論文2001-38SP-3-3

DCT 직류 값을 이용한 움직임 추정기 설계에 관한 연구

(A Study on Motion Estimator Design Using DCT DC Value)

李 權 喆 *, 朴 鍾 鎭 *, 趙 源 敬 *

(Kwon Cheol Lee, Jin Jong Park, and Won Kyung Cho)

요 약

정보량이 많은 고화질의 동영상을 실시간으로 전송하기 위하여 압축 알고리즘을 필수적으로 사용하고 있으며, 시간적 중복성을 제거하는 동영상의 압축방법은 움직임 추정 알고리즘을 사용한다. 본 연구에서 설계하고자 하는 움직임 추정기는 블록정합 알고리즘이며, MPEG 부호기에서 사용되는 DCT 연산 결과인 DC 값을 이용하여 화면의 밝기를 판단한다. 움직임 추정기는 휘도 신호 8비트 모두를 사용하지 않고, 화면 밝기에 따른 비트 플레인(bit plane)에서 3비트만 선택하는 비교선택기를 이용한다. 본 연구에서 제안한 비교 선택기는 I-Picture만을 계산한다. I-Picture에 의해 계산된 선택 비트는 I, P와 B Picture의 움직임 추정 연산에 사용함으로서 움직임 추정기의 크기를 줄일 수 있는 구조를 제안하였다. 제안된 움직임 추정기의 고찰을 위하여 실험에 사용된 표준 동영상의 해상도는 352×288이며, DCT 연산의 처리 블록은 8×8이며, 탐색영역은 23×23이다. 제안된 알고리즘은 C언어로 모델링하였으며, 기존 완전탐색방법과 PSNR을 비교한 결과 사람의 시각으로 거의 구별할 수 없는 작은 차이(0~0.83dB)가 나타남을 알 수 있었다. 본 연구에서 제안한 움직임 추정기의 하드웨어 크기는 기존 구조Ⅰ보다 38.3%, 기존 구조Ⅱ보다 30.7% 줄일 수 있었고, 메모리 크기는 기존 구조Ⅰ,Ⅱ보다 31.3% 줄일 수 있었다.

Abstract

The compression method is necessarily used to send the high quality moving picture that contains a number of data in image processing. In the field of moving picture compression method, the motion estimation algorithm is used to reduce the temporal redundancy. Block matching algorithm to be usually used is distinguished partial search algorithm with full search algorithm. Full search algorithm be used in this paper is the method to compare the reference block with entire block in the search window. It is very efficient and has simple data flow and control circuit. But the bigger the search window, the larger hardware size, because large computational operation is needed. In this paper, we design the full search block matching motion estimator. Using the DCT DC values, we decide luminance. And we apply 3 bit compare-selector using bit plane to I(Intra coded) picture, not using 8 bit luminance signals. Also it is suggested that use the same selective bit for the P(Predicted coded) and B(Bidirectional coded) picture. We compare based full search method with PSNR(Peak Signal to Noise Ratio) for C language modeling. Its condition is the reference block 8×8, the search window 24×24 and 352×288 gray scale standard video images. The result has small difference that we cannot see. And we design the suggested motion estimator that hardware size is proved to reduce 38.3% for structure I and 30.7% for structure II. The memory is proved to reduce 31.3% for structure I and II.

* 正會員, 慶熙大學校 電子工學科

(Depart. Electronics of Kyunghee University)

接受日字:2000年7月29日, 수정완료일:2001年1月9日

Ⅰ. 서 론

정보량이 많은 고화질 동영상을 고속으로 전송하기 위해서는 데이터 압축이 필수적이며, 동영상 데이터의 압축방법은 동영상 데이터의 중복성(Redundancy)을 최소화하는데 있다. 움직임 추정 방법은 연속되는 동영상 프레임간의 중복성을 제거함으로서 데이터 압축 효과를 얻을 수 있다[1]. 움직임 추정 방법 중 보편적으로 많이 사용되는 블록 정합 알고리즘[2,3]은 현재 프레임을 여러 개의 기준블록으로 분할하여 이전 프레임의 탐색 영역 안에서 가장 유사한 블록을 찾아 기준블록에 대한 상대 위치를 추출하는 방법이며, MPEG 1[4,5],2[6],4나 H.261[7], H.262, H.263[8]등 대부분의 표준 동영상 압축 방법으로 채택되었다. 블록 정합 알고리즘 중에는 3단계 탐색[8], 2차원 대수 탐색[9], One-at-a-time 탐색[10], Conjugate Direction 탐색[11] 같은 부분 탐색과 완전 탐색 알고리즘으로 구별되는데, 본 논문은 완전 탐색 알고리즘을 적용하였으며, 기준 블록을 탐색영역 내의 모든 블록과 비교하는 방법으로 성능이 우수하고 데이터 흐름과 제어 회로가 비교적 간단하다는 장점이 있다. 그러나 탐색 영역이 커질 경우 많은 연산을 필요로 하여 하드웨어 크기가 커지는 단점이 있다. 또한 본 연구에서 제안한 방법은 완전 탐색 알고리즘 뿐만 아니라 부분 탐색 알고리즘에도 적용할 수 있다.

본 논문에서 제안한 움직임 추정기는 DCT 직류 값을 이용하여 화면의 밝기를 판별한 후, 비트 플레인을 이용하여 휘도 8비트 신호 중 정보가 모여있는 부분의 3비트만 선택하여 사용함으로서 움직임 추정기의 전체적인 연산 비트 크기를 줄일 수 있는 구조이다. 제안한 구조의 움직임 추정기는 DCT 연산기가 포함되어 있는 MPEG 부호화기에서 적용 할 수 있으며, 제안한 구조의 기능 검증을 위하여 DCT 연산의 기준블록은 8×8으로 하고, 탐색영역은 24×24으로 하며, 352×288 그레이 스케일의 표준 동영상을 사용하였다. 제안된 알고리즘은 C언어로 모델링하여 기능을 검증하였고, VHDL을 이용하여 움직임 추정기를 기술하였으며, 합성 도구를 이용하여 제안된 움직임 추정기를 합성하고 이를 토대로 하드웨어 크기를 고찰하였다.

