논문 02-27-11A-5 한국통신학회논문지 '02-11 Vol.27 No.11A

움직임 벡터 예측과 서브샘플링을 이용한 새로운 고속 블록기반 움직임 벡터 탐색 알고리즘

준회원 이 성 호*, 명 진 수*, 조 용 국*, 정회원 오 승 준*, 안 창 범**

A New Fast Block-based Motion Vector Search Algorithm Using Subsampling and Motion Vector Prediction

Sung-Ho Lee*, Jin-Soo Myung*, Yong-Gook Cho* *Associate Members,*

Seoung-Jun Oh*, Chang-Beom Ahn** *Regular Members*

요 약

최근 들어 ASIC(Application Specific IC)이나 소형 시스템에서 사용할 수 있는 더 빠르고 정확한 움직임 벡터 예측방법이 요구되고 있다. 전역탐색방법은 정확도는 좋지만 많은 계산량이 요구된다. 기존의 고속 알고리즘들은 탐색점의 개수를 줄임으로써 계산량을 줄인 반면 움직임 벡터 예측의 정확도가 낮다. 본 논문에서는 기존의 고속 알고리즘 보다 높은 정확도를 가지면서 또한 낮은 계산량을 가지는 효과적인 움직임 예측 탐색 알고리즘인 NSSP(A New fast block-based Search algorithm using Subsampling and motion vector Prediction)를 제안한다. NSSP는 서브샘플링 방법을 위한 디더링 패턴과 주위 블록의 움직임벡터를 이용하는 방법을 기반으로 하고 있다. NSSP는 기존 알고리즘들과 유사한 시각적 화질을 제공하면서 입력 비디오 영상에 따라 최고 50% 까지 계산량을 줄일 수 있다.

ABSTRACT

Recently we need a faster and more accurate motion vector search algorithm in the area of designing an ASIC (Application Specific IC) and a small device. The full search algorithm is computationally intensive even though it can provide the best visual quality. Many algorithms with a reduced number of search locations have been proposed. However, most of these algorithms resulted in a significant loss in estimation accuracy as compared to the full search algorithm. In this paper, we propose a new fast and efficient search algorithm for block motion estimation that can produce proper visual quality with a significantly reduced computational complexity. The proposed algorithm, NSSP(A New fast block-based Search algorithm using Subsampling and motion vector Prediction), is based on the ideas of dithering pattern for pixel decimation and using motion vectors at adjacent macro blocks for prediction. Experimental results show that our approach reduces computational complexity by up to about 50% compared to the well-known conventional fast algorithms while providing comparable visual quality.

I. 서 론

컴퓨터가 보편화되고 인터넷이 활성화되면서 멀티미디어 컨텐츠를 쉽게 접할 수 있게 되었고 제작도 보편화되고 있다. 요즘 모든 방송국의 웹사이트(Web site)에서는 고속 인터넷 서비스와 향상된 멀티미디어 스트리밍 서비스를 통하여 일반인에게 VOD(Video On Demand) 서비스를 지원하고 있다.

* 광운대학교 전자공학부 멀티미디어 연구실(shlee@media.gwu.ac.kr), ** 광운대학교 전기공학과(cbahn@daisy.gwu.ac.kr)

논문번호 : 020201-0429, 접수일자 : 2002년 4월 29일

※ 본 연구는 한국과학재단 목적기초연구 (R01-2002- 000-00179-0) 지원으로 수행되었습니다.

또한 차세대 휴대통신서비스인 IMT-2000에서는 기존 휴대 통신기에서 제공하지 못했던 동영상서비스를 지원하고 있다. 이러한 수요와 공급으로 인해 비디오 영상에 대한 실시간 인코딩이 더욱더 요구되어지고 있다.

비디오 영상 압축은 디지털 비디오를 전송하고 저장하기 위하여 가장 핵심이 되는 기술이다. 720×420 인 NTSC 비디오 신호를 압축하지 않을 경우 249×2^{20} 비트가 필요하며 이러한 방대한 정보를 화상회의와 같은 실시간 서비스에 적용하는 것은 불가능하다. 따라서 ISO MPEG-1/2/4 와 ITU-T H.261/263/263+[1][2][3][4]와 같은 압축방식이 필요하다. 이러한 동영상 압축 표준방식에서는 비디오 신호가 가지는 프레임간 중복성을 제거하기 위해 움직임 보상방식(Motion compensation)을 사용한다. 움직임 보상방식에서는 비디오 영상이 가지는 움직임 벡터(Motion Vector: MV)가 필요하다. 영상압축 인코더에서 가장 많은 계산량을 차지하는 부분이 바로 움직임벡터를 추정하는 부분이다. 따라서 실시간으로 비디오 영상을 인코딩하기 위해서는 움직임 벡터를 추정하기 위하여 요구되는 계산량을 현저하게 줄이는 것이 관건이다.

전역 탐색법(Full Search: FS)은 탐색 영역 내에 있는 모든 탐색점에 대한 SAD(Sum of Absolute Difference)값을 계산하고, 그중 가장 작은 SAD값을 가지는 탐색점을 움직임 벡터로 추정한다. FS는 가장 정확하게 움직임 벡터를 찾는 반면 계산량이 많아 실시간 인코딩에 적합하지 않다. FS의 단점인 속도 문제를 개선하기 위하여 많은 고속 움직임 벡터 예측 방법들이 제안되었다.

