論文98-35S-5-9

움직임경계블록의 영역분할을 이용한 프레임간 내삽

(Interframe Interpolation using Segmentation of Blocks on Motion Boundary)

李 起 同 *, 金 東 郁 *, 姜 應 寬 *, 崔 宗 秀 *

(Ki Dong Lee, Dong Wook Kim, Eung Kwan Kang, and Jong Soo Choi)

요 약

블록 기반 움직임 보상형 프레임간 내삽기법은 블록단위의 움직임벡터를 이용해 건너뛴 프레임을 내삽하기 때문에, 내삽된 영상에서 심한 블록효과가 나타난다. 본 논문에서는 이와 같은 블록효과를 감소시키기 위한 프레임간 내삽기법을 제안한다. 첫째, 순방향 움직임벡터를 이용해 수신측에서 쉽게 역방향 움직임벡터를 구하는 방법을 제안한다. 드러나는 영역과 가려지는 영역을 효과적으로 예측하기 위해서는 송신측으로부터 전송 받는 순방향 움직임벡터뿐만 아니라 역방향 움직임벡터도 필요하기 때문이다. 둘째, 움직이는 물체의 경계에 위치한 블록들을 몇 개의 영역으로 분할하고, 분할된 영역의 움직임벡터를 인접한 블록들의 움직임벡터로부터 추정하는 방법을 제안한다. 동일한 블록일지라도 움직이는 물체와 배경이 서로 다른 움직임을 갖을 수 있게 하여 블록효과를 줄이고자 하였다. 모의 실험결과, 주관적, 객관적 화질 면에서 제안한 방법이 기존의 방법에 비해 우수한 것을 알 수 있다.

Abstract

Block-based interframe interpolation algorithms cause severe block effect because the algorithm interpolates the skipped frame by using block based motion vector. Therefore, in this paper, we propose an algorithm that reduces the block effect in the interpolated frames. First, we propose an algorithm that obtains backward motion vector by using forward motion vector received from the transmitter. In order to predict well covered and uncovered region, backward motion vector is needed as well as forward motion vector. Second, we propose the algorithm which segments the motion boundary blocks into regions and obtains the motion vector of each region from candidates that consist of the motion vectors of neighbor blocks. This algorithm makes it possible that the moving object and the background, in spite of being in the same block, have different motion vectors from each other so that the block effect can be reduced. According to the results of simulation, the proposed algorithm is superior to conventional algorithm in subjective quality as well as in objective quality

I. 서 론

움직임 보상형 프레임간 내삽은 그림 1에서와 같이 두 장의 참조프레임 R_1, R_2 사이에 하나 이상의 프레임을 삽입하는 기법으로, 동영상 압축이나 프레임율 변환 또는 느린 화면 재생을 위한 목적으로 사용될 수 있다[1]. 동영상 압축의 경우, 송신측에서는 N개의 프레임씩 건너뛰면서 전송하고, 수신측에서는 전송받은

* 正會員, 中央大學校 電子工學科

(Dept. of Elec. Eng., Chung Ang Univ.)

※ 본 연구는 정보통신연구관리단(대학기초연구지원사업)의 연구비 지원으로 수행되었으며 지원에 감사드립니다.

接受日字: 1997年7月19日, 수정완료일: 1998年4月2日

프레임 사이의 움직임 정보를 이용해 건너뛴 프레임을 재구성한다. 따라서, 기존의 동영상 압축기법과 함께 사용함으로써 그 압축효과를 더욱 향상 시킬 수 있다[2].

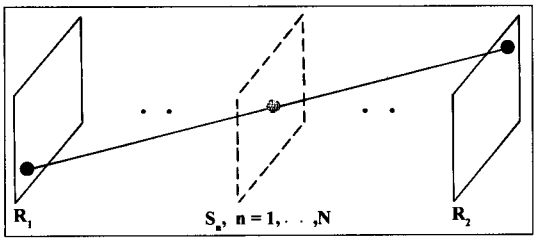


그림 1. 움직임 보상형 프레임간 내삽
Fig. 1. Motion compensated interframe interpolation.

