HEVC 고속 부호화를 위한 SIMD 기반 Search Box 기법의 정수 화소 단위 움직임 추정 방법

석진욱 [1], 김연희 [1], 기명석 [1], 김휘용 [1]

한국전자통신연구원 방송통신미디어연구소 [1]
{jnwseok, kimyounhee, serdong, hykim5}@etri.re.kr[1]

Novel Motion Estimation Scheme to Integer Pixel with a Search Box based on SIMD for Fast HEVC encoding

Jinwuk Seok[1], Younhee Kim[1], MyungSeok Ki[2], Hui Yong Kim[1]

Broadcasting and Telecommunications Media Research Laboratory, Electronics and Telecommunications Research Institute (ETRI) [1],

요 약

본 논문은 4K UHD 입력 영상에 대한 HEVC 고속 부호화를 위하여 대부분의 상용 CPU 및 AP 에서 사용되고 있는 SIMD (Single Instruction Mutiple Data) 명령어를 사용한 고속의 정수 화소 단위 움직임 추정 방법에 대한 연구이다. 특히, IT 기기에서의 고속 동영상 부호화를 위해 기존의 SIMD 명령어를 개량하여 동일한 CPU 실행시간에 다수의 움직임 추정을 수행할 수 있는 SIMD 명령어를 사용하여 보다 같은 실행시간에 보다 넓은 영역에 대한 움직임 벡터 탐색을 수행할 수 있도록 Search Box 기법을 새로이 개발하고 이를 토대로 기존 HEVC 에서 사용되고 있는 움직임 추정 방법에 대하여 연산시간을 줄이는 동시에 화질 열화를 최소화 시킬 수 있는 방법에 대하여 논한다.

1. 서론

가장 최신의 비디오 부호화 표준인 HEVC (High Efficiency Video Coding)는 기존의 H.264/AVC (Advanced Video Coding)에 비해 동일 주관적 화질 대비 두 배정도의 부호화 효율을 나타내는 것으로 알려져 있는 반면 우수한 부호화 효율을 위해 HEVC 표준은 다양한 부호화 도구들을 제공하여 연산시간이 다른 부호화 표준에 비해 매우 크다. 특히 많은 연산시간을 사용하는 움직임 벡터 추정 연산의 경우 전체 연산시간의 50% 이상을 차지하는 것으로 알려져 있어 실시간으로 HD 혹은 4K UHD 와 같은 고해상도 영상의 부호화를 어렵게 하고 있다[1].

이러한 이유로, 움직임 추정을 고속화 하기 위한 여러 가지 방법들이 제안 되었으나[2][3][4], 많은 경우, 기존 부호화 압축 등에서 사용되던 방법에서 크게 벗어나지 않은데다, 특히 4K UHD 의 경우에는 적합하지 않는 경우도 많다. HEVC 의 참조 소프트웨어에서는 TZSearch 방법을 고속 부호화를 위한 움직임 추정 방법으로 제안하고 있는데, 이 방법은 기존의 패턴 기반의 움직임 추정에 더하여 패턴기반 움직임 추정이 국소 최소 점에서 빠져 나오지 못할 경우를 대비하여 점 방식 탐색 (Raster search) 방식을 추가하여 국소 최소 점에 대한 대비를 하고 있다[5]. 하지만 이러한 방법은 기본적으로 상용 CPU 에서 제공하는 부호화 기법들을 충분히 활용하지 못할 뿐 아니라, 점 방식 탐색의 경우, 탐색을 위한 연산수가 탐색 영역이 늘어남에 따라 제곱으로 늘어나게 되는 단점이 있어, 4K UHD 영상의 실시간 부호화를 어렵게 만든다.

