# H.264 동영상 압축을 위한 낮은 복잡도를 갖는 부 화소 단위에서의 움직임 추정

*이윤화, 신현철
한양대학교 전자전기제어계측공학과
e-mail : yhlee@digital.hanyang.ac.kr, shin@hanyang.ac.kr

# Sub-pixel Motion Estimation Algorithm with Low Computation Complexity for H.264 Video Compression

*Yun-Hwa LEE, Hyunchul SHIN
Department of Electronics, Electrical, Control and Instrumentation Engineering
Hanyang University

## Abstract

Motion Estimation(ME) is an important part of video compression, because it requires a large amount of computation. Half-pixel and quarter-pixel motion estimation allows high video compression rates but it also has high computation complexity. In this paper we suggest a new and efficient motion estimation algorithm for half-pixel and quarter-pixel motion estimation using SAD values. In the method, an integer-pixel motion vector is found and then only three neighboring points of the integer-pixel motion vector is evaluated to find the half-pixel motion vector. The quarter-pixel motion vector is also found by using a similar method. Experimental results of our method shows 20% reduction in computation time, when compared with those of a conventional method, while producing same quality motion vectors.

## I. 서론

오늘날 고속 네트워크의 인프라 확대와 디지털 기술의 발전으로, 동영상을 제한된 용량의 채널을 통하여 전송하거나, 저장매체에 저장하기 위해, 동영상 압축의 기술 개발이 활발히 진행 되고 있다. 일반적으로 영상는 방식으로 나뉘어 질 수 있다. 시간적 중복성을 제거하기 위해서는 움직임 추정 (Motion Estimation)과 움직임 보상 (Motion Compensation)을 사용한다. 이 과정은 동영상 압축에서 연산량의 60% 이상을 차지하고 있는 중요한 처리 과정이다.[JZY03]

움직임 추정의 방법은 추정의 기본 단위에 따라 크게 화소 순환 알고리듬 (Pel Recursive Algorithm: PRA)과 블록 정합 알고리듬 (Block Matching Algorithm: BMA)으로 나누어진다. 두 방법 중 수행시간이 적게 소요되고 간단한 블록단위를 기준으로 하는 블록 정합 알고리듬이 기존의 압축표준에서 많이 사용되어 왔다 [TKK81]. 현재 프레임과 가장 비슷한 위치의 블록을 찾기 위한 측정 방법으로는 평균 자승오차(Mean Squared Error: MSE), 절대차의 합 (Sum of Absoulute Differences: SAD)등이 있다.

동영상에서 물체의 움직임은 샘플링 격자 간격 (Sampling Grid Distance)의 정수배 단위로만 일어나지는 않기 때문에 정수 격자 단위에서의 움직임 벡터가 실질적인 움직임 벡터가 아닐 수 있다. 따라서 분수화소(Fractional pel)단위의 움직임 추정을 통한 움직임 벡터는 압축률을 증가 시킬 수 있다.이런 이유로 부 화소 단위에서의 움직임 추정이 H.264의 표준으로 채택되었다.

본 논문에서는 1/2과 1/4화소 단위에서의 움직임 추정 시, 탐색 점을 8개에서 3개로 줄여, 기존의 8개의

점에서의 움직임 추정 방법보다 연산량을 줄이는 낮은 복잡도의 알고리듬을 제안하였다. 제안한 방법에서는 기존 알고리듬 보다 약 20% 연산시간이 감소하는 성능을 보였다.

Ⅱ절에서는 H.264에서의 1/2 화소와 1/4화소 단위에서의 움직임 추정에 대한 기존 연구방법을 기술한다. Ⅲ절에서는 H.264에서 1/2화소와 1/4 화소단위에서의 제안한 움직임 추정 알고리듬을 기술하고 Ⅳ절에서는 제안한 방법으로 실험 한 결과를 설명하고, Ⅴ절에서는 결론을 맺는다.