고속 움직임 벡터 예측 알고리즘은 탐색점이 최소 SAD값을 가지는 점으로 움직일 때 SAD 값이 단순하게 증가한다고 가정한다. 일반적으로 이러한 고속 알고리즘은 탐색 영역 내에서 탐색점의 개수를 줄임으로써 속도를 향상시킨다. 3단계 탐색(Three-Step Search: TSS) 방법[5]은 널리 알려진 고속 움직임 벡터 예측 방법이다. TSS 방법은 FS에 비해 계산량이 적어서 속도는 빠른 반면 정확도가 떨어진다. 따라서 속도와 정확도 사이에 타협점을 찾아 적절한 방법을 적용시키는 것이 매우 중요하다.

그러나 기존의 일반적인 탐색점 수를 줄이는 방법으로는 정확도를 유지하면서 속도를 크게 개선하기 힘들다. 따라서 SAD값을 구할 때 블록의 대표값들을 이용해 서브샘플링을 하여 SAD값을 구하는 방법들이 제안되었다[6][7]. 이 방법들은 속도에 중점을 두었으므로 정확도의 유지가 관건이 된다.

탐색방법에 있어서 주변 움직임 벡터를 이용하는 방법으로는 Nearest Neighbors Search 방법[8]이 대표적인데 속도뿐 아니라 정확도면에서도 좋은 결과를 보여준다.

본 논문은 최적의 탐색점을 찾기 위하여 요구되는 계산량을 효과적으로 줄이기 위해 서브샘플링 방법으로 SAD값을 구하고, 이에 부합되는 탐색 알고리즘을 설계하여 정확도를 유지하면서도 계산 속도를 크게 향상시키는 방법을 제안한다.

Ⅱ. 블록기반 움직임 벡터 예측방법

움직임 벡터를 예측하는 방법에는 화소 반복법(Pel-Recursive Algorithm)과 블록 정합법(Block Matching Algorithm)이 있다. 화소단위로 움직임벡터를 추정하는 화소반복법은 탐색영역결정이 불필요하고 보다 정확한 값을 예측하며 적은 움직임에 유효하다. 그러나 잡음에 약하고 움직이는 물체의 경계 영역에서 보상효과가 나쁘며 계산량이 많은 것이 단점이다. 따라서 적당한 크기의 단위를 정해서 블록 단위로 움직임벡터를 추정하여 계산량이 적은 블록 정합 방법이 많이 쓰이고 있다.

블록 정합법을 기반으로 하여 움직임 벡터를 추정하기 위기 위해서는 다음과 같은 조건을 전제로 한다. 첫째, 한 물체의 움직임은 연속하는 프레임 내에서 전형성을 가지며 동일한 운동을 한다. 둘째, 영상내의 물체는 딱딱한 형태를 가져 중간에 분리되지 않는다. 그러므로 블록내의 모든 화소는 동일한 움직임을 가지며, 각 블록은 서로 독립적이라는 가정을 한다. 이 방법은 잡음에 강하고 계산량이 적어 실시간 처리에 유용하다.

그림 1은 블록 정합법을 나타내고 있다. 블록 정합법은 탐색 영역 내에 있는 탐색점들에 대한 SAD값 중에서 가장 작은 값을 가지는 탐색점으로 향하는 벡터 값을 움직임 벡터로 정한다. SAD값을 구하는 방법은 식(1)과 같다.

$$SAD(i,j)=\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}|C(x+k,y+l)-R(x+i+k,y+j+l)| \quad (1)$$

식(1)에서 $C(x+k,y+l)$는 현재 프레임의 블록 내부에 있는 화소이고, $R(x+i+k,y+j+l)$은 참조 프레임의 화소다. i와 j의 범위는 $-d\le i,j\le d$ 로서 탐색 영역을

나타내며, M과 N은 블록의 크기이다. 탐색 영역에서 SAD(*i,j*)가 최소가 되는 점의 블록을 가장 정합이 잘된 블록으로 결정하고 이 때의 (*i,j*)를 움직임 벡터로정한다.

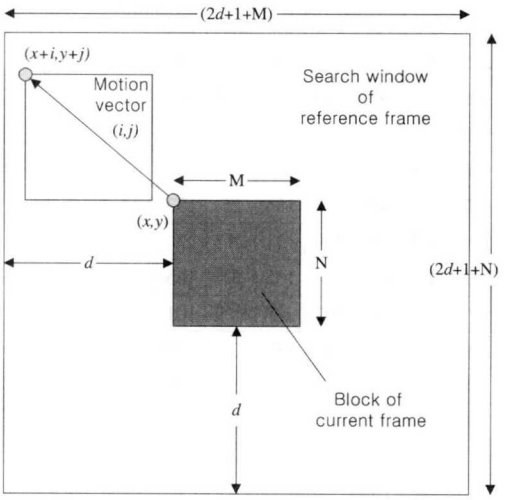


그림 1. 블록 정합법

Ⅲ. 새로운 고속 움직임 벡터 예측 방법

1. 서브샘플링을 이용한 움직임벡터 예측

본 논문에서 제안하는 움직임 벡터 예측 방법인 NSSP(A New fast block-based Search algorithm using Subsampling and motion vector Prediction)에서는 서브샘플링과 이웃 매크로블록의 움직임 벡터를 이용하여 현재 매크로블록의 움직임 벡터를 예측한다.

