SAD 함수 모델링을 이용한 고속 반화소 추정 알고리즘

이윤구, 최부림, 나종범
한국과학기술원 전자전산학과

Fast Half-Pixel Motion Estimation Based on SAD Curve Modeling

Yun-Gu Lee, Boorym Choi, and Jong Beom Ra
Image System Lab., Dept. of EECS, KAIST
E-mail: jbra@ee.kaist.ac.kr

Abstract

많은 고속 정화소 움직임 추정 알고리즘이 개발됨에 따라, 최근에는 10 개의 탐색점만으로 정화소 움직임 벡터를 찾을 수 있게 되었다. 반면에, 반화소 움직임 추정에서는 정화소 움직임 벡터주변의 8 개의 반화소 탐색점을 검색해야 한다. 그러므로 반화소 움직임 추정 알고리즘의 계산량을 줄일 필요성이 생기게 되었다. 본 논문에서는 directional search 와 SAD 함수의 선형 모델링을 이용한 고속 반화소 움직임 추정 알고리즘을 제안한다. 제안된 알고리즘은 PSNR 열화 없이 2.21 개의 탐색만으로도 움직임 벡터를 찾을 수 있게 해준다. 특히, 조절 가능한 파라미터를 이용하면, 약간의 PSNR 감소와 함께 0.34 개의 탐색만으로 움직임 벡터를 추정을 할 수 있게 해준다.

I. 서론

비디오 코딩에서 움직임 추정은 블록 매칭 알고리즘 (BMA)을 이용하여 수행된다. BMA 에서 부호화될 현재장은 겹치지 않는 블록으로 나뉘어 진 뒤, 각각의 현재 블록과 가장 비슷한 블록을 이전 장에서 찾게 된다. 전역탐색기법 (full search block matching algorithm)에서는 주어진 탐색영역안의 가능한 모든 후보 블록을 현재 블록과 비교한다. 아래 식은 여기에 비교 척도로 사용되는 SAD (sum of absolute differences)이다.

$$SAD(u,v)=\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\left|I_t(i,j)-I_{t-1}(i+u,j+v)\right| \qquad (1)$$

여기서 $I_t(j, j)$와 $I_{t-1}(i, j)$는 각각 시간 t, t+1 에서의 luminance 화소 값이다. (u, v)는 움직임을 의미하며 N 은 블록의 크기이다.

실제의 움직임은 화소 단위로 이루어지지 않는다. 그러므로 정화소 단위의 움직임 벡터는 정확도가 떨어진다. 따라서 움직임 벡터의 정확도를 높이기 위하여 반화소 단위로 움직임 추정을 하게 된다. H.263, MPEG-1/2/4 와 같은 국제 표준에서는 반화소 단위로 하는 움직임 추정을 채택하고 있다 [1-4]. 반화소 정확도로 움직임 추정은 일반적으로 두단계의 과정을 거쳐 이루어진다. 첫번째 단계에서 주어진 탐색영역에서 정화소 단위로 움직임 추정을 하여 정화소 움직임 벡터를 얻는다. 두번째 단계에서는 정화소 움직임벡터 주변의 8 개의 반화소 탐색점을 탐색하여 최종 움직임 벡터를 얻는다. 정화소 움직임 추정의 탐색영역은 반화소 움직임 추정보다 몇 십 배에서 몇 백배 큰 계산량을 가지고 있었기 때문에 대부분의 연구는 고속 정화소 움직임 추정에 관해 이루어졌다. 그 결과 고속 정화소 움직임 추정알고리즘은 10 개 이하의 탐색점의 탐색으로도 수행 가능하게 되었다. 따라서 8 개의 반화소를 탐색하는 반화소 움직임 추정알고리즘의 계산량이 상대적으로 커지게 되었고 고속 반화소 움직임 추정 알고리즘이 필요하게 되었

다 [5-7].

본 논문에서는 SAD 함수의 선형 모델링과 directional search 를 이용한 움직임 추정 알고리즘을 제안한다. 제안된 알고리즘에서 사용하는 조절 가능한 파라미터를 이용하면 적은 PSNR 감소와 탐색점 사이의 관계를 조정하여 원하는 속도와 성능을 얻을 수 있게 된다. 2 장에서는 기존의 방법에 대하여 간단히 살펴보고 3 장에서는 제안된 알고리즘을 살표본다. 실험결과는 4 장에 기술되어 있으며, 5 장에서는 결론을 맺는다.

