에 비해 연산량만 줄일 뿐 PSNR에 영향을 주지 않는 무손실(lossless) 제거 방법이다. 손실성 제거 방법에서는 다단계 연속 제거 알고리즘의 각 레벨에 적절한 가중치를 부여함으로써 손실을 감안하고서라도 조건에 만족하지 않는 후보 블록을 과감히 생략하여 연산량을 줄인다. MSEA의 SSAD에 가중치를 부여한 값을 SMSEA라 하면 l-level에서의 SMSEA는 다음과 같다.

$$SMSEA_l=MSEA_l\times\lambda_l\ (\lambda\ge 1) \quad (12)$$

λ의 값이 작을수록 후보를 잘 못 제거할 가능성이 줄어들며, λ의 값이 커질수록 연산량이 줄어들게 되므로 적절한 λ를 선정해야 한다. λ_l이 모두 1일 경우는 SEA와 같다.

3. 제안하는 알고리즘

제안하는 알고리즘에서는 한 프레임 내의 공간적 중복성과 연속한 프레임간의 시간적 중복성을 이용한 Lossy strict MSEA를 사용한다.

먼저 각 블록에 적응적으로 가중치를 적용하기 위해서는 블록의 움직임 정도를 구분해야 한다. 움직임 정도는 크게 static과 dynamic으로 나누어 각 블록의 움직임 벡터를 구함과 동시에 판별한다.

$$B_c(m,n)=\begin{cases}static & ,(v_x^2+v_y^2)<th_{cond}\\ dynamic & ,otherwise\end{cases} \quad (13)$$

한 프레임 내에서는 이웃하는 블록끼리는 움직임 정도가 비슷하다는 공간적 중복성을 이용한다. <그림 1>과 같이 블록 v_1부터 v_{12}까지 12개의 블록 중에서 식 (13)을 이용하여 static으로 판별된 블록의 개수를 기준으로 현재 블록의 움직임 정도를 추측한다. 현재 블록의 움직임 정도는 크게 static, middle, dynamic 세 가지로 구분하였다.

	FS		SEA		MSEA		Lossy		Proposed	
	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)
Akiyo	42.34	491.03	42.34	22.65	42.34	10.04	42.03	9.68	41.70	9.58
Foreman	31.81	496.52	31.81	91.27	31.81	30.19	30.91	11.51	30.99	12.23
Football	27.08	144.39	27.08	75.82	27.08	20.10	25.23	4.26	26.06	5.14

<표 1 > 기존 알고리즘과 제안하는 알고리즘의 성능 비교

v_1	v_2	v_3	v_4	v_5
v_6	v_7	v_8	v_9	v_{10}
v_{11}	v_{12}	v		

<그림 1> 현재 블록이 참고하는 이웃 블록 후보

C_s는 이웃하는 블록 중 static블록의 개수이며, th_s, th_d는 각각 static과 dynamic을 구분하기 위한 임계값이다.

$$B_c = \begin{cases} static & , C_s > th_s \\ middle & , th_d < C_s < th_s \\ dynamic & , otherwise \end{cases} \quad (14)$$

식 (14)를 이용하여 현재 블록의 움직임 정도를 추측하면 각각의 움직임 정도에 맞게 설정된 가중치 λ_l을 적용한다.

기존의 알고리즘에서는 움직임 벡터를 구하고자 할 때 현재 블록에서 $(x,y)=(0,0)$에서의 SAD를 시작 기준으로 설정한다. 하지만 제안하는 알고리즘에서는 전체적인 시간을 단축시키기 위한 방법으로는 연속하는 프레임간의 시간적 중복성을 이용한다. 연속하는 프레임에서 같은 위치의 블록 사이에는 매우 유사한 움직임 벡터 값을 갖는 점을 이용하여 $(x,y)=B_p(v_x,v_y)$일 때의 SAD값을 시작 기준으로 설정한다.

