H.264 동영상 표준 부호화 방식을 위한 고속 움직임 추정 기법

학생회원 윤 성 현*, 최 권 열*, 정회원 이 성 수*, 홍 민 철*

Fast Motion Estimation Algorithm for H.264 Video Coding Standard

Sung-Hyun Yoon*, Kwon-Yul Choi* ***Student Members,***
Seongsoo Lee*, Min-Cheol Hong* ***Regular Members***

요 약

본 논문에서는 H.264 동영상 표준 부호화 방식을 위한 고속 움직임 추정 기법을 제안한다. 제안된 고속 움직임 추정 기법은 움직임 벡터가 국부적으로 일정한 상관관계를 유지하고 있다는 특성을 이용하여 각 블록의 움직임 추정 시 부호화된 인접 블록의 움직임 벡터 값으로부터 주어진 탐색 영역 내에서 수평, 수직 방향 각각 독립적으로 불필요한 탐색 영역을 제거하여 축소된 비대칭 가변 크기 탐색 영역을 결정한다. 또한, 예측된 탐색 영역 제약 조건을 이용하여 가변 단계 탐색 방식을 취하는 것을 특징으로 한다. 실험 결과를 통해 제안된 방식에 의해 움직임 추정 탐색 지점이 전영역 탐색 방식(Full Search)에 비해 평균 98% 이상 절감 되었으며, PSNR과 비트율은 평균적으로 전영역 탐색 방식과 동일 수준인 것을 결과로부터 확인 할 수 있었다.

Key Words : H.264, fast motion estimation, local statistics, variable step search.

ABSTRACT

In this paper, we propose fast motion estimation algorithm. Local statistics of a motion vector is highly correlated to motion vectors of its neighboring blocks. According to the property, block-based motion search range is adaptively determined in order to reduce unnecessary search points. Based on the determined search range, motion vector is obtained by variable step search motion estimation. Experimental results show that comparing to Full search motion estimation, the motion searching points of proposed algorithm is reduced as much as 98%. Moreover, PSNR and Bit Rate are almost same to Full search method.

Ⅰ. 서 론

일반적인 디지털 동영상 정보는 정보량이 방대한 이유로 부호화 방식에 대해 많은 연구가 진행되어 왔으며, 특히 1988년에 디지털 정보의 부호화 및 저장에 대한 표준 규격의 필요성이 대두되면서 ITU에서는 유무선 통신망 환경에서 동영상 서비스를 위한 표준 규격 제정을 위해 노력해 왔다. 이동망과 같은 새로운 통신 채널의 급속한 보급에 따라 기존 압축방법에 비해 압축률이 더욱 향상된 동영상 부호화 방식의 필요성이 증대되었다. H.264 동영상 표준 부호화 방식은 기존의 동영상 압축 방식보다 2배 이상의 압축률을 제공하므로, 동일 전송 채널폭을 사용할 경우 개선된 화질을 제공할 수 있다.

H.264 동영상 표준 부호화 방식의 높은 압축률 제공은 여러 기술에 채택되어 다양한 응용분야에

* 숭실대학교 정보통신전자 공학부 ({xizang, tantis}@vipl.ssu.ac.kr, {sslee, mhong}@e.ssu.ac.kr),
논문번호 : KICS2005-09-379, 접수일자 : 2005년 9월 21일
* 본 연구는 한국 학술진흥재단 협동연구 지원사업 (KRF-2004-042-D00152) 지원으로 이루어 졌음.

사용되는 기술적 동향에도 불구하고 높은 압축률에 따르는 계산량 증가의 문제로 인해 실시간 구현에 어려움이 따른다.

