DOI: 10.3745/KIPSTB.2010.17B.6.421

JPEG2000의 동적 ROI 코딩에서 개략적인 분할을 이용한 빠른 마스크 생성 기법

박 재 흥[†] · 이 점 숙[††] · 서 영 건[†††] · 홍 도 순[††] · 김 현 주[††††]

요 약

인터넷이 느린 환경이나 이미지가 큰 경우에 서버로부터 이미지를 전부 보는 데는 시간이 걸린다. 이를 위해, 이미지의 일부를 우선적으로 처리하는 방법이 필요하여, JPEG2000에서는 ROI(Region-of-Interest) 코딩으로 제공하고 있다. ROI 코딩은 이미지의 thumbnail을 사용자에게 보내어 개략적인 이미지를 보고 먼저 보고 싶어 하는 영역을 지정하면, 이 영역에 관한 정보가 서버로 전달되어, 서버는 사용자에게 이 영역을 우선적으로 전송한다. 이를 처리하기 위해, 기존의 방법은 사용자가 지정한 영역을 계산할 때, 섬세하게 계산을 함으로써 영역에 관한 마스크 정보 생성 시간을 많이 요구하였는데, 본 연구에서 제안하는 방법은 개략적으로 영역을 계산하고, ROI 마스크의 생성시간을 줄이는 방법을 제안한다. 각 블록에서 ROI와 배경이 섞여 있는 블록이면 적절한 경계선을 찾기 위하여 코드블록의 가장자리를 이진 탐색하여 두 경계지점을 얻고, 두 점 간의 중간에 위치한 코드블록 내의 경계점을 얻기 위하여, 두 점 간의 중간 지점을 이진 탐색한다. 이렇게 얻어진 세 점을 지나는 두 직선의 방정식을 이용해서 개략적인 ROI 코딩을 한다. 제안한 방법은 품질 면에서는 큰 차이가 없지만 실행 속도 면에서는 현저하게 빠르다는 것을 보인다.

키워드 : JPEG2000, ROI 코딩, 관심영역

A Technique Getting Fast Masks Using Rough Division in Dynamic ROI Coding of JPEG2000

Jae-Heung Park[†] · Jum-Sook Lee[††] · Yeong-Geon Seo[†††] · Do-Soon Hong[††] · Hyun-Joo Kim[††††]

ABSTRACT

It takes a long time for the users to view a whole image from the image server under the low-bandwidth internet environments or in case of a big sized image. In this case, as there needs a technique that preferentially transfers a part of image, JPEG2000 offers a ROI(Region-of-Interest) coding. In ROI coding, the users see the thumbnail of image from the server and specifies some regions that they want to see first. And then if an information about the regions are informed to the server, the server preferentially transfers the regions of the image. The existing methods requested a huge time to compute the mask information, but this thesis approximately computes the regions and reduces the creating time of the ROI masks. If each code block is a mixed block which ROI and background are mixed, the proper boundary points should be acquired. Searching the edges of the block, getting the two points on the edge, to get the boundary point inside the code block, the method searches a mid point between the two edge points. The proposed method doesn't have a big difference compared to the existing methods in quality, but the processing time is more speedy than the ones.

Keywords : JPEG2000, ROI Coding, ROI

1. 서 론

인터넷이 활발하게 이용되면서 몇 년 전에는 불가능할 것으로 여겨졌던 서비스들이 이제는 우리 생활에 밀접하게 다가와 있다. 특히 영상은 지금 인터넷 트래픽의 95% 이상을 차지하며, 많은 응용 분야에서 사용되고 있다[1]. 그래서 영상 데이터는 양이 많아 데이터량을 줄이는 것과, 먼 곳으로 오류 없이 빠르게 전송하는 것이 필요하게 되었다. 이런 분야는 이미 많은 연구가 되어 왔고[2, 3], 최근에는 많은 데이터량 중에서 특정 부분만 미리 빠르게 볼 수 있게 하는 ROI 코딩을 JPEG2000 표준[4, 5]에서 지원하고 있다.

† 정 회 원 : 경상대학교 컴퓨터과학과 교수
†† 준 회 원 : 경상대학교 컴퓨터과학과 박사과정
††† 종신회원 : 경상대학교 컴퓨터교육과 교수, 컴정연구소원(교신저자)
†††† 정 회 원 : 진주산업대학교 컴퓨터공학과 교수

논문접수 : 2010년 8월 17일
수 정 일 : 1차 2010년 9월 27일, 2차 2010년 10월 11일
심사완료 : 2010년 10월 15일

JPEG2000의 특징은 무손실/손실 압축, 무손실 코딩에의 내포된 손실, 화소 정확성과 해상도에 의한 점진적인 전송, 비트 에러 그리고 관심영역 부호화 등이 있다[6, 7]. 특히 ROI 코딩은 영상 내의 관심영역을 배경보다 먼저 전송하여 사용자가 볼 수 있게 할 수 있으며, 낮은 압축률로 배경보다 고품질로 영상을 저장하는 것을 말한다. 이렇게 하면, 사용자 입장에서 원하는 부분을 먼저 볼 수 있으며, 그리고 고품질의 영상으로 볼 수 있다. 또한 저비트율의 통신 환경에서는 전체 영상을 모두 받지 않고 원하는 관심영역만 볼 수도 있다[8-10]. ROI 코딩 방법은 정적과 동적으로 나뉜다. 정적 ROI 코딩 방법은 이미지 코딩 과정에서 ROI 마스크를 생성한 후 웨이블릿 계수 단위로 우선적 처리하는 방법이다. 이 방법은 인코딩 시에 ROI가 이미 결정되어 압축되는데, 일반적으로 이미지 전송은 압축 후에 필요에 의해 수행되기 때문에 ROI 코딩 시간이 이미지 전송 시간에 직접적으로 영향을 미치지는 않는다. 표준에서는 Maxshift[9]와 Scaling based[10] 방법이 있다.

