論文99-36S-12-11

예측 비디오 코딩을 위한 통합 움직임 보상 알고리즘

(Integration of Motion Compensation Algorithm for Predictive Video Coding)

陰 湜 珉 *, 朴 根 秀 **, 宋 文 豪 *

(Ho-Min Eum, Geun-Soo Park and S. Moon-Ho Song)

요 약

많은 경우의 예측 비디오 압축 표준에서는, BMA에 의해 매크로 블록당 하나의 움직임 벡터가 계산되는 방식인 BMC방식이 널리 사용되고 있다. 그러나 BMC에 의해 예측된 움직임 벡터 필드는 블록당 하나의 움직임 벡터를 사용하기 때문에 불연속적이며, 불연속적인 움직임 벡터 필드로 인해 블록화 현상을 나타낸다. 따라서 이를 제거하는 효과적인 방법은 움직임 벡터 필드를 평활화(smoothing)하는 방법일 것이다.

최적 평활화 과정은 비디오 시퀀스의 움직임 종류에 따라 다를 것이다. 본 논문에서는 움직임 벡터를 평활화하는 몇 개의 방법들을 고려할 것이다. 어떠한 방법이든 BMA로 구한 움직임 벡터는 더 이상 최적화된 움직임 벡터가 아닐 것이므로, DFD(displaced frame difference)의 놈(norm)을 최소화하는 최적 움직임 벡터를 찾아야 한다. 본 논문에서는 conjugate gradient 알고리즘을 사용하여 DFD의 놈을 최소화하는 최적움직임 벡터를 찾는 통합 알고리즘을 제안한다. 이 통합 알고리즘은 ATMC(affine transform based motion compensation), BTMC(bilinear transform based motion compensation), 그리고 본 논문에서 제안하는 FMC(filtered motion compensation)의 세가지 방식에 대하여 적용되고 BMC에 대비해서 평가되어 졌다.

Abstract

In a number of predictive video compression standards, the motion is compensated by the block-based motion compensation (BMC). The effective motion field used for the prediction by the BMC is obviously discontinuous since one motion vector is used for the entire macro-block. The usage of discontinuous motion field for the prediction causes the blocky artifacts and one obvious approach for eliminating such artifacts is to use a smoothed motion field.

The optimal procedure will depend on the type of motion within the video. In this paper, several procedures for the motion vectors are considered. For any interpolation or approaches, however, the motion vectors as provided by the block matching algorithm(BMA) are no longer optimal. The optimum motion vectors(still one per macro-block) must minimize the of the displaced frame difference (DFD). We propose a unified algorithm that computes the optimum motion vectors to minimize the of the DFD using a conjugate gradient search. The proposed algorithm has been implemented and tested for the affine transformation based motion compensation (ATMC), the bilinear transformation based motion compensation (BTMC) and our own filtered motion compensation (FMC). The performance of these different approaches will be compared against the BMC.

* 正會員, 高麗大學校 電波工學科
(Radio sciences & engineering, Korea University)
* 正會員, 三星電子 情報通信總括 네트웍 事業部
(Network systems division, Information and communication business. Samsung electronics)
接受日字:1999年7月29日, 수정완료일:1999年11月30日

Ⅰ. 서 론

비디오 프레임간의 예측 부호화(predictive interframe image coding)는 영상 시퀀스에서 프레임 사이의 시간적 중복성(temporal redundancy)을 제거함으로써 효과적인 데이터 압축을 가능하게 한다. 이러한 예측 부호화에서 가장 널리 쓰이는 블록 기반의 움직임 보상(block based motion compensation : BMC)은 효율적인 정보의 압축이 가능하다는 우수한 성능을 이유로 MPEG, H.261, H.263 등의 영상압축 표준안에서 중요하게 사용되고 있고, 다양한 연구들을 통해서 그 성능의 우수성이 분석되었으며, 성능의 개선 및 보완이 계속 제안되고 있다[1-3].

BMC에서는 용이한 구현 및 계산으로 널리 알려진 block matching algorithm(BMA)을 사용하여 움직임 벡터(motion vector)를 찾는다. 즉, 현재 프레임을 정사각형의 블록들로 나누어서 각각 블록들과의 상관관계(correlation)가 높은 블록을 전 프레임으로부터 찾는 것이다. 이렇게 BMA에 의해서 찾아진 움직임 벡터에 대응되는 전 프레임의 블록을 이용하여 예측영상이 합성될 때, displaced frame difference(DFD)의 놈(norm)이 최소화된다. 이 과정에서 찾아진 현재 프레임에서의 블록 단위의 움직임 벡터는 각 블록에 대해서 전 프레임에서 가장 잘 대응되는 블록의 상대적 위치를 의미한다. 부호화기(encoder)에서는 이러한 움직임 벡터와 전 프레임을 이용하여 예측 프레임을 합성하고 현재 프레임의 원 영상(original image)과의 차 영상(difference image)을 움직임 벡터 정보와 함께 복호기(decoder)로 전송한다. 전송된 차 영상과 움직임 벡터로 이루어진 데이터는 원래 보내져야 할 현재 프레임의 데이터보다 정보량이 훨씬 적으므로 이로부터 정보의 압축 효과를 얻을 수 있다. 복호기에서는 전송받은 차 영상 및 움직임 벡터 정보와 전 프레임을 가지고 현재 프레임을 복원할 수 있다.

그러나 BMC에서는 블록 단위의 움직임 벡터 하나가 블록 내의 모든 화소의 움직임을 동일하게 모델링하므로 효과적인 영상 압축을 얻을 수 있는 장점이 있는 반면에, 복호기에서 복원된 영상에서는 인접한 블록의 경계부분에서 블록단위의 움직임 크기가 다른 만큼의 불연속 현상이 관찰될 수 있다. 이 불연속 현상을 블록화 현상(blocky artifact)이라고 일컫는다. 이 현상은 움직임이 적은 영상에서는 크게 문제가 되지 않을 수 있으나 움직임이 큰 영상에서는 심각한 화질 열화를 초래할 수 있다. 따라서, 이 블록화 현상의 제거는 영상의 화질을 향상시키는 면에서 매우 중요하다.

BMC에서는 영상의 움직임을 부분적으로 일정(piece-wise constant)하고, 블록의 경계에서 불연속적인 벡터 필드(vector field)로 모델링한다. 그러나 실제 영상 시퀀스에서는 회전, 뒤틀림, 확대/축소 등의 복잡한 움직임을 포함하고 있으므로 BMC는 이러한 영상 시퀀스를 다루는데 문제가 있다. 이러한 BMC의 단점을 극복하고 실제 영상 시퀀스에서의 움직임 벡터 필드를 효과적으로 근사하기 위하여 여러 가지 움직임 모델의 새로운 접근 방법들이 제안되었다[4-12]. 이러한 방법들은 크게 두 가지의 범주로 나눌 수 있는데, 하나는 움직임 벡터들을 조작하는 것이고[4-8,12], 다른 하나는 예측영상이 합성될 때 영상을 평활화(smoothing)하는 것이다[9-11]. 본 논문에서는 전자인 움직임 벡터를 조작하는 접근 방법을 중심으로 살펴볼 것이다.

