일반논문-08-13-1-15

인접 블록 움직임 벡터의 지역적 통계 특성을 이용한 고속 움직임 추정 기법

김 기 범[a], 정 찬 영[a], 홍 민 철[a]‡

Fast Motion Estimation Using Local Statistics of Neighboring Motion Vectors

Kibeom Kim[a], Chan-Young Jeong[a], and Min-Cheol Hong[a]‡

요 약

본 논문에서는 인접 블록 움직임 벡터의 통계적 특성을 이용한 가변 탐색 스텝의 고속 움직임 추정 기법에 대해 제안한다. 인접 블록 움직임 벡터들 사이의 상관 관계를 이용하여 움직임 추정을 위한 탐색 영역을 적응적으로 결정하였으며, 이를 통해 불필요한 탐색 지점 수를 제거할 수 있었다. 이와 같이 결정된 탐색 영역을 기반으로 가변 탐색 스텝 움직임 추정을 적용하였으며 움직임 추정을 위한 연산량을 줄일 수 있었다. 실험 결과를 통해 제안 방식이 H.264 JM의 고속 전 대역 spiral 탐색 기법과 기타 고속 움직임 추정 방식과 비교하여 부호화 성능의 저하 없이 움직임 추정을 위한 탐색 지점 수 및 연산 량이 급격히 감소됨을 확인할 수 있었다.

Abstract

In this paper, we propose a variable step search fast motion estimation algorithm using local statistics of neighboring motion vectors. Using the degree of correlation between neighboring motion vectors, motion search range is adaptively adjusted to reduce unnecessary search points. Based on the adjusted search range, motion vector is obtained by variable search step. Experimental results show that the proposed algorithm has the capability to dramatically reduce the search points and computing cost for motion estimation, comparing to fast full spiral search motion estimation and other fast motion estimation.

Keywords: variable step search, motion estimation, local statistics, H.264, adjusted search range

I. 서 론

H.264 동영상 표준 부호화 방식은 기존 동영상 부호화 방식과 비교해 더 향상된 화질을 유지하는 동시에 더 높은 압축 효율을 얻기 위해 개발되었다[1]. H.264 동영상 표준 부호화 방식은 기존의 동영상 부호화 방식에 비해 우수한 압축 효율을 보이므로 DMB(Digital Multimedia Broadcasting), Mobile Phone 그리고, 초 해상도 영상 전송 같은 다양한영상 서비스에 활용될 것으로 기대된다. 부호화 성능 효율은 화면 내 인트라 예측, 4x4 블록 기반의 변환, 다양한 블록 크기의 움직임 추정 등의 새로운 기술들을 사용함으로써 가능해졌다[1,2,3]. H.264 동영상 표준 부호화방식이 우수한 압축률을 보임에도 불구하고, 이로 인해 발생되는 막대한 연산량과 복잡성으로 인해 최적화된 고속의 방식들의 연구가 요구된다. 특히, H.264의 움직임 추정은 그림1에서 보는 것과 같이 많은 계산량을 요구한다. 그러므로, 현재 상용화되고 있는 실시간 H.264 인코더의 부호화 성능을 개선하기 위해 저연산 및 저전력 움직임 추정 기법이 요구된다.

a) 숭실대학교 정보통신전자공학부
School of Electronic Engineering, Soongsil University
‡ 교신저자 : 홍민철(mhong@ssu.ac.kr)
* 본 연구는 서울시 산학연 협력사업(과제번호 10544) 및 숭실대학교 교내 연구비 지원 사업에 의해 수행되었음.

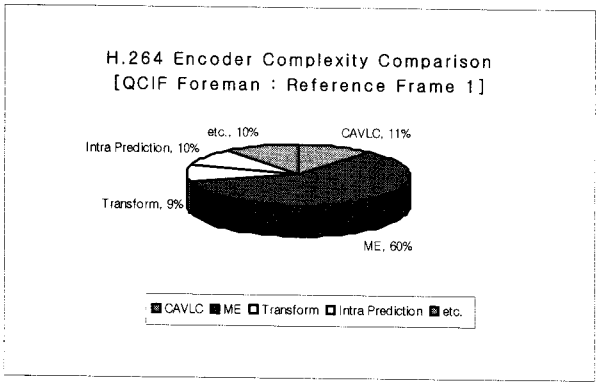


그림 1. H.264 부호화부 연산량 비교
Fig. 1. H.264 encoder complexity comparison

움직임 추정은 영상들 사이의 시간적 중복성을 제거하기 위해 사용되는 기법으로, 영상들의 화소 단위 또는 블록 단위로 수행될 수 있으며, 가장 정확한 움직임은 화소 단위로 움직임 벡터를 예측하여 획득될 수 있으나, 모든 화소에 대해 움직임 추정을 하는 이유로 많은 연산량을 요구하며, 노이즈에 의해 훼손된 화소인 경우 움직임 벡터 결정에 어려움이 있다[2].

BMA(Block Matching Algorithm)는 블록 내부의 모든 화소들은 동일한 움직임을 가진다는 가정하에 움직임 추정의 복잡성을 줄이기 위해 일반적으로 사용된다.

