(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0102493
(43) 공개일자 2010년09월24일

(51) Int. Cl.
*H04N 7/24* (2006.01)
(21) 출원번호 10-2009-0020918
(22) 출원일자 2009년03월11일
심사청구일자 없음

(71) 출원인
**경희대학교 산학협력단**
경기도 용인시 기흥구 서천동 1 경희대학교 국제캠퍼스내
(72) 발명자
**박광훈**
경기도 성남시 분당구 분당동 45번지, 동아빌라 B동-302호
**김경용**
전라남도 영광군 영광읍 남천리 349-2
(74) 대리인
**특허법인 신지, 천성훈, 유경열, 박진석**

전체 청구항 수 : 총 26 항

# (54) 블록 기반 적응적 비트플레인 코딩을 이용한 깊이 정보맵 코딩 방법 및 장치

## (57) 요 약

### 1. 청구범위에 기재된 발명이 속한 기술분야

본 발명은 깊이 정보맵(depth map) 코딩 시, 깊이 정보의 화질 및 코딩 효율을 향상 시키기 위한 방법 및 장치에 관한 것임.

### 2. 발명이 해결하려고 하는 기술적 과제

본 발명에서는 깊이 정보맵 코딩 시, 깊이 정보맵의 특성을 최대한 이용함으로써 깊이 정보맵의 화질 및 코딩 효율을 향상 시키는 방법을 제안한다.

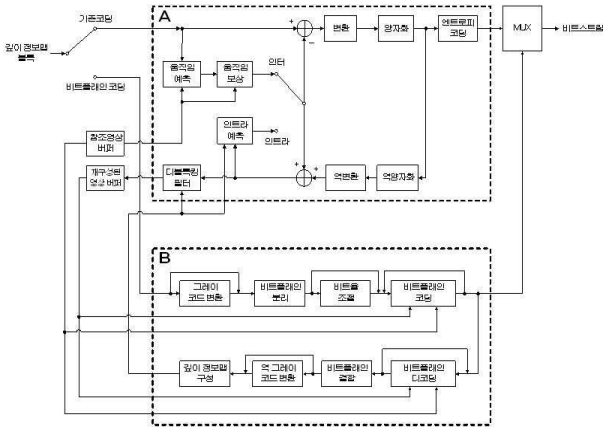
### 3. 발명의 해결방법의 요지

본 발명은 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법 및 장치를 이용하여 깊이 정보맵의 화질 및 코딩 효율을 향상시키는 방법을 포함함.

### 4. 발명의 중요한 용도

본 발명은 깊이 정보맵에 대한 코딩 효율 향상 및 화질 향상에 이용된다.

*대 표 도* - 도8



이 발명을 지원한 국가연구개발사업
과제고유번호 R0A-2005-000-10061-0
부처명 한국과학재단
연구관리전문기관
연구사업명 국가지정연구실사업
연구과제명 실감방송을 위한 유니버셜 스케일러블 비디오 코딩기술
기여율
주관기관 경희대학교산학협력단
연구기간 2005년 04월 01일 ~ 2010년 03월 31일

## *특허청구의 범위*

**청구항 1**

블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 인코딩하는 방법 및 장치.

**청구항 2**

제1항에 있어서, 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법 중 최적의 모드를 선택하기 위한 모드 선택 방법.

**청구항 3**

깊이 정보맵을 인코딩 하는데 있어서, 원하는 비트율을 얻기 위한 비트율 조절 단계; 깊이 정보맵을 비트플래인 단위로 분리하는 단계; 깊이 정보맵 블록 각각의 픽셀을 그레이 코드로 변환하는 단계; 비트플래인을 인코딩하는 단계; 및 전체 블록을 인코딩하는 전체 인코딩 단계를 포함하는 인코딩 방법 및 장치.

**청구항 4**

제3항에 있어서, 깊이 정보맵을 MxN픽셀 크기의 블록 단위로 나누어서 각각의 블록을 비트플래인 인코딩하는 방법 및 장치.

**청구항 5**

제3항에 있어서, 깊이 정보맵을 인코딩하는 방법에 있어서 임의의 블록 단위로 코딩할 비트플래인의 수를 결정하는 방법을 이용한 비트율 조절 방법.

**청구항 6**

제5항에 있어서, 깊이 정보맵 블록 혹은 분리된 비트플래인의 이진 블록에 다운샘플링을 적용하여 비트율을 조절하는 방법, 깊이 정보맵 블록을 비트플래인 단위로 분리한 후 분리된 비트플래인의 이진 블록들 중 특정 비트플래인의 이진 블록을 코딩하지 않는 방법, 방법, 상기 방법들을 조합하여 비트율을 조절하는 방법.

**청구항 7**

제3항에 있어서, 깊이 정보맵 블록의 비트플래인 블록을 인코딩하는 방법에 있어서, 참조 영상 및 재구성된 영상의 각각의 픽셀에 그레이 코드 변환을 수행하는 단계; 참조영상 및 재구성된 영상에 대한 비트플래인 분리 단계; 선택된 비트플래인 블록의 움직임을 예측하는 단계; 예측된 움직임을 통해 선택된 비트플래인 블록의 움직임을 보상하는 단계; 선택된 비트플래인 블록의 모드를 결정하는 단계; 선택된 비트플래인 블록을 CAE인코딩하는 단계; 선택된 비트플래인 블록의 코딩 정보들을 모아서 비트스트림을 생성하는 단계; 및 전체 비트플래인 블록을 인코딩하는 전체 인코딩 단계를 포함하는 인코딩 방법 및 장치.

**청구항 8**

제7항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록의 모드를 결정하는 방법에 있어서 인트라 픽쳐 모드 결정 방법과 인터 픽쳐 모드 결정 방법 및 결정된 비트플래인 블록의 모드를 인코딩하는 방법.

**청구항 9**

제7항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록의 움직임 벡터의 예측값을 선택된 비트플래인 블록에 인접하는 주변 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 정보를 이용하여 결정하는 방법.

**청구항 10**

제7항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE인코딩하는데 있어서, 선택된 비트플래인 블록에 대한 비트율 조절 단계; 참조 비트플래인 블록의 다운샘플링 단계; 재구성된 비트플래인 영상과 참조 비트플래인 블록 혹은 다운샘플링된 참조 비트플래인 블록을 이용한 컨텍스트 계산 단계; 확률 테이블을 이용한 산술 인코딩 단계; 및 전체 비트플래인 블록을 인코딩하는 전체 인코딩 단계를 포함하는 인코딩 방법 및 장치.

**청구항 11**

제10항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE인코딩하는데 있어서, 선택된 비트플래인 블록에 대한 변환 비율을 결정하는 방법.

**청구항 12**

제10항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE인코딩하는데 있어서, 선택된 비트플래인 블록의 모드에 따라 컨텍스트 템플릿을 구성하는 방법.

**청구항 13**

제10항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE인코딩하는데 있어서, 선택된 비트플래인 블록의 픽셀을 산술 인코딩하는데 있어서, 선택된 비트플래인 블록의 픽셀을 스캔하는 방법.

**청구항 14**

제10항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 런 길이 코딩 방법과 가변 길이 코딩 방법을 이용한 비트플래인 코딩 방법.

**청구항 15**

제10항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 컨텍스트 기반 적응적 산술 인코딩을 응용한 비트플래인 코딩 방법.

**청구항 16**

제10항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAC혹은 JBIG의 방법 혹은 Quad Tree 방법을 이용한 비트플래인 코딩 방법.

**청구항 17**

블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 디코딩하는 방법 및 장치.

**청구항 18**

깊이 정보맵 블록을 디코딩 하는데 있어서, 비트플래인을 디코딩하는 단계; 깊이 정보맵 블록의 각각의 픽셀을 그레이 코드로 변환하는 단계; 각각의 비트플래인을 결합하는 단계; 깊이 정보맵을 구성하는 단계; 및 전체 블록을 디코딩하는 전체 디코딩 단계를 포함하는 디코딩 방법 및 장치.

**청구항 19**

제18항에 있어서, 깊이 정보맵의 비트플래인의 블록을 디코딩 하는데 있어서, 참조 영상 및 재구성된 영상의 각각의 픽셀에 그레이 코드 변환을 수행하는 단계; 참조영상 및 재구성된 영상에 대한 비트플래인 분리 단계; 선택된 비트플래인 블록의 디코딩 정보들을 분리해주는 단계; 선택된 비트플래인 블록의 움직임을 보상하는 단계; 선택된 비트플래인 블록에 대한 동일 레벨 블록 디코딩하는 단계; 선택된 비트플래인 블록을 CAE디코딩하는 단계; 및 전체 비트플래인 블록을 디코딩하는 전체 디코딩 단계를 포함하는 디코딩 방법 및 장치.

**청구항 20**

제19항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록의 움직임 벡터의 예측값을 선택된 비트플래인 블록에 인접하는 주변 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 정보를 이용하여 결정하는 방법.

**청구항 21**

제19항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE디코딩하는데 있어서, 선택된 비트플래인 블록에 대한 변환 비율 디코딩 단계; 참조 비트플래인 블록의 다운샘플링 단계; 재구성된 비트플래인 영상과 참조 비트플래인 블록 혹은 다운샘플링된 참조 비트플래인 블록을 이용한 컨텍스트 계산 단계; 확률 테이블을 이용한 산술 디코딩 단계; 디코딩된 비트플래인 블록에 대한 업샘플링 단계; 및 전체 비트플래인 블록을 디코딩하

는 전체 디코딩 단계를 포함하는 디코딩 방법 및 장치.

**청구항 22**

제21항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE디코딩하는데 있어서, 선택된 비트플래인 블록의 모드에 따라 컨텍스트 템플릿을 구성하는 방법.

**청구항 23**

제21항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAE디코딩하는데 있어서, 선택된 비트플래인 블록의 픽셀을 산술 디코딩하는데 있어서, 선택된 비트플래인 블록의 픽셀을 스캔하는 방법.

**청구항 24**

제21항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 런 길이 디코딩 방법과 가변 길이 디코딩 방법을 이용한 비트플래인 디코딩 방법.

**청구항 25**

제21항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 컨텍스트 기반 적응적 산술 디코딩을 응용한 비트플래인 디코딩 방법.

**청구항 26**

제21항에 있어서, 깊이 정보맵 블록의 선택된 비트플래인 블록을 CAC혹은 JBIG의 방법 혹은 Quad Tree 방법을 이용한 비트플래인 코딩 방법.

# *명 세 서*

## *발명의 상세한 설명*

### *기 술 분 야*

[0001] 본 발명은 깊이 정보맵 코딩에 관한 것으로, 보다 구체적으로 깊이 정보맵(depth map) 코딩 시, 깊이 정보의 화질 및 코딩 효율을 향상 시키기 위한 방법 및 장치에 관한 것이다.

### *배 경 기 술*

[0002] 실감 미디어의 관심이 대폭적으로 증가하면서 또한 그에 대한 연구가 활발히 진행되고 있다. 이와 관련된 연구로써 JVT(Joint Video Team of ISO/IEC JTC1/SC29/WG11 MPEG and ITU-T SG16 Q.6 VCEG)에서는 2008년 7월 복수의 카메라로부터 입력 받은 여러 뷰(multiple view)의 영상을 효율적으로 코딩하여 사용자에게 전달하기 위한 다시점 비디오 코딩(Multi-view Video Coding, H.264/AVC Amendment 4) 표준을 완료하였고, 또한 현재 MPEG에서는 3D Video에 대한 표준이 진행 중이다. 3D Video는 N개의 시점 영상과 N개의 깊이 정보맵(depth map)을 함께 전송하여 전송 받은 시점 영상과 깊이 정보맵을 이용한 뷰 인터폴레이션(view interpolation)을 수행함으로 전송 받은 시점 영상보다 더 많은 시점의 영상을 생성할 수 있도록 하는 방법이다. 3D Video는 3차원 디스플레이 시스템을 지원할 수 있으며, 대표적인 시스템의 예로 FTV(Free View-point TV)를 들 수 있다.

[0003] 다시점 비디오 코딩과 3D Video 표준 모두 다양한 시점을 제공함으로써 사용자에게 현장감을 전해줄 수 있는 기술이다. 하지만 다시점 비디오 코딩은 고정된 개수 카메라로부터 입력 받은 고정된 수의 시점만을 사용자에게 보여줄 수 있을 뿐이다. 다시점 비디오 코딩에서는 좀 더 많은 시점을 지원하기 위해서는 더욱 많은 카메라를 사용해야 하며, 또한 카메라 수만큼의 시점 데이터를 전송해야 한다. 하지만 현존하는 전달미디어(방송, 통신, 저장 미디어)의 전송 대역폭과 저장 용량에는 한계가 존재하기 때문에 많은 뷰의 시점 데이터를 전달하는 데는 한계가 따른다.

[0004] 이러한 문제점을 보완할 수 있는 방법으로 3D Video 표준이 대두되었는데, 3D Video는 다시점 비디오 코딩과 달리 카메라로부터 입력 받은 시점 이외에 깊이 정보맵(depth map)을 함께 전송하여 입력 받은 시점 이외에 다양한 시점을 깊이 정보맵을 이용해 사용자가 원하는 시점을 무한대까지 생성할 수 있도록 지원한다. 따라서 3D

Video에서는 다시점 비디오 코딩 방법과 같이 많은 수의 영상 시점을 모두 전송할 필요가 없고, 소수의 시점 영상과 깊이 정보맵만을 전송하면 되기 때문에 대역폭과 저장공간을 절약할 수 있다는 장점이 있다.

[0005] 3D Video에서는 입력 받은 시점 영상을 인코딩/디코딩하는 과정 이외에 깊이 정보맵을 생성하는 과정, 깊이 정보맵을 인코딩/디코딩하는 과정, 깊이 정보맵을 이용하여 가상의 시점 영상을 생성하는 방법이 추가적으로 필요하다. 따라서 현재 3D Video 표준에서는 주로 깊이 정보맵을 생성하는 방법과 깊이 정보맵을 이용하여 가상의 시점 영상을 생성하는 방법에 대한 연구를 수행하고 있다. 하지만, 깊이 정보맵을 코딩하는 방법에 대해서는 MPEG-C Part 3에서 3D 영상을 깊이 정보을 이용하여 랜더링하기 위해 픽쳐 내의 실 세계에서 가장 가까이 있는 객체의 평면 공간의 위치와 실 세계에서 가장 멀리 있는 객체의 평면 공간의 위치를 보내기 위한 파라미터(parameter)의 코딩 방법만 정의하고 있을 뿐 아직 깊이 정보맵 자체의 코딩 방법은 정의하지 않고 있다. 따라서 깊이 정보맵 코딩 방법에 대한 연구가 필요하다고 할 수 있다.

[0006] 깊이 정보맵이란, 현재 시점에서 카메라와 실제 사물(object)과의 거리를 시점 영상과 동일한 해상도로 각 픽셀에 해당하는 깊이 정보를 일정한 비트수로 표현하는 것이다. 깊이 정보맵의 일 예로 도 1은 MPEG의 다시점 비디오 코딩의 테스트 시퀀스인 "Breakdancers" 영상 시퀀스(도 1 왼쪽 영상)의 깊이 정보맵(도 1 오른쪽 영상)을 보여주고 있다. 실제 도 1의 깊이 정보맵은 각 픽셀에 대응되는 깊이 정보를 8비트로 표현한 것으로 카메라와 가까울수록 큰 값(밝은 값)으로 표현된 것이다.

[0007] 도 1의 깊이 정보맵을 이용하여 각 픽셀에서 실 세계에서의 거리(Z)를 구하는 방법의 일 예는 아래 수학식 1과 같다.

*수학식 1*

$$Z = Z_{far} + v \cdot \frac{Z_{near} - Z_{far}}{255} \quad \text{with } v \in [0, \ldots, 255]$$

[0008]

[0009] v는 도 1의 깊이 정보맵에서 실제 표현되는 깊이 정보값이고, $Z_{far}$와 $Z_{near}$는 실제 MPEG-C Part 3에서 정의하는 파라미터로써 영상 안에서 보여지는 실 세계에서의 가장 먼 부분($Z_{far}$)과 가까운 부분($Z_{near}$)의 실제 위치를 나타낸다. 따라서 깊이 정보맵에 표현되는 깊이 정보는 영상에서 보여지는 실 세계에서의 가장 먼 부분과 가까운 부분을 $2^n$ (n: 깊이 정보맵을 표현하는 비트수, 도 1의 깊이 정보맵과 상기 수식에서는 n=8) 등분한 것을 표현한 것이다.

[0010] 현재 깊이 정보맵의 코딩 방법으로 H.264/AVC나 다시점 비디오 코딩과 같은 기존의 표준 동영상 코딩 방법을 그대로 사용하는 것이 일반적이며, 기존의 표준 동영상 코딩 방법인 H.264/AVC의 인코더 구조도의 일 예는 도 2와 같다.

[0011] 도 2의 H.264/AVC 인코더 구조도에서 데이터를 처리하는 단위는 가로 세로 16x16픽셀 크기의 매크로블록(Macroblock)이며, 영상을 입력 받아 인트라 모드 또는 인터 모드로 인코딩이 수행되어 비트스트림을 출력한다. 인트라(Intra) 모드일 경우, 스위치가 인트라로 전환이 되며, 인터(Inter) 모드일 경우에는 스위치가 인터로 전환이 된다. 코딩 과정의 주요한 흐름은 먼저 입력된 블록 영상에 대한 예측 블록을 생성한 후, 입력된 블록과 예측 블록의 차분을 구해 그 차분을 코딩하는 것이다. 먼저 예측 블록의 생성은 인트라 모드와 인터 모드에 따라 수행이 된다. 먼저 인트라 모드일 경우에는 "인트라 예측" 과정에서 현재 블록의 이미 코딩된 주변 픽셀값을 이용하여 공간적 예측을 수행하여 예측 블록을 생성하며, 인터 모드일 경우에는 "움직임 예측" 과정에서 참조 영상 버퍼에 저장되어 있는 참조 영상에서 현재 입력된 블록과 가장 매치가 잘 되는 영역을 찾아 움직임 벡터(Motion Vector)를 구한 후, 구한 움직임 벡터를 이용하여 "움직임 보상"을 수행함으로써 예측 블록을 생성한다. 상기 설명한 것과 같이 현재 입력된 블록과 예측 블록의 차분을 구하여 잔여 블록(Residual Block)을 생성한 후, 이에 대한 코딩을 수행한다. 잔여 블록에 대한 코딩은 "변환", "양자화", "엔트로피 코딩"의 순서로 수행이 된다. 먼저 "변환" 과정에서는 입력된 잔여 블록을 입력 받아 변환(Transform)을 수행하여 변환계수(Transform Coefficient)를 출력한다. 그리고 "양자화" 과정에서는 입력된 변환계수를 양자화 파라미터에 따라 양자화를 수행한 양자화된 계수(Quantized Coefficient)를 출력한다. 그리고 "엔트로피 코딩" 과정에서는 입력된 양자화된 계수를 확률 분포에 따른 엔트로피 코딩을 수행하여 비트스트림으로 출력된다. H.264/AVC는 프레임간(Inter-frame) 예측 코딩을 수행하기 때문에 현재 코딩된 영상을 이 후에 입력된 영상의 참조 영상으로 사용하기 위해 디코딩하여 저장할 필요가 있다. 따라서 양자화된 계수를 "역양자화"과정과 "역변환"을 수행하여 예측 영상과 가산기를 통해 재구성된 블록을 생성한 다음 "디블록킹 필터"를 통해 코딩 과정에서 발생한 블록킹

현상(Blocking Artifact)을 제거한 후, "참조 영상 버퍼"에 저장한다.

[0012] 기존의 표준 동영상 코딩 방법인 H.264/AVC의 디코더 구조도의 일 예는 도 3과 같다.

[0013] 도 3의 H.264/AVC 디코더 구조도에서 데이터를 처리하는 단위는 가로 세로 16x16픽셀 크기의 매크로블록(Macroblock)이며, 비트스트림을 입력 받아 인트라 모드 또는 인터 모드로 디코딩이 수행되어 재구성된 영상을 출력한다. 인트라(Intra) 모드일 경우, 스위치가 인트라로 전환이 되며, 인터(Inter) 모드일 경우에는 스위치가 인터로 전환이 된다. 디코딩 과정의 주요한 흐름은 먼저 예측 블록을 생성한 후, 입력 받은 비트스트림을 디코딩한 결과 블록과 예측 블록을 더하여 재구성된 블록을 생성하는 것이다. 먼저 예측 블록의 생성은 인트라 모드와 인터 모드에 따라 수행이 된다. 먼저 인트라 모드일 경우에는 "인트라 예측" 과정에서 현재 블록의 이미 코딩된 주변 픽셀값을 이용하여 공간적 예측을 수행하여 예측 블록을 생성하며, 인터 모드일 경우에는 움직임 벡터를 이용하여 참조 영상 버퍼에 저장되어 있는 참조 영상에서 영역을 찾아 "움직임 보상"을 수행함으로써 예측 블록을 생성한다. "엔트로피 디코딩" 과정에서는 입력된 비트스트림을 확률 분포에 따른 엔트로피 디코딩을 수행하여 양자화된 계수(Quantized Coefficient)를 출력한다. 양자화된 계수를 "역양자화"과정과 "역변환"을 수행하여 예측 영상과 가산기를 통해 재구성된 블록을 생성한 다음 "디블록킹 필터"를 통해 블록킹 현상(Blocking Artifact)를 제거한 후, "참조 영상 버퍼"에 저장한다.

[0014] 이러한 동영상 표준 코덱을 사용함으로 높은 압축률을 얻을 수 있지만, 이는 실제 일반 영상의 특성에 맞도록 설계된 코딩 방법으로 깊이 정보맵의 특성을 최대한 사용하지 못한다는 단점이 있다. 보통 깊이 정보맵을 사용하여 영상을 합성하는 장치에서는 깊이 정보맵의 정확도에 따라 화질에 큰 영향을 미치기 때문에, 깊이 정보맵의 특성을 이용하여 최적의 효율을 얻을 수 있는 코딩 방법이 필요하다.

