BFS를 이용한 추가 메모리를 요구하지 않는 제로트리 압축기법

(Zero-tree Packetization without Additional Memory using BFS)

김 충 길 † 정 기 동 ††
(Chung-Gil Kim) (Ki-Dong Chung)

요 약 SPIHT는 수행속도가 빠르고 효율적인 웨이블릿 기반의 이미지 압축 알고리즘으로 잘 알려져 있다. 그러나, SPIHT는 알고리즘을 수행하는 과정에서 발생하는 제로트리 및 계수의 상태를 저장하기 위하여 리스트 구조를 사용하고 있어 추가 메모리를 요구하며, 비트율의 증가에 따라 메모리 요구량이 증가하는 단점을 가진다. 본 논문에서는 SPIHT 알고리즘을 수행하는데 있어 추가 메모리를 요구하지 않는 MZC-BFS 알고리즘을 제안한다. 제안된 기법은 peano 코드를 이용하여 완벽한 너비우선 순서에 따라 공간트리를 탐색하며, 부호화 과정에서 이전상태 중요계수 테스트 및 복원과정에서 계수의 LSB를 이용함으로써 SPIHT에서 리스트 문제를 제거한다. MZC-BFS는 SPIHT에 비하여 리스트를 사용하지 않기 때문에 하드웨어 구현이 간단하고 수행속도가 빠를 뿐 아니라 추가 메모리를 요구하지 않기 때문에 하드웨어 제작 비용을 절감할 수 있다.

키워드 : 이미지 압축, SPIHT, 제로트리

Abstract SPIHT algorithm is a wavelet based fast and effective technique for image compression. It uses a list structure to store status information which is generated during set-partitioning of zero-tree. Usually, this requires lots of additional memory depending on how high the bit-rate is. Therefore, in this paper, we propose a new technique called MZC-BFS, which needs no additional memory when running SPIHT algorithm. It explicitly performs a breadth first search of the spatial-tree using peano-code and eliminates additional memory as it uses pre-status significant test for encoding and LSB bits of some coefficients for decoding respectively. This method yields nearly the same performance as SPIHT. This may be desirable in fast and simple hardware implementation and reduces the cost of production because no lists and additional memory are required.

Key words : Image Compression, SPIHT, Zero-tree

1. 서 론

JPEG와 같은 기존의 영상압축 표준은 DCT 변환에 기반하고 있다. JPEG은 낮은 비트율에서 블록의 경계가 눈에 두드러지는 블록킹 현상이 나타난다. 이는 JPEG에서 사용하는 DCT가 영상 이미지를 8×8 블록으로 분해한 후 각각의 블록을 독립적으로 부호화 하기 때문이다. 최근 DCT의 블록킹 효과를 해결하기 위한 기법으로 웨이블릿 변환을 이용한 이미지 압축기법에 대한 연구가 활발히 진행되고 있다. 웨이블릿 변환에 기반한 이미지 압축기법으로 잘 알려진 EZW(Embedded ZeroTree Wavelet)[1] 및 SPIHT(Set Partitioning In Hierarchical Trees)[2] 등은 동일한 비트율에서 JPEG 보다 우수한 이미지 품질을 생성한다.

EZW와 SPIHT는 알고리즘이 간단하고 속도가 빠른 이미지 압축기법으로 잘 알려져 있다. 이들 알고리즘은 웨이블릿 변환된 영상의 부대역 사이에 존재하는 자기-유사성(self-similarity)을 이용하여 제로트리를 정의하고, 이를 이용하여 보다 지배적인 에너지를 가진 계수를 우선적으로 부호화 함으로써 임베디드 코드를 생성한다. 따라서 점진적인 이미지 전송이 가능하며, 정확히 원하는 비트율에서 압축과 복원을 중단할 수 있다. 이들 기법은 웨이블릿 변환된 이미지에서 보다 지배적 에너지를 가진 계수를 우선적으로 분리시키기 위하여 중요계수 테스트(significant test)를 수행하며, 중요계수 테스트의 결과에 따라 계수집합 분할 규칙(set partitioning

† 비 회 원 : 부산대학교 멀티미디어협동과정
chgkim@melon.cs.pusan.ac.kr
†† 종신회원 : 부산대학교 전자전기정보컴퓨터공학부 교수
kdchung@pusan.ac.kr
논문접수 : 2003년 9월 3일
심사완료 : 2004년 1월 19일

rule)을 적용한다. SPIHT는 계수집합 분할 규칙에서 EZW 보다 효율적이며, 동일한 비트율에서 보다 우수한 품질의 복원 이미지를 생성한다.

