(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
H03M 7/46 (2006.01)

(45) 공고일자 2007년07월02일
(11) 등록번호 10-0733949
(24) 등록일자 2007년06월25일

(21) 출원번호	10-2001-7012203	(65) 공개번호	10-2002-0008133
(22) 출원일자	2001년09월25일	(43) 공개일자	2002년01월29일
심사청구일자	2005년03월24일		
번역문 제출일자	2001년09월25일		
(86) 국제출원번호	PCT/US2000/007955	(87) 국제공개번호	WO 2000/59116
국제출원일자	2000년03월24일	국제공개일자	2000년10월05일

(81) 지정국

국내특허 : 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바베이도스, 불가리아, 브라질, 벨라루스, 캐나다, 스위스, 중국, 쿠바, 체코, 독일, 덴마크, 에스토니아, 스페인, 핀란드, 영국, 그루지야, 헝가리, 이스라엘, 아이슬랜드, 일본, 케냐, 키르키즈스탐, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 노르웨이, 뉴질랜드, 슬로베니아, 슬로바키아, 타지키스탄, 투르크맨, 터어키, 트리니아드토바고, 우크라이나, 우간다, 우즈베키스탄, 베트남, 폴란드, 포르투칼, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 아랍에미리트, 안티구와바부다, 코스타리카, 도미니카, 알제리, 모로코, 탄자니아, 남아프리카, 인도, 그라나다, 시에라리온, 세르비아 앤 몬테네그로, 크로아티아, 인도네시아, 가나, 감비아, 짐바브웨,

AP ARIPO특허 : 케냐, 레소토, 말라위, 수단, 스와질랜드, 우간다, 시에라리온, 가나, 감비아, 짐바브웨, 탄자니아,

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르키즈스탐, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크맨,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀란드, 사이프러스,

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기니, 말리, 모리타니, 니제르, 세네갈, 차드, 토고, 기니 비사우,

(30) 우선권주장

09/277,255	1999년03월26일	미국(US)
09/280,135	1999년03월26일	미국(US)
09/276,954	1999년03월26일	미국(US)

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이

(72) 발명자
맬바,헨리큐,에스.
미국98053워싱턴주레드몬드233알디애비뉴엔,이.2302

(74) 대리인
백만기
이중희
주성민

(56) 선행기술조사문헌
WO 9317524 A1
US 05818877 A
EP 0940994 A2
WO 9854093 A1
US 05717394 A

심사관 : 송병준

전체 청구항 수 : 총 17 항

(54) 유한 알파벳 데이터의 비손실 적응 인코딩

(57) 요약

인코더는 데이터-종속 데이터 구조들을 사용할 필요 없이 크고 작은 웨이브렛 계수들(wavelet coefficients)을 개별적인 그룹들로 클러스터하기 위해 양자화된 웨이브렛 계수들을 리오더한다. 계수들은 그 후 코드워드들에서 소비되는 비트들의 수를 최소화하면서, 양자화된 계수들의 스트링들을 나타내기 위해 코드워드 사용을 제어하는 파라미터를 계속해서 갱신하는 런 길이 코드(run-length code)를 기초로 하여 적응적으로(adaptively) 인코딩된다. 인덱스 매트릭스는 상부 좌측 코너에 최저(coarsest) 계수들을 포함하고, 훨씬 더 큰 블록들의 로우 하이 및 하이 로우 서브 밴드들을 교대로 채워서, 로우 하이 서브 밴드들이 매트릭스의 상부를 포함하고, 하이 로우 서브 밴드들이 매트릭스의 좌측을 포함하게 한다. 최단 코드워드들은 길이가 2^k인 가장 확률이 큰 문자(most likely character)의 런을 나타내도록 할당되는데, k는 파라미터이다. k는 발생된 연속 문자들을 기초로 하여 조정된다. k는 문자가 동일할 때 증가되고, 문자가 상이할 때 감소된다. 디코더는 상술된 바의 역순으로 작용한다.

대표도

도 5

특허청구의 범위

청구항 1.

적응 런-길이 인코더(adaptive run-length encoder)를 위하여 스트링의 길이를 초기화하는 단계 및

인코딩 파라미터에 기초하여 상기 스트링의 길이를 변경하는 단계를 포함하는

유한 알파벳 데이터 문자를 인코딩하는 방법으로서,

이미 인코딩된 문자를 근거로 상기 인코딩 파라미터를 변경하는 단계

를 포함하는 인코딩 방법.

청구항 2.

제1항에 있어서,

상기 인코딩 파라미터는 예상되는 문자가 발생할 때마다 증가되고 상기 예상되는 문자는 0인 인코딩 방법.

청구항 3.
삭제

청구항 4.

제1항에 있어서,

상기 인코딩 파라미터는 예상되는 문자가 발생하지 않을 때마다 감소되고 상기 예상되는 문자는 0인 인코딩 방법.

청구항 5.
삭제

청구항 6.

제1항에 있어서,

상기 문자들은 비트 평면 기반으로 인코딩되는 인코딩 방법.

청구항 7.

제1항에 있어서, 상기 인코딩 파라미터는 2의 거듭제곱(power of two)인 인코딩 방법.

청구항 8.
삭제

청구항 9.
삭제

청구항 10.
삭제

청구항 11.

컴퓨터가 제1항의 방법을 실행하도록 하는 명령어(instruction)들이 기록되어 있는 컴퓨터-판독 가능 매체.

청구항 12.
삭제

청구항 13.

제1항에 있어서,

스케일링 배율(scaling factor)이 상기 인코딩 파라미터에 적용되어, 상기 인코딩 파라미터를 변경하는 단계는 상기 이미 인코딩된 문자에 기초하여 상기 스케일링된 인코딩 파라미터(scaled encoding parameter)의 값을 조정하는 단계를 포함하며,

후속하여 만들어진 데이터 문자들(subsequently encountered data characters)의 인코딩은 상기 스케일링 배율로 나뉘어진 상기 스케일링된 인코딩 파라미터의 조정된 값과 같은 인코딩 파라미터를 사용하는 단계를 포함하는 인코딩 방법.

청구항 14.

제1항에 있어서,

상기 데이터는 웨이브렛 계수를 포함하는 인코딩 방법.

청구항 15.

제1항에 있어서,

상기 데이터는 멀티미디어 데이터와, ECG 데이터 및 수명 측정 유형(biometric type)의 데이터와 같은 물리적 데이터를 포함하는 인코딩 방법.

청구항 16.

제1항에 있어서,

상기 데이터는 팩스로 전송될 이미지의 스캔을 나타내는 인코딩 방법.

청구항 17.

삭제

청구항 18.

삭제

청구항 19.

삭제

청구항 20.

적응 런-길이 디코더를 위해 스트링 길이를 초기화하는 단계 및

디코딩 파라미터에 기초하여 상기 스트링 길이를 변경하는 단계를 포함하는,

역방향 적응 런 길이 인코딩(backward adaptive run length encoging)을 사용하여 인코딩된 유한 알파벳 데이터 문자들을 디코딩하는 방법으로서,

이미 디코딩된 문자들에 기초하여 상기 디코딩 파라미터를 변경하는 단계

를 포함하는 디코딩 방법.

청구항 21.

제20항에 있어서,

상기 디코딩 파라미터는 예상되는 문자가 발생할 때마다 증가되고 상기 예상되는 문자는 0인 디코딩 방법.

청구항 22.
삭제

청구항 23.

제20항에 있어서,

상기 디코딩 파라미터는 예상되는 문자가 발생하지 않을 때마다 감소되고 상기 예상되는 문자는 0인 디코딩 방법.

청구항 24.
삭제

청구항 25.

제20항에 있어서,

상기 문자들은 비트 평면 기반으로 디코딩되는 디코딩 방법.

청구항 26.

제20항에 있어서,

상기 디코딩 파라미터는 2의 거듭제곱(power of two)인 디코딩 방법.

청구항 27.

제20항에 있어서,

스케일링 배율(scaling factor)이 상기 디코딩 파라미터에 적용되어, 상기 디코딩 파라미터를 변경하는 단계는 상기 이미 디코딩된 문자에 기초하여 상기 스케일링된 디코딩 파라미터의 값을 조정하는 단계를 포함하고,

후속하여 만들어진 인코딩된 데이터 문자들(subsequently encountered encoded data characters)의 상기 디코딩은 상기 스케일링 배율로 나뉘어진 상기 스케일링된 디코딩 파라미터의 조정된 값과 같은 디코딩 파라미터를 사용하는 단계를 포함하는 디코딩 방법.

청구항 28.
삭제

청구항 29.

컴퓨터가 제20항의 방법을 실행하도록 하는 명령어(instruction)들이 기록되어 있는 컴퓨터-판독 가능 매체.

청구항 30.
삭제

명세서

기술분야

본 발명은 일반적으로 이미지 압축 분야에 관한 것으로 특히 디지털 화상들의 향상된 웨이브렛 인코딩 및 디코딩에 관한 것이다.

삭제

삭제

<저작권 통지/허용>

배경기술

디지털 화상(digital picture)는 웹 페이지, CD-ROM 백과 사전, 디지털 카메라 등과 같은 다양한 애플리케이션에서 사용된다. 대부분의 경우, 소량의 기억 장치에 적합하고 단 시간에 다운로드되기 위해 화상들을 압축할 필요가 있다. 예를 들어, 전형적인 디지털 카메라에서, 화상들은 픽셀 당 12 내지 24 비트의 해상도인 상태에서 1024 ×768 화소들(픽셀들)의 해상도를 갖는다. 각각의 이미지의 원 데이터(raw data)는 따라서 대략 1.2 내지 2.5 메가바이트이다. 예를 들어, 몇몇 화상들을 컴퓨터 디스켓에 삽입하기 위해, 각각의 화상에 의해 사용되는 데이터의 양을 감소시킬 필요가 있다. 달성되는 압축률이 클수록, 더 많은 화상들이 디스켓 또는 메모리 카드에 삽입되고 전화선과 같은 대역폭 제한 전송 매체를 통해 보다 신속하게 전달될 수 있다.

이미지 압축은 지난 20년간 광범위하게 연구되어 왔다. ISO(International Standards Organization)의 JPEG(joint photographic experts group) 위원회에 의해 정의된 JPEG 표준은 1992년에 정의된 것으로 가장 인기 있는 디지털 화상 압축 방법이다. JPEG에서, (8 ×8 크기의) 작은 정사각 픽셀 블록들이 DCT(discrete cosine transform)에 의해 주파수 영역으로 맵핑된다. DCT 계수들은 양자화(quantized)되고(스케일링 배율(scale factor)로 나누어져서 가장 가까운 정수로 반올림됨), 고정 지그재그 스캔 패턴(fixed zigzag scan pattern)을 통해 1차원 벡터로 맵핑된다. 이 벡터는 런 길이(run length)와 허프만 인코딩의 결합을 통해 인코드된다.

JPEG의 작은 8 ×8 블록들의 독립적인 프로세싱은 구현의 관점에서 볼 때 이점이 있는데, 특히 저-비용 하드웨어에 유리하다. 그러나, 블로킹 아티팩트(blocking artifacts)라는 JPEG에서의 주요한 문제점을 야기하기도 한다. 인접 블록들로부터의 양자화 오류들이 블록들 사이(among blocks)에서 상관되지 않고 블록들 내(within the blocks)에서 상관되기 때문에, 인코딩시 인접 블록들 간의 전위차로 인해 재생 이미지에서 8 ×8 블록들의 경계들이 보이게 된다. 이러한 아티팩트들은 타일링(tiling) 또는 블로킹 아티팩트들이라고 하는데, 이들은 오버랩핑 기반 함수들(overlapping basis functions)로 변환함으로써 감소될 수 있다(그러나 완전히 제거되지는 않는다).

블로킹 아티팩트를 제거하기 위한 효율적인 방법은 블록 DCT(block DCT)를 웨이브렛 분해(wavelet decompositoin)로 대체하는 것이다. 웨이브렛 분해는 효율적인 시간-주파수 표현(time-frequency representation)을 제공한다. 웨이브렛 계수 양자화 및 인코딩에 의해 매우 양호한 압축 성능이 얻어질 수 있다.