따라서 전체 움직임 추정 연산 량을 줄이기 위해서는 움직임 추정을 위한 탐색 점의 개수를 줄이면서도 전체적인 압축 성능이 떨어지지 않도록 해야 한다. 서로 상반되는 이러한 문제를 해결하기 위하여 본 논문에서는 동일한 연산시간에 보다 많은 탐색 점에 대하여 움직임 추정을 가능하게 하는 SIMD 기반 Search Box 기법의 정수 화소 움직임 추정 법을 제안하였고 4K UHD 영상에 대한 실험을 통하여 제안 방법의 효과를 검증하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존 HEVC 의 정수 화소 움직임 추정 방법에 대하여 설명한다. 3 장에서는 본 연구에서 제안한 SIMD 기반 Search Box 기법의 정수 화소 움직임 추정 법을 설명한다. 4 장에서는 이에 대한 실험결과를 제시하고 마지막으로 5 장에서는 제안한 방법에 대한 검토 및 결론을 명시한다.

2. HEVC 정수 화소 움직임 추정

2.1. 패턴 탐색 기반 HEVC 정수 화소 움직임 추정

HEVC 참조 소프트웨어에서는 전체 영상에 대한 움직임

추정 및 TZSearch 로 명명된 패턴 기반 움직임 추정을 제공한다. 이 중에서 전체 영상에 대한 움직임 추정의 경우, 자명한 방법이기 때문에 본 논문에서는 논외로 한다. HEVC 참조 소프트웨어에서의 패턴 기반 움직임 추정은 다이아몬드 패턴 탐색과 정방형 패턴 탐색의 2 가지 방법이 제공된다.

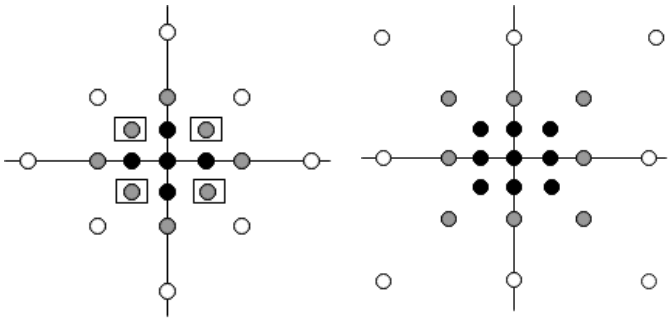

그림 1. 다이아몬드 탐색 패턴 (좌), 정방형 탐색 패턴(우)

각 패턴 탐색 영역은 n-Iteration 에 대하여 2^n 의 크기로 커진다. 그러므로 HEVC 참조 소프트웨어에서 제공하는 기본 탐색 영역인 [-64, 64] 영역까지의 패턴 탐색은 7 Iteration 으로 가능하며 정방형 탐색, 다이아몬드 탐색 모두 $8\times n+5$ 개의 탐색 점을 가지면서 정수 화소 움직임 추정을 수행하게 된다. 3840x2160 해상도를 가지는 4K UHD 영상의 경우, 움직임이 크게 나타나는 영상에 대하여는 기본 탐색영역으로는 불충분하여 [-256, 256]으로 탐색 영역을 확장하여야 하며, 정수 화소 움직임 추정을 위한 탐색 점의 개수는 Iteration 의 수가 9 로 늘어나면서 최대 77 개의 탐색 점을 가지게 된다. 하지만 이 과정에서 탐색 점 사이의 거리가 늘어나게 되면서 국소 최소 점(Local minuma)에 빠질 확률이 커지게 되어 국소 최소 점을 찾기 위한 패턴 탐색의 Iteration 수가 5 이상, 혹은 탐색 영역이 [-32, 32] 이상이 되면 점 방식 탐색을 수행한다.

