論文2003-40SP-4-2

시공간 특성을 이용한 고속 움직임 벡터 예측 방법

(A Fast Motion Estimation Scheme using Spatial and Temporal Characteristics)

魯 大 榮 *, 張 祐 演 *, 吳 承 埈 *, 石 玟 秀 **

(Dae-Young No, Ho-Yun Jang, Seoung-Jun Oh, and Minsoo Suk)

요 약

움직임 예측은 화질을 유지하면서 영상을 낮은 비트율로 부호화하는 중요한 기술이다. 일반적인 전역 탐색 방법을 사용하면 많은 계산량이 요구된다. 이전의 많은 고속 움직임 예측 방법들은 탐색점의 수를 줄이는데 초점을 두고 있기 때문에 움직임 벡터 예측의 정확도가 낮다. 그러므로 본 논문에서는 주위 블록간의 시공간적 상관관계를 이용하는 새로운 움직임 예측 방법을 제안한다. 신뢰할 수 있는 예측 움직임 벡터(Reliable Predicted Motion Vector : RPMV)를 정의한 후 전역 탐색 방법과 결과를 비교하여 RPMV의 성능을 검증한다. 검증된 RPMV 의 크기와 방향을 이용하는 새로운 움직임 벡터 예측 방법을 제안한다. 실험을 통해 제안된 방법을 현재까지 제안된 방법 중 효율적인 것으로 알려진 Nearest Neighbor 방법과 비교하였을 때 약 11~14% 정도 속도 향상이 있었다.

Abstract

The Motion Estimation (ME) process is an important part of a video encoding systems since they can significantly reduce bitrate with keeping the output quality of an encoded sequence. Unfortunately this process may dominate the encoding time using a straightforward full search algorithm (FS). Up to now, many fast algorithms can reduce the computation complexity by limiting the number of searching locations. This is accomplished at the expense of less accuracy of motion estimation. In this paper, we introduce a new fast motion estimation method based on the spatio-temporal correlation of adjacent blocks. A reliable predicted motion vector (RPMV) is defined. The reliability of RPMV is shown on the basis of motion vectors achieved by FS. The scalar and the direction of RPMV are used in our proposed scheme. The experimental results show that the proposed method is about 11~14% faster than the nearest neighbor method which is a wellknown conventional fast scheme.

Keywords : BMA, Motion estimation, Motion vector prediction, H.263

* 正會員, 光云大學校 電子工學科
(Department of Electronics Engineering, Kwangwoon University)
** 正會員, 成均館大學校 情報通信工學部
(School of Information and Communication Engineering, Sunkyunkwan University)

※ 본 연구는 2003년도 광운대학교 교내학술연구비 지원과 한국과학재단 목적기초연구 (R01-2002-000-00179-0) 지원으로 수행되었음.
接受日字:2003年6月18日, 수정완료일:2003年7月18日

1. 서 론

컴퓨터와 초고속 인터넷이 보급되고 활성화되면서 멀티미디어 컨텐츠들이 보편화 되었고, 기존의 아날로그 방식의 매체들은 디지털로 변화되어 저장되거나 전송되고 있다. 이러한 기술을 활용하여 누구나 쉽게 웹 사이트에서 VOD(Video On Demand) 서비스나 화상 통신 등의 서비스를 받을 수 있다. 디지털 방송이 실현되어 기존의 아날로그 TV에서는 접해보지 못했던 다양한 디지털 영상 서비스들을 제공받을 수 있게 된다. 이러한 다양한 수요와 공급으로 인해 디지털 비디오 영상에 대한 실시간 부호화가 절실히 요구된다.

영상압축은 디지털 비디오를 저장하거나 전송하기 위하여 가장 중요한 기술이다[1]. 예를 들어, 한 화면의 크기가 352×288 화소(pixel)로 구성된 CIF 크기의 비디오 신호를 압축하지 않고 전송하면 프레임 당 약 1200K비트(kbits)가 필요하며, 30 fps(frame per second)로 전송할 경우에는 약 35Mbps(bits per second) 의 대역폭을 요구한다. 이렇게 비디오 데이터는 비트양이 너무 많기 때문에 방대한 통신 대역폭과 저장 공간이 필요하다. 그러므로 방대한 양의 비디오 데이터를 대폭 줄이기 위해서는 압축 기술이 요구된다. 그러나 고속의 비디오 데이터를 실시간으로 압축하려면 계산량이 크게 요구되어 화상 회의와 같은 실시간 비디오 서비스에 적용하기에 많은 문제점이 있다. 따라서 동영상 압축에 대한 연구가 활발히 진행되어지고 있으며 관련 표준들이 제정되었다. 대표적인 동영상 압축 표준으로는 ISO(International Standardization Organization)의 MPEG(Moving Picture Expert Group) 계열과 ITU-T(International Telecommunications Union-Telecommunication)의 H.26x 계열이 있다[2-7].

동영상 압축 방식에서는 비디오 신호의 시간상의 중복성을 제거하여 비디오 데이터를 압축하기 위해서 움직임 보상(Motion Compensation : MC)방식을 사용한다. 이러한 MC를 사용하기 위해서는 움직임 벡터(Motion Vector : MV)를 예측하여야 한다. 움직임 벡터를 추정하는 부분은 부호화기에서 가장 많은 계산량을 요구하기 때문에 실시간으로 동영상을 압축하기 위해서는 움직임 벡터를 예측할 때 소요되는 계산량을 줄여야 한다.