Ⅱ. 움직임 추정 알고리즘

1. 개요

영상 시퀀스(Image Sequence)에서 움직임이란 시간축으로 연속된 화면에서의 차이로 정의된다. 만약 화면이 거의 변하지 않으면 두 개의 연속된 화면은 상관관계가 높을 것이다. 따라서 화면의 차이는 적을 것이고 이러한 차이만을 고려한다면 영상압축의 양도 줄어들 것이다. 이러한 사실을 바탕으로 화면간의 영상을 압축하는데 움직임 추정을 이용한다[16]. 움직임 추정 알고리즘은 해당 화소의 이웃하는 화소들로부터 각 화소의 변위를 회귀적으로 움직임을 예측하는 화소 반복 알고리즘(PRA, Pel-Recursive Algorithm)과 이전 프레임의 주어진 탐색 영역 안에서 현재 프레임 내의 한 기준 블록에 대해 블록단위로 비교하여 가장 유사한 블록을 찾는 블록 정합 알고리즘(BMA, Block Matching Algorithm)이 있다[2,3]. 화소 반복 알고리즘은 블록정합 알고리즘보다 예측이 정확하지만, 각 화소의 변위를 회귀적으로 모두 계산하여야 하기 때문에 연산량이 매우 많다. 따라서 화소 반복 알고리즘을 하드웨어로 구현하는데 있어 고속의 연산기를 필요로 한다. 블록 정합 알고리즘의 규칙적인 연산은 하드웨어 구현이 용이하지만, 완전 탐색(Full Search)에 필요한 많은 연산량이 문제점이다. 완전 탐색 알고리즘에서 연산량을 줄이는 방법으로 부분 탐색(Partial Search) 방법들이 연구되었다. 부분 탐색 방법은 연산량을 줄일 수는 있으나 이를 하드웨어로 실현할 때는 연산의 비 규칙성 때문에 연산기의 제어가 복잡해지고, 블록 탐색 과정에서 각 단계에서 잘못된 정합 과정에 따라 최종적으로 잘못된 결과를 얻을 수 있는 문제점이 있다.

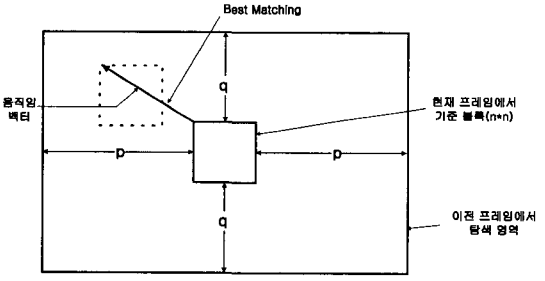


그림 1. 블록 정합 알고리즘의 연산 방법
Fig. 1. Operation Methods of Block Matching Algorithms.

2. 블록 정합 알고리즘

MPEG에서 움직임 벡터의 추정은 매크로 블록 단위

로 연산이 이루어지며, 블록 단위의 움직임 추정 방식에서 블록 정합 알고리즘은 예측 효율과 추정의 정확도, 계산상의 복잡도, 그리고 움직임 벡터 표현을 위한 부가정보량 등의 서로 상충되는 요건들을 대체로 잘 만족시킨다. 이와 같은 블록 정합 알고리즘은 MPEG과 같은 블록 단위의 부호화 기법에서 구현이 용이하기 때문에 널리 이용한다.

1) 블록 정합의 기준

블록 정합 알고리즘에서 최적의 정합 블록을 찾기 위한 정합 기준으로는 식 (1), (2), (3)와 같이 예측 오차나 블록의 상관성을 계산하는 함수들이 사용된다.

최소 MAD(Mean of Absolute Difference) :

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

최소 MSE(Mean of Square Error) :

$$MSE(i,j)=\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}|x_t(k,l)-x_{t-1}(k+i,l+j)|^2 \quad (2)$$

최대 NCCF(Normalized Cross Correlation Function) :

$$NCCF(i,j)=\frac{\sum_{k=1}^{N}\sum_{l=1}^{N}x_t(k,l)\cdot x_{t-1}(k+i,l+j)}{[\sum_{k=1}^{N}\sum_{l=1}^{N}x_t^2(k,l)]^{1/2}\cdot[\sum_{k=1}^{N}\sum_{l=1}^{N}x_{t-1}^2(k+i,l+j)]^{1/2}} \quad (3)$$

여기서 $x_t(k,l)$은 현재 프레임의 (k,l)위치에 있는 화소의 밝기 값이고, $x_{t-1}(k+i,l+j)$는 이전 프레임의 (k,l) 위치에서 (i,j)만큼 이동한 위치에 있는 화소의 밝기 값이다. 식 (1), (2), (3)와 같은 정합 기준들이 블록 정합 알고리즘의 성능이나 정확도에 심각한 영향을 주지 않으며, 최소 MAD 함수는 계산이 비교적 단순하고 VLSI 구현이 용이하기 때문에 블록 상관성 계산에 많이 사용한다.