## *발명의 내용*

### *해결 하고자하는 과제*

[0015] 본 발명에서는 깊이 정보맵 코딩 시, 깊이 정보맵의 특성을 최대한 이용함으로써 깊이 정보맵의 화질 및 코딩 효율을 향상 시키는 방법을 제안한다.

### *과제 해결수단*

[0016] 본 발명은 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법 및 장치를 이용하여 깊이 정보맵의 화질 및 코딩 효율을 향상시키는 방법을 포함한다.

### *효 과*

[0017] 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법을 이용하여 화질 및 코딩효율을 향상시키는 효과가 있다.

### *발명의 실시를 위한 구체적인 내용*

[0018] 이하, 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세하게 설명한다. 본 발명의 실시예를 설명함에 있어 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 또한, 후술되는 용어들은 본 발명의 실시예에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

[0019] 깊이 정보맵은 일반적인 실사 영상과는 다르게 상당히 완만한 특성을 갖는데, 도 4를 통해서 쉽게 그 특성을 알 수 있다. 도 4는 도 1의 실제 영상과 깊이 정보맵의 각 픽셀의 레벨(실제 영상에서는 휘도(luminance) 성분의

레벨 즉 밝기를 뜻하고, 깊이 정보맵에서는 깊이의 레벨을 뜻한다)을 표현한 3차원 그래프인데, 실제 영상의 그래프(도 4 왼쪽 그래프)에서는 픽셀 사이에 변화가 심한 형태를 보여주고 있음을 확인할 수 있고, 반면 깊이 정보맵의 그래프(도 4 오른쪽 그래프)는 상당히 완만한 형태를 보여주고 있음을 확인할 수 있다.

[0020] 그리고 또한 깊이 정보맵을 비트플래인(bit-plane) 단위로 표현할 수 있는데, 깊이 정보맵의 비트플래인은 실사 영상의 비트플래인에 비해 상당히 단조로운 형태를 보여준다. 도 5 도 1의 실제 영상과 깊이 정보맵을 비트플래인 단위로 표현한 모습을 보여준다. 실제 영상의 비트플래인(도 5 왼쪽)은 최상위 비트플래인(MSBP; most significant bit-plane)은 단조로운 형태 정보를 가지지만 하위 비트플래인으로 갈수록 상당히 복잡해지며, 최하위 비트플래인(LSBP; least significant bit-plane)부터 세 개의 비트플래인은 거의 백색 잡음(white noise) 형태로 상당히 복잡함을 알 수 있다. 하지만 깊이 정보맵의 비트플래인(도 5 가운데)은 최상위 비트플래인부터 최하위 비트플래인으로 갈수록 복잡해지긴 하나, 전체 비트플래인이 실제 영상의 비트플래인에 비해 상당히 단조로운 형태를 보임을 알 수 있다. 또한 깊이 정보맵의 각 비트플래인 모습은 어떤 일정한 형태를 유지하고 있음을 확인할 수 있다. 그리고 보통 비트플래인을 표현할 때 그레이 코드(gray code)를 적용하면 각 비트플래인 단위에서 주변 비트값의 유사도가 증가하게 되는데 깊이 정보맵의 각 레벨 값에 그레이 코드를 적용한 비트플래인의 모습은 도 5 오른쪽과 같고, 더욱 단조로운 형태를 보이는 것을 쉽게 확인할 수 있다.

[0021] 일 예로 도 6은 Ballet 영상의 깊이 정보맵에서 객체 경계 부분 매크로블록의 비트플래인 영상을 분석한 것으로써, 실제 매크로블록과, 실제 매크로블록을 비트플래인으로 분리한 결과와, 실제 매크로블록에 그레이 코드를 적용한 후 비트플래인으로 분리한 결과를 보여준다.

[0022] 도 6의 실제 깊이 정보맵의 매크로블록을 비트플래인 단위로 분리한 이진 영상들을 살펴보면, MSB 비트플래인에서부터 LSB 비트플래인까지의 이진 영상이 단조로운 형태로 되어 있는 것을 알 수 있다. 특히 MSB 비트플래인과 MSB-1 비트플래인 그리고 MSB-3 비트플래인은 이진 영상이 완전히 일치하며, 또한 MSB-2 비트플래인과 MSB-4 비트플래인도 이진 영상이 완전히 일치한다는 것을 알 수 있다. 그리고 MSB, MSB-1, MSB-3 비트플래인의 이진 영상을 MSB-2, MSB-4 비트플래인의 이진 영상과 비교해보면 완전히 반전되게 일치한다는 것을 알 수 있으며, 또한 MSB-5 비트플래인의 이진 영상을 MSB-6 비트플래인의 이진 영상과 비교해보면 완전히 반전되게 일치한다는 것을 알 수 있다. 그리고 LSB 비트플래인의 이진 영상은 다른 어떠한 비트플래인의 이진 영상과도 일치하지 않고 전혀 다른 형태를 가진다는 것을 알 수 있다.

[0023] 따라서, 도 6의 실제 깊이 정보맵의 매크로블록에 그레이 코드를 적용한 후 비트플래인 단위로 분리하게 되면, 현재 비트플래인의 이진 영상과 현재 비트플래인보다 한 단계 상위 비트플래인의 이진 영상이 완전히 일치하는 경우에는 현재 비트플래인의 이진 영상은 모두 '0(0)'이 되며, 현재 비트플래인의 이진 영상과 현재 비트플래인보다 한 단계 상위 비트플래인의 이진 영상이 완전히 반전되게 일치하는 경우에는 현재 비트플래인의 이진 영상은 모두 '255(1)'가 된다. 그 결과 도 6의 실제 깊이 정보맵의 매크로블록에 그레이 코드를 적용한 후 비트플래인 단위로 분리한 이진 영상들을 살펴보면, MSB-1 비트플래인의 이진 영상은 모두 '0(0)'이 되며, MSB-2 비트플래인, MSB-3 비트플래인, MSB-4 비트플래인, MSB-6 비트플래인의 이진 영상은 모두 '255(1)'가 된다.

[0024] 이와 같은 특성을 이용하여 깊이 정보맵에 그레이 코드를 적용하여 비트플래인 단위로 코딩할 수 있는데 도 6의 실제 깊이 정보맵의 매크로블록에 그레이 코드를 적용한 후 비트플래인 코딩 시, MSB 비트플래인과 MSB-5 비트플래인 그리고 LSB 비트플래인의 이진 영상의 경우에은 이진 영상 압축 방법을 이용하여 코딩하고 그 이외의 비트플래인의 이진 영상의 경우에는 현재 이진 영상이 모두 '0(0)'인지 아니면 모두 '255(1)'인지에 대한 정보만 코딩하면 된다. 이러한 비트플래인 코딩 방법은 객체 경계 부분에서 코딩 효율이 좋지 못한 DCT 기반 동영상 압축 방법보다 상당히 월등한 코딩 효율을 가질 수 있다.

[0025] 깊이 정보맵을 그레이 코드로 변환했을 때 정보의 손실이 발생을 한다면 그레이 코드를 기존의 깊이 정보맵으로 복원할 때, 정확한 값을 복원할 수 없다. 따라서 "비트플래인 코딩" 과정에서 비트량 조절을 위해 정보의 손실을 허용하기 위해서는 MSB 비트플래인부터 LSB 비트플래인의 순서로 차례대로 코딩할 비트플래인의 수를 증가시키면서 코딩을 수행해야 한다.

[0026] 또 다른 일 예로 도 7은 Ballet 영상의 깊이 정보맵에서 배경 부분 매크로블록의 비트플래인 영상을 분석한 것으로써, 실제 매크로블록과, 실제 매크로블록을 비트플래인으로 분리한 결과와, 실제 매크로블록에 그레이 코드를 적용한 후 비트플래인으로 분리한 결과를 보여준다.

[0027] 도 7의 실제 깊이 정보맵의 매크로블록을 비트플래인 단위로 분리한 이진 영상들을 살펴보면, 이진 영상들이 단조롭지만 도 6에서의 비트플래인 단위로 분리한 이진 영상들 보다 대체적으로 복잡한 형태를 가진다는 것을 알

수 있다. 그리고 MSB 비트플래인과 MSB-1 비트플래인은 모두 '0(0)'이고, MSB-2 비트플래인은 모두 '255(1)'로써, MSB-2 비트플래인부터 값이 존재한다는 것을 알 수 있다. 또한 MSB-4 비트플래인의 이진 영상을 LSB 비트플래인의 이진 영상과 비교해보면 완전히 일치한다는 것을 알 수 있으며, MSB-4, LSB 비트플래인의 이진 영상을 MSB-3 비트플래인의 이진 영상과 비교해보면 완전히 반전되게 일치한다는 것을 알 수 있다.

[0028] 따라서, 도 7의 실제 깊이 정보맵의 매크로블록에 그레이 코딩을 수행한 비트플래인에서 MSB, MSB-1, MSB-2, MSB-4 비트플래인은 이진 영상은 모두 '0(0)'이거나 모두 '255(1)'이므로 비트플래인 코딩 시 현재 비트플래인의 이진 영상이 모두 '0(0)'인지 아니면 모두 '255(1)'인지에 대한 정보만 보내주면 된다. 그리고 그레이 코딩된 비트플래인에서 MSB-3, MSB-5, MSB-6, LSB 비트플래인의 이진 영상은 이진 영상 압축 알고리즘을 이용하여 코딩해야 한다. 하지만 도 7의 실제 깊이 정보맵의 매크로블록은 카메라로부터 거리가 멀리 떨어져 있어서 픽셀값이 작고, 블록 내의 픽셀값 간의 상관도가 높기 때문에 DCT 기반 동영상 압축 알고리즘을 이용하여 코딩할 경우 비트플래인 단위 이진 영상 코딩보다 높은 코딩 효율을 가질 수 있다.

[0029] 이러한 사실을 통해서 비트플래인 단위 이진 영상 코딩 방법과 DCT 기반 동영상 압축 방법의 장점을 적절하게 이용하기 위한 방법이 필요하며, 또한 비트플래인 코딩시 원하는 비트율을 얻기 위한 방법이 필요하다.

[0030] [발명의 방법]

[0031] 본 발명에서는 비트플래인 단위 이진 영상 코딩 방법과 DCT 기반 동영상 압축 방법의 장점을 적절하게 이용하고 또한 비트플래인 코딩 시 원하는 비트율을 얻기 위해서 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법을 제안한다.

[0032] **[방법] 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법**

[0033] **<인코더>**

[0034] 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 인코더 구조도의 일 예는 도 8과 같다.

[0035] 도 8의 인코더 구조도는 입력된 깊이 정보맵을 MxN 블록 단위로 분리하여 코딩되며, MxN 블록 단위로 입력된 깊이 정보맵 블록은 비트플래인 코딩 모드 혹은 기존 코딩 모드 중에 적응적으로 선택되어 코딩이 수행이 된다. 기존 코딩 모드가 선택이 되었을 경우에는 도 8에서 'A'로 표시되어 있는 부분이 수행되며, 'A'는 깊이 정보맵 블록을 입력 받아 기존의 동영상 코딩 방법(MPEG-1, MPEG-2, MPEG-4 Part 2 Visual, H.264/AVC, SVC, MVC, VC-1, AVS, KTA, 등)으로 인코딩하여 비트스트림을 출력한다. 비트플래인 코딩 모드가 선택이 되었을 경우에는 'B'로 표시되어 있는 부분이 수행되며, 'B'는 n-bit로 표현되는 깊이 정보맵 블록과 참조 영상, 양자화 파라미터를 입력 받아 비트플래인 단위 코딩 방법을 이용하여 인코딩하고 비트스트림을 출력한다.

[0036] 도 8에서는 기존 코딩 모드 'A'의 일 예로 H.264/AVC의 인코더 구조도를 보여준다. 'A'의 인코더 구조도에서 데이터를 처리하는 단위는 가로 세로 16x16픽셀 크기의 매크로블록(Macroblock)이며, 인트라 모드 또는 인터 모드로 인코딩이 수행된다. 인트라(Intra) 모드일 경우, 'A' 내부의 스위치가 인트라로 전환이 되며, 인터(Inter) 모드일 경우에는 스위치가 인터로 전환이 된다. 코딩 과정의 주요한 흐름은 먼저 입력된 매크로블록에 대한 예측 블록을 생성한 후, 입력된 매크로블록과 예측 블록과의 차분을 구해 그 차분을 코딩하는 것이다. 먼저 예측 블록의 생성은 인트라 모드와 인터 모드에 따라 수행이 된다. 먼저 인트라 모드일 경우에는 "인트라 예측" 과정에서 현재 매크로블록의 이미 코딩된 주변 픽셀값을 이용하여 공간적 예측을 수행하여 예측 블록을 생성하며, 인터 모드일 경우에는 참조 영상 버퍼에 저장되어 있는 참조 영상에서 현재 입력된 블록과 가장 매치가 잘 되는 영역을 찾아 움직임 벡터(Motion Vector)를 구하는 과정인 "움직임 예측"을 수행하고, 구한 움직임 벡터를 이용하여 참조 영상 버퍼에 저장되어 있는 참조 영상에서 예측 블록을 가져오는 과정인 "움직임 보상"을 수행해서 예측 블록을 생성한다. 상기 설명한 것과 같이 현재 입력된 블록을 생성한 예측 블록과 차분을 구하여 잔여 블록(Residual Block)을 생성한 후, 이에 대한 코딩을 수행한다. 잔여 블록에 대한 코딩은 "변환", "양자화", "엔

트로피 코딩"의 순서로 수행이 된다. 먼저 "변환" 과정에서는 입력된 잔여 블록을 입력 받아 변환(Transform)을 수행하여 변환계수(Transform Coefficient)를 출력한다. 그리고 "양자화" 과정에서는 입력된 변환계수를 양자화 파라미터에 따라 양자화를 수행한 양자화된 계수(Quantized Coefficient)를 출력한다. 그리고 "엔트로피 코딩" 과정에서는 입력된 양자화된 계수를 확률 분포에 따른 엔트로피 코딩을 수행하여 결과를 출력한다. 수행된 결과는 "MUX"에 입력되어 비트스트림으로 출력이 된다. H.264/AVC는 프레임간(Inter-frame) 예측 코딩을 수행하기 때문에 현재 코딩된 영상을 이후에 입력된 영상의 참조 영상으로 사용하기 위해 디코딩하여 저장할 필요가 있다. 따라서 양자화된 계수를 "역양자화"과정과 "역변환"을 수행하여 예측 영상과 가산기를 통해 재구성된 블록을 생성한 다음 "디블록킹 필터"를 통해 코딩 과정에서 발생한 블록킹 현상(Blocking Artifact)을 제거한 후, "재구성된 영상 버퍼"에 저장한다.

[0037] 도 8에서 비트플래인 코딩 모드 'B'는 비트플래인 단위 코딩 방법의 인코더 구조도를 보여준다. 'B'의 인코더 구조도에서 데이터를 처리하는 단위는 가로 세로 MxN픽셀 크기의 블록이다.

[0038] 여기에서 깊이 정보맵을 코딩할 때 데이터를 처리하는 단위는 다양하게 적용될 수 있다.

[0039] (1) 하나의 일 실시 예로서, MxN픽셀 크기의 블록을 작은 블록으로 나누어서 서브 블록을 구성하고 구성된 서브 블록이 각각 다른 방법이나 다른 블록 모드로 코딩될 수 있다.

[0040] (2) 또 다른 실시 예로서, MxN픽셀 크기의 블록들을 큰 블록 혹은 더 큰 영역으로 합쳐서 코딩될 수 있다.

[0041] (3) 또 다른 실시 예로서, 위 (1)과 (2)의 방법을 조합하여 여러 크기의 블록들로 코딩될 수 있다.

[0042] 이 외에도 위와 같이 깊이 정보맵을 코딩할 때 데이터를 처리하는 단위는 다양한 방법으로 변경될 수 있다.

[0043] 도 8의 "그레이 코드 변환" 과정에서는 깊이 정보맵 블록을 입력 받아서 각각의 픽셀을 그레이 코드로의 변환을 수행한다. 그레이 코드는 연속하는 값을 표현할 때, 1-비트만을 변경하여 표현하는 코드이다. 일반적으로 깊이 정보맵은 주변 픽셀과 상당히 유사한 특성을 갖는데, 깊이 정보맵 그대로를 비트플래인으로 나누었을 경우에는 각 비트플래인의 픽셀은 픽셀의 유사도와 상관없이 서로 다른 값을 가질 수 있다. 일 예로 8-비트로 표현되는 깊이 정보맵에서 연속되는 두 개의 픽셀이 '127'과 '128'의 값을 가진다고 가정할 경우에 '127'은 이진수로 $(01111111)_2$ 로 표현이 되며 '128'은 이진수로 $(10000000)_2$ 로 표현이 된다. 이와 같은 경우에는 깊이 정보 자체는 유사한 값을 갖지만, 비트플래인 별로 비교를 했을 경우에는 모든 비트에서 다른 값을 가지는 것을 확인할 수 있다. 하지만 깊이 정보맵을 그레이 코드로 변경하였을 경우에는 실제 값이 '1' 차이가 나면 1-비트만 차이가 나도록 되어 있기 때문에 각 비트플래인 안에서 주변 비트값의 유사도를 높여줄 수 있다. 실제 m-비트 깊이 정보 픽셀의 이진값 $(a_{m-1}\cdots a_2a_1a_0)_2$ 를 그레이 코드 $(g_{m-1}\cdots g_2g_1g_0)_2$ 로 변경하는 방법은 아래 수학식 2와 같다.

*수학식 2*

$$g_i = a_i \oplus a_{i+1} \qquad 0 \le i \le m-2$$
$$g_{m-1} = a_{m-1}$$

[0044]

[0045] 여기에서 $\oplus$ 는 XOR(eXclusive OR) 연산을 의미한다. 실제 '127'과 '128'의 값을 그레이 코드로 변환을 수행하면 '127'은 이진수로 $(11000000)_2$ 로 표현이 되며, '128'은 이진수로 $(01000000)_2$ 로 표현이 된다. 그레이 코드로 변환을 수행하였을 때 각 비트의 유사도가 증가한 것을 쉽게 확인할 수 있다. "그레이 코드 변환" 과정은 수행될 수도 있고 수행되지 않을 수도 있다.

[0046] 도 8의 "비트플래인 분리" 과정에서는 n-bit로 표현되는 깊이 정보맵 블록을 입력 받아서 n개의 비트플래인 블록으로 분리한다. 도 8의 "비트율 조절" 과정에서는 입력된 깊이 정보맵 블록에 비트율 조절 기능이 필요할 경우 선택적으로 사용할 수 있는 과정으로써, 실제 코딩 시에 원하는 비트율을 얻기 위해 코딩될 비트율을 조절한다. "비트율 조절"에 대한 설명은 추후에 자세히 설명한다. 여기에서 "비트플래인 분리" 과정이 수행된 후 "비트율 조절" 과정이 수행될 수 있으며, 혹은 "비트율 조절" 과정이 수행된 후 "비트플래인 분리" 과정이 수행될 수 있다. "비트플래인 분리" 과정과 "비트율 조절" 과정을 거친 분리된 비트플래인 블록들은 "비트플래인 코딩" 과정으로 반복하여 입력이 되어 "비트플래인 코딩"을 수행한 결과로 비트스트림을 출력한다.

[0047] 도 8의 참조 영상 버퍼에 저장되어 있는 참조영상과 재구성된 영상 버퍼에 저장되어 있는 재구성된 영상은 "비

트플래인 코딩" 과정으로 입력된다. 도 8의 "비트플래인 코딩" 과정에서는 각각의 비트플래인을 인코딩하여 비트스트림을 출력한다. "비트플래인 코딩"을 수행하는 방법은 추후에 자세히 설명한다.

[0048] 도 8에서 비트플래인 코딩이 선택되어 "비트플래인 코딩"을 수행하고, 코딩을 수행한 데이터는 이후에 입력되는 깊이 정보맵의 코딩을 위해 "비트플래인 디코딩"을 수행한다. 도 8의 참조 영상 버퍼에 저장되어 있는 참조영상과 재구성된 영상 버퍼에 저장되어 있는 재구성된 영상은 "비트플래인 디코딩" 과정으로 입력된다. "비트플래인 디코딩" 과정에서는 비트스트림을 입력 받아 비트플래인 수만큼 반복해서 수행이 되며, 입력된 비트스트림을 각 비트플래인 별로 디코딩하여 n개의 비트플래인 영상을 출력한다. "비트플래인 디코딩"을 수행하는 방법은 추후에 자세히 설명한다. "비트플래인 결합" 과정에서는 출력된 각각의 비트플래인 영상을 결합하여 n-bit로 표현되는 영상을 출력한다. "역 그레이 코드 변환" 과정에서는 그레이 코드로 표현된 깊이 정보맵을 원래의 형태로 복원하는 것으로 실제 m-비트 깊이 정보 픽셀의 그레이 코드 $(g_{m-1}\cdots g_2 g_1 g_0)_2$ 를 이진값 $(a_{m-1}\cdots a_2 a_1 a_0)_2$ 로 변경하는 방법은 아래 수학식 3과 같다.

*수학식 3*

$$a_{m-1} = g_{m-1}$$
$$a_i = a_{i+1} \oplus g_i \qquad 0 \le i \le m-1$$

[0049]

[0050] 여기에서 $\oplus$ XOR(eXclusive OR) 연산을 의미한다.

[0051] "깊이 정보맵 구성" 과정에서는 n-bit로 표현되는 영상을 실제 깊이 정보맵의 비트수에 맞게 구성하여 깊이 정보맵을 출력한다.

[0052] 여기에서 깊이 정보맵을 구성하는 방법은 다양하게 적용될 수 있다.

