정규논문 (Regular Paper)
방송공학회논문지 제19권 제5호, 2014년 9월 (JBE Vol. 19, No. 5, September 2014)
http://dx.doi.org/10.5909/JBE.2014.19.5.699
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

HEVC 부호화기에서 GPU 기반 정수화소 움직임 추정을 고속화하기 위한 적응적인 탐색영역 결정 방법

김 상 민[a], 이 동 규[a], 심 동 규[b], 오 승 준[a]‡

Adaptive Search Range Decision for Accelerating GPU-based Integer-pel Motion Estimation in HEVC Encoders

Sangmin Kim[a], Dongkyu Lee[a], Dong-Gyu Sim[b], and Seoung-Jun Oh[a]‡

요 약

본 논문은 High Efficiency Video Coding (HEVC) GPU 기반 정수화소(integer-pel) 움직임 추정(Motion Estimation)을 고속화하기 위한 적응적인 탐색영역 결정 방법을 제안한다. 적응적인 탐색영역은 Motion Vector Difference (MVD)를 이용하여 결정한다. 먼저, 입력 영상의 MVD를 분석하여 입력 영상을 두 모델로 분류한다. 이후 분류된 각 모델의 MVD 특성에 따라 적응적인 탐색영역을 결정한다. 제안하는 알고리즘을 GPU 기반 정수화소 움직임 추정에 적용하기 위해 움직임 추정의 시작점은 이전 프레임의 Motion Vector (MV)로 결정한다. 위 과정은 CPU에서 이뤄지며, CPU는 움직임 추정의 시작점과 적응적인 탐색영역을 GPU에 전송한다. 이후 GPU는 정수화소 움직임 추정을 병렬로 수행한다. 제안하는 알고리즘은 참조 모델 대비 1.1%의 BD-rate 상승과 전체 부호화 시간의 37.9% 감소 및 951.2배 빠른 정수화소 움직임 추정 수행 시간을 얻는다. 또한, 적응적인 탐색영역이 적용되지 않은 단순 병렬화 알고리즘 대비 57.5%의 정수화소 움직임 추정 시간 감소와 0.6% BD-rate 상승을 얻는다.

Abstract

In this paper, we propose a new Adaptive Search Range (ASR) decision algorithm for accelerating GPU-based Integer-pel Motion Estimation (IME) of High Efficiency Video Coding (HEVC). For deciding the ASR, we classify a frame into two models using Motion Vector Differences (MVDs) then adaptively decide the search ranges of each model. In order to apply the proposed algorithm to the GPU-based ME process, starting points of the ME are decided using only temporal Motion Vectors (MVs). The CPU decides the ASR as well as the starting points and transfers them to the GPU. Then, the GPU performs the integer-pel ME. The proposed algorithm reduces the total encoding time by 37.9% with BD-rate increase of 1.1% and yields 951.2 times faster ME against the CPU-based anchor. In addition, the proposed algorithm achieves the time reduction of 57.5% in the ME running time with the negligible coding loss of 0.6%, compared with the simple GPU-based ME without ASR decision.

Keyword : HEVC, Fast ME, Adative Search Range, MVD, GPGPU

Ⅰ. 서 론

높은 화질의 영상 서비스에 대한 수요가 늘어남에 따라 국제 표준화 그룹인 Moving Picture Experts Group (MPEG)과 Video Coding Experts Group (VCEG)은 공동으로 차세대 동영상 압축 표준 High Efficiency Video Coding (HEVC)를 개발하여 최근 표준화 작업을 완료하였다. HEVC는 이전 표준 H.264/AVC 대비 객관적 화질측면에서 약 40%정도 높은 부호화 효율을 보이고 있다. 하지만, 이러한 높은 압축률을 얻기 위하여 적용된 다양한 부호화 크기와 새로운 기술들로 그 복잡도 역시 증가하였다. 특히, 움직임 추정(Motion Estimation : ME)의 복잡도는 전체 부호화기 복잡도의 70%이상을 차지하고 있으며, 연산 시간 역시 큰 비중을 차지한다[1].

이러한 움직임 추정을 고속화하기 위해서 다양한 연구가 진행되어 왔다[2]-[7]. 이 연구들은 크게 Central Processing Unit (CPU) 상에서의 고속화 방법[2]-[4]과 Graphics Processing Unit (GPU)상에서의 병렬 고속화 방법으로 나눌 수 있다[5]-[7]. CPU 상에서 진행된 연구 three-step search[2]와 four-step search[3] 그리고 diamond search[4]는 탐색영역 내에 존재하는 탐색 점을 선택적으로 결정하여 탐색함으로써 움직임 추정의 고속화를 이뤄낸다. GPU를 이용한 병렬화 방법[5]-[7]은 General-Purpose computing on GPU (GPGPU)를 이용하여 복잡도가 높은 움직임 추정을 전역탐색 기반에서 단순 병렬화 함으로써 고속화를 이뤄냈다. 이는 [2]-[4]와 같이 분기문이 많이 필요한 알고리즘을 병렬화 할 경우 Single Instruction Multiple Threads (SIMT) 구조에 적합하지 않아 병렬화 이득을 최대화 할 수 없기 때문이다. 하지만, SIMT 구조에 적합한 전역탐색 기반의 움직임 추정 알고리즘은 GPGPU 구현을 통해 더욱 빠른 움직임 추정 고속화를 얻을 수 있다.

본 논문은 SIMT 구조에 적합한 전역탐색 기반의 적응적인 탐색영역 결정 방법을 제안하고, 결정된 탐색영역을 GPU 기반 정수화소 (integer-pel) 움직임 추정에 적용함으로써 더욱 빠른 정수화소 움직임 추정 고속화를 이룬다. 적응적인 탐색영역을 결정하기 위하여 제안하는 알고리즘은 전역(global) 움직임 특성과 지역(local) 움직임 특성 모두를 반영한다. 전역 움직임 특성을 반영하기 위하여 입력영상의 Motion Vector Difference (MVD)를 프레임 단위로 분석하여 움직임 정도를 파악하고, 입력 프레임을 두 가지 모델로 분류한다. 이후 각 모델의 지역 움직임 특성을 파악하기 위해 Coding Tree Unit (CTU) 단위로 MVD를 파악하여 탐색영역을 결정한다. 움직임 추정의 병렬화는 [7]에 제안된 알고리즘을 사용한다. [7]은 Nvidia사의 GPGPU 기술 CUDA(Compute Unified Device Architecture)를 이용하여 전역탐색 기반의 정수화소 움직임 추정 방법을 프레임 단위로 구현한다. 움직임 추정을 병렬 처리할 경우 각 Prediction Unit (PU)의 움직임 추정 시작점을 찾는 과정에서 (Advanced Motion Vector Prediction : AMVP) 인접 PU 간의 종속성 문제가 발생한다. 이를 해결하기 위하여 본 논문은 이전 프레임의 Motion Vector (MV)를 이용하여 움직임 추정의 시작점으로 삼는다[8][9]. 적응적인 탐색영역과 움직임 추정의 시작점은 CPU에서 결정되며, 이 값들은 GPU에 전송된다. 이후 GPU는 정수화소 움직임 추정을 병렬로 처리한다. 본 논문은 low complexity 환경에서 부호화 성능을 최대한 유지하면서 움직임 추정을 고속화하는 것을 목표로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 HEVC의 움직임 추정 방법 및 CUDA에 대해 소개한다. 3장에서는 기존의 움직임 추정 병렬화 방법을 살펴보고, 4장에서 제안하는 탐색영역 결정 방법을 설명한다. 5장에서는 제안된 알고리즘의 실험결과를 분석하고, 6장에서 결론을 맺는다.

