(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0083980
(43) 공개일자 2010년07월23일

(51) Int. Cl.
H04N 7/24 (2006.01) *H04N 13/00* (2006.01)
(21) 출원번호 10-2009-0003348
(22) 출원일자 2009년01월15일
심사청구일자 없음

(71) 출원인
삼성전자주식회사
경기도 수원시 영통구 매탄동 416
경희대학교 산학협력단
경기도 용인시 기흥구 서천동 1 경희대학교 국제캠퍼스내
(72) 발명자
박태성
경기도 수원시 영통구 영통동 신안아파트 533동 507호
김종호
경기도 수원시 영통구 매탄동 1237-1 (18/3) -304
(뒷면에 계속)
(74) 대리인
이건주

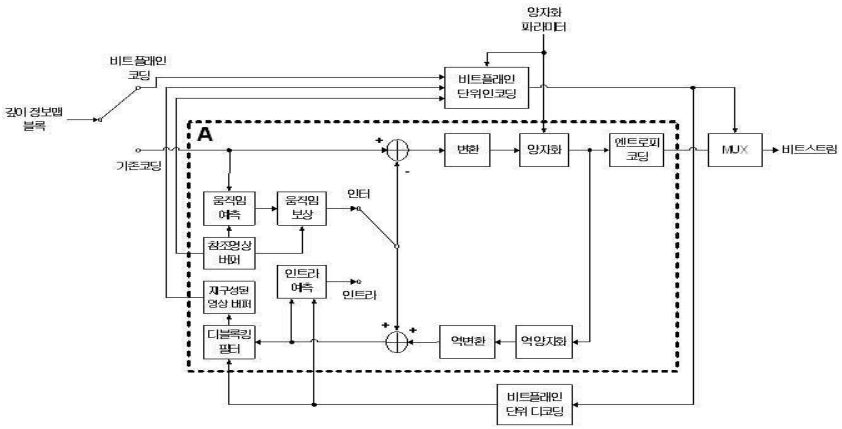
전체 청구항 수 : 총 2 항

(54) 적응적 블록기반 깊이 정보맵 코딩 방법 및 장치

(57) 요 약

본 발명은 깊이 정보맵 코딩시, 깊이 정보맵을 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법을 포함하며, 깊이 정보맵 코딩시, 깊이 정보맵을 비트플래인 단위로 분리하여 각각의 비트플래인을 코딩하는 방법을 기존의 동영상 코딩 알고리즘과 함께 적응적으로 선택하여 사용하는 방법을 포함한다.

대 표 도 - 도4



(72) 발명자

오윤제

경기도 수원시 영통구 영통동 두산아파트 805동 106호

서덕영

경기도 성남시 분당구 수내동 푸른마을벽산아파트 201동 202호

박광훈

경기도 성남시 분당구 분당동 45번지 동아빌라 B동 302호

김규헌

서울특별시 강남구 청담1동 현대청담3차아파트 103동 1504호

박민우

경기도 수원시 권선구 세류2동 1167-26

김경용

전라남도 영광군 영광읍 남천리 349-2

특허청구의 범위

청구항 1

깊이 정보맵 코딩시, 깊이 정보맵을 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법 및 장치.

청구항 2

깊이 정보맵 코딩시, 깊이 정보맵을 비트플래인 단위로 분리하여 각각의 비트플래인을 코딩하는 방법을 기존의 동영상 코딩 알고리즘과 함께 적응적으로 선택하여 사용하는 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명은 깊이 정보맵(depth map) 코딩 시, 깊이 정보의 화질 및 코딩 효율을 향상시키기 위한 깊이 정보맵 코딩 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 Amendment 4) 표준을 완료하였고, 또한 현재 MPEG에서는 3D Video에 대한 표준이 진행중이다. 3D Video는 N개의 시점 영상과 N개 이하의 깊이 정보맵(depth map)을 함께 전송하여 전송받은 시점 영상과 깊이 정보맵을 이용하여 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 수의 뷰를 전달하는데는 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 영상(들)을 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) 코딩 방법만 정의하고 있을 뿐 아직 깊이 정보맵 자체의 코딩 방법은 정의하지 않고 있다. 따라서 깊이 정보맵 코딩에 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 비디

오 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 대응하는 화면에 보이는 깊이 정보를 8비트로 표현한 것으로 카메라와 가까울수록 큰 값으로 표현된 것이다.

[0007] 도 1의 깊이 정보맵을 이용하여 각 픽셀에서 실 세계에서의 거리(Z)를 구하는 방법의 일 예는 아래 수식과 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 코딩은 H.264나 다시점 비디오 코딩과 같은 기존의 표준 코덱을 그대로 사용하는 것이 일반적이다. 이러한 동영상 표준 코덱을 사용함으로 높은 압축률을 얻을 수 있지만, 깊이 정보맵의 특성을 최대한 사용하지는 못하는 단점이 있다. 보통 깊이 정보맵을 사용하여 영상을 합성하는 장치에서는 깊이 정보맵의 정확도에 따라 화질에 큰 영향을 미치기 때문에, 깊이 정보맵 코딩에 있어서 최적의 효율을 얻는 방법이 필요하다.

발명의 내용

해결 하고자하는 과제

[0011] 본 발명은 깊이 정보맵의 화질 및 코딩 효율을 향상 시키는 깊이 정보맵 코딩 방법 및 장치를 제공하고자 한다.

과제 해결수단

[0012] 이를 달성하기 위한 본 발명의 일 형태에 따르면, 본 발명은 깊이 정보맵 코딩 시, 깊이 정보맵을 비트플래인 단위로 분리하여 각각의 비트플래인을 코딩하는 방법을 기존의 동영상 코딩 알고리즘과 함께 적응적으로 선택하여 사용하는 방법 및 장치를 이용하여 깊이 정보맵의 화질 및 코딩 효율을 향상시키는 방법을 포함함을 특징으로 한다.

효 과

[0013] 본 발명은 깊이 정보맵 코딩시, 깊이 정보맵을 비트플래인 단위로 분리하여 각각의 비트플래인을 코딩하는 방법을 기존의 동영상 코딩 알고리즘과 함께 적응적으로 선택하여 사용하는 방법을 이용하여 화질 및 코딩효율을 향상시키는 효과가 있다.

발명의 실시를 위한 구체적인 내용

[0014] 이하 첨부된 도면을 참조하여 본 발명을 구성하는 장치 및 동작 방법을 본 발명의 실시 예를 참조하여 상세히 설명한다. 하기 설명에서는 구체적인 구성 소자 등과 같은 특정 사항들이 나타나고 있는데 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐 이러한 특정 사항들이 본 발명의 범위 내에서 소정의 변형이나 혹은 변경이 이루어질 수 있음은 이 기술분야에서 통상의 지식을 가진 자에게는 자명하다 할 것이다. 또한, 본 발명을 설명함에 있어서 본 발명과 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에 그 상세한 설명을 생략하기로 한다.

[0015] 깊이 정보맵은 일반적인 실사 영상과는 다르게 상당히 완만한 특성을 갖는데, 도 2를 통해서 쉽게 그 특성을 알 수 있다. 도 2는 도 1의 실제 영상과 깊이 정보맵의 각 픽셀의 레벨(실제 영상에서는 휘도(luminance) 성분의 레벨 즉 밝기를 뜻하고, 깊이 정보맵에서는 깊이의 레벨의 뜻한다)을 표현한 3차원 그래프인데, 실제 영상의 그래프(도 2 왼쪽 그래프)에서는 픽셀 사이에 변화가 심한 형태를 보여주고 있음을 확인할 수 있고, 이에 비해 깊

이 정보맵의 그래프(도 2 오른쪽 그래프)는 상당히 완만한 형태를 보여주고 있음을 확인할 수 있다.

[0016] 그리고 또한 깊이 정보맵을 비트플래인(bit-plane) 단위로 표현할 수 있는데, 깊이 정보맵의 비트플래인은 실사 영상의 비트플래인에 비해 상당히 단조로운 형태를 보여준다. 도 3은 도 1의 실제 영상과 깊이 정보맵을 비트플래인 단위로 표현한 모습을 보여준다. 실제 영상의 비트플래인(도 3 왼쪽)은 최상위 비트플래인(MSBP; most significant bit-plane)은 단조로운 형태 정보를 가지지만 하위 비트플래인으로 갈수록 상당히 복잡해 지며, 최하위 비트플래인(LSBP; least significant bit-plane)부터 세 개의 비트플래인은 거의 white noise 형태로 상당히 복잡함을 알 수 있다. 하지만 깊이 정보맵의 비트플래인(도 3 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 gray code를 적용하면 각 비트플래인 단위에서 중복도가 높아지는데 깊이 정보맵의 각 레벨 값에 gray code를 적용한 비트플래인의 모습은 도 3 오른쪽과 같고, 더욱 단조로운 형태를 보이는 것을 직접 확인할 수 있다.

[0017] 이러한 사실을 통해서 깊이 정보맵을 비트플래인으로 표현하였을 때 갖는 높은 중복성을 이용하기 위한 비트플래인 단위 코딩 방법을 적절하게 코딩할 수 있는 방법이 필요하다.

[0018] 본 발명에서는 깊이 정보맵의 코딩 시에 깊이 정보맵의 특성에 맞게 코딩하는 방법으로 깊이 정보맵을 비트플래인 단위로 분리하여 각각의 비트플래인을 코딩하는 방법을 기존의 동영상 코딩 알고리즘과 함께 적응적으로 선택하여 사용하는 방법을 제안한다.

[0019] <<기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법>>

[0020] <인코더>

[0021] 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 인코더 구조도의 일예는 도 4와 같다.

[0022] 도 4의 인코더 구조에서는 MxN 블록 단위로 코딩이 되는 것으로써, MxN 블록 단위로 입력된 깊이 정보맵 블록은 비트플래인 코딩 모드 혹은 기존 코딩 모드 중에 선택적으로 코딩이 수행이 된다. 비트플래인 코딩 모드가 선택이 되었을 경우에는 “비트플래인 단위 인코딩” 이 수행이 되며, n-bit로 표현되는 깊이 정보맵 블록과 참조 영상과 양자화 파라미터를 입력받아 실제 코딩을 수행한다. 기존 코딩 모드가 선택이 되었을 경우에는 도 4에서 'A'로 표시되어 있는 기존의 동영상 코딩 방법(MPEG-1, MPEG-2, MPEG-4 Part 2 Visual, H.264/AVC, SVC, MVC, VC-1, AVS, KTA, 등)을 수행한다.

