(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년11월19일
(11) 등록번호 10-1463279
(24) 등록일자 2014년11월12일

(51) 국제특허분류(Int. Cl.)
H04N 19/91 (2014.01)
(21) 출원번호 10-2009-7019348
(22) 출원일자(국제) 2008년02월15일
심사청구일자 2013년02월13일
(85) 번역문제출일자 2009년09월16일
(65) 공개번호 10-2009-0115208
(43) 공개일자 2009년11월04일
(86) 국제출원번호 PCT/SE2008/000125
(87) 국제공개번호 WO 2008/100206
국제공개일자 2008년08월21일
(30) 우선권주장
0700446-8 2007년02월16일 스웨덴(SE)
(뒷면에 계속)
(56) 선행기술조사문헌
US5349348 A
WO2005050567 A2
JP2003198378 A
전체 청구항 수 : 총 23 항

(73) 특허권자
모바일 이미징 인 스웨덴 아베
스웨덴 에스-223 70 룬트 쉴레배겐 17 이데온 리서치 파크
(72) 발명자
니에미, 사미
스웨덴 에스-239 32 스카뇌르 외른배겐 14
스텐, 요한
스웨덴 에스-211 55 말뫼 아미랄스가탄 26
(74) 대리인
제일특허법인, 김수진

심사관 : 오석환

(54) 발명의 명칭 **데이터 스트림의 생성 및 데이터 스트림 내부의 위치들의 식별**

(57) 요 약

가변 길이 코딩(Variable Length Coding) 방식을 이용하여 인코딩되는 데이터 스트림을 생성하기 위한, 컴퓨터 프로그램 제품들(products)을 포함하는 장치들 및 방법들이 개시된다. 가변 길이 코딩 방식에 따라 다수의 코드 워드들을 포함하는 데이터 스트림에 대한 코드 워드들이 인코딩된다. 분리 마커가 데이터 스트림 내의 인코딩된 데이터 블록들 사이에 삽입된다.

(30) 우선권주장
0701690-0 2007년07월11일 스웨덴(SE)
60/891,439 2007년02월23일 미국(US)

특허청구의 범위

청구항 1

가변 길이 코딩 방식에 의해 인코딩되는 데이터 스트림을 생성하기 위한 방법으로서,

가변 길이 코딩 방식에 따라, 복수의 코드 워드들(code words)을 포함하는 데이터 스트림에 대해 코드 워드들을 인코딩하는 단계와,

상기 데이터 스트림의 인코딩된 코드 워드들 사이에 분리 마커(separation marker)를 삽입하는 단계를 포함하되,

상기 분리 마커의 삽입은 특정 코드 워드 이후에 적어도 16개의 연속적인 2진수 1들을 삽입함으로써 수행되는

데이터 스트림 생성 방법.

청구항 2

가변 길이 코딩 방식에 의해 인코딩되는 데이터 스트림을 생성하기 위한 방법으로서,

가변 길이 코딩 방식에 따라, 복수의 코드 워드들을 포함하는 데이터 스트림에 대해 코드 워드들을 인코딩하는 단계와,

상기 데이터 스트림의 인코딩된 코드 워드들 사이에 분리 마커를 삽입하는 단계를 포함하되,

상기 분리 마커의 삽입은 특정 코드 워드 이후에 적어도 16개의 연속적인 2진수 제로(0)들을 삽입함으로써 수행되는

데이터 스트림 생성 방법.

청구항 3

제 1 항 또는 제 2 항에 있어서,

상기 특정 코드 워드는 데이터 블록의 마지막 코드 워드인

데이터 스트림 생성 방법.

청구항 4

제 1 항에 있어서,

상기 분리 마커의 삽입은 2개 이상의 특정 코드 워드들 사이에 적어도 16개의 연속적인 2진수 1들을 삽입함으로써 수행되는

데이터 스트림 생성 방법.

청구항 5

가변 길이 코딩 방식에 의해 인코딩되는 데이터 스트림을 생성하기 위한 방법으로서,

가변 길이 코딩 방식에 따라, 복수의 데이터 블록들을 포함하는 데이터 스트림의 데이터 블록들을 인코딩하는 단계와,

상기 데이터 스트림의 인코딩된 데이터 블록들 사이에 분리 마커를 삽입하는 단계를 포함하되,

상기 분리 마커는 적어도 16개의 연속적인 2진수 1들의 2진수 시퀀스인

데이터 스트림 생성 방법.

청구항 6

가변 길이 코딩 방식에 의해 인코딩되는 데이터 스트림을 생성하기 위한 방법으로서,

가변 길이 코딩 방식에 따라, 복수의 데이터 블록들을 포함하는 데이터 스트림의 데이터 블록들을 인코딩하는

단계와,

상기 데이터 스트림의 인코딩된 데이터 블록들 사이에 분리 마커를 삽입하는 단계를 포함하되,

상기 분리 마커는 적어도 16개의 연속적인 2진수 제로(0)들의 2진수 시퀀스인

데이터 스트림 생성 방법.

청구항 7

제 5 항 또는 제 6 항에 있어서,

상기 분리 마커를 삽입하는 단계는 이전의 블록이 인코딩된 이후 및 다음에 인코딩되는 데이터 블록이 이전의 데이터 블록에 부가되기 이전의 시간 지점에서 상기 분리 마커를 삽입하는 단계를 더 포함하는

데이터 스트림 생성 방법.

청구항 8

제 1 항, 제 2 항, 제 4 항, 제 5 항 또는 제 6 항 중 어느 한 항에 있어서,

상기 인코딩 및 상기 분리 마커의 삽입은 하드웨어에 의해 수행되는

데이터 스트림 생성 방법.

청구항 9

제 1 항, 제 2 항, 제 4 항, 제 5 항 또는 제 6 항 중 어느 한 항에 있어서,

상기 데이터 스트림에서 상기 분리 마커의 삽입은 상기 가변 길이 코딩 방식에 의해 인코딩되는 상기 데이터 스트림과 연계하여 수행되는

데이터 스트림 생성 방법.

청구항 10

제 6 항에 있어서,

상기 데이터 블록들의 인코딩은 JPEG 표준에 따라 수행되며, 각각의 데이터 블록은 상기 JPEG 표준에 따른 데이터 단위(unit)에 상응하는

데이터 스트림 생성 방법.

청구항 11

제 5 항에 있어서,

상기 분리 마커 이전에 8개의 2진수 제로(0)들을 삽입하는 단계를 더 포함하는

데이터 스트림 생성 방법.

청구항 12

제 5 항에 있어서,

상기 분리 마커 이전에 8 비트들의 세트를 삽입하는 단계를 더 포함하고, 상기 8 비트들의 세트는 2진수 제로값의 적어도 1 비트를 포함하며, 상기 2진수 제로값의 적어도 1 비트는 상기 8 비트들의 세트에서 미리 결정된 위치에 배열되는

데이터 스트림 생성 방법.

청구항 13

제 12 항에 있어서,

상기 2진수 제로값의 적어도 1 비트의 상기 미리 결정된 위치는 상기 8 비트들의 세트에서 최하위(the least

significant) 비트의 위치인

데이터 스트림 생성 방법.

청구항 14

제 6 항에 있어서,

상기 분리 마커 이전에 8개의 2진수 1들을 삽입하는 단계를 더 포함하는

데이터 스트림 생성 방법.

청구항 15

가변 길이 코드들의 스트림 내에서 특정 관심 코드 워드에 관련된 데이터를 조회(retrieve)하기 위한 방법으로서,

상기 가변 길이 코드들의 스트림 내에서 미리 정의된 분리 마커의 위치를 식별하는 단계 - 상기 미리 정의된 분리 마커의 위치를 식별하는 단계는 상기 분리 마커 옆의 가변 길이 코드들의 스트림에 배열된 미리 결정된 심볼을 식별하는 단계를 포함하고, 상기 미리 결정된 심볼 또는 상기 미리 정의된 분리 마커 중 하나로부터 상기 미리 정의된 분리 마커 또는 상기 미리 결정된 심볼로의 이동은 상기 분리 마커의 위치로써 식별됨 - 와,

상기 가변 길이 코드들의 스트림 내에서 상기 특정 관심 코드 워드의 시작 위치를 계산하는 단계와,

상기 특정 관심 코드 워드에 관련된 데이터를 조회하는 단계를 포함하는

데이터 조회 방법.

청구항 16

제 15 항에 있어서,

상기 미리 정의된 분리 마커는 적어도 16개의 연속적인 2진수 1들의 2진수 시퀀스이거나, 적어도 16개의 연속적인 2진수 제로(0)들의 2진수 시퀀스인

데이터 조회 방법.