동적 ROI 코딩 방법은 인코딩된 비트스트림의 일부를 사용자에게 전송한 후, 사용자가 관심영역을 지정하면, 그 때 ROI가 지정된다. 이 방법에는 표준에서 제안하고 있는 묵시적[9] 방법이 있으며, 동적 ROI 방법에서는 무엇보다도 ROI 마스크를 빠르게 생생해야 한다. [11]에서는 하나의 블록 내에서 ROI와 배경 영역을 구분하는데, 가장자리를 탐색하여 배경과 ROI가 나뉘는 두 점을 얻어, 두 점을 직선으로 연결하여 개략적인 마스크를 생성하였다. 또한, 자동적으로 관심영역을 추출하는 연구도 활발히 진행되고 있다[12, 13]. ROI 기법은 사용자의 관심영역을 빠르게 지정하고 그 정보를 서버로 전송해야만 진정한 서비스라고 할 수 있다. 하지만 기존의 방법들은 영역을 처리하는 데 시간을 소모한다거나, 영역을 대충 결정하는 문제가 있었다. 시간을 소모한다는 것은 블록 내의 모든 픽셀을 탐색하여 마스크를 결정하는 것이고, 영역을 대충 결정한다는 것은 블록 내에서 한 픽셀이라도 ROI 영역에 속하면 그 블록 전체가 ROI 블록으로 처리되는 문제 등이다.

ROI 마스크는 한 블록의 전체가 ROI인 경우는 ROI 마스크 처리되며, 한 블록 내에 ROI와 배경이 섞여 있는 경우에는 일부가 ROI 마스크로 처리 되어야 한다. 이렇게 두 가지가 섞여 있는 경우에, 빠르게 ROI 마스크를 계산하고 서버로 전송하는 데, 화질은 최대한 보장해 주는 것이 중요하다. 기존의 ROI 마스크 생성 방법들은 ROI 마스크를 생성하기 위해 한 블록을 모두 탐색하면서 마스크 테이블을 만들지만, 제안한 방법은 한 코드블록의 네 꼭짓점에서 가장자리를 따라 배경과 ROI를 나누는 점을 이진 탐색을 하여 구한다. 이 때, 이진 탐색을 3회만 한다. 구해진 두 점을 연결하면 직선이 되어 이를 보정하기 위해 두 점의 중간 지점을 한 번 더 탐색하여 한 점을 구하여, 세 점을 연결하는 배경과 ROI를 나누는 마스크를 구한다.

2. ROI 코딩

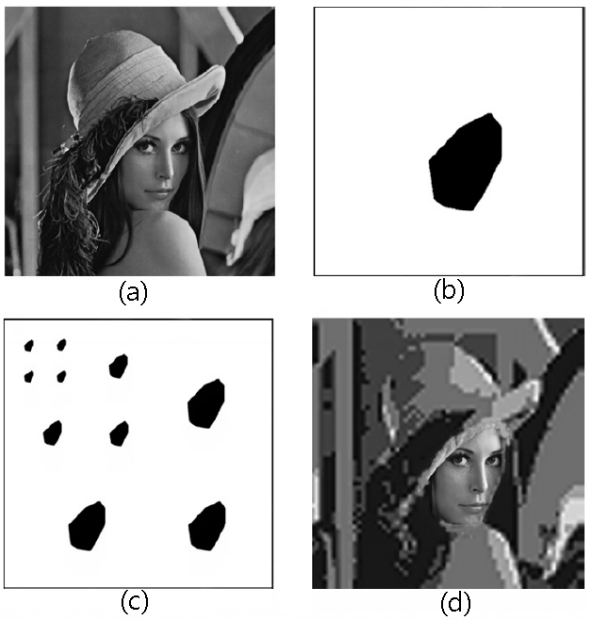
인코딩 과정에서 관심영역을 미리 알지 못하는 경우에 디코딩 과정에서 ROI를 지정하게 된다. 이럴 경우 디코딩 과정에서 사용자가 원하는 ROI 모양 정보를 인코더에 넘겨주면 인코더에서는 그 정보를 기반으로 ROI 코딩이 되어 있지 않은 압축된 비트스트림으로부터 ROI를 추출하여 우선처리를 하게 한다. 이것을 동적 ROI 코딩 방법이라고 한다.

Maxshift 방법[9]은 양자화된 계수 중에 ROI 계수와 배경 계수를 구분하여, 배경 계수 중에서 가장 큰 계수 값, s를 구한 다음, ROI 계수를 s보다 높은 비트-평면에 이동시키는 방법으로서 JPEG2000 Part1 표준이다. (식 1)은 s를 구하는 수식이고, (식 2)는 ROI 처리 후의 계수, $a'(u,v)$를 구하는 수식이다.

$$s \geq \max(M_b) \qquad \text{(식 1)}$$

$$a'(u,v) = \begin{cases} a(u,v), & M(u,v) = 0 \\ 2^s a(u,v), & M(u,v) = 1 \end{cases} \qquad \text{(식 2)}$$

(식 1)에서 $\max(M_b)$은 각 서브밴드에서 양자화된 배경 계수 중에 가장 큰 값을 의미한다. (식 2)에서 $M(u,v)$는 ROI 마스크 정보로서 계수가 ROI에 속하는 좌표인 경우는 1, 배경에 속하는 좌표인 경우는 0을 의미한다. 장점은 인코딩 시에 ROI 모양 정보 대신 s값만 가지게 되므로 코딩 효율이 좋다. 단점은 ROI가 모두 복원될 때까지 배경을 얻을 수 없다. 즉 ROI 중요도 조절이 불가능하다. 또한 다수 ROI 지원이 어렵다. (그림 1)은 임의의 ROI를 갖는 Lenna 영상을 0.26bpp로 Maxshift 압축을 한 후에 재구성된 영상을 보인다.