[0023] 도 4에서는 기존 코딩 모드의 'A' 일 예로 H.264/AVC의 인코더를 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 블록에 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 블록을 생성한 예측 블록과 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 결과를 출력한다. 수행된 결과는 “MUX”에 입력되어 비트스트림으로 출력이 된다. 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, "재구성된 영상 버퍼"에 저장한다.

[0024] 도 4에서 비트플래인 코딩이 선택되었을 경우, "비트플래인 단위 인코딩" 과정에서의 인코더 구조도의 일 예는 도 5와 같다.

[0025] 도 5의 "비트율 조절" 은 비트율 조절 기능이 필요할 경우에 선택적으로 사용할 수있는 과정으로써, 실제 코딩시에 원하는 비트율을 얻기 위해 코딩할 비트플래인의 수를 결정한다.

[0026] 코딩할 비트플래인의 수를 결정하는 일 예로 현재 입력되는 양자화 파라미터(Quantization Parameter; QP)에 따라 실제 코딩을 수행할 비트플래인을 MSB 비트플래인부터 LSB 비트플래인의 순서대로 선택하는 방법이 있을 수 있다. 실시 일 예로 QP가 32일 때에는 LSB 와 LSB+1 비트플래인을 제외한 모든 비트플래인을 코딩하도록 설정할 수 있다.

[0027] 코딩할 비트플래인의 수를 결정하는 또다른 일 예로 헤더 정보 혹은 매크로블록 데이터 정보에 코딩을 수행할 비트플래인의 수를 결정하는 신택스 엘리먼트(syntax element)를 코딩하여 전송할 수 있다. 헤더 정보에 코딩할 비트플래인의 수를 코딩하는 구현의 실시 일 예로 동영상 표준인 H.264/AVC의 Picture Parameter Set(PPS; pic_parameter_set_rbsp) 신택스에 코딩하는 방법은 다음과 같다.

표 1

[0028]

pic_parameter_set_rbsp( ) {	C	Descriptor
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
max_bitplane_num	1	u(n)
...		
}		

[0029] 'max_bitplane_num'은 코딩을 수행할 비트플래인의 수를 결정하는 신택스 앨리먼트이며, MSB 비트플래인부터 LSB 비트플래인의 순서로 결정한다.

[0030] 헤더 정보에 코딩할 비트플래인의 수를 코딩하는 구현의 실시 일 예로 동영상 표준인 H.264/AVC의 Slice Header(slice_header) 신택스에 코딩하는 방법은 다음과 같다.

표 2

[0031]

slice_header( ) {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
max_bitplane_num	1	u(n)
}		

[0032] 'max_bitplane_num'은 코딩을 수행할 비트플래인의 수를 결정하는 신택스 앨리먼트이며, MSB 비트플래인부터 LSB 비트플래인의 순서로 결정한다.

[0033] 그리고 "비트율 조절" 수행 여부에 대한 플래그 정보를 코딩할 수 있다. "비트율 조절" 수행 여부에 대한 플래그 정보하는 구현의 실시 일 예로 동영상 표준인 H.264/AVC의 Picture Parameter Set(PPS; pic_parameter_set_rbsp) 신택스에 코딩하는 방법은 다음과 같다.

표 3

[0034]

pic_parameter_set_rbsp( ) {	C	Descriptor
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
rate_control_flag	1	u(1)

...		
}		

[0035] 'rate_control_flag'가 1이면 “비트율 조절”을 수행하고, 0이면 “비트율 조절”을 수행하지 않는다. 만약 일예로 Slice Header에 코딩을 수행할 비트플래인의 수를 결정하는 신택스 앨리먼트가 존재한다면, Slice Header 신택스에 코딩하는 다음과 같다. 'rate_control_flag'가 1일 때만 코딩을 수행할 비트플래인의 수를 결정하는 신택스 앨리먼트인 'max_bitplane_num'의 코딩을 수행한다.

표 4

[0036]

slice_header( ) {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
if(rate_control_flag = = 1)		
max_bitplane_num	1	u(n)
}		

[0037] 도 5의 “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 수식과 같다.

수학식 2

[0038]
$$g_i = a_i \oplus a_{i+1} \qquad 0 \le i \le m-2$$
$$g_{m-1} = a_{m-1}$$

[0039] 여기서 $\oplus$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.

[0040] 도 5의 “비트플래인 분리” 과정에서는 n-bit로 표현되는 깊이 정보맵 블록을 입력받아서 n개의 비트플래인으로 분리한다. 분리한 n개의 비트플래인은 각각 “비트플래인 코딩”으로 반복하여 입력이 되며, “비트플래인 코딩”을 수행한 결과를 “MUX”로 입력하여 비트스트림으로 출력한다.

[0041] “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 6과 같다.

[0042] 도 6에서 현재 비트플래인 블록은 현재 코딩하려고 하는 블록에서 코딩을 수행할 비트플래인 블록을 의미한다. 참조 비트플래인 영상은 도 4의 “참조 영상 버퍼”의 참조 영상에서 현재 코딩을 수행하려고 하는 비트플래인과 동일한 비트플래인 영상이고, 재구성된 비트플래인 영상은 도 4의 “재구성된 영상 버퍼”의 현재 재구성된 깊이 정보맵 영상에서 현재 코딩을 수행하려고 하는 비트플래인과 동일한 비트플래인 영상이다.

[0043] “움직임 예측” 과정에서는 현재 비트플래인 블록과 가장 유사한 부분을 참조 비트플래인 영상에서 검색하여 가장 잘 매칭이 되는 영역과의 움직임 변위 즉 움직임 벡터를 계산하여 출력하며, “움직임 보상” 과정에서는 “움직임 예측” 과정에서 생성된 움직임 벡터를 이용하여 참조 비트플래인 영상으로부터 움직임 보상된 비트플래인 블록을 출력한다. “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 경우에는 인트라(intra) 픽쳐 모드 결정 방법과, 현재 깊이 정보맵 프레임의 코딩 모드가 시간 방향 예측을 수행할 경우에는 인터(inter)픽쳐 모드 결정 방법을 수행하며, 각각의 상세한 설명은 추후에 기술한다. “CAE(Context-based Arithmetic Encoding) 코딩” 과정은 현재 비트플래인 블록이 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 기술한다. “멀티플렉서” 에서는 현재 비트플래인 블록의 움직임 벡터와 현재 비트플래인 블록의 모드, 그리고 “CAE 코딩” 결과를 입력으로 받아 “비트스트림”을 생성한다.

[0044] 도 6의 “모드 결정” 부분에서 인트라 픽쳐 모드 결정 방법은 도 7과 같으며, 자세한 알고리즘은 다음과 같다.

[0045] Step 1) 현재 비트플래인 블록의 모든 픽셀값이 동일한 값이면 Step 2로 분기한다. 만약 그렇지 않으면 Step 3으로 분기한다.

[0046] 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.

[0047] Step 3) 현재 비트플래인 블록의 모드를 'intraCAE'로 한다. 'intraCAE' 모드의 코딩 방법은 추후에 자세히 설명한다.

[0048] 도 6의 “모드 결정” 부분에서 인터 픽쳐 모드 결정 방법은 도 8과 같으며, 자세한 알고리즘은 다음과 같다.

[0049] 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.

[0050] 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.

[0051] Step 3) 현재 비트플래인 블록의 모든 픽셀값이 동일한 값이면 Step 4로 분기한다. 만약 그렇지 않으면 Step 5로 분기한다.

[0052] 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.

[0053] Step 5) 움직임 예측을 수행한다.

[0054] 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.

[0055] 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

[0056] Step 8) 'intraCAE' 모드 코딩과 'interCAE'모드 코딩을 각각 수행한다. 'intraCAE'와 'interCAE'모드의 코딩 방법은 추후에 자세히 설명한다.

[0057] Step 9) 만약 'intraCAE' 모드 코딩 결과의 비트량이 'interCAE' 모드 코딩 결과의 비트량 보다 작다면 Step 10으로 분기한다. 만약 그렇지 않다면 Step 11로 분기한다.

[0058] Step 10) 현재 비트플래인 블록의 모드를 'intraCAE'로 한다. 'intraCAE' 모드 코딩 방법은 추후에 설명한다.

[0059] Step 11) 현재 비트플래인 블록의 모드를 'interCAE'로 한다. 'inter CAE' 모드 코딩 방법은 추후에 설명한다.

[0060] 현재 비트플래인 블록의 움직임 벡터 예측값(MV_P) 결정 방법의 일 예의 순서도는 도 9와 같으며, 자세한 알고리즘은 다음과 같다.

[0061] Step 1) 현재 비트플래인 블록의 좌측의 경계가 영상의 경계(LeftBoundary)가 아니고 현재 비트플래인 블록의 좌측 비트플래인 블록 모드(LeftMBMode)가 시간 방향 예측(inter)을 수행하였다면, Step 2로 분기한다. 만약 그렇지 않다면 Step 3으로 분기한다.

[0062] Step 2) 현재 비트플래인 블록의 움직임 벡터 예측값(MV_P)을 현재 비트플래인 블록의 좌측 비트플래인 블록의 움직임 벡터(MV_{Left})로 설정한다.

[0063] Step 3) 현재 비트플래인 블록의 상단의 경계가 영상의 경계(AboveBoundary)가 아니라면, Step 4로 분기한다. 만약 현재 비트플래인 블록의 상단의 경계가 영상의 경계(AboveBoundary)라면, Step 8로 분기한다.

[0064] Step 4) 현재 비트플래인 블록의 상단의 경계가 영상의 경계(AboveBoundary)가 아니고 현재 비트플래인 블록의 상단 비트플래인 블록 모드(AboveMBMode)가 시간 방향 예측(inter)을 수행하였다면, Step 5로 분기한다. 만약 그렇지 않다면 Step 6으로 분기한다.

