움직임 벡터 추정을 위한 동적 영역 결정 방식

이민구, 홍민철
숭실대학교 정보통신공학과

Dynamic Search Area Decision For Motion Vector Estimation

Min-Gu Lee and Min-Cheol Hong
School of Electronic Engineering, Soongsil University
Email: mhong@e.ssu.ac.kr

요약

본 논문에서는 동영상 압축 방식에서 압축 효율을 증가시키는 움직임 벡터 추정 방식의 개선된 방법에 대해 제안한다. 움직임 벡터 추정 방식의 하나인 BMA(block matching algorithm) 중에 가장 좋은 성능을 보이는 FSA(full search algorithm)은 움직임의 정도를 예측하지 않고 모든 블록에 대해 고정적인 탐색 영역이 결정되어 있다. 이는 계산량을 상당히 크게 하는 원인이 된다. 제안된 방식은 블록 사이의 움직임이 유사한 특성을 이용하여 블록 단위의 동적인 탐색 영역을 결정하여 계산량을 효율적으로 줄이면서 FSA와 유사한 특성을 유지하게 된다.

1. 서론

동영상 압축 표준화 기구인 ISO 의 MPEG-1,2,4와 ITU-T 의 H.261, H.263, H.26L 에서는 연속하는 프레임간의 시간상 중복하는 정보를 줄여 압축 효율을 증가시키는 움직임 벡터 추정 방식을 사용하고 있다. 움직임 벡터 추정 및 보상 방식은 압축 효율을 개선시키는 장점이 있으나 부호화 부의 계산량을 가중시켜 실시간 처리에 가장 큰 문제점으로 다루어지고 있다. 이로 인해, 움직임 벡터 추정을 위한 저 계산량 방식에 대한 연구가 필요하다.

움직임 벡터 추정에 대한 대부분의 발전된 방식들은 BMA(block matching algorithm)이라는 블록 단위로 움직임 벡터를 찾는 기법에 있었다. BMA 기법을 사용한 것 중에 하나인 FSA(full search algorithm)은 기존의 움직임 추정 방식 중에서 가장 좋은 성능을 나타내지만 고정적인 탐색 영역의 모든 지점에서 BDM(the difference of block distortion measure)을 계산하며 그 중에서 BDM 의 최소값을 찾아 움직임 벡터를 결정하기에 상당한 계산량을 갖는다. 더욱이 더 많은 블록으로 구성된 고해상도 프레임의 동영상 경우에 증가된 블록의 개수에 비례하여 위의 과정을 통한 계산량을 더해 주어야 한다. 그러므로 FSA 으로는 계산량의 부담으로 인해 현재 인터넷이나 무선환경에서 실시간으로 고화질의 동영상 신호 전송하기는 어렵다. 또한 HDTV 나 고해상도의 TV 에도 적용하기에도 부담이 된다.

이와 같은 FSA 의 계산량의 부담을 줄이기 위해 여러 방식이 제안되었다. 탐색 영역 내에서 일정한 패턴에 의해 지점 수를 줄이는 방식[1-2], 삼각부등식을 이용한 탐색 지점 소거 방식[3-4], BDM 을 계산하는 블록 안의 화소 수를 줄이는 방식[5-6], 탐색 영역을 동적으로 조절하는 방식[7-9], 등이 있었다. 하지만 이러한 기존의 방법은 계산량을 줄일 수 있었으나 FSA 보다 좋지 않은 성능을 보여 왔다. 이 때문에 기존의 방법을 사용하고자 해도 고화질을 만족해 줄 수 없기에 다른 대안을 찾기 어려웠다. 하드웨어 기술이 발전하며 어느 정도의 계산량은 감당할 수 있고 고화질을 만족시키는 방법에 대해 초점이 모아지고 있어 FSA 의 성능은 높이 평가되고 있다. 그리하여 FSA 와 유사한 성능을 유지하면서 이보다 계산량을 효율적으로 줄인 방식을 요구하고 있다.