SPIHT는 중요계수 테스트 과정에서 테스트될 계수집합과 계수의 상태를 유지하기 위하여 리스트 구조를 사용하고 있으며, 리스트를 처리하기 위한 관리기능과 리스트 저장을 위한 추가의 메모리를 요구하고 있다. SPIHT에서 요구되는 리스트의 최대 요구량은 이미지의 형태 및 요구된 압축 비트율에 의존적이기 때문에 요구되는 추가 메모리의 크기는 가변적이다. 이러한 추가 메모리의 가변적 요구 특성은 SPIHT 알고리즘을 하드웨어로 구현할 때 문제점으로 지적되고 있다.

그 동안 SPIHT의 리스트 사용에 따른 문제점을 해결하기 위한 연구가 진행되었다. Wheeler[3]는 웨이블릿 이미지를 4개의 동일 크기의 공간블록(spatial block)으로 분할함으로써 리스트의 최대 요구량을 줄이는 기법을 제안하였으며, Lin[4], Kutil[5], Wheeler[6]는 이미지의 픽셀 수에 대응되는 일정한 크기의 상태맵을 이용하여 SPIHT에서 리스트를 제거한 기법을 제안하였다.

본 논문에서는 리스트 및 상태맵과 같은 추가 메모리를 요구하지 않으면서 SPIHT 알고리즘을 수행할 수 있는 MZC-BFS(Memoryless Zero-tree Compression using Breadth First Search) 알고리즘을 제안한다. MZC-BFS는 peano 코드를 이용하여 공간트리를 너비우선에 따라 탐색한다. 본 논문에서는 SPIHT 알고리즘의 수행에 있어 리스트를 사용함으로 발생하는 문제점을 제거하기 위한 기법으로 압축과정에 있어서 이전상태 중요계수 테스트 기법을 제안하며 복원과정에 있어서 계수의 LSB를 사용한 상태 저장과 이를 위한 상태정보 배치기법을 제안한다. 본 논문의 구성은 제2장에서는 관련연구에 대하여 간략히 살펴보고, 제3장에서는 MZC-BFS 알고리즘을 기술하며, 제4장에서는 실험을 통하여 본 논문에서 제안한 MZC-BFS 알고리즘의 성능을 평가한다. 그리고 5장에서는 결론을 맺는다.

2. 관련연구

2.1 제로트리 압축 기법

일반적인 이미지는 저주파 성분이 강한 특성을 가진다. 피라미드 분해된 웨이블릿 이미지는 서로 대응되는 부대역간에 공간적 자기-유사성(spatial self-similarity)을 가지며, 그림 1과 같이 동일한 위치를 참조하는 계수에 대하여 부모-자식 관계를 가지는 공간트리(spatial orientation tree)를 형성한다.

Shapiro[1]는 임계치(threshold)를 주어 계수(coefficient)의 절대치가 임계치보다 작은 경우를 비중요계수(insignificant), 반대의 경우를 중요계수(significant)로

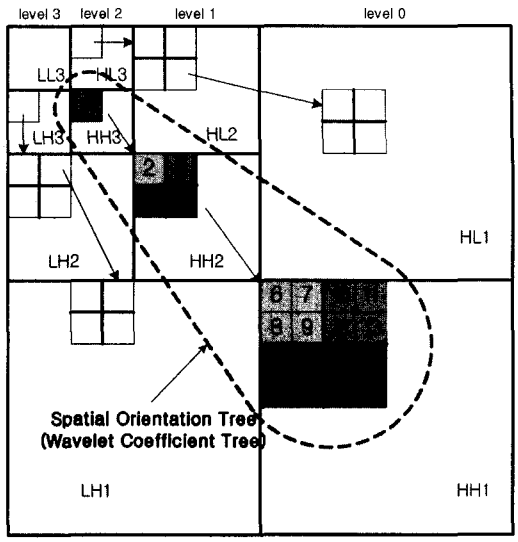


그림 1 웨이블릿 이미지에서 공간트리의 예

정의하고 비중요계수로 구성된 공간트리를 제로트리(zerotree)로 정의하였으며, 이를 기반으로 EZW 알고리즘을 제안하였다. EZW는 정해진 스캔 순서에 따라 계수 및 공간트리에 대한 중요계수 테스트(significant test)를 수행하며, 제로트리 조건을 만족하지 못한 공간트리에 대하여 계수집합 분할 규칙(set partitioning rule)을 적용한다. EZW는 계수집합 분할 과정을 통하여 보다 지배적인 에너지를 가진 계수를 우선적으로 부호화 함으로써 점진적인 이미지 복원이 가능한 임베디드 코드를 생성한다.

