(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년08월02일
(11) 등록번호 10-2562028
(24) 등록일자 2023년07월27일

(51) 국제특허분류(Int. Cl.)
H04N 19/70 (2014.01) *H04N 19/119* (2014.01)
H04N 19/174 (2014.01) *H04N 19/46* (2014.01)
(52) CPC특허분류
H04N 19/70 (2015.01)
H04N 19/119 (2015.01)
(21) 출원번호 10-2020-7031463
(22) 출원일자(국제) 2019년03월25일
심사청구일자 2020년12월14일
(85) 번역문제출일자 2020년10월30일
(65) 공개번호 10-2020-0140314
(43) 공개일자 2020년12월15일
(86) 국제출원번호 PCT/EP2019/057432
(87) 국제공개번호 WO 2019/197145
국제공개일자 2019년10월17일
(30) 우선권주장
1805886.7 2018년04월09일 영국(GB)
(56) 선행기술조사문헌
JP2016046778 A*
KR1020150029723 A*
US20150016503 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
캐논 가부시끼가이샤
일본 도쿄도 오오따꾸 시모마루꼬 3쪼메 30방 2고
(72) 발명자
타케 조나탕
프랑스 35160 탈렁삭 라 프로아르디에르
우에드라오고 나엘
프랑스 35330 모르 드 브르타뉴 쿠투스
(뒷면에 계속)
(74) 대리인
장수길, 이중희

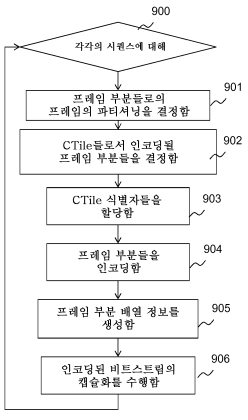
전체 청구항 수 : 총 19 항 심사관 : 김혜린

(54) 발명의 명칭 **프레임 부분들을 갖는 비디오 데이터를 인코딩 또는 디코딩하기 위한 방법 및 장치**

(57) 요 약

본 발명은 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 비트스트림으로 인코딩하는 방법에 관한 것이며, 이 방법은 적어도 하나의 프레임 부분을 하나 이상의 제1 인코딩된 유닛으로 인코딩하는 단계를 포함하며, 여기서 이 방법은, 적어도 하나의 프레임 부분 식별자를 제1 인코딩된 유닛들로 시그널링하는 단계 - 프레임 부분 식별자는 하나의 인코딩된 프레임 부분을 식별함 -, 및 프레임 부분 식별자를 포함하는 프레임 부분 배열 정보 및 프레임 부분에 관한 공간적 정보를 제공하는 단계를 더 포함한다.

대 표 도 - 도9



(52) CPC특허분류
H04N 19/174 (2015.01)
H04N 19/46 (2015.01)
(72) 발명자
드누알 프랑크
프랑스 35190 생 도미뇌크 라 빌 에 레 8
마즈 프레데릭
프랑스 35850 랑강 뤼 데 티욀 6

명 세 서

청구범위

청구항 1

프레임 부분들로 공간적으로 분할되는 프레임을 비트스트림으로 인코딩하는 방법으로서, 상기 방법은:

상기 프레임 내의 프레임 부분을 상기 비트스트림으로 인코딩하는 단계;

상기 프레임 내의 프레임 부분들 각각의 식별자를 상기 비트스트림으로 시그널링하는 단계; 및

상기 프레임 내의 프레임 부분의 위치에 관한 공간적 정보를 시그널링하는 단계를 더 포함하고,

상기 식별자 및 상기 공간적 정보는 상기 비트스트림 내의 하나의 파라미터 세트에서 시그널링되고, 상기 식별자를 나타내는 데 이용되는 비트 수의 정보는 상기 비트스트림에서 더 시그널링되는, 방법.

청구항 2

제1항에 있어서,

상기 프레임 부분은 독립적으로 인코딩되는, 방법.

청구항 3

제2항에 있어서,

상기 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 제공하는 단계를 더 포함하는, 방법.

청구항 4

삭제

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

제1항에 있어서,

상기 파라미터 세트는 하나 이상의 프레임에 관한 정보에 대하여 전용인, 방법.

청구항 9

삭제

청구항 10

제1항에 있어서,

상기 공간적 정보는 코딩 트리 유닛 어드레스에 의해 주어지는 상기 프레임 부분의 위치를 포함하는, 방법.

청구항 11

제1항에 있어서,

상기 공간적 정보는 상기 프레임 부분의 크기를 포함하는, 방법.

청구항 12

제10항에 있어서,

상기 프레임 부분의 위치는 프레임과 관련하여 주어지는, 방법.

청구항 13

제1항에 있어서,

상기 파라미터 세트는, 주어진 후필터링 알고리즘이 상기 프레임 부분에 이용될 수 있는지를 나타내는 플래그를 포함하는, 방법.

청구항 14

프레임 부분들로 공간적으로 분할되는 프레임을 포함하는 비디오 데이터를 비트스트림으로부터 디코딩하는 방법으로서, 상기 방법은:

상기 프레임 내의 프레임 부분들 각각의 식별자 및 상기 프레임 내의 프레임 부분의 위치에 관한 공간적 정보를 상기 비트스트림 내의 하나의 파라미터 세트로부터 획득하는 단계 - 획득될 상기 식별자 및 상기 공간적 정보는 상기 하나의 파라미터 세트에서 시그널링되고, 상기 식별자를 나타내는 데 이용되는 비트 수의 정보는 상기 비트스트림에서 더 시그널링됨 -;

상기 공간적 정보에 기반하여 프레임 내의 상기 프레임 부분의 위치를 결정하는 단계; 및

상기 비트스트림으로부터 상기 프레임 부분을 디코딩하는 단계를 포함하는, 방법.

청구항 15

제14항에 있어서,

상기 프레임 부분은 독립적으로 인코딩되는, 방법.

청구항 16

제14항에 있어서,

상기 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 획득하는 단계를 더 포함하는, 방법.

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

제14항에 있어서,

상기 파라미터 세트는 하나 이상의 프레임에 관한 정보에 대하여 전용인, 방법.

청구항 22

삭제

청구항 23

제14항에 있어서,

상기 공간적 정보는 코딩 트리 유닛 어드레스에 의해 주어지는 상기 프레임 부분의 위치를 포함하는, 방법.

청구항 24

제14항에 있어서,

상기 공간적 정보는 상기 프레임 부분의 크기를 포함하는, 방법.

청구항 25

제23항에 있어서,

상기 프레임 부분의 위치는 프레임과 관련하여 주어지는, 방법.

청구항 26

제14항에 있어서,

상기 파라미터 세트는, 주어진 후필터링 알고리즘이 상기 프레임 부분에 이용될 수 있는지를 나타내는 플래그를 포함하는, 방법.

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

프레임 부분들로 공간적으로 분할되는 프레임을 비트스트림으로 인코딩하기 위한 디바이스로서, 상기 디바이스는:

상기 프레임 내의 프레임 부분을 상기 비트스트림으로 인코딩하도록 구성된 회로를 포함하며, 상기 회로는, 또한,

상기 프레임 내의 프레임 부분들 각각의 식별자를 상기 비트스트림으로 시그널링하고,

상기 프레임 내의 프레임 부분의 위치에 관한 공간적 정보를 시그널링하도록 구성되고,

상기 식별자 및 상기 공간적 정보는 상기 비트스트림 내의 하나의 파라미터 세트에서 시그널링되고, 상기 식별자를 나타내는 데 이용되는 비트 수의 정보는 상기 비트스트림에서 더 시그널링되는, 디바이스.