청구항 17

제 15 항 또는 제 16 항에 있어서,

상기 특정 관심 코드 워드에 관련된 데이터를 조회하는 단계는, 상기 데이터 스트림 내에서 상기 특정 관심 코드 워드의 상기 계산된 시작 위치를 조회하는 단계와, 상기 가변 길이 코드들의 스트림의 특징들(features)의 리스트에 상기 시작 위치를 삽입하는 단계를 포함하는

데이터 조회 방법.

청구항 18

제 15 항 또는 제 16 항에 있어서,

상기 특정 관심 코드 워드에 관련된 데이터를 조회하는 단계는 상기 특정 관심 코드 워드에 의해 표현되는 값을 조회하는 단계를 포함하는

데이터 조회 방법.

청구항 19

제 18 항에 있어서,

상기 가변 길이 코드들의 스트림은 이미지의 압축된 표현(representation)이며, 상기 특정 관심 코드 워드는 상기 가변 길이 코드들의 스트림의 데이터 단위의 DC-계수인

데이터 조회 방법.

청구항 20

제 15 항 또는 제 16 항에 있어서,

상기 가변 길이 코드들의 스트림으로부터 상기 미리 정의된 분리 마커를 제거하는 단계를 더 포함하는

데이터 조회 방법.

청구항 21

제 15 항 또는 제 16 항에 있어서,

상기 미리 결정된 심볼은 EOB(End Of Block) 심볼인

데이터 조회 방법.

청구항 22

제 15 항 또는 제 16 항에 있어서,

상기 가변 길이 코드들의 스트림 내의 상기 특정 관심 코드 워드의 시작 위치의 계산은 상기 미리 정의된 분리 마커의 위치 및 상기 분리 마커의 공지된 길이에 기초하는

데이터 조회 방법.

청구항 23

삭제

청구항 24

제 15 항 또는 제 16 항에 있어서,

상기 가변 길이 코드들의 스트림은 복수의 코드 워드들을 포함하는 데이터 블록들을 포함하며, 상기 특정 관심 코드 워드는 데이터 블록의 제 1 코드 워드이고, 이에 따라 상기 특정 관심 코드 워드의 상기 시작 위치의 계산은 상기 데이터 블록의 시작 위치의 계산에 상응하는

데이터 조회 방법.

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

명 세 서

기 술 분 야

[0001] 본원 발명은, 가변 길이 코딩(Variable Length Coding) 방식을 이용하여 인코딩되는 데이터 스트림을 생성하기 위한, 컴퓨터 프로그램 제품들(products)을 포함하는 장치들 및 방법들과, 가변 길이 코드들의 스트림 내부의 코드 워드(code word)의 위치를 식별하기 위한 방법에 관한 것이다.

배 경 기 술

[0002] 오늘날 사진들, 스틸 사진들, 그래픽 등과 같은 이미지들은 일반적으로 디스플레이를 구비한 모든 전자 제품에서 볼 수 있다. 그러나, 이미지들을 디스플레이상에 보여지게 하는 것은 충분하지 않다; 캡쳐된 이미지들의 크기가 증가하기 때문에, 이미지들이 합리적인 시간내에 디스플레이될 수 있는 것이 또한 중요하다. 디스플레이상에 이미지들을 디스플레이하기 위한 전자 제품들의 사용자들은 대부분의 경우 이미지의 뷰(view)를 변경할 수 있는 것에 관심이 있다. 사용자에 의해 요청되는 몇몇 공통된 동작들은 이미지의 상세를 보기 위해 이미지를 줌(zoom)하는 것, 특징부(feature)를 추적하기 위해 줌된 이미지를 팬(pan)하는 것, 디스플레이상에서 보는 것을 용이하게 하기 위하여 이미지들을 회전하는 것이다. 또한, 이러한 전자 제품들의 사용자들에 의해 다루어지는 이미지들은 증가하는 해상도를 가진다, 즉, 이미지를 정의하는 화소들의 수가 점점 더 증가된다.

[0003] 상기 전자 장치들은, 예를 들어, 휴대 전화들, PDA, 팜탑(palm tops), 또는 다루어질 이미지들의 관점에서 제한된 처리 용량을 가지는 다른 장치들일 수 있다. 예를 들어, 이미지들을 보기 위한 디스플레이를 구비하는 다수의 상기 전자 장치들은 연속적으로 제공되는 뷰들 사이에 빈번하게 발생하고 성가신 지연들을 제공하지 않고 줌, 팬 등과 같은 동작들을 실행하는 데 충분한 처리 능력을 포함하지 않는다. 이에 의해, 이미지를 연속적으로 줌인 또는 줌아웃하는 것이 긴 지연의 중간 이미지들을 갖는 다수의 이미지들의 제공으로서 경험될 수 있다는 결과에 이른다. 따라서, 연속된 줌의 경험이 달성되지 않으며, 이는 사용자를 화나게 할 수 있다. 이는 또한 사용자에 의한 잘못된 처리 또는 입력들을 발생시킬 수 있다.

[0004] 이 문제를 해결하기 위한 하나의 일반적인 방법은 장치의 처리 용량을 증가시키거나 사용자가 연속된 흐름의 경험을 예상하는 동작들을 회피하는 것이다.

발명의 상세한 설명

[0005] 본 발명의 목적은 가변 길이 코딩된 스트림들 및/또는 이미지들에 대한 연산을 개선하고 상기 스트림들 및/또는 이미지들에 대한 연속된 연산들의 사용을 위한 경험을 개선하고, 캡쳐된 이미지를 스크린에 디코딩하는 시간을 개선하기 위한 것이다.

[0006] 본 발명의 일 양상에 따르면, 상기 목적은 가변 길이 코딩 방식에 의해 인코딩되는 데이터 스트림을 생성하기 위한 방법에 의해 성취된다. 상기 방법은, 가변 길이 코딩 방식에 따라, 다수의 코드 워드들(code words)을 포함하는 데이터 스트림에 대해 코드 워드들을 인코딩하는 단계, 및 상기 데이터 스트림의 인코딩된 코드 워드들 사이에 분리 마커(separation marker)를 삽입하는 단계를 포함한다.

[0007] 분리 마커를 도입함으로써, 가변 길이 코드들의 데이터 스트림에 대한 부분적인 액세스가 용이해진다. 예를 들어, 가변 길이 코딩을 이용하여 코딩되는 이미지의 임의의 영역을 표현하는 이미지 데이터에 대한 액세스를 가속하는 것이 가능해진다. 또한, 마커는 액세스될 데이터의 앞에 배열된 모든 단일 코드를 디코딩할 필요 없이 가변 길이 코드들의 데이터 스트림 내에서 데이터의 일부를 형성하는 데이터를 액세스할 수 있게 한다. 이에 의해, 시간 및 프로세싱 용량이 절약될 수 있다. 동일한 이점이 동일한 수송 또는 저장 매체, 예를 들어, 다중화된 데이터 스트림들을 공유하는 별개의 데이터 스트림들을 지시하기 위하여 마커들의 사용에 적용된다.

[0008] 일 실시예에서, 상기 분리 마커의 삽입은 특정 코드 워드 이후에 적어도 16개의 연속적인 2진수 1들을 삽입함으로써 수행된다. 다른 실시예에서, 상기 분리 마커의 삽입은 연속적인 1들 대신에 적어도 16개의 연속적인 2진수 제로(0)들을 삽입함으로써 수행된다. 16개의 연속적인 이진수 1들 및 제로들을 포함하는 마커를 사용함으로써, 마커에 대한 검색이 빠르고 효율적으로 수행될 수 있다. 그 이유는 바이트 레벨, 즉 8비트상에서 마커를 검출하는 것이 가능해지기 때문이다. 바이트 레벨에서의 검출이 가능한 것은, 가변 길이 코드들의 데이터 스트림 내에 있는 마커들의 일부, 16개의 연속적인 이진수 1들 및 제로들이 단지 이진수 1들 또는 이진수 제로들을 포함하는 바이트에 의해 항상 표현될 것이기 때문이다.

[0009] 다른 실시예에 따르면, 상기 특정 코드 워드는 데이터 블록의 마지막 코드 워드이다. 마커가 1의 마지막과 다른 데이터 블록의 시작을 식별하게 함으로써, 개별 데이터 블록들에 대한 액세스가 용이해지고 실질적으로 더 빠르게 수행될 수 있다.

[0010] 또 다른 실시예에 따르면, 상기 분리 마커의 삽입은 2개 이상의 특정 코드 워드들 사이에 적어도 16개의 연속적인 2진수 1들을 삽입함으로써 수행된다.