[0053] (1) 하나의 일 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기위해 코딩되지 않은 비트플래인 블록의 모든 값을 '0(0)'으로 설정하여 구성할 수 있다.

[0054] (2) 또 다른 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기 위해 코딩되지 않은 비트플래인 블록의 모든 값을 '255(1)'로 설정하여 구성할 수 있다.

[0055] (3) 또 다른 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기 위해 코딩되지 않은 비트플래인 블록을 모두 '0(0)'으로 구성할지 혹은 모두 '255(1)'로 구성할지에 대한 정보를 코딩하여 코딩되지 않은 비트플래인 블록을 구성할 수 있다. 코딩되지 않은 비트플래인 블록을 어떻게 구성할지에 대한 정보는 비트스트림에 포함될 수 있다.

[0056] (4) 또 다른 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기 위해 코딩되지 않은 비트플래인 블록을 그 블록에 인접하는 주변 비트플래인 블록의 픽셀값을 이용한 패딩(Padding) 방법을 통해 구성할 수 있다. 여기에서 주변 비트플래인 블록의 픽셀값을 이용한 패딩(Padding)은 여러 가지의 패딩 방향을 가질 수 있으며, 해당 패딩 방향에 대한 정보는 비트스트림에 포함된다.

[0057] 이 외에도 위와 같이 깊이 정보맵을 구성하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0058] 구성된 깊이 정보맵은 기존 코딩 모드 'A'에서 인트라 예측에 사용되거나 디블록킹 필터를 거쳐 재구성된 영상 버퍼에 저장된다. 이때 현재 블록에 대한 디블록킹 필터의 수행 여부 정보를 코딩하여 그 정보를 통해 디블록킹 필터를 수행할 수 있다. 여기에서 구성된 깊이 정보맵은 디블록킹 필터를 거치지 않고 재구성된 영상 버퍼에 저장될 수 있다. 또한 구성된 깊이 정보맵은 디블록킹 필터를 거친 후 인트라 예측에 사용될 수 있다.

[0059] "비트플래인 코딩"을 수행하는 방법의 실시 일 예로 국제 동영상 표준인 MPEG-4 Part 2 Visual(ISO/IEC 14496-2)의 이진 형상 코딩(binary shape coding)에서 사용하는 방법을 응용하여 사용할 수 있다. 제안하는 "비트플래인 코딩"의 일 예는 도 9와 같다.

[0060] 도 9에서 입력된 현재 비트플래인 블록은 현재 코딩하려고 하는 깊이 정보맵 블록에서 코딩을 수행할 비트플래

인 블록을 의미한다. 도 8의 참조 영상 버퍼에 저장되어 있는 참조영상과 도 8의 재구성된 영상 버퍼에 저장되어 있는 재구성된 영상은 각각 "그레이 코드 변환"과정과 "비트플래인 분리" 과정을 거쳐 그레이 코드가 적용된 각각의 비트플래인으로 분리되고, 현재 코딩할 비트플래인과 동일한 비트플래인의 참조 비트플래인 영상은 "움직임 예측"과 "CAE인코딩" 과정으로 입력된다.

[0061] 도 9의 "움직임 예측" 과정에서는 현재 비트플래인 블록과 가장 유사한 부분을 참조 비트플래인 영상에서 검색하여 가장 잘 매치가 되는 영역과의 움직임 변위 즉 움직임 벡터를 계산하여 출력한다. "움직임 보상" 과정에서는 "움직임 예측" 과정에서 생성된 움직임 벡터를 이용하여 참조 비트플래인 영상으로부터 움직임 보상된 비트플래인 블록을 출력한다.

[0062] "모드 결정" 과정에서는 현재 비트플래인 블록의 모드를 결정하는 부분으로써 현재 비트플래인 블록과 움직임 보상된 비트플래인 블록을 통해 결정이 되며, 현재 깊이 정보맵 프레임의 코딩 모드가 시간 방향 예측을 수행하지 않을 경우에 사용되는 인트라(Intra) 픽쳐 모드 결정 방법과, 현재 깊이 정보맵 프레임의 코딩 모드가 시간 방향 예측을 수행할 경우에 사용되는 인터(Inter)픽쳐 모드 결정 방법을 수행하며, 각각의 상세한 설명은 추후에 자세히 설명한다.

[0063] 여기에서 결정된 비트플레인 블록의 모드는 다양한 방법으로 코딩할 수 있다.

[0064] (1) 하나의 일 예로서, 현재 비트플레인 블록의 모드를 고정길이부호화를 통해 코딩할 수 있다.

[0065] (2) 또 다른 예로서, 현재 비트플레인 블록의 모드를 비트플래인 블록 모드의 출현 빈도에 따른 확률을 이용한 가변길이부호화를 통해 코딩할 수 있다. 이때 미리 계산된 확률 테이블을 이용하여 코딩할 수 있다.

[0066] (3) 또 다른 예로서, 현재 비트플레인 블록의 모드를 현재 비트플래인 블록에 인접하는 주변의 비트플래인 블록 모드를 고려하여 상황에 맞게 적응적으로 변화된 확률을 이용한 산술 코딩 방법을 통해 코딩할 수 있다.

[0067] 이 외에도 위와 같이 비트플레인 블록 모드를 코딩하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0068] "CAE(Context-based Arithmetic Encoding) 코딩" 과정은 현재 비트플래인 블록의 픽셀값이 모두 '0' 또는 '1'일 경우가 아닌 경우와, 참조 비트플래인에서 현재 비트플래인 블록과 동일한 위치의 참조 비트플래인 블록, 또는 움직임 보상을 수행한 참조 비트플래인 블록간의 오차가 허용 오차 범위를 초과할 경우에 수행되며, 현재 비트플래인 블록 내의 각 픽셀은 인트라 모드일 경우에는 각 픽셀의 주변의 픽셀 정보를 기반으로, 인터 모드일 경우에는 각 픽셀의 주변의 픽셀 정보와 현재 픽셀에 대응하는 참조 비트플래인 영상의 픽셀과 그 주변 픽셀 정보를 기반으로 하는 이진 산술 코딩(Binary Arithmetic Coding)을 수행한다. "CAE 코딩" 방법에 대한 상세한 설명은 추후에 자세히 설명한다. "멀티플렉서" 에서는 현재 비트플래인 블록의 움직임 벡터와 현재 비트플래인 블록의 모드, 그리고 "CAE 코딩" 결과를 입력으로 받아 "비트스트림"을 생성한다.

[0069] 도 9의 "모드 결정" 부분에서 인트라 픽쳐 모드 결정 방법의 실시 일 예의 순서도는 도 10과 같으며, 자세한 알고리즘은 다음과 같다.

[0070] Step 1) 현재 비트플래인 블록의 모든 픽셀값이 동일한 값이면 Step 2로 분기한다. 만약 그렇지 않으면 Step 3으로 분기한다.

[0071] Step 2) 현재 비트플래인 블록의 모든 픽셀값이 '1'이면 현재 비트플래인 블록의 모드를 'all_1'로 설정한다. 만약 그렇지 않다면 현재 비트플래인 블록의 모드를 'all_0'로 설정한다.

[0072] Step 3) 현재 비트플래인 블록의 모드를 'intraCAE'로 한다. 'intraCAE' 모드의 코딩 방법은 추후에 자세히 설명한다.

[0073] 도 9의 "모드 결정" 부분에서 인터 픽쳐 모드 결정 방법의 실시 일 예의 순서도는 도 11과 같으며, 자세한 알고리즘은 다음과 같다.

[0074] Step 1) 현재 비트플래인 블록의 움직임 벡터의 예측값($MV_P$)에 대응하는 참조 비트플래인 블록과 현재 비트플래인 블록간의 오차($A_{err}$)가 허용 오차 범위($B_{err}$)와 동일하거나 작다면 Step 2로 분기한다. 만약 그렇지

않다면 Step 3으로 분기한다.

[0075] Step 2) 현재 비트플래인 블록 모드를 'No Update Without MV'로 한다. 'No Update Without MV' 모드는 현재 비트플래인 블록의 재구성된 비트플래인 블록을 현재 비트플래인 블록의 주변으로부터 움직임 벡터의 예측값($MV_P$)에 대응하는 참조 비트플래인 블록으로 하며, 현재 비트플래인 블록의 모드 정보만을 전송하고 이외의 추가 데이터는 전송하지 않는다. 움직임 벡터의 예측값($MV_P$) 결정 방법은 추후에 자세히 설명한다.

[0076] Step 3) 현재 비트플래인 블록의 모든 픽셀값이 동일한 값이면 Step 4로 분기한다. 만약 그렇지 않으면 Step 5로 분기한다.

[0077] Step 4) 현재 비트플래인 블록의 모든 픽셀값이 '1'이면 현재 비트플래인 블록의 모드를 'all_1'로 설정한다. 만약 그렇지 않다면 현재 비트플래인 블록의 모드를 'all_0'으로 설정한다.

[0078] Step 5) 움직임 예측을 수행한다.

[0079] Step 6) 움직임 예측에서 계산한 움직임 벡터에 대응하는 참조 비트플래인 블록과 현재 비트플래인 블록간의 오차($C_{err}$)가 허용 오차 범위($B_{err}$)와 동일하거나 작다면 Step 7로 분기한다. 만약 그렇지 않다면 Step 8로 분기한다.

[0080] Step 7) 현재 비트플래인 블록의 모드를 'No Updata with MV'로 한다. 'No Updata with MV' 모드는 현재 비트플래인 블록의 재구성된 비트플래인 블록을 움직임 예측을 통해 계산한 움직임 벡터에 대응하는 참조 비트플래인 블록으로 하며, 현재 비트플래인 블록의 모드 정보와 움직임 벡터만을 전송하고 이외의 추가 데이터는 전송하지 않는다

[0081] Step 8) 'intraCAE' 모드 코딩과 'interCAE'모드 코딩을 각각 수행한다. 'intraCAE'와 'interCAE'모드의 코딩 방법은 추후에 자세히 설명한다.

[0082] Step 9) 만약 'intraCAE' 모드 코딩 결과의 비트량이 'interCAE' 모드 코딩 결과의 비트량 보다 작다면 Step 10으로 분기한다. 만약 그렇지 않다면 Step 11로 분기한다.

[0083] Step 10) 현재 비트플래인 블록의 모드를 'intraCAE'로 한다. 'intraCAE' 모드 코딩 방법은 추후에 설명한다.

[0084] Step 11) 현재 비트플래인 블록의 모드를 'interCAE'로 한다. 'Inter CAE' 모드 코딩 방법은 추후에 설명한다.

[0085] 현재 비트플래인 블록의 움직임 벡터 예측값($MV_P$) 결정 방법의 실시 일 예의 순서도는 도 12와 같으며, 자세한 알고리즘은 다음과 같다.

[0086] Step 1) 현재 비트플래인 블록에서 좌측의 경계가 영상의 경계(LeftBoundary)가 아니고 현재 비트플래인 블록에서 좌측 비트플래인 블록 모드(LeftMBMode)가 시간 방향 예측(inter)을 수행하였다면, Step 2로 분기한다. 만약 그렇지 않다면 Step 3으로 분기한다.

[0087] Step 2) 현재 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 현재 비트플래인 블록에서 좌측 비트플래인 블록의 움직임 벡터($MV_{Left}$)로 설정한다.

[0088] Step 3) 현재 비트플래인 블록에서 상단의 경계가 영상의 경계(AboveBoundary)가 아니라면, Step 4로 분기한다. 만약 현재 비트플래인 블록에서 상단의 경계가 영상의 경계(AboveBoundary)라면, Step 8로 분기한다.

[0089] Step 4) 현재 비트플래인 블록에서 상단의 경계가 영상의 경계(AboveBoundary)가 아니고 현재 비트플래인 블록에서 상단 비트플래인 블록 모드(AboveMBMode)가 시간 방향 예측(inter)을 수행하였다면, Step 5로 분기한다. 만약 그렇지 않다면 Step 6으로 분기한다.

[0090] Step 5) 현재 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 현재 비트플래인 블록에서 상단 비트플래인 블록의 움직임 벡터($MV_{Above}$)로 설정한다.

[0091] Step 6) 현재 비트플래인 블록에서 우측의 경계 또는 상단의 경계가 영상의 경계(AboveRightBoundary)

가 아니고 현재 비트플래인 블록에서 우측 상단 비트플래인 블록 모드(AboveRightMBMode)가 시간 방향 예측(inter)을 수행하였다면, Step 7로 분기한다. 만약 그렇지 않다면 Step 8로 분기한다.

[0092] Step 7) 현재 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 현재 비트플래인 블록에서 우측상단 비트플래인 블록의 움직임 벡터($MV_{AboveRight}$)로 설정한다.

[0093] Step 8) 현재 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 수평 성분 값과 수직 성분 값 모두 '0'으로 설정한다.

[0094] 또한, 여기에서 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 결정하는 방법은 다양하게 적용될 수 있다.

[0095] (1) 하나의 일 실시 예로서, 현재 비트플레인 블록에 인접하는 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터들을 이용하여 현재 비트플래인 블록의 움직임 벡터 예측값($MV_p$)을 결정할 수 있다.

[0096] (2) 또 다른 실시 예로서, 도 12의 순서도에서 현재 비트플래인 블록과 인접하는 주변 비트플래인 블록 대신 깊이 정보맵 블록의 움직임 벡터를 이용하여 현재 비트플래인 블록의 움직임 벡터 예측값($MV_p$)을 결정할 수 있다.

[0097] (3) 또 다른 실시 예로서, 현재 비트플레인 블록에서 좌측 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{Left}$)와 상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{Above}$), 그리고 우측상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{RightAbove}$)의 중간값(Median)을 이용하여 현재 비트플래인 블록의 움직임 벡터 예측값($MV_p$)을 결정할 수 있다. 여기에서 우측상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{RightAbove}$)가 유효하지 않다면, 좌측상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{LeftAbove}$)를 대신 사용할 수 있다.

[0098] 이 외에도 위와 같이 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 결정하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0099] 도 9의 "모드 결정" 부분에서 결정된 현재 비트플래인 블록의 모드가 'intraCAE'와 'interCAE'로 선택 되었을 경우, 컨텍스트 기반 산술 인코딩(CAE, Context-based Arithmetic Encoding) 방법을 사용하며, "CAE 인코딩"의 구조도의 일 예는 도 13과 같다.

[0100] 도 13의 "비트율 조절" 과정은 현재 비트플래인 블록 모드가 'intraCAE' 모드와 'interCAE'모드일 때 크기 변환(Size Conversion) 방법을 통해 수행된다. 현재 비트플래인 블록을 변환 비율(Conversion Ratio; CR) 1/2 혹은 1/4 크기로 다운샘플링(down-sampling)한 후, 다시 현재 비트플래인 블록 크기로 업샘플링(up-sampling)하여 생성한 비트플래인 블록과 현재 비트플래인 블록간의 오차를 계산하고, 만약 그 오차가 허용 오차 범위와 같거나 작다면, 현재 비트플래인 블록을 1/2 혹은 1/4로 다운샘플링하여 생성한 블록에 대한 코딩을 수행한다. 그렇지 않다면 현재 비트플래인 블록을 그대로 코딩한다. 변환 비율을 설정하는 방법은 순서도는 도 14과 같으며, 자세한 알고리즘은 다음과 같다.

[0101] Step 1) 현재 비트플래인 블록에 대해 변환비율(CR)을 1/4로 설정하고, 현재 비트플래인 블록의 변환비율에 따라 크기변환 즉 1/4 다운샘플링을 수행하고 다시 업샘플링을 수행한 비트플래인 블록을 생성한다.

[0102] Step 2) 현재 비트플래인 블록과 현재 비트플래인 블록에 변환비율(CR) 1/4로 다운샘플링하고 다시 업샘플링을 수행한 블록간의 오차($D_{err}$)가 허용 오차($B_{err}$) 범위보다 크다면 Step 3으로 분기한다. 만약 그렇지 않다면 알고리즘을 종료한다.

[0103] Step 3) 현재 비트플래인 블록에 대해 변환비율(CR)을 1/2로 설정하고, 현재 비트플래인 블록의 변환비율에 따라 크기변환 즉 1/2 다운샘플링을 수행하고 다시 업샘플링을 수행한 블록을 생성한다.

[0104] Step 4) 현재 비트플래인 블록과 현재 비트플래인 블록에 변환비율(CR) 1/2로 다운샘플링하고 다시 업샘플링을 수행한 비트플래인 블록 간의 오차($E_{err}$)가 허용 오차($B_{err}$) 범위보다 크다면 Step 5로 분기한다. 만약

그렇지 않다면 알고리즘을 종료한다.

[0105] Step 5) 현재 비트플래인 블록에 대해 변환비율(CR)을 1로 설정하고, 알고리즘을 종료한다.

[0106] 도 13의 "다운샘플링" 과정은 "비트율 조절" 과정에서 출력된 변환비율(CR)에 따라서 현재 비트플래인 블록에 대응하는 참조 비트플래인 영상의 참조 비트플래인 블록에 대한 다운샘플링(Down-sampling)을 수행하여 컨텍스트 계산에 사용할 수 있도록 한다. 만약 변환비율(CR)이 '1' 이라면 다운샘플링을 수행하지 않는다.

[0107] 도 13의 "컨텍스트 계산" 과정에서는 현재 비트플래인 블록의 모드가 'intraCAE' 모드일 경우에는 현재 비트플래인 블록의 코딩할 픽셀의 주변의 픽셀값들을 기반으로, 'interCAE' 모드일 경우에는 현재 비트플래인 블록의 코딩할 픽셀의 주변의 픽셀값들과 현재 블록의 코딩할 픽셀에 대응하는 참조 비트플래인 영상의 픽셀과 그 주변 픽셀값들을 통해 컨텍스트 템플릿(context template)를 구성한다. "산술 인코딩" 과정에서는 "컨텍스트 계산" 과정에서 구성된 컨텍스트 템플릿을 인덱스(index)로 하는 확률 테이블을 참조하여 현재 코딩할 픽셀에 대한 산술 인코딩을 수행해서 비트스트림을 생성한다.

[0108] 'intraCAE' 모드에서 컨텍스트 템플릿을 구성할 때 사용하는 픽셀들은 도 15와 같고 현재 픽셀(X)을 중심으로 10개의 주변 픽셀들을 (c9 c8 c7 c6 c5 c4 c3 c2 c1 c0) 형태의 10비트 컨텍스트 템플릿을 형성한 후에, 이것을 산술 코딩시의 확률 테이블의 인덱스로 사용한다. 그리고 'interCAE' 모드에서 컨텍스트 템플릿을 구성할 때 사용하는 픽셀들은 도 16과 같고 현재 픽셀(X)을 중심으로 4개의 주변 픽셀들과, 현재 블록에 대응하는 참조 블록에서 현재 픽셀에 대응하는 픽셀(c6)과 그 주변 4개 픽셀들을 (c8 c7 c6 c5 c4 c3 c2 c1 c0) 형태의 9비트 컨텍스트 템플릿을 형성한 후에, 이것을 산술 코딩시의 확률 테이블의 인덱스로 사용한다.

[0109] 여기에서 'intraCAE' 모드와 'interCAE' 모드에서 컨텍스트 템플릿을 구성하는 방법은 다양하게 적용할 수 있다.

[0110] (1) 하나의 일 예로서, 'intraCAE' 모드에서 비트플래인 블록의 코딩할 픽셀(X)과 인접하는 주변의 픽셀들 중 다양한 개수의 픽셀들을 이용하여 컨텍스트 템플릿을 구성할 수 있다. 실시 예로서, 'intraCAE' 모드에서 현재 픽셀(X)을 중심으로 인접하는 4개의 주변 픽셀들을 (c3 c2 c1 c0) 형태의 4비트 컨텍스트 템플릿을 구성할 수 있다.

[0111] (2) 또 다른 실시 예로서, 'interCAE' 모드에서 현재 비트플래인 블록의 코딩할 픽셀(X)과 인접하는 주변의 픽셀들과 현재 비트플래인 블록의 코딩할 픽셀(X)에 대응하는 참조 비트플래인 영상의 픽셀과 그 주변 픽셀들 중 다양한 개수의 픽셀들을 이용하여 컨텍스트 템플릿을 구성할 수 있다. 실시 예로서, 'interCAE' 모드에서 현재 픽셀(X)을 중심으로 인접하는 4개의 주변 픽셀들을 (c3 c2 c1 c0) 형태의 4비트 컨텍스트 템플릿을 구성할 수 있다.

[0112] 이 외에도 위와 같이 컨텍스트 템플릿을 구성하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0113] 도 13에서 "컨텍스트 계산"과 "산술 인코딩"은 현재 비트플래인 블록 또는 다운샘플링된 비트플래인 블록의 픽셀 수만큼 반복된다. 이때 현재 비트플래인 블록의 픽셀을 코딩하는 순서는 위에서 아래로 왼쪽에서 오른쪽으로 가로 방향 스캔 또는 세로 방향 스캔을 적응적으로 수행할 수 있으며, 두 가지 스캔 모드를 모두 수행한 후, 비트량이 적은 스캔 모드를 선택하고, 스캔 모드 정보는 비트스트림에 저장한다.

[0114] 여기에서 현재 비트플래인 블록의 픽셀을 스캔하는 방법은 다양하게 적용할 수 있다.

[0115] (1) 하나의 일 실시 예로서, 도 17의 (a)처럼 현재 비트플래인 블록의 픽셀을 왼쪽에서 오른쪽으로 위에서 아래로 스캔을 수행하거나 위에서 아래로 왼쪽에서 오른쪽으로 스캔을 수행할 수 있다.