청구항 31

프레임 부분들로 공간적으로 분할되는 프레임을 비트스트림으로부터 디코딩하기 위한 디바이스로서, 상기 디바이스는:

상기 프레임 내의 프레임 부분들 각각의 식별자 및 상기 프레임 내의 프레임 부분의 위치에 관한 공간적 정보를 상기 비트스트림 내의 하나의 파라미터 세트로부터 획득하고 - 획득될 상기 식별자 및 상기 공간적 정보는 상기 비트스트림 내의 하나의 파라미터 세트에서 시그널링되고, 상기 식별자를 나타내는 데 이용되는 비트 수의 정보

는 상기 비트스트림에서 더 시그널링됨 -;

상기 공간적 정보에 기반하여 프레임 내의 상기 프레임 부분의 위치를 결정하며;

상기 비트스트림으로부터 상기 프레임 부분을 디코딩하도록 구성된 회로를 포함하는, 디바이스.

청구항 32

삭제

청구항 33

프로그래밍가능한 장치를 위한 컴퓨터 판독가능한 저장 매체 상에 저장된 컴퓨터 프로그램으로서,

상기 컴퓨터 프로그램은, 상기 프로그래밍가능한 장치에 로딩되고 상기 프로그래밍가능한 장치에 의해 실행될 때, 제1항 내지 제3항, 제8항, 제10항 내지 제16항, 제21항, 제23항 내지 제26항, 제30항 및 제31항 중 어느 한 항에 따른 방법을 구현하기 위한 명령어들의 시퀀스를 포함하는, 컴퓨터 판독가능한 저장 매체 상에 저장된 컴퓨터 프로그램.

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

청구항 51

삭제

청구항 52

삭제

청구항 53

삭제

청구항 54

삭제

청구항 55

삭제

청구항 56

삭제

청구항 57

삭제

발명의 설명

기 술 분 야

[0001] 본 개시내용은 공간적 부분들을 포함하는 비디오 데이터를 인코딩 또는 디코딩하기 위한 방법 및 디바이스에 관한 것이다.

배 경 기 술

[0002] HEVC 타일들이 병렬 인코딩을 위해 도입되고 설계된다. 그러나, 큰 크기의 비디오 콘텐츠로 인해, 타일들이 상이하게 이용되는 몇 가지 이용 사례들이 존재한다. 특히, 개별 타일들 또는 타일들의 세트를 스트리밍하기 위한 필요성이 생겼다. 일부 애플리케이션은 또한, 새로운 비디오 시퀀스를 구성하기 위해, 동일한 시퀀스로부터의 또는 상이한 시퀀스들로부터의, 상이한 타일들의 조합에 대한 필요성을 진전시켰다.

[0003] HEVC에서의 현재 메커니즘들은 이러한 종류의 시나리오들을 염두에 두고 설계되지 않았다. 현재 HEVC 메커니즘을 이용하여 이러한 시나리오들을 구현하는 것은 타일들에 대한 인코딩 제약들을 추가하는 것을 의미하고, 디코딩 시에 타일들의 임의적인 구성은 데이터의 재기입 프로세스들을 수반한다. 특히, 슬라이스 세그먼트 헤더들의 조작을 포함하는 데이터의 재기입이 일반적으로 요구된다.

발명의 내용

[0004] 본 발명은 앞서 언급한 관심사들 중 하나 이상을 다루도록 창안되었다. 본 발명은 비트스트림 내의 프레임 부분들의 정의 및 이들 프레임 부분들의 시그널링에 관한 것이다. 본 발명은 디코딩 시에 이러한 프레임 부분들의 추출 및 재조합을 용이하게 하면서 그렇게 할 때 요구되는 재기입 프로세스들을 제한하는 것을 목표로 한다.

[0005] 본 발명의 제1 양태에 따르면, 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 비트스트림으로 인코딩하는 방법이 제공되며, 이 방법은 적어도 하나의 프레임 부분을 하나 이상의 제1 인코딩된 유닛으로 인코딩하는 단계를 포함하며, 여기서 이 방법은, 적어도 하나의 프레임 부분 식별자를 제1 인코딩된 유닛들로 시그널링하는 단계 - 프레임 부분 식별자는 하나의 인코딩된 프레임 부분을 식별함 -, 및 프레임 부분 식별자를 포함하는 프레임 부분 배열 정보 및 프레임 부분에 관한 공간적 정보를 제공하는 단계를 더 포함한다.

[0006] 본 발명의 제1 양태는 HEVC 타일들과 같은 공지된 타일링 설계에 비해 압축을 개선할 가능성을 가능하게 하면서 더 많은 유연성 및 더 간단한 조작들을 제공하는 이점을 갖는다.

[0007] 실시예에서, 프레임 부분 배열 정보는 하나의 제2 인코딩된 유닛으로 제공된다.

[0008] 실시예에서, 적어도 하나의 프레임 부분은 독립적으로 인코딩된다.

[0009] 실시예에서, 이 방법은 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 제공하는 단계를 더 포함한다.

[0010] 실시예에서, 하나 이상의 제1 인코딩된 유닛은 각각의 프레임 부분에 대해 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 포함한다.

[0011] 실시예에서, 하나 이상의 제1 인코딩된 유닛은 적어도 하나의 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 포함한다.

[0012] 실시예에서, 하나 이상의 제1 인코딩된 유닛은 프레임 부분을 인코딩하는 데 이용되는 인코딩 제약들의 레벨을 나타내는 플래그를 포함한다.

[0013] 실시예에서, 프레임 부분은 슬라이스이고, 제1 인코딩된 유닛은 데이터 부분을 포함하는 슬라이스 유닛이고, 플래그는 슬라이스 유닛의 데이터 부분의 슬라이스 세그먼트 헤더에 포함된다.

[0014] 실시예에서, 프레임 부분은 슬라이스이고, 제1 인코딩된 유닛은 데이터 부분을 포함하는 슬라이스 유닛이고, 프레임 부분 식별자는 슬라이스 유닛의 데이터 부분의 슬라이스 세그먼트 헤더에 포함된다.

[0015] 실시예에서, 제1 인코딩된 유닛은 헤더 부분 및 인코딩된 프레임 부분을 포함하는 데이터 부분을 포함하고, 프레임 부분 식별자는 헤더 부분에 포함된다.

[0016] 실시예에서, 프레임 부분 식별자는 모든 프레임 부분 인코딩된 유닛들에서 시그널링되고, 미리 정의된 프레임 부분 식별자 값은 프레임 부분이 독립적으로 인코딩되지 않았음을 나타낸다.

[0017] 실시예에서, 제2 인코딩된 유닛은 하나 이상의 프레임에 관한 정보에 대하여 전용인 파라미터 세트이다.

[0018] 실시예에서, 제2 인코딩된 유닛은 프레임 부분 정보에 대하여 전용인 파라미터 세트이다.

[0019] 실시예에서, 제1 인코딩된 유닛은 프레임 부분이 독립적으로 인코딩되었음을 나타내는 특정 유형을 갖는다.

[0020] 실시예에서, 프레임 부분 식별자는 고정된 미리 결정된 비트 수를 이용하여 인코딩된다.

[0021] 실시예에서, 프레임 부분 식별자는 시그널링된 비트 수를 이용하여 인코딩된다.

[0022] 실시예에서, 공간적 정보는 코딩 트리 유닛 어드레스에 의해 주어지는 프레임 부분의 위치를 포함한다.

[0023] 실시예에서, 공간적 정보는 샘플 어드레스에 의해 주어지는 프레임 부분의 위치를 포함한다.

