논문 2006-43SP-4-7

미디어프로세서 상의 고속 움직임 탐색을 위한 Hexagon 모양 라인 탐색 알고리즘

(Hexagon-shape Line Search Algorithm for Fast Motion Estimation on Media Processor)

정 봉 수*, 전 병 우**

(Bongsoo Jung and Byeungwoo Jeon)

요 약

대부분의 고속 블록 움직임 추정 알고리즘은 탐색점의 수를 줄여서 연산량을 감소시킨다. 하지만 이러한 고속 움직임 추정 알고리즘들은 비정규화 데이터 흐름 때문에 멀티미디어 프로세서에서는 좋은 성능을 보이기 어렵다. 미디어 프로세서에서는 내부 메모리에서 데이터의 효과적인 재사용이 SAD 명령의 수를 줄이는 것보다 더욱 중요하다. 이는 수행 사이클의 성능이 외부 메모리 액세스의 횟수에 매우 의존적이기 때문이다. 따라서 본 논문에서는 내부 메모리로부터 데이터를 효과적으로 재사용할 수 있는 라인 탐색 패턴과 라인 탐색 패턴에서 불필요한 SAD 연산을 줄이기 위한 서브 샘플링 방법을 적용한 Hexagon 모양 라인 탐색(Hexagon-shape line search, HEXSLS) 기법을 제안한다. 모의실험을 통하여 HEXSLS기법의 MAE 성능은 전역 탐색 블록 정합(FSBMA) 기법과 비슷하고, Hexagon 기반 탐색(Hexagon-based search) 보다 우수한 성능을 가짐을 보인다. 또한 HEXSLS는 Hexagon 기반 탐색이나 예측 라인 탐색(predictive line search) 기법보다 적은 외부 메모리의 액세스가 발생한다. 결과적으로, 제안한 HEXSLS 기법은 종래의 기법과 비교하여 미디어 프로세서에서 매우 낮은 수행 사이클 성능을 보인다.

Abstract

Most of fast block motion estimation algorithms reported so far in literatures aim to reduce the computation in terms of the number of search points, thus do not fit well with multimedia processors due to their irregular data flow. For multimedia processors, proper reuse of data is more important than reducing number of absolute difference operations because the execution cycle performance strongly depends on the number of off-chip memory access. Therefore, in this paper, we propose a Hexagon-shape line search (HEXSLS) algorithm using line search pattern which can increase data reuse from on-chip local buffer, and check sub-sampling points in line search pattern to reduce unnecessary SAD operation. Our experimental results show that the prediction error (MAE) performance of the proposed HEXSLS is similar to that of the full search block matching algorithm (FSBMA), while compared with the hexagon-based search (HEXBS), the HEXSLS outperforms. Also the proposed HEXSLS requires much lesser off-chip memory access than the conventional fast motion estimation algorithm such as the hexagon-based search (HEXBS) and the predictive line search (PLS). As a result, the proposed HEXSLS algorithm requires smaller number of execution cycles on media processor.

Keywords : Fast motion estimation, multimedia processors, sub-sampling line pattern, memory access, data reuse

I. 서 론

블록 정합 움직임 추정은 MPEG-1/2/4 와 ITU-T H.261/263/264 같은 여러 영상 압축 표준에서 프레임간의 시간적 중복성을 없애기 위하여 널리 사용되는 방법으로써, 동영상 압축효율에 가장 큰 영향을 주는 부분이다. 하지만 움직임 추정은 부호화기에서 가장 많은 계산량을 요구하는 부분이기도 하다[15][16]. 일반적으로 블록 정합 왜곡은 연산이 간단한 식 (1)의 SAD (sum of absolute difference)를 사용한다.

* 학생회원, ** 정회원, 성균관대학교 정보통신공학부
(Sungkyunkwan University, School of Information and Communication Engineering)
접수일자: 2005년12월13일, 수정완료일: 2006년6월27일

$$SAD(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|a(m,n)-b(m+x,n+y)|, \quad -P \le (x,y) \le P-1 \tag{1}$$

여기서 $a(m,n)$은 현재 블록, $b(m,n)$은 탐색 지점의 참조 블록, M,N은 블록의 수평, 수직의 크기, P는 탐색 영역의 크기를 나타낸다. 그리고 (x,y)는 탐색 영역에서 탐색 위치이다. 여기서 최소 SAD값을 가지는 (x,y)변위가 최적의 움직임 벡터로 선택이 된다.

움직임 추정 방법 중 전역탐색 블록 정합(Full search block-matching algorithm, FSBMA)방법은 탐색영역 내의 모든 위치를 탐색하기 때문에 최적의 움직임 벡터를 찾을 수 있다. 하지만 전역 탐색에서 하나의 블록에 대하여 탐색을 수행할 때 ADD, ABS, SUB 연산을 필요로 하는 SAD계산이 $(2P+1)^2 \times M \times N$번과 같이 요구 되어 실시간 동영상 압축 시스템에 적용하기에는 문제가 있다. 따라서 실시간 동영상 압축 시스템을 구현하기 위하여 고속 움직임 추정 알고리즘의 연구가 활발히 진행 되었다.

대부분의 고속 움직임 추정 알고리즘은 다양한 탐색 패턴을 이용하여 탐색점을 줄이는 연구가 대부분이다. 대표적으로 3단계탐색(TSS)[1], 4단계탐색(4SS)[2], 다이아몬드탐색(DS)[3][4], Hexagon 탐색(HEXBS)[5], 그리고 CDS (Cross-Diamond Search)[6] 알고리즘들이 있다. 이러한 기법들은 Unimodal 오류 표면(Unimodal error surface) 의 가정과 움직임 벡터의 Center-biased 특징을 기초로 탐색점의 수를 줄이는 것이다. 하지만, 영상의 물체가 빠르거나 복잡한 움직임을 가지는 경우 국부 최소점에 쉽게 빠지기 때문에 영상의 화질저하가 크게 발생한다. 이러한 문제점을 해결하기 위해 움직임 벡터의 시/공간 상관성을 이용하여 고속 움직임 추정을 수행하는 알고리즘이 연구되었다[7][8][9]. 하지만 이러한 고속 움직임 추정 알고리즘들은 복잡한 제어 명령(Control operation)과 탐색 패턴의 의존성 때문에 하드웨어 구현이나 병렬 처리가 가능한 미디어 프로세서들에 사용되기에는 비효율적이다. 또한 탐색 패턴의 의존성으로 참조 프레임이 위치한 외부메모리의 접근이 빈번히 발생하여 탐색 속도의 저하를 가져온다.

최근 Huang[11]은 미디어 프로세서에서 외부메모리의 접근을 줄이기 위하여 라인 탐색 패턴(line search pattern)을 제안하였다. 하지만 라인 탐색 패턴은 탐색 영역의 수평라인의 모든 지점의 SAD를 계산하는 단점과 수직 움직임 벡터가 많이 발생하는 영상에서는 여전히 외부 메모리의 접근이 많이 발생하는 문제점이 있다.

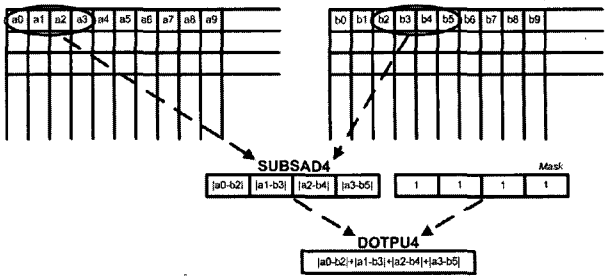


그림 1. TMS320C64x™에서 4개의 화소데이터를 위한 SAD 계산 예

Fig. 1. Illustration of SAD calculation for four 8-bit pixel data on TMS320C64x™.