기본적으로 H.264 동영상 표준 부호화 방식의 전영역 탐색 방식에 따른 탐색 영역은 그림 1과 같이 결정된다. 탐색 원점의 cx, cy는 부호화된 인접 블록의 움직임 벡터 중간값(median)으로 결정 된다[1-3]. 이와 같은 탐색 원점의 이동은 움직임 벡터의 정확도를 높이고 전송할 움직임 벡터 정보를 줄일 수 있는 장점이 있다. 그림 1에서와 같이 ω의 탐색 영역 크기를 갖는 경우 탐색 영역 내 정화소 수는 $(2\omega+1)^2$이 된다. H.264 동영상 표준 부호화 방식의 움직임 추정(motion estimation)은 1/4화소 위치까지 움직임을 추정한다. 그림 2에서와 같이 결정된 정화소를 중심으로 8개의 1/2화소 위치에서 움직임을 추정하고 이후 결정된 위치를 중심으로 8개의 1/4화소 위치에서 움직임을 추정한다.

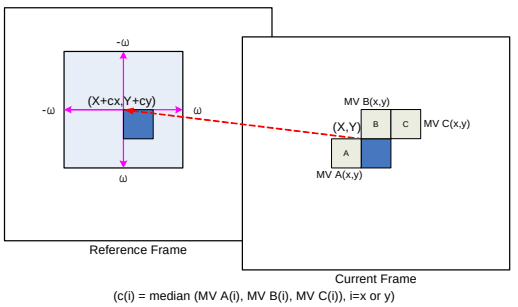

그림 1. H.264 전영역 탐색 방식의 탐색 영역

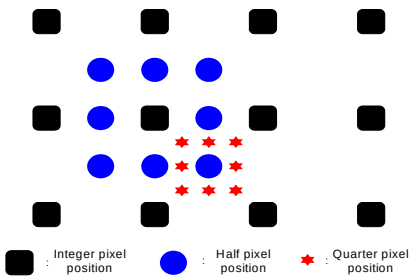

그림 2. H.264 부화소 움직임 추정 위치

H.264 동영상 표준 부호화 방식은 가변크기 블록 움직임 추정 방식으로 인하여 매크로 블록(macro block)에 대한 반복적인 움직임 추정 연산으로 압축 효율은 개선되었지만 계산량이 증가하는 문제점이 나타난다. 매크로 블록을 세분화하는 각 크기별 모드(mode)는 그림 3과 같다[1-3].

움직임 추정 과정의 과다한 연산량을 절감하기 위해 많은 고속 움직임 추정 방식이 개발되어 왔다. 대표적인 고속 움직임 추정 방식은 TSS(Three Step Search), FSS(Four Step Search)방식 등이 있으며[4-6] 상기 방식 방식들의 변형된 형태가 주를 이루고 있다[7-9]. 기존의 고속 움직임 추정 방식은 전영역 탐색 방식의 계산량 문제를 크게 개선시킬 수 있으나 움직임 벡터 오류에 의해 압축 효율성이 크게 떨어지는 단점이 있다. 또한, 움직임 추정 영역을 적응적으로 결정하여 불필요한 움직임 벡터 후보군을 제거하는 움직임 탐색 결정 방식과 관련된 기법들이 제안되었다[10,11]. 상기 방식들은 움직임 탐색 영역 결정시 움직임 벡터의 국부 통계 특성을 반영하지 않기 때문에 움직임 벡터의 정확성이 감소되어 압축 효율이 저하되는 단점을 갖고 있다.

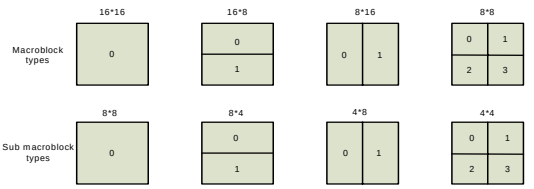

그림 3. H.264의 모드별 블록형태

H.264 동영상 표준 부호화 방식의 부호화 과정에서 움직임 추정 과정은 전체 부호화 시간의 약 60~70%를 차지한다. 움직임 추정 과정의 계산량 감소 기법 연구는 막대한 계산량을 가지는 H.264 동영상 표준 부호화 방식의 실시간 구현을 가능하게 하여 다양한 멀티미디어 시스템에 적용 가능하기 위한 필수적인 기법이라 할 수 있다.