(그림 1) (a) Lenna 원 이미지, (b) 공간 도메인에서 ROI 위치 (c) 2-level Maxshift ROI 계수 마스크
(d) 0.26bpp에서 Maxshift 압축 후의 재구성된 Lenna

묵시적 ROI 방법[9]은 전체 손실 최소화에 의해서 코드블록 공헌도를 할당하기 때문에, 손실 감소와 ROI가 일치하도록 코드블록 공헌도를 할당한다. (식 3)은 손실 계산방법을 나타낸다.

$$D_j^{n_i}=\begin{cases} W_{ROI}w_b \sum_{u,v\in B_j}(\widehat{a^{n_i}}(u,v)-a(u,v))^2, & ROI \text{ 코드블록} \\ w_b \sum_{u,v\in B_j}(\widehat{a^{n_i}}(u,v)-a(u,v))^2, & \text{그외} \end{cases} \quad \text{(식)}$$

(식 3)에서 W_{ROI}은 가중치이고, $D_j^{n_i}$은 n_i에서의 가중 MSE 손실이고, w_b은 B_j을 포함하는 서브밴드의 가중치이고, B_j은 j번째 코드블록이고, $a(u,v)$은 계수이고, $\hat{a}(u,v)$은 절단점 n_i에 관련된 양자화된 계수들이고, n_i은 절단점을 의미한다. 이 방법의 장점은 복잡도가 낮아 구현이 쉽고, 비트-평면 이동을 하지 않는다. 단점은 코드블록단위로 ROI가 지정되기 때문에 다각형 모양만 지원하다. 또한 ROI 코드블록 안에 포함된 배경 계수도 ROI 처리를 함으로서 ROI 코딩 성능이 떨어진다.

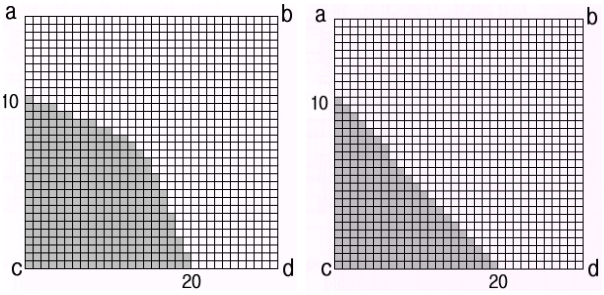
Maxshift 방법과 수정된 묵시적 방법[10]은 모든 픽셀을 탐색하고, 묵시적 방법은 최악의 경우에 모든 픽셀을 탐색한다. 모든 픽셀을 탐색하는 것은 한 블록 내에서 ROI 픽셀과 배경 픽셀을 픽셀 단위로 정확하게 구분해 내기 위한 것이다. 묵시적 방법은 한 블록 내에 하나의 픽셀이라도 ROI 픽셀에 속하면 그 블록 전체가 ROI 블록으로 결정된다.

수정된 묵시적 ROI 방법은 알고리즘 복잡도 없이 ROI 코드블록에 포함된 배경 계수의 우선권을 줄여 묵시적 방법을 보안한 방법이다. ROI 코드블록 내의 배경 계수의 우선권 조절은 배경 계수의 k LSB만큼 절단하는 방법이다. (식 4)는 ROI 처리 전의 ROI 코드블록 내의 계수인 $a(u,v)$를 우선권 조절 후의 계수인 $\overline{a}(u,v)$로 변환하는 식이다.

$$\overline{a}(u,v)=\begin{cases} sign[a(u,v)]\left\lfloor \frac{|a(u,v)|}{2^k} \right\rfloor 2^k, & \text{배경 계수} \\ a(u,v), & ROI \text{계수} \end{cases} \quad \text{(식 4)}$$

장점은 계수 단위로 ROI 처리가 가능하며, 전체 ROI 전송이 묵시적 ROI 방법보다 약 24% 속도가 빠르다는 것이다. 그리고 ROI 크기가 작을수록 무손실 재구성 비트율도 낮아진다. 단점은 ROI 코드블록에 포함된 k LSB의 절단 때문에, 무손실 ROI 코딩 방법과는 호환되지 않는다.

기울기 기반 방법[11]은 블록 내의 계수를 전부 탐색하지 않고 일부만 탐색하여 개략적인 정보를 이용해 빠르게 마스크를 생성하는 방법이다. 개략적인 정보는 블록의 가장자리를 따라 ROI와 배경 간의 경계를 탐색한 후, 탐색된 두 가장 자리를 직선으로 연결하여 한쪽은 배경, 한쪽은 ROI로 간주한다. (그림 2)의 (a)의 회색 부분은 ROI 영역이고 오른쪽은 배경이다. (b)는 (a)에서 나누어진 영역을 경계선으로



(a) 원래 경계 (b) 경계선 구분

(그림 2) [11] 방법의 배경과 ROI 구분 예

그은 그림을 보이고 있다. 이 방법은 빠르지만 정밀도는 조금 떨어진다.