본 논문에서는 Hexagon 모양을 기반으로 움직임 탐색 성능을 높이고 외부 메모리 접근을 줄이기 위해 라인 탐색 방법을 적용하며, 또한 라인 탐색 패턴의 탐색 점수와 외부메모리 접근 횟수를 줄이기 위한 Hexagon 모양 라인 탐색(HEXagon-Shape Line Search, HEXSLS) 기법을 제안한다. 제안한 기법을 Texas Instruments사의 고성능 미디어 프로세서인 TMS320C6415™에 구현하여 성능을 평가한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 미디어 프로세서의 일반적인 특징을 기술 하고, Ⅲ장에서는 기존의 고속 움직임 추정 알고리즘을 나타내고, Ⅳ장에서는 제안 알고리즘을 설명하고, Ⅴ장에서는 제안 알고리즘의 성능을 보이고, 마지막으로 결론을 맺는다.

Ⅱ. 미디어 프로세서의 특징

일반적으로 미디어 프로세서는 여러개의 명령어를 병렬로 처리하기 위해 VLIW (Very Long Instruction Word) 구조를 가진다. 멀티미디어 데이터의 처리시에는 단순 반복적인 연산을 하는 경우가 많이 발생한다. 따라서 루프문 내에서 단순 반복적인 연산을 수행할 때 조건 연산에 의한 오버헤드를 줄이기 위해 제로-오버헤드 루프 제어(zero-overhead loop control) 기능을 가지고 있다.

SAD계산 량을 줄이기 위한 미디어 프로세서의 주요한 특징은 다음과 같다[10][11].

1) 메모리 접근과 명령어의 패치를 위한 넓은 데이터 패스를 가진다.

2) 계산 중심의 멀티미디어 태스크를 위한 특별한 명령어 집합을 가지고 있다.

표 1. SAD연산과 외부메모리접근의 소요 사이클 비교
Table 1. Cycle counts comparison of SAD operation and off-chip memory access.

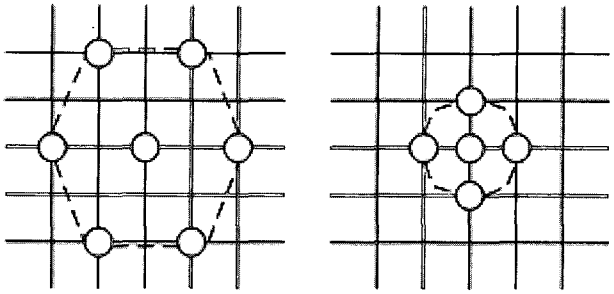
명령	매크로블록의 SAD연산	매크로블록의 외부메모리접근	47x16블록의 외부메모리접근
사이클 (Cycles)	75	3248	9910

3) 여러 개의 데이터를 병렬로 처리하기 위한 SIMD (Single Instruction and Multiple Data) 명령어 구조를 가지고 있다.

TI TMS320C64x™ 미디어 프로세서의 경우 64 비트 데이터 패스를 가지고 있으며, 이것은 8개의 8비트 화소 데이터를 한 번에 액세스할 수 있다. 또한 VLIW 구조는 8개의 기능 유닛을 가지고 있고, 이러한 기능 유닛에서 몇 가지 명령들은 SIMD 구조를 가지고 있다[14]. 예를 들어, SUBABS4 명령어는 4개의 화소데이터에 대해 뺄셈과 절대치 연산을 한 번에 수행 할 수 있을 뿐만 아니라, 이 작업은 하나 이상의 기능 유닛에서 동시에 수행 될 수 있다. 따라서 미디어 프로세서는 블록 정합 오류를 구하기 위한 SAD 명령을 일반적인 마이크로프로세서보다 훨씬 효과적으로 계산할 수 있다.

그림 1은 TI TMS320C64x™ 미디어 프로세서에서 SAD 명령을 SUBABS4와 DOTPU4 명령을 사용하여 수행하는 예를 도시하였다. 먼저, 메모리에 위치한 4개의 8비트 데이터를 SUBABS4명령어를 이용하여 병렬로 뺄셈과 절대치 연산을 수행한다. 그 다음 DOTPU4 명령어로 4개의 연산결과를 하나의 결과로 더한다. 이렇게 화소데이터를 4개씩 움직이면서 SAD 연산하여 하나의 블록에 해당하는 정합오류를 구할 수 있다. 따라서 16x16 매크로블록의 경우, 64번의 SUBABS4연산, 64번의 DOTPU4연산, 그리고 64번의 ADD연산이 필요하다. 하나의 연산에 1사이클이 필요하다고 가정한다면, 매크로블록의 SAD연산에 192사이클이 필요하게 된다. 하지만 미디어 프로세서는 데이터간의 의존성이 없으면, 명령어를 병렬로 처리 가능하기 때문에 더욱 적은 사이클을 소모하게 된다.

표 1은 블록 정합 오류 계산에 소모되는 SAD연산의 사이클 수와 외부 메모리(off-chip memory) 액세스 사이클 수를 TMS320C6415™에 구현하여 비교한 것이다. 수행 사이클의 수를 보면 하나의 매크로블록의 블록 정합 오류를 계산하는데 외부 메모리 액세스에 필요한 사이클이 SAD 연산 사이클과 비교하여 약 40배 이상 소모됨을 알 수 있다. 여기서 SAD연산 사이클은 SAD연산을 위한 데이터가 내부메모리에 위치한 다음의 결과이다. 이것은 움직임 추정에서 외부 메모리 접근 횟수를 줄이는 것이 매우 중요함을 나타낸다. "47x16블록의 외부메모리접근"은 수평 탐색영역의 크기가 32인 경우 하나의 수평라인 위치의 SAD 연산을 위하여 외부 메모리 액세스에 필요한 사이클 수를 나타낸 것으로 하나의 매크로블록의 외부메모리접근 사이클 수와 비교하여 약 3배 소모됨을 알 수 있다. 이것은 32개 위치의 SAD 연산을 위하여 한 번에 외부메모리를 액세스하여 내부메모리에 위치시켜 SAD 연산을 수행한다면 매우 많은 외부메모리 접근 횟수를 줄일 수 있음을 알 수 있다.



(a) LHSP (b) SHSP

그림 2. Hexagon 탐색의 두가지 패턴
Fig. 2. Two types of hexagon-based search pattern.

움직임 추정 알고리즘을 더욱 효과적으로 병렬 처리하기 위해서는 탐색 패턴의 의존성이 적어야하고 복잡한 제어명령의 사용으로 인한 복잡도가 적어야한다. 하지만 Hexagon 탐색 기법과 같은 일반적인 고속 움직임 추정 방법은 다음의 탐색 지점이 현재의 탐색 결과에 의존적이기 때문에 다음 탐색 지점의 데이터를 미리 가져 올 수 없다. 따라서 외부 메모리(off-chip memory)의 접근을 줄이기 위해서는 정규화된 탐색 패턴을 가져야 미디어 프로세서의 내부 메모리(on-chip memory)에서 데이터를 효과적으로 재사용할 수 있다.

