論文96-33B-8-9

저해상도 양자화된 이미지를 이용하여 연산량을 줄인 움직임 추정 기법

(A Motion Estimation Algorithm with Low Computational Cost Using Low-resolution Quantized Image)

李 誠 洙 *, 蔡 洙 翊 *

(Seongsoo Lee and Soo-Ik Chae)

요 약

본 논문에서는 완전탐색 알고리듬의 연산량을 줄이기 위해 저해상도 양자화를 사용하는 움직임 추정 기법을 제안한다. 제안하는 알고리듬은 저해상도 이미지를 비교하여 움직임 벡터 후보위치들을 결정하는 저해상도 탐색과, 이들 움직임 벡터 후보위치에 대해서 원해상도 이미지를 비교하여 움직임 벡터를 결정하는 원해상도 탐색으로 구성된다. 저해상도 이미지는 기준블록과 탐색영역의 화소값을 기준블록의 평균값을 뺀 후 2비트로 양자화하여 생성한다. 움직임 벡터 후보위치들은 기준블록과 탐색영역의 양자화된 코드값이 서로 일치하지 않는 화소의 개수를 세어 결정된다. 이 알고리듬은 완전탐색 알고리듬에 비하여 연산량이 1/12로 감소하였고, 성능 저하는 0.03~0.12 dB임을 모의실험을 통하여 확인하였다.

Abstract

In this paper, we propose a motion estimation algorithm using low resolution quantization to reduce the computation of the full search algorithm. The proposed algorithm consists of the low resolution search which determines the candidate motion vectors by comparing the low-resolution image and the full resolution search which determines the motion vector by comparing the full-resolution image on the positions of the candidate motion vectors. The low resolution image is generated by subtracting each pixel value in the reference block or the search window by the mean of the reference block, and by quantizing it in 2 bit resolution. The candidate motion vectors are determined by counting the number of pixels in the reference block whose quantized codes are unmatched to those in the search window. Simulation results show that the required computational cost of the proposed algorithm is reduced to 1/12 of the full search algorithm while its performance degradation is 0.03~0.12 dB.

I. 서 론

동영상은 그 특성상 시간 영역과 공간 영역, 코드들 사이에서 많은 상관관계를 가지므로, 이를 제거하면 압축률을 높일 수 있다. 대부분의 동영상 압축 기법은 시간 영역의 상관관계를 제거하는데 움직임 추정 기법(motion estimation algorithm)[1]을, 공간 영역의 상관관계를 제거하는데 DCT 변환(discrete cosine transform)[2]을, 코드들 사이의 확률적 상관관계를 제거하는데 가변길이 부호화(variable length coding)[3]를 사용한다.

움직임 추정 기법 중 가장 많이 사용되는 블록 정합 알고리듬(block matching algorithm)은 현재 프레임

* 正會員, 서울大學校 電氣工學部 및 半導體 共同硏究所
(School of Electrical Engineering & Inter University Semiconductor Research Center)

接受日字:1996年2月24日, 수정완료일:1996年7月24日

을 여러 개의 기준블록(reference block)으로 분할하여 이전 프레임의 탐색영역(search window) 안에서 가장 닮은 블록을 찾아 기준블록에 대한 상대 위치를 움직임 벡터(motion vector)로 정한 후 두 블록간의 화소값 차이와 움직임 벡터만을 전송하는 기법으로, 국제 표준인 H.261, H.262, H.263등 대부분의 표준 동영상 압축 기법에 채택되었다.

블록 정합 알고리듬 중에서 완전탐색 알고리듬(full search algorithm)[1]은 기준블록을 탐색영역 내의 모든 블록과 비교하는 방법으로 성능이 우수하고 데이터 흐름과 제어회로가 비교적 간단하다는 장점이 있으나, 탐색영역이 커질 경우 막대한 연산량을 필요로 하는 단점이 있다. 완전탐색 알고리듬의 이러한 단점을 보완하기 위해 많은 고속탐색 알고리듬(fast algorithm)이 제안되었다[4-8]. 이들 알고리듬은 탐색영역 내의 여러 블록 중 일부분에 대해서만 비교하거나[4-6] 기준블록의 여러 화소 중 일부만을 비교하기 때문에[7-8] 연산량은 줄일 수 있으나 완전탐색 알고리듬에 비해 성능이 떨어지고 제어회로가 복잡해지는 단점이 있다.