II. 기존의 방법

고속 반화소 움직임 추정 알고리즘 중의 하나인 HVDR [8]은 정화소 움직임 벡터 주변의 좌우, 위아래의 네 개의 반화소 탐색점을 탐색한 뒤, 그 결과에 따라 한 개의 대각선 탐색점을 추가로 탐색하는 방법이다. 이 방법의 경우에는 항상 5 개의 탐색점을 탐색한다. PPHPS [9]의 경우에는 주변의 정화소 탐색점의 SAD 값을 이용하여 반화소 위치의 SAD 값들을 예측을 한 뒤, 이 값들을 이용하여 반화소 움직임 추정의 속도를 향상시킨다. 예측된 SAD 값중 가장 작은 값을 갖는 위치를 찾은 뒤, 그 주변의 반화소 탐색점을 포함하여 세개의 반화소 탐색점의 SAD 값을 실제로 구한다. 이 방법의 경우에는 항상 세 개의 SAD 값을 탐색하기 때문에 위의 HVDR 방법보다 속도가 빠르다. 그러나 SAD 함수를 예측하기 위한 추가적인 계산량이 필요하다.

III. 제안한 알고리즘

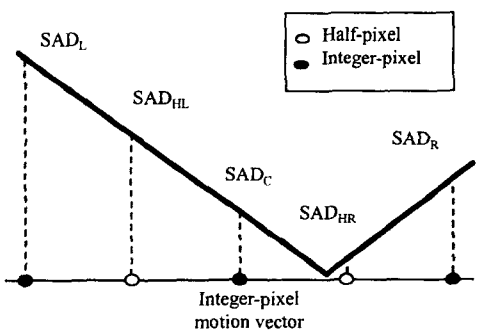

그림 1. SAD 값의 모델링

SAD 함수 모델링 통해 SAD 값을 예측하여 반화소 움직임 추정 알고리즘의 계산량을 줄인다. 그러나 보통 예측된 SAD 값은 실제 SAD 값과 다르기 때문에 제안된 알고리즘은 제안된 기준에 따라 예측된 SAD 값의 정확도를 판별하여 실제 SAD 값이 필요한 경우에만 계산해 본다. 따라서 대부분 SAD 의 실제 탐색은 이루어지지 않게 되어 계산량은 줄게 되며 PSNR 성능은 크게 줄어들지 않는다.

3.1 선형 모델링

정화소 움직임 벡터 주변의 반화소 탐색점의 SAD 값을 예측하기 위해서는 정화소 움직임 벡터 주변의 정화소 탐색점의 SAD 값들이 필요하다. 최근에 대부분의 정화소 움직임 추정 알고리즘은 gradient descent search 방식을 채택하고 있기 때문에 정화소 움직임 벡터의 좌우, 위아래의 SAD 값들은 얻을 수 있다.

PPHPS 는 2D 로 SAD 함수를 모델링을 수행하는 반면에 제안된 알고리즘은 1D 로 SAD 함수를 모델링 한다. 또한 1D 의 모델링은 직선으로 모델링을 수행한다. 이미 움직임 벡터가 찾아진 극소 구간에서 SAD 함수는 직선으로도 잘 모델링 되어지며 정확도가 크게 떨어지지 않는다. 특히, 직선으로 모델링을 할 경우, 간단하게 덧셈과 뺄셈만으로도 연산이 가능하다.

그림 1 은 수평방향으로 SAD 함수를 모델링 한 그림이다. 그림 1 에서 x 축은 탐색점 위치를 나타내며 중심점은 정화소 움직임 벡터의 위치이다. 그리고 y 축은 SAD 값을 나타낸다. SAD 함수는 두 개의 직선의 방정식으로 모델링된다. 두 개의 직선의 방정식은 서로 다른 부호를 가지며 절대값은 같은 값을 갖는다. 그림 1 의 SAD_C 를 정화소 움직임 벡터의 SAD 값이라고 하자. SAD_L 과 SAD_R 은 각각 정화소 움직임 벡터의 왼쪽과 오른쪽 정화소 탐색점의 SAD 값이라고 하자. SAD_L 과 SAD_R 의 값은 정화소 움직임 추정으로부터 이미 얻어진 값이다. 이때, $(-1, SAD_L)$과 $(1,SAD_R)$ 두 점은 두 개의 직선의 방정식 위에 각각 위치 하며 $(0, SAD_C)$는 두 개의 직선 중 하나 위에 존재한다. (SAD_L 과 SAD_R 이 같은 값일 경우에는 두 직선의 교점 위에 존재 한다.) 위의 조건으로 두 개의 직선의 방정식은 계산할 수 있으며 직선으로부터 두 개의 반화소 위치의 SAD_{HL} 과 SAD_{HR} 의 값도 계산할 수 있다.