[0116] (2) 또 다른 실시 예로서, 도 17의 (b)처럼 현재 비트플래인 블록의 픽셀을 왼쪽에서 오른쪽으로 그리고 다시 오른쪽에서 왼쪽으로 지그재그(zigzag) 순으로 위에서 아래로 스캔을 수행하거나 위에서 아래로 아래에서 위로 지그재그(zigzag) 순으로 왼쪽에서 오른쪽으로 스캔을 수행할 수 있다.

[0117] (3) 또 다른 실시 예로서, 도 17의 (c)처럼 현재 비트플래인 블록의 픽셀을 왼쪽 위에서부터 대각선 방향으로 오른쪽 위에서 왼쪽 아래로 스캔하거나 왼쪽 아래에서부터 대각선 방향으로 왼쪽 위에서 오른쪽 아래로 스캔

을 수행할 수 있다.

[0118] (4) 또 다른 실시 예로서, 도 17의 (d)처럼 현재 비트플래인 블록의 픽셀을 왼쪽 위에서부터 대각선 방향 지그재그(zigzag) 순으로 왼쪽 위에서 오른쪽 아래로 스캔을 수행하거나 왼쪽 아래에서부터 대각선 방향 지그재그(zigzag) 순으로 왼쪽 아래에서 오른쪽 위로 스캔을 수행할 수 있다.

[0119] 이 외에도 위와 같이 현재 비트플래인 블록의 픽셀을 스캔하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0120] 도 9의 "비트플래인 코딩"을 수행하는 방법의 또 다른 실시 일 예로 상기 살펴본 MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법이 있다.

[0121] ① MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법의 일 예로 "비트율 조절" 부분의 변환비율(CR)을 다양하게 적용하여 코딩을 수행할 수 있다.

[0122] ② MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법의 또 다른 일 예로 블록 단위에서 움직임 정보를 각 비트플래인 블록 별로 모두 저장하지 않고, 블록 단위에서 하나 이상의 특정 비트플래인의 움직임 정보만 저장할 수도 있다.

[0123] 도 9의 "비트플래인 코딩"에서 "CAE인코딩"을 수행하는 과정은 [방법 #1] 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인 코딩 방법과 [방법 #2] 컨텍스트 기반 적응적 산술 코딩을 응용한 비트플래인 코딩 방법, 그리고 [방법 #3] CAC(Constant Area Coding) 혹은 JBIG(Joint Bi-Level Image Processing Group)의 방법 혹은 Quad Tree 방법을 이용한 비트플래인 코딩 방법으로 대체되어 수행될 수 있다.

[0124] [방법 #1] 도 9의 "비트플래인 코딩"에서 "CAE인코딩"을 수행하는 방법 대신에 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용하여 비트플래인 블록을 코딩할 수 있는데 제안하는 방법의 일 예는 도 18과 같다.

[0125] 도 18은 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding)방법을 사용하여 비트플래인 블록을 코딩하는 방법을 나타내며, 비트플래인 블록을 입력받아 비트플래인 코딩을 수행하여 비트스트림을 출력한다. "픽셀 스캔 및 런 길이 저장" 과정에서는 입력된 비트플래인 블록은 도 17의 픽셀 스캔 방법을 이용한 스캔을 수행하여 비트플래인 블록의 각 심볼(Symbol)에 대한 런(Run)의 길이들을 저장한다. 여기에서 각 심볼(Symbol)에 대한 런(Run)의 길이들을 세는 방법은 도 19와 같으며, 자세한 알고리즘은 다음과 같다.

[0126] Step 1) 이전 픽셀 값(PrevPixel)을 '0'으로 설정하고 런(Run)의 길이(nCount)를 '0'으로 설정한다.

[0127] Step 2) 도 17의 스캔 순서에 따라 다음 픽셀을 읽어 오는 함수(NextPixel())을 통해 현재 픽셀 값(CurrPixel)을 읽어온다.

[0128] Step 3) 만약 현재 픽셀 값(CurrPixel)이 이전 픽셀 값(PrevPixel)과 다르다면 Step 5로 분기한다. 만약 그렇지 않다면, Step 4로 분기한다.

[0129] Step 4) 런(Run)의 길이(nCount)를 '1' 증가시킨 후 Step 2로 분기한다.

[0130] Step 5) 런(Run)의 길이(nCount)를 저장한다.

[0131] Step 6) 런(Run)의 길이(nCount)를 '0'으로 설정한다.

[0132] Step 7) 도 17의 스캔 순서에 따라 다음 픽셀을 읽어 오는 함수(NextPixel())을 통해 읽어온 픽셀값이 존재하지 않는다면, 알고리즘을 종료한다. 그렇지 않다면 Step 2로 분기한다.

[0133] 도 18의 "가변길이 코딩" 과정에서는 입력된 런(Run)의 길이들을 확률 분포에 따라 가변 길이 코딩(Variable Length Coding ) 방법을 수행하여 비트스트림을 출력한다. 이때 입력된 런(Run)의 길이들을 코딩하기 위한 방법으로 산술 코딩(Arithmetic Coding) 방법을 이용하여 코딩할 수도 있다.

[0134] 도 18의 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 사용하여 비트플래인 블록을 코딩하는 방법에서 입력된 비트플래인 블록을 작은 블록으로 나누어서 서브 블록을 구성하고 그 서브 블록 각각을 코딩할 수 있으며, 제안하는 방법의 일 예는 도 20과 같다.

[0135] 도 20의 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인코딩 방법에서 비트플래인 블록을 서브 블록으로 나누어서 코딩하는 방법을 나타내며, 비트플래인 블록을 입력 받아 서브 블록으로 나누어서 각각의 서브 블록에 비트플래인 코딩을 수행하여 비트스트림을 출력한다. "서브 블록 분리" 과정에서는 입력된 비트플래인 블록을 작은 블록으로 나누어서 서브 블록을 구성한다. "픽셀 스캔 및 런 길이 저장" 과정과 "가변 길이 코딩" 과정은 나누어진 서브 블록의 개수만큼 반복 수행된다. "픽셀 스캔 및 런 길이 저장" 과정에서는 입력된 서브 비트플래인 블록에 도 17의 픽셀 스캔 방법을 이용한 스캔을 수행하여 비트플래인 블록의 런(Run)의 길이를 저장한다. 여기에서 런(Run)의 길이를 정하는 방법은 도 19와 같다.

[0136] [방법 #2] 도 9의 "비트플래인 코딩"에서 "CAE인코딩"을 수행하는 방법 대신에 현재 코딩할 비트플래인 블록에 인접하는 주변 비트플래인 블록의 코딩 정보를 이용한 컨텍스트 기반 적응적 산술 코딩(CABAC; Context-based Adaptive Binary Arithmetic Coding) 방법을 응용하여 코딩을 수행할 수 있는데, 제안하는 방법의 일 예는 도 21과 같다.

[0137] 도 21은 현재 코딩할 비트플래인 블록에 인접하는 주변 비트플래인 블록의 코딩 정보를 이용한 비트플래인 코딩 방법으로 컨텍스트 기반 적응적 산술 코딩(CABAC; Context-based Adaptive Binary Arithmetic Coding) 방법을 응용한 방법을 나타내며, 비트플래인 블록을 입력 받아 컨텍스트 기반 적응적 산술 코딩을 응용한 비트플래인 코딩을 수행하여 비트스트림을 출력한다. 여기에서 입력된 비트플래인 블록을 작은 블록으로 나누어서 서브 블록을 구성하고 각 서브 블록에 도 21의 방법을 각각 적용할 수 있다. "컨텍스트 구성" 과정에서는 입력된 비트플래인 블록에서 현재 코딩을 수행할 픽셀과 인접하는 주변 픽셀들을 이용하여 컨텍스트 템플릿을 생성한다. " 산술 코딩" 과정에서는 생성된 컨텍스트 템플릿을 이용하여 현재 픽셀에 대해 산술 코딩을 수행한다.

[0138] [방법 #3] 도 9의 "비트플래인 코딩"에서 "CAE인코딩"을 수행하는 방법 대신에 현재 코딩할 비트플래인 블록을 CAC(Constant Area Coding) 혹은 JBIG(Joint Bi-Level Image Processing Group)의 방법 혹은 Quad Tree 방법을 이용한 비트플래인 코딩 방법을 수행할 수 있다.

[0139] "비트플래인 디코딩"을 수행하는 방법의 실시 일 예로 국제 동영상 표준인 MPEG-4 Part 2 Visual(ISO/IEC 14496-2)의 이진 형상 코딩(binary shape coding)에서 사용하는 방법을 응용하여 사용할 수 있다. 제안하는 "비트플래인 디코딩"의 일 예는 도 22와 같다.

[0140] 도 22은 비트플래인 디코딩을 수행하는 구조도를 나타내며, 비트스트림을 입력 받아서 비트플래인 디코딩을 수행하여 재구성된 비트플래인 블록을 출력한다. 도 8의 참조 영상 버퍼에 저장되어 있는 참조영상과 도 8의 재구성된 영상 버퍼에 저장되어 있는 재구성된 영상은 각각 "그레이 코드 변환"과정과 "비트플래인 분리" 과정을 거쳐 각각의 비트플래인으로 분리되고, 현재 코딩할 비트플래인 블록과 동일한 비트플래인의 참조 비트플래인 영상은 "움직임 보상"과 "CAE 디코딩"과정으로 입력된다.

[0141] 도 22의 "DE-MUX" 과정에서는 비트스트림을 입력으로 받아 현재 비트플래인 블록의 움직임 벡터와 현재 비트플래인 블록의 모드, 그리고 "CAE 디코딩" 과정에서 사용될 비트스트림을 출력한다.

[0142] 여기에서 비트플레인 블록의 모드는 다양한 방법으로 디코딩할 수 있다.

[0143] (1) 하나의 일 실시 예로서, 현재 비트플레인 블록의 모드를 고정길이복호화를 통해 디코딩할 수 있다.

[0144] (2) 또 다른 실시 예로서, 현재 비트플레인 블록의 모드를 비트플래인 블록 모드의 출현 빈도에 따른 확률을 이용한 가변길이복호화를 통해 디코딩할 수 있다. 이때 미리 계산된 확률 테이블을 이용할 수 있다.

[0145] (3) 또 다른 실시 예로서, 현재 비트플레인 블록의 모드를 현재 비트플래인 블록에 인접하는 주변의 비트플래인 블록 모드를 고려하여 상황에 맞게 적응적으로 변화된 확률을 이용한 산술 코딩 방법을 통해 디코딩할 수 있다.

[0146] 이 외에도 위와 같이 비트플레인 블록 모드를 디코딩하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0147] 입력 받은 현재 비트플래인 블록의 모드에 따라 디코딩을 수행하게 된다. 만약 비트플래인 블록의 모드가 'No Update Without MV'라면, 움직임 벡터 예측값($MV_P$)을 움직임 벡터로 하여 "움직임 보상"을 수행하여 움직임 벡터에 대응하는 참조 비트플래인 블록을 출력한다.

[0148] 여기에서 디코딩할 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 결정하는 방법은 다양하게 적용될 수 있다.

[0149] (1) 하나의 일 실시 예로서, 현재 디코딩할 비트플레인 블록에 인접하는 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터들을 이용하여 현재 비트플래인 블록의 움직임 벡터 예측값($MV_p$)을 결정할 수 있다.

[0150] (2) 또 다른 실시 예로서, 도 12의 순서도에서 현재 디코딩할 비트플래인 블록과 인접하는 주변 비트플래인 블록 대신 깊이 정보맵 블록의 움직임 벡터를 이용하여 현재 비트플래인 블록의 움직임 벡터 예측값($MV_p$)을 결정할 수 있다.

[0151] (3) 또 다른 실시 예로서, 현재 디코딩할 비트플레인 블록에서 좌측 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{Left}$)와 상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{Above}$), 그리고 우측상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{RightAbove}$)의 중간값(Median)을 이용하여 현재 비트플래인 블록의 움직임 벡터 예측값($MV_p$)을 결정할 수 있다. 여기에서 우측상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{RightAbove}$)가 유효하지 않다면, 좌측상단 비트플래인 블록 혹은 깊이 정보맵 블록의 움직임 벡터($MV_{LeftAbove}$)를 대신 사용할 수 있다.

[0152] 이 외에도 위와 같이 디코딩할 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 결정하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0153] 만약 비트플래인 블록의 모드가 'No Update With MV'라면, 전송 받은 움직임 벡터값을 이용하여 "움직임 보상"을 수행하여 움직임 벡터에 대응하는 참조 비트플래인 블록을 출력한다. 만약 비트플래인 블록의 모드가 'all_0' 이라면, "동일 레벨 블록 디코딩"이 수행이 되며, 블록 내의 모든 픽셀값을 '0'으로 설정하여 출력한다. 만약 비트플래인 블록의 모드가 'all_1' 이라면, "동일 레벨 블록 디코딩"이 수행이 되며, 블록 내의 모든 픽셀값을 '1'로 설정하여 출력한다. 만약 비트플래인 블록의 모드가 'intraCAE' 라면, "CAE(Context-based Arithmetic Encoding) 디코딩"이 수행되며, 현재 비트플래인 블록에서 디코딩을 수행할 각 픽셀의 주변 픽셀 정보를 기반으로 이진 산술 코딩(Binary Arithmetic Coding)을 수행한다. 만약 비트플래인 블록의 모드가 'interCAE' 라면, "CAE(Context-based Arithmetic Encoding) 디코딩"이 수행이 되며, 현재 비트플래인 블록에서 디코딩을 수행할 각 픽셀의 주변 픽셀 정보와 현재 픽셀에 대응하는 참조 영상의 픽셀과 그 주변 픽셀 정보를 기반으로 하는 이진 산술 디코딩(Binary Arithmetic Deoding)을 수행한다.

[0154] 현재 비트플래인 블록의 모드가 'intraCAE'와 'interCAE'로 선택 되었을 경우, 컨텍스트 기반 산술 인코딩(CAE, Context-based Arithmetic Encoding) 방법을 사용하며, CAE 디코딩의 구조도의 일 예는 도 23과 같다.

[0155] 도 23의 "변환비율 디코딩" 과정에서는 입력된 비트스트림에서 변환비율(CR)을 디코딩하여 출력한다. "다운샘플링" 과정에서는 입력된 변환비율에 따라 참조 비트플래인 블록에 대한 다운샘플링을 수행하여 컨텍스트 계산에 사용할 수 있도록 한다. 만약 변환비율(CR)이 '1' 이라면 다운샘플링을 수행하지 않는다. "컨텍스트 계산" 과정에서 컨텍스트 템플릿(context template)의 구성은 상기 도 13의 "컨텍스트 계산"에서 설명된 것과 동일하다. "산술 디코딩" 과정에서는 "컨텍스트 계산" 과정에서 구성된 컨텍스트 템플릿을 인덱스(index)로 하는 확률 테이블을 참조하여 입력된 비트스트림을 산술 디코딩을 수행하여 현재 픽셀을 생성한다.

[0156] 여기에서 'intraCAE' 모드와 'interCAE' 모드에서 컨텍스트 템플릿을 구성하는 방법은 다양하게 적용할 수 있다.

[0157] (1) 하나의 일 예로서, 'intraCAE' 모드에서 비트플래인 블록의 코딩할 픽셀(X)과 인접하는 주변의 픽셀들 중 다양한 개수의 픽셀들을 이용하여 컨텍스트 템플릿을 구성할 수 있다. 실시 예로서, 'intraCAE' 모드에서 현재 픽셀(X)을 중심으로 인접하는 4개의 주변 픽셀들을 (c3 c2 c1 c0) 형태의 4비트 컨텍스트 템플릿을 구성할 수 있다.

[0158] (2) 또 다른 실시 예로서, 'interCAE' 모드에서 현재 비트플래인 블록의 코딩할 픽셀(X)과 인접하는 주변의 픽셀들과 현재 비트플래인 블록의 코딩할 픽셀(X)에 대응하는 참조 비트플래인 영상의 픽셀과 그 주변 픽셀들 중 다양한 개수의 픽셀들을 이용하여 컨텍스트 템플릿을 구성할 수 있다. 실시 예로서, 'interCAE' 모드에서 현재 픽셀(X)을 중심으로 인접하는 4개의 주변 픽셀들을 (c3 c2 c1 c0) 형태의 4비트 컨텍스트 템플릿을 구성할 수 있다.

[0159] 이 외에도 위와 같이 컨텍스트 템플릿을 구성하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0160] "업샘플링" 과정에서는 디코딩된 비트플래인 블록에 업샘플링(Up-sampling)을 수행하여 재구성된 비트플래인 블록을 생성한다. 만약 변환비율이 '1'이라면 업샘플링을 수행하지 않는다.

[0161] 도 22의 "비트플래인 디코딩"을 수행하는 방법의 또 다른 실시 일 예로 상기 살펴본 MPEG-4 Part 2 Visual의 이진 형상 디코딩 방법을 응용하는 방법이 있다.

[0162] ① MPEG-4 Part 2 Visual의 이진 형상 디코딩 방법을 응용하는 방법의 일 예로 "비트율 조절" 부분의 디코딩된 변환비율(CR)을 다양하게 적용하여 디코딩을 수행할 수 있다.

[0163] ② MPEG-4 Part 2 Visual의 이진 형상 디코딩 방법을 응용하는 방법의 또 다른 일 예로 블록 단위에서 하나 이상의 특정 비트플래인에 저장되어 있는 움직임 정보를 각 비트플래인에서 선택적으로 사용할 수 있도록 하는 방법이 있다.

[0164] 도 22의 "비트플래인 디코딩"에서 "CAE디코딩"을 수행하는 방법은 [방법 #1] 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인 디코딩 방법과 [방법 #2] 컨텍스트 기반 적응적 산술 코딩을 응용한 비트플래인 디코딩 방법, 그리고 [방법 #3] CAC(Constant Area Coding) 혹은 JBIG(Joint Bi-Level Image Processing Group)의 방법 혹은 Quad Tree 방법을 이용한 비트플래인 디코딩 방법으로 대체되어 수행될 수 있다.

[0165] [방법 #1] 도 22의 "비트플래인 코딩"에서 "CAE인코딩"을 수행하는 방법 대신에 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용하여 비트플래인 블록을 코딩할 수 있는데 제안하는 방법의 일 예는 도 24와 같다.

[0166] 도 24는 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding)방법을 사용하여 비트플래인 블록을 디코딩하는 방법을 나타내며, 비트스트림을 입력받아 비트플래인 디코딩을 수행하여 재구성된 비트플래인 블록을 출력한다. "가변길이 디코딩" 과정에서는 입력된 비트스트림을 가변길이 디코딩하여 런(Run)의 길이들을 출력한다. "비트플래인 블록 구성" 과정에서는 비트스트림의 스캔 순서에 대한 정보와 런(Run)의 길이들을 이용하여 비트플래인 블록을 구성한다.

[0167] 도 24의 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 사용하여 비트플래인 블록을 디코딩하는 방법에서 입력된 비트스트림을 각 서브 블록으로 나누어서 디코딩을 수행할 수 있으며, 제안하는 방법의 일 예는 도 25와 같다.

[0168] 도 25의 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인코딩 방법에서 비트플래인 블록을 서브 블록으로 나누어서 디코딩하는 방법을 나타내며, 비트스트림을 입력 받아 각 서브 블록으로 나누어서 디코딩을 수행하여 재구성된 비트플래인 블록을 출력한다. "가변길이 디코딩" 과정과 "서브 블록 구성" 과정은 나누어진 서브 블록의 개수만큼 반복 수행된다. "가변길이 디코딩" 과정에서는 입력된 비트스트림을 가변길이 디코딩하여 각 서브 블록에 대한 런(Run)의 길이들을 출력한다. "서브 블록 구성" 과정에서는 비트스트림의 스캔 순서에 대한 정보와 런(Run)의 길이들을 이용하여 서브 비트플래인 블록을 구성한다. "비트플래인 블록 구성" 과정에서는 서브 비트플래인 블록들을 이용하여 비트플래인 블록을 출력한다.

[0169] **<디코더>**

[0170] 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 디코더 구조도의 일 예는 도 26과 같다.

[0171] 도 26에서 입력된 비트스트림은 비트플래인 디코딩 모드 혹은 기존 디코딩 모드를 통해 디코딩되어 재구성된 깊이 정보맵 블록을 출력한다. 비트플래인 디코딩 모드가 선택되었을 경우에는 'C'로 표시되어 있는 부분이 수행되며, 'C'는 비트스트림을 입력 받아 비트플래인 수만큼 반복해서 비트플래인 디코딩을 수행하고 각각의 비트플래인 영상을 결합하여 N-bit로 표현되는 재구성된 깊이 정보맵 블록을 출력한다. 기존 디코딩 모드가 선택되었을 경우에는 'D'로 표시되어 있는 부분이 수행되며, 'D'는 비트스트림을 입력받아 기존의 동영상 코딩 방법(MPEG-1, MPEG-2, MPEG-4 Part 2 Visual, H.264/AVC, SVC, MVC, VC-1, AVS, KTA, 등)으로 디코딩하여 재구성된 깊이 정보맵 블록을 출력한다.

[0172] 도 26에서 비트플래인 디코딩 모드 'C'는 비트플래인 단위 코딩 방법의 디코더 구조도를 보여준다. 'C'의 디코더 구조도에서 데이터를 처리하는 단위는 가로 세로 16x16픽셀 크기의 블록이다. 도 26의 참조 영상 버퍼에 저장되어 있는 참조영상과 재구성된 영상 버퍼에 저장되어 있는 재구성된 영상은 "비트플래인 디코딩" 과정으로 입력된다. "비트플래인 디코딩" 과정에서는 비트스트림을 입력 받아 인코딩된 비트플래인 수만큼 반복해서 수행이 되며, 입력된 비트스트림을 각 비트플래인 별로 디코딩하여 n개의 비트플래인 영상을 출력한다. "비트플래인 디코딩"을 수행하는 방법은 도 22의 "비트플래인 디코딩" 방법을 이용한다. "비트플래인 결합" 과정에서는 출력된 각각의 비트플래인 영상을 결합하여 n-bit로 표현되는 영상을 출력한다. "역그레이 코드 변환" 과정에서는 그레이 코드로 표현된 영상을 원래의 형태로 복원한다. "깊이 정보맵 구성" 과정에서는 n-bit로 표현되는 영상을 실제 깊이 정보맵의 비트수에 맞게 구성하여 깊이 정보맵을 출력한다.