본 논문에서는 탐색영역 내의 모든 블록에 대해서 모든 화소를 사용하여 비교함으로서 고속탐색 알고리듬의 단점인 성능저하를 줄이고, 화소값을 저해상도 양자화하여 비교함으로서 필요한 연산량을 크게 줄이는 저해상도 양자화 움직임 추정 기법을 제안한다. II장에서는 블록 정합 알고리듬 중 완전탐색 알고리듬과 고속탐색 알고리듬에 대하여 고찰하고 III장에서는 제안하는 저해상도 양자화를 이용한 움직임 추정 알고리듬에 대하여 설명한다. IV장에서는 기존의 블록 정합 알고리듬과 제안하는 알고리듬의 연산량을 계산하고 V장에서는 모의실험을 통하여 성능을 비교하며, VI장에서는 결론을 맺는다.

II. 블록 정합 알고리듬

블록 정합 알고리듬은 현재 프레임을 N×M 크기를 갖는 여러 개의 작은 블록(이하 기준블록이라 칭함)으로 나눈 다음, 각각의 기준블록을 이전 프레임의 정해진 영역(기준블록이 상하좌우로 -p에서 +(p-1) 화소만큼 이동한 영역, 이하 탐색영역이라 칭함)내에 있는 여러 개의 블록과 비교하여 가장 닮은 블록(이하 정합블록이라 칭함)을 찾아내어, 기준블록에 대한 정합블록의 상대 위치를 움직임 벡터로 정하는 기법이다(그림 1). 기준 블록과 '가장 닮은' 정합블록을 찾아내는 판단기준은 차의 제곱 합(SSD : sum of squared difference)과 차의 절대값 합(SAD : sum of absolute difference) 등을 들 수 있는데, 대부분의 블록 정합 알고리듬에서는 곱셈기를 사용하지 않고 간단한 하드웨어로 구현할 수 있는 차의 절대값 합을 주로 사용한다. 차의 절대값 합은 수식 (1)과 같이 나타낼 수 있다.

$$SAD(u,v) = \sum_{i=1}^{N}\sum_{j=1}^{M} | R(i,j) - S(i+u, j+v) |, \quad -p \le u, v < p \tag{1}$$

수식 (1)에서 기준블록의 크기는 N×M, R(i,j)과 S(i,j)는 각각 기준블록과 탐색영역의 (i,j)번째 위치의 화소값을 나타내며, 움직임 벡터는 SAD(u,v)가 최소가 되는 (u,v) 좌표로 정의한다.

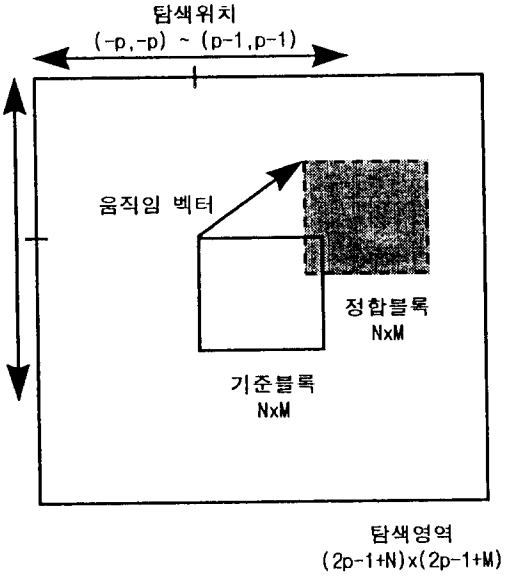

그림 1. 블록 정합 알고리듬의 개념도
Fig. 1. Conceptual diagram of the block matching algorith.

블록 정합 알고리듬 중에서 완전탐색 알고리듬은, 근래에 들어와서 압축해야 할 동영상의 화면 크기 및 프레임 전송속도(frame rate)가 커짐에 따라 필요한 연산량이 매우 커지는 문제점을 가지고 있다. 그림 1과 같이 움직임 벡터의 범위가 x,y방향 모두 -p에서 +(p-1)일 때, 기준블록 하나에 대해서 완전탐색 알고리듬을 수행하려면 $4p^2$번의 차의 절대값 합 계산이 필

요하므로, 하나의 프레임이 n개의 기준블록으로 구성되고 프레임 전송속도가 f 프레임/초 일 때 차의 절대값 합 계산을 초당 $4p^2\times n\times f$번 수행하여야 한다. 기준블록의 크기가 N×M 화소일 때 차의 절대값 합 계산은 각각 N×M번씩의 뺄셈과 절대값 계산, 누산으로 구성되므로, 완전탐색 알고리듬의 연산량은 뺄셈과 절대값 계산, 누산이 각각 $4p^2\times n\times f\times N\times M$번이다. 화면 크기가 720×480 화소, 프레임 전송속도가 30 프레임/초, 기준블록의 크기가 16×16 화소이고 순방향 예측(forward prediction)만을 사용하는 H.262 (MPEG2) MP@ML P-픽처에 대해 움직임 벡터의 범위가 x,y방향 모두 -32에서 31까지인 완전탐색 알고리듬을 수행한다면 이때 필요한 연산량은 뺄셈과 절대값 계산, 누산이 각각 초당 4.25×10^{10}번이다.