다수의 웨이브렛-기반 이미지 압축 시스템들이 지난 몇 년간 기술적인 문헌에 보고되어 왔다. 웨이브렛의 경우, JPEG 보다 더 양호한 압축률을 달성할 수 있는데, 통상 20% 내지 50% 정도 더 양호하다. 더 중요하게, 웨이브렛 변환은 JPEG에서 문제가 되는 블로킹 아티팩트들을 갖지 않는 화상를 생성한다. 따라서, 웨이브렛-기반 변환(wavelet-based transform)은 점점 더 인기를 얻고 있다. 실제로, JPEG의 다음 개정안인 JPEG2000에서 고려되고 있는 모든 제안은 웨이브렛을 사용한다.

몇몇 종래의 웨이브렛 변환은 이미지들을 16 서브밴드(subbands)에 대응하는 계수들로 분해한다. 이는 빅 블록 포맷(big block format)이라고 하는 4 ×4 서브밴드 매트릭스를 야기하는데, 이는 채널들의 오더링(ordering) 및 스펙트럼 분해(spectral decomposition)를 나타낸다. 글자 L과 H는 각각의 서브밴드에 대한 로우 패스 필터링(low pass filtering) 및 하이 패스 필터링(high pass filtering)을 식별하기 위해 각각 사용된다. 제1 서브밴드는 LL 및 HL 계수들을 포함하는데, 여기서 각 세트의 첫 번째 글자는 수평 필터링에 대응하고, 두 번째 글자는 수직 필터링에 대응한다. 각각의 서브밴드 필터링 결합에는 2개의 단계가 사용된다. 오더링은 좌측에서 우측으로 또한 하부에서 상부로 갈수록 증가하는 주파수에 대응한다. 상기 오더링은 고정되어서 인코딩 및 디코딩 모두가 고정된 방식으로 동작하게 한다. 계수들의 양자화가 실행되고, 그 후 계수들이 몇몇 형태로 압축 인코딩되는데, 여기에는 이미지를 더 압축하기 위한 적응 허프만 인코딩(adaptive Huffman encodding) 또는 산술 인코딩(arithmetic encoding)을 포함한다. 이러한 형태들의 인코딩은 데이터 유형에 좌우되는 제로 트리 구조들(zero tree structures)을 포함해서 꽤 복잡할 수 있다. 이러한 인코더들은 꽤 복잡한데, 다수의 인코더들은 상이한 이미지들을 압축시키기 위해 변경될 필요가 있다. 따라서 인코더들을 하드웨어로 구현하는 것은 어려운 일이다.
오더링 웨이브렛 계수(ordering wavelet coefficients)에 대한 제로 트리 기반 접근의 복잡도에 대한 하나의 해결책이 오덴트리히 이 외(Ordentlich E. et al)의 "웨이브렛 압축 계수의 임베디드 코딩을 위한 저 복잡도 모델링 접근(A low complexity modeling approach for embedded coding of wavelet compression coefficients)", 프로시딩 DCC '98 데이타 압축 컨퍼런스(CAT. No. 98TB100225), 프로시딩 DCC '98 데이타 압축 컨퍼런스, 스노우버드, UT, USA, 1998년 3월 30일-4월 1일, 제408쪽-제417쪽, XP0009526 1998 로스 알라미토스, CA, USA, IEEE Comput. Soc., USA ISBN 0-1886-8406-2에 개시되어 있다. 오덴트리히는 2단계 과정을 개시하였는데, 상기에서는 한 집합의 계수가 상기 집합 외부의 이전에 인코드된 정보의 함수로 분해되어 2 이상의 계수의 집합으로 된다. 제2 단계는 상기 2개 이상의 계수의 집합을 선택된 오더링에 으이해 제한되는 일상적인 컨텍스트 모델링을 사용하여 분해한다. 결과적인 문자들은 그 다음 적응 런 길이 롤롬 라이스 인코더(adaptive Run-Length Golomb-Rice encoder)를 사용하여 인코드된다.

따라서, 하드웨어 또는 소프트웨어로 구현될 수도 있는 유사한 유한 알파벳 데이터(finite alphabet data) 및 웨이브렛 압축 계수들(wavelet compression coefficient)에서 동작하는 간단한 인코딩 기술이 필요하다.

<요약>

큰 크기의 값보다 적은 크기의 값이 발생할 확률이 더 큰 부호화된 정수 데이터에 대한 적응 인코딩(adaptive encoding)이 실행된다. 인코딩은 비트 평면에서 실행되는데, 이는 재생 정밀도(precision of reconstruction)의 확장성(scalability)에 대해 비손실(오류 없음)부터 다양한 레벨의 근사 재생(approximatoin reconstruction)까지를 허용한다. 허프만 인코딩과 달리, 코드표가 필요하지 않은데, 그 이유는 간단한 규칙들이 입력 스트링으로부터 코드워드를 결정하기 때문이다.

한 형태로, 각각의 비트 평면에서, 최단 코드워드(싱글 0)가 할당되어 길이가 2^k인 가장 확률이 큰 입력, 0의 런(run)을 나타낸다. 여기서, k는 코드워드들에서 소비되는 비트들의 수를 최소화하면서, 양자화 계수들의 스트링들을 나타내는데 사용되는 코드워드들을 제어하는 파라미터이다. k는 보다 긴 런들이 발생할 때 증가되고 다른 경우 예를 들어 런에서 상이한 심볼들이 발생할 때 감소되도록 적용된다. 비트 평면의 인코딩은 적응 산술 인코더(adaptive arithmetic encoder)와 같은 임의의 효율적인 엔트로피 인코더로 실행될 수 있지만, 한 구현에서, 새로운 적응 런-길이 + 골롬-라이스 인코더(adaptive Run-Length plus Golomb-Rice encoder)가 사용된다.

제로트리와 같은 데이터-종속 데이터 구조들을 사용할 필요 없이, 또는 트리의 세트 파티션들을 위한 개별적인 리스트를 사용할 필요 없이, 하드웨어가 보다 쉽게 구현될 수 있고 소프트웨어 구현이 보다 신속하게 실행될 수도 있다.

발명의 상세한 설명

본 발명의 일례의 실시예들에 대한 이하의 상세한 설명에서, 본 발명의 일례의 실시예들의 일부를 형성하며, 본 발명이 실행될 수도 있는 특정 일례의 실시예들이 실례로서 도시된 추가된 도면들이 참조되었다. 실시예들은 충분히 상세하게 설명되었으므로 본 기술 분야에 숙련된 자들이 본 발명을 구현할 수 있으며, 다른 실시예들이 사용될 수도 있고 논리적, 기계적, 전기적 변형들 및 다른 변형들이 본 발명의 범위 내에서 이루어질 수 있음을 알 것이다. 따라서, 이하의 상세한 설명은 제한적인 의미가 아니고, 본 발명의 범위는 추가된 청구항들에 의해서만 정의된다.

상세한 설명은 다수의 섹션들로 나누어져 있다. 제1 섹션은 본 발명을 구현하는 컴퓨터 시스템의 동작을 설명한 것이다. 그 다음 양자화된 웨이브렛 계수들의 고정 리오더링(fixed reordering) 및 적응 런-길이 인코딩에 대한 하이 레벨 설명이

이어진다. 또한 인코딩된 데이터를 위한 디코더가 설명된다. 하이 레벨 설명에서 선택된 블록들에 대한 보다 상세한 설명이 흐름도를 사용해서 기술된다. 이어서 소프트웨어 애플리케이션 오피스 슈트(software application office suite)에서의 인코더 및 디코더 사용이 전반적으로 설명된다. 결론은 몇몇 잠정적인 장점들에 대해 기술하고 또한 다른 대안 실시예들에 대해 기술한다.

하드웨어 및 동작 환경(Hardware and Operating Environment)

도 1은 본 발명이 구현될 수 있는 적합한 컴퓨팅 환경을 간단하고 전체적으로 도시한 것이다. 본 발명은 개인용 컴퓨터(PC)에 의해 실행되는 명령들을 포함하는 컴퓨터 실행 가능 프로그램 모듈들의 일반적인 문맥으로 기술된다. 프로그램 모듈들은 특정 작업들을 실행하거나 특정한 추상적인 데이터 유형들을 구현하는 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조 등을 포함한다. 본 기술 분야에 숙련된 자들은 멀티미디어 기능을 갖는 핸드-헬드 장치들, 멀티프로세서 시스템, 마이크로프로세서-기반 프로그래밍 가능한 가전 제품, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터 등을 포함하는 다른 컴퓨터 시스템 구성들에 의해서도 본 발명이 실행될 수 있음을 알 것이다. 본 발명은 또한 통신망들을 통해 연결된 원격 처리 장치들에 의해서 태스크들이 실행되는 분산 컴퓨팅 환경에서도 실행될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈들은 로컬 및 원격 메모리 기억 장치들 모두에 위치할 수도 있다.

도 1은 프로세싱 유닛(21), 시스템 메모리(22), 및 시스템 메모리와 다른 시스템 컴포넌트들을 프로세싱 유닛(21)에 결합시키는 시스템 버스(23)를 포함하는 종래의 개인용 컴퓨터(20) 형태의 범용 컴퓨팅 장치를 도시한 것이다. 시스템 버스(23)는 메모리 버스 또는 메모리 컨트롤러, 주변 버스, 및 로컬 버스를 포함하는 다양한 유형들 중 임의의 유형일 수 있고, 다양한 버스 구조들 중 임의의 버스 구조를 사용할 수도 있다. 시스템 메모리(22)는 판독 전용 메모리(ROM)(24) 및 랜덤-액세스 메모리(RAM)(25)를 포함한다. ROM(24)에 기억된 기본 입출력 시스템(BIOS)(26)은 개인용 컴퓨터(20)의 컴포넌트들 간에 정보를 전달하는 기본 루틴들을 포함한다. 바이오스(BIOS)(26)는 또한 시스템을 위한 개시 루틴들을 포함한다. 개인용 컴퓨터(20)는 하드 디스크(도시되지 않음)에 대한 판독 및 기록을 위한 하드 디스크 드라이브(27), 제거 가능 자기 디스크(29)에 대한 판독 및 기록을 위한 자기 디스크 드라이브(28), 및 CD-ROM 또는 다른 광 매체와 같은 제거 가능 광 디스크(31)에 대한 판독 및 기록을 위한 광 디스크 드라이브(30)를 더 포함한다. 하드 디스크 드라이브(27), 자기 디스크 드라이브(28) 및 광 디스크 드라이브(30)는 각각 하드 디스크 드라이브 인터페이스(32), 자기 디스크 드라이브 인터페이스(33) 및 광-드라이브 인터페이스(34)에 의해 시스템 버스(23)에 접속된다. 드라이브 및 관련 컴퓨터 판독 가능 매체는 개인용 컴퓨터(20)를 위해 컴퓨터 판독 가능 명령들, 데이터 구조들, 프로그램 모듈들 및 다른 데이터의 비휘발성 기억 장치를 제공한다. 본 명세서에 기술된 일례의 환경이 하드 디스크, 제거 가능 자기 디스크(29) 및 제거 가능 광 디스크(31)를 사용하더라도, 본 기술 분야에 숙련된 자들은 컴퓨터에 의해 액세스될 수 있는 데이터를 기억할 수 있는 다른 유형들의 컴퓨터 판독 가능 매체가 일례의 운영 환경에서 사용될 수도 있음을 알 것이다. 상기 매체는 자기 카셋트, 플래시-메모리 카드, DVD(digital versatile disk), 베르눌리 카트리지(Bernoulli cartridge), RAM, ROM 등을 포함할 수도 있다.

프로그램 모듈들은 하드 디스크, 자기 디스크(29), 광 디스크(31), ROM(24) 및 RAM(25)에 기억될 수도 있다. 프로그램 모듈들은 운영 체제(35), 하나 이상의 응용 프로그램들(36), 다른 프로그램 모듈들(37) 및 프로그램 데이터(38)를 포함할 수도 있다. 사용자는 키보드(40) 및 포인팅 장치(42)와 같은 입력 장치들을 통해 개인용 컴퓨터(20)에 명령들 및 정보를 입력할 수도 있다. 다른 입력 장치들(도시되지 않음)은 마이크로폰, 죠이스틱, 게임 패드, 위성 디시(satellite dish), 스캐너 등을 포함할 수도 있다. 여타 입력 장치들은 종종 시스템 버스(23)에 결합된 직렬 포트 인터페이스(46)를 통해 프로세싱 유닛(21)에 접속되는데, 병렬 포트, 게임 포트, 또는 USB(universal serial bus)와 같은 도 1에 도시되지 않은 다른 인터페이스들을 통해 접속될 수도 있다. 모니터(47) 또는 다른 디스플레이 장치는 비디오 어댑터(48)와 같은 인터페이스를 통해 시스템 버스(23)에 접속된다. 모니터 외에, 개인용 컴퓨터들은 통상 스피커 및 프린터와 같은 다른 주변 출력 장치들(도시되지 않음)을 포함한다.