2) 완전 탐색 블록 정합 알고리즘

완전 탐색 방식은 탐색 범위내의 모든 후보들을 탐색한다. 탐색 범위 내의 모든 후보들을 탐색하기 때문에 최대의 성능을 얻을 수 있고, 데이터의 처리 과정이 규칙적이어서 병렬 처리 구조의 설계가 용이하다. 프레임율이 Ff이며, 한 프레임이 Np개의 가로축 화소와 Nl개의 세로축 화소로 구성되었을 경우 블록 정합 알고리즘의 연산량은 식 (4)와 같다. 그림 1과 같이 기준 블록의 크기가 n×n개의 화소로 구성되고, 최대 허용 가능한 이동 변위를 p라고 하면, 탐색영역 내의 각각의 후보블록 당 연산 횟수는 식(1)에서 알 수 있듯이 3n2-1(n2의 뺄셈, n2의 절대값 계산, n2-1의 누적연산)의 연산이 필요하다. 또한 최소값을 찾기 위해 3번의 추가 연산이 필요하다. 따라서 (2p+1)2(p=q이면)개의 모든 후보 블록들을 비교하기 위해서는 총 (2p+1)2×(3n2+2)의 연산이 필요하다. 이는 한 개의 기준 블록을 탐색 영역 내의 모든 후보 블록들과 정합하는데 필요한 연산량이다. 한 프레임 내의 가능한 기준블록 수는 단순히 (Np×Nl)/n2으로 구할 수 있다. 한 프레임의 모든 기준블록에 대한 연산량을 계산해보면 (3n2+2)×(2p+1)2×{(Np×Nl)/n2}이 된다. 따라서 1초에 연산해야 할 연산량 Op는 식(4)와 같이 정의 할 수 있다.

$$Op=(2p+1)^2\times(3n^2+2)\times(N_l\times N_p)/n^2\times F_f \quad (4)$$

블록의 크기 n=8인 경우 국제적으로 규격화되어 있는 영상 형태에 따른 연산량 Op는 표 1과 같이 프레임의 크기와 탐색 영역에 따라 달라짐을 알 수 있다.

표 1. 움직임 추정 알고리즘의 처리 규격에 따른 초당 연산량(n=8)

Table 1. Computational operations per second to spec. for motion estimation algorithms(n=8).

수준	규격	탐색영역	프레임 수	초당 연산수 (GOP)
표준 CIF	352×288	+8,-7	10	2,923
			30	8,768
Main	720×576	+8,-7	10	11,956
			30	35,869
		+16,-15	30	138,886
High	1920×1152	+16,-15	60	1,481.145

Ⅲ. 비교선택기의 제안 및 움직임 추정기의 설계

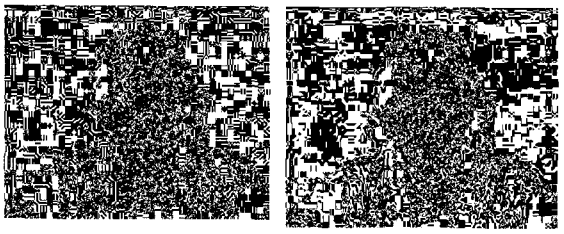
1. 제안된 움직임 추정기의 처리 방법

한 개의 화소는 n개의 비트로 구성되었으며, 그림 2(a)은 8비트로 표현된 원래의 영상 데이터를 보여준다. 또한 그림 2(b)는 8비트로 구성된 원래 영상 데이터의 최하위 1비트만을 표시한 것이고, 그림 2(i)는 원래 영상 데이터의 최상위 1비트(8번째)만을 표시한 것이다.

즉, 영상의 밝기에 따라 적절한 몇 개의 비트만으로도 원래 영상과 유사한 영상을 얻을 수 있음을 알 수 있다.

그림 3은 블록 당 휘도 평균값이 증가함에 따라 DCT 직류 값이 증가하며, DCT 직류 값은 화면의 밝기 정도를 나타냄을 알 수 있다. 따라서, 본 논문에서는 비트 플레인을 이용하여 휘도 신호 8비트 중 DCT 직류 값에 따라 레벨 비트를 정하고 좌·우 비트를 추가하여 3비트를 결정하여 처리함으로 프레임 메모리와

(a) Miss America Orignal Data



(b) 1'st Bit (c) 2'nd Bit

(d) 3'rd Bit (e) 4'nd Bit

(f) 5'nd Bit (g) 6'nd Bit

(h) 7'nd Bit (i) 8'nd Bit

그림 2. 영상 데이터의 각 비트 플레인 데이터
Fig. 2. Bit plane image of orignal image data.

움직임 추정기의 하드웨어 크기를 줄일 수 있는 비교 선택기를 제안한다. 표 2의 DC 직류 값에 따른 선택 비트의 결정은 패턴 영상을 사용하여 계산된 값이며, 패턴 영상은 선택된 비트에 불규칙적으로 데이터를 생성하고, 생성된 패턴 영상을 DCT 연산 후 DC 값을 추출한 평균값이다.

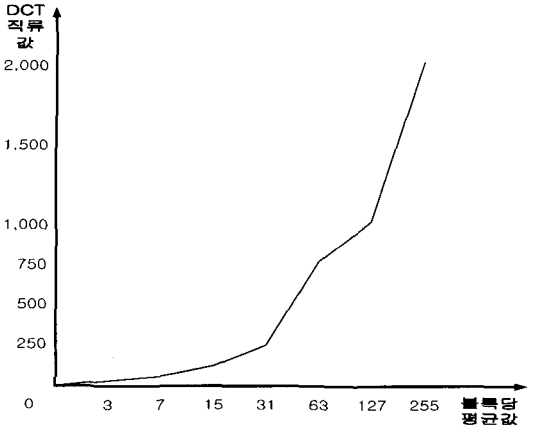


그림 3. 휘도 신호에 따른 DCT 직류 값의 변화
Fig. 3. DCT DC values as to luminance.

표 2. DCT 직류 값에 따른 선택 비트
Table 2. Selective bits as to DCT DC values.