동영상 압축을 위한 몇 가지 프레임간 내삽기법이 제안되었는데, C. K. Wong 와 O. C. Au는 블록 기반 움직임벡터를 이용해 건너뛴 프레임을 내삽하는 MCTI (motion compensated temporal interpolation)기법과 MCTI기법의 수행속도를 개선시킨 FMCTI(fast MCTI)기법을 제안하였다[3]. 동일 블록내에 있는 모든 화소들이 동일한 움직임을 갖는다는 가정아래 건너뛴 프레임을 내삽하기 때문에 움직임의 경계에서 심한 블록효과가 나타난다. 움직임 경계에 위치한 블록 내에는, 동일 블록일지라도, 서로 다른 움직임이 존재할 수 있음에도 불구하고 이를 고려하지 않았기 때문이다. 또한, 이와 같은 블록효과를 감소시키기 위해서, 내삽하고자 하는 프레임의 전, 후에 위치한 3장의 참조프레임을 이용해 내삽될 프레임 내에 있는 화소들을 움직이는 물체, 드러난 배경, 가려진 배경, 정적인 배경 등으로 분류하여 서로 다르게 보상하는 MMCTI(modified MCTI)기법이 제시되었다[4]. 그러나 이 기법으로는 내삽될 프레임 내에 있는 화소들에 대한 정확한 분류가 어렵기 때문에 블록효과가 크게 감소하지 않는다. 이 외에도 여러 가지 움직임 보상형 프레임간 내삽기법들이 제안되었지만, 이들은 기존의 동영상 송수신시스템에서 사용되는 블록단위의 움직임정보를 이용하지 않고 별도로 움직임을 추정하기때문에 수신측에서 많은 계산량이 요구된다[1] [2] [5].

본 논문에서는 블록효과로 인한 문제점을 해결하기 위해 두 가지 방법을 제안한다. 첫째, 부호화 과정에서 생성되는 순방향 움직임벡터를 이용해 수신측에서 손쉽게 역방향 움직임벡터를 구하는 방법을 제안한다 수신측에서 드러나는 영역과 가려지는 영역을 보다 정확히 예측해 건너뛴 프레임을 내삽하기 위해서는 송신측으로부터 전송 받은 순방향으로 추정된 움직임벡터뿐만 아니라 역방향 움직임벡터도 필요하다. 그러나 역방향 움직임벡터를 얻기 위해 블록정합방법과 같은 기존의 움직임추정방법을 사용하는 것은 지나치게 많은 계산량을 요구한다. 따라서 이미 알고있는 정보를 이용해 역방향 움직임벡터를 쉽게 구하는 방법을 제안한다. 둘째, 움직임경계에 위치한 블록들을 영역분할하고, 각각의 영역에 서로 다른 움직임벡터를 할당하는 방법을 제안한다. 동일한 블록 내에 있을지라도, 영역마다 서로 다른 움직임을 갖을 수 있어, 기존의 블록 기반 움직임 보상형 프레임간 내삽기법에서 발생하는 문제점을 해결할 수 있다.

Ⅱ장에서는 순방향 움직임벡터를 이용해 역방향 움직임벡터를 구하는 방법과 오정합에 의해 잘못 구해진 움직임벡터를 교정하는 방법에 대해 제시하고, Ⅲ장에서는 움직임 경계에 위치한 블록들의 영역별 움직임 추정 방법에 대해 알아본다. 그리고 Ⅳ장에서는 모의실험 결과를 보이며, 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 움직임벡터 추정