SPIHT는 보다 향상된 계수집합 분할 규칙을 통하여 EZW 보다 효율적인 비트스트림을 생성한다. SPIHT는 계수집합 분할 규칙에 필요한 상태를 유지하기 위하여 3개의 리스트 LIS(List of Insignificant Set), LIP(List of Insignificant Pixels) 및 LSP(List of Significant Pixels)를 사용한다. LIS 리스트는 부호화 과정에서 탐색된 모든 제로트리를 원소로 가지며, LIP와 LSP는 각각 공간트리에서 분리된 비중요계수와 중요계수를 원소로 가진다. SPIHT 알고리즘에서 중요계수 테스트는 임계치를 2^n, 좌표 (r,c)에 대한 계수값을 $c_{r,c}$, 계수집합을 T라고 할 때

$$S_n(T)=\begin{cases}1, \max\limits_{(r,c)\in T}\{|c_{(r,c)}|\}\geq 2^n\\ 0, otherwise\end{cases} \quad (1)$$

식 (1)과 같이 정의된다.

SPIHT 알고리즘의 계수집합 분할 규칙은 그림 2와 같으며, 그림 2의 알고리즘에서 사용된 계수집합은 그림 3과 같이 정의된다. $O(r,c)$는 노드 (r,c)의 자식노드

(offspring)를 원소로 가지는 집합을 말하며, $D(r,c)$는 (r,c)가 루트인 공간트리에서 루트 (r,c)를 제외한 노드들로 구성된 집합을 의미한다.

```
for all T in LIS
  if (T is D tree)
    if (S_n(D(r,c)) = 1)
      append all coefficient (k,l) ∈ O(r,c) to LIP or LSP
      remove D(r,c) and append L(r,c) to LIS
  else
    if(S_n(L(r,c)) = 1)
      remove L(r,c) and append all subtree D(k,l) to LIS
```

그림 2 SPIHT의 계수집합 분할 규칙

$O(r,c)$: set of coordinates of all offspring of node (r,c)

$D(r,c)$: set of coordinates of all descendants of the node (r,c)

$L(r,c)$: $D(r,c)-O(r,c)$

그림 3 SPIHT의 계수집합

SPIHT는 부호화 과정에서 리스트 구조를 사용하고 있기 때문에 리스트 관리기능 및 저장을 위한 추가 메모리를 요구한다. 요구되는 메모리의 크기는 이미지의 형태와 압축 비트율에 의존적이다. 즉, 비트율이 높아질 경우 요구되는 추가 메모리의 최대요구량은 증가하며 이미지의 형태에 따라 요구되는 추가 메모리의 최대 요구량은 가변적이다. 이러한 메모리 요구특성은 SPIHT 알고리즘의 하드웨어 구현에 있어서 문제점으로 지적되고 있다.

SHPIT에서 리스트의 사용에 따른 문제점을 해결한 대표적인 연구로는 SMAWZ(Significance Map Based Adaptive Wavelet Zerotree)[5]와 NLS(No List SPIHT)[6]가 있다. SMAWZ는 이미지의 픽셀 수를 i개 라고 할 때, 각각 i비트, $i/4$비트, $i/16$비트 용량을 가지는 3개의 비트맵과 공간트리에 대한 깊이우선탐색(Depth First Search)을 사용하여 SPIHT에서 리스트를 제거하였으며, NLS는 선형 인덱싱(linear-indexing) 주소기법, 공간트리에 대한 너비우선탐색(Breadth First Search)과 픽셀당 4비트의 상태맵을 사용한다. 이들 알고리즘에서 사용된 상태맵의 크기는 이미지의 형태 및 압축 비트율과 무관하며, 고정된 크기의 메모리를 요구하고 있다. 그러나 상태맵은 이미지의 픽셀수에 의존적이기 때문에 이미지가 커질 경우 상태맵을 위한 메모리 요구량이 증가하는 문제점을 가진다.

2.2 Peano 인덱스

일반적으로 제로트리에 기반한 압축기법에서 공간트리에 대한 너비우선 탐색은 중요계수를 우선적으로 탐색할 수 있기 때문에 깊이우선 탐색 보다 우수한 성능을 가진다[6]. SPIHT는 공간트리를 분할할 때 분할된 서브트리를 리스트의 끝에 추가하기 때문에 완벽한 너비우선 탐색을 수행하지 못한다. 그러나, Peano 코드[7]를 이용할 경우 간단한 산술연산을 이용하여 공간트리에 대한 완벽한 너비우선 탐색을 쉽게 구현할 수 있다.

