H.264 동영상 압축을 위한 부 화소 단위에서의 고속 움직임 추정 방법

(A Fast Sub-pixel Motion Estimation Method for H.264 Video Compression)

이 윤 화 † 최 명 훈 †† 신 현 철 †††
(Yunhwa Lee) (Myunghoon Choi) (Hyunchul Shin)

요 약 움직임 추정은 H.264의 비디오 코딩 과정에서 가장 많은 연산량을 차지하는 중요한 처리과정이다. 움직임 추정 과정에서 정수배 화소 단위에서의 탐색에 비하여, 1/2 화소 (half-pixel)와 1/4 화소 (quarter-pixel) 단위까지의 움직임 추정은 영상압축률을 높일 수 있지만, 계산의 복잡도가 늘어나는 문제가 있다. 본 논문에서는 각 블록간의 절대 오차 값인 SAD (Sum of Absolute Difference)가 최소 점을 기준으로 포물선 모양의 분포를 나타내는 특성 및 1/2 단위와 1/4 단위의 화소 보간 특성을 이용하여 움직임 추정 과정에서 탐색 점을 줄임으로써 처리속도를 증가시키고, 계산의 복잡도를 줄이는 알고리듬을 제안하였다. 제안한 방법에서는, 정수 화소 단위에서의 가장 작은 SAD를 갖는 점을 기준으로 주위 8점 가운데 두 번째로 SAD가 작은 점을 찾아 해당 방향으로 1/2 화소 단위의 움직임 추정을 행하였고, 1/4 화소 단위에서도 1/2 화소단위에서 두 번째로 SAD가 작은 점 방향으로 움직임 추정을 실행하였다. 그 결과 기존 알고리듬에 비해 비교적 화질에 변화가 없고, 인코더 처리과정 에서 약 20%의 빠른 속도로 처리하는 결과를 보였다.

키워드 : H.264 인코더, 부 화소 움직임 추정, 낮은 복잡도, SAD, 영상압축

Abstract Motion Estimation (ME) is an important part of video coding process and it takes the largest amount of computation in video compression. Half-pixel and quarter-pixel motion estimation can improve the video compression rate at the cost of higher computational complexity. In this paper, we suggest a new efficient low-complexity algorithm for half-pixel and quarter pixel motion estimation. It is based on the experimental results that the sum of absolute differences(SAD) shows parabolic shape and thus can be approximated by using interpolation techniques. The sub-pixel motion vector is searched from the minimum SAD integer-pixel motion vector. The sub-pixel search direction is determined toward the neighboring pixel with the lowest SAD among 8 neighbors. Experimental results show that more than 20% reduction in computation time can be achieved without affecting the quality of video.

Key words : H.264 encoder, sub-pixel motion estimation, SAD, Video compression

1. 서 론

오늘날 고속 네트워크의 인프라 확대와 디지털 기술의 발전으로 네트워크를 통한 멀티미디어 정보의 전송이 늘어나고 있으며, 이에 따라 동영상 압축 등의 기술개발이 활발히 진행되고 있다. 일반적으로 영상압축 방법은 시간적 중복성, 공간적 중복성을 제거하는 방식으로 나뉘어 질 수 있다. H.264에서는 프레임의 상관관계를 이용하여, 프레임사이의 움직임 예측을 통한 시간적 중복성을 제거하기 위해서 움직임 추정(Motion Estimation)과 움직임 보상(Motion Compensation)을 사용한다. 이 과정은 동영상 압축에서 일반적으로 연산량의 60% 이상을 차지하고 있는 중요한 처리 과정이다[3].

움직임 추정의 방법은 추정의 기본 단위에 따라 크게 화소 순환 알고리듬(Pel Recursive Algorithm: PRA)과

· 본 연구보고서는 정보통신부의 출연금등으로 수행한 정보통신연구개발사업의 연구결과입니다.
† 학생회원 : 한양대학교 전자전기제어계측공학과
yhlee@digital.hanyang.ac.kr
†† 정 회 원 : 삼성전자 반도체 메모리사업부 플래시 메모리 개발팀 연구원
mh95.choi@samsung.com
††† 종신회원 : 한양대학교 전자전기제어계측공학과 교수
shin@hanyang.ac.kr
논문접수 : 2005년 11월 10일
심사완료 : 2006년 2월 24일

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[6]. 현재 프레임과 가장 비슷한 위치의 블록을 찾기 위한 추정 방법으로는 평균 자승오차(Mean Squared Error: MSE)와 절대차의 합(Sum of Absolute Differences: SAD)등이 사용된다.