a) 광운대학교 전자공학과 (Dept. of Electronic Engineering, Kwangwoon university)
b) 광운대학교 컴퓨터공학과 (Dept. of Computer Engineering, Kwangwoon university)
‡ Corresponding Author : 오승준(Seoung-Jun Oh)
E-mail: sjoh@kw.ac.kr
Tel: +82-2-940-5102
※ 이 논문은 2014년도 광운대학교 교내학술연구비 및 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발 사업의 일환으로 수행하였음. [14-000-11-002, 방송용 영상 인식 기반 객체 중심 지식융합 미디어 서비스 플랫폼 개발]
· Manuscript received July 10, 2014 Revised September 4, 2014 Accepted September 4, 2014

II. HEVC 움직임 추정 방법과 CUDA

1. HEVC 움직임 추정 방법

H.264/AVC에서 16x16 크기의 매크로 블록(Macroblock : MB)을 기본 단위로 부호화하는데 비해 HEVC는 64x64~16x16 크기의 코딩 트리 유닛(Coding Tree Unit : CTU)을 정의하고, 64x64~8x8 크기의 쿼드트리(quad tree) 형태로 분할된다. 이를 Coding Unit (CU)라고 한다. 움직임 추정을 위하여 각 CU는 PU로 분할되며, 움직임 추정을 통해 참조 프레임에서 현재 PU와 가장 유사한 참조 블록을 찾는다. 움직임 추정을 위해 사용되는 식은 (1)과 같다.

$$J(\Phi,\lambda) = SAD(\Phi) + \lambda B(\Phi) \quad (1)$$

J는 율-왜곡(Rate-Distortion : RD)을 나타내며, *SAD*는 Sum of Absolute Differences (SAD)를 의미한다. *B*는 MVP와 추정된 MV간의 MVD를 부호화하기 위한 비트 량이다. λ는 라그랑지안 승수(lagrangian multiplier)로 에러의 양과 비트량 간의 단위를 일치시키기 위하여 사용된다. 움직임 추정은 *J*값을 최소로 하는 참조 블록으로부터 MV를 얻는다. 움직임 추정의 탐색 방법은 탐색영역 내에 있는 모든 탐색점을 순차적으로 탐색하는 전역탐색 방법과 제한된 탐색점에서 탐색하는 고속탐색 방법이 존재한다. 부호화기는 탐색을 시작하기에 앞서 부호화 성능을 높이기 위해 움직임 추정의 시작점을 결정한다. H.264/AVC는 현재 MB 주변 블록의 MV를 이용하여 MVP를 결정하고, 움직임 추정의 시작점으로 사용한다. HEVC는 AMVP를 이용하여 현재 PU의 공간적 후보들의 MV뿐만 아니라 시간적 후보들의 MV를 고려하여 MVP를 결정한다. 그림 1은 MVP를 시작점으로 MV를 찾는 움직임 추정 과정을 보여준다.

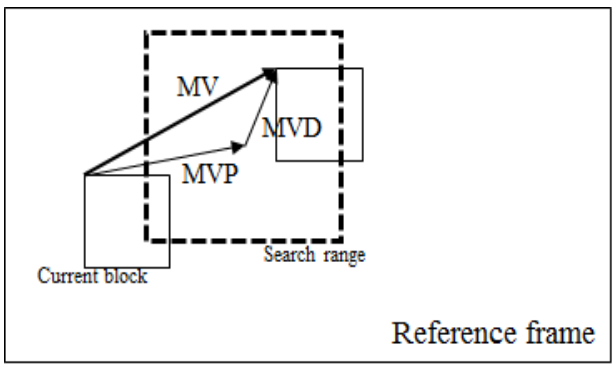


그림 1. 움직임 추정 처리 과정
Fig. 1. ME process

2. CUDA 프로그래밍 모델

CUDA 프로그램은 CPU에서 순차적으로 처리되는 host 프로그램과 GPU에서 다중 스레드(thread)에 의해 처리되는 device 프로그램으로 구성된다. 이 예로 커널(kernel)이라 불리는 함수는 CPU에서 호출되어 SIMT 방식으로 GPU에서 실행된다. CUDA는 수많은 스레드들을 다루기 위해 계층적인 스레드 구조를 가진다. 하나의 스레드 블록(thread block)은 동시에 수행되는 스레드들의 집합으로 정의되고, 그 상위 개념으로 독립적인 스레드 블록들의 집합을 그리드(grid)라고 정의한다. 스레드 볼록 내에 모든 스레드는 GPU의 온-칩(on-chip)메모리인 공유 메모리(shared memory)를 통해서 데이터를 공유할 수 있다.

3. CUDA 하드웨어 구조

Kepler 구조의 GPU는 다수의 Streaming Multiprocessor (SMX)로 구성되어 있으며, 하나의 SMX는 다수의 Streaming Processor (SP)들과 공유 메모리, 레지스터, 글로벌 메모리, 워프 스케줄러(warp scheduler)등으로 이뤄져 있다. 하나의 SMX에서 처리되는 32개의 스레드를 워프(warp)라 하고, 워프 스케줄러는 워프 그룹 단위로 스레드를 스케줄링 한다. 레지스터는 가장 빠른 메모리로 각 스레드는 자신의 레지스터를 가지며 다른 스레드의 레지스터에 접근할 수 없다. 공유 메모리는 온-칩 메모리로 레지스터만큼 빠른 메모리 이며, 스레드 블록 내의 스레드간 통신이 가능하다.

IV. 움직임 추정의 병렬화

본 장은 기존의 GPU를 이용한 움직임 추정 병렬화에 대

해 간단히 살펴보고, 본 논문에서 참고한 [7]의 알고리즘을 소개한다. [5]는 전역탐색을 기반으로 정수화소 및 부화소 (fractional-pel) 움직임 추정을 병렬화 하였지만 GPU의 온-칩 메모리를 사용하지 않아 데이터 전송 시 높은 메모리 접근 지연시간(latency)을 갖는다. [6]는 메모리 사용을 최적화하여 전역탐색 기반의 정수화소 움직임 추정을 병렬화 한다. [7]은 정수화소 움직임 추정을 위하여 계층적인(hierarchical) SAD 연산방법을 사용하고, 메모리 사용을 최적화 한다. 또한 Concurrent Parallel Reduction (CPR)을 제안하여 최소 SAD를 구하는 과정에서 발생하는 기존 Parallel Reduction (PR)의 활성 스레드 저하 문제와 동기화(synchronization) 문제를 해결한다. 다음은 [7]이 사용한 계층적인 SAD 연산과 CPR에 대한 설명이다.

[7]은 프레임 단위로 정수화소 움직임 추정을 진행하며, 하나의 스레드 블록이 하나의 MB에 할당되어 계층적으로 SAD 값을 구한다. 이를 위해 MB내 4x4 블록들의 SAD가 먼저 계산된다. 4x4 블록 당 계산되는 SAD들을 SAD group이라 정의하고, MB내 총 16개의 SAD group들은 CPR에 의해 최소 SAD값을 얻게 된다. 기존 PR은 반복적인 연산중에 활성 스레드들의 개수가 반으로 줄어들어 스레드 활용(utilization)이 저하되는 문제를 갖고 있다. 또한 이에 따르는 data hazard 문제는 스레드간의 동기화 과정을 필요로 하기 때문에 병렬화 성능을 저하시킨다. 하지만 CPR은 각 SAD group에 스레드를 할당하여 각 SAD group을 한 워프씩 병렬처리 한다. 즉, 연산 과정에서 스레드간 동기화 없이 동시에 처리한다. 결국, CPR에 의해서 16개의 4x4 블록의 IMV(integer-pel MV)들을 얻게 된다. 이후 8x4, 4x8 블록의 SAD는 4x4 블록들의 SAD를 더해서 구하고, 8x4, 4x8 블록에 해당하는 16개의 새로운 SAD group이 생성된다. 이 SAD group 역시 CPR로 처리되며, 16개의 IMV가 얻어진다. 같은 방법으로 다른 블록의 IMV들을 얻을 수 있다. 본 논문에서는 [7]의 알고리즘을 HEVC 정수화소 움직임 추정에 적용하여 하나의 32x32 CTU에 하나의 스레드 블록을 할당한다. 이후 8x8 PU를 시작으로 계층적인 SAD 연산을 시작하고, CPR을 통해 각 PU의 IMV를 구한다.

