논문 2007-44SP-3-9

# H.264/AVC의 효율적인 전 영역 움직임 추정을 위한 새로운 움직임 벡터 예측 방법 제안

## ( New Motion Vector Prediction for Efficient H.264/AVC Full Pixel Motion Estimation )

최 진 하*, 이 원 재*, 김 재 석*

( Jinha Choi, Wonjae Lee, and Jaeseok Kim )

요 약

본 논문은 영상 부호화 표준인 H.264/AVC에서 중요한 과정인 움직임 추정에서 효율적인 전 영역 추정을 위해 인접 서브 매크로 블록과 독립적인 새로운 움직임 예측 방식을 제안한다. H.264/AVC는 높은 압축 효율을 위해 H.264/AVC에서는 움직임 추정 과정에서 7가지의 다양한 가변 블록을 사용 한다. 그러나 다양한 가변 블록으로 인해 반복적인 연산이 많아지고 복잡해져 움직임 추정에 많은 시간이 소요된다. 이로 인해 실시간 인코딩이 어려워지게 되었다. 이를 해결하기 위해 많은 고속 움직임 추정 방식이 제안되었으나 전 영역 움직임 추정에 비해 영상의 품질(PSNR)이 떨어지고 비트 수가 증가하게 된다. 제안된 독립적 움직임 예측 방식과 SAD 값을 공유하여 움직임 추정을 수행할 경우 기존 전 영역 탐색에 비해 반복적인 연산 양이 80%정도 감소하게 된다. 또한 연산양이 감소됨에도 불구하고 시뮬레이션 결과 Y PSNR은 최대 0.04 dB 이하의 변화만을 가져왔고 인코딩된 Bit 수는 평균적으로 약 0.6% 정도만 증가하였다.

Abstract

H.264/AVC has many repeated computation for motion estimation. Because of that, it takes much time to encode and it is very hard to implement into a real-time encoder. Many fast algorithms were proposed to reduce computation time but encoding quality couldn't be qualified. In this paper we proposed a new motion vector prediction method for efficient and fast full search H.264/AVC motion estimation. We proposed independent motion vector prediction and SAD share for motion estimation. Using our algorithm, motion estimation reduce calculation complexity 80% and less distortion of image (less PSNR drop) than previous full search scheme. We simulated our proposed method. Maximum Y PSNR drop is about 0.04 dB and average bit increasing is about 0.6%

**Keywords**: H.264, AVC, 움직임 추정, 움직임 벡터 예측, SAD 공유

## I. 서 론

최근 DMB나 DVB-H, PMP와 같이 손안의 멀티미디어 기기의 사용량이 증가함에 따라 데이터 양이 많은 동영상의 압축 기술이 중요시 되었다. 이에 H.264/AVC와 같이 H.263이나 MPEG-2 등 기존 압축 기술에 비해 압축 효율이 증가한 기술들이 선보이게 되었다. 영상 압축에 있어서 핵심 기술은 움직임 추정이라 할 수 있다. 이에 H.264/AVC에서는 기존의 압축 기술에 비해 움직임 추정에 있어 그림 1과 같이 7 가지 가변 블록을 사용하여 보다 정확한 움직임 추정을 하게 된다.

H.264/AVC는 움직임 추정을 수행할 때 정수 화소만이 아닌 1/4 화소 단위까지 탐색을 행하여 기존 압축 기술 방식에 비해 보다 세밀한 움직임까지 추정하게 된

* 정회원, 연세대학교 전기전자공학과
(Department of Electrical and Electornics Engineering, Yonsei University)

※ 본 연구는 정보 통신부 및 정보통신 연구 진흥원 대학 IT 연구센터 육성·지원 사업 및 2006년도 교육인적자원부 BK21 사업의 일환인 연세대학교 전기전자공학부 TMS 사업단의 지원을 받아 연구되었음

접수일자: 2007년1월11일, 수정완료일: 2007년4월11일

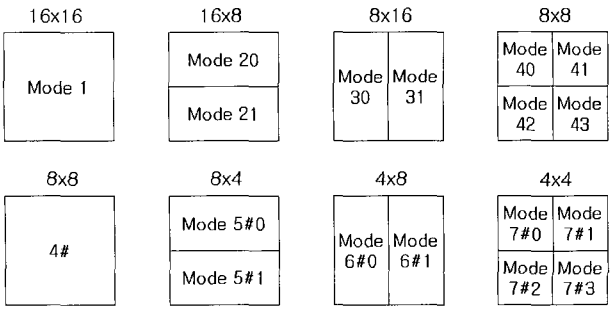



그림 1. 7가지 가변 블록 모드
Fig. 1. 7 variable block sized mode.

다. 기존 압축 기술의 경우 움직임 추정 시 참조하는 영상을 전 프레임 하나만을 참조했다. 이러한 단일 참조 영상의 경우 몇 가지 영상이 반복적으로 나타나는 현상이 발생할 경우 이로 인한 열화 현상이 심하게 발생하는데 H.264/AVC에서는 복수의 참조 영상을 통해 움직임 추정을 하여 반복적인 현상으로 인한 열화 현상이 줄어들게 된다.