[0011] 일 실시예에서, 가변 길이 코딩 방식에 의해 인코딩되는 데이터 스트림을 생성하기 위한 방법은 가변 길이 코딩 방식에 따라, 다수의 데이터 블록들을 포함하는 데이터 스트림의 데이터 블록들을 인코딩하는 단계; 및 상기 데이터 스트림의 인코딩된 데이터 블록들 사이에 분리 마커를 삽입하는 단계를 포함한다.

[0012] 상기 실시예들 중 하나와 관련하여 언급된 것처럼, 분리 마커의 도입은 가변 길이 코드들의 데이터 스트림에 대한 부분적인 액세스를 용이하게 한다. 예를 들어, 가변 길이 코딩을 이용하여 코딩되는 이미지의 임의의 영역을 표현하는 이미지 데이터에 대한 액세스를 가속하는 것이 가능해진다. 또한, 마커는 액세스될 데이터의 앞에 배열된 모든 단일 코드를 디코딩할 필요 없이 가변 길이 코드들의 데이터 스트림 내에서 데이터의 일부를 형성하는 데이터를 액세스할 수 있게 한다. 이에 의해, 시간 및 프로세싱 용량이 절약될 수 있다. 동일한 이점이 동일한 수송 또는 저장 매체, 예를 들어, 다중화된 데이터 스트림들을 공유하는 별개의 데이터 스트림들을 지시하기 위하여 마커들의 사용에 적용된다. 또한, 마커가 데이터 블록을 식별하게 함으로써, 개별 데이터 블록들에 대한 액세스가 용이해지고 실질적으로 더 빠르게 수행될 수 있다.

[0013] 일 실시예에 따르면, 상기 분리 마커를 삽입하는 단계는 이전의 블록이 인코딩된 이후 및 다음에 인코딩되는 데이터 블록이 이전의 데이터 블록에 부가되기 이전의 시간 지점에서 상기 분리 마커를 삽입하는 단계를 더 포함한다.

[0014] 다른 실시예에서, 상기 인코딩 및 상기 분리 마커의 삽입은 하드웨어에 의해 수행된다. 이에 의해, 마커의 삽입을 가속화한다.

[0015] 일 실시예에서, 상기 데이터 스트림에서 상기 분리 마커의 삽입은 상기 가변 길이 코딩 방식에 의해 인코딩되는 상기 데이터 스트림과 연계하여 수행된다. 상기 가변 길이 코딩 방식에 의해 인코딩되는 상기 데이터 스트림과 연계하여 마커를 삽입함으로써, 인코딩은 마커들이 전체 스트림을 디코딩함으로써 데이터 스트림내로 삽입되는 경우보다 실질적으로 더 빨리 수행될 수 있다.

[0016] 다른 실시예에 따르면, 상기 데이터 블록들의 인코딩은 JPEG 표준에 따라 수행되며, 각각의 데이터 블록은 상기 JPEG 표준에 따른 데이터 단위(unit)에 상응한다. 상기 마크들을 JPEG 데이터 블록들을 포함하는 데이터 스트림들 내로 삽입함으로써, 인코딩된 데이터 스트림에 의해 표현되는 이미지의 특정 부분들의 액세스는 더 빨라지고 더 적은 프로세싱 용량을 요한다. 따라서, 이미지의 일부분들에 대한 연산들은 용이해질 수 있다.

[0017] 다른 실시예에 따르면, 상기 방법은 상기 분리 마커가 이진수 1들을 포함하면, 상기 분리 마커 이전에 8개의 2진수 제로(0)들을 삽입하는 단계를 포함하거나, 또는 상기 분리 마커가 이진수 0들을 포함하면, 상기 분리 마커 이전에 8개의 2진수 1들을 삽입하는 단계를 포함한다. 상기 분리 마커 이전에 이러한 시퀀스들 중 어느 하나 또는 다른 공지된 시퀀스를 삽입하는 단계에 의해, 분리 마커의 데이터 스트림 내에서의 정확한 개시점의 식별이 용이해진다. 상기 분리 마커 이전에 삽입된 시퀀스는 대안으로 8비트들의 세트일 수 있고, 상기 8 비트들의 세트는 2진수 제로값의 적어도 1 비트를 포함하며, 상기 2진수 제로값의 적어도 1 비트는 상기 8 비트들의 세트에서 미리 결정된 위치에 배열된다. 또한, 상기 2진수 제로값의 적어도 1 비트의 상기 미리 결정된 위치는 상기 8 비트들의 세트에서 최하위(the least significant) 비트의 위치일 수 있다.

[0018] 본 발명의 다른 양상에 따르면, 상기 목적은 가변 길이 코드들의 스트림 내에서 특정 관심 코드 워드에 관련된 데이터를 조회(retrieve)하기 위한 방법에 의해 달성된다. 상기 방법은, 상기 가변 길이 코드들의 스트림 내에서 미리 규정된 마커의 위치를 식별하는 단계; 상기 가변 길이 코드들의 스트림 내에서 상기 특정 관심 코드 워드의 시작 위치를 계산하는 단계; 및 상기 특정 관심 코드 워드에 관련된 데이터를 조회하는 단계를 포함한다.

[0019] 데이터 스트림내의 관심 있는 위치를 식별하기 위하여 가변 길이 코드들의 데이터 스트림내에 미리 정의된 마커를 이용함으로써, 이 위치의 액세스는 전체 데이터 스트림이 위치를 발견하기 위해 디코딩되어야 하는 경우보다 적은 시간 및 처리 용량을 요할 수 있다. 예를 들어, 가변 길이 코딩을 이용하여 코딩되는 이미지의 임의의 영역을 표현하는 이미지 데이터에 대한 액세스를 가속하는 것이 가능해진다.

[0020] 마커는 또한, 액세스될 데이터의 앞에 배열된 모든 단일 코드를 디코딩할 필요 없이 가변 길이 코드들의 데이터 스트림 내에 위치된 데이터를 액세스할 수 있게 한다. 이에 의해, 시간 및 프로세싱 용량이 절약될 수 있다. 동일한 이점이 동일한 수송 또는 저장 매체, 예를 들어, 다중화된 데이터 스트림들을 공유하는 별개의 데이터 스트림들을 지시하기 위하여 마커들의 사용에 적용된다.

[0021] 특정 실시예에 따르면, 상기 방법은 상기 가변 길이 코드들의 스트림 내에서 적어도 하나의 부가적인 미리 규정된 마커의 위치를 식별하는 단계; 상기 가변 길이 코드들의 스트림 내에서 적어도 하나의 부가적인 특정 관심 코드 워드의 시작 위치를 계산하는 단계; 및 상기 적어도 하나의 부가적인 특정 관심 코드 워드에 관련된 데이터를 조회하는 단계를 더 포함한다. 데이터 스트림을 디코딩하는 데 다수의 마커들이 식별되고 사용되기 때문에, 상기 프로세스는 심지어 더 적은 시간 및 처리 전력을 요할 수 있다.

[0022] 다른 실시예에서, 상기 특정 관심의 코드 워드에 관련된 데이터를 조회하는 단계는 상기 데이터 스트림 내에서 상기 특정 관심의 코드 워드의 상기 계산된 시작 위치를 조회하는 단계, 및 상기 가변 길이 코드들의 스트림의 특징들(features)의 리스트에 상기 시작 위치를 삽입하는 단계를 포함한다. 상기 특정 관심의 코드 워드의 위치를 조회하고 이 위치를 저장함으로써, 데이터 스트림을 맵핑하는 것이 가능하고 이에 의해 데이터 스트림의 부분들에 대한 심지어 더 빠른 미래의 액세스들을 수행하는 것을 가능하게 한다.

[0023] 또 다른 실시예에 따르면, 상기 특정 관심 코드 워드에 관련된 데이터를 조회하는 단계는 상기 특정 코드 워드에 의해 표현되는 값을 조회하는 단계를 포함한다. 이 실시예는 보다 드물게 사용될 특정 값들에 대한 액세스를 가속한다. 예를 들어, 저장되지 않을 예정인 이미지들에 대한 연산을 수행할 때가 그러하다.

[0024] 일 실시예에서, 상기 가변 길이 코드들의 스트림은 이미지의 압축된 표현(representation)이며, 상기 특정 관심 코드 워드는 가변 길이 코드들의 스트림의 데이터 단위의 DC-계수이다. 이에 의해, 캡쳐된 이미지들의 감소된 버전들의 빠른 생성이 용이해진다.