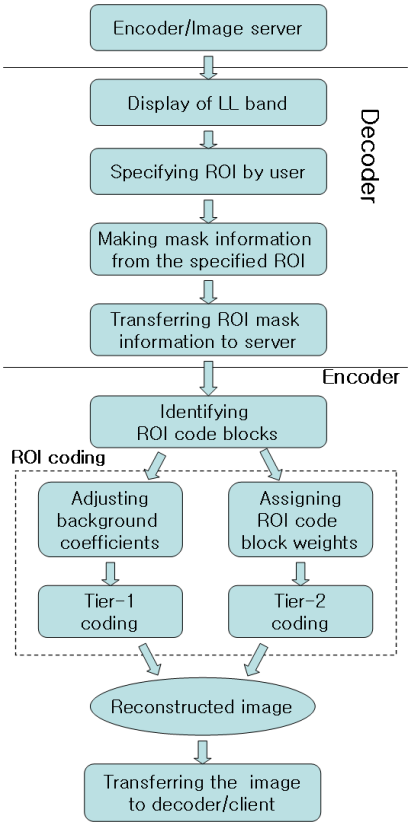
기존의 방법은 ROI와 배경의 경계를 찾기 위하여 많은 시간을 소모한다. 모든 픽셀을 탐색하는 경우는 경계는 명확하게 찾지만 탐색 시간이 많이 걸리는 단점이 있다. [11]에서 제안한 방법은 이를 개선하기 위한 것이지만, 가장자리 점을 탐색해 나가는 과정에서 시간을 소모한다. 이에 반해, 제안하는 방법은 약간의 품질은 떨어지지만 탐색시간과 마스크 생성 시간을 줄인다. 또한 품질은 기존의 방법에 비해 큰 차이를 보이지는 않는다.

3. 세 점 기반의 경계선을 이용한 마스크 생성

(그림 3)은 JPEG2000의 전체 구성도이고, ROI 코딩은 점선으로 표시된 부분만 해당된다. 인코더 또는 서버에서 전송된 이미지는 개략적인 이미지인 LL 밴드만 우선 전송되며 사용자는 이 개략적인 이미지를 보고 먼저 볼 영역을 표시한다. 이렇게 표시된 영역을 ROI라 하며, 지정된 ROI에 관한 정보는 서버로 전송되어 ROI 코딩된다. 이 과정에서 빠른 처리를 요하기 때문에 정확한 ROI 영역을 표시하기보다 약간의 품질은 떨어지더라도 빠르게 처리하는 것이 중요하다[11]. ROI에 포함되는 마스크 정보는 (식 5)와 같이 구성된다. 값이 1이면 ROI에 해당되는 픽셀을 의미하고 0이면 배경이 된다.

$$M(x,y)=\begin{cases} 1, & ROI \\ 0, & \text{배경} \end{cases} \quad \text{(식 5)}$$

웨이블릿 도메인에서의 ROI 마스크 정보는 이미지 도메인에서의 ROI 마스크 정보와 달라서, IDWT를 이용하면 구할 수 있다. 즉, 웨이블릿 도메인에서의 ROI 마스크 생성 과정은 우선 이미지 도메인에서의 ROI 마스크와 마지막 IDWT에 의해 2개의 서브밴드 안에 어떤 위치의 계수가 ROI 계수로 재구성하기 위해 사용되었는지를 추적하게 된다. 그런 후에, 두 서브밴드의 ROI 마스크와 마지막 이전의 IDWT에 의해 각각 2개의 서브밴드 안에 어떤 위치의 계수가 ROI 계수로 재구성하기 위해 사용되었는지를 추적하게 된다. 이렇게 하여 각 분해레벨에서 모든 서브밴드 내에 어떤



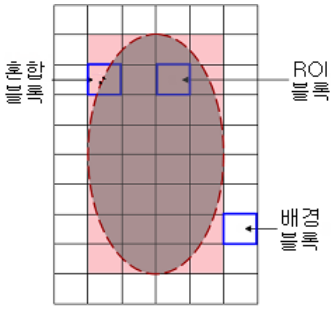
(그림 3) ROI 코딩을 지원하는 JPEG2000

위치의 계수가 ROI 계수로 재구성하기 위해 사용되었는지를 추적하게 된다.

3.1 ROI 지정과 경계선 찾기

각 코드블록은 32x32 픽셀로 하며, 64x64도 가능하다. 다만 코드블록이 커지면 ROI의 정밀도가 떨어진다. 한 코드블록이 모든 픽셀을 탐색하여 정확하게 ROI를 정의한다는 것은 많은 시간이 소요될 뿐 아니라, 사람의 눈으로는 정확한 것과 유사한 것을 거의 구분하지 못한다. 그러므로 오히려 ROI 영역을 개략적으로 정의하고 빠른 시간 안에 마스크를 생성하는 것이 중요하다.

(그림 4)와 같이 사용자가 개략적인 이미지를 보고 ROI을 정의하면, 모든 코드블록은 다음 종류 중 한 가지로 분류된다.



(그림 4) ROI 지정 예

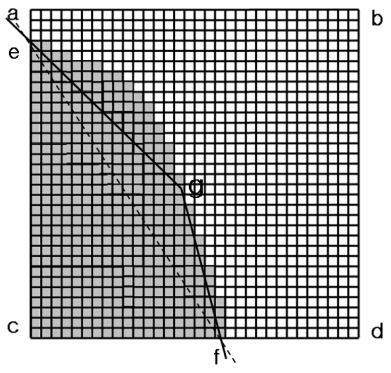
- ROI 코드블록 (abcd = 1111)
- 배경 코드블록 (abcd = 0000)
- 혼합 코드블록 (abcd ≠ 1111 and abcd ≠ 0000)

ROI로 지정된 영역 내부의 코드블록은 ROI 코드블록이며, 외부는 배경 코드블록, 경계선에 있는 것은 혼합 코드블록이다. 배경 코드블록은 ROI 코딩에서는 무시되며 ROI 코드블록은 전체가 ROI 이므로 마스크도 쉽게 구할 수 있다. 다만 경계선에 있는 혼합 코드블록은 한 블록 내에 어느 경계를 기준으로 한쪽은 ROI 영역이고, 한쪽은 배경 영역이 되는지를 빠른 시간 안에 계산하여 서버로 전송하여야 한다.