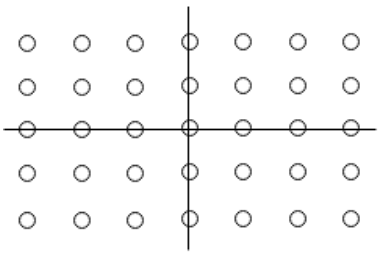

그림 2. 점 방식 탐색 (Raster Search) 패턴

점 방식 탐색의 경우 탐색 점과 탐색 점 사이의 간격은 정수 화소 기준으로 5 로 참조 소프트웨어에서는 구현되어 있다. 따라서, 점 방식 탐색은 탐색 범위가 [-64,64]의 경우에 최대 625 개 이며 [-256. 256]의 경우 최대 10404 개로 크게 늘어난다. 실험적으로 해상도 1920×1080 의 HD 급 영상의 경우, 탐색 범위가 [-64,64]가 되어도 충분하지만, 3840×2160 의 4K UHD 영상의 경우 최소 [-128,128]에서 최대 [-256, 256]의 탐색 범위를 가져야 한다. 그 이유는 탐색 범위가 작을수록, 4K UHD 영상내부의 객체의 움직임 추정이 충분하지 않아 영상과 영상 사이의 간격이 큰 경우에 많은 CU 의 예측 모드가 INTRA 로 모드가 추정되고 이 때문에 비트 량이 급격히 증가하는 경우가 발생한다. 하지만 탐색 범위가 크면 클수록 부호화 시간이 늘어나게 되는데 보통 [-64,64]의 경우보다 [-256,256]의 탐색 범위가 최소 2 배 가량 연산 시간이 증가하게 된다.

2.2. Bi-Prediction HEVC 정수 화소 움직임 추정 및 참조 소프트웨어의 정수화소 움직임 추정 알고리즘

움직임 추정을 위한 참조 영상이 하나인 경우에 사용되는 패턴 탐색 방법과 달리, 참조 영상이 두 개 이상의 경우가 되는 Bi-Ptrediction 의 경우에는 참조 영상이 하나인 경우보다 영상간 시간 차이가 작거나, 이미 하나의 참조 영상에서 패턴 탐색에 의한 움직임 예측 정보가 존재하는 경우이기 때문에, 2.1 절에서와 같이 광범위한 움직임 예측을 하지 않고 국소적인 탐색 영역에 대하여 움직임 예측을 수행한다. 따라서, 탐색 범위는 참조 소프트웨어에서는 이 경우에 4 를 주어 탐색을 수행하지만, 고속화를 위해 탐색 범위를 1 로 줄여도 화질 열화는 그렇게 심각하지 않다.

이상의 결과를 종합하여 HEVC 참조 소프트웨어에서의 정수 화소 움직임 추정 알고리즘은, 기본 값으로 정해진 알고리즘 정의 변수에서 움직임 벡터 $\boldsymbol{V_x} \in \mathbb{R}^2$ 에 대하여 SAD 값의 정의를 식 (1)과 같을 때 다음과 같이 정의된다.

$$h(V_x) \equiv \sum_{k\in M} |P_k - (R_k^{L(z,b)} + V_x)|_2 \tag{1}$$

Pattern Search Algorithm

Step 1 $\begin{cases} h(V_A) < h(V_0) & V_s = V_A \\ h(V_A) \geq h(V_0) & V_s = V_0 \end{cases}$

Step 2 *Motion Estimation with Pattern Search for IDist*

Step 3 $\begin{cases} \min\limits_{x\in\rho_{n+1}} h(V_x(n+1)) < \min\limits_{x\in\rho_n} h(V_x(n)) & R(n+1) = 0 \\ \min\limits_{x\in\rho_{n+1}} h(V_x(n+1)) < \min\limits_{x\in\rho_n} h(V_x(n)) & R(n+1) = R(n)+1 \end{cases}$

Step 4 *If* $R(n) < 3$ *then* IDist = IDist × 2, $n \leftarrow n+1$ *and go to step 2*

Step 5 *If* $|V_x(n) - V_s|_2 = 1$ *then Neighboring 2 point Search.*
If $|V_x(n) - V_s|_2 \leq 32$ *then go to step 9*

Step 6 *Motion Estimation with Raster Search for all Search Range*

Step 7 *Motion Estimation with Pattern Search if* $|V_x(n) - V_R|_2 > 0$, *after* $\text{IDist} = \frac{1}{2}|V_x(n) - V_R|_2$

Step 8 *If* $|V_x(n) - V_s|_2 = 1$ *then Neighboring 2 point Search.*
If $|V_x(n) - V_R|_2 > 0$ *then go to step 7*

Step 9 *End*

위 알고리즘에서 V_A 는 최적 AMVP 후보 벡터를, V_0 는 제로벡터를, V_S 는 시작 벡터를 의미하며 V_R 은 점 방식 탐색으로 찾아진 최소 벡터를 의미한다.