완전탐색 알고리듬의 연산량을 줄이기 위한 고속탐색 알고리듬은 크게 선택적 탐색 알고리듬(selective search algorithm)과 화소 부표본화 알고리듬(pixel subsampling algorithm)으로 나눌 수 있다. 선택적 탐색 알고리듬은 정합블록을 찾아내는 기준인 차의 절대값 합이 최적의 움직임 벡터에서 멀어질수록 단조 증가한다는 가정 아래, 첫 단계에서 여러 개의 탐색위치에 대해 차의 절대값 합을 계산한 다음, 차의 절대값 합이 가장 작은 탐색위치를 시작점으로 하여 다음 단계의 탐색위치를 결정하는 방법이다. 그러나 실제 영상에서는 이러한 가정이 항상 성립하지는 않으므로 국부 극소에 빠질 가능성이 크고, 이에 따른 성능 저하가 발생하게 된다.

화소 부표본화 알고리듬은 블록의 모든 화소가 동일한 거리만큼 움직였다는 가정 아래, 기준블록의 여러 화소 중 일부만을 사용하여 차의 절대값 합을 계산한 후 이를 비교하여 움직임 벡터를 찾아내는 방법이다. 화소 부표본화 알고리듬은 탐색영역 내의 모든 탐색위치를 탐색하면 국부 극소에 빠질 가능성은 적으나, 기준블록의 여러 화소중 일부분만을 비교하기 때문에 수직선이나 수평선 등이 있을 경우 성능 저하가 발생하게 된다.

III. 저해상도 양자화 움직임 추정 알고리듬

1. 기본 개념

선택적 탐색 알고리듬은 국부 극소에 의한 성능 저하가 있고, 화소 부표본화 알고리듬은 수직선, 수평선 등에 의한 성능 저하가 존재한다. 또한 완전탐색 알고리듬은 필요한 연산량이 매우 크다. 이러한 문제점을 개선하기 위해서는 모든 탐색위치에 대해서 모든 화소를 사용하여 비교하되, 각 화소를 비교하는데 필요한 연산량을 줄이는 것이 바람직하다.

비교하려는 화소값이 8비트 해상도보다 낮은 비트 해상도를 가진다면 연산의 숫자는 줄지 않더라도 연산에 필요한 하드웨어가 작아지게 된다. 따라서 화소값을 양자화하여 비트 해상도를 낮춘 화소들을 비교하여 움직임 추정 알고리듬을 수행한다면 기존의 원해상도 완전탐색 알고리듬보다 훨씬 작은 하드웨어로 저해상도 완전탐색이 가능해진다.

그러나, 화소값의 비트 해상도를 낮춘다는 것은 곧 화소값이 가지고 있는 정보를 잃어버린다는 것을 의미하므로 이에 따른 성능 저하는 피할 수 없게 된다. 본 논문에서 제안하는 저해상도 양자화 움직임 추정 기법은 이러한 문제를 해결하기 위하여 먼저 양자화로 비트 해상도를 낮춘 저해상도 이미지(low resolution image)를 비교하여 복수개의 움직임 벡터 후보위치(candidate motion vector)를 결정한 후, 이들 움직임 벡터 후보위치에 대해서 원해상도 이미지(full resolution image)를 비교하여 움직임 벡터를 결정함으로써 성능 저하를 줄인다. 저해상도 양자화 움직임 추정 기법은 그림 2와 같이 저해상도 탐색(low resolution search)과 원해상도 탐색(full resolution search)으로 구성된다.

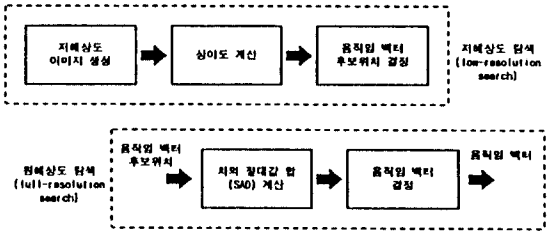

그림 2. 저해상도 양자화 움직임 추정 기법의 블록도.

Fig. 2. The block diagram of the low resolution quantization motion estimation algorithm.

저해상도 탐색은 기준블록과 탐색영역 내의 모든 블록의 저해상도 이미지를 비교하여 움직임 벡터 후보위치를 결정하는 일종의 완전탐색 알고리듬으로, 양자화에 필요한 파라미터를 결정하고 저해상도 이미지를 생

성하는 단계와 움직임 벡터 후보위치를 결정하기 위한 판단기준인 상이도(dissimilarity measure)를 계산하는 단계, 계산된 상이도를 비교하여 움직임 벡터 후보위치를 결정하는 단계로 구성된다. 원해상도 탐색은 일종의 선택적 탐색 알고리듬으로, 저해상도 탐색에서 결정된 움직임 벡터 후보위치에 대해서 기존의 블록 정합 알고리듬처럼 차의 절대값 합을 계산하여 움직임 벡터를 결정한다.