FS(Full Search) BMA는 참조 영상의 탐색 영역내부에 존재하는 모든 후보 탐색 지점을 비교 함으로써 최적의 움직임 벡터 예측을 보장한다. 그러나 이 방식은 많은 계산량을 요구하는 단점을 갖고 있다[2]. 움직임 추정의 복잡성 문제를 해결하기 위하여, TSS(Three Step Search), FSS(Four Step Search) 및 이와 유사한 많은 고속 움직임 추정 기법이 개발 되었다[2,4,5,6,7,8,9].

움직임 추정의 성능과 계산량, 그리고 전력 소모량은 탐색 영역의 크기에 의존한다. 탐색 영역의 크기가 작은 경우 움직임 벡터의 정확성은 감소하지만, 계산량의 이득이 존재한다[2]. 반대로, 탐색 영역의 크기가 커지면, 움직임 벡터의 정확성은 증가하지만 계산량의 문제점이발생한다. 이런 관계의 처리를 위해, 동적 탐색 영역 결 정 (Dynamic Search Range) 기법이 참고 문헌[10,11] 에서 소개되었다. 동적 탐색 영역 결정 방식은 탐색 영역의 크기를 조절하기 위해 DFD(Displaced Frame Difference)나 DBD(Displaced Block Difference)의 임계치를 사용한다. 다른 고속 움직임 추정 기법과 비교해, 이 방식들은 우수한 부호화 효율을 보임에도 불구하고, 여전히 많은 계산량을 요구한다.

잘 알려진 것과 같이 한 블록의 움직임 벡터는 인접한 블록과 높은 상관성을 보인다. 그러므로 인접 블록의 상관도에 따라 움직임 추정을 위한 탐색 영역을 줄이는 것이 가능하다. 본 논문에서는 인접한 움직임 벡터의 국부적인 통계특성을 이용한 고속 움직임 추정기법을 제안한다. 불필요한 탐색 지점을 제거하기 위해, 현재 블록과 인접한 블록사이의 움직임 벡터들 간의 상관성을 나타내는 국부적 통계 특성을 정의하여 국부적인 탐색 영역을 결정한다. 더불어 조절된 탐색 영역을 기반으로 탐색 지점의 수를 절감하기 위해 VSS(Variable Step Search) 기법을 수행한다.

이 논문은 다음과 같이구성된다. 2장에서 제안하는 고속 움직임 추정 기법을 소개한다. 측정된 지역적 통계 특성이 정의되고 이를 기반으로 하는 적응적 탐색 영역 결정 방식에 대해 기술한다. 또한 TSS 의 변형된 형태인 VSS 기법이 기술된다. 마지막으로, 실험 결과와 결론을 3장과 4장에 걸쳐 기술한다.

II. 제안 방식

1. 탐색 영역 결정 기법

일반적인 동영상은 인접 영상과 현재 부호화 영상의 블록 사이에 높은 상관성을 보인다[10]. 이러한 특성으로 인해 움직임 벡터의 예측이나 움직임 추정 등이 이루어 진다. 그러나 기존의 움직임 추정 방식은 미리 지정된정방형의 움직임 탐색 영역 내에서 움직임 추정을 하게 되며 실제 최적의 움직임 벡터가 위치하는 곳이 원점에서 멀리 떨어진 곳이 아닐 때에도 불필요한 위치를 탐색하여 연산량이 증가하는 결과를 나타낸다[9]. 이와 같이 불필요한 연산을 감소시키기 위해 부호화 블록과 인접한 블록들과의 국부 움직임 벡터의 통계적 특성을 이용하여 수평 수직 각각의 방향에 대해 움직임 추정 탐색 영역을 결정한다.

움직임 추정 대상 블록과 인접 블록, 그리고 국부 움직임 벡터의 통계적 특성을 알아보기 위해 실시한실험에서 인접 블록의 움직임 벡터 크기가 모두 0이거나 방향이 같은 경우, 또 인접 블록의 움직임 벡터 합이 작은 경우는기타 경우보다 움직임벡터가 작게 결정되는 특징을 볼 수 있었다. 그림2는 Container 영상에서 인접 블록의 상이한 통계적 특성에 따른 부호화 블록의 움직임 벡터가 인접 블록의 움직임 벡터의 median 값과 차이인 MVD (Motion Vector Difference)의 분포를 나타낸다.

