움직임 예측과 신경 회로망을 이용한 고속 움직임 추정 알고리즘

정회원 최 정 현*, 이 경 환*, 이 법 기*, 정 원 식*, 김 경 규*, 김 덕 규*

Fast Motion Estimation Algorithm Using Motion Vector Prediction and Neural Network

Jung-Hyun Choi*, Kyeong-Hwan Lee*, Bub-Ki Lee*, Won-Sik Cheong*, Kyoung-Kyoo Kim*, and Duk-Gyoo Kim* *Regular Members*

요 약

본 논문에서는, 움직임 예측과 신경 회로망을 이용한 고속 움직임 추정 알고리즘을 제안하였다. 움직임 벡터의 공간적 상관성이 높다는 점을 고려하여, 현재 블록의 움직임 벡터를 인접 블록들의 움직임 벡터들로 예측하였다. 학습 시간이 빠르고 2차원 적응적 특성의 KSFM(Kohonen self-organizing feature map) 신경망을 이용하여, 움직임 벡터의 코드북(codebook)을 설계하였다. 2차원 코드북상에서 서로 비슷한 코드벡터들(codevectors)은 가까이 위치하므로, 예측 코드벡터로부터 코드북상에서 점진적으로 움직임을 추정하였다. 모의 실험 결과, 제안한 방법이 적은 계산량으로도 우수한 성능을 나타냄을 확인하였다.

ABSTRACT

In this paper, we propose a fast motion estimation algorithm using motion prediction and neural network. Considering that the motion vectors have high spatial correlation, the motion vector of current block is predicted by those of neighboring blocks. The codebook of motion vector is designed by Kohonen self-organizing feature map(KSFM) learning algorithm which has a fast learning speed and 2-D adaptive characteristics. Since the similar codevectors are closely located in the 2-D codebook, the motion is progressively estimated from the predicted codevector in the codebook. Computer simulation results show that the proposed method has a good performance with reduced computational complexity.

I. 서 론

동영상 신호의 압축을 위해서는, 연속된 프레임간의 시간 중복성을 제거하는 움직임 보상 부호화가 효율적이다. 움직임 보상 부호화는, 이전 프레임과 현재 프레임간의 움직임을 추정하는 과정과, 추정한 움직임 벡터를 사용하여 만든 예측 영상과 원 영상과의 차(displaced frame difference; DFD)를 부호화하는 과정으로 구성된다. 움직임 추정 방법으로는, 블록 단위의 정합을 통해 움직임을 추정하는 블록 정합 방법(block matching algorithm; BMA)과 화소 단위의 움직임을 추정하는 화소 순환 알고리즘(pel recursive algorithm; PRA)이 있다. 동영상 부호화에서는 적은 계산량과 하드웨어 구현의 용이성 때문에 BMA가 주로 사용된다.[1],[2]

전역 탐색 블록 정합 방법(full search BMA; FSBMA)에서는, 이전 프레임의 설정된 탐색 영역(search area)내의 모든 탐색점들(search points)에 대하여 정합을 행하여, 현재 블록의 움직임을 추정한다. 이 방법은 최적의 움직임 벡터를 추정할 수

* 경북대학교 전자전기공학부(jhchoi@palgong.kyungpook.ac.kr)
논문번호 : 99036-0128, 접수일자 : 1999년 1월 28일

있지만 많은 계산량을 필요로 한다.

움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여 탐색점의 수를 줄임으로써 고속의 움직임 추정이 가능한 3단계 탐색(three-step search; TSS) 방법[2]이 제안되었다. 이 방법은 움직임을 단계적으로 추정하는 방법으로서, FSBMA보다 계산량을 많이 줄일 수 있으나, 추정 과정에서 국부 최소 (local minimum)에 빠지는 단점이 있다.

인접 블록들의 움직임 벡터를 이용하여 움직임 벡터를 예측하고, 예측된 움직임 벡터를 탐색 기준으로 단계적인 탐색을 행하는 방법이 제안되었다[3]. 이 방법은 움직임 예측을 이용하여, TSS 방법보다 탐색점 수가 적으면서도 더 나은 성능을 얻을 수 있었다.