[0065] Step 5) 현재 비트플래인 블록의 움직임 벡터 예측값(MV_P)을 현재 비트플래인 블록의 상단 비트플래인 블록의 움직임 벡터(MV_{Above})로 설정한다.

[0066] Step 6) 현재 비트플래인 블록의 우측의 경계 또는 상단의 경계가 영상의 경계(AboveRightBoundary)가 아니고 현재 비트플래인 블록의 우측 상단 비트플래인 블록 모드(AboveRightMBMode)가 시간 방향 예측(inter)을 수행하였다면, Step 7로 분기한다. 만약 그렇지 않다면 Step 8로 분기한다.

[0067] Step 7) 현재 비트플래인 블록의 움직임 벡터 예측값(MV_P)을 현재 비트플래인 블록의 우측상단 비트플래인 블록의 움직임 벡터($MV_{AboveRight}$)로 설정한다.

[0068] Step 8) 현재 비트플래인 블록의 움직임 벡터 예측값(MV_P)을 수평 성분 값과 수직 성분 값을 (0, 0)으로 설정한다.

[0069] 상기 “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 구조도는 도 10과 같다.

[0070] 도 10의 “비트율 조절” 과정은 현재 비트플래인 블록 모드가 'intraCAE' 모드와 'interCAE' 모드일 때 수행되며 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 계산하여 그 오차가 허용 오차 범위와 같거나 작다면, 현재 비트플래인 블록을 1/2 혹은 1/4로 다운샘플링하여 생성한 블록에 대한 코딩을 수행한다. 변환 비율을 설정하는 방법은 순서도는 도 11과 같으며, 자세한 알고리즘은 다음과 같다.

[0071] Step 1) 현재 비트플래인 블록에 대해 변환비율(CR)을 1/4로 설정하고, 현재 비트플래인 블록의 변환비율에 따라 크기변환 즉 1/4 다운샘플링과 업샘플링을 수행한 비트플래인 블록을 생성한다. Step 2로 분기한다.

[0072] Step 2) 현재 비트플래인 블록과 변환비율(CR) 1/4로 크기변환을 수행한 블록간의 오차가 허용 오차 범위보다 크다면 Step 3으로 분기한다. 만약 그렇지 않다면 알고리즘을 종료한다.

[0073] Step 3) 현재 비트플래인 블록에 대해 변환비율(CR)을 1/2로 설정하고, 현재 비트플래인 블록의 변환비율에 따

라 크기변환 즉 1/2 다운샘플링과 업샘플링을 수행한 블록을 생성한다. Step 4로 분기한다.

[0074] Step 4) 현재 비트플래인 블록과 변환비율(CR) 1/2로 크기변환을 수행한 비트플래인 블록 간의 오차가 허용 오차 범위보다 크다면 Step 5로 분기한다. 만약 그렇지 않다면 알고리즘을 종료한다.

[0075] Step 5) 현재 비트플래인 블록에 대해 변환비율(CR)을 1로 설정한다. 알고리즘을 종료한다.

[0076] 도 10의 “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.

[0077] 도 10의 “컨텍스트 계산” 과정에서는 현재 비트플래인 블록의 모드가 'intraCAE' 모드일 경우에는 현재 비트플래인 블록의 코딩할 픽셀의 주변의 픽셀값들를 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 수행하여 비트스트림을 생성한다.

[0078] 'intraCAE' 모드에서 컨텍스트 템플릿을 구성할 때 사용하는 픽셀들은 도 12와 같고 현재 픽셀(X)를 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 13과 같고 현재 픽셀(X)를 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.

[0079] 도 10에서 “컨텍스트 계산”과 “산술 인코딩”은 현재 비트플래인 블록 또는 다운샘플링된 비트플래인 블록의 픽셀 수만큼 반복된다. 현재 비트플래인 블록의 픽셀은 가로 방향 또는 세로 방향으로 스캔을 적응적으로 수행할 수 있으며, 두 가지 스캔 모드를 모두 수행한 후, 비트량이 적은 스캔 모드를 선택하고, 스캔 모드 정보는 비트스트림에 저장한다.

[0080] ”비트플래인 코딩”을 거친 후에 나온 각 비트플래인의 비트스트림의 구성은 일 예로 도 14와 같이 최상위(MSB; Most Significant Bit) 비트플래인으로부터 최하위(LSB; Least Significant Bit) 순으로 차례로 구성할 수 있다.

[0081] “비트플래인 코딩”을 거친 후에 나온 각 비트플래인의 비트스트림의 구성의 또다른 일 예로 도 15와 같이 각 비트플래인 사이에 구분자를 두어 각 비트플래인 사이를 구분할 수도 있다.

[0082] “비트플래인 코딩”을 거친 후에 나온 각 비트플래인의 비트스트림의 구성의 또다른 일 예로 도 16과 같이 각 비트플래인 사이에 헤더(header) 정보를 넣을 수도 있다. 헤더 정보는 각 비트플래인의 속성을 정의하거나 코딩에 필요한 정보 등을 포함할 수 있다.

[0083] 도 4의 “비트플래인 코딩”을 수행하는 방법의 또다른 실시 일 예로 상기 살펴본 MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법이 있다.

[0084] ① MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법의 일 예로 전체 코딩 과정중에서 일부의 과정을 제거하고 수행하는 방법이 있다. 일 예로 “CAE 코딩” 과정에서 도 10의 “비트율 조절” 부분을 제거하고 수행할 수 있다. 즉, 변환비율(CR)을 항상 '1'로 하여 코딩을 수행할 수 있다. 또 다른 일 예로 픽쳐 모드와 관계 없이 비트플래인 블록 코딩 시에 인트라(intra) 코딩만을 수행하는 방법이 있다. 이와 같은 경우에는 상기 설명한 것 중, “all_0”, “all_1”, “intraCAE” 세 가지 모드만을 사용할 수 있다.

[0085] ② MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법의 또 다른 일 예로 “CAE 코딩” 과정에서 각 컨텍스트 템플릿에 대한 확률을 변경할 수 있다. 기존의 방법에서는 이진 형상 영상(shape image)에 적합한 확률이었으나, 이를 깊이 정보맵에 적합한 확률로 수정하여 사용할 수도 있다.

[0086] ③ MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법의 또 다른 일 예로 움직임 정보를 각 비트플래인 블록별로 저장하지 않고, 각 블록 단위로 움직임 정보를 하나만을 저장할 수도 있다.

[0087] 도 4의 “비트플래인 코딩”을 수행하는 방법의 또다른 실시 일 예로 Run Length Coding 방법과 Variable Length Coding 방법을 사용하여 코딩을 수행할 수도 있다.

[0088] 도 4의 "비트플래인 코딩"을 수행하는 방법의 또다른 실시 일 예로 주변의 코딩 정보를이용한 Context-based 코딩과 Arithmetic Coding 방법을 사용하여 수행할 수도 있다.

[0089] 도 4의 "비트플래인 코딩"을 수행하는 방법의 또다른 실시 일 예로 다양한 이진 영상 코딩 방법을 사용하여 코딩을 수행할 수도 있다. 일 예로 CAC(Constant Area Coding) 혹은 JBIG(Joint Bi-Levelnary Image Processing Group)의 방법을 사용할 수도 있다.

[0090] 도 4에서 비트플래인 코딩이 선택이 되어 "비트플래인 단위 코딩"을 수행하고, 코딩을수행한 데이터는 이후에 입력되는 깊이 정보맵의 코딩을 위해 "비트플래인 단위 디코딩"을 수행한다. "비트플래인 단위 디코딩" 과정의 일 예는 도 17과 같다.

[0091] 도 17의 "비트플래인 디코딩" 과정에서는 입력된 비트스트림을 참조 영상과 재구성된 영상을 이용해서 각 비트플래인 별로 디코딩하여 n개의 비트플래인 영상을 출력한다. "비트플래인 결합" 과정에서는 출력된 각각의 비트플래인 영상을 결합하여 n-bit로 표현되는 영상을 출력한다.

[0092] 도 17의 "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 수식과 같다.

수학식 3

[0093]
$$a_{m-1} = g_{m-1}$$
$$a_i = a_{i+1} \oplus g_i \qquad 0 \le i \le m-1$$

[0094] 여기서 $\oplus$는 XOR(eXclusive OR) 연산을 의미한다.

[0095] 도 17의 "깊이 정보맵 구성" 과정에서는 실제 깊이 정보맵의 비트수에 맞게 구성하여 "재구성된 깊이 정보맵"을 출력한다.

[0096] "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 18과 같다.

[0097] 도 18의 "DE-MUX" 과정에서는 비트스트림을 입력으로 받아 현재 비트플래인 블록의 움직임 벡터와 현재 비트플래인 블록의 모드, 그리고 "CAE 디코딩" 과정에서 사용될 비트스트림을 출력한다. 입력 받은 현재 비트플래인 블록의 모드에 따라 디코딩을 수행하게 된다. 만약 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. 만약 모드가 'No Update With MV'라면 전송 받은 움직임 벡터값을 이용하여 "움직임 보상"을 수행하여 움직임 벡터에 대응하는 참조 비트플래인 블록을 출력한다. 만약 모드가 'all_0' 이라면, "동일 레벨 블록 디코딩"이 수행이 되며, 블록 내의 모든 픽셀값을 '0'으로 설정하여 출력한다. 만약 모드가 'all_1' 이라면, "동일 레벨 블록 디코딩"이 수행이 되며, 블록 내의 모든 픽셀값을 '1'로 설정하여 출력한다. 만약 모드가 'intraCAE' 라면, "CAE(Context-based Arithmetic Encoding) 디코딩"이 수행이 되며, 현재 비트플래인 블록에서 디코딩을 수행할 각 픽셀의 주변 픽셀 정보를 기반으로 이진 산술 코딩(Binary Arithmetic Coding)을 수행한다. 만약 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.

[0098] 현재 비트플래인 블록의 모드가 'intraCAE'와 'interCAE'로 선택 되었을 경우, 컨텍스트 기반 산술 인코딩(CAE, Context-based Arithmetic Encoding) 방법을 사용하며, CAE 디코딩의 구조도는 도 19와 같다.

