논문 02-27-8A-11 한국통신학회논문지 '02-8 Vol.27 No.8A

저 메모리를 갖는 제로트리 부호화

신 철*, 김 호 식*, 정회원 유 지 상*

Low Memory Zerotree Coding

Cheol Shin*, Ho-Sik Kim*, Ji-sang Yoo* *Regular Members*

요 약

SPIHT(set partitioning in hierarchical tree)[1]는 제로트리 알고리즘 중 효율적이며 잘 알려져 있다. 그러나 높은 메모리 요구로 인해 하드웨어 구현에 큰 어려움을 가지고 있다. 본 논문에서는 저 메모리 사용과 빠른 제로트리 부호화 알고리즘을 제안한다. 메모리를 줄이고 빠른 코딩을 위한 방법으로 다음 3가지를 제안한다. 첫 번째, 리프팅(lifting)을 이용한 웨이블릿(wavelet) 변환은 기존의 필터뱅크 방식의 변환보다 저 메모리와 계산량의 감소를 가진다. 두 번째 방법은 웨이블릿 계수들을 블록으로 나누어 각각 부호화 한다. 여기서 블록은 제로트리 구조가 유지되는 STB(spatial tree-based block)이다. 마지막으로 Wheeler와 Pearlmandl 제안한 NLS(no list SPIHT)[2]를 이용한 부호화이다. NLS의 효율성은 SPIHT와 거의 같으며 작고 고정된 메모리와 빠른 부호화 속도를 보여준다.

ABSTRACT

The SPIHT(set partitioning in hierarchical tree)[1] is efficient and well-known in the zerotree coding algorithm. However SPIHT's high memory requirement is a major difficulty for hardware implementation. In this paper we propose low-memory and fast zerotree algorithm. We present following three methods for reduced memory and fast coding speed. First, wavelet transform by lifting has a low memory requirement and reduced complexity than traditional filter bank implementation. The second method is to divide the wavelet coefficients into a block and to code each of them. This block is STB(spatial tree-based block) which is zerotree preserving block. Finally, we use NLS algorithm proposed by Wheeler and Pearlman in our codec. Performance of NLS is nearly same as SPIHT and reveals low and fixed memory and fast coding speed.

I. 서 론

멀티미디어 정보 중 영상의 데이터는 대용량으로 한정된 대역폭을 통하여 전송되거나 저장하기 위해서는 압축이 필요하다. 따라서 영상 신호의 중복성을 효과적으로 제거하여 높은 압축비를 얻는 반면 복구시 재구성 영상에서의 왜곡을 최소로 할 수 있는 알고리즘에 대한 연구가 끊임없이 진행되어 왔다.

웨이블릿(wavelet)은 1983년 Morlet에 의해 소개된 이후 신호를 분석하고 해석하는데 효과적인 수학적 도구로 알려져, 순수 수학 분야로부터 여러 응용분야에 이르기까지 폭 넓게 연구되어져 왔다. 신호 해석의 방법으로써 웨이블릿 변환이 소개된 이후 웨이블릿 변환을 이용한 영상 부호화에 대한 연구가 활발히 진행되고 있다. 웨이블릿 변환은 계층적인 주파수 특성을 갖는 영상 신호를 효과적으로 부호화 할 수 있다. 웨이블릿으로 변환된 계수는 서로 다른 주파수 특성을 갖고 이들간에는 어느 정도의 상관관계가 존재한다. 웨이블릿 변환을 이용하여 영상을 다해상도로 분해한 후 영상을 압축하는 것은 기존의 DCT변환 방식의 블록간 왜곡현상과 에지 열화의 단점을 극복하면서 활발히 연구되어지고 있다.

웨이블릿 변환을 이용한 부호화 기법 중 제로트

* 광운대학교 전자공학과 디지털 미디어 연구실(dustdin@image.gwu.ac.kr, hosik@image.gwu.ac.kr, jsyoo@daisy.gwu.ac.kr)
논문번호 : K01202-0917, 접수일자 : 2001년 9월 17일
※ 본 연구는 한국과학재단 목적기초연구(R01-2001-000-00350-0)지원으로 수행되었습니다.