DCT 직류값	선택비트
0-24	1,2,3
25-56	2,3,4
57-120	3,4,5
121-248	4,5,6
249-760	5,6,7
761-2040	6,7,8

비교 선택기의 구조는 그림 4와 같다. 비교 선택기의 비교기는 DCT 연산기에서 입력된 DCT 직류 값을 표

2와 같이 판별하여 선택비트 신호를 출력한다. 또한 비교 선택기의 선택블록는 비교기에서 출력한 선택신호에 따라 8비트 영상 데이터 중 3비트를 선택하여 출력한다.

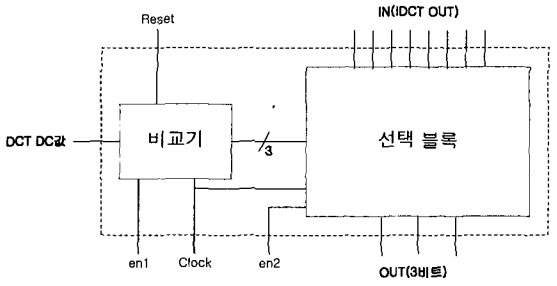


그림 4. 비교 선택기의 구조
Fig. 4. Structure of compare selector.

비교선택기는 그림 5와 같이 부호화 과정에서 I(Intra) 화면에만 선택적으로 적용하고, P(Predicted) 화면과 B(Bidirectional) 화면에서는 I(Intra) 화면에서 계산된 선택 비트를 사용한다.

2. 움직임 추정기의 설계

본 논문에서 제안한 움직임 추정기의 DCT 처리를 위한 기준 블록의 크기는 8×8이고, 탐색 영역의 크기는 24×24이며, 영상 신호는 8비트 그레이 스케일 동영상이다. 설계하고자 하는 움직임 추정기의 블록도는 그림 6과 같다.

한 프레임에 대한 움직임 추정 연산은 각 탐색영역에 대해 각각 움직임 벡터를 찾는 과정이다. 실험에 사용한 영상 크기는 352×288이며, 기준 블록이 8×8이므로 1,584개의 블록 연산이 요구된다. 또한 각각 블록 연산은 세 부분의 과정으로 나누어진다. 첫 번째 과정은 수직 또는 수평의 8개 화소에 대한 절대값 차를 누적해 가는 과정(PE 배열)이고, 두 번째 과정은 첫 번째 과정에서 누적된 값을 수평 또는 수직으로 8번 누적해 가는 과정(3단 가산기)이다. 세 번째 과정은 두 번째 과정에서 누적된 값을 탐색 영역(24×24)에서 16×16 블록에 대한 256번의 연산에 의해 계산된 256개의 누적 값 중 가장 작은 값을 찾고, 가장 작은 값의 위치 벡터 값을 블록의 움직임 벡터로 정한다. 최적의 정합 블록을 찾기 위한 정합 기준으로 사용한 것은 최소

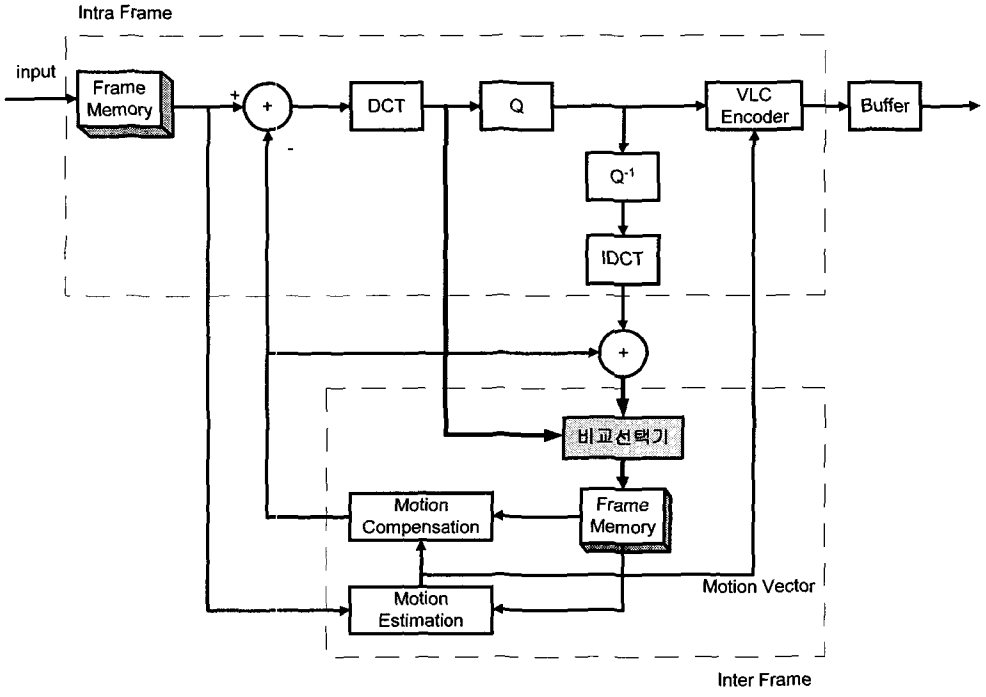


그림 5. 비교선택기를 포함한 MPEG 부호화기
Fig. 5. MPEG encoder with compare selector.