[0099] 도 19의 "변환비율 디코딩" 과정에서는 입력된 비트스트림에서 변환비율(CR)을 디코딩하여 출력한다. "다운샘플링" 과정에서는 입력된 변환비율에 따라 참조 비트플래인 블록에 대한 다운샘플링을 수행하여 컨텐스트 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 10의 "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. "업샘플링

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

[0100] 도 17의 “비트플래인 디코딩”을 수행하는 방법의 또다른 실시 일 예로 상기 살펴본 MPEG-4 Part 2 Visual의 이진 형상 디코딩 방법을 응용하는 방법이 있다.

[0101] ① MPEG-4 Part 2 Visual의 이진 형상 디코딩 방법을 응용하는 방법의 일 예로 전체 디코딩 과정에서 일부의 과정을 제거하고 수행하는 방법이 있다. 일 예로 “CAE 디코딩” 과정에서 변환비율(CR)을 항상 '1'인 데이터만을 디코딩할 수 있다. 또 다른 일 예로 픽쳐 모드와 관계 없이 비트플래인 디코딩을 수행할 때 인트라(intra) 디코딩만을 수행하는 방법이 있다. 이와 같은 경우에는 상기 설명한 모드 중, “all_0”, “all_1”, “intraCAE” 세 가지 모드만을 디코딩할 수 있다.

[0102] ② MPEG-4 Part 2 Visual의 이진 형상 디코딩 방법을 응용하는 방법의 또 다른 일 예로 “CAE 디코딩” 과정에서 각 컨텍스트 템플릿에 대한 확률을 변경할 수 있다. 기존의 방법에서는 이진 형상 영상에 적합한 확률이었으나, 이를 깊이 정보맵에 적합한 확률로 수정하여 사용할 수도 있다.

[0103] ③ MPEG-4 Part 2 Visual의 이진 형상 코딩 방법을 응용하는 방법의 또 다른 일 예로 블록 단위로 저장되어 있는 움직임 정보를 각 비트플래인에서 사용할 수 있도록 하는 방법이 있다.

[0104] 도 17의 “비트플래인 디코딩”을 수행하는 방법의 또다른 실시 일 예로 Run Length Coding 방법과 Variable Length Coding방법을 사용하여 디코딩을 수행할 수도 있다.

[0105] 도 17의 “비트플래인 디코딩”을 수행하는 방법의 또다른 실시 일 예로 주변의 코딩 정보를 이용한 Context-based 디코딩 방법과 Arithmetic Coding 방법을 사용한 디코딩 방법을 수행할 수도 있다.

[0106] 도 17의 “비트플래인 디코딩”을 수행하는 방법의 또다른 실시 일 예로 다양한 이진 영상 코딩 방법을 사용하여 디코딩을 수행할 수도 있다. 일 예로 CAC(Constant Area Coding) 혹은 JBIG(Joint Bi-Levelnary Image Processing Group)의 방법을 사용할 수도 있다.

[0107] <디코더>

[0108] 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 디코더 구조도의 일 예는 도 20과 같다.

[0109] 도 20에서 입력된 비트스트림은 비트플래인 디코딩 모드 혹은 기존 디코딩 모드를 통해 디코딩되어 재구성된 깊이 정보맵 블록을 출력한다. 비트플래인 디코딩 모드가 선택되었을 경우에는 “비트플래인 단위 디코딩”이 수행되며, 비트스트림을 입력받아 각 비트플래인 별로 디코딩하고 각각의 비트플래인 영상을 결합하여 N-bit로 표현되는 재구성된 깊이 정보맵 블록을 출력한다. 기존 디코딩 모드가 선택되었을 경우에는 도 20에서 'B'로 표시되어 있는 기존의 동영상 코딩 방법(MPEG-1, MPEG-2, MPEG-4 Part 2 Visual, H.264/AVC, SVC, MVC, VC-1, AVS, KTA, 등)을 수행하며, 비트스트림을 입력받아 디코딩하여 재구성된 깊이 정보맵 블록을 출력한다.

[0110] 도 20에서 기존 디코딩 모드의 'B'의 일 예로 H.264/AVC의 디코더를 보여준다. 'B'의 디코더 구조도에서 데이터를 처리하는 단위는 가로 세로 16x16픽셀 크기의 매크로블록(macroblock)이며, 인트라 모드 또는 인터 모드로 디코딩이 수행된다. 인트라(intra) 모드일 경우, 'B' 내부의 스위치가 인트라로 전환이 되며, 인터(inter) 모드일 경우에는 스위치가 인터로 전환이 된다. 디코딩 과정의 주요한 흐름은 먼저 예측 블록을 생성한 후, 입력받은 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 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.

[0111] 도 20에서 “비트플래인 단위 디코딩” 과정은 도 17의 “비트플래인 단위 디코딩” 과정과 동일하며, 상기 설명과 동일함으로 설명은 생략한다.

[0112] <모드 선택 방법>

[0113] 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택할 수 있는 모드 선택 방법이 필요한데, 본 발명에서 제안하는 모드 선택 방법의 일 예는 도 21과 같으며, 자세한 알고리즘은 다음과 같다.

[0114] Step 1) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드의 코스트(J_A)를 계산하고 또한 비트플래인 코딩 모드의 코스트(J_B)를 계산한다.

[0115] Step 2) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드의 코스트(J_A)가 비트플래인 코딩 모드의 코스트(J_B)보다 작다면, Step 3로 분기한다. 만약 그렇지 않다면, Step 4로 분기한다.

[0116] Step 3) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드로 코딩한다.

[0117] Step 4) 현재 깊이 정보맵 블록을 비트플래인 코딩 모드로 코딩한다.

[0118] 도 21의 코딩 모드 결정 순서도에서 각 코딩 모드에 대한 코스트(J_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 아래와 같다.

수학식 4

[0119] $$J_i = SAD_i + \lambda_{SAD} \cdot R_i$$

[0120] SAD_i (sum of absolute difference)는 원본 영상과 재구성된 영상 간의 예측 오차의 절대값의 합을 의미하고, 'λ_{SAD}'는 SAD를 사용할 때의 "Lagransian" 상수를 나태내며, R_i는 해당하는 블록 모드로 코딩을 수행하였을 때의 비트량을 의미한다.

[0121] 또한 코스트를 계산하는 방법의 또 다른 일 예로 SAD(sum of absolute difference)를 사용할 수도 있다. 또한 코스트를 계산하는 방법의 또 다른 일 예로 SSD(sum of square difference)를 사용할 수도 있다.

[0122] <제안하는 코딩 방법의 전체적인 코딩 흐름 및 실시 구현의 일 예>

[0123] 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 전체적인 코딩 흐름은 도 22와 같으며, 자세한 알고리즘은 다음과 같다.

[0124] 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 21의 방법을 사용할 수 있다.

[0125] Step 2) 모드 결정 과정에서 비트플래인 단위 코딩 모드가 선택이 되었다면, 비트플래인 단위 코딩을 수행할지에 대한 플래그 정보인 'bitplane_coding_flag'를 1로 설정하고, 만약 기존 방법 코딩 모드로 선택이 되었다면 'bitplane_coding_flag'를0으로 설정한다.

[0126] Step 3) 만약 bitplane_coding_flag가 '0'이라면 Step 4로 분기한다. 만약 그렇지 않다면 Step 5로 분기한다.

[0127] Step 4) 현재 깊이 정보맵 블록을 기존 동영상 코딩 모드로 코딩한다.

[0128] Step 5) 현재 깊이 정보맵 블록을 비트플래인 코딩 모드로 코딩한다.

[0129] 상기 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법을 국제 동영상 표준인 H.264/AVC에 실제 구현한 일 예로 Macroblock layer(macroblock_layer) 신택스를 다음과 같이 수정하여 구현할 수 있다.

표 5

[0130]

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++){		
bab_type		u(v) | ae(v)
if(bab_type = = NoUpdateWithMV || bab_type = = InterCAE){		
mvd_x		u(v) | ae(v)
mvd_y		u(v) | ae(v)
}		
if(bab_type == IntraCAE || bab_type == InterCAE){		
scan_type		u(1) | ae(v)
binary_arithmetic_code( )		
}		
}		
}		
}		

[0131] “bitplane_coding_flag”는 현재의 깊이 정보맵 블록이 비트플래인 단위 코딩 모드로 코딩되었는지의 여부를 나타내며, “bitplane_coding_flag” 값이 '1'일 경우에는 비트플래인 단위 코딩을 수행한 것이며, '0'일 경우에는 기존 방법으로 코딩을 수행한 것이다. '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' 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.

[0132] 본 발명에서 제안하는 방법의 우수성을 검증하기 위해 본 발명의 발명자는 H.264/AVC의 참조 소프트웨어인 JM(Joint Model) 13.2와 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 영상 시퀀스와 Breakdancers XGA(1024x768) 15Hz 영상 시퀀스에 대한 비교를 수행하였다.

[0133] 도 23, 도 24는 H.264/AVC(Anchor)와 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 H.264/AVC의 결과보다 성능이 월등하게 향상된 것을 확인할 수 있으며, 평균적인 PSNR 향상을 나타내는 BD-PSNR 방법을 사용하여 성능을 비교하였을 때 평균적으로 0.65 dB 향상이 되었고, 평균적인 bit-rate 감소량을 나타내는 BD-rate 방법을 사용하여 성능을 비교하였을 때 평균적으로 9.8% 감소하였다.

[0134] 본 발명에서 깊이 정보맵을 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 인코더 장치도는 도 25와 같다.