최신 동영상 비디오 부호화 표준인 H.264/AVC는 가변 블록 등 추가된 움직임 추정을 통해 압축 효율이 우수한 장점을 가지게 된다. 하지만 높은 압축 효율과 좋은 영상 품질을 가지기 위해 움직임 추정 시 필요한 움직임을 찾기 위해서는 반복적인 연산이 필요하다. 이러한 반복적인 연산으로 인해 압축에 필요한 연산 양이 많아지고 연산 시간이 늘어나게 된다. 따라서 움직임 추정에 필요한 시간을 줄이기 위해 많은 고속 움직임 추정 알고리즘들이 제안되었다. 하지만 TSS (Three Step Search)와 같은 기존 고속 탐색 알고리즘의 경우 지엽적인 최소값 (Local Minimum)을 최적 값으로 찾게 되어 영상의 품질 (PSNR) 저하가 이루어지는 단점이 있다.[1]

또한 특정 조건을 만족하면 탐색을 멈추는 Early termination과 같은 고속 탐색 알고리즘은 매크로 블록 당 움직임 추정에 필요한 평균 시간을 줄이게 되는데 이러한 경우 termination이 발생하지 않는 극단적인 경우(Extreme case)에서는 Early termination의 의미가 사라지고 Early termination을 사용하지 않는 탐색에 비해 추정 시간 감소가 극히 적게 된다. 평균 시간의 단축을 고려한 Early termination 방식의 경우 실제로 프레임 당 추정시간이 불규칙하게 되므로 하드웨어를 이용한 실시간 인코더를 구현 시 기능적인(Functional) Timing을 맞추기 까다롭게 되고 주어진 영상의 복잡함에 따라 인코딩 시간과 영상의 품질이 고르지 못한 경우가 발생한다.[2]

고속 움직임 추정을 포함한 기존 움직임 추정 알고리즘의 경우 움직임 추정을 수행할 때 이전 움직임 추정 과정에 영향을 받는 종속적인 움직임 추정 현상이 발생한다. 이로 인해 움직임 추정을 수행할 때 이전 움직임 추정이 완료될 때까지 기다리게 되므로 수행 시간이 증가하게 되고 하드웨어 구현에 타이밍 분배가 힘들어 지고 파이프라인이나 병렬처리가 까다로운 단점을 가져오게 된다.

본 논문에서는 기존 움직임 추정에서 이전 수행 시간 동안 기다리게 되는 문제점을 해결하고 연산 양과 연산 시간을 개선시킬 수 있는 움직임 예측 방식과 이를 이용한 움직임 추정 방법을 제안한다. 본 논문은 Ⅱ장에서는 기존 H.264/AVC의 움직임 추정 방식의 특징을 분석하고 기존 움직임 예측방식에 대해서 분석한다. 또한 기존 고속 알고리즘에 대해서 분석한다. 그리고 Ⅲ장에서는 움직임 추정을 빠르고 효율적으로 할 수 있는 움직임 예측 방식을 제안한다. 제안된 방식의 성능 분석은 Ⅳ장에서 실험 결과를 통해 설명하겠으며 Ⅴ장에서는 본 논문의 결론을 내리고자 한다.

## Ⅱ. 움직임 추정 방식의 특징

### 1. SAD와 비용 함수 (Cost)

SAD (Sum of Absolute Differences)는 두 영상의 차이에 대한 비교를 위해 자주 사용된다. 일반적으로 두 신호의 차이를 비교할 때는 MSE (Mean Square Error)를 사용하여 에러의 정도를 판단한다. 하지만 영상의 경우 처리할 데이터의 양이 많고 동영상 인코딩 시 빠른 처리 시간을 요구하므로 곱하기와 루트를 사용하는 MSE에 비해 좀 더 연산이 간단한 SAD가 주로 사용되게 된다. 일반적인 SAD만을 사용하게 될 경우 수식적인 최적 값을 찾아 실제 움직임과 관계가 적게 되고 움직임 벡터가 불규칙해지는 문제점이 존재한다. 이로 인해 영상의 품질(PSNR)이 저하되고 인코딩된 비트수가 증가한다. 이러한 문제점을 줄이기 위해서 통계적으로 만들어진 라그랑제 상수를 더하여 Cost 함수를 만들게 되는데 Cost 함수 수식은 수식 (1)과 같다.

$$Cost = SAD + \lambda \cdot MVBits \quad (1)$$

표 1은 H.264/AVC의 인코딩 수행 시 SAD만을 사용한 움직임 추정과 Cost 함수를 이용한 움직임 추정의 비교를 나타낸 표이다. (Baseline profile, 100 프레임, 탐색영역 +15~-15, 다섯 이전 화면 참조) SAD만을 사

표 1. SAD만을 사용한 경우와 Cost 함수를 사용한 경우의 Y PSNR과 비트 율 비교

Table 1. Y PSNR and Bit rate comparison between using SAD only and using Cost function.

| | SAD만 사용 | | Cost 함수 사용 | |
|---|---|---|---|---|
| | PSNR | Bit rate | PSNR | Bit rate |
| container.qcif | 35.74 | 115.64 | 35.93 | 46.00 |
| carphone.qcif | 36.05 | 241.36 | 36.45 | 89.86 |
| foreman.qcif | 34.05 | 548.64 | 35.13 | 111.96 |

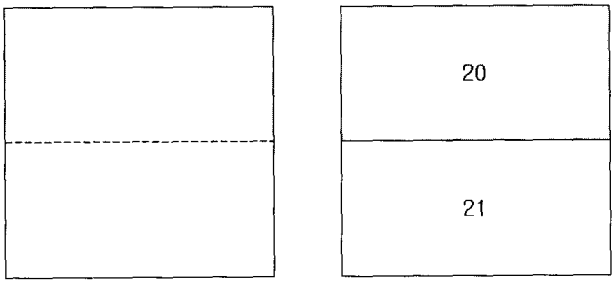



그림 2. Mode 1과 Mode 2의 SAD 값 비교 그림

Fig. 2. SAD region comparison between mode 1 and 2.