4. 실험결과

실험은 CIF 영상(352×288)을 이용하여 진행하였다. 블록 사이즈 N은 16이었으며 level은 3-level까지 수행하였다. <표 1>은 기존의 알고리즘을 사용하였을 때의 PSNR과 수행시간을 비교해 놓은 것이다. lossy 알고리즘은 각 단계별로 가중치 값을 λ_0=4.0, λ_1=2.5, λ_2=2.0, λ_3=1.5 로 설정하였다.[3] 제안하는 알고리즘에서는 static, middle, dynamic으로 구분한 각각의 블록에 실험적으로 얻은 가중치를 적용하였다. Akiyo와 같은 정적인 영상의 경우는 기존의 lossy 알고리즘보다 PSNR이 감소하였으나 탐색 시간이 감소하였다. 반면에 비교적 동적인 영상의 경우에는 움직임 벡터의 탐색 시간은 증가하였으나 PSNR이 증가함에 따라 더 정확한 움직임 벡터를 탐색한다는 것을 확인 할 수 있었다. Akiyo 영상과 같은 정적인 영상의 경우 PSNR은 감소하였지만 영상을 분석해 본 결과 사람이 인지하기에 매우 적은 차이가 있었다. 반면에 Foreman, Football과 같은 동적인 영상에서는 제안하는 알고리즘을 통해 눈에 띄게 화질이 개선되는 것을 확인할 수 있었다. 다음의 <그림 2>는 기존의 lossy 알고리즘과 논문에서 제안하는 알고리즘을 통해 얻은 football 영상을 비교한 것이다. 그림 2(a)는 loosy 알고리즘, 2(b)는 제안하는 알고리즘을 이용하여 보간한 football 영상이며 영상에 표시한 부분과 같이 기존의 알고리즘보다 제안하는 알고리즘이 더 좋은 성능을 나타냄을 확인 할 수 있었다.

(a)Lossy 알고리즘

(b)제안하는 알고리즘

<그림 2> 기존의 알고리즘과 제안하는 알고리즘을 이용하여 비교한 영상

5. 결론

제안하는 알고리즘을 통하여 실험한 결과 모든 영상에서 PSNR과 탐색 속도가 좋아지는 것은 아니었다. 하지만 실험결과에서 언급하였듯이 PSNR이 감소한 경우는 대부분의 움직임 벡터가 작은 정적인 영상이었으며 사람이 눈으로 그 차이를 쉽게 구분하지 못하는 정도였다. 반대로 동적인 영상에서는 PSNR이 증가함과 동시에 움직임 벡터의

탐색 시간도 같이 증가하였지만 기존에 걸리던 시간과 차이가 많이 나지 않았다. 이러한 결과를 통해 본 논문에서 제안하는 알고리즘을 영상에 적용할 경우 PSNR과 탐색 시간 사이에 약간의 손익을 감소하더라도 영상의 움직임 정도에 따라 적응적으로 대응하여 움직임 벡터를 탐색할 수 있다.

6. 감사의 글

"본 연구는 지식경제부 및 산업전략기술개발사업의 연구결과로 수행 되었음, 하드웨어 기반 변복조기 결합형 실시간 멀티 코덱 트랜스코더)

7. 참고문헌

1. Li. W., and Salari, E.: 'Successive Elimination Algorithm for Motion Estimation', *IEEE Trans. Image Process.*, 1995, **4**, (1), pp. 105-107
2. Gao, X.Q., Duanmu, C.J., and Zou, C.R.: 'A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation', *IEEE Trans.* Image Process., 2000, **9**, (3), pp. 501-504
3. Y. Song, Z. Liu, T. Ikenaga, S. Goto.: 'Lossy Strict Multilevel Successive Elimination Algorithm for Fast Motion Estimation', *IEEE Int. Symp. Intelligent Signal Processing and Comm. Systems*(ISPACS2006)., Tottori, Japan. pp. 431-434, Dec. 2006.