일반적인 블록 정합법은 SAD값을 계산할 때 매크로블록 내부의 256개 모든 화소들의 차를 더하여 SAD값을 계산한다. SAD값을 계산 할 때 256개 화소를 모두 계산하지 않고 256개 중 일부 화소만을 사용한다면 어는 정도 정확도는 떨어지지만 계산량을 많이 줄일 수 있다. 본 논문에서는 향상된 움직임 벡터 예측 속도를 유지하면서 서브샘플링에서 발생할 수 있는 오류를 보완하기 위하여 입력 영상에 존재하는 중복성을 이용하여 매크로블록 내부 화소들에 대한 대표값을 정하고, 정해진 대표값을 이용하여 SAD값을 계산하다. 서브샘플링 방법은 다음과 같다.

먼저 매크로블록의 서브샘플링을 위하여 디더링 패턴(Dithering pattern)을 이용한다. 서브샘플링할 화소를 결정하는 N×N 크기의 디더링 행렬(Dithering matrix)을 D^N이라 하고 모든 원소가 "1"인 행렬을 U^N이라 한다. 디더링 행렬의 각 원소를 $D^N_{x,y}$이라 표시하고, x와 y는 각각 행과 열을 의미한다. H.263과 MPEG 표준과 같은 비디오 영상 압축 표준에서는 움직임 예측을 매크로블록 단위로 수행한다. 따라서 본 논문에서는 디더링 행렬의 크기를 16×16으로 하고 TWSS(Two-Step Search)[9]에서 제안한 초기 디더링 행렬을 이용한다. 이때의 초기 디더링 행렬은 식(2)와 같으며, 행렬 U^N은 식(3)과 같다.

$$D^2=\begin{bmatrix}0 & 2\\ 3 & 1\end{bmatrix} \tag{2}$$

$$U^N=\begin{bmatrix}1 & 1 & . & . & . & 1\\ 1 & 1 & . & . & . & 1\\ . & . & & & & \\ . & . & & & & \\ 1 & & & & & \end{bmatrix} \tag{3}$$

N×N 크기의 디더링 행렬 D^N은 식(2)와 (3)을 이용하여 식(4)와 같이 정리된다.

$$D^N=\begin{bmatrix}4D^{N/2}+D^{N/2}_{0,0}U^{N/2} & 4D^{N/2}+D^{N/2}_{0,1}U^{N/2}\\ 4D^{N/2}+D^{N/2}_{1,0} & 4D^{N/2}+D^{N/2}_{1,1}U^{N/2}\end{bmatrix} \tag{4}$$

따라서 16×16 크기인 디더링 행렬 D^{16}은 초기 디더링 행렬 D^2로부터 식(4)를 적용하여 구해진다. 그림 2는 식(4)를 이용하여 계산한 16×16 크기의 디더링 행렬 D^{16}의 패턴이다. 디더링 행렬 D^{16}을 살펴보면 16×16 크기의 매크로블록을 4개의 서브블록(Sub-block)으로 나누어 각각에 차례로 하나씩 순서를 부여한다는 것을 알 수 있다. 따라서 주어진 순서대로 서브샘플링을 한다면 화소들 사이에 공간상의 중복성 때문에 이러한 샘플링값이 각 서브블록을 대표하는 값이 될 수 있다. 이렇게 매크로블록을 서브샘플링하여 SAD값을 결정하는 함수는 다음과 같다.

$$SAD_S(i,j)=\sum_{D^N_{k,l}=T_l}^{T_h}|C(x+k,y+l)-R(x+i+k,y+j+l)| \tag{5}$$

식(5)에서 SAD_S는 서브샘플링된 SAD값을 의미하며 T_l 과 T_h 는 각각 디더링 행렬을 서브샘플링할 때 시작하는 점과 끝나는 점을 가리킨다. 탐색영역 내에서 $SAD_S(i,j)$이 가장 적은 (*i,j*)를 움직임벡터로

0	128	32	160	8	136	40	168	2	130	34	162	10	138	42	170
192	64	224	196	200	72	232	104	194	66	226	98	202	74	234	106
48	176	16	144	56	184	24	152	50	178	18	146	58	186	26	154
240	112	208	80	248	120	216	88	242	114	210	82	250	122	218	90
12	140	44	255	4	255	36	164	14	142	46	174	6	134	38	166
204	76	236	172	196	132	228	100	206	78	238	110	198	70	230	102
60	188	28	108	52	68	20	148	62	190	30	158	54	182	22	150
252	124	220	156	244	180	212	84	254	126	222	94	246	118	214	86
3	131	35	92	11	116	43	171	1	129	33	161	9	137	41	159
195	67	227	163	203	139	235	107	193	65	225	97	201	73	233	105
51	179	19	99	59	75	27	155	49	177	17	145	57	185	25	153
243	115	211	147	251	187	219	91	241	114	209	81	249	121	217	89
15	143	47	83	7	123	39	167	13	141	45	173	5	133	37	165
207	79	239	175	199	135	231	103	205	77	237	109	197	69	229	101
63	191	31	159	55	71	23	151	61	189	29	157	53	181	21	149
255	127	223	95	247	119	215	87	253	125	221	93	245	117	213	85

그림 2. 디더링 행렬 D^{16}

결정한다.