본 논문에서는 인접된 블록 간의 움직임 벡터의 국부 통계 특성이 유사한 특성을 이용하여 움직임 추정을 위한 블록의 탐색 영역을 결정하고, 결정된 움직임 탐색 영역 내에서 TSS 방식의 변형된 형태인 다단계 탐색 과정을 취하여 전영역 탐색 움직임 추정 방식과 유사한 압축 효율을 유지한 고속 움직임 추정 방식에 대해 기술한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 적응적 탐색 영역 결정 방식과 가변 단계 움직임 추정 방식을 결합한 고속 움직임 추정 기법에 대해 설명하고, Ⅲ장에서 제안된 기법에 의한 실험 결과를 보이며, Ⅳ장에서는 결론을 맺는다.

Ⅱ. 제안 방식

2.1 적응적 탐색 영역 결정 방식

영상의 움직임 벡터는 대부분 인접한 블록의 움

직임 벡터와 유사한 움직임을 갖는 특성을 나타낸다. 기존의 움직임 추정 방식은 미리 지정된 정방형의 움직임 탐색 영역 내에서 움직임 추정을 하게 되므로 불필요한 위치를 탐색하여 연산량이 증가하는 결과를 나타낸다.

이러한 불필요한 계산을 감소시키기 위해 움직임 추정 대상 블록의 인접 블록들과의 국부 움직임 벡터의 통계적 특성을 이용하여 수평 수직 방향 각각의 움직임 추정 탐색 영역을 결정하였다. 인접 블록들의 위치는 그림 4와 같다. 가변크기 블록을 사용하는 H.264 동영상 표준 부호화 방식에 따라 각 모드(mode)별 인접 블록을 결정하였으며, 부호화 블록의 위치에 따라 존재하지 않는 인접 블록이 발생하는 경우 존재하지 않는 블록의 움직임 벡터 크기 값은 주어진 탐색 영역 크기로 대체 하였다. 예외적으로, 블록 C가 존재하지 않고 블록 D는 존재하는 경우 블록 C의 움직임 벡터 값을 블록 D의 값으로 대체하여 사용한다. 그림 4에서 블록 E는 현재 부호화 대상 블록이며 좌상측 좌표 위치와 우하측 좌표 위치를 표기하였다. 이때 $block\ size_h$는 해당 모드에 대한 수평 방향의 크기 값이며, $block\ size_v$는 해당 모드에 대한 수직 방향의 크기 값이다. 블록 A, B, C는 블록 E와 상관관계가

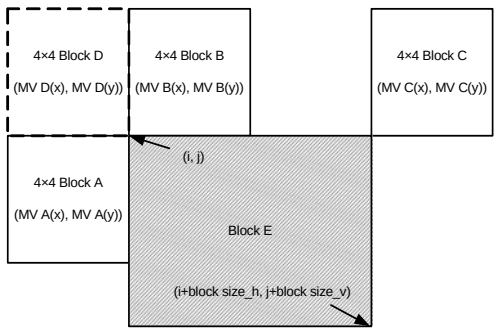

그림 4. 부호화 대상 블록과 인접 블록 위치

가장 높은 인접 4×4 크기 블록으로 각 블록은 수평 수직 방향의 움직임 벡터가 결정된 상태이다. 주어진 탐색 영역 크기를 식 (1)과 같이 정의 한다.

$$search\ range(i) \le |\omega|, \quad i = x, y \tag{1}$$

식 (1)의 ω는 움직임 추정 시 미리 정의된 최대 움직임 탐색 영역 값을 의미하며, x 및 y는 수평 및 수직 방향을 의미한다.