2. 저해상도 이미지 생성

각 화소가 가지고 있는 정보를 가능한 한 잃어버리지 않고 양자화 시키려면 각 화소를 그대로 양자화하는 것보다는 화소의 평균값을 뺀 후 양자화하는 것이 유리하다. 블록 정합 알고리듬의 목적은 기준블록과 가장 닮은 블록을 탐색영역 내에서 찾아내는 데 있으므로, 블록 정합 알고리듬을 수행하여 최종적으로 찾아낸 정합블록의 평균값과 기준블록의 평균값은 비슷하다고 가정할 수 있다. 탐색영역의 평균보다는 기준블록의 평균이 계산하기 쉽기 때문에, 기준블록과 탐색영역의 화소값 모두 기준블록의 평균을 뺀 후 양자화한다.

저해상도 이미지의 비트 해상도가 너무 낮으면 양자화로 인해 잃어버리는 정보의 양이 너무 많아서 큰 성능 저하를 초래하고, 비트 해상도가 너무 높으면 성능 개선에 비해 연산량 증가가 훨씬 크다. 비트 해상도를 1비트에서 4비트까지 변화시키며 모의실험을 수행하였을 때의 PSNR 및 연산량은 그림 3과 같으며, PSNR 및 연산량을 고려할 때의 최적 비트 해상도는 2비트임을 알 수 있다.

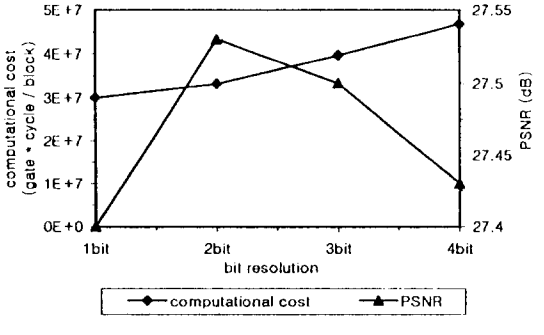

그림 3. 비트 해상도에 따른 PSNR 및 연산량
Fig. 3. The PSNR and the computational cost vs. bitresolution.

저해상도 탐색에서 생기는 성능 저하는 양자화로 인한 화소값 정보 손실에서 기인하므로, 성능저하를 줄이려면 양자화 오차(quantization error)가 최소가 되도록 양자화 경계(quantization threshold)를 각 기준블록 마다 적응적으로 정해야 한다. 양자화 오차를 계산하는 방법으로는 오차의 제곱의 평균(mean squared error)과 오차의 절대값의 평균(mean absolute error) 등을 들 수 있는데 본 논문에서는 곱셈기 없이 간단한 하드웨어만으로 구현할 수 있는 오차의 절대값의 평균을 사용하였다. 이때 기준블록의 양자화 오차 $\overline{E_q}$를 나타내면 수식 (2)와 같다.

$$\overline{E_q} = \frac{1}{N \times M} \sum_{i=1}^{L} \sum_{R(x,y) \in X_i} |R(x,y) - RR(x,y)| \qquad (2)$$

수식 (2)에서 기준블록의 크기는 N×M이며, R(x,y), RR(x,y)는 각각 기준블록의 (x,y)번째 위치의 화소값과 복원화소값(reconstructed value of quantization)을, L은 양자화 구간의 개수(저해상도 이미지의 비트 해상도가 k비트일 때 $L=2^k$), X_i는 i번째 양자화 구간을 나타낸다.

비선형 양자화 이론(nonlinear quantization theory)에 따르면 양자화 오차를 최소로 하는 양자화 경계는 수식 (3)에서 구할 수 있으며[19], 2비트 양자화의 경우 4개의 양자화된 코드의 복원화소값의 비를 각각 -3 : -1 : 1 : 3으로 놓고 수식 (3)을 풀면 세개의 양자화 경계 t_1, t_2, t_3는 수식 (4)와 같이 t_1, t_3은 기준블록의 평균편차 σ_m의 3/2, 3/2배로, t_2는 0으로 주어진다.

$$\frac{\partial \overline{E_q}}{\partial t_i} = 0, \quad \frac{\partial \overline{E_q}}{\partial r_j} = 0 \text{ for } i = 1..L-1, \; j = 1..L \qquad (3)$$

$$t_3 = -t_1 = \frac{3}{2}\sigma_m, \quad t_2 = 0$$
$$\sigma_m = \sum_{i=1}^{N} \sum_{j=1}^{M} |R(i,j) - \bar{R}| \qquad (4)$$

