SPIHT 기반 문서 부호화와 복호화 시스템의 성능 향상

(Improvement of SPIHT-based Document Encoding and Decoding System)

장 준 † 이 호 석 ††
(Joon Jang) (Hosuk Lee)

요 약 본 논문은 문자가 포함된 영상에서 문자의 가독성은 유지하면서 영상을 압축하고 복원하는 효율적인 방법을 제시한다. 본 논문에서 제시하는 방법은 문서 분할, Quincunx 다운샘플링, (5/3) 웨이블릿 리프팅 그리고 서브밴드별 SPIHT(Set Partitioning In Hierarchical Trees) 부호화 방법을 기반으로 하여 구축되었다. 부호화 과정에서는 *Quincunx* 다운샘플링과 서브밴드 SPIHT 부호화 방법을 사용하여 부호화 수행 시간을 단축하였으며 산술 부호화를 적용하여 SPIHT 부호기의 비트스트림을 더욱 압축하였다. 실험에서는, 복원된 영상을 제시하여 시스템의 성능을 확인할 수 있도록 하였으며, 여러 가지 양자화를 적용하여 제안한 SPIHT에 기반한 문서 압축 시스템의 압축율과 PSNR을 비교하고 분석하였다.

키워드 : 문서 부호화와 복호화, 웨이블릿 리프팅, (5/3) 변환, SPIHT, 양자화, 다운샘플링과 업샘플링

Abstract In this paper, we present a document image compression system based on segmentation, Quincunx downsampling, (5/3) wavelet lifting and subband-oriented SPIHT coding. We reduced the coding time by the adaptation of subband oriented SPIHT coding and Quincunx downsampling. And to increase compression rate further, we applied arithmetic coding to the bitstream of SPIHT coding output. Finally, we present the reconstructed images for visual comparison and also present the compression rates and PSNR values under various scalar quantization methods.

Key words : document encoding & decoding, wavelet lifting, (5/3) transform, quantization, SPIHT, downsampling & upsampling

1. 서 론

일반적으로 컬러 영상 압축의 표준인 JPEG(Joint Phothographic Experts Group)과 같은 압축 방법은 문자가 포함된 영상에서는 적절하지 못하다. JPEG은 영상에서의 고주파 성분은 제거하여도 영상의 질(quality)은 크게 저하되지 않는다는 가정에 기반을 두고 DCT (Discrete Cosine Transform)을 사용한다. 자연적인 대부분 영상은 이러한 가정을 만족시키지만, 문자가 포함된 영상은 이러한 가정을 만족시키지 못한다. 그러므로 문자의 가독성을 유지하기 위해서 다른 효율적인 부호화 방법이 필요하다.

† 비 회 원 : (주)지오매니아
cjun73@dreamwiz.com
†† 정 회 원 : 호서대학교 컴퓨터공학부 교수
hslee@office.hoseo.ac.kr
논문접수 : 2002년 1월 7일
심사완료 : 2003년 4월 1일

이 논문에서 소개하는 기본 방법은 문자가 포함된 영상을 세 개의 영역으로 구분하는 것이다. 즉, 원래의 영상을 배경 영역(background), 문자 영역(foreground), 그리고 비트맵(bitmap)으로 구분하는 것이다. 배경 영역과 문자 영역은 저해상도(low-resolution)의 칼라 영상이고, 비트맵은 고해상도(high-resolution)의 2-레벨 영상이다. 이렇게 분리된 배경 영역과 문자 영역에 대하여서는 문자의 가독성을 손상시키지 않는 범위 내에서 적절한 양자화 방법을 적용하여 영상을 부호화하고 복호화한다. 문서 분할 모듈은 multiscale bicolor clustering 알고리즘과 배경영역/문자영역 반전 알고리즘을 사용하여 구현하였다[1].

SPIHT(Set Partitioning In Hierarchical Trees) 부호화 알고리즘[2]은 EZW(Embedded Zerotree Wavelet) 부호화 알고리즘[3]을 개선한 방법이지만 영상의 크기가 커지면 부호화 하는데 시간이 너무 많이 소요된다

는 단점을 가지고 있다. 이러한 점을 개선하기 위하여 SPIHT 부호화 방법을 한번에 전체 웨이블릿 계수에 대하여 적용하는 것이 아니라 서브밴드별로 나누어 적용하는 서브밴드 SPIHT 부호화 방법을 제안한다.

분할된 배경 영역에 대하여서는 Quincunx 다운샘플링[4,5]을 수행하여, 배경 영역을 두 개의 영상으로 나눈다. 이후 나누어진 2개의 배경 영역과 문자 영역에 대하여 웨이블릿 변환을 수행한다. 웨이블릿 변환[4,5]은 웨이블릿 리프팅[6,7]을 사용한다. 웨이블릿 리프팅은 기존의 웨이블릿 변환에 비하여 여러 가지 장점을 가지고 있다. 그리고 웨이블릿 계수를 부호화하기 위하여 서브밴드 SPIHT 부호화 알고리즘[1]을 사용한다. 서브밴드 SPIHT 부호화 알고리즘은, 기존의 SPIHT 알고리즘을 개선한 것으로 웨이블릿 계수를 대역별로 분할하여 각각의 서브밴드에 대하여 SPIHT 부호화를 적용하는 방법이다. 마지막으로 압축율을 더욱 높이기 위하여 산술 부호화[8]를 적용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 웨이블릿 리프팅에 대하여 소개한다. 3장에서는 SPIHT 부호화 개선 방법에 대하여 소개한다. 4장에서는 문서 분할, Quincunx 다운샘플링, 부호기, 그리고 복호기에 대하여 설명한다. 5장의 실험에서는 복원된 영상을 제시하고, 여러 가지 양자화 적용에 의한 압축율과 PSNR 값을 제시한다. 마지막으로 6장에서는 결론과 향후 연구방향에 대하여 소개한다.