[0135] 도 25에서 입력된 영상은 n-bit로 표현되는 깊이 정보맵이며, 출력은 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법으로 코딩된 결과인 비트스트림이다. “모드 결정부”에서는 n-bit로 표현되는 깊이정보맵을 입력받아 상기 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택할 수 있는 모드 선택 방법을 수행하며, 수행하는 방법의 일 예로 상기 **도** 21의 모드 선택 방법을 사용할 수 있다. “모드 결정부”에서 비트플래인 코딩 모드가 선택되었을 경우, 비트플래인 단위 코딩 방법인 “비트플래인 단위 인코딩부”가 수행되며, 수행하는 방법의 일 예로 상기 도 5의 비트플래인 단위 인코딩 방법을 사용할 수 있다. “모드 결정부”에서 기존 코딩 모드가 선택되었을 경우, 기존 코딩 방법인 도 25의 'A'가 수행되며, 수행하는 방법의 일 예로 상기 도 4의 기존 코딩 방법('A')을 사용할 수 있다. “참조 영상 버퍼부”에는 시간 방향으로 이전에 코딩되고 디코딩된 깊이 정보맵이 저장된다. “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. “비트플래인 단위 디코딩부”에서는 “모드 결정부”에서 비트플래인 코딩 모드가 선택되었을 경우에 수행되며, 수행하는 방법의 일예로 상기 도 17의 비트플래인 단위 디코딩 방법을 사용할수 있다.

[0136] 본 발명에서 깊이 정보맵을 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 디코더 장치도는 도 26과 같다.

[0137] 도 26에서 입력은 비트스트림이며, 출력은 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법으로 디코딩된 결과인 재구성된 깊이 정보맵이다. “코딩 모드 판단부”에서는 비트스트림을 입력받아 현재 디코딩할 깊이 정보맵의 코딩 정보를 디코딩하여, 비트플래인 디코딩 모드로 디코딩할지 아니면 기존 디코딩 모드로 디코딩할지 여부를 판단한다. 비트플래인 디코딩 모드가 선택되었을 경우, “비트플래인 단위 디코딩부”가 수행되며, 수행하는 방법의 일 예로 상기 도 17의 “비트플래인 단위 디코딩” 방법을 수행할 수 있다. 기존 디코딩 모드가 선택되었을 경우, 도 26의 기존 디코딩 방법('B')을 수행하며, 수행하는 방법의 일 예로 상기 도 20의 기존 디코딩 방법('B')을 수행할 수 있다. “참조 영상 버퍼부”에는 시간 방향으로 이전에 코딩되고 디코딩된 깊이 정보맵이 저장된다. “디블록킹 필터부”에서는 코딩 과정에서 발생한 블록킹 현상(blocking artifact)을 제거한 후, “재구성된 영상 버퍼부”에 저장한다.

[0138] 상기와 같이 본 발명의 일 실시예에 따른 적응적 블록기반 깊이 정보맵 코딩 방법 및 장치의 동작 및 구성이 이루어질 수 있으며, 한편 상기한 본 발명의 설명에서는 구체적인 실시예에 관해 설명하였으나 여러 가지 변형이 본 발명의 범위를 벗어나지 않고 실시될 수 있다.

도면의 간단한 설명

[0139] 도 1은 실제 영상(왼쪽)과 깊이 정보맵영상(오른쪽)을 나타내는 예시도

[0140] 도 2는 도 1의 실제 영상과 깊이 정보맵의 각 픽셀의 level을 표현한 3차원 그래프(왼쪽)와, 실제 영상의 그래프(오른쪽)의 깊이 정보맵의 그래프를 나타내는 도면

[0141] 도 3은 도 1의 실제 영상의 비트플래인 표현(왼쪽)과, 깊이 정보맵의 비트플래인(가운데)과, 깊이 정보맵의 비트플래인에 gray code를 적용(오른쪽)한 비트플래인 분석을 나타낸 도면

[0142] 도 4는 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 인코더의 블록 구성도

[0143] 도 5는 비트플래인 단위 인코딩의 블록 구성도

[0144] 도 6은 비트플래인 코딩을 수행을 위한 일 예시 블록 구성도

[0145] 도 7은 인트라 픽쳐 모드 결정을 위한 동작 흐름도

[0146] 도 8은 인터 픽쳐 모드 결정을 위한 동작 흐름도

[0147] 도 9는 움직임 예측값(MV_P) 결정 방법의 동작 흐름도

[0148] 도 10은 CAE 인코딩의 블록 구성도

[0149] 도 11은 변환비율(CR; Conversion Ratio) 결정의 동작 흐름도

[0150] 도 12는 intraCAE를 위한 컨텍스트 템플릿을 나타낸 도면

[0151] 도 13은 interCAE를 위한 컨텍스트 템플릿을 나타낸 도면

[0152] 도 14는 각 비트플래인의 비트스트림의 구성의 일 예시도

[0153] 도 15는 각 비트플래인의 비트스트림의 구성의 일 예시도

[0154] 도 16은 각 비트플래인의 비트스트림의 구성의 일 예시도

[0155] 도 17은 비트플래인 단위 디코딩의 블록 구성도

[0156] 도 18는 비트플래인 디코딩을 수행하는 방법의 일 예를 나타낸 블록 구성도

[0157] 도 19는 CAE 디코딩의 블록 구성도

[0158] 도 20은 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 디코더의 블록 구성도

[0159] 도 21은 코딩 모드 결정 방법의 일 예를 나타낸 흐름도

[0160] 도 22는 "bitplane_coding_flag" 값에 따른 코딩 모드 수행 동작 흐름도

[0161] 도 23은 실제 제안하는 방법의 실험 결과(Ballet)를 나타낸 그래프(Anchor: 기존의 방법, Proposal: 제안하는 방법)

[0162] 도 24는 실제 제안하는 방법의 실험 결과(Breakdancers)를 나타낸 그래프(Anchor: 기존의 방법, Proposal: 제안하는 방법)

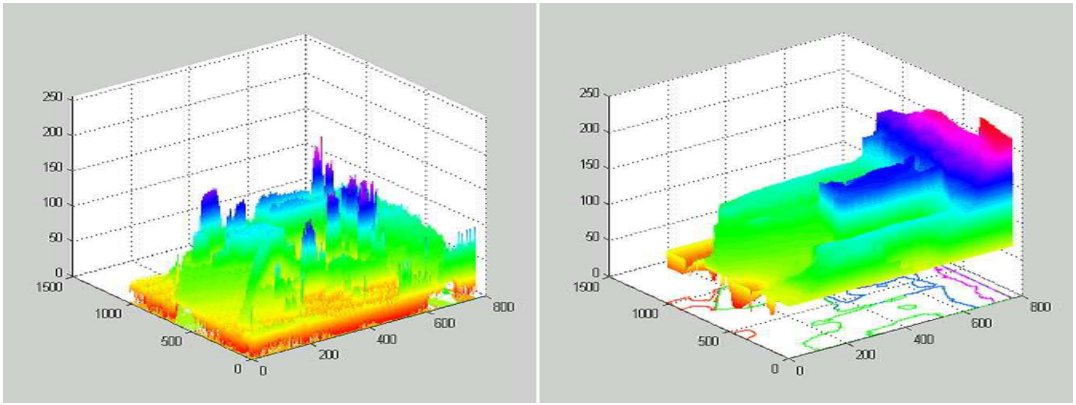
[0163] 도 25는 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 인코더 장치의 블록 구성도