개인용 컴퓨터(20)는 원격 컴퓨터(49)와 같은 하나 이상의 원격 컴퓨터들에 접속된 논리 접속부들을 사용해서 네트워크 환경에서 동작할 수도 있다. 원격 컴퓨터(49)는 다른 개인용 컴퓨터, 서버, 라우터, 네트워크 PC, 피어 장치(peer device), 또는 다른 일반적인 네트워크 노드일 수도 있다. 원격 컴퓨터(49)는 통상 개인용 컴퓨터(20)와 관련해서 상술된 구성 요소 다수 또는 모두를 포함한다; 그러나, 도 1에는 기억 장치(50)만이 도시되어 있다. 도 1에 도시된 논리 접속부들은 근거리 통신망(LAN)(51) 및 광역 통신망(WAN)(52)을 포함한다. 이러한 네트워크 환경은 사무실에서 흔히 볼 수 있는 것으로, 기업간 컴퓨터 네트워크, 인트라넷 및 인터넷이다.

LAN 네트워크 환경에 배치될 때, PC(20)는 네트워크 인터페이스 또는 어댑터(53)를 통해 LAN(51)에 접속된다. 인터넷과 같은 WAN 네트워크 환경에서 사용될 때, PC(20)는 통상 네트워크(52)를 통한 통신을 설정하기 위해 모뎀(54) 또는 다른 수단을 포함한다. 모뎀(54)은 PC(20)의 내부에 있을 수도 있고 외부에 있을 수도 있으며, 직렬 포트 인터페이스(46)를 통

해 시스템 버스(23)에 접속된다. 네트워크 환경에서, PC(20) 내부에 상주하는 것으로 도시된 MS 워드를 포함해서 그와 같은 프로그램 모듈들 또는 그 일부분들은 원격 기억 장치(50)에 기억될 수도 있다. 물론, 도시된 네트워크 접속부들은 예시적인 것으로, 컴퓨터들 간의 통신 링크를 설정하기 위한 다른 수단이 사용될 수도 있다.

소프트웨어는 객체 지향 프로그래밍 방법들을 포함하는 다수의 상이한 방법들을 사용해서 설계될 수 있다. C++ 및 자바는 객체 지향 프로그래밍과 관련된 기능을 제공하는 일반적인 객체 지향 컴퓨터 프로그래밍 언어들의 2가지 일예들이다. 객체 지향 프로그래밍 방법들은 데이터 멤버들(변수들) 및 데이터에 대해 작용하는 멤버 함수들(메소드들(methods))을 클래스(class)라고 하는 단일 엔티티(entity)로 캡슐화하는 수단을 제공한다. 객체 지향 프로그래밍 방법들은 현존 클래스들을 근거로 새로운 클래스들을 생성하는 수단을 제공한다.

객체는 클래스의 인스턴스(instance)이다. 객체의 데이터 멤버들은 컴퓨터 메모리 내부에 기억된 속성들(attributes)이고, 메소드들은 이러한 데이터에 대해 작용하는 실행가능한 컴퓨터 코드이다. 이들은 함께 잠정적으로 다른 서비스들을 제공한다. 객체라는 개념이 본 발명에서 개발되는데, 본 발명의 특정 양상들은 한 실시예에서 객체들로 구현된다.

인터페이스는 명명 유닛(named unit)으로 조직화된 관련된 함수들의 집단이다. 각각의 인터페이스는 몇몇 식별자에 의해 고유하게 식별될 수도 있다. 인터페이스들은 인스턴시에이션(instantiation)을 갖지 않는다. 즉, 인터페이스는 인터페이스에 의해 지정되는 메소드들을 구현하기 위해 필요한 실행 가능 코드를 갖지 않는 정의 그 자체이다. 객체는 인터페이스에 의해 지정된 메소드들을 위해 실행 가능 코드를 제공함으로써 인터페이스를 지원할 수도 있다. 객체에 의해 제공되는 실행 가능 코드는 인터페이스에 의해 지정되는 정의들에 순응하여야 한다. 객체는 또한 추가의 메소드들을 제공할 수도 있다. 본 기술 분야에 숙련된 자들은 인터페이스들이 객체 지향 프로그래밍 환경에서의 사용에 있어서 또는 객체 지향 프로그래밍 환경에 의해 제한되지 않음을 알 것이다.

하이 레벨 인코더 및 디코더 설명(High Level Encoder and Decoder Description)

웨이브렛 변환 기반 이미지 픽셀 인코더(wavelet transform based image pixel encoder)의 간단한 블록도가 도 2에 도시되어 있고, 대응하는 디코더는 도 3에 도시되어 있다. 인코더 및 디코더는 각각의 입력 및 출력으로 이미지 픽셀 데이터를 사용하는 것으로 기술되었지만, 필요에 따라 다른 데이터가 변환될 수도 있다. 도시된 실시예에서, 이미지 픽셀 데이터는 웨이브렛 변환 블록(wavelet transform block)(210)으로 제공되고, 웨이브렛 변환 블록(210)은 공지된 방식으로 동작해서 웨이브렛 계수들을 양자화 블록(quantization block)(220)에 제공한다. 웨이브렛 계수들은, 배경 섹션에서 기술된 바와 같이, 큰 블록 포맷(big block format)이다. 양자화는, 임계값 T를 정의하는 양자화 단계(quantization step)에 의해 제어되는 균일 양자화기(uniform quantizer)에 의해 실행된다. 따라서, 단계들 사이에 속하는 각각의 계수는 단계의 중간 값으로 표현된다. T가 작을 수록, 양자화시 손실이 적게 발생된다. 따라서, 블록(220)의 출력은 양자화된 웨이브렛 계수인 일련의 정수이다. 다수의 다른 애플리케이션에서와 같이, 양자화기는 일반적인 반올림(normal rounding)을 근거로 하거나 또는 반내림(rounding towards zero)을 근거로 할 수도 있다("불감대(dead zone)"를 갖는 양자화기로 알려져 있음).

리오더링 및 블로킹 함수 또는 블록(230)은 웨이브렛 계수들을 유사한 값들의 클러스터들로 그룹화한다. 이는 제로일 확률이 가장 큰 빈도수(frequency) 계수들의 블록들을 클러스터링 또는 그룹화시키게 된다. 데이터가 단조 감소(monotonically decaying)하는 진폭 분포를 갖는 경향이 있다는 점에서, 리오더링은 유사한 데이터의 그룹화의 확률을 증가시킨다. 제1 블록들은 보다 큰 진폭의 데이터를 갖는 경향이 있는 반면, 다음 블록들에서는 웨이브렛 계수들의 진폭들이 감쇠하는 경향이 있다. 스캐닝 순서를 고정함으로써 그룹화가 실행되는데, 이는 데이터 독립적이다. 이러한 그룹화의 일세트가 도 4에 도시되어 있는데, 일례로 웨이브렛 계수 의 64개 블록들을 갖는다. 도 4에서, 낮은 빈도수(frequency) 성분들은 그룹의 상부 좌측 코너를 향하여 배치되는데, 계수 블록들은 각각의 레벨에서 로우-하이 서브밴드 및 하이-로우 서브밴드로부터 교체된다. 리오더링 및 블로킹 블록(230)은 표시된 스캐닝 순서로 매크로블록 시퀀스를 제공한다. 제1 블록, 0은 웨이브렛 트리의 레벨 0의 모든 계수들을 포함한다. 이는 최저 해상도(coarsest resolution)에 대응한다. 블록들 0 내지 3은 레벨 1의 모든 계수들을 포함한다. 블록들 0 내지 15는 레벨 2의 모든 계수들을 포함한다. 레벨 3은 블록들 0 내지 63을 포함한다. 블록들은 각각의 레벨에서 로우-하이 및 하이-로우 서브밴드들로부터 교체되고, 로우-하이는 시퀀스의 상부라는 점에 주목하라. 이하에 기술되는 수학적 기술 섹션(Mathematical Description section)에서, 그러한 특정 오더링의 장점들을 기술한다. 본 기술 분야에 숙련된 자들이 알 수 있는 바와 같이, 다른 오더링들도 가능하지만, 상기 오더링이 다른 오더링 보다 양호하게 작용하는 것이 명백하다. 그 후 비트들은 최상위 비트로부터 시작해서 순차적으로 인코딩된다.

적응 인코딩 블록(240)은 매크로블록들을 수신하고 비손실 방식으로 인코딩한다. 블록들의 클러스터링은 큰 제로의 클러스터들을 갖는 압축 데이터를 제공한다. 비트 평면을 기반으로 인코딩함으로써 데이터를 더 리오더링하면, 보다 긴 제로

스트링들을 찾을 확률이 커진다. 제1 비트 평면의 최상위 비트로부터 시작해서 긴 제로 스트링의 확률을 보다 크게 한다. 또한, 이는 가장 적합한 데이터가 먼저 인코딩되도록 보장한다. 제3 또는 제4 비트 평면들이 인코딩될 때, 홀수들은 1에 비해 0에 대해 대략 동일하고, 스트레이트 이진 인코딩(straignt binary encoding)이 효율적으로 사용될 수도 있다.

인코더는 적응 런-길이 변형을 갖는 골롬-라이스 인코더를 개조한 것이다. 간단히 말해서, 2^k 제로 스트링이 제로와 동일한 단일 비트로 구성된 코드워드로 표현된다. 제로 코드워드로 표현되는 제로 스트링의 길이는 파라미터 k에 의해 제어되고, 파라미터 k는 관측된 제로 빈도수를 근거로 데이터가 만들어질 때마다 변한다. 제로 값이 인코딩되는 경우, 제로들이 발생할 확률이 보다 큰 것으로 가정되고, 따라서 파라미터 k의 값은 증가된다. 제로가 아닌 값이 만들어지는 경우, k는 감소된다. 증가량 및 감소량을 적절히 제어함으로써, 인코더는 실제로 그 확률을 추정하는 오버헤드 없이도 제로의 변화 확률을 갖는 비트들의 스트링을 잘 추적할 수 있다. 피드백 루프(245)는 인코더(240)의 역방향 적응성(backwards adaptive nature)을 나타내기 위해 사용된다. 이러한 인코딩은 효율적인 압축을 제공하고 입력되는 데이터의 통계치 변경에 대한 신속한 적응성을 제공한다. 인코더(240)는 가장 적절한 정보가 비트 스트림의 초반부에 제공된다는 점에서 효율적으로 점진적인 비트 스트림 출력을 제공한다. 최하위 비트들이 최종 비트 평면에서 인코딩되기 때문에, 보다 낮은 해상도 비트 스트림들의 경우, 해상도 충실도 블록(resolution fidelity block)(250)으로 표시된 것과 같이 실제로 포기(discard)되거나 인코딩되지 않을 수도 있다. 이는 데이터의 저 대역폭 전송에 유용하다.

도 3의 블록 형태로 도시된 바와 같이, 디코딩은 본래 인코딩 및 데이터 변환의 반대이다. 도 2의 인코더에 의해 생성되는 것과 같은 인코딩된 데이터의 비트 스트림은 비손실 적응 디코딩 블록(310)에서 수신된다. 비트 스트림은 디코더로부터, 로컬 기억 장치로부터, 또는 원격 디코더 또는 기억 장치로부터 위성 전송, 케이블 전송 또는 다른 네트워크와 같은 다수의 현존하는 전송 매체들 중의 하나를 통하여 직접 수신될 수도 있다. 디코딩 블록(310)은 순방향 이송 라인(feed forward line)(315)을 통해 인코딩 중에 개발된 규칙들을 수신한다. 블록(310)은 본래 사용될 스트링 길이를 수신하고, 규칙들에 따라 데이터를 재생한다. 다시 말해서, 블록 레벨에서 동작하지만, 본 발명의 요구 사항은 아니다. 이것은 단지 대량의 메모리를 요구하는 이미지 또는 다른 데이터를 전부 표시하여 동시에 작업하는 것보다는, 또는 이러한 메모리를 사용할 수 없는 경우 페이징하는 것보다는 보다 편리하게 해 준다. 비트 평면의 최종 비트를 디코딩하지 않음으로써 한 형태의 충실도 감소가 블록(310)에서 수행될 수도 있다. 이는 실제로 파라미터 T에 의해 제어되는 단계 사이즈를 효과적으로 배가시킨다. 이는 데이터의 충실도를 감소시키는 간단한 방법이다.