일반적으로 이미지의 픽셀은 2차원 좌표에 의하여 참조되어진다. 크기가 $R\times C$인 이미지에서 행과 열의 좌표 r, c에 대한 2진 표현을 $r_{m-1}r_{m-2}\cdots r_1r_0$, $c_{m-1}c_{m-2}\cdots c_1c_0$, ($r_i$, c_i는 비트)라고 할 때, 픽셀 (r, c)에 대한 peano 인덱스 i는 $r_{m-1}c_{m-1}\cdots r_1c_1r_0c_0$로 표현된다. 즉 r과 c의 비트를 인터리빙 함으로써 간단히 구할 수 있다.

Peano 코드는 공간트리의 처리에 적합한 몇가지 연산을 제공한다. 먼저, 계수에 대한 인덱스의 증가를 통하여 공간트리의 너비우선탐색을 수행할 수 있어 2차원 좌표에 비하여 계산이 간단하고 수행 속도가 빠르다. 또한, Peano 코드는 임의의 노드에 대하여 부모노드와 자식노드의 좌표를 쉽게 계산할 수 있다. 공간트리가 정확히 4개의 자식노드를 가지기 때문에 임의의 노드 i에 대한 부모노드의 인덱스는 $i/4$이고, 자식노드의 인덱스는 $4i+k(k=0,1,2,3)$가 된다.

3. MZC-BFS 알고리즘

본 논문에서는 SPIHT 알고리즘에서 계수집합 분할 과정과 리스트에 속한 노드들의 존재조건을 관찰함으로써 리스트를 사용하지 않고 제로트리를 부호화 할 수 있는 MZC-BFS 알고리즘을 제안한다. MZC-BFS는 이미지 압축 과정에 있어서 SPIHT 알고리즘의 리스트의 역할을 대신할 수 있는 이전상태 중요계수 테스트 기법을 사용하며, 복원과정에 있어 리스트를 제거하기 위한 기법으로 복원 계수의 LSB의 사용과 이를 위한 주소변환 기법을 사용한다. 이들 기법은 SPIHT 알고리즘의 수행에 있어서 리스트를 제거할 뿐 아니라 추가의 메모리를 요구하지 않고 SPIHT의 리스트와 동일한 기능을 수행할 수 있다.

SPIHT의 계수집합 분할 규칙에서 임의의 노드 g를 루트로 하는 공간트리 $D(g)$는 $S_n(D(g))=1$일 때 4개의 자식노드 $i(i\in O(g)$, $g=i/4)$와 노드 g를 루트로 하는 1개의 공간트리 $L(g)$로 분할되며, 공간트리 $L(g)$는 $S_n(L(g))=1$일 때 자식노드 i를 루트로 하는 4개의 서브트리 $D(i)$로 분할된다. 다시 말해서, 노드 i는 $S_n(D(g))$의 결과가 1일 때 부호화 과정에 참여하게 된다. 또한, 공간트리 $D(i)$는 $S_n(L(g))=1$일 때 부호화 과정에 참여하게 된다. 따라서 이전상태 중요계수 테스트 $S_n(D(i/4))$는 리스트를 사용하지 않고 노드 i에 대한 부호화 시작 시점을 결정하며, $S_n(L(i/4))$는 $D(i)$에 대한 부호화 시작

시점을 결정할 수 있다.

SPIHT 알고리즘에서 임계치가 2^n일 때, 노드 i가 LIP의 원소이면, 모든 $m(n<m)$에 대하여 $S_m(i)=0$을 만족한다. 또한 노드 i가 LSP의 원소라면, $S_m(i)=1$을 만족하는 적어도 하나의 m이 존재하고 모든 $m'(n<m'\leq m)$에 대하여 $S_{m'}(i)=1$을 만족한다. 따라서 $S_{n+1}(i)$가 0이면 노드 i는 LIP의 원소이고, 1이면 LSP의 원소임을 알 수 있다. SPIHT에서 노드 i에 대한 부호화 과정은 노드 i가 속한 리스트에 의하여 결정되기 때문에 이전상태 중요계수 테스트 함수 $S_{n+1}(i)$는 노드 i에 대하여 요구되는 부호화 과정이 주부호화 과정인지 종속부호화 과정인지를 결정할 수 있다.

SPIHT의 계수집합 분할 규칙에서 임계치가 2^n일 때, 노드 i를 루트로 하는 공간트리 $D(i)$가 LIS의 원소이면 모든 $m(n<m)$에 대하여 $S_m(D(i))=0$을 만족한다. 또한 공간트리 $D(i)$가 LIS의 원소가 아니라면 적어도 하나의 m이 존재하고 모든 $m'(n<m'\leq m)$에 대하여 $S_{m'}(D(i)=1)$을 만족한다. 즉, $S_{n+1}(D(i))$는 공간트리 $D(i)$의 인코딩 동작을 결정할 수 있으며, 공간트리 $L(i)$에 대하여 동일한 개념을 적용할 수 있다.

