2001年度 第14回 信號處理合同學術大會 論文集 第14卷 1號

저 메모리를 갖는 제로트리기반 영상 압축

신 철 김호식 유지상

광 운 대 학 교 전 자 공 학 과

Low Memory Zerotree Coding

Cheol Shin, Ho-Sik Kim and Ji-Sang Yoo

Digital Media Lab., Dept. of Electronic Eng., Kwangwoon University
dustdin@explore.gwu.ac.kr, hosik@explore.gwu.ac.kr and jsyoo@daisy.gwu.ac.kr

요 약

제로트리 부호화 알고리즘 중 효율적이며 잘 알려진 SPIHT는 높은 메모리 요구로 인해 하드웨어 구현에 큰 어려움을 가지고 있다. 이 논문에서는 저 메모리 사용과 빠른 제로트리 부호화 알고리즘을 제안한다.

메모리를 줄이고 빠른 코딩을 위한 방법으로 다음 3가지를 사용한다. 첫 번째, 리프팅을 이용한 웨이블릿 변환은 기존의 필터뱅크 방식의 변환보다 저 메모리와 계산량의 감소를 가진다. 두 번째, 웨이블릿 변환된 계수들은 블럭으로 나누어져 각각 코딩된다. 여기서 블록은 제로트리 구조가 유지되는 STB(spatial tree-based block)이다. 세 번째, Wheeler 와 Pearlman이 제안한 NLS (no list SPIHT)를 이용한 부호화이다. NLS는 효율성에서 SPIHT와 거의 같으며 작고 고정된 메모리와 빠른 부호화 속도를 보여준다.

1. 서론

멀티미디어 정보 중 영상의 데이터는 대용량으로 한정된 대역폭을 통하여 전송되거나 저장하기 위해서는 압축이 필요하다. 최근 웨이블릿 변환을 이용한 영상의 압축은 DCT변환 방식의 블록간 왜곡현상과 에지 열화의 단점을 극복하면서 활발히 연구되어지고 있다.

웨이블릿 변환을 이용한 부호화 기법 중 제로트리 부호화 기법은 매우 높은 압축 성능을 보이고 있다. 이러한 알고리듬 중에서 우수한 성능을 가지는 SPIHT[1]은 주요 임계치를 이용하여 웨이블릿 계수를 중요 계수와 비중요 계수로 분류하고 부대역간의 상관관계를 이용하여 부호화한다. 그러나 부,복호 과정에서 3개의 리스트를 사용함으로써 높은 메모리를 요구한다. 이러한 고 메모리 요구는 제한된 메모리 사용 환경에서는 부적절한 것이다. SPIHT의 이러한 메모리 사용을 줄이기 위해 Wheeler와 Pearlman은 리스트 대신 상태 마커(state maker)를 사용한 NLS(no list SPIHT)[2]를 제안하였다. 본 논문에서는 변형된 NLS알고리즘을 사용하여 NLS보다 좀더 작은 메모리 사용과 효율성을 보여 준다. 여기서의 기본적인 생각은 다른 알고리즘 Tanbman와 Zakhor[3]의 논문에도 적용 가능 할 것이다.

높은 메모리 사용은 하드웨어 설계시 비용의 증가를 가져온다. 좀더 단순한 웨이블릿 변환과 영상압축 방법은 하드웨어 설계에서 꼭 필요한 기술이다. 리프팅(lifting)을 이용한 웨이블릿 변환은 기존의 필터뱅크 방식보다 메모리 사용이 거의 없고 계산량 또한 적다. 따라서 변형된 NLS는 하드웨어의 적용이 아주 적당하다고 할 수 있다.

본 논문의 구조는 다음과 같다. 2장에서 리프팅을 이용한 웨이블릿 변환과 NLS알고리즘에 대해 차례로 기술하고 3장에서는 이 논문에서 제안하는 알고리즘을 기술한다. 4장에서는 실험 결과를 고찰하고 향후 보완해야 할 미비점 등에 대해 기술한다.

