論文98-35S-12-8

움직임 벡터 분포 특성을 이용한 고속 적응 블럭 정합 알고리즘

(Fast Adaptive Block Matching Algorithm using Characteristic of the Motion Vector Distribution)

申 容 達 *, 金 榮 春 *

(Yong-Dal Shin and Young-Choon Kim)

요 약

본 논문에서는 움직임 벡터의 분포특성을 이용한 고속 적응 블럭 정합 알고리즘을 제안하였다. 제안 방법에서는 움직임 벡터에 따른 MAD(0,0)의 분포특성을 분석하여 블럭을 움직임이 없는 블럭, 작은 블럭, 중간정도인 블럭 혹은 큰 블럭으로 분류한 후 각 블럭의 특성에 따라 적응적으로 움직임 벡터를 추정한다. 제안한 방법의 성능을 평가하기 위해서 컴퓨터 시뮬레이션을 수행하였다. 이 결과로부터 제안 방법의 PSNR은 기존의 NTSS 방법과 거의 비슷하면서도 계산량이 30.44% ~ 40.27% 감소되는 효과적인 방법임을 확인할 수 있었다.

Abstract

We present a fast adaptive block matching algorithm using characteristic of the motion vector distribution. In the presented method, the block is classified into one of four motion categories: stationary block, quasi-stationary block, medium-motion block or high-motion block according to characteristic of the MAD(0,0) distribution for motion vector, each block estimates the motion vector adaptively. By the simulation, the PSNR of our algorithm is similar to NTSS method. The computation amount of the presented method decreased 30.44% ~ 40.27% more than NTSS method.

I. 서 론

통신 매체의 발달로 고속으로 정보를 전달하는 것이 가능하게 되었으며, 특히 여러 가지 정보 중에서 동영상 정보를 전송하기 위한 표준안으로서 MPEG이 소개되고 있다. 동영상은 대체로 움직임 보상 부호화 (motion compensation coding)를 이용하여 높은 데이터 압축이 이루어진다. 이 방법에 의한 데이터의 압축은 이전 프레임을 이용하여 움직임 추정 및 보상을 수행하고, 이때 추정된 움직임 벡터 (motion vector)에 의해서 보상된 영상과 원 영상과의 차이 신호를 부호화 한다.[1] 이 방법은 크게 화소 순환 알고리즘 (pel recursive algorithm: PRA)과 블럭 정합 알고리즘 (block matching algorithm: BMA)으로 나누어진다.[1][2] 블럭 정합 알고리즘은 화소 순환 알고리즘에 비하여 수행시간이 적게 소요되고 하드웨어 구현이 용이하기 때문에 움직임 보상 부호화에 널리 사용되고있다.

블럭 정합 알고리즘은 입력영상을 임의의 작은 블럭으로 나눈 후, 한 블럭내의 모든 화소는 동일 방향의 움직임을 갖는다고 가정하여 움직임 벡터를 추정하는 것으로, 이전 프레임에서 설정된 탐색영역에서 어떤 블럭이 현재프레임의 정해진 블럭으로 이동하였는가 찾는 것이다. N×N 화소를 갖는 블럭에서 프레임간 화소의 최대 움직임의 거리를 P화소라 가정할 때 상

* 正會員, 永同大學校 電子工學部

(School of Electronic Engineering, YoungDong University)

接受日字: 1998年6月26日, 수정완료일: 1998年10月9日

하 좌우로 최대 P화소 만큼의 움직임이 발생할 수 있으므로 탐색영역의 크기는 (2P+N)×(2P+N)이며, 이 탐색영역에서 정합할 수 있는 탐색점의 수는 $(2P+1)^2$ 이다. 이때 서로 정합될 수 있는 두 블럭사이에서 왜곡의 정도를 나타내는 척도로는 평균절대오차 (mean absolute difference: MAD), 평균자승오차 (mean squared difference: MSD)가 널리 사용되고 있다.[1][2] MAD 및 MSD는 각각

$$MAD(k,l)=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|X_t(i,j)-X_{t-1}(i+k,j+l)| \quad (1)$$

$$MSD(k,l)=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[X_t(i,j)-X_{t-1}(i+k,j+l)]^2 \quad (2)$$