용할 경우와 Cost 계산식을 같이 사용한 경우의 영상의 품질 차이(Y PSNR)는 표 1과 같이 SAD만을 사용한 쪽이 떨어짐을 알 수 있다. 또한 모션 벡터의 불규칙성으로 인해 비트 율도 매우 커짐을 알 수 있다.

SAD 값의 경우 그림 2와 같이 현재 같은 탐색 지점에서 Mode 1의 SAD 값은 수식 (2)처럼 Mode 2에서 Sub MB 20과 Sub MB 21의 SAD 값의 합과 같다. (여기서 지정한 모드는 그림 1의 모드 설정과 같다.) 이는 같은 지점에서 Sub MB 20과 Mode 1의 위쪽 부분의 화소 차이 값이 동일하고 Mode 1의 아래쪽 부분과 Sub MB 21의 화소 차이 값이 같다는 것이다. 따라서 수식 (2)와 같이 Mode 1의 SAD는 Mode 2의 SAD 합으로 사용할 수 있다. 또한 수식 (3) 과 같이 Cost를 구할 때 이 값을 사용할 수 있다. 특정 움직임 추정 지점에서 모든 모드에 대해 하위 모드의 SAD를 합해서 Cost를 구하게 되는데 이를 SAD 공유라고 한다.

$$SAD_{Mode\,1} = SAD_{Mode\,20} + SAD_{Mode\,21} \tag{2}$$

$$Cost_{Mode\,1} = SAD_{Mode\,20} + SAD_{Mode\,21} + \lambda \bullet MVBits \tag{3}$$

## 2. H.264/AVC의 움직임 벡터 예측 방식

H.264/AVC의 전 영역 움직임 추정 시 움직임 벡터 예측 방법은 기본적으로 움직임 벡터의 중간 값 예측 (Median Prediction) 방법을 사용한다.

그림 3과 같이 H.264/AVC에서 사용하는 중간 값 예측 방식은 현재 구하려는 매크로 블록(Macro Block, 이하 MB)의 왼쪽(A), 위쪽(B), 오른쪽 대각선 방향(C)의 움직임 벡터 x,y 각각 세 개의 값의 중간 값을 예측 움직임 벡터로 취한다. 영상의 가장 오른쪽 MB의 경우 C 방향이 존재하지 않을 수 있는데 이 경우에는 왼쪽 대각선 방향(D)으로 대신한다.

그림 4와 같은 경우는 예외적인 경우로 추정하려는 매크로 블록이 정사각형이 아닐 때(16x8, 8x16 형태의 MB) 움직임 추정을 하려는 현재 매크로 블록에서 가까운 위치의 움직임 벡터를 예측 벡터로 그대로 가져다 쓴다.

$$MVD(x,y) = Current\,MV(x,y) - Predicted\,MV(x,y) \tag{4}$$

이러한 중간 값 예측 방식을 사용하는 이유는 H.264/AVC에서 움직임 벡터를 엔트로피 코딩할 때 움직임 벡터를 그대로 저장하는 것이 아닌 수식(4)와 같이 이전 움직임 벡터와의 차이 값을 코딩하여 저장하는

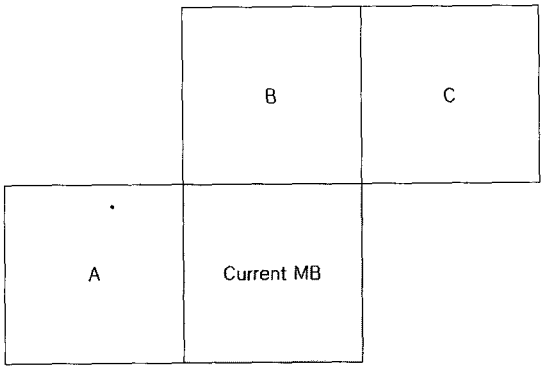



그림 3. 움직임 벡터의 중간 값 예측

Fig. 3. Median Prediction of Motion Vector Prediction.

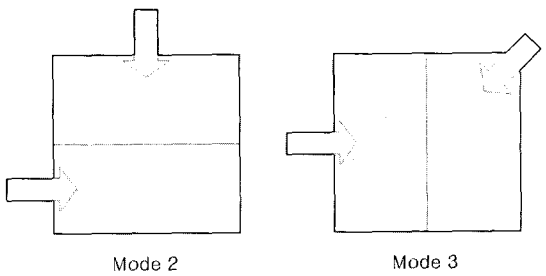



그림 4. 움직임 벡터 예측에서 예외

Fig. 4. Exception of Motion Vector Prediction.

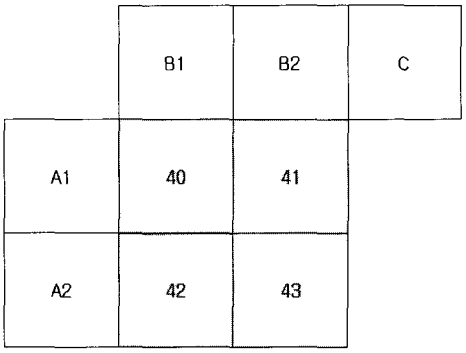



그림 5. Mode 4에서의 움직임 벡터 예측
Fig. 5. MV prediction of mode 4.

방식을 사용하기 때문이다. 따라서 움직임 벡터를 예측을 통해 움직임 벡터의 차이 값을 최소화하여 압축 효율이 높아지게 된다.