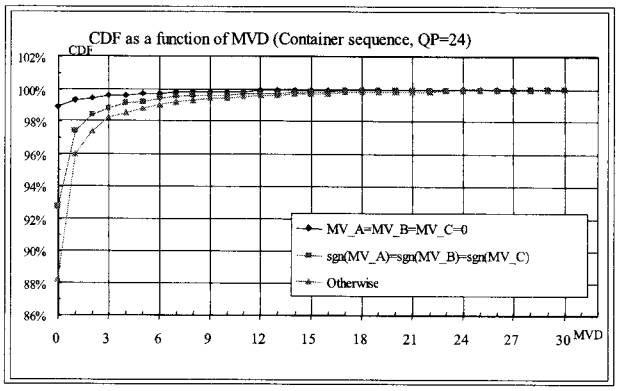


그림 2. 인접블록 통계 특성에대한 MVD의 CDF
Fig. 2. CDF as a function of MVD for local statistics of neighboring motion vectors

그림 2에 나타난 특성을 이용하여 인접 블록의 움직임 벡터에 따라 적정한 영역 내에서만 움직임 추정을 수행함으로써 탐색 지점 감소를 통한 연산량 절감이 가능하다. 인접한 움직임 벡터의 값에 따라 MVD(Motion Vector Difference)가 적게 나오는 지역은 적은 크기의 탐색 영역을 설정하고, 기타 블록에서는 넓은 영역을설정하여 기존의 전 영역 탐색과 성능비교에서 질적 차이가 생기지 않도록 한다. 이와 같은 과정은 수평및 수직 방향에 대해각각 독립적으로 진행이 되므로 x, y 방향에서 서로 다른 크기의 탐색 영역을 가지게 된다.

본 논문에서는 다음과 같은 방식을 제안 하고자 한다. 인접 블록들의 위치는 그림 3과 같이 나타난다. 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. 이와 같이 인접 블록의 움직임 벡터를 모두 정의한 후에 탐색 영역을 결정하기 위한 연산을 수행한다. 그림 3에서 블록 E는 현재 부호화 대상 블록이며 좌상측 좌표 위치를 (i,j), 우하측 좌표위치를 $(i+blocksize_h,\ j+blocksize_v)$로 표기 하였다. 이때 $block\ size_h$는 해당 블록의 수평 방향의 크기를 나타내고 $block\ size_v$는 블록의 수직 방향의 크기를 나타낸다.

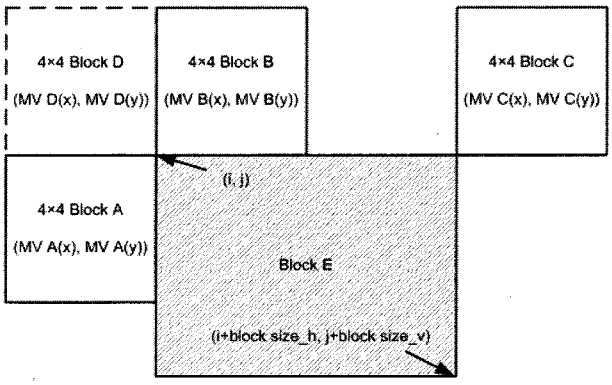


그림 3. 움직임 추정을 위한 인접 블록의 위치
Fig. 3. Neighboring blocks' location for motion estimation

블록 A, B, C는 블록 E와 상관관계가 가장 높은 인접 4×4 크기 블록으로 각 블록은 이미 수평 수직 방향의 움직임 벡터가 결정된 상태이다. 상기 인접 블록의 움직임 벡터를 이용하기 위해 먼저 주어진 탐색 영역 크기를 식 (2)과 같이 정의 한다.

$$SearchRange(i) \le |\omega|,\ i = x, y \qquad (2)$$

식 (2)는 부호화 조건으로 정의된 최대 움직임탐색 영역의 값을 의미하며, x 및 y는 각각 수평 및 수직 방향을의미한다. 그리고 인접한 세 개 블록의 움직임 벡터를 이용하여 식 (3)을 정의 할 수 있다.

$$MV_{\max}(i) = \max(|MV_A(i)|, |MV_B(i)|, |MV_C(i)|) \qquad (3)$$

식 (3)의 값은 움직임 벡터 추정 오류를줄이기 위한 방안으로 인접 블록 움직임 벡터 최대치를 최소 탐색 영역 크기로 정의한다. 이 과정을 통해 부호화블록은 가능성 있는 움직임 벡터 지점을 탐색 영역 내부에 포함할 수 있게 된다.

$$\alpha(i) = |MV_A(i)| + |MB_B(i)| + |MV_C(i)| \quad (4)$$

식 (4)는 인접 영역의 상관도와 움직임 벡터의 크기를 추정할 수 있는 척도로 사용된다. 즉, $\alpha(i)$ 가 작은 경우 부호화 블록은 전반적으로 적은 움직임을 가지는 영역에 속하므로 현재 블록 또한 적은 움직임 벡터를 가질 확률이높으며, 반대로$\alpha(i)$ 가 큰 경우 많은 움직임을 가지는 영역에 속하므로 현재 부호화 대상 블록의 움직임 또한 큰 움직임 벡터일 가능성이 높다. 특히 $\alpha(i)$ 가 0이 되는 경우는인접한 세 블록의 움직임 벡터가 모두 0이므로 현재 부호화 대상 블록 또한 0에 가까운 적은 값의 움직임 벡터를 가질 확률이 높아진다.