블록화 현상은 위에서 언급하였듯이 블록단위로 하나의 일정한 움직임 벡터를 사용하여 예측 영상을 합성하는 데에서 기인한다. 물론 블록화 현상은 양자화 행렬(quantization matrix)과도 관계가 있지만, 본 논문에서는 양자화로 야기되는 블록화 현상은 고려하지 않았다. 본 논문에서는 예측 영상을 합성하기 전에 움직임 벡터를 보간(interpolation)하는 방법을 고려한다. 따라서, 공간적으로 평활화 된 움직임 벡터들은 블록화 현상을 완전히 제거할 것이다. 그러나 이 방법에서 BMA에 의한 움직임 벡터는 더 이상 최적화 된 움직임 벡터가 아니므로 최적화 된 움직임 벡터를 찾는 과정이 필요하다.

본 논문에서는 움직임 벡터를 평활화하는 몇 가지 방법들에 대하여 고려하였다. 어떤 방법에서든지 BMA에 의한 움직임 벡터는 더 이상 최적화 된 움직임 벡터가 아니다. 매크로블록 당 하나의 최적화 된 움직임 벡터는 DFD의 놈을 최소화하는 것이어야 한다. 이러한 관점에서 본 논문에서는 conjugate gradient method[14]를 사용하여, DFD의 놈을 최소화하는 최적화 된 움직임 벡터를 찾는 통합 알고리즘을 제안한다. 그리고 제안된 알고리즘은 affine 변환 움직임 보상(affine transform based motion compensation : ATMC), bilinear 변환 움직임 보상(bilinear transform based motion compensation : BTMC), 그리고 공간적으로 평

활한 움직임 벡터를 찾는 부분에서 band-limited 보간 필터가 사용되는 filtered motion compensation(FMC), 이 세 가지 방법에 대하여 구현하고 테스트하였다. 각 방법들의 성능은 BMC와 비교되어졌다.

본 논문은 다음과 같이 구성되어 있다. 먼저 II장에서는 ATMC, BTMC 그리고 FMC를 실행하는 통합된 최적 움직임 벡터 추정 알고리즘을 정리할 것이다. 그리고 III장에서는ATMC, BTMC 그리고 FMC의 구현 방법을 설명하고 ATMC와 BTMC도 FMC의 특별한 경우임을 보인다. 알고리즘의 성능은 IV장에서 평가되어질 것이다. 시뮬레이션 결과에서 위 세 가지 방법에 한해서 블록화 현상이 완벽하게 제거됨을 알 수 있다. 또한 BMC에 비교해서 더 높은 PSNR을 보임을 알 수 있다. 마지막으로 V장에서 결론을 맺을 것이다.

II. 통합 알고리즘

예측 부호화를 기반으로 하는 기존의 동영상 정보 압축 방식에서는 예측 영상을 블록단위로 합성함으로 인하여 블록화 현상을 유발시키는 단점을 갖는다. 이에 블록화 현상을 제거하는 근본적인 해결 방안으로서, 본 논문에서는 ATMC, BTMC, 그리고 FMC방식을 수행하는 통합 알고리즘을 제안한다. 이 세 가지 방식에 적용되는 통합 알고리즘은 블록화 현상에 근본적으로 접근하여 그 원인을 제거하고 블록화 현상을 완벽하게 제거한다.

기존의 동영상 정보 압축 부호화기에서는 프레임간 정보압축을 위해서 BMA를 이용한 움직임 벡터 추정기(motion vector estimator)와 BMC를 이용한 움직임 보상기(motion compensator)가 주로 사용된다. 그러나 제안된 본 통합 알고리즘 에서는 기존의 방식에서 조금 변형된 움직임 벡터 추정 방식과 보상 방식을 취함으로써, 블록화 현상을 제거하고 예측성능을 향상시킨다.

1. 최적 움직임 벡터 추정 알고리즘

본 알고리즘에서는 다양한 움직임 모델을 수용하기 위해서 움직임 추정기와 보상기에서는 반복적인 움직임 벡터 최적화 작업을 수행한다. 이 과정이 통합알고리즘에서 가장 핵심이 되는 알고리즘으로서 부호화기에서 수행된다. 수식전개에 의해서 살펴보면 다음과 같다.

본 알고리즘의 최종목표는 블록화 현상을 제거하고 궁극적으로 Displaced Frame Difference(DFD)의 놈(norm)을 최소화함을 목표로 한다.

최적화 된 움직임 벡터는 DFD의 놈, 예를 들면 Estimated Mean Square Error (EMSE)를 최소화한다. EMSE는 불필요한 상수를 고려하지 않고 다음과 같이 쓰여질 수 있다.

$$EMSE = \sum_{r \in R} |DFD(r, u(r))|^2 \tag{1}$$

위 식(1)에서 R은 영상의 범위, $r=(x, y)$는 화소 좌표, $u(r)$은 위치 r에서의 움직임 벡터를 의미한다. DFD는 다음과 같이 정의된다.

$$DFD(r, u(r)) = I_n(r) - I_{n-1}(r + u(r)) \tag{2}$$

위 식(2)에서 $I_n(r)$과 $I_{n-1}(r+u(r))$은 각각 현재(n번째) 프레임과 움직임 벡터에 의해 이동된 전(n-1 번째) 프레임을 의미한다. 식에서 알 수 있듯이, DFD는 현재 프레임과 이동된 전 프레임사이의 차이다. 그리고 예측된 프레임, $I_{n-1}(r+u(r))$은 전 프레임, $I_{n-1}(r)$과 해당하는 움직임 벡터, $u(r)$를 이용해서 합성된 영상이다.

EMSE의 최적화는 비선형문제이며 선형화에 의해서 수치해가 얻어질 수 있다. 연속적 선형화를 이용하기 위하여 최적화 된 움직임 벡터, $u(r)$에 근접한 움직임 벡터, $u^*(r)$를 가정한다. 이 오차를 $\delta = u(r) - u^*(r)$로 정의하면 DFD가 $u(r)$에 대해서 비선형이므로 식(2)는 $u^*(r)$을 중심으로 한 테일러 전개에 의해서 다음과 같이 전개된다.

$$\begin{aligned} DFD(r, u(r)) &= I_n(r) - I_{n-1}(r + u^*(r)) + \nabla I_{n-1}(r + u^*(r))^T \delta + h.o.t. \\ &= DFD(r, u^*(r)) + \nabla I_{n-1}(r + u^*(r))^T \delta + h.o.t. \\ &\approx DFD(r, u^*(r)) + \nabla I_{n-1}(r + u^*(r))^T \delta \end{aligned} \tag{3}$$

위 식(3)에서 ∇는 gradient연산자, $[\frac{\partial}{\partial x} \ \frac{\partial}{\partial y}]^T$를 의미하고 $h.o.t.$는 δ에 대한 고차항(higher order term)으로서 매우 작은 값이므로 근사화에서 생략될 수 있다. 이를 이용하여 식(1)은 다음과 같이 쓰여진다.

$$EMSE \approx \sum_{r \in R} |DFD(r, u^*(r)) + \nabla I_{n-1}(r + u^*(r))^T \delta|^2 \tag{4}$$