리(zerotree) 부호화 기법은 매우 높은 압축 성능을 보이고 있다.[3-4] 이러한 알고리즘 중에서 우수한 성능을 가지는 SPIHT(set partitioning in hierarchical tree)는 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)를 제안하였다. NLS는 입력영상의 메모리 크기와 그보다 56%많은 고정된 메모리를 요구한다. 또한 리스트 대신 각 계수당 4비트의 메모리를 요구하는 상태마커를 사용하여 어떻게 복호화 해야할지를 결정하게 한다. 계수들의 인덱싱(indexing)방법은 2차원배열 인덱스(index)에서 1차원 배열 인덱스를 사용하므로 해서 몇몇 계산과 구조의 용이성을 보여준다.

본 논문에서는 변형된 NLS 알고리즘을 사용하여 NLS보다 좀더 작은 메모리 사용과 효율성을 보여준다. 이 방법은 Tanbman와 Zakhor[5]의 알고리즘에도 적용 가능 할 것이다. 웨이블릿 변환된 영상은 4개의 블록으로 나누고 각각의 블록은 독립적으로 NLS 알고리즘을 수행한다. 여기서 블록은 제로트리가 유지되는 STB(spatial tree-based block)이다. 이렇게 함으로써 생기는 메모리 사용은 NLS가 사용하는 메모리의 1/4만 필요하게 된다. STB를 각각 부호화한 비트스트림(bit stream)은 하나의 단일 비트스트림으로 재구성되어 복호기에 보내어진다. 단일 비트스트림 재구성에서 각각의 STB 비트스트림을 얼마만큼 더 할당해야 할지는 복호과정동안 주어진 비트율에서 원 영상과의 왜곡율이 최소가 되는 문제와 관련이 있다. 본 논문에서는 각 STB에 대하여 어느 일정 임계값을 선택하여 임계값보다 큰 계수의 개수의 합을 구하여 그 비율만큼의 비트를 각각의 STB에 할당하였다.

많은 메모리 사용은 하드웨어 설계시 비용의 증가를 가져온다. 좀더 단순한 웨이블릿 변환은 하드웨어 설계에서 꼭 필요한 기술이다. 리프팅(lifting)을 이용한 웨이블릿 변환은 기존의 필터뱅크 방식보다 메모리 사용이 거의 없고 계산량 또한 적다. 기존의 필터뱅크 방식은 웨이블릿 변환을 하기 위해 보조 메모리를 사용 하지만 리프팅 방식은 원 영상을 저장해 두었던 메모리에 웨이블릿 계수를 보조메모리 없이 다시 저장할 수 있다. 필터 뱅크 방식의 웨이블릿 변환은 입력신호에 대한 저주파수, 고주파수 필터를 사용하여 분해되고 저주파수 필터에 의해 분해된 신호는 같은 방식으로 반복 분해되는 방식이다. 반면 리프팅 방식은 빠른 계산을 위해 저주파수와 고주파수 필터사이의 유사성을 이용하여 웨이블릿 변환을 수행한다. 본 논문에서는 웨이블릿 변환에서 많이 쓰이는 (9-7) 필터[9]를 팩토링(factoring) 과정을 통하여 리프팅으로 변환한 후 사용하였다[10].

본 논문의 구조는 다음과 같다. 2장에서 리프팅을 이용한 웨이블릿 변환과 NLS 알고리즘에 대해 차례로 기술하고 3장에서는 이 논문에서 제안하는 알고리즘을 기술한다. 4장에서는 실험 결과를 보였고 5장에서는 결론 및 향후 연구방향 등에 대해 기술한다.