2. 웨이블릿 리프팅

웨이블릿 리프팅 방법은 웨이블릿 변환을 구현하기 위한 다른 방법이다. 웨이블릿 리프팅 방법을 간략하게 소개하면 다음과 같다. 원래의 신호를 $\lambda_{0,k}=f(k)$ for $k\in\mathbb{Z}$ 라고 하자. 이 신호로부터 가능하다면 적은 양의 계수를 얻기를 원하고 허용 가능한 범위 내에서 근사값(approximation)을 구하기를 원한다고 하자. 즉, 적은 계수들을 사용하고, 손실되는 정보도 정확히 제어하고, 실제 신호와 근사화된 신호 사이의 오차도 작게 만들기로 한다. 그러면, 간단히 짝수 신호들만을 서브샘플링(subsampling)하여 계수를 구할 수가 있다. 이것을 다음과 같이 수식 (1)로 표현할 수 있다[6,7].

$$\lambda_{-1,k} = \lambda_{0,2k} \quad \text{for } k \in \mathbb{Z} \tag{1}$$

그러나 이 과정에서 우리는 얼마나 많은 정보가 손실되었는지 구할 필요가 있다. 바꾸어 말하면 $\{\lambda_{-1,k}\}$에서 원래의 신호 $\{\lambda_{0,k}\}$를 복구하는데 필요한 정보를 알아야 한다. 우리는 이 차이를 부호화하기 위해 $\{\gamma_{-1,k}\}$ 계수를 사용하고, 이 계수들은 웨이블릿 계수로 부르기로 한다. 이 차이는 간단하게, $\gamma_{-1,k} = \lambda_{0,2k+1}$ for $k\in\mathbb{Z}$로 표현할 수 있다. 그러나, 앞에서 언급한 방법으로는 만족 할만한 효과를 얻을 수 없다. 왜냐하면, 홀수 신호들의 값이 적은 경우는 웨이블릿 계수는 적을 수도 있다. 어떠한 경우는 거의 없을 수도 있다. 그러므로, 서브샘플된 계수 $\{\lambda_{-1,k}\}$로부터 원래의 신호 $\{\lambda_{0,k}\}$를 복원하기 위해서는 보다 정교한 방법이 필요하다. 주변의 샘플 계수들이 연관성을 가지고 있다고 가정하면, 홀수 계수 $\lambda_{0,2k+1}$를 주변의 짝수 계수 $\lambda_{-1,k}$ 와 $\lambda_{-1,k+1}$의 평균값으로 예측할 수 있다. 즉, 웨이블릿 계수를 홀수 계수값 $\lambda_{-1,k}$과 이것의 예측값 사이의 차이(difference)로 생각할 수 있다.

$$\gamma_{-1,k} := \lambda_{0,2k+1} - \frac{1}{2}(\lambda_{-1,k} + \lambda_{-1,k+1}) \tag{2}$$

만일 신호가 많은 연관성(correlation)을 가지고 있다면, 대부분의 웨이블릿 계수들은 작은 값을 가질 것이다. 여기서, 웨이블릿 계수는 $\{\lambda_{-1,k}\}$로부터 원래의 신호 $\{\lambda_{0,k}\}$를 복원하기 위하여 필요한 세부값(detail)으로 생각할 수 있다. 주파수의 관점에서 보면, 웨이블릿 계수는 원래 신호의 고주파 부분을 나타내고, $\{\lambda_{-1,k}\}$은 저주파 부분을 나타낸다. 이러한 방법을 반복적을 수행하여 웨이블릿 변환을 구현할 수 있다. 즉, 다음 단계에서는 $\{\lambda_{-1,k}\}$에 대해서, 더 낮은 단계의 계수 $\{\lambda_{-2,k}\}$와 서브 샘플링을 통해서 $\{\gamma_{-2,k}\}$를 구한다. 각각의 단계의 $\lambda_{j,k}$ 계수들의 평균값은 동일하여야 하므로 $\sum_k \lambda_{-1,k} = \frac{1}{2}\sum_k \lambda_{0,k}$을 만족해야 한다. 따라서 웨이블릿 계수 $\gamma_{-1,k}$ 을 가지고 $\lambda_{-1,k}$을 보정할 필요성이 있다. 이때 이웃하는 웨이블릿 계수들을 사용한다.

$$\lambda_{-1,k} \mathrel{+}= 1/4(\gamma_{-1,k-1} + \gamma_{-1,k}) \tag{3}$$

다시 말하면, 웨이블릿 리프팅은 두 단계로 구성된다. 처음은 웨이블릿 계수를 계산하는 단계, 두 번째는 웨이블릿 계수를 가지고 서브샘플된 계수들을 보정하는 단계이다.

웨이블릿 리프팅에서 계수의 계산은 메모리의 추가 사용이 없이 원래의 데이터 장소에서 직접 계산할 수 있다. 본 논문에서는 (5/3) 웨이블릿 리프팅 방법을 사용하였다[9]. (5/3) 웨이블릿 리프팅 변환식은 수식 (4)와 같다. 여기서 $s[n]$은 lowpass 서브밴드 신호, $d[n]$은 highpass 서브밴드 신호, $s_0[n]$은 $x[2n]$, 그리고 $d_0[n]$은 $x[2n+1]$을 나타낸다. $x[n]$은 입력 신호이다.