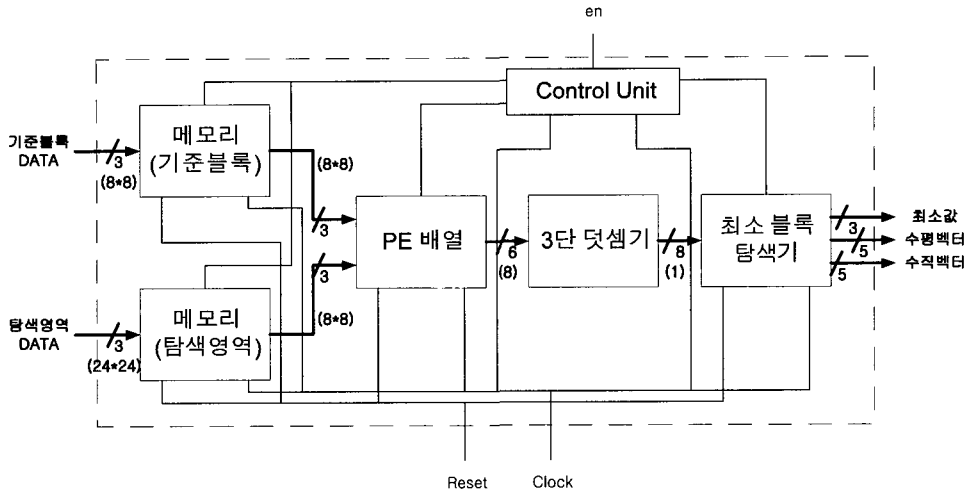


그림 6. 움직임 추정기의 전체 구조
Fig. 6. Block diagram of motion estimator.

MAD(Mean of Absolute Difference) 방식이다.

1) 움직임 추정기의 프로세서 요소(Process Element) 배열

프로세서 요소 배열은 수직 또는 수평으로의 8개 화소에 대한 절대값 차를 찾아 계속적으로 누적해 간다. 프로세서 요소 배열의 전체 구조는 그림 7과 같다.

프로세서 요소 배열은 64개의 프로세서 요소 블록으로 구성되며, 각 프로세서 요소 블록은 그림 8과 같다. 각 프로세서 요소는 기준 블록과 탐색 영역의 데이터 차에 대한 절대값을 계산하고, 두 블록의 차이값과 이전 프로세서 요소에서의 출력값을 가산하여 레지스터에 저장하는 누적기로 구성된다[17]. 프로세서 요소 블록을 구성하는 구조에서 절대값 차를 계산하는 부분의 구조는 3비트의 가산기와 1의 보수기를 사용한 간단한

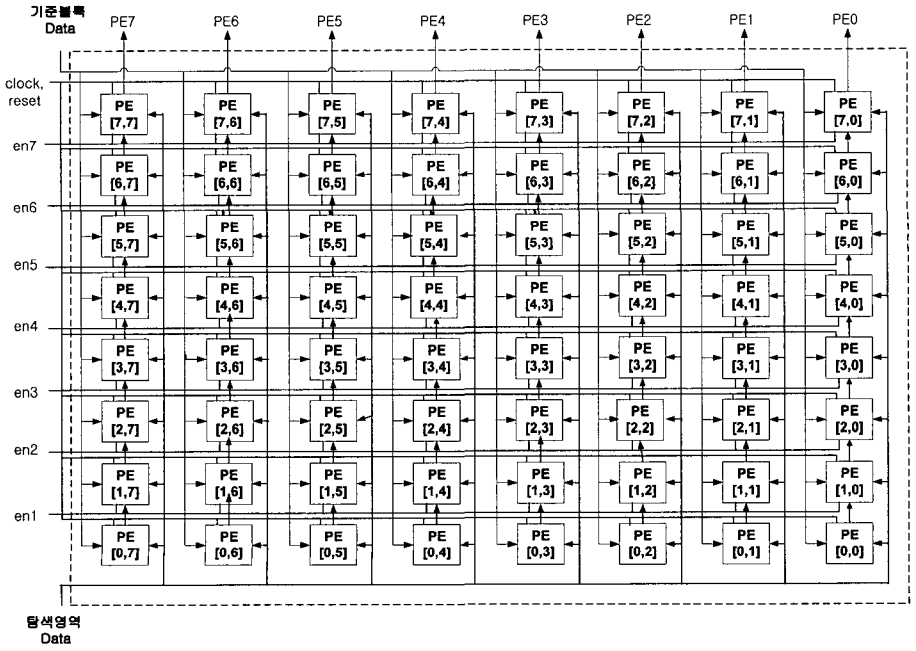


그림 7. 움직임 추정기의 프로세서 요소 배열
Fig. 7. Processing elements array of motion estimator.

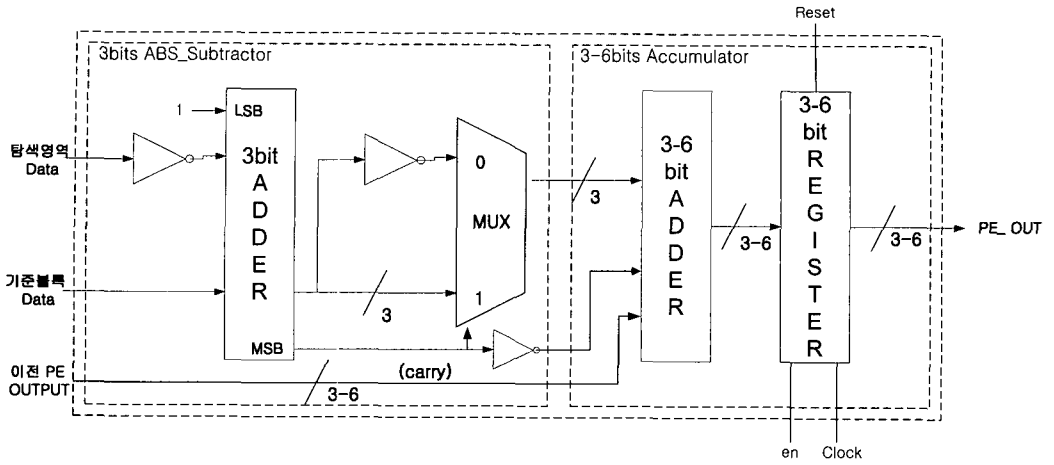


그림 8. 프로세서 요소 블록 구조
Fig. 8. Structure of processing elements block.