이다. 여기서 $X_t(i,j)$는 현재 프레임, $X_{t-1}(i,j)$는 이전 프레임, (k,l)은 탐색 영역에서의 탐색점이다. 각 탐색점에 대해 MAD 혹은 MSD가 최소가 되는 (k,l)을 움직임 벡터로 정의한다. MAD는 MSD에 비하여 계산량이 적고 하드웨어 구현이 용이하여 널리 이용하고 있지만, MSD보다 정확한 움직임 벡터를 탐색하지 못하는 단점을 지니고 있다. 이렇게 정합될 수 있는 모든 화소에 대하여 오차를 구하고 그 중 가장 작은 오차를 갖는 탐색점의 값을 움직임 벡터로 결정하는 방법을 전 탐색 블럭 정합 알고리즘 (full search block matching algorithm)라 한다. 이 방법은 탐색영역내에서 최적인 움직임 벡터를 추정할 수 있으나, 정합하는 경우 탐색점이 너무 많으므로 많은 계산량을 필요로 하는 단점이 있다. 따라서 계산량을 줄이면서 거의 정확한 움직임 벡터를 찾을 수 있는 고속의 블럭 정합 알고리즘 즉, 2차원 로그 탐색 알고리즘, 3단계 탐색 (three step search: TSS) 알고리즘, 교차 탐색 (cross search) 알고리즘, 새로운 3단계 탐색 (new three step search: NTSS) 알고리즘 등이 연구되었다.[1]-[6] 이들 방법 중에서 계산량이 간단하면서도 거의 정확한 움직임 벡터를 추정하는 방법은 새로운 3단계 탐색 알고리즘[6]이다.

TSS 및 NTSS 방법에서는 움직임 벡터를 추정하는 과정에서, 제 1 단계에서는 블럭의 움직임 정도는 고려하지 않고 항상 고정적으로 탐색점을 각각 9개 및 17개 사용하여 움직임 벡터를 추정한다. 이와 같은 방법으로 움직임 벡터를 추정할 경우 움직임이 없는 블럭 및 적은 블럭에서는 계산량이 증가되는 단점이 있다.

본 논문에서는 움직임 벡터의 분포특성을 이용한 고속 적응 블럭 정합 알고리즘을 제안하였다. 제안 방법에서는 움직임 벡터에 따른 MAD(0,0)의 분포특성을 분석하여 블럭을 움직임이 없는 블럭, 작은 블럭, 중간 정도인 블럭 혹은 큰 블럭으로 분류한 후, 각 블럭의 특성에 따라 적응적으로 움직임 벡터를 추정한다.

제안한 방법의 성능을 평가하기 위해서 컴퓨터 시뮬레이션을 수행하였다. 이 결과로부터 제안 방법의 PSNR 성능은 기존의 NTSS 방법과 거의 비슷하면서도 계산량이 30.44% ~ 40.27% 감소되는 효과적인 방법임을 확인할 수 있었다.

Ⅱ. 제안한 적응 블럭 정합 알고리즘

블럭 정합 알고리즘은 현재 프레임 t의 (i,j)에 있는 블럭을 이전 프레임 t-1에 있는 블럭 (i,j)와 정합을 할 때 평균절대오차는 항상 탐색영역의 모든 탐색점에 대하여 평가를 해야만 움직임 추정이 정확하게 된다. 이와 같이 모든 탐색점에 대하여 움직임 벡터를 추정한다면 그 계산량은 매우 많아진다. 또한 고속으로 움직임 벡터를 추정하기 위해서 탐색점 수를 너무 적게 하여 블럭 정합 알고리즘을 수행하게 되면 국부최소에 빠져서 부정확한 움직임 벡터를 추정하게 되어 복원 영상에서 화질이 저하된다.