특정 탐색 지점에서 움직임 벡터 예측 값은 Mode 별, Sub MB 별로 각각 다르게 존재하게 된다. 따라서 움직임 벡터의 차이 값을 저장하기 위해 사용되는 MV bit 값은 Mode 별, Sub MB 별, 탐색 위치별로 각각 다른 값을 갖게 된다. H.264/AVC의 움직임 추정 방식에서 사용하는 중간 값 예측의 경우 근처의 움직임 벡터를 이용하여 예측 값을 가지므로 근처 움직임 벡터에 따라 움직임 예측 값이 결정되는 종속적인 추정이 된다.

예를 들어 그림 5와 같이 Mode 4에서 움직임 벡터를 예측할 때 Sub Macro Block(이하 Sub MB) 40의 경우 A1(왼쪽)의 움직임 벡터 값과 B1(위쪽)의 움직임 벡터 값, B2(오른쪽 위)의 움직임 벡터 값의 중간 값을 사용한다. Sub MB 41에서는 B2와 C의 움직임 벡터 값과 Sub MB 40에 해당하는 움직임 벡터 값의 중간 값을 사용하게 된다. 이는 Sub MB 41의 움직임 예측은 Sub MB 40의 움직임 예측과 독립적이지 않고 Sub MB 40 부분의 움직임 벡터 값에 영향을 받는 종속적인 움직임 예측이다. 두 작업이 종속적으로 일어나게 되는 경우 한 작업이 완료된 후에 다른 작업을 시작할 수 있다. 따라서 두 Sub MB의 움직임 벡터를 하드웨어로 구현 할 때 동시에 처리를 할 수 없게 되는 단점이 존재 한다. 또한 Mode 4의 모든 Sub Mode를 동시에 처리할 수 없으므로 SAD를 공유할 수 경우 없게 된다.

## 3. 기존 움직임 추정 알고리즘

기존 고속 움직임 추정 알고리즘의 경우 특정 지점을 대표로 해서 움직임 벡터를 찾는 방식이 많다. 예를 들어 그림 6의 Three Step Search(이하 TSS) 방식의 경우 처음 9개 지점의 Cost를 구한 뒤 그중 가장 최적의

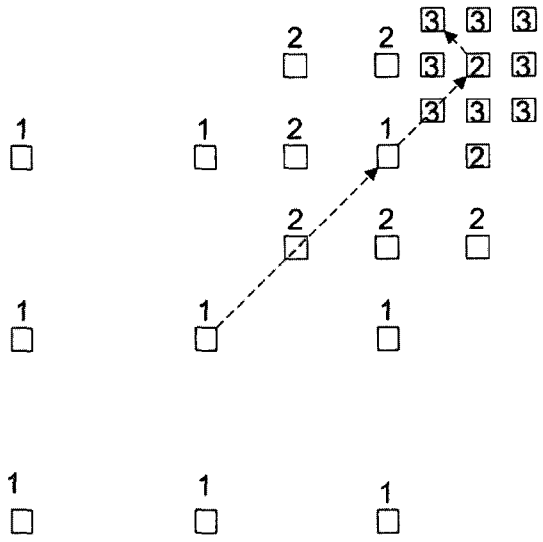
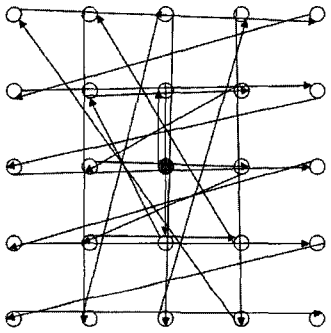
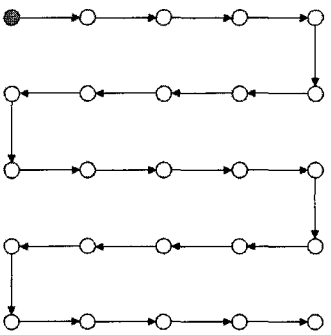



그림 6. Three Step Search 방식
Fig. 6. Three Step Search method.



(a) 나선형 탐색
(a) Spiral Search



(b) 단순 선형 탐색
(b) Plain Search

그림 7. 전 영역 움직임 추정의 탐색 방법
Fig. 7. Searching pattern of full pixel motion estimation.

Cost를 찾아 다시 주변 8개와 비교하고 최적의 Cost 주변에서 8개를 추가로 비교하게 된다.

New Three Step Search, Novel Four Step Search, Diamond Search, Hexagon Based Search, UMHexagonS와 같은 기존의 많은 고속 움직임 추정

알고리즘의 경우 또한 최적의 Cost 주변으로 다시 최적 값을 찾는 과정을 반복하게 된다. 이 경우 최적 Cost 값 주변에서 다시 추정을 시작하므로 이전 추정 과정의 결과에 따라 종속적으로 추정을 하고 Mode 별로 최적 값의 방향이 다르게 되므로 SAD를 공유할 수 없게 된다.[3]~[6]

그리고 기존의 전 영역 탐색을 할 때에도 그림 6의 (a)와 같은 나선형 추정을 사용하면 최초의 시작점이 움직임 벡터의 예측 값에서 출발하므로 각 Mode 별로 탐색하는 영역이 달라져 SAD 값을 공유할 수 없게 된다. 따라서 SAD를 공유하기 위해서는 그림 7의 (b)와 같은 움직임 탐색의 방식을 사용해야 한다.

그림 7에서 하이라이트 된 지점은 움직임 벡터 추정의 시작부분이다. 그림 7의 (a)에서의 시작점은 예측된 움직임 벡터에 해당하는 점이다. 그림 7의 (b)에서는 움직임 벡터의 가장 가장자리 점이 된다. 그림 7의 경우 움직임 추정 영역이 (-2~+2)인 경우를 예를 든 것인데 그림 7에서처럼 탐색 범위가 -2~+2 인 경우 (-2, -2) 지점이 시작 지점이 된다.