[0024] 실시예에서, 공간적 정보는 프레임 부분의 크기를 포함한다.

[0025] 실시예에서, 프레임 부분의 위치는 프레임과 관련하여 주어진다.

[0026] 실시예에서, 몇 개의 파라미터 데이터 유닛들이 동일한 프레임 부분에 대한 상이한 프레임 부분 배열을 포함하는 비트스트림에서 시그널링된다.

[0027] 실시예에서, 제2 인코딩된 유닛은 주어진 후필터링(post-filtering) 알고리즘이 프레임 부분에 이용될 수 있는지를 나타내는 플래그를 포함한다.

[0028] 실시예에서, 동일한 프레임 부분 식별자가 프레임 부분 세트를 정의하는 몇 개의 프레임 부분을 식별하는 데 이용될 수 있다.

[0029] 실시예에서, 헤더 부분은 계층 식별자를 포함하고, 계층 식별자는 프레임 부분 식별자를 시그널링하는 데 이용된다.

[0030] 본 발명의 제2 양태에 따르면, 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 적어도 하나의 비트스트림으로부터 디코딩하는 방법이 제공되며, 이 방법은, 프레임 부분 식별자를 포함하는 프레임 부분 배열 정보 및 프레임 부분에 관한 공간적 정보를 비트스트림으로부터 획득하는 단계, 비트스트림 내의 하나 이상의 제1 인코딩된 유닛으로부터 적어도 하나의 프레임 부분을 추출하는 단계 - 프레임 부분은 프레임 부분 식별자를 포함함 -, 공간적 정보에 기반하여 프레임 내의 프레임 부분의 위치를 결정하는 단계, 및 결정된 위치에 따라 프레임 부분을 프레임으로 렌더링하도록 프레임 부분을 디코딩하는 단계를 포함한다.

[0031] 실시예에서, 프레임 부분 배열 정보는 하나의 제2 인코딩된 유닛으로 제공된다.

[0032] 실시예에서, 적어도 하나의 프레임 부분은 독립적으로 인코딩된다.

[0033] 실시예에서, 이 방법은 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 획득하는 단계를 더 포함한다.

[0034] 실시예에서, 하나 이상의 제1 인코딩된 유닛은 각각의 프레임 부분에 대해 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 포함한다.

[0035] 실시예에서, 하나 이상의 제1 인코딩된 유닛은 적어도 하나의 프레임 부분이 독립적으로 인코딩되었음을 나타내는 플래그를 포함한다.

[0036] 실시예에서, 하나 이상의 제1 인코딩된 유닛은 프레임 부분을 인코딩하는 데 이용되는 인코딩 제약들의 레벨을 나타내는 플래그를 포함한다.

[0037] 실시예에서, 프레임 부분은 슬라이스이고, 제1 인코딩된 유닛은 데이터 부분을 포함하는 슬라이스 유닛이고, 플래그는 슬라이스 유닛의 데이터 부분의 슬라이스 세그먼트 헤더에 포함된다.

[0038] 실시예에서, 프레임 부분은 슬라이스이고, 제1 인코딩된 유닛은 데이터 부분을 포함하는 슬라이스 유닛이고, 프레임 부분 식별자는 슬라이스 유닛의 데이터 부분의 슬라이스 세그먼트 헤더에 포함된다.

[0039] 실시예에서, 제1 인코딩된 유닛은 헤더 부분 및 인코딩된 프레임 부분을 포함하는 데이터 부분을 포함하고, 프레임 부분 식별자는 헤더 부분에 포함된다.

[0040] 실시예에서, 프레임 부분 식별자는 모든 프레임 부분 인코딩된 유닛들에서 시그널링되고, 미리 정의된 프레임 부분 식별자 값은 프레임 부분이 독립적으로 인코딩되지 않았음을 나타낸다.

[0041] 실시예에서, 제2 인코딩된 유닛은 하나 이상의 프레임에 관한 정보에 대하여 전용인 파라미터 세트이다.

[0042] 실시예에서, 제2 인코딩된 유닛은 프레임 부분 정보에 대하여 전용인 파라미터 세트이다.

[0043] 실시예에서, 제1 인코딩된 유닛은 프레임 부분이 독립적으로 인코딩되었음을 나타내는 특정 유형을 갖는다.

[0044] 실시예에서, 프레임 부분 식별자는 고정된 미리 결정된 비트 수를 이용하여 인코딩된다.

[0045] 실시예에서, 프레임 부분 식별자는 시그널링된 비트 수를 이용하여 인코딩된다.

[0046] 실시예에서, 공간적 정보는 코딩 트리 유닛 어드레스에 의해 주어지는 프레임 부분의 위치를 포함한다.

[0047] 실시예에서, 공간적 정보는 샘플 어드레스에 의해 주어지는 프레임 부분의 위치를 포함한다.

[0048] 실시예에서, 공간적 정보는 프레임 부분의 크기를 포함한다.

[0049] 실시예에서, 프레임 부분의 위치는 프레임과 관련하여 주어진다.

[0050] 실시예에서, 몇 개의 파라미터 데이터 유닛들이 동일한 프레임 부분에 대한 상이한 프레임 부분 배열을 포함하는 비트스트림으로부터 획득된다.

[0051] 실시예에서, 제2 인코딩된 유닛은 주어진 후필터링 알고리즘이 프레임 부분에 이용될 수 있는지를 나타내는 플래그를 포함한다.

[0052] 실시예에서, 동일한 프레임 부분 식별자가 프레임 부분 세트를 정의하는 몇 개의 프레임 부분을 식별하는 데 이용될 수 있다.

[0053] 실시예에서, 헤더 부분은 계층 식별자를 포함하고, 계층 식별자는 프레임 부분 식별자를 시그널링하는 데 이용된다.

[0054] 본 발명의 제3 양태에 따르면, 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 포함하는 새로운 비트스트림을 생성하는 방법이 제공되며, 이 방법은, 복수의 비트스트림으로부터 추출되고 새로운 비트스트림으로 병합될 복수의 프레임 부분을 결정하는 단계 - 복수의 비트스트림은 제1항 내지 제25항 중 어느 한 항에 따라 인코딩됨 -, 추출될 프레임 부분들의 프레임 부분 식별자들을 결정하는 단계, 새로운 비트스트림에 대한 프레임 부분 배열 정보를 생성하는 단계, 복수의 비트스트림으로부터 추출될 복수의 프레임 부분을 추출하는 단계, 및 복수의 프레임 부분 및 생성된 프레임 부분 배열 정보를 새로운 비트스트림에 임베딩하는 단계를 포함한다.

[0055] 실시예에서, 이 방법은, 추출된 프레임 부분에 대한 새로운 프레임 부분 식별자를 결정하는 단계, 및 프레임 부분 식별자를 추출된 프레임 부분 내에 새로운 프레임 부분 식별자로 대체하는 단계를 더 포함한다.

[0056] 실시예에서, 복수의 프레임 부분을 추출하는 단계는, 복수의 비트스트림을 파싱하는 단계, 및 결정된 프레임 부분 식별자들 중 하나를 포함하는 프레임 부분 인코딩된 데이터 유닛들을 추출하는 단계를 포함한다.

[0057] 본 발명의 제4 양태에 따르면, 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 비트스트림으로 인코딩하기 위한 디바이스가 제공되며, 이 디바이스는 적어도 하나의 프레임 부분을 하나 이상의 제1 인코딩된 유닛으로 인코딩하도록 구성된 회로를 포함하며, 방법은, 적어도 하나의 프레임 부분 식별자를 제1 인코딩된 유닛들로 시그널링하는 것 - 프레임 부분 식별자는 하나의 인코딩된 프레임 부분을 식별함 -, 및 프레임 부분 식별자를 포함하는 프레임 부분 배열 정보 및 프레임 부분에 관한 공간적 정보를 제공하는 것을 더 포함한다.