2. 영상 압축

2.1 리프팅을 이용한 웨이블릿 변환

영상을 압축하는데 있어 웨이블릿을 이용한 방식은 기존의 DCT를 이용한 방식에 비해 블록킹 현상이 나타나지 않고 각 서브 밴드별로 처리가 가능하여 압축률을 높일 수 있다. 웨이블릿 변환은 Morlet에 의해 소개된 이후에 효율이 높고 계산량은 적은 웨이블릿에 대한 연구가 많이 진행되었다[4]. 이들 중 리프팅을 이용한 변환은 기존 필터 뱅크 방식에 비해 계산량이 줄어 속

표 1. (9.7)필터 계수값

n	0	±1	±2	±3	±4
$2^{-1/2}h_n$	0.602949	0.266864	−0.078223	−0.016864	0.026749
$2^{-1/2}\tilde{h}_n$	0.557543	0.295636	−0.028772	−0.045636	0

도가 빠르고 메모리를 적게 사용하는 장점이 있다[5].

리프팅을 이용한 웨이블릿 변환은 크게 분할(split), 예측(predict), 갱신(update)의 3 단계로 구성되어있다. 각 단계를 수식으로 설명하면 다음과 같다.

분할(split) : 입력 신호 x_l을 짝수 성분 s_i 과 홀수 성분 d_i으로 분할한다.

$$s_i^{(0)} = x_{2i} \ , \ d_i^{(0)} = x_{2i+1}$$

예측(predict) : $s_i^{(k-1)}$를 갖고 $d_i^{(k-1)}$를 예측하여 고주파 성분 $d_i^{(k)}$를 구한다.

$$d_i^{(k)} = d_i^{(k-1)} + \sum_j p_j^{(k)} \cdot s_{i+j}^{(k-1)} \ , k = \{1,\ldots,K\}$$

갱신(update) : $s_i^{(k-1)}$와 $d_i^{(k)}$를 갖고 원신호 x_n의 저주파 성분 $s_i^{(k)}$를 구한다.

$$s_i^{(k)} = s_i^{(k-1)} + \sum_j u_j^{(k)} \cdot d_{i+j}^{(k)} \ , k = \{1,\ldots,K\}$$

여기서, k는 예측과 갱신 단계가 반복되는 계층 수, p_j와 u_j는 각각 예측 계수, 갱신 계수이다.

본 논문에서 사용된 필터는 웨이블릿 변환에서 많이 사용되어지는 (9-7)필터를 사용하였다[6]. 표 1에 (9-7) 필터의 Analysis와 Synthesis 필터 계수가 각각 나타나 있다. 이 필터 계수들을 Factoring Algorithm [7]을 적용시켜 리프팅으로 변환시키면 아래 수식과 같이 된다. 입력 신호를 x_l이라 하면,

$$\begin{aligned}
s_i^{(0)} &= x_{2i} \ , \ d_i^{(0)} = x_{2i+1} \\
d_i^{(1)} &= d_i^{(0)} + \alpha(s_i^{(0)} + s_{i+1}^{(0)}) \\
s_i^{(1)} &= s_i^{(0)} + \beta(d_i^{(1)} + s_{i-1}^{(1)}) \\
d_i^{(2)} &= d_i^{(1)} + \gamma(s_i^{(1)} + s_{i+1}^{(1)}) \\
s_i^{(2)} &= s_i^{(1)} + \delta(d_i^{(2)} + d_{i-1}^{(2)}) \\
s_i &= \zeta s_i^{(2)} \ , \ d_i = d_i^{(2)} / \zeta
\end{aligned}$$

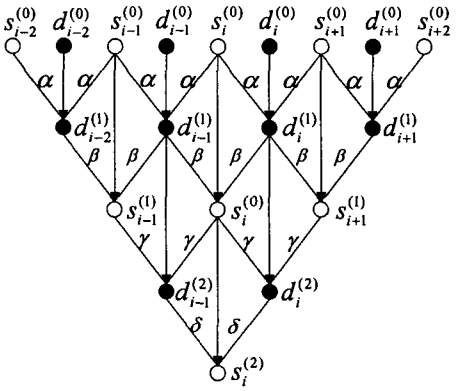

그림 1. (9,7)필터의 리프팅 구조

$\alpha = -1.586134342, \beta = -0.052980118$
$\gamma = 0.8829110762, \delta = 0.4435068522$
$\zeta = 1.149604398$

위 수식을 구조적으로 나타내면 그림 1과 같이 된다.