## Ⅲ. 제안된 움직임 벡터 예측 알고리즘

기존 H.264/AVC의 움직임 벡터 추정 방식을 사용하면 종속적인 움직임 추정으로 인해 모든 Sub 모드를 동시에 처리가 불가능 하고 SAD 또한 다시 사용할 수 없는 단점이 있다. 또한 고속 탐색 알고리즘들의 경우 Local Minimum에 빠지는 단점이 존재하므로 전 영역 탐색을 기반으로 추정을 하는 것이 영상 품질이 뛰어나게 된다.

따라서 본 논문에서는 모든 Mode가 SAD를 공유 가능하면서 모든 움직임 벡터 예측이 독립적으로 이루어져 Cost 값 추정이 동시에 가능한 방식을 제안한다. Mode 1의 경우 하나의 MB만을 추정하고 Mode 2와 3의 경우 주변에 가까운 값으로 움직임 벡터 예측을 하므로 기존 H.264에서도 그림 7의 (b)와 같은 방식의 움직임 추정을 한다면 같은 움직임 벡터 지점에서 SAD를 공유하며 동시에 연산이 가능하다. 하지만 다른 Mode의 경우 앞부분에서 설명한 것과 같이 종속적인 움직임 예측을 사용하므로 SAD를 공유하면서 동시에 연산이 불가능하다. 따라서 다음과 같은 변형된 움직임 예측 방식을 사용하여 그림 8과 같은 SAD 공유를 사용할 수 있다.

제안된 움직임 벡터 예측은 독립적인 움직임 예측을 위해 중간 값 예측을 위해 참조하는 세 방향의 벡터에서 근처 Sub MB에 종속적인 예측을 없애는 방식이다. 예를 들면 그림 9에서 701 블록의 움직임 벡터를 예측

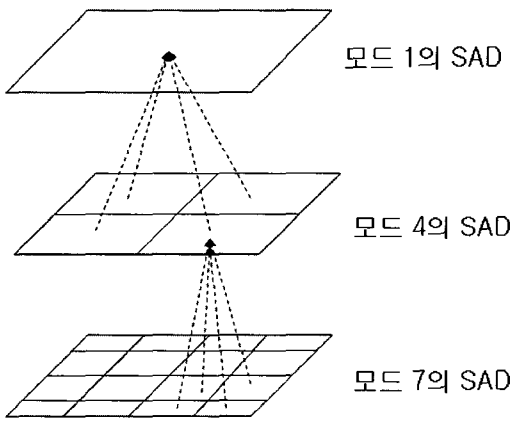



그림 8. SAD 공유 예시
Fig. 8. Example of SAD Share.

| | B1 | B2 | B3 | B4 | C |
|---|---|---|---|---|---|
| A1 | 700 | 701 | 710 | 711 | |
| A2 | 702 | 703 | 712 | 713 | |
| A3 | 720 | 721 | 730 | 731 | |
| A4 | 722 | 723 | 732 | 733 | |

(a) 기존 움직임 벡터 예측 방식
(a) Orignal MV prediction

| | B1 | B2 | B3 | B4 | C |
|---|---|---|---|---|---|
| A1 | 700 | 701 | 710 | 711 | |
| A2 | 702 | 703 | 712 | 713 | |
| A3 | 720 | 721 | 730 | 731 | |
| A4 | 722 | 723 | 732 | 733 | |

(b) 제안된 움직임 벡터 예측 방식
(b) Proposed MV prediction

그림 9. Mode 7에서의 움직임 벡터 예측
Fig. 9. MV prediction of mode 7.

하기 위해 기존 예측 방법에서는 세 방향 예측 값 중 하나인 700 블록의 움직임 벡터 값 (왼쪽 방향의 움직임 벡터 값)을 알아야 했다. 하지만 이럴 경우 두 예측이 독립적으로 이루어 지지 않게 되므로 700 블록과 같은 방향에 있으면서 독립적인 움직임 벡터 값인 A1 값으로 대체한다.

그림 9의 702 블록에서도 기존 움직임 예측 방식은 A2와 700 블록, 701 블록의 움직임 벡터 값을 예측해야 했다. 하지만 제안된 움직임 예측 방식은 700 블록 대신 B1의 움직임 벡터를, 701 블록 대신 B2의 움직임 벡터를 움직임 벡터 예측에 사용한다. 같은 방식으로 나머지 모든 Sub 모드 중 종속적인 예측을 해야 하는 경우 같은 방향성을 가지는 가장 가까운 MB의 외부 값을 가져온다. 이렇게 되면 모든 모드가 동시에 독립적으로 Cost를 계산 할 수 있게 되고 이로 인해 SAD를 재사용하여 1번의 탐색만으로 모든 모드의 움직임 벡터 값을 찾을 수 있게 된다.

$$MVbits \propto |CurrentMV - ProposedPredictedMV| \quad (5)$$

이와 같은 움직임 예측 방식을 사용하게 되면 수식 (5)와 같이 Cost를 구할 때 사용하는 MVbits 값이 결정되고 따라서 특정 움직임 추정 지점에서 모든 서브 모드의 Cost 계산 시 SAD 공유가 가능하여 모드별 독립적인 계산이 가능하다.

제안된 움직임 벡터 예측 방식은 움직임 벡터의 특징에서 기인한다. 현재 MB에서 대부분의 움직임 벡터는 이전 MB와 비슷한 움직임 벡터를 추정한다. 또한 움직임 벡터는 특정 경향성을 가지므로 같은 방향의 움직임 벡터가 계속 결정될 확률이 높다. 또한 예측 벡터를 그대로 사용하는 Mode 1, 2, 3이 많은 결정 Mode가 된다.