(그림 5)와 같이 혼합 블록인 경우는 a-c와 c-d의 가장자리를 따라 얻어지는 두 점 e와 f를 얻고, e와 f의 중간점 g를 얻는다. 세 점이 구해지면, e-g를 지나는 직선과 f-g를 지나는 직선의 왼쪽 부분이 ROI로 설정된다. <표 1>에서 R은 오른쪽, L은 왼쪽 T는 위쪽, B는 아래쪽을 의미한다. g점을 탐색하기 위한 변의 위치는 ROI가 오른쪽에 위치하면 b-d 변에서 탐색하여 왼쪽(L) 방향으로 탐색한다(b-d→L). 이 때 탐색 변의 위치는 ROI가 걸쳐 있는 중간점에 시작한다. e와 f를 얻을 때, 변의 탐색 방향은 <표 1>에 보인 것처럼 탐색 순서를 따른다. 예를 들어 abcd가 0010이면 e를 얻기 위하여 a→c로, f를 얻기 위하여 c→d로 탐색한다.

<표 1>에서 abcd의 값이 0110과 1001은 ROI가 한 블록 내에서 두 개로 나뉘어져 있는 경우인데, 이 경우는 실제로 일어 날 수 없는 것으로 간주하여 본 연구에서는 제외했다. 클라이언트 영역의 비트 테이블(BT[32][32])로부터 마스크 정보를 구하는 알고리즘은 다음과 같다.

```
a,b,c,d = 0 or 1 of 4 points from BT[32][32] ;
id = a<<3 | b<<2 | c<<1 | d ;
if (id is 0 or 15) current_cb = id ;
else if (id is 6 or 9)
        current_cb = UNDEFINED ;
else begin
 (* e를 얻기 위한 첫 탐색 *)
   (* <표 1>의 탐색 순서에서 첫 번째 탐색 방향 *)
  e = scan(c→d) or scan(a→b) or scan(a→c)
```



(그림 5) 혼합 블록과 네 꼭짓점(abcd), 세점(efg)

〈표 1〉 ROI 분포에 따른 탐색 순서

a b c d	탐색순서	ROI위치	g점 탐색 방향
0 0 0 0	배경 블록		
1 1 1 1	ROI 블록		
0 0 0 1	c→d, b→d	R	b-d→L
0 0 1 0	a→c, c→d	L	a-c→R
0 0 1 1	a→c, b→d	B	c-d→T
0 1 0 0	a→b, b→d	R	b-d→L
0 1 0 1	a→b, c→d	R	b-d→L
0 1 1 0	제외		
0 1 1 1	a→b, a→c	R	b-d→L
1 0 0 0	a→b, a→c	L	a-c→R
1 0 0 1	제외		
1 0 1 0	a→b, c→d	L	a-c→R
1 0 1 1	a→b, b→d	L	a-c→R
1 1 0 0	a→c, b→d	T	a-b→B
1 1 0 1	a→c, c→d	R	b-d→L
1 1 1 0	c→d, b→d	L	a-c→R

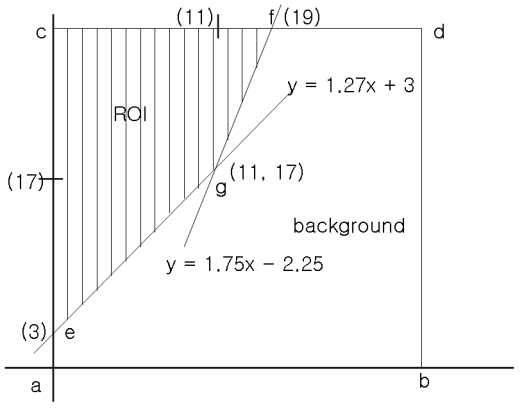
```
(* f를 얻기 위한 두 번째 탐색 *)
  (* <표 1>의 탐색 순서에서 두 번째 탐색 방향 *)
  f = scan(b→d) or scan(c→d) or scan(a→c)
(* g를 얻기 위한 세 번째  탐색 *)
  (* abcd가 0001인 경우, 즉 id가 1인 경우
     b-d 에지에서 Left 방향으로 탐색
     f는 b-d 에지에서 경계점
     31은 b-d 에지에서 d 점, 즉 끝 점     *)
  if (id is 1) g = scan(b-d→L, f, 31) ;
  (* abcd가 0010인 경우, 즉 id가 2인 경우
     a-c 에지에서 Right 방향으로 탐색
     e는 a-c 에지에서 경계점
     31은 a-c 에지에서 c 점, 즉 끝 점     *)
  else if (id is 2) g = scan(a-c→R, e, 31) ;
  else if   (* id = 3,4,5,7,8,10,11,13,14 생략 *)
  end if // of  end if
end if   // of begin
(* 함수 scan(), direct 방향에서 중앙으로 탐색
 탐색 시작 위치는 start_idx, 종료위치는 end_idx *)
integer scan(direct, start_idx = 0, end_idx = 31)
begin
   idx1 = 0 ; (* 이진 탐색을 위한 첫 idx *)
   idx2 = 31 ;(* 이진 탐색을 위한 끝 idx *)
   temp = (idx1 + idx2) / 2 ; (* 중간점 *)
   (* mid : 시작점과 끝점의 중간 점 *)
   mid = (start_idx + end_idx) / 2 ;
   (* temp와 mid를 index로 하여 이진탐색
      처음 탐색 시작점의 비트값(1 또는 0)과
      같으면 탐색 방향으로 다르면 반대 방향으로
      인덱스(temp) 조정,  이 과정을 3회만 반복 *)
   return (temp) ;
end function scan
```

이 알고리즘이 수행되고 나면, abcd 4 비트와 efg의 각 값을 나타내기 위한 5비트씩 3개, 합하여 19비트의 ROI 마스크 정보가 서버로 전송된다. 알고리즘에서 '<<"는 비트단위의 Left-Shift를 의미하며, '|'는 비트 단위의 OR를 의미한다. scan() 함수에서는 가장자리 한 라인이나 e 또는 f에서 시작하여 꼭짓점까지의 중간 한 라인에서 0과 1로 접하는 점을 이진 탐색한다. 여기에서 최적의 점을 찾을 수도 있지만 3회만 이진 탐색하여도 최대 3픽셀 거리에 위치하게 된다.