[0164] 도 26은 기존의 동영상 코덱과 비트플래인 단위 코딩 방법을 적응적으로 선택하여 코딩하는 방법의 디코더 장치의 블록 구성도

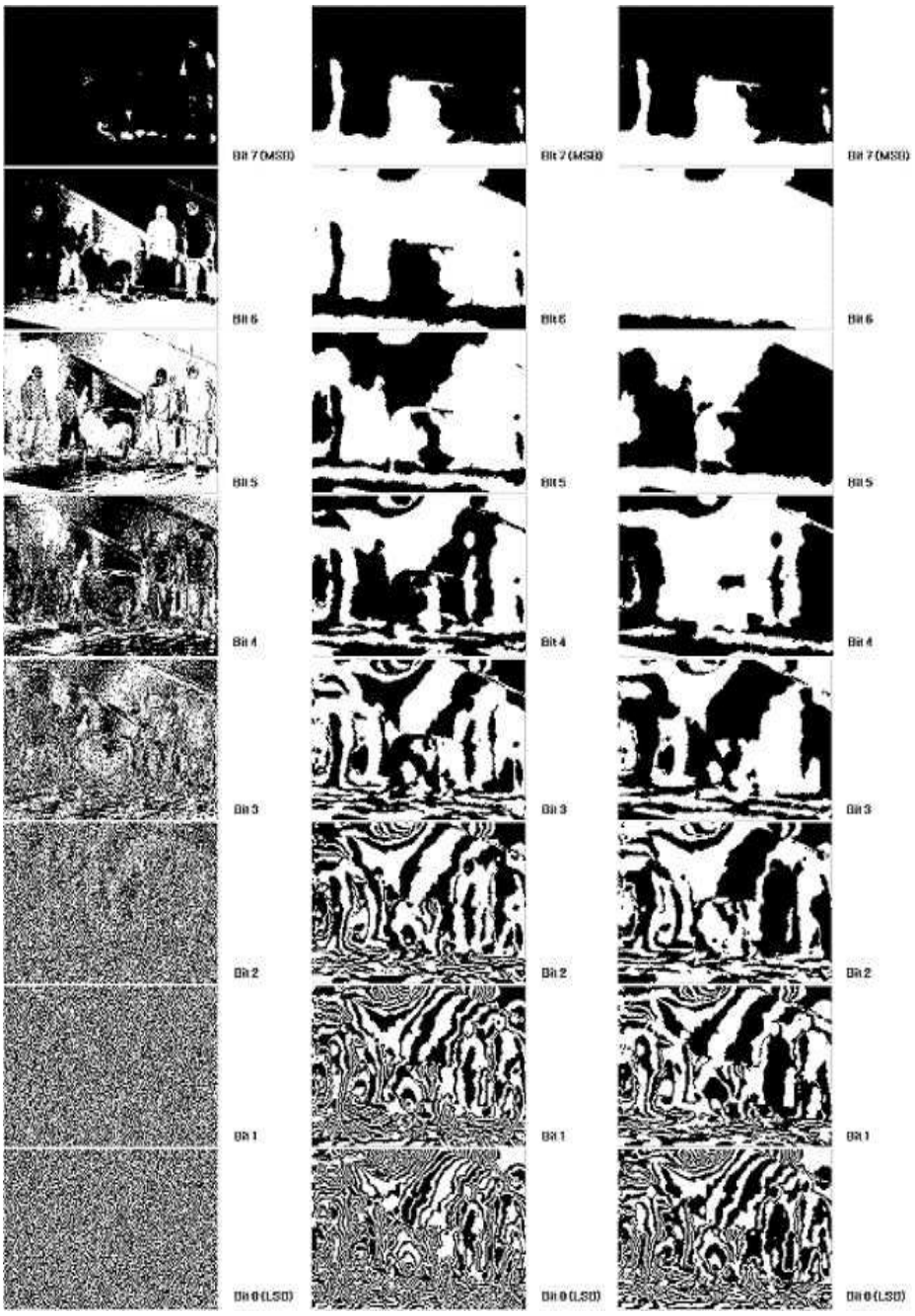
도면

도면1

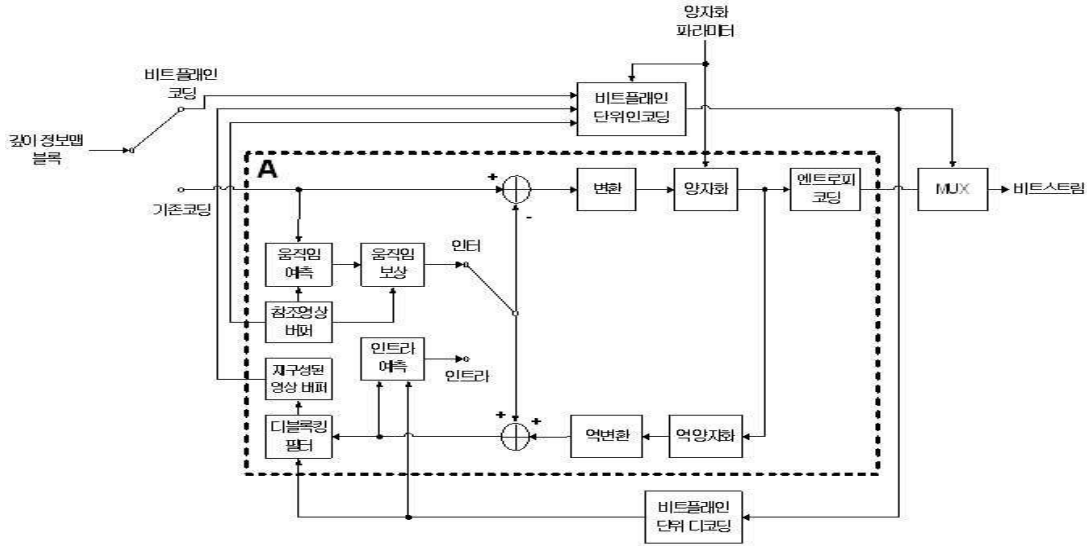
도면2



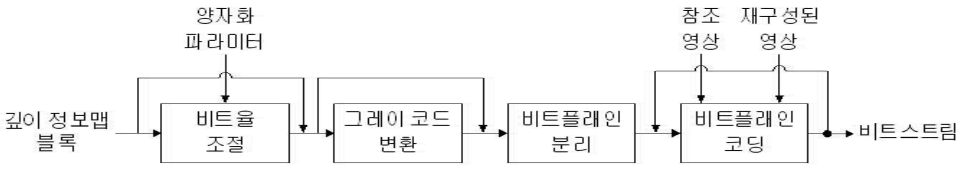
도면3


Bit 7 (MSB)
Bit 6
Bit 5
Bit 4
Bit 3
Bit 2
Bit 1
Bit 0 (LSB)
Bit 7 (MSB)
Bit 6
Bit 5
Bit 4
Bit 3
Bit 2
Bit 1
Bit 0 (LSB)
Bit 7 (MSB)
Bit 6
Bit 5
Bit 4
Bit 3
Bit 2
Bit 1
Bit 0 (LSB)

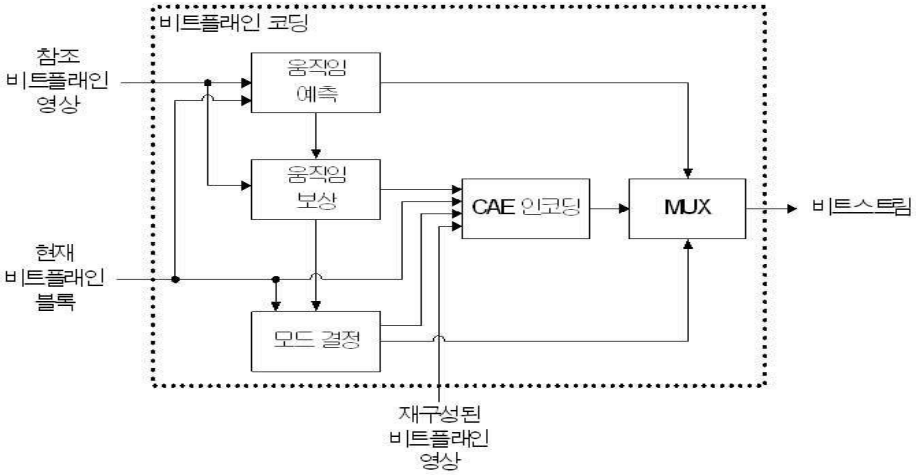
도면4


양자화
파라미터
비트플래인
코딩
깊이 정보맵
블록
비트플래인
단위인코딩
A
기존코딩
변환
양자화
엔트로피
코딩
MUX
비트스트림
움직임
예측
움직임
보상
인터
참조영상
버퍼
인트라
예측
인트라
재구성된
영상 버퍼
디블록킹
필터
역변환
역양자화
비트플래인
단위 디코딩

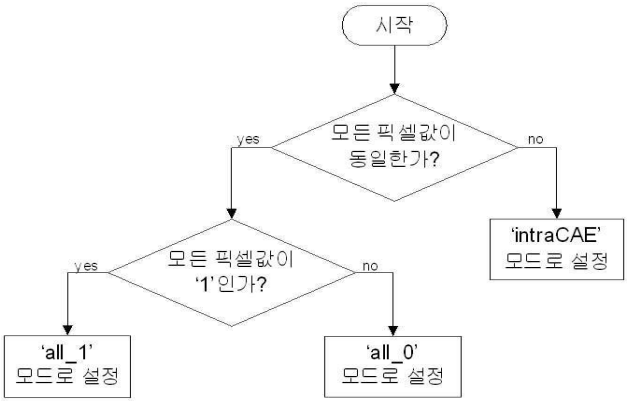
도면5


양자화
파라미터
참조
영상
재구성된
영상
깊이 정보맵
블록
비트율
조절
그레이 코드
변환
비트플래인
분리
비트플래인
코딩
비트스트림

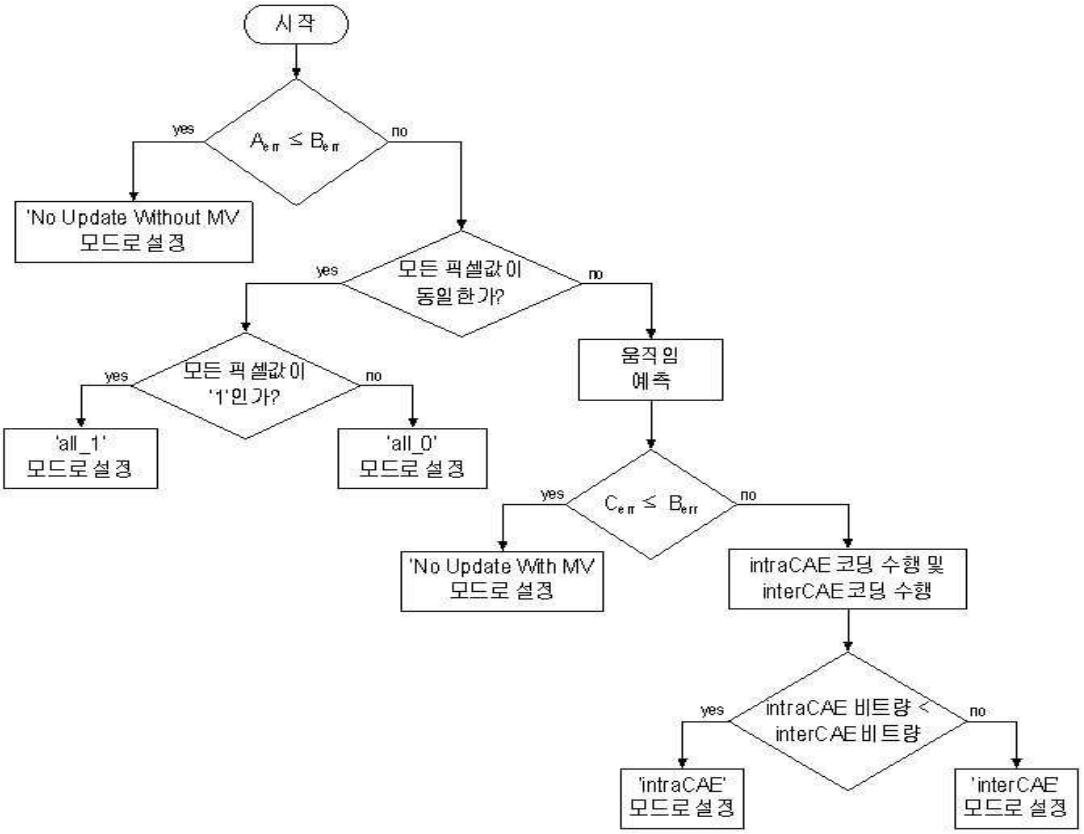
도면6


비트플래인 코딩
참조
비트플래인
영상
움직임
예측
움직임
보상
CAE 인코딩
MUX
비트스트림
현재
비트플래인
블록
모드 결정
재구성된
비트플래인
영상