3. 고속 HEVC 부호화를 위한 SIMD 기반 Search Box 기법의 정수 화소 움직임 추정

영상 부호화에 있어 정수 화소 움직임 추정은 부호화기의 압축 성능을 좌우할 정도로 매우 중요한 프로세스이지만, 정수 화소 움직임 추정을 위한 목적함수의 형태는 전형적인 멀티 모달 문제로서 일반적인 비선형 최적화 알고리즘을 통해서는 압축 성능을 끌어올리기가 쉽지 않다. 특히, 작은 탐색 영역에서는 볼록(convex) 함수의 특징을 대체로 만족하지만, 전역 탐색에서는 이러한 볼록 다양체 (convex manifold)가 다수 결합된 형태로 나타나기 때문에 고속화를 위해 탐색 영역을 줄일 경우 압축 성능의 심대한 하락이 나타난다. 그러므로, 지금까지의 영상 부호화에서는 비선형 최적화의 방식보다는 발견적 방법에 기반한 움직임 추정 알고리즘이 일반적으로 사용되었으며, 본 연구에서도 영상 압축 성능을 지키기 위하여 발견적 방법에 기반한 알고리즘을 제안한다.

최근의 상용 CPU 나 AP 들은 별도의 영상 부호화

솔루션을 사용하지 않고 영상 부호화를 수행하기 위해 각종 SIMD 솔루션들을 탑재하고 있는데 Intel 계열의 CPU 의 경우 8 비트 화소들에 대하여 한번에 8 개의 가로 축 방향의 SAD 연산을 처리할 수 있도록 하는 _mm_mpsadbw_epu8 SIMD 명령어를 Intel SSE 4.1 에서 지원하고 있다[5]. 따라서, 기존 탐색 점 방식에 대하여 움직임 추정을 위한 연산 량을 효과적으로 줄이기 위해서 가로축 방향의 8 개 화소에 대응할 수 있도록 세로축 방향에서 8 개의 처리 선과 1 개의 중앙선을 포함하여 9 개의 세로축 방향의 SIMD 처리 선을 한번에 처리할 수 있도록 한다. 이에 의해, 그림 3 과 같이 9x8 의 크기를 가지는 Search Box 를 구성할 수 있으며 이를 움직임 추정을 위한 기본 패턴으로 사용한다.

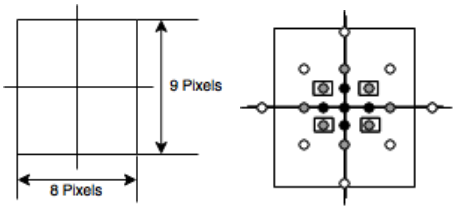

그림 3. 9x8 크기의 Search Box 패턴(좌) 와 기존 다이아몬드 탐색 방법과의 탐색 범위 비교(우)

제안한 방식의 Search Box 의 경우 연산 량이 기존 방식에 대하여 8 배 빠른 연산이 가능하므로 세로 방향으로 9 번의 연산이 추가 되어도 약 1.1 배의 연산 량으로 기존 방법에서 8 개의 탐색 점을 찾는 동안 9 배가 많은 72 번의 탐색 점을 찾을 수 있다. 따라서, Bi-Prediction 의 경우에는 한번의 연산으로 기존 방식에서 탐색 영역 4 에 해당하는 영역을 근사하게 탐색 가능하며 기존 패턴 탐색 방식보다 탐색 영역 4 까지는 기존 방식보다 약 2 배 정도 빠른 속도로 탐색이 가능하다.