Ⅱ. 영상 압축

1. 리프팅을 이용한 웨이블릿 변환

기존의 영상 압축에서 많이 사용되어진 DCT 변환은 영상을 블록으로 잘라 처리하기 때문에 영상의 압축율이 높아지게 되면 블록화 현상이 일어나고 영상의 질이 많이 저하되는 문제점을 갖고 있다. 영상을 압축하는데 있어 웨이블릿을 이용한 방식은 기존의 DCT를 이용한 방식에 비해 블록킹 현상이 나타나지 않고 각 서브 밴드별로 처리가 가능하여 압축율을 높일 수 있다. 웨이블릿 변환은 Morlet에 의해 소개된 이후에 효율이 높고 계산량은 적은 웨이블릿에 대한 연구가 많이 진행되었다[6]. 이들 중 리프팅(lifting)을 이용한 변환은 기존 필터 뱅크 방식에 비해 계산량이 절반정도로 줄어 속도가 빠르고 메모리를 적게 사용하며 정수 대 정수 웨이블릿 변환이 용이하여 무 손실 영상 압축을 할 수 있고 역 변환을 쉽게 구현할 수 있다는 장점이 있다[7].

리프팅은 쌍직교(biorthogonal) 웨이블릿 변환의 공간축 상에서의 구현 방법론이다[8]. 리프팅을 이용한 웨이블릿 변환은 그림 1과 같이 크게 분할(split), 예측(predict), 갱신(update)의 3 단계로 구성된다. 각 단계를 수식으로 설명하면 다음과 같다.

분할(split) : 입력 신호를 이웃한 두 성분으로 분할 한다. 예를 들면, 입력 신호 $x[n]$을 짝수 번째 신호 $x_e[n]$과 홀수 번째 신호 $x_o[n]$로 분할한다.

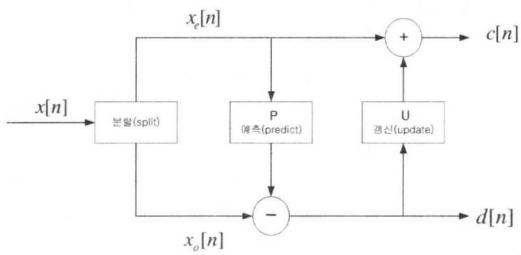


그림 1. 리프팅의 구조도

예측(predict) : 예측 연산자 P를 이용하여 $x_e[n]$로부터 $x_o[n]$를 예측할 때 얻어지는 에러인 웨이블릿 계수 $d[n]$를 구한다.

$$d[n] = x_o[n] - P(x_e[n])$$

갱신(update) : 입력 신호 $x[n]$을 근사화하여 나타내는 스케일링(scaling) 계수 $c[n]$를 얻기 위해 $x_e[n]$과 $d[n]$을 결합한다. 이것은 웨이블릿 계수에 갱신 연산자 U를 적용한 다음 $x_e[n]$를 더하여 구한다.

$$c[n] = x_e[n] + U(d[n])$$

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

본 논문에서 사용된 필터는 웨이블릿 변환에서 많이 사용되어지는 (9-7)필터를 사용하였다. 표 1에 (9-7) 필터의 분석과 합성 필터 계수를 각각 나타내었다. 이 필터 계수들을 팩토링(factoring) 알고리즘을 적용시켜 리프팅으로 변환시키면 아래 수식과 같이 된다. 입력 신호를 x_i이라 하면,

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

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

위 수식을 구조적으로 나타내면 그림 2와 같이 된다. 수식을 간단히 설명하면, 들어온 입력 신호 x_i을 짝수 번째 신호와 홀수 번째 신호로 나누어 각각 $s_i^{(0)}$과 $d_i^{(0)}$이라 하고 이웃하는 짝수번째 신호 $s_i^{(0)}$과 $s_{i+1}^{(0)}$을 이용하여 $d_i^{(2)}$을 예측한다. 다음 갱신 단계에서는 이웃하는 홀수번째 신호 $d_{i-1}^{(1)}$과 $d_i^{(1)}$ 을 이용하여 $s_i^{(1)}$를 구한다. 위 예측과 갱신 단계를 한 단계씩 더 반복하여 구한 $s_i^{(2)}$에 스케일링 계수 ζ를 곱하고 $d_i^{(2)}$에는 ζ를 나누어 최종적인 계수값들을 구한다.

