ISSN:1226-7244 (Print)
ISSN:2288-243X (Online) j.inst.Korean.electr.electron.eng.Vol.21,No.1,100~103,March 2017
논문번호17-01-17 http://dx.doi.org/10.7471/ikeee.2017.21.1.100

HEVC 다단계 움직임 추정 기법에서 단위 연산기 개수의 최적화 방법

Optimization Method on the Number of the Processing Elements in the Multi-Stage Motion Estimation Algorithm for High Efficiency Video Coding

이 성 수*★
Seongsoo Lee*★

Abstract

Motion estimation occupies the largest computation in the video compression. Multiple processing elements are often exploited in parallel to meet processing speed. More processing elements increase processing speed, but they also increase hardware area. therefore, it is important to optimize the number of processing element. HEVC (high efficiency video coding) usually exploits multi-stage motion estimation algorithms for low computation and high performance. Since the number and position of search points are different in each stage, the utilization of the processing elements is not always 100% and the utilization is quite different with the number of processing elements. In this paper, the optimizing method is proposed on the number of processing elements. It finds out the optimal number of the processing elements for the given multi-stage motion estimation algorithm by calculating utilization and execution cycle of the processing elements.

요 약

움직임 추정기는 동영상 압축에서 가장 많은 연산량을 차지하는 연산으로, 처리 속도를 맞추기 위해 다수의 단위 연산기를 병렬로 사용하는 경우가 많다. 단위 연산기를 많이 사용할수록 처리 속도가 빨라지지만 하드웨어 면적도 커지기 때문에 단위 연산기의 개수를 최적화하는 것이 중요하다. HEVC(high efficiency video coding)의 경우 연산량을 줄이고 성능을 높이기 위해서 다단계 움직임 추정 기법을 주로 사용하는데, 각 단계마다 탐색점의 개수 및 위치가 다르기 때문에 단위 연산기의 사용률이 항상 100%가 되지 않으며 단위 연산기의 개수에 따라 사용률이 크게 달라진다. 본 논문에서는 단위 연산기의 사용률과 연산 사이클을 계산하여 주어진 움직임 추정 기법에 최적화된 단위 연산기 개수를 찾아내는 방법을 제안한다.

Key words: HEVC, Motion Estimation, Processing Element, Utilization, Optimization

* School of Electronic Engineering, Soongsil University
★ Corresponding author
e-mail: sslee@ssu.ac.kr tel: 02-820-0692
※ Acknowledgment
"This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Korea (2014R1A1A2059625)."
Manuscript received Mar. 29, 2017; accepted Mar. 30. 2017

Ⅰ. 서론

동영상 압축에서 가장 많은 연산량을 차지하는 연산은 움직임 추정(motion estimation)으로 [1]-[3], 전체 연산량의 50~90%를 차지한다. 따라서 움직임 추정은 거의 하드웨어로 구현되며, 처리 속도를 맞추기 위해 다수의 단위 연산기(processing element)를 병렬로 사용하는 경우가 많다. 단위 연산기가 많으면 그만큼 빠른 시간 내에 움직임 추정을 수행할 수 있지만, 그만큼 하드웨어 크기가 커져서 칩 가격 상승, 전력 소모 증가 등의 문제가 발생한다. 따라서 움직임 추정기를 설계하는 과정에서 단위 연산기를 몇 개로 할지 결정하는 것이 중요한데, 지금까지는 대부분 설계자가 자신의 경험에 비추어 4, 8. 16, 32 등 2의 n제곱 중에서 선택해왔다.

HEVC(high-efficiency video coding)[4]에서는 연산량을 줄이고 성능을 높이기 위해서 TZS(test zone search)[5]와 같은 다단계 움직임 추정 기법을 주로 사용한다. 이때, 각 단계마다 탐색점(search point)의 개수 및 위치가 다르기 때문에 단위 연산기의 사용률이 항상 100%가 되지 않으며 단위 연산기의 개수에 따라 사용률이 크게 달라진다.