본 논문에서는 움직임 벡터에 따른 MAD(0,0)의 분포 특성을 분석하여, 이 MAD(0,0)의 크기에 따라 적응적으로 움직임 벡터를 추정하는 고속 적응 블럭 정합 알고리즘을 제안하였다. 제안한 방법은 NTSS 가제 1단계에서 무조건 17개의 탐색점에 대해서 블럭정합을 수행하기 때문에 계산량이 증가되는 단점을 개선할 수 있는 방법이다. 그림 1에서는 FOOTBALL영상에 대하여 탐색영역이 -7 ~ 7, 블럭크기 8×8에서 전역탐색 알고리즘을 이용하여 얻은 움직임 벡터의 분포를 나타낸 것이다. 이 그림에서 X, Y축은 변위 (displacement), Z 축은 전체영상 60 프레임에서 움직임 벡터의 분포를 백분율(%)로 표시한 것으로서, 움직임 벡터가 (0,0)인 경우는 24.67%, -1 ~ 1일 때는 53.16 %, -2 ~ 2일 때는 61.98 %, -3 ~ 3 인 경우는 65.18 %, 나머지는 34.82 %의 분포를 가지고 있다. 또한 TABLE TENNIS 영상에서 움직임 벡터가 (0,0)인 경우는 57.65%, -1 ~ 1일 때는 65.44 %,

-2 ~ 2일 때는 74.92 %, -3 ~ 3 인 경우는 79.34 %, 나머지는 20.66 %의 분포를 가지고 있다. 따라서 이들 영상에서 전체의 움직임 벡터 중 65.18 ~ 79.34 %가 -3 ~ 3사이에 존재함을 알 수 있다.

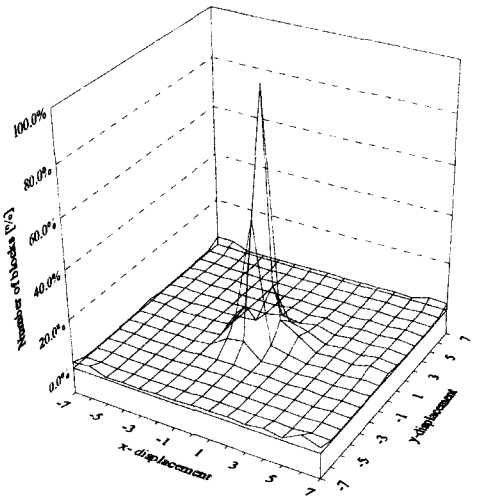

그림 1. 전역탐색 알고리즘을 이용한 FOOTBALL 영상의 움직임 벡터의 분포

Fig. 1. The distribution of motion vector using full search algorithm for FOOTBALL image.

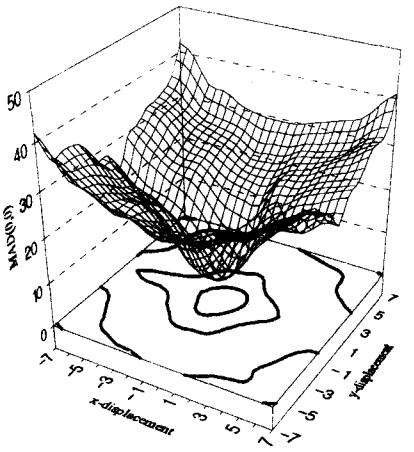

그림 2. FOOTBALL 영상에서 움직임 벡터 따른 MAD(0,0) 값의 크기

Fig. 2. The degree of MAD(0,0) values for motion vector of FOOTBALL image.

그림 2에서는 FOOTBALL영상에 대하여 전역탐색 알고리즘을 이용하여 움직임 벡터를 얻었으며, 이 움직임 벡터에 따른 MAD(0,0) 값의 크기 정도를 나타낸 것이다. 이 그림에서 X, Y축은 변위, Z 축은 움직임 벡터에 따른 MAD(0,0)의 평균을 표시한 것이며, 바닥은 움직임 벡터에 따른 MAD(0,0)의 분포를 등고선으로 나타낸 것으로, 중심점으로 부터 등고선의 표시가 $0 \le MAD(0,0) < 10$, $10 \le MAD(0,0) < 20$, $20 \le MAD(0,0) < 30$ 및 $30 \le MAD(0,0)$이다. 이 그림에서 MAD(0,0)가 작을수록 움직임 벡터는 거의 (0,0)에 집중 혹은 움직임 벡터가 적고, MAD(0,0)가 서서히 증가하면서 움직임 벡터도 (0,0)에서 점차로 증가한다.