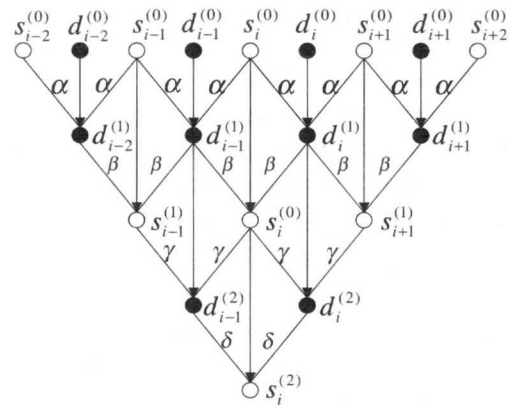


그림 2. (9,7) 필터의 리프팅 구조

2. NLS 알고리즘

입력영상에 대해 웨이블릿 변환 후 NLS는 비트 플레인(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 Descendants

표 2. 상태 마커

State Marker	Description
MIP	임계값보다 작거나 아직 테스트되지 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

의 최대값들의 배열의 크기는 XYC/4이고 Grand-descendants의 최대값들의 배열의 크기는 XYC/16 이다. 또한 계수의 크기와 부호를 분리 저장하기 위해 XYC만큼 메모리가 필요하며 상태 마커는 XY/2가 필요하다. 따라서 NLS를 수행하는 동안 입력 영상 크기만큼의 영상 메모리와 그 보다 56% 많은 메모리가 필요하게 된다.

일반적인 영상 부호화기에서 영상을 2차원 배열의 인덱싱을 이용하여 처리하지만 NLS 알고리즘에서는 보다 계산의 용이성을 위해 1차원 배열의 인덱싱을 사용한다. 그림 3은 2차원 배열을 1차원 배열로 바꾸는 방법을 나타내고 있다. 영상의 수평과 수직방향의 크기를 $R=C=2^N$이라고 하고 r과 c는 2차원 배열의 수평과 수직의 인덱스라 하자. 그러면 수평방향 인덱스 r의 이진표현은 $r=[r_{Lr-1},\cdots,r_1,r_0]$로 나타내어지고, 또한 수직방향의 인덱스 c의 이진 표현은 $c=[c_{Lr-1},\cdots,c_1,c_0]$과 같이 표현된다. 이것을 1차원 배열로 나타내면 $i=[r_{Lr-1},c_{Lr-1},\cdots,r_1,c_1,r_0,c_0]$ 로 정의할 수 있다. 즉 2차원 배열의 수평, 수직 인덱스 r과 c의 각 비트를 교차시켜 1차원 배열의 인덱스 i를 만들어 낸다.

NLS 알고리즘을 간단히 설명하면 각 비트 플레인마다 3개의 패스가 존재한다. 첫 번째는 IP(insignificant pixel) 패스이다. SPIHT의 LIP 패스와 유사하다. 각 계수들 중 상태 마커가 MIP인 것을 찾아 임계값 보다 크면 MNP를 할당한다. 두 번째는 IS(insignificant set) 패스는 SPIHT의 LIS 패스와

	0	1	2	3	4	5	6	7
0	0	1	4	5	16	17	20	21
1	2	3	6	7	18	19	22	23
2	8	9	12	13	24	25	28	29
3	10	11	14	15	26	27	30	31
4	32	33	36	37	48	49	52	53
5	34	35	38	39	50	51	54	55
6	40	41	44	45	56	57	60	61
7	42	43	46	47	58	59	62	63

그림 3. R=8, C=8인 1차원 배열의 인덱싱

유사하다. 각 계수들을 조사하여 MD이고 임계값보다 크면 자기를 포함하는 4개의 계수들을 MCP로 하고 직계 Descendant는 MG로 한다. 계수의 상태 마커가 MG이고 임계값보다 크면 자기를 포함하는 4개의 계수들을 MD로 할당한다. 그렇지 않고 계수가 MCP를 가진다면 임계값을 조사하여 크면 MNP를 할당하고 작으면 MIP를 할당한다. 마지막으로 REF(refinement) 패스이다. 이것은 SPIHT의 LSP 패스와 유사하다. 계수가 MSP이면 계수값의 더 정확한 값을 보내게 된다. 계수가 MNP이면 MSP로 바꾼다. 이때 세 패스를 원하는 비트율이 될 때까지 반복한다.