순 방향 변환

$$d[n]=d_0[n]-\left\lfloor \frac{1}{2}(s_0[n+1]+s_0[n]) \right\rfloor \quad (4)$$
$$s[n]=s_0[n]+\left\lfloor \frac{1}{4}(d[n]+d[n-1])+\frac{1}{2} \right\rfloor$$

역 방향 변환

$$s[n]=s_0[n]-\left\lfloor \frac{1}{4}(d[n]+d[n-1])+\frac{1}{2} \right\rfloor$$
$$d[n]=d_0[n]+\left\lfloor \frac{1}{2}(s_0[n+1]+s_0[n]) \right\rfloor$$

3. SPIHT 부호화 개선

3.1 SPIHT 부호화 알고리즘 개선

SPIHT 부호화 알고리즘[2]은 EZW 부호화 알고리즘[3]을 개선한 알고리즘이다.

SPIHT 알고리즘에서 부호화 과정의 중요도 지도(significant map)는 각 threshold에 대하여 LSP(List of Significant Pixels), LIP(List of Insignificant Pixels), LIS(List of Significant Sets)의 리스트를 통하여 구해진다. 또한 전체 알고리즘은 초기화, 정렬단계, refinement pass, 양자화 값 갱신의 네 가지 단계로 구성된다.

그러나 SPIHT 부호화 알고리즘은 수행시간이 많이 소요되는 단점이 있다. 즉, 영상의 크기가 커지면 구성되는 zerotree의 깊이가 증가하고 검색해야 할 노드(node)의 수가 증가하게 되어 수행시간이 많이 소요된다. 다음 [그림 1]은 영상의 크기와 부호화 소요시간과의 관계를 그래프로 나타낸 것이다.

[그림 1]에서 알 수 있듯이, Lena 영상의 경우에 64×64 크기의 영상까지는 1초 이내에 수행되며 128×128 크기의 영상까지는 비교적 수행시간이 적게 소요된다. 그러나 영상의 크기가 256×256으로 커지면 수행시간이 7분20초(440초)가 소요되는 것을 알 수 있다. 이것은 LIS 리스트의 중요도 검사 때 하나의 노드의 자손 중에 중요한 정보를 가지는 자손이 존재하는지 검사하는 부분에서 수행시간이 많이 소요되기 때문이다.

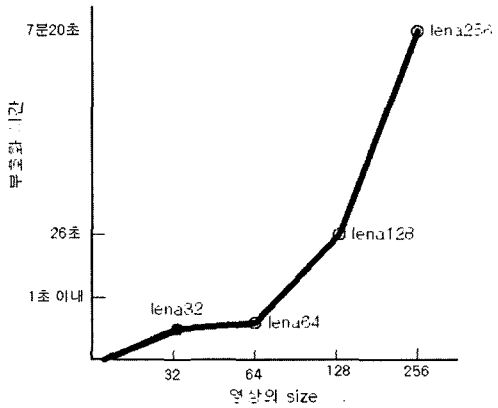


그림 1 원래 SPIHT 방법의 부호화 소요시간

위의 실험 결과와 같이 크기가 작은 영상에 대해서는 SPIHT 부호화가 우수한 효과를 보이지만 크기가 큰 영상에 대해서는 부호기로서의 역할을 하기에는 수행 시간이 너무 많이 소요되는 것을 알 수 있다. 이 문제를 해결하기 위해 원 영상을 웨이블릿 변환한 다음, 계수들을 서브밴드로 분할하고 각각의 분할된 서브밴드에 대하여 SPIHT 부호화를 적용하는 방법을 고안하였다. 여기서 또한 웨이블릿 변환의 특성을 살리기 위하여 n-레벨로 분해된 웨이블릿 영상에 대하여서는 n개의 서브밴드로 나누어 적용하였다. [그림 2]는 웨이블릿 영상을 n개의 서브밴드로 나누고, 각 서브밴드에 대하여 SPIHT 방법을 적용하는 방법을 나타낸 것이다. 즉, 원래의 SPIHT 부호화 방법을 각 서브밴드에 대하여 각각 적용하는 것이다. EZW, SPIHT, 그리고 개선한 SPIHT에 대한 시간복잡도는 3.2절에서 제시하기로 한다.

그러나 위에서 제안한 방법으로 부호화하기 위해서는 부수적인 작업이 필요하다. 즉, 개선된 서브밴드 SPIHT 부호화 방법에서는 영상 분해 레벨, 각 서브밴드에 대한 threshold 값, 그리고 정렬단계와 refinement 단계에서 생성하는 비트스트림 량에 대한 정보를 복호기에 전달하여야 한다.

다음은 개선된 서브밴드 SPIHT 부호화 방법을 구현하기 위하여 필요한 자료구조이다. 아래는 생성되는 비트스트림 양을 저장하기 위한 세부 자료구조이다.

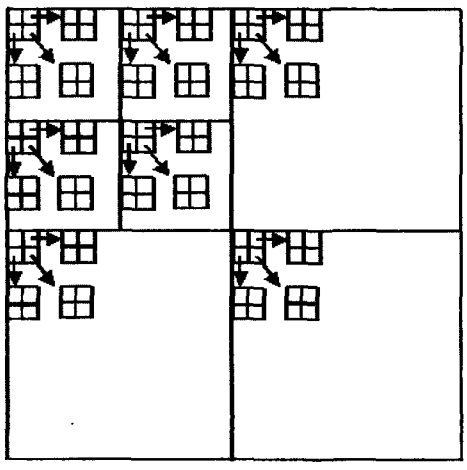


그림 2 서브밴드 SPIHT 부호화

```
• typedef bitstreamInfo {
  long LIPBit;          // LIP 리스트 검색에 대한 생성 비트량
  long LISBit;          // LIS 리스트 검색에 대한 생성 비트량
  long RefinementBit; // LSP 리스트 검색에 대한 생성 비트량
  } bitstreamInfo;
```