가장 일반적인 움직임 예측 방법인 전역 탐색(Full Search : FS) 방법은 가장 좋은 화질을 보여주지만 탐색 영역을 모두 검색하기 때문에 많은 계산량이 요구되며 실시간으로 동영상을 압축하는데 문제가 된다. 3단계 탐색(Three Step Search : TSS) 방법 등으로 대표되는 탐색 횟수를 줄이는 고속 움직임 예측 방법들이 제안되었지만 화질의 저하를 감수해야 한다[8]. 최근에는 속도와 화질 사이의 타협점을 찾아 적절한 방법을 적용한 알고리즘들이 많이 제안되었다[9-15]. Nearest Neighbour 탐색 방법과 같이 주위 블록과의 상관관계를 이용하는 방법들은 속도뿐만 아니라 PSNR 및 시각적인 화질에서도 좋은 결과를 보여준다[16, 17]. Nearest Neighbour 탐색 방법은 주변 블록과의 공간적인 상관도를 이용하여 국부 영역에서는 다이아몬드 탐색(Diamond Search : DS) 방법을 적용하고, 광역 영역에서는 TSS 방법을 적절히 적용하고 있다.

본 논문에서는 시간적·공간적인 블록간의 상관도를 이용하여 신뢰할 수 있는 예측된 움직임 벡터(Reliable Predicted Motion Vector : RPMV)를 정의하고, RPMV의 크기와 방향 정보를 TSS 방법과 적절히 혼용하여 기존의 고속움직임 예측 방법보다 탐색횟수가 더 적으면서 화질 저하를 최소화하는 방법을 제안한다.

Ⅱ. 신뢰할 수 있는 예측 움직임 벡터

이전 프레임과 현재 프레임과의 시간적 유사성을 이용하여 움직임벡터를 예측하고 두 프레임간의 차이 값만을 부호화하게 되면 영상 정보를 낮은 비트율로 줄일 수 있다. 움직임 벡터를 예측하기 위하여 일반적으로 다음과 같은 가정을 한다.

(a) 물체의 단일성(Proximity Translation) : 움직임 영상 신호의 물리적 성질을 고려하여 영상내의 물체는 딱딱한 형태를 가지며 중간에 분리되지 않는다는 가정이다.

(b) 강도의 불변성(Intensity Stability) : 움직임으로 인해 물체가 이동했을 경우에도 물체의 밝기가 크게 변화하지 않는다는 가정이다. 마찬가지로 움직이는 물체의 부피도 일정하다.

(c) 선형 운동 : 한 물체의 움직임은 짧은 시간 내에 연속하는 프레임 내에서 전형성을 가지며 동일한 운동을 한다.

움직임 벡터를 예측하는 방법에는 화소 반복법과 블

록 정합법이 있으며, 일반적으로 블록 정합법을 사용한다[1].

2.1 RPMV에 대한 정의

고속 움직임 벡터 예측 방법들은 주위 블록과의 상관관계를 이용하여 PMV(Predicted Motion Vector)를 구한 후 TSS 방법에 적절히 응용하고 있다. 이 방법은 영상에 대한 가정 중 첫 번째 가정인 물체의 단일성에 기반을 두고 공간적인 상관관계만을 고려한 것이다. 제안된 방법은 세 번째 가정인 선형 운동을 반영하여 시간적인 블록간의 상관관계도 함께 고려한다. 즉 한 물체의 움직임은 짧은 시간 내에 연속하는 프레임 내에서 전형성을 가지며 동일한 운동을 하기 때문에 이전 프레임에서 현재 블록과 같은 위치에 있는 블록은 현재 블록과 동일한 움직임을 갖는다. 공간적·시간적 상관관계로부터 RPMV를 구하는 방법은 식 (1)과 같다.

$$\begin{aligned} SPMV &= median(MV0, MV1, MV2) \\ TPMV &= TPMV_t = MV_{t-1} \\ RPMV &= \min SAD(SPMV, TPMV) \end{aligned} \tag{1}$$

식 (1)에서 SPMV(Spatial Predicted Motion Vector)는 H.263+에서 제시하는 방법으로 <그림 1>에서 보여주는 주위 블록 MB0, MB1, MB2의 각각의 움직임 벡터 MV0, MV1, MV2들의 중간값이다. TPMV (Temporal Predicted Motion Vector)는 이전 프레임에서 현재 블록과 같은 위치에 있는 블록의 움직임 벡터를 그대로 사용한다. SPMV와 TPMV의 위치에서 구한 SAD 값을 비교하여 더 작은 값을 가지는 것을 RPMV로 택한다.

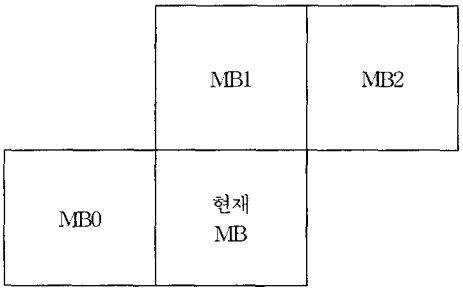


그림 1. 참조할 주위 매크로 블록
Fig. 1. Adjacent reference macroblocks.

2.2 RPMV의 신뢰성 검증

RPMV는 식 (2)와 같이 벡터의 기본 요소인 크기와 방향으로 표시할 수 있다.

$$\begin{aligned} RPMV &= (x_{RPMV}, y_{RPMV}) \\ |RPMV| &= \sqrt{x^2_{RPMV} + y^2_{RPMV}} \\ \theta_{RPMV} &= \tan^{-1}\left(\frac{y_{RPMV}}{x_{RPMV}}\right) \end{aligned} \tag{2}$$

식 (2)에서 $|RPMV|$는 크기를 θ_{RPMV}는 방향을 나타낸다. 예측기로서의 RPMV의 신뢰성을 검증하기 위해 실제 움직임 벡터와 RPMV와의 거리와 방향을 비교하였다.