[0058] 본 발명의 제5 양태에 따르면, 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 적어도 하나의 비트스트림으로부터 디코딩하기 위한 디바이스가 제공되며, 이 디바이스는, 프레임 부분 식별자를 포함하는 프레임 부분 배열 정보 및 프레임 부분에 관한 공간적 정보를 비트스트림으로부터 획득하고, 비트스트림 내의 하나 이상의 제1 인코딩된 유닛으로부터 적어도 하나의 프레임 부분을 추출하고 - 프레임 부분은 프레임 부분 식별자를 포함함 -, 공간적 정보에 기반하여 프레임 내의 프레임 부분의 위치를 결정하며, 결정된 위치에 따라 프레임 부분을 프레임으로 렌더링하도록 프레임 부분을 디코딩하도록 구성된 회로를 포함한다.

[0059] 본 발명의 제6 양태에 따르면, 프레임 부분들로 공간적으로 분할되는 프레임들을 포함하는 비디오 데이터를 포함하는 새로운 비트스트림을 생성하기 위한 디바이스가 제공되며, 이 디바이스는, 복수의 비트스트림으로부터 추출되고 새로운 비트스트림으로 병합될 복수의 프레임 부분을 결정하고 - 복수의 비트스트림은 제1항 내지 제25항 중 어느 한 항에 따라 인코딩됨 -, 추출될 프레임 부분들의 프레임 부분 식별자들을 결정하고, 새로운 비트스트림에 대한 프레임 부분 배열 정보를 생성하고, 복수의 비트스트림으로부터 추출될 복수의 프레임 부분을 추출하며, 복수의 프레임 부분 및 생성된 프레임 부분 배열 정보를 새로운 비트스트림에 임베딩하기 위한 회로를 포함한다.

[0060] 본 발명의 제7 양태에 따르면, 프로그래밍가능한 장치를 위한 컴퓨터 프로그램 제품이 제공되며, 이 컴퓨터 프로그램 제품은 프로그래밍가능한 장치에 로딩되고 이에 의해 실행될 때 본 발명에 따른 방법을 구현하기 위한 명령어들의 시퀀스를 포함한다.

[0061] 본 발명의 제8 양태에 따르면, 본 발명에 따른 방법을 구현하기 위한 컴퓨터 프로그램의 명령어들을 저장하는 컴퓨터 판독가능한 저장 매체가 제공된다.

[0062] 본 발명의 제9 양태에 따르면, 실행 시에 본 발명에 따른 방법이 수행되게 하는 컴퓨터 프로그램이 제공된다.

[0063] 본 발명에 따른 방법들의 적어도 부분들은 컴퓨터에 의해 구현될 수 있다. 따라서, 본 발명은 완전 하드웨어 실시예, 완전 소프트웨어 실시예(펌웨어, 상주 소프트웨어, 마이크로-코드 등을 포함함) 또는 일반적으로 모두가 본 명세서에서 "회로", "모듈" 또는 "시스템"으로 지칭될 수 있는 소프트웨어 및 하드웨어 양태들을 조합하는 실시예의 형태를 취할 수 있다. 또한, 본 발명은, 매체에 구현된 컴퓨터 이용가능한 프로그램 코드를 갖는 임의의 유형적 표현 매체로 구현된 컴퓨터 프로그램 제품의 형태를 취할 수 있다.

[0064] 본 발명이 소프트웨어로 구현될 수 있으므로, 본 발명은 임의의 적절한 운반 매체 상의 프로그래밍가능한 장치에의 제공을 위한 컴퓨터 판독가능한 코드로 구현될 수 있다. 유형적 비일시적인 운반 매체는, 플로피 디스크, CD-ROM, 하드 디스크 드라이브, 자기 테이프 디바이스 또는 솔리드 스테이트 메모리 디바이스 등의 저장 매체를 포함할 수 있다. 일시적인 운반 매체는 전기 신호, 전자 신호, 광학 신호, 음향 신호, 자기 신호 또는 전자기 신호, 예를 들어 마이크로파 또는 RF 신호와 같은 신호를 포함할 수 있다.

도면의 간단한 설명

[0065] 이제, 본 발명의 실시예들은 단지 예로서 다음의 도면들을 참조하여 설명될 것이다.

도 1은 본 발명이 통합될 수 있는 시스템을 도시한다.

도 2는 블록 기반 비디오 인코더, 예를 들어 HEVC의 이미지 코딩 구조를 도시한다.

도 3은 HEVC에서 슬라이스 세그먼트들 및 타일들이라 불리는 두 종류의 파티션들에 따른 이미지의 분할을 도시한다.

도 4는 이미지의 경계들과 교차하는 CTU들에 대해 HEVC에서 이용되는 쿼드-트리 추론 메커니즘을 도시한다.

도 5는 예를 들어 HEVC에서 이용되는 경계 확장 메커니즘을 도시한다.

도 6은 HEVC 비트스트림 구성의 예를 도시한다.

도 7은 관심 영역(ROI) 스트리밍을 위한 HEVC 파티셔닝의 예를 도시한다.

도 8a 및 도 8b는 관심 영역들의 조합에 대한 2개의 상이한 이용 시나리오 예를 도시한다.

도 9는 본 발명이 통합되는 비디오 인코더의 전형적인 인코딩 프로세스를 도시한다.

도 10은 본 발명이 통합되는 비디오 디코더의 전형적인 디코딩 프로세스를 도시한다.

도 11은 본 발명의 예시적인 이용을 설명한다.

도 12는 캡슐화 단계들에 대한 상세들을 제공한다.

도 13a, 도 13b 및 도 13c는 인코딩 프로세스에 의해 수행되는 프레임 부분 배열의 시그널링을 도시한다.

도 14는 비-그리드 기반 파티셔닝의 예를 도시한다.

도 15는 CTile 식별자를 시그널링하는 대안적인 실시예들을 도시한다.

도 16a는 CTile마다의 의존성 리스트를 포함하는 XPS를 도시한다.

도 16b는 CTile 의존성들의 제1 예를 도시한다.

도 16c는 CTile 의존성들의 제2 예를 도시한다.

도 17(17a 및 17b)은 CTile가 연속적으로 인코딩된 프레임들 사이에서 위치 또는 크기를 변경할 수 있는 실시예들의 예를 제공한다.

도 18은 본 발명의 하나 이상의 실시예의 구현을 위한 컴퓨팅 디바이스의 개략적인 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0066] 비디오 시퀀스의 프레임들을 공간적 프레임 부분들로 인코딩하는 것은 특히, 예를 들어 소위 360도 비디오들의 스트리밍에 관련된 시나리오들에서 유용하며, 이는 사실상 360도 파노라마 비디오 또는 구형 비디오의 전형적인 2D 비디오 표현 상으로의 투영의 결과이다.

[0067] 360도 비디오들(또는 단지 360 비디오)은 양호한 사용자 경험을 제공하기 위해 매우 높은 해상도를 가질 수 있

는 비디오들이다. 헤드 장착형 디스플레이의 내부에(또는 스크린 상에) 표시될 때, 360 비디오 콘텐츠의 공간적 하위 부분만이 사용자에게 제시된다.