3.1. Search Box 패턴 탐색 방법의 구현

제안한 9x8 Search Box 방법의 경우 가로축의 화소수가 8 인 관계로 중심점이 (i, j)=(3.5, 4) 가 되는 문제점이 있다. 따라서, 중심점 좌표를 정수로 맞추기 위해서는 적절하게 중심점 오프셋을 주어 중심점이 (3, 4) 혹은 (4, 4) 가 되도록 해야 한다.

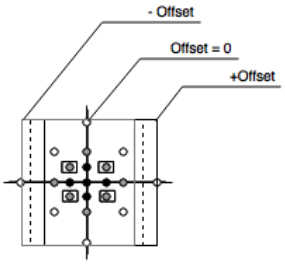

그림 3. Search Box 에 가로축 Offset 을 적용했을 경우

가로 화소 Offset 을 사용하여 중심점을 정수 화소 단위로 정의할 수 있게 되면 이를 통해 Search Box 기반의 패턴 탐색을 수행할 수 있게 된다. 기존의 패턴 탐색의 경우 기본적으로 3 번의 Iteration 에서 최소 점을 찾게 되면 1 차 패턴 탐색을 중지하는 것과 등가로 Search Box 의 경계선 내에서 최소 점이 발견되면 탐색을 멈추도록 하며 경계선에서 최소 점이 발견되면 다음과 같이 Search Box 의 중심점을 이동하여 계속 탐색을 수행하도록 한다.

$$V_s(n+1)=\bar{V}_x(n)+\begin{cases}\begin{bmatrix}s(d,D)\\-D\end{bmatrix} & \text{with offset}=s(d,0.5) \quad \text{if } x(j)=j_t\\ \begin{bmatrix}D\\s(d,D)\end{bmatrix} & \text{with offset}=0.5 \quad \text{if } x(i)=i_r\\ \begin{bmatrix}s(d,D)\\D\end{bmatrix} & \text{with offset}=s(d,0.5) \quad \text{if } x(j)=j_b\\ \begin{bmatrix}-D\\s(d,D)\end{bmatrix} & \text{with offset}=0.5 \quad \text{if } x(i)=i_L\end{cases} \tag{2}$$

식(2)에서 $\bar{V}_x(n)$ 은 $\bar{V}_x(n)=\underset{x\in\rho_n}{min}V_x(n)$ 를 만족하는 국소 최소점이며 d 는 $d=\bar{V}_x(\cdot)-V_s(\cdot)$ 로 정의되는 값이며, D 는 Search Box 가 실제 움직여야 하는 Derivation 값으로서 D=4 이며 (i, j)는 Search Box 내 화소 x 의 가로축 및 세로축 좌표로서 다음의 범위를 가진다.

$$\forall i\in\mathbb{R}[i_L,i_R],\ j\in\mathbb{R}[j_h,j_b] \text{ where } i_L<i_R,\ j_h<j_b \tag{3}$$

또한 함수 s(x,y)는 다음의 값을 나타내는 3 상 상태함수이다.

$$s(x,y)=\begin{cases}x<0 & -y\\ x>0 & y\\ x=0 & 0\end{cases} \tag{4}$$

식 (2)를 만족하는 Search Box 의 중심점 갱신 방정식은 다음의 그림으로 표현된다.

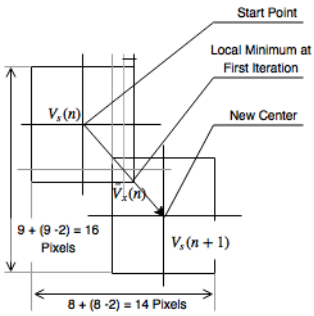

그림 4. Search Box 의 탐색 중심점 갱신의 개념도

3.2. Search Box 패턴 탐색 기반 정수 화소 움직임 추정

제안한 9x8 Search Box 기반의 패턴 탐색의 경우 대체로 3 회 정도에서 Search Box 내에 위치하는 최소 점을 찾을 수 있다. 그러나, 이렇게 찾은 최소 점은 탐색 영역이 협소하기 때문에 HD 혹은, 4K UHD 에서 충분히 작은 최소 점이 되지 못할 가능성이 높다. 그러므로 3.1 절에서 제안한 Search Box 패턴 방식을 보완하기 위하여 기존의 패턴 탐색 알고리즘과 제안한 Search Box 패턴 탐색 알고리즘을 결합시킨다.