최근에 Lee 등은 움직임 벡터의 공간적 상관성을 이용하여 움직임 벡터를 벡터 양자화(vector quantization; VQ)하는 방법을 제안하였다[4]. 이 방법에서는, 모든 블록에 대해 FSBMA를 수행하여 구한 움직임 벡터를 입력 벡터로 두고, LBG 알고리즘[5][6]으로 반복 수행하여 코드북(codebook)을 만들고, 이를 이용하여 움직임 벡터를 벡터 양자화하였다.

본 논문에서는, 움직임 예측과 신경 회로망을 이용한 고속 움직임 추정 방법을 제안하였다. KSFM 신경망을 이용하여 움직임 벡터의 코드북을 만들고, 인접 블록의 움직임 벡터들로써 예측된 움직임 벡터를 기준으로, 2 차원 구조의 코드북상에서 점진적으로 움직임 벡터를 탐색하였다. 동영상의 초기 프레임들에서 FSBMA로 구해진 움직임 벡터들을 훈련 벡터(training vector)로 두고, 기존 LBG 방법에 비해 학습시간이 짧고, 2차원적으로 인접 코드벡터들(codevectors)간에 서로 연관성을 가지고 학습시킬 수 있는 KSFM 신경망[7]을 사용하여 코드북을 설계하였다. 이 때 코드북은 2차원 형태로 구해지며, 서로 비슷한 움직임 벡터들은 코드북 내에서 가깝게 위치하고, 서로 차이가 큰 움직임 벡터들은 멀리 떨어져 위치한다. 이와 같은 2차원 구조의 코드북 특성을 이용하여, 제안한 방법에서는 인접 블록들로 예측한 코드벡터를 기준으로 코드북상에서 점진적으로 움직임 추정을 행하였다. 또한 각 단계에서 정합 오차(matching error)를 문턱치(threshold)와 비교해서, 조건이 만족되면 탐색을 중도 정지(half stop)하였다.

제안한 방법의 성능을 평가하기 위하여 모의실험을 수행한 결과, 제안한 방법이 TSS 방법[2]에 비해 아주 적은 계산량에서도 FSBMA 방법과 비슷한 성능을 얻을 수 있음을 확인할 수 있었다. 또한 LBG를 이용하여 움직임 벡터를 벡터 양자화한 Lee 등의 방법[4]에 비해 적은 계산량에서도 우수한 결과를 나타내었다.

Ⅱ. 기존의 움직임 추정방법

1. 3 단계 탐색(TSS) 방법[2]

전역 탐색 블록 정합 방법(FSBMA)에서는, 탐색 영역내의 모든 탐색점들에 대하여 정합을 행하므로, 최적의 움직임을 추정할 수 있지만 많은 계산량을 필요로 한다. 이를 개선하기 위해, 움직임 추정 오차는 움직임 방향으로 단조 감소한다는 가정을 이용하여, 탐색 영역의 일부 탐색점에 대해서만 탐색을 행함으로써 계산량을 크게 줄일 수 있는 TSS 방법이 제안되었다. 이 방법에서는 단계를 진행하면서 참조 탐색점(reference search point)의 간격을 줄여서, 점점 세밀하게 움직임 벡터를 추정한다. 탐색 영역을 -7~7로 두었을 때, 단계별 탐색 과정은 다음과 같다.

step 1; 참조 탐색점의 간격을 4로 두고, 원점과 그 주위 8개의 참조 탐색점에 대하여 블록 정합을 행하여, 최소 오차의 탐색점 (SP_1)을 찾는다. (탐색점 수 = 9)

step 2; 참조 탐색점의 간격을 2로 두고, SP_1 주위의 8개의 참조 탐색점에 대하여 블록 정합을 행하여, 최소 오차의 탐색점 (SP_2)을 찾는다. (탐색점 수 = 8)

step 3; 참조 탐색점의 간격을 1로 두고, SP_2 주위의 8개의 참조 탐색점에 대하여 블록 정합을 행하여, 최소 오차의 탐색점 (SP_3)을 찾아서 SP_3를 최종 움직임 벡터로 결정한다. (탐색점 수 = 8)