본 논문에서는 단위 연산기의 사용률과 연산 사이클을 계산하여 주어진 움직임 추정 기법에 최적화된 단위 연산기 개수를 찾아내는 방법을 제안하고 TZS에 이를 적용하여 최적화된 단위 연산기 개수를 찾아내었다.

Ⅱ. 단위 연산기의 사용률 및 수행 시간 계산

단위 연산기의 사용률(utilization)은 움직임 추정기가 동작하는 동안에 전체 단위 연산기 중에서 몇 개가 실제로 블록 정합(block matching)을 수행하는지를 나타내는 지표로, 사용률이 높을수록 단위 연산기가 쉬지 않고 효율적으로 움직임 추정을 수행한다는 것을 의미한다.

단위 연산기가 블록 정합을 수행하는데 걸리는 수행 시간은 실제 비교하는 화소의 개수에 비례한다. 본 논문에서는 하나의 단위 연산기가 매 사이클마다 하나의 화소만을 비교한다고 가정하

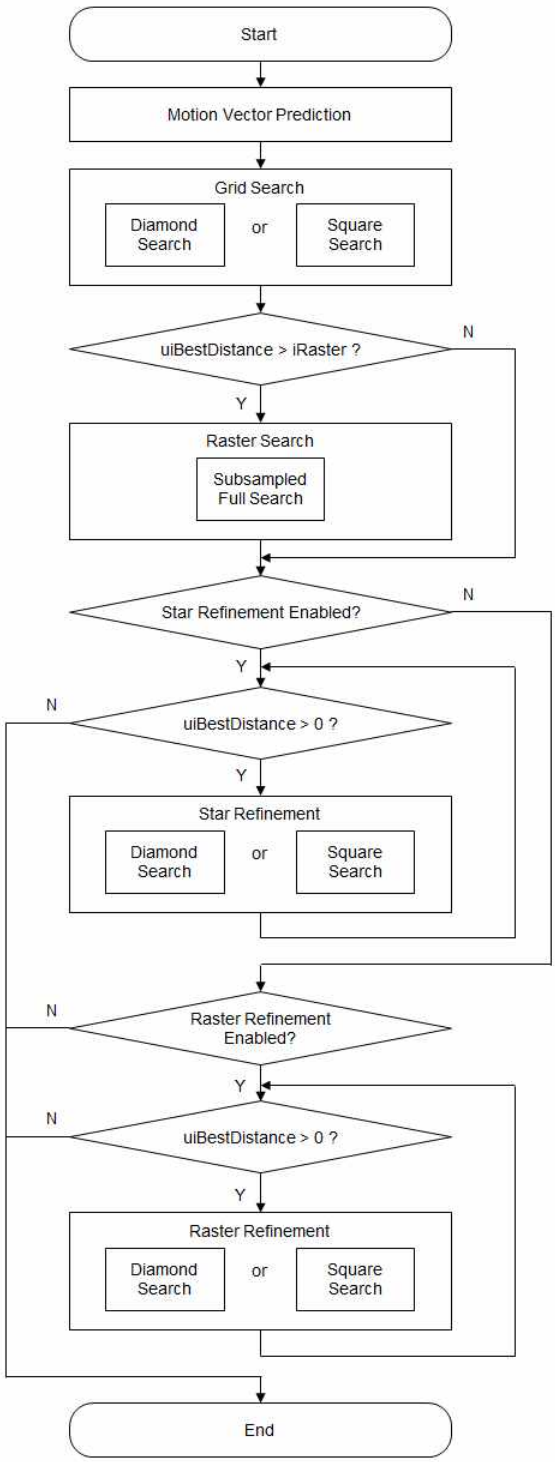

Fig. 1. TZS motion estimation algorithm
그림 1. TZS 움직임 추정 기법

였다. 대부분의 경우 예측 유닛(prediction unit) 내의 모든 화소를 비교하기 때문에 예측 유닛의 크기가 32×32 화소일 때 하나의 단위 연산기가 한 번의 블록 정합을 수행하는 시간은 1024 사이클이지만, 일부 움직임 추정 기법에서는 부표본화(subsampling) 등을 통해 이보다 적은 수의 화