블록(310)으로부터 출력된 데이터는 블록(230)으로부터 출력된 정수 데이터와 동일하여야 한다. 그러나, 실제로 고 주파수 웨이브렛 계수들을 사용하지 않고 블록(320)에서 지시된 때에 블록(320)의 이미지의 고해상도 층들이 제거될 수도 있다. 이는 이미지 또는 이미지 집합을 디스플레이하는데 사용되는 윈도우(window)가 작은 경우에 유용하다. 블록(330)은 그 후 블록들을 다시 고유 위치들로 언셔플(unshuffle)하거나 리오더하는데 사용된다. 리오더 블록(330)의 출력은 수신된 비트 스트림의 헤더에 의해 제공되는 스텝 사이즈(step size)를 사용해서 블록(340)에서 재승산(remultiplied)될 필요가 있는 정수들이다. 이는 재생 웨이브렛 계수들을 제공한다. 헤더는 또한 이미지 사이즈가 얼마나 큰지에 대한 정보 및 다른 표준 이미지 포맷 데이터를 제공한다. 블록(350)에서 공지된 방식으로 역 웨이브렛 변환이 실행된다. 선택된 소정의 충실도 또는 해상도 감소 외의 손실만이 양자화 단계에서 발생되는데, 이는 T 파라미터의 변경에 의해 제어될 수 있음에 주목하여야 한다.

해상도 감소 옵션 블록(320)은 소수의 다른 방법들로 동작할 수 있다. 데이터를 제거하는 한 가지 방법은 관련된 정수들을 제로화하는 것이다. 해상도를 감소시키는 다른 방법은 소정의 지점에서 값들을 제로화하도록 명령받을 수 있는 언셔플 블록(330)의 동작을 변경하는 것이다. 언셔플 블록(330) 및 제로들이 시작되는 역 웨이브렛 변환 블록(350) 양자를 구별함으로써, 이러한 블록은 상기 지점들에서 실제 데이터의 불필요한 프로세싱을 제거하도록 용이하게 수정될 수 있다.

본 발명의 적응 인코딩 및 디코딩은 변하는 통계치를 갖는 클러스터링된 제로들을 포함하는 데이터에 대해 매우 잘 동작한다. 이러한 유형의 데이터는 또한 제로들의 어느 한 측에서 확률의 근접한 지수함수적인 감소를 갖는 높은 확률의 데이터를 갖는 것으로 특징화될 수도 있다. 정적 이미지 데이터 및 비디오와 같은 멀티미디어 데이터는 이러한 특징을 갖는다. 더 나아가, 다수의 유형의 물리적 데이터의 변환 또한 이러한 유형의 특징을 갖는다. 물리적 데이터를 캡쳐(capture)할 때, 정보는 통상 몇몇 장소에서 발생하는데, 이는 다른 데이터의 대부분이 제로임을 의미한다. 이러한 데이터의 대칭성은 이러한 유형의 인코딩이 가장 잘 동작하게 하는 바람직한 특성이기도 하다. 다시 말해서, 정보 스파이크의 어느 한 측에서 음(negative) 및 양(positive) 값 양자의 지수적인 감소는 유익한 것이다. 이러한 물리적 데이터의 일례들은 ECG 및 기타 수명 측정 유형(biometric type)의 데이터를 포함한다.

인코딩의 수학적 설명

도 2 및 도 3과 관련해서 상술된 변환 및 인코딩 및 디코딩을 수학적으로 설명한다. 이하의 단계들은 인코딩 알고리즘을 정의한다:

1. 이미지 어레이 x(m,n), m = 0,1,...M-1, n = 0,1,..., N-1이 주어지는 경우, 웨이브렛 변환 계수들 X(r,s)를 계산하라, r = 0,1,..., M-1, s = 0,1,..., N-1 이다.

2. 각각의 계수 X(r,s)는

수학식 1

$$q(r,s) = \mathrm{sgn}(X(r,s))\left\lfloor |X(r,s)|/T \right\rfloor$$

에 따라 양자화된다. 여기서, sgn(·)은 일반적인 시그넘 함수(signum function)이고 T는 양자화 임계값이다. 본 단계는 연속 웨이브렛 계수들 X(r,s)을 정수 시퀀스 q(r,s)로 맵핑한다. 본 단계는 정보 손실을 야기하는 유일한 단계이다.

3. 양자화된 계수들은 리오더되고 이하의 수학식에 따라 블록들로 그룹화된다;

l = 0,1,...,L-1이고 k = 0,1,...,K-1인 동안,

수학식 2

$$u_k(l) = q\left(r_k + \mathrm{mod}(l, M_B), s_k + \lfloor l/M_B \rfloor\right)$$

삭제

이다. 여기서, $L = M_B N_B$ 는 블록 사이즈이고, $K = MN/L$은 블록들의 총 수이고, M_B 및 N_B는 $M_B = M/2^J$ 및 $N_B = N/2^J$로 정의된다. 파라미터 J는 $u_k(l)$로 그룹화되는 양자화된 계수들의 사각 블록들의 사이즈를 제어해서, 블록 사이즈를 제어한다.

각각의 k에 있어서, 상부 좌측 코너 인덱스들 (r_k, s_k)은 상술된 스캔 순서에 따라 정의된다.

4. 블록들은 $U_i = \{u_k(l)\}$ 형태로, 고정된 크기 LK_B의 매크로블록들 U_i로 그룹화되는데, 여기서 $k = iK_B, iK_B + 1, \ldots + iK_B + K_B - 1$ 이다. 각각의 매크로블록에 있어서, 비트 평면들은 적응 런-길이/라이스(RLR) 코더에 따라 연속해서 양자화된다. 실제 RLR 출력 비트들에 뒤이은 U_i를 위하여 RLR 코드에 의해 사용되는 비트들의 수의 바이너리 인코딩이 출력 비트스트림에 추가된다.

이하의 단계들은 PWC 비트스트림을 디코딩하는데 사용된다:

1. $i = 0,1,\ldots,I_{max}-1$인 동안, 매크로블록들 U_i의 RLR-코드화 비트들을 디코딩하라. $I_{max} < K$이면, 웨이브렛 계수들의 저해상도 버젼(lower resolution version)이 복구된다. 소정의 재생 정확성(desired reconstruction accuracy)이 주어지는 경우, 각각의 매크로블록 내에서 단지 처음 소수의 비트 평면들만이 디코딩됨에 주목한다. 디코딩되지 않도록 선택된 q(r,s)의 비트 평면들의 모든 비트들은 제로로 설정된다. $I_{max} < K$를 선택함으로써 해상도 확장성(resolution scalability)을 얻을 수 있는 반면, 충실도 확작성은 각각의 매크로블록의 비트 평면들의 부집합만을 디코딩함으로써 얻어질 수 있다.

2. q(r,s)를 복구한 후에, 웨이브렛 계수들은

수학식 3

$$\hat{X}(r,s) = \begin{cases} 0, & q(r,s) = 0 \\ T[q(r,s) + 1/2], & q(r,s) > 0 \\ T[q(r,s) - 1/2], & q(r,s) < 0 \end{cases}$$

에 의해 재생된다. 수학식 3의 재생 규칙과 결합된 수학식 2의 양자화 규칙은 라플라스 (양측 지수적(double-sided exponential)) 확률 분포를 갖는 랜덤 변수들의 최소-엔트로피 스칼라 양자화의 최적에 가까운, 원점 주위에 불감대를 갖는 균일 양자화기를 포함하는 것에 주목하여야 한다.

PWC 인코더의 단계 3에서 기술된 바와 같이, 웨이브렛 계수들을 리오더하기 위하여 상부 좌측 코너 인덱스들의 시퀀스 (r_k, s_k)가 정의된다. 도 4에 도시된 스캐닝 순서가 사용되는데, $M_B = M/2^J$ 및 $N_B = N/2^J$는 각각의 블록의 사이즈를 제어한다. 파라미터 J는 블록 0이 최저 해상도의 모든 웨이브렛 계수들, 예를 들면, 모든 스케일링 함수 계수들을 정확하게 포함하도록 선택되어야 한다. 따라서, J는 웨이브렛 변환에서 사용되는 해상도 레벨들의 수(트리 깊이; tree depth)와 동일하여야 한다. 도 4로부터 모든 상부 좌측 코너 인덱스들의 시퀀스 (r_k, s_k)를 추론하는 것은 쉽다.

임의의 소정 레벨 해상도의 계수들의 완전한 세트를 디코딩하기 위해, 인덱스 0 내지 $K_{max}-1$의 모든 블록들을 사용하는 것이 바람직하다는 점이 도 4로부터 명백하다. 여기서, K_{max}는 4의 거듭제곱(power)이다. 따라서, PWC 디코더의 단계 1에서, $I_{max}-1$은 K_{max}가 4의 거듭제곱이 되도록 선택된다.

동일한 해상도 레벨 내에서의 로우-하이(LH) 및 하이-로우(HL) 웨이브렛 계수들을 다르게 스캐닝하는 이유는 단순하다. 원본 이미지가 소정의 공간 위치에서 특정 특성을 가지면(또는 특성을 갖지 않으면), 상기 위치에 대응하는 LH 및 HL 서브밴드들의 클러스터들은 큰(또는 작은) 값들을 갖게 된다. 따라서, 동일한 공간 위치에 대응하는 LH 및 HL 서브밴드들로부터의 블록들의 쌍들이 매크로블록에서 인접해서 나타나거나 또는 적어도 서로 근접하거나 가까이 있음을 보장함으로써, 보다 쉽게 크고 작은 값들의 클러스터들을 생성할 수 있다. 이는 양자화된 계수들의 비트 평면들의 긴 0들의 런들의 확률을 증가시킨다.

도 7의 흐름도는 도 4에 도시된 순서로 계수 블록들을 기록하는데 사용되는 알고리즘을 도시한 것이다. 알고리즘은 희망대로 컴퓨터 프로그램 명령들 또는 하드웨어, 펌웨어 또는 소정의 상기 모든 수단의 결합을 통하여 구현될 수 있다. 알고리즘은 시작 블록(710)에서 개시된다. M ×N 양자화된 웨이브렛 계수들을 포함하는 입력 매트릭스 Q는 블록(715)에서 판독된다. 계수들은 양자화 블록(220)에 의해 제공되는 것과 같은 계수들이다. 웨이브렛 레벨들의 수, JW로 공지된 방식으로 블록(720)에서 정의된다. 블록(725)에서, 블록 사이즈가 NH ×NV로 정의되는데, NH는 $M/(2^{JW})$와 동일하고 NV는 $N/(2^{JW})$와 동일하다. 제1 출력 블록은 그 후 블록(730)에서 기록되고, IH 및 IV는 사이즈가 보다 큰 다른 블록들의 기록을 위한 루프들을 정의하는데 사용되기 위하여, 각각 NH 및 NV로 초기화된다. 간단한 예를 들면, 도 4에서 매트릭스 Q는 16 ×16이고 4 레벨들을 가지며 블록 사이즈가 1이라고 가정하자. 초기 IH 및 IV는 1이 된다. 다른 일례들에서, 블록 사이즈는 8 ×8 또는 16 ×16과 같이 보다 크거나 훨씬 높게 된다.

결정 블록(740)은 IH가 M 보다 작은지를 검사함으로써 계수들의 전체 매트릭스가 기록되었는지를 결정하는데 사용된다. IH가 M 보다 여전히 작으면, 보다 많은 계수들이 기록될 필요가 있다. 도 4에 도시된 바와 같이, 계수들의 제1 블록들은 1 ×1 크기이고, 그 후 2 ×2, 4 ×4 등으로 증가된다. 다음 세트의 흐름도 블록들은 1로부터 블록(745)에서 IH/NH로 설정된 블록 사이즈 파라미터 NBLK로 루프함으로써 연속 블록들을 기록하는데 사용된다. 블록(750)에서 정의된, I를 사용하는 내포된(nested) 루프와, 블록(755)에서 정의된, J를 사용하는 내포된 루프는 블록(760)에서 출력 블록들 LH 및 HL의 기록 순서를 제어하는데 사용된다. J는 NEXT 명령문(statement)(762)에서 증가되고, I는 NEXT 명령문(764)에서 증가된다. 그 결과, 이러한 특정의 구현에서는 블록들의 행(row)들이 먼저 기록된다. 원한다면 열들이 먼저 기록될 수도 있고, 또는 임의의 다른 기록 순서가 사용될 수도 있다. 처음에는 이러한 루프를 통하여, 16 ×16 크기의 매트릭스 및 4 레벨들이 주어지면, NBLK는 1이 되고, 이에 따라 블록들(430 및 440) 만이 기록된다.