FSA 가 비효율적인 계산량의 부담을 가지고 있을 수 밖에 없는 이유는 보통 프레임간의 움직임이 있는 블록의 수가 적고 움직임이 존재하는 경우도 움직임의 크기가 작은 것이 대다수 인데 움직임의 크기에 상관없이 모든 블록에 대해 고정적인 탐색 영역이 결정되어 있기 때문이다. 그러하기에 계산량의 부담을 줄일 수 있는 방법 중의 하나는 움직임의 정도를 예측하여 움직임의 정도에 따라 동적으로 탐색 영역을 결정하는 것이다.

본 논문에서는 움직임의 정도를 예측하는 방법과 이에 따라 동적인 탐색 영역을 결정하는 방법에 대해 제안하였다. 움직임의 정도를 예측하는 방법은 임의 블록의 움직임 벡터가 공간상에 인접한 블록들과 매우 비슷한 특성을 갖는다는 점을 고려하여 인접한 블록의 움직임 벡터를 이용하였고 이에 따라 동적인 탐색 영역을 결정하되 움직임 벡터 추정의 오류를 피하기 위한 조건을 추가하였다.

본 논문은 다음과 같이 구성되었다. 2장에서는 FSA 의 계산량 부담 문제에 대해 다루었고, 3장에서는 2장에서의 문제를 해결하기 위한 제안된 알고리즘을 설명하였고, 4장에서는 실험을 통해 제안된 알고리즘을 검증하였다. 마지막으로 결론과 차후 연구 과제가 5장에 이어진다.

2. FSA 의 계산량 부담 문제

움직임 벡터 추정을 $N \times N$ 크기의 블록 단위로 하는 경우, 일반적으로 BMA 는 연속하는 프레임을 현재 프레임과 이전 프레임으로 나누고 방법에 따라 각각 다르게 정의한 탐색 지점에서 블록 단위로 BDM 을 계산하여 최소값을 갖는 위치로 움직임 벡터를 다음과 같이 추정한다.

$$E(x,y)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|\hat{f}_{t-1}(i+x,j+y)-f_t(i,j)\right|,\ -w\le x,y\le w \quad (1)$$

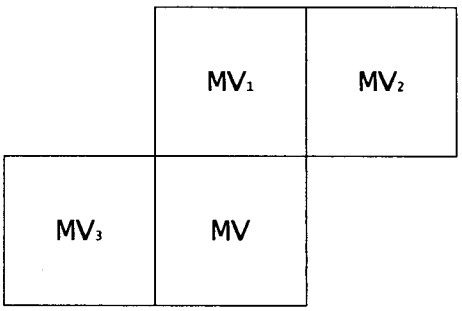
식 (1)은 BDM 의 측정 방식 중 하나인 SAD(the sum of absolute difference)이고 (x,y) 는 움직임 벡터를 찾고자 탐색하는 수평과 수직 방향의 지점, $\hat{f}_{t-1}$ 와 f_t 는 압축된 이전과 입력 영상이고 w 는 움직임 벡터가 가질 수 있는 최대값이다. 일반적으로 w 이 크면 클수록 계산량은 증가하고 w 이 작으면 작을수록 계산량은 줄어들지만 계산량을 줄이기 위해 사용자에 의해 설정된 움직임 탐색 영역 w 을 작게 하는 것은 영상의 압축 효율을 감소시키는 결과를 낳게 된다.

FSA 는 움직임 벡터의 최대값인 w 이 고정되어 있고 w 안의 모든 탐색 지점에서 SAD 를 계산하기 위해 블록의 움직임의 정도에 상관 없이 $(2w+1)^2$ 의 상당한 계산량을 갖는다. 대개 블록의 움직임 벡터가 원점에 몰려 있기 때문에 이를 중심으로 일정 영역 내에 모든 지점에서 탐색을 하는 방식은 움직임의 크기가 이 영역을 벗어나지 않는 한 움직임 벡터를 정확히 찾아 좋은 성능을 갖게 된다. 그러나 고정적인 탐색 영역으로 인해 계산량 부담 문제를 벗어날 수 없는 것이다.