결정된 각각의 움직임 벡터를 이용하여 식 (2)와 같이 정의 할 수 있다.

$$MV_{\max}(i) = 2 \times \max(|MV\ A(i)|, |MV\ B(i)|, |MV\ C(i)|) \tag{2}$$

식 (2)의 $MV_{\max}(i)$는 움직임 벡터 추정 오류를 줄이기 위해 인접 블록 움직임 벡터 최대치를 두 배하여 오류를 줄이기 위한 최소 탐색 영역 크기 값으로 정의한다.

$$\alpha(i) = |MV\ A(i)| + |MV\ B(i)| + |MV\ C(i)| \tag{3}$$

식 (3)은 인접 영역의 상관도와 움직임 벡터의 크기를 추정할 수 있는 척도로 사용된다. 즉, $\alpha(i)$가 작은 경우 인접 블록이 가질 수 있는 움직임 벡터의 크기 종류가 제한되므로 상대적으로 상관도가 높으며, 현재 부호화 대상 블록의 움직임 또한 작은 영역 내에서 일어 날 수 있다고 추정할 수 있다. $\alpha(i)$가 큰 경우 반대로 각각의 인접 블록이 가질 수 있는 움직임 벡터의 크기 종류가 다양해지므로 상관도가 낮아지며, 현재 부호화 대상 블록의 움직임이 다양한 크기와 큰 움직임 벡터를 가질 수 있다고 추정할 수 있다. 식 (3)에 의한 $\alpha(i)$ 값에 의해 식 (4) 와 같이 주어진 탐색 영역을 국부 통계 특성을 반영하는 두 가지 탐색 영역 중 하나로 결정 한다.

$$\beta(i) = \begin{cases} \dfrac{\omega + 4}{8}, & for\ \ \alpha(i) < 2 \\ \dfrac{\omega + 2}{4}, & otherwise \end{cases} \tag{4}$$

식 (2)에서 구해진 $MV_{\max}(i)$는 현재 부호화 대상 블록의 예상되는 움직임 벡터를 찾을 수 있는 최소 탐색 영역의 크기이므로 움직임 벡터 추정 오류를 최소화하기 위한 영역을 다음과 같이 정의한다.

$$\gamma(i) = \max(\beta(i), MV_{\max}(i)) \tag{5}$$

식 (5)에 의해 결정된 탐색 영역은 식 (1)에서 정의된 움직임 탐색 영역의 제약 조건을 위배할 수 있으므로 식 (6)과 같이 ω와 $\gamma(i)$ 의 크기 비교에 의해 작은 값을 선택하게 된다.

$$\delta(i) = \min(\omega, \gamma(i)) \tag{6}$$

따라서 수평 및 수직 방향의 탐색 영역 dx, dy는 식 (7)과 같이 결정된다.

$$-\delta(x) \le dx \le \delta(x), \quad -\delta(y) \le dy \le \delta(y) \tag{7}$$

2.2 다단계 고속 움직임 추정 기법

식 (7)에 의해 결정된 수평 및 수직 방향의 움직임 탐색 영역을 이용한 적응적 탐색 영역 결정 방식 (ASRD; Adaptive Search Range Decision)을 이용한 전영역 탐색 방식은 아래와 같은 SAD 값을 최소화하는 지점을 움직임 벡터로 결정하게 된다.

$$SAD_{(m,n)} = \sum_{i=0}^{U-1}\sum_{j=0}^{V-1}|f_t(X,Y) - \hat{f}_{t-1}(X+dx, Y+dy)| \tag{8}$$
$$(\text{for } -\omega \le dx, dy \le \omega,\ X = U\times m + i,\ Y = V\times n + j)$$

식 (8)에서 U, V는 움직임 추정을 위한 블록의 크기를 의미하고, m, n은 움직임 추정 블록의 위치를 나타낸다. 식 (8)의 ASRD에 의한 움직임 벡터 결정 방식은 식 (7)에 의해 결정된 움직임 탐색 영역 내의 불필요한 지점을 연산 과정에 포함시키게 되므로 연산량 절감에 한계가 있다. 연산량의 절감을 위해 식 (7)을 기반으로 TSS의 변형 형태인 다단계 움직임 탐색 과정을 수행하게 된다.