이러한 디더링 패턴을 이용하여 최소 48개 서브샘플을 취하여 SAD값을 계산해도 256개의 화소를 모두 취하여 SAD값을 계산할 때와 비교하여 크게 성능이 떨어지지 않는다[10][11]. 따라서 매크로블록 내부의 256개의 화소 차를 모두 계산하여 SAD값을 구할 필요 없이 48개 이상을 서브샘플링하여 SAD값을 구하면 계산량을 현저히 줄일 수 있다. 따라서 본 논문은 그림 2의 D^{16}의 패턴에서 0에서 47까지의 화소들을 취하여(즉, T_l=0, T_h=47) SAD값을 구한다. 본 논문에서는 고속으로 움직임 벡터를 예측하기 위하여 서브샘플링을 이용하며 다음에 설명한 새로운 탐색 방법에 적용한다.

그림 3. 참조할 주위 매크로블록

2. 주변 움직임 벡터를 이용한 예측

영상의 공간적 유사성과 연속성을 이용하는 방법으로 이미 구해진 주위 움직임 벡터를 이용하여 현재 매크로블록의 움직임 벡터를 추정한다. 이 추정된 움직임 벡터를 Predicted Motion Vector(PMV)라 한다[8]. 그림 3은 참조할 주위 매크로블록을 나타내고 있다. MB0, MB1, MB2의 각각의 수평, 수직방향 움직임 벡터들 각각의 중간값을 취하여 PMV를 구한다. 주변 매크로블록의 코딩모드에 따라 PMV를 구하는 방법은 식(6)과 같다.

```
if(MB0 = INTRA & MB1 = INTER)
    PMV.x = MB1.x, PMV.y = MB1.y
else if(MB0 = INTER & MB1 = INTRA)
    PMV.x = MB0.x,  PMV.y = MB0.y
else if (MB0 = INTER & MB1 = INTER &
         MB2 = INTRA)
    PMV.x = (MB0.x+MB1.x)/2 ,
    PMV.y = (MB0.y+MB1.y)/2
else if (MB0 = INTRA & MB1 = INTRA )
    PMV.x = SKIP , PMV.y = SKIP
else
    PMV.x = Median(MB0.x,MB1.x,MB2.x) ,
    PMV.y = Median(MB0.y,MB1.y,MB2.y)      (6)
```

여기서 PMV.x와 PMV.y는 각각 PMV의 수평과 수직 방향 성분이고 INTRA와 INTER는 매크로블록의 코딩모드를 나타낸다. PMV.x와 PMV.y가 SKIP일 경우 주변블록의 움직임벡터를 참조하는 방법을 사용하지 않는다.

3. 새로운 고속 움직임 벡터 예측 방법

본 논문에서 제안하는 방법인 NSSP는 앞서 Ⅲ.1, 2절에서 설명한 서브샘플링 방법과 주변 블록의 움직임 벡터를 참조하는 방법을 사용한다. NSSP는 서브샘플링을 통하여 계산 속도를 향상시키고, 주변 블록의 움직임 벡터를 이용하여 보다 정확한 움직임 벡터를 찾는다.