MZC-BFS는 너비우선 탐색 과정에서 제로트리에 속한 노드들에 대한 효율적 탐색을 위하여 $Skip(i)$ 함수를 사용한다. 임의의 노드 i가 제로트리의 원소라면 노드 i의 모든 형제노드 또한 제로트리의 원소이기 때문에 이들 노드들은 인코딩 과정에서 제외된다. 노드 i가 제로트리의 원소일 때 인코딩 과정에서 제외될 형제노드의 개수를 구하기 위한 $Skip(i)$ 함수는 그림 4와 같이 정의된다.

```
Skip(i)
  if S_n(D(i/4)) = 0
    return 4×Skip(i/4)
  else
    return 1
```

그림 4 MZC-BFS의 $Skip(i)$

MZC-BFS의 인코딩 알고리즘을 간략히 기술하면 그림 5와 같다.

MZC-BFS의 인코딩 알고리즘에서 사용된 이전상태 중요계수 테스트 함수 $S_{n+1}(T)$는 계수값을 기반으로 하기 때문에 복원 알고리즘에는 적용할 수 없다. 따라서 본 논문에서는 이미지 복원과정에서 구성된 공간트리 $D(i)$와 $L(i)$에 대한 상태정보를 일부 계수의 LSB를 사용함으로써 추가 메모리를 요구하지 않고 이미지를 복원하기 위한 공간트리 상태정보 배치기법을 제안한다.

```
for each refinement pass n
  EncodingPass()

EncodingPass()
  i = 0
  while i < I
    if S_n(i/4) = 1
      if S_{n+1}(i) = 0
        output S_n(i)
        if S_n(i) = 1
          output sign bit of node i
      else
        output n-th significant bit
      if S_n(L(i/4)) = 1
        if S_{n+1}(D(i)) = 0
          output S_n(D(i))
        if S_n(D(i)) = 1 and S_{n+1}(L(i)) = 0
          output S_n(L(i))
      i = i + 1
    else
      i = i + Skip(i)
```

그림 5 MZC-BFS의 인코딩 알고리즘

점진적인 이미지 전송에 있어 웨이블릿 변환된 이미지의 왜곡은 픽셀의 수 N에 대하여 $|c_i|^2/N$의 비율로 감소한다[2]. 즉, 계수의 MSB에 대한 오류는 큰 이미지 왜곡을 유발하지만 LSB의 오류로 인한 이미지 왜곡은 무시될 수 있다. SPIHT의 경우 비트율이 낮을 경우 계수의 하위 비트들에 대한 정보는 생성되지 않으며 복원과정에서 이들 값은 0의 값을 가진다. MZC-BFS는 복원과정에 필요한 공간트리의 재구성 상태를 유지하기 위하여 추가 메모리를 사용하지 않고 일부 계수의 LSB를 이용한다. 일부 계수에 대한 LSB의 이용은 비트율이 높아질 때 LSB의 값을 복구함에 따라 공간트리에 대한 상태정보를 잃게 되기 때문에 정확한 이미지를 복원할 수 없다. 따라서 본 논문에서는 높은 비트율에서 완전한 이미지 복원을 위하여 $D(i)$와 $L(i)$에 대해 각각 식 (2) 및 식 (3)과 같은 주소변환 기법을 제안하며, 복원 과정에서 가장 낮은 임계치 n=0일 때 계수의 LSB를 0으로 설정함으로써 LSB에 대한 오류를 점진적으로 복원시킨다. 수식에서 i_{level}은 그림 1에서 노드 i가 속한 부대역 레벨을 나타낸다. 그림 6은 $D(i)$에 대한 상태정보 배치를 위한 주소변환의 예를 보여주고 있다.

$$d(i) = (4i+3)\times 4^{i_{level}-1} \tag{2}$$

$$l(i) = (4i+2)\times 4^{i_{level}-1} \tag{3}$$

MZC-BFS의 디코딩 알고리즘은 그림 7과 같으며, $B_0(i)$는 계수 i의 LSB에 대한 비트연산을 의미한다.