다음은 각각의 서브밴드에 대한 좌표를 저장하기 위한 세부 자료구조이다.

```
• typedef struct subImageInfo {
    int basePoint_X;  // 서브밴드의 시작 X좌표
    int basePoint_Y;  // 서브밴드의 시작 Y좌표
    Int RowBoundary;// 서브밴드의 끝 X좌표
    int ColBoundary;  // 서브밴드의 끝 Y좌표
  } subImageInfo;
```

다음은 각 서브밴드에 대한 threshold 값과 각 서브밴드에 대한 비트스트림 출력량을 저장하는 자료구조이다.

```
• typedef struct passInfo {
    short int threshold;   // 서브밴드에 대한 threshold 값
    bitstreamInfo* bitInfo;// 각 threshold에 대한 비트스
                           트림 출력 정보
  } passInfo;
```

다음 [그림 3]은 개선된 서브밴드 SPIHT 부호화 방법에서 영상의 크기과 수행 소요 시간과의 관계를 나타낸 그래프이다. [그림 1]과 [그림 3]을 비교하여 보자. [그림 3]에서, 32 × 32, 64 × 64, 128 × 128 크기의 Lena 영상의 경우에 부호화에 소요되는 시간이 1초 이내인 것을 알 수 있다. [그림 1]에서는 128 × 128 크기의 Lena 영상의 경우에 26 초가 소요되었다. 더욱이, [그림 3]을 보면 256 × 256 크기의 Lena 영상의 부호화에 79 초 이내가 소요되는 것을 알 수 있다. [그림 1]을 보면,

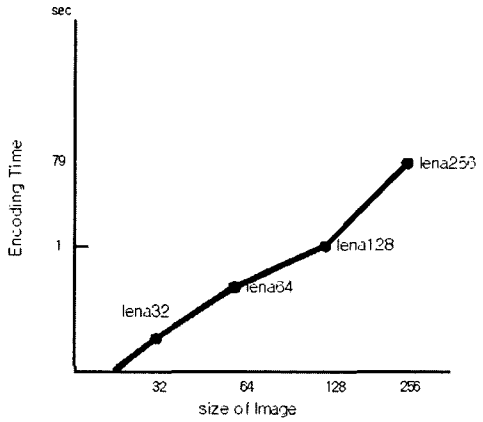


그림 3 개선된 SPIHT 방법의 부호화 소요 시간

256 × 256 크기의 Lena 영상의 경우에 440 초가 소요되는 것을 알 수 있다. 이 그래프들은 개선된 서브밴드 SPIHT 부호화 방법의 뚜렷한 성능 향상을 보여준다. 512 × 512 크기의 Lena 영상을 예로 들어, 원래의 SPIHT 부호화 방법에서는 LIS 리스트에서 검색하여야 할 노드의 개수가 87,380개 였다. 그러나 개선된 SPIHT 부호화 방법에서는 3-레벨로 분해하였을 경우에, LIS 리스트에서 검색하여야 할 노드의 개수가 2,728개로 줄어 들었다. 거의 84,652개의 노드가 줄어들었으며, 이는 SPIHT 부호화 소요 시간을 단축시켜, 전체적으로 큰 성능향상을 가지고 오는 것을 알 수 있었다.

3.2 시간 복잡도 비교

다음은 EZW, SPIHT, 개선된 SPIHT의 시간 복잡도이다. 먼저 EZW의 시간 복잡도 다음과 같다.

· EZW : $O(\log_2 M\,(N^2 + \text{no. of significant coefficients}))$

(단, M은 웨이블릿 계수 최대값, N은 계수의 전체크기)

EZW는 dominant pass와 subordinate pass 두 부분으로 구성되며 dominant pass를 수행한 다음에 subordinate pass를 수행한다. Dominant pass 부분에서는 최대 웨이블릿 계수값의 $\frac{1}{2}$ 값을 문턱값(threshold)으로 설정한 다음, 각 문턱값에 대하여 모든 계수를 조사하여 중요한(significant) 계수를 찾아내며 sbordinate pass 부분에서는 찾아낸 중요한 계수를 부호화한다. 그러므로 위와 같은 시간 복잡도를 구할 수 있다.

다음은 SPIHT의 시간 복잡도이다.

· SPIHT : $O(\log_2 M\,(\text{no. of elements in LIP} + \text{no. of elements in LIS}))$

(단, M은 웨이블릿 계수 최대값)

SPIHT는 sorting pass와 refinement pass로 구성되며 LIS, LIP, LSP 세 가지의 리스트 구조를 사용한다. LIS는 전체 계수의 루트로 초기화되며 궁극적으로 전체 계수를 나타낸다. LIP는 중요하지 않은(insignificant) 계수를 저장하고, LSP는 중요한(significant) 계수를 저장한다. SPIHT는 sorting pass 부분에서 LIS와 LIP 리스트를 조사하여 중요한 계수를 찾아낸 다음, 이 계수들을 부호화한다. 따라서, 알고리즘의 복잡도는 궁극적으로 LIS 와 LIP 리스트의 크기 - 각 리스트에 포함된 원소의 개수 - 에 의하여 결정된다.

개선한 SPIHT의 서브밴드 단위 시간 복잡도는 다음과 같다.

· 개선한 SPIHT :

$O(\log_2 M$ (no. of elements in $SLIP_d$ + no. of elements in $SLIS_d$))

(단, M은 웨이블릿 계수 최대값, d는 분해레벨)

원래의 SPIHT 알고리즘은 웨이블릿 계수의 개수가 많지 않은 경우에는 효율적이지만, 계수의 개수가 증가하면 상당히 비효율적이다. 따라서 각 서브밴드별로 SPIHT 알고리즘을 수행시키면, 적은 개수의 원소를 포함한 LIP와 LIS 리스트를 가지고 SPIHT를 수행시킬 수가 있으며, 따라서 전체 수행속도가 상당히 빠르게 된다. 식의 d는 웨이블릿 변환의 분해레벨(decomposition level)을 의미하며 $SLIP_d$와 $SLIS_d$는 레벨 d의 경우에 각 서브밴드의 LIP와 LIS 리스트를 의미한다. LIP와 LIS 리스트에 포함된 계수의 개수에는 대체적으로 다음의 관계가 있다. 즉, d=1일 때, 각 서브밴드 LIP($SLIP$)와 LIS($SLIS$) 계수의 개수는 원래의 LIP와 LIS 개수의 $\frac{1}{4}$이 된다.