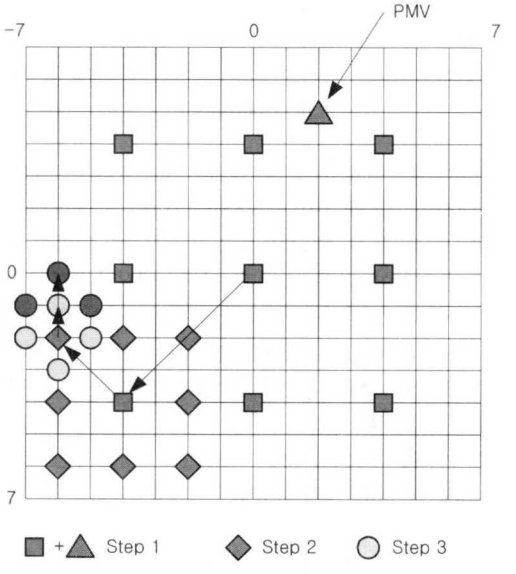


그림 4. NSSP 탐색 방법

그림 4는 탐색영역이 $\pm d = \pm 7$ 인 경우의 NSSP 방법을 보여준다. Step 1~Step 3 과정은 전화소(full-pel) 탐색과정을 나타낸 것이다. Step 1에서는 현재 매크로블록의 위치를 나타내는 (0,0)과 $\lceil d/2 \rceil$ 간격으로 위치한 8개 탐색점의 위치에 있는 매크로블록들에 대한 SAD값을 구한다. 여기에 PMV를 새로운 탐색점으로 추가하여 총 10개의 탐색점에 대해서 SAD값을 구한다. 이때 SAD값은 서브샘플링 방법으로 계산되고, SAD_S로 표시한다. 10개 SAD_S 중에서 최소값을 가지는 탐색점의 위치를 잠정적인 움직임 벡터 MV로 저장하고, 그 값을*MinSad*에 저장한다. Step 2에서는 Step 1에서 구해진 탐색점의 위치 (i,j)와 $\lceil d/4 \rceil$ 간격으로 위치한 8개 탐색점들의 위치에 대해서 SAD_S를 구하고, Step 1과 마찬가지로 최소값을 가지는 탐색점의 위치를 MV로 저장하고 그 SAD_S를 *MinSad*에 저장한다. Step 3 에서는 Step 2에서 구한 MV 위치에서 상하좌우에 있는 탐색점들에 대한 SAD_S를 구하고, 마찬가지로 최소값을 갖는 탐색점을 찾아 그 위치와 최소값을 각각 MV와 *MinSad*에 저장한다. Step 3에서는 중앙 위치에 있는 탐색점이 최소값을 가지면 탐색을 종료하고, 그렇지 않은 경우에는 다시 최소점으로 이동하여 Step 3 과정을 반복한다. Step 3을 반복하는 최대 회수는 실험적으로 4회로 정하였다. Step 4 과정에서는 반화소(half-pel) 탐색을 수행한다. 반화소 탐색은 움직임 벡터로 예측된 점을 중심으로 주변 8개의 반화소 탐색점을 만들고 각각의 SAD값을 구한다. Step 4에서 반화소 탐색에 대한 SAD값을 계산할 때 서브샘플링 방법을 적용하는 가에 따라 NSSP-1 방법과 NSSP-2 방법으로 분류된다. NSSP-1 방법은 반화소 탐색에서 서브샘플링방법을 사용하지 않는 경우이고, NSSP-2 방법은 사용하는 경우이다. 이 과정에서 구한 SAD값들과 *MinSad* 중에서 최소값을 가지는 위치를 최종 움직임 벡터로 정한다. NSSP 알고리즘을 정리하면 다음과 같다.

Algorithm NSSP

초기화 과정:

$A=\{(i,j)|i,j=\pm\lceil d/2 \rceil \text{ and } 0\} \cup \text{PMV}$

/* 집합 A는 (0,0)과 주위 8개 탐색점들, 그리고 PMV의 위치 집합 */

$B(i,j)=\{(i+k,j+l)|k,l=\pm\lceil d/4 \rceil \text{ and } 0, \text{ except } k,l=0\}$

/* 집합 B(i,j)는 (i,j)를 제외한 (i,j) 주위 8개 탐색점들의 위치 집합 */

$C(i,j)=\{(i-1,j),(i+1,j),(i,j-1),(i-1,j+1)\}$

/* 집합 C(i,j)는 (i,j) 상하좌우 4개 탐색점들의 위치 집합 */

Step 1:

For all $(m,n)\in A$, find a pair of (m,n) such that $SAD_S(m,n)$ is minimum.

$MV=(m,n)$; $MinSad=SAD_S(m,n)$; $B(i,j) = B(m,n)$;

Step 2:

For all $(m,n)\in B$, find a pair of (m,n) such that $SAD_S(m,n)$ is minimum.

if ($SAD_S(m,n) < MinSad$)

  $MV=(m,n)$; $MinSad=SAD_S(m,n)$; $C(i,j) = C(m,n)$;

Step 3:

for(r=0; r<4; r++) { /* 최대 4회 반복 */

  For all $(m,n)\in C$, find a pair of (m,n) such that $SAD_S(m,n)$ is minimum.

  if ($SAD_S(m,n) < MinSad$)

    $MV=(m,n)$; $MinSad=SAD_S(m,n)$; $C(i,j)=C(m,n)$;

```
    else break;  /* 중앙점이 최소이면 종료 */
}
```

Step 4: /* 반화소(half pel) 탐색 */
For all $(m,n)\in\{(i,j)|\ (i,j)$ are 8 half-pel's around MV$\}$, find a pair of (m,n) such that $SAD_S(m,n)$ is minimum.
if ($SAD_S(m,n)$ < *MinSad*) MV=(m,n);

Ⅳ. 실험 결과

H.263+ 코덱의 움직임 예측 모듈을 기존 방법과 NSSP로 대치하여 각 방법들의 성능을 비교한다. 사용한 실험 시퀀스는 H.263과 MPEG에서 성능 검사를 위하여 가장 보편적으로 사용하는 움직임이 적은 컨테이너(Container), 움직임이 약간 있는 코스트가드(Coastguard), 움직임이 많은 스테판(Stefan) 시퀀스이다. 각 비디오 시퀀스에 대하여 CIF 크기를 사용하고 비트율을 고정한 경우와 프레임율을 고정한 경우로 분류하여 실험한다. 그리고 NSSP에서는 반화소 탐색에서 서브샘플링 방법을 적용시키지 않은 것(NSSP-1)과 서브샘플링 방법을 적용시킨 것(NSSP-2)으로 분류하여 성능을 비교한다. 또한 실험에서 비교 알고리즘인 전역탐색(Full Search: FS), 3단계 탐색(Three-Step Search: TSS), Nearest Neighbors Search(NEAR) 방법에 대해서도 NSSP와 마찬가지로 반화소 탐색방법을 적용하여 실험하였다.