[0068] 따라서, 예를 들어, 사용자가 보고 있는 영역들에 대해서만 고품질을 갖는 공간적 프레임 부분들을 요청하기 위해, DASH(dynamic adaptive streaming over HTTP)와 같은 스트리밍 프로토콜들을 이용하는 것이 흥미롭다. 보이지 않는 영역들(즉, 사용자가 보고 있지 않은 영역들)의 경우, 공간적 프레임 부분들은 간단히 스킵될 수 있다.

[0069] 본 발명의 애플리케이션은 사용자의 보는 방향에 스트리밍을 적응시키는 스트리밍 접근법들을 지칭한다. 즉, 이것은 뷰포트 의존 스트리밍을 지칭한다. 이러한 접근법의 경우, 저장 비용, 계산 비용, 및 사용자 경험 사이의 하나의 양호한 절충은 시퀀스들을 다양한 품질들을 갖는 독립적인 공간적 프레임 부분들로 인코딩하는 것이다. 그 후, 프레임 부분들은, 요구들 및 대역폭 제약들에 따라, 랜덤하게 액세스되고, 추출되고/되거나 다른 프레임 부분 시퀀스들과 조합될 수 있다. 이것은 임의의 추가적인 인코딩 또는 트랜스코딩을 요청하지 않는다. 이러한 시나리오들의 예가 도 8a를 참조하여 설명된다.

[0070] 애플리케이션은 시스템 운영자로부터 요청된 구성과 일치시키기 위해 여러 상이한 비디오들의 공간적 프레임 부분들이 새로운 비디오에서 재구성되는 비디오 감시 시스템에 관한 것이다. 예를 들어, 운영자는 원래의 비디오의 부분들만을 원할 수 있다. 이 애플리케이션은 특히 도 8b에 도시된다.

[0071] 마지막으로, 다른 애플리케이션에서, 전체 비디오 시퀀스로부터 추출된 하나의 단일 공간적 프레임 부분만을 포함하는 새로운 "비디오"는 새로운 비디오 내의 공간적 프레임 부분의 새로운 위치가 상이한 경우 인코딩 파라미터들의 재기입을 수반할 수 있다.

[0072] HEVC를 이용할 때, 공간적 프레임 부분들의 인코딩은 HEVC 타일들에 기반한다. 그러나, HEVC 타일들 및 보다 일반적으로 HEVC형 타일들은 전술한 애플리케이션들을 다루도록 설계되지 않는다.

[0073] 도 1은 본 발명이 통합될 수 있는 시스템(예를 들어, 상호작용적 스트리밍 비디오 시스템)의 실시예를 설명한다.

[0074] 비디오 비트스트림은 서버, 또는 프록시 서버(100)로부터 네트워크(101)를 통해 클라이언트(102)로 전송된다. 서버(100)는 블록 기반 비디오 코덱, 예를 들어 HEVC 비디오 코덱의 사양들에 따르는 비디오 인코더(103)에 의해 생성된 비디오 스트림들(또는 비디오 파일들)을 이용한다.

[0075] 인코더는, 후술되는 바와 같이, 본 발명에 따라 일부 공간적 프레임 부분들에 대한 공간적 랜덤 액세스를 제공하면서, 상이한 레이트/왜곡 상충관계를 갖는 비디오 시퀀스들의 세트를 압축한다.

[0076] 서버(100)는 상호작용적 스트리밍을 위한 이용가능한 비디오 스트림들의 설명에 대한 요청을 통신 네트워크(101)를 통해 수신한다. 통신 네트워크(101)는 인터넷 프로토콜 표준에 기반한다. IP 네트워크(101)를 통해 미디어 프리젠테이션을 전송하는 데 이용되는 표준 프로토콜은 바람직하게는 MPEG DASH(Dynamic Adaptive Streaming over HTTP)이다. 그러나, 본 발명은 또한 임의의 다른 스트리밍 프로토콜들에 이용될 수 있다.

[0077] 도 2는 두 종류의 파티션: 슬라이스 세그먼트들 및 공간적 프레임 부분들에 따른 이미지의 분할을 도시한다. 이미지(206)는 3개의 슬라이스 세그먼트로 분할된다. 슬라이스 세그먼트는 이미지 또는 전체 이미지의 일부이다. 각각의 슬라이스 세그먼트는 (HEVC의 코딩 유닛들에 대응할 수 있는) 정수개의 코딩 블록들을 포함한다. 코딩 블록들은 샘플들로 이루어진다.

[0078] 두 종류의 슬라이스 세그먼트: 독립적인 슬라이스 세그먼트들(207) 및 의존적인 슬라이스 세그먼트들(208)이 있다. 각각의 슬라이스 세그먼트는 패킷 지향 및 비트스트림 지향 전송 시스템들 모두에서 이용하기 위한 일반 포맷을 갖는 구조인 하나의 NAL 유닛에 임베딩된다. 두 가지 유형의 슬라이스 세그먼트들 간의 차이는 독립적인 슬라이스 세그먼트 헤더에 지정된 데이터가 슬라이스 세그먼트의 인코딩된 블록들을 디코딩하는데 필요한 모든 파라미터들을 정의한다는 사실에 있다. 한편, 의존적인 슬라이스 세그먼트는 감소된 헤더를 가지며, 첫 번째 선행하는 독립적인 슬라이스 세그먼트에 의존하여, 그 헤더에서 이용가능하지 않은 파라미터들을 추론한다. 슬라이스 내의 첫 번째 코딩 유닛의 어드레스는 독립적인 슬라이스 세그먼트 헤더에서 지정된다.

[0079] 도 3은 공간적 프레임 부분들(SPF)로의 다른 파티셔닝이 프레임(305)에 도시된 바와 같이 독립적으로 코딩된 직사각형 영역들에서 각각의 프레임을 분할하게 하는 것을 도시한다.

[0080] HEVC형 타일들과 마찬가지로, 공간적 프레임 부분은 정수개의 코딩 블록들을 포함한다. 슬라이스 경계들과 유

사하게, SPF 경계들(310)은 모든 인트라 예측 메커니즘들을 깨뜨린다.

[0081] HEVC형 타일들과 마찬가지로, SPF는 디코딩 프로세스를 초기화하는 데 이용되는, 특정 NAL 유닛에 포함된 픽처 파라미터 세트에서 정의된다. PPS NAL 유닛은 픽처 내의 타일 행들의 수 및 타일 열들의 수 및 그 연관된 크기들을 지정할 수 있는 신택스 요소들을 포함한다. 다른 파라미터 세트 NAL 유닛들(예컨대, 비디오 파라미터 세트 또는 VPS, 시퀀스 파라미터 세트들 또는 SPS)은 비트스트림의 인코딩 구성을 기술하는 파라미터들을 전달한다. 본 발명에서, 이것은 XPS(X는 와일드카드 문자로서 이용됨)로서 이들 파라미터 세트 중 임의의 것으로 지칭된다. 하나의 슬라이스 세그먼트 내의 SPF 위치들, 예를 들어 비트들에서의 오프셋은 슬라이스 세그먼트 헤더의 끝에서 이용가능한 신택스 요소들로 식별된다.

[0082] SPF 및 슬라이스 세그먼트들은 공동으로 이용될 수 있지만 일부 제약들을 갖는다. 다음의 조건들 중 하나 또는 둘 다가 검증되어야 한다:

[0083] - 하나의 슬라이스(또는 슬라이스 세그먼트)의 모든 코딩 블록들이 동일한 SPF에 속하거나,

[0084] - 하나의 SPF의 모든 코딩 블록들이 동일한 슬라이스(또는 슬라이스 세그먼트)에 속한다.