위의 EMSE를 이산화(discretize) 하고 순서대로(lexicographical order) 정렬하면 DFD행렬의 벡터 형

내, b와 개선 벡터, δ_p는 다음과 같이 표현된다.

$$b(u)=\begin{vmatrix} DFD((1,1),u(1,1)) \\ DFD((1,2),u(1,2)) \\ \vdots \\ DFD((N,M),u(N,M)) \end{vmatrix},\quad \delta_p=\begin{vmatrix} \delta_h(1,1) \\ \delta_h(1,2) \\ \vdots \\ \delta_h(N,M) \\ \delta_v(1,1) \\ \vdots \\ \delta_v(N,M) \end{vmatrix} \tag{5}$$

위 식(5)에서 N과 M은 프레임의 화소 단위 크기로서 N은 세로방향, M은 가로방향의 크기이다. 그리고 δ_p는 움직임 벡터의 두 성분(δ_h, δ_v)으로 구성되어 있으며 아래 첨자 (h,v)는 각각 (가로, 세로) 방향을 의미한다. 예를 들어, Common Intermediate Format (CIF) 프레임은 (M,N)=(288, 352)이고, 가로로 22개, 세로로 18개의 매크로 블록으로 나누어진다. 식(5)의 정의에 의해서 EMSE는 다음과 같이 이산화(discretize) 된다.

$$EMSE_D=\|\boldsymbol{A}(u)\ \delta_p+b(u)\|^2 \tag{6}$$

위 식(6)에서 $\nabla I_{n-1}(r+u^*(r))^T$의 이산화 된 표현, $\boldsymbol{A}(u)$와 DFD의 이산화 된 표현, b는 두 항 모두 식(6)에 표시된 바와 같이 $u(r)$의 함수이고, $\boldsymbol{A}(u)$는 사이즈가 $(MN\times 2MN)$이며, EMSE의 테일러 전개식에서 첫 번째 항에서 비롯된 것이다. δ_p는 영상의 모든 화소에서 정의되며, 듬성듬성한(sparse) 움직임 벡터 필드(예를 들어 매크로 블록단위의 계)의 선형 또는 비선형 변환으로 모델링 된 것이다. 또한 매크로블록단위 움직임 벡터의 사전적 순서(lexicographical order)로 정리된 항을 δ_b로 나타내면 다음과 같이 두 움직임 벡터 필드의 관계를 설정할 수 있다.

$$\delta_p=\boldsymbol{H}(\delta_b) \tag{7}$$

다시 말하자면, 식(7)의 $\boldsymbol{H}$는 듬성듬성하게 매크로 블록단위로 샘플링 된 움직임 벡터 필드를 화소 단위로 샘플링 된 움직임 벡터 필드로 대응시키는 선형 또는 비선형의 어떤 변환을 의미한다. 식(5)와 (7)에서의 정의를 사용해서 이산화 된 EMSE는 다음과 같이 쓰여진다.

$$EMSE_D=\|\boldsymbol{A}(u)\boldsymbol{H}(\delta_b)+b(u)\|^2 \tag{8}$$

모든 δ_b와 $\boldsymbol{H}$에 대한 위 식(8)의 최적화는 EMSE의 감소로 귀결된다. δ_b와 $\boldsymbol{H}$에 대해서 동시에 최적화 하는 문제는 해결하기 어려운 비선형 문제이므로 먼저 $\boldsymbol{H}$를 고정시킴으로써 문제해결을 용이하게 만들 수 있다. 예를 들면, 움직임 벡터 필드를 band-limited되어 있다고 가정할 경우, $\boldsymbol{H}$는 band-limited 보간 필터를 취하는 변환을 사용할 수 있겠다. 그러나 $\boldsymbol{H}$를 고정시켜 놓아도 EMSE의 최소화 해는 비 선형 방정식 문제의 해이므로 δ_b에 대한 EMSE의 최적화는 여전히 어려운 문제이다. 그렇지만 $u(r)$에 대해 근접한 초기치를 사용할 수 있다면 수치해는 다음의 반복적인 방식에 의해서 구해질 수 있다.

$$\begin{aligned} \boldsymbol{A}(u^{(k)})\boldsymbol{H}(\delta_b^{(k+1)}) &= -b(u^{(k)}) \\ u^{(k+1)} &= u^{(k)}+\delta_b^{(k+1)} \end{aligned} \tag{9}$$

위 알고리즘(식(9))을 해가 수렴할 때까지 반복한다. 초기 벡터, $u^{(0)}(r)$를 얻기 위해서는 기존의 Block matching algorithm(BMA)를 사용한다. 이 때, BMA로 구해진 벡터가 최적의 벡터에 가까운 값이고 반복을 통해서 빠르게 수렴할 수 있다고 가정한다. 실제로, CIF 영상에서 네 번 또는 그 이하에서 수렴되었다고 할 수 있었다.

위의 알고리즘이 ATMC(affine transform-based motion compensation), BTMC(bilinear transform-based motion compensation), FMC(filtered motion compensation)의 세 가지 방법에 관하여 적용되었다. 이 세 가지 방법들의 공통점은 보간(interpolation) 과정이 선형 변환이라는 것이다. 즉, 식(7)이 다음과 같은 행렬 연산으로 δ_b와 δ_p의 관계를 나타낼 수 있다.

$$\delta_p=\boldsymbol{H}\delta_b \tag{10}$$

따라서 식(9)와 같은 알고리즘이 이 세 가지 방법에서는 식(7)대신 식(10)을 사용하여 적용되는 것이다.

2. 속도와 정확성을 위한 알고리즘

위의 알고리즘을 수행하기 위해, 식(9)에서의 $\boldsymbol{AH}\delta_b=-b$에서 δ_b를 구하기 위해서 sparse한 행렬에서 빠른 속도와 좋은 수렴특성을 갖는 conjugate gradient 알고리즘[14]을 사용하였고, 화소 단위의 움직임 벡터로 영상을 합성할 때는 sub-pixel의 개념을 도입하였다.

1) Conjugate-Gradient 알고리즘

Conjugate gradient 알고리즘은 $\boldsymbol{Qx}=\boldsymbol{b}$ ($\boldsymbol{Q}$는 symmetric positive 행렬)에서 $\boldsymbol{x}$를 numerical method을 이용하여 적은 계산량으로 빠르게 구하기 위한 방식이다. 알고리즘은 다음과 같다.

Starting at any $\boldsymbol{x}_o \in \boldsymbol{R}^n$ define $\boldsymbol{d}_0 = -\boldsymbol{g}_0 = \boldsymbol{b} - \boldsymbol{Qx}_0$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$$

$$\alpha_k = -\frac{\boldsymbol{g}_k^T \boldsymbol{d}_k}{\boldsymbol{d}_k^T \boldsymbol{Q}\boldsymbol{d}_k}$$

$$\boldsymbol{d}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k$$

$$\beta_k = -\frac{\boldsymbol{g}_{k+1}^T \boldsymbol{Q}\boldsymbol{d}_k}{\boldsymbol{d}_k^T \boldsymbol{Q}\boldsymbol{d}_k}$$

where $\boldsymbol{g}_k = \boldsymbol{Qx}_k - \boldsymbol{b}$

식(9)에서 $\boldsymbol{AH}\delta_b = -b$에서 δ_b를 구해야 하므로, 양변에 $(\boldsymbol{AH})^*$를 곱하여 $(\boldsymbol{AH})^* \boldsymbol{AH}\delta_b = -(\boldsymbol{AH})^* b$로 만든다. 이로써 $(\boldsymbol{AH})^* \boldsymbol{AH}$와 $(\boldsymbol{AH})^* b$이 각각 위 알고리즘에서 $\boldsymbol{Q}$와 $\boldsymbol{b}$ 역할을 하게 된다.