3.2 에러 허용치

각 방향의 움직임 벡터의 값을 찾기 위하여 예측된 반화소 SAD 값은 정화소 움직임 벡터 위치의 SAD 값

과 비교된다. 그러나 예측된 SAD 값은 항상 실제 값과 같지 않기 때문에 비교 결과가 항상 정확하지는 않다. SAD_E 를 예측된 SAD 값이라고 가정했을 때, SAD 값은 SAD_E-e 와 SAD_E+e 사이에 있다고 가정한다. 여기서 e 는 에러허용치이다.

이제, 두 개의 예측된 반화소 탐색점의 SAD 값중에서 작은 값을 SAD_H 라 하자. 만약 (SAD_C-SAD_H)가 e 보다 크면 예측된 SAD_H 는 최소값으로 간주된다. 또한 (SAD_H-SAD_C)가 e 보다 크면 SAD_C 는 최소값으로 간주된다. 그러나 |SAD_H-SAD_C|가 e 보다 작으면 실제 SAD_H 값은 계산된다. 수평방향으로도 같은 작업을 반복한다. 이 과정을 통해 각 방향의 움직임 벡터와 해당하는 SAD 값을 얻는다. 이때, 각 방향으로 최대 탐색은 한 개의 SAD 탐색뿐이다.

3.3 보정

위에서 기술한 봐와 같이 x 와 y 방향 각각의 움직임 벡터와 그에 해당하는 (예측된 혹은 실제) SAD 값이 얻어졌다. 이 결과에 따라 추가적으로 대각선 방향으로 하나의 탐색점의 탐색 여부를 판단하게 된다. 만약 두 개의 움직임 벡터 중 하나라도 0 인 값을 가지면 대각선 방향으로 탐색은 이루어지지 않는다. 그렇지 않은 경우에는 대각선 방향으로 한 개의 점을 추가적으로 더 탐색한다. 예를 들어, x 와 y 방향의 움직임 벡터가 각각 -0.5 와 -0.5 라고 하자. 이 경우에는 대각선 방향인 (-0.5, -0.5)에 해당하는 움직임 벡터가 추가적으로 탐색된다. 계산된 혹은 예측된 SAD 값 중에서 가장 작은 값을 갖는 위치가 최종 움직임 벡터가 된다. 그러나 만약 움직임 벡터가 각각 -0.5 와 0 이라면 최종 움직임 벡터는 (-0.5, 0)이 된다. 따라서 제안된 알고리즘은 최대 세개의 탐색점을 탐색하게 된다. 그러나 실제 계산량은 훨씬 적게 된다.

IV. 결과

실험에서는 두개의 CIF (352x288)영상과 네개의 SIF 영상이 사용되었으며 각 영상은 Carphone, Foreman, Flower garden, Football, Table tennis, Mobile 이 사용 되었다. 각 영상은 100 장으로 이루어져 있으며 frame rate 는 30Hz 이다. 코딩은 H.263 을 이용하여 수행하였다. 첫번째장을 제외하면 모두 P frame 으로 인코딩 되었다. 정화소 움직임 추정 알고리즘으로 방법 A 에서는 전역탐색알고리즘이 사용되었으며 방법 B,C,D,E,F,G 에서는 UCBDS [6]이 사용되었다. 반화소 움직임 추정 알고리즘으로 방법 A, B 에서는 전역탐색 알고리즘이, C 에서는 HVDR 이, D 에서는 PPHPS 가 사용되었다. 방법 E, F, G 에서는 제안된 방법이 사용되었으며, 각각 e 값은 0, 50, infinite 값이 사용되었다.