Ⅲ. 제안된 알고리즘

그림 4에서 보듯이 제안된 알고리즘의 기본적인 생각은 웨이블릿 계수를 4개의 블록으로 나누는 것이다. 그리고 각 블록마다 독립적으로 NLS 알고리즘을 수행하고 각각의 비트 스트림을 조합하여 전송단에 보낸다.

1. STB(Spatial Tree-Based Block)

웨이블릿 계수를 블록으로 나눌 때 가장 중요한 것은 트리 구조가 그대로 유지되면서 블록으로 나누는 것이다. 이러한 생각은 몇몇 부호화기에서 사용되었다. Creusere는 병렬처리를 위한 EZW에 이 기술을 사용했고[11], Rogers와 Cosman은 error resilie nce를 위해 유사한 방법을 사용했다[12]. Wheeler와 Pearlman은 저 메모리를 위한 SPIHT를 위해 이 기술을 이용했다[13].

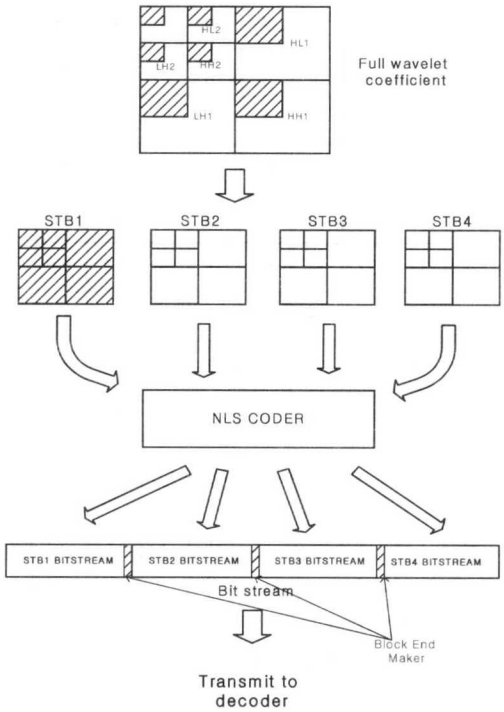


그림 4. 제안된 알고리즘 구조

본 논문은 빠른 알고리즘 수행속도와 고정된 저 메모리를 갖는 NLS 알고리즘을 더욱더 작은 메모리를 가지면서 압축 효율성 또한 거의 같은 효과를 보기 위해 이 기술을 이용한다. 그림 4에서 보듯이 4개의 블록 STB1, STB2, STB3, STB4로 나눔으로 해서 메모리는 NLS가 필요로 하는 크기의 1/4만 있으면 된다. 즉 입력영상의 크기 메모리와 그 보다 14%더 많은 메모리가 필요로 하게 된다. 따라서 NLS보다 적은 메모리 사용이 가능하다.

필터뱅크 방식의 웨이블릿 변환 구조는 그림 5와 같이 제로트리 구조를 가지지만 리프팅의 결과는 그렇지 않다. 그림 5는 필터뱅크방식의 웨이블릿 구조와 리프팅의 웨이블릿 변환 구조와의 차이를 3레벨까지 웨이블릿 변환한 8*8영상을 이용하여 나타내었다. 본 논문에서는 그림 5와 같이 리프팅에 의한 웨이블릿 변환 구조를 곧바로 1차원 배열의 인덱싱을 가지는 배열로 바꾸었다. 그림 4는 제안된 알고리즘의 기본적인 생각을 그림으로 나타낸 것이지 그림대로 알고리즘을 수행한 것이 아니다. 즉, 리프팅에 의해 웨이블릿 변환된 계수는 곧바로 NLS 부호화기의 메모리로 들어가 중간에 임시 저장하는 메모리는 필요없게 된다.