TSS 방법에서는, 매 블록당 25개의 탐색점에 대하여 블록 정합을 행하기 때문에, 탐색 영역 내의 모든 탐색점에 대해서 탐색하는 FSBMA에 비해 계산량이 크게 감소한다. 그러나, 일부 탐색 영역에 대해서만 탐색을 행하므로, 국부 최소에 빠지는 문제점이 있다. 특히 움직임 추정 오차가 가정과 달리 움직임 방향으로 단조 감소하지 않는 경우, 추정 오

차가 매우 커지는 단점이 있다. 움직임 벡터가 (3, -3)일 때의 예를 그림 1에 나타내었다.

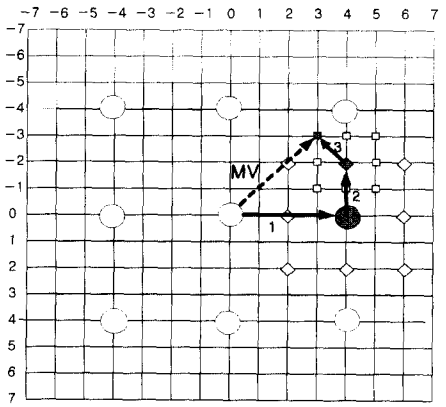


그림 1. 움직임 벡터가 (3, -3)일 때 TSS 방법의 예

2. 움직임 벡터의 벡터 양자화 방법[4]

Lee 등은 움직임 벡터를 추정하는 방법으로서, 움직임 벡터를 벡터 양자화하는 방법을 제안하였다. 이 방법에서는, 모든 블록의 움직임 벡터를 입력 벡터로 두고, LBG 방법으로 반복 수행하여 움직임 벡터에 대한 코드북을 만들고, 이를 이용하여 움직임 벡터를 양자화하였다. 이 방법의 구체적인 알고리듬을 설명하면 다음과 같다.

먼저 모든 동영상의 블록들에 대해 FSBMA로 실제의 움직임 벡터 δ를 구한다. 만약, 코드북의 크기를 M이라 했을 때, 정해진 탐색 영역에 대하여 탐색점 M개를 균일하게 분포시키고, 이들 각각을 초기 코드북의 코드벡터로 둔다.

$$C = \{\delta_1, \delta_2, \cdots, \delta_M\} \tag{1}$$

여기서, C는 코드북이며, δ_i는 코드벡터이다. (x, y)위치에서의 t번째 프레임과 $t-1$번째 프레임의 밝기값을 각각 $s_t(x, y)$와 $s_{t-1}(x, y)$로 정의하면, 현재 블록 s_t^B는

$$s_t^B = \{ s_t(x, y),\ (x, y) \in block\ B \} \tag{2}$$

와 같고, 이전 프레임의 탐색 영역 m_{t-1}^B는

$$m_{t-1}^B = \{ s_{t-1}(x, y),\ (x, y) \in search\ region\ \text{for}\ block\ B \} \tag{3}$$

과 같이 나타낼 수 있다.

코드북은 다음과 같이 *clustering* 조건과 *centroia* 조건을 반복 수행하여 구한다.

Clustering condition

$$R_i \equiv \{ s_t^B,\ m_{t-1}^B:\ d(s_t^B, m_{t-1}^B, \delta_i) \le d(s_t^B, m_{t-1}^B, \delta_j),\ 1 \le j \le N \} \tag{4}$$

여기서, $d(s_t^B, m_{t-1}^B, \delta_i)$는 코드벡터 δ_i에 대한, 현재 블록 s_t^B의 탐색 영역 m_{t-1}^B에서의 오차를 나타낸다.

Centroid condition

$$\delta_i = cent(R_i) \tag{5}$$

이 방법은, 수렴할 때까지 반복을 계속하는 LBG 방법을 이용하므로, 움직임 벡터의 코드북을 설계시 계산량이 많다. 또한 움직임을 추정할 때, 코드북의 코드벡터 수만큼의 블록 정합을 행하여야 한다.