움직임 벡터에 따른 MAD(0,0) 값의 크기를 근거로 하여 제안한 고속 적응 블럭 정합 알고리즘의 흐름도는 그림 3과 같다. 이 그림에서 움직임 정도의 척도는 MAD(0,0)를 이용한다. $MAD(0,0) < th_1$이면 움직임이 거의 없는 블럭, $th_1 \le MAD(0,0) < th_2$ 이면 움직임이 적은 블럭, $th_2 \le MAD(0,0) < th_3$ 이면 중간정도인 블럭, 이 외는 움직임이 큰 블럭으로 분류한 후, 각 블럭에 대해서 적응적으로 블럭 정합 알고리즘을 사용하여 움직임 벡터를 추정한다. 여기서, th_1 , th_2 및 th_3는 문턱값이다.

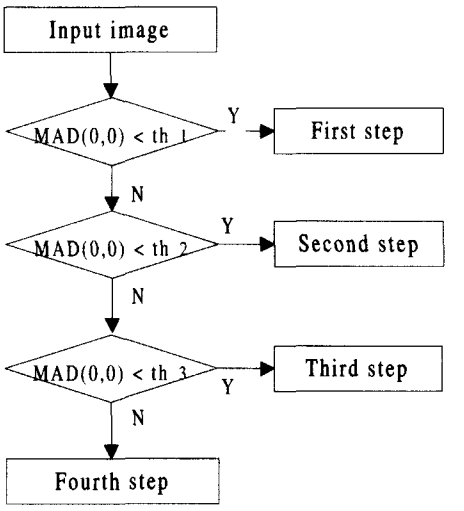

그림 3. 제안한 고속 적응 블럭 정합 알고리즘 블록선도

Fig. 3. The block diagram of the proposed fast adaptive BMA.

그림 3에서 제 1, 2, 3, 및 4단계는 다음과 같다.

1. 제 1단계: $MAD(0,0) < th_1$ 인 경우이며, 이는 움직임이 거의 없는 블럭으로 (0,0)을 움직임벡터로 결정한다.
2. 제 2단계: $th_1 \le MAD(0,0) < th_2$ 경우이며, 이는

움직임이 적은 블럭으로 그림 4와 같이 검은색 원점인 MAD(0,0)를 중심으로 인근의 8개의 탐색점(검은 네모점)을 이용하여 블럭정합을 하여 가장 작은 MAD를 갖는 탐색점 (흰 삼각형)을 찾는다. 그리고 이 탐색점을 중심으로 검은 삼각형인 인근의 8개에 대하여 한번 더 블럭정합을 수행한다. 이와 같이 블럭정합을 하여 가장 작은 MAD를 갖는 탐색점을 움직임 벡터로 결정한다.

3. 제 3단계: $th_2 \le MAD(0,0) < th_3$ 인 경우이며, 이는 움직임이 중간정도의 블럭으로 그림 5와 같이 검은색 원점인 MAD(0,0)를 중심으로 검은색 네모점 8개의 탐색점에 대하여 블럭정합을 수행한다. 이 중에서 가장 작은 MAD를 갖는 탐색점을 기준으로하여, 이 탐색점 주위 검은점 삼각형인 인근의 8개에 대하여 또다시 블럭정합을 수행하여, 가장 작은 MAD를 갖는 탐색점을 움직임 벡터로 결정한다.
4. 제 4단계: $MAD(0,0) \ge th_3$ 인 경우이며, 이는 움직임 큰 영역으로서 탐색점의 수를 많이하여 움직임 벡터를 결정하여야 한다. 따라서 이를 위해서 움직임 벡터 결정하기 위해서 첫 번째 단계에서 탐색점을 17개 선택하는 NTSS방법을 이용한다.

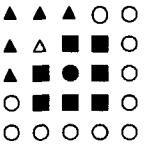

그림 4. 제안 방법의 제 2단계
Fig. 4. The second step of the proposed method.

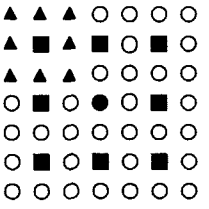

그림 5. 제안 방법의 제 3단계
Fig. 5. The third step of the proposed method.