위의 알고리즘에서 $\boldsymbol{x}_k$의 값이 수렴하게 될 때에 (본 논문의 시뮬레이션에서는 $\| \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \| \le 0.001 \| \boldsymbol{x}_k \|$ 일 때), 이 값을 우리가 구하는 δ_b의 값으로 볼 수 있다. 위의 방법의 적용으로 pseudo-inverse 방식에 비해서 정확도를 대략 90%이상으로 유지하면서 계산 시간을 30% 이내로 줄일 수 있다.

2) sub-pixel

또한 구해진 움직임 벡터는 정수의 값이 아니므로 정확한 계산을 위해서 sub-pixel의 개념을 적용하였다. 즉, 영상을 복원할 때에 움직임 벡터가 정수개의 화소(pixel) 만큼 떨어진 곳을 가리키는 것이 아니라 소수점이 있는 값을 가지게 되는 경우에, 그 움직임 벡터가 가르치는 곳의 주위의 화소들을 이용하여 보간(interpolation)을 하는 방법으로 계산의 정확성을 높일 수가 있다. MPEG 등의 표준에서도 예측 성능을 높이기 위하여 half-pixel개념을 도입하고 있다. 본 알고리즘의 구현에서는 bilinear 보간을 사용하여 sub-pixel의 개념을 도입하였다.

즉, $(r+u(r))$이 가리키는 포인트가 (x', y')라고 할 때, 그 포인트에서의 화소의 값은 다음과 같이 구해진다.

$$\begin{aligned} I_{n-1}(x', y') = {} & (1-\alpha)((1-\beta)I_{n-1}(X, Y) + \beta I_{n-1}(X+1, Y)) \\ & + \alpha((1-\beta)I_{n-1}(X, Y+1) + \beta I_{n-1}(X+1, Y+1)) \end{aligned} \tag{11}$$

여기서, (X, Y)는 (x', y')의 정수 부분이고 (α, β)는 (x', y')의 나머지 분수 부분이다.

Ⅲ. 통합 알고리즘의 적용

본 장에서는 Ⅱ장에서 제안된 통합 알고리즘을 FMC, BTMC, ATMC에 적용하여 각각의 구현 방법을 제시할 것이다. 또한, 수식을 전개하는 과정에서 BTMC, ATMC가 FMC의 특별한 경우임이 자연스럽게 드러날 것이다.

1. FMC

본 알고리즘에서 식(10)에서의 $\boldsymbol{H}$가 2-D 필터와 컨볼루션(convolution)을 수행하는 선형변환일 때 FMC (filtered motion compensation)가 된다. FMC에서 움직임 보상을 하는 방법은 다음과 같다.

움직임 추정기에서 구해지는 최적 움직임 벡터 값은 BMA와 달리 단 한번의 연산으로 구해지지 않고 Ⅱ.1절에 소개된 통합 알고리즘(식(9))에 의해 수 차례의 반복 수렴 작업으로 구해진다. 따라서, 최적치로의 빠른 수렴을 위해서 적절한 초기값을 설정해주는 것이 중요하다. 초기값으로서 가장 알맞은 값은 BMA를 이용해서 구해진 블록단위의 움직임 벡터 값이다. 위와 같은 이유로 먼저 BMA를 수행한다. 그 후, 구해진 초기 벡터를 <그림 1>과 같은 미리 제작된 2-D 필터를 이용해서 화소 단위로 전환하고 이 전환된 움직임 벡터와 전 프레임을 가지고 움직임 보상기에서 예측 프레임을 화소 단위로 합성한다. 그 다음 과정으로서 예측 프레임과 현재 프레임사이의 차 영상을 구한 후, 차 영상의 놈을 최소화하는 개선 벡터를 구하여 초기 벡터를 갱신(update)한다. 갱신된 벡터를 이용하여 움직임 보상 작업을 반복적으로 행하여 차 영상의 놈이 최소가 될 때까지 구하는 것이 Ⅱ장에서의 최적 움직임 벡터를 구하는 알고리즘이다.

이 알고리즘을 구현하는데 있어서 conjugate-gradient 알고리즘을 사용한다. 이때 행렬 $\boldsymbol{A}$를 곱하는 과정은 각 방향 차분(difference) 성분을 곱해줌으로서 구현할 수가 있고, $\boldsymbol{H}$를 곱하는 과정은 2-D 필터링을 해줌으

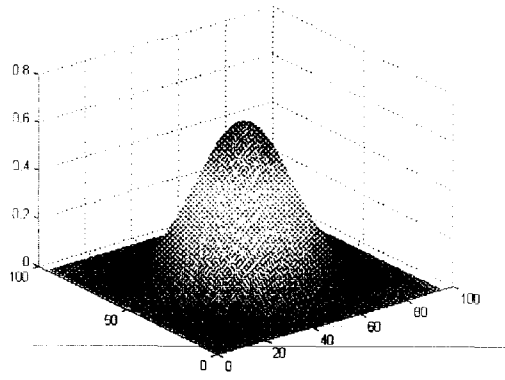


그림 1. FIR 필터 커널
Fig. 1. FIR filter kernel.

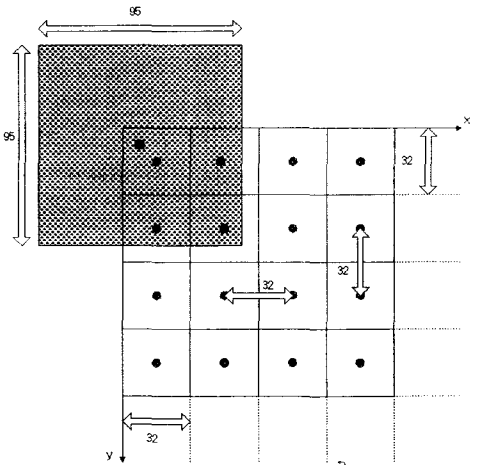


그림 2. 컨볼루션에 의한 필터링
Fig. 2. Filtering by convolution.

로서 구현할 수 있다. 즉, 매크로 블록 단위의 모션벡터가 그 매크로 블록의 가운데에 위치하고 나머지 부분에서는 0인 움직임 벡터 영상과 필터 커널(filter kernel)을 컨볼루션 함으로써 얻어질 수 있다. 이때, 16X16 크기의 매크로 블록에서 정 가운데 값이 존재하지 않으므로 사이사이에 0을 집어넣어서 한 매크로 블록의 크기를 31X31로 만든 뒤에 가운데 점인 (16, 16)의 위치에 움직임 벡터가 있다고 보고, 95X95크기의 필터와 컨볼루션을 취한 뒤에 다시 사이사이의 값을 빼서 원래의 한 매크로 블록이 16X16인 화소 단위의 움직임 벡터들을 얻는 방법을 사용한다. 이 과정이 <그림 2>에 나타나 있다. 검은 점으로 표시된 매크로 블록의 가운데에 움직임 벡터값이 있고, 색칠된 사각형은 <그림 1>과 같은 필터를 의미한다.