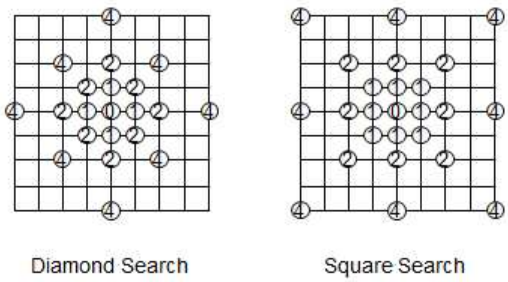

Fig. 2. Diamond pattern and square pattern
그림 2. 다이아몬드 패턴과 사각 패턴

소를 비교하기 때문에 수행 시간이 더 짧아질 수 있다. 움직임 추정기 및 단위 연산기의 구조를 바꾸어 하나의 단위 연산기가 여러 개의 화소를 비교하도록 설계하면 이에 맞추어 수행 시간을 다시 계산할 수 있다.

그림 1은 TZS 움직임 추정 기법을 나타낸 것인데, 움직임 벡터 예측(motion vector prediction), 그리드 탐색(grid search), 래스터 탐색(raster search), 스타 정교화(star refinement), 래스터 정교화(raster refinement)의 다섯 단계로 구성된다. 그리드 탐색, 스타 정교화, 래스터 정교화의 경우, 그림 2와 같이 다이아몬드 패턴과 사각 패턴 중에서 하나를 정하여 사용한다. 스타 정교화와 래스터 정교화는 둘 다 사용하지 않거나 둘 중 하나만을 사용하는데, 이들 정교화는 탐색을 끝내는 조건이 정해진 횟수가 아니기 때문에 실시간 탐색을

Table 1. Utilization and execution cycles with the number of processing elements using diamond pattern
표 1. 다이아몬드 패턴을 사용할 때 단위 연산기의 개수에 따른 사용률과 수행 시간

Number of Processing Elements	Motion Vector Prediction		Grid Search		Raster Search		Overall	
	Utilization (%)	Execution Cycles	Utilization (%)	Execution Cycles	Utilization (%)	Execution Cycles	Utilization (%)	Execution Cycles
1	100.0	4096	100.0	45056	100.0	6084	100.0	55236
2	100.0	2048	100.0	22528	99.4	3060	99.9	27636
3	66.7	2048	97.8	15360	98.8	2052	94.6	19460
4	100.0	1024	100.0	11264	98.3	1548	99.8	13836
5	80.0	1024	97.8	9216	99.4	1224	96.4	11464
6	66.7	1024	91.7	8192	97.1	1044	89.7	10260
7	57.1	1024	89.8	7168	96.6	900	86.8	9092
8	50.0	1024	91.7	6144	96.0	792	86.7	7960
9	44.4	1024	97.8	5120	98.8	684	89.9	6828
10	40.0	1024	88.0	5120	99.4	612	81.8	6756
11	36.4	1024	100.0	4096	96.0	576	88.2	5696
12	33.3	1024	91.7	4096	93.9	540	81.3	5660
13	30.8	1024	84.6	4096	100.0	468	76.0	5588
14	28.6	1024	78.6	4096	92.9	468	70.6	5588
15	26.7	1024	97.8	3072	93.9	432	81.3	4528
16	25.0	1024	91.7	3072	96.0	396	76.9	4492

Table 2. Utilization and execution cycles with the number of processing elements using square pattern
표 2. 사각 패턴을 사용할 때 단위 연산기의 개수에 따른 사용률과 수행 시간