[0173] 여기에서 깊이 정보맵을 구성하는 방법은 다양하게 적용될 수 있다.

[0174] (1) 하나의 일 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기위해 디코딩되지 않은 비트플래인 블록의 모든 값을 '0(0)'으로 설정하여 구성할 수 있다.

[0175] (2) 또 다른 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기 위해 디코딩되지 않은 비트플래인 블록의 모든 값을 '255(1)'로 설정하여 구성할 수 있다.

[0176] (3) 또 다른 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기 위해 디코딩되지 않은 비트플래인 블록을 모두 '0(0)'으로 구성할지 혹은 모두 '255(1)' 구성할지에 대한 정보를 디코딩하여 디코딩되지 않은 비트플래인 블록을 구성할 수 있다.

[0177] (4) 또 다른 실시 예로서, 현재 블록을 실제 깊이 정보맵의 비트수에 맞게 구성하기 위해 디코딩되지 않은 비트플래인 블록을 인접하는 주변 비트플래인 블록의 픽셀값을 이용하여 구성할 수 있다. 여기에서 비트스트림으로부터 해당 패딩(Padding) 방향에 대한 정보를 디코딩하여 주변 비트플래인 블록의 픽셀값을 이용한 패딩(Padding)을 통해 비트플래인 블록을 구성할 수 있다.

[0178] 이 외에도 위와 같이 깊이 정보맵을 구성하는 방법은 상식적으로 다양한 방법으로 변경될 수 있다.

[0179] 구성된 깊이 정보맵은 기존 디코딩 모드 'D'에서 인트라 예측에 사용되거나 디블록킹 필터를 거쳐 재구성된 영상 버퍼에 저장되거나 "재구성된 깊이 정보맵"으로 출력된다. 이때 비트스트림으로부터 디코딩된 현재 디코딩된 블록에 대한 디블록킹 필터의 수행 여부 정보를 통해 현재 디코딩된 블록에 디블록킹 필터를 수행할 수 있다. 여기에서 구성된 깊이 정보맵은 디블록킹 필터를 거치지 않고 재구성된 영상 버퍼에 저장될 수 있다. 또한 구성된 깊이 정보맵은 디블록킹 필터를 거친 후 인트라 예측에 사용될 수 있다.

[0180] 도 26에서는 기존 디코딩 모드 'D'의 일 예로 H.264/AVC의 디코더를 보여준다. 'D'의 디코더 구조도에서 데이터를 처리하는 단위는 가로 세로 16x16픽셀 크기의 매크로블록(Macroblock)이며, 인트라 모드 또는 인터 모드로 디코딩이 수행된다. 인트라(Intra) 모드일 경우, 'D' 내부의 스위치가 인트라로 전환이 되며, 인터(Inter) 모드일 경우에는 스위치가 인터로 전환이 된다. 디코딩 과정의 주요한 흐름은 먼저 예측 블록을 생성한 후, 입력 받은 비트스트림을 디코딩한 결과 블록과 예측 블록을 더하여 재구성된 깊이 정보맵 블록을 생성하는 것이다. 먼저 예측 블록의 생성은 인트라 모드와 인터 모드에 따라 수행이 된다. 먼저 인트라 모드일 경우에는 "인트라 예측" 과정에서 현재 블록의 이미 코딩된 주변 픽셀값을 이용하여 공간적 예측을 수행하여 예측 블록을 생성하며,

인터 모드일 경우에는 움직임 벡터를 이용하여 참조 영상 버퍼에 저장되어 있는 참조 영상에서 해당 영역을 찾아 "움직임 보상"을 수행함으로써 예측 블록을 생성한다. "엔트로피 디코딩" 과정에서는 입력된 비트스트림을 확률 분포에 따른 엔트로피 디코딩을 수행하여 양자화된 계수(Quantized Coefficient)를 출력한다. 양자화된 계수를 "역양자화"과정과 "역변환"을 수행하여 예측 영상과 가산기를 통해 재구성된 블록을 생성한 다음 "디블록킹 필터"를 통해 블록킹 현상(Blocking Artifact)를 제거한 후, "재구성된 영상 버퍼"에 저장한다.

[0181] **<모드 선택 방법>**

[0182] 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법을 적응적으로 선택할 수 있는 모드 선택 방법이 필요한데, 본 발명에서 제안하는 모드 선택 방법의 일 예는 도 27과 같으며, 자세한 알고리즘은 다음과 같다.

[0183] Step 1) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드의 코스트($Cost_A$)를 계산하고 또한 비트플래인 코딩 모드의 코스트($Cost_B$)를 계산한다.

[0184] Step 2) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드의 코스트($Cost_A$)가 비트플래인 코딩 모드의 코스트($Cost_B$)보다 작다면, Step 3로 분기한다. 만약 그렇지 않다면, Step 4로 분기한다.

[0185] Step 3) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드로 코딩한다.

[0186] Step 4) 현재 깊이 정보맵 블록을 비트플래인 코딩 모드로 코딩한다.

[0187] 도 27의 코딩 모드 결정 순서도에서 각 코딩 모드에 대한 코스트($Cost_i$)는 다양한 방법으로 계산할 수 있는데, 코스트를 계산하는 방법의 일 예로 "Lagrangian optimization technique"을 사용할 수 있으며, 일 예로 *SAD*(sum of absolute difference)를 사용하는 방법의 해당하는 수식은 수학식 4와 같다.

*수학식 4*

[0188] $$Cost_i = SAD_i + \lambda_{MODE} \cdot R_i, \qquad \lambda_{MODE} = c \cdot (QUANT)^2$$

[0189] $SAD_i$ (sum of absolute difference)는 원본 영상과 재구성된 영상 간의 예측 오차의 절대값의 합을 의미하고, $\lambda_{MODE}$는 양자화 파라미터의 제곱에 상수 $C(C= 0.85)$를 곱하여 생성된 값으로 매크로블록 모드에 대한 "Lagransian" 상수를 나태내며, $R_i$는 해당하는 매크로블록 모드로 실제 코딩을 수행하였을 때의 발생 비트량을 의미한다. 그리고 코스트를 계산하는 방법의 또 다른 일 예로 *SSD*(sum of square difference)를 사용할 수도 있다.

[0190] **<비트플래인 코딩 시 비트율 조절 방법>**

[0191] 비트플래인 단위 코딩에서 비트량 조절을 위한 방법으로 깊이 정보맵 블록 혹은 분리된 비트플래인의 이진 블록에 다운샘플링을 적용하여 비트플래인 코딩을 수행하거나 깊이 정보맵 블록을 비트플래인 단위로 분리한 후 분리된 비트플래인의 이진 블록들 중 특정 비트플래인의 이진 블록을 코딩하지 않는 방법을 이용할 수 있다. 여기에서 특정 비트플래인의 이진 블록을 코딩하지 않는 방법을 이용할 때 깊이 정보맵을 그레이 코드로 변환했을 때 정보의 손실이 발생을 한다면 그레이 코드를 기존의 깊이 정보맵으로 복원할 때, 정확한 값을 복원할 수 없다. 따라서 "비트플래인 코딩" 과정에서 비트량 조절을 위해 정보의 손실을 허용하기 위해서는 MSB 비트플래인부터 LSB 비트플래인의 순서로 차례대로 코딩할 비트플래인의 수를 증가시키면서 코딩을 수행해야 한다.

[0192] 도 8의 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법에서 비트플래인 단위 코딩 시 비트율 조절 과정은 [방법 #1] 깊이 정보맵 블록 혹은 분리된 비트플래인의 이진 블록에 다운샘플링을 적용하여 비트율을 조절하는 방법, [방법 #2] 깊이 정보맵 블록을 비트플래인 단위로 분리한 후 분

리된 비트플래인의 이진 블록들 중 특정 비트플래인의 이진 블록을 코딩하지 않는 방법을 통해 비트율을 조절하는 방법, [방법 #3] [방법 #1]과 [방법 #2]를 조합하여 비트율을 조절을 적용할 수 있다.

[0193] [방법 #1] 깊이 정보맵 블록 혹은 분리된 비트플래인의 이진 블록에 다운샘플링을 적용하여 비트율을 조절하는 방법의 일 예는 도 28과 같다.

[0194] 도 28은 깊이 정보맵 블록을 입력받아 깊이 정보맵 블록 혹은 분리된 비트플래인의 이진 블록에 다운샘플링을 적용하여 다운 샘플링된 깊이 정보맵 블록을 출력하거나 다운 샘플링된 비트플래인 블록들을 출력한다. "비트플래인 분리" 과정에서는 n-bit로 표현되는 깊이 정보맵 블록을 입력 받아서 n개의 비트플래인 블록으로 분리한다. "샘플링 레이트 결정" 과정에서는 깊이 정보맵 블록 혹은 비트플래인 단위로 분리된 비트플래인 블록들을 입력받아 최적의 샘플링 레이트(sampling rate)를 결정한다. 최적의 샘플링 레이트(sampling rate)를 결정하는 방법은 추후에 자세히 설명한다. "다운샘플링" 과정에서는 깊이 정보맵 블록을 입력 받아서 수평 방향 혹은 수직 방향으로 최적의 샘플링 레이트(sampling rate)에 따라 다운샘플링을 수행한다. 여기에서 깊이 정보맵 블록을 다운 샘플링 할 때 필터를 적용하여 다운샘플링을 수행하는 방법을 사용할 수 있고, 도 29와 같이 필터를 적용하지 않고 다운샘플링을 수행하는 방법을 사용할 수 있다.

[0195] 도 29에서 왼쪽 영상은 원본 영상이고, 오른쪽 영상은 수평 수직 방향으로 1/2 다운샘플링을 수행한 결과이다. 원본 영상(도 29의 왼쪽 영상)에서의 흰색과 검은색 블록은 픽셀을 의미하며, 검은색 블록은 다운샘플링된 영상을 구성하기 위해 사용될 픽셀을 의미한다. 여기에서 각 픽셀은 깊이 정보맵 블록의 픽셀값이 될 수 있으며, 또한 비트플래인 블록의 픽셀값이 될 수 있다.

[0196] 도 28의 "샘플링 레이트 결정" 과정에서 최적의 샘플링 레이트(sampling rate)를 결정하는 방법의 순서도의 일 예는 도 30과 같고, 자세한 알고리즘은 다음과 같다.

[0197] Step 1) 샘플링 레이트(sampling rate)를 '1'로 초기화한다.

[0198] Step 2) 샘플링 레이트(sampling rate)에 1/2을 곱한다.

[0199] Step 3) 설정된 샘플링 레이트(sampling rate)를 이용하여 수평 또는 수직 방향으로 다운샘플링을 수행한다.

[0200] Step 4) 다운샘플링된 블록에 비트플래인 코딩을 수행한 후 코스트($Cost_i$) 값을 계산한다. 코스트($Cost_i$) 값을 계산하는 방법은 상기 절에 자세히 설명되어 있다.

[0201] Step 5) 만약 현재 설정된 샘플링 레이트(sampling rate)가 가장 낮은 샘플링 레이트(sampling rate)라면, Step 6으로 분기한다. 만약 그렇지 않다면 Step 2로 분기한다. 이때 가장 낮은 샘플링 레이트(sampling rate)는 사용자에 의해 임의의 값으로 설정될 수 있다.

[0202] Step 6) 만약 모든 다운샘플링된 블록 중에서 현재 다운샘플링된 블록의 코스트($Cost_i$) 값이 최소라면, Step 7으로 분기한다. 만약 그렇지 않다면 Step 2로 분기한다.

[0203] Step 7) 현재 다운샘플링된 블록의 샘플링 레이트(sampling rate)를 최적의 샘플링 레이트로 설정한다.

[0204] [방법 #2] 현재 깊이 정보맵 블록에 대한 비트율 조절을 위해 깊이 정보맵 블록을 비트플래인 단위로 분리한 후 분리된 비트플래인의 이진 블록들 중 특정 비트플래인의 이진 블록을 코딩하지 않는 방법을 통해 비트율을 조절할 수 있으며, 일 예로 코딩할 비트플래인을 결정하는 방법을 사용할 수 있다. 이때 코딩할 비트플래인을 결정하는 방법은 그레이 코드 변환된 깊이 정보맵의 특성 때문에 MSB 비트플래인부터 LSB 비트플래인의 순서로 차례대로 코딩할 비트플래인을 결정해야 한다. 따라서 코딩할 비트플래인을 결정하는 방법은 코딩할 비트플래인의 수를 결정하는 방법이라 할 수 있다. 코딩할 비트플래인의 수를 결정하는 방법은 비트율-왜곡 최적화 방법을 이용하여 블록 단위로 결정할 수 있으며, 코딩할 비트플래인의 수에 대한 정보는 고정길이 부호화 또는 양자화 파라미터 및 주변 상황에 따른 가변길이 부호화를 통해 코딩되어 비트스트림에 포함된다.

[0205] 코딩할 비트플래인의 수를 결정하는 방법의 순서도의 일 예는 도 31과 같고, 자세한 알고리즘은 다음과 같다.

[0206] Step 1) 코딩할 비트플래인의 수(N)를 '0'으로 설정한다.

[0207] Step 2) 코딩할 비트플래인의 수(N)를 '1' 증가시킨다.

[0208] Step 3) 코딩할 비트플래인의 수(N)에 따라 MSB 비트플래인부터 LSB 비트플래인까지 차례대로 코딩 모드를 설정한다. 예를 들어 N이 '1'일 경우는 MSB 비트플래인만 코딩하는 모드이고, N이 '2'일 경우에는 MSB와 MSB-1 비트플래인을 코딩하는 모드이고, N이 '3'개일 경우에는 MSB와 MSB-1, MSB-2 비트플래인을 코딩하는 모드이고, N이 'm'일 경우에는 MSB와 MSB-1, MSB-2, MSB-3, … , MSB-(m-1) 비트플래인을 코딩하는 모드이다.

[0209] Step 4) 해당 코딩 모드로 코딩을 수행한 후 해당 코딩 모드의 코스트($Cost_i$) 값을 계산한다. 코스트($Cost_i$) 값을 계산하는 방법은 상기 절에 자세히 설명되어 있다.

[0210] Step 5) 만약 해당 코딩 모드가 모든 코딩 모드 중 코스트($Cost_i$) 값이 최소라면, Step 6으로 분기한다. 만약 그렇지 않다면 Step 2로 분기한다.

[0211] Step 6) 가장 최소의 코스트($Cost_i$) 값을 가진 코딩 모드를 통해 코딩할 비트플래인의 수를 결정한다.

[0212] [방법 #3] 현재 깊이 정보맵 블록에 대한 비트율 조절을 위해 [방법 #1]과 [방법 #2]를 조합하여 비트율을 조절할 수 있다. 실시 일 예로 코딩할 깊이 정보맵에 [방법 #1]을 수행하여 다운샘플링된 깊이 정보맵을 생성하고 [방법 #2]를 수행하여 다운샘플링된 깊이 정보맵의 코딩할 비트플래인의 수를 결정함으로써 비트율 조절을 수행할 수 있다.

[0213] **<제안하는 코딩 방법의 전체적인 코딩 흐름 및 실시 구현의 일 예>**

[0214] 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 전체적인 코딩 흐름은 도 32와 같으며, 자세한 알고리즘은 다음과 같다.

[0215] Step 1) 현재 깊이 정보맵 블록에 대해 비트플래인 단위 코딩을 수행할지 기존 방법으로 코딩을 수행하지에 대한 모드 결정을 수행한다. 수행하는 방법의 일 예로 상기 도 27의 방법을 사용할 수 있다.

[0216] Step 2) 모드 결정 과정에서 비트플래인 코딩 모드가 선택이 되었다면, 비트플래인 코딩을 수행할지에 대한 플래그 정보인 'bitplane_coding_flag'를 '1'로 설정하고, 만약 기존 코딩 모드로 선택이 되었다면 'bitplane_coding_flag'를 '0'으로 설정한다.

[0217] Step 3) 만약 bitplane_coding_flag가 '0'이라면, Step 4로 분기한다. 만약 그렇지 않다면 Step 5로 분기한다.

[0218] Step 4) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드로 코딩한다.

[0219] Step 5) 현재 깊이 정보맵 블록을 비트플래인 코딩 모드로 코딩한다.

[0220] 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법에서 비트플래인 단위 코딩 시 비트율 조절 방법으로 코딩할 비트플래인의 수를 결정하는 방법의 전체적인 코딩 흐름은 도 33과 같으며, 자세한 알고리즘은 다음과 같다. 이때 도 33은 깊이 정보맵이 8-bit로 표현되어 있는 경우의 실시 일 예이다.

[0221] Step 1) 코딩할 비트플래인의 수를 결정하며, 수행하는 방법의 일 예로 도 31의 방법을 사용할 수 있다.

[0222] Step 2) 코딩할 비트플래인의 수(bitplane_num)를 설정한다.

[0223] Step 3) 만약 bitplane_num이 '8'이 아니라면, Step 4로 분기한다. 만약 그렇지 않다면 MSB, MSB-1, MSB-2, MSB-3, MSB-4, MSB-5, MSB-6, LSB 비트플래인을 코딩한다.

[0224] Step 4) 만약 bitplane_num이 '7'이 아니라면, Step 5로 분기한다. 만약 그렇지 않다면 MSB, MSB-1, MSB-2, MSB-3, MSB-4, MSB-5, MSB-6 비트플래인을 코딩한다.

[0225] Step 5) 만약 bitplane_num이 '6'이 아니라면, Step 6으로 분기한다. 만약 그렇지 않다면 MSB, MSB-1,

MSB-2, MSB-3, MSB-4, MSB-5 비트플래인을 코딩한다.

[0226] Step 6) 만약 bitplane_num이 '5'가 아니라면, Step 7로 분기한다. 만약 그렇지 않다면 MSB, MSB-1, MSB-2, MSB-3, MSB-4 비트플래인을 코딩한다.

[0227] Step 7) 만약 bitplane_num이 '4'가 아니라면, Step 8로 분기한다. 만약 그렇지 않다면 MSB, MSB-1, MSB-2, MSB-3 비트플래인을 코딩한다.

[0228] Step 8) 만약 bitplane_num이 '3'이 아니라면, Step 9로 분기한다. 만약 그렇지 않다면 MSB, MSB-1, MSB-2 비트플래인을 코딩한다.

[0229] Step 9) 만약 bitplane_num이 '2'가 아니라면, MSB 비트플래인을 코딩한다. 만약 그렇지 않다면 MSB, MSB-1 비트플래인을 코딩한다.

[0230] 상기 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법을 국제 동영상 표준인 H.264/AVC에 실제 구현한 일 예로 Macroblock layer(macroblock_layer) 신택스를 다음의 표 1 및 표 2와 같이 수정하여 구현할 수 있다.(여기서, 표 1과 표 2는 원래 연속된 하나이나, 지면의 한계로 이를 분할하여 표현한다)

표 1

| macroblock_layer( ) { | C | Descriptor |
|---|---|---|
| if ( ! mb_skip_flag ) | | |
| **bitplane_coding_flag** | 2 | u(1) \| ae(v) |
| if (!bitplane_coding_flag) { | | |
| **mb_type** | 2 | ue(v) \| ae(v) |
| if( mb_type == I_PCM ) { | | |
| while( !byte_aligned( ) ) | | |
| **pcm_alignment_zero_bit** | 2 | f(1) |
| for( i = 0; i < 256; i++ ) | | |
| **pcm_sample_luma[ i ]** | 2 | u(v) |
| for( i = 0; i < 2 * MbWidthC * MbHeightC; i++ ) | | |
| **pcm_sample_chroma[ i ]** | 2 | u(v) |
| } else { | | |
| noSubMbPartSizeLessThan8x8Flag = 1 | | |
| if( mb_type != I_NxN && MbPartPredMode( mb_type, 0 ) != Intra_16x16 && NumMbPart( mb_type ) == 4 ) { | | |
| sub_mb_pred( mb_type ) | 2 | |
| for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ ) | | |
| if( sub_mb_type[ mbPartIdx ] != B_Direct_8x8 ) { | | |
| if( NumSubMbPart( sub_mb_type[ mbPartIdx ] ) > 1 ) | | |
| noSubMbPartSizeLessThan8x8Flag = 0 | | |
| } else if( !direct_8x8_inference_flag ) | | |
| noSubMbPartSizeLessThan8x8Flag = 0 | | |
| } else { | | |
| if( transform_8x8_mode_flag && mb_type == I_NxN ) | | |
| **transform_size_8x8_flag** | 2 | u(1) \| ae(v) |
| mb_pred( mb_type ) | 2 | |
| } | | |
| if( MbPartPredMode( mb_type, 0 ) != Intra_16x16 ) { | | |
| **coded_block_pattern** | 2 | me(v) \| ae(v) |
| if( CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && ( mb_type != B_Direct_16x16 \|\| direct_8x8_inference_flag ) ) | | |
| **transform_size_8x8_flag** | 2 | u(1) \| ae(v) |
| } | | |
| if( CodedBlockPatternLuma > 0 \|\| CodedBlockPatternChroma > 0 \|\| MbPartPredMode( mb_type, 0 ) == Intra_16x16 ) { | | |
| **mb_qp_delta** | 2 | se(v) \| ae(v) |
| residual( ) | 3 \| 4 | |
| } | | |
| }else | | |
| for( i=0; i < max_bitplane_num; i++ ){ | | |
| **xor_flag** | 2 | u(1) \| ae(v) |
| **bab_type** | 2 | u(v) \| ae(v) |
| if( bab_type = = NoUpdateWithMV \|\| bab_type = = InterCAE ){ | | |
| **mvd_x** | 2 | u(v) \| ae(v) |

[0231]

표 2

| | | |
|---|---|---|
| **mvd_y** | 2 | u(v) \| ae(v) |
| } | | |
| if( bab_type == IntraCAE \|\| bab_type == InterCAE ){ | | |
| **scan_type** | 2 | u(1) \| ae(v) |
| binary_arithmetic_code( ) | 3 \| 4 | |
| } | | |
| } | | |
| } | | |
| } | | |

[0232]

[0233] "bitplane_coding_flag"는 현재의 깊이 정보맵 블록이 비트플래인 코딩 모드로 코딩되었는지의 여부를 나타내며, "bitplane_coding_flag" 값이 '1'일 경우에는 비트플래인 코딩을 수행한 것이며, '0'일 경우에는 기존 방법으로 코딩을 수행한 것이다. 'bitplane_num'은 상기 도 31의 방법을 이용한 비트율 조절 방법으로 현재의 깊이 정보맵 블록에서 코딩된 비트플래인의 수를 나타낸다. 여기에서 상기 도 28의 방법을 이용하여 비트율 조절을 수행할 경우, 'bitplane_num'은 'downsampling_rate'로 변경되어 사용되며, 'downsampling_rate'는 현재의 깊이 정보맵 블록에 대한 수평 또는 수직 방향으로의 다운샘플링의 샘플링 레이트(sampling rate)를 나타낸다. 깊이 'bab_type' 은 상기 <인코더> 방법에서 비트플래인 단위 코딩을 수행하였을 때, 비트플래인 블록의

모드를 나타낸다. 'mvd_x'와 'mvd_y'는 현재 비트플래인 블록에서 움직임 벡터의 차분값($MV_D$; Motion Vector Difference)의 수평과 수직 성분을 나타낸다. 움직임 벡터의 차분값은 현재 비트플래인 블록의 움직임 벡터(MV)와 현재 비트플래인 블록의 움직임 벡터 예측값($MV_P$)을 차분함으로 구한다($MV_D = MV - MV_P$). 디코더에서는 전송받은 움직임 벡터의 차분값을 움직임 벡터 예측값과 합산하여 움직임 벡터를 계산한다($MV = MV_D + MV_p$). "scan_type"은 현재 비트플래인 블록을 컨텍스트 기반 산술 인코딩(CAE, Context-based Arithmetic Encoding)을 수행할 때, 수평 방향 스캔을 수행할지 수직 방향 스캔을 수행할지에 대한 플래그 정보로써, 'scan_type'이 0일 때는 수평(horizontal) 방향 스캔을 수행하며, 'scan_type'이 1일 때는 수직(vertical) 방향 스캔을 수행한다. "binary_arithmetic_code()"는 컨텍스트 기반 산술 인코딩(CAE, Context-based Arithmetic Encoding)을 통해 코딩된 데이터를 의미한다.