특히, H를 2-D LPF(Low Pass Filter)를 사용하여 band-limited 보간(interpolation)을 하는 선형 변환으로 설정하면 본 통합 알고리즘은 BLMC(band-limited motion compensation)를 수행하게 된다[8]. <그림 1>의 필터는 stop band에서 40dB의 attenuation을 갖도록 설계된 least squares FIR LPF의 point spread function이다. 블록의 경계에도 0이 삽입되므로 한 매크로 블록의 크기는 32X32라고 할 수 있다. 사각형 모양의 점이 현재 필터링 되는 점을 나타낸다. 주변 블록의 움직임 벡터가 현재 블록의 점을 보간하는데 곱해지게 된다. 따라서 필터의 사이즈가 주변 블록까지 아홉 개의 블록에 걸치는 사이즈가 되어야 하므로 블록의 사이즈는 95X95가 되는 것이다. <그림 2>에서 사각형 점의 값을 보간 하기 위해서는 필터에 걸치는 네 개의 움직임 벡터를 이용한다. 사각형필터가 벡터 영상의 가운데 부분으로 움직였을 때는 주위의 아홉 개의 움직임 벡터를 이용하면 되는 것이다.

2. BTMC

BTMC(bilinear transform based motion compensation)도 FMC와 같은 방식으로 적용할 수 가 있다. 차이점이 있다면, FMC에서는 블록의 정가운데에 움직임 벡터가 있는 반면에 BTMC에서는 움직임 벡터가 격자(grid) 단위로 있다는 점이다. BTMC에서는 각 매크로 블록의 꼭지점에 움직임 벡터가 있다고 보고 bilinear transform에서 얻어진 보간공식을 이용해서 화소 단위 움직임 벡터를 구한다고 볼 수가 있다.[6]

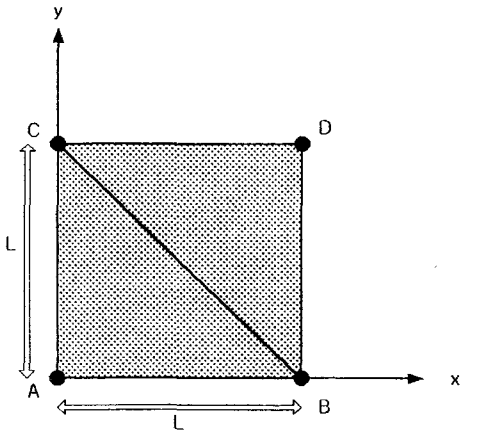


그림 3. ATMC, BTMC에서의 보간
Fig. 3. Interpolation in ATMC, BTMC.

BTMC는 <그림 3>에서 A, B, C, D 네 점의 움직임 벡터 u_A, u_B, u_C, u_D를 이용하여 화소 단위의 움직임 벡터를 구하는데 다음의 관계식을 이용한다.[6]

$$u(x,y) = (1-\frac{y}{L})[(1-\frac{x}{L})u_A+\frac{x}{L}u_B] + \frac{y}{L}[(1-\frac{x}{L})u_C+\frac{x}{L}u_D] \quad (12)$$

따라서 $h(\cdot)$(식(10)의 $\boldsymbol{H}$를 구현하기 위한 변수)를 식(13)과 같이 설정하면, BTMC 움직임 벡터 필드와의 컨볼루션을 통해 주위의 4개의 격자의 움직임 벡터에 곱해져서 그 블록 내에서의 위치에 따른 보간을 수행하게 할 수 있다.

$$h(x,y)=\begin{cases}(1+\frac{x}{L})(1+\frac{y}{L}), & -L<x,y<0\\ (1+\frac{x}{L})(1-\frac{y}{L}), & -L<x<0,\ 0<y<L\\ (1-\frac{x}{L})(1+\frac{y}{L}), & 0<x<L,\ -L<y<0\\ (1-\frac{x}{L})(1-\frac{y}{L}), & 0<x,y<L\end{cases} \quad (13)$$

위와 같이 2L X 2L 크기의 $h(\cdot)$를 만들면 <그림 4>와 같은 모양의 커널이 된다. 이 커널을 사용하여<그림 5>에서 처럼 컨볼루션을 수행하였을 때에 그 결과가 식(12)와 같게 된다. (i,j)번째 매크로 블록에서 $u_G(i,j)$가 u_A역할을 하고 $u_G(i+1,j)$, $u_G(i,j+1)$, $u_G(i+1,j+1)$이 각각 u_B, u_C, u_D가 되는 것이다. 즉, 블록 내에서의 $(x',y')=(x-iL,y-jL),0<x',y'<L$ 위치에 커널의 원점이 있을때 $u_G(i,j)$는 커널에서 $(-x',-y')$의 위치이므로 $(1-\frac{x'}{L})(1-\frac{y'}{L})$이 곱해지고, $u_G(i+1,j)$는 $(L-x',-y')$의 위치이므로 $\frac{x'}{L}(1-\frac{y'}{L})$이 곱해지고, $u_G(i,j+1)$는 $(-x',L-y')$의 위치이므로 $\frac{y'}{L}(1-\frac{x'}{L})$이 곱해지고, $u_G(i+1,j+1)$는 $(L-x',L-y')$의 위치이므로 $\frac{y'}{L}\frac{x'}{L}$이 곱해진다. 따라서 식(18)과 같은 식으로 보간된다. <그림 5>에서 색칠된 사각형 모양이 <그림 4>의 커널을 의미하며, 사각형 점의 위치를 보간하는 데에 걸치는 네 개의 격자위치의 움직임 벡터가 이용된다.

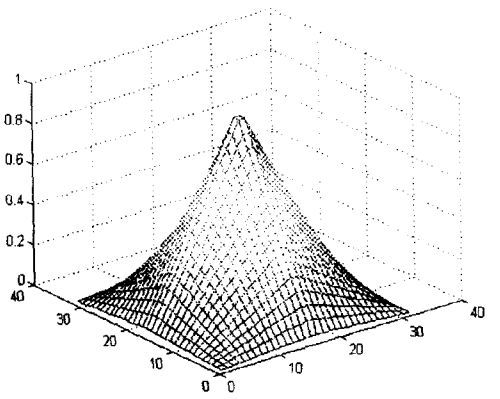


그림 4. bilinear 변환 커널
Fig. 4. Bilinear transform kernel.

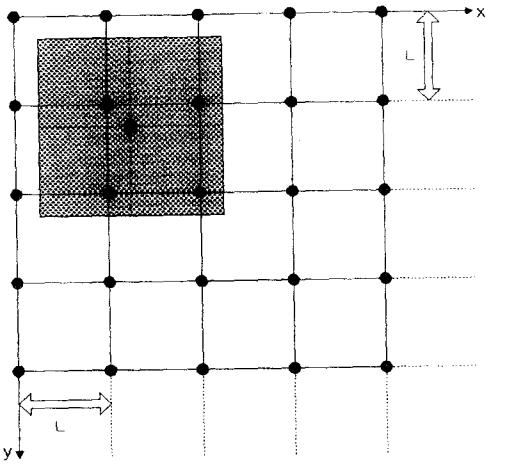


그림 5. BTMC를 수행하는 컨볼루션
Fig. 5. Convolution for BTMC.