움직임 추정 알고리듬에 있어서 참조 프레임에서의 탐색 영역 내의 모든 탐색 점에서 탐색을 행하여 움직임 벡터를 찾는 방법인 전역 탐색 알고리듬(Full Search Block Matching Algorithm : FSBMA)은 높은 정밀도를 가지고 있지만, 연산량이 많고 처리시간이 길다는 단점이 있다. 따라서 비슷한 성능을 가지면서 탐색 영역 내의 비중이 높은 탐색 점에서만 탐색을 행하는 고속 탐색 알고리듬(Fast Search Algorithm :FSA)이 주로 사용되고 있고, 고속 탐색 알고리듬으로 TSS[6], NTSS[7], FSS[8]등이 개발되었다.

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 1/2화소와 1/4화소에서의 움직임 추정이 H.264의 표준으로 채택되었다. 1/2화소단위에서의 움직임 추정은 정수배 격자 단위에서 찾아진 정수화소 위치를 중심으로 1/2화소 간격 떨어진 8개의 점에서 블록과의 오차 값을 계산하여 1/2화소 단위에서의 움직임 벡터를 구할 수 있다. 1/4화소 단위의 움직임 추정도 위의 방법과 동일하게 이뤄진다. 현재 고속 탐색 움직임 추정 알고리듬이 개발되면서, 정수배 단위에서의 탐색 점이 적게는 10개 가까이 됨에 따라[9], 1/2과 1/4화소 단위에서의 탐색 점의 수를 줄이는 방법들이 연구되기 시작했다.

본 논문에서는 1/2과 1/4 화소단위에서의 움직임 추정 시, 탐색 점을 8개에서 3개로 줄여 기존의 8개의 점에서의 움직임 추정 방법보다 연산시간을 줄이는 낮은 복잡도의 알고리듬을 제안하였다. 제안한 방법에서는 탐색 영역 내에서의 블록단위당 오차추정 값들의 분포 특성을 이용하여 가장 작은 오차 값을 갖는 점의 ±1 화소 내에서 두 번째로 작은 오차 값이 분포되어 있다는 가정 하에 두 번째로 작은 오차 값을 찾아 그 방향에서만 움직임 추정을 하여 기존 알고리듬 보다 인코더 처리과정 에서 20%이상의 연산시간이 감소하는 성능을 보였다. 제안한 방법의 성능을 평가하기 위해서 JM8.2상에서 실험결과를 비교하였다[4].

2절에서는 H.264에서의 1/2 화소와 1/4화소 단위에서의 움직임 추정에 대한 기존 연구방법을 기술한다. 3절에서는 H.264에서 1/2화소와 1/4 화소단위에서의 제안한 움직임 추정 알고리듬을 기술하고, 4절에서는 제안한 방법으로 실험한 결과를 설명하고, 5절에서 결론을 맺는다.

2. 기존의 움직임 추정 기법

2.1 오차 값 및 오차 표면(Error Surface)의 특성

움직임 추정 과정에서 블록단위당 움직임 벡터를 구할 때 가장 비슷한 블록을 찾는 오차 추정 값으로 연산과정에서 곱셈이 필요 없고 계산이 간단한 SAD가 주로 사용된다.

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

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

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

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

그림 1은 식 (2)를 이용하여 정수화소 단위에서 찾아진 최소 SAD를 갖는 위치에서 ±1내에 있는 부 화소단위에서의 SAD오차 값들을 표현한 그래프이다[2]. 그림 1과 같은 SAD 오차 표면은 표 1에서의 실험값을 통해 나타날 확률이 높음을 알 수 있다. 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.

표 1에서는 Claire, News, Salesman, Trevor 시퀀스의 SAD 오차표면 특성을 나타낸다. 실험결과는 표에서와 같이 96% 이상이 가장 작은 SAD값을 기준으로 주변의 SAD값들이 포물선 모양의 분포를 나타내었다[1]. SAD 오차표면의 이웃 화소간의 포물선 분포를 이용하여 정수 화소 단위에서의 가장 작은 SAD를 갖는 화소 점을 찾고, 그 위치를 중심으로 ±1내의 화소 주변 값들의 SAD를 계산한 후, 계산된 SAD값들의 분포특성을 이용해 1/2 화소와 1/4 화소단위에서의 가장 작은 SAD를 갖는 화소 위치를 예측 할 수 있다.