다음 LH 및 HL 블록들의 기록에 이어, (780)에서 출력 블록을 기록하는 위치들을 정의하기 위하여 I 및 J를 다시 사용하여, (770) 및 (775)에서 제2 세트의 내포된 루프들이 설정(set up)된다. 출력 블록은 동일한 레벨의 HH 블록들에 대응하는데, 처음에 거치는 것은 블록(450)이다. NEXT J 및 NEXT I 명령어들은 각각 (782) 및 (784)에서 내포된 루프를 완료한다. HH 블록은 내포된 루프들이 동일하기 때문에 상기 LH 및 HL 블록들로서 동시에 기록될 수 있었음에 주목하여야 한다. 이 레벨의 모든 블록들이 기록된 후에, IH 및 IV는 (790)에서 2 지수로 증가되고, (740)에서 IH가 M 보다 여전히 작은지를 알기 위해 비교된다. IH가 M 보다 작지 않으면, 본 발명에 따라 완전한 리오더 세트의 웨이브렛 계수들이 제공된 후에 알고리즘은 (795)에서 종료된다.

2번째 내포된 루프들을 통해 블록들(455, 460, 470)이 기록되고, 3번째 내포된 루프들을 통해 블록들(480, 475, 490)이 기록된다. 보다 높은 레벨들을 갖는 보다 큰 매트릭스 사이즈들 또한 고려된다.

디코딩을 목적으로 초기 순서를 복구하기 위해, 간단히 기록된 방식과 동일한 방식으로 리오더링 알고리즘의 출력을 판독할 수 있다. 필요한 것은 초기 매트릭스의 크기 및 기록된 레벨들의 수에 관한 정보이다. 기록 순서는 단지 계수들을 제공하는 초기 순서와 반대이다. 직접 맵핑이 또한 사용될 수도 있지만, 이는 상당한 추가 대역폭을 필요로 한다.

비트 평면 인코딩의 상세한 설명

인코딩 블록(240)에 의해 실행되는 프로세스는 표 1을 참조하여 쉽게 이해될 수 있다. 비트 평면들은 입력 양자화 웨이브렛 계수들 또는 다른 데이터의 이진(bianry) 표현(크기 + 부호)의 특정 인덱스의 비트들의 시퀀스들이다. 예를 들어 표 1은 값들의 시퀀스 {9, -6, 1, 0, -2, 3, -4, -1, 2}의 비트 평면들을 도시한 것이다. 표에서, 비트 평면 4는 시퀀스 {100000000} 이고, 비트 평면 3은 시퀀스 {010000100} 이고, 비트 평면 2는 시퀀스 {010011001}이고, 비트 평면 1은 시퀀스 {101001010} 이다.

[표 1]

정수 데이터의 비트 평면 분해

데이터 값 →	9	-6	1	0	-2	3	-4	-1	2
부호 비트 →	0	1	0	0	1	0	1	1	0
비트 평면 4 →	1	0	0	0	0	0	0	0	0
비트 평면 3 →	0	1	0	0	0	0	1	0	0
비트 평면 2 →	0	1	0	0	1	1	0	0	1
비트 평면 1 →	1	0	1	0	0	1	0	1	0

표 1의 입력 데이터에서, 보다 작은 크기의 값들이 발생할 확률이 보다 큰 것으로 보이는데, 이는 통상 양자화된 웨이브렛 데이터 및 유한 알파벳 데이터이다. 보다 큰 크기의 입력 값들이 발생할 확률이 적기 때문에, 보다 큰 비트 평면들이 높은 빈도수의 제로들을 도시하는 경향이 있음을 상기 패턴으로부터 알 수 있다. 비트 평면 1(최하위 비트) 및 부호 비트 평면은 통상 거의 동일한 빈도수의 0들 및 1들을 갖는다.

도 5의 흐름도는 단계(505)에서 시작해서 비트 평면들을 통해 입력 데이터를 효율적으로 인코딩하기 위한 알고리즘을 도시한 것이다. 비트 평면들은 먼저 단계(510)에서 N개의 숫자들을 포함하는 입력 버퍼 x로부터 판독된다. 비트 평면들의 수, bamx는 단계(515)에서 계산되고, 유효 플래그 벡터 sflg는 단계(520)에서 모두 제로들로 설정된다.

단계(525)에서, 비트 평면 인덱스 변수 bit는 bmax와 동일하게 설정되어서, 인코딩은 최상위 비트 평면으로부터 시작한다. 인덱스 "bit"에 의해 지시되는 비트들의 값들은 단계(530)에서 비트 평면 벡터 bp를 형성한다. 각각의 평면 bp에 있어서, 비트들은 블록들(535, 540)에 표시된 바와 같이 2개의 부집합들로 분할된다. x1은 보다 높은 평면들에서 "1" 엔트리가 보이지 않는 위치들에 대응하는데, 이를 유효 비트들(significant bits)이라고 한다. x2는 보다 높은 평면들에서 "1"이 거의 이미 보여지는 위치들에 대응하는데, 극치 비트들(refinement bits)이라고 한다.

블록(545)에서, x1은 적응 런-길이 골롬-라이스(ARLGR) 인코더로 인코딩되는데, 이는 x1의 높은 빈도수의 0들에 유익하다. x1에서 1과 동일한 모든 비트의 경우, 부호 비트도 또한 인코딩되고 출력 코드의 끝에 추가된다.

블록(550)에서, x2는 스트레이트(straight) 이진 인코딩으로 인코딩된다. 이는 출력 스트림에 x2 비트들을 추가함으로써 실행된다. 0들 및 1들이 x2에서는 통상 거의 동일하게 발생하기 때문에, 인코딩 효율성 면에서 볼 때 최소 손실이 야기된다.

부호 비트들은 비트 평면으로서 처리되지 않기 때문에, 부호 비트들은 비트 평면으로 처리되지 않음을 주목한다. 각각의 비트 평면의 x1 벡터들을 코드화하는 프로세스에서 부호 비트들이 송신된다. 따라서, 알파벳 {0, + 1, -1}, 즉, 비트 플러스 부호로부터 도출된 것과 같은 벡터 x1을 생각할 수 있다.

도 5의 흐름도의 중요한 특성은 x1에 속한 비트들 및 x2에 속한 비트들에 대한 정보가 명시적으로 인코딩될 필요가 없다는 점이다. 벡터 sflg는 x1로의 비트 할당을 제어하고, sflg는 먼저 모두 0들로 초기화된 후, 단계(555)에서 각각의 비트 평면이 인코딩된 후에 갱신된다. 따라서, 디코더는 sflg의 변경을 쉽게 추적할 수 있다. 다음 비트 평면에 이어, 단계(560)에서 bit가 감소되고 단계(565)에서 최종 평면이 디코딩되었는지를 알기 위해 검사된다. 그렇지 않으면, 제어는 다음 비트 평면의 인코딩을 위해 블록(530)으로 진행한다. 저 해상도 코딩이 요구되는 경우 비트가 0과 동일하거나 크면, 단계(570)에서 모든 x1 및 x2 인코딩들의 출력들을 포함하는 출력 버퍼가 기록되고, 단계(575)에서 프로세스가 종료한다.

적응 런-길이 + 골롬-라이스(ARLGR) 코더는 인코딩 게인이 존재하는 곳이다. 많은 0들을 갖는 긴 벡터들 x1을, 보다 적은 0들을 갖는 보다 압축된 코드로 맵핑한다. ARLGR 인코더는 이하에 도시된 바와 같이 관련 부호 비트들을 갖거나 갖지 않은 이진 시퀀스들을 인코딩하는데 사용될 수 있다. ARLGR 인코더를 이해하기 위해서는, 먼저 런-길이 인코딩 및 골롬-라이스 코딩의 기본을 고려한다.

일반적인 형태에서, 런-길이(RL) 코딩 배후의 기본 아이디어는 입력 데이터 벡터의 동일한 값의 긴 스트링들을, 반복될 값 및 값들이 반복되어야 하는 횟수를 규정하는 코드로 대체하는 것이다. 상기 반복 스트링들이 충분히 길고 충분한 빈도수를 갖는 경우, RL 코딩은 데이터 벡터를 표현하는데 필요한 비트들의 수를 상당히 감소시킬 수 있다.

RL 코딩은 0 또는 1이 발생할 확률이 상당히 큰 이진 데이터의 인코딩에도 적용될 수 있다. 한 일례는 그래픽스 파일들인데, 예를 들어, 디지털화된 블랙을 화이트 배경에 그려넣는 것이다. 화이트 화소들(픽셀들)이 0과 동일한 비트로 표현되고 블랙 도트들이 1과 동일한 비트로 표현되는 경우, 0들이 발생할 확률이 훨씬 더 크다는 점은 명백하다. 실제로, 다수의 표준 그래픽스 파일 포맷들은 RL 코딩을 사용한다.

1996년에, 골롬(Golomb)은 양수(positive number) 표현을 위한 간단한 코드를 제안하였다. 숫자들이 기하학적 확률 분포를 갖는 소스로부터 도출되는 경우, 즉, Prob{x=n}=ab^n인 경우 - 여기서, a 및 b는 파라미터들이다 -, 골롬 코드는 실제로 최적(최소 예상 길이)임이 후에 알려졌다. 몇 년 후, 라이스(Rice)는 실제로 구현하기 매우 쉬운 골롬 코드의 부집합을 독립적으로 유도해냈다. 이 코드들은 골롬-라이스 코드들로 공지되어 있다.

본 발명에서, 이진 숫자(digit)들의 소스를 위한 골롬-라이스 코드들은 RL 코드들과 결합된다. 결과 런-길이 = 골롬-라이스 코드가 표 2에 도시되어 있다. 코드는 파라미터 k에 의해 특징화된다. 파라미터 k는 코드워드 0과 관련된 런의 길이를 제어하는데; 최대 런 길이는 2^k과 동일하다.

[표 2]

심볼들 ∈ {0,1}을 생성하는 소스의 런-길이 + 골롬-라이스 인코딩

K	입력 스트링	출력 이진 코드
0	0	0
	1	1
1	00	0
	1	10
	01	11
2	0000	0
	1	100
	01	101
	001	110
	0001	111

3	00000000	0
	1	10000
	01	10010
	001	10011
	0001	10101
	00001	10111
	000001	11000
	0000001	11010
	00000001	11100

이전에 논의된 비트 평면 인코더의 x1 벡터의 인코딩을 위해, 각각의 논제로(nonzero) 비트의 코드워드에 부호를 추가할 필요가 있다. 이를 위해, 표 3에 도시된 바와 같이 RLGR 코드의 간단한 확장이 사용된다.

[표 3]

심볼들 ∈ {0, + 1, -1}을 생성하는 소스의 런-길이 + 골롬-라이스 인코딩

K	입력 스트링	출력 이진 코드
0	0	0
	+1	10
	−1	11
1	00	0
	+1	100
	−1	101
	0+1	110
	0−1	111
2	0000	0
	+1	1000
	−1	1001
	0+1	1010
	0−1	1011
	00+1	1100
	00−1	1101
	000+1	1110
	000−1	1111

3	00000000	0
	+1	10000
	-1	10001
	0+1	10010
	0-1	10011
	00+1	10100
	00-1	10101
	000+1	10110
	000-1	10111
	0000+1	11000
	0000-1	11001
	00000+1	11010
	00000-1	11011
	000000+1	11100
	000000-1	11101
	0000000+1	11110
	0000000-1	11111

입력 벡터들의 소정의 소스의 경우, {0,1} 또는 {0, + 1, -1} 알파벳들을 사용해서, 파라미터 k는 예상 코드 길이를 최소화하도록 선택되어야만 한다. (610)에서 제1 심볼로부터 시작되어, xindex = 1 이고, symbol은 x(xindex)로 설정되고, runmax는 2^k으로 설정된다.