수식 (3),(4)에서 기준블록의 크기는 N×M이고, t_i는 i번째 양자화 경계를, r_i는 i번째 양자화 구간에서의 복원화소값을, L은 양자화 구간의 개수를, R(i,j)는 기준블록의 (i,j)번째 화소값을, $\bar{R}$은 기준블록의 화소 평균값을 나타낸다. 수식 (4)는 덧셈, 뺄셈, 절대값 계산, 쉬프트만으로 계산할 수 있으므로, 양자화 경계를 결정하는 하드웨어는 비교적 쉽게 구현할 수 있다. 평균값 계산 때와 마찬가지로 블록 정합 알고리듬에서 최종적으로 찾아낸 정합블록의 화소값은 기준블록과

같은 확률분포를 가진다고 가정할 수 있으므로 본 논문에서는 기준블록의 화소값에서 구한 수식 (3)의 양자화 경계를 기준블록과 탐색영역 모두의 양자화 경계로 정한다. 기준블록과 탐색영역의 화소값을 양자화하여 저해상도 이미지를 생성하는 과정은 그림 4와 같다.

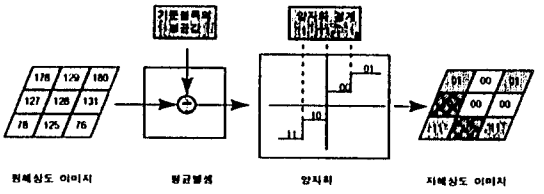

그림 4. 저해상도 이미지 생성 과정
Fig. 4. The generation of low-resolution image.

3. 상이도 결정

저해상도 탐색에서 움직임 벡터 후보위치를 결정하기 위한 상이도는 여러 가지로 정의할 수 있으나, 본 논문에서는 하드웨어 구현을 간단하게 하기 위해서 양자화된 코드값이 서로 일치하지 않는 화소의 숫자를 상이 화소수(DPC: different pixel count)로 정의하고 이를 상이도로 사용한다. 상이 화소수의 정의는 수식 (5)와 같다.

$$DPC(u,v) = \sum_{i=1}^{N}\sum_{j=1}^{M} \overline{\delta}\,[RL(i,j), SL(i+u, j+v)], \quad -p \le u, v < p \tag{5}$$

수식 (5)에서 기준블록의 크기는 N×M, RL(i,j), SL(i,j)는 각각 기준블록과 탐색영역내의 (i,j)번째 화소값의 양자화된 코드이며 $\overline{\delta}(x,y) = 1\ (x \neq y$ 일때)로 주어진다. $0\ (x = y$ 일때)

4. 움직임 벡터 후보위치 및 움직임 벡터 결정

본 논문에서 제안하는 알고리듬은 저해상도 탐색으로 찾은 움직임 벡터를 복수개의 움직임 벡터 후보위치로 정한 후, 각 움직임 벡터 후보위치에서 원해상도 탐색을 실시함으로서 움직임 벡터를 찾는다. 움직임 벡터의 후보위치 개수는 많을수록 성능은 높아지지만 연산량도 따라서 증가하므로 적절한 개수를 선택하여야 한다.

움직임 벡터 후보위치를 구하는 방법은 매 탐색위치열(row of search position, 움직임 벡터의 y값이 동일한 탐색위치)에 대해 상이도를 구한 후, 상이도가 가장 작은 순서대로 몇개의 위치를 움직임 벡터 후보위치로 정한다. 움직임 벡터 후보위치를 매 탐색위치열마다 구하는 이유는 제안하는 알고리듬을 하드웨어로 구현할 때 저해상도 탐색과 원해상도 탐색을 파이프라이닝하도록 하기 위해서이다. 저해상도 탐색부에서 하나의 탐색위치열을 처리하고 나서 움직임벡터 후보위치를 구하면 원해상도 탐색부에서는 이 움직임벡터 후보위치 각각에 대해 차의 절대값 합을 계산하게 되고, 이 동안 저해상도 탐색부에서는 다음 탐색위치열을 처리할 수 있게 된다. 탐색영역이 ±32×±32일 때 탐색영역의 매 탐색위치열 (64 탐색위치)에서 결정되는 움직임벡터 후보위치의 개수를 1개에서 4개까지 변화시키며 모의실험을 수행하였을 때의 PSNR 및 연산량은 그림 5와 같으며, PSNR 및 연산량을 고려할 때의 최적 움직임벡터 후보위치 개수는 매 탐색위치열 당 2개임을 알 수 있다.