3. ATMC

ATMC(affine transform based motion compensation)도 FMC와 같은 방법에 적용할 수 있다.

ATMC에서는 <그림 3>에서 보는 바와 같이 한 블록을 두 개의 삼각형 모양의 patch로 나누어서 화소 단위의 움직임 벡터를 구한다. 즉 <그림 3>에서 아래 삼각형내의 화소에 해당하는 움직임 벡터를 구하는데는 A, B, C점에서의 움직임 벡터 u_A, u_B, u_C 를 이용하고 위 삼각형에서는 B, C, D의 u_B , u_C, u_D 세 개를 이용하는 것이다. 이를 수식으로 나타내면 다음과 같다.[6]

$$u(x,y)=\begin{cases}(1-\frac{x}{L}-\frac{y}{L})u_A+\frac{x}{L}u_B+\frac{y}{L}u_C, & x+y\le L\\ (1-\frac{y}{L})u_B+(1-\frac{x}{L})u_C+(\frac{x}{L}+\frac{y}{L}-1)u_D, & else\end{cases} \quad (14)$$

따라서, 다음과 같이 2L×2L 크기의 $h(\cdot)$(식(10)의 H를 구현하기 위한 변수)를 설정하면 affine transform을 수행하는 커널을 만들 수가 있다.

$$h(x,y)=\begin{cases}(\frac{x}{L}+\frac{y}{L}+1), & -L<x,y<0,\ x+y>-L\\ (1-\frac{x}{L}), & 0<x<L,\ -L<y<0,\ x+y>0\\ (1-\frac{y}{L}), & -L<x<0,\ 0<y<L,\ x+y>0\\ (1+\frac{y}{L}), & 0<x<L,\ -L<y<0,\ x+y<0\\ (1+\frac{x}{L}), & -L<x<0,\ 0<y<L,\ x+y<0\\ (1-\frac{x}{L}-\frac{y}{L}), & 0<x,y<L,\ x+y<L\end{cases} \quad (15)$$

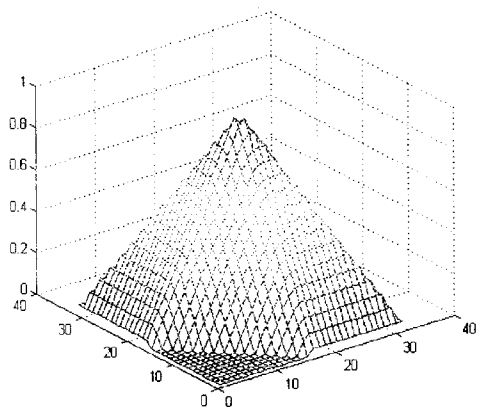


그림 6. affine 변환 커널
Fig. 6. Affine transform kernel.

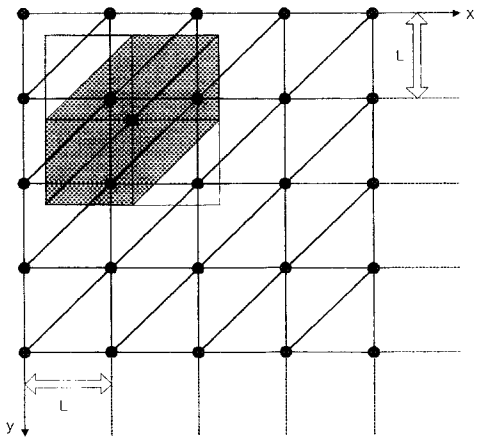


그림 7. ATMC를 수행하는 컨볼루션
Fig. 7. Convolution for ATMC.

그러면, <그림 6>과 같은 모양의 커널이 되고 <그림 7>에서와 같이 컨볼루션으로 affine transform을 수행할 수가 있다. <그림 7>에서 사각형 점의 위치를 보간하는 데에 색칠된 육각형 모양에 걸리는 세 개의 격자의 위치에서의 움직임 벡터 값이 곱해지게 된다.
이와 같은 방법으로 ATMC와 BTMC에서도 FMC와 같은 방식으로 알고리즘을 적용시킬 수 있다. 즉, ATMC와 BTMC는 특별한 필터 커널을 사용하는 FMC의 특별한 경우라고 볼 수 있다.

Ⅳ. 시뮬레이션 결과

이번 장에서는 BMC의 성능을 개선하기 위한 방법으로서의, ATMC, BTMC 방식과 FMC방식의 성능을 비교 분석하였다. FMC, ATMC와 BTMC도 Ⅲ장의 내용대로 구현하였다. Ⅳ.1절에서는 움직임 벡터 필드를 각각 BMC와 ATMC, BTMC, FMC를 이용하여 복원하였을 때의 성능을 평가하고, Ⅳ.2절에서는 각각 알고리즘을 CCITT시퀀스인 "deadline"시퀀스에 적용하였다.

1. Motion field example

본 논문에서는 ATMC, BTMC 방식과 FMC방식의 성능을 평가하고 분석하기 위하여 비선형 움직임 벡터 필드인 $\sin(ax)\sin(ay)$를 움직임 벡터 필드(motion field)로 사용하였다. <그림 8>은 이 원래의 움직임 벡터 필드를 보여준다. <그림 9>는 이 움직임 벡터 벡터 필드를 매크로 블록 단위로 샘플링한 뒤에 BMC에서 사용되어지는 움직임 벡터 필드를 구한 것이다. 눈에 선명하게 보이는 블록 모양의 움직임 벡터 필드가 블록화 현상을 야기하는 원인이 되는 것이다.

이를 없애기 위해 ATMC, BTMC, FMC에 사용되어지는 움직임 벡터 필드가 <그림 10>, <그림 11>, <그림 12>에 보여진다. <그림 10>은 ATMC에서 쓰이는 움직임 벡터들의 모습인데, 블록 모양은 나타나지 않으나 에러를 유발하는 삼각형 patch 모양을 볼 수가 있다. <그림 11>에서는 BTMC의 움직임 벡터 필드이고 사각형 모양의 patch가 눈에 보인다. 이 또한 완벽한 모션벡터의 복원이 아닐 것이다. <그림 12>는 FMC에 의한 움직임 벡터 필드이다. 가장자리를 제외한 영역에서 거의 완벽하게 복원됨을 알 수가 있다. 따라서 이와 같은 움직임 벡터 필드에서는, 눈으로 보기에는 FMC가 다른 방법들보다 우수함을 알 수가 있다. 또한, 이들

의 복원 정도를 정량적으로 분석하기 위하여 다음과 같은 측정 지수를 계산하였다.

$$F = 20\log\frac{\|u_{true} - u_{BMC}\|}{\|u_{true} - u_{IMC}\|} \quad (16)$$

여기서 u_{true}는 실제의 움직임 벡터 필드이고 u_{BMC}는 BMC에 적용되는 움직임 벡터 필드이고 u_{IMC}는 ATMC, BTMC, FMC에 적용되는 움직임 벡터 필드를

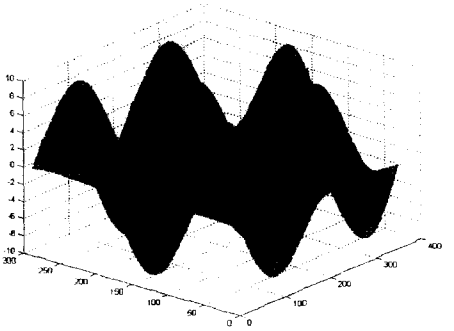


그림 8. 원래의 모션 벡터 필드
Fig. 8. Original motion vector field.