전술한 바와 같이 움직임 추정은 탐색 전 움직임 추정의 시작점을 결정한다. 하지만, 움직임 추정을 프레임 단위로 병렬 처리하면 AMVP 과정에서 인접 PU간의 종속성 문제가 발생하여 공간적 후보를 고려할 수 없다. 이는 현재 PU가 스레드에 의해 처리될 때 주변 PU 역시 동시에 처리되고 있으므로 현재 PU는 주변 PU의 MV를 이용할 수 없기 때문이다. H.264/AVC 역시 같은 이유로 탐색영역의 시작점을 찾는데 문제를 일으킨다. [7]는 탐색영역의 시작점을 사용하지 않았다. 이것은 부호화 성능 저하의 원인이 될 수 있다. 본 논문은 시간적 후보를 고려하여 탐색영역의 시작점을 결정함으로써 성능 저하를 최소화한다[8][9]. 이를 Search Starting Point (SSP)라 명명한다. SSP는 현재 CTU와 대응되는 이전 프레임 CTU의 모든 MV 평균으로 정의되며, 현재 CTU 내의 모든 PU는 하나의 SSP를 공유한다.

Ⅴ. 제안하는 방법

기존의 GPU 기반 고속 움직임 추정 알고리즘[5]-[7]은 움직임 추정의 단순 병렬화를 통하여 고속화를 이루었다. 이는 [2]-[4]와 같이 많은 분기문이 필요한 알고리즘을 병렬화 할 경우 SIMT 구조에 적합하지 않아 병렬화 성능을 얻기 어렵기 때문이다. 하지만, SIMT 구조에 적합한 전역탐색 기반의 움직임 추정 알고리즘은 GPGPU 구현을 통해 더욱 빠른 움직임 추정 고속화를 얻을 수 있다. 본 논문은 SIMT 구조에 적합한 전역탐색 기반에서 MVD를 이용한 적응적인 탐색영역 결정 방법을 제안하고, 결정된 탐색영역을 GPU 기반 정수화소 움직임 추정에 적용함으로써 더욱 빠른 정수화소 움직임 추정 고속화를 이룬다. 적응적인 탐색영역을 결정하기 위하여 제안하는 알고리즘은 전역 움직임 특성과 지역 움직임 특성 모두를 반영한다. 전역 움직임 특성을 반영하기 위하여 입력영상의 MVD를 프레임 단위로 분석하여 움직임 정도를 파악하고, 입력 프레임을 두 가지 모델로 분류한다. 이후 각 모델의 지역 움직임 특성을 파악하기 위해 CTU 단위로 MVD를 파악하고 탐색영역을 결정한다. 그림 2는 제안하는 알고리즘의 흐름도이다. 실선은 알고리즘의 흐름을 나타내고, 점선은 데이터의 흐름을 나타낸다. CPU에서 SSP와 적응적인 탐색영역을 정하고 이 값을 GPU에 전송한다. GPU는 전송받은 정보를 이용해서

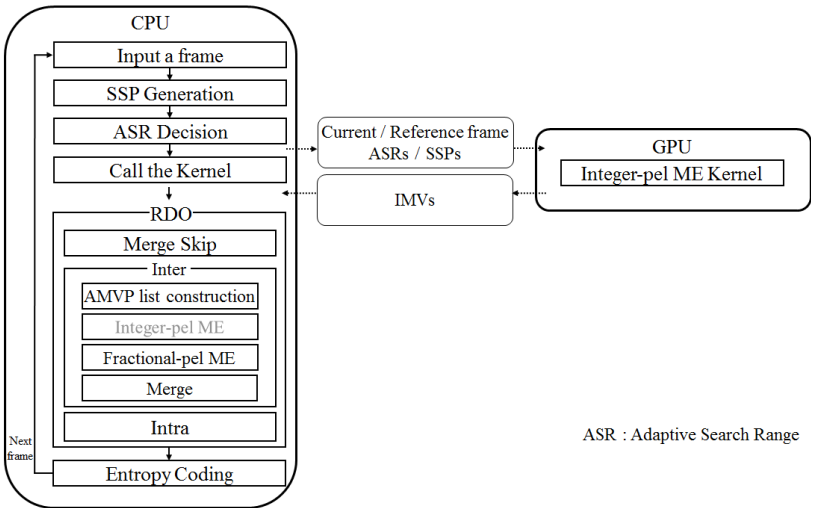


그림 2. 제안하는 알고리즘의 전체 흐름도
Fig. 2. Overflow of the proposed algorithm

현재 프레임의 정수화소 움직임 추정을 진행하고, CPU는 이 과정이 끝날 때까지 대기한다. 따라서 제안하는 알고리즘은 Rate-Distrotion Optimization (RDO) 과정 전에 입력 프레임 내의 모든 PU의 IMV를 얻을 수 있다. 제안하는 알고리즘은 RDO 과정에서 정수화소 움직임 추정을 제외하고 HEVC test model (HM)의 방법을 그대로 이용한다. RDO 과정 전에 정수화소 움직임 추정을 거쳤기 때문에 RDO 과정에서 정수화소 움직임 추정을 생략하고 미리 얻은 IMV를 사용하여 부화소 움직임 추정을 진행한다. 나머지 과정은 HM의 방법을 그대로 이용한다.

1. 움직임 특성에 따른 영상 분류

움직임 추정은 MV를 얻기 위해 MVP를 시작점으로 삼고 진행되어 MVD를 찾는다. 따라서 움직임 추정 관점에서 영상의 움직임 특성을 파악하는 기준으로 MVD가 고려된다. 제안하는 알고리즘은 입력 프레임의 탐색영역을 결정하기 위하여 전역 움직임 특성과 지역 움직임 특성 모두를 반영한다. 먼저, 전역 움직임 특성을 살펴보기 위하여 입력 영상을 프레임 단위로 살펴본다. 이때 프레임내 우세한(dominant) 움직임의 특성을 파악하고 연산 량을 최소화하기 위하여 체스보드 거리 측정법(chess board distance : L^{∞})을 이용하고, 이를 MVD_{chess}라 정의한다. 본 논문은 프레임 내의 움직임이 많고 적음을 파악하기 위해 한 프레임당 MVD_{chess}값이 0인 비율을 이용한다. 이 비율은 D_{motion} (Degree of Motion)이라 정의되고, 프레임의 전역 움직임 특성을 대표한다. D_{motion}값이 0에 가까울수록 입력 프레임은 움직임이 적은 프레임, 100에 가까울수록 움직임이 많은 프레임으로 판단될 수 있다. 표 2 실험환경에서 D_{motion}을 조사하여 그림 3의 분포를 얻었다. 그림 3에 따르면 Class B 영상은 크게 두 모델로 분류 될 수 있음을

$$MVD_{Chess} = \begin{cases} MVD_x, & |MVD_x| \geq |MVD_y| \\ MVD_y, & |MVD_x| < |MVD_y| \end{cases} \quad (2)$$