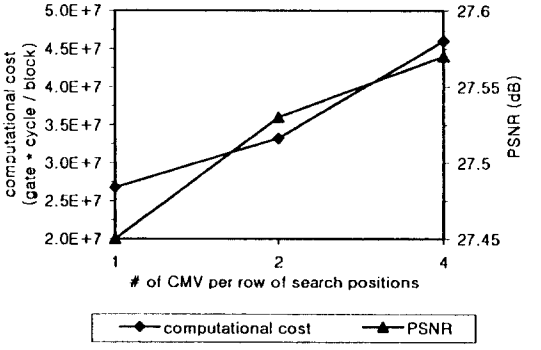

그림 5. 탐색위치열 당 움직임 벡터 후보위치 개수에 따른 PSNR 및 연산량
Fig. 5. The PSNR and the computational cost vs. the number of CMV per row of search positions.

Ⅳ. 움직임 추정 기법의 연산량 계산

블록 정합 알고리듬을 수행하는 움직임 추정기의 복잡도에서 단위처리기(processing element)들이 차지하는 비중은 제어회로를 제외하고는 매우 크며, 단위처리기 전체의 복잡도는 단위처리기 개수와 각 단위처리기의 복잡도의 곱으로 나타낼 수 있다. 움직임 추정기의 처리능력(throughput)은 기준블록 하나를 처리하는데 걸리는 싸이클 수에 반비례하므로, 본 논문에서는 동일한 처리능력을 갖는 움직임 추정기의 복잡도를 비교하기 위해서 단위처리기 전체의 복잡도(게이트 수)에 기준블록 당 소요되는 싸이클 수를 곱한 값을 단위처리기 총복잡도(total complexity of processing ele-

ments)로 정의하고, 이를 각 움직임 추정 알고리듬의 연산량을 비교하는 척도로 삼는다.

정합블록을 결정하는 판단 기준으로 차의 절대값 합을 사용하는 기존의 블록 정합 알고리듬은 기준블록의 크기가 16×16 화소일 때 8비트 뺄셈기 한 개, 8비트 절대값 계산기 한 개, 16비트 누산기 한 개로 이루어지는 단위처리기를 사용한다. 이에 비해서 제안하는 알고리듬의 저해상도 탐색은 움직임 벡터 후보위치를 결정하는 판단 기준으로 상이 화소수를 사용하기 때문에 덧셈기 대신에 간단한 로직 게이트로 구현할 수 있다. 더욱이 여러 화소를 한꺼번에 처리하는 단위처리기를 그림 6에서처럼 낮은 비트수의 덧셈기 트리(adder tree)로 구현할 수 있어서 매 화소를 처리할 때마다 필요했던 누산기를 하나로 줄일 수 있어 단위처리기 총 복잡도를 크게 줄일 수 있다. 원해상도 탐색은 기존의 블록 정합 알고리듬과 동일한 구조의 하드웨어로 구현되지만, 처리해야 할 움직임 벡터 후보 위치의 수가 작기 때문에 기존의 블록 정합 알고리듬에 비해 필요한 단위처리기의 숫자가 작아 단위처리기 총복잡도는 역시 줄어든다. 다만 제안하는 알고리듬은 저해상도 이미지를 생성하기 위해서 기준블록의 평균값을 계산하는 덧셈기와 그림 7과 같은 양자화기가 필요하게 된다.

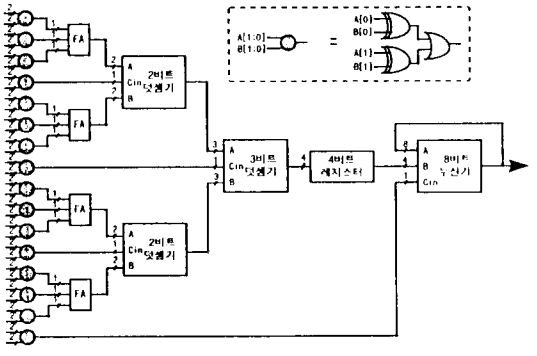

그림 6. 저해상도 탐색에서 사용되는 단위처리기
Fig. 6. The processing element in the low-resolution search.

완전탐색 알고리듬과 4:1 교번 부표본화 알고리듬(4:1 alternate subsampling)[7], 1차원 전역탐색 알고리듬(one-dimensional full search)[5]과 제안하는 알고리듬에 대하여 단위처리기 총복잡도를 계산한 결과는 표 1과 같다. 제안하는 알고리듬의 단위처리기 총복잡도에서 단위처리기 전체의 게이트 수 합에는 기준블록의 평균을 계산하는 하드웨어와 양자화기가 포함되었다. 표 1에서 알 수 있듯이 제안하는 알고리듬의 단위처리기 총복잡도는 완전탐색 알고리듬의 1/12로 감소하였다.