## Ⅱ. 기존의 움직임 추정 방법

### 2.1. 오차 값 및 오차 표면(Error Surface)의 특성

움직임 추정 과정에서 블록단위당 움직임 벡터를 구할 때 가장 비슷한 블록을 찾는 오차 측정값으로 연산과정에서 곱셈이 필요 없고 간단한 SAD가 사용되었다.

$$SAD = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}|c_{ij} - r_{ij}| \qquad (1)$$

식 (1)에서 $N \times N$은 블록 사이즈, $c_{ij}$와 $r_{ij}$는 현재 프레임과 참조 프레임의 $(i, j)$ 위치의 샘플을 의미한다. 정수화소에서 찾아진 결과의 ±1 거리의 화소 내에서 부 화소(Sub pixel)의 농도는 식(2)와 같이 양선형 보간 (Bilinear Interpolation) 기법을 이용하여 얻어진다.

$$\begin{aligned} I(x_s,y_s) = {} & I(x,y)(x+1-x_s)(y+1-y_s) \\ & + I(x+1,y)(x_s-x)(y+1-y_s) \\ & + I(x,y+1)(x+1-x_s)(y_s-y) \\ & + I(x+1,y+1)(x_s-x)(y_s-y) \end{aligned} \qquad (2)$$

$I(x,y)$는 $(x,y)$의 정수 화소 점에서의 농도, $(x_s,y_s)$는 부 화소의 위치이다.

그림 1은 식 (2)를 이용하여 정수화소 단위에서 찾아진 최소 SAD를 갖는 위치에서 ±1내에 있는 부 화소 단위에서의 SAD오차 값들을 표현한 그래프이다. 그림 1을 통해 정수배 화소단위에서 찾아진 SAD값을 중심으로 ±1내의 8점은 더 큰 값을 가지며, 이 범위에서 부 화소 단위의 움직임 벡터를 포함 한다는 가정을 할 수 있다. 대부분의 고속 탐색 알고리즘은 이 가정을 기초로 두고 있다. SAD 오차표면의 이웃 화소 간 포물선 분포를 이용하여 정수 화소 단위에서의 가장 작은 SAD를 갖는 화소 주변 값들 중에서 두 번째로 작은 SAD값을 찾을 수 있다. 탐색에 있어서의 복잡도를 줄이기 위해 두 번째 작은 SAD를 갖는 방향으로 반 화소 단위에서의 가장 작은 SAD를 갖는 화소 위치를 예측 하여 움직임 추정을 할 수 있다.

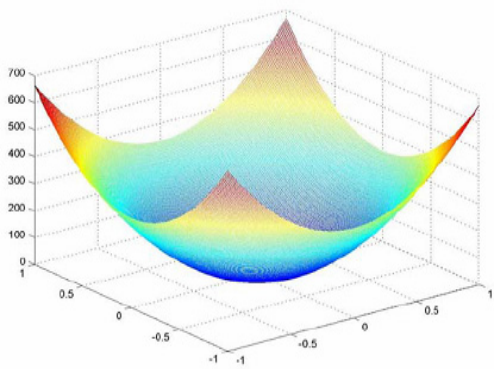


그림 1. 전형적인 오차표면 [HMW05]

### 2.2 1/2 화소와 1/4 화소 보간법

H.264에서는 1/2 단위의 화소는 식 (4)에서와 같이 6-tap FIR 필터(6-tap Finite Impulse Response Filter)를 이용하여 보간한다.

$$\begin{aligned} b_1 &= (E - 5F + 20G + 20H - 5I + J) \\ h_1 &= (A - 5C + 20G + 20M - 5R + T) \\ b &= (b_1 + 16) >> 5 \\ h &= (h_1 + 16) >> 5 \end{aligned} \qquad (4)$$

$b$ 와 $h$는 각각 1/2 화소 단위에서의 보간된 화소 값들을 나타내고, $A, C, E, F, G, H, M, I, J, T$는 정수 단위의 화소 값들이다. 1/2 단위의 화소 값을 생성하기 위해서는 주위의 정수단위 화소 값 6개를 이용한 연산이 필요하다.