Number of Processing Elements	Motion Vector Prediction		Grid Search		Raster Search		Overall	
	Utilization (%)	Execution Cycles	Utilization (%)	Execution Cycles	Utilization (%)	Execution Cycles	Utilization (%)	Execution Cycles
1	100.0	4096	100.0	49152	100.0	6084	100.0	59332
2	100.0	2048	100.0	24576	99.4	3060	99.9	29684
3	66.7	2048	100.0	16384	98.8	2052	96.6	20484
4	100.0	1024	100.0	12288	98.3	1548	99.8	14860
5	80.0	1024	96.0	10240	99.4	1224	95.0	12488
6	66.7	1024	100.0	8192	97.1	1044	96.4	10260
7	57.1	1024	98.0	7168	96.6	900	93.2	9092
8	50.0	1024	100.0	6144	96.0	792	93.2	7960
9	44.4	1024	88.9	6144	98.8	684	84.0	7852
10	40.0	1024	96.0	5120	99.4	612	87.8	6756
11	36.4	1024	87.3	5120	96.0	576	80.3	6720
12	33.3	1024	100.0	4096	93.9	540	87.4	5660
13	30.8	1024	92.3	4096	100.0	468	81.7	5588
14	28.6	1024	85.7	4096	92.9	468	75.8	5588
15	26.7	1024	80.0	4096	93.9	432	71.2	5552
16	25.0	1024	100.0	3072	96.0	396	82.6	4492

수행할 때에는 일반적으로 사용하지 않는다.

스타 정교화와 래스터 정교화는 사용하지 않고, 탐색 범위를 ±32, 래스터 간격을 5라고 가정했을 때, 단위 연산기의 개수에 따른 사용률과 수행 시간은 표 1 및 표 2와 같다.

지금까지는 움직임 추정기를 설계할 때에 단위 연산기의 개수를 2의 n제곱으로 결정하는 것이 훨씬 효율적이라고 알려졌으나 표 1 및 표 2를 보면 꼭 그렇지만은 않다는 것을 알 수 있다. 예를 들어 다이아몬드 패턴을 사용하는 경우에서 단위 연산기가 11~14개인 경우는 수행 시간이 거의 비슷하고 단위 연산기의 사용률은 11개일 때가 가장 높으므로 단위 연산기의 최적 개수는 11개가 된다.

III. 결론

본 논문에서는 단위 연산기의 사용률과 연산 사이클을 계산하여 주어진 움직임 추정 기법에 최적화된 단위 연산기 개수를 찾아내는 방법을 제안하고 TZS에 이를 적용하여 최적화된 단위 연산기 개수를 찾아내었다.

본 논문에서는 HEVC 움직임 추정기에서 가장 많이 사용되는 TZS를 예로 들었으나, 다른 기법도 동일한 방법으로 분석하고 단위 연산기의 개수를 최적화할 수 있다.

References

[1] H. Yang and S. Lee, "Motion Estimation Algorithm to Guarantee Hard Realtime Operation," *j.inst.Korean.electr.electron.eng,* vol. 17, no. 1, pp. 36-43, 2013.
DOI : 10.7471/ikeee.2013.17.1.036

[2] A. Hur, T. Park, and S. Lee, "Design of HEVC Motion Estimation Engine with Search Window Data Reuse and Early Termination," *j.inst.Korean.electr.electron.eng,* vol. 20, no. 3, pp. 273-178, 2016.
DOI : 10.7471/ikeee.2016.20.3.273

[3] T. Park, A. Hur, and S. Lee, "Reusing Search Window Data and Exploiting Early Termination in Variable Block Size Motion Estimation," *j.inst.Korean.electr.electron.eng,* vol. 20. no. 1, pp. 111-114, 2016.
DOI : 10.7471/ikeee.2016.20.1.111

[4] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. Circuits Syst. Video Technol.* vol. 22, no. 15, pp. 1649-1668, 2012.
DOI: 10.1109/TCSVT.2012.2221191

[5] HEVC software repository HM-11 reference model, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-11.0-dev/