식 (3)은 유클리디언 거리(Euclidean distance)를 구하는 방법이며, 식 (4)는 맨하튼 거리(Manhattan distance)를 측정하는 방법이다.

$$d_E = \sqrt{(x_1 - x_2)^2} + \sqrt{(y_1 - y_2)^2} \tag{3}$$

$$d_{L1} = |x_1 - x_2| + |y_1 - y_2| \tag{4}$$

식 (5)는 두 벡터 $\boldsymbol{X}_1 = (x_1, y_1)$과 $\boldsymbol{X}_2 = (x_2, y_2)$가 이루는 각을 구하는 식이다. 두 벡터간의 이루는 각은 방향 오차라고 한다.

$$\begin{aligned} \theta &= \cos^{-1}\left(\frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2}}\right) \\ \boldsymbol{X}_1 \cdot \boldsymbol{X}_2 &= |\boldsymbol{X}_1||\boldsymbol{X}_2| \cos\theta = x_1 x_2 + y_1 y_2 \end{aligned} \tag{5}$$

<그림 2~4>는 식 (3)을 사용하여 실제 움직임 벡터와 RPMV와의 거리를 측정한 것이며, <그림 5~7>은 두 벡터 간의 방향 오차를 실험하여 도시한 것이다. 대부분의 시퀀스(sequence)들은 프레임별 평균 거리가 4보다 작으며, foreman, stefan, table 시퀀스처럼 큰 움직임이 많은 것은 거리가 4보다 크다는 것을 알 수 있다. 움직임이 작은 시퀀스들은 대부분 RPMV와 방향이 비슷했으며, 움직임이 큰 시퀀스들도 움직임 벡터는 RPMV의 방향으로부터 $\pm\pi/2$ 이내의 방향 오차를 가진다는 것을 알 수 있다. RPMV는 실제 움직임 벡터와의 거리가 대부분이 4 이내에 존재하며 거리가 멀더라도 비슷한 방향에 있는 것을 알 수 있다.

Ⅲ. 제안된 움직임 벡터 예측 방법

<그림 2~7>이 보여 주듯이 일반적으로 RPMV는 실제 움직임 벡터와의 거리가 대부분이 4 이내에 존재

하며 거리가 4 보다 멀리 떨어져 있더라도 비슷한 방향에 있다. 이러한 RPMV의 특성을 이용하면 움직임 벡터를 빨리 예측 할 수 있다. <그림 8>은 RPMV의 방향과 거리를 적용하여 움직임 벡터를 예측하는 제안된 방법에 대한 기본 개념을 보여준다.

<그림 8>에서 점 영역은 RPMV를 중심으로 일정

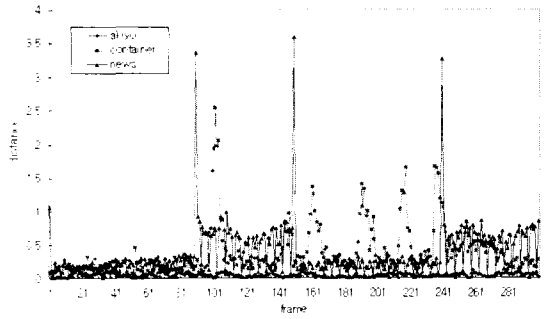


그림 2. FS와 RPMV 간의 프레임별 평균 거리 측정 1
Fig. 2. Mean distance per frame 1 between FS and RPMV.

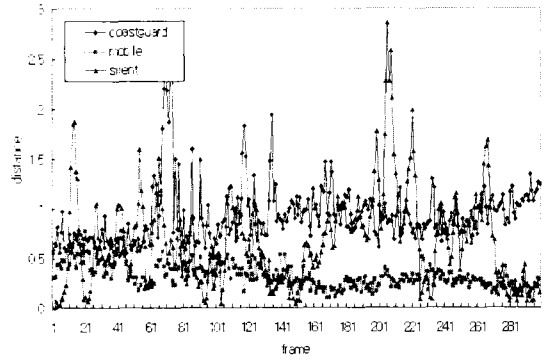


그림 3. FS 와 RPMV 간의 프레임별 평균 거리 측정 2
Fig 3. Mean distance per frame 2 between FS and RPMV.

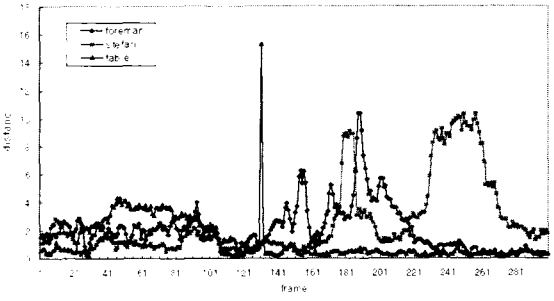


그림 4. FS와 RPMV 간의 프레임별 평균 거리 측정 3
Fig. 4. Mean distance per frame 3 between FS and RPMV.

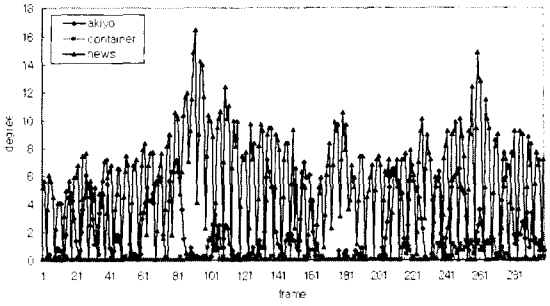


그림 5. 세 가지 시험 시퀀스에서 FS 와 RPMV 간의 프레임별 평균 방향 오차 측정(akiyo, container, news)
Fig. 5. Mean directional error per frame between FS and RPMV in three test sequences (akiyo, container, and news).

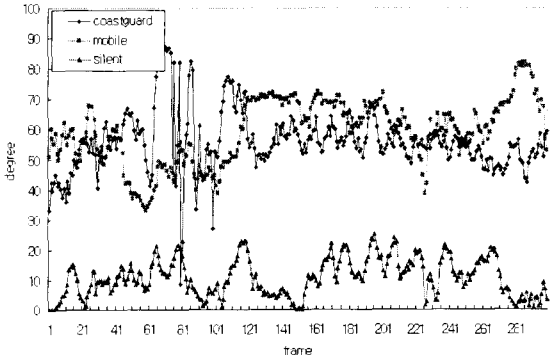


그림 6. 세 가지 시험 시퀀스에서 FS 와 RPMV 간의 프레임별 평균 방향 오차 측정(cosatguard, mobile, silent)
Fig. 6. Mean directional error per frame between FS and RPMV in three test sequences (cosaguard, mobile, and silent).