이러한 문제를 해결하기 위한 방법으로 w 을 모든 블록에 대해 고정적으로 두지 않고 움직임의 정도를 예측하여 움직임의 정도에 따라 블록마다 동적으로 결정할 수 있다면 계산량을 효율적으로 줄일 수 있다. 보통 동영상에서 움직임이 있는 블록은 인접한 블록들과 비슷한 움직임의 크기와 방향을 갖고 움직이는 물체는 인접한 블록들로 이루어져 있는 경우가 많다. 즉, 대개 인접한 블록들의 움직임 벡터의 상관 관계는 높으므로 인접한 블록들의 움직임 벡터를 이용하면 움직임의 정도의 예측이 가능할 수 있다.

3. 제안된 알고리즘

한 프레임에서 블록 단위로 SAD 를 계산할 때, 프레임의 맨 위의 왼쪽 블록에서 시작하여 왼쪽에서 오른쪽으로, 위에서 아래의 순서로 각 블록에 대한 움직임 벡터를 추정해 간다. 그리하여 그림 1 과 같이 현재 블록의 움직임 벡터에 대해 상관 관계가 높은 인접한 블록은 현재 블록의 상위 블록, 상위 오른편 블록, 왼편 블록이라는 것을 알 수 있다. 이렇게 이미 계산된 블록들의 움직임 벡터를 이용하여 현재 블록의 움직임 정도를 예측하기 때문에 움직임 예측을 위해 특별 한 계산 없이 이를 쉽게 예측할 수 있다.



MV : 현재 블록의 움직임 벡터
MV_1: 상위 블록의 움직임 벡터
MV_2: 상위의 오른편 블록의 움직임 벡터
MV_3: 왼편 블록의 움직임 벡터

그림 1. 현재 블록과 인접한 블록의 움직임 벡터

그러므로 FSA 의 모든 블록에 대한 고정적인 탐색 영역 대신에 블록마다 동적인 탐색 영역을 수직, 수직 방향에 대해 각각 다음과 같은 단계로 정의하였다.

Step 1 : 움직임 정도 예측

$$\text{max\_mv\_x} = \max(|MV_1\_x|, |MV_2\_x|, |MV_3\_x|)$$
$$\text{max\_mv\_y} = \max(|MV_1\_y|, |MV_2\_y|, |MV_3\_y|) \quad (2)$$

Step 2 : 동적 영역 결정

$$\text{dynamic\_search\_area\_x} = \max(2*\text{max\_mv\_x}, |w|/4)$$
$$\text{dynamic\_search\_area\_y} = \max(2*\text{max\_mv\_y}, |w|/4) \quad (3)$$

Step 3 : 새로운 탐색 영역 결정

$$\text{new\_search\_area\_x} = \min(\text{dynamic\_search\_area\_x}, |w|/4)$$
$$\text{new\_search\_area\_x} = \min(\text{dynamic\_search\_area\_x}, |w|/4) \quad (4)$$

단계 1 에서는 인접한 블록들의 움직임 벡터 중 절대값이 최대인 것을 식 (2)에서 매개변수로 정의하였다 또한 단계 2 을 통해 이 매개변수를 2 배 해 준다. 이는 인접한 블록들의 움직임 벡터와 현재 블록의 움직임 벡터의 상관 관계가 낮은 non-stationary 영역일 때, 예측한 움직임보다 더 큰 움직임이 발생하여 탐색 영역이 이에 비해 작다면 움직임 벡터 추정의 오류를 발생할 수 있기 때문이다. 이러한 과정을 통해 인접한 블록의 움직임 벡터로 탐색 영역을 결정하기 위한 현재 블록의 움직임의 정도를 예측하였다.

단계 2 에서는 움직임 벡터가 가질 수 있는 최소 크기를 식 (3)에서 $|w|/4$ 로 정의하였다. 그리고 단계 1 에서 정의한 매개변수와 비교하여 더 큰 값을 동적 탐색 영역으로 결정하였다. 이는 인접한 블록들의 움 직임 벡터가 모두 '0'의 값을 가질 때, 탐색 영역이 '0'으로 정의되어 움직임 벡터 추정의 오류를 발생할 수 있기 때문이다. 이러한 경우는 단지 블록 단위로 동적인 탐색 영역을 결정하는 과정에서 한 블록에만 영향을 미치지 않고 연속적으로 오류를 발생할 수 있