표 2. D_{motion} 조사를 위한 실험 환경
Table 2. Experimental environment for D_{motion} research

reference model	HM 10.0
configuration	low delay P
QP	22,27,32,37
sequence	Class B
number of frame	100
CTU size	32x32
reference frame	1
search type	full search
search range	±16

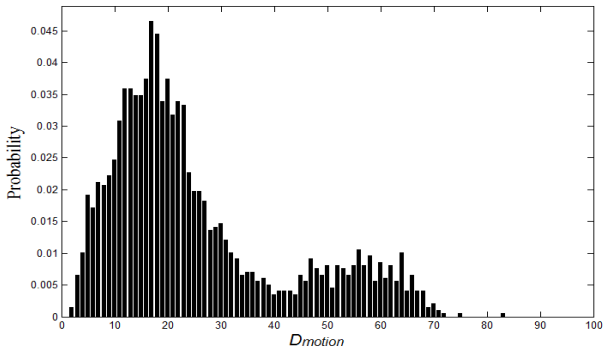


그림 3. Class B 영상의 D_{motion} 분포
Fig. 3. D_{motion} Distribution of Class B sequences

알 수 있다. 그림 4는 D_{motion}의 분포를 2개의 가우시안 분포(Gaussian distribution)로 모델링한 결과를 나타낸다. 두 분포를 분류하기 위하여 오차 최소화(error minimization) 방법을 사용하여 문턱 값 37%를 얻었다. 이로써 입력영상의 각 프레임 당 D_{motion}값을 문턱 값 37%와 비교하여 프레임의 전역 움직임 특성을 파악 할 수 있다.

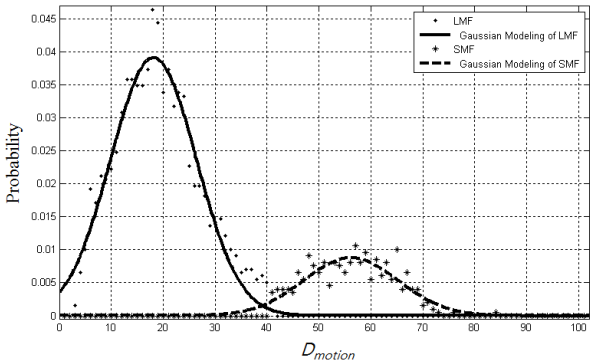


그림 4. Class B 영상 D_{motion} 의 가우시안 모델링
Fig. 4. Gaussian modeling of D_{motion} distribution of Class B sequences

본 논문은 현재 프레임과 이전 프레임 간의 전역 움직임 특성의 연관성(correlation)이 매우 높다고 가정한다. 따라서 현재 프레임의 전역 움직임 특성은 이전 프레임의 D_{motion}값을 통해서 얻어진다. 만약, 이전 프레임의 D_{motion}값이 37%보다 작을 경우 현재 프레임을 움직임이 많은 프레임으로 판단하고, LMF(Large Motion Frame)라 정의한다. 반대의 경우 SMF(Small Motion Frame)로 판단한다.

2. 적응적인 탐색영역 결정 방법

전술한 바와 같이 제안하는 알고리즘은 입력 프레임의 전역 움직임 특성뿐만 아니라 지역 움직임 특성을 고려하여 적응적인 탐색영역을 결정한다. 따라서 LMF와 SMF로 판정된 각 프레임들의 지역 움직임 특성을 살펴본다. 지역 움직임 특성은 각 프레임의 CTU 단위로 조사된다. 본 논문은 CTU 단위의 지역 움직임 특성을 MVD_{CTU}로 정의한다. MVD_{CTU}는 한 CTU 내에 모든 MVD_{chess}의 평균값으로 정의된다. 그림 5는 LMF와 SMF의 MVD_{CTU}분포를 각각 나타낸다. 이 두 분포는 라플라시안 분포(Laplacian distribution)를 따르고 있다. 본 논문은 부호화 성능을 최대한 유지하면서 움직임 추정을 고속화하기 위해 문턱 값을 정하여 적응적 탐색영역을 적용한다. 예를 들어, SMF의 MVD_{CTU}가 문턱 값 TH_{SMF} 내에 존재한다면 탐색영역(Search Range : SR)이 적용되고, LMF의 경우 TH_{LMF}을 문턱 값으로 하고 탐색영역 SR을 적용한다. 이 문턱 값과 적응적인 탐색영역의 크기

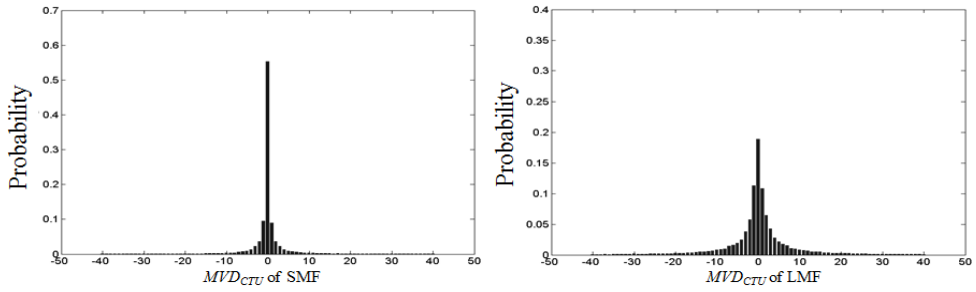


그림 5. SMF와 LMF의 MVD_{CTU} 분포
Fig. 5. Distributions of MVD_{CTU} in SMF and LMF

는 실험적으로 정한다. 표 3은 MVD_{CTU} 분포에서 0을 기준으로 75%~95% 에 포함되는 MVD_{CTU} 값을 1/4화소(quarter-pel) 기준으로 나타낸 것이다. 예를 들어 LMF의 경우 $-6 \le MVD_{CTU} \le 6$에 해당하는 MVD_{CTU}가 전체 분포 중 75%를 차지함을 의미한다.

표 3. 문턱 값 구간과 MVD_{CTU}
Table 3. Threshold section and MVD_{CTU}

Section / Mode	75%	85%	95%
LMF	-6~6	-12~12	-32~32
SMF	-1~1	-3~3	-11~11

본 논문에서는 표 3에서 제시된 3구간 중에서 실험을 통하여 TH_{LMF}와 TH_{SMF}를 결정한다. 실험환경은 표 4와 같다. 정수화소 움직임 추정의 시간 측정을 위해 TR(Time Reduction)을 이용한다. TR는 식 (3)에 정의된다.

표 4. 문턱 값과 적응적 탐색영역 결정을 위한 실험 환경
Table 4. Test environment for threshold and adaptive search range decision

reference model	HM 10.0
configuration	low delay P
QP	22,27,32,37
sequence	Class B
number of frame	100
CTU size	32x32
reference frame	1
search type	full search
search range	±8

$$TR = \frac{T_{anchor} - T_{proposed}}{T_{anchor}} \times 100(\%) \quad (3)$$

T_{anchor}는 참조 소프트웨어에서의 연산 시간이며 $T_{proposed}$는 제안된 알고리즘의 연산 시간이다. 그림 6,7은 적응적 탐색영역과 문턱 값 결정 실험결과를 보여준다. 그림 6은 TH_{LMF}와 TH_{SMF}를 75%부터 95%까지 변경 시켰을 때의 움직임 추정 TR을 나타낸다. 이에 따르면, TH_{LMF}와 TH_{SMF} 모두 95%로 설정할 경우 가장 빠른 움직임 추정을 이뤄낼 수 있다. 하지만 본 논문은 움직임 추정의 고속화뿐만 아니라 부호화 성능 저하를 최소화하기 위해 각 문턱 값의 부호화 성능도 함께 고려해야 한다. 그림 7은 TH_{LMF} 85% TH_{SMF} 95% 일 때 이후의 문턱 값에서 Class B 영상 중 Basketball Drive 영상의 Bjontegaard-delta bitrate (BD-rate)[10]가 0.6%에서 1.1%로 두 배 가까이 상승한 것을 확인할 수 있다. 따라서 본 논문은 TH_{LMF}와 TH_{SMF}의 값을 각각 85%와 95%로 결정한다. 만약, 입력영상의 임의의 프레임이 LMF로 분류되고, 현재 CTU의 MVD_{CTU}가 TH_{LMF}에 포함된다면 적응적인 탐색영역을 적용하고, 포함되지 않는다면 기본 탐색영역을 적용하여 정수화소 움직임 추정을 진행한다.