Ⅲ. 기존 고속 움직임 탐색 기법: Hexagon 탐색, 예측 라인 탐색 (PLS)

기존 제안된 고속 움직임 탐색 기법으로는 간단한 Hexagon 탐색 (Hexagon-based Search, HEXBS) 기법을 들 수 있다[5]. 그림 2에서 보는 바와 같이 HEXBS 기법은 탐색의 중심 화소와 그 중심으로부터 큰 육각형 모양으로 이루어진 LHSP(Large Hexagon Search Pattern)와, 중심 화소로부터 거리가 1의 화소들로 이루

어진 SHSP(Small Hexagon Search Pattern)의 두 가지 탐색 패턴을 가지고 있다. LHSP는 중심을 포함하여 7개의 탐색 위치 점을, SHSP는 중심을 포함하여 5개의 탐색 위치 점을 갖고 있으며 둘 다 모두 육각형 모양을 갖는 것을 특징으로 한다.

주어진 블록에 대하여 참조 영상 내에서 최적 움직임 벡터를 찾기 위해 영상을 x,y 좌표계로 표현할 때 움직임 벡터가 (0,0) 지점을 시작점으로 하여 최소 SAD를 갖는 지점을 찾아 나간다. 먼저 LHSP의 중심점을 (0,0)으로 하여 각 9개 지점에 대하여 최소 SAD를 갖는 지점을 조사한 다음, 그 지점을 다시 LHSP의 중심점으로 하여 동일한 과정을 반복한다. 이러한 반복 과정을 수행하다가 최소 SAD를 갖는 지점이 LHSP의 중심점이 될 경우에는 그 위치를 SHSP의 중심점으로 하여 마지막으로 최소 SAD를 갖는 지점을 조사하고 그 지점을 최적 움직임 벡터로 정한다.

HEXBS 기법의 경우 초기 LHSP의 탐색에서 다음 번의 LHSP를 구성할 경우, 추가로 항상 3개의 탐색 위치만을 새롭게 구성하여 조사하면 된다. 이것은 DS (Diamond Search)기법이 최소 SAD 지점에 따라 탐색 위치점의 개수가 달라지는 것과는 차이가 있다. HEXBS 기법은 DS 기법과 비교하여 좀더 정규화된 탐색 패턴이라고 할 수 있다. 하지만, 여전히 다음의 탐색 작업이 현재 탐색 단계에서의 최소 SAD 지점에 따라 달라지기 때문에 다음 탐색 패턴 위치의 데이터를 미리 가져 올 수 없는 문제점이 있다. 또한 (0,0) 지점을 중심으로 탐색을 시작하기 때문에 비교적 움직임이 작은 영상의 경우에는 LHSP의 반복 탐색 수행 횟수가 적지만 움직임이 큰 영상의 경우에는 수행 횟수가 증가하게 되어 외부 메모리 접근이 많이 발생하여 속도 향상 측면에서 효율이 떨어지게 된다. 또한 탐색 시작점으로부터 정합 오류가 최소가 되는 지점까지 거리가 멀 경우, 탐색 도중 국부 최소치(local minimum)에 빠져 최적 움직임 벡터를 찾는데 실패할 확률도 커진다.

이러한 문제점을 극복하기 위해 최적 움직임 벡터를 찾을 확률을 높이고, 정규화된 탐색 패턴을 구성하여 외부 메모리 접근을 줄인 예측 라인 탐색(Predictive Line Search, PLS) 기법이 제안되었다[11]. PLS의 구체적 알고리즘은 다음과 같다. 먼저 예측 움직임 벡터 지점을 지나는 탐색 영역 크기의 하나의 수평라인을 p라고 하면, p-1, p, p+1의 3개의 라인내의 모든 위치를 조사한다. 그리고 최소 SAD를 가지는 방향으로 추가 라인을 조사한다. 만약 최소 SAD 지점이 조사한 3개의

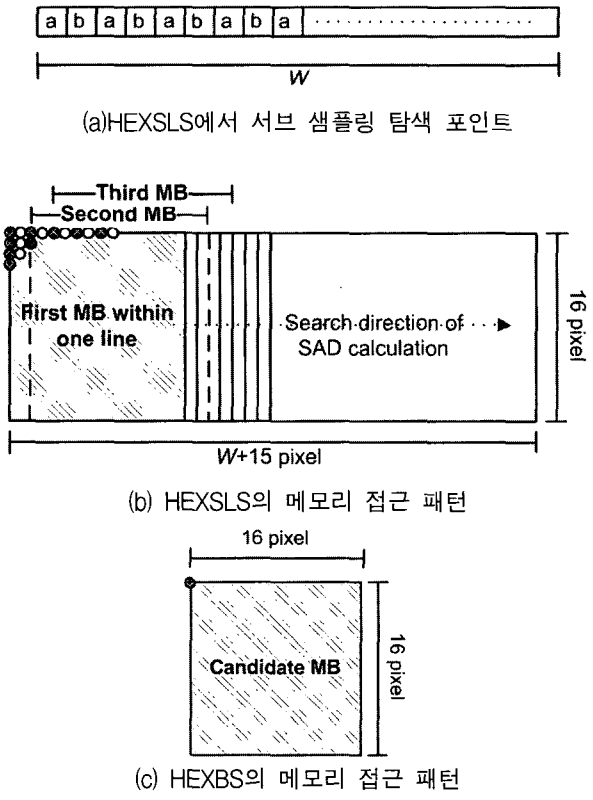


그림 3. 서브 샘플링과 메모리 액세스 패턴
Fig. 3. Sub-sampling and memory access patterns.

라인 중에서 중심 라인에 위치하지 않으면 라인 탐색을 계속 수행한다. 중심 라인에서 최소 SAD 지점이 찾아질 때, 탐색을 중단하고 그 지점을 최적의 움직임 벡터로 선택한다.

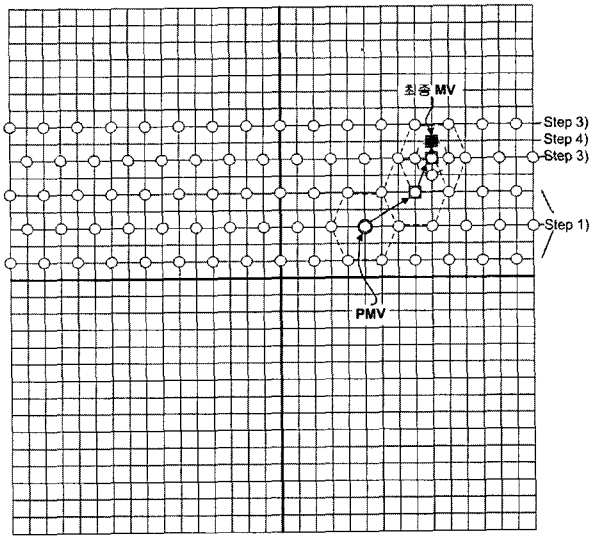
이러한 PLS 기법은 공간상의 움직임 벡터간의 상관도를 이용하여 종래 Center-biased 기법들에 비해 부호화 효율을 높이고, 탐색 영역에서 하나의 수평라인에서 효과적으로 데이터를 재사용함으로써 외부 참조 메모리의 액세스 시간을 줄여 속도 측면에서 모두 우수한 성능을 보였다. 본 논문에서는 기존의 HEXBS 기법의 탐색 모양을 유지하면서 PLS 기법의 라인 탐색 패턴을 적용하여 외부 참조 프레임 메모리의 액세스를 줄여 속도를 높이고, Hexagon 패턴을 수평 방향으로 더욱 많이 탐색함으로써 탐색 효율을 더욱 증가시키고자 한다. 따라서 HEXBS의 탐색 패턴과 PLS 기법을 상호 유기적으로 적용한 Hexagon 모양 라인 탐색 기법(HEXSLS)을 제안한다.