먼저, $MAD(0,0) < th_1$인 블럭은 움직임이 거의 없는 블럭으로 제안방법의 제 1 단계를 수행한다. 이 블럭은 동영상에서 배경 (background) 및 움직임이 적은 전경 (foreground)으로 구성된다. 배경 영역은 동영상 중에서 대부분 차지하고 있으며, 이 영역에서는 프레임이 변화되어도 움직임이 거의 발생되지 않는다. 또한 전경부분 중에서 프레임이 변해도 움직임의 변화가 거의 없는 영역들이 많이 포함되어 있다. 이와 같이 움직임의 정도가 거의 없는 영역에서는 탐색점 전부에 대하여 탐색할 필요가 없다. 그러므로 이 영역에서는 움직임이 없는 영 움직임 벡터 (수평방향으로의 움직임 0, 수직방향으로의 움직임 0)로 정하게 되어 움직임 벡터를 추출하는데 사용되는 계산량이 필요없다. 블럭크기가 8×8 및 16×16이며 탐색영역이 -7~7인 경우에서 전 탐색 블럭 정합 알고리즘에서는 225개, NTSS 방법의 제 1단계에서는 17개, TSS 방법의 제 1단계에서는 9개의 탐색점에 따른 계산량을 필요로 하기 때문에 제안방법에서는 계산량이 매우 감소된다. 또한 PSNR 성능면에서도 배경 혹은 움직임의 변화가 거의 없기 때문에 오차가 거의 차이 나지 않는다.

$th_1 \le MAD(0,0) < th_2$인 블럭은 움직임이 적은 블럭으로 제안방법의 제 2단계를 수행한다. 이 경우는 탐색점을 먼저 9개를 선택하고, 최소 MAD가 되는 탐색점 인근의 8개 탐색점에 대하여 블럭정합을 한다. 이 방법에서의 계산량은 첫 번째의 탐색점 9개와 두 번째의 탐색점이 (-1,-1), (-1,1), (1,-1), (1,1)에서는 계산량이 5개, (-1,0), (1,0), (0, -1), (0,1)에서는 3개의 탐색점의 계산량이 필요로 하므로 계산량이 많이 줄어든다.

$th_2 \le MAD(0,0) < th_3$ 인 블럭은 움직임이 중간 정도로 제안방법의 제 3단계를 수행한다. 이 경우에서는 첫 번째 탐색으로 9개의 탐색점을 필요로하고, 두 번째 탐색에서는 8개의 탐색점을 필요로한다. 따라서 총 17개의 탐색점에 대하여 블럭 정합을 행한다.

$MAD(0,0) \ge th_3$ 인 블럭은 움직임이 큰 블럭으로 제안방법의 제 4단계를 수행한다. 이 경우에서는 첫 번째 단계에서 많은 탐색점을 필요로 하는 기존의 NTSS 방법을 수행하기 때문에 첫 번째 탐색은 17개로 하여 블럭정합을 수행하여 MAD의 값이 최소인 탐색점이 MAD(0,0)의 인근의 8개에 위치할 경우는 $th_1 \le MAD(0,0) < th_2$ 인 블럭에서와 같이 탐색점이 어느 위치에 있는가에 따라 5개 혹은 3개가 된다. 그러나 이 외의 탐색점에서 MAD가 최소일 경우는 3단계 탐색 알고리즘과 같이 8개, 8개 수행하여 총 33개의 탐색점에 대하여 블럭정합을 행하여 움직임 벡터를 결정하기 때문에 많은 계산량을 필요로 한다.

Ⅲ. 실험 결과 및 고찰

본 논문에서 제안한 방법에 대하여 성능을 평가하기 위하여 컴퓨터 모의 실험을 수행하였다. 본 실험에서는 720×480 크기의 TABLE TENNIS 및 FOOTBALL 각 30, 60 프레임을 사용하였다. FOOTBALL은 카메라 및 물체의 움직임 빠른 영상이며, TABLE TENNIS은 정지된 카메라를 이용하여 많은 배경을 가지면서 줌아우트(zoom out) 하는 영상으로 비교적 움직임이 적은 영상이다. 본 실험에서는 블럭 크기를 8×8 및 16×16, 탐색범위 -7~7로, 정합의 척도는 평균절대오차를 사용하여 계산량 및 PSNR

$$PSNR = 10 \log \frac{255^2}{\sigma_e^2} \quad [\text{dB}]$$

에 대하여 평가하였다. 여기서, σ_e^2은 원 영상과 움직임 보상된 영상의 평균 자승 오차이다.