본 논문에서 제안한 다단계 탐색 방식은 다음과 같이 수행된다. 식 (7)에서 결정된 ASRD를 이용하여 첫 단계 탐색 크기를 다음과 같이 결정한다.

$$k_{first} = \max\{2^{M} < \max(\delta(x), \delta(y))\} \tag{9}$$

식 (9)의 M은 정수를 의미하며, 탐색 단계 수는 $(M+1)$이 된다. 이후 각 단계의 탐색 크기는 식 (10)과 같이 전단계의 1/2 크기를 사용한다.

$$k_t = (k_{t-1})/2 \tag{10}$$

움직임 벡터 결정은 그림 5와 같이 TSS와 유사하게 이루어지며, 제안 방식의 특성은 $\max(\delta(x), \delta(y))$의 값이 작을수록 인접된 블록들의 움직임과 움직임 추정을 수행하는 블록간의 상관관계가 크므로 탐색 단계 및 탐색 지점들의 수는 작아지며, 상기 값이 클수록 인접 블록들과의 상관관계가 작으므로 탐색 단계 및 탐색 지점들의 수가 커지게 된다. 탐색 단계는 $k_t = 1$이 될 때까지 진행한 후 종료하게 된다. 단, 각 단계에서 그림 5에 사선으로 표시된 영역과 같이, 식 (7)에 의해 결정된 영역을 벗어나는 위치는 탐색 지점에서 제외한다.

상기와 같이 제안된 고속 움직임 추정 방식은 인접 블록들의 움직임 벡터의 통계적인 특성을 이용하여 탐색 영역을 결정하고, 결정된 탐색 영역에서 탐색 지점을 축소하는 것을 특징으로 한다.

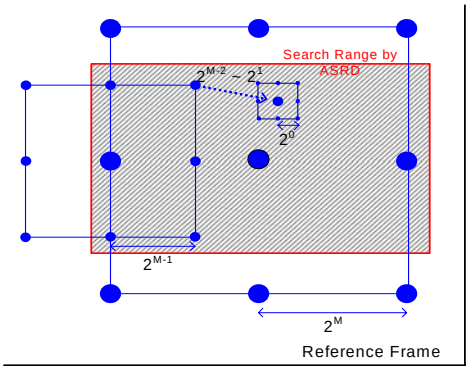

그림 5. 고속 움직임 추정 기법

Ⅲ. 실험 결과

본 논문에서 실험은 다양한 영상을 여러 양자화 크기 값(QP)에 대해 실험하였다. JM 9.0 Baseline Profile Level 3.0을 기준으로 RD(Rate Distortion) optimize off 설정으로 실험하였으며, 주어진 움직임 탐색 영역이 32인 경우에 대한 QCIF Foreman, Container, Stefan, Claire 영상을 10 frames/sec로 부호화한 실험 결과를 기술한다.

제안 방식을 Full Search 움직임 추정 방식 및 기존의 TSS 방식과의 성능 비교를 위해, 주어진 움직임 탐색 영역 크기가 32인 경우 TSS를 적용한 Fixed Step Search (움직임 탐색 영역이 32인 경우 5 Step Search)와 비교하였으며, 제안된 방식을 ASRD만 사용한 경우와 ASRD 및 다단계 움직임 추정 방식을 결합시킨 경우에 대해서도 비교하였다.

성능 비교를 위해 PSNR (Peak Signal to Noise Ratio)를 사용하였으며, $M\times N$ 크기의 8bits 영상에 대하여 식 (11)과 같이 표현된다.

$$PSNR = 10\log\frac{MN\times 255^{2}}{\|f-\hat{f}\|^{2}} \tag{11}$$

식 (11)의 f 및 $\hat{f}$은 각기 원영상 및 부호화된 영상을 의미하며, $\|\cdot\|$은 유클리디안놈(Euclidean norm)을 나타낸다.