Ⅳ. 제안된 고속 움직임 탐색기법

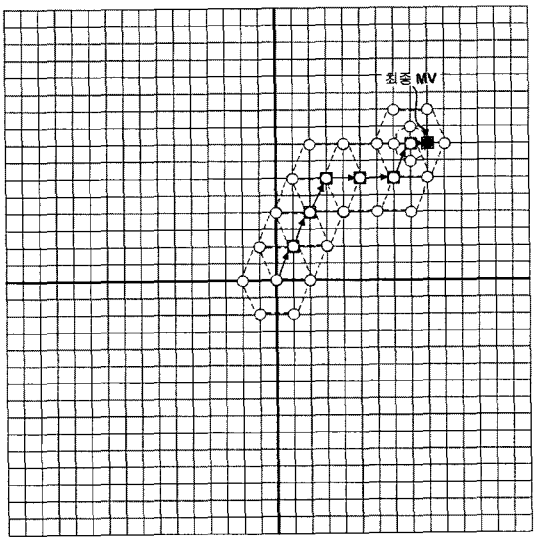
Ⅱ장에서 논의한 미디어 프로세서의 특징을 주로 고

려하여, 제안 알고리즘은 기본적으로 정규화된 탐색 패턴을 가지는 라인 탐색 방법을 이용한다. 또한 라인 탐색 패턴의 불필요한 SAD 계산을 줄이고 Hexagon 패턴으로 구성하기 위하여 하나의 수평 라인에서 서브 샘플링(sub-sampling) 위치에서만 SAD를 계산한다. 먼저, 그림 3(a)에 나타낸 것처럼 하나의 수평 라인에서 모든 위치의 SAD를 조사하는 대신 단지 "a" 혹은, "b"의 서브 샘플링 위치만을 조사한다. 만약 현재의 탐색 라인이 "a" 위치를 탐색한다면, 다음 단계의 수평 라인은 "b" 위치를 탐색한다. 즉, 수평 라인의 서브 샘플링 위치는 항상 이웃하는 수평 라인과 다른 위치를 탐색하도록 한다. 따라서 기존의 라인 탐색 알고리즘과 비교하여 하나의 수평라인에서 SAD연산량은 절반으로 감소하게 된다. 그림 3(b)와 3(c)는 제안한 HEXSLS 기법과 기존 HEXBS 기법의 외부 메모리 액세스 예를 보여준다. 여기서 탐색 영역 P를 16으로 가정한다면, W는 32가 된다. 제안한 HEXSLS 기법은 하나의 수평 라인을 탐색하기 위해, 그림 3(b)에서 보여주는 것처럼 외부 메모리에서 내부 메모리로 (32 + 15) x 16 = 752바이트 액세스가 필요하다. 그 다음, 외부 메모리의 액세스 없이 내부 메모리에서 16개의 서브 샘플링 위치를 탐색할 수 있다. 여기서, 다음 블록의 SAD 계산은 현재 블록의 SAD를 계산한 후, 내부 메모리에서 2픽셀 이동한 위치에서 계산하면 된다. 즉, 하나의 매크로블록의 SAD계산을 위해 필요한 외부 메모리 액세스는 47바이트 이다. 또한 탐색 라인상의 정규화된 패턴 때문에 HEXSLS 기법은 VLIW 구조의 명령어의 병렬성을 높일 수 있고, 제로-오버헤드 제어 구조로 조건 분기 명령에 의한 오버헤드를 줄일 수 있다. 반면, HEXBS는 단지 하나의 탐색점의 SAD를 계산하기 위해 내부 메모리로 16 x 16 = 256바이트의 외부 메모리 액세스가 필요하다. 16 위치의 탐색을 위하여 752 바이트 외부 메모리 액세스와 단지 하나의 탐색 위치를 위한 256 바이트 외부 메모리 액세스를 비교하면, HEXSLS 기법이 더욱 효과적인 메모리 액세스를 수행함을 알 수 있다. 또한 제안 알고리즘은 DMA (Direct memory access)를 이용하여 현재 라인 탐색 중에 미리 다음 탐색 라인을 외부 메모리에서 내부 메모리로 쉽게 가져 올 수 있다.

외부 프레임 메모리에서 하나의 수평라인을 접근할 때, 제안 알고리즘은 이전의 탐색한 라인에서 하나의 라인씩 건너뛰면서 수평라인을 접근한다. 이것은 움직임이 수직으로 크게 발생한 경우 외부 메모리의 접근을 줄일 수 있기 때문이다. 또한 국부 최소에 빠지는 것을 감소



(a) HEXSLS 기법



(b) HEXBS 기법

그림 4. HEXSLS와 HEXBS 기법의 예: MV(9,-8)
Fig. 4. An example of HEXSLS and HEXBS: MV(9,-8).

시키고 탐색 성능을 높이기 위하여 초기 탐색 라인을 (0,0) 지점이 아닌 예측 움직임 벡터(predicted motion vector, PMV)의 수직벡터지점을 지나는 수평 라인을 초기 탐색 라인으로 한다. 이것은 현재 매크로블록의 움직임 벡터와 이웃 블록의 움직임 벡터사이에 상관도가 높기 때문이다[16]. 예측 움직임 벡터는 현재 블록의 이웃하는 3개의 블록의 Median 움직임벡터를 취한다[16].

본 논문에서 제안하는 HEXSLS 기법의 수행 순서를 단계별로 정리하면 다음과 같다.

단계 1 (Start): 예측 움직임 벡터(PMV)의 y 성분을 p로 설정한다. 또한 라인 p상의 PMV의 x 성분이 나타

내는 위치를 초기 서브 샘플링 패턴("a" 혹은 "b")으로 설정한 후, p, p-2, p+2 수평라인에서 SAD_{min}의 위치를 찾는다. 여기서 p-2, p+2의 서브 샘플링 위치는 라인 p의 서브 샘플링 위치와 다른 위치이다. SAD_{min}이 p+2 라인에 위치하면 단계 2를, p-2 라인에 위치하면 단계 3을, 그렇지 않으면 단계 4를 수행한다.

단계 2 (Direction search of line p+2): p=p+2로 설정한다. p 라인과 다른 서브 샘플링 위치의 p+2 라인을 탐색한다. SAD_{min}이 p 라인에 위치하면 단계 4를, 그렇지 않으면 단계 2를 반복한다.

단계 3 (Direction search of line p-2): p=p-2로 설정한다. p 라인과 다른 서브 샘플링 위치의 p-2 라인을 탐색한다. SAD_{min}이 p 라인에 위치하면 단계 4를, 그렇지 않으면 단계 3을 반복한다.

단계 4 (Refinement): 이전 단계에서 찾은 최적의 위치를 SHSP의 중심으로하여 최소 SAD를 갖는 지점을 탐색한다.

단계 5 (End): 최종적으로 찾은 SAD_{min}을 가지는 위치를 최적의 움직임 벡터로 선택한다.