또한 움직임의 정도에 대한 문턱값은 움직임 벡터에 따른 MAD(0,0)의 분포 특성을 이용하여 결정하였다. 제안 방법에서는 MAD(0,0)에 따라 적응적으로 블럭 정합 알고리즘을 수행하기 위한 문턱값 th_1, th_2 및 th_3는 영상의 특성에 따라 조금씩 차이가 있다. 시뮬레이션 결과에서 블럭크기 8×8인 경우, FOOTBALL 영상에서 th_1=3.29, th_2=6.94 및 th_3=13.01, TABLE TENNIS 영상에서th_1=8.54, th_2=11.32 및 th_3=12.85 이 거의 최적에 가까운 값이다. 여기서, th_1, th_2 및 th_3는 각각 움직임 벡터가 (0,0), (-1 ~ 1) 및 (-2 ~ 2)일 때의 MAD(0,0)의 평균값이다. 그러나 본 논문에서는 영상에 따라 각각 다른 문턱값을 사용하기보다는 동일한 문턱값을 사용하기 위해서 PSNR 및 계산량을 고려하여 th_1=4.0, 4.5, 5.0 th_2=9.0, 9.5, 10.0 및 th_3=13.0, 14.0, 15.0 등 여러 값을 이용하여 모의 실험한 결과 th_1=4.5, th_2=9.5 및 th_3=13.0에서 가장 좋은 결과를 얻었다.

제안 방법과 기존의 NTSS 방법에 대하여 프레임 변화에 따른 PSNR 결과는 그림 6 및 7에서와 같다. 그리고 각 영상들에 대한 평균 PSNR 결과는 표 1과 같다.

그림 6과 7에서와 같이 두 영상 모두에서 기존의 NTSS 방법과 제안한 방법에서의 PSNR이 거의 비슷하다는 것을 알 수 있다. 또한 표 1에서와 같이 각 블럭크기에 대하여 기존의 NTSS 방법과 거의 같은 PSNR을 유지한다는 것을 알 수 있다. 이 결과로부터 MAD(0,0)의 크기에 따라 적응적으로 블럭 정합 알고리즘을 수행하여 움직임 벡터를 결정하더라도 PSNR에는 영향을 거의 미치지 않음을 알 수 있다. 기존의 NTSS 및 제안 방법에 대한 계산량을 비교한 결과는 표 2와 같다. 이 때 이 계산량은 움직임 벡터를 추정하기 위해서 요구되는 탐색점의 수를 나타낸 것이다.

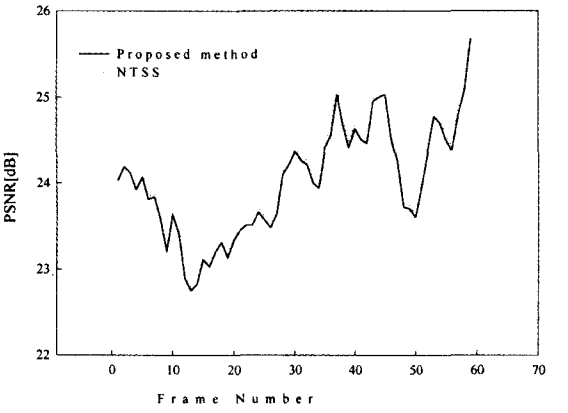

그림 6. FOOTBALL 영상의 블럭크기 8×8 (탐색범위: -7~7)에 대한 PSNR

Fig. 6. PSNR of block size 8×8 (search area: -7~7) for FOOTBALL image.