1. 비트율을 고정한 실험

CIF 시퀀스에 대하여 목표 비트율을 300 kbps (kilo bits per second)로 고정하고 실험한다. 각 알고리즘의 성능은 프레임율(Frame), 인코더의 움직임 벡터 모듈에서 소요하는 시간(Time), 그리고 휘도성분의 PSNR을 측정하여 비교한다. 표 1은 각 실험 시퀀스에 대하여 비트율을 고정하여 인코딩한 경우에 대한 성능 평가 결과이다.

표 1. 프레임율, 움직임 탐색 시간, PSNR 성능 비교. (a) 스테판 (b) 코스트가드 (c) 스테판 시퀀스(CIF 크기 150 프레임 사용, 300 kbps)

구 분	FS	TSS	NEAR	NSSP-1	NSSP-2
Frame*1	27.42	23.05	27.22	27.02	26.82
Time*2	296.28	32.99	30.22	23.21	14.90
PSNR*3	25.68	24.94	25.56	25.41	24.99

주) *1 Frame(단위:fps), *2 Time(ms/frame), *3 PSNR(dB)

(a) Stefan

구 분	FS	TSS	NEAR	NSSP-1	NSSP-2
Frame	28.81	28.61	28.81	28.81	28.81
Time	342.06	33.23	29.54	24.23	15.39
PSNR	28.03	27.67	28.03	27.79	27.30

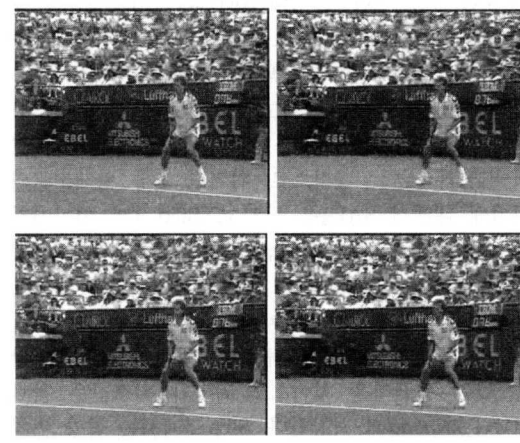
(b) Coastguard

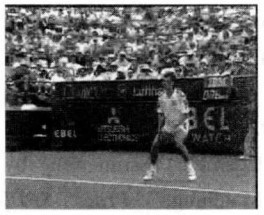
구 분	FS	TSS	NEAR	NSSP-1	NSSP-2
Frame	28.81	28.81	28.81	28.81	28.81
Time	292.75	30.54	28.94	23.75	14.08
PSNR	34.69	34.66	34.68	34.66	34.44

(c) Container

프레임율 면에서 콘테이너와 코스트가드 시퀀스에 대하여 NSSP-1이나 NSSP-2 방법을 적용한 경우에 프레임율은 NEAR 방법과 차이가 없었다. 그러나 움직임이 많은 스테판 시퀀스에서 NSSP-1 방법과 NSSP-2 방법을 적용한 경우에 각각 0.2 fps, 0.4 fps 감소하였다. PSNR 면에서 컨테이너 시퀀스에 대하여 NSSP-1 방법과 NSSP-2 방법을 적용한 경우에 NEAR 방법에 비해 각각 0.02dB, 0.22dB 감소하였고, 코스트가드 시퀀스에 대해서는 각각 0.24dB, 0.73dB 감소하였고, 스테판 시퀀스에 대해서는 각각 0.15dB, 0.57dB 감소하였다. 인코딩 시간 측면에서는 NSSP-2 방법이 다른 고속 예측 방법에 비하여 48~55% 정도 속도가 향상되었고, NSSP-1 방법은 NSSP-2 방법을 제외한 다른 고속 예측 방법들에 비하여 18~30% 정도 속도가 향상되었다.

그림 5는 스테판 시퀀스를 각 탐색방법을 사용하여 300kbps로 인코딩 하였을 때 150번째 프레임의 복원된 결과 영상을 나타낸 것이다. NSSP-1, NSSP-2 두 방법 모두 다른 고속알고리즘이나 FS에 비해 화질이 떨어지지 않음을 알 수 있다.





FS	TSS
NEAR	NSSP-1
NSSP-2	

그림 5. 각 탐색방법을 사용했을 때의 스테판 150번째 프레임 결과 영상(300kbps).

2. 프레임율을 고정한 실험

원영상의 프레임율을 30fps로 고정하고, 목표 비트율을 정하지 인코딩 한다. 따라서 각각의 알고리즘 성능에 따른 결과는 생성된 비트스트림의 비트율에 따라 판단할 수 있다. 각 알고리즘에 대한 성능은 비트율(Bitrate), 인코더의 움직임 벡터 모듈에서 걸린 시간(Time), 그리고 휘도성분의 PSNR로 표시한다. 표 2는 각 실험 영상들에 대하여 프레임율을 고정하여 인코딩한 경우에 대한 성능 평가 결과이다.