· no. of elements of $SLIP_d \approx \frac{1}{4^d} \times$ (no. of elements of LIP_1)

· no. of elements of $SLIP_d \approx \frac{1}{4^d} \times$ (no. of elements of LIS_1)

즉, 각 분해레벨 d가 증가할수록 LIP와 LIS 리스트에 포함되는 원소의 개수는 레벨 1의 LIP와 LIS 리스트에 포함된 원소의 개수에 대하여 $\frac{1}{4^d}$로 줄어들게 된다. 따라서, SPIHT를 각 서브밴드별로 수행시키면 적은 개수의 원소를 포함한 LIP와 LIS 리스트를 가지고 SPIHT를 수행시킬 수가 있기 때문에, 부호화 전체의 효율은 상당히 좋아지게 되는 것이다. 이러한 결과는 3.1절의 실험에서도 입증하였다. 다음은 개선한 SPIHT 전체 시간 복잡도이다.

·개선한 SPIHT 전체 시간 복잡도 :

$\sum_{d=1}^{3\log_2 M+1} O(\log_2 M$ (no. of elements in $SLIP_d$ + no. of elements in $SLIS_d$))

(단, M은 웨이블릿 계수 최대값, d는 분해레벨)

4. 시스템 구조

4.1 문서 분할

입력된 문서는 배경 영역(background)과 문자 영역(foreground), 그리고 비트맵(bitmap)으로 분할된다[5]. 문서 분할은 K-평균 칼라 클러스터링 알고리즘을 기반으로 하여 멀티스케일 칼라 클러스터링 알고리즘과 문자 영역/배경 영역 반전 알고리즘으로 구성된다[1]. 다음 [그림 4]는 문서 분할의 결과이다.

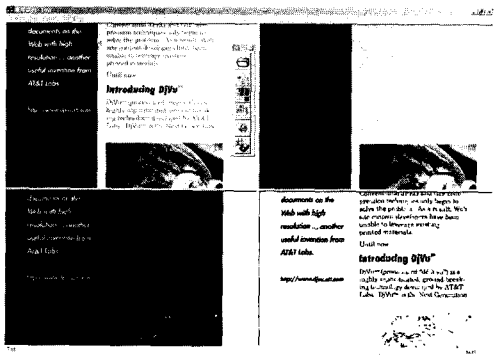


그림 4 문서 분할

4.2 다운샘플링과 업샘플링

분할된 문서의 배경 영역에 대해서는 Quincunx 다운샘플링[4,5]을 수행한다. 문자 영역에 대해서는 다운샘플링을 수행하지 않는다. 다음 [그림 5]는 Quincunx 다운샘플링 방법과 결과를 보여준다.

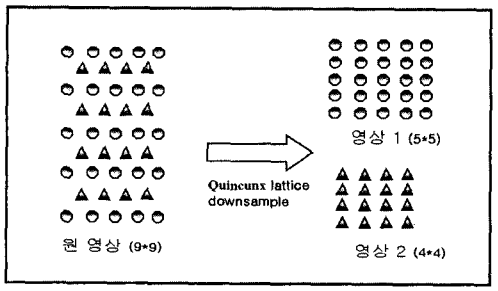


그림 5 Quincunx 다운샘플링

즉, 원래의 배경 영역을 두 개의 영상으로 다운샘플링한다. 이후에 복구할 때는, 각각 복구된 두 영상을 원래의 영상의 좌표에 맵핑하게 된다. 각각의 좌표에 맵핑된 영상의 비어있는 부분을 복원하기 위한 방법으로는 짝수 열에서는 우측의 화소를 복사하고, 홀수 열에서는 좌측의 화소를 복사한다. 이 방법은 매우 간단하나 효과적이다. 그러나 복원된 배경 영상은 물체에서 경계 부분에 계단 현상이 나타나는 경우가 있다. 이러한 문제점을 해결하기 위하여 메디안 필터링을 통해 물체의 경계 부분을 부드럽게 하였다.

4.3 부호기 구조

이 절에서는 부호기에 대하여 설명한다. 부호기는 문

서 분할 모듈, 웨이블릿 리프팅 모듈, 비트맵 영상을 부호화 하는 JBIG 모듈, 웨이블릿 변환된 문자 영역과 배경 영역을 부호화 하는 서브밴드 SPIHT 모듈, 산술 부호화 모듈로 구성된다. 문자가 포함된 영상이 입력되면, 문서 분할 모듈에서 문자 영역, 배경 영역, 비트맵으로 분리하게 된다. 비트맵 영상은 다음 단계에서 JBIG 부호기에 의해 부호화하고 비트스트림으로 출력되게 된다. 문자 영역은 RGB 칼라 모델에서 YCbCr 칼라 모델로 변환하고, 4:1:1 다운샘플링을 수행하게 된다. YCbCr로 변환된 영상은 웨이블릿 리프팅을 수행하여 웨이블릿 계수로 변환된다. 이러한 계수들에 대하여 스칼라 양자화를 수행한다. 스칼라 양자화는 일률적으로 적용하지 않고 각각의 밴드에 적절한 양자화 값을 독립적으로 적용하게 된다. 즉 최상위 밴드의 영상에 대해서는 작은 값을 양자화 값으로 적용하고, 최하위 밴드에 대해서는 좀더 큰 값을 양자화 값으로 적용한다. 양자화된 계수들은 서브밴드 SPIHT 부호화 모듈에 입력하여 부호화를 수행한다. SPIHT 부호화에 의하여 생성된 비트스트림을 다시 더욱 압축하기 위해 산술 부호화를 수행하게 된다. 배경 영역은 문자 영역과 동일한 수행과정을 거치게 되지만, Quincunx 다운샘플링 과정을 수행하여 영상을 둘로 나누고 두 개의 영역에 대해 동일한 과정을 수행하게 된다. 다음 [그림 6]은 부호기의 전체 구조를 간략하게 나타낸 것이다.