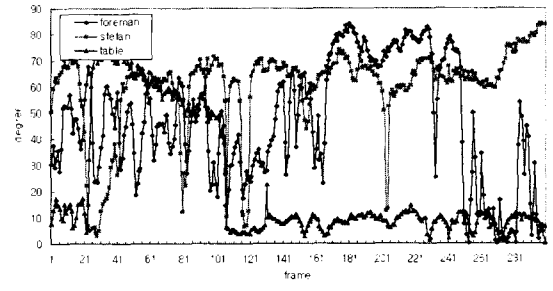
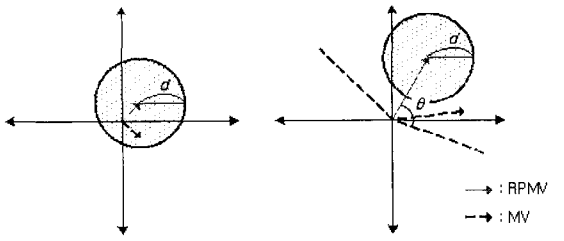


그림 7. 세 가지 시험 시퀀스에서 FS 와 RPMV 간의 프레임별 평균 방향 오차 측정(foreman, stefan, table)
Fig. 7. Mean directional error per frame between FS and RPMV in three test sequences (foreman, stefan, and table).



(a) |RPMV| ≤ d (b) |RPMV| > d

그림 8. RPMV의 크기에 따른 구분
Fig. 8. Classification of RPMV in terms of scalar.

거리 d 만큼 떨어져 있는 부분이다. θ는 RPMV와 움직임 벡터간의 방향오차이다. 움직임이 적은 영상일수록 주위 블록의 움직임 벡터로부터 구해진 RPMV의 크기는 d 보다 작다. <그림 8(a)>가 이러한 경우인데, 움직임 벡터는 RPMV를 중심으로 하는 점 영역 내에 대부분이 존재하게 된다. 움직임이 큰 영상일수록 RPMV의 크기도 d 보다 클 수 있으며, 실제 움직임 벡터는 RPMV로부터 거리가 멀어지게 되어 <그림 8(b)> 처럼 점 영역 밖에서 예측될 수 있다. 그러나 앞장에서 설명했듯이 실제 움직임 벡터는 RPMV의 방향에서 $\pm\theta$의 방향 오차 내에 존재한다.

3.1 탐색 위치 구분

RPMV의 크기와 방향에 따른 방향 오차내의 탐색점들을 구하는 것은 많은 계산을 필요로 한다. 제안하는 방법은 <그림 9>와 같이 탐색 영역을 9개의 탐색 위치로 구분하여 RPMV를 포함하는 위치를 찾는다. (0, 0)에서부터 d_T 이내에 있는 위치를 0 위치라고 하고 나머지 부분은 $\pi/4$ 간격으로 나누어 1~8 위치라고 구분하였다.

만약 $2d_T$ 가 TSS 방법의 초기 탐색 크기와 같다면 ($2d_T = \lceil p/2 \rceil$) 9개로 구분된 각각의 탐색 위치에서의 대표점들은 TSS 방법 1 단계에서 사용되는 탐색점들로 쉽게 대치할 수 있다. 먼저, RPMV의 크기와 방향을 이용하여 RPMV가 포함되는 위치를 찾는다. RPMV의 위치가 0이면 영상의 움직임은 작으며 실제 움직임 벡터는 RPMV와 가까운 곳에 존재한다고 판단하고 RPMV를 중심으로 움직임 벡터를 예측한다. RPMV의 위치가 0 이 아니면 실제 움직임 벡터는 RPMV와의 거리가 멀어질 수 있지만 $\pm\theta$ 방향 오차 내에 존재할 것이다. RPMV를 포함하는 위치와 $\pm\theta$ 방향 오차를 가지는 탐색 위치를 정한 후, 각 탐색 위치에서의 대표 탐색점들과 RPMV 간의 SAD 값을 비교하여 움직임 벡터를 예측한다. <그림 9>에서 방향 오차 θ는 $\pi/4$, $\pi/2$, $3\pi/4$ 가 될 수 있다.

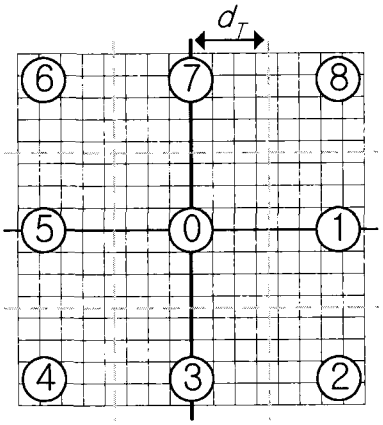


그림 9. 탐색 위치 구분
Fig. 9. Classification of search positions.

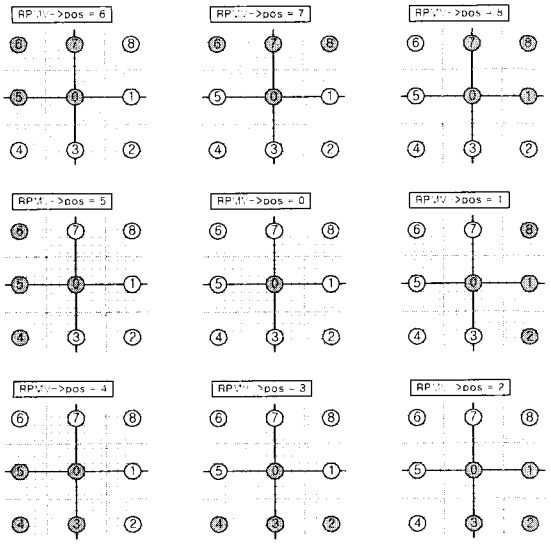


그림 10.RPMV의 위치에 따른 탐색 위치 결정 ($\theta \le \pi/4$)
Fig. 10. Search positions in each position of RPMV ($\theta \le \pi/4$).