Ⅲ. 움직임 예측과 신경회로망을 이용한 움직임 추정

1. KSFM을 이용한 움직임 벡터의 벡터 양자화

FLOWER GARDEN 영상에 대한 움직임 벡터의 공간적인 분포를 그림 2에 나타내었다. 그림 2에서와 같이 움직임 벡터는 특정한 위치에 집중하여 분포함을 확인할 수 있고, 이를 이용하여 움직임 벡터를 벡터 양자화하고 코드벡터에 대하여서만 탐색을 행함으로 효율적인 움직임 추정이 가능하다.

제안한 방법에서는 KSFM 신경망[7]을 이용하여 움직임 벡터의 코드북을 설계하였다. LBG 방법의 VQ 코드북 설계 방법[5][6]은 큰 메모리가 필요하며, 수렴 속도가 느리고, 초기 코드북의 적절한 선택이 어려운 단점이 있다. 이에 반해 KSFM 신경망 코드북 설계 방법은, 학습 전과정에서 벡터들을 저장하지 않으므로 메모리 사용 효율이 높고, 속도가 빠르며, 최종 코드북이 초기화의 영향을 거의 받지 않는다. 또한 코드 벡터들이 상호 병렬적으로 학습되어 2차원 형태의 코드북을 구성하므로 움직임 벡터의 탐색에 적합하다.

그림 3에는 KSFM 신경망의 구조를 나타내었는데, 입력층(input layer)과 2차원 출력층(output layer)으로 구성되어 있다. 학습 과정은 먼저 N개의 입력으로부터 M개의 출력 신경 벡터, 즉 대표 벡

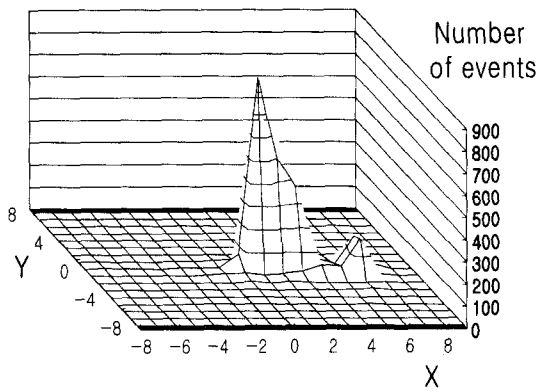


그림 2. FLOWER GARDEN 영상에 대한 움직임 벡터의 공간적인 분포

터 사이의 연결강도 $W_i(0)$, $i=1,2,\cdots,M$을 초기화시키고, 입력층에서 움직임 벡터들을 입력 벡터로 제시한다. 입력 벡터와 모든 신경 벡터들간의 거리 계산을 하여 최소 거리에 있는 출력 신경 벡터를 승리 벡터로 결정하고, 이 때 2차원적으로 연결된 출력층의 벡터들이 병렬로 상호 작용하여, 하나의 입력 벡터에 대해 승리한 신경 벡터와 그 이웃의 출력 벡터까지 입력 벡터쪽으로 갱신된다. t 시간에서 입력 벡터 X에 대한 승리벡터가 j 번째 신경 벡터라면, $t+1$ 시간에서 모든 출력 벡터들은

$$W_i(t+1) = W_i(t) + \varepsilon(t)N(j,t)(X - W_i(t)) \quad (6)$$

와 같이 갱신된다. 학습율을 나타내는 함수 $\varepsilon(t)$와, 승리한 j신경 벡터 부근의 갱신되는 이웃 신경 벡터의 범위를 결정하는 함수 $N(j,t)$는

$$\begin{aligned} \varepsilon(t) &= 0.9\,(1-\frac{t}{T}) \\ N(j,t) &= \exp[-\frac{\|i-j\|^2}{2\times 0.1^{t/T}}] \end{aligned} \quad (7)$$

와 같다. 여기서, t는 현재의 입력 반복수이고, T는 총 입력 반복수이며, $\|i-j\|$은 i 신경 단위와 최소 거리로 판정된 j 신경 단위와의 2차원 평면상의 거리를 나타낸다. $\varepsilon(t)$는 학습율로서 학습과정이 경과함에 따라 '0'으로 감소하는 함수이고, $N(j,t)$는 승리한 j 신경 벡터 부근의 갱신되는 이웃 신경 벡터의 범위를 결정하는 함수로 학습 과정이 경과함에 따라, 또는 먼 거리에 있는 신경 벡터일수록 감소하는 함수이다.