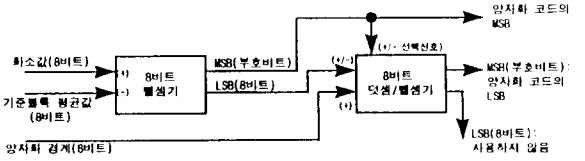

그림 7. 저해상도 탐색에서 사용되는 양자화기
Fig. 7. The quantizer in the low-resolution search.

표 1. 단위처리기 총복잡도
Table 1. Total complexity of processing elements.

블록 정합 알고리듬	기준블록 당 처리시간 (싸이클 / 기준블록)	단위처리기 전체의 게이트 수 추산량 (게이트)	단위처리기 총복잡도 (게이트 · 싸이클 / 기준블록)
완전탐색	1,024	4.00×10^{5}	4.10×10^{8}
4:1 교번 부표본화	1,028	1.00×10^{5}	1.03×10^{8}
1차원 전역탐색	1,920	1.25×10^{4}	2.40×10^{7}
제안하는 알고리듬	1,024	3.24×10^{4}	3.31×10^{7}

V. 모의실험 결과

본 논문에서는 여러 가지 블록 정합 알고리듬의 성능을 비교하는 척도로 원래 영상에 대한 복원 영상의 평균 PSNR(peak signal-to-noise ratio)를 사용하였다. 모의실험에 사용된 영상은 CCIR601 표준영상(704×480 화소, 8비트/화소, 30프레임/초)인 "football", "table tennis", "popple", "flower garden"이며 각각 40 프레임씩에 대하여 수행되었다. 움직임 벡터의 범위는 (-32, -32) ~ (31,31), 저해상도 탐색에서 저해상도 이미지의 비트해상도는 2비트, 움직임 벡터 후보위치 숫자는 매 탐색위치열 당 2개로 하였다. 완전탐색 알고리듬과 4:1 교번 부표본화 알고리듬, 1차원 전역탐색 알고리듬과 제안하는 알고리듬에 대하여 평균 PSNR을 구한 결과는 표 2 및 그림 8,9와 같다.

표 2. 평균 PSNR
Table 2. The average PSNR.

블록 정합 알고리듬	사용 영상별 PSNR (dB)			
	football	table tennis	popple	flower garden
완전탐색	25.11	28.26	30.01	27.03
4:1 교번 부표본화	24.65	28.04	29.77	26.96
1차원 전역탐색	23.67	27.29	29.76	26.66
제안하는 알고리듬	24.99	28.23	29.92	26.97

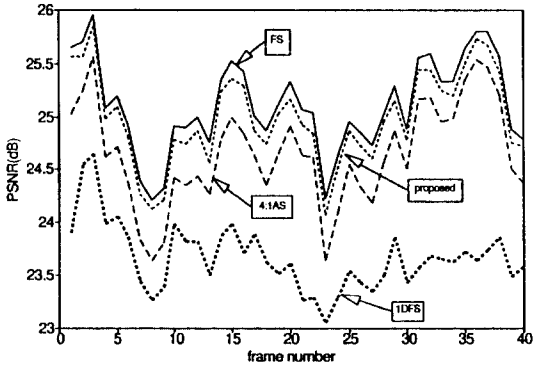

그림 8. "football" 영상에서의 PSNR 그래프
Fig. 8. The PSNR graph in "football" sequences.

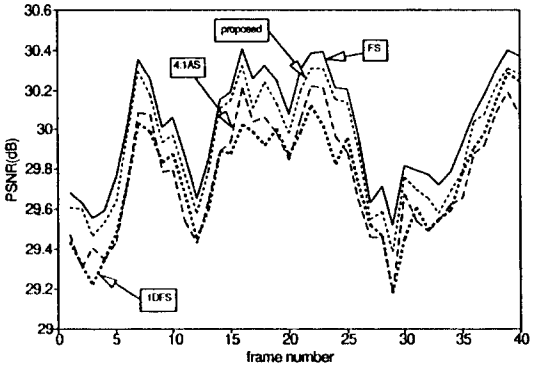

그림 9. "popple" 영상에서의 PSNR 그래프
Fig. 9. The PSNR graph in "popple" sequences.

표 1,2에서 알 수 있듯이 제안하는 알고리듬은 완전탐색 알고리듬에 비해 단위처리기 총복잡도가 1/12로 감소하였으며, 화소의 해상도를 낮춤으로 인해 생기는 PSNR 저하는 0.03~0.12 dB였다. 4:1 교번 부표본화 알고리듬과 비교하면 단위처리기 총복잡도는 1/3로 감소하였으며, PSNR은 0.01~0.34 dB 높다는 것을 알 수 있다. 제안하는 알고리듬은 1차원 전역탐색 알고리듬에 비해서 단위처리기 총복잡도가 1.4배이고 PSNR은 0.16~1.32dB 높으므로, PSNR의 개선 효과가 단위처리기 총복잡도의 증가보다 크다는 것을 알 수 있다.