[0085] 이것은 하나의 슬라이스(또는 슬라이스 세그먼트)가 몇 개의 전체 SPF들을 포함할 수 있거나 단지 단일 타일의 하위 부분일 수 있다는 것을 의미한다. 둘째, SPF는 몇 개의 전체 슬라이스들(또는 슬라이스 세그먼트들)을 포함할 수 있거나 단지 단일 슬라이스(또는 슬라이스 세그먼트)의 하위 부분일 수 있다.

[0086] 도 4는 단지 예시 목적을 위해, 이미지의 경계들과 교차하는 코딩 유닛들에 대해 HEVC에서 이용되는 쿼드-트리 추론 메커니즘을 개략적으로 도시한다. HEVC에서, 이미지들은 코딩 유닛 크기의 폭 및 높이 배수를 갖도록 제약되지 않는다. 그 후, 프레임의 최우측 코딩 유닛들은 이미지의 우측 경계(401)와 교차할 수 있고, 프레임의 최하단 코딩 유닛들은 이미지의 하단 경계(402)와 교차할 수 있다. 이들 경우들에서, HEVC는 경계들과 교차하는 코딩 유닛들에 대한 쿼드-트리 추론 메커니즘을 정의한다. 이 메커니즘은, 경계와 교차하는 더 이상의 CU가 없거나, 또는 이들 코딩 유닛들에 대해 최대 쿼드-트리 깊이에 도달될 때까지, 이미지 경계와 교차하는 코딩 유닛의 임의의 CU를 재귀적으로 분할하는 것으로 구성된다. 예를 들어, 코딩 유닛(403)은 자동으로 분할되지 않는 반면, 코딩 유닛들(404, 405 및 406)은 자동으로 분할된다. 추론된 쿼드-트리의 시그널링이 없다: 디코더는 이미지 경계들에 대해 동일한 쿼드-트리를 추론해야 한다. 그러나, 자동으로 획득된 쿼드-트리는, 예를 들어, 407에서와 같이, (최대 쿼드-트리 깊이에 도달되지 않은 경우) 그 코딩 유닛들에 대한 분할 정보를 시그널링함으로써 프레임 내부에 있는 코딩 유닛들에 대해 더 정밀해질 수 있다.

[0087] 도 6은 서버로부터 클라이언트로 전송된 전형적인 비디오 비트스트림(600)을 도시한다. 비트스트림은 HEVC 또는 블록 기반 비트스트림에 따른다.

[0088] 비트스트림(600)은 네트워크 추상 계층(NAL) 유닛들의 연속으로서 구성된다. 여러 종류들(유형들)의 NAL 유닛들이 있다. 파라미터 세트 NAL 유닛들(예컨대, HEVC에 대한 VPS, SPS 및 PPS)은 시퀀스들을 인코딩하는 데 이용되는 코딩 도구들의 구성을 설명한다. 이들은 또한 이미지들의 특성들(해상도, 프레임 레이트 등)에 대한 일부 정보를 설명한다.

[0089] 제1 NAL 유닛(601)은 전체 비트스트림에 대한 정보를 제공하는 비디오 파라미터 세트(VPS)이다. 특히, 이는 비트스트림 내의 규모조정성 계층들의 수를 나타낸다.

[0090] 후속하는 NAL 유닛(602)은 시퀀스 파라미터 세트(SPS)이다. 이것은 시퀀스 레벨 파라미터들을 제공한다. 픽처 레벨 파라미터들을 제공하는 픽처 파라미터 세트(PPS) NAL 유닛(603)이 이어진다. 이어서, 슬라이스 세그먼트들(604)이 제공될 수 있다. 프레임마다 하나의 슬라이스 세그먼트를 갖는 것이 일반적이다. 슬라이스 세그먼트들(604)은 다양한 NAL 유닛 유형들(CRA, IDR, BLA, RASL, RADL, STSA, TSA 또는 TRAIL ...)을 갖는 NAL 유닛들에 포함될 수 있다. 슬라이스 세그먼트를 포함하는 NAL 유닛은 NAL 헤더(605)로 구성되고(NAL 헤더의 추가 설명은 도 10의 설명과 함께 제공될 것임), 원시 바이트 시퀀스 페이로드(RBSP)(606)로 구성된다. NAL 헤더(605)는 NAL 유닛 유형을 포함하는 정보를 포함한다. RBSP(즉, NAL 유닛 데이터)는 NAL 유닛 유형에 특정한 정보를 포함한다. 슬라이스 세그먼트들의 경우, RBSP는 슬라이스 세그먼트 데이터(608)가 뒤따르는 슬라이스 세그먼트 헤더(607)를 포함한다. 슬라이스 세그먼트 데이터는 슬라이스 세그먼트의 래스터 스캔 순서화된 코딩 트리 유닛들(609)의 인코딩된 데이터의 연속이다.

[0091] 실시예(여기에 도시되지 않음)에서, (타일링 파라미터 세트에 대한) TPS라고 불리는 파라미터 세트가 슬라이스 세그먼트 NAL 유닛들 이전에 비트스트림에 삽입될 수 있다. 대응하는 파라미터들은 새로운 TPS가 발견될 때까

지 유효하다. TPS는 프레임들의 파티셔닝 구성을 설명한다.

[0092] 다른 실시예에서, 비트스트림에 TPS가 존재하지 않으면, 비트스트림에 하나의 공간적 프레임 부분만이 존재하는 것으로 가정된다. 공간적 프레임 부분은 비디오 프레임들과 동일한 치수들을 가지며, 그 원점에 위치된다.

[0093] 비트스트림은 독립적인 프레임 부분들 또는 관심 영역들(ROI)을 포함할 수 있다. 도 7은 프레임 내의 직사각형 영역들로서 여기서 고려되는 관심 영역들의 예를 개략적으로 도시한다. ROI들은 HEVC 및 블록 기반 코덱들에서 잘 알려져 있다.

[0094] 스트리밍 ROI들 또는 독립적인 프레임 부분들은 파티셔닝 전략을 암시한다. 타일 경계들의 도입은 일부 HEVC 예측 메커니즘들을 깨뜨리기 때문에 코딩 효율에 영향을 미친다.

[0095] 도 7에서, 프레임(700)은 4x4 SPF 그리드로 분할된다. 미리 정의된 ROI(701)에 액세스하기 위해, SPF 6, 7, 10 및 11에 대한 대응하는 슬라이스 세그먼트들을 임베딩하는 NAL 유닛들이 선택되어 클라이언트에 전송된다.

[0096] 바람직하게는, 본 발명에서, 하나의 독립적인 슬라이스 세그먼트 및 0개 이상의 의존적인 슬라이스 세그먼트가 SPF에 임베딩된다. 이점은 ROI를 포함하는 프레임의 다른 부분들에 독립적으로 이 ROI에 대한 액세스를 보증하는 것이다.

[0097] 실제로, HEVC 및 더 일반적으로 블록 기반 코덱들의 경우, HEVC 타일들 등은 그 경계들에서 모든 인트라-프레임 예측 메커니즘들(루프 필터링 처리를 제외함)을 깨뜨린다는 것을 상기한다. 따라서, 공간적 예측 메커니즘이 허용되지 않는다. 그러나, 여러 예측 메커니즘들은 압축을 개선하기 위해 비디오 시퀀스의 프레임들 사이의 데이터의 시간적 중복성에 의존한다. 예를 들어, HEVC 타일에서의 하나의 블록은 부분적으로 또는 전체적으로 현재 HEVC 타일 경계들 밖에 있는 예측자 블록으로부터 예측될 수 있다. 더욱이, 예측자 블록은 또한 부분적으로 또는 전체적으로 프레임 경계들 밖에 있을 수 있는데, 그 이유는 HEVC가 이미지의 경계를 확장하기 위한 잘 알려진 경계 확장 메커니즘을 제공하여 예측자 블록이 부분적으로 또는 전체적으로 참조 이미지 밖에 있을 수 있게 하기 때문이다.