[0025] 일부 다른 실시예들에서, 본 발명의 제 1 양상과 관련하여 언급된 마크들에 대응하는 마커들 및 본 발명의 제 1 양상과 관련하여 언급된 마커들에 선행하는 코드들의 배열은 본 발명의 제 2 양상에 따른 방법에서 사용될 때 실질적으로 동일한 이점들을 제공한다.

[0026] 또 다른 실시예에서, 상기 미리 규정된 마커의 위치를 식별하는 단계는 상기 마커 다음의 상기 가변 길이 코드들의 스트림에 배열된 미리 결정된 심볼을 식별하는 단계를 더 포함하고, 상기 미리 결정된 심볼 또는 상기 미리 규정된 마커 중 하나로부터 상기 미리 규정된 마커 또는 상기 미리 결정된 심볼로의 변환(transition)은 상기 마커의 위치로서 식별된다. 이 실시예를 실행함으로써, 미리 규정된 마커의 위치의 결정이 용이해진다.

[0027] 다른 실시예에 따르면, 미리결정된 심볼은 EOB(End Of Block) 심볼이다.

[0028] 또 다른 실시예에서, 상기 특정 코드 워드의 가변 길이 코드들의 스트림 내에서 상기 시작 위치의 계산은 상기 미리 규정된 마커의 위치 및 상기 마커의 공지된 길이에 기초한다.

[0029] 또 다른 방법에 따라, 상기 가변 길이 코드들의 스트림은 다수의 코드 워드들을 포함하는 데이터 블록들을 포함하며, 상기 특정 관심 코드 워드는 데이터 블록의 제 1 코드 워드이고, 이에 따라 상기 특정 관심 코드 워드의 시작 위치의 계산은 상기 데이터 블록의 시작 위치의 계산에 상응한다. 마커가 1의 마지막과 다른 데이터 블록의 시작을 식별하게 함으로써, 개별 데이터 블록들에 대한 액세스가 용이해지고 실질적으로 더 빠르게 수행될 수 있다.

실 시 예

[0030] 본 발명은 가변 길이 코딩(VLC) 방식들 또는 엔트로피 코딩(entropy coding) 방식들에 의해 코딩되었던 데이터 블록들의 스트림의 빠른 인덱싱을 용이하게 하기 위한 방법들, 장치들, 및 컴퓨터 프로그램 제품들에 관한 것이다. 본 발명의 일 실시예에 따르면, 빠른 인덱싱은 데이터 스트림내에 순차적으로 배열된 데이터 블록들 사이에 데이터 블록 분리 마커(marker)의 삽입에 의해 성취된다. 본원에서, 데이터 스트림은, 동일한 데이터 파일의 일부인 것에 의해, 네트워크를 통해 전송되는 동일한 데이터 메시지의 일부인 것에 의해(이러한 메시지는 서로 다른 패키지들로 분리되거나 네트워크에 의존하여 하나의 연속된 스트림으로 전송될 수 있다), 스트리밍 오디오 또는 스트리밍 비디오 또는 이들의 조합과 같은 동일한 정보 스트림의 일부인 것에 의해, 서로와 관련된 데이터로서 이해되어야 한다.

[0031] 본 발명의 실시예들은 데이터 블록들이 VLC 방식 또는 엔트로피 코딩 방식(예, 허프만(Huffman) 인코딩, 산술 인코딩 등)을 이용하여 인코딩되는 모든 형태의 데이터 스트림에 적용할 수 있다. 이러한 인코딩 방식들은 정보를 압축하기 위하여 설계되었으며 이러한 인코딩 방식들을 이용하는 하나의 효과는 인코딩된 데이터 스트림내의, 코드 워드들(code words)이라고도 불리는, 모든 심볼들이 동일한 코드 길이를 가지지는 않는다는 것이다.

[0032] 그러나, 코드들의 방식은 하나의 심볼이 종료되고 다른 심볼이 시작되는 곳을 알기 위하여 스트림내의 모든 이전의 심볼들을 처리할 필요가 있다.

[0033] 본 출원의 문맥에서, 심볼은 유닛들의 세트로부터의 하나의 유닛으로서 이해되어야 하며, 여기서 각각의 유닛은 유닛들의 세트에 할당된 개별 비트 코드를 가진다. 본 출원에서, 데이터 블록은 보다 큰 데이터 스트림의 부분(예를 들어, 이미지 데이터의 부분, 오디오 스트림의 부분, 이미지 스트림의 직사각형 이미지 블록 등)를 나타내는 일부 데이터로서 이해되어야 한다.

[0034] 일 실시예에서, 둘 이상의 가변 길이 코딩된 스트림들이 보다 구체적인 규칙들에 따라(예를 들어, 제1 신호로부터 n 심볼들을 변경하고, 제2 신호로부터 m 심볼들을 변경하고, 제3 신호로부터 k 심볼들을 변경하는 것에 의해) 혼합되는 것이 바람직할 수 있다.

[0035] 이는, 모든 심볼들이 서로 다른 신호들 사이의 경계들을 발견하기 위하여 처리되어야만 하기 때문에, 수신단에서 신호들을 분리하는 데 대단한 어려움을 겪는다.

[0036] 다른 실시예에서, 특정 심볼, 즉 예를 들면 가변 길이 코딩된 스트림내의 모든 n 번째 심볼을 빠르게 찾는 것이 바람직할 수도 있다. 이는 또한 통상적으로 모든 심볼들이 원하는 심볼들의 위치를 찾기 위하여 처리될 것을 요한다.

[0037] 또 다른 실시예에서, 하나의 통신 채널을 통해 둘 이상의 신호들을 전송할 수 있는 것이 바람직하다. 이는 통상적으로 둘 이상의 신호들을 변경함으로써 달성되어, 제1 신호의 n 비트들이 먼저 전송된 후, 제2 신호의 m 비트들이 전송되고, 그 후 제1 신호의 b 비트들이 전송되고,… 등과 같이 이어진다.

[0038] 본 발명은 가변 길이 인코딩 프로세스에서 생성되는 스트림내에 삽입될 수 있는 특수한 마커들을 도입함으로써 가변 길이 코딩된 스트림들의 이용에 있어서의 전술한 문제점들을 해결한다. 이러한 마커들은 데이터 스트림내에서 특정 심볼들을 찾는 프로세스의 속도를 높이기 위해 사용될 수 있다. 부가적으로, 본원 발명은 최초의 스트림 또는 스트림들을 생성 또는 재생성하기 위하여 효율적인 방법으로 상기 마커들을 제거하는 방법을 도입한다.

[0039] *상기 마커의 예시적인 사용:*

[0040] 가변 길이 코딩된 스트림은 서로 다른 비트 길이들을 갖는 심볼들로 이루어진다. 다음의 심볼들은, 예를 들어, 다음의 코드 워드들에 의해 표현될 수 있다:

[0041] A = 10

[0042] B = 110

[0043] C = 111

[0044] D = 0000

[0045] E = 0001

[0046] 다음의 심볼들 "CBCDBEDCABCBECB"에 대한 비트 스트림은 다음과 같이 된다:

[0047] 11111011 10000110 00010000 11110110 11111000 01111110

[0048] 비트 스크림을 최초 심볼들(original symbols)로 디코딩하기 위해서 비트 스트림은 좌측으로부터 디코딩되어야 하며 비트들은 유효한 심볼이 생성될 때까지 판독된다. 전체 비트 스트림을 디코딩함으로써 심볼들 사이의 경계들이 위치된 곳을 이해할 수 있다.

[0049]

C	B	C	D	B	E	D	C	A	B	C	B	E	C	B
111	110	111	0000	110	0001	0000	111	10	110	111	110	0001	111	110

[0050] 이러한 접근법의 문제점은, 특히 많은 심볼들이 처리되어야 할 때 소프트웨어에서 실행하는 것이 비교적 시간 소비적이라는 점이다.

[0051] 16진수로 기록된 동일한 비트 스트림은 다음과 같다:

[0052] FB 86 10 F6 F8 7E

[0053] 본 발명의 다양한 실시예들에 따르면, 비트 마커들 또는 심볼 분리 마커들(symbol separating markers)이 특정 마커의 빠른 발견을 가능하도록 도입된다.

[0054] 비트 마커들은 *다수의 여덟개 및 적어도 16 비트만큼 긴 길이의* 일련의 이진수들(binary ones)로 이루어질 수 있다. 가장 흥미있는 마커는 16개의 이진수들, 즉 11111111 11111111이다.