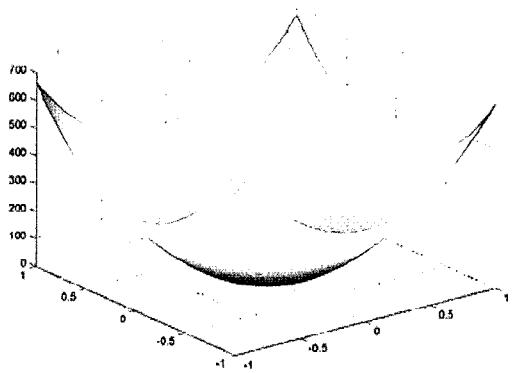

그림 1 전형적인 오차 표면

표 1 정수화소 단위에서 찾아진 위치의 ±1내에 있는 이웃 화소들의 SAD 분포 통계

Sequence	Number of frames	Number of blocks	Number of single valley blocks	Ratio
Claire	494	48807	47140	96.58%
News	300	29601	28908	97.66%
Salesman	449	44352	44106	99.45%
Trevor	150	14751	14199	96.26%

2.2 1/2 화소와 1/4화소 단위의 움직임 추정

인접 프레임간의 시간적 중복성을 줄이기 위한 움직임 추정 과정에서 더 정확한 움직임 추정을 통하여 높은 압축 효율을 얻기 위해 부 화소 단위의 움직임 추정 기법이 행해지고 있다. 현재 프레임과 이전 프레임과의 관계를 다음 식 (3)과 같이 나타낼 때, 수평 수직 방향으로의 실제 움직임 거리(Displacement distance)인 d_i, d_j는 반드시 샘플링 간격의 정수 배로만 나타나지 않는다.

$$I_t(i,j) = I_{t-1}(i-d_i, j-d_j) \quad (3)$$

$I_t(i,j)$는 현재 프레임의 (i,j)위치에서의 휘도 값, $I_{t-1}(i-d_i, j-d_j)$는 이전 프레임의 $(i-d_i, j-d_j)$에서 화소의 휘도 값이다.

2.2.1 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 보간 한다.

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

b와 h는 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.

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

식 (5)에서 G ,H, b 값은 정수 화소 값이고, a와 c 값은 보간 된 1/2 화소 값을 나타낸다. 1/4 화소가 수평 방향에 있을 때, 양 옆의 화소 값을 이용하여 보간 하고, 수직 방향일 때는 위, 아래의 화소 점을, 대각선 방향일 경우는 대각선 방향의 화소 점을 이용한다.

2.2.2 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 8점을 탐색 하여 각 화소단위에서의 움직임 벡터를 찾는다. 결과적으로 정수 화소에서만 움직임 추정을 하는 경우보다 1/2 화소단위에서 8개, 1/4 화소단위에서 8개 등 총 16개의 점에서 SAD 연산이 필요하게 되는 단점이 있다. 이러한 추가적인 연산량을 줄이기 위해, 1/2화소 단위에서 화소 간 보간 및 블록 정합을 하면서 오차 값이 적고 고속 처리가 요구되는 2SS[1], PPHPS[5] 알고리듬이 개발되었다.