그림 4는 HEXSLS 기법과 HEXBS 기법의 예를 보여준다. 그림 4의 예에서 PMV는 (5,-3), 최적의 움직임 벡터는 (9,-8), 그리고 탐색 영역은 (-16,15) 라고 가정한다. 먼저 그림 4(a)에서, (단계 1) p 라인은 y=-3에 위치하는 탐색 영역의 전체 수평라인이고 탐색할 3개의 라인은 각각 -5 라인, -3 라인, -1 라인이다. 최소 SAD지점은 (8,-5) 에서 찾아졌다. (단계 3) 여기서 찾은 -5 라인은 조사한 3개의 라인들 중에 경계라인에 위치하기 때문에 추가로 -7 라인을 탐색한다. 그 다음 조사한 최소 SAD지점은 (9,-7)에 위치하였다. 이 지점은 여전히 탐색 라인의 경계위치이다. (단계 3) 따라서, p=-9인 추가 라인을 탐색한다. 그러나 이번에는 (9,-7)위치가 여전히 최소 SAD이다. (단계 4) 이 위치는 경계라인이 아니기 때문에, (9,-7) 위치에 이웃하는 SHSP 모양의 4개의 위치를 조사하고, (단계 5) 마지막 (9,-8) 위치가 최적의 움직임 벡터로 탐색되었다. 따라서 HEXSLS 기법은 5번의 라인 탐색을 수행하였다. 만약 PLS 기법으로 탐색을 하였다면, 그림 4의 예에서 8번의 라인 탐색을 수행 할 것이다. 결과적으로 HEXSLS 기법은 PLS 기법보다 3회 적은 라인 탐색을 수행하여 외부메모리 액세스를 줄임으로써 최적 움직임 벡터를 빠르게 찾을 수 있다. 그림 4(b)는 HEXBS 기법의 경우에 탐색을 수행하는 예를 보여준다. 여기서 HEXBS 기법은 SAD를 계산하는 탐색 위치의 수는 적게 발생한다. 하지만 불규칙적인 다음 단계 진행으로 인하여 사전에 외부메모리에서 내부메모리로의 데이터 이동이 어렵다. 결과적으로 외부메모리 액세스가 빈번하게 발생하여 탐색 속도의 저하를 가져온다.

그림 4에서와 같이, HEXSLS 기법은 기본적으로 HEXBS 기법의 패턴으로 구성되어 있다. 즉, HEXSLS 기법은 HEXBS의 비정규화된 탐색 기법에 대하여 탐색 전략을 변경하고 메모리 액세스 패턴을 수정하여 HEXBS 기법의 속도와 예측 성능을 최적화 한 것이라고 말할 수 있다. 비록 HEXSLS 기법이 HEXBS 기법에 비하여 더 많은 탐색 위치를 요구하지만, HEXSLS 기법은 정규화된 패턴으로 외부메모리의 액세스를 줄이고 내부 메모리에서 데이터 재사용을 높이므로 병렬처리를 지원하는, 미디어 프로세서에서 더 빠르게 최적의 움직임 벡터를 찾을 수 있다.

V. 실험 결과

제안된 HEXSLS 기법의 성능 평가를 위해 TI TMS320C6415™ 미디어 프로세서에 기존 방법으로 전역 탐색(FSBMA) 기법, Hexagon 탐색(HEXBS) 기법[5], 예측 라인 탐색(PLS) 기법[11]을 구현하여 성능비교를 하였다. 블록 정합 왜곡은 SAD를 이용하였고, 탐색 블록의 크기는 16x16으로, 탐색 윈도우는 (-16, 15)로 하였다. 모의 실험에는 동영상의 특성이 다양하게 나타나도록 7개의 시험 영상 시퀀스를 사용하였다. 사용한 시험 영상 시퀀스의 크기는 352x288, 그리고 부호화시 프레임 간격은 1로 하였다.

각 기법의 성능을 비교하기 위해 다음과 같은 성능 지표를 사용하였다.

1) 화소당 평균 절대 오차(MAE) : 원 영상과 움직임 보상된 영상간의 왜곡값으로 다음과 같이 정의 된다.

$$MAE = \frac{1}{K \times L} \sum_{m=0}^{K-1} \sum_{n=0}^{L-1} |I_{org}(m,n) - I_{pred}(m,n)|,$$

여기서 영상의 크기는 $K \times L$이고, $I_{org}(m,n)$는 원 영상, $I_{pred}(m,n)$는 움직임 보상된 영상이다.

2) 프레임당 평균 움직임 벡터 정확도(MV hit-rate) : 움직임 벡터 정확도는 FSBMA에서 찾은 움직임 벡터 값과 일치되는 움직임 벡터의 비율이다.

3) 프레임당 평균 소요 사이클(Cycle) : 각 기법들이 TMS320C6415™에서 소요되는 실제 클럭수 이다.

표 2. MAE와 움직임 벡터 정확도의 성능비교
Table 2. Performance comparison in terms of MAE and motion vector hit-rate (%).

Sequence	FSBMA		HEXBS		PLS		HEXSLS	
	MAE	Hit-rate	MAE	Hit-rate	MAE	Hit-rate	MAE	Hit-rate
Foreman	3.422	100	4.254	65.85	3.483	94.26	3.530	89.40
Stefan	8.105	100	11.288	66.63	8.289	94.33	8.309	87.97
Table Tennis	2.713	100	2.989	88.62	2.875	95.96	2.874	94.29
Tempete	5.811	100	5.916	93.73	5.879	97.46	5.887	95.62
Paris	2.113	100	2.172	97.15	2.124	99.51	2.138	98.75
Mobile and calender	7.675	100	7.958	90.93	7.681	99.58	7.707	98.29
Silent	1.791	100	1.884	93.45	1.841	97.68	1.837	96.75
평균	4.519	100	5.209	85.19	4.596	96.97	4.612	94.44

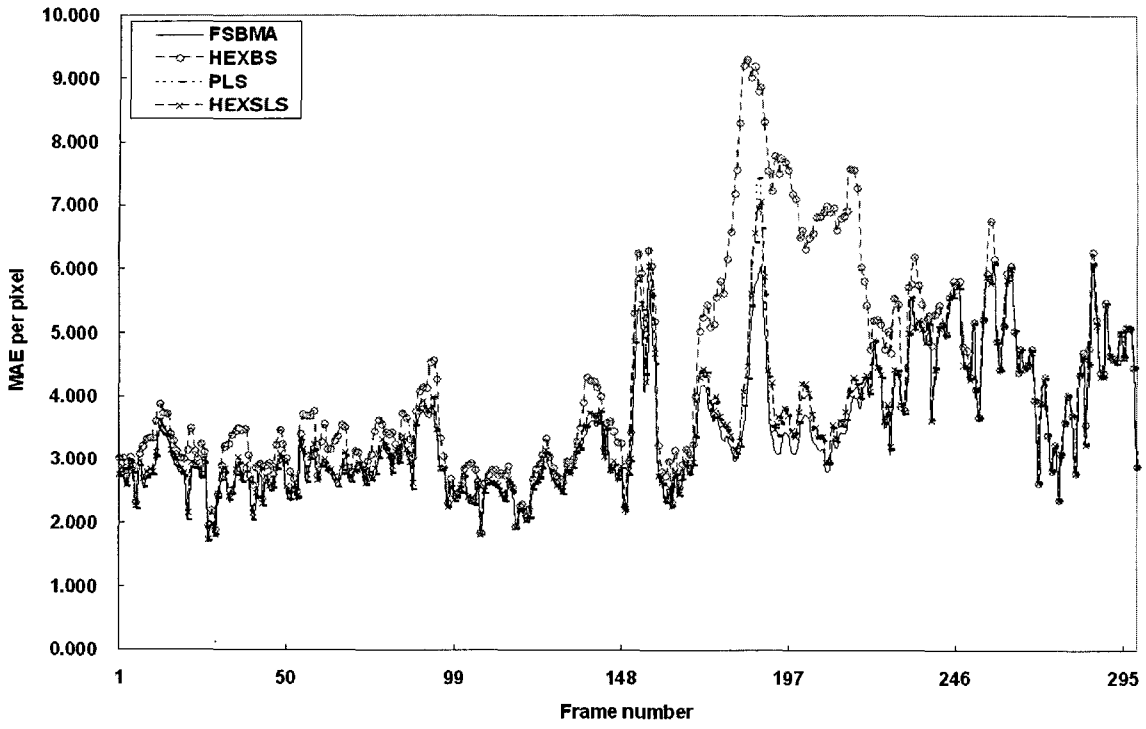
표 3. 프레임 당 평균 탐색 라인 수
Table 3. Average number of search lines per frame.