LL3 0	HL3 1	HL2 4	HL2 5	HL1 16	HL1 17	HL1 20	HL1 21
LH3 2	HH3 3	HL2 6	HL2 7	HL1 18	HL1 19	HL1 22	HL1 23
LH2 8	LH2 9	HH2 12	HH2 13	HL1 24	HL1 25	HL1 28	HL1 29
LH2 10	LH2 11	HH2 14	HH2 15	HL1 26	HL1 27	HL1 30	HL1 31
LH1 32	LH1 33	LH1 36	LH1 37	HH1 48	HH1 49	HH1 52	HH1 53
LH1 34	LH1 35	LH1 38	LH1 39	HH1 50	HH1 51	HH1 54	HH1 55
LH1 40	LH1 41	LH1 44	LH1 45	HH1 56	HH1 57	HH1 60	HH1 61
LH1 42	LH1 43	LH1 46	LH1 47	HH1 58	HH1 59	HH1 62	HH1 63

(a)

LL3 0	HL1 16	HL2 4	HL1 17	HL3 1	HL1 20	HL2 5	HL1 21
LH1 32	HH1 48	LH1 33	HH1 49	LH1 36	HH1 52	LH1 37	HH1 53
LH2 8	HL1 18	HH2 12	HL1 19	LH2 9	HL1 22	HH2 13	HL1 23
LH1 34	HH1 50	LH1 35	HH1 51	LH1 38	HH1 54	LH1 39	HH1 55
LH3 2	HL1 24	HL2 6	HL1 25	HH3 3	HL1 28	HL2 7	HL1 29
LH1 40	HH1 56	LH1 41	HH1 57	LH1 44	HH1 60	LH1 45	HH1 61
LH2 10	HL1 26	HH2 14	HL1 27	LH2 11	HL1 30	HH2 15	HL1 31
LH1 42	HH1 58	LH1 43	HH1 59	LH1 46	HH1 62	LH1 47	HH1 63

(b)

그림 5. 8×8 영상의 3레벨 웨이블릿 변환한
(a) 필터뱅크 방식 과 (b)리프팅 방식의 차이

2. 비트 할당(Bit Allocation)

그림 4에서 각각의 STB들의 비트스트림(bit stream)을 하나의 단일 비트스트림으로 만들 때 바이트 단위나 패킷단위로 각 블록들의 비트스트림값을 서로 교차(interleave)하면서 만들지 않고 하나의 STB 비트스트림 뒤에 다음 STB 비트스트림을 붙이는 형식으로 단일 비트스트림을 만들었다. 그리고 각 STB의 비트스트림을 구별하기 위해 STB 비트스트림 사이에는 마커를 삽입했다. 따라서 복호 과정에서 주어진 비트율에서 복원영상과 원 영상의 왜곡율이 최소가 되는 각 STB들의 비트를 어떻게 주어져야 할지가 주요 관점이다.

여기서는 아주 간단한 방법으로 정해진 비트율에서 각 STB의 비트스트림이 얼마만큼의 비트를 할당 할 것인지를 결정한다. 4개의 STB에서 큰 계수

의 값들이 많은 STB에 더 많은 비트를 할당해야 하는 것은 당연하다. 그림 6은 512×512 lenna 영상의 각 STB별 계수값들의 분포를 계수값이 -100에서 100사이의 값들만 나타내었다. 그림 6에서 보면 각각의 STB는 서로 다른 계수의 분포를 가짐을 알 수 있다. STB3의 분포는 다른 STB보다 큰 계수가 많음을 알 수 있다. 이것은 STB3의 부호화 과정에서 더 많은 비트를 할당해야 함을 보인다.