형태로 되어 있다. 그림 8의 구조에서 두 입력 중 한쪽의 입력을 1의 보수를 취한 후 가산기에서 두 입력 값을 더한 후 캐리가 발생하면 합에 1을 더하고, 캐리가 발생하지 않을 경우에 대해서는 그 합의 1의 보수를 결과로 선택한다. 따라서 두 입력의 절대값 차가 출력을 얻을 수 있다.

표 4는 프로세서 요소 배열의 위치에 따라 필요한 가산기와 레지스터의 비트 수를 보여준다. PE0은 앞단의 누적 값이 없으므로 가산기와 레지스터가 필요하지 않고 두 번째 단인 PE1은 PE0의 출력 값과 자신의 출력 값이 더해져서 누적되며, 각 단의 출력 값이 누적됨에 따라 소요되는 가산기와 레지스터가 커져야 한다.

표 3. 프로세서 요소의 하드웨어 크기
Table 3. Hardware size of processing elements.

구분	가산기	레지스터
PE7	6	6
PE6	6	6
PE5	6	6
PE4	5	6
PE3	5	5
PE2	4	5
PE1	3	4
PE0	×	×

2) 3단 가산기

각 프로세서 요소에서 출력된 두 블록의 차이 값은 동시에 8개가 출력되며, 이를 누적하기 위하여 3단 가산기를 사용한다. 3단 가산기의 구조는 그림 9와 같으며, 8개의 프로세서 요소의 누적된 값을 수직으로 8번 누적하여 결과를 출력한다. 3단 가산기도 PE 배열과 마찬가지로 표 5과 같이 각 단마다 다른 크기를 갖는다.

표 4. 3 단 가산기 각 단의 데이터 크기(bit)
Table 4. Data size of 3-step adder in each step.

구분	가산기
3단	8
2단	7
1단	6

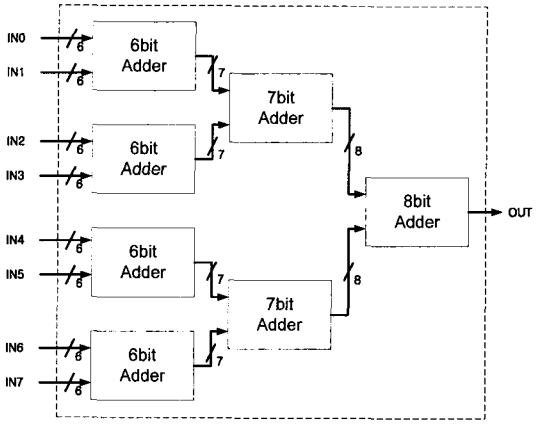


그림 9. 3단 가산기 구조
Fig. 9. Structure of 3-step adder.

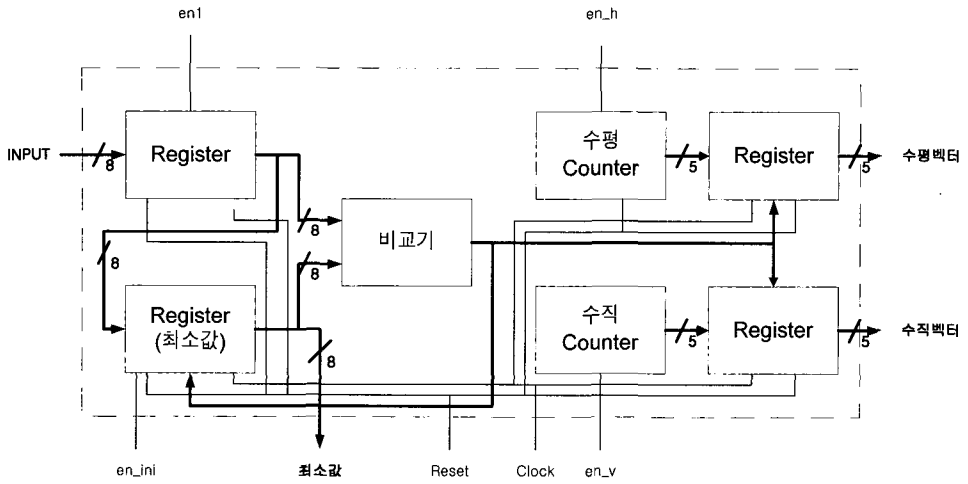


그림 10. 최소 블록 탐색기의 구조
Fig. 10. Structure of minimum value block searcher.

3) 최소 블록 탐색기

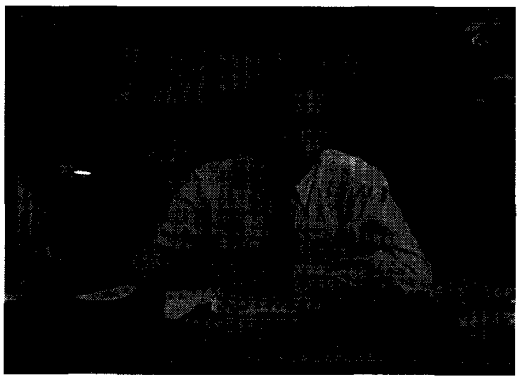
최소 블록 탐색기는 3단 가산기에서 누적된 256개의 값 중에서 가장 작은 값을 계산하고, 가장 작은 값에 해당하는 위치 정보를 현재 선택된 블록의 움직임 벡터로 출력한다. 최소 블록 탐색기는 그림 10과 같이 비교기와 두 개의 카운터로 구성되는데, 비교기는 256개의 정합된 값 중에서 가장 작은 값을 계산하기 위하여 사용되고, 두 개의 카운터는 탐색 영역의 수직과 수평의 해당하는 위치정보를 계산하기 위하여 사용된다.

Ⅳ. 실험 및 고찰

본 논문에서 제안한 구조의 성능 평가는 프레임 크기가 352×288인 클라넷와 미스아메리카, 세일즈맨의 30 프레임 그레이 스케일의 동영상을 사용하였고, 제안된 움직임 추정기의 구조는 C 언어로 모델링하였다.