1/4 화소 단위에서의 화소 값은 이중 선형 보간 (Bi-Linear Interpolation)의 식 (5)를 이용하여 보간 한다. 식 (5)에서 G ,H, b 값은 정수 화소 값이고, a와 c 값은 보간 된 1/2 화소 값을 나타낸다.

$$\begin{aligned} a &= round((G + b)/2) \\ c &= round((H + b)/2) \end{aligned} \qquad (5)$$

### 2.3 일반적인 1/2화소와 1/4 화소에서의 움직임 추정

일반적으로 1/2화소와 1/4화소 단위에서의 움직임 추정은 그림 2와 같이 각각 화소 단위에서 구해진 최소 오차 점을 기준으로 하여 상하 좌우 대각선의 8점에서 탐색 하여 각 화소단위에서의 움직임 벡터를 찾는다.

결과적으로 정수 화소에서만 움직임 추정을 하는 경우보다 1/2 화소단위에서 8개, 1/4화소단위에서 8개, 총 16개의 점에서 SAD 연산이 필요하게 되는 단점이 있다. 따라서 1/2화소 단위에서 화소 간 보간 및 블록 정합을 하면서 오차 값이 적고 낮은 복잡도의 고속 처리가 요구되는 2SS[BZJ03], PPHPS[CDU03] 알고리듬이 개발되었다.

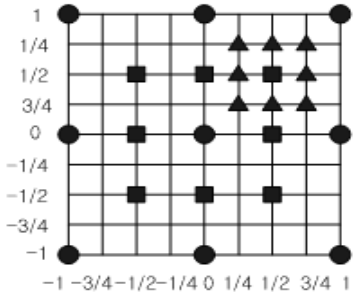


그림 2. 정수화소, 1/2 화소, 1/4 화소의 위치

## III. 제안한 움직임 추정 기법

### 3.1. 제안한 1/2 화소 단위에서의 움직임 추정

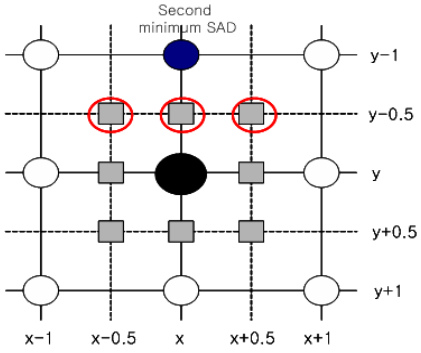


그림 3. 제안한 반 화소 단위에서의 움직임 추정 과정

본 논문에서는 그림1에서와 같은 SAD 오차표면의 포물선 분포 특징과 식 (4)에와 같이 6-tap FIR 필터로 1/2 화소를 보간 하는 H.264의 특성 이용하여 SAD가 가장 작은 위치와 SAD가 두 번째로 작은 위치 사이의 1/2 화소와 1/4 화소에서의 움직임 추정을 제안 하였다. 결국, 기존의 1/2과 1/4 화소 단위에서 각각 8개의 점에서의 탐색을 평균 3점으로 줄여 움직임 추정 하여 연산의 복잡도를 줄이고, 처리 시간을 빠르게 한다.