1. 역방향 움직임벡터의 추정

건너뛴 프레임을 수신측에서 효과적으로 내삽하기 위해서는, 먼저 내삽될 프레임을 드러나는 영역, 가려지는 영역, 정적인 배경, 움직이는 물체 등으로 구분해 이를 서로 다르게 보상해주어야 한다[2] [4] [6]. 일반적인 동영상 부호화 기법을 고려해볼 때, 수신측은 송신측으로부터 순방향으로 추정된 움직임벡터만을 전달받는다. 여기서 순방향 움직임벡터란, 시간적인 순서로 볼 때, 앞에 위치한 프레임을 참조프레임으로하여 뒤에 올 프레임을 예측하기위해 사용되는 움직임 벡터를 뜻하고, 역방향 움직임 벡터란 그 반대를 의미한다.
그런데 순방향으로 추정된 움직임벡터만을 사용할 경우, 그림 2에서 볼 수 있는 것처럼 가려지는 영역은 잘 예측할 수 있는 반면, 드러나는 영역의 예측은 어렵게 된다. 따라서 가려지는 영역뿐 아니라 드러나는 영역도 예측할 수 있게 하기 위해서는 순방향 움직임벡터뿐만 아니라 역방향 움직임벡터도 필요하다. 이를 위해서는 수신측에서 블럭정합방법과 같은 기존의 움

직임탐색 방법을 이용해 별도로 역방향 움직임벡터를 구할 수도 있지만 이와 같은 방법은 지나치게 많은 계산량을 요구한다. 따라서 본 논문에서는 송신측으로부터 받은 순방향 움직임벡터를 이용해 보다 쉽게 역방향 움직임 벡터를 예측하는 방법을 제안한다.

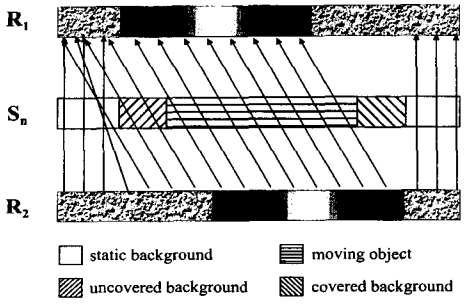


그림 2. 순방향 움직임벡터를 이용한 내삽
Fig. 2. Interframe interpolation using forward motion vector.

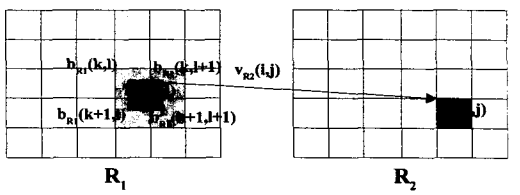
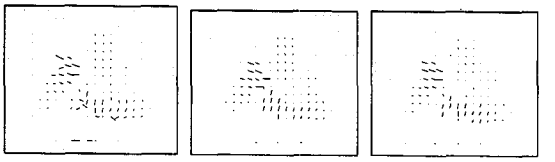


그림 3. 역방향 움직임벡터 추정
Fig. 3. Estimation of backward motion vector.

그림 3에서, R_1, R_2는 송신측으로부터 전송 받은 이미 알고 있는 참조프레임이고, $b_{R2}(i, j)$와 $b_{R1}(k, 1)$은 R_2프레임 내의 (i, j)번째 블록과 R_1프레임 내의 번째 블록을 각각 의미한다. 그리고 $v_{R2}(i, j)$는 블록 $b_{R2}(i, j)$에 할당된 순방향 움직임벡터이고, $b'_{R2}(i, j)$는 $b_{R2}(i, j)$가 참조하는 영역을 나타낸다. $0 \le i, k \le I$, $0 \le j, 1 \le J$ 일 때, R_1프레임에서, 영역 $b'_{R2}(i, j)$에 의해 겹쳐지는 각각의 블록에 $-v_{R2}(i, j)$를 할당하면, R_1프레임에 속한 각 블록에는 여러 개의 움직임벡터가 할당된다. 이들을 블록$b_{R1}(k, 1)$의 후보벡터집합 $C(b_{R1}(k, 1))$라 하면, 역방향 움직임벡터 $v_{R1}(k, 1)$은 다음 식을 만족하는 v_c와 같다.