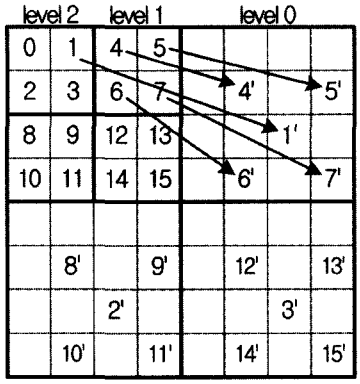


그림 6 $D(i)$의 상태정보 배치

```
for each refinement pass n
   DecodingPass()

DecodingPass()
   i = 0
   while i < I
      if n = 0   B0(i) = 0
      if B0(d(i/4)) = 1
         if Sn+1(i) = 0
            input one bit
            if bit = 1
               input sign bit and set coefficient
         else
            input one bit and set refinement bit
         if B0(l(i/4)) = 1
            if B0(d(i)) = 0
               input one bit and B0(d(i)) = bit
            if B0(d(i)) = 1 and B0(l(i)) = 0
               input one bit and set B0(l(i)) = bit
         i = i + 1
      else
         i = i + Skip(i)
```

그림 7 MZC-BFS의 디코딩 알고리즘

4. 실험 결과

MZC-BFS 알고리즘의 성능을 평가하기 위하여 메모리 사용량, 이미지 품질 및 압축과 복원에 필요한 수행시간에 대하여 실험하였다. 실험에 사용된 이미지는 512(512 크기의 Lena와 Barbara를 이용하였다. 실험에 사용된 웨이블릿 이미지는 Geoff Davis[8]의 라이브러리에서 Antonini[9] 9-7 탭 필터를 이용하였으며, 이미지 품질을 향상시키기 위한 추가적인 기법은 사용하지 않았다.

표 1은 MZC-BFS, SPIHT, NLS, SMAWZ에 대하여 Barbara 이미지의 압축과정에서 요구되는 추가 메모리 요구량을 비교한 결과이다. SPIHT의 경우 리스트 한 개에 필요한 메모리 크기는 6바이트로 가정하였으며,

표 1 Barbara의 비트율에 따른 추가 메모리 요구량 (byte)

bitrate	MZC-BFS	SPIHT	NLS	SMAWZ
0.1	0 K	61 K	128 K	42 K
0.5	0 K	270 K	128 K	42 K
1.0	0 K	618 K	128 K	42 K
1.5	0 K	882 K	128 K	42 K
2.0	0 K	1222 K	128 K	42 K

표 2 이미지 크기에 따른 추가 메모리 요구량

Image size	MZC-BFS	NLS	SMAWZ
256(256	0 K	32 K	10.5 K
512(512	0 K	128 K	42 K
1024(1024	0 K	512 K	168 K
2048(2048	0 K	2048 K	672 K

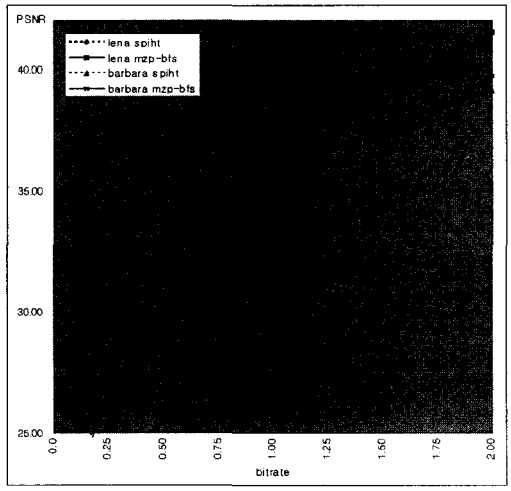


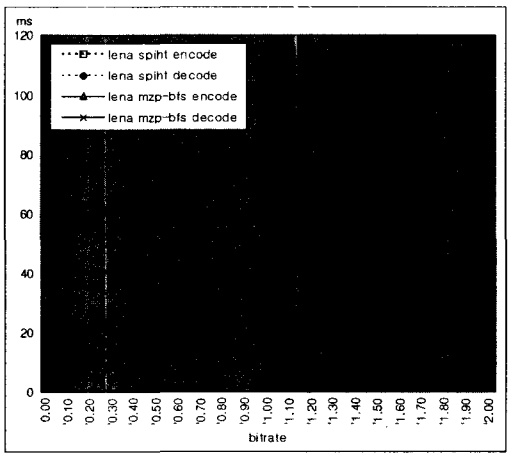
그림 8 비트율에 따른 PSNR값 비교

LIS, LIP 및 LSP에 대한 각각의 최대 메모리 요구량을 합하였다. 표 2는 이미지 크기에 따른 MZC-BFS, NSL 및 SMAWZ에 대한 추가 메모리 요구량을 비교한 결과이다.