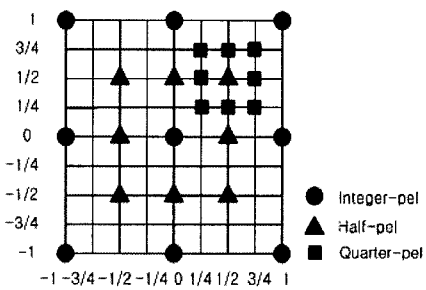

그림 2 정수화소, 1/2화소, 1/4화소의 위치

2.2.3 2SS(Two-Step Search) 알고리듬

2SS 알고리듬은 1/2 화소 단위에서의 움직임 벡터를 찾기 위해 수평과 수직방향으로의 이웃 1/2 화소 위치에서 순차적으로 탐색하는 기법이다. 다음 그림 3은 전형적인 2SS 과정을 보여주고 있다. 정수배 화소 단위에서의 가장 작은 SAD 값은 알고 있고, 수평과 수직 방향으로의 1/2 화소 단위에서 움직임 추정을 한다. 따라서 대부분의 경우 1/2 화소 단위의 4개점에서 움직임 추정을 한다. 만약 정수배 단위에서의 찾아진 화소 점이 프레임의 맨 위, 맨 아래 또는 좌우의 끝에 위치 할 경우, 1/2 화소 단위에서의 움직임 추정에서 수평과 수직 방향으로의 탐색점이 줄어든다.

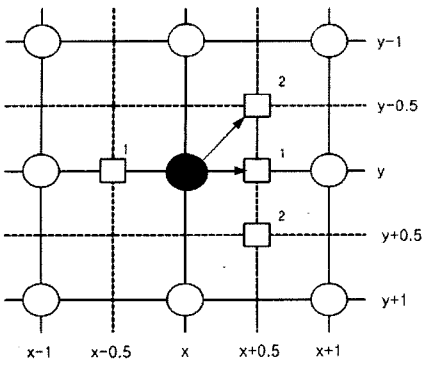

그림 3 2SS 과정

단계 1. 정수배 화소 단위에서 찾은 SAD가 가장 작은 화소 위치인 (x_1, y_1)를 중심으로 탐색을 시작한다. 먼저 수평 위치의 $(x_1-0.5, y_1)$, (x_1, y_1), $(x_1+0.5, y_1)$에서 블록 단위의 SAD를 계산하여 비교한다. 이중 가장 작은 SAD를 갖는 화소를 수직 방향 탐색의 중심인 (x_2, y_2)로 정한다.

단계 2. 수직 방향의 화소 점인 $(x_2, y_2-0.5)$,(x_2, y_2), $(x_2, y_2+0.5)$에서 블록 단위의 SAD를 계산하여 비교한다. 이때 가장 작은 SAD를 갖는 화소점이 1/2 단위에서의 움직임 벡터가 된다.

2SS 알고리듬과 제안한 알고리듬의 실험을 통한 비교 결과, 제안한 알고리듬은 평균 세 점에서 탐색하고, 2SS는 평균 네 점에서 탐색함으로 2SS 알고리듬 보다 약 25%의 연산 처리 시간을 줄였다.

3. 새로운 고속 움직임 추정 기법

3.1 새로운 1/2 화소 단위에서의 고속 움직임 추정

본 논문에서는 그림 1에서와 같은 SAD 오차표면의 포물선 분포 특징을 이용하여, 정수화소 단위에서 가장 작은 SAD를 갖는 점 주위의 정수 화소 8점에서 두 번째로 작은 SAD를 갖는 점을 찾아, 그 방향으로의 부화소 단위의 움직임 벡터 값을 찾도록 한다. 따라서 1/2 화소와 1/4 화소 단위에서의 움직임 벡터를 정수 화소 단위의 SAD값 위치를 통해 구한다.

또한, 식 (4)와 같이 6-tap FIR 필터로 1/2 화소를 보간 하는 H.264의 특성에서 각각, G와 H에 20이 곱하여져 있다. G와 H는 b라는 1/2 위치의 화소를 보간 하는데 있어, 양 옆의 정수 화소를 나타낸다. 따라서 1/2 화소는 보간 하려는 화소 양 옆의 점이 가장 큰 영향을 미친다는 것을 발견할 수 있었다. 위의 두 가지 특성을 이용하여 SAD가 가장 작은 위치와 SAD가 두 번째로 작은 위치 사이의 1/2 화소와 1/4 화소에서의 움직임 추정을 제안 하였다. 즉, 기존의 방법이 1/2과 1/4 화소 단위에서 각각 8개의 점을 탐색 하던 것을 평균 3점 탐색으로 줄여서 움직임 추정 과정에서의 연산의 복잡도를 줄이고, 처리 시간을 빠르게 하는 장점을 가지게 하였다. (x,y)점에서 SAD값이 최소라고 할 때, 1/2화소 단위의 움직임 벡터는 다음과 같이 구한다.