$$\min_{v_c \in C(b_{R1}(k,1))} |BDIF\ (b_{R1}\ (k,1),\ v_c)| \qquad (1)$$

이때, $BDIF(r_R, v)$는 R프레임에 속한 영역 r_R과, 영역 r_R이 움직임벡터 v를 가지고 시간적으로 뒤에 위치한 프레임에서 참조하는 영역사이의 차를 의미한다. 따라서 후보벡터들 중에서 그 값이 가장 작은 벡터를 블록 $b_{R1}(k,1)$의 역방향 움직임벡터로 선택한다. 제안한 방법은 계산량이 비교적 많은 기존의 탐색과정을 거치지 않기 때문에 적은 계산량으로 원하는 결과를 얻을 수 있다. 그림 4에서, (b)는 제안한 방법을 이용해서 얻은 역방향 움직임 벡터를 나타낸 것이다.



(a) (b) (c)

그림 4. 오정합에 의한 움직임벡터 제거
(a)BMA로 구한 순방향 움직임벡터 (b) 제안된 방법으로 얻은 역방향 움직임 벡터 (c) 교정된 순방향 움직임벡터
Fig. 4. Elimination of motion vector estimated by mismatching.
(a) Forward motion vector estimated by BMA (b) Backward motion vector obtained by proposed algorithm (c) Corrected forward motion vector

2. 오정합에 의한 움직임벡터 제거

순방향 움직임벡터의 집합을 $V_{R2} = \{v_{R2}(i,j);\ 0 \le i \le I,\ 0 \le j \le J\}$, 위에서 얻은 역방향 움직임벡터의 집합을 $V_{R1} = \{v_{R1}(i,j);\ 0 \le i \le I,\ 0 \le j \le J\}$이라 하고 앞 절에서 제시한 순방향 움직임벡터를 이용해 역방향 움직임벡터를 얻는 과정을 $F(\cdot)$라 하면 다음과 같은 식으로 나타낼 수 있다.

$$V_{R1} = F(V_{R2}) \qquad (2)$$

이때, $F(\cdot)$는 송신측으로부터 받은 순방향으로 추정된 움직임벡터 중에서 역방향으로도 그 움직임이 일치하는 움직임벡터만을 선택하기 때문에, 이 과정에서 오정합에 의해 잘못 구해진 움직임벡터는 제거된다.

$$V'_{R2} = F(V_{R1}) \qquad (3)$$

따라서, 식 3과 같이, 앞 절에서 얻은 역방향 움직임벡터 V_{R1}을 이용하면 오정합에 의해 잘못 구해진 움직임벡터가 제거된 V'_{R2}를 얻을 수 있다. 그림 4에서, (a)는, 송신측으로부터 받은, 블록정합방법에 의해 순방향으로 추정된 움직임벡터이며, (c)는 제안한 방법을 이용해 얻은 교정된 움직임벡터이다.

Ⅲ. 움직임 경계에 위치한 블록의 영역분할

1. 영역분할의 필요성

앞에서도 언급한 것처럼, 물체의 움직임이 있을 때 움직이는 물체의 경계에서는 드러나는 영역과 가려지는 영역이 발생한다. 그런데 기존의 블록기반 움직임 보상형 프레임간 내삽기법은 동일한 블록 내에 있는 모든 화소들이 동일한 움직임을 갖는다는 가정아래 건너뛴 프레임을 내삽한다. 움직이는 물체의 경계에 위치한 블록은 동일 블록 내에 서로 다른 둘 이상의 움직임을 포함할 수 있음에도 불구하고 하나의 움직임벡터로 표현하기 때문에 드러나는 영역과 가려지는 영역을 제대로 보상할 수 없고, 이로 인해 블록효과가 심하게 나타나게 된다[3]. 따라서 움직임 경계에 위치한 블록들에 대해서만 밝기값에 근거해 영역을 분할하고 영역별로 서로 다른 움직임을 갖을 수 있게 한다면 드러나는 영역과 가려지는 영역을 잘 표현할 수 있어 블록효과를 감소시킬 수 있다.