2장에서 살펴본 바와 같이 선택적 탐색 알고리듬은 국부 극소가, 화소 부표본화 알고리듬은 수직선, 수평선이 성능 저하의 주원인이다. 표 2 및 그림 8,9에서 알 수 있듯이 움직임의 크기가 크기 때문에 국부 극소가 일어날 가능성이 많은 "football"과 "table tennis"에서는 선택적 탐색 알고리듬인 1차원 전역 탐색 알고리듬의 성능 저하가 화소 부표본화 알고리듬인 4:1 교번 부표본화 알고리듬에 비해 훨씬 크다. 수직선, 수평선이 많은 "popple"과 "flower garden"에서는 4:1 교번 부표본화 알고리듬의 성능이 이보다 연산량이 훨씬 작은 1차원 전역 탐색 알고리듬과 큰 차이를 보이지 않는다. 이에 비해서 제안하는 알고리듬은 모의실험에 사용된 네개의 표준영상 모두에 대해서 완전탐색 알고리듬과 비슷한 성능을 보였다. 따라서 제안하는 알고리듬은 2장에서 열거한 완전탐색 알고리듬과 고속탐색 알고리듬의 문제점을 효과적으로 개선하였음을 알 수 있다.

VI. 결 론

본 논문에서는 화소의 비트 해상도를 낮추어 비교함으로서 완전탐색 알고리듬과 고속탐색 알고리듬의 문제점을 개선한 저해상도 양자화 움직임 추정 기법을 제안하였다. 제안하는 알고리듬은 화소의 해상도를 낮춘 저해상도 이미지를 비교하여 움직임 벡터 후보위치를 결정하고, 이들 움직임 벡터 후보위치에 대해 원래 해상도로 비교하여 움직임 벡터를 결정한다.

각 화소가 가지고 있는 정보를 가능한 한 잃어버리지 않고 저해상도 이미지를 생성하기 위해 기준블록과 탐색영역의 화소값은 각각 기준블록의 평균값을 뺀 후 2비트로 양자화한다. 양자화 오차에 의한 성능 저하를 최소로 하기 위해 양자화 경계는 기준블록의 표준편차에 의해 적응적으로 결정된다. 움직임 벡터 후보위치들은 기준블록과 탐색영역의 양자화된 코드값이 서로 일치하지 않는 화소의 개수를 세어 결정된다.

모의실험 결과 제안하는 알고리듬은 단위처리기 총

복잡도로 표시된 연산량을 완전탐색 알고리듬의 1/12로 감소시켰고, PSNR로 표시된 성능 저하는 0.03~0.12 dB였다. 또한 고속탐색 알고리듬의 단점인 국부 극소나 수평선, 수직선 등으로 인한 성능 저하도 개선되었다.

참 고 문 헌

[1] J.R.Jain *et al.*, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Comm.*, vol. COM-29, pp. 1799-1808, Dec. 1981.

[2] N.Ahmed *et al.*, "Discrete cosine transform," *IEEE Trans. Comm.*, vol. COM-23, pp. 90-93, Jan. 1974.

[3] D.Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40. pp. 1098-1101, 1952.

[4] T.Koga *et al.*, "Motion compensated interframe coding for videoconferencing," *Proc. Nat. Telecommun. Conf.*, pp. G5.3.1-G5.3.5, 1981.

[5] M.J.Chen *et al.*, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 504-509, vol. 4, no. 5, Oct., 1994.

[6] R.Srinivasan *et al.*, "Predictive coding based on efficient motion estimation," *IEEE Trans. Comm.*, vol. COM-33, pp. 888-896, Aug. 1985.

[7] B.Liu *et al.*, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 148-157, vol. 3, no. 2, Apr. 1993.

[8] M.Biering *et al.*, "Displacement estimation by hierarchical block matching," *Proc. SPIE Visual Commun. Image Processing*, pp. 942-951, 1988.

[9] Jayant *et al.*, *Digital coding of waveforms*, Prentice-Hall, pp. 131, 1984.

저 자 소 개

李 誠 洙(正會員)

1968년 5월 27일생. 1991년 2월 서울대학교 전자공학과 졸업(학사). 1993년 2월 서울대학교 대학원 전자공학과 졸업(석사). 1993년 ~ 현재 서울대학교 대학원 전기공학부 박사과정 재학중. 주관심분야는 영상처리 알고리듬, 영상처리 집적회로 설계 등임.

蔡 洙 翊(正會員) 第 31卷 B編 第 5號 參照.
현재 서울대학교 전기공학부 교수