[0234] 본 발명에서 제안하는 방법의 우수성을 검증하기 위해 본 발명의 발명자는 H.264/AVC의 참조 소프트웨어인 JM(Joint Model) 13.2에 실제 구현하여 기존 방법과 제안하는 방법의 비교를 수행하였다. 비트플래인 단위 코딩에서 현재 비트플래인 코딩 모드는 "all_0", "all_1", "intraCAE" 세 가지 모드만 허용하였다. 기존 코딩 방법의 조건은 H.264/AVC의 Baseline Profile을 사용하였으며, Ballet XGA(1024x768) 15Hz 영상의 깊이 정보맵과 Breakdancers XGA(1024x768) 15Hz 영상의 깊이 정보맵에 대한 비교를 수행하였다.

[0235] 도 34, 도 35는 H.264/AVC를 이용한 기존 방법(Anchor)과 제안하는 방법(Proposed)에 대한 비트율 대비 PSNR(Peak Singal-to-Noise Ratio)에 대한 RD(Rate-distortioon)-Curve를 나타낸다. 그래프 결과를 통해 본 발명에서 제안하는 방법의 결과가 기존 방법의 결과보다 성능이 월등하게 향상된 것을 확인할 수 있으며, 평균적인 PSNR 향상을 나타내는 BD-PSNR 방법을 사용하여 성능을 비교하였을 때 0.87 dB ~ 0.89 dB 향상되었고, 평균적인 bit-rate 감소량을 나타내는 BD-rate 방법을 사용하여 성능을 비교하였을 때 12.62 % ~ 12.89 % 감소하였다.

[0236] <발명의 장치>

[0237] 본 발명에서 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 인코더 장치도는 도 36과 같다.

[0238] 도 36에서 입력된 영상은 n-bit로 표현되는 깊이 정보맵이며, 출력은 깊이 정보맵을 블록으로 나눈 후, 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법을 이용한 비트플래인 단위 코딩 방법과 기존의 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법으로 코딩된 결과인 비트스트림이다. "참조 영상 버퍼부"에는 시간 방향으로 이전에 코딩되고 디코딩된 깊이 정보맵이 저장된다. "재구성된 영상 버퍼부"에는 현재 깊이 정보맵을 인코딩하고 디코딩한 깊이 정보맵이 저장된다. "모드 결정부"에서는 n-bit로 표현되는 깊이 정보맵 블록을 입력 받아 상기 도 27의 모드 선택 방법(블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법을 적응적으로 선택할 수 있는 모드 선택 방법)을 수행하여 코딩 모드에 대한 정보를 비트스트림으로 출력한다.

[0239] "모드 결정부"에서 기존 코딩 모드가 선택되었을 경우, 기존 코딩 방법인 도 36의 'A'가 수행된다. "예측부"에서는 "참조 영상 버퍼부"에 저장된 참조 영상과 "디코딩부"의 결과 영상과 비트플래인 코딩으로 인코딩하고 디코딩한 영상을 입력 받아 인트라(Intra) 및 인터(Inter) 모드에 따라 예측 블록을 출력하여 "인코딩부"에 전달한다. "인코딩부"에서는 예측블록과 깊이 정보맵을 입력 받아 인코딩을 수행해서 비트스트림을 출력한다. "디코딩부"에서는 예측블록과 "인코딩부"에서 인코딩된 비트스트림을 입력 받아 디코딩해서 재구성된 깊이 정보맵을 출력하고 "예측부"와 "디블록킹 필터부"로 전달한다. "디블록킹 필터부"에서는 코딩 과정에서 발생한 블록킹 현상(blocking artifact)을 제거한 후, "재구성된 영상 버퍼부"에 저장한다. "멀티플랙서부"에서는 비트플래인 코딩 모드를 수행하여 출력된 비트스트림과 기존 코딩 모드를 수행하여 출력된 비트스트림이 조합되어 하나의 비트스트림이 출력된다.

[0240] "모드 결정부"에서 비트플래인 코딩 모드가 선택되었을 경우, 비트플래인 단위 코딩 방법인 도 36의 'B'가 수행된다. "그레이 코드 변환부"에서는 깊이 정보맵 블록을 입력 받아 그레이 코드로의 변환을 수행한다. "비트플래인 분리부"에서는 N-bit로 표현되는 깊이 정보맵 블록을 입력 받아 N개의 비트플래인 블록으로 분리한다. "비트

율 조절부"는 옵션에 따라 사용할 수도 있고 사용하지 않을 수도 있다. "비트율 조절부"에서는 깊이 정보맵 블록을 입력 받아 상기 도 28의 방법이나 도 31의 방법으로 코딩을 수행할 비트플래인을 선택하며, 코딩을 수행할 비트플래인의 수에 대한 정보는 비트스트림으로 출력된다. 참조 영상 버퍼에 저장되어 있는 참조 영상과 재구성된 영상 버퍼에 저장되어 있는 재구성된 영상은 "비트플래인 코딩부"로 입력된다. "비트플래인 코딩부"에서는 입력된 참조 영상과 재구성된 영상을 통해 코딩할 비트플래인의 수만큼 반복해서 각각의 비트플래인을 인코딩하고 그 결과를 비트스트림으로 출력한다. 비트플래인 블록을 코딩하는 방법은 상기 도 9에서의 MPEG-4 Part 2 Visual의 형상 코딩 방법을 응용한 비트플래인 코딩 방법을 사용할 수 있다. 비트플래인 코딩이 선택되어 "비트플래인 코딩"을 수행하고, 코딩을 수행한 데이터는 이후에 입력되는 깊이 정보맵 블록의 코딩을 위해 "비트플래인 디코딩부"를 수행한다. "비트플래인 디코딩부"는 인코딩된 비트스트림을 입력 받아 비트플래인 수만큼 반복해서 수행되며, 입력된 비트스트림을 각 비트플래인 별로 디코딩하여 n개의 비트플래인 영상을 출력한다. "비트플래인 디코딩부"는 상기 도 22에서 MPEG-4 Part 2 Visual의 형상 디코딩 방법을 응용한 비트플래인 코딩 방법을 사용할 수 있다. "비트플래인 결합부"에서는 "비트플래인 디코딩부"에서 출력된 N개의 비트플래인들을 N-bit에 맞게 결합을 수행한다. "역 그레이 코드 변환부"에서는 그레이 코드로 표현된 깊이 정보맵 블록을 원래의 형태로의 복원을 수행한다. "깊이 정보맵 구성부"는 실제 깊이 정보맵의 비트수에 맞게 구성하여 재구성된 깊이 정보맵을 출력한다.

[0241] 본 발명에서 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 디코더 장치도는 도 37과 같다.

[0242] 도 37에서 입력은 비트스트림이며, 출력은 깊이 정보맵을 블록으로 나눈 후, 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법을 이용한 비트플래인 단위 코딩 방법과 기존의 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법으로 디코딩된 결과인 재구성된 깊이 정보맵이다. "참조 영상 버퍼부"에는 시간 방향으로 이전에 디코딩된 깊이 정보맵이 저장된다. "재구성된 영상 버퍼부"에는 현재 깊이 정보맵을 인코딩하고 디코딩한 깊이 정보맵이 저장된다. "코딩 모드 판단부"에서는 비트스트림을 입력받아 현재 디코딩할 깊이 정보맵의 코딩 정보를 디코딩하여, 비트플래인 디코딩 모드로 디코딩할지 아니면 기존 디코딩 모드로 디코딩할지 여부를 판단한다.

[0243] 비트플래인 디코딩 모드가 선택되었을 경우, 비트플래인 디코딩 방법인 도 37의 'C'가 수행된다. "비트플래인 디코딩부"는 비트스트림을 입력 받아 비트플래인 수만큼 반복해서 수행되며, 입력된 비트스트림을 각 비트플래인 별로 디코딩하여 n개의 비트플래인 영상을 출력한다. "비트플래인 디코딩부"는 상기 도 22에서 MPEG-4 Part 2 Visual의 형상 디코딩 방법을 응용한 비트플래인 코딩 방법을 사용할 수 있다. "비트플래인 결합부"에서는 "비트플래인 디코딩부"에서 출력된 N개의 비트플래인들을 N-bit에 맞게 결합을 수행한다. "역 그레이 코드 변환부"에서는 그레이 코드로 표현된 깊이 정보맵 블록을 원래의 형태로의 복원을 수행한다. "깊이 정보맵 구성부"는 실제 깊이 정보맵의 비트수에 맞게 구성하여 재구성된 깊이 정보맵 블록을 출력한다.

[0244] 기존 디코딩 모드가 선택되었을 경우, 기존 디코딩 방법인 도 37의 'D'가 수행된다. "예측부"에서는 "참조 영상 버퍼부"에 저장된 참조 영상과 "디코딩부"의 결과 영상과 비트플래인 코딩으로 디코딩한 영상을 입력 받아 인트라(Intra) 및 인터(Inter) 모드에 따라 예측 블록을 출력하여 "디코딩부"에 전달한다. "디코딩부"에서는 예측블록과 비트스트림을 입력 받아 디코딩해서 재구성된 깊이 정보맵 블록을 출력하고 "예측부"와 "디블록킹 필터부"로 전달한다. "디블록킹 필터부"에서는 재구성된 깊이 정보맵 블록을 입력받아 코딩 과정에서 발생한 블록킹 현상(blocking artifact)을 제거한 후, "재구성된 영상 버퍼부"에 저장하거나 필터링을 수행한 재구성된 깊이 정보맵 블록을 출력한다.

[0245] 이상의 설명은 본 발명의 일 실시예에 불과할 뿐, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 본 발명의 본질적 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현할 수 있을 것이다. 따라서, 본 발명의 범위는 전술한 실시예에 한정되지 않고 특허 청구범위에 기재된 내용과 동등한 범위 내에 있는 다양한 실시 형태가 포함되도록 해석되어야 할 것이다.

## *도면의 간단한 설명*

[0246] 도 1은 실제 영상(왼쪽)과 깊이 정보맵영상(오른쪽)의 일 예

[0247] 도 2는 H.264/AVC의 인코더 구조도

[0248] 도 3은 H.264/AVC의 디코더 구조도

[0249] 도 4는 도 1의 실제 영상과 깊이 정보맵의 각 픽셀의 level을 표현한 3차원 그래프; (왼쪽) 실제 영상의 그래프, (오른쪽) 깊이 정보맵의 그래프

[0250] 도 5는 비트플래인 분석: 도 1의 실제 영상의 비트플래인 표현(왼쪽), 깊이 정보맵의 비트플래인(가운데), 깊이 정보맵의 비트플래인에 그레이 코드를 적용(오른쪽)

[0251] 도 6은 Ballet 영상의 깊이 정보맵에서 객체 경계 부분 매크로블록의 비트플래인 분석

[0252] 도 7은 Ballet 영상의 깊이 정보맵에서 배경 부분 매크로블록의 비트플래인 분석

[0253] 도 8은 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 인코더 구조도의 일 예

[0254] 도 9는 "비트플래인 코딩"을 수행하는 방법의 일 예

[0255] 도 10은 인트라 픽쳐 모드 결정 방법

[0256] 도 11은 인터 픽쳐 모드 결정 방법

[0257] 도 12는 움직임 예측값(MVP) 결정 방법 순서도

[0258] 도 13은 "CAE 인코딩" 구조도의 일 예

[0259] 도 14는 변환비율(CR; Conversion Ratio) 결정 순서도

[0260] 도 15는 intraCAE를 위한 컨텍스트 템플릿

[0261] 도 16은 interCAE를 위한 컨텍스트 템플릿

[0262] 도 17은 스캔 순서의 일 예

[0263] 도 18은 런 길이 코딩(Run Length Coding)과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인 코딩 방법의 일 예

[0264] 도 19는 각 심볼(Symbol)에 대한 런(Run)의 길이들을 세는 방법

[0265] 도 20은 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인 코딩 방법에서 비트플래인 블록을 서브 블록으로 나누어서 코딩하는 방법의 일 예

[0266] 도 21은 컨텍스트 기반 적응적 산술 코딩 방법을 응용한 비트플래인 코딩 방법의 일 예

[0267] 도 22는 "비트플래인 디코딩"을 수행하는 방법의 일 예

[0268] 도 23은 "CAE 디코딩" 구조도의 일 예

[0269] 도 24는 런 길이 코딩(Run Length Coding)과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인 디코딩 방법의 일 예

[0270] 도 25는 런 길이 코딩(Run Length Coding) 방법과 가변 길이 코딩(Variable Length Coding) 방법을 이용한 비트플래인 코딩 방법에서 비트플래인 블록을 서브 블록으로 나누어서 디코딩하는 방법의 일 예

[0271] 도 26은 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 디코더 구조도의 일 예

[0272] 도 27은 코딩 모드 결정 방법의 일 예

[0273] 도 28은 깊이 정보맵 블록 혹은 분리된 비트플래인의 이진 블록에 다운샘플링을 적용하여 비트율을 조절하는 방법의 일 예

[0274] 도 29는 다운샘플링하는 방법의 일 예

[0275] 도 30은 최적의 샘플링 레이트(sampling rate)를 결정하는 방법의 순서도의 실시 일 예

[0276] 도 31은 코딩할 비트플래인의 수를 결정하는 방법의 순서도의 실시 일 예

[0277] 도 32는 "bitplane_coding_flag" 값에 따른 코딩 모드 수행 방법의 실시 일 예

[0278] 도 33은 "bitplane_num" 값에 따른 비트플래인 코딩 수행 방법의 순서도의 실시 일 예

[0279] 도 34는 Ballet 영상의 깊이 정보맵 실험 결과 그래프 (Anchor: 기존의 방법, Proposed: 제안하는 방법)

[0280] 도 35는 Breakdanceres 영상의 깊이 정보맵 실험 결과 그래프 (Anchor: 기존의 방법, Proposed: 제안하는 방법)

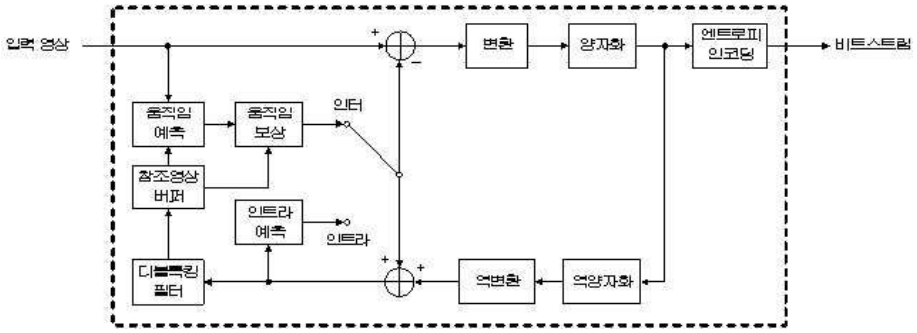
[0281] 도 36은 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 인코더 장치도

[0282] 도 37은 블록 단위 비트플래인 코딩 방법과 기존의 블록 단위 동영상 코딩 방법을 적응적으로 선택하여 코딩하는 방법과, 깊이 정보맵을 비트플래인 단위로 분리하여 코딩할 비트플래인을 적응적으로 선택하는 방법의 디코더 장치도.