2. 움직임 경계 블록

블록효과는 원래의 영상과 움직임 보상영상의 차가 상대적으로 큰 블록에서 발생한다고 볼 수 있다. 그림 5는 Salesman 영상의 9번 프레임과 11번 프레임 사이의 DFD(Displaced Frame Difference)를 보인다. 움직이는 물체의 경계에서 그 값이 크게(검게) 나타나는 것을 볼 수 있다. 따라서 프레임 내에 있는 어떤 블록이 움직임경계에 있는 블록인지 아닌지를 판단하기 위해 다음과 같이 BDIF값을 이용할 수 있다.

$$|\mathrm{BDIF}(b_{R1}(i,\ j),\ v_{R1}(i,\ j)| \geq \mathrm{TH} \qquad (4)$$

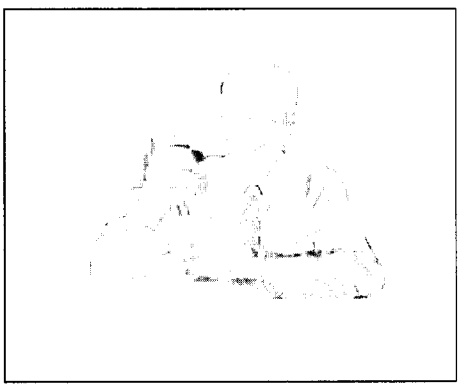


그림 5. Salesman영상의 9번과 11번 프레임 사이의 DFD

Fig. 5. DFD between 9th frame and 11th frame of "Salesman" sequence.

여기서, BDIF값이 문턱치 TH 보다 크면, 그 블록은 움직임 경계에 위치한 블록이라고 판단하고 이를 움직임 경계 블록이라 부르기로 한다.

3. 움직임 경계 블록의 영역분할과 분할된 영역의 움직임 추정

먼저, 움직임 경계 블록으로 판정된 블록을 몇 개의 영역으로 분할한다. 분할된 각 영역의 움직임은 탐색에 의한 정합방법을 이용해서 얻을 수 있으나, 앞에서도 언급한 바와 같이 이는 지나치게 많은 계산을 요구한다. 따라서 본 절에서는 Ⅱ장에서 구한 블록에 기반한 순방향, 역방향 움직임 벡터를 이용해 간단하게 움직임 경계 블록 내에 있는 영역들의 움직임벡터를 구하는 방법을 제안한다. 그림 6에서, $b_{R1}(i,\ j)$를 움직임 경계 블록이라 하고, 영역분할하면 다음과 같이 나타낼 수 있다.

$$b_{R1}(i,\ j) = \bigcup_{k=1}^{K} r_{R1}\ (i,j,k) \qquad (5)$$

여기서 $r_{R1}(i,j,k)$는 블록 $b_{R1}(i,j)$내에 있는 k번째 영역을 나타낸다. $r_{R1}(i,j,k)$의 움직임벡터 $r_{rR1}(i,j,k)$는 다음 식을 만족하는 v_c와 같다.

$$C(r_{R1}(i,j,k))=\{v_{R1}(i,j),\ v_{R1}(i-1,j),\ v_{R1}(i+1,j),\ v_{R1}(i,j-1),\ v_{R1}(i,j+1)\} \qquad (6)$$

$$\min_{v_c \in C(r(i,j,k))} |\mathrm{DFD}\ (r(i,j,k),\ v_c)|,\ 0 \leq k \leq K \qquad (7)$$

여기서 $C(r_{R1}(i,j,k))$는 $r_{R1}(i,j,k)$의 후보벡터를 말하며, $r_{R1}(i,j,k)$를 포함한 블록과 4근방에 위치한 블록의 움직임벡터를 후보벡터로 선택한다. 후보벡터들 중에서 BDIF가 가장 작은 벡터를 $r_{R1}(i,j,k)$의 움직임벡터로 할당한다. R_2프레임에 속한 움직임 경계 블록도 마찬가지 방법으로 영역별 움직임벡터를 구할 수 있다.