<그림 10>은 방향 오차를 $\pi/4$로 할 때의 RPMV의 위치에 따른 탐색 위치를 나타낸다. 예를 들어, RPMV의 위치가 1이라고 하면 1 위치와 $\pm\pi/4$ 의 방향 내에 있는 탐색 위치는 8, 1, 2, 0 이 된다. 이렇게 새로 구한

탐색 위치들을 RPMV와 $\pi/4$ 방향 오차를 갖는 탐색 위치라고 한다.

<표 1~3>은 여러 시퀀스에 대해서 RPMV의 위치에서 각각의 방향 오차 내에 실제의 움직임 벡터가 존재하는 정도를 백분율로 나타낸 것이다. RPMV가 0 위치 일 때는 실제의 움직임 벡터도 0 위치인 경우만 표시하였다. 실제 움직임 벡터는 RPMV와 유사한 방향에 있으며 방향 오차를 크게 할수록 방향 오차 내에 존재한 가능성이 높아지는 것을 알 수 있다.

움직임이 작은 시퀀스일수록 대부분의 경우 0 위치에서 RPMV와 실제 움직임 벡터가 존재하며, 움직임이 큰 시퀀스들의 경우 RPMV와 실제 움직임 벡터는 80% 이상 유사한 방향에서 존재한다.

표 1. akiyo 시퀀스에서 방향 오차 내에 있는 움직임 벡터의 백분율(단위 : %)

Table 1. The percentile of MVs within each directional error range in akiyo sequence(unit : %).

RPMV의 위치		예측 방향 오차 θ		
위치	횟수	$\frac{1}{4}\pi$	$\frac{1}{2}\pi$	$\frac{3}{4}\pi$
0	118379	99.91	99.91	99.91
1	7	100.00	100.00	100.00
2	1	100.00	100.00	100.00
3	4	100.00	100.00	100.00
4	2	100.00	100.00	100.00
5	6	100.00	100.00	100.00
6	1	100.00	100.00	100.00
7	4	100.00	100.00	100.00
8	0	0	0	0

표 2. coastguard 시퀀스에서 방향 오차 내에 있는 움직임 벡터의 백분율 (단위: %)

Table 2. The percentile of MVs within each directional error range in coastguard sequence(unit: %).

RPMV의 위치		예측 방향 오차 θ		
위치	횟수	$\frac{1}{4}\pi$	$\frac{1}{2}\pi$	$\frac{3}{4}\pi$
0	114420	95.70	95.70	95.70
1	1523	96.13	97.24	98.36
2	35	94.29	97.14	100.00
3	98	97.96	98.98	98.98
4	39	89.74	92.31	97.44
5	262	91.60	92.75	94.27
6	64	90.62	96.88	100.00
7	1896	97.84	99.05	99.74
8	67	88.06	97.01	98.51

표 3. stefan 시퀀스에서 방향 오차 내에 있는 움직임 벡터의 백분율(단위 : %)

Table 3. The percentile of MVs within each directional error range in stefan sequence(unit : %).

RPMV		예측 방향 오차 θ		
위치	횟수	$\frac{1}{4}\pi$	$\frac{1}{2}\pi$	$\frac{3}{4}\pi$
0	72720	92.33	92.33	92.33
1	15684	93.45	96.89	98.71
2	481	95.38	97.90	99.12
3	895	95.46	97.77	98.69
4	556	82.64	92.71	98.29
5	26193	90.53	92.86	96.30
6	501	85.40	88.99	93.74
7	796	80.29	90.15	96.72
8	578	79.90	92.16	97.61

3.2 RPVM를 이용한 움직임 벡터 예측 방법

RPMV를 이용하여 움직임 벡터를 예측하는 방법은 <그림 11>의 순서도와 같다. P()는 주어진 벡터의 탐색 위치를 반환하는 함수이다. RP는 RPMV의 위치이며, SPs는 TSS 1 단계의 탐색점들 중에서 $\pm\theta$ 방향 오차 내에 있는 탐색점들이다. P_minSAD()는 주어진 탐색위치의 대표점들 중에서 최소 SAD 값을 가지는 탐색점의 위치를 구하는 함수이다. DS는 다이아몬드 탐색(Diamond Search)이다. 본 논문에서는 ±15 탐색영역에 대해서 $\theta=\pi/4$와 $d_T=4$를 적용하였으며 각 위치들의 대표 탐색점들은 TSS 1 단계에서 사용되는 탐색점들을 이용하였다.

먼저 RPMV의 크기와 방향으로부터 RP를 결정한다. RP가 0 위치이며 움직임이 작다고 판단하고 RPMV를 중심으로 국부 영역 탐색에 대해서 효율적인 DS를 수행하여 빠르게 움직임 벡터를 예측한다. RP가 0 이외의 위치일 경우에는 TSS 1 단계의 탐색점들 중에서

RP와 $\pm\pi/4$ 방향 오차 내에 있는 위치에 속하는 SPs들을 선택하여 RPMV와 SAD 값을 비교한다. RPMV에서 최소 SAD 값을 가지면 RPMV를 중심으로 DS 방법을 수행하고 그렇지 않다면 최소 SAD값을 가지는 탐색점을 중심으로 TSS 방법의 나머지 단계를 수행한 후 DS 방법으로 움직임 벡터를 보정한다.

DS 방법은 NNS 방법에서 사용된 방법과 같다. 가운데 주위의 상, 하, 좌, 우 4개의 탐색점을 탐색하면서 주위의 점들 중에서 SAD 값이 최소이면 그 점을 중심으로 주위 4개의 탐색점들을 탐색하는 과정을 반복한다. 가운데의 점에서 SAD 값이 최소이면 탐색을 중단한다. 제안된 방법에서는 반복 횟수를 최고 4번까지 허용하여 계산량 증가를 막는다.