[0055] 만약 우리가 스트림 내의 심볼 "A" 이후에 경계를 마킹하기를 원한다면, 스트림 내의 심볼 "A" 이후에 16개의 이진수들을 삽입함으로써 행해질 것이다:

C	B	C	D	B	E	D	C	A	비트 마커	B
111	110	111	0000	110	0001	0000	111	10	1111111111111111	110

C	B	E	C	B
111	110	0001	111	110

[0056]

[0057] 따라서 결과적인 비트 스트림은 다음과 같다:

[0058] 11111011100001100001000011110111111111111111111101111100001111110

[0059] 16진수들로 기록된 동일한 비트 스트림은 다음과 같다:

[0060] FB 86 10 F7 FF FE F8 7E

[0061] 이제, 우리는 스트림 내에 16개의 이진수들을 기록했기 때문에, 우리의 마커는 결코 "00"이 이어지지 않는 적어도 하나의 "FF"로 언제나 귀결될 것이다; 자연스럽게 발생하는 FF 코드들은 비-마커(non-marker)를 표시하기 위하여 "FF00"으로 표시될 수 있음을 의미한다; 모든 다른 FF 발생은 본 발명의 다양한 실시예들에 따라 마커임이 보장된다.

[0062] 이는 우리가 연속된 바이트들로서 보여지는 비트 스트림내에 바이트 FF를 쉽게 위치시킬 수 있고, 우리는 그 후 간단한 산술 로직을 이용함으로써 마커를 제거하고 최초 스트림을 복구할 수 있음을 의미한다.

[0063] 바이트들은 마커가 직접 복사될 수 있기 전의 바이트와 마커 이후의 하나의 바이트까지이다.(The bytes up to the byte before the marker can be copied directly, as well as the bytes one byte after the marker) 나머지 작업은 괄호내에 표시된, 이진수들을 삽입함으로써 분리되었던 바이트를 복구하는 것이다.

[0064] FB 86 10 (F7 FF FE) F8 7E

[0065] 이진수들로 기록된 F7 FF FE는 다음과 같이 보여진다:

B1	B2	B3
11110111	11111111	11111110

[0066]

[0067] 또는 보다 일반적으로

B1	B2	B3
abcde111	11111111	11111fgh

[0068]

[0069] 삽입된 비트들을 제거하는 동작은 매우 간단한데, 이는 삽입된 이진수들이 이들을 결합할 때 비트 마스크들로서 보여질 수 있기 때문이다.

[0070] 최초 바이트 = B1 & B3 = abcde111 & 11111fgh = abcdefgh

[0071] 나머지 스트림은 다음과 같이 된다.

[0072] FB 86 10 (F7&FE) F8 7E = FB 86 10 F6 F8 7E

[0073] 만약 16개의 심볼들 이전 또는 이후의 심볼이 알려져 있고 적어도 하나의 0을 포함한다면, 표시된 심볼들 사이의 경계의 비트 위치, 즉 마커의 위치를 찾는 것은 쉬운 일이다. 마커가 제거되었을 때 마커의 위치는 다음 심볼의 위치이다.

[0074] 이는 "FF"의 왼쪽으로 첫 번째 0을 찾고 0을 포함하는 심볼의 외관(appearance) 및/또는 코드를 알고 그 후 심볼이 종료하는 곳을 계산하는 것에 의해 달성될 수 있다.

[0075] 이를 달성하는 다른 방법은 "FF" 이전의 바이트에 대한 256개의 엔트리들의 검색 테이블을 이용하는 것에 의하는 것이고, 여기서 테이블 내의 값으로서 비트 위치 (p)를 가진 알려진 심볼을 갖는 모든 가능한 엔트리들이 입력된다.

[0076] *아래의 예는 검색 기술을 예시한다:*

[0077] 16개의 1들 이전의 알려진 심볼이 10이라고 가정하면, 바이트 1에 대한 유일한 가능성들은 다음과 같고, 여기서 x들은 이진수 1들 및 이진수 0들의 모든 순열이며, p는 16개의 1들의 첫 번째 비트를 나타낸다.

[0078] Byte1 Byte2

[0079] xxxxxx10 p1111111, 64 순열들, pos = 8

[0080] xxxxx10p 11111111, 32 순열들, pos = 7

[0081] xxxx10p1 11111111, 16 순열들, pos = 6

[0082] xxx10p11 11111111, 8 순열들, pos = 5

[0083] xx10p111 11111111, 4 순열, pos = 4

[0084] x10p1111 11111111, 2 순열, pos = 3

[0085] 10p11111 11111111, 1 순열, pos = 2

[0086] 0p111111 11111111, 1 순열, pos = 1

[0087] 상기 예에서, F7(11110111)에 대한 검색은 pos=6, (111101p11==111101111)로 귀결될 것이다.

[0088] 만약 심볼이 알려지지 않았다면, 예를 들어 8 비트 길이의 총 길이를 갖는 것이 바람직한 알려진 코드 (K)를 언제나 삽입하는 것이 가능할 것이며, 이는 우리가 표시된 심볼들 사이의 경계를 위치시키기 위해 필요로 되는 심볼을 아는 동시에 이를 쉽게 제거할 수 있게 해준다.

[0089] 이러한 목적으로 사용하기에 매우 양호한 코드는 "00000000"인데, 그 이유는 이것이 마커를 제거할 때 비트 마스크로서 사용될 수 있기 때문이다, 즉:

[0090] K = 0000 0000

[0091] 이러한 수정된 스트림은 다음의 외관을 가질 것이다:

C	B	C	D	B	E	D	C	A	K
111	110	111	0000	110	0001	0000	111	10	00000000

비트 마커	B	C	B	E	C	B
1111111111111111	110	111	110	0001	111	110

[0092]

[0093] 바이트들로 정렬되면 수정된 스트림은 다음의 외관을 가질 것이다:

[0094] **11111011 10000110 00010000 11110000 00000111 11111111 11111110 11111000 01111110**

[0095] 그리고 16진수들로 기록될 때 이는 다음과 같다:

[0096] FB 86 10 (F0 07 FF FE) F8 7E

[0097] 상기로부터, FF를 찾는 것이 가능하며, 알려진 심볼 K를 나타내는 FF 이전의 바이트를 검색함으로써, 심볼 K가 종료되는 곳을 이해할 수 있고, 따라서 표시된 심볼들 사이의 경계를 나타내는, K 이전의 심볼이 종료되는 곳을 찾기 위하여 역으로 계산할 수 있다.

[0098] 심볼 K 는 제1 위치에서 오른쪽으로부터 첫 번째 이진수 0을 가지며, 따라서, 심볼 K 이전의 심볼은 발견된 이진수 0의 왼쪽에서 7번째 위치에서 종료된다.

[0099] F0 07 FF FE = 11110000 00000111 11111111 11111110

[0100] 그 후, 아래에서 x로 표시된, FF의 왼쪽으로 이진수 0의 첫 번째 발생을 발견함으로써 제거가 실행될 수 있다.

[0101] 11110000 0000x111 11111111 11111110

[0102] 그리고 상기 발생이 발견된 이후에, 발견된 이진수 0의 왼쪽 7개 비트들로부터 시작하는 다음 24개 비트들을 제거하고, 이는 r로 표시된다:

[0103] 11110rrr rrrrrrrr rrrrrrrr rrrrr110

[0104] 이제, 4개 비트들을 하나로 합치는 것이 가능하고, 이는 다음과 같은 결과를 초래한다:

[0105] 1111110 = F6

[0106] 따라서, 전체 결과 바이트 어레이는 오리지널(original) 스트림이 된다:

[0107] FB 86 10 F6 F8 7E.

[0108] 0인 K가 마스크(mask)로서 작용할 수 있기 때문에, 상기 마커의 제거는 몇몇 단순한 산술 로직에 의하여 용이하게 수행된다:

[0109]

B1	B2	B3	B4
abcd0000	00000111	11111111	1111efgh

[0110] B = (B1|B2)&B4 = (F0|07)&FE = F7&FE = F6

[0111] 둘 이상의 신호들이 동일한 채널을 통해 송신될 때, 상기 마킹 방법을 사용하는 실시예는 다음과 같이 주어진다.

[0112] 단 하나의 채널을 통해 송신될 2개 비트 스트림들을 고려하라.