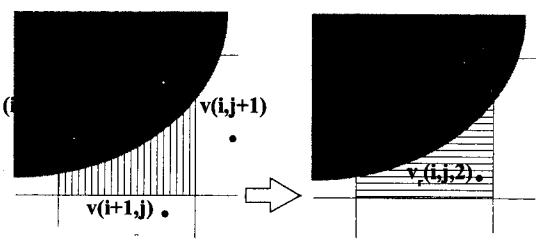


그림 6. 움직임 경계 블록 내에 있는 영역의 움직임 벡터 추정

Fig. 6. Estimation of regions in motion boundary block.

Ⅱ장에서 얻은 순방향 움직임벡터와 역방향 움직임벡터 그리고 Ⅲ장에서 얻은 움직임 경계 블록의 영역별 움직임정보를 이용해서 건너뛴 프레임을 내삽한다. 순방향 움직임벡터를 이용해 가려지는 영역을 구하고, 역방향 움직임벡터를 이용해 드러나는 영역을 구한다.

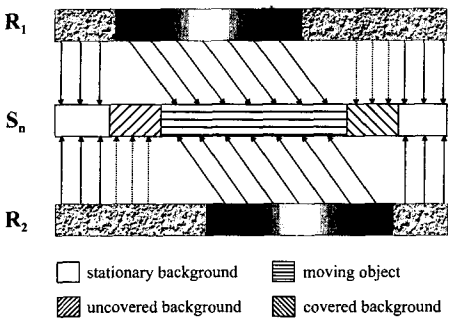


그림 7. 건너뛴 프레임의 재구성
Fig. 7. Reconstruction of skipped frame.

그 이외의 영역은 움직이는 물체 또는 정적인 배경에 해당하며 각각의 영역은 그림 7과 같이 처리한다. 움직이는 물체와 정지한 배경영역은 R_1, R_2 프레임을 이용하고, 드러난 배경영역은 R_2프레임에서, 가려진 배경영역은 R_1프레임에서 그 값을 가져와 내삽한다.

Ⅳ. 실험결과

본 절에서는 제안한 방법의 성능을 평가하기 위해 352×288 크기의 "Salesman"영상을 1번 프레임부터 100번 프레임까지 사용하였다. 손에 든 상품을 설명하는 장면으로 손의 움직임이 비교적 많은 영상이다. 성능비교대상은 Chi-Kong Wong와 Oscar C. Au에 의해 제안된 MCTI기법[3]과 MMCTI기법[4]으로 하였다. 순방향 움직임벡터는 블록정합방법을 이용해 구했으며, 블록의 크기는 16으로 했다. N=1인 경우, 즉 하나의 프레임을 건너뛴 경우에는 탐색영역을 ±15로 하였으며, N=2인 경우에는 탐색영역을 ±31로 하여 정수단위 움직임을 추정하였다. 그리고 움직임 경계블록의 판정을 위한 문턱치 TH는 실험을 통해 적절한 상수 값을 선택 하였다.

그림 8은, N=1이고 10번째 프레임을 내삽하는 경우, 드러나는 영역과 가려지는 영역을 나타낸 그림이다. 흰색은 드러난 영역을 나타내고, 검은색은 가려진 영역을 나타낸다.

그림 8. 드러난 영역과 가려진 영역
Fig. 8. Uncovered and covered region.