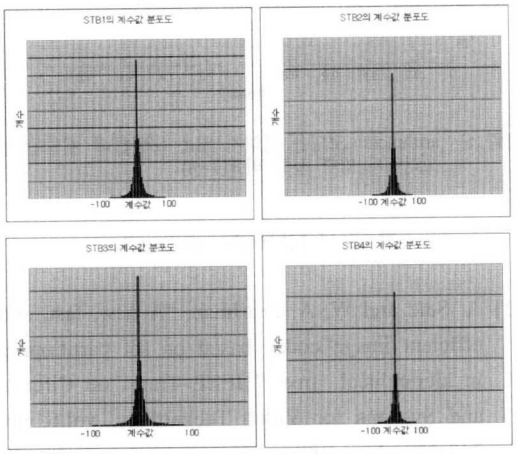


그림 6. 512×512 lenna 영상의 각 블록별 계수값 분포도

본 논문에서는 각각의 STB의 큰 계수값들의 많고 적음을 어느 일정한 임계값을 선택하여 임계값 이상의 계수들의 수를 합하여 그 크기만큼의 비율로 각 블록에서 얼마만큼의 비트를 할당할지를 결정했다. 즉, 4개의 STB에서 어느 일정한 임계값 T보다 큰 계수의 개수의 합을 각각 s_1, s_2, s_3, s_4 라하고 전체 개수의 합 $t_c = s_1 + s_2 + s_3 + s_4$ 된다. 복호 과정에서 주어진 비트가 B라면 각각의 STB에 주어지는 비트는 각각

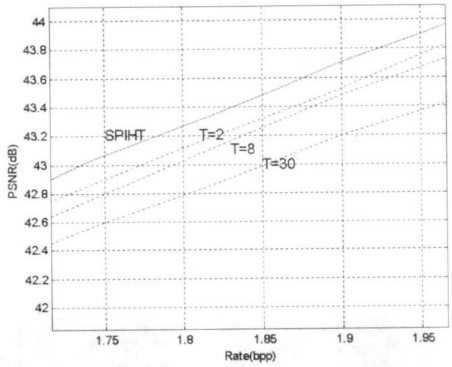
$$STB1_B = \frac{B s_1}{t_c}, \quad STB2_B = \frac{B s_2}{t_c}$$

$$STB3_B = \frac{B s_3}{t_c}, \quad STB4_B = \frac{B s_4}{t_c}$$

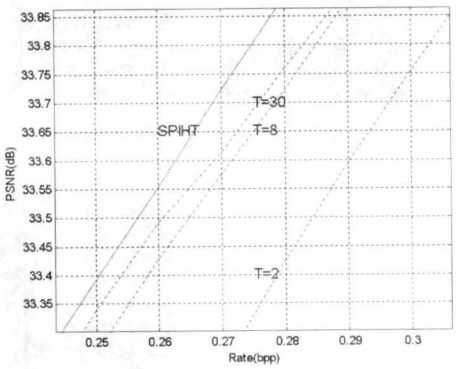
가 된다.

Ⅳ. 실험 결과 및 분석

주어진 비트에서 각각의 STB에 얼마만큼의 비트를 할당하여 부호화 할지를 결정하기 위해 어느 일정한 임계값을 구해야 한다. 그림 7은 512×512 lenna 영상을 이용하여 5레벨까지 웨이블릿 분해한 후 여러 가지 임계치를 적용하여 arithmetic 코딩을 행하지 않은 SPIHT와 제안된 알고리즘의 결과를 나타낸다. 실선은 SPIHT의 결과이고 점선은 임계치를 T=2, 8, 30으로 행한 제안된 알고리즘의 결과이다. 그림 7은 화소당 할당되는 비트가 많을 때 (a)와 적을 때 (b)의 경우 임계치에 따른 결과가 다르게 나왔다. 그림 7의 (a)를 보면 1.85bpp에서 임계치 T=2인 경우가 T=8 또는 T=30인 경우 보다 좋게 나왔고 그림 7의 (b)를 0.28bpp에서보면 T=30인 경우가 T=2 또는 T=8인 경우 보다 좋은 결과가 나왔다. 이것은 많은 비트를 할당하여 복호화 할 경우 NLS 알고리즘은 웨이블릿 계수가 작은 값들도 부호화 한다는 의미이므로 각각의 STB에 할당해야할 비트를 결정하기 위한 임계값 또한 작아져야 하고 적은 비트를 할당하여 복호화 할 경우는 웨이블릿 계수의 큰 값들만 부호화 되므로 각각의 STB에 할당 해야할 비트를 결정하기 위한 임계값은 커져야 한다는 의미이다.