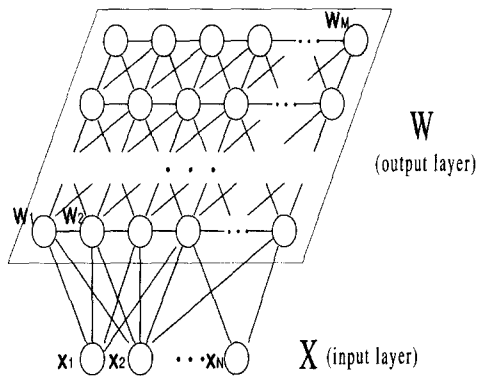


그림 3. KSFM 신경망의 구조

2. 인접블록의 움직임 벡터를 이용한 움직임 벡터 예측

인접한 블록의 움직임 벡터들은 높은 공간적 상관성을 가지며, 이를 이용하여 움직임 추정 과정에서 현재 블록의 움직임 벡터를 예측할 수 있다.

본 논문에서 현재 블록의 움직임 벡터를 예측하기 위하여 사용된 3개의 인접 블록들을 그림 4에 나타내었다.

$$G = [\,G_1,\ G_2,\ G_3\,]^T = [1/3, 1/3, 1/3]^T \quad (8)$$

$$\begin{aligned} P &= [\,P_x,\ P_y\,]^T \\ &= [\,\sum_{k=1}^{3} G_k B_{xk},\ \sum_{k=1}^{3} G_k B_{yk}\,]^T \end{aligned} \quad (9)$$

3개 인접 블록의 움직임 벡터에 식 (8)의 동일한 가중치를 각각 곱하여, 식 (9)와 같이 선형 조합으로 평균값을 구해서, 현재 블록의 예측 움직임 벡터로 사용하였다. 이때, B는 인접 블록의 움직임 벡터이고, P는 현재 블록에 대한 움직임 벡터의 예측값을 나타낸다.

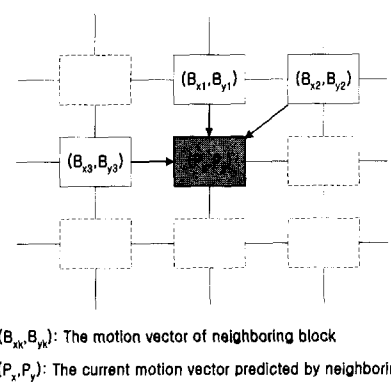


그림 4. 현재 블록의 움직임 벡터 예측

움직임 벡터를 VQ할 때, 즉 코드북에서 코드벡터를 탐색할 때, 예측된 현재 블록의 움직임 벡터와 가장 거리가 가까운 코드벡터를 초기 코드벡터로 사용한다. 초기 코드벡터에 대하여 BMA를 행하여 *SAD*(sum of absolute difference)를 문턱치 (*TH*)와 비교한다. 만약 *SAD*가 문턱치보다 작으면 탐색을 중단하고, 초기 코드벡터를 현재 블록의 움직임 벡터로 결정한다. 만약 *SAD*가 문턱치보다 크면 점진적으로 움직임 벡터를 구한다. 인접 블록으로부터 움직임을 예측함으로써, 초기 코드벡터에 대한 별도의 부가 정보는 필요 없다.

3. 점진적인 코드북 탐색

제안한 방법에서는, 앞 절에서와 같이 현재 블록의 예측 움직임 벡터 F 와 가장 가까운 거리에 있는 코드벡터를 코드북 평면에서의 초기 탐색점으로 하여 *SAD*를 구한다. 만약 *SAD*가 문턱치보다 작으면 탐색을 중단하고, 초기 코드벡터를 현재 블록의 움직임 벡터로 결정한다. 만약 *SAD*가 문턱치보다 크면 점진적으로 움직임 벡터를 구한다.