(a) Claire

(b) Miss America



(c) Salesman

그림 11. 실험에 사용한 샘플 영상
Fig. 11. Sample images.

표 5는 실험에 사용한 동영상의 첫 프레임 DCT 직류 값과 표 2에 따른 선택 비트를 보여준다. 표 5와 같

이 세 개의 실험 데이터는 모두 선택 비트가 5, 6, 7임을 알 수 있었다.

표 5. 동영상에 대한 DCT 직류 값과 선택 비트

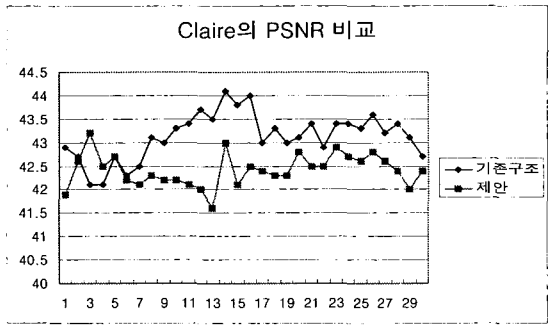
Table 5. DCT DC value and selective bits in moving image.

구분	Claire	Miss America	Salesman
DCT DC값(8×8 환산)	745	506	584
선택비트	5,6,7	5,6,7	5,6,7

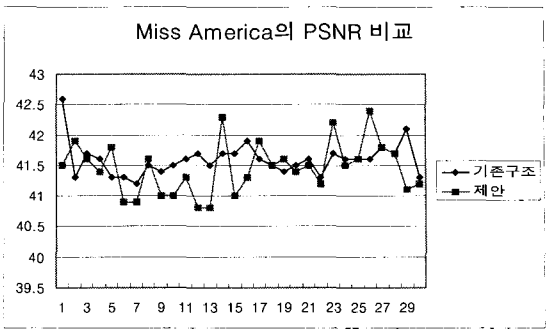
동영상에 대한 기존의 완전탐색기법과 본 논문에서 제안한 방법의 성능 평가는 PSNR(Peak Signal to Noise Ratio)을 사용하였다. PSNR은 다음 식 (5)와 같이 정의된다.

$$PSNR = 20\log_{10}\frac{255}{\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}|x_t(k,l)-x_{t-1}(k+i,l+j)|^2} \quad (5)$$

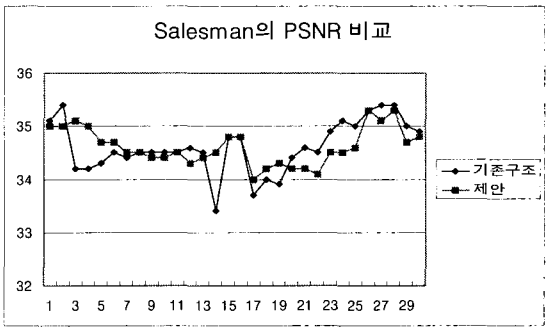
여기서 N2은 영상의 크기이고, $x_t(k,l)$은 현재 프레임의 (k,l)위치에 있는 화소의 밝기 값이고, $x_{t-1}(k+i,l+j)$는 이전 프레임의 (k,l)위치에서 (i,j)만큼 이동한 위치에 있는 화소의 밝기 값이다. 원래영상에 대한 복원영상의 평균 오차인 PSNR은 그림 12와 같다. 실험 결과는 본 논문에서 제안한 방법이 기존 완전탐색 기법과 거의 유사함을 알 수 있다. 또한 동영상에 대한 PSNR 평균값은 표 8과 같다.



(a) Claire



(b) Miss America



(c) Salesman

그림 12. 동영상의 각 프레임에 따른 PSNR 비교(dB)

Fig. 12. PSNR Comparison of Sample Moving Picture Frame.

표 6. PSNR 평균값 비교(dB)

Table 6. PSNR average value comparison.

구분	기존완전탐색방법	제안 방법
Claire	43.2	42.4
Miss America	41.6	41.5
Salesman	34.6	34.6

또한 기능이 검증된 움직임 추정 알고리즘은 VHDL을 이용하여 기술하였고, VHDL 컴파일러 및 시뮬레이터를 사용하여 움직임 추정기의 기능을 시뮬레이션 하였으며, 합성 결과는 표 7과 같다. 본 논문에서 제안한 움직임 추정기는 하드웨어 구현 시 하드웨어 크기를 기존 구조 I [18]보다 38.3%를 줄일 수 있었으며, , 기존 구조II[19]보다는 30.7% 줄일 수 있었다. 또한 움직임 추정기에서 사용하는 메모리는 기존구조 I ,II보다 31.3% 줄일 수 있었다.

표 7. 움직임 추정기의 합성 결과
Table 7. Synthesis results of motion estimator.
(단위 : Gate)

구성부	기존구조 I	기존구조 II	제안 구조
비교선택기	-	-	4,421
PE 배열	24,334	21,633	10,587
3단 가산기	1,047	931	573
최소블록 탐색기	627	594	470
프레임 메모리 (352×288)	1.6M	1.6M	1.1M
기준블록 메모리 (8×8)	512	512	192
탐색영역 메모리 (24×24)	4.6K	4.6K	1.7K

V. 결 론

HDTV와 같은 영상 처리 분야에서는 많은 데이터를 포함하는 고화질의 동영상을 고속으로 전송하기 위하여 압축기법을 필수적으로 사용하고 있다. 동영상 정보의 압축기법 중에서 시간적 중복성을 제거하는데는 움직임 추정기법을 사용한다. 움직임 추정기법 중 가장 많이 사용되는 블록 정합 알고리즘은 부분 탐색과 완전 탐색 알고리즘으로 구별되는데, 본 논문에서 제안한 구조는 모든 블록 정합 알고리즘에 적용 할 수 있다. 본 연구에서 설계한 움직임 추정기는 완전 탐색 알고리즘을 적용하였으며, DCT 직류 값을 이용하므로 DCT 연산기가 포함되어 있는 부호화기에 적용이 가능하다. 움직임 추정 방법은 DCT 직류 값에 따라 화면의 밝기를 판단하여 휘도 신호 8비트 모두를 사용하지 않고, 비트 플레인을 이용하여 그 중에 정보가 모여있는 3비트만 선택하여 연산하는 방법이다.