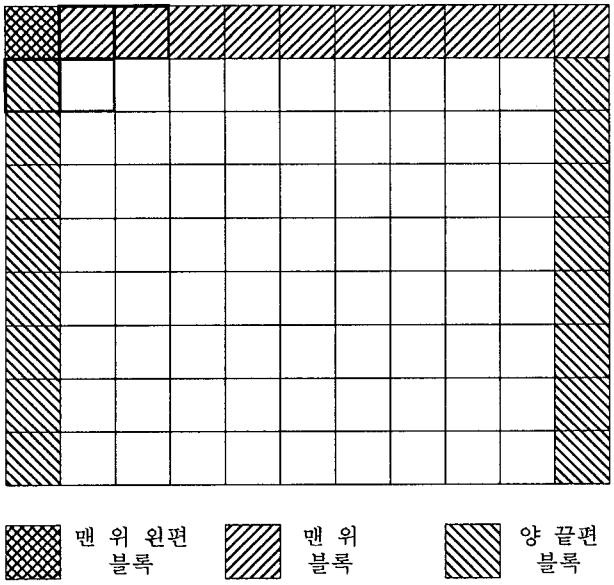


그림 2. 프레임 안의 블록의 위치

다. 단계 2는 이를 방지하지 위한 조건을 둔 것이다.

마지막으로 단계 3 에서는 움직임 벡터가 가질 수 있는 최대 크기를 식 (4)에서 $|w|$로 정의하였다. 그리고 단계 2 에서 결정한 동적 탐색 영역과 비교하여 더 작은 값을 각 블록에 대한 새로운 탐색 영역으로 결정하였다. 이는 단계 2 에서 결정된 동적 탐색 영역이 너무 커 불필요한 계산을 할 수 있는 경우를 피하기 위한 조건이다.

이와 같이 움직임 벡터를 추정하는 과정에서 주의할 점은 한 프레임 안에 모든 블록에 대해 위의 단계가 적용되지 않는다는 것이다. 그림 2 와 같이 프레임의 맨 위 왼편 블록은 그림 1 에서 정의한 인접한 블록의 움직임 벡터 MV_1, MV_2, MV_3를 모두 알 수 없고 이를 제외한 맨 위 블록들은 인접한 블록 중에 MV_2, MV_3를 알 수 없어 위의 단계를 적용하기에는 움직임의 정도를 예측하기 위한 정보가 부족하여 무리가 있다. 이처럼 탐색 영역을 결정하는 인접하는 블록의 움직임 벡터를 두 개 이상 알 수 없는 경우는 위의 단계에 상관없이 탐색 영역의 크기를 $|w|$으로 한다. 하지만 프레임에서 위에서 언급한 경우를 제외한 양 끝편 블록과 같이 인접하는 블록의 움직임 벡터를 한 개만 알 수 없는 경우는 위의 절차를 그대로 사용한다. 이 때 알 수 없는 하나의 블록은 움직임 벡터 값을 '0'의 값으로 두어 무시하고 나머지 두 블록에 의해 탐색 영역을 결정한다.

본 논문에서 제안된 움직임 벡터 추정을 위한 동적 탐색 영역 결정 방식은 블록마다 동적인 탐색 영역을 결정하여 FSA 의 장점인 일정한 크기 안의 움직임 벡터를 정확하게 예측하고 FSA 의 단점인 움직임 적은 대부분의 블록들에 대해 불필요한 계산을 피할 수 있는 효과적인 방법이다.

4. 실험 결과 및 분석

제안된 방식을 H.263+ TMN8(Test Model Near-term)에 적용하여 실험하였고 고정적인 탐색 영역을 갖는 FSA 과 비교하였다. 그리고 영상간의 움직임 정도가 다른 여러 영상에 대해 다양한 비트 율 및 frame rate 에 적용하였다. 또한 w의 크기도 달리 해 보았다. 본 논문에서는 이 중, QCIF 크기를 갖고 30 frames/sec 인 300 frames 의 동영상이 입력 되었을 때, 움직임 벡터가 가질 수 있는 최대값 $w = 15$ 에서 움직임 벡터 추정하는 동영상 부호화 과정을 통해 10 frames/sec 로 압축하여 100 frame 을 출력한 Foreman 과 News 에 대해 기술하기로 한다. 제안된 알고리즘의 성능을 측정하기 위해 PSNR(Peak to Signal to Noise Radio)가 사용되었으며 제안된 알고리즘의 계산량을 측정하기 위해 실험한 동영상의 각 블록 당 평균 탐색 지점 수로 다음의 식을 이용하였다.