인코딩 프로세스의 개요로서, 소스 심볼을 인코딩한 후에, kp는 방출된 출력 코드를 근거로 하여 조정된다. 출력 코드가 0이었고, k ≠1이면, kp는 선정된(predefined) 증가 단계 **Up**만큼 증가된다. 즉, **kp = kp + Up**로 설정된다. 출력 코드가 0이었고, k = 0이면, kp는 선정된 증가 단계 **Uq**만큼 증가된다. 즉, **kp = kp + Uq**로 설정된다.

출력 코드가 1로 시작되었으면(논제로 입력에 대응), kp는 선정된 감소 단계 **Dn**만큼 감소된다. 즉, **kp = kp - Dn**으로 설정된다. 다음 입력 심볼을 인코딩하기 위한 k 값은 $k = \lfloor \mathbf{kp}/\mathbf{L} \rfloor$ 로 설정된다.(즉, **kp/L**의 끝수를 버리고 가장 가까운 정수로 내림).

알고리즘은 간단한 방법을 근거로 한다. 0들의 런이 발생하면, k는 증가되어, 보다 긴 0들의 시퀀스들이 싱글 출력 비트 = 0에 의해 캡쳐되게 한다. 논제로 심볼이 발생하면, k는 감소되어 지나치게 긴 출력 코드들의 발생을 방지한다. 상술된 보조 파라미터 **kp** 및 스케일링 배율 **L**을 사용해서 상술된 바와 같이 부동 소수점 연산을 사용할 필요 없이, 소수부 단계들(fractional steps)에서 k를 조정할 수 있게 한다.

ARLGR 인코더에서 테스트된 데이터의 대부분의 경우, 통상적인 파라미터 선택들에 있어서: **L** = 4, **Up** = 4, **Dn** = 5, **Uq** = 2 인 경우, 성능은 괘 양호했다(인코딩 비율이 소스 엔트로피들에 거의 근접함). 몇몇 경우, 상기 파라미터들을 조정함으로써 조금 더 양호한 성능을 도출할 수도 있다.

도 6의 흐름도에 대한 설명으로 돌아가서, 블록들(602, 604, 606, 608, 610, 612)에 설명된 파라미터들의 초기화 및 정의에 이어서, k는 먼저 단계(614)에서 0과 동일한지의 여부를 알기 위해 검사된다. k = 0 이고 symbol = 0 이면, Uq는 단계(618)에서 kp에 가산된다. 제로는 단계(620)에서 출력 버퍼에 추가되고, kp가 단계(622)에서 범위 밖이면, 즉 kpmax 보다 크면, kp는 클리핑된다(clipped). 단계(624)에서, k는 스케일링 배율 kp/L보다 작은 가장 큰 정수로 결정된다. xindex는 증가되고, 단계(628)에서 결정되는 것에 의해 M 보다 작은 경우, 다음 symbol이 단계(612)에서 선택된다. M 보다 큰 경우, 출력 비트 버퍼가 단계(630)에서 기록되고, 프로세스는 단계(640)에서 종료된다.

결정 블록(616)으로 다시 돌아가서, symbol이 0과 동일하지 않은 경우, 1이 단계(642)에서 출력 비트 버퍼에 추가되고, 데이터가 부호 비트를 가지면, symbol의 부호 비트가 단계(644)에서 출력 비트 버퍼에 추가되며, 프로세싱은 단계(622)로 이어져서 kp가 범위 내에 있는지의 여부를 검사한다.

k가 블록(614)에서 1과 동일하지 않으면, 단계(650)에서 symbol에 대한 자세한 검사가 이루어진다. symbol이 0과 동일하지 않으면, 1이 단계(652)에서 출력 비트 버퍼에 추가되고, run의 k-비트 값이 단계(654)에서 출력 비트 버퍼에 추가된다. 단계(656)에서, Dn이 kp로부터 감산되고, 프로세싱은 선택적인 부호 비트가 추가되는 단계(644)로 진행한다.

symbol이 단계(650)에서 0인 것으로 판정되면, run이 단계(622)에서 runmax와 동일한지의 여부를 검사한다. 동일하지 않으면, kp는 단계(622)에서 kpmax를 초과하지 않도록 클리핑된다. run이 단계(662)에서 runmax와 동일했으면, 0이 단계(664)에서 출력 비트 버퍼에 추가되고, run은 단계(666)에서 0으로 설정된다. 최종적으로, Up가 kp에 가산되고, 프로세싱은 다시 kp를 클리핑하기 위한 블록(622)으로 돌아오며, 단계(624)에서 k를 설정하고, (626)에서 xindex를 증가시키고, 단계(628)에서 최종 심볼이 처리되었는지를 검사한다. 그렇다면, 정보가 단계(630)에서 출력 비트 버퍼에 기록되고 프로세스는 단계(640)에서 종료한다.

표 4에는 양자화된 웨이브렛 계수들에 대한 비트 평면 인코더 사용 결과들이 도시되어 있다. 간단한 비트 평면 인코더가 계산이 보다 간단함에도 불구하고, 적응 산술 인코더들(종래 기술의 인코더로 생각됨) 보다 더 양호하게 실행됨에 주목한다.

[표 4]

입력으로서 양자화되고 리오더된 웨이브렛 계수들의 바이트 출력 코드 길이

데이터 집합 (길이 = 30,000 값)	본 발명의 비트-평면 인코더	적응 산술 인코더	적응 ELS 인코더
웨이브렛 데이터, 적은 빈도수	8,359	12,748	12,129
웨이브렛 데이터, 중간 빈도수	4,906	5,608	5,022

산술 인코더들에 의해 공유되지 않은 인코더의 주요한 장점은 확장성(scalability)이다. 기술된 비트 평면 인코딩에 있어서, 신호의 저 충실도 버젼은 평면 1 보다 큰 비트 평면의 디코딩 프로세스를 정지함으로써 쉽게 얻어질 수 있다. 이는 정보의 순차적인 전송 및 재생을 가능케 하고, 인터넷과 같은 통신 채널들의 중요한 특징을 허용한다. 확장성의 다른 애플리케이션은, 예를 들면 디지털 카메라에 있다. 사용자가 보다 많은 화상을 찍기 원하고, 이미 기억된 화상들의 품질을 기꺼이 희생시키고자 하면, 현존 이미지들의 보다 낮은 비트 평면들이 새로운 화상들을 위해 기억 용량을 늘릴 수 있도록 제거될 수 있다.

ARLGR 인코더가 비트 평면 인코더의 사용과 관련해서 기술되었지만, 값 0이 값 1 보다 발생할 확률이 훨씬 많은 이진 데이터를 위한 범용 인코더로서도 매우 유용하게 사용될 수 있다. 이는 특히, 확률 분포가 일정하게 변하는 경우에 유용하다. 예를 들어, 480 ×640 픽셀들의 해상도로 스캔된 흑백 그림을 인코딩할 때의 문제점을 생각해 보자. 화이트 = 0 및 블랙 = 1 로 맵핑한다고 가정하면, ARLGR 인코더는 데이터에 직접 적용될 수도 있다. 그러나, 인코더(240)는 1들의 런들을 매우 잘 처리하지 못하고, 따라서 차 연산자(difference operator)가 먼저 픽셀들의 모든 행들에 대해 적용된다. 제2 행으로부터 시작해서 아래로 이동할 때, 상기 행의 동일한 픽셀들로서 동일한 컬러를 가지면, 각각의 픽셀 값은 0으로 대체되고, 상이한 컬러를 가지면 1로 대체된다. 이는 열들에 대해서도 반복된다. 결과 비트들은 ARLGR 인코더(240)에 의해 인코딩된다.

이는 정보의 어떠한 손실도 없이 화이트 또는 블랙 런들이 0들의 런들로 맵핑될 수 있도록 한다. 따라서, 이는 데이터가 ARLGR 인코딩에 보다 적합하게 되도록 한다. 표 5는 간단한 인코더와 다른 수단들과의 성능 비교를 도시한 것이다.

[표 5]

일반적인 흑백 화상 데이터를 인코딩하기 위한 바이트 출력 코드 길이

기술된 ARLGR 인코더	CCITT 팩스 표준	적응 ELS 인코더	적응 산술 인코더
3,294	5,926	5,331	3,393

ARLGR 인코더(240) 알고리즘은 표준 팩스 인코딩 알고리즘을 거의 2배 능가한다. 팩스 알고리즘에 의해 사용되는 바이트들의 55%만을 사용한다. 실제로, 새로운 ARLGR-기반 인코더는 특정 이미지에 대한 적은 마진으로 종래 기술의 적응 산술 인코더를 훨씬 능가했다. 또한, 최저 계산 복잡성(lowest computation complexity)을 갖는다. 이는 단지 하나의 일례이며, 결과들이 사용된 이미지 및 파라미터의 조정에 따라 변할 수도 있음에 주목하여야 한다.

도 8에는 오피스 프로그램들의 슈트(suite)의 블록도(810)가 전반적으로 도시되어 있다. 한 특정 오피스 슈트는 워드 프로세싱, 이메일, 스프레드시트, 프리젠테이션 툴, 사진 조작 프로그램들 및 브라우저들과 같은 애플리케이션들을 포함하는, 다수의 하이 레벨 애플리케이션들(812)을 포함한다. 하위 레벨 소프트웨어, 하드웨어 또는 소프트웨어와 하드웨어의 결합 중 적어도 2개의 기능들(826, 818)이 상기 애플리케이션들을 지원한다. 도시된 기능들은 비디오 입출력 기능(826) 및 팩스/스캐너 기능(818)을 포함한다. 상기 레벨의 다수의 다른 기능들이 상주할 수도 있다.

특히, 비디오 기능은 외부 소스로부터의 비디오 및 이미지 데이터를 디스플레이하고 수신할 수 있는 기능을 모두 제공한다. 비디오 및 팩스/스캐너 기능들은 본 명세서에 기술되어 있으며, 블록(832)에 도시된 인코더 및 디코더를 사용해서 상술된 바와 같은 인코딩 및 디코딩 기능들을 제공한다. 로우(raw) 이미지 또는 다른 적합한 데이터가 픽셀 또는 다른 형태로 캡쳐되면, 인코딩을 위해 인코더(832)가 사용된다. 또한, 인코딩된 데이터가 본 명세서에 기술된 인코딩 유형을 사용해서 임의의 소스로부터 획득되면, 디코더(832)가 상기 데이터를 수신하는 애플리케이션에 의해 호출되어서, 상기 데이터를 디스플레이 가능하거나 유용한 포맷으로 변환하거나 디코딩한다.

훨씬 많은 애플리케이션들을 통합할 수도 있는 마이크로소프트 오피스 또는 수반(follow-on) 제품들과 같은 통합 오피스 슈트를 포함할 수도 있는 다수의 애플리케이션들이, 압축되거나 압축 해제될 필요가 있는 데이터를 처리할 수 있는 확률이 훨씬 더 많아졌음에 주목하여야 한다. 본 발명은 JPEG에 존재하는 블로킹 아티팩트들을 제거하고, 소정의 소프트웨어, 하드웨어 또는 하이브리드 형태들로, 보다 덜 복잡하게 구현되는 다른 형태의 코딩 기법들을 대안으로 제공한다. 인코더/디코더(832)는 또한, 쉽게 상기 오피스 슈트로 통합될 수 있다.

결론

양자화된 웨이브렛 계수들의 리오더링은 데이터-종속 데이터 구조들을 사용할 필요 없이, 크고 작은 웨이브렛 계수들을 개별적인 그룹들로 클러스터하기 위해 실행된다. 계수들은 그 후, 코드워드들에서 사용되는 비트들의 수를 최소화하면서, 양자화된 계수들의 스트링들을 표현하는데 사용되는 코드워드들을 제어하는 파라미터를 계속해서 변경시키는 런-길이 코드를 근거로 적응 인코딩된다. 오더링 패턴이 고정되고, 계수 인코딩이 각각의 이미지를 위해 수정된 테이블를 요구하지 않기 때문에, 본 발명은 보다 쉽게 하드웨어 또는 소프트웨어로 구현될 수 있다. 다른 장점들은 블로킹 아티팩트들을 제거할 수 있고, 이미지 데이터에 대해 임의의 소망의 압축률로 단일 경로(single path) 인코딩할 수 있다는 점들을 포함한다.