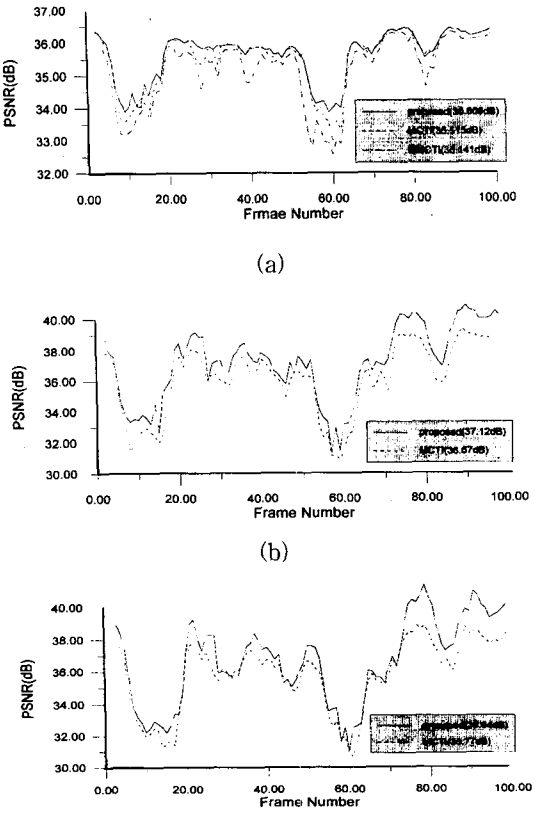


그림 9. PSNR 비교 (a) N = 1 (b) N = 2, n = 1 (c) N = 1, n =2
Fig. 9. Comparison for PSNR. (a) N = 1 (b) N = 2, n = 1 (c) N = 1, n =2

기존의 방법과 제안한 방법의 객관적인 성능 비교를 위해, 그림 9에 얻어진 PSNR을 보였다. 참조프레임으로는 내삽하고자 하는 프레임의 전,후 프레임을 원 영상 그대로 사용했다. 한 장의 프레임을 건너뛴 N=1인

경우와 두 장의 프레임을 건너뛴 N=2인 경우에 대해 각각 실험하였다. MMCTI기법은 특성상 N이 1보다 큰 경우에는 불가능하기때문에 N=1인 경우만을 실험하였다. 예를 들어, N=1인 경우 10번 프레임을 내삽하기 위해서는 9번과 11번 프레임을 사용하며, 11번 프레임을 내삽하기 위해서는 10번과 12번 프레임을 사용한다. N=2인 경우에는 먼저 내삽되는 프레임(n=1인 내삽 프레임)과 나중에 내삽되는 프레임(n=2인 내삽프레임)으로 나누어 성능평가를 실시했다. N=1인 경우, 제안한 방법이 기존의 방법에 비해 평균 0.1dB 정도 화질이 개선되었으나, 비교적 움직임이 큰, 즉 드러나는 영역과 가려지는 영역이 많이 나타나는 구간에서는 그 차가 크게 나타나는 것을 볼 수 있다. N=2인 경우에는, N=1인 경우에 비해 참조프레임간의 움직임이 크기 때문에 평균 0.5dB이상 성능이 개선된 것을 수 있다.

그림 10은 주관적화질을 비교하기 위해, 기존의 방법과 제안한 방법에 의해 내삽된 영상을 보인 것이고, 그림 11은 원영상과 내삽된 프레임 사이의 FD(Frame difference)를 보인 것이다. 움직임이 많은 오른손과 외손주변에서 블록효과가 많이 줄어들었다.

계산량을 비교해보면, 한 프레임이 M×N개의 블록을 갖고, 탐색범위가 SX×SY일 때, 전역탐색기법을 이용해 역방향 움직임벡터를 구하면 M×N×SX×SY회의 블록비교를 위한 계산량이 필요하다. 반면에 제안한 방법을 이용해 역방향 움직임벡터를 구하면 블록당 후보벡터의 수가 평균 4개 미만이기 때문에, 4×M×N회 미만의 블록비교가 필요하며, 순방향 움직임벡터의 교정을 위해서도 동일한 회수 만큼의 블록비교가 필요하다. 그리고 영역별 움직임벡터를 구하기 위해서는, 5개의 후보벡터를 이용하기 때문에 각각의 영역에

(a) (b) (c)

그림 10. 내삽된 프레임(10번 프레임, N=1)
(a) MCTI (b) MMCTI (c) 제안한 방법
Fig. 10. Interpolated frame(10th frame, N=1).
(a) MCTI (b) MMCTI (c) proposed

(a) (b) (c)

그림 11. 원 영상과의 FD (FD > 10)
(a) MCTI (b) MMCTI (c) 제안한 방법
Fig. 11. FD between original frame and interpolated frame (FD > 10).
(b) MCTI (b) MMCTI (c) proposed

대해 5회씩 영역비교를 하게 된다. 그런데 식5에 따르면 이들 영역들의 합은 하나의 블록과 같기 때문에, 움직임 경계블록 하나의 영역별 움직임벡터를 구하기 위해서는 5회의 블록비교를 위한 계산량이 필요하다.