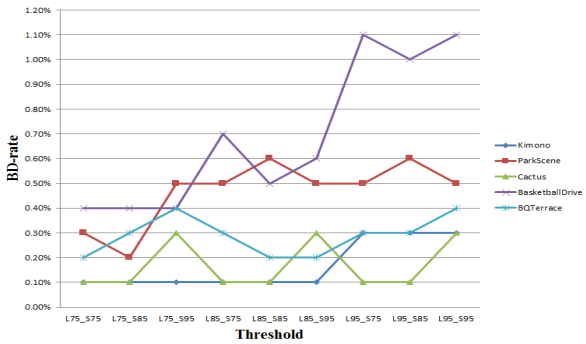


그림 6. 문턱 값 결정을 위한 부호화 성능 비교
Fig. 6. Comparison of coding performance for threshold decision

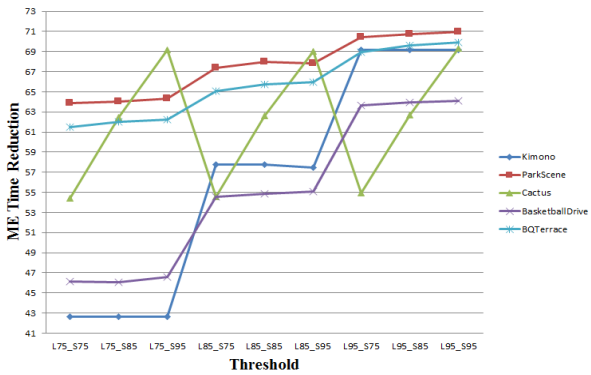


그림 7. 문턱 값 결정을 위한 움직임 추정 TR 비교
Fig. 7. Comparison of ME TR for threshold decision

TH_{LMF}와 TH_{SMF}가 결정되었으므로 적응적인 탐색영역을 결정해야 한다. 본 논문은 이를 위하여 3가지의 탐색영역 4x4, 8x8, 12x12 중에서 실험을 통하여 TH_{LMF}와 TH_{SMF}에 적용될 탐색영역을 결정한다. 그림 8,9는 각 탐색영역의 정

수화소 움직임 추정 *TR*과 BD-rate 결과를 나타낸다. 이 결과를 살펴보면 부호화 성능 하락을 최소화하면서 정수화소

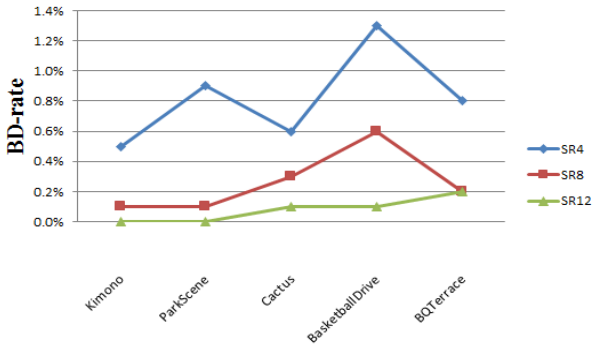


그림 8. SR 결정을 위한 부호화 성능 비교
Fig. 8. Comparison of coding performance for SR decision

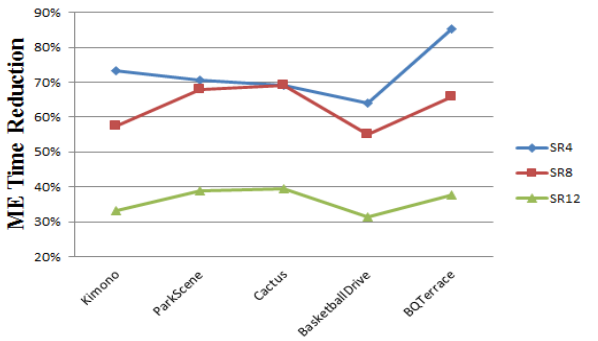


그림 9. SR 결정을 위한 움직임 추정 *TR* 비교
Fig. 9. Comparison of ME *TR* for SR decision

```
1.    Calculate
1.1   the D_motion of the current frame
1.2   the MVD_CTUs of all CTUs of the current frame
2.    if D_motion < 37 then
2.1     for number of CTU iterations do:
2.1.1     if |MVD_CTU| <= TH_SMF then
            set ±8 for search range of current CTU
2.1.2     else
            set ±16 for search range of current CTU
3.    else
3.1     for number of CTU iterations do:
3.1.1     if |MVD_CTU| <= TH_LMF then
            set ±8 for search range of current CTU
3.1.2     else
            set ±16 for search range of current CTU
```

그림 10. 적응적인 탐색영역 결정을 위한 수도코드
Fig. 10. Pseudo code for adaptive search range decision

움직임 추정의 고속화를 이뤄내는 탐색영역은 8x8이라는 것을 알 수 있다. 따라서 본 논문은 문턱 값 TH_{LMF}와 TH_{SMF}를 만족하는 CTU에 대해 적응적인 탐색영역 8x8을 적용한다. 그림 10은 제안하는 알고리즘의 수도코드(pseudo code)를 나타낸다.

Ⅴ. 실험결과

제안된 알고리즘의 성능을 평가하기 위하여 HEVC 참조 소프트웨어 HM 10.0에 CUDA 기반 GPU 병렬화 기법을 적용하였다. 실험을 위해 사용된 CPU는 Intel i7 3.07GHz, 16GB DRAM이고, GPU는 GeForce GTX 460, 1GB DRAM 이다. 세부적인 실험환경은 표 5에 정리되어 있다. 부호화 속도의 측정은 *TR*을 사용하였다. 또한 부호화 성능을 위하여 BD-rate와 RD 곡선을 보였다. 본 논문은 제안된 알고리즘의 일반적인 성능 평가를 위해 학습 집합(training set)인 Class B 영상뿐만 아니라 검증 집합(test set)인 Class A,C 영상에 적용하였다.