제안한 방법을 단계별로 설명하면 다음과 같다.

Step 1 ; 식 (9)을 사용해 초기 예측 움직임 벡터 F 를 구한다. F와 가장 가까운 코드벡터를 초기 탐색점 (MV_1)으로 두고 BMA를 행한 후, $SAD(MV_1) \le TH$이면 MV_1을 현재 블록의 움직임 벡터로 결정한다. 만약 $SAD(MV_1) > TH$ 이면, ***Step 2***를 행한다.

Step 2 ; 2차원 코드북 내에서 MV_1을 중심으로 3×3 탐색창, 즉 주위의 8개의 코드벡터에 대하여 BMA를 행하여 최소 *SAD*의 코드벡터 MV_2를 구한다. $SAD(MV_2) \le TH$ 이면, MV_2을 현재 블록의 움직임 벡터로 결정한다. 만약 $SAD(MV_2) > TH$ 이면, ***Step 3***를 행한다.

Step 3 ; 나머지 코드벡터들 모두에 대해서 BMA를 행하여, 최소 *SAD*의 코드벡터 MV_3를 구하고, $SAD(MV_1)$, $SAD(MV_2)$ 및 $SAD(MV_3)$ 중 가장 작은 값을 현재 블록의 움직임 벡터로 결정한다.

Ⅳ. 실험 결과 및 고찰

제안한 방법의 성능을 평가하기 위하여 FSBMA 방법, TSS 방법[2], 및 Lee 등의 방법[4]과 비교하여 모의 실험하였다. 실험 영상으로는 352×240 크기의 FLOWER GARDEN과 MOBILE 각각 70프레임씩 사용하였다. 블록의 크기는 8×8로 하였고, 탐색 영역은 -7~7로 하였다. 성능을 비교하기 위하여 화질의 척도로는 PSNR을 사용하였다.

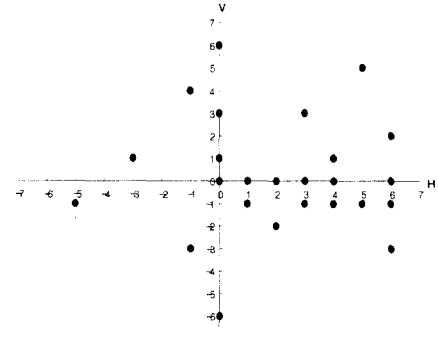
$$PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} [\, s_t(x,y) - \bar{s}_t(x,y)]^2} \tag{10}$$

여기서, $X \times Y$는 영상 한 프레임의 크기이고, $s_t(x,y)$는 t 번째 프레임의 원 영상의 화소값이고, $\bar{s}_t(x,y)$는 움직임 보상된 영상의 화소값이다. 블록 정합 척도로는 *SAD*를 사용하였다.

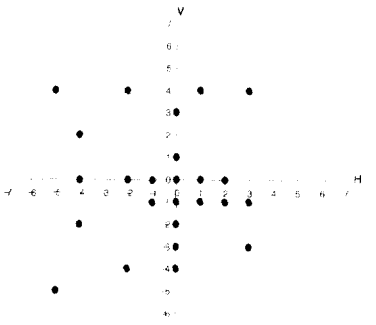
$$SAD(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} | \; S_t(m,n) - S_{t-1}(m+i, n+j) \; | \tag{11}$$

여기서, 블록의 크기는 $M \times N$ 이고, (i,j)는 이전 영상의 탐색 영역내의 탐색점 좌표를 나타낸다. 그리고 계산량의 척도로는 블록당 블록 정합 횟수를 사용하였다.

70 프레임의 FLOWER GARDEN 영상에 대해서 FSBMA으로 최적의 움직임 벡터를 찾았을 때, 원 영상 블록과 보상 블록과의 절대치 오차, 즉 블록 *SAD*의 평균이 486.59 이고, 표준 편차는 약 423 이고, MOBILE 영상 70 프레임에 대해서는, 블록 *SAD*의 평균은 552.22 이고, 표준 편차는 약 427 이다. 그러므로 제안한 방법에서의 문턱치 (*TH*)는, FSBMA로 수행한 블록 *SAD*의 평균 이하의 값을 균일하게 분포시켜, 200, 300, 400, 500 및 600으로 두고 실험하였다.