표 2. 비트율, 움직임 탐색 시간, PSNR 성능 비교. (a) 스테판 (b) 코스트가드 (c) 스테판 시퀀스 (CIF 크기 150 프레임 사용, 30 fps)

구 분	FS	TSS	NEAR	NSSP-1	NSSP-2
Bitrate*4	743.89	1019.63	776.68	790.07	829.73
Time	262.06	31.13	29.43	22.93	14.41
PSNR	29.70	29.60	29.67	29.65	29.58

주) *4 Bitrate(bps)

(a) Stefan

구 분	FS	TSS	NEAR	NSSP-1	NSSP-2
Bitrate	516.75	569.39	516.05	537.17	582.40
Time	313.50	31.33	28.35	23.10	14.41
PSNR	29.88	29.82	29.88	29.82	29.72

(b) Coastguard

구 분	FS	TSS	NEAR	NSSP-1	NSSP-2
Bitrate	126.07	124.79	124.52	125.13	137.88
Time	330.60	30.51	28.92	21.36	18.90
PSNR	31.76	31.73	31.73	31.72	31.66

(c) Container

비트율 면에서 스테판 시퀀스에서 NSSP-1 방법의 경우 FS 방법에 비해 6% 증가하였고 NEAR 방법에 비해서는 2% 증가하였으나 TSS 방법에 비해서는 비트율이 23% 감소하였다. NSSP-2 방법의 경우 FS 방법에 비해 비트율이 12% 증가하였고 NEAR 방법에 비해서는 7% 증가하였다. 코스트가드 시퀀스에 대하여 NSSP-1 방법과 NSSP-2 방법을 적용한 경우 NEAR 방법에 비해 각각 4%, 13% 비트율이 증가하였다. 컨테이너 시퀀스에 대해 NSSP-1 방법은 다른 알고리즘과 유사한 비트율을 나타냈지만 NSSP-2 방법은 약 10%정도 비트율이 증가하였다. PSNR 면에서는 비트율 증감의 영향으로 모든 시퀀스에서 유사한 PSNR수치를 나타냈다.

속도측면에서는 NSSP-1 방법을 적용하였을 때 TSS 와 NEAR 방법에 비해 19~30% 정도 속도가 향상되었고, NSSP-2 방법을 적용한 경우에는 35~54% 정도 속도가 향상되었다.

그림 6은 각 탐색방법을 사용하여 스테판 CIF 시퀀스를 인코딩 하였을 때의 결과 영상의 프레임별 비트량을 비교한 것이다. TSS방법이 가장 많은 비트량을 발생시키고, FS방법이 가장 적은 비트량을 발생시킨다. 제안한 NSSP-1방법과 NSSP-2방법은 FS와 큰 차이를 보이지 않는 것을 알 수 있다.

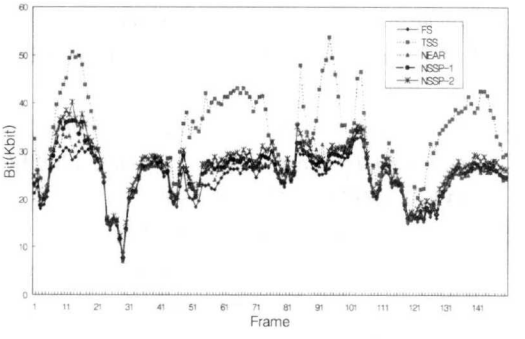


그림 6. 프레임별 비트량 성능 비교(스테판 시퀀스)

V. 결 론

본 논문에서는 디더링 패턴을 이용한 서브샘플링 방법과 주위의 움직임벡터를 이용한 탐색 방법을 결합한 새로운 탐색방법인 NSSP(A New fast block-based Search algorithm using Subsampling and motion vector Prediction)를 제안하였다. 반화소 탐색을 수행할 때 서브샘플링 방법 적용 여부에 따라 NSSP를 NSSP-1과 NSSP-2로 분류하였다.

NSSP-1을 사용한 경우에 기존의 TSS와 NEAR 방법에 비해 PSNR값은 입력 비디오 영상에 따라 유사하거나 0.1~0.2dB 정도 감소하였으나, 속도는 18~30% 정도 향상되었다. NSSP-2를 사용한 경우에 PSNR값은 기존 방법과 비교하였을 때 유사한 수준이었으나, 움직임이 많은 영상에서는 최대

0.73dB까지 감소하였다. 그러나 속도는 TSS 방법과 NEAR 방법에 비해 48~55% 정도 향상되었다.

시각적 화질측면에서 NSSP-1과 NSSP-2는 모두 FS 방법을 적용하여 얻은 결과와 유사하였다. 따라서 최고 55% 정도까지 속도가 향상된 NSSP는 기존의 고속 예측 방법들에 비해 실제적으로 유용한 방법이라고 할 수 있다.

참 고 문 헌

[1] *Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s: Video*, ISO/IEC 11172-2 (MPEG-1 Video), 1993.