$$\text{Relative Operation (\%)} = \frac{\text{the number of search points in the proposed algorithm}}{\text{the number of search points in the TMN8}} \times 100 \quad (5)$$

그림 3 에서 Foreman 경우는 100 frame 의 평균 PSNR 가 제안된 알고리즘과 기존의 방식이 비트 율에 상관없이 유사하다는 것을 볼 수 있다. 반면에 계산량은 그림 4 에서 나타나는 바와 같이 제안된 알고리즘이 기존의 방식에 비해 비트 율에 따라 지속적으로 줄어들며 32~38% 정도 줄어드는 것을 알 수 있다. 또한 Foreman 보다 움직임의 크기가 작은 News 도 그림 5, 6 에서 볼 수 있듯이 Foreman 과 마찬가지로 제안된 알고리즘과 기존의 방식이 비트 율에 따라 성능의 변화는 거의 없으나 계산량이 꾸준히 줄어드는 것을 알 수 있다. 하지만 News 움직임의 크기가 Foreman 보다 작기 때문에 13% 정도로 계산량의 줄어드는 크기가 더 큰 것을 알 수 있다. 즉, 움직임의 정도에 따라 탐색영역을 동적으로 결정하였기에 나타나는 현상으로 볼 수 있다.

5. 결론 및 차후 연구 과제

동영상 압축 방식의 시간상 중복성을 줄여주는 움직임 벡터 추정 방식에서 계산량을 효율적으로 줄이면서 기존의 방식 중 성능이 가장 좋은 방식인 FSA 과 유사한 성능을 유지하는 방식에 대해 제안하였다. 블록 단위로 움직임 벡터 추정하는 과정에서 블록마다 동적인 탐색 영역을 결정하였다. 움직임 벡터 추정을 하고자 하는 블록에 대한 움직임의 정도를 예측하는 매개변수가 정의되었고 새로운 탐색 영역을 정의하였다. 제안된 방식의 이점은 실험을 통하여 검증할 수 있었다.

현재는 움직임의 정도에 더 민감하게 반응하는 탐색 영역에 대해 연구 중에 있다. 단지 탐색 영역을 결정하게 하는 인접한 블록의 움직임 벡터 크기 외에 움직임의 방향을 고려한 방법과 본 논문에서 정의한 움직임 벡터가 가질 수 있는 최소 크기를 변화시켜 움직임의 정도에 따라 민감하게 반응하는 방법에 대한 것이다. 이는 제안된 방식의 성능과 별차이 없이 계산량을 더욱 줄여 줄 수 있을 것으로 생각된다.