[0098] 마지막으로, 예측자 블록은 서브-픽셀 위치에 배치될 수 있다. 이것은 참조 블록 픽셀 값이 서브-픽셀 보간 필터의 결과라는 것을 의미하며, 이는 예측자 블록에 대응하는 전체 픽셀 좌표에 위치한 픽셀의 블록 밖의 최대 4개의 픽셀의 범위로부터 서브-픽셀 값을 생성한다. 그 결과, 시간적 예측은 HEVC 타일 경계들 밖에 위치한 픽셀 데이터의 세트와 HEVC 타일 내의 블록들 간의 코딩 의존성을 도입할 수 있다.

[0099] 시간적 예측에 수반되는 제2 HEVC 메커니즘은 움직임 벡터 예측자를 이용하는 움직임 벡터의 예측 인코딩으로 구성된다.

[0100] 마지막으로, HEVC는 인접 타일들의 픽셀들 간의 의존성들을 도입하는 루프 필터들의 세트를 제공한다. 이 루프 필터들은 특히 잔차 블록들의 양자화에 의해 도입되는 일부 아티팩트들을 제거하는 디블로킹 필터들 및 SAO 필터이다. HEVC는 이들 루프 필터들이 타일 또는/및 슬라이스 경계들에서 디스에이블되는지를 나타내는 플래그들을 픽처 파라미터 세트들에서 제공한다. 디스에이블될 때, 타일들 간의 코딩 의존성은 이러한 압축 도구들에 의해 도입되지 않는다.

[0101] 관심 영역을 디코딩하는 것(관심 영역을 독립적으로 디코딩하는 것을 의미함)을 보증하기 위한 솔루션은 이전에 언급된 예측 메커니즘들의 일부 또는 전부를 디스에이블하는 것이다.

[0102] 이것은 결과적인 비트스트림에 대해 덜 효율적인 압축 및 더 높은 비트레이트로 이어진다. 결과적인 비트스트림의 비트레이트를 최적화하기 위해 관심 영역 이용 시나리오에 따라 예측 메커니즘들의 활성화/비활성화를 적응시키는 것이 가능하다.

[0103] 도 8a 및 도 8b는 이미 전술한 관심 영역들의 조합에 대한 2개의 상이한 애플리케이션 예를 도시한다.

[0104] 예를 들어, 제1 예에서, 도 8a는 4개의 관심 영역으로 구성된 2개의 상이한 비디오 스트림으로부터의 2개의 프레임(800 및 801)을 나타낸다. 제1 비디오 스트림(800)은 고품질의 인코딩 파라미터들을 가지며, 제2 비디오 스트림(801)은 저품질 및 이에 따른 낮은 비트레이트 버전이다. 클라이언트는 관심 영역 #3의 고품질 버전을 영역들 1, 2 및 4에 대한 저품질 관심 영역들과 효율적으로 조합한다. 이것은 다른 덜 중요한 영역들에 대해 비트레이트를 비교적 낮게 유지하면서 관심 영역 #3의 품질을 강조하는 것을 허용한다.

[0105] 제2 예에서, 4개의 비디오 스트림(803, 804, 805 및 806)의 세트가 도 8b에 표현되어 있다. 이 이용 시나리오에서, 클라이언트는 각각의 비디오 스트림의 상이한 관심 영역들의 새로운 모자이크 비디오를 형성한다. 이것

은 각각의 비디오 스트림의 관심 영역을 결과적인 비디오 스트림 내의 새로운 위치로 재배열하거나 조합한다.

[0106] 본 발명의 실시예에 따르면, 여기서 제약된 타일(이하의 설명에서 CTile로 단축됨)이라고 불리는 공간적 프레임 부분을 정의하는 것이 제안된다. 이것은 랜덤하게 액세스되고 디코딩 에러 없이 완전히 디코딩될 수 있는 공간적 프레임 부분들로 분할되는 프레임들의 시퀀스에 속하는 공간적 프레임 부분을 지칭한다. CTile의 디코딩은 그 공간적 위치 및/또는 그 이웃과는 독립적으로 수행될 수 있다. 다시 말해서, CTile는 디코더가 임의의 에러 없이 이를 항상 디코딩할 수 있는 방식으로 독립적으로 인코딩 또는 인코딩된다.

[0107] CTile를 형성하는 샘플들에 대응하는 인코딩된 데이터는 독립적으로 인코딩된다. 예를 들어, 하나의 파서가 CTile에 대응하는 샘플들을 추출할 수 있도록, 데이터는 슬라이스들(또는 유사한 특징들을 갖는 프레임의 임의의 다른 부분들)을 형성하는 인코딩된 유닛들 또는 NAL 유닛들로 인코딩된다. 그 결과, 2개의 CTile는 인코딩된 유닛들의 2개의 상이한 세트로 인코딩된다. 임의의 공간적 위치에서 디코딩되기 위해, CTile는 다른 CTile들을 포함하는 슬라이스의 일부가 될 수 없다. 따라서, CTile에 대응하는 슬라이스의 인코딩된 데이터는 다른 슬라이스들로부터의 인코딩된 데이터와는 독립적이다.

[0108] 슬라이스에 대응하는 인코딩된 데이터는 몇 개의 슬라이스 세그먼트 인코딩된 유닛들로 추가로 분할될 수 있다.

[0109] 실시예에서, CTile들은 엄격하게 독립적으로 디코딩가능하며, 이는 CTile들을 형성하는 인코딩된 데이터를 파싱하는데 필요한 모든 데이터가 CTile에 포함된다는 것을 의미한다. 또한, 예측 메커니즘들은 동일한 CTile의 인코딩된 데이터로부터 계산된 예측 정보를 이용한다. 인터 예측의 경우, 참조 블록들은 다른 프레임에서의 동일한 CTile로부터 검색된다.

[0110] 다른 실시예들에서, 인코딩 제약들이 해제될 수 있다.

[0111] 첫 번째의 다른 실시예에서, 경계 확장 메커니즘은 이미지 경계들에 이용된다. 도 5에 개략적으로 도시된 이러한 메커니즘은 비제약적이고 더 효율적인 움직임 보상을 허용하기 위해 CTile 경계들에 이용된다.

[0112] 도 5는 예를 들어 HEVC에서 이용되는 경계 확장 메커니즘을 간략화된 방식으로 도시한다. 이 메커니즘은 프레임 밖의 샘플 값들을 참조하여 인터 예측(현재 프레임 밖의 데이터의 이용을 허용하는 잘 알려진 예측 모드)에서의 움직임 보상을 허용한다.

[0113] 프레임(502)을 인코딩하는 동안 블록(501)을 예측할 때, 참조 프레임(504)의 경계와 교차하는 블록(503)으로부터의 예측을 허용하는 것이 유용하다. 이것은, 예를 들어, 이전 프레임에서 부분적으로 시야 밖에 있었던 동일한 콘텐츠로부터 이동 콘텐츠를 예측하는 것을 허용한다. 바람직하게, 샘플 패딩 방법은 참조 픽처의 프레임 경계들의 모든 주위의 마진 내의 샘플들에 액세스할 수 있게 하도록 정의된다.

[0114] 두 번째의 다른 실시예에서, 움직임 벡터 예측자들(또는 임의의 다른 예측자)의 도출 메커니즘들은 임의의 이웃 타일 정보 또는 타일링 구성에 전혀 의존하지 않는 방식으로 CTile들에 대해 허가될 수 있다.