표 1은 전체 99개 인터(inter) 프레임 부호화 시 움직임 추정 소요 시간을 나타낸다. 수치적으로 다른 방법 보다 제안된 고속 움직임 추정 기법이 가

장 적은 움직임 추정 시간을 소요하는 것을 확인할 수 있다. 또한, QP 값이 커짐에 따라 비교 방식보다 움직임 추정 소요시간이 절감되고, 영상의 움직임 정도가 작을수록 제안 방식에 의한 소요시간 이득이 커짐을 확인할 수 있었다.

표 2 및 3에는 QP 함수에 따른 PSNR 및 비트율 비교를 나타내었다. 전반적으로 Full Search 방식과 비교하여 유사한 성능을 보이지만, QP 값이 커짐에 따라 움직임 벡터 추정 오류에 의한 residual 값이 증가하여 비트율의 증가가 다소 이루어졌다. 반면에 Fixed Step Search 방식과는 유사한 성능을 보임을 확인할 수 있었다.

표 1. QP 함수에 대한 움직임 추정 소요 시간 비교 (99 frames, [sec])

QP		16	24	32	40
Foreman	Full Search	233.98	206.50	162.18	126.60
	ASRD	73.90	67.10	56.19	43.04
	Fixed Step	5.63	4.88	4.53	4.14
	제안방식	4.07	4.30	3.80	2.74
Container	Full Search	156.91	141.85	104.61	77.45
	ASRD	29.35	27.85	22.27	15.22
	Fixed Step	3.71	3.21	3.31	2.69
	제안방식	2.52	2.45	2.10	1.52
Stefan	Full Search	307.29	283.49	240.88	193.69
	ASRD	110.39	103.16	90.71	73.65
	Fixed Step	6.02	5.00	4.90	4.65
	제안방식	5.77	5.48	4.49	4.08
Claire	Full Search	92.55	76.47	55.02	45.20
	ASRD	17.56	14.22	9.78	8.69
	Fixed Step	3.39	2.44	2.29	2.04
	제안방식	1.80	2.14	1.44	1.50

표 2. QP 함수에 대한 PSNR 비교 (dB)

QP		16	24	32	40
Foreman	Full Search	45.01	38.61	33.06	28.04
	ASRD	45.02	38.60	33.04	28.03
	Fixed Step	44.98	38.57	32.99	27.99
	제안방식	44.98	38.56	32.99	27.98
Container	Full Search	45.00	38.60	33.11	27.85
	ASRD	44.99	38.60	33.08	27.83
	Fixed Step	44.99	38.60	33.09	27.84
	제안방식	44.99	38.60	33.07	27.82
Stefan	Full Search	44.58	37.50	30.62	24.76
	ASRD	44.58	37.49	30.61	24.76
	Fixed Step	44.55	37.43	30.54	24.70
	제안방식	44.55	37.43	30.54	24.72
Claire	Full Search	47.75	42.56	36.89	31.31
	ASRD	47.75	42.55	36.88	31.30
	Fixed Step	47.73	42.52	36.88	31.28
	제안방식	47.74	42.50	36.83	31.21

표 3. QP 함수에 대한 비트율 비교 (kbits/sec)

QP		16	24	32	40
Foreman	Full Search	340.11	127.16	46.09	18.30
	ASRD	335.76	125.98	46.17	19.09
	Fixed Step	342.74	130.21	48.25	19.29
	제안방식	338.72	128.93	48.00	19.59
Container	Full Search	203.64	54.84	14.59	5.15
	ASRD	203.15	55.04	14.58	5.15
	Fixed Step	203.42	54.69	14.62	5.19
	제안방식	203.51	54.70	14.59	5.14
Stefan	Full Search	701.21	332.78	125.40	42.40
	ASRD	695.27	330.61	125.52	42.78
	Fixed Step	719.22	351.42	135.27	47.17
	제안방식	715.94	347.29	134.54	47.06
Claire	Full Search	102.10	35.21	11.59	4.33
	ASRD	101.81	35.16	11.57	4.36
	Fixed Step	102.29	35.55	11.58	4.34
	제안방식	101.60	35.30	11.66	4.33