Sequence	FSBMA	PLS	HEXSLS
Foreman	12672.00	1389.87	1277.37
Stefan	12672.00	1300.42	1238.94
Table Tennis	12672.00	1297.42	1231.22
Tempete	12672.00	1218.01	1199.27
Paris	12672.00	1217.85	1192.84
Mobile and calender	12672.00	1263.72	1214.71
Silent	12672.00	1282.54	1233.13
평균	12672.00	1281.40	1226.79

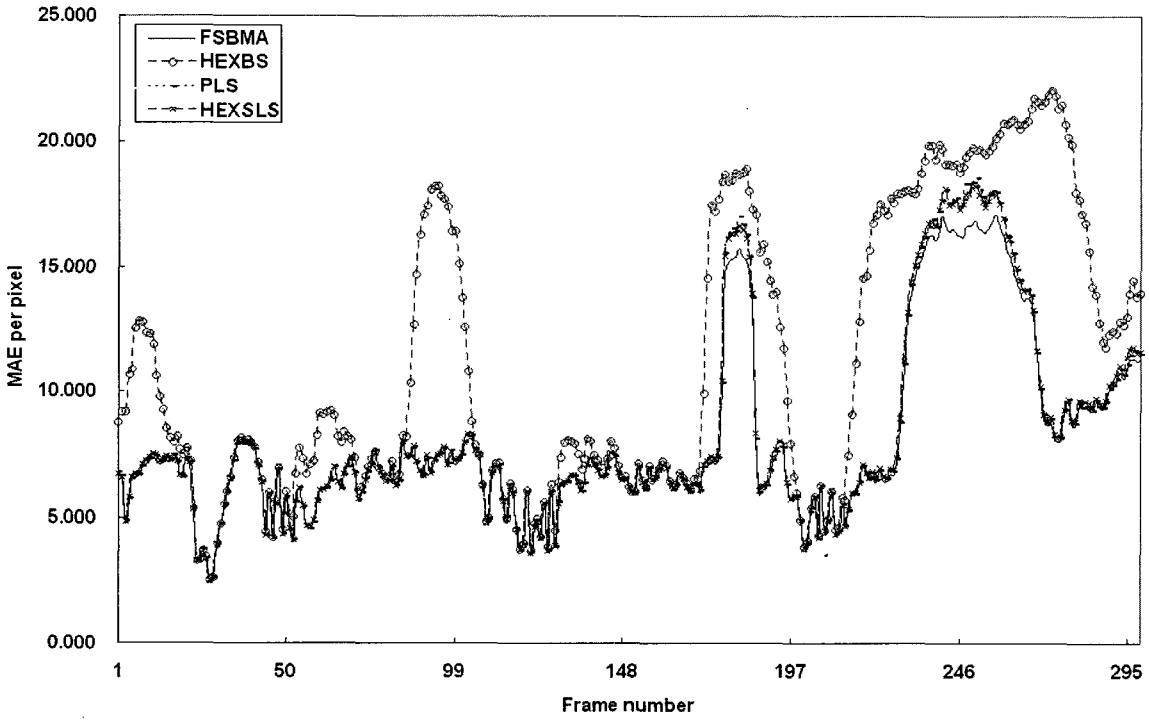
표 4. TMS320C6415^{TM}에서 프레임 당 평균과 최대 소요 사이클 수 [MCycles]
Table 4. Average and maximum cycle counts per frame under TMS320C6415^{TM} [MCycles].

Sequence	FSBMA		HEXBS		PLS		HEXSLS	
	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
Foreman	177.11	177.12	55.61	98.96	24.44	37.90	22.94	28.52
Stefan	177.11	177.12	62.85	76.68	23.24	28.69	22.27	25.72
Table Tennis	177.11	177.16	49.55	69.55	23.21	42.32	22.24	29.58
Tempete	177.11	177.12	45.72	47.39	22.56	24.23	21.87	22.58
Paris	177.11	177.12	44.97	48.31	22.13	22.86	21.79	22.10
Mobile and calender	177.11	177.12	46.71	58.38	22.13	22.89	21.81	22.14
Silent	177.11	177.12	46.97	54.63	23.00	27.41	22.21	23.99
평균	177.11	177.13	53.43	73.15	23.36	33.29	22.33	26.60
Gain	1.00	1.00	3.31	2.42	7.58	5.32	7.93	6.66

표 2는 MAE와 움직임 벡터 정확도 성능을 보여준다. Silent 와 Paris 같은 느린 움직임의 영상에 대해 모든 고속 탐색 기법들은 FSBMA와 비교하여 비슷한 MAE 성능을 보여준다. 여기서 HEXSLS의 MAE 성능은 모든 영상에 대하여 PLS와 비슷한 성능을 보이며, HEXBS와 비교하여 매우 우수한 성능을 보인다. 이것은 제안한 HEXSLS의 경우 HEXBS보다 수평 방향으로 많은 탐색을 수행하여 국부 최소점에 빠지는 경우가 적게 발생하기 때문이다. 특히, HEXSLS의 경우 Foreman, Stefan과 같은 매우 빠른 움직임의 영상에서도 FSBMA와 비슷한 MAE 성능을 보여준다. 이것은 제안한 HEXSLS기법이 다양한 영상에서 매우 효과적인 방법임을 의미한다. 움직임 벡터 정확도는 HEXSLS가 HEXBS보다 모든 영상에 대하여 높은 움직임 벡터 탐색 성능을 보여준다. 특히, 빠른 움직임의 영상에서 더욱더 좋은 성능을 보여준다. 반면, PLS와 비교하여 비슷한 움직임 벡터 정확도를 보여주며, FSBMA의 움직임 벡터에 매우 근접한 성능을 보여준다. 움직임 벡터 정확도는 MAE 성능에 영향을 미치는 요소로 정확도가 FSBMA에 근접 할수록 MAE 성능도 좋아지게