<표 4>는 CIF 크기의 대표적인 시퀀스 300 프레임에 대해서 RPMV와 TSS 1 단계의 9개 탐색점들과 SAD 값을 비교하여 RPVM가 더 작은 값을 가지는 경우를 측정한 결과이다. RPMV는 움직임이 적은 시퀀스들에 대해서는 99% 이상이 선택되어지며, 움직임이 많은 시퀀스에서도 RPMV가 선택될 정도가 89%가 넘는다.

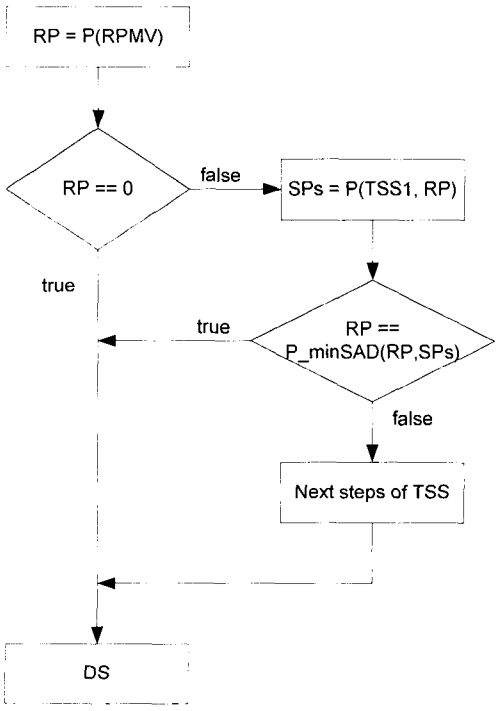


그림 11. 제안된 방법의 순서도
Fig. 11. The flow chart of the proposed scheme.

표 4. 각 시험 시퀀스에서 RPMV가 MV로 선택된 횟수를 보여주는 표
Table 4. The table shows how many times RPMV is selected as a MV in each test sequence.

시퀀스 종류	횟수	백분율(%)
akiyo	118224	99.84
container	117744	99.44
coastguard	114380	96.60
silent	114214	96.46
forman	110383	93.25
stefan	106005	89.52
table	113672	96.00

대부분의 시퀀스들은 RPMV가 선택되어지며 이 경우의 탐색 횟수는 최대 15번이다. SPs중에서 최소 SAD 값이 나타날 경우에는 기존의 TSS 경로보다 더 적은 탐색 횟수를 갖는다.

Ⅳ. 실험 및 고찰

H.263+ 부호화기[18]의 움직임 벡터 예측 모듈을 기존의 방법들과 제안하는 방법으로 대치하여 성능을 비교하였다. 실험에 사용한 비디오 시퀀스는 H.26x 와 MPEG에서 보편적으로 사용하는 akiyo, coastguard, foreman, stefan 시퀀스이다. 사용한 시퀀스의 크기는 CIF 이다. 펜티엄 3 800 MHz CPU와 256 MB SDRAM이 장착된 PC에서 실험하였다. 비트율을 고정한 경우와 비트율을 고정하지 않은 경우로 구분하여 실험하였고, 탐색영역의 크기는 ±15 로 하였다.

4.1 비트율을 고정하지 않은 실험

시퀀스의 프레임율을 30 fps(frames per second)로 고정한 후, 목표 비트율을 정하지 않고 부호화하였다. 따라서 각 방법의 성능에 따른 결과는 비트율의 수치와 움직임 벡터 예측 모듈의 속도를 비교하여 판단할 수 있다.

<그림 12~15>는 실험 시퀀스에 대한 각 방법들의 프레임별 비트량을 비교한 것이다. <표 5>는 300 프레임을 부호화하여 최종 생성된 비트스트림 (bitstream)의 크기를 측정한 것이며 <표 6>은 각 방법들 간의 프레임 당 평균 속도를 비교한 것이다.

제안한 방법은 기존 움직임 예측 방법보다 더 적은

비트스트림을 생성하였으며, stefan 시퀀스의 경우 프레임별 비트량은 FS 방법과 거의 유사한 결과를 보였다. 또한 모듈의 평균 속도는 가장 빠른 결과를 보여주며 특히 NNS 방법보다 2~3 ms/frame 정도 빠른 것을

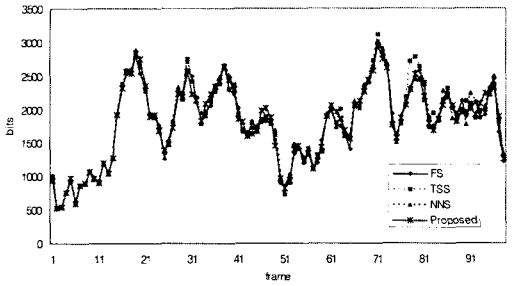


그림 12. akiyo 시퀀스에서의 프레임별 비트량 비교 (100 프레임)

Fig. 12. Comparison of bits per frame in akiyo sequence(100 frames).

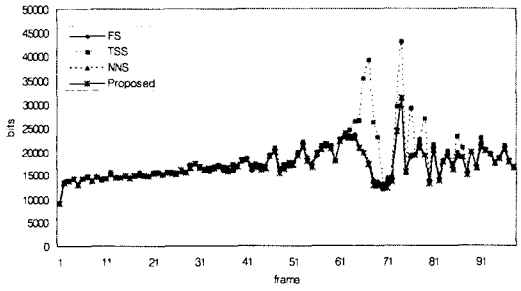


그림 13. coastguard 시퀀스에서의 프레임별 비트량 비교(100 프레임)

Fig. 13. Comparison of bits per frame in coastguard sequence(100 frames).

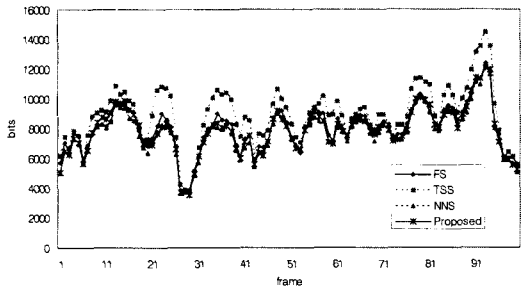


그림 14. foreman 시퀀스에서의 프레임별 비트량 비교 (100 프레임)

Fig. 14. Comparison of bits per frame in foreman sequence(100 frames).