정수배 화소 단위에서 SAD 값이 가장 작은 점을 찾는다. 찾아진 점을 $(x,y)$로 설정한다. SAD 값의 분포특성을 이용하여 $(x,y)$의 ±1 내에 있는 이웃 정수 화소 점에서 SAD를 구하여 비교한다. 정수 화소 단위에서 두 번째로 SAD가 작은 점 찾는다. 두 번째로 SAD가 작은 방향과 같은 방향의 1/2 화소 점과 영상의 수평이동이 잦은 특성을 이용하여 수평위치에 있는 점들에도 움직임 추정을 하게 된다. 그림7과 같이 $(x,y-1)$수직위치가 두 번째로 작은 SAD를 갖는 정수 화소 점이라면, 같은 방향의 1/2 화소 점과 좌우 위치에 있는 점을 선택하여, 3점에서 1/2 화소단위의 오차 값인 SAD값을 계산한다. 1/2 화소 단위의 3점에서 얻은 SAD값 중 가장 작은 값과 기존 정수 화소 단위에서의 SAD값과 비교하여 1/2 화소 단위에까지 움직임 추정 시, 움직임 벡터 값을 찾을 수 있다. SAD값이 두 번째로 작은 위치가 $(x+1,y)$과 같이 수평 위치 일 때, $(x+0.5,y)$,$(x-0.5,y)$의 총 두 점에서 1/2 화소 단위에서의 움직임 벡터를 구하게 된다.

### 3.2. 제안한 1/4 화소 단위에서의 움직임 추정

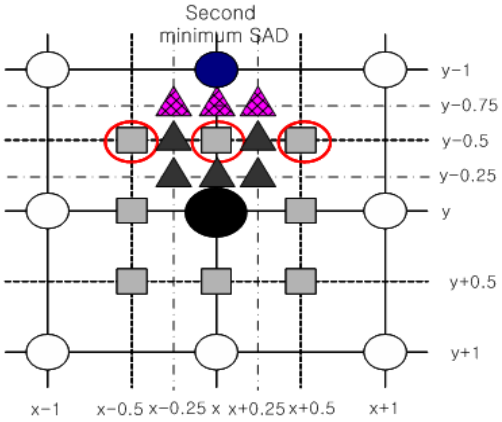


그림 4. 제안한 1/4화소 단위에서의 움직임 추정 과정

1/2 화소 단위에서 찾아진 점 주위의 1/4 화소단위의 8개의 점 중, SAD 에러 표면특성을 이용하여 다음 그림 8과 같이 3개의 점에서만 탐색을 하였다. 1/2 화소 단위에서 찾은 위치가 정수 화소 단위에서 두 번째로 작은 SAD값을 가지는 방향과 같을 때, 1/2 화소 단위에서의 움직임 추정 방법과 같이 동일한 방향의 1/4 화소 점과 수평위치의 좌우 점을 선택하여 탐색을 시작한다. 1/2 화소단위에서 찾은 위치가 정수화소 단위에서 두 번째로 작은 SAD를 가지는 점과 다를 때, 1/2 화소 단위에서 찾아진 SAD가 가장 작은 점 방향으로 1/4 화소 단위에서 3점을 선택하여 움직임 벡터를 구하게 된다.

## IV. 실험결과

본 논문에서 제안한 방식의 성능을 테스트하기 위해 QCIF 포맷의 시퀀스 Akiyo, Salesman, Claire, Hall를 H.264 부호화 환경 아래에서 실험 하였다. 각각의 영상이 움직임 추정 과정에서 사용된 블록의 크기는 7종류의 가변 블록을 지원하였으며 탐색 영역은 33x33 사

이즈가 사용되었다. 각 시퀀스는 20 프레임 단위당 실험 되었고, 사용된 서버사양은 CPU는 Opteron 2.5 GHz 이고 메모리는 4G를 지원한다. 실험은 JM8.2에서 정수 화소 단위에서 구현된 고속 탐색 알고리즘을 그대로 사용하였고, 1/2 과 1/4 화소 단위에서 탐색 점을 줄이는 제안한 방법과 기존의 8개의 화소위치에서 전부 탐색하는 방법과 비교 하여 그 결과가 표 2에 표시되 있다. 실험 결과, 제안한 알고리듬은 기존 알고리듬과 비교해 화질과 비트율에서 거의 차이가 없었고, 전체 움직임 추정 연산 시간에서 약 20%의 감소를 보였다. 1/2 화소와 1/4 화소 단위에서의 제안한 움직임 추정 알고리듬은 기존 알고리듬의 부 화소 단위에서의 움직임 추정 보다 약 40% 정도의 연산처리시간 감소를 확인할 수 있었다.