움직임 탐색 영역을 추정할 수 있는 또 다른 척도로 인접 블록의 움직임 벡터 크기와 방향의 동일함을 고려할 수 있다. 앞의 그림 2에서 나타난 바와 같이 부호화 블록의 움직임 벡터는 인접 블록의 움직임 벡터 방향이 모두 같은 경우 적은 값을 갖게 되며, 이와 같은 경우는 부호화 블록이 인접 블록과 동일 객체 내부에 존재할 확률이 높다. 본 논문에서는 인접 블록과의 상관성을 확인하기 위하여 움직임 벡터의 방향성과 더불어 크기의 동일함을 고려하였다. 인접 블록 움직임 벡터의 방향과 크기의 동일한 여부를 판별하기 위해 다음과 같은 식을 정의하였다.

$$Flag(i) = \begin{cases} 1, & for\ MV_A(i) = MV_B(i) = MV_C(i) \\ 0, & for\ \quad else \end{cases} \quad i = x, y \quad (5)$$

식 (4)에 의한 $\alpha(i)$ 값과 식 (5)에 의한 $Flag(i)$ 에 따라 주어진 탐색 영역을 국부 통계 특성을 반영하는 탐색 영역 중 하나로 결정한다. 이때 식 (5)에 의한 $Flag(i)$ 는 식 (4)에 의한 $\alpha(i)$ 보다 우선시 되는 조건으로 $Flag(i)$ 값이 1인 경우는 탐색 영역의 크기를 가장 작게 하고 그렇지 않은 경우 $\alpha(i)$ 에 의해 탐색 영역의 크기를 적응적으로 결정하게 된다.

$$\beta(i) = \begin{cases} \dfrac{\omega + 8}{16}, & for\ \ Flag\ (i) = 1 \\ \dfrac{\omega + 2}{8}, & for\ \ Flag\ (i) \neq 1 \,\&\, \alpha(i) \leq 2 \\ \dfrac{\omega + 1}{4}, & for\ \ Flag\ (i) \neq 1 \,\&\, \alpha(i) > 2 \end{cases} \quad i = x, y \quad (6)$$

식 (6)에서 구한 값은 인접 블록의 상관성에 따른 최소한의 탐색 영역 값이므로 다음의 식 (7)을 적용 하여 너무 적은 탐색 영역으로 인한 움직임추정 오류를 방지 한다.

$$\gamma(i) = \max(\beta(i), MV_{\max}(i)) \quad (7)$$

식 (7)에서 구한 $\gamma(i)$ 값을 최종 탐색 영역으로 결정 하기 전에 이 값이 식 (2)의 주어진 탐색 영역을 벗어 나는지 여부를 확인한다. 원래의 탐색 영역 보다 더 큰 영역에 대한 추정은 더 많은 연산을 소요하기 때문이다. 그래서 ω 과 $\gamma(i)$ 사이에 작은 값이 선택 된다.

$$\delta(i) = \min(\omega, \gamma(i)) \quad (8)$$

x, y 방향으로 각각의 탐색 영역 결정은 다음 식 (8)에 의해 다음과 같이 결정된다.

$$\begin{aligned} -\delta(x) \leq dx \leq \delta(x) \\ -\delta(y) \leq dy \leq \delta(y) \end{aligned} \quad (9)$$

이와 같이 결정된 dx, dy 영역 내에서 움직임 추정을 수행하게 된다.

2. Variable Step Search

탐색 영역 결정 기법을통해 전 영역 탐색 기법과 비교하여 많은 연산량 절감을 기대할 수 있지만 탐색 영역 내의 모든 화소에 대해서 SAD연산을 수행하기에는 많은 어려움이 따른다. 이러한 이유로 본 논문에서는 기존 TSS[3-5] 기법

을 변형한 VSS기법을 제안하고자 한다.

VSS 기법은 잘 알려진 TSS 기법의 변형을 통해 만들어 졌기 때문에 실제 구현에 용이한 이점이 있다. VSS 기법의 단계는 다음과 같다.

Step 1: 최대 탐색 단계와 step_size를 결정한다.

$$\max_{step}(i)=\max_{M(i)}\lfloor 2^{M(i)}<\delta(i)\rfloor+1 \quad i=x,y \tag{10}$$

최대 탐색 단계는 식 (10)을 통해 얻어지고 step-size는 식 (11)을 통해 계산된다. VSS 기법은 TSS와 달리 정방형이 아닌 탐색 영역 내에서이루어 지는 기법이므로 x, y 방향에 대해 독립적으로 식이 적용된다.

$$step\_size(i)=2^{M(i)} \tag{11}$$

Step 2: $step\_size(i)$ 와 동일한 거리에 위치한 9곳의 탐색 지점 중 최소 SAD를 가지는 한 지점을 결정한다.

Step 3: $step\_size(i)$ 를 반으로 줄여 Step 2를 반복한다. 이때 $step\_size(i)$ 가 x, y 방향 모두 1이 되면 종료한다. x, y 방향 중 한 방향이 먼저 1이 되면 다른 방향이 1이 될 때까지 계속 1을 유지하게 된다.

그림 4는 $\delta(x)$=17, $\delta(y)$=11 인 경우 VSS의 예를 나타낸 것이며 수행과정은 아래와 같다.