제안된 알고리즘은 기준 블록 8×8, 탐색 영역 23×23, 352×288 gray scale 표준 동영상에 적용한 결과 기존 알고리즘에 비하여 PSNR 값의 차이는 0~0.83dB으로 큰 차이가 없음을 알 수 있었고, 실제 복원된 동영상을 시각으로 확인한 결과 기존 알고리즘과의 차이를 구별할 수 없을 정도이다. 움직임 추정기를 VHDL을 이용하여 기술하였으며, 합성은 Synopsys Synthesis을 이용하고, 합성 라이브러리는 삼성 KG90 0.35㎛ SOG 공정을 사용하였다. 합성한 결과 본 연구에서 제안한 움직임 추정기는 코어의 크기에서 기존구조 I 에 비하여 38.3%를 줄일 수 있었고, 기존구조 II에 비하여 30.7% 줄일 수 있었다. 또한 움직임 추정기에서 사용하는 메모리 크기는 기존구조 I, II보다 31.3% 줄일 수 있었다.

앞으로 DCT 직류 값에 따른 선택 비트를 정하는데 있어서 보다 정확한 움직임 추정을 하기 위해 더 많은 연구가 필요하며, 프레임 사이의 변화가 큰 동영상의 처리에서 B-Picture와 P-Picture의 선택 비트 결정 방법에 대한 연구가 필요하다. 본 연구에서 제안된 움직임 추정기는 DCT 연산기를 포함하고 있는 MPEG2 Encoder 시스템의 적용해 봄으로써 시스템 수준의 검증을 필요로 한다.

참 고 문 헌

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

[2] T.Komarek et al., "Array Architecture for Block Matching Algorithms", IEEE Trans. on Circuit and System, vol 36, pp. 1301~1308, Oct. 1989.

[3] Luc De Vos et al., "Parameterizable VLSI Architectures for the Full-Search Block-Matching Algorithm", IEEE Trans. on Circ. and Syst., vol. 36, pp. 1309~1316, Oct. 1989.

[4] Simulation Model Editorial Group, "MPEG video simulation model three(SM3)", ISO/IEC JTC1/SC29/WG11, MPEG90, no. 41, July, 1990.

[5] MPEG Editorial Group, "Coding of moving pictures and associated audio for DSM at up to about 1.5 Mbits/s: ISO/IEC11172-2 IS", ISO/IEC JTC1/SC29/WG11, Apr. 1993.

[6] MPEG Test Model Editing Committee, "Generic coding of moving pictures and associated audio: MPEG-2 Test Model5(TM5)", ISO/IEC JTC1/SC29/WG11, MPEG94, no.702, Mar. 1994.

[7] ITU-T Rec. H.261 : "Video Codec for Audio Visual Services at p×64kb/s", Mar. 1993.

[8] ITU-T Rec. H.263 : "Video Coding for Low

Bitrate Communication", Oct. 1995.

[9] T.Koga and et al. "Motion compensated interframe coding for video-conferencing," Proc. Nat. Telecommun. Conf., New Orleans, LA, pp. G5.3.1-G5.3.5, 1981

[10] R.Srinvasan and K.R.Rao, "Predictive coding based on efficient motion estimation", ICC'1984, pp. 541~526. 1984.

[11] R.Srinvasan and K.R.Rao, "Predictive coding based on efficient motion estimation", IEEE Trans. on Commun., vol. COM-33, no. 8, pp. 888~896, Aug. 1985.

[12] 후지와라 히로시, "최신 MPEG", 교보문고, 1995.

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

[14] 대우전자 영상연구소, "MPEG 비디오", 연암출판사, 1995.

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

[16] H.G.Musman et al., "Advances in Picture Coding", Proc. IEEE, vol. 73, pp. 523~548, Apr. 1985.

[17] Takao ONOYE, et al., "A VLSI Architecture for MPEG2 MP@HL Real time Motion Estimation", IEEE'96, pp. 664-667, 1996.

[18] Renaud Pacalet, et al., "A Real Time MPEG2 MP@ML Motion Estimator Chipset", ISSCC'97/ SESSION16, pp. 260~261, 1997.

[19] 이태호, "완전탐색 블록정합 이동예측기 구조의 최적화 설계", 한국과학기술대학 석사학위논문, Feb. 1997.

저 자 소 개

李 權 喆(正會員)

1967년 3월 13일생. 1992년 2월 경희대학교 전자공학과 공학사. 1994년 2월 경희대학교 전자공학과 공학석사. 1999년 2월 경희대학교 전자공학과 공학박사 수료. 1999년 8월~2001년 2월 주식회사 앤스랩. 주관심분야 : 신호처리 및 영상처리, 회로 및 시스템, ASIC 설계

朴 鍾 鎭(正會員)

1966년12월17일생. 1989년 2월 경희대학교 전자공학과 공학사. 1991년 8월 경희대학교 전자공학과 공학석사. 2001년 2월 경희대학교 전자공학과 공학박사. 1999년 5월~현재 주식회사 앤스랩. 2000년 2월~현재 대림대학 전자정보통신학과 겸임교수. 주관심분야 : 디지털 신호 처리 시스템, 통신 시스템, ASIC 설계

趙 源 敬(正會員) 第 29卷 B編 第 2號 參照