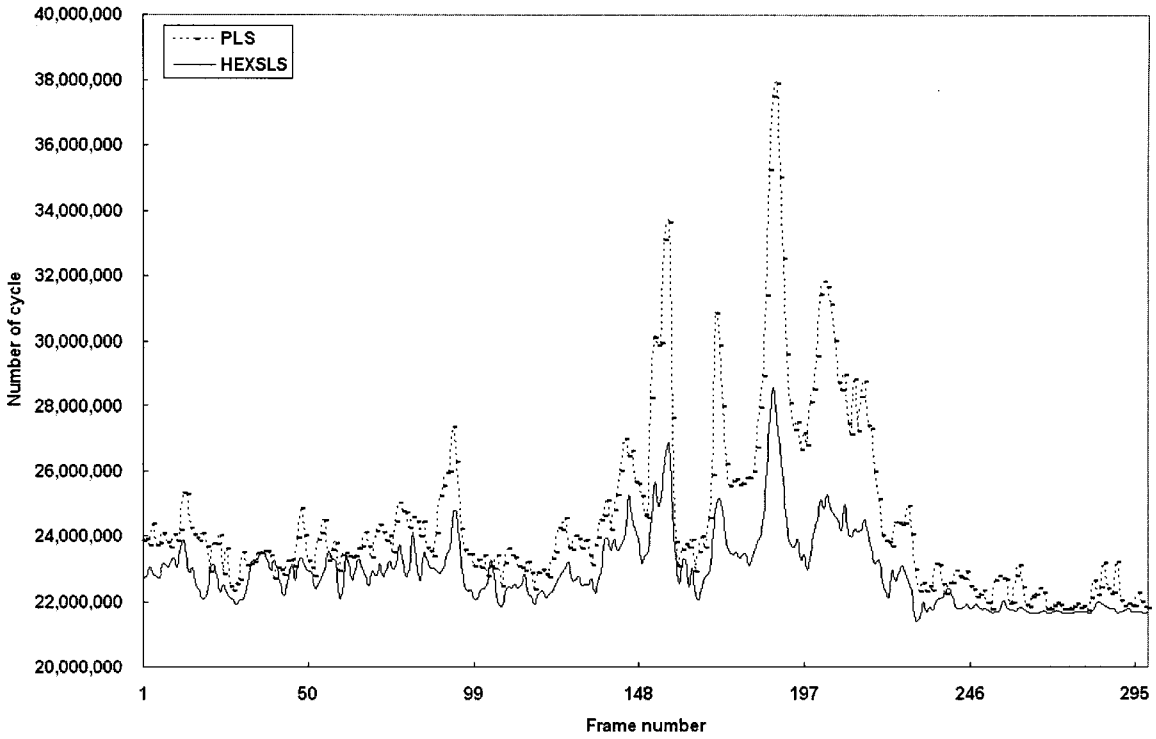
(a) Foreman 영상



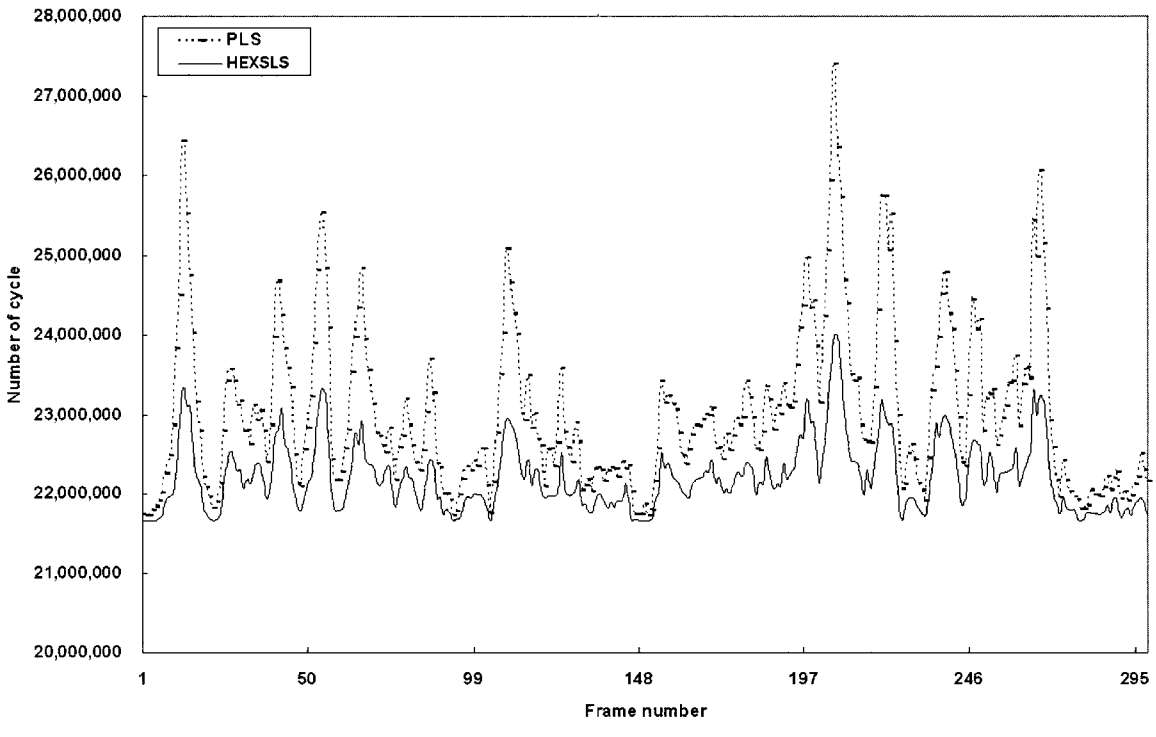
(b) Stefan 영상

그림 5. 각 프레임 별 MAE 성능 비교

Fig. 5. MAE performance comparison at each frame.



(a) Foreman 영상



(b) Stefan 영상

그림 6. 각 프레임 별 수행 사이클 비교
Fig. 6. Number of cycle comparison at each frame.

된다. 따라서 표 2의 결과에서 HEXSLS 기법은 매우 신뢰성이 높은 기법이라 말할 수 있다.

그림 5는 Foreman과 Stefan 영상에서 각 프레임별 MAE 성능을 보여준 것이다. 여기서 모든 프레임에 대하여 제안한 HEXSLS 기법은 FSBMA와 매우 비슷한 성능을 보여주는 반면, HEXBS는 각 프레임별 MAE의 성능 변화가 심하게 발생함을 볼 수 있다.

표 3은 제안한 HEXSLS 기법과 PLS 기법의 프레임당 평균 수평 라인의 접근 수를 비교한 것이다. 여기서 평균 수평 라인의 접근 수는 외부 메모리(off-chip memory)의 접근 횟수와 관련이 있다. HEXSLS가 PLS와 비교하여 더 적은 수평 라인의 접근이 발생하는 것을 알 수 있다. 또한 FSBMA와 비교하여 탐색 라인이 급격히 줄어들었음을 볼 수 있다.

미디어 프로세서에서 실제 소요되는 사이클을 측정하기 위해 움직임 추정 기법들을 TI TMS320C6415™에 구현하여 코드 컴포우저 스튜디오(code composer studio, CCS) 툴을 이용하여 측정하였다. 여기서, TMS320C6415™의 L2-data 캐쉬는 사용하지 않고 L2-SRAM 모드로 사용하였다. 일반적으로 미디어 프로세서는 메모리의 접근을 DMA를 이용하여 버스트(burst)하게 데이터를 전송하기 위해 내부 로컬 버퍼 기법을 이용하기 때문이다. 표 4를 보면, HEXSLS 기법이 모든 영상 시퀀스에 대하여 다른 고속 탐색 기법보다 항상 적은 소요 사이클이 발생하는 것을 볼 수 있다. HEXSLS는 FSBMA와 비교하여 평균 7.93배 사이클 감소를 가져오며, 가장 좋지 못한 경우에도 6.66배 사이클 감소를 가져온다. HEXBS 기법과 비교하여, 제안한 HEXSLS 기법은 더 많은 탐색 포인트를 가지지만, 평균 2.4배의 수행 사이클 감소를 가져온다. 이것은 탐색 윈도우에서 하나의 수평라인의 정규화된 데이터 흐름으로 내부메모리에서 화소를 재사용하기 때문에 적은 외부메모리(off-chip memory) 액세스가 발생하기 때문이다. 표 4의 최대 소요 사이클에서 제안한 HEXSLS 방법은 평균 소요 사이클과의 차가 크지 않지만, PLS의 경우 평균 소요 사이클과 최대 소요 사이클 사이의 차가 크게 발생한다. 실시간 비디오 코딩에서 최대 소요 사이클은 평균 소요 사이클 보다 더욱 중요하다. 따라서 HEXSLS는 PLS와 HEXBS와 비교하여 더욱 우수한 알고리즘이라고 말할 수 있다.