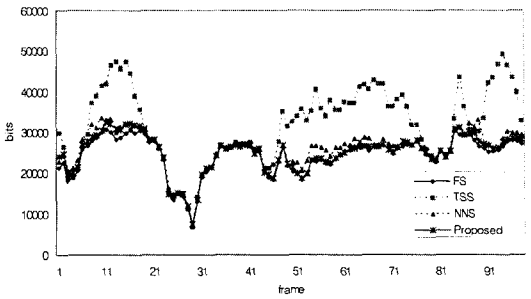


그림 15. stefan 시퀀스에서의 프레임별 비트량 비교 (100 프레임)

Fig. 15. Comparison of bits per frame in stefan sequence(100 frame).

표 5. 재구성된 시험 시퀀스의 총 데이터 량 (300 frame 사용, 단위 : bytes)

Table 5. Total bytes in reconstructed video sequences(300 frame).

구 분	FS	TSS	NNS	Proposed
akiyo	67,982	68,273(4.3)	68,188(3.0)	68,086(1.5)
coastguard	569,804	598,014(49.5)	569,042(-1.3)	564,821(-8.7)
foreman	366,987	429,097(169.2)	368,626(4.5)	367,829(2.3)
stefan	1,277,540	1,615,230(264.3)	1,374,093(75.6)	1,347,162(54.5)

주: 숫자 옆의 ( )는 FS에 대한 상대적인 비트량을 표시한 것임(단위 : 천분율)

표 6. 각 방법의 연산 속도 비교(300 frame, 단위 : ms/frame)

Table 5. Comparison in terms of computation time(300 frame, unit : ms/frame)

구 분	FS	TSS	NNS	Proposed
akiyo	574.67	37.51	24.86	21.44
coastguard	568.47	37.55	24.61	23.12
foreman	569.21	37.43	25.37	22.98
stefan	569.69	36.92	25.76	23.13

알 수 있다. 이것은 11~14% 정도 속도가 향상된 것이다.

4.2 비트율을 고정한 실험

비디오 시퀀스에 대해서 목표 비트율을 고정하고 CIF 크기의 300 프레임 시퀀스를 부호화하였다. Rate Control 방법은 프레임율을 변화시키지 않는 알고리즘

을 사용하였으며 각 움직임 벡터 예측 방법들의 성능은 화질적인 측면에서 비교하였다.

<그림 16>은 200 kbps로 부호화된 coastguard 시퀀스이며 <그림 17>은 300 kbps 로 부호화된 stefan 시퀀스이다. <표 7~8>은 각 시퀀스들의 휘도 성분의 PSNR을 측정한 것이다.

제안된 방법은 비트율을 고정하였을 경우에도 시각적인 측면이나 PNSR에서 거의 유사한 결과를 보여준다.

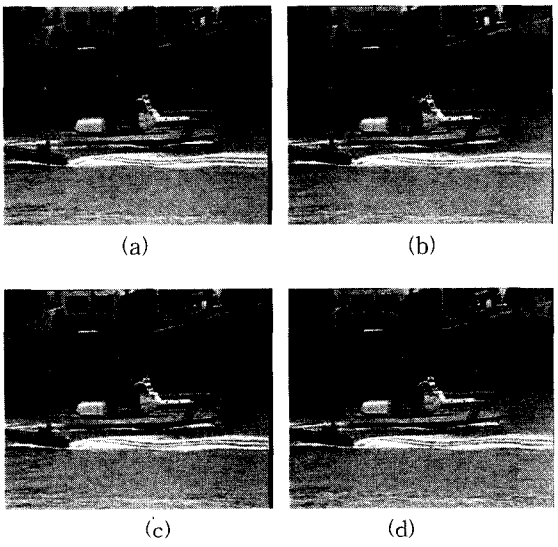


그림 16. 200kbps로 부호화된 시퀀스의 시각적 화질 비교

Fig. 16. Comparison in terms of visual quality in coastguard sequence encoded at 200kbps: (a) FS (b) TSS (c) NNS and (d) Proposed

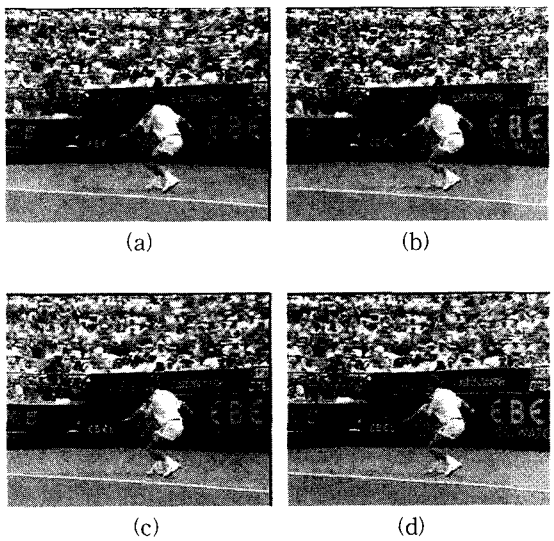


그림 17. 300kbps로 부호화된 영상의 시각적 화질 비교

Fig. 17. Comparison in terms of visual quality in stefan sequence encoded at 300kbps: (a) FS (b) TSS (c) NNS and (d) Proposed

표 7. 200 kbps 로 부호화된 시퀀스의 PSNR 비교

Table 7. Comparison in terms of PSNR in test sequences encoded at 200kbps.

구 분	FS	TSS	NNS	Proposed
akiyo	40.80	40.80	40.81	40.81
coastguard	27.17	27.00	27.18	27.17
foreman	30.33	29.84	30.32	30.29
stefan	24.71	24.54	24.73	24.70

표 8. 300 kbps 로 부호화된 시퀀스의 PSNR 비교

Table 8. Comparison in terms of PSNR in test sequences encoded at 300kbps.