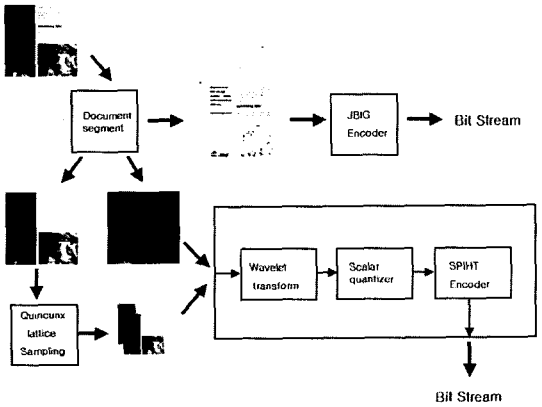


그림 6 부호기 구조

4.4 복호기 구조

복호기는 부호기의 역과정을 거치게 된다. 문서 복원 모듈에 입력되는 비트스트림은 비트맵, 문자 영역 그리고 두 개의 배경 영역이다. 두 개의 배경 영역을 각각 back1과 back2로 부르기로 한다. 두 개의 배경 영역 back1과 back2로부터 하나의 배경 영역을 복구한다. 배경 영역이 복구되었으면, 다음에 비트맵 정보를 참조하여 임의의 좌표가 '1'의 값을 가지고 있다면 문자 영역 값을 선택하여 복원될 영상에 맵핑하고, '0'의 값을 가지고 있으면 배경 영역의 값을 맵핑하는 작업을 수행한다. 다음 [그림 7]은 복호기 전체 구조를 간략하게 나타낸 것이다.

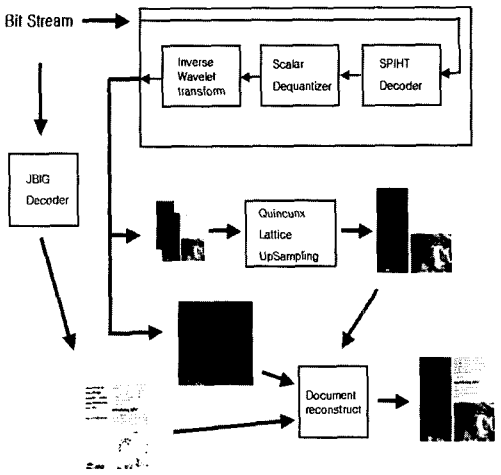


그림 7 복호기 구조

5. 시스템 실험

5.1 실험 환경

시스템의 실험 환경은 다음과 같다.

· CPU : AMD 1.2GHz . RAM : 512 MBytes

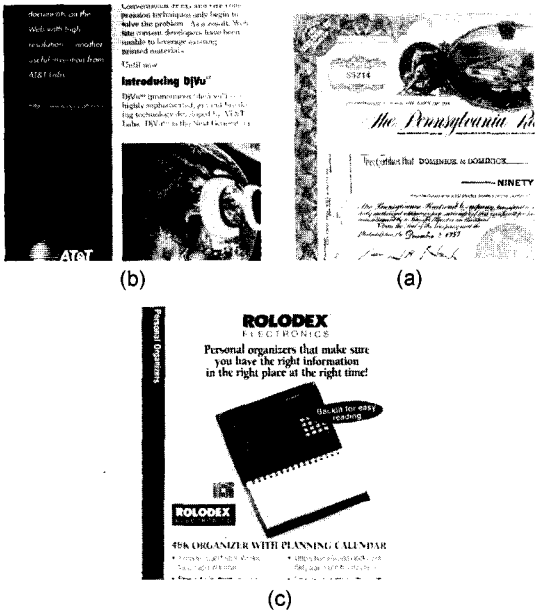
· OS : MS Windows 2000 Professional

실험 영상은 DjVu brochure, Pennsylvania railroad stock certificate, Franklin electronic publishers 영상들을 512 ×512 크기를 가지는 영상으로 편집하여 사용하였다. 영상의 분해를 위한 웨이블릿 리프팅은 (5/3) 리프팅 방법을 사용하였으며, 분해 레벨은 3으로 하였다. 실험 방법은 복구 영상과 원래 영상을 양자화(scalar quantization)에 따라 압축률과 PSNR(Peak Signal-to-Noise Ratio)의 값을 측정하여 비교하는 것으로 하였다. 다음은 PSNR을 구하기 위한 공식이다.

$$\mathbf{PSNR} = 10\log_{10}\left(\frac{(\text{peak pixel value})^2}{\mathbf{MSE}}\right)\mathbf{dB} \qquad (5)$$

$$\mathbf{MSE} = \frac{\sum [f(i,j)-F(i,j)]^2}{N^2}$$

여기서 MSE는 원 영상과 복원 영상간의 평균 제곱 오차를 나타낸다. 다음 [그림 8]의 (a),(b),(c)는 실험 영상들이다.