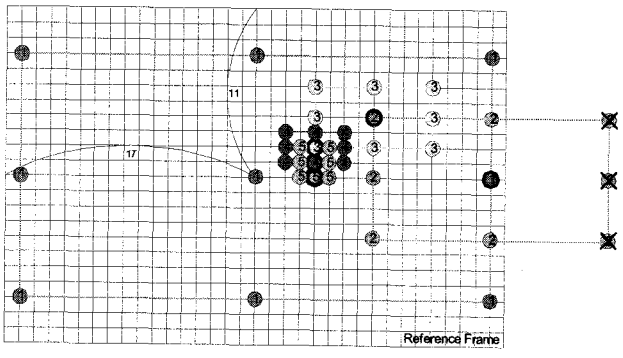


그림 4. Variable Step Search의 예
Fig. 4. Example of Variable Step Search

1st Step : $\delta(x)$=17, $\delta(y)$=11에 대한 $\max_{step}(x)=5$ 이고 $\max_{step}(y)=4$가 된다. 그리고, 수직 수평 방향에 대한 step_size는 $step\_size(x)=16$ 이고 $step_size(y)=8$이 된다.

2nd Step : 그림 4에서 1로 표기된 것과 같이 x, y 각각의 방향으로 step-size 만큼 떨어진 9개 후보 지점에 대해 SAD를 구한다. 그 중 최적의 지점을 다음 단계의 원점으로 정한다.

3rd Step : x, y 각각의 step-size 를 반으로 줄인다. 그래서 $step-size(x)=8$이고 $step-size(y)=4$가 된다. 그러면 그림 4에서 2로 표기된 것과 같이 원점을 중심으로 step-size 만큼 떨어진 9개 후보 지점에 대해 SAD를 구한다. 이 때 그림4의 우측 열에 존재하는 지점과 같이 탐색 지점이 탐색 영역을 벗어 나는 경우 탐색 지점에서 제외된다. 최적의 지점을 다음 단계의 원점으로 정한다.

4th Step : x, y 각각의 step-size를 반으로 줄인다. 그래서 $step-size(x)=4$이고 $step-size(y)=2$가 된다. 그러면 그림 4에서 3로 표기된 것과 같이 원점을 중심으로 step-size 만큼 떨어진 9개 후보 지점에 대해 SAD를 구하여 최적의 지점을 다음 단계의 원점으로 정한다.

5th Step : x, y 각각의 step-size를 반으로 줄인다. 그래서 $step-size(x)=2$이고 $step-size(y)=1$이 된다. 그러면 그림 4에서 4로 표기된 것과 같이 원점을 중심으로 step-size 만큼 떨어진 9개 후보 지점에 대해 SAD를 구하여 최적의 지점을 다음 단계의 원점으로 정한다.

6th Step : x, y 각각의 step-size를 반으로 줄인다. 그래서 $step-size(x)=1$이고 $step-size(y)=1$이 된다. 그러면 그림 4에서 5로 표기된 것과 같이 원점을 중심으로 step-size 만큼 떨어진 9개 후보 지점에 대해 SAD를 구하여 최적의 지

점을 찾으면 그 지점이 정화소 단위의 최적의 위치가 된다.

VSS 기법은 비정방형으로 나타나는 적응적 탐색 영역 결정 기법에 적합한 방식으로 기존 TSS 기법에 비해 좁은 영역에서 단계별 탐색을 수행함으로 인해 TSS 기법이나 확장된 FSS 기법에 비해 움직임 추정 효율이 더 높게 나타난다.

III. 실험 결과

본 논문에서는 다양한영상에서의 결과를 확인하기 위하여 그림 5와 같이 Stefan, Foreman, Container, 및 Claire 영상을 사용하였다. QP에 따른 성능을 비교하기 위해 QP 16, 24, 32, 40에 따른 성능을 비교하였다. 성능 비교를 위해 H.264 JM(Joint Model) 11.0 Baseline에 기술된 FFSS(Fast Full Spiral Search) 움직임 추정 방식과 고정스텝 탐색 움직임 추정방식 (예를 들어, 식 (2)의 W가 32인 경우 6 step search 움직임 추정 방식 및 참고문헌 [12]에 기술된 VSS 방식과 비교하였다. 실험결과에서 기술된 ASRD (Adaptive Search Range Decision)는 본 논문에서 제안된 탐색 영역 결정기법이며, 제안된VSS는 제안된 ASRD기법에 VSS를 적용한 방식을 의미한다.