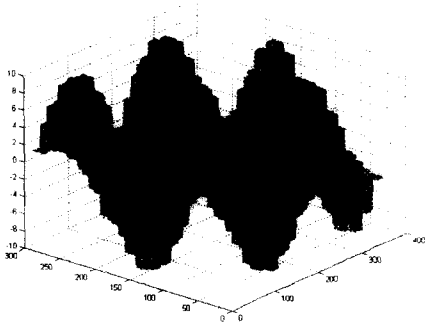


그림 9. BMC에서 사용되는 움직임 벡터 필드
Fig. 9. Motion vector field used in BMC.

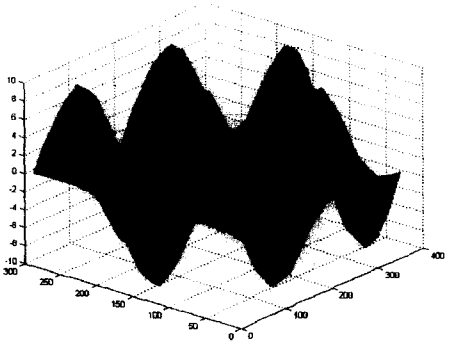


그림 10. ATMC에서 사용되는 움직임 벡터 필드
Fig. 10. Motion vector field used in ATMC.

나타낸다. 이 지수를 계산한 결과 ATMC에서는 11.5dB, BTMC에서는 12.4dB 그리고 FMC에서는 18.4dB를 얻을 수 있었다. 역시 이와 같은 Band-limited 움직임 벡터 필드에서는 FMC가 다른 방법들에 비해 우수함을 알 수 있다

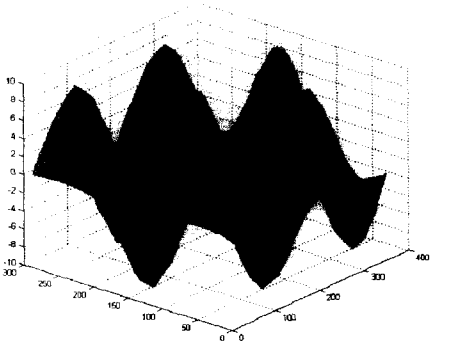


그림 11. BTMC에서 사용되는 움직임 벡터 필드
Fig. 11. Motion vector field used in BTMC.

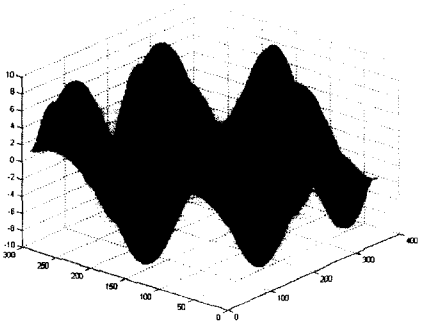


그림 12. FMC에서 사용되는 움직임 벡터 필드
Fig. 12. Motion vector field used in FMC.

2. 비디오 시퀀스에 적용

또한, ATMC, BTMC 그리고 FMC 알고리즘을 CCITT시퀀스인, "deadline", 시퀀스에 적용하여 평가였다. 시퀀스는 화소 단위로 (352×288), 블록단위로 (16×16)의 크기를 갖는다. 블록화 현상의 제거는 육안으로 기존의 방식인 BMC와 비교 평가되었고, 21개의 프레임에 적용하여 각각의 경우에 예측 정확도를 정량적으로 분석하기 위해 PSNR값 또한 기존의 방식의 값과 비교되었다.

<그림 13>은 "deadline"시퀀스의 7번째 원영상, <그림 14>는 6번째 영상과 BMA로 구해진 움직임 벡터를 사용하여 BMC로 예측된 영상이고, <그림 15>는 최적 움직임 벡터 추정 알고리즘으로 구한 움직임 벡터를 이용하여, 6번째 영상으로부터 FMC로 예측된 영상이다. <

그림 16>은 같은 알고리즘을 사용하여 ATMC로 예측된 영상이고, <그림 17>은 BTMC로 예측된 영상이다.

그림 13. deadline sequence의 7번째 영상
Fig. 13. 7th image of deadline sequence.

그림 14. BMC 방식으로 합성된 예측 영상
Fig. 14. Predictive image by BMC.

그림 15. FMC방식으로 합성된 예측영상
Fig. 15. Predictive image by FMC.

그림 16. ATMC방식으로 합성된 예측 영상
Fig. 16. Predictive image by ATMC.

BMC로 합성된 영상인 <그림 14>에서는 손과 안경 근처에서 블록화 현상이 관찰된다. 그러나, FMC로 합성된 영상인 <그림 15>와 ATMC로 합성된 영상인 <그림 16> 그리고 BTMC로 합성된 <그림 17>에서는 이와 같은 화질을 낮추는 블록화 현상이 발생하지 않았다.

<그림 18>에서는 기존의 BMC방식과 FMC, ATMC, BTMC의 네 가지 방식에 따라, 21개의 프레임에 대하여 적용한 결과 도출된 PSNR 값을 그린 것이다. FMC, ATMC, BTMC에 의해 예측된 경우 BMC보다 월등한 PSNR값을 보임을 알 수가 있고, 이 경우에는 FMC 방식이 가장 좋게 나타났음을 알 수가 있다. 21개 프레임에 걸친 PSNR들의 평균값들도 이와 같은 사실을 뒷받침한다. BMC를 적용하였을 때의 PSNR들의 평균값은 33.29dB인데 비해, ATMC에서는 33.91dB이고 BTMC에서는 33.99dB이다. 그리고 FMC에서의 평균값은 34.26dB이다.

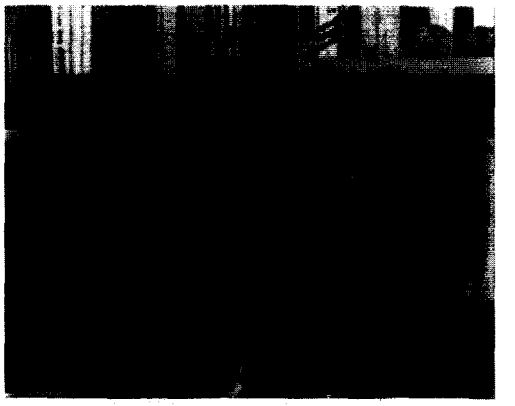


그림 17. BTMC방식으로 합성된 예측영상
Fig. 17. Predictive image by BTMC.