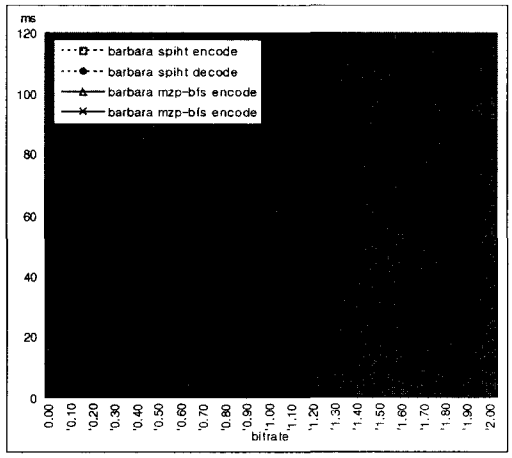
그림 8은 MZC-BFS와 SPIHT에 대하여 비트율에 의한 이미지의 품질을 측정한 결과이며, 공간트리에 대한 루트 노드의 스캔순서는 morton 방식을 사용하였다. 그림 8에서 MZC-BFS 알고리즘은 SPIHT 알고리즘과 거의 동일한 이미지 품질을 보여주고 있다. MZC-BFS와 SPIHT는 구간에 따라 이미지 품질의 차이가 역전되는 현상을 보이고 있는데, 이러한 현상은 SPIHT와 MZC-BFS가 생성하는 정보의 순서에 차이를 가지기 때문이다. SPIHT의 경우 압축된 비트스트림은 주부호화 과정에서 생성된 정보와 종속부호화 과정에서 생성된 정보가 구간적으로 반복되는 패턴을 가지는데 비하

여, MZC-BFS 알고리즘은 픽셀들의 스캔 순서에 따라 주부호화 과정의 정보와 종속부호화 과정의 정보가 혼합되어 전송되게 된다. 따라서 주어진 비트율에 대한 압축과정의 중단 시점이 주부화 과정에서 중단되는 경우와 종속부호화 과정에서 중단되는 경우에 따라 이미지 품질의 역전현상이 나타난다.

그림 9는 SPIHT와 MZC-BFS 알고리즘의 수행속도를 측정한 결과이다. 수행속도의 측정에서 SPIHT 알고리즘은 리스트에 대한 메모리 할당과 해제에 대한 영향을 최소화 하기 위하여 충분한 크기의 배열을 할당하여

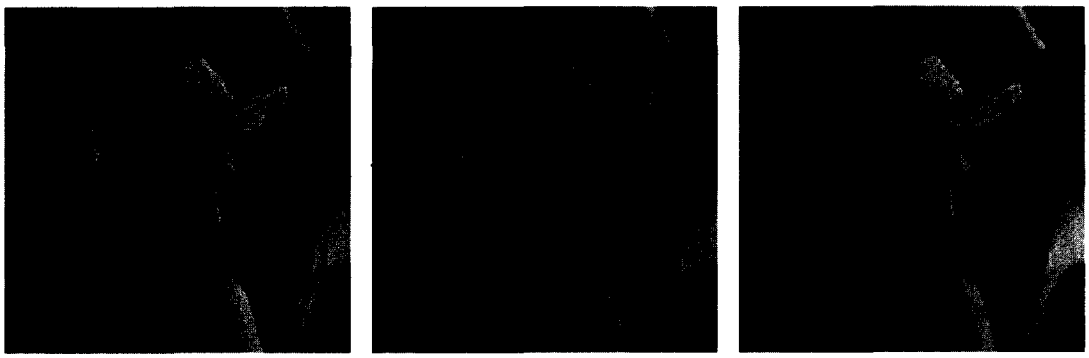


(a) Lena



(b) Barbara

그림 9 Lena와 Barbara의 비트율에 따른 수행속도



(a) Original Lena (b) SPIHT(31.48dB) (c) MZC-BFS(31.48dB)

(a) Original Barbara (b) SPIHT(25.66dB) (c) MZC-BFS(25.65dB)

그림 10 0.2bpp에서 Lena 및 Barbara에 대한 복원 이미지

리스트의 기능을 수행하도록 하였으며, MZC-BFS에서는 peano 코드에 대한 연산에 적합하도록 웨이블릿 이미지를 재배치한 후 알고리즘을 수행하였다. 실험결과에서 SPIHT의 경우 복원과정보다 압축과정의 수행시간이 높게 나타난 것은 $D(T)$와 $L(T)$에 대한 중요계수테스트를 위한 메모리 접근에 원인이 있으며, MZC-BFS의 경우 복원과정의 수행시간이 높게 나온 것은 $d(i)$와 $l(i)$ 계산에 필요한 픽셀의 피라미드 레벨을 구하는데 있어 반복문을 수행하기 때문이다. 임의의 픽셀에 대한 레벨은 주소에 대한 2진 비트열에서 최초로 1이 나타나는 자리수에 의존적이기 때문에 조합 논리회로로 구현이 가능하다. 따라서, MZC-BFS에서 픽셀의 피라미드 레벨에 대한 계산을 하드웨어로 구현할 경우 압축과 복원에 있어 수행시간은 거의 동일하다.