[0113]

	A	B	C	D	B	E	D	A	C	B	D
스트림 A	10	110	111	0000	110	0001	0000	10	111	110	0000

B	E	C	B
110	0001	111	110

[0114] 16진법으로 이것을 기록하면 다음과 같이 된다:

[0115] B7 0C 21 7C 18 7E

[0116]

	A	B	E	D	C	E	B	A	B	A	C
스트림 B	10	110	0001	0000	111	0001	110	10	110	10	111

D	A	B	D	A
0000	10	110	0000	10

[0117] 16진법으로 이것을 기록하면 다음과 같이 된다:

[0118] B0 87 1 D 6B 85 82

[0119] 스트림들은 스트림 A의 5개 코드워드들이 송신되고, 그후 스트림 B의 8개 코드워드들이 송신되고, 그 후, 스트림 A의 5개 코드워드들이 송신되고, 나머지도 이와 같은 방식으로 송신되도록, 본 실시예에서 스트림들은 변경된다:

	A	B	C	D	B
스트림 A	10	110	111	0000	110

	A	B	E	D	C	E	B	A
스트림 B	10	110	0001	0000	111	0001	110	10

	E	D	A	C	B
스트림 A	0001	0000	10	111	110

	B	A	C	D	A	B	D	A
스트림 B	110	10	111	0000	10	110	0000	10

	D	B	E	C	B
스트림 A	0000	110	0001	111	110

[0120]

[0121] 다음의 이진 스트림을 초래한다:

[0122] 10110111000011010110000100001110001110100001000 0

[0123] 10111110110101110000101100000100000110000111111 0

[0124] 16진수로 쓰여지면:

[0125] B7 0D 61 0E 3A 10 BE D7 0B 04 18 7E

[0126] 이제 수신기는 2개의 오리지널 스트림들을 재생성하기(recreate) 위하여 모든 VLC 심볼들을 프로세싱해야 한다.

[0127] 본 발명의 실시예들은 보다 효율적인 방식으로 2개의 스트림들을 분리할 수 있기 위하여 특별한 마커들을 도입하여, 마커의 삽입 방식이 일반적으로 하기의 실시예들에서보다 훨씬 더 드물게 발생하고, 따라서, 더 많은 오버헤드를 요구하지 않는다는 것을 유념하라.

	A	B	C	D	B		E
스트림 A	10	110	111	0000	110	00000000111111111111111	0001

D	A	C	B		D
0000	10	111	110	00000000111111111111111	0000

B	E	C	B
110	0001	111	110

[0128]

[0129] B7 0C 01 FF FE 21 7C 01 FF FE 18 7E

	A	B	E	D	C	E	B	A
스트림 B	10	110	0001	0000	111	0001	110	10

				B	A	C	D
	00000000111111111111111			110	10	111	0000

A	B	D	A	
10	110	0000	10	00000000111111111111111

[0130]

[0131] B0 87 1D 00 7F FF EB 85 82 00 FF FF

[0132] 이제 스트림들은 FF까지 제1 스트림을 출력하고, 그 후 FF까지 제2 스트림으로 계속되고, 그 후 다음 FF까지 제1 스트림으로 계속되고, 나머지도 이와 같은 방식으로 수행됨으로써, FF에서 스트림들은 변경될 수 있어, 하기의 송신된 스트림을 초래할 수 있다:

[0133] B7 0C 01 FF B0 87 1 D 00 7F FF FE 21 7C 01 FF EB 85 82 00 FF FF FE 18 7E

[0134] 이제 수신기가 스트림들을 분리하기 위하여 스트림들을 FF로 분리하여, 상기 프로세스를 반전시킨다:

[0135] Stream A: B7 0C 01 FF FE 21 7C 01 FF FE 18 7E

[0136] Stream B: B0 87 1 D 00 7F FF EB 85 82 00 FF FF

[0137] 각각의 스트림은 이제 단순한 산술 로직에 의하여 마커들을 제거하도록 프로세싱될 수 있다:

[0138] B = (b1|b2)&b4,

[0139] 스트림 A: B7 0C 01 FF FE 21 7C 01 FF FE 18 7E

[0140] 스트림 B: B0 87 1 D 00 7F FF EB 85 82 00 FF FF 는,

[0141] 스트림 A: B7 (0C|01)&FE 21 (7C|01)&FE 18 7E = B7 0C 21 7C 18 7E

[0142] 스트림 B: B0 87 1 D (00|7F)&EB 85 (82|00)&FF = B0 87 1 D 6B 85 82

[0143] 가 되고, 이것은 오리지널 스트림이다. 수신측상의 분리 동작은 임의의 비트 동작들 또는 가변 길이 코딩 없이 전체적으로 수행되었다.

[0144] 추가의 실시예들에 따라, 심볼 K는 미리 정의된 비트 위치에서 미리 정의된 0을 갖는 임의의 7 비트 수일 수 있다. 상기 설명은 미리 정의된 0의 위치를 공지함으로써 마커의 시작점을 발견하기 위하여 전개될 수 있다. 실시예들 중 하나에 따라, 0은 최하위 비트이고, 최상위 7개 비트가 다른 데이터, 예를 들어, 0 내지 127까지 증가하는 수에 대하여 사용될 수 있다. 상기 방식으로, 단지 바이트 스트림을 관찰함으로써 128개에 달하는 마킹된 코드워드들을 묵과하는 것이 용이할 것이다.

[0145] 본 발명의 일 양상에 따라, 상기 개시된 일반적인 방법은 JPEG 스트림들을 프로세싱하는데 사용하기 위하여 구현될 수 있다. DU의 끝을 나타내는 특별한 마커들을 포함하는 논-컴플라이언트(non-compliant) JPEG 데이터의 대안적 스트림을 생성하기 위하여 최소한의 변화로 현재 JPEG 인코더를 변경하는 방법이 하기에 개시된다. 논-컴플라이언트 JPEG 데이터는 그 후 각각의 DU의 시작을 맵핑하기 위하여 분석될 수 있고, 논-컴플라이언트 마커들은 비휘발성 메모리에 논-컴플라이언트 JPEG 데이터를 저장하는 동안 제거될 수 있다.

[0146] 추가로, 코드워드가 이전에 인코딩된 유닛을 생성하기 위하여 디코딩될 수 있는 일련의 비트들로서 이해되도록 의도된다는 것을 유념해야 한다. 모든 코드워드들이 동일한 길이는 아니다. 이러한 사실에도 불구하고, 코드워드들이 정확히 단일 심볼의 가변 길이 표시로서 판독됨을 의미하지는 않는다. 그러한 더 복잡한 코드워드들의 실시예는 jpeg 스트림에서 사용되는 1들이다. 본 명세서에서 정의된 바와 같은 JPEG 코드워드는 0이 아닌 수의 크기와 제로 런 렝쓰(zero run length)에 대한 심볼의 결합물로 0이 아닌 수를 수반하는, 크기에 기초하여 수의 최상위 비트들을 수반하는 일련의 0들을 인코딩한다.

[0147] 따라서, 0이 아닌 수를 수반하는 일련의 0들 또는 그들의 서브세트들 중 임의의 것에 대하여 생성되는 일련의 전체 비트들은 우리의 정의에 따라 코드워드로서 판독될 수 있다.

[0148] 코드워드로서 판독될 다른 서브세트들은 0이 아닌 수의 크기와 제로 런 렝쓰에 대한 심볼의 결합물, 또는 최상위 비트들을 나타내는 비트들일 것이다.

[0149] JPEG 스트림은 64개 계수들의 데이터 유닛들(DU들)로 구성된다. 계수들은 가변 길이의 허프만(Huffman) 코딩된 심볼들에 의하여 표현되고, 통상적으로 마지막 계수들은 0이고, EOB(End Of Block) 심볼은 끝에 0들을 갖는 데이터 유닛(DU)을 종결하는데 이용된다.

[0150] 몇몇 애플리케이션들에 대하여, DU들의 비트 위치들을 아는 것은 매우 유용하다.

[0151] 1) 작은 스크린상의 큰 JPEG 이미지의 인스턴트(instant) 디스플레이. DU 위치들이 공지되면, 블럭의 제1 계수만을 디코딩하고, 따라서, 오리지널 이미지의 스케일링 다운된(scaled down) 버전을 생성하는 것이 매우 신속하다.

[0152] 2) DU 위치들이 공지되면, 임의의 DU에 랜덤하게 액세스하는 것이 가능하다. 완전한 DC 계수들이 마찬가지로 공지되면, 이전 블럭들을 디코딩할 필요 없이 JPEG의 영역을 디코딩하는 것이 가능하다.