(a) DjVu brochure (512 × 512)
(b) Pennsylvania railroad stock certificate (512 × 512)
(c) Franklin electronic publishers (512 × 512)

그림 8 실험 영상

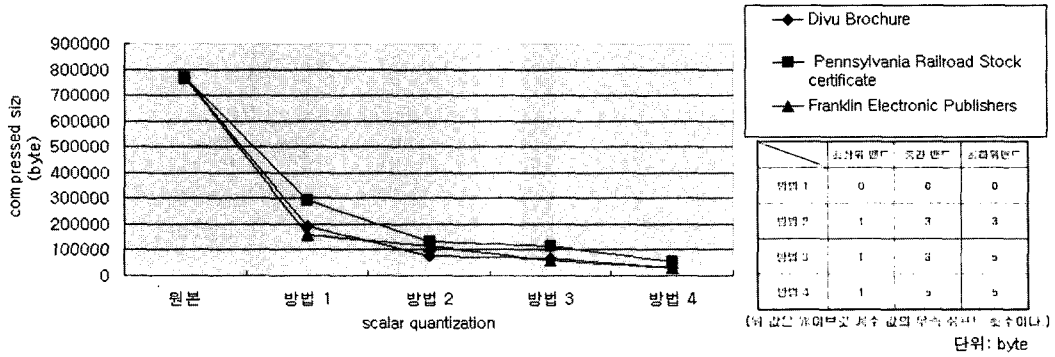
5.2 실험 결과

실험 영상인 DjVu brochure, Pennsylvania railroad stock certificate, Franklin electronic publishers 영상을 압축하여 얻어지는 압축률과 비트량은 [그림 9]와 같다. [그림 10]은 PSNR 값에 대한 그래프이다. 각각의 방법은 양지화를 각 대역별로 적용하는 수치를 조정하여 나타낸 것으로 방법 1은 양자화를 적용하지 않는 방법이고, 방법 2는 최상위 밴드의 계수 값을 1비트 우측으로 쉬프트(shift)하고, 나머지 밴드의 계수 값을 3비트 우측으로 쉬프트한 방법이다. 방법 3은 최상위 밴드와 중간 밴드는 방법 2와 같고, 최하위 밴드의 계수 값을 5비트 우측으로 쉬프트한 방법이다. 방법 4는 최상위 밴드의 계수 값을 1비트 우측 쉬프트하고, 나머지 밴드는 5비트 우측으로 쉬프트한 방법이다.

5.3 성능 분석

본 논문에서 구현한 SPIHT에 기반한 문서의 부호화와 복화화 시스템은 [그림 9]와 [그림 10]에서 보는 바와 같은 좋은 결과를 나타내었다. 본 연구의 주된 관점은 문자가 포함된 영상에서 문자의 가독성을 유지하면서 압출률을 높이는 부분과 기존의 SPIHT 부호화 알고리즘의 성능 향상에 있다고 하였다.

본 논문에서 구현한 SPIHT에 기반한 시스템은 부호화 소요시간의 관점에서는 [그림 3]을 그리고 압축율의 관점에서는 [그림 9]와 [그림 10]의 결과를 보여 주었다. 또 문자의 가독성 유지라는 관점에서 볼 때, 문자가 포함된 문서에 대해서 최대 약 3%-6%(33:1-16:1)까지 압축할 수 있었다. 결론적으로 PSNR을 25db 범위로 유지하면서 압축률은 높이고 부호화 소요시간 단축이라는 결과를 얻을 수 있었다. PSNR 수치가 조금 낮은 것은 배경 영역에 대한 Quincunx 다운샘플링때문이라고 생각하며, 시각적인 차원에서 복원 영상의 품질에 큰 차이가 없었다. 그리고 문자 영역에 대해서는 Quincunx 다운샘플링을 적용하지 않음으로써 문자의 가독성은 손상



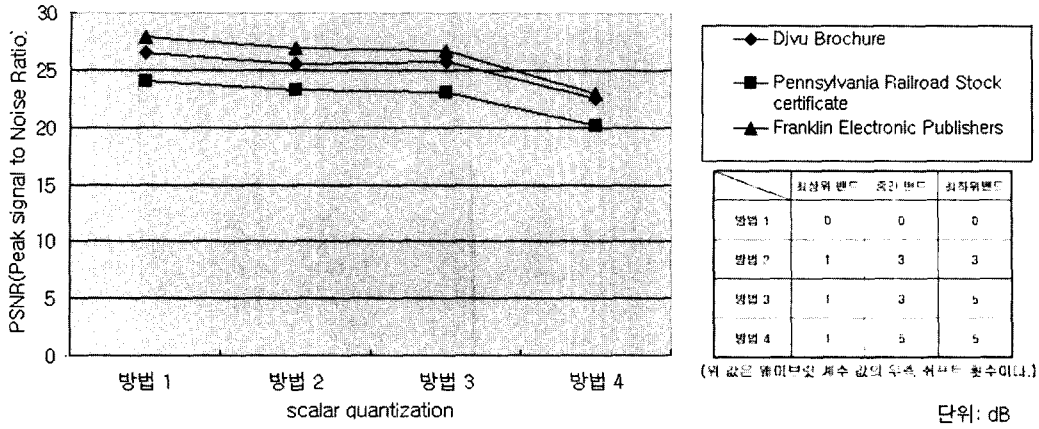
	최상위 밴드	중간 밴드	최하위밴드
방법 1	0	0	0
방법 2	1	3	3
방법 3	1	3	5
방법 4	1	5	5

(위 값은 대역별로 적용 값의 우측 쉬프트 수 입니다.)