그림 6는 Foreman과 Silent 영상에서 각 프레임별 소요 사이클을 보여준다. 여기서 HEXSLS 기법이 PLS와 비교하여 항상 적은 사이클이 소요되는 것을 볼 수 있다. 특히, 움직임이 적은 부분에서는 PLS와 비교하여 HEXSLS 기법은 비슷한 성능을 보이지만, 빠른 움직임이나 수직방향으로 움직임이 많이 발생하는 경우 PLS의 경우 프레임별 소요 사이클이 크게 변화하는 것을 볼 수 있다. 움직임 추정에서 수행 사이클의 큰 변동은 비디오 압축의 실시간 구현에 많은 문제점이 발생한다. 따라서 제안한 HEXSLS는 다양한 영상에서 PLS보다 더욱 강인하고 안정적인 방법이라고 말 할 수 있다. 비록 HEXSLS와 PLS는 유사한 탐색 패턴을 가져도, HEXSLS는 여전히 PLS보다 적은 수행 사이클만이 요구된다. 이것은 HEXSLS는 탐색 라인의 수가 적고, 추가로 하나의 수평라인에서 적은 SAD 연산을 가지기 때문이다.

또한 HEXSLS는 HEXBS와 같은 Hexagon 모양의 패턴을 가지지만, 외부 메모리의 액세스가 적게 발생되기 때문에 HEXBS보다 다소 SAD 연산이 많이 발생하더라도 더 적은 수행 사이클을 요구한다.

위의 모든 실험 결과를 통하여, 제안한 HEXSLS는 TI TMS320C64x™ 와 같은 미디어 프로세서에서 수행 사이클과 탐색 성능에서 매우 효과적인 기법이라고 말할 수 있다.

VI. 결 론

본 논문은 미디어 프로세서에서 Hexagon 탐색 패턴의 비 정규화된 탐색 방법을 라인 탐색 기법을 적용하여 정규화된 탐색 방법을 가지는 Hexagon 모양 라인 탐색(HEXagon-Shape Line Search, HEXSLS) 기법을 제안하였다. 또한 탐색 라인에서 서브 샘플링 위치만을 탐색하고 수직으로 하나의 위치를 건너뛰어 탐색함으로서 PLS기법보다 외부 메모리의 접근을 줄이고 SAD 탐색 수를 줄였다. 제안한 HEXSLS 기법은 FSBMA와 PLS 기법과 비교하여 비슷한 MAE 성능을 유지하면서 미디어 프로세서에서 사이클 소모가 줄어들었다. HEXSLS는 FSBMA와 HEXBS보다 대략 7.93배, 2.4배 적은 수행 사이클을 보여 주었으며, 반면 MAE 성능은 HEXBS보다 우수하다. 이것은 HEXSLS는 멀티미디어 프로세서에서 실시간 비디오 코딩 응용에 매우 효과적인 기법임을 말한다.

참 고 문 헌

[1] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T.

Ishiguro, "Motion compensated interframe coding for video conferencing," in Proc. Nat. Telecommun. Conf., New Oreleans, LA, pp.G5.3.1-G5.3.5, Nov. 1981.
[2] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.4, pp.313-317, Aug. 1994.
[3] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block matching motion estimation," IEEE Trans. on Image Processing, vol. 9, pp. 287-290, Feb. 2000.
[4] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 8, pp. 369-377, Aug. 1998.
[5] G. Zhu, X. Lin, and L. -P. Chau, "Hexagon-based search pattern for fast block motion estimation," IEEE Trans. on Circuits Syst. Video Technol., vol. 12, no. 5, pp. 349-355, May. 2002.
[6] C. H. Cheung and L. M. Po, "A novel cross-diamond search pattern for fast block motion estimation," IEEE Trans. on Circuits Syst. Video Technol., vol. 12, no. 12, Dec. 2002.
[7] J. -Y. Nam, J. -S. Seo, J. -S. Kwak, M. -H. Lee, and Y. H. Ha, "New fast-search algorithm for block matching motion estimation using temporal and spatial correlation of motion vector," IEEE Trans. on Consumer Electron., vol. 46, no. 4, pp. 934-942, Nov. 2000.
[8] P. I. Hosur, "Motion adaptive search for fast motion estimation," IEEE Trans. Consumer Electron., vol. 49, no. 4, pp.1330-1340, Nov. 2003.
[9] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive motion vector field adaptive search technique (PMVFAST) - Enhancing block based motion estimation," Proc. SPIE Visual Commun. Image Process., San Jose, CA, pp.883-892, Jan. 2001.
[10] I. Kuroda and T. Nishitani, "Multimedia processors," Proc. IEEE, vol. 86, pp.1203-1221, Jun. 1998.
[11] Y. W. Huang, S. Y. Chien, B. Y. Hsieh, and L. G. Chen, "Predictive Line Search: An efficient motion estimation algorithm for MPEG-4 encoding systems on multimedia processors," IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 1, pp.111-117, Jan. 2003.
[12] Texas Instruments, Inc., TMS320C64x technical overview, Literature Number : SPRU395B, Jan. 2001.
[13] Texas Instruments, Inc., TMS320C6000 optimizing compiler user's guide, Literature Number : SPRU187K, Oct. 2002.
[14] N. Kehtarnavaz, DSP System Design: Using the TMS320C6000, Prentice Hall, Upper Saddle River, New Jersey 2001.
[15]최웅일, 전병우, "H.264 표준의 가변 움직임 불록을 위한 고속 움직임 탐색 기법", 대한전자공학회논문지 SP편 제41권 제6호, 209-220쪽, 2004년 11월.
[16] 정봉수, 전병우, "예측 움직임 벡터와 블록 정합 오류 특성을 이용한 고속 움직임 추정 알고리즘", 대한전자공학회 신호처리 소사이어티 추계학술대회, 제 26권 제2호, 145-148쪽, 2003년 11월.

저 자 소 개

정 봉 수(학생회원)
2002년 2월 성균관대학교 전기 전자 및 컴퓨터공학부 졸업(학사)
2004년 8월 성균관대학교 정보통신공학부 졸업 (석사)
2004년 9월~현재 성균관대학교 전자 컴퓨터 공학과 박사 과정
<주관심분야: Resource-Aware 영상 압축, 저복잡도 영상 압축, 멀티미디어 VLSI, 멀티미디어 통신, 에러 강인 디지털 비디오, >

전 병 우(정회원)
1985년 2월 서울대학교 전자공학과 졸업 (학사)
1987년 2월 서울대학교 전자공학과 졸업 (석사)
1992년 12월 Purdue Univ, School of Elec. 졸업 (공학박사)
1993년~1997년 8월 삼성전자 신호처리연구소 수석연구원
1997년 9월~현재 성균관대학교 정보통신공학부 부교수
<주관심분야: 멀티미디어 영상 압축, 영상인식, 신호처리>