그림 10. 'foreman' 35번째 프레임에서의 결정 모드
Fig. 10. Decided Mode of 'Foreman' 35th frame.

예를 들어 그림 10은 foreman 영상의 35번 째 프레임에서 결정된 모드를 나타낸 것인데 81%가 상위 모드로 선택됨을 알 수 있다. 상위 모드가 가장 많이 선택되는 것이 본 알고리즘에서 PSNR 변화가 적은 이유 중 하나가 된다. 따라서 기존 종속적인 전 영역 예측과 제안된 독립적인 예측에 영상의 품질에는 차이가 거의 없게 되고 연산 양이 줄어들게 된다.

## IV. 실험 결과

인코더를 하드웨어로 구현 할 시 가장 중요한 사항은 MB 당 타이밍(연산 시간) 문제와 하드웨어의 크기, 전력 사용 문제이다. 이는 움직임 추정에 필요한 탐색 사이클 수와 연산 양 추정을 통해 비교해 볼 수 있다. 물론 빠른 추정과 적은 하드웨어 사용을 하면서도 기존 전 영역 탐색에 비해 PSNR 저하가 거의 없어야 고속 추정에 의미가 있다. 한 지점에서 주어진 특정 Mode의 Sub MB 크기만큼 빼고 절대 값을 취해 더하여 라그랑제 상수 값과 합하게 되면 한 지점에서 특정 Mode의 Cost 값이 구해지게 된다. 연산 양 비교를 위해 이러한 연산 회수를 각각 비교하면 연산에 필요한 하드웨어 크기와 파워 사용량을 추정할 수 있다. (여기서 기준 단위를 16x16 단위의 MB를 일련의 과정을 통해 Cost를 구하는 것을 단위 1이라고 한다.)

표 2는 기존 Full search 방식과 제안된 방식을 사용한 Full search의 연산 양을 비교한 표이다. 제안된 움직임 벡터 예측과 SAD를 사용한 결과 기존 전 영역 추정에 비해 80%의 연산양이 감소하였음을 알 수 있다.

대기 사이클 수는 움직임 추정에 필요한 최소 탐색 횟수로 SAD를 최대한 공유하고 병렬 처리 과정 없이 탐색하는 하나의 움직임 벡터 지점을 의미한다. 이 경우 각각의 과정(차이 값 결정, SAD 연산, Cost 연산, Cost 비교 과정)에서 빠른 처리를 위해 파이프라인 방식을 고려해야 하므로 모든 stage는 동일한 클럭이 소요된다고 가정할 때 대기 사이클 당 N 클럭이 소요된다고 하면 한 MB의 최적 Cost와 움직임 벡터를 찾는

표 2. 연산양 비교
Fig. 2. Computation Comparison.

| | 연산양 (15x15) | 연산양 (7x7) |
|---|---|---|
| Full Search | 4805 | 1125 |
| Proposed | 961 | 225 (80%↓) |

표 3. 전 영역 탐색을 하는 기존 방식과 제안된 방식의 성능 비교
Table 3. Performance comparison between previous method and proposed method.

| 영상 | Full Search | | | | Proposed | | | |
|---|---|---|---|---|---|---|---|---|
| | Y PSNR (dB) | U PSNR (dB) | V PSNR (dB) | Bit (Bytes) | Y PSNR (dB) | U PSNR (dB) | V PSNR (dB) | Bit (Bytes) |
| grandma.qcif | 36.31 | 39.80 | 40.28 | 374864 | 36.30 | 39.81 | 40.32 | 373144 |
| | | | | | -0.01 | +0.01 | +0.04 | -0.46% |
| highway.qcif | 37.23 | 37.89 | 38.68 | 692072 | 37.23 | 37.90 | 38.73 | 696408 |
| | | | | | - | +0.01 | +0.05 | +0.63% |
| claire.qcif | 39.25 | 39.65 | 42.08 | 330480 | 39.31 | 39.62 | 42.05 | 330392 |
| | | | | | +0.06 | -0.03 | -0.03 | -0.03% |
| container.qcif | 35.61 | 40.65 | 40.25 | 468072 | 35.61 | 40.68 | 40.25 | 465360 |
| | | | | | - | +0.03 | - | -0.58% |
| carphone.qcif | 35.25 | 39.82 | 40.60 | 1149536 | 35.30 | 39.79 | 40.61 | 1173024 |
| | | | | | +0.05 | -0.03 | +0.01 | +2.04% |
| news.qcif | 35.29 | 39.82 | 40.36 | 655264 | 35.25 | 39.81 | 40.30 | 656320 |
| | | | | | -0.04 | -0.01 | -0.06 | +0.16% |
| foreman.qcif | 34.77 | 39.13 | 40.68 | 1291064 | 34.84 | 39.13 | 40.62 | 1311280 |
| | | | | | +0.07 | - | -0.06 | +1.57% |
| stefan.qcif | 30.42 | 36.77 | 36.36 | 876984 | 30.54 | 36.75 | 36.34 | 890448 |
| | | | | | +0.12 | -0.02 | -0.02 | +1.54% |
| average PSNR drop | - | - | - | - | +0.03 | -0.01 | -0.01 | +0.61% |

데 필요한 최소 클럭을 대기 사이클과 N 클럭의 곱으로 추정할 수 있다. 이러한 대기 사이클이 클수록 하드웨어로 구현할 때 타이밍 전략이 어려워지는 단점이 존재한다.