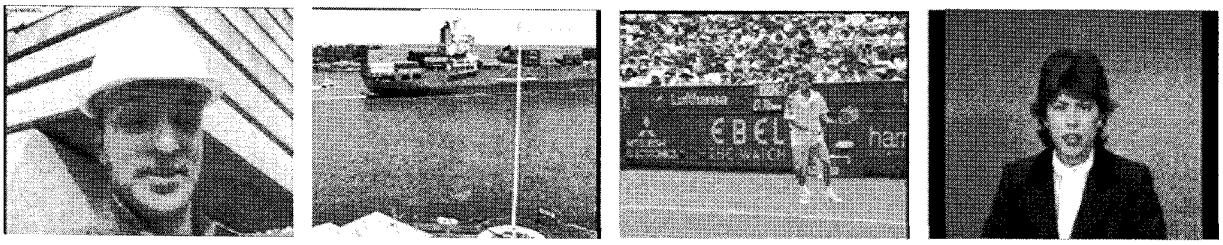
실험 환경은 windowsXP, Pentium 4 CPU 2.8GHz, 512MB RAM을 기반으로, 탐색 영역 32, QCIF급 영상을 이용하여 실험하였다. 제안된 방식의 객관적인 성능을 측정하기 위해PSNR(Peak Signal to Noise Ratio)이 사용되었으며, 이는 화소당 8bits로 구성된 $M \times N$ 영상 크기에 대해 다음과 같이 정의된다.

$$PSNR = 10 \log \frac{MN \times 255^2}{\|f - \tilde{f}\|^2} \quad (12)$$

식 (12)의 $\|\cdot\|$은 유클리안 놈(Euclidean norm)을 나타내고 f와 $\tilde{f}$는 각각 원 영상과 복원된 영상을 나타낸다.

표1. 양자화 인덱스 함수에 대한 PSNR 비교 (단위: dB)
Table 1. PSNR comparisons as a function of Quantization index (unit: dB)

Sequence	Method	QP			
		16	24	32	40
Foreman (QCIF)	FFSS	45.01	38.61	33.06	28.04
	ASRD	45.01	38.60	33.02	28.05
	6StepSearch	44.95	38.57	32.99	27.99
	Ref. [12] 방식	44.99	38.56	32.99	27.98
	제안된VSS	45.00	38.58	32.98	27.98
Container (QCIF)	FFSS	45.00	38.60	33.11	27.85
	ASRD	45.00	38.58	33.07	27.83
	6StepSearch	44.99	38.60	33.09	27.84
	Ref. [12] 방식	44.99	38.60	33.07	27.82
	제안된VSS	45.00	38.57	33.06	27.80
Stefan (QCIF)	FFSS	44.58	37.50	30.62	24.76
	ASRD	44.57	37.49	30.58	24.73
	6StepSearch	44.55	37.43	30.54	24.70
	Ref. [12] 방식	44.55	37.43	30.54	24.72
	제안된VSS	44.56	37.43	30.53	24.70
Claire (QCIF)	FFSS	47.00	41.12	35.29	30.31
	ASRD	47.00	41.15	35.27	30.29
	6StepSearch	47.00	41.11	35.28	30.25
	Ref. [12] 방식	47.00	41.11	35.25	30.26
	제안된VSS	47.00	41.12	35.26	30.27



(a) (b) (c) (d)

그림 5. 실험에 사용된 영상 ; (a) QCIF Foreman, (b) QCIF Container, (c) QCIF Stefan, (d) QCIF Claire
Fig. 5. Video sequences used for experimental evaluation ; (a) QCIF Foreman, (b) QCIF Container, (c) QCIF Stefan, (d) QCIF Claire

표 2. 양자화 인덱스 함수에 따른 Bit rate 비교 (단위: kbits/sec)
Table 2. Bit rate comparisons as a function of quantization index (unit: kbits/sec)

Sequence	Method	QP			
		16	24	32	40
Foreman(QCIF)	FFSS	340.1	127.2	46.1	18.3
	ASRD	333.2	125.8	46.9	19.6
	6StepSearch	342.7	130.2	48.3	19.3
	Ref. [12] 방식	338.7	128.9	48.0	19.6
	제안된VSS	333.9	127.2	47.8	20.1
Container(QCIF)	FFSS	203.6	54.8	14.6	5.2
	ASRD	203.2	55.0	14.6	5.2
	6StepSearch	203.4	54.7	14.6	5.2
	Ref. [12] 방식	203.5	54.7	14.6	5.2
	제안된VSS	203.4	55.2	14.6	5.2
Stefan(QCIF)	FFSS	701.2	332.8	125.4	42.4
	ASRD	694.4	329.9	126.9	44.1
	6StepSearch	719.2	351.4	135.3	47.2
	Ref. [12] 방식	715.9	347.3	134.5	47.1
	제안된VSS	708.3	342.2	132.2	47.9
Claire(QCIF)	FFSS	84.7	28.5	9.3	3.5
	ASRD	84.7	28.5	9.4	3.5
	6StepSearch	85.4	28.9	9.5	3.6
	Ref. [12] 방식	85.3	28.7	9.4	3.6
	제안된VSS	84.5	28.5	9.3	3.6

표 3. 양자화 인덱스 함수에 따른SAD/MB 비교
Table 3. SAD/MB comparisons as a function of Quantization index