[2] *Information Technology-Generic Coding of Moving Pictures and Associated Audio Information: Video*, ISO/IEC 13818-2-ITU-T Rec. H.262 (MPEG-2 Video), 1995.

[3] *Standardization Sector of ITU, Video Coding for Low Bitrate Communication*, ITU-T Rec. H.263, Mar. 1996.

[4] *"Information Technology-Generic Coding of Audio-Visual Objects" Part 2: Visual*, ISO/IEC 14496-2 (MPEG-4 Video), 1999.

[5] K. I. T. Koga, A.Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. NTC'81*, New Orleans, Dec. 1992, vol. 28, pp. 239-251.

[6] Y.K. Kang, Y. Wang and H. Kuroda, "A Globally Adaptive Pixel-Decimation Algorithm for Block-Motion Estimation," *IEEE Trans. on Circuits Syst. Video Tech*, vol 10, no 9, pp. 1006-1011, Sep. 2000.

[7] J.N. Kim and T.S. Choi, "A Fast Full-Search Motion Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan," *IEEE Trans. on Circuits Syst. Video Tech*, vol 10, no 9, pp. 1040-1048, Oct. 2000.

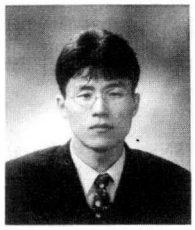
[8] M. Gallant, G. Côté, and F. Kossentini, "An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding," *IEEE Tr. on Image Processing*, VOL 8, NO 12, December 1999.

[9] Fang-Hsuan Cheng and San-Nan Sun, "New Fast and Efficient Two-step Search Algorithm for Block Motion Estimation," *IEEE Trans. on Circuits Syst. Video Tech*, vol 9, no 7, pp.977-983, Oct. 1999.

[10] M. Bierling, "Displacement Estimation by Hierarchical Block Matching," *Proc. SPIE Visual Commun. Image Processing '88*, vol 1001, pp.942-951, 1988.

[11] 김철중, 채병조, 오승준, 정광수, "서브샘플링을 이용한 수정된 Two-Step 고속 움직임 예측 알고리즘," *한국정보과학회 춘계종합학술발표회*, pp.459-462 April, 2001.

이 성 호(Sung-Ho Lee) 준회원

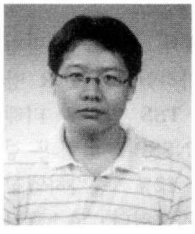


1998년 2월 : 광운대학교 전자공학과 졸업
2002년 8월 : 광운대학교 전자공학과 석사
1998년 1월~2000년 7월 : LG 산전
2001년 9월~현재:(주)인티스 정보통신연구소 연구원
<주관심 분야> 멀티미디어, 신호처리

명 진 수(Jin-Soo Myung) 준회원

1999년 2월 : 광운대학교 전자공학과 졸업
2001년 2월 : 광운대학교 전자공학과 석사
2002년 3월~현재 : 광운대학교 전자공학과 박사과정
2002년 3월~현재 : (주)인티스 정보통신연구소 연구원
<주관심 분야> 실시간 비디오 처리 및 압축, Embedded System, Transcoder

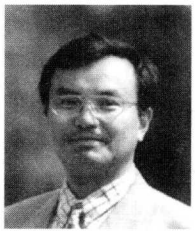
조 용 국(Yong-Gook Cho) 준회원



2001년 2월 : 광운대학교 전자공학부 졸업
2001년 3월~현재 : 광운대학교 전자공학과 석사과정
2001년 6월~현재 : (주)인티스 정보통신연구소 연구원

<주관심 분야> 영상 후처리 기법, 동영상 포맷변환, 동영상 크기 변환

오 승 준(Seoung-Jun Oh) 정회원



1980년 2월 : 서울대학교 전자공학과 졸업(학사)
1982년 2월 : 서울대학교 전자공학과 대학원 졸업(석사)
1986년 7월~1986년 8월 : NSF Supercomputer Center 초청 학생연구원
1987년 5월~1988년 5월 : Northeast Parallel Architecture Center 학생연구원
1988년 5월 : 미국 Syracuse University 전기 및 컴퓨터공학과 졸업(박사)
1982년 3월~1992년 8월 : 한국전자통신연구원 근무 (멀티미디어연구실 실장)
1992년 9월~현재 : 광운대학교 전자공학부 및 정보통신연구원 교수 (멀티미디어연구실)
2000년 3월~현재 : (주)인티스 정보통신연구소 연구소장
<주관심 분야> 비디오 처리, 비디오 및 영상압축, 멀티미디어시스템

안 창 범(Chang-Beom Ahn) 정회원

1981년 : 서울대학교 전자공학과 공학사
1983년 : 한국과학기술원 전기 및 전자공학과 공학석사
1986년 : 한국과학기술원 전기 및 전자공학과 공학박사
1986년~1991년 : University of California, Irvine 연구조교수
1991년~1992년: 생산기술연구원 부교수
1992년~현재: 광운대학교 전기공학과 교수
<주관심 분야> 영상처리, 영상압축, 다차원신호처리, 의학영상시스템