(a)



(b)

그림 5. (a) FLOWER GARDEN 및 (b) MOBILE 영상에 대한 2차원 공간상에서의 코드북

본 논문에서는 동영상의 초기 두 프레임의 움직임 벡터들을 훈련 벡터로 사용하여 25개의 코드벡터를 가지는 코드북을 설계하였다. 코드북의 크기를 25로 둔 것은 TSS 방법과 성능 비교를 위해서다.

그림 5에서 FLOWER GARDEN과 MOBILE 영상에 대해 구한 코드북을 2차원 공간상에 도시하였으며, 여기서 각 코드벡터는 움직임 벡터이다.

(a)

(b)

(c)

(d)

그림 6. (a) FLOWER GARDEN 원영상과, (b) FSBMA, (c) TSS, 및 (d) 문턱치 300의 제안한 방법으로 구한 움직임 보상된 차 영상

그림 6에는 기존의 방법과 제안한 방법으로 구한 움직임 보상된 차영상을 나타내었다. 제안한 방법에 대한 차영상이 FSBMA의 차영상과 거의 비슷하게 나타나며, TSS방법[2]의 차영상보다는 훨씬 우수함을 확인할 수 있다.

제안한 방법의 문턱치 (TH)를 변화시켰을 때, PSNR과 계산량을 나타내는 블록당 블록 정합 횟수를 표 Ⅰ, Ⅱ에 나타내었다.

표 Ⅰ에서와 같이, 제안한 방법은 FSBMA보다 FLOWER GARDEN인 경우는 0.26~0.37 dB정도, MOBILE인 경우는 0.11~0.20 dB정도 낮아지지만, TSS보다는 FLOWER GARDEN인 경우는 2.69~2.80 dB정도, MOBILE인 경우는 1.09~1.18 dB정도로 큰 증가가 있음을 알 수 있었다. 또한 Lee 등의 방법[4] 에 비해서도 문턱치의 변화에 따라 차이는 있지만, 우수한 PSNR을 나타낸다.

그리고 표 Ⅱ에서는 계산량의 대부분을 차지하는 블록당 블록 정합 횟수를 측정하였는데, 제안한 방법이 FSBMA보다 90~95 % 정도 감소했으며, TSS 방법[2] 및 Lee 등의 방법[4]보다 23.9~64.2 % 정도 감소했다.

또한, FSBMA 및 TSS 방법의 움직임 벡터 전송 비트수는 블록당 8 비트가 필요한데 비해, 제안한 방법에서는 5 비트가 필요하므로 블록당 3비트 만큼의 전송 비트수를 줄일 수 있다. 제안한 방법에서는 TSS 방법[2]과의 성능 비교를 위해 코드북의 크기를 25로 하였는 데, 만약 5비트의 어드레스(address)에 해당하는 32 크기의 코드북을 사용할 때는 더 나은 화질을 얻을 수 있다.

제안한 방법에서는 TSS 방법[2]에 비해서 부호기에서의 VQ 코드북 설계시의 계산량과 메모리가 필요하다. 그러나 움직임 추정시에는 TSS 방법[2]보다 적은 계산량으로 더 정확한 결과를 얻을 수 있다.

또한 KSFM 신경망을 이용하여 움직임 벡터의 코드북을 설계하는 제안한 방법과 LBG를 이용하는 Lee 등의 방법[4]의 코드북 생성시간을 비교해 보면, 전체 오차가 수렴할때까지 훈련(training)을 반복하는 Lee 의 방법[4]이 제안한 방법보다 5배 이상 걸린다.

이상의 결과로부터, 움직임 예측과 KSFM 신경망을 이용한 제안한 고속 움직임 추정 방법이 기존의 방법들보다 비트수와 계산량은 더 줄이면서도, FSBMA과 비슷한 수준의 PSNR을 얻을 수 있음을 확인할 수 있었다.