단계 1. 정수배 화소 단위에서 SAD 값이 가장 작은 점 (x,y)를 구한다(그림 4 참조). SAD 값의 분포특성을 이용하여 (x,y)의 ±1 내에 있는 이웃 정수 화소 점에서 SAD를 구하여 비교한다. 정수 화소 단위에서 두 번째로 SAD가 작은 점을 찾는다.

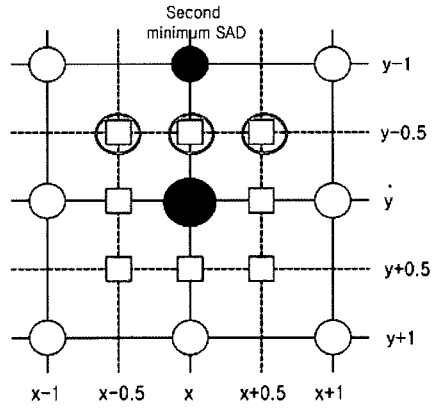

그림 4 제안한 1/2화소 단위에서의 움직임 추정 과정

단계 2. 단계 1에서 찾아진 두 번째로 SAD가 작은 방향과 같은 방향의 1/2 화소 점과 영상의 수평이동이 잦은 특성을 이용하기 위하여 수평위치에 있는 점들에서도 움직임 추정을 한다.

(i) $(x,y-1)$의 수직위치가 두 번째로 작은 SAD를 갖는 정수 화소 점일 경우, $(x,y-0.5)$, $(x\pm0.5,y-0.5)$의 세 점에서의 1/2 화소단위의 SAD값을 계산한다. $(x,y+1)$이 두 번째로 작은 SAD를 갖는 화소일 경우에도 비슷하게 처리한다.

(ii) SAD값이 두 번째로 작은 위치가 $(x\pm1,y)$로 수평위치일 경우, $(x+0.5,y)(x-0.5,y)$의 총 두 점에서 1/2 화소 단위에서의 SAD를 계산한다.

(iii) $(x+1,y-1)$과 같은 대각선 위치의 점에서 SAD가 두 번째로 작은 경우, 대각선 방향의 반 화소 점 $(x+0.5,y-0.5)$,$(x-0.5,y+0.5)$과 최저점 주변의 두 점 $(x,y-0.5)$,$(x+0.5,y)$등 총 4점에서 SAD를 계산하여 움직임 벡터를 구한다. 다른 방향의 대각선 방향일 경우에도 비슷한 방법을 적용한다.

단계 3. 1/2 화소 단위의 세 점에서 얻은 SAD값 중 가장 작은 값과 기존 정수 화소 단위에서의 SAD값을 비교하여, 1/2 화소 단위에서의 움직임 벡터 값을 찾을

수 있다.

1/2 화소 단위에서의 움직임 추정은 수직 위치의 탐색 점은 3점이고, 수평 위치에서의 탐색 점은 2점이며, 대각선 위치의 탐색 점은 4점이 된다. 따라서 제안한 알고리듬에서는 평균 3점정도 탐색한다.

모든 경우에 3점을 탐색하도록 할 수도 있으나, 실험 결과 위의 방법이 더욱 효과적이었다. 예를 들어 위의 단계 2)에서 (i)의 경우에는 제안한 방법과 같이 탐색하고, (ii)의 경우 중에 $(x-1,y)$에서 SAD 값이 두 번째로 작을 때에는 $(x-0.5,y),(x-0.5,y\pm0.5)$의 세 점을 탐색하고, (iii)의 경우에는 $(x+0.5,y-0.5),(x+0.5,y),(x,y-0.5)$의 세 점을 탐색하는 방법을 (3,3,3)방법이라 하고, 제안한 방법을 (3,2,4)방법이라고 하자. 두 방법의 성능은 표 2에 보인 바와 같이 PSNR은 같으나, Bit-rate에서 제안한 (3,2,4)방법이 약간 우수하다.