표 4. QP 함수에 대한 화소당 평균 SAD 비교

QP		16	24	32	40
Foreman	Full Search	36.42	48.53	75.21	122.64
	ASRD	36.73	48.80	75.83	123.74
	Fixed Step	38.92	50.37	76.99	124.27
	제안방식	38.52	50.02	77.07	124.75
Container	Full Search	26.51	40.70	66.56	111.87
	ASRD	26.51	40.76	67.01	112.34
	Fixed Step	26.69	40.88	67.07	112.21
	제안방식	26.67	40.86	67.28	112.71
Stefan	Full Search	71.62	80.30	111.05	182.08
	ASRD	73.18	81.98	112.21	182.92
	Fixed Step	88.86	96.00	120.82	187.62
	제안방식	88.30	95.32	120.21	186.66
Claire	Full Search	16.02	22.91	38.89	69.45
	ASRD	16.09	23.08	39.29	70.52
	Fixed Step	16.32	23.23	39.24	70.34
	제안방식	16.24	23.28	39.65	71.31

움직임 벡터의 정확도을 비교하기 위해 표 4에 움직임 추정을 위한 화소당 평균 SAD 값을 나타내었다. Full Search 방식과 비교하여 ASRD 방식이 가장 우월한 움직임 벡터 정확성을 나타내었으며, 대부분의 경우에 제안된 ASRD와 다단계 움직임 추정 기법이 Fixed Step Search 방식보다 움직임 벡터 추정 성능이 우수함을 확인할 수 있었다.

표 5에 화면당 움직임 탐색 지점 수에 대한 비교를 나타내었다. Full Search 방식 및 Fixed Step 방식은 영상의 움직임 정도 및 QP 값에 무관하게 탐색 지점 수가 변화되지 않으며, ASRD 방식은 QP 값이 커짐에 따라 움직임 벡터의 크기에 대한

표 5. QP 함수에 대한 화면당 움직임 탐색 지점 수

QP		16	24	32	40
Foreman	Full Search	17,218,278	17,218,278	17,218,278	17,218,278
	ASRD	3,538,559	3,404,439	3,226,041	2,849,780
	Fixed Step	235,422	235,422	235,422	235,422
	제안방식	184,937	183,670	181,265	175,248
Container	Full Search	17,218,278	17,218,278	17,218,278	17,218,278
	ASRD	1,764,449	1,738,423	1,713,680	1,704,731
	Fixed Step	235,422	235,422	235,422	235,422
	제안방식	151,231	148,747	146,744	146,208
Stefan	Full Search	17,218,278	17,218,278	17,218,278	17,218,278
	ASRD	3,539,637	3,399,399	3,206,095	2,921,268
	Fixed Step	235,422	235,422	235,422	235,422
	제안방식	203,760	202,852	199,896	195,203
Claire	Full Search	17,218,278	17,218,278	17,218,278	17,218,278
	ASRD	1,765,466	1,757,170	1,734,398	1,720,585
	Fixed Step	235,422	235,422	235,422	235,422
	제안방식	151,488	150,113	148,548	147,646

제약 조건으로 탐색 지점수가 작아지며, 영상간의 움직임 정도가 작아짐에 따라 탐색 지점 수가 현저하게 감소하는 것을 확인할 수 있었다. ASRD 방식을 기반으로 하는 다단계 움직임 추정 기법은 Full Search 방식의 0.85%~1% 정도의 탐색 지점이 요구되며, Fixed Step 방식과 비교하여 60%~85% 정도의 탐색 지점이 필요함을 확인할 수 있었다. 상기의 결과로부터 제안 방식은 Full Search 방식과 유사한 성능을 유지하면서 극소의 연산량을 필요로 하는 효과적인 방식임을 확인할 수 있었다.