표 5. 제안된 알고리즘 부호화 성능 평가를 위한 실험 환경
Table 5. Test environment for coding performance evaluation of the proposed algorithm

configuration	low delay P
QP	22,27,32,37
sequence	Class A,B,C
number of frame	100
CTU size	32x32
reference frame	1
search type	full search
search range	±16
prediction mode	no 4x8 and 8x4 inter predictions
AMP	off

그림 11은 Class A와 C 영상의 D_{motion}분포를 나타낸다. Class A 영상의 경우 대부분의 프레임들이 37% 보다 작은 D_{motion} 값을 갖고 있음을 보인다. 따라서 본 논문에서 제시한 문턱 값(D_{motion} 37%)에 의해 대부분의 프레임들은 LMF로 판정되고, 그에 맞는 탐색영역이 결정된다. 한편, Class C 영상의 D_{motion}을 두 개의 가우시안 분포로 모델링 한 후

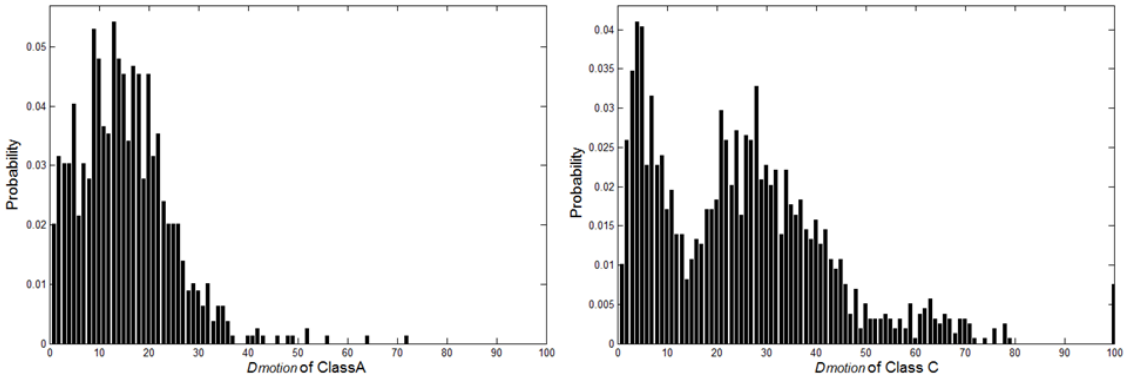


그림 11. Class A와 C 영상의 D_{motion} 분포들
Fig. 11. D_{motion} Distributions of Class A and Class c sequences

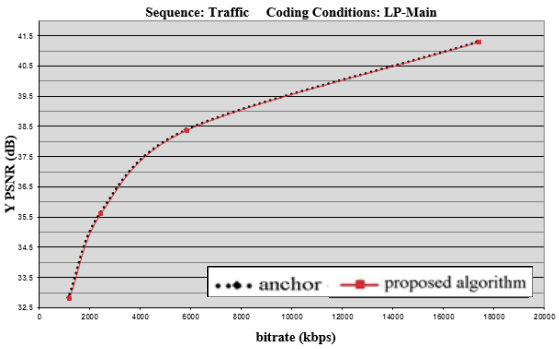
오류 최소화 방법을 이용하면 문턱 값 D_{motion} 16%을 얻게 된다. 이는 본 논문에서 제안한 문턱 값 37%를 적용하면 정수화소 움직임 추정 속도 측면에서 손해가 있음을 의미한다. 하지만 문턱 값 37%는 16%에 비해 보수적인 값임으로 화질 측면에서는 이득이 있다.

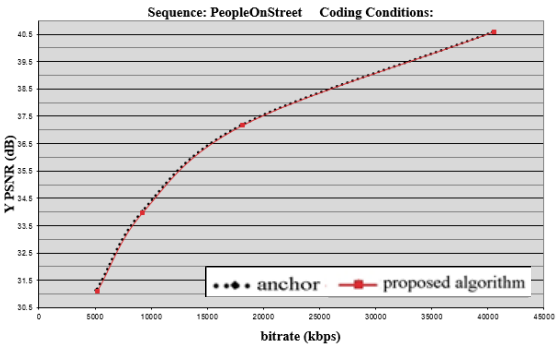
표 6는 제안된 알고리즘과 참조 모델과의 성능 비교이다. 참조 모델은 표 5의 세부사항을 따른다. 제안된 알고리즘은 참조 모델 대비 학습 집합(Class B) 뿐만 아니라 검증 집합(Class A,C)에서도 Y BD-rate 1% 내외의 부호화 성능을 내고 있으며, 평균 37.9%의 부호화 *TR* 성능을 가진다. 부호화 성능 감소의 원인은 SSP의 사용과 적응적인 탐색영역의 적용이다. SSP의 사용으로 인한 부호화 손실은 평균 Y BD-rate 0.5%이다. 하지만 시작점을 사용하지 않고 움직임 추정 알고리즘을 진행하면 SSP로 인한 평균 손실 0.5% 보다 0.6% 높은 평균 Y BD-rate

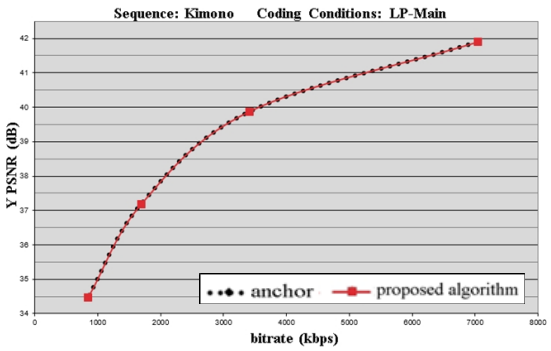
1.1% 상승을 얻는다. 또한, 그림 12은 실험에 사용된 모든 영상에 대해 제안된 알고리즘과 참조 모델 간의 RD 곡선이 매우 유사함을 나타낸다. 따라서 제안하는 알고리즘은 학습 집합뿐만 아니라 검증 집합에서도 그 성능을 유지한다.

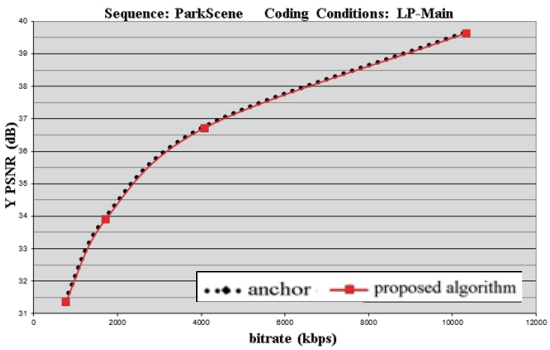
표 6. 제안된 알고리즘의 부호화 성능
Table 6. Coding performance of the proposed algorithm

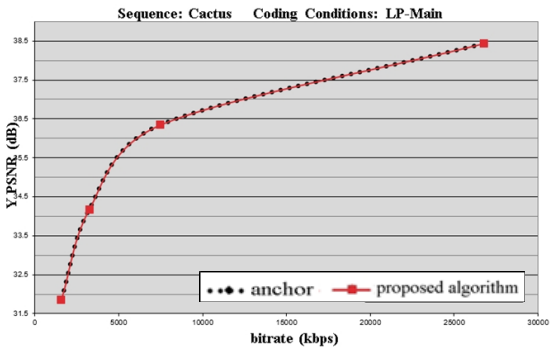
Sequence		BD-rate (%)			Encoding Time (h)		Encoding *TR* (%)
		Y	U	V	Anchor	Algorithm	
Class A	Traffic	1.0	0.9	0.7	1.90	1.06	44.2
	PeopleOnStreet	0.7	2.1	1.8	2.39	1.55	34.9
	Class A average	0.9	1.5	1.3	2.14	1.31	39.5
Class B	Kimono	0.5	0.5	0.7	1.10	0.69	37.7
	ParkScene	1.7	1.3	1.7	1.04	0.62	40.2
	Cactus	0.7	0.9	0.9	1.06	0.65	39.1
	BasketballDrive	2.3	3.6	2.2	1.11	0.71	36.4
	BQTerrace	0.4	0.5	-0.4	1.14	0.73	36.5
	Class B average	1.1	1.4	1.0	1.09	0.68	38.0
Class C	BasketballDrill	2.9	3.3	4.5	0.22	0.14	36.8
	BQMall	0.9	1.1	0.9	0.22	0.13	37.7
	PartyScene	0.1	0.5	0.4	0.27	0.13	30.0
	RaceHorses	1.2	1.2	1.7	0.27	0.19	29.0
	Class C average	1.3	1.5	1.9	0.24	0.16	33.4
	Total average	**1.1**	**1.5**	**1.4**	**0.97**	**0.61**	**37.9**