구 분	FS	TSS	NNS	Proposed
akiyo	42.41	42.38	42.39	42.38
coastguard	28.68	28.48	28.67	28.67
foreman	31.93	31.18	31.83	31.78
stefan	25.51	25.06	25.50	25.58

V. 결 론

본 논문에서는 시공간 정보를 이용한 고속 움직임 벡터 탐색 방법을 제안하였다. 주위 블록의 움직임이나 이전 프레임의 같은 위치에 있는 블록의 움직임들은 현재 블록의 움직임 벡터와 상관관계가 높다. 이러한 벡터의 상관도를 이용하면 움직임 벡터를 빠르게 예측할 수 있다. 특히, 시공간 정보를 가지고 있는 벡터의 방향과 크기를 이용하여 효율적으로 탐색점의 수를 줄일 수 있었다.

제안한 방법은 기존의 고속 움직임 예측 방법들보다 PSNR 측면에서는 유사하나 속도는 크게 향상되었다. 평균적으로 21~23 ms/frame 정도이며, 특히 NNS 방법보다는 2~3 ms/frame 정도 빠름을 알 수 있다. 제안한 방법을 적용하였을 때 화질 저하가 거의 없이 기

존 방법들보다 11~14% 정도 속도가 향상되었으므로 실시간 시퀀스 부호화 서비스에 적합하다.

참 고 문 헌

[1] A. Murat Tekalp, Digital Video Processing, Prentice Hall. 1995.

[2] ISO/IEC, JTC1/SC29/WG11, Generic coding of moving pictures and associated audio, Part 2 : Video, IS 11172-2, Mar. 1995.

[3] ISO/IEC, JTC1/SC29/WG11, Generic coding of moving pictures and associated audio, Part 2 : Video, IS 13818-2, Mar. 1995.

[4] ISO/IEC, JTC1/SC29/WG11, Generic coding of moving pictures and associated audio, Part 2 : Video, IS 14496-2, Nov. 1998.

[5] ITU-T, Video Coding for Low Bitrate Communication, Draft Recommendation H.263, July 1995.

[6] ITU-T, Video Coding for Low Bitrate Communication, Draft Recommendation H.263+, July 1998.

[7] ITU-T, Joint Final Committee Draft of Joint Video Specification, Draft Recommendation H.264, July 2002.

[8] B. Furht, J. Greenberg and R. Westwater, Motion Estimation Algorithms for Video Compressiong, Florida Atlantic Univ. 1997.

[9] M. Bierling, "Displacement Estimation by Hierarchical Block Matching" Proc. Visual Comm. and Image Processing, SPIE Vol 1001, pp. 942~951, 1988.

[10] M. Gallant, G. Côté, and F. Kossentini, "An Efficient Computation Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding" IEEE Trans. on Image Processing, Vol 8, No 12, pp. 1816~1823, Dec. 1999.

[11] J.N. Kim and T.S. Choi, "A Fast Full Search Motion Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan" IEEE Trans. Circuits Syst. Video Technol. Vol. 10, No. 7, pp. 1040~1048, Oct. 2000.

[12] 김철중, 채병조, 오승준, 정광수, "서브샘플링을 이용한 새로운 고속 움직임 예측 알고리즘", 한국정보과학회 가을 학술발표회, Vol. 28, No.2, pp. 781~783, 2001

[13] Y. Nie, and K.K. Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation" IEEE Trans. on Image Processing, Vol 11, No 12, pp. 1442~1449 Dec. 2002.

[14] A. Chimienti, C. Ferraris, and D. Pau "A Complexity-Bounded Motion Estimation Algorithm" IEEE Trans. on Image Processing, Vol 11, No 4, pp. 387~392, Apr. 2002.

[15] C.H. Cheung and L.M. Po "Adjustable Partial Distortion Search Algorithm for Fast Block Motion Estimation" IEEE Trans. Circuits Syst. Video Technol. Vol. 13, No. 1, Jan. 2003.

[16] 이성호, 명진수, 오승준, 정광수, "움직임 특성을 이용한 새로운 고속 움직임 예측 방법", 신호처리 합동학술대회 2002, Vol. 15, No. 1, p. 295. 2002

[17] J.N. Kim and T.S. Choi, "Computational reduction using UESA, adaptive partial sum form gradient magnitude for fast motion estimation", in Proc. PCS, pp. 107~111, 1999.

[18] ITU-T recommendation H.263 software implementation, Digital Video Coding Group, Telenor R&D, 1995.

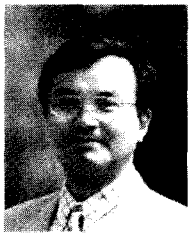
저 자 소 개

魯 大 榮(正會員)

2001년 2월 : 광운대학교 전자공학부 졸업(학사). 2001년 3월~현재 : 광운대학교 대학원 전자공학과 석사과정. 2001년 6월~현재 : 인티스(주)정보통신연구소 연구원. <주관심분야 : 영상 처리, 멀티미디어 시스템>

張 祐 演(正會員)

2001년 2월 : 광운대학교 전자공학부 졸업(학사). 2003년 2월 : 광운대학교 대학원 전자공학과 졸업(석사). 2001년 6월~현재 : 인티스(주) 정보통신연구소 연구원. <주관심분야 : 영상 처리, 영상 압축>



吳 承 埈(正會員)

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. <주관심분야 : 비디오처리, 비디오 및 영상압축, 멀티미디어시스템>

石 玟 秀(正會員)

1969년 : 서울대학교 공과대학 전자공학과(수료). 1970년 : University of California, Davis(학사). 1972년 : University of California, Davis(석사). 1974년 : University of California, Davis(박사). 1979년~1982년 : 한국과학기술원 교수. 1983년~1994년 : Syracuse University 교수. 2001년~현재 : 성균관대학교 교수. <주관심분야 : 패턴인식, 신호처리 및 통신>