표 2 제안한 (3,2,4)방법과 (3,3,3)방법의 성능 비교

Sequence		PSNR (dB)	Bit-rate (Kbit/s)
Akiyo_qcif	Proposed (3,2,4) method	40.3	38.6
	(3,3,3) method	40.3 (-0.0)	38.9 (+0.3)
Salesman_qcif	Proposed (3,2,4) method	38.4	65.8
	(3,3,3) method	38.4 (-0.0)	65.9 (+0.1)
Claire_qcif	Proposed (3,2,4) method	40.4	40.5
	(3,3,3) method	40.4 (+0.0)	40.3 (-0.2)

3.2 새로운 1/4 화소 단위에서의 고속 움직임 추정

1/2 화소 단위에서 찾아진 점 주위의 1/4 화소단위의 8개의 점 중, SAD 오차 표면특성을 이용하여, 평균적으로 3개의 점에서 탐색을 하였다. 다음 그림 5와 같이 1/2 화소 단위에서 찾은 위치가 $(x,y-0.5)$로 정수 화소 단위에서 두 번째로 작은 SAD값을 가지는$(x,y-1)$방향과 같을 때, 1/2 화소 단위에서의 움직임 추정 방법과 같이, 수평위치에서 $(x,y-0.75)$, $(x\pm0.25,y-0.75)$점을 선택하여 탐색을 시작한다.

1/2 화소 단위에서 찾은 위치가 정수화소 단위에서 두번째로 작은 SAD를 가지는 점과 다를 때, 즉, SAD가 두 번째로 작은 위치가 $(x,y-1)$ 방향이고, 1/2 화소 단위에서 구해진 SAD가 가장 작은 위치가 $(x+0.5,y-0.5)$로 다른 방향이라면, 1/4화소 위치에서는 1/2 화소 단위에서 구해진 점 방향으로의 세 점 $(x+0.5,y-0.75)$, $(x+0.75,y-0.5)$,$(x+0.75,y-0.75)$에서 SAD를 계산한다. 이는 더 작은 화소 단위인 1/2 화소 단위에서 찾아진 SAD 방향이 정수 화소에서의 방향 보다 1/4 화소 단위에서의 움직임 벡터를 탐색하는데 있어 더 큰 영향을 주기 때문이다. 1/2화소와 1/4 화소 단위에서의 SAD 값 연산 시, 매 화소 값마다가 아닌 블록 단위의 한 행을 기준으로 계산하며, 계산 도중 최소 SAD값을 넘는 값이 나올 때는 연산을 중지함으로써 연산과정에서 연산 효율을 높였다. 제안한 방법으로의 1/2화소와 1/4 화소단위에서의 움직임 추정은 탐색 점을 줄여 연산의 복잡도를 줄이면서 빠른 처리가 가능한 결과를 확인하였다.

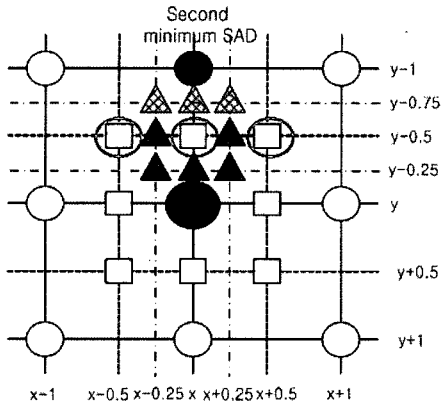

그림 5 제안한 1/4화소 단위에서의 움직임 추정 과정

4. 실험 결과

4.1 일반적인 움직임 추정 방법과의 성능 비교

본 논문에서 제안한 방식의 성능을 테스트하기 위해 QCIF 포맷의 시퀀스 Akiyo, Salesman, Claire, Hall, News를 H.264 부호화 환경 아래에서 실험 하였다. 각각의 영상이 움직임 추정 과정에서 사용된 블록의 크기는 4×4, 4×8, 8×4, 8×8, 16×8, 8×16, 16×16의 7종류의 가변 블록을 지원하였으며, 탐색 영역은 33×33 사이즈가 사용되었다. 각 시퀀스는 20 프레임 단위로 실험 하였고, 사용된 서버사양은 CPU는 Opteron 2.5 GHz 이고 메모리는 4G를 지원한다.