그림 10은 압축률 0.2bpp에서 Lena와 Barbara에 대한 복원 이미지를 보여주고 있으며, SPIHT와 MZC-BFS에 의해 복원된 이미지는 시각적으로 거의 동일한 이미지 품질을 보여주고 있다.

5. 결 론

SPIHT는 알고리즘이 간단하고 속도가 빠른 제로트리 기반 이미지 압축 알고리즘으로 잘 알려져 있다. 그러나 SPIHT는 부호화 과정에서 리스트 구조를 사용하기 때문에 추가의 메모리를 요구한다. SPIHT에서 추가 메모리의 요구량은 이미지의 형태와 비트율에 의존적이기 때문에 하드웨어 구현에 있어서 문제점으로 지적된다.

본 논문에서는 이전상태 중요계수 테스트 및 복원과정에서 계수의 LSB를 이용하여 SPIHT에서 리스트를 제거한 기법을 제안하였다. 제안된 기법은 리스트를 제거할 뿐만 아니라 계수값 만을 참조하므로 추가 메모리를 전혀 요구하지 않는다. MZC-BFS는 SPIHT의 계수 집합 분할 규칙을 따르고 있어 복원 이미지의 품질에 있어서 SPIHT와 거의 동일한 성능을 가진다. MZC-BFS는 알고리즘의 수행에는 있어 추가 메모리를 전혀 요구하지 않기 때문에 하드웨어 구현에 있어 비용을 절감할 수 있으며, 리스트를 사용하지 않기 때문에 메모리 접근 횟수가 급격히 줄어 수행속도가 빠르다. 또한 리스트 등에 대한 처리기능 및 추가의 메모리 등에 대한 부담이 없기 때문에 경량화가 요구되는 모바일 환경 등에 적합할 것으로 기대된다.

참 고 논 문

[1] Jerome M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Trans. on Signal Processing, vol.41, no.12, pp: 3445-3462, December 1993.

[2] Amir Said, William A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Trans. CSVT, vol.6, no.3, pp

[3] Frederick W. Wheeler, William A. Pearlman, "Low-Memory Packetized SPIHT Image Compression," Thirty-Third Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, Oct. 24-27, 1999.

[4] Wen-Kuo Lin, Neil Burgess, "Listless Zerotree Coding for Color Images," in 32nd Asilomar Conference on Signals, System and Computers, CA, USA, November 1998.

[5] R. Kutil, "A significance map based adaptive wavelet zerotree codec (SMAWZ)," In S. Panchanathan, V. Bove, and S.I. Sudharsanan, editors, Media Processors 2002, volume 4674 of SPIE Proceedings, pages 61-71, January 2002.

[6] Frederick W. Wheeler, William A. Pearlman, "SPIHT Image compression Without Lists," IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2000), Istanbul, Turkey, June 5-9, 2000.

[7] G. Seetharaman, B. Zavidovique, "Image Processing in a Tree of Peano-coded Images," In IEEE Workshop on Computer Architecture for Machine Perception. (Editor) Charles C. Weems Jr., pp.229-235, Octover 1997.

[8] http://www.geoffdavis.net/

[9] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," IEEE Trans. Image Processing, vol.1, pp.205-220, April 1992.

김 충 길

1994년 부산대학교 전자계산학과 졸업(학사). 1996년 부산대학교 전자계산학과 졸업(석사). 1998년~2000년 창신대학 전임강사. 2000년~2002년 성심외국어대학 전임강사. 2004년 현재 부산대학교 멀티미디어협동과정 박사수료. 2004년 현재 (주)프리디지텔 대표. 관심분야는 이미지 압축, VoIP

정 기 동

1973년 서울대학교 졸업(학사). 1975년 서울대학교 대학원 졸업(석사). 1986년 서울대학교 대학원 계산통계학과 졸업(이학박사). 1990년~1991년 MIT, South Carolina 대학 교환 교수. 1995년~1997년 부산대학교 전자계산소 소장. 1978년~현재 부산대학교 전자계산학과 교수 1997년~현재 부산대학교 대학원 멀티미디어 협동과정 교수. 관심분야는 병렬처리, 멀티미디어