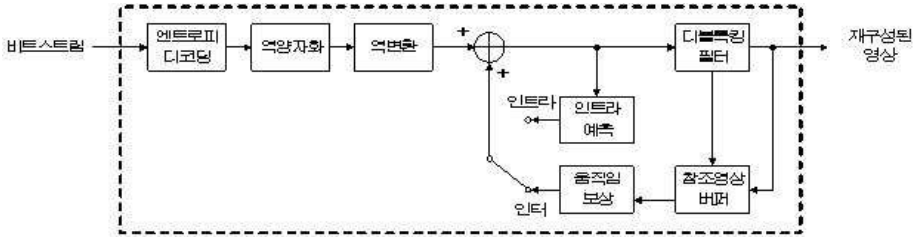
## 도면

### 도면1

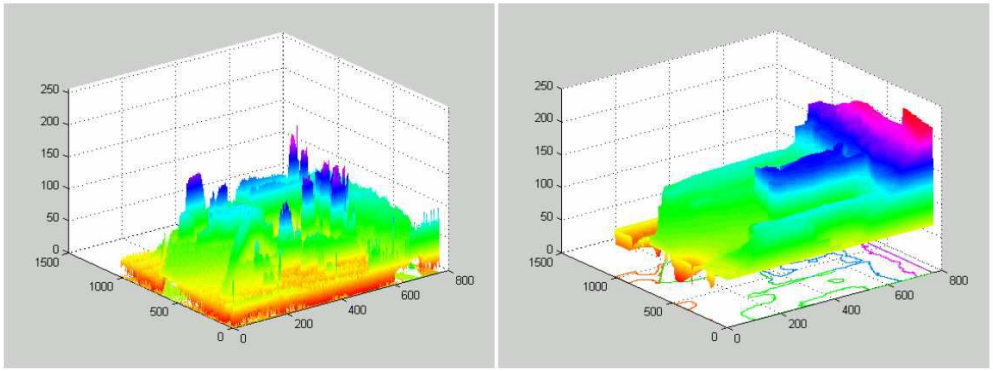
### 도면2



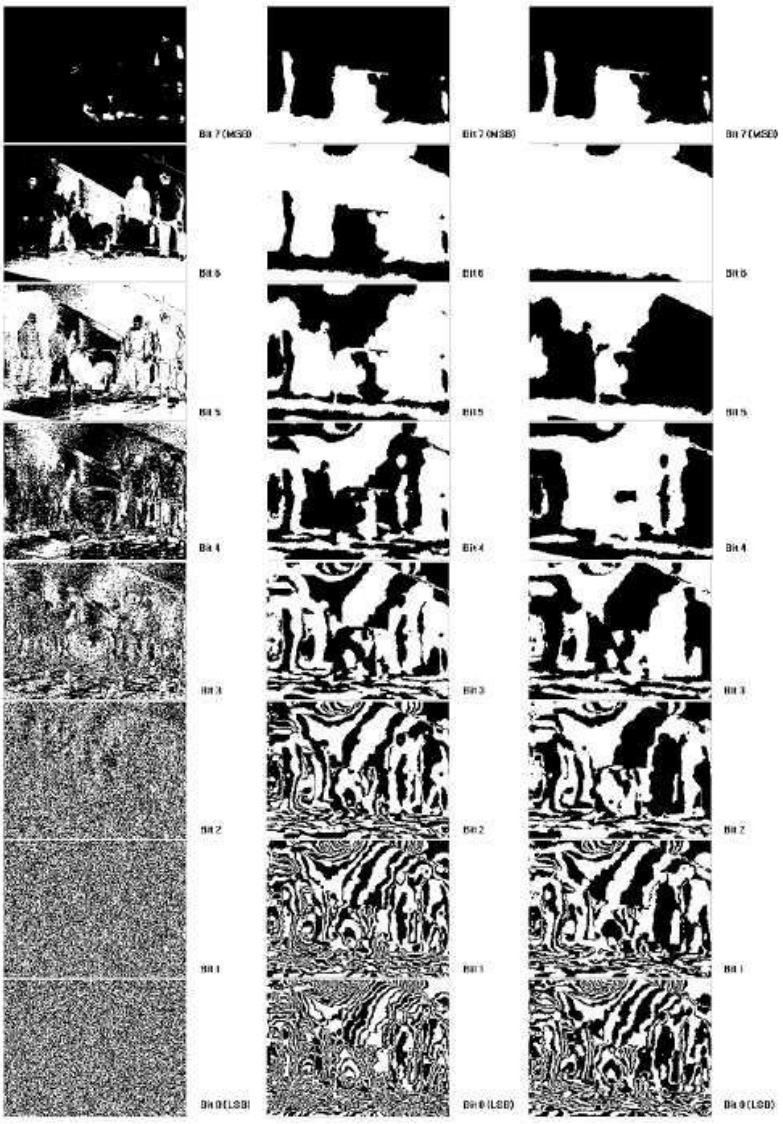
도면3


비트스트림
엔트로피 디코딩
역양자화
역변환
디블록킹 필터
재구성된 영상
인트라
인트라 예측
움직임 보상
인터
참조영상 버퍼

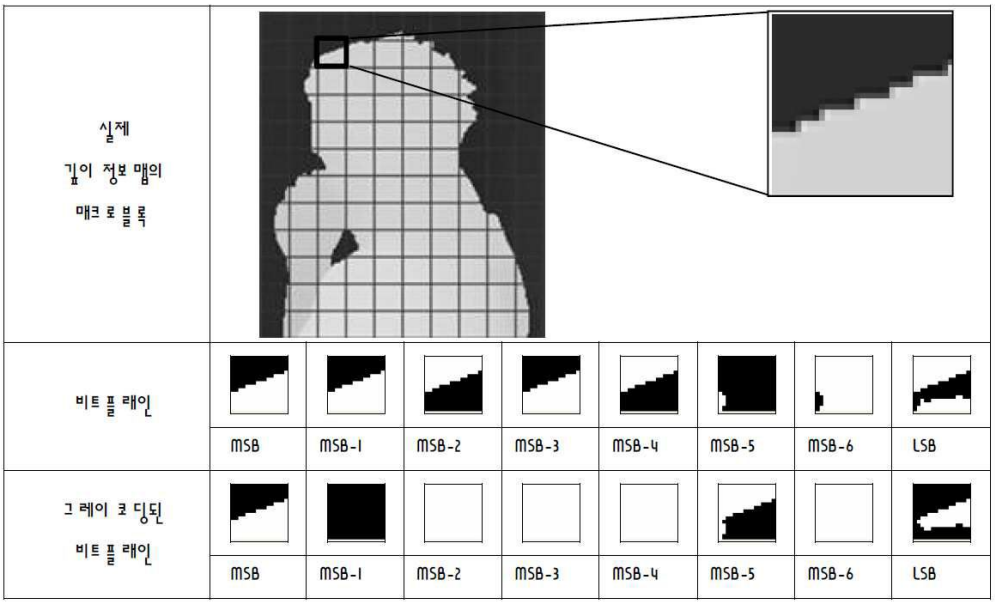
도면4


250
200
150
100
50
0
1500
1000
500
0
0
200
400
600
800

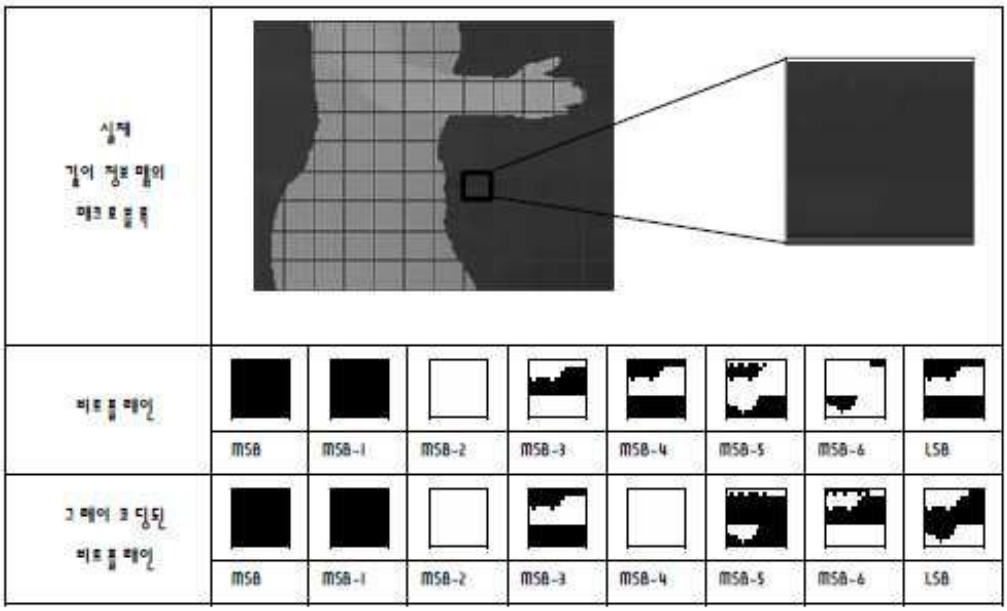
도면5



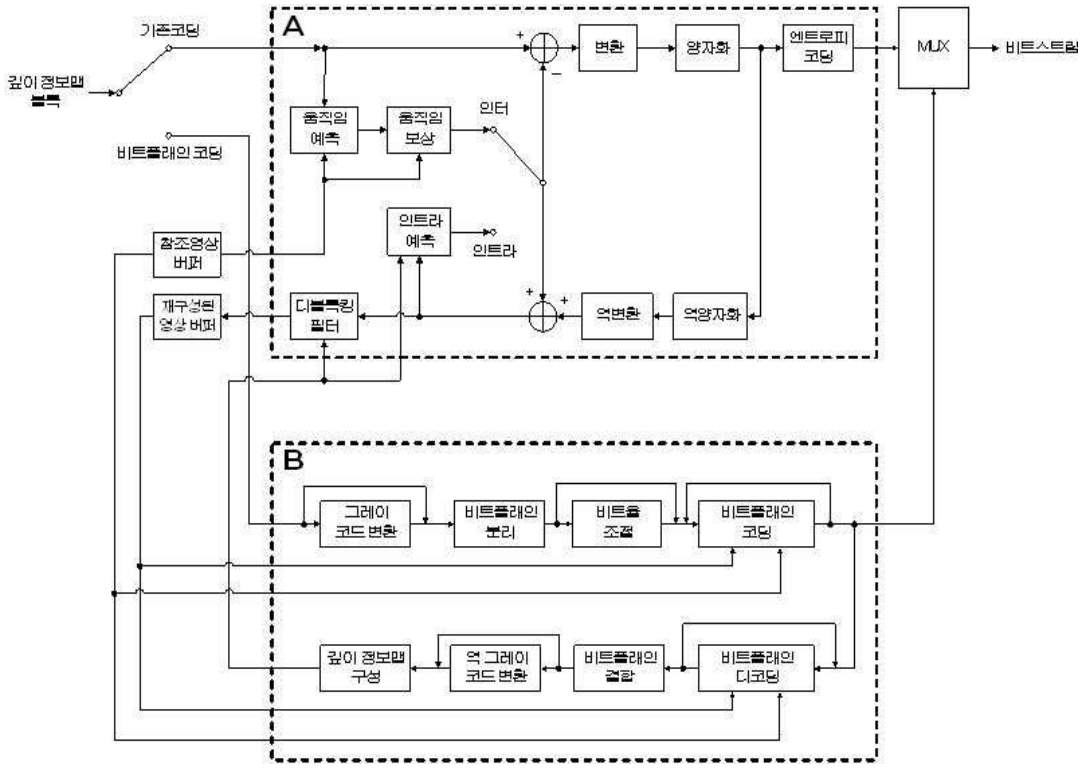
도면6


실제
깊이 정보 맵의
매크로블록
비트플레인
MSB
MSB-1
MSB-2
MSB-3
MSB-4
MSB-5
MSB-6
LSB
그레이 코딩된
비트플레인
MSB
MSB-1
MSB-2
MSB-3
MSB-4
MSB-5
MSB-6
LSB

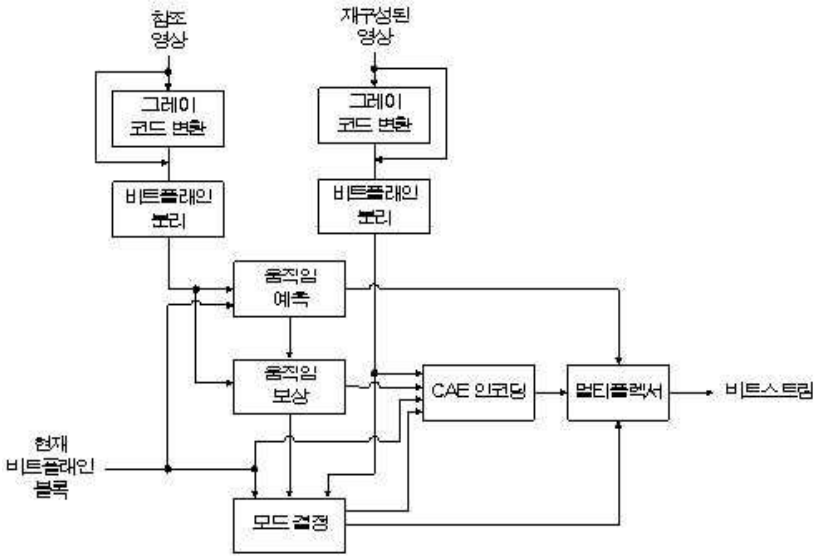
도면7


실제
깊이 정보 맵의
매크로블록
비트플레인
MSB
MSB-1
MSB-2
MSB-3
MSB-4
MSB-5
MSB-6
LSB
그레이 코딩된
비트플레인
MSB
MSB-1
MSB-2
MSB-3
MSB-4
MSB-5
MSB-6
LSB

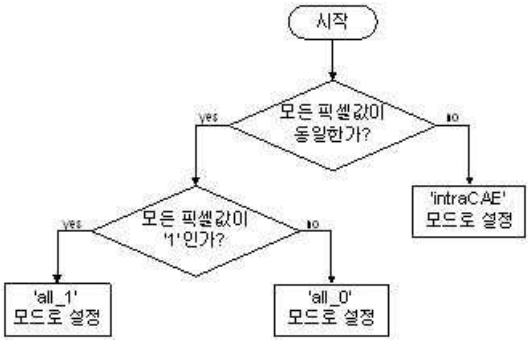
도면8


기존코딩
깊이 정보맵 블록
비트플레인 코딩
A
변환
양자화
엔트로피 코딩
MUX
비트스트림
움직임 예측
움직임 보상
인터
인트라 예측
인트라
참조영상 버퍼
재구성된 영상 버퍼
디블록킹 필터
역변환
역양자화
B
그레이 코드 변환
비트플레인 분리
비트율 조절
비트플레인 코딩
깊이 정보맵 구성
역 그레이 코드 변환
비트플레인 결합
비트플레인 디코딩

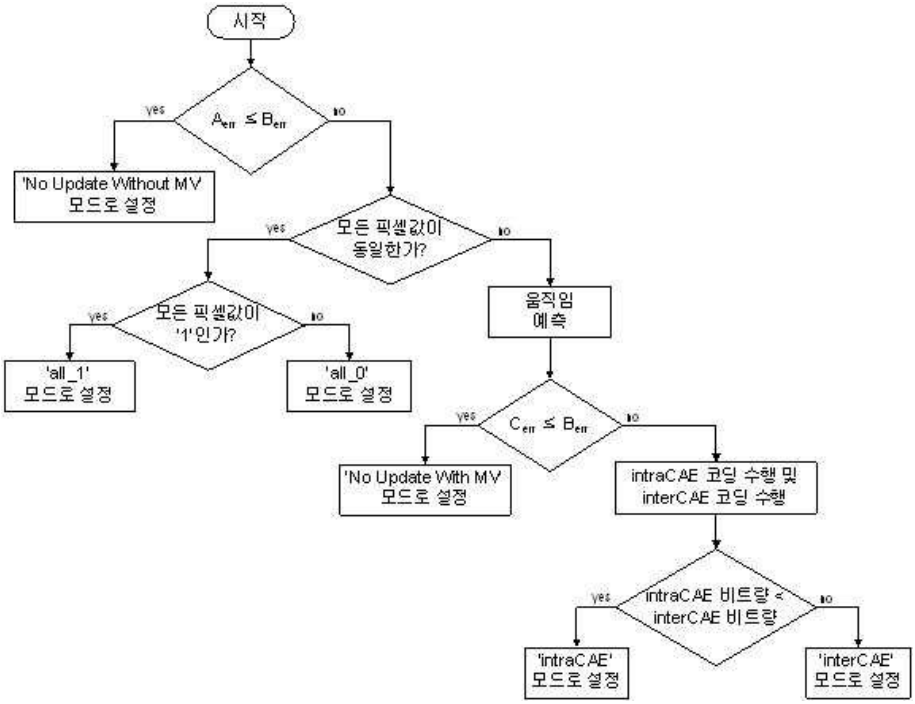
도면9


참조 영상
재구성된 영상
그레이 코드 변환
그레이 코드 변환
비트플레인 분리
비트플레인 분리
움직임 예측
움직임 보상
CAE 인코딩
멀티플렉서
비트스트림
현재 비트플레인 블록
모드 결정

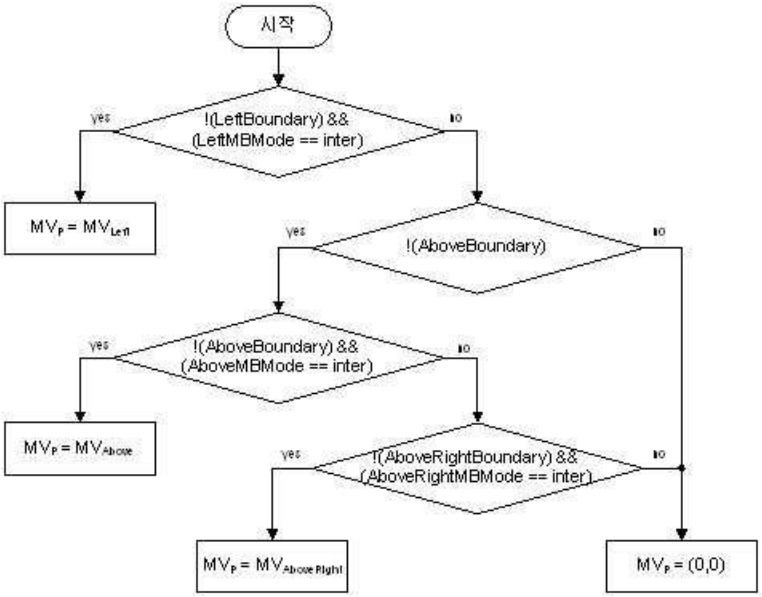
도면10


시작
모든 픽셀값이 동일한가?
yes
no
'intraCAE' 모드로 설정
모든 픽셀값이 '1'인가?
yes
no
'all_1' 모드로 설정
'all_0' 모드로 설정

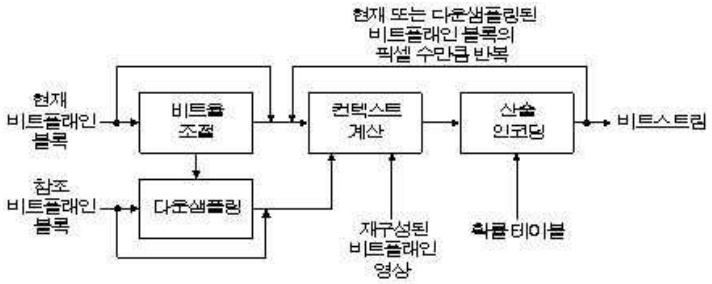
도면11


시작
$A_{err} \leq B_{err}$
yes
no
'No Update Without MV' 모드로 설정
모든 픽셀값이 동일한가?
yes
no
모든 픽셀값이 '1'인가?
yes
no
'all_1' 모드로 설정
'all_0' 모드로 설정
움직임 예측
$C_{err} \leq B_{err}$
yes
no
'No Update With MV' 모드로 설정
intraCAE 코딩 수행 및 interCAE 코딩 수행
intraCAE 비트량 < interCAE 비트량
yes
no
'intraCAE' 모드로 설정
'interCAE' 모드로 설정

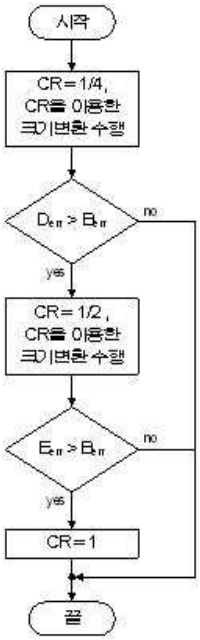
도면12


시작
!(LeftBoundary) &&
(LeftMBMode == inter)
yes
no
MVP = MVLeft
!(AboveBoundary)
yes
no
!(AboveBoundary) &&
(AboveMBMode == inter)
yes
no
MVP = MVAbove
!(AboveRightBoundary) &&
(AboveRightMBMode == inter)
yes
no
MVP = MVAbove Right
MVP = (0,0)

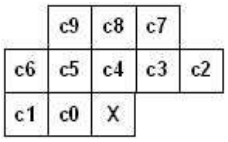
도면13


현재 또는 다운샘플링된
비트플레인 블록의
픽셀 수만큼 반복
현재
비트플레인
블록
비트율
조절
컨텍스트
계산
산술
인코딩
비트스트림
참조
비트플레인
블록
다운샘플링
재구성된
비트플레인
영상
확률테이블