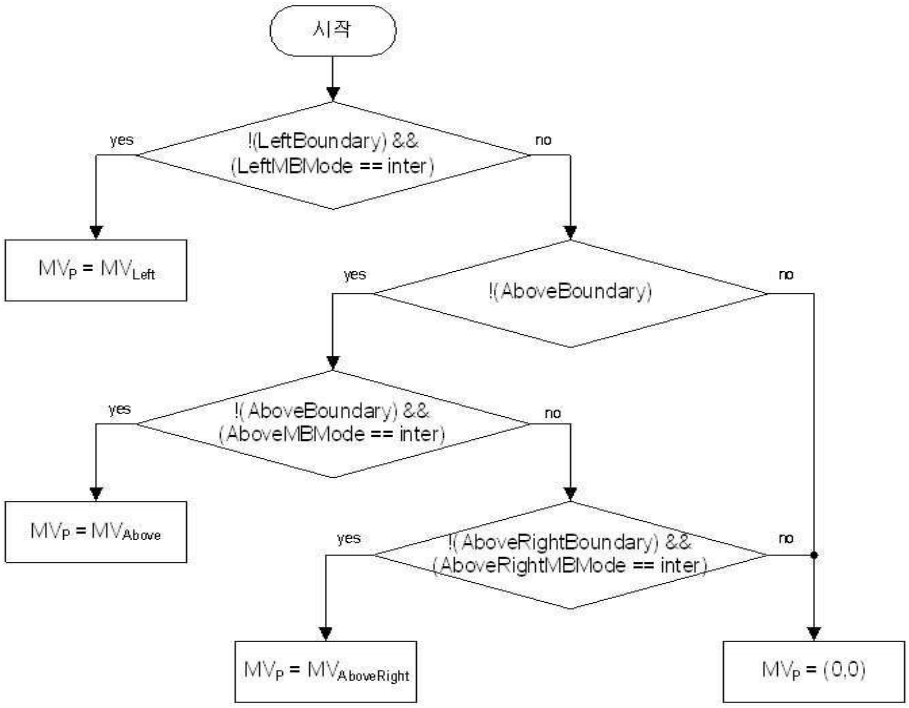
도면7


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

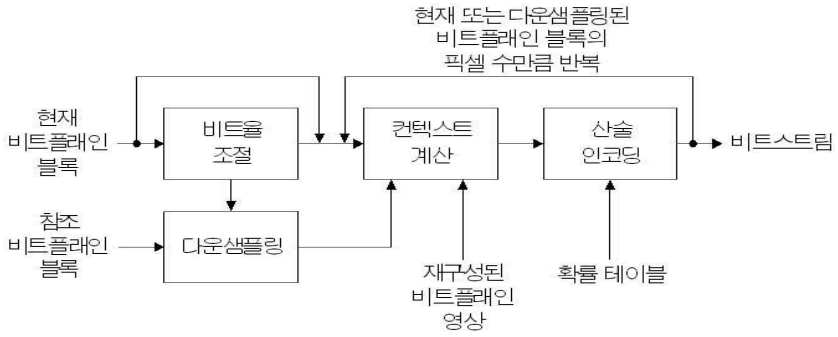
도면8


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

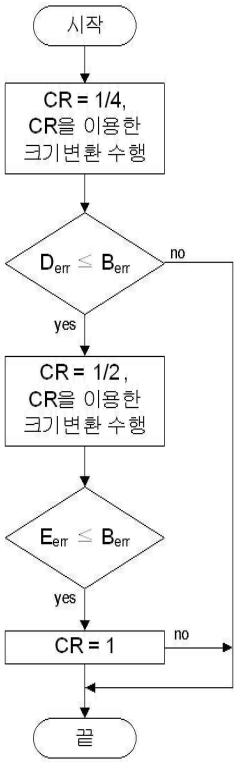
도면9


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

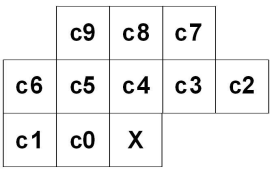
도면10


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
확률 테이블

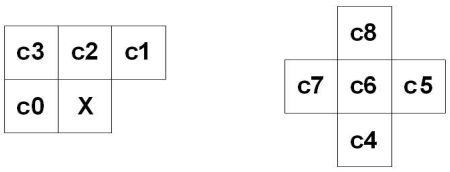
도면11



도면12



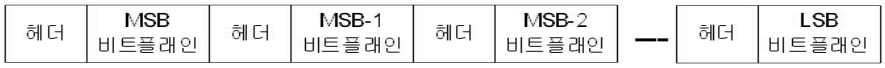
도면13



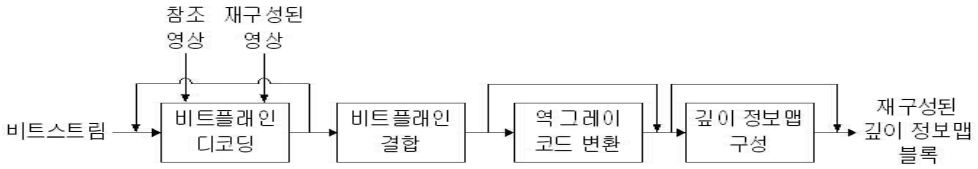
도면14

도면15

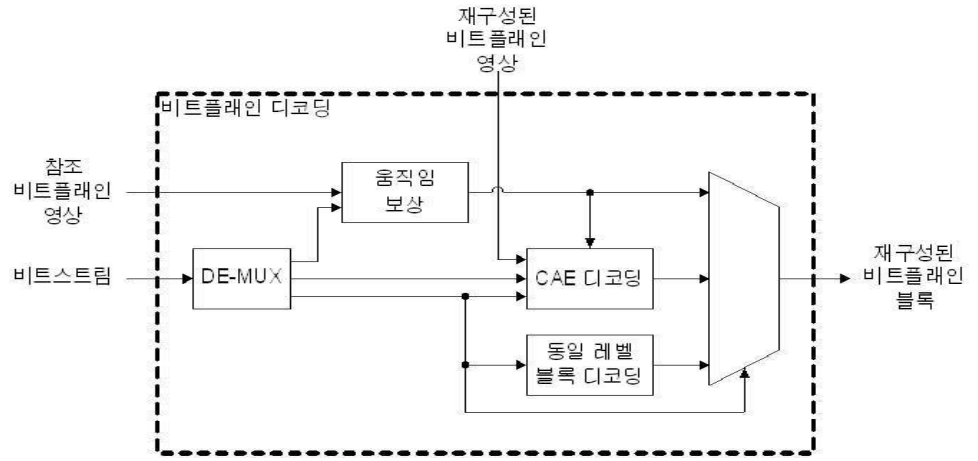
도면16



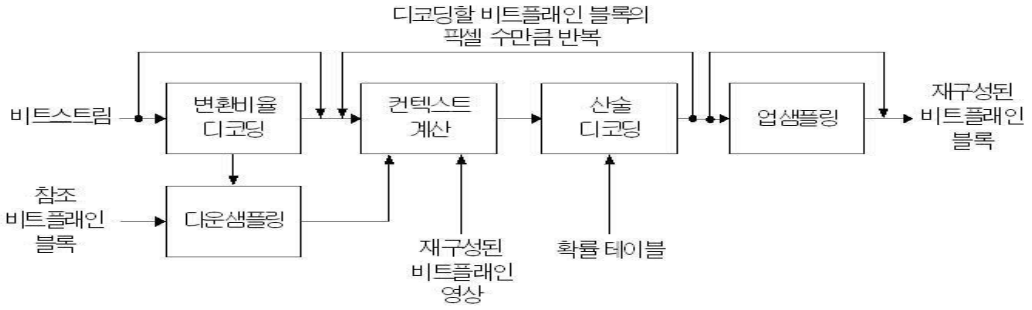
도면17



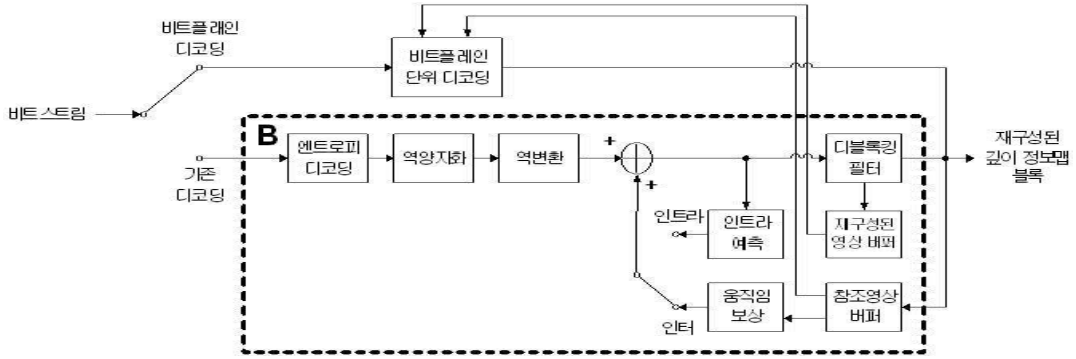
도면18



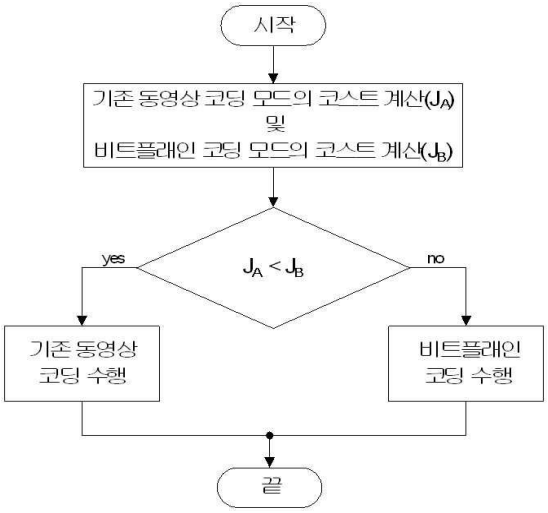
도면19



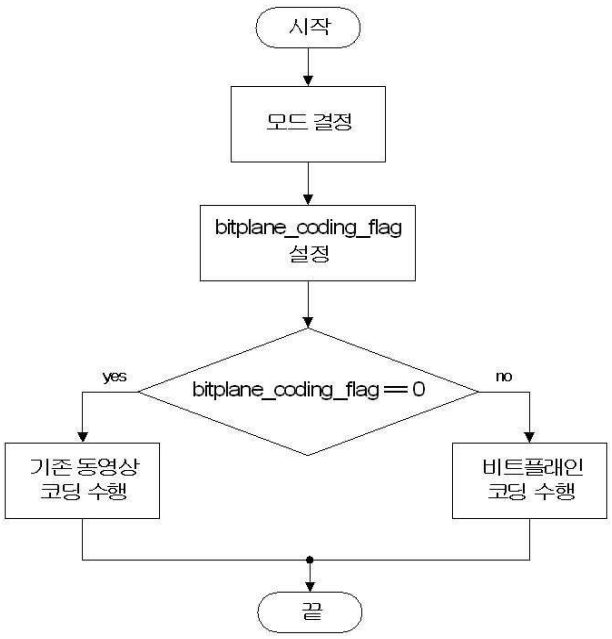
도면20