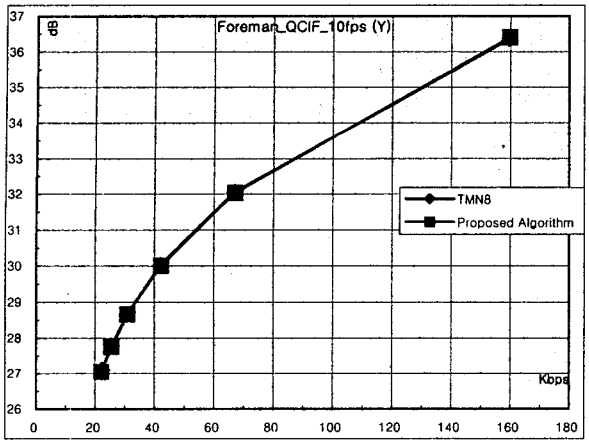


그림 3. QCIF Foreman 동영상의 비트 율에 따른 PSNR 비교

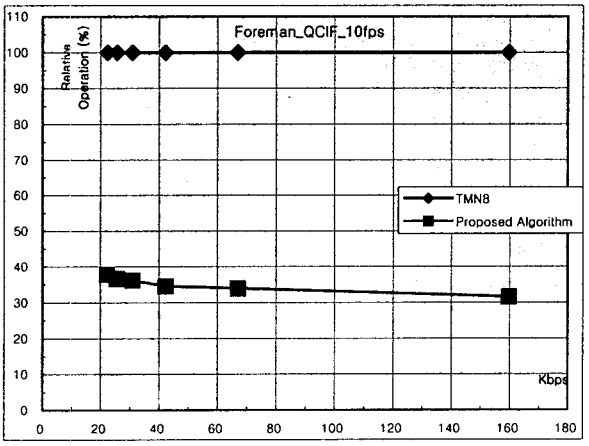


그림 4. QCIF Foreman 동영상의 비트 율에 따른 계산량 비교

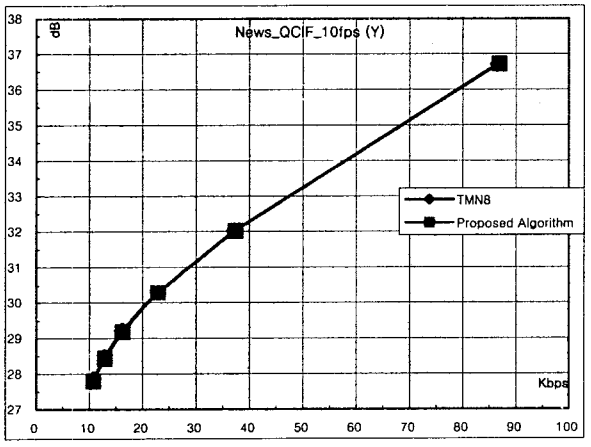


그림 5. QCIF News 동영상의 비트 율에 따른 PSNR 비교

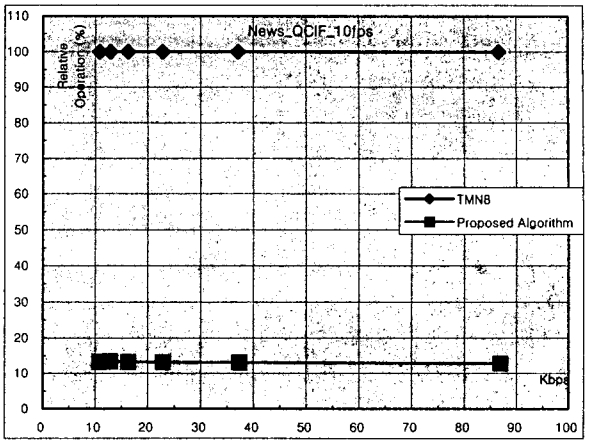


그림 6. QCIF News 동영상의 비트 율에 따른 계산량 비교

참고 문헌

[1] J. R. jain and A. K. jain, "Displacement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799-1808, Dec. 1981.

[2] V. Christopoulos and J. Cornelis, "A Center-Biased Adaptive Search Algorithm for Block Motion Estimation," *IEEE Trans. Circuits and Systems for Video Tech*, vol. 10, no. 3, pp. 423–426, Apr. 2000.

[3] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing*, vol. 4, no. 1, pp. 105-107, Jan. 1995.

[4] M. Brunig and W. Niehsen, "Fast Full-Search Block Matching," *IEEE Trans. Circuits and Systems for Video Tech*, vol. 11, no. 2, pp. 241–247, Feb. 2001.

[5] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits and Systems for Video Tech*, vol. 3, no. 2, pp. 148–157, Apr. 1993.

[6] Y. –L. Chan and W. –C. Siu, "New Adaptive Pixel Decimation for Block Motion Vector Estimation," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 6, no. 1, pp. 113-118, Feb. 1996.

[7] L. W. Lee, J. F. Wang, J. Y. Lee, and J. D. Shie, "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 3, no. 1, pp. 85-87, Feb. 1993.

[8] J. Feng, K. T. Lo, H. Mehrpour and A.E. Karbowiak, "Adaptive Block Matching Algorithm," *Electron. Lett.*, vol. 32, pp. 1542-1543, Aug. 1995.

[9] H. S. Oh and H. K. Lee, "Adaptive adjustment of the search window for block-matching algorithm with variable block size," *IEEE Trans. Consumer Electron.*, vol. 44, pp. 659-666, Aug. 1998.