실험은 JM8.2 오픈 Source를 이용하여 구현하였다. JM8.2 Source 상에서는 Hardamard 변환이 사용 되어질 경우, 정수 단위에서 찾아진 SAD가 가장 작은 점에서 반 화소 단위의 SAD 계산을 다시 한번 하게 되는 특성이 있다. 즉, 그림 5의 (x,y)점에서 다시 SAD계산을 한다. 따라서 다음 결과에 나오는 실험치는 위와 같은 JM8.2 특성으로 인하여 계산 수치와 약간 다를 수 있다. 정수 화소 단위에서는 JM8.2에 구현된 고속 탐색 알고리즘을 그대로 사용하였고, 1/2과 1/4 화소 단위에서 탐색 점을 줄이는 제안한 방법과 기존의 8개의 화소

표 3 JM8.2와의 제안한 ME 전체 처리 과정에 대한 성능 비교

	Akiyo_qcif		Salesman_qcif		Claire_qcif		Hall_qcif		News_qcif	
	JM8.2	Proposed	JM8.2	Proposed	JM8.2	Proposed	JM8.2	Proposed	JM8.2	Proposed
PSNR(dB)	40.6	40.6 (-0.0)	38.5	38.5 (+0.0)	40.7	40.7 (-0.0)	39.3	39.3 (-0.0)	39.0	39.1 (+0.1)
Bit-rate (Kbit/s)	43.9	43.8 (-0.1)	113.4	115.0 (+1.6)	48.9	49.1 (+0.2)	47.9	48.0 (+0.1)	93.4	94.6 (+1.2)
Encoder처리 시간(sec)	44.9	35.4 (78.8%)	42.1	34.1 (81.0%)	45.0	34.8 (77.3%)	47.7	37.6 (78.8%)	47.2	38.5 (81.5%)
Sub-pel 연산처리 시간(sec)	8.03	5.05 (60.4%)	7.65	4.54 (57.1%)	7.91	4.51 (59.4%)	8.73	4.49 (57.1%)	8.59	4.90 (51.5%)

위치에서 전부 탐색하는 방법을 비교 하여, 그 결과를 표 3에 정리하였다. 실험 결과, 제안한 알고리듬은 기존 알고리듬과 비교해 화질을 판단할 수 있는 PSNR이 0.1dB 좋아지는 경우도 있으나 같은 성능으로 볼 수 있다. 비트율(Bit-rate)은 기존의 8점 탐색 방법보다 2% 이내에서 늘어나는 경우가 있으나 역시 비슷한 수준이다. 연산 시간은 기존의 8점의 1/2 화소 단위 탐색에 비해 인코더 처리 과정에서 약 20%정도의 감소하였다.

1/2 화소와 1/4 화소 단위에서의 제안한 움직임 추정 방법의 효율을 입증하기 위해서, 기존의 8점에서 탐색하는 부 화소 단위에서의 움직임 추정 결과와 제안한 3점 탐색 알고리듬의 부 화소 단위에서의 움직임 추정 결과를 비교하였다. 표 3에서 부 화소 단위의 연산처리 시간을 보면 제안한 3점 탐색 알고리듬이 부 화소 단위에서 기존의 8점 탐색 알고리듬보다 약 40%정도의 연산처리 시간의 감소가 있었다. 따라서 제안한 3점 탐색알고리듬은 기존의 8점 탐색 알고리듬과 PSNR에서 비슷한 성능을 가지며 부 화소 단위에서의 움직임 추정에서 40% 정도의 연산시간 감소를 보였다.

4.2 2SS 움직임 추정 방법과의 성능비교

최근 연구된 고속 1/2 화소 단위에서의 움직임 추정 기법인 2SS 알고리듬과 제안한 방법과의 실험을 통한 성능 비교 결과는 표 4에 나타내었다. 실험에 사용된 시퀀스는 Akiyo, Salesman, Claire, Hall 이고, 각각 100 프레임씩 사용하여 JM8.2상에서 검증하였다. 표 4를 보면, 1/2 화소 단위의 움직임 추정 과정에서, 제안한 3점 탐색 방법과 2SS방법은 화질 및 비트율면에서 비슷한 성능을 갖는다. 그러나 연산 처리 시간을 비교하면, 제안한 3점 탐색 방법은 8개의 1/2 화소 점에서 움직임 추정하는 일반적인 방법보다 60%의 연산 시간 감소가 있고, 2SS 보다 25% 이상의 연산시간 감소가 있다.