2.2 NLS 알고리즘

입력영상에 대해 웨이블릿 변환 후 NLS는 비트플랜(bit plane)방식에 따라 계수들의 값들을 양자화 한다. NLS는 리스트를 사용하지 않는 대신 표 2와 같은 상태 마커를 사용한다. 따라서 각 계수들은 4비트의 메모리를 요구한다 .또한 미리 각 트리에 대한 Descendants의 최대값을 구하여 정렬하고 Grand-descendants의 최대값도 구하여 정렬시킨다. 계수들의 값들은 그 크기와 부호를 분리하여 정렬시킨다. 이렇게 미리 각 트리의 최대값을 미리 구해 놓으므로 해서 간단하고 빠른 알고리즘 수행을 할 수 있다.

NLS에서 사용되는 메모리 크기를 계산하면, 영상의 크기를 수평방향으로 X, 수직 방향으로 Y라 하고 계수의 바이트 수를 C라 한다면 Descendants의 최대값들의 배열의 크기는 XYC/4이고 Grand-descendants의 최대값들의 배열의 크기는 XYC/16이다. 또한 계수의 크기와 부호를 분리 저장하기위해 XYC만큼 메모리가 필요하며 상태 마커는 XY/2가 필요하다. 따라서 NLS를 수행하는 동안 입력영상 크기만큼의 영상 메모리와 그 보다 56%많은 메모리가 필요하게 된다.

NLS알고리즘을 간단히 설명하면 각 비트플랜마다 3개의 패스가 존재한다. 첫 번째는 IP(insignificant pixel) 패스이다. SPIHT의 LIP pass와 유사하다. 각 계수들 중 상태 마커가 MIP인 것을 찾아 임계값 보다 크면 MNP를 할당한다. 두 번째는 IS(insignificant set) 패스는 SPIHT의 LIS 패스와 유사하다. 각 계수들을 조사하여 MD이고 임계값 보다 크면 자기를 포함하는 4개의 계수들을 MCP로 하고 직계 Descendant는 MG로 한다. 계수의 상태 마커가 MG이고 임계값보다 크면 자기를 포함하는 4개의 계수들을 MD로 할당한다. 그렇

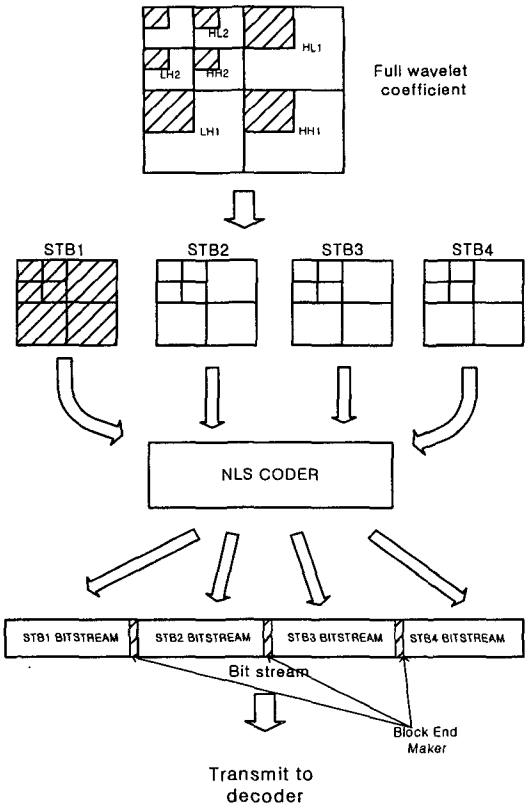

그림 2. 제안된 알고리즘 구조

치 않고 계수가 MCP를 가진다면 임계값을 조사하여 크면 MNP를 할당하고 작으면 MIP를 할당한다. 마지막으로 REF(refinement) 패스이다. 이것은 SPIHT의 LSP패스와 유사하다. 계수가 MSP이면 계수값의 더 정확한 값을 보내게 된다. 계수가 MNP이면 MSP로 바꾼다. 이 세 패스를 원하는 비트율이 될 때까지 반복한다.