결합된 패턴 탐색 알고리즘은 다음과 같다.

Combined Pattern Search algorithm

Step1: *Motion Estimation with Search Box for once*

Step2: *If* $\frac{1}{num(M)}h(\bar{V}_x)>\eta$ or $x\in\partial\rho$ *do Pattern Search from Step2 to Step4 in Pattern Saerch Algorithm with iDist=8, else go to Step 4*

Step 3 *Motion Estimation with Search Box once for Refinement of Pattern Search in Step 2*

Step 4 *If* $20<\frac{1}{num(M)}h(\bar{V}_x)<30$ *then do the Raster Search for all Search Range*

Step 5 *Motion Estimation with Search Box for* $x\in\rho$ *or Iteration Limit (= 3 or less)*

Step 6 *End*

4. 실험결과

본 연구에서는 4K UHD (3840×2190) 고속 부호화에서 제안한 정수 화소 움직임 추정 방법의 유효성을 확인하기 위해 총 10 종의 테스트 영상을 표 1 의 실험조건 하에서 4 개의 양자화 계수 (22, 27, 32, 37)를 사용하여 실험하고 이때의 비트 율과 PSNR 값간의 BD-rate 결과와 부호화 속도 결과를 확인 하였다

표 1. 실험조건

Sequence	3 종의 4K60P 데이터	Frame rate	60
		Profile	Main10
Number of encoded frames	32	Bit depth	10
Open GOP size	8	Max CU size	64×64

비교대상은 SSE 2.0 으로 SIMD 고속화가 완료된 HM14.0 HEVC 부호화기로서 정수 화소 움직임 추정 부분만 본 논문에서 제안한 방법이 구현되어 있는가의 차이를 가진다. 또한, Tile 병렬화가 구현되어 각 영상들은 총 54 개의 Tile 이 병렬화 되어 동시에 부호화가 이루어지도록 한다.

표 2. 정수 화소 움직임 추정 알고리즘 실험결과

		Anchor		Test		Delta		
	QPI Slice	kbps	PSNR-YUV	kbps	PSNR-YUV	delta BitRate	delta Time	Avg Delta Time
Ready SetGo	22	40440.44	42.18	40679.45	42.19	0.5910%	-0.1069	-4.16%
	27	16610.67	40.75	16722.48	40.74	0.6731%	-0.0134	
	32	8565.825	38.86	8626.545	38.86	0.7089%	-0.0254	
	37	4871.13	36.78	4904.535	36.77	0.6858%	-0.0207	
Shake NDry	22	81252.36	40.27	81327.8	40.27	0.0928%	-0.0933	-5.53%
	27	29944.01	39.03	29922.35	39.02	-0.0723%	-0.0707	
	32	13413.99	37.30	13414.61	37.30	0.0046%	-0.0361	
	37	5834.67	35.52	5813.685	35.52	-0.3597%	-0.0211	
India	22	120613.3	42.95	120764.2	42.95	0.1251%	-0.1048	-6.40%
	27	61583.85	39.80	61692.15	39.80	0.1759%	-0.0591	
	32	31475.99	36.86	31535.43	36.86	0.1889%	-0.0533	
	37	15333.33	34.22	15385.05	34.23	0.3373%	-0.0387	
						0.26%	-5.36%	-5.36%

실험에서 사용한 영상들은 움직임 추정의 효과를 잘 표현할 수 있는 영상들로서 대체로 서로 다른 영상 객체들이 서로 다른 방향의 움직임을 나타내고 있는 것으로 사용하였으며 대체로 비트 율은 높으나 PSNR 은 높게 나타날 수 있는 것을 선택하여 잘못된 알고리즘이나 구현 실수로 인한 손실이 크게 나타날 수 있는 영상을 사용하였다. 실험결과, 대체로 QP 의 크기와는 관계 없이 BD-rate 특성 및 속도 특성을 나타내고 있음을 알 수 있다.