표 3에서 시뮬레이션에 사용한 영상은 grandma, highway, claire, container, carphone, news, foreman, stefan이다. 시뮬레이션에 사용한 reference code는 JM 9.6 을 사용하였다. 인코딩한 프레임 수는 stefan을 제외한 모두 300 frame을 인코딩 하였으며 (stefan은 100 frame) Baseline profile을 사용하여 CAVLC 엔트로피 코딩을 수행하였다. 또한 참조 프레임은 5프레임으로 하였다. 탐색영역은 가로, 세로 +15~-15로 하였다. [7]

시뮬레이션에 사용된 영상 중 grandma, claire 등은 적은 움직임을 갖는 영상이고 carphone, foreman이 중간 정도의 움직임 영상이다. 또 news와 stefan의 경우 움직임이 큰 영상이다. 탐색 영역(Search range)으로 사용한 -15 ~ +15 기준으로 하여 표 4의 연산 양과 대기 사이클 계산을 하였다. 표 3에서 볼 수 있듯이 제안된 방식을 사용한 결과 Y PSNR의 저하가 최대 0.04 dB에 그쳤으며 비트 증가량은 평균 0.6%가 증가하는데 그쳤다. 이는 움직임이 적거나 약간 있는 영상뿐만 아니라 highway와 stefan과 같은 많은 움직임을 갖는 영상으로 인코딩한 결과도 영상의 품질차이가 거의 없으며 Bit 증가량도 그리 높지 않음을 알 수 있다. 그에 비해 연산 양과 대기 사이클은 기존 전영역 움직임 추정에 비해 각각 1/5 (80% 감소), 1/41 (약 98% 감소)에 달하므로 인코딩 속도가 빠르고 하드웨어 구현에 효율적임을 알 수 있다. grandma나 claire 같이 움직임이 작은 영상의 경우 PSNR 변화는 거의 없고 오히려 Bit 수가 감소하였다. 이는 일반적인 움직임 추정을 행할 때

표 4. 기존 고속 탐색 알고리즘과의 비교
Table 4. Performance comparison between fast algorithms and proposed method.

| | 연산 양 (MV +16 ~ -16) | 연산 양 (MV +7 ~ -7) | 대기 사이클 | Average Y PSNR drop (MV +16 ~ -16, dB) |
|---|---|---|---|---|
| Full Search | 4805 (961 × 5) | 1125 (225 × 5) | 41 (41 서브모드) | - |
| NTSS | 231 (33 × 7) | 231 (33 × 7) | 123 (3 × 41) | -0.09 |
| DS | 511 (73 × 7) | 189 (24 × 7) | 492 (12 × 41) | -0.11 |
| Hexagon | 329 (47 × 7) | 126 (18 × 7) | 492 (12 × 41) | -0.15 |
| UMHexagonS | 693 (99 × 7) | 693 (99 × 7) | 287 (7 × 41) | -0.03 |
| Proposed FS | 961 | 225 | 1 (독립) | +0.06 |

주변 움직임 벡터의 경향과 크게 다른 움직임 벡터가 발생하고 근처 움직임 벡터가 이를 움직임 예측으로 참조하게 되는 현상이 발생한다. 이러한 특정 움직임 추정 지점에서 벡터 에러 확산 현상이 발생하는 경우 화질의 열화가 크게 될 수 있다. 하지만 제안된 움직임 예측을 통한 참조의 경우 매크로 블록 내에서는 이러한 현상이 줄어들게 되고 이로 인해 움직임 벡터의 평탄화가 이루어지게 되어 Bit 양이 줄기 때문이다.

표 4에서는 기존 고속 탐색 알고리즘과의 비교에는 문헌에서 가장 많이 인용되는 적은 움직임을 갖는 claire 영상, 중간 움직임을 가지는 carphone 영상과 foreman 영상, 많은 움직임을 가지는 stefan 영상을 PSNR 비교에 사용하였다.[8]~[10] 고속 탐색모드는 모드별로 각각 다른 지점을 최적 값으로 찾아 가므로 최대 탐색 지점에서 7가지 모드만큼 곱한 값이 연산 양이 된다. 전 영역 추정의 경우 모드 1~3을 동시에 추정할 수 있으므로 5가지 모드만큼 곱한 값을 연산 양으로 사용한다. 또한 모든 모드가 다른 최적 값을 찾아가므로 탐색 포인트는 41가지의 Sub 매크로 블록이 모두 다르게 된다. 따라서 탐색 영역의 41 배가 탐색 포인트 수가 된다. 움직임 추정 시간이 case에 따라 다르게 변하여 평균 인코딩 속도를 올리는 움직임 추정 방식의 경우 실시간으로 인코딩하는 하드웨어를 구현하기 위해서는 어느 경우에서든 실행이 되어야 하므로 최악의 경우(Extreme case)를 고려해야 한다. 따라서 DS (Diamond Search)나 Hexagon 방식의 경우 가장 멀리 있는 움직임 벡터((+15, +15) 지점)를 직선 이동(수직 이동 후 수평 이동)으로 찾는 경우를 extreme case로 정하고 연산 양을 계산하였다. 또한 대표적인 Fast ME 방식인 UMHexagonS with Early termination과 같은 기법이나 다른 Fast ME 방식들과 Early Termination을 조합한 경우 Extreme case에서는 Early termination을 사용하지 않은 방식과 같거나 오히려 비교를 위해 더 많은 시간을 사용하게 되므로 비교에서 제외하였다.[11]

표 4에서 보는 바와 같이 기존 고속 탐색 알고리즘의 경우 NTSS. DS, Hexagon 방식의 경우 연산양의 감소는 크지만 영상의 품질(PSNR) 저하가 크게 되는 것을 알 수 있다. 또한 종속적인 탐색으로 인해 이전 수행이 끝날 때까지 기다려야 하는 시간인 대기 사이클 수가 많이 필요해진다. 이로 인해 하드웨어로 구현할 때 파이프라인 설계나 병렬 설계 등 Timing 설계가 힘들어지게 된다. UMHexagonS 방식의 경우 다른 고속 추정 알고리즘에 비해 영상의 저하는 거의 없지만 99개의 Search point를 한꺼번에 나누어 찾는 것이 아닌 7번에 나누어 찾게 되므로 대기 사이클이 크게 된다.