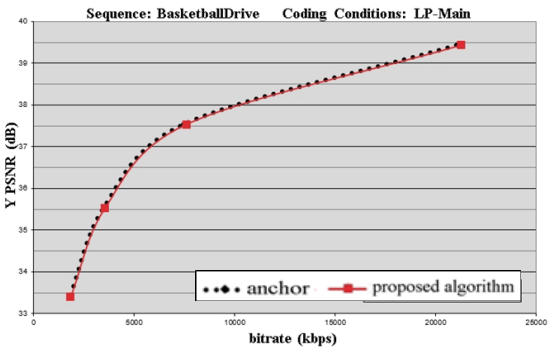

Sequence: Traffic Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm

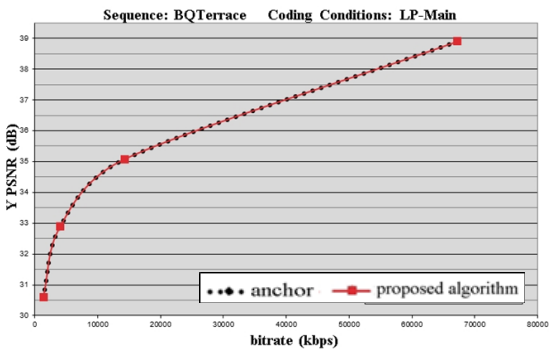

Sequence: PeopleOnStreet Coding Conditions:
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm

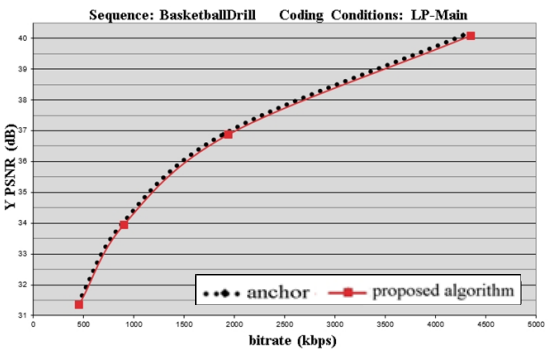

Sequence: Kimono Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm


Sequence: ParkScene Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm


Sequence: Cactus Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm


Sequence: BasketballDrive Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm


Sequence: BQTerrace Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm


Sequence: BasketballDrill Coding Conditions: LP-Main
Y PSNR (dB)
bitrate (kbps)
anchor
proposed algorithm

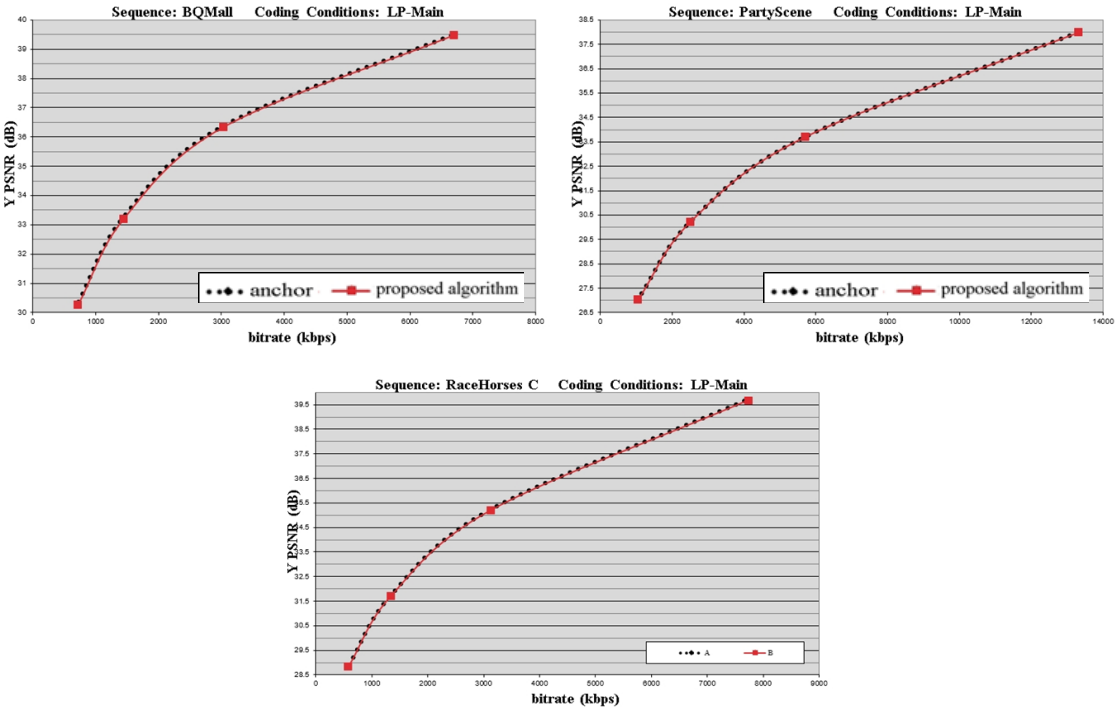


그림 12. 참조 모델과 제안된 알고리즘의 RD 곡선 비교
Fig. 12. RD curve comparisons between anchor and the proposed algorithm

표 7은 제안된 알고리즘의 정수화소 움직임 추정 연산 시간을 참조 모델과 비교한 결과이다. 제안된 알고리즘은 참조 모델 대비 951.2배 빠른 정수화소 움직임 추정 연산 속도를 가진다.

표 7. 제안된 알고리즘의 움직임 추정 수행 시간
Table 7. ME execution time of the proposed algorithm

Sequence		ME Time (s)		Speed-up
		Anchor	Algorithm	
Class A	Traffic	3020.19	2.86	1054.5x
	PeopleOnStreet	3020.68	2.98	1013.7x
	Class A average	3020.43	2.92	1034.1x
Class B	Kimono	1512.00	1.79	844.6x
	ParkScene	1512.02	1.56	972.0x
	Cactus	1512.79	1.51	1001.0x
	BasketballDrive	1512.39	1.83	828.6x
	BQTerrace	1511.60	1.62	930.7x
	Class B average	1512.16	1.66	915.4x
Class C	BasketballDrill	293.85	0.30	973.1x
	BQMall	294.26	0.30	982.0x
	PartyScene	294.35	0.29	1023.3x
	RaceHorses	294.42	0.35	839.6x
	Class C average	294.22	0.31	954.5x
	Total average	**1343.50**	**1.40**	**951.2x**

본 논문은 움직임 추정의 단순 병렬화 알고리즘과 SIMT 구조를 고려한 제안된 알고리즘의 성능을 비교하기 위한 실험을 진행하였다. 그 결과를 표 8에 정리하였다. 이 실험 결과의 참조 모델은 제안된 알고리즘에서 적응적인 탐색영역만을 적용하지 않고 단순 병렬화한 것이다. 제안된 알고리즘은 단순 병렬화 알고리즘 대비 0.6%의 BD-rate 상승을 얻었지만, 57.5%의 추가적인 움직임 추정 *TR* 상승을 얻는다.