제안한 방법에서는 동영상의 초기 프레임으로 움직임 벡터의 코드북을 사용하기 때문에, 시간적인 상관성이 큰 동영상 신호에서는 효율적이다. 그러나 장면 전환 및 움직임이 크고 불규칙한 영상에 대해서는 움직임 벡터 코드북의 주기적인 갱신이 필요할 것이다.

표 Ⅰ. 각 방법의 PSNR[dB] 비교

Sequence		FLOWER GARDEN	MOBILE
FSBMA		25.31	23.96
TSS[2]		22.25	22.67
Lee's[4]		24.96	23.76
Proposed	TH=200	25.05	23.85
	TH=300	25.04	23.84
	TH=400	25.02	23.82
	TH=500	24.98	23.79
	TH=600	24.94	23.76

표 Ⅱ. 각 방법의 블록당 블록 정합 횟수 비교

Sequence		FLOWER GARDEN	MOBILE
FSBMA		225	225
TSS[2]		25	25
Lee's[4]		25	25
Proposed	TH=200	17.49	19.03
	TH=300	15.16	16.22
	TH=400	12.78	13.64
	TH=500	10.72	11.54
	TH=600	8.97	9.73

V. 결 론

본 논문에서는 움직임 벡터의 예측과 신경 회로망을 이용한 고속 움직임 추정 방법을 제안하였다. 동영상의 초기 프레임들의 움직임 벡터를 입력 벡터로 하여, 움직임 벡터의 코드북을 설계하였다. 이때 코드북 생성 시간이 빠르고, 인접 코드벡터간에 연관성을 가지고 학습하는 KSFM 신경 회로망을 이용하였으며, 최종 코드북에서는 코드벡터가 2차원 형태로 상관성을 지닌다. 제안한 방법에서의 움직임 추정은, 코드북의 전체 코드벡터에 대해서 탐색을 행하지 않고, 인접 블록의 움직임 벡터들로부터 현재 블록의 움직임 벡터를 예측하고, 이를 중심으로 코드북상에서 점진적인 탐색을 행하였다.

모의 실험 결과, 제안한 방법이 기존의 방법들보다 비트수와 계산량은 더 줄이면서도, 움직임 추정 방법 중 최적인 FSBMA에 근접한 PSNR을 얻을 수 있음을 확인할 수 있었다.

참 고 문 헌

[1] H. Gharavi, Mike Mills, "Block matching Motion Estimation Algorithms-New Results," *IEEE Trans. on Circuits and System*, vol. 37, No. 5, pp. 649-651, 1990.

[2] T. Koga, K. Iinuma, A. Hirano, Y. Iiyima and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing," *Proc. NTC. '81*, pp. G5.3.1-G5.3.5, New Orleans, LA, Dec. 1981.

[3] L. Luo, C. Zou, X. Gaom, "A New Prediction Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Trans. On Consumer Electronics*, vol. 43, No. 1, pp. 56-61, 1997.

[4] Y. Y. Lee and John W. Woods, "Motion Vector Quantization for Video Coding," *IEEE Trans. on Image Proc.* vol 4. No. 3, Mar. 1995.

[5] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Comm.*, vol. COM-28, pp. 84-95, Jan. 1980.

[6] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*: Kluwer Academic Publishers, 1992.

[7] N. M. Nasrabadi and Y. Feng, "Vector Quantization of Images Based upon the Kohonen Self-Organizing Feature Maps," *IEEE Int'l. Conf. Neural Network*, San Diego, CA, vol. 1, pp. 101-108, 1988.

최 정 현(Jung-Hyun Choi) 정회원
통신학회논문지 제24권 제1B호 참조

이 경 환(Kyeong-Hwan Lee) 정회원
통신학회논문지 제24권 제1B호 참조

이 법 기(Bub-Ki Lee) 정회원
통신학회논문지 제24권 제1B호 참조

정 원 식(Won-Sik Cheong) 정회원
통신학회논문지 제24권 제1B호 참조

김 경 규(Kyoung-Kyoo Kim) 정회원
통신학회논문지 제24권 제1B호 참조

김 덕 규(Duk-Gyoo Kim) 정회원
통신학회논문지 제24권 제1B호 참조