3. 제안된 알고리즘

본 장에서는 제안된 알고리즘에 대한 과정을 설명을 한다. 그림 2에서 보듯이 제안된 알고리즘의 기본적인 생각은 웨이블릿 계수를 4개의 블록으로 나누는 것이다. 그리고 각 블록마다 독립적으로 NLS알고리즘을 수행하고 각각의 비트 스트림을 조합하여 전송단에 보낸다.

3.1 STB(Spatial Tree-Based Block)

웨이블릿 계수를 블록으로 나눌 때 가장 중요한 것은 트리구조가 그대로 유지되면서 블록을 나누는 것이다. 이러한 생각은 몇몇 코덱에서 사용되었다. Creusere는 병렬처리를 위한 EZW[8]에 이 기술을 사용했고, Rogers 와 Cosman은 error resilience[9]위해 유사한 방법을 사용했다. Wheeler와 Pearlman은 저 메모리를 위한 SPIHT[10]를 위해 이 기술을 이용했다.

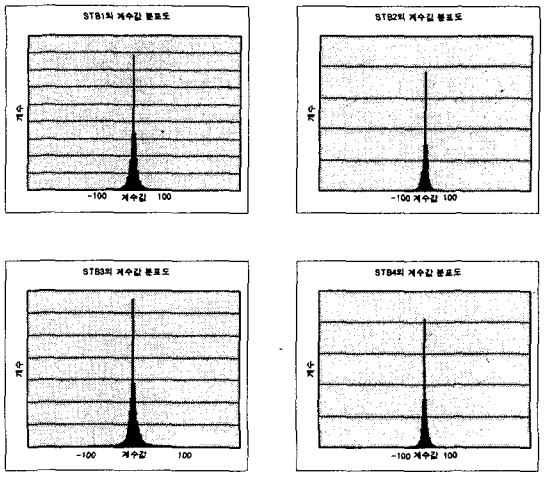

그림 3. 512*512 lenna 영상의 각 블록별 계수값 분포도

본 논문은 빠른 알고리즘 수행속도와 고정된 저 메모리를 갖는 NLS알고리즘을 더욱더 작은 메모리를 가지면서 압축 효율성 또한 거의 같은 효과를 보기 위해 이 기술을 이용한다.

그림 2에서 보듯이 4개의 블록을 나누므로 해서 생기는 메모리의 감소량은 NLS가 필요로 하는 메모리의 1/4만 있으면 된다. 즉 입력영상의 크기메모리와 그 보다 14%더 많은 메모리가 필요로 하게 된다. 따라서 NLS보다 적은 메모리 사용이 가능하다.

3.2 Bit Allocation

그림 2에서 각 블록들의 비트스트림(bitstream)을 하나의 단일 비트스트림으로 만들 때 바이트단위나 패킷단위로 각 블록들의 비트스트림값을 서로 교차하면서

표 2. 상태 마커

State Marker	Description
MIP	임계값보다 작거나 아직 테스트 되지 않음
MNP	임계값보다 큰 값이지만 REF패스를 통과하지 않음
MSP	임계값보다 큰 값이고 REF패스를 통과
MCP	MIP와 비슷하지만 IS패스에서만 존재
MD	계수값은 처음 1세대 자식
MG	계수값은 처음 2세대 자식
MN2	MD의 처음 2세대 자식
MN3	MD또는 MG의 처음 3세대 자식
:	:
MN6	MD또는 MG의 처음 6세대 자식