[0153] 방법은 블럭의 끝을 나타내는 마커들을 생성하기 위하여 하드웨어에 존재할 수 있고, SW 컴포넌트를 수신하는 것은 DU 위치들을 발견하기 위하여 사용할 수 있는 인코더 컴포넌트의 변형을 허용하고, 그 후에 용이하게 마커들을 제거하고, 따라서, 인코딩된 이미지의 스케일링 다운된 버전 및/또는 DU 위치 데이터베이스의 빠른 생성을

허용한다. 인코딩은 속도 문제로 인하여 HW에서 수행될 수 있다. 이것은 JPEG 데이터 스트림의 데이터 유닛들의 시작 위치들을 저장하기 위하여 보조 메모리를 요구하지 않고 표준 JPEG HW가 변형되도록 허용하기 위한 단순한 접근법이다. 보조 메모리는 2*number_of_data_units (16 bit) => 2mpix = 60000 DU's => 120kb 메모리일 필요가 있고, 이는 큰 실리콘 영역을 점유하는 0 대기 상태 메모리인 것이 바람직하므로, 매우 값비싸다.

[0154] 하기의 설명은 상기 JPEG 데이터의 인코딩 및 디코딩 모두를 커버한다.

[0155] 인코더:

[0156] 1) EOB가 항상 생성되도록 HW를 변형한다. 대안적으로, HW가 EOB를 항상 끝에 둠을 보장하기 위하여 독창적인 Q 테이블들을 사용한다. 0이 항상 보장된 계수 nr 64로 삽입된다면, 하드웨어는 EOB를 삽입할 필요가 있을 것이다.

[0157] 2) 오리지널 호프만 코딩된 스트림에서 발생하는 각각의 FF에 대하여 트레일링 "00"이 삽입되는 것을 확인하라. 이것은 유효 JPEG 스트림에서 이미 수행된다.

[0158] 3) 각각의 DU(또는 EOB) 이후에 적어도 16개의 이진수 1들을 제외하고 다수의 8을 부가한다.

[0159] 이것은 또한 EBO 심볼이 12개의 트레일링 이진수 1들과 함께 보통 2-4 바이트 심볼을 포함하기 위해 변형되고, DC 테이블들이 실제 DC 심볼 이전에 4개 1들을 포함하기 위해 변형되도록 허프만 테이블들을 변형함으로써 수행될 수 있다. 그 후, 인코더는 항상 비트 스트림에서 EOB들을 인코딩하도록 강제된다. 이것은 EOB와 DC 사이에서 항상 16개의 이진수 1들의 마커가 존재할 것임을 의미한다.

[0160] 본 실시예의 예시로서, 보통의(normal) EOB 및 새로운(new) EOB는 주어진 보통의 EOB에 대한 상기 논의된 조정의 관점에서, 하기와 같을 수 있다:

[0161] 보통 EOB: 1010

[0162] 새로운 EOB: 1010111111111111

[0163] 또한, 보통 DC들 및 새로운 Dc들은 하기의 테이블에서 개시되는 바와 같이 서로 관련될 수 있다.

[0164]

통상의 DCs:	새로운 DCs:
000	1111000
001	1111001
010	1111010
011	1111011
100	1111100
110	1111110
1110	11111110
11110	111111110
111110	1111111110
1111110	11111111110
11111110	111111111110
111111110	1111111111110

[0165] 이러한 방식은 상기 테이블의 제3 DC인 새로운 DC의 실시예 및 상기 EOB의 실시예를 사용하여, 제3 DC와 결합된 EOB가 "1010 1111111111111111 010"에 대응하는 결과를 초래한다. 따라서, 결과는 EOB 이후에 16개의 이진 1들을 부가할 때와 동일하다.

[0166] 방법의 일 실시예에 의하여 16개의 이진 1들을 부가하는 실시예가 하기에 주어지고, 하기의 실시예는 일반적인 허프만 코딩된 스트림으로 일반화된다.

[0167] 실시예는 하기의 허프만 테이블을 사용한다:

[0168] A = IO (EOB)

[0169] B = 110 C = 111

[0170] D = 0000

[0171] E = 0001

[0172] 하드웨어에서 인코딩된 심볼들 "CBCDBEDCABCBECB"의 허프만 코딩된 스트링은 하기의 비트 스트림을 초래한다:

[0173] 11111011 10000110 00010000 11110110 11111000 01111110 ...

[0174] FB 86 10 F6 F8 7E

[0175] 밑줄친 부분은 EOB를 나타낸다.

[0176] 다음 16개의 이진의 1들(binary ones)이 EOB 이후 삽입된다:

11111011 10000110 00010000 11110111 11111111 11111110 11111000 01111110 ...

[0177] FB 86 10 F7 FF FE F8 7E

[0178] 이제 디코딩의 예가 제시되며, 디코딩은 소프트웨어에서 수행될 수 있다.

[0179] 스텝 1. "00" 트레일링을 갖지 않는 FF 찾기(seek). JPEG 스트림에서 단지 "FF 00"만이 허용되나, 이러한 룰들(rules)에 따라 발생되지 않을 수 있다. 찾아낸 FF가 우리의 마커(our marker)라는 것을 100% 확신한다는 것을 의미한다. (재시작(restart) 마커들은 JPEG 스트림에 사용되지 않게 제공된다)

[0180] 이제, EOB 심볼(본 예에서는 10)을 향한 시크 다운(seek down) 및 다음 DU의 시작은 EOB 직후 발견된다.

[0181] 16개의 1들을 제거하고, 다음 DU에 대한 비트 어드레스를 기록한다(remember).

11111011 10000110 00010000 11110110 11111000 01111110 ...

[0182] FB 86 10 F7 FF FE F8 7E
FB 86 10 F6 F8 7E

[0183] 알 수 있듯이, 단지 1 바이트(byte)만이 변조되며, 2 바이트들은 제거되며, 나머지 바이트들은 미리 정확하게 워드(word) 정렬된다 :

[0184] FB 86 10 F7 FF FE F8 7E
FB 86 10 F6 F8 7E

[0185] 바람직하게(even better) 1들의 삽입된 숫자가 32인 경우, 나머지는 길게 워드 정렬되어 32 비트(bit) memcpy에 대해 허용된다. 이는 2 이상의 바이트들의 부가로 파일의 크기가 증가되어 네트워크의 버스에 대해 데이터를 이동할 때 대역폭 요구조건들(requirements)을 증가시키게 되어 항상 바람직한 것은 아닐 수 있다.

[0186] FB 86 10 F7 FF FF FF FE F8 7E
FB 86 10 F6 F8 7E

[0187] 1에 2바이트들을 결합하는 프로세스는 매우 간단하다:

[0188] FB 86 10 F7 FF FE F8 7E
FB 86 10 F6 F8 7E

[0189] F7 및 FE를 결합하는 연산(operation)은 매우 간단한 것으로, 이는 이들 중 논리(logical) "and" 연산과 마찬가지이며, 삽입된 16개의 1들이 비트(bit) 마스크로 작용하기 때문이다.

Byte1 = F7
Byte2 = FE
Joined Byte = Byte1 & Byte2

[0190]

[0191] 아래 예를 참조할 때 상기의 것을 보다 쉽게 볼 수 있다:

abcde111 11111111 11111fgh

Byte1 = abcde111
Byte2 = 11111fgh
JoinedByte = abcde111 & 11111fgh = abcdefgh

[0192]

[0193] 상기 예는 스트림에서 모든 DU들의 비트 어드레스들을 신속하게 발견하는 프로세스, 및 오리지널 스트림을 재생시키기 위해 마커들을 신속하게 제거하는 프로세스를 나타낸다.

[0194] 알고리즘은 각각의 DU의 DC 계수들을 발견하는데 이용하기 위해 변조될 수 있다. 다음, DC 계수들을 디코딩함으로써, 각각의 DC 데이터는 이미지의 8배 작은 버전을 재생성하는 데 이용될 수 있다. 4배 작은 표현(representation)이 요구되는 경우 동일한 것이 수행될 수 있고, 여기서 DC 계수 및 순차적(subsequent) AC의 넘버는 계수들에 대해 2×2 IDCT를 수행할 수 있도록 디코딩된다. 마커들이 DC 계수들의 위치를 표시함으로써, 다른 계수들을 디코딩하는 후프만(Huffman)을 스킵(skip)할 수 있게 되어, 캡쳐링 모멘트(capturing moment)에서 작은 스크린들에 대한 이미지들의 보다 신속한 스케일링이 허용된다. 이러한 구현은 감소된 크기의 이미지를 디스플레이하는 프로세스를 가속시키기 위한 임의의 애플리케이션에서 활용될 수 있다.