Sequence	Method	QP			
		16	24	32	40
Foreman(QCIF)	FFSS	2103.4	2406.9	2552.9	2486.3
	ASRD	2130.3	2432.3	2541.0	2447.5
	6StepSearch	2232.9	2425.6	2472.9	2513.6
	Ref. [12] 방식	2224.7	2480.8	2576.6	2579.1
	제안된VSS	2188.3	2412.4	2529.6	2560.4
Container(QCIF)	FFSS	1144.2	1176.3	1451.0	1933.2
	ASRD	1137.5	1179.2	1449.0	1945.3
	6StepSearch	1138.0	1171.4	1457.4	1933.1
	Ref. [12] 방식	1151.1	1190.9	1466.7	1947.7
	제안된VSS	1142.6	1184.3	1443.1	1944.8
Stefan(QCIF)	FFSS	4646.7	4914.7	5881.4	6229.3
	ASRD	4846.7	5098.5	6022.4	6129.4
	6StepSearch	5691.0	5761.0	6111.2	5986.0
	Ref. [12] 방식	5628.9	5834.0	6106.7	6086.6
	제안된VSS	5569.4	5656.6	6098.8	6087.2
Claire(QCIF)	FFSS	601.5	802.2	1035.7	1356.2
	ASRD	604.3	801.8	1045.7	1373.4
	6StepSearch	622.4	811.5	1030.9	1365.1
	Ref. [12] 방식	609.8	810.2	1035.9	1372.8
	제안된VSS	609.4	808.2	1034.4	1375.9

표 4. 양자화 인덱스 함수에 따른 탐색 지점 비교
Table 4. Search points comparisons as a function of Quantization index

Sequence	Method	QP			
		16	24	32	40
Foreman (QCIF)	FFSS	1,255,943	995,656	619,282	338,983
	ASRD	172,070	147,655	105,731	68,716
	6StepSearch	21,815	20,556	17,862	14,128
	Ref. [12] 방식	17,136	16,037	14,753	12,517
	제안된VSS	14,795	14,373	13,160	11,079
Container (QCIF)	FFSS	842,823	632,393	382,670	210,360
	ASRD	77,928	65,730	45,236	27,968
	6StepSearch	19,083	17,231	13,824	10,490
	Ref. [12] 방식	13,259	12,887	10,617	8,518
	제안된VSS	11,422	10,670	9,311	7,587
Stefan (QCIF)	FFSS	1,356,658	1,217,454	940,924	584,357
	ASRD	198,638	185,073	159,738	116,992
	6StepSearch	21,803	20,783	19,116	16,554
	Ref. [12] 방식	19,870	18.907	17,231	14,725
	제안된VSS	17,072	16,408	15,323	13,703
Claire (QCIF)	FFSS	485,966	409,060	269,165	138,753
	ASRD	48,643	45,039	32,261	16,714
	6StepSearch	14,799	13,973	11,504	8,558
	Ref. [12] 방식	10,522	9,909	8,758	6,867
	제안된VSS	9,906	9,563	8,271	6,571

표 1및 2에 각 방식의 양자화 인덱스 함수에 대한PSNR 및 비트율 비교를 나타내었다. 동일 비트율 기준으로ASRD는 FFSS와 유사한 성능을 보였으며, 제안된 VSS 방식은6 step search 및 기존 VSS 방식과 유사한 성능을 보임을 확인할 수 있었다. 그러나 Stefan과 같이 움직임이 큰 영상의 경우, FFSS와 비교하여 동일 비트율에서 최대 0.06 dB 정도의 오차가 발생함을 확인할 수 있었다. 상기와 같이 움직임이 큰 영상인 경우 인접 블록들의 움직임 벡터와 부호화 블록 움직임 벡터의 상관관계가 감소 함으로서 움직임 추정 영역에서 오류가 발생하게 되고, 이로 인해 움직임 보상 영상과 원영상과의 차이 신호인 잔여 영상의 값이 상대적으로 커지게 되어 압축 효율의 감소가 발생하게 된다.

표 3에 부호화 대상블록과 참조 블록 간의 움직임 벡터 정확성을 나타내는 SAD(Sum of Absolute Difference) 를 나타내었다. 표 3에 나타난 바와 같이 ASRD는 FFSS와 유사한 성능을 보이며, 제안된 VSS 기법은6StepSearch와 유사한 성능을 보이며, 기존 VSS과 비교하여 약간의 개선이 있음을 확인할 수 있다. 이는 제안한 방식이 기타 방식과

비교하여 움직임 추정에서 결정되는 움직임 벡터가 유사한 국부 최소값 (local minimum)에서 결정되는 이유에 기인한다. 특히, 기존 VSS 방식과 비교하여 제안된 ASRD 기법이 국부 통계 특성에 따라움직임 추정 영역을 상대적으로 정확히 예측한 결과로 분석되었다.