5. 결론

본 연구에서는 HEVC 부호화기의 부호화 속도 향상을 위한 SIMD 기반 Search Box 기법의 정수 화소 움직임 추정 법을 제안하였다. 실험 결과, 제안한 방법은 기존 HM 참조 소프트웨어에서 사용한 방법대비 3.48 %의 속도 상승과 0.6%의 극히 미미한 BD-rate 손실을 평균적으로 보이고 있어 제안한 방법의 유효성을 입증할 수 있었다.

제안한 방법이 부호화 시간을 크게 단축하지 못하고 잇는 것으로 보이는데, 그 이유는 첫째, 비교대상이 이미 기존 알고리즘을 상당히 SIMD 를 통해 상당히 최적화 되어 있고, Tile 병렬화를 통해 정수 화소 움직임 추정이 상대적으로 부호화 시간에서 차지하는 비중이 작기 때문이다. 일례로 많은 연산시간을 차지하는 점 방식 탐색을 수행하지 않도록 하더라도 속도 상승률은 1% 정도로 극히 미미했으며 오히려 고속화에 의해 PSNR 이 하락할 경우, 이로 인한 연산 부하가 증가하여 연산시간이 증가하는 현상을 볼 수 있었다.

제안한 방법에서 사용한 Search Box 방법에서는 Search Box 내의 SAD 값의 총합을 구할 수 있기 때문에 이를 사용하여 보다 평균적인 특성을 가지는 움직임 벡터 추정 지표를 유도할 수 있다. 향후에는 언급한 평균적인 특성을 가지는 지표를 사용하여 기존의 발견적 방법의 기반한 움직임 추정법 대신 보다 이론적 기반을 갖춘 새로운 정수 화소 움직임 추정 법을 개발할 수 있을 것으로 보이며 본 방법을 기반으로 하는 부 화소 움직임 추정 법을 차후 연구할 예정이다.

감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (B0101-15-295, 초고품질 콘텐츠 지원 UHD 실감방송/디지털시네마/사이니지 융합 서비스 기술 개발)

참고문헌

[1] N. Purnachand, LN. Alves, A. NAvarro, "Improvements to TZ search motion estimation algorithm for multiview video coding", *IEEE IWSSIP 2012*, Vienna Apr. 2012.

[2] N. Purnachand, LN. Alves, A. NAvarro, " Fast Motion Estimation Algorithm for HEVC," *Proc. IEEE Int. Conf. Consumer Electronics*, pp. 34-37, Sep. 2012.

[3] Zhaoqing Pan, et al., " Early termination for TZSearch in HEVC Motion Estimation," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 1389-1393, May. 2013.

[4] W. Hong, "Coherent Block-Based Motion Estimation for Motion-Compensated Frame Rate Up-Conversion", *International Conference on Consumer Electronics*, Jan. 2010.

[5] Intel Intrinsic Guide, https://software.intel.com/sites/landingpage/IntrinsicsGuide.

Notations

$z \in \{LIST\_0, LIST\_1\}$	참조 영상의 List index
$L(z,b)$	z 로 특정되는 List 내의 참조영상의 지표가 b 인 참조 영상
$M \in \{2N\times 2N, N\times 2N, 2N\times N\}$	참조영상 $L(z,b)$의 특정크기 블록,
$P_k \in \mathbb{R}^2,\ k \in M$	현재 영상의 정수 화소
$\lvert x\rvert_2$	$\forall x \in \mathbb{R}^2,\ \lvert x\rvert_2 \equiv \lvert x_1\rvert + \lvert x_2\rvert$
ρ_n	n 번째 Iteration 에서 탐색 영역
$num(M)$	특정크기 블록 M 이 가진 화소 수
η	특정 블록의 크기로 SAD 값을 나눈 평균 SAD 에 대한 프로세스 Threshold 값 20 으로 설정한다.
$\partial\rho$	집합 혹은 영역 ρ의 경계면