다음은 Low delay P 환경 Common Test Condition (CTC) 대비 표 5에 제시된 참조 모델 및 제안하는 알고리즘의 부호화 성능을 제시한다. 먼저 CTC 대비 참조 모델의 부호화 성능은 Class A,B 그리고 C 영상에서 평균 Y BD-rate 17.5% 상승이다. 또한, CTC 대비 제안하는 알고리즘의 평균 부호화 *TR* 및 평균 정수화소 움직임 추정 *TR*은 각각 46.2%와 99.8%이며 부호화 성능은 평균 Y BD-rate 18.8% 상승이다. 이와 같이 상대적으로 큰 부호화 성능 저하를 얻는 이유는 표 5에 제시된 제약조건 때문이다. 하지만 본 논문의 목표가 low delay P 환경에서 low complexity 알고리즘 구현이기 때문에 위와 같은 제한점을 두었고, 이 참조 모델과의 성능을 비교했다. 비록 본 논문의 알고리즘이 CTC 환경에서 용인되는 성능을 얻기 위해서는 새로운 문턱 값을 정하기 위한 모델링의 과정을 거쳐야 하지만 MVD를 통계적으로 이용하여 탐색영역을 적응적으로 줄이는 알고리즘의 기본 개념을 적용하는 것에는 문제가 없다. 따라서 향후 CTC 환경에서 제안하는 알고리즘을 적용하기 위한 연구가 필요하다.

Ⅵ. 결 론

본 논문은 MVD를 이용한 적응적인 탐색영역 결정 알고리즘을 제안하고, 결정된 탐색영역을 GPU기반 정수화소 움직임 추정에 적용함으로써 단순 병렬화 이득뿐만 아니라 적응적인 탐색영역으로 인한 추가적인 움직임 추정 고속화를 이루었다. 제안된 알고리즘은 프레임 단위로 전역 움직임 정도를 파악하여 입력 영상을 두 가지 모델, LMF와 SMF로 분류하고, 각 모델의 지역 움직임 특성을 CTU 단위로 파악하여 적응적인 탐색영역을 적용하였다. 이와 같은 적응적인 탐색영역은 CPU에서 결정이 되고, 이 결과는 GPU로 전송되어 정수화소 움직임 추정이 시작된다. 움직임 추정은 프레임 단위로 진행되며, 움직임 추정 탐색영역의 시작점은 SSP를 이용한다. 제안된 알고리즘은 일반적인 성능 평가를 위하여 학습 집합뿐만 아니라 검증 집합에서 실험을 진행하였다. 실험 결과는 CPU 기반의 참조 모델 대비 1.1%의 BD-rate 상승, 37.9%의 부호화 및 951.2배 빠른

표 8. 적응적인 탐색영역의 부호화 성능
Table 8. Coding performance of adaptive search range

Sequence		BD-rate (%)			IME Time (ms)		IME TR (%)
		Y	U	V	Anchor	Algorithm	
Class A	Traffic	0.7	0.9	0.9	7314.6	2864.2	60.8
	PeopleOnStreet	0.3	1.4	1.3	7314.6	2979.8	59.3
	Class A average	0.5	1.1	1.1	7314.6	2922.0	60.1
Class B	Kimono	0.2	0.3	0.2	3742.6	1790.2	52.2
	ParkScene	0.8	0.6	0.7	3744.0	1555.5	58.5
	Cactus	0.3	0.6	0.8	3744.1	1511.2	59.7
	BasketballDrive	1.0	1.9	1.2	3742.6	1825.3	51.2
	BQTerrace	0.2	1.2	0.3	3744.1	1624.1	56.6
	Class B average	0.5	0.9	0.6	3743.5	1661.3	55.6
Class C	BasketballDrill	1.9	2.1	2.9	718.7	302.0	58.0
	BQMall	0.6	0.7	0.9	718.3	299.6	58.3
	PartyScene	0.0	0.3	-0.1	717.7	287.7	60.0
	RaceHorses	0.8	0.9	1.2	718.8	350.7	51.2
	Class C average	0.8	1.0	1.2	718.4	310.0	56.9
	Total average	**0.6**	**1.0**	**0.9**	**3292.7**	**1399.1**	**57.5**

움직임 추정 고속화를 얻었다. 한편, 단순 병렬화 방법 대비 0.6%의 부호화 성능 감소 및 57.5%의 움직임 추정 *TR*을 이루었다.

참 고 문 헌 (References)

[1] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC Complexity and Implementation Analysis," IEEE Transactions on Circuit Systems for Video Technoogy, vol. 22, no. 12, Dec. 2012.

[2] R. Li, B. Zeng, M. L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 4, no: 4, pp. 438-442, Aug. 1994

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

[4] S. Zhu, K. K. Ma, "A new diamond search algorithm for fast block matching motion estimation," Information, Communications and Signal Processing, ICICS., 292-296, vol.1, Sep. 1997.

[5] W. N. Chen, H. M. Hang, "H.264/AVC motion estimation implementation on Compute Unified Device Architecture (CUDA)," IEEE International Conference on Multimedia and Expo, pp. 697-700, April, 2008.

[6] J. Zhou, L. Jiao, X. Cao, "Implementation of parallel full search algorithm for motion estimation on multi-core processors," International Conference on Next Generation Information Technology, Gyeongju, pp. 31-35, June 2011.

[7] D. K. Lee, S. J. Oh, "Variable block size motion estimation implementation on compute unified device architecture (CUDA)," IEEE International Conference on Consumer Electronics, Las Vegas, pp. 635-636, Jan. 2013.

[8] R. Rodriguez, L. Martinez, "Accelerating H.264 Inter Prediction in a GPU by using CUDA," International Conference on Consumer Electronics (ICCE), Las Vegas, pp. 463-464, Jan. 2010.

[9] R. Rodriguez, J. L. Martinez, "Reducing Complexity in H.264/AVC Motion Estimation by using a GPU," IEEE International Workshop on Multimedia Signal Processing (MMSP), Hangzhou, pp. 1-6, Oct. 2011

[10] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," document VCEG-M33, ITU-T SG16 Q.6 Video Coding Experts Group (VCEG), April 2001.

저 자 소 개

김 상 민

- 2013년 2월 : 광운대학교 전자공학과 학사
- 2013년 3월 ~ 현재 : 광운대학교 전자공학과 석사과정
- 주관심분야 : 영상압축, 병렬화

이 동 규

- 2012년 2월 : 광운대학교 전자공학과 학사
- 2014년 2월 : 광운대학교 전자공학과 석사
- 2014년 3월 ~ 현재 : 광운대학교 전자공학과 박사과정
- 주관심분야 : 영상압축, 컴퓨터 비전

저 자 소 개

심 동 규

- 1993년 2월 : 서강대학교 전자공학과 공학사
- 1995년 2월 : 서강대학교 전자공학과 공학석사
- 1999년 2월 : 서강대학교 전자공학과 공학박사
- 1999년 3월 ~ 2000년 8월 : 현대전자 선임연구원
- 2000년 9월 ~ 2002년 3월 : 바로비젼 선임연구원
- 2002년 4월 ~ 2005년 2월 : University of Washington Senior research engineer
- 2005년 3월 ~ 현재 : 광운대학교 컴퓨터공학과 교수
- 주관심분야 : 영상신호처리, 영상압축, 컴퓨터비전

오 승 준

- 1980년 2월 : 서울대학교 전자공학과 학사
- 1982년 2월 : 서울대학교 전자공학과 석사
- 1988년 5월 : 미국 Syracuse University 전기/컴퓨터공학과 박사
- 1982년 3월 ~ 1992년 8월 : 한국전자통신연구원 멀티미디어연구실 실장
- 1986년 7월 ~ 1986년 8월 : NSF Supercomputer Center 초청 학생연구원
- 1987년 5월 ~ 1988년 5월 : Northeast Parallel Architecture Center 학생연구원
- 1992년 3월 ~ 1992년 8월 : 충남대학교 컴퓨터공학부 겸임교수
- 1992년 9월 ~ 현재 : 광운대학교 전자공학과 교수
- 2002년 3월 ~ 현재 : SC29-Korea 의장 및 MPEG Forum 부의장
- 주관심분야 : 비디오 데이터 처리, 영상압축, 멀티미디어시스템