디코더는 상술된 인코딩 및 블로킹을 역순으로 적용하는 것으로 기술되었다. 인코딩된 계수들의 디코딩이 먼저 실행된 후, 계수들은 언셔플된다. 언셔플된 계수들은 그 후, 역 웨이브렛 변환을 거쳐, 이미지 픽셀들과 같이 변환되고 압축된 데이터를 복구한다. 적응 산술 코딩은 또한 유사한 압축 장점들을 가질 수 있도록 리오더링과 관련해서 사용될 수도 있는데, 이는 조금 더 높은 복잡성을 야기한다.

제로 트리와 같은 데이터-종속 데이터 구조들, 또는 트리의 세트 분할을 위한 분리 리스트를 사용할 필요 없이, 보다 쉽게 하드웨어가 구현된다. 본 출원은 본 발명의 임의의 응용 또는 변화를 포함하도록 의도된 것이다.

도면의 간단한 설명

도 1은 본 발명이 구현될 수도 있는 컴퓨터 시스템의 블록도이다.

도 2는 비손실 적응 방식으로 웨이브렛 계수들을 리오더하고 인코딩하는 인코더의 블록도이다.

도 3은 도 2의 인코더에 의해 생성된 인코드 계수들을 디코딩하고 언셔플(unshuffle)하는 디코더의 블록도이다.

도 4는 도 2의 인코더에 의해 생성된 리오더 웨이브렛 계수들의 블록도이다.

도 5는 계수들을 비트 평면들로 분리하는 도 2의 계수 인코더의 하이 레벨 동작을 도시한 흐름도이다.

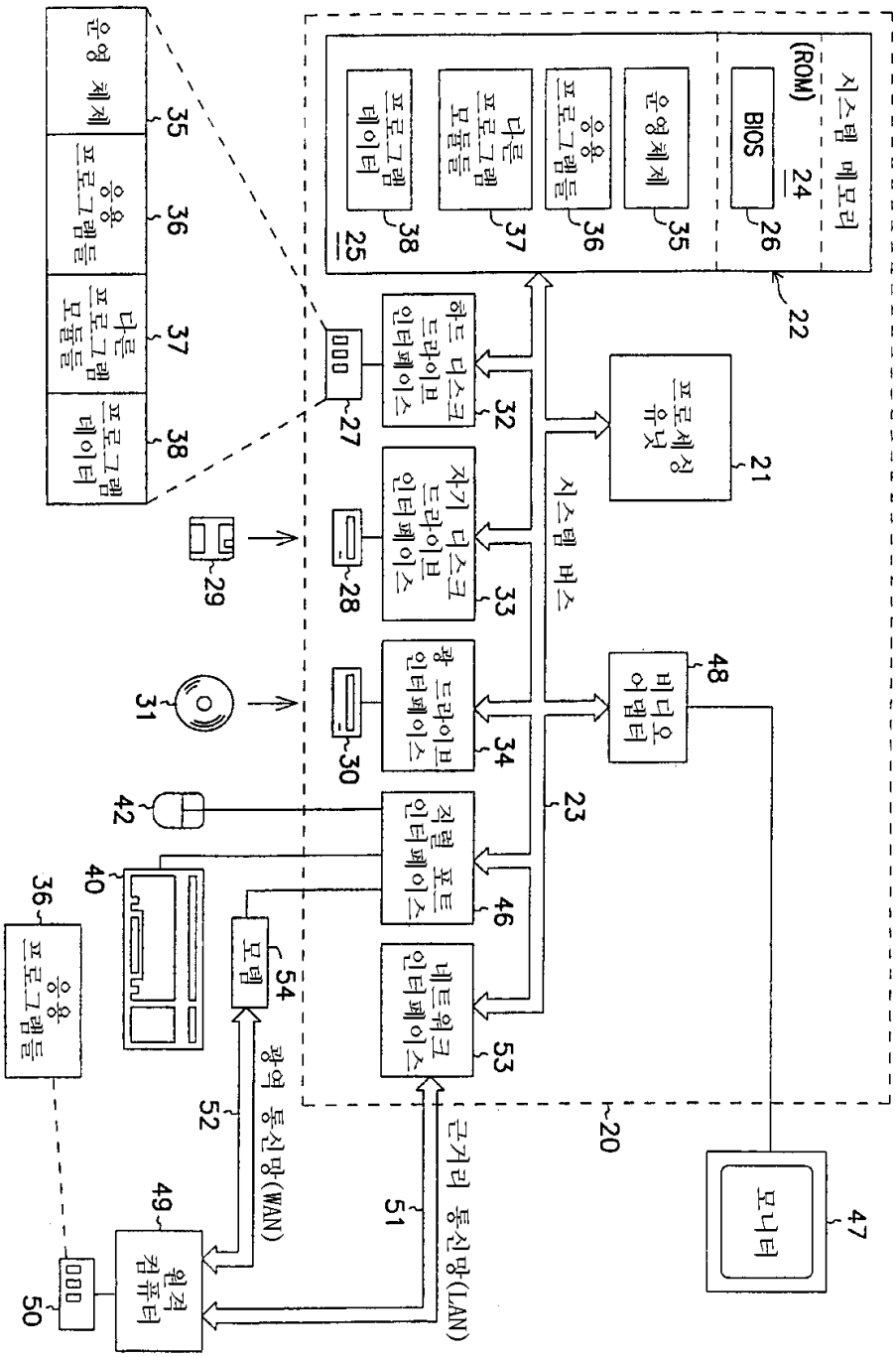
도 6은 도 2의 런-길이 적응 인코더의 동작을 더 상세히 도시한 흐름도이다.

도 7은 도 4에 도시된 바와 일치하는 리오더 방식으로 계수 매트릭스를 기록하는 것을 도시한 흐름도이다.

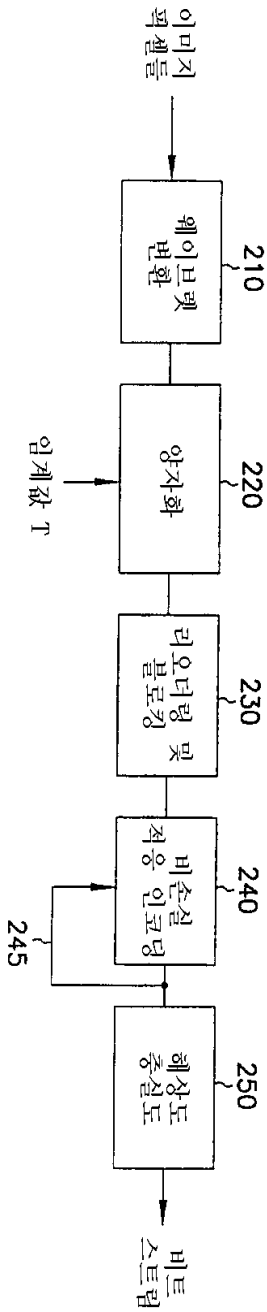
도 8은 이미지 데이터를 처리하는 소프트웨어 애플리케이션 슈트(software application suite)에서 도 2의 인코더 및 도 3의 디코더를 사용하는 것을 도시한 블록도이다.

도면

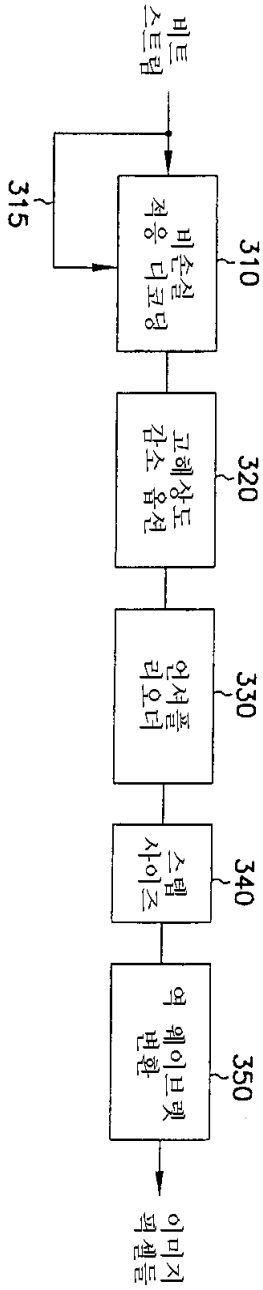
도면1



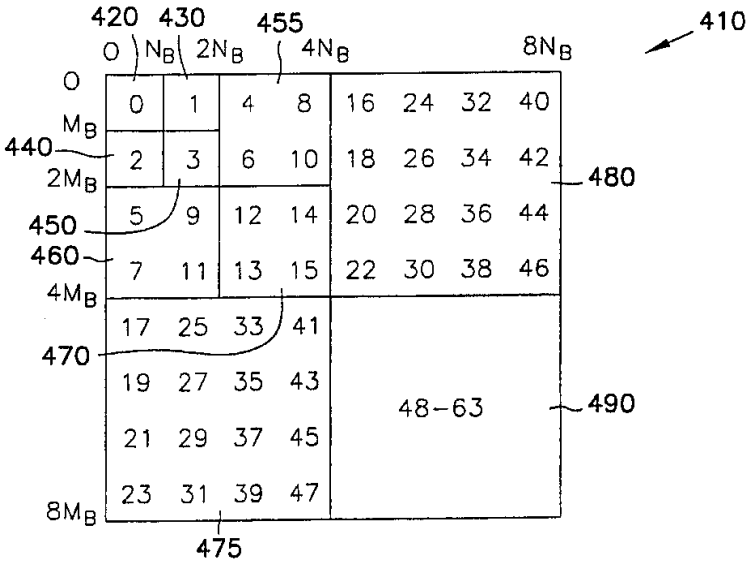
도면2


이미지 픽셀들
웨이브렛 변환
210
양자화
220
임계값 T
리오더링 및 블로킹
230
비손실 적응 인코딩
240
245
해상도 증진도
250
비트 스트림

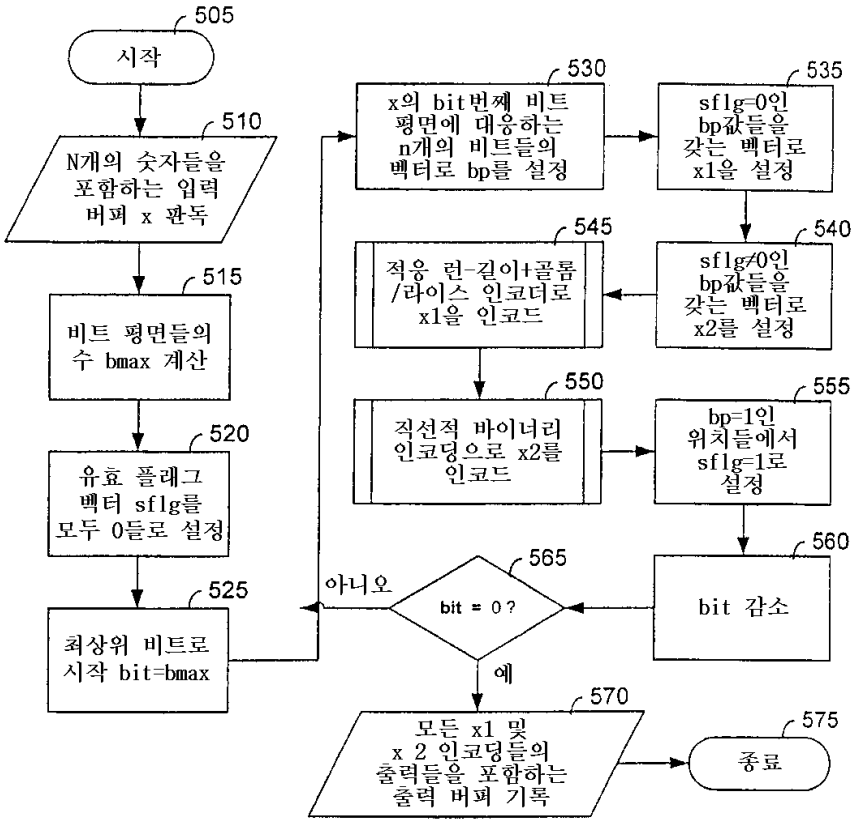
도면3


비트 스트림
315
비손실 적응 디코딩
310
고해상도 감소 옵션
320
언셔플 리오더
330
스텝 사이즈
340
역 웨이브렛 변환
350
이미지 픽셀들