위와 같이 제안된 ASRD는 FFSS와 유사한 성능을 나타내고, 제안된 VSS 기법은 6Stepsearch 및 기존 VSS와 유사한 성능이 있는 반면 제안된 ASRD 및 VSS 방식은 기존방식과 연산량에서 상당한 이득이 있었음을 확인할 수 있었다. 움직임 추정시 소요되는 연산량을 비교하기 위한 탐색지점을 비교 결과를 표 4에 나타내었다. 제안된 ASRD 방식을 이용하여 FFSS대비 85% 정도의 탐색지점 절감을 이루어졌으며, 제안된 VSS 방식을 이용하여 6StepSearch및 기존 VSS 방식과 비교하여 각각 평균30% 및 10% 정도의 탐색지점 절감이 있었음을 확인할 수 있었다.

이와 같은 결과를 통해본 논문에서 제안한 ASRD 및 VSS방식은 동일 비트율에서 0.05dB 정도의 압축효율 손실이 있었으나, 부호화 연산량을 나타내는 탐색지점의 개수는 급격히 줄어든다는 것을 확인할 수 있다. 즉, 부호화에 소요되는 시간이 상당한 이득이 있었음을 확인할 수 있었다.

V. 결 론

본 논문에서는 H.264 동영상 표준 부호화 방식에서 인접 블록의 정보를 이용한 고속 움직임추정 기법을 제안하였다. 실험 결과 본 논문에서 제안한 VSS기법은 기존의 전 영역 탐색에 비해 탐색 지점은 평균 99% 절감되었고, 6StepSearch 및 기존 VSS기법과 비교하여 각각 30% 및 10%의 속도 절감을 기대할 수 있다. PSNR의 경우 최대 0.08dB 이내의 차이를 나타냈으며, Bit Rate는 0.02% 증가하는 것을 확인할 수 있었다. 결국 제안한 기법은 영상의 질적 저하 없이 고속 움직임 추정이 가능하다. 이처럼 부호화 블록과 상관도가 높은 인접 블록의 움직임 벡터를 이용함으로써 압축의 효율은 높이고 연산량은 감소 시킬 수 있었다. 앞으로 고속 영상 압축 관련 기술에 응용할 수 있을 것으로 기대된다.

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), May 2003.

[2] I. Richardson, H.264 and MPEG-4 Video Compression, Wiely, 2003.

[3] T. Wiegand, G. Sullivan, G. Njontegaard and A. Lutjra, "Overview of the H.264/AVC Video Coding Standard," IEEE Trans. on Circuit and Systems for Video Tech., vol.13, no. 9, pp. 560-576, July 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 Systems 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, April 1993.

[8] Y.-L. Chan and W.-C. Siu, "New adaptive pixel decimation for block motion vector estimation," IEEETrans. Circuits and Systems for Video Tech., vol. 6, no. 1, pp. 113-118, Feb. 1996.

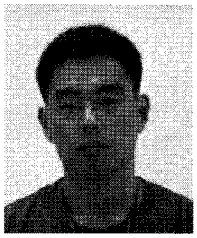
[9] J. Feng, K. T. Lo, H. Mehrpour, and A. E. Karbowiak, "Adaptive block matching algorithm," Electronic Letter, vol. 31, no. 18, pp. 1542-1543, Aug. 1995.

[10] L. W. Lee, J. F. Wang, J. Y. Lee, and J. D. Shie, "Dynamic search window adjustment and interlace search for block-matching algorithm," IEEE Trans. Circuits and Systems for Video Tech., vol. 3, no. 2, 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 Electronics, vol. 44, no. 3, pp. 659-666, Aug. 1998.

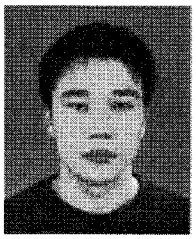
[12] 윤성현, 최권열, 이성수, 홍민철, "H.264 동영상 표준 부호화 방식을 위한 고속 움직임 추정 기법," 한국통신학회논문지, vol. 30, no. 11C, pp. 1091-1097, 2005년 11월.

저 자 소 개



김 기 범

- 2006년 2월 : 숭실대학교 정보통신·전자공학부 졸업 (학사)
- 2006년 3월 ~ 현재 : 숭실대학교 정보통신공학과 석사과정
- 주관심분야 : 동영상 부호화, H.264/AVC



정 찬 영

- 2007년 2월 : 숭실대학교 정보통신·전자공학부 졸업 (학사)
- 2007년 3월 ~ 현재 : 숭실대학교 정보통신공학과 석사과정
- 주관심분야 : 동영상 부호화, Multiview video coding

홍 민 철

- 1988년 2월 : 연세대학교 전자 공학과 졸업 (학사)
- 1990년 8월 : 연세대학교 전자 공학과 졸업 (석사)
- 1990년 7월 ~ 1991년 8월 : LG정보통신 연구원
- 1997년 9월 : Northwwestern University, 전기 및 컴퓨터 공학과 졸업 (박사)
- 1997년 9월 ~ 1998년 8월 : Northwestern University, Research Fellow
- 1998년 9월 ~ 2000년 2월 : LG전자, 선임연구원
- 2000년 3월 ~ 현재 : 숭실대학교, 부교수
- 주관심분야 : 영상 복원 및 enhancement, 정지 및 동영상 필터링, 동영상 부호화, 비선형 필터링, Signal Blind Deconvolution