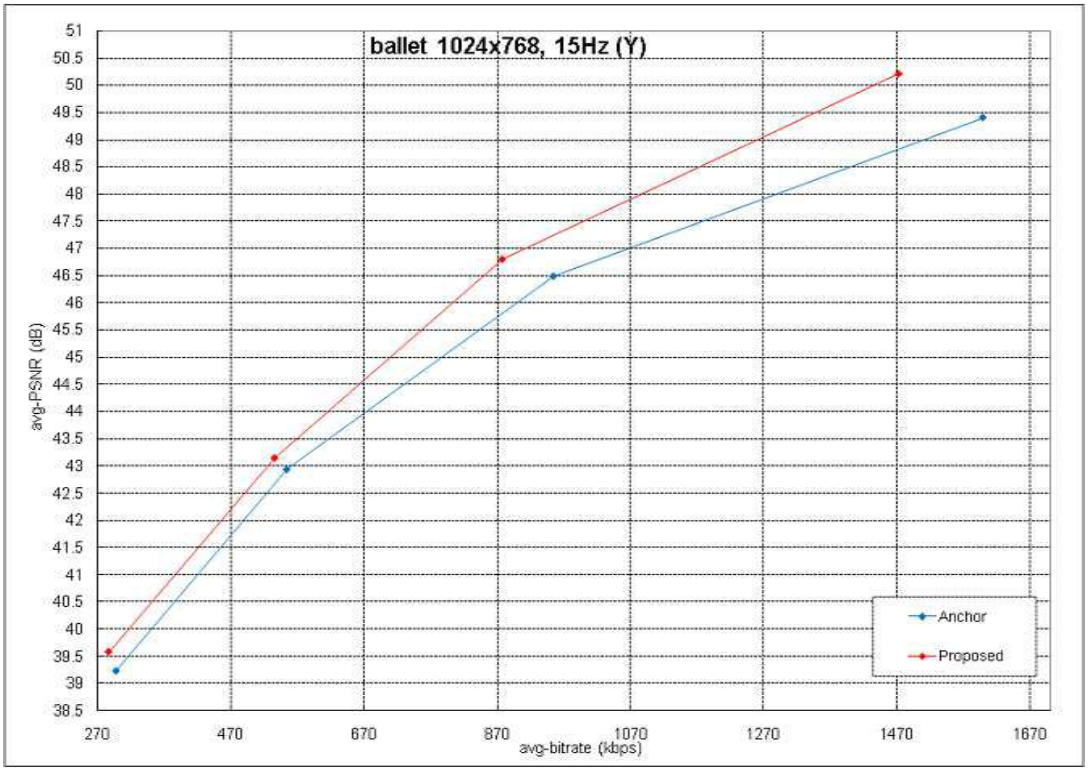
도면21



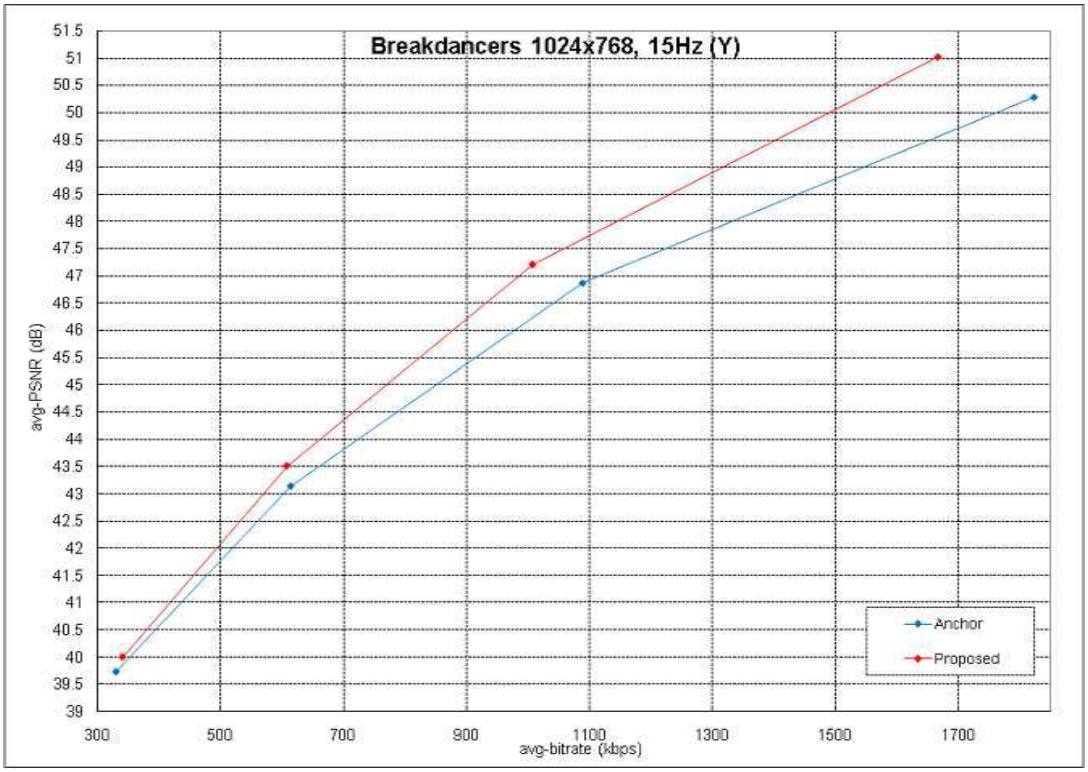
도면22



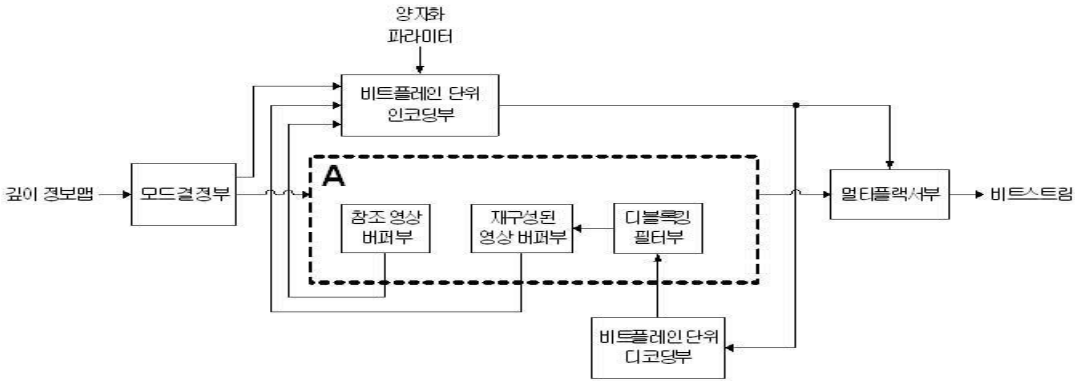
도면23


ballet 1024x768, 15Hz (Y)
avg-PSNR (dB)
avg-bitrate (kbps)
51
50.5
50
49.5
49
48.5
48
47.5
47
46.5
46
45.5
45
44.5
44
43.5
43
42.5
42
41.5
41
40.5
40
39.5
39
38.5
270
470
670
870
1070
1270
1470
1670
Anchor
Proposed

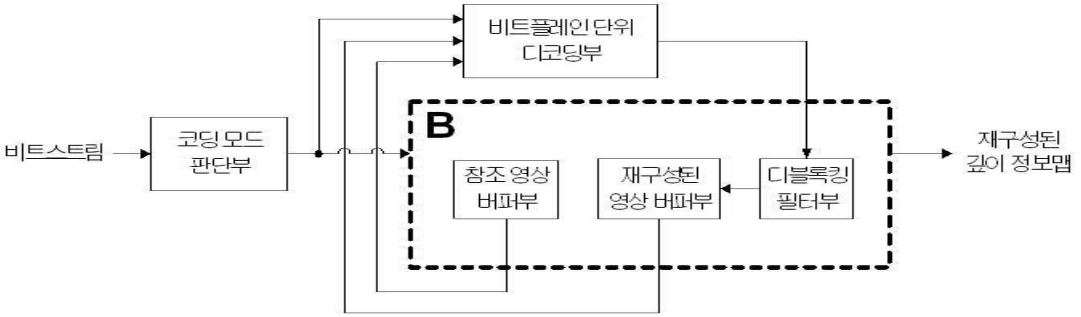
도면24


Breakdancers 1024x768, 15Hz (Y)
avg-PSNR (dB)
avg-bitrate (kbps)
51.5
51
50.5
50
49.5
49
48.5
48
47.5
47
46.5
46
45.5
45
44.5
44
43.5
43
42.5
42
41.5
41
40.5
40
39.5
39
300
500
700
900
1100
1300
1500
1700
Anchor
Proposed

도면25


양자화
파라미터
비트플레인 단위
인코딩부
깊이 정보맵
모드결정부
A
참조 영상
버퍼부
재구성된
영상 버퍼부
디블록킹
필터부
멀티플렉서부
비트스트림
비트플레인 단위
디코딩부

도면26


비트플레인 단위
디코딩부
비트스트림
코딩 모드
판단부
B
참조 영상
버퍼부
재구성된
영상 버퍼부
디블록킹
필터부
재구성된
깊이 정보맵