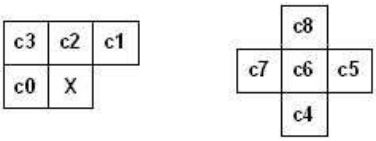
도면14



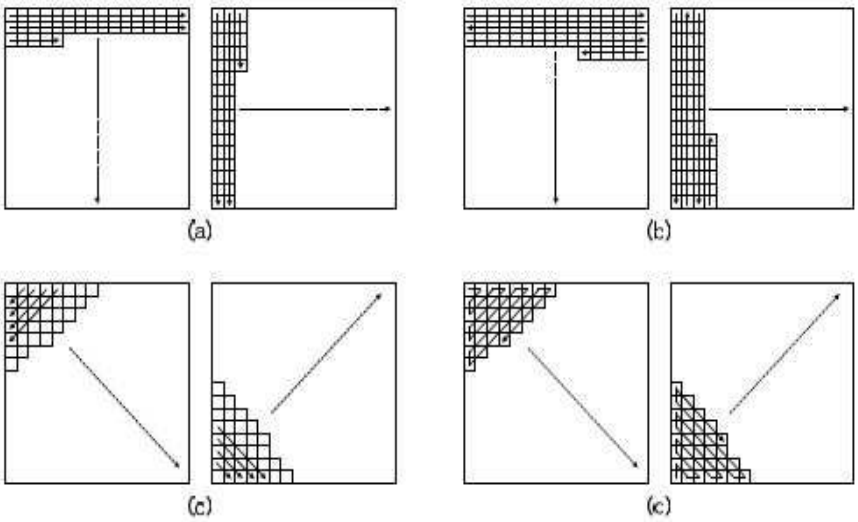

도면15



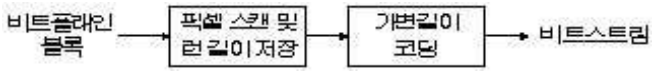

도면16



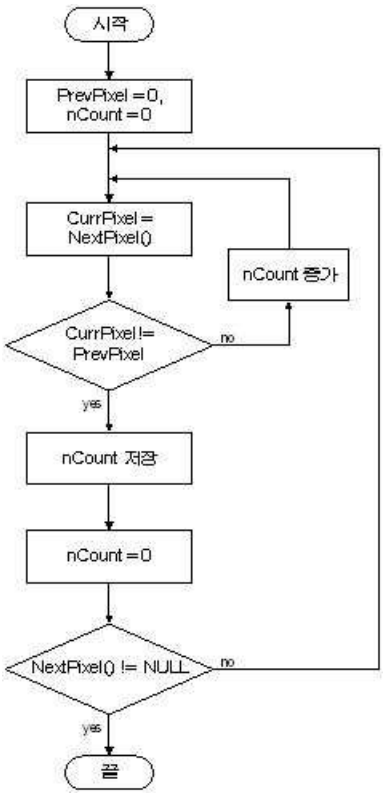
도면17



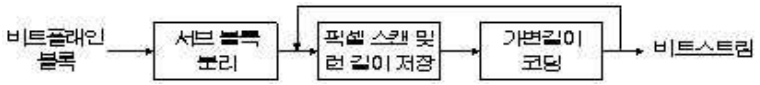

도면18



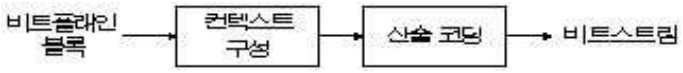

도면19



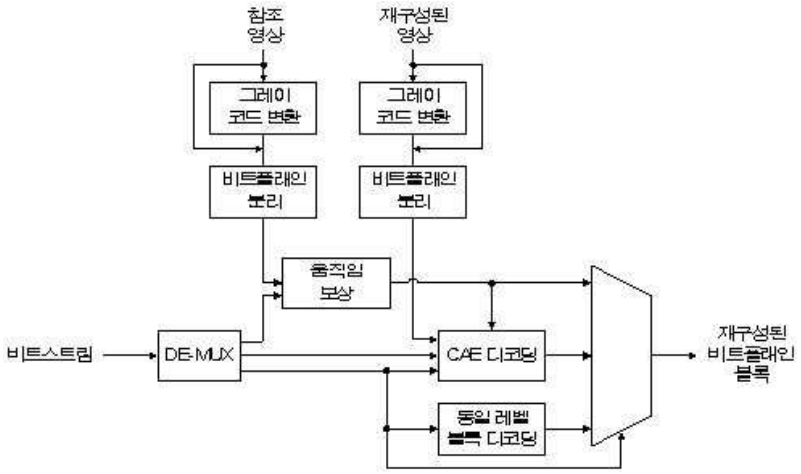
도면20



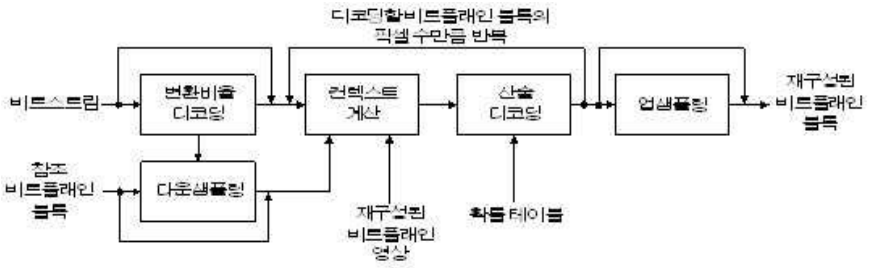

도면21



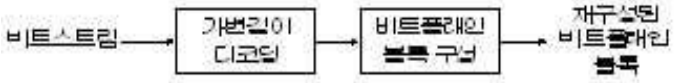

도면22



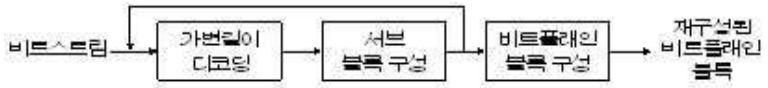

도면23



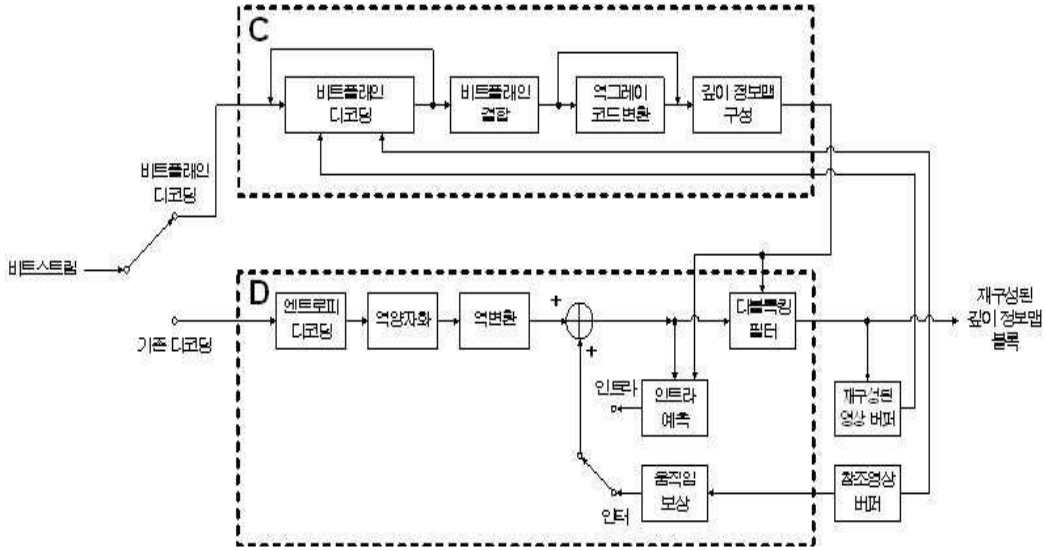

도면24



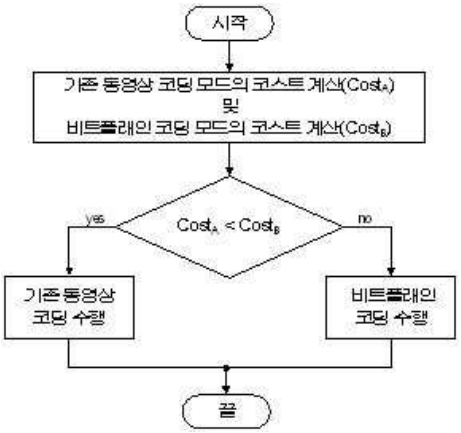
도면25




도면26

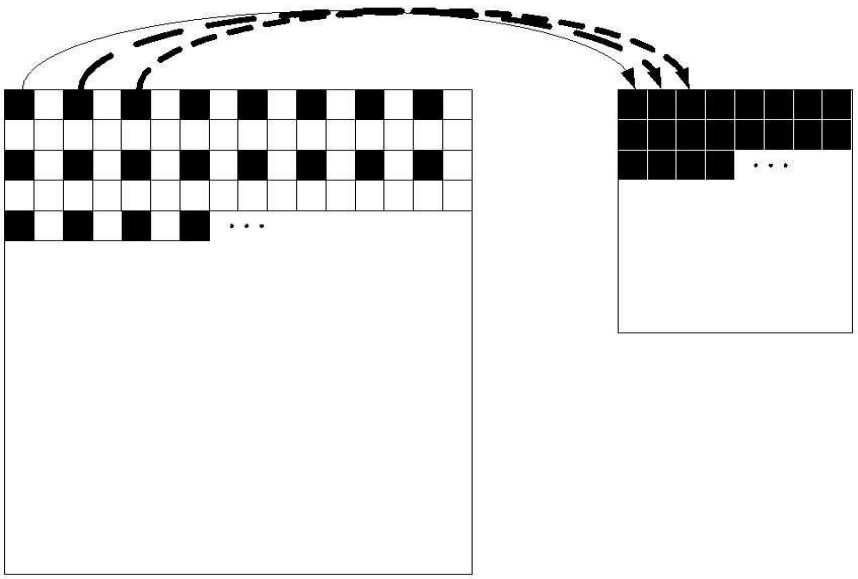



도면27

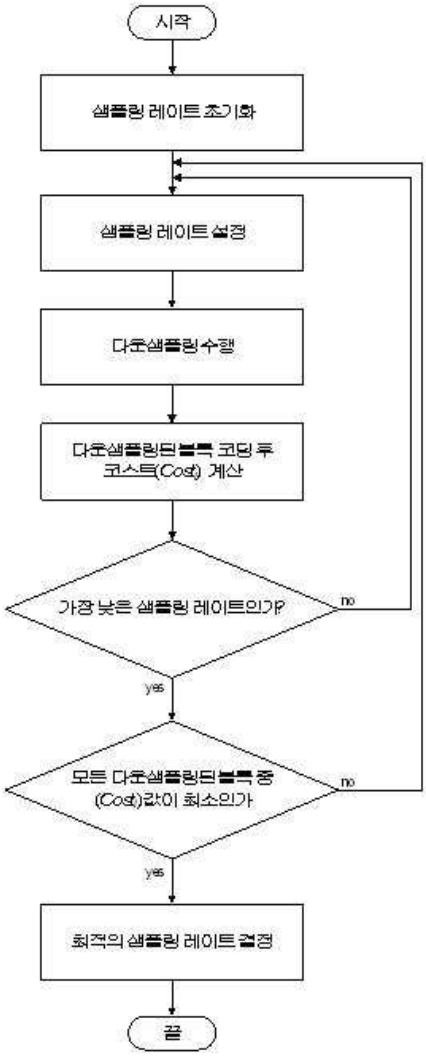



도면28

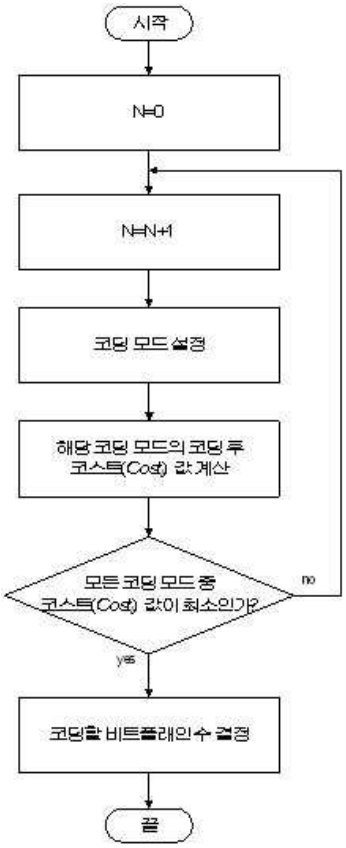
도면29



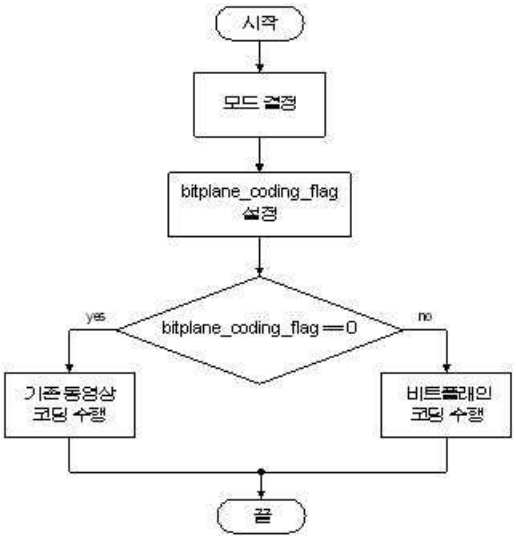
도면30


시작
샘플링 레이트 초기화
샘플링 레이트 설정
다운샘플링 수행
다운샘플링된 블록 코딩 후
코스트(Cost) 계산
가장 낮은 샘플링 레이트인가?
no
yes
모든 다운샘플링된 블록 중
(Cost)값이 최소인가
no
yes
최적의 샘플링 레이트 결정
끝

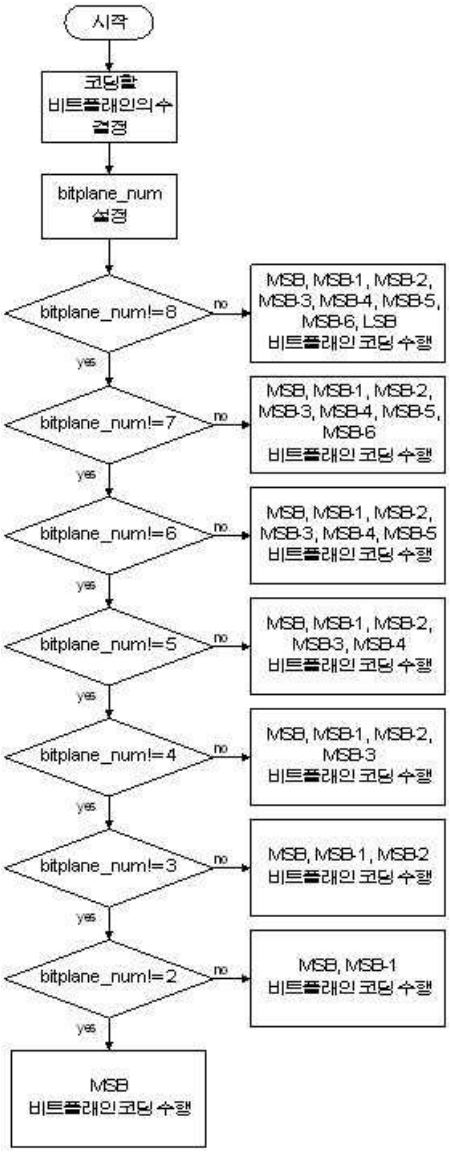
도면31


시작
N=0
N=N+1
코딩 모드 설정
해당 코딩 모드의 코딩 후
코스트(Cost) 값 계산
모든 코딩 모드 중
코스트(Cost) 값이 최소인가?
no
yes
코딩할 비트플레인수 결정
끝

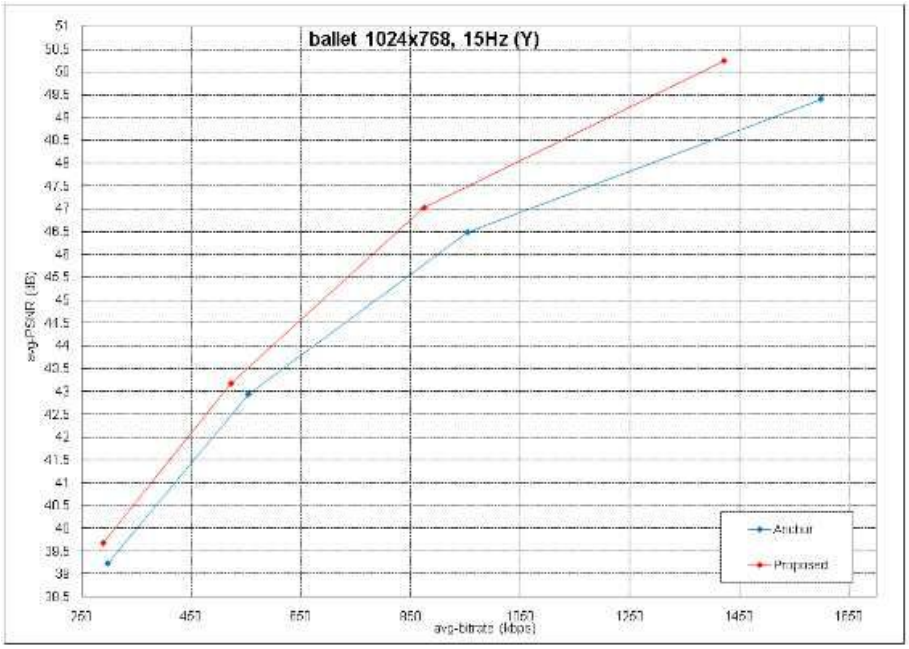
도면32


시작
모드 결정
bitplane_coding_flag
설정
bitplane_coding_flag == 0
yes
no
기존 동영상
코딩 수행
비트플레인
코딩 수행
끝

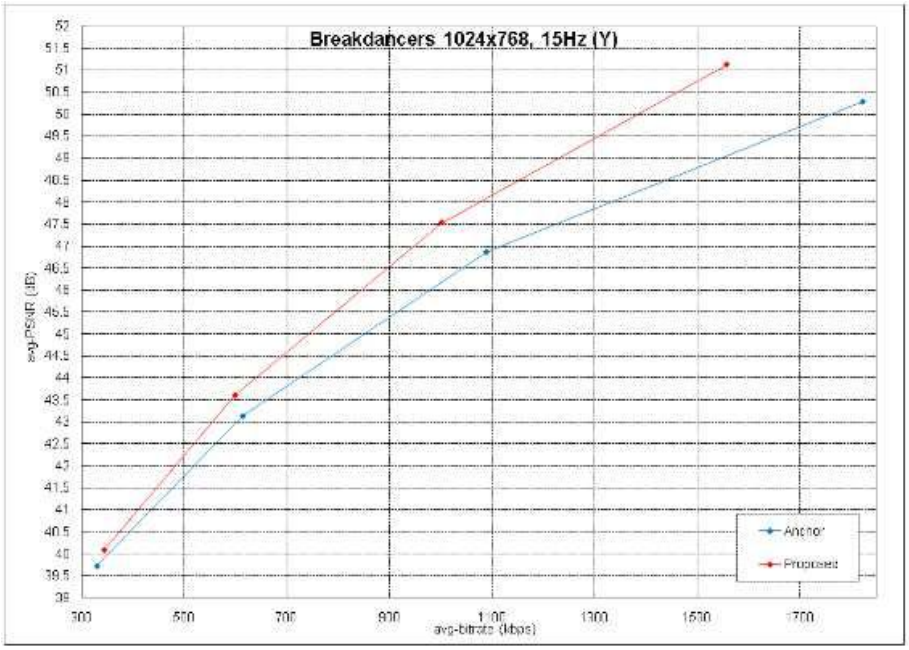
도면33


시작
코딩할 비트플래인의수 설정
bitplane_num 설정
bitplane_num!=8
no
MSB, MSB-1, MSB-2, MSB-3, MSB-4, MSB-5, MSB-6, LSB 비트플래인 코딩 수행
yes
bitplane_num!=7
no
MSB, MSB-1, MSB-2, MSB-3, MSB-4, MSB-5, MSB-6 비트플래인 코딩 수행
yes
bitplane_num!=6
no
MSB, MSB-1, MSB-2, MSB-3, MSB-4, MSB-5 비트플래인 코딩 수행
yes
bitplane_num!=5
no
MSB, MSB-1, MSB-2, MSB-3, MSB-4 비트플래인 코딩 수행
yes
bitplane_num!=4
no
MSB, MSB-1, MSB-2, MSB-3 비트플래인 코딩 수행
yes
bitplane_num!=3
no
MSB, MSB-1, MSB-2 비트플래인 코딩 수행
yes
bitplane_num!=2
no
MSB, MSB-1 비트플래인 코딩 수행
yes
MSB 비트플래인코딩 수행

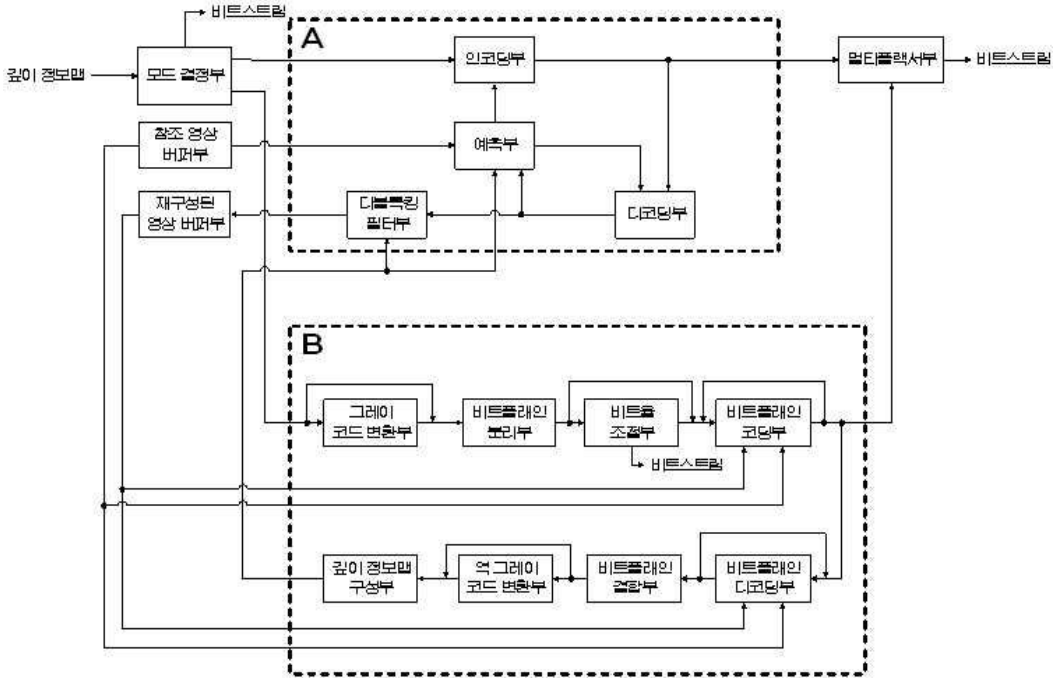
도면34


ballet 1024x768, 15Hz (Y)
avg-PSNR (dB)
avg-bitrate (kbps)
Anchor
Proposed

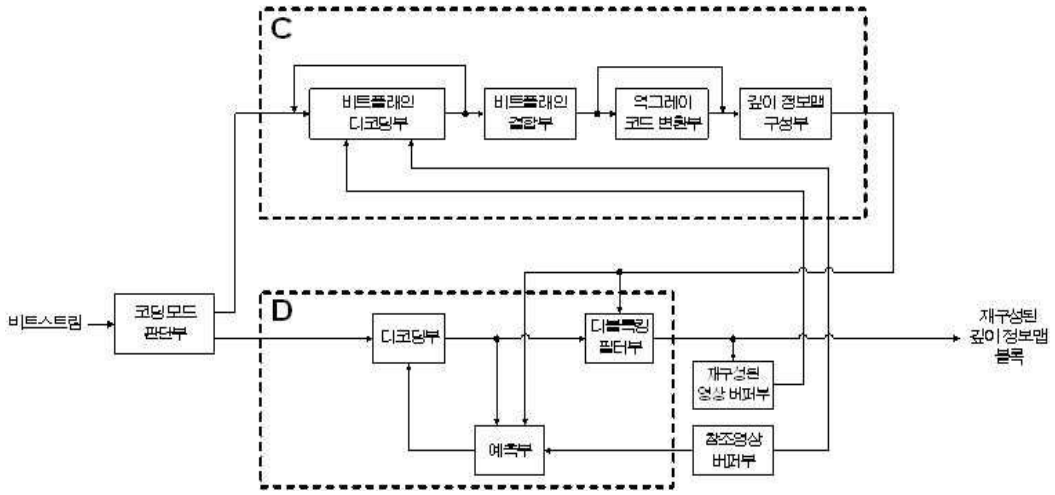
도면35


Breakdancers 1024x768, 15Hz (Y)
avg-PSNR (dB)
avg-bitrate (kbps)
Anchor
Proposed

도면36


비트스트림
깊이 정보맵
모드 결정부
A
인코딩부
멀티플렉서부
비트스트림
참조 영상 버퍼부
예측부
재구성된 영상 버퍼부
디블록킹 필터부
디코딩부
B
그레이 코드 변환부
비트플레인 분리부
비트율 조절부
비트플레인 코딩부
비트스트림
깊이 정보맵 구성부
역그레이 코드 변환부
비트플레인 결합부
비트플레인 디코딩부

도면37


C
비트플레인 디코딩부
비트플레인 결합부
역그레이 코드 변환부
깊이 정보맵 구성부
비트스트림
코딩모드 판단부
D
디코딩부
디블록킹 필터부
재구성된 깊이정보맵 블록
재구성된 영상 버퍼부
예측부
참조영상 버퍼부