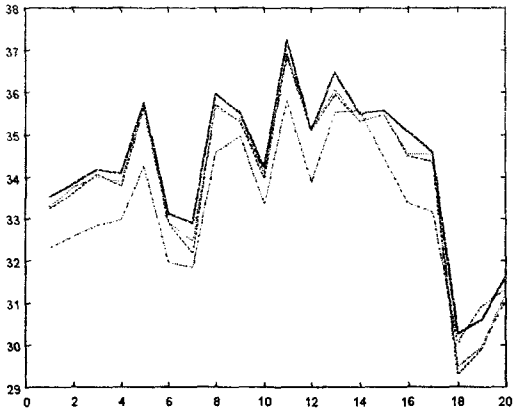


그림 18. PSNR값의 비교 : BMC(dash-dot), ATMC(dotted), BTMC(dashed), FMC(solid)
Fig. 18. Comparison of PSNR: BMC(dash-dot), ATMC(dotted), BTMC(dashed), FMC(solid).

V. 결 론

BMC 방식은 현재 프레임을 예측하는데 있어서 불연속적인 움직임 벡터 필드를 사용하므로 블록화 현상(blocky artifact)을 야기한다. 그래서 본 논문에서는 블록화 현상을 제거하기 위해 예측 영상을 합성하기 전에 움직임 벡터를 화소 단위로 보간하는 알고리즘을 사용하였다.

이러한 알고리즘은 FMC 뿐만 아니라 ATMC와 BTMC을 구현하는 데에 적용할 수 있으므로, 본 논문에서는 BMC에서 야기되는 블록화 현상을 없애고 예측 성능을 높이는 움직임 보상 방법으로서, FMC를 ATMC와 BTMC에 더불어서 비교 평가하였다. 또한 ATMC와 BTMC는 특별한 필터 커널을 사용하는 FMC의 특별한 경우임을 보였다.

본 논문상의 실험 결과로는 FMC가 가장 우수하였으나, 비디오 시퀀스에 따라서 또는 특정 장면에서는 ATMC나 BTMC가 더 우수한 경우도 있을 수 있다. 그렇기 때문에, 본 논문에서 제안된 알고리즘에서, 부호화기(encoder)가 세 가지 방법 중에 가장 높은 PSNR을 갖는 방법을 적용시키는 방법으로 개선할 수도 있다. 이는 H를 고정되었다고 보고 최적화 했던 본 논문의 문제에 부분적으로 최적화 된 답을 제공할 수 있을 것이다.

본 논문의 결과에서 알 수 있듯이, 보간을 이용하여 화소 단위로 움직임 벡터 필드를 부드럽게 하는 FMC(ATMC, BTMC)는 블록화 현상을 완벽하게 제거할 수 있었고 PSNR 값도 BMC보다 월등한 값을 보였다.

이러한 방법들은, 부호화 과정에서 최적화 된 움직임 벡터를 찾기 위해서 복잡한 계산을 해야 되는 반면에, 복호기(decoder)에서는 단지 예측 영상을 합성하기 전에 보간과정만 거치면 되므로 BMC보다 조금 많은 계산만으로 블록화 현상을 제거하고 예측 성능을 높이는 성과를 얻을 수가 있다.

감사의 글

이 논문은 1998년 한국학술진흥재단의 학술연구비에 의하여 지원되었음(총괄과제번호 E-00261).

참 고 문 헌

[1] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799-1808, 1981.

[2] H. G. Musmann, P. Pirsch, and H. Grallert, "Advances in picture coding", *Proc. IEEE*, vol. 73, no. 4, pp. 523-548, 1985.

[3] M. Ghanbari, "The cross-search algorithm for motion estimation", *IEEE Trans. Com.*, vol. 38, pp. 950-953, 1990.

[4] H. Brusewitz, "Motion compensation with triangles", in *Proc. 3rd International Conference on 64 kbit/s Coding of Moving Video*, (Rotterdam, Netherlands), Sept. 1990.

[5] G. J. Sullivan and R. L. Baker, "Motion compensation for video compression using control grid interpolation", in *Proc. ICASSP 91*, pp. 2713-2716, (Toronto, Canada), May 1991.

[6] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations", *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 4, no. 3, pp. 339-356, 1994.

[7] Y. Nakaya and H. Harashima, "An iterative motion estimation method using triangular patches for motion compensation", in *Proc. SPIE Visual Comm. and Image Processing.*

pp. 546-557, (Boston, MA), Nov. 1991.

[8] Y. S. Jung, G. S. Park, and S. M. Song, "Elimination of blocky artifacts in predictive video coding, in *Proc. SPIE Visual Comm and Image Processing*", vol. 3309, (San Jose, CA), pp. 232240, 1998.

[9] H. Watanabe and S. Singhal, "Windowed motion compensation", *in Proc. SPIE Visual Comm. Image Processing*, vol. 1605, pp. 582-589, 1991.

[10] S. Nogaki and M. Ohta, "An overlapped block motion compensation for high quality motion picture coding", *in Proc. of IEEE International. Symposium on Circuits and Systems*, pp. 184-187, 1992.

[11] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation : an estimation -theoretic approach", *IEEE Trans. On Image Processing*, vol. 3, no. 5, 1994.

[12] Geun-Soo Park, Ho-min Eum and S. Moon-Ho Song, "Filtered motion compensation for video compression", *Proc. ITC-CSCC'98* (Sokcho, Korea) , vol. 1, pp. 59-62 July 1998.

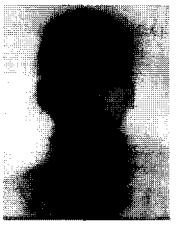
[13] B. Porat, *A course in Digital Signal Processing*. John Wiley and Sons, New York, 1997.

[14] David G Luenberger, "Linear and nonlinear programming", 2nd edition, Addison wesley, Menlo Park, 1984.

저 자 소 개

陰 湜 珉(學生會員)

1974년 2월 14일생. 1998년 2월 고려대학교 전파공학과 졸업(공학사). 1998년 3월~현재 고려대학교 대학원 전파공학과 석사과정. 주관심분야 : 동영상압축



朴 根 秀(正會員)

1974년 4월 9일생. 1997년 고려대학교 전파공학과 졸업(공학사). 1999년 2월 고려대학교 대학원 전파공학과 졸업(공학석사). 현재 삼성전자 정보통신총괄 네트웍 사업부 연구원. 주관심분야 : 동영상 압축

宋 文 豪(正會員)

1959년 2월 5일생. 1982년 6월 M.I.T 대 전자공학과 졸업(공학사). 1985년 6월 U.C.L.A. 대 전자공학과 졸업(공학석사). 1991년 12월 U.S.C. 대 전자공학과 졸업(공학박사). 1982년 6월~1983년 11월 미국 Litton 산업 연구원. 1983년 11월~1991년 12월 미국 Hughes 항공회사 선임연구원. 1992년 1월~1993년 12월 Stanford 대 Research Associate. 1994년 1월~1995년 2월 University of California 조교수. 1995년 3월~현재 고려대학교 전파공학과 부교수. 주관심분야 : 동영상 압축, 레이다 신호처리, CDMA 코드획득, 메디컬 영상신호처리