3.2 ROI 마스크 생성

얻어진 ROI 마스크 정보를 이용하여 최종 ROI 마스크 테이블이 만들어지고, 이후에 EBCOT과정을 통하여 ROI 영역은 우선 처리되어 디코더로 전송되도록 해야 한다. 이 과정은 신속히 처리되어야 하며, 불필요한 계수는 탐색할 필요가 없게 만들어야 한다. 본 연구에서 사용되는 방법은 ROI와 배경을 구분하여 ROI 부분만 정확하게 탐색하여 1로 설정한다.

두 점을 지나는 일차 방정식 두 식을 구하고 이 식을 기준으로 왼쪽(위쪽) 또는 오른쪽(아래쪽) 만을 탐색하여 ROI 영역으로 지정한다. (그림 5)의 예에서 얻어진 경계 부분의 방정식은 $y = 1.75x - 2.25$와 $y = 1.27x + 3$이며, 왼쪽 부분이 ROI가 된다. (그림 5)에서는 경계선의 기울기가 ⊖로 보이지만, 모서리 a 부분에서의 인덱스가 0이므로, x 축 기준으로 180도 회전시켜야 일반적인 2차원 평면이 될 수 있다. 그래서 기울기가 ⊕가 된다. ROI 영역만 탐색하기 위해서는 기울기가 ⊕ 또는 ⊖ 인지를 구분하여 탐색해야 하며, ROI 영역이 왼쪽에 있는지 오른쪽에 있는지도 구분하여 탐색해야 한다. 이런 네 가지 경우를 고려하여 최적의 탐색 알고리즘은 다음과 같다. 여기서 a1과 a2는 두 경계 직선의



(그림 6) (그림 5)로부터 얻어진 경계선의 방정식

각각의 기울기이며, b1은 e점을 지나는 직선의 y 절편이며, b2는 f 점을 지나는 직선의 y 절편이다. 두 직선의 교차점은 xx, yy이다.

○ ROI 분포가 L인 경우
  max(0, b1) <= y < min(yy, b1)인 y에 대해
    0 <= x <= min(xx, (yy-b1)/a1)인 x 에 대해
      mask[y][x] = 1 ;
  max(yy, b2) <= y <= min(31, b2)인 y에 대해
    0 <= x <= min(xx, (31-b2)/a2)인 x 에 대해
      mask[y][x] = 1 ;
○ ROI 분포가 T인 경우
  0 <= x < xx인 x와 b1 <= y <= 31인 y에 대해
      mask[y][x] = 1 ;
  xx <= x < 31인 x와 b2 <= y <= 31인 y에 대해
      mask[y][x] = 1 ;
○ (ROI 분포가 R 또는 B인 경우 생략)

이 알고리즘에서는 마스크 테이블의 32X32 영역 내에서 ROI 영역만을 탐색하여 1로 설정하도록 계산되어야 하기 때문에 x, y의 인덱스 값 계산이 네 가지 경우로 나누어서 계산되었다. (그림 6)은 ROI 위치가 L의 경우인 (그림 5)의 예를 보인 것이다. x가 가질 수 있는 값의 범위는 0~31이나, 그림에서 보듯이 19를 초과하는 범위는 수행할 필요가 없다. 그래서 x는 0에서 19까지의 값만 처리한다. 여기서 19는 31=1.75x-2.25 식에서 x=33.25/1.75, 즉, (b-31)/a로부터 얻어진다. y의 범위도 x와 마찬가지로 0~31이나 그림에서 3 미만은 의미가 없다. y는 x=0일 때이므로, y는 3, 즉, (ax+b)에서 시작한다.

3.3 코드블록의 우선 처리

EBCOT에서 각 품질 레이어는 코드블록의 임베디드 비트 스트림으로부터 임의의 공헌도를 포함한다. 따라서 코드블록 우선적 처리는 각 코딩 패스에서 손실률을 조절한 후, PCRD(Post-Compression Rate-Distortion) 최적화 알고리즘을 다시 수행한다. 전체 손실 최소화에 의해서 코드블록 공헌도를 할당하기 때문에, 관심영역은 손실 감소와 관심영역이 일치하도록 코드블록 공헌도를 할당한다. (식 3)에 의해 손실 계산 방법은 이루어진다.

4. 실험 및 평가

실험을 위해서 객관적 화질 평가인 PSNR과 ROI 마스크 생성 시간들을 비교한다. 동일한 실험 조건을 위해 W_{ROI}의 값은 4096로 설정하였으며, Maxshift 방법을 제외한 모든 방법에서 k의 값은 5로 설정하였다. 관심영역의 모양은 타원이며, 크기는 전체 이미지의 25%이며, Taehee 영상을 사용하였다.

4.1 객관적 화질 평가

기존 ROI 코딩 방법들은 마스크 생성을 위해 모두 순차 탐색을 하는 반면에, 제안한 방법은 (그림 6)과 같이 일부 마스크 정보만 탐색한다. 샘플당 n 비트의 길이를 가지는 이미지를 위한 PSNR은 (식 6)과 같고

$$PSNR = 10\log_{10}\frac{(2^n-1)^2}{MSE} \qquad \text{(식 6)}$$

MSE는 (식 7)과 같다.