[0195] 이러한 애플리케이션의 일례로는 뷰 파인더(view finder)로서 디스플레이를 이용하는 이미지 습득(acquisition) 애플리케이션들이 있다. 이러한 애플리케이션들의 일반적 문제점 중 하나로는 이미지 뷰(image view)를 나타내는 이미지의 표현의 지연이 있다. 다른 말로, 사용자가 디스플레이 상의 원하는 뷰를 볼 때, 모멘트는 이미 사라져 버릴 수 있다. 이와 같은 애플리케이션에서, DC 계수들의 위치를 나타내는 마커들을 포함하는 데이터 스트림을 제공하고 마커들을 찾아내고, 마커들에서 DC 계수들에 관련된 정보를 검색하고(retrieve), 디스플레이상에 제시되는 정보로부터 감소된 사이즈의 이미지를 생성하는 디스플레이 프로세스를 구현함으로써, 실질적으로 지연이 감소될 수 있다.

[0196] 또한, 본 발명의 일 양상에 따라, DU의 마지막(end)을 나타내는 특정한 마커들을 포함하는 비준수(non-compliant) JPEG 데이터의 대안적 스트림(alternative stream)을 생성하기 위해 최소 변화들로 JPEG 인코더(encoder)를 변조시키는 방법이 제시된다. 다음 비준수 JPEG 데이터는 각각의 DU를 개시를 맵핑하도록 분석될 수 있고, 비준수 마커들은 비휘발성 메모리에 저장되는 동안 제거될 수 있다.

[0197] 상기 예들에서, 마커는 적어도 16개의 이진의 1들로 설정된다. 마커는 적어도 16개의 이진의 0들, 즉 0000000000000000로 설정된다. 마커가 16개의 이진의 0들로 설정되면, 마커 이전에 삽입된 인지된(known) 코드는 "11111111"일 수 있다. 따라서, 코드 000000001111111111111111는 111111110000000000000000로 반전된다.

[0198] 본 발명은 디지털 전자 회로, 또는 컴퓨터 하드웨어, 펌웨어, 소프트웨어 또는 이들의 조합들에서 구현될 수 있다. 본 발명의 장치는 프로그램가능한 프로세서에 의한 실행을 위해 기계-판독가능한 저장 디바이스에서 실체적으로 구체화되는 컴퓨터 프로그램 제품에서 구현될 수 있고; 본 발명의 방법 단계들은 입력 데이터를 동작시키고 출력 데이터를 생성함으로써 본 발명의 기능들을 수행하기 위해 명령들의 프로그램을 실행하는 프로그램가능한 프로세서에 의해 실행될 수 있다. 본 발명은 적어도 하나의 입력 디바이스에서, 데이터 저장 시스템으로부터 데이터 및 명령들을 수신하고, 데이터 저장 시스템으로 데이터 및 명령들을 전송하도록 결합된 적어도 하나의 프로그램가능 프로세서, 및 적어도 하나의 출력 디바이스를 포함하는 프로그램가능한 시스템상에서 실행될 수 있는 하나 이상의 컴퓨터 프로그램들에서 바람직하게 구현될 수 있다. 각각의 컴퓨터 프로그램은 고급 절차형(high-level procedural) 또는 객체-지향형(object-oriented) 프로그래밍 언어, 또는 원할 경우 어셈블리 또는 기계 언어에서 구현될 수 있으며; 임의의 경우, 상기 언어는 컴파일드(compiled) 또는 인터프리터(interpreted) 언어일 수 있다. 예를 들어, 적절한 프로세서들로는 범용성 및 특정 용도의 마이크로프로세서들이 포함된다. 일반적으로, 프로세서는 판독-전용 메모리 및/또는 랜덤 액세스 메모리로부터의 명령들 및 데이

터를 수신한다. 일반적으로, 컴퓨터는 데이터 파일들을 저장하기 위한 하나 이상의 대용량(mass) 저장 디바이스들을 포함하며; 이러한 디바이스들은 내부 하드 디스크들 및 이동식 디스크들과 같은 자기(magnetic) 디스크들, 광자기(magneto-optical) 디스크들 및 광학 디스크들을 포함한다. 컴퓨터 프로그램 명령들 및 데이터를 실체적으로 구체화하기에 적합한 저장 디바이스들은 예를 들어 EPROM, EEPROM 및 플래시 메모리 디바이스들과 같은 반도체 메모리 디바이스들; 내부 하드 디스크들 및 이동식 디스크들과 같은 자기 디스크들; 광자기 디스크들; 및 CD-ROM 디스크들을 포함하는 모든 형태의 비휘발성 메모리를 포함한다. 언급된 임의의 것은 ASIC(application-specific integrated circuit)들에 의해 보완되거나 또는 ASIC들에 통합될 수 있다.

[0199] 사용자와의 인터랙션을 제공하기 위해, 본 발명은 사용자에게 정보를 디스플레이하기 위한 모니터 또는 LCD 스크린과 같은 디스플레이 디바이스를 갖는 컴퓨터 시스템상에서 구현될 수 있다. 사용자는 키보드 및 포인팅 디바이스, 이를 테면 마우스, 트랙볼, 마이크로폰, 터치-감지 디스플레이, 트랜듀서 카드 판독기, 자기 또는 페이퍼 테입 판독기, 테이블렛, 스타일러스(stylus), 음성(voice) 또는 수기 인식기, 또는 물론 다른 다른 컴퓨터들과 같은 임의의 다른 공지된 입력 디바이스를 통해 컴퓨터 시스템에 입력을 제공할 수 있다. 컴퓨터 시스템은 컴퓨터 프로그램들이 사용자들과 인터랙팅하는 그래픽 사용자 인터페이스를 제공하도록 프로그램될 수 있다.

[0200] 마지막으로, 프로세서는 프로세서가 네트워크로부터 정보를 수신하거나 또는 앞서 개시된 방법 단계들을 수행하는 과정에서 네트워크로 정보를 출력할 수 있는 네트워크 접속을 이용하여 컴퓨터 또는 텔레통신 네트워크, 예를 들면 인터넷 네트워크, 또는 인트라넷 네트워크와 접속된다. 프로세서를 이용하여 실행되는 명령들의 시퀀스로서 종종 표현되는 이러한 정보는 예를 들어, 반송파(carrier wave)로 구체화되는 컴퓨터 데이터 신호 형태로 네트워크로부터 수신되고 네트워크로 출력된다. 앞서 개시된 디바이스들 및 물질들은 컴퓨터 하드웨어 및 소프트웨어 기술들에서의 당업자들에게 친숙할 것이다.

[0201] 본 발명은 컴퓨터 시스템들에 저장된 데이터를 호출하는(involving) 다양한 컴퓨터-구현 동작들을 이용한다는 것이 주목된다. 제한되는 것은 아니지만, 이러한 동작들은 물리적 양들의 물리적 조작(physical manipulation)을 요구하는 것들이다. 통상적으로, 필수적이지는 않지만, 이러한 양들은 저장될 수 있는, 전송될 수 있는, 조합될 수 있는, 비교될 수 있는 그리고 다르게 조작될 수 있는 전기적 또는 자기적 신호들의 형태를 취한다. 본 발명의 일부를 형성하는 본 발명에 개시된 동작들은 기계적 동작들에 유용한다. 수행되는 조작들은 종종 이를 테면, 생성(producing), 식별(identifying), 실행(running), 결정(determining), 비교(comparing), 실행(executing), 다운로딩(downloading) 또는 검출(detecting)과 관련하여 언급된다. 때로는 공통적 사용을 위해 편의상 원칙적으로 상기 전기적 또는 자기적 신호들이 비트들(bits), 값들(values), 엘리먼트들(elements), 변수들(variables), 문자들(characters), 데이터 또는 이와 유사한 것으로서 언급된다. 그러나, 이러한 및 유사한 모든 항목들은 적절한 물리적 양들과 연관되며 단지 이러한 양들에 편의상 적용되는 라벨들이라는 것을 주목해야 한다.

[0202] 또한, 본 발명은 앞서 언급된 동작들을 수행하기 위한 디바이스, 시스템 또는 장치에 관한 것이다. 시스템은 요구되는 목적을 위해 특별하게 구성되거나, 또는 컴퓨터에 저장된 컴퓨터 프로그램에 의해 선택적으로 활성화되거나 구성되는 범용성 컴퓨터일 수 있다. 앞서 제시된 프로세서들은 본질적으로 임의의 특정 컴퓨터 또는 다른 컴퓨팅 장치와 관련되는 것은 아니다. 특히, 다양한 범용성 컴퓨터들은 본 발명의 설명에 따라 기록된 프로그램들을 이용할 수 있고, 혹은 대안적으로, 요구되는 동작들을 수행하기 위해 보다 특정화된 컴퓨터 시스템을 구성하는 것이 편리할 수 있다.

[0203] 본 발명의 다양한 구현예들이 개시되었다. 그럼에도 불구하고, 본 발명의 범주 및 사상을 이탈하지 않고 다양한 변조들이 이루어질 수 있다는 것이 이해되 것이다.