## V. 결론

압축효율이 뛰어난 H.264 인코더의 가장 큰 연산을 차지하는 움직임 추정 과정에서 보다 효율적인 처리가 가능하도록 낮은 복잡도의 1/2 화소와 1/4화소 단위에서의 움직임추정 방법을 제안하였다. 이는 SAD의 에러 표면 특성과 화소의 보간 특성을 이용하여 부 화소 단위에서의 오차 값이 작은 방향을 예측하여 움직임 벡터를 찾는 방법이다. 예측된 방향으로만 탐색하여 기존의 각각8개의 탐색 점을 3점으로 줄이게 되었다. 제안한 방법으로 5개의 영상을 JM8.2[IPH01]를 통해 실험한 결과, 일반적인 움직임 추정 방법과 비슷한 성능을 가지면서 전체 움직임 추정 처리 과정으로는 약 20%의 연산 시간의 감소가 있었고, 부 화소 단위에서만의 움직임 추정 처리과정으로는 약 40%정도의 연산 시간 감소가 있었다.

## 감사의 글

본 연구보고서는 정보통신부의 출연금등으로 수행한 정보통신 연구개발 사업의 연구결과 입니다.

## 참고문헌

[JZY03] Jianning Zhang, Yuwen He, Shiqiang Yang, Yuzhuo Zhong, "Performance and complexity Joint Optimization For H.264 Video Coding," Circuits and Systems, ISCAS '03. Proceedings of the 2003 International Symposium on Vol 2, pp. 25-28, May 2003

[TKK81] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in Proc. NTC81, New Orleans, LA, pp. C9.6.1-9.6.5, Nov. 1981

[IPH01] http://iphome.hhi.de/

[BZJ03] Bo Zhou, Jian Chen " A Fast Two- Step Search Algorithm for Half-pixel Motion Estimation" Electronics, Circuits and Systems, ICECS 2003. Proceedings of the 2003 10th IEEE International conference on Vol 2, pp.611-614 Dec 2003

[CDU03] Cheng Du, Yun He, and Junli Zheng, "PPHS: A Parabolic Prediction-Based, Fast Half-Pixel Search Algorithm for Very Low Bit-Rate Moving-Picture Coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No.6, June 2003

[HMW05] Hoi-Ming Wong, Au, O.C., Jinxin Huang, Shiju Zhang, Yan, W.N. "Sub-Optimal Quarter-Pixel Inter-Prediction Algorithm (SQIA)", Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on Volume 2, pp. 921 - 924, March 18-23, 2005

표 1. JM8.2 와의 제안한 ME 전체 처리 과정에 대한 성능 비교

| | Akiyo_qcif | | Salesman_qcif | | Claire_qcif | | Hall_qcif | |
|---|---|---|---|---|---|---|---|---|
| | JM8.2 | Proposed | JM8.2 | Proposed | JM8.2 | Proposed | JM8.2 | Proposed |
| PSNR(dB) | 40.596 | 40.593 (-0.003) | 38.533 | 38.543 (+0.01) | 40.956 | 40.926 (-0.03) | 39.3 | 39.9 (-0.00) |
| Bit-rate (k/bit) | 43.91 | 43.85 (-0.06) | 113.38 | 116.80 (+3.42) | 104.59 | 106.56 (+1.97) | 47.94 | 48.13 (+0.19) |
| ME처리시간 (sec) | 44.936 | 35.39 (78.76%) | 42.08 | 34.086 (81.68%) | 44.98 | 34.856 (81%) | 47.706 | 37.59 (77.50%) |
| 부 화소 단위 처리시간 (sec) | 8.036 | 5.05 (60.37%) | 7.65 | 4.546 (57.08%) | 7.913 | 4.516 (59.43%) | 8.736 | 4.496 (57.07%) |