(a) 많은 비트로 복호화 한 경우



(b) 적은 비트로 복호화 한 경우

그림 7. arithmetic 코딩을 적용하지 않은 SPIHT와 512x512 lenna영상의 여러 가지 임계치를 적용한 제안한 알고리즘의 압축 성능 비교

그림 8은 512×512 lenna, goldhill 그리고 barbara 영상을 이용하여 각각의 STB에 할당 해야 할 비트를 결정하기 위한 임계값 T=8로 하고 5레벨까지 웨이블릿 분해하여 실험한 결과이다. 실선은 SPIHT의 결과이고 점선은 제안된 알고리즘의 결과이다. 제안된 알고리즘의 시각적 비교를 위해 그림 9에 512×512 lenna 영상을 다양한 압축비를 적용하여 SPIHT와 비교하여 나타내었다. 그림 9에서 시각적으로 SPIHT와 제안된 알고리즘의 결과가 거의 같음을 알 수 있다.

V. 결 론

본 논문에서는 리프팅을 이용한 웨이블릿 변환과 변형된 NLS 알고리즘을 이용하여 저 메모리 제로트리 부호화 기법을 제안하였다. 리프팅을 이용한 웨이블릿 변환은 메모리 사용이 거의 없고 변형된 NLS알고리즘은 원 NLS 알고리즘보다 1/4적은 메모리 사용을 보여 주었다. 또한 제안된 알고리즘의 압축 효율성은 SPIHT와 거의 유사함을 보여 주었다. 본 논문에서 제안된 알고리즘은 제약된 메모리 사용환경에서 유용할 것이라 생각된다.

SPIHT 장점 중 하나는 점진적 부호화가 가능하다는 것이다. 하지만 본 논문은 각각의 STB는 점진적 부호화지만 전체는 점진적 부호화가 아니다. 이는 앞으로 연구해야 할 점이라 생각된다.

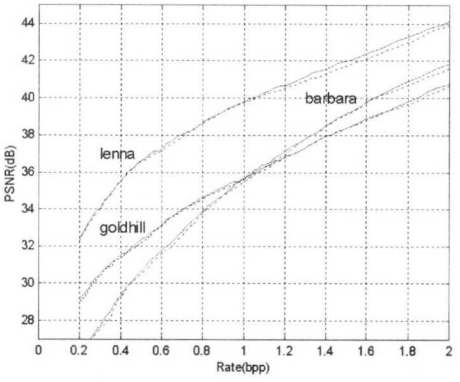


그림 8. arithmetic 코딩을 행하지 않은 SPIHT와 제안된 알고리즘의 압축 성능 비교(T=8)

참 고 문 헌(References)

[1] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical 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. 2047-20 50

[3] J. Shapiro, "Embedded image coding using

(a) 원 영상 (b) SPIHT 0.25bpp (c) SPIHT 0.12bpp (d) SPIHT 0.06bpp

(e) 제안한 방법 0.25bpp (f) 제안한 방법 0.12bpp (g) 제안한 방법 0.06bpp

그림 9. T=8일 때 여러 가지 압축비로 복원한 512×512 lenna영상

zerotrees of wavelet coefficients," IEEE Trans. On Signal Processing, vol.41, no.12, pp.3445-3463, December 1993.

[4] Z.Xiong, K. Ramchandran, and M. Orchard, "Space-frequency quantization for wavelet image coding," IEEE trans. Image Processing, vol. 6, pp. 677-693, May 1997

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

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

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

[8] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," J. Appl. Comp. Harm. Anal., vol. 3, no. 2, pp. 186-200, 1996.

[9] 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

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

[11] 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.

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

[13] 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

신 철(Cheol Shin)
2000년 : 광운대학교 제어계측공학과 학사
2002년 : 광운대학교 전자공학과 석사
2002년~현재 : TLI 연구원
<주관심 분야> 신호 및 영상처리, 보안 시스템

김 호 식(Ho-sik Kim)
2001년 : 광운대학교 제어계측공학과 학사
2001년~현재 : 광운대학교 전자공학과 석사과정
<주관심 분야> 신호 및 영상처리, 영상 압축

유 지 상(Ji-sang Yoo)
1985년 : 서울대학교 전자공학과 학사
1987년 : 서울대학교 전자공학과 석사
1993년 : Purdue 대학교 전기공학과 박사
1997년~현재 : 광운대학교 전자공학과 부교수
<주관심 분야> 비선형 신호처리, 신호 및 영상처리