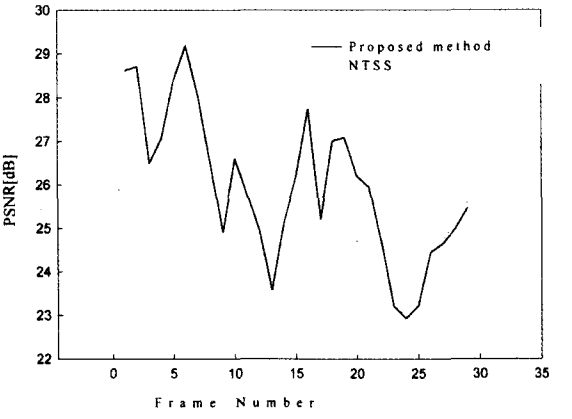

그림 7. TABLE TENNIS 영상의 블럭크기 8×8 (탐색범위: -7~7)에 대한 PSNR

Fig. 7. PSNR of block size 8×8 (search area: -7~7) for TABLE TENNIS image.

이 표에서 보는바와 같이 블럭 크기 8×8 및 16×16 인 경우 기존의 NTSS 방법에 비하여 제안방법의 계산량이 FOOTBALL 영상에서는 각각 40.27% 및 36.64% 감소되었으며, TABLE TENNIS 영상에서

각각 32.77%, 30.44% 감소되었다. 이와 같이 계산량이 감소되는 이유는 대부분의 영상에서는 움직임이 없는 배경부분 및 움직임이 적은 블럭 즉, FOOTBALL 영상에서는 65.18 %, TABLE TENNIS 영상에서는 79.34 % 존재하기에 움직임의 정도에 따라 적응적으로 움직임 벡터를 추정하기 때문이다.

표 1. 각 방법에 대한 평균 PSNR [dB]
Table 1. Average PSNR [dB] for each method.

Block size	Image	Proposed method	NTSS
8×8	FOOTBALL	24.02	24.06
	TABLE TENNIS	25.97	26.00
16×16	FOOTBALL	22.80	22.83
	TABLE TENNIS	25.51	25.53

표 2. 각 방법에 대한 계산량 [%]
Table 2. The computation amount [%] for each method.

Block size	Image	Proposed method	NTSS
8×8	FOOTBALL	59.73	100.00
	TABLE TENNIS	67.23	100.00
16×16	FOOTBALL	63.36	100.00
	TABLE TENNIS	69.56	100.00

이상의 결과로부터 제안 방법이 블럭 정합 알고리즘 방법중 성능이 우수한 NTSS 방법과 비교해서 PSNR은 비슷하면서도 계산량을 많이 줄여서 고속으로 움직임 벡터를 추정할 수 있는 효과적인 방법임을 확인할 수 있었다.

Ⅳ. 결 론

본 논문에서는 움직임 벡터의 분포특성을 이용한 고속 적응 블럭 정합 알고리즘을 제안하였다. 제안 방법에서는 움직임 벡터에 따른 MAD(0,0)의 분포특성을 분석하여 블럭을 움직임이 없는 블럭, 작은 블럭, 중간 정도인 블럭 혹은 큰 블럭으로 분류한 후 각 블럭의 특성에 따라 적응적으로 움직임 벡터를 추정한다.

제안한 방법의 성능을 평가하기 위해서 컴퓨터 시뮬레이션을 수행하였다. 이 결과로부터 제안 방법의 PSNR은 기존의 NTSS 방법과 거의 비슷하면서도 계산량이 30.44% ~ 40.27% 감소되는 효과적인 방법임을 확인할 수 있었다.

참 고 문 헌

[1] H. G. Musmann, P. Pirch and H. J. Grallert, "Advances in picture coding," *Proc. of IEEE*, vol. 73, no. 4, pp. 523-548, Apr. 1985.

[2] A. K. Jain, "Image data compression: A review," *Proc. of IEEE*, vol. 69, no. 3, pp. 349-389, Mar. 1981.

[3] A. M. Tekalp, Digital video processing, Prentice Hall, 1995.

[4] M. Ghanbari, "The cross search algorithm for motion estimation," *IEEE Trans. on Commun.*, vol. COM-38, no. 7, pp. 950-953, July 1990.

[5] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 6, pp. 148-157, Apr 1993.

[6] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuit and Systems for Video Technology*, vol. 4, no. 4, pp. 438-442, Aug. 1994.

저 자 소 개

申 容 達(正會員) 第 34卷 S編 第 9號 參照
현재 영동대학교 전자공학부 조교수

金 榮 春(正會員) 第 33卷 B編 第 6號 參照
현재 영동대학교 전자공학부 전임강사