Ⅳ. 결 론

본 논문에서는 H.264 동영상 표준 부호화 방식을 위한 고속 움직임 추정 기법에 관하여 제안하였다. 움직임 추정을 위한 탐색 영역을 인접 블록의 움직임 벡터 크기 값을 이용하여 적응적으로 결정하여 움직임 추정에 불필요한 영역을 줄이고, 적응적으로 결정된 영역의 가변성에 따라 단계를 조정하여 탐색 지점의 개수를 줄이는 가변 단계별 탐색 방식을 혼용한 기법이다.

실험 결과를 통해 제안된 방식은 Full Search 방식과 PSNR 및 비트율에서 매우 유사한 성능을 유지하며, 움직임 탐색 지점 수 및 움직임 추정에 소요되는 시간 등의 연산량이 현저하게 감소됨을 확인할 수 있었다. 실험 결과에는 기술되지 않았지만 고해상도 영상에서는 블록간의 상관관계가 커지므로 연산량의 이득이 더욱 커짐을 확인할 수 있었다. 상기와 같이 제안된 방식은 H.264 기반의 부호화부의 실시간 구현과 관련된 다양한 응용 분야에 활용 가능할 것으로 판단된다.

참 고 문 헌

[1] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*(ITU-T Rec. H.264|ISO/IEC 14496-10 AVC) JVT-G050, Geneva, Switzerland, 23-27 May, 2003.

[2] T. Wiegand, G. Sullivan, G. Bjontegaard and A, Luthar, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Circuit and Systems for Video Technology*, vol.13, pp.560-576, July, 2003.

[3] Iain E.G. Richardson, *H.264 and MPEG-4 Video* Compression, Wiley, 2003.

[4] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing." Proc. NTC81, pp.G5.3.1-5.3.5, Dec. 1981.

[5] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Tech.*, vol.4, no.4, pp.438-442, Aug. 1994.

[6] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits and Sys-

tems for Video Tech, vol.6, no.3, pp.313-317, June. 1996.

[7] 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.

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

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

[10] 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.

[11] 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.

윤 성 현 (Sung-Hyun Yoon) 학생회원

2003년 2월 숭실대학교 정보통신전자공학부 학사

2005년 8월 숭실대학교 정보통신공학과 공학석사

<관심분야> 영상 압축, 고속 움직임 추정

최 권 열 (Kwon-Yul Choi) 학생회원

1999년 3월 숭실대학교 정보통신전자공학부 입학, 재학중

<관심분야> 영상 압축, 복원

이 성 수 (Seongsoo Lee) 정회원

1991년 2월 서울대학교 전자공학과 학사

1993년 2월 서울대학교 전자공학과 석사

1998년 8월 서울대학교 전기공학부 박사

1998년 11월~2000년 3월 Research Associate, University of Tokyo

2000년 4월~2002년 8월 이화여자대학교 정보통신학과 연구전임강사

2002년 9월~현재 숭실대학교 정보통신전자공학부 조교수

<관심분야> 저전력 및 멀티미디어 SoC 설계, 저전력 및 멀티미디어 알고리즘 등

홍 민 철 (Min-Cheol Hong) 정회원

1988년 2월 연세대학교 전자공학과 (공학사)

1990년 8월 연세대학교 전자공학과 (공학석사)

1997년 9월 Northwestern University 전기및컴퓨터 공학과 (공학박사)

1990년 7월~1991년 8월 (주)LG 정보통신 (연구원)

1997년 9월~1998년 8월 Northwestern University (Research Fellow)

1998년 8월~2000년 2월 (주)LG 전자 (선임연구원)

2000년 3월~현재 숭실대학교 정보통신전자공학부 조교수

<관심분야> 영상복원, 비선형 영상처리 및 필터링, 차세대 동영상 압축 방식, Blind image deconvolution