5. 결 론

최근 인터넷과 멀티미디어의 급속한 발전에서 제한된 전송선로와 저장 공간에 보다 많은 양의 정보를 전달하고 저장하는 기술인 영상 압축 기술의 발달은 큰 기여를 하였다. 압축효율이 뛰어난 H.264 인코더의 주된 연산을 차지하는 움직임 추정 과정에서 보다 효율적인 처리가 가능하도록 낮은 복잡도의 1/2 화소와 1/4화소 단위에서의 움직임추정 방법을 제안하였다. 이는 SAD의 오차 표면 특성과 화소의 보간 특성을 이용하여 부 화소 단위에서의 오차 값이 작은 방향을 예측하여 움직임 벡터를 찾는 방법이다. 예측된 방향으로만 탐색하여 기존의 8개의 탐색 점을 평균 3점으로 줄였다. 제안한 방법으로 5개의 영상을 JM8.2를 통해 실험한 결과, 일반적인 움직임 추정 방법과 비슷한 성능을 가지면서 인코더 처리 과정에서 약20%의 연산 시간의 감소가 있었고, 부 화소 단위에서만의 움직임 추정 처리과정에서는 약 40%정도의 연산 시간 감소가 있었다. 또한, 1/2 화소 단위의 움직임 추정 방법인 2SS 알고리듬과의 비교에서, 1/2 화소 단위의 움직임 추정에 있어 25%이상의 빠른 속도로 처리함을 확인 하였다. 따라서 본 논문에서 제안한 방법이 효율적인 움직임 추정 기법임을 확인하였다.

표 4 2SS 방법과 제안한 방법과의 움직임 추정 성능 비교

	Akiyo_qcif			Salesman_qcif			Claire_qcif			Hall_qcif		
	JM8.2	2SS	Proposed	JM8.2	2SS	Proposed	JM8.2	2SS	Proposed	JM8.2	2SS	Proposed
PSNR(dB)	40.3	40.3 (-0.0)	40.3 (-0.0)	38.4	38.4 (-0.0)	38.4 (-0.0)	40.4	40.4 (-0.0)	40.4 (-0.0)	39.2	39.1 (-0.1)	39.2 (-0.0)
Bit-rate (Kbit/s)	37.1	38.1 (+1.0)	38.6 (+1.5)	63.8	65.0 (+1.2)	65.8 (+2.0)	38.5	42.0 (+3.5)	40.5 (+2.0)	59.5	59.8 (+0.3)	60.7 (+1.2)
Half-pel 연산처리 시간(sec)	21.2	14.9 (70.3%)	8.87 (41.8%)	22.4	14.4 (64.3%)	9.46 (42.2%)	22.2	14.3 (64.4%)	8.81 (39.7%)	21.8	14.2 (65.1%)	8.58 (39.3%)

참 고 문 헌

[1] B. Zhou and J. 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.

[2] H. M. Wong, O. C. Au, J. Huang, S. Zhang and W. N. Yan, "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.

[3] J. Zhang, Y. He, S. Yang and Y. 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.

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

[5] C. Du, Y. He and J. 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.

[6] 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.

[7] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., Vol. 4, pp. 438-442, Aug. 1994.

[8] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion esitmation," IEEE Trans. Circuits Syst. Video Technol., Vol. 6, pp. 313-317, June 1996.

[9] C.-H. cheung and L.-M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. video Technol., Vol. 12, pp. 1168-1177, Dec. 2002.

이 윤 화

2002년 한라대학교 전자공학과 학사. 2004년 한양대학교 전자전기제어계측공학과 석사과정. 관심분야는 H.264, MPEG-4, ASIC설계, 저전력 설계

최 명 훈

2003년 한양 대학교 전자컴퓨터 공학부 학사. 2005년 한양 대학교 전자전기제어계측 공학과 석사. 관심분야는 H.264, VLSI, 반도체 설계

신 현 철

1978년 서울대학교 전자공학과 학사. 1980년 한국과학기술원 전기 및 전자공학 석사 1983년~1987년 U.C. Berkeley Ph.D. 1983년~1987년 Fulbright scholarship. 1987년~1989년 MTS, AT&T Bell Lab's, Murray Hill N.J., USA. 1989년~현재 한양대학교 전자컴퓨터공학부 교수. 1997년~현재 IDEC 한양대학교 지역센터 센터장. 관심분야는 CAD&VLSI, 통신용 멀티미디어 반도체 설계, 저전력 설계