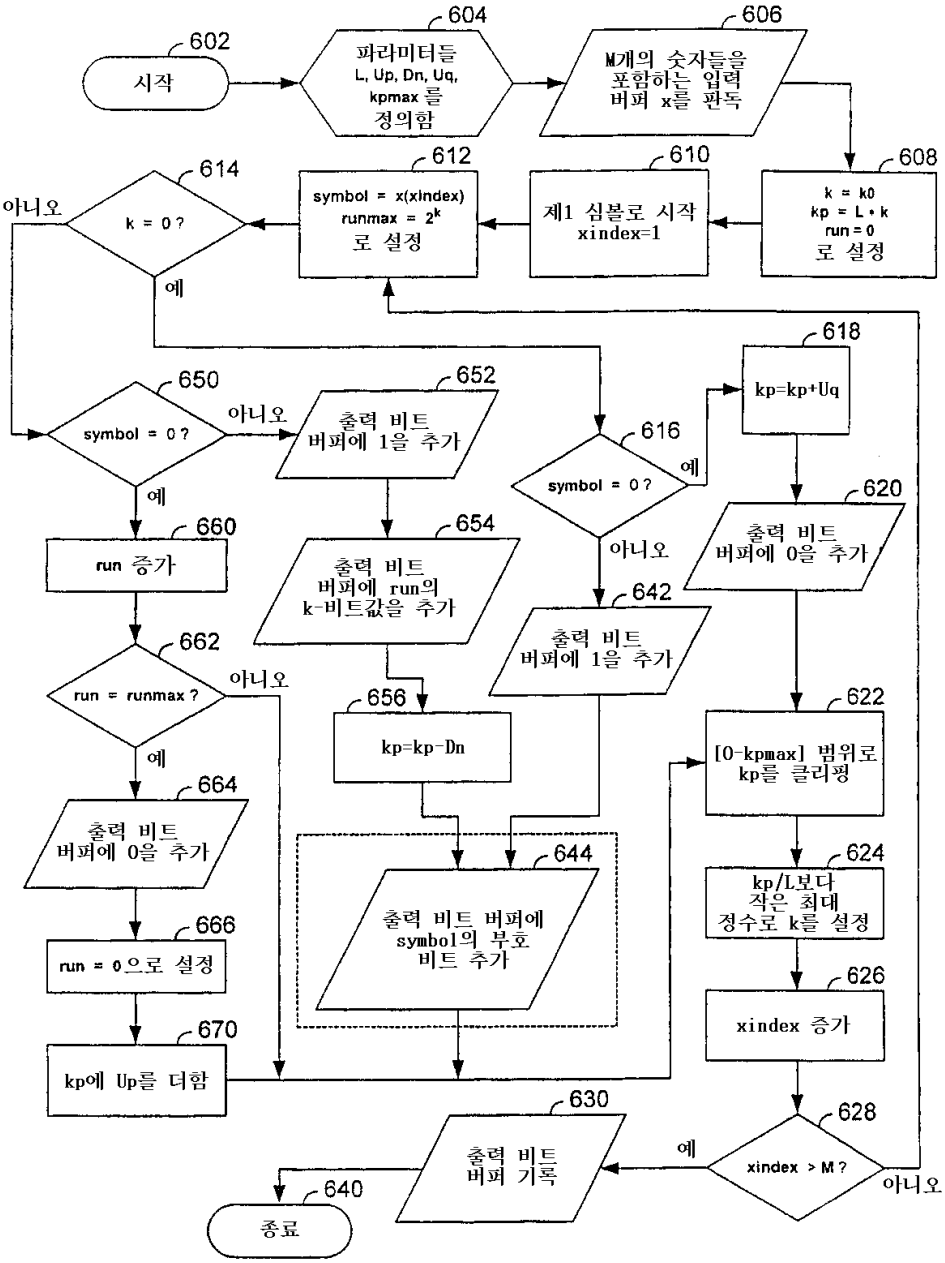
도면4


420
430
455
410
0
N_B
$2N_B$
$4N_B$
$8N_B$
0
M_B
$2M_B$
$4M_B$
$8M_B$
440
450
460
470
475
480
490
0 1 4 8 16 24 32 40
2 3 6 10 18 26 34 42
5 9 12 14 20 28 36 44
7 11 13 15 22 30 38 46
17 25 33 41
19 27 35 43
21 29 37 45
23 31 39 47
48-63

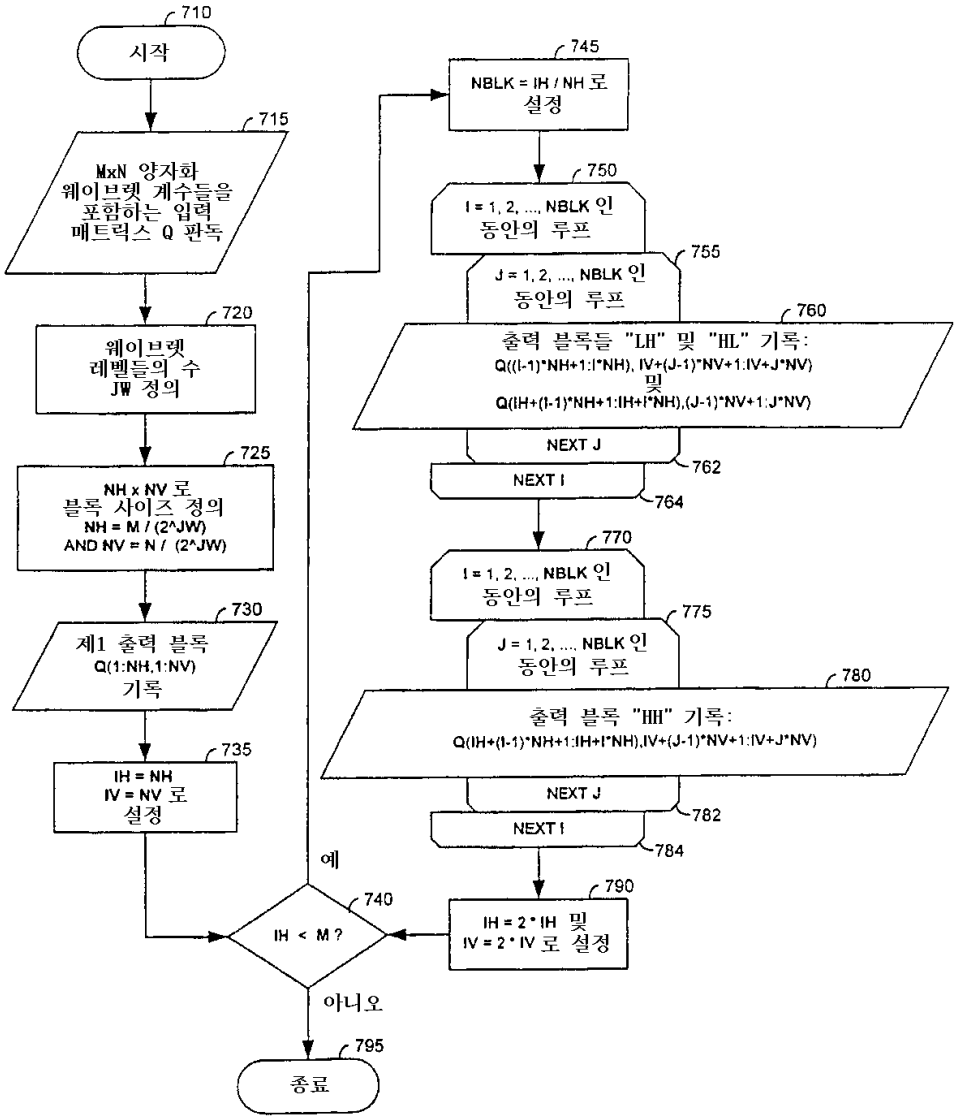
도면5


505
시작
510
N개의 숫자들을 포함하는 입력 버퍼 x 판독
515
비트 평면들의 수 bmax 계산
520
유효 플래그 벡터 sflg를 모두 0들로 설정
525
최상위 비트로 시작 bit=bmax
530
x의 bit번째 비트 평면에 대응하는 n개의 비트들의 벡터로 bp를 설정
535
sflg=0인 bp값들을 갖는 벡터로 x1을 설정
540
sflg≠0인 bp값들을 갖는 벡터로 x2를 설정
545
적응 런-길이+골롬/라이스 인코더로 x1을 인코드
550
직선적 바이너리 인코딩으로 x2를 인코드
555
bp=1인 위치들에서 sflg=1로 설정
560
bit 감소
565
bit = 0?
아니오
예
570
모든 x1 및 x 2 인코딩들의 출력들을 포함하는 출력 버퍼 기록
575
종료

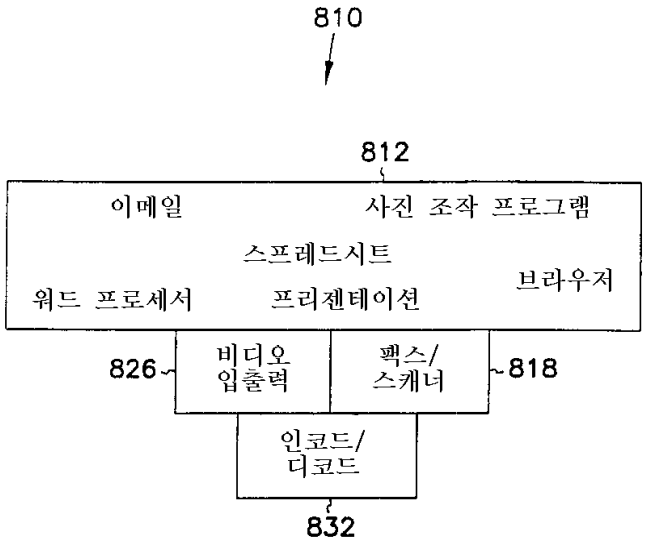
도면6


602
시작
604
파라미터들 L, Up, Dn, Uq, kpmax 를 정의함
606
M개의 숫자들을 포함하는 입력 버퍼 x를 판독
608
k = k0
kp = L • k
run = 0
로 설정
610
제1 심볼로 시작
xindex=1
612
symbol = x(xindex)
runmax = 2^k
로 설정
614
k = 0 ?
아니오
예
650
symbol = 0 ?
아니오
예
652
출력 비트 버퍼에 1을 추가
654
출력 비트 버퍼에 run의 k-비트값을 추가
656
kp=kp-Dn
660
run 증가
662
run = runmax ?
아니오
예
664
출력 비트 버퍼에 0을 추가
666
run = 0으로 설정
670
kp에 Up를 더함
616
symbol = 0 ?
예
아니오
618
kp=kp+Uq
620
출력 비트 버퍼에 0을 추가
642
출력 비트 버퍼에 1을 추가
644
출력 비트 버퍼에 symbol의 부호 비트 추가
622
[0-kpmax] 범위로 kp를 클리핑
624
kp/L보다 작은 최대 정수로 k를 설정
626
xindex 증가
628
xindex > M ?
예
아니오
630
출력 비트 버퍼 기록
640
종료

도면7


710
시작
715
MxN 양자화 웨이브렛 계수들을 포함하는 입력 매트릭스 Q 판독
720
웨이브렛 레벨들의 수 JW 정의
725
NH x NV 로 블록 사이즈 정의
NH = M / (2^JW)
AND NV = N / (2^JW)
730
제1 출력 블록 Q(1:NH,1:NV) 기록
735
IH = NH
IV = NV 로 설정
740
IH < M ?
예
아니오
745
NBLK = IH / NH 로 설정
750
I = 1, 2, ..., NBLK 인 동안의 루프
755
J = 1, 2, ..., NBLK 인 동안의 루프
760
출력 블록들 "LH" 및 "HL" 기록:
Q((I-1)*NH+1:I*NH), IV+(J-1)*NV+1:IV+J*NV)
및
Q(IH+(I-1)*NH+1:IH+I*NH),(J-1)*NV+1:J*NV)
762
NEXT J
764
NEXT I
770
I = 1, 2, ..., NBLK 인 동안의 루프
775
J = 1, 2, ..., NBLK 인 동안의 루프
780
출력 블록 "HH" 기록:
Q(IH+(I-1)*NH+1:IH+I*NH),IV+(J-1)*NV+1:IV+J*NV)
782
NEXT J
784
NEXT I
790
IH = 2 * IH 및
IV = 2 * IV 로 설정
795
종료

도면8


810
812
이메일
사진 조작 프로그램
스프레드시트
워드 프로세서
프리젠테이션
브라우저
826
비디오 입출력
팩스/ 스캐너
818
인코드/ 디코드
832