표 1 과 2 는 제안된 방법이 다른 알고리즘보다 빠르다는 것을 보여준다. 예측에러 허용치인 e 값을 무한대로 할 경우 모든 예측된 SAD 값은 불확실하게 되며 모든 실제 SAD 값이 계산된다. 따라서 이 경우에 전체 탐색 포인트는 2.21 포인트가 된다. 이 경우에는 제안된 알고리즘은 다른 고속 알고리즘보다 속도는 빠르며 PSNR 성능은 같다. e 를 0 으로 설정한 경우에는 모든 SAD 값은 정확하다고 판단되며 실제 SAD 값은 계산되지 않는다. 이 경우에 전체 탐색점은 0.34 가 된다. 그리고 반화소 전역탐색 기법과 비교해서 0.11dB 의 성능저하가 생긴다. 에러 허용치 e 값을 50 으로 할 경우에는 기존의 고속 반화소 움직임 추정 알고리즘인 PPHPS 보다 3.2 배나 빨라지며 PSNR 성능저하도 단지 0.05dB 이다. 제안된 알고리즘에서 SAD 값을 예측은 상당히 적기 때문에 무시할만한 값이다.

V. 결론

비디오 코딩에서 사용되는 반화소 움직임 추정 방법을 제안하였다. 제안된 알고리즘은 움직임 벡터를 x 방향과 y 방향으로 나누어 예측한다. 그리고 그 결과에 따라 대각선 방향으로의 탐색여부를 결정하게 된다. 제안된 알고리즘은 사용된 파라미터에 따라 PSNR 와 속도를 조절할 수 있다. 제안된 알고리즘은 기존의 고속 알고리즘에 비해 성능저하 없이 속도를 빠르게 할 수 있을 뿐만 아니라 약간의 PSNR 감소와 함께 속도를 크게 향상 시킬 수 있다.

참고문헌

[1] ISO/IEC JTC1/SC29/WG11,"ISO/IEC CD 11172:Information technology," MEPG I Committee Draft, Dec. 1991.

[2] ISO/IEC JTC1/SC29/WG11,"ISO/IEC CD

13818:Information technology," MEPG 2 Committee Draft, Dec. 1993.

[3] ISO/IEC JTC1/SC29/WG11, MPEG99/5477, "MPEG-4 Video Verification Model version 14.2," Dec. 1999.

[4] International Telecommunication Union, "Video codec for low bitrate communication," ITU-T Recommendation H.263, Mar. 1996.

[5] L. Liu and E. Feig, "A Block-based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," IEEE Trans. Circuts Syst. Video Technol., vol. 6, pp. 419-422, Aug. 1996.

[6] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 4, pp. 369-378, Aug. 1998.

[7] Shan Zhu and Kai-Kuang Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Trans. Image Processing, vol. 92, no. 2, pp. 287-290, Feb. 2000.

[8] K.H. Lee, J.H. Choi, B.K. Lee, and D.G. Kim "Fast Two-Step Half-Pixel Accuracy Motion Vector Prediction," Electronics Letters, vol. 36, pp. 625-627, Mar. 2000.

[9] C. Du and Y. He, "A Comparative Study of Motion Estimation for Low Bit Rate Video Coding," SPIE Proc. VCIP2000, vol. 4067, no. 3, pp. 1239-1249, Jun. 2000.

표 1. 제안된 알고리즘과 기존의 알고리즘과의 비교

	Carphone		Foreman		Table tennis		Flower garden		Mobile		Football		Average	
	PSNR (dB)	kbps	PSNR (dB)	kbps	PSNR (dB)	kbps	PSNR (dB)	kbps	PSNR (dB)	kbps	PSNR (dB)	kbps	PSNR (dB)	Search points
A	34.72	304	32.83	293	28.62	286	26.12	1059	24.35	1099	28.49	1056	29.19	8
B	34.38	303	32.74	307	28.6	301	26.10	1085	24.34	1117	28.12	1011	29.05	8
C	34.36	305	32.70	307	28.58	301	26.10	1089	24.34	1119	28.09	1019	29.03	5
D	34.37	304	32.72	307	28.59	302	26.09	1097	24.32	1128	28.09	1021	29.03	3
E	34.24	311	32.59	313	28.48	308	26.03	1120	24.30	1142	27.99	1049	28.94	0.34
F	34.30	308	32.66	310	28.54	307	26.05	1115	24.31	1140	28.04	1041	28.98	0.93
G	34.36	305	32.70	307	28.59	301	26.10	1089	24.34	1119	28.09	1019	29.03	2.21

표 2. 영상에 따른 제안된 알고리즘의 탐색점 수

	Carphone	Foreman	Table tennis	Flower garden	Mobile	Football
E	0.37	0.39	0.33	0.27	0.27	0.39
F	1.27	1.22	1.18	0.60	0.43	0.9
G	2.19	2.24	2.20	2.12	2.13	2.38