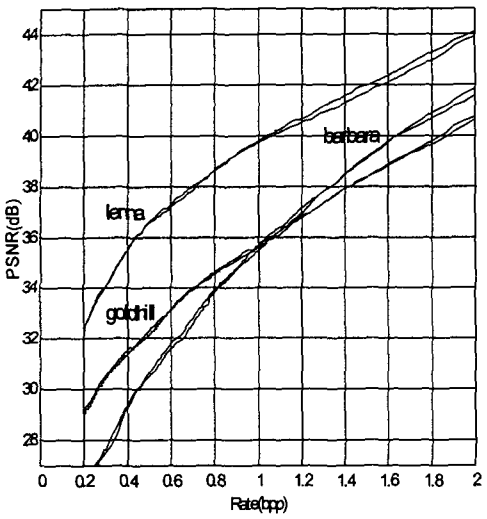

그림 4. Binary uncoded SPIHT와 제안된 알고리즘의 압축 성능 비교

만들지 않고 하나의 블록 비트스트림뒤에 다음 블록 비트스트림을 붙이는 형식으로 단일 비트스트림을 만들었정에서 주어진 비트율에서 복원영상과 원 영상의 왜곡율이 최소가 되는 각 블록들의 비트를 어떻게 주어져야 할지가 주요 관점이다.

여기서는 아주 간단한 방법으로 정해진 비트율에서 각 블록의 비트스트림이 얼마만큼의 비트를 할당 할 것인지를 결정한다. 4개의 블록에서 큰 계수의 값들이 많은 블록에 더 많은 비트를 할당해야 하는 것은 당연하다. 그림 3는 512*512 lenna 영상의 각 블록별 계수값들의 분포를 계수값이 -100에서 100사이의 값들만 나타내었다. 이 논문에서는 어느 일정한 임계값을 선택하여 임계값 이상의 계수들의 개수 합을 구하여 그 크기만큼의 비율로 각 블록에서 얼마만큼의 비트를 할당 할지를 결정했다. 실험 결과 임계값이 8인 경우 가장 좋은 결과가 나왔는데 이는 그림2에서 보듯이 임계값 8보다 크거나 작은 계수들의 합은 그 크기가 비슷하므로 어느 블록이 계수들의 값이 많은지를 구별하기에 적당치 않기 때문이다.

4. 실험 결과 및 분석

실험은 512*512 lenna, goldhill 그리고 barbara 영상을 이용하였다. 실험 결과는 그림 4와 같이 나왔다. 여기서는 5레벨까지 웨이블릿 분해하였다. 그림 4에서의 결과는 arithmetic 코딩을 행하지 않은 SPIHT와 제안된 알고리즘의 결과이다. 실선은 SPIHT의 결과이고 점선은 제안된 알고리즘의 결과이다.

제안된 알고리즘의 압축효율성은 SPIHT와 거의 유사하고 작고 고정된 메모리의 사용은 제약된 메모리 사용환경에서 유용할 것이라 생각된다.

SPIHT장점 중 하나는 점진적 부호화가 가능하다는 것이다. 하지만 본 논문은 각각의 STB는 점진적 부호화지만 전체는 점진적 부호화가 아니다. 이는 앞으로 개선 해야 할 점이라 생각된다.

5. 참고문헌

[1] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchal trees." *IEEE Trans. on Circuits and Systems for Video Technology*, vol.6, pp. 243-250, June 1996

[2] F.W. Wheeler and W.A. Pearlman "SPIHT image compression without lists," *2000 IEEE International Conference* on, vol. 4 , 2000 pp. 2047-2050

[3] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, Sept. 1994, pp. 572-588.

[4] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine*, pp. 14-38, 1991.

[5] W. Sweldens and P. Schröder. "Building your own wavelets at home," *Wavelets in Computer Graphics*, pp. 15-87, ACM SIGGRAPH Course notes, 1996.

[6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE trans. Image Processing*, vol. 1, no.2, pp. 205-220, Apr. 1992

[7] I. Daubechies and W. Sweldens, "Factoring wavelet and subband transforms into lifting steps," *Journal of Fourier Analysis and Applications*, 4(3):245-267, 1998.

[8] C. D. Creusere. "Image coding using parallel implementation of the embedded zerotree wavelet algorithm," *In Proc. of the IS&T/SPIE Symposium on Electronic Imaging*, vol. 2668, 1996.

[9] J. K. Rogers and P. C. Cosman. "Wavelet zerotree image compression with packetization,"*IEEE Signal Processing Letters*, 5(5):105-107, May 1998.

[10] F.W. Wheeler and W.A. Pearlman. "Low-memory packetized SPIHT image compression," *Conference Record of the Thirty-Third Asilomar Conference* on , vol. 2, 1999 pp. 1193-1197