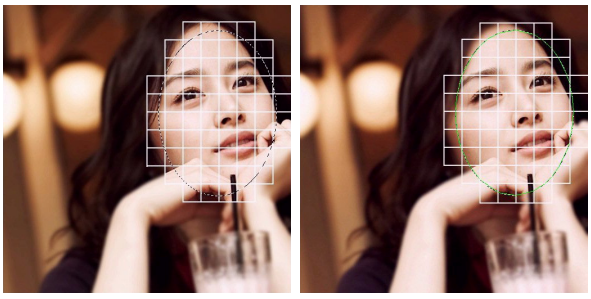
$$MSE = \frac{\sum(\hat{x}-x)^2}{A} \qquad \text{(식 7)}$$

여기서 x는 원 화소 값을 의미하고, $\hat{x}$은 재구성된 화소 값을 의미하고, A는 ROI를 의미하고, 따라서 ROI PSNR은 ROI의 R-D 성능을 측정하여 구할 수 있다. <표 2>는 Taehee 이미지에 대한 각 비트율에서 ROI의 PSNR을 나타내고 있다. 표에서 보면 제안한 방법이 기존의 방법에 비해 객관적 화질이 비슷하다는 것을 알 수 있다.

(그림 7)은 묵시적 방법과 제안 방법에 의해 지정된 ROI의 영역을 객관적 관점으로 표시하고 있다. 사각형은 64x64 크기의 ROI 블록이다. ROI 코드블록의 가장 자리를 따라 혼합 블록이 존재하며, 이 블록에서 ROI 계수와 배경 계수가 나누어지는 영역을 보이고 있다. (a)의 네모 안에 있는 모든 블록은 배경보다 화질이 좋게 표시되며, (b)는 제안한 방법으로 가장 자리의 혼합 블록 내에서도 안쪽의 ROI 계수는 좋은 화질로, 바깥의 배경 ROI 계수는 덜 좋은 화질로 표시되어 있다.

<표 2> ROI PSNR 비교

코딩방법 / 비트율	묵시적 [9]	Maxshift [9]	수정된 묵시적[10]	기울기기반방법[11]	제안 방법
0.0625	24.200	27.885	24.668	24.616	24.634
0.125	27.299	30.289	28.383	28.208	28.290
0.25	30.919	33.876	32.617	32.455	32.522
0.5	35.897	39.014	38.288	38.110	38.209
1.0	43.230	47.733	46.312	46.129	46.200



(a) (b)

(그림 7) 묵시적 방법과 제안 방법에서의 ROI 지정 예

4.2 ROI 마스크 생성 시간 비교

대부분의 ROI 코딩 방법들은 마스크 생성을 위해 순차 탐색을 하는 반면에, 제안한 방법은 <표 3>과 같이 일부 마스크 정보만 탐색한다. Maxshift 방법과 수정된 묵시적 방법은 모든 픽셀을 탐색하고, 묵시적 방법은 최악의 경우에 모든 픽셀을 탐색하지만, 첫 번 탐색에서 ROI로 판정되면 그 코드블록이 모두 ROI 블록으로 간주되므로 1회에 탐색이 끝날 수도 있다. 기울기 기반 방법은 우선 모서리 네 점을 먼저 탐색하므로 4회, 두 변을 따라 탐색하므로 평균 16x2회 탐색을 하게 된다. 제안한 방법은 기울기 기반 방법과 유사하나 탐색 시에 이진 탐색을 3회 하며 중간점을 한 번 탐색한다. <표 3>은 디코더에서 ROI 정보를 얻기 위해 탐색하는 시간을 나타내고 있다. 코드블록의 크기가 32일 때, n^2은 1024 회를 탐색해야 하며, 기울기 기반 방법의 혼합 블록인 경우에는 꼭짓점 탐색 4회, 가장 자리를 탐색하기 위해 32회를 해야 한다. 제안한 방법에서는 꼭짓점 탐색 4회, 두 변의 가장 자리 2진 탐색을 위해 3회 x 2 = 6회, 두 점의 중간점을 탐색하기 위해 3회를 의미한다.

<표 4>는 ROI 마스크 생성 시에 소요되는 시간을 표현하고 있다. 제안된 방법은 ROI 영역을 구분하기 위해 1차원 방정식 두개로 표현되어 있으므로 인덱스 값의 변화를 정확하게 ROI 영역 내에서만 움직이도록 되어있다. ROI 정확도는 모든 영역을 탐색하여 하나의 계수 범위까지 마스크를 표현해 주는 Maxshift와 수정된 묵시적 방법이 가장 좋다. 하지만, 제안된 방법은 전부를 탐색하지 않고도 거의 비슷한 ROI 마스크를 얻을 수 있다.

〈표 3〉 ROI 정보 평균 탐색 시간

코딩방법 / 탐색횟수	Maxshift [9]	묵시적 [9]	수정된 묵시적[10]	기울기기반방법[11]	제안 방법
배경 블록	n^2	n^2	n^2	4	4
ROI 블록	n^2	1	n^2	4	4
혼합블록	n^2	$n^2/2$	n^2	4 + n	4+6+3

〈표 4〉 마스크 생성 시간 비교

코딩방법 / 탐색횟수	Maxshift [9]	묵시적 [9]	수정된 묵시적[10]	기울기기반방법[11]	제안 방법
배경 블록	0	0	0	0	0
ROI 블록	n^2	n^2	n^2	n^2	n^2
혼합 블록	n^2	n^2	n^2	$n^2/2$	$n^2/2$

5. 결 론

JPEG2000에서 기존의 방법들은 관심영역을 처리하는 데 시간을 소모한다거나, 영역을 대충 결정하는 문제가 존재한다. 이러한 문제점들을 개선하여 ROI와 배경을 구분하는 데 개략적인 구분 정보를 이용하여, 빠른 ROI 영역을 탐색한 후 마스크를 생성하여, 결코 품질 면에서는 뒤떨어지지 않는 방법을 본 연구에서는 제안했다. 우선 코드블록의 네 모서리를 탐색하고, 이 정보를 기반으로 코드블록의 가장자리 두 방향을 탐색하여 배경과 ROI의 경계의 두 점을 얻는다. 두 점을 직선으로 그으면 중간 부분에 많은 차이가 나므로 중간점을 한 번 더 탐색하여 차이를 보정하였다. 이 방법은 코드블록의 탐색 시간을 줄여줄 뿐 아니라, 인코더에서 마스크 테이블을 생성하는 시간을 줄였다. 또한 다른 방법에 비해 본래의 ROI 모양과 근접한 ROI 마스크 테이블을 얻을 수 있었다.