## V. 결 론

본 논문에서는 기존 전 영역 탐색의 경우 움직임 벡터 예측이 종속적으로 이루어지는 단점을 없애기 위해 독립적인 움직임 벡터 예측이 가능하도록 새로운 움직임 벡터 예측 방법을 제안하였다. 또한 독립적인 움직임 예측이 되는 경우 움직임 추정 시 SAD 공유를 통해 SAD를 재사용하는 방법을 소개하였다. 이러한 움직임 벡터 예측과 SAD 공유를 통해 모든 모드를 동시에 계산할 수 있게 되어 움직임 추정에 필요한 연산 양과 하드웨어 복잡 도를 줄이고 움직임 추정에 필요한 클럭 사이클 수도 줄이게 되는 효과를 가져다준다. 제안된 방식을 실험한 결과 기존 전 영역 추정에 비해 연산 양은 80% 줄이면서도 비트 증가율은 0.61%, PSNR 감소

는 0.04dB 이하로 변화가 거의 없음을 본 논문을 통해 알 수 있다. 본 논문의 하드웨어 효율성을 증명하기 위해 본 논문을 바탕으로 HD급 고화질 영상에서도 실시간 움직임 추정이 가능한 움직임 추정기와 이를 적용한 H.264/AVC 부호화기를 SoC로 구현할 예정이다.

## 참 고 문 헌

[1] Reoxiang Li, Bing Zeng and M.L. Liou, "A new three-step search algorithm for block motion estimation", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, Issue 4, Page(s):438 - 442, Aug. 1994.

[2] Jiangfeng Xu, Zhibo Chen and Yun He, "Efficient Fast ME Predictions and Early-termination Strategy Based on H.264 Statistical Characters", Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Volume 1, 15-18, Page(s):218 - 222, Dec. 2003.

[3] Lai-Man Po, Wing-Chung Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation in Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, Volume 6, Issue 3, Page(s):313 - 317, June 1996.

[4] Ce Zhu, Xiao Lin and Lap - Pui Chau, "Hexagon-Based Search Pattern for Fast Block Motion Estimation", IEEE Transactions on Circuits and Systems for Video Technology. Volume 12, Issue 5, Page(s):349 - 355, May 2002.

[5] Rahman, C.A, Badaway, W., "UMHexagonS algorithm based motion estimation architecture for H.264/AVC", Fifth International Workshop on System-on-Chip for Real-Time Applications, 2005. Proceedings. Page(s):207 - 210, 20-24 July 2005.

[6] Shan Zhu, Kai-Kuang Ma, "A new diamond search algorithm for fast block-matching motion estimation", IEEE Transactions on Image Processing, Volume 9, Issue 2, Page(s):287 - 290, Feb. 2000.

[7] Arizona State University YUV Image Sequences "http://trace.eas.asu.edu/yuv/index.html".

[8] Xiang Li, Eric Q. Li and Yen-Kuang Chen, "Fast Multi-frame Motion Estimation Algorithm with Adaptive Search Strategies in H.264", IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). Vol. 3, 17-21, Page(s):iii - 369 - 72, May 2004.

[9] 김미영, "움직임 벡터의 빠른 추정을 위한 HDS 기법", 한국해양정보통신학회 논문지 8(2)권, pp. 338-343, 2004년

[10] 최웅일, 전병우, "H.264 표준의 가변 움직임 블록을 위한 고속 움직임 탐색 기법", 대한전자공학회 논문지, 제 41권 SP편 6호, 209-220 쪽, 2004년 11월

[11] 이융기, 이영렬, "가변 크기 블록에서 정수단위 화소 움직임 벡터의 빠른 검색", 대한전자공학회 논문지, 제 40권 SP편 5호, 388-396 쪽, 2003년 9월

## 저 자 소 개


**최 진 하**(정회원)

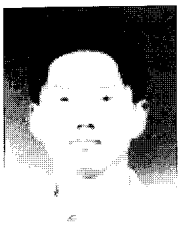
2005년 연세대학교 전기전자 공학부 학사 졸업.
2007년 연세대학교 전기전자 공학과 석사 졸업.
2007년~현재 연세대학교 전기 전자공학과 박사 과정

<주관심분야 : H.264/AVC, ISP, SoC 설계>

**김 재 석**(정회원)

1977년 연세대학교 전자공학과 학사 졸업.
1979년 한국과학기술원 전기 및 전자공학과 석사 졸업.
1988년 Rensselaer Polytechnic Institute 전자공학과 박사 졸업.
1993년~1995년 한국전자통신연구원 책임 연구원.
1995년~현재 연세대학교 전기전자공학과 교수.

<주관심분야: 통신 및 영상 시스템, VLSI 신호 처리, 임베디드 S/W 및 SoC 구현>



**이 원 재**(정회원)

2001년 연세대학교 전자공학과 학사 졸업.
2003년 연세대학교 전기전자 공학과 석사 졸업.
2003년~현재 연세대학교 전기 전자공학과 박사과정

<주관심분야: H.264/AVC, ISP, SoC 설계>