단위: byte

	Djvu Brochure	압축률	Pennsylvania Railroad Stock certificate	압축률	Franklin Electronic Publishers	압축률
원본	768488	100%	768488	100%	768488	100%
방법 1	191477	23.7%	292595	35.9%	155547	19.2%
방법 2	77397	9.22%	133001	15.6%	116210	8.0%
방법 3	68296	8.00%	115755	13.4%	59517	7.0%
방법 4	31439	3.30%	55173	5.7%	28567	3.1%

그림 9 양자화 방법에 대한 압축율



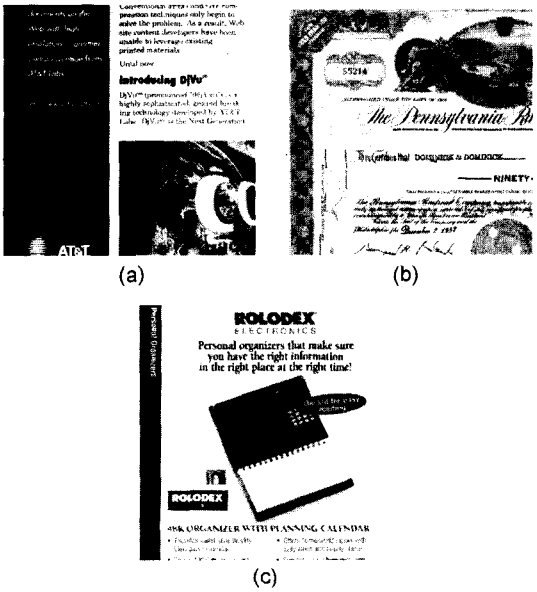
	최상위 밴드	중간 밴드	최하위밴드
방법 1	0	0	0
방법 2	1	3	3
방법 3	1	3	5
방법 4	1	5	5

(위 값은 웨이브릿 계수 값의 우측 쉬프트 횟수이다.)

	Djvu Brochure	Pennsylvania Railroad Stock certificate	Franklin Electronic Publishers
방법 1	26.52	23.99	27.89
방법 2	25.5	23.3	26.91
방법 3	25.72	23.05	26.65
방법 4	22.5	20.21	22.97

그림 10 양자화 방법에 대한 PSNR

되지 않는 결과를 보여 주었다. [그림 11]의 (a),(b),(c)는 방법 3으로 스칼라 양자화한 영상을 복원한 복원 영상이다. [그림 8]의 원래의 영상과 비교하여 큰 차이가 없는 것을 알 수 있다.



(a) DjVu brochure (512 × 512)
(b) Pennsylvania railroad stock certificate (512 × 512)
(c) Franklin electronic publishers (512 × 512)

그림 11 복원한 영상

6. 결 론

본 논문은 SPIHT 부호화 소요시간의 단축과 문자가 포함된 영상에서 문자의 가독성을 유지하면서 압축률은 높이기 위한 연구에 대하여 소개하였다. SPIHT 부호화 소요시간의 단축은 Quincunx 다운샘플링과 서브밴드 SPIHT 부호화 방법을 사용함으로서 이룩하였다. 그리고 압축율은 문자의 가독성을 유지하면서 최대 약 3% -6% (33:1-16:1)까지 할 수가 있었다.

향후의 발전 방향으로는 부호화 시간을 더욱 단축시키기 위하여 각 서브밴드가 일정 크기 이상이라면, 이를 다시 분할하여 부호화하는 방법을 생각해 볼 수 있다. 또 부호화 압축률의 향상이라는 관점에서 보면, SPIHT 부호화 알고리즘은 웨이블릿 변환된 영상의 특성에 따라 압출률이 다를 수 있으므로, 배경 영역과 문자 영역 각각에 대해 서로 다른 웨이블릿 변환을 적용한다면 압축률의 향상을 얻을 수 있을 것이다.

참 고 문 헌

[1] Leon Bottou, Patrick Haffner, Paul Howard, Patrice Simard, Yoshua Bengio and Yann LeCun, "High Quality Document Image Compression with DjVu," Journal of Electronic Imaging, 7(3), pp. 410 425, July 13, 1998.

[2] Amir Said and William 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, June 1996.
[3] Jerome M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Trans. on Signal Processing, Vol. 41, No. 12, Dec. pp 3445-3462, 1993.
[4] Raughuveer M. Rao and Ajit S. Bopardikar, Wavelet Transforms - Introduction to Theory and Application, Addison-Wesley, 1998.
[5] Gilbert Strang and Truong Nguyen, Wavelets and Filter Banks, Wellesley-Cambridge Press, 1997.
[6] Wim Sweldens, "Building your own wavelets at home," In Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes, 1996.
[7] Wim Sweldens, "The lifting scheme : A custom design construction of biorthogonal wavelets," Applied and Computational Harmonic Analysis, Vol. 3, No. 2, pp. 186-200, 1996.
[8] Mark Nelson, The Data Compression Book, Henry Holt & Company, November 1995.
[9] Michael David Adams and Faouzi Kossentini, "Reversible Integer-to-Integer Wavelet Transforms for Image Compression : Performance Evaluation and Analysis," IEEE Trans. on Image Processing, Vol. 9, No. 6, pp. 1010-1024, June 2000.

장 준

1993년 3월~2000년 2월 호서대학교 컴퓨터공학부 공학사 졸업. 2000년 3월~2002년 2월 호서대학교 컴퓨터공학부 대학원 석사 졸업. 2002년 3월~현재 (주) 지오매니아 근무. 관심분야는 영상처리, 디지털지도, CAD

이 호 석

1979년 3월~1983년 2월 서울대학교 전자계산기공학과 공학사 졸업. 1983년 3월~1985년 2월 서울대학교 컴퓨터공학과 대학원 석사 졸업. 1989년 3월~1993년 8월 서울대학교 컴퓨터공학부 대학원 박사 졸업. 1994년~현재 호서대학교 컴퓨터공학부 근무. 관심분야는 영상처리, 웨이블릿, MPEG-4/7/21, 프로그래밍 언어