제안한 방법은 개략적인 모양으로 ROI를 구하였음에도 화질 면에서는 결코 기존의 방법에 뒤떨어지지 않았으며, ROI 정보 평균 탐색 시간은 상수로써 나타낼 수 있었는데 비해 기존의 방법은 ROI의 크기에 비례하여 계산 복잡도를 나타내었다. 마스크 생성 시간도 기존의 방법에 비해 반으로 줄였다. 향후 연구 방향은 ROI 결정 시간에 큰 영향을 미치는 ROI의 자동 추출 방법과 본 연구 결과를 결합하여 훨씬 더 좋은 서비스를 연구하는 것이다.

참 고 문 헌

[1] Y. Shujjing, G. Yanfeng and Z. Ye, “An Automatic ROI Coding Algorihtm Based on Multi-Resolution Target Detection”, ISSCAA 2nd Int'l Symposium on Digital Object Identification, pp.1-4, 2008.

[2] 강기준 외, “JPEG2000 이미지에서 적응적 코드블록 판별 알고리즘을 이용한 동적 고속 관심영역 코딩 방법”, 한국정보처리학회 논문지 B, 제14-B권, 5호, pp. 321~328, 2007.10.

[3] J. Hara, “An Implementation of JPEG 2000 Interactive Image Communication System”, ISCAS 2005, Vol.6, pp.5922-5925, May 2005.

[4] J. In, S. Shirani, and F. Kossentini, “On RD Optimized Processive Image Coding Using JPEG”, IEEE Transactions on Image Processing, Vol.8, No.11, pp.1630-1638, Nov. 1999.

[5] Ahn, C. B., Kim, I. Y. and Han, S. W., “Medical Image Compression Using JPEG Progressive Coding”, Nuclear Science Symposium and Medical Imaging Conference, 1993 IEEE Conference Record, pp.1336-1339, Nov. 1993.

[6] ISO/IEC International Standard 15444-1, ITU Recommendation T.800, “JPEG2000 Image Coding System”, 2000.

[7] C. Christopoulos, A. Skodras and T. Ebrahimi, ”The JPEG2000 Still Image Coding System : An Overview”, IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp.1103~1127, Nov. 2000.

[8] Kong H-S, Vetro A., Hata T. and Kuwahara N., “Fast Region-of-Interest Transcoding for JPEG2000 Images”, Mitsubishi Electric Reseerch Laboratories, Inc., Dec. 2005.

[9] M. Boliek and C. Christopoulos, “JPEG 2000 Part I Final Committee Draft Version 1.0”, ISO/IEC JTC 1/SC 29/WG 1 N1646R, March 2000.

[10] M. Boliek, E. Majani, J. S. Houchin, J. Kasner and M. L.

Carlander, "JPEG 2000 Part II Final Committee Draft", ISO/IEC JTC 1/SC 29/WG 1 N2000, Dec. 2000.
[11] 박순화 외 4명, "관심영역 코딩을 위한 기울기 정보기반의 빠른 마스크 생성 기법", 한국컴퓨터정보학회 논문지, 제14권, 제1호, pp.81-89, 2009.
[12] 박재홍 외 4명, "JPEG2000에서 ROI의 자동 추출과 우선적 처리", 한국컴퓨터정보학회 논문지, 제14권, 제6호, pp.127-136, 2008.
[13] P. G. Tahoees, J. R. Varela, M. J. Lado and M. Souto, "Image Compression : Maxshift ROI encoding options in JPEG2000", Computer Vision and Image Understanding, Vol.109, Iss. 2, pp.139-145, 2008.

박 재 흥
e-mail : pjh@gnu.ac.kr
1978년 충북대학교 수학교육과
1989년 중앙대학교 대학원 전산과 박사
1983년~현 재 경상대학교 컴퓨터과학과 교수
관심분야 : 소프트웨어 공학, 소프트웨어 신뢰성

이 점 숙
e-mail : alleya@gnu.ac.kr
1994년 경상대학교 컴퓨터과학과 학사
1999년 경상대학교 컴퓨터교육과 석사
2003년~현재 경상대학교 컴퓨터과학과 박사 과정
관심분야 : MPEG, JPEG2000, wavelet

서 영 건
e-mail : young@gnu.ac.kr
1987년 경상대학교 전산과 학사
1997년 숭실대학교 전산과 박사
1989년~1992년 삼보컴퓨터
1997년~현 재 경상대학교 컴퓨터교육과 교수, 컴정연구소원
2001년~현 재 경상대학교 컴정통신연구소원
관심분야 : 멀티미디어통신, 영상인식, 원격교육

홍 도 순
e-mail : hong-d-s@hanmail.net
1988년 경상대학교 전산과 학사
1993년 경상대학교 전산과 석사
1993년~현 재 충렬여자고등학교 교사
2006년~현 재 경상대학교 컴퓨터과학과 박사 과정 수료
관심분야 : MPEG, JPEG2000, wavelet

김 현 주
e-mail : hjkim@jinju,ac.kr
1988년 경상대학교 전산과 학사
1990년 숭실대학교 전산과 석사
1993년~1997년 제일정밀
2002년~현 재 진주산업대학교 컴퓨터공학과 교수
관심분야 : 정보검색, XML