V. 결 론

본 논문에서는 기존의 블록기반 움직임 보상형 프레임간 내삽기법에서 발생하는 블록효과를 감소시키기 위한 새로운 프레임간 내삽기법을 제안하였다. 양방향 움직임벡터를 모두 사용해서 드러나는 영역과 가려지는 영역을 보다 정확히 예측 및 보상하였고, 움직임 경계에 위치한 블록을 영역분할해서 동일한 블록에 속했을지라도 서로 다른 움직임을 가질 수 있게 했다. 또한, 역방향 움직임벡터나 영역별 움직임을 구하기 위한 새로운 기법을 제시하였다. 모의실험결과, 제안한 방법이 기존의 방법에 비해 주관적 화질은 물론 객관적 화질도 우수함을 보여준다.

참 고 문 헌

[1] C. L. Huang and T. T. Choa, Motion-compensated interpolation for scan rate up-conversion, *Optical Engineering*, vol. 35, no. 1, pp. 166-176, Jan. 1996.

[2] J. R. Corbera and J. Sklansky, Interframe interpolation of cinematic sequences, *J. Visual Communication and Image Representation*, vol. 4, no. 4, pp. 392-406, Dec. 1993.

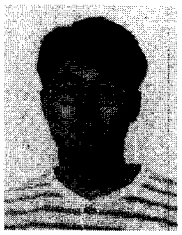
[3] C. K. Wong and O. C. Au, Fast motion compensated temporal interpolation for video, *Proc. SPIE Visual Comm. and Image Proc.*, vol. 2501, pp. 1108-1118, Taipei, Taiwan, 24-26 May 1995.

[4] C. K. Wong and O. C. Au Modified motion compensated temporal frame interpolation for very low bit rate video, *Proc. IEEE ICASSP*, vol. 6, pp. 2327-2330, Atlanta, Georgia USA, 7-10 May 1996.

[5] P. Csillag and L. Boroczky, Estimation of accelerated motion for motion-compensated frame interpolation, *Proc. SPIE Visual Comm. and Image Proc.*, vol. 2727, pp. 604-614, Orlando, Florida, 17-20 Mar. 1996.

[6] C. Cafforio, F. Rocca, and S. Tubaro, Motion compensated image interpolation, *IEEE Trans. Commun.*, vol. 38, no. 2, pp. 215-222, Feb. 1990.

저 자 소 개



李 起 同(正會員)

1996.2 : 중앙대학교 전자공학과 (공학사). 1998.2 : 중앙대학교 대학원 전자공학과 (공학석사). 1998.2 ~ 현재 : ㈜현대전자 멀티미디어 연구소 연구원. 주 관심분야는 동영상 부호화

姜 應 寬(正會員)

1993.2 : 중앙대학교 전자공학과 (공학사). 1995.2 : 중앙대학교 대학원 전자공학과 (공학석사). 1995.1 ~ 1997.5 : (주) 현대전자 정보통신연구소 연구원. 1997.9 ~ 현재 : 중앙대학교 대학원 전자공학과 박사과정 재학 중. 주관심분야는 영상처리, 컴퓨터비젼 등임

金 東 郁(正會員)

1987.2 : 성균관대학교 전자공학과 (공학사). 1992.2 : 중앙대학교 대학원 전자공학과 (공학석사). 1996.8 : 중앙대학교 대학원 전자공학과 (공학박사). 1997.3 ~ 1998.2 : 충남산업대학교 전자공학과 전임강사. 1998.3 ~ 현재 : 전주대학교 전자공학과 조교수. 주 관심분야는 동영상 부호화

崔 宗 秀(正會員) 第 28卷 B編 第 5號 參照
현재: 중앙대학교 전자공학과 교수