[0115] 도 9는 본 발명에 따른 비디오 인코더에서 구현된 인코딩 프로세스의 예를 도시한다.

[0116] 먼저, 각각의 고려된 입력 비디오 시퀀스(900)에 대해, 인코더는 단계(901)에서 프레임 부분들로의 프레임들의 파티셔닝(즉, 프레임 부분 배열)을 결정한다. 일부 실시예들에서, 프레임 부분의 크기는 하나의 프레임 부분이 단일 관심 영역 또는 관심 영역의 부분을 커버하도록 미리 결정된다. 예를 들어, 프레임 부분은 512x512 픽셀들의 크기를 가질 수 있다.

[0117] 그 다음, 인코더는 단계(902)에서 어느 프레임 부분들이 CTile들로서 인코딩되어야 하는지를 결정한다. 예를 들어, 이러한 프레임 부분은 클라이언트가 단독으로 디코딩하기를 원할 수 있거나, 또는 클라이언트가 하나 이상의 다른 관심 영역과 구성하기를 원할 수 있는 관심 영역(ROI)에 대응할 수 있다.

[0118] 이어서, 인코더는 단계(903)에서 식별자들을 결정하여 이들을 모든 CTile들에 할당한다. 변형예에서, 인코더는 식별자들을 결정하여 이들을 CTile들의 선택에만 할당할 수 있다. 변형예에서, CTile들의 식별자들이 추론될 수 있다.

[0119] 동일한 식별자들이 몇 개의 인코딩된 프레임들에의 CTile들에 할당될 때, 이는 이들 CTile들이 동일한 CTile 시퀀스에 속한다는 것을 의미한다. CTile들의 시퀀스(또는 CTile 시퀀스)는 다른 프레임 부분들과 독립적으로 디코딩될 수 있다. CTile 시퀀스로부터의 데이터만이 CTile 시퀀스를 디코딩하는데 필요하다. 다시 말해서, CTile 시퀀스로부터의 CTile들은 시간적 의존성들을 함께 가질 수 있다.

[0120] CTile들 및 CTile 식별자들을 결정한 후에, 인코더는 인코딩 구성에 따라 프레임 부분들을 압축 및 인코딩(904)한다. 프레임 부분들의 인코딩은 임의의 디코더가 이전에 설명된 바와 같이 이들을 디코딩할 수 있는 것을 보장한다.

[0121] 인코더는 단계(905)에서 프레임 부분 배열 정보를 생성한다. 프레임 부분 배열 정보는 프레임 부분들로의 프레임 파티셔닝의 설명 파라미터들을 결정하는 것으로 구성된다. 이것은 또한 CTile 식별자들을 프레임 내의 또는 프레임들의 시퀀스 내의 그 위치에 연관시킴으로써 CTile들의 설명 파라미터들을 결정하는 것으로 구성된다. 도 13a, 도 13b 및 도 13c(제1 대안) 또는 도 15(제2 대안)를 참조하여 설명되는 상이한 시그널링 대안들이 제안된다. 단계(905)는 파라미터 세트들(XPS)에서 시그널링 타일링 파라미터들을 생성하는 것을 포함한다. 변형예에서, 단계(905)는 인코딩 단계(904) 이후가 아니라 그 이전에 수행된다.

[0122] 단계(906)는 비트스트림으로의 XPS의 NAL 유닛 및 압축된 CTile 프레임 부분들의 임의적 캡슐화를 포함한다.

[0123] 예를 들어, 스트리밍 프로토콜에 기반하여, 이 단계는 예를 들어 ISO BMF 파일 포맷과 같은 상위 레벨 비디오 설명 포맷 내부의 비트스트림의 캡슐화를 추가로 포함할 수 있다. 이것은 또한 예를 들어 오디오 데이터와 비디오 데이터를 다중화하는 것을 허용할 수 있다.

[0124] 단계들(901, 902 및 903)은 미리 결정된 프레임 부분들의 위치들을 제공하고, 프레임 부분들이 CTile들인지 그리고 어느 식별자가 CTile들에 이용되어야 하는지에 관한 정보를 제공하는 하나 이상의 구성 파일을 이용하여 구현될 수 있다. 대안적인 실시예에서, 프레임 부분들 및 CTile들은 예를 들어 심층 신경망 또는 더 간단한 세그먼트화 알고리즘을 이용하여 비디오 콘텐츠의 분석으로부터 자동으로 결정될 수 있다.

[0125] 일부 실시예들에서 설명된 바와 같이, 단계(901)는 전체 비디오 시퀀스 내에서(또는 비디오 세그먼트 내의 적어도 몇 개의 연속적인 프레임들에 대해) 일정한 파티셔닝을 결정하는 데 이용될 수 있다. 이것은 CTile들의 위치들 및 크기들이 CTile 시퀀스의 일부를 포함하는 적어도 몇 개의 프레임들 내의 전체 CTile 시퀀스 내에서 일정하다는 것을 의미한다.

[0126] 대안적으로, 결정된 프레임 부분들은 프레임들 사이에서 가변 크기들 및 위치들을 가질 수 있다.

[0127] 도 10은 비디오 디코더에서 구현된 디코딩 프로세스의 예를 도시한다. 디코딩 프로세스는 위에서 정의된 바와 같은 CTile들의 이용을 수반한다.

[0128] 먼저, 비디오 디코더는 파라미터 세트들(XPS)을 포함하는 NAL 유닛들을 추출한다. 프레임 부분 배열 정보는 단계(1000)에서 파라미터 세트들로부터 획득된다.

[0129] 각각의 고려된 프레임 부분(1001)에 대해, 디코더는 단계(1002)에서 프레임 부분 배열 정보로부터 프레임 부분이 CTile인지를 결정한다.

[0130] 프레임 부분이 CTile로서 시그널링되는 경우, 테스트(1003) 후의 분지 '예'에서, 디코더는 프레임 부분 배열 정보로부터 CTile 식별자를 추출(또는 추론)하고, 단계(1005)에서 CTile 식별자 및 그 식별자와 연관되는 CTile 위치 정보에 의해, CTile의 디코딩 위치를 결정한다. 그렇지 않고, 프레임 부분이 CTile가 아닌 경우, 테스트(1003) 후의 분지 '아니오'에서, 디코더는 단계(1005)에서 프레임 부분 인코딩된 데이터에서 설명되는 포지셔닝 정보 및 XPS 정보로부터 프레임 부분의 디코딩 위치를 결정한다.

[0131] 마지막으로, 디코더는 프레임 부분이 CTile인지 여부를 고려하여 프레임 부분 인코딩된 데이터를 디코딩하고(1006), 렌더링 픽처 버퍼 내부에 디코딩된 샘플 값들을 넣는다.

[0132] 도 11은 도 9의 인코딩 프로세스로 생성된 2개의 비트스트림의 병합 프로세스(도 8a 및 도 8b에서의 애플리케이션들 참조)의 예를 설명한다. 병합 프로세스는 추출된 CTile들이 클라이언트에 전송될 새로운 비디오 비트스트림으로 조합된다는 것을 의미한다.

[0133] 병합 프로세스는 단계(1100)에서 하나 이상의 비디오 비트스트림으로부터 추출되고 새로운 비트스트림으로 병합될 CTile들의 세트를 결정하는 것으로 시작한다. 예를 들어, 그래픽 사용자 인터페이스는 사용자가 CTile들의 세트를 선택하고 또한 이들을 프레임에서 재배열하는 것을 허용한다. 다른 예에서, 이 선택은 비트스트림의 콘텐츠에 기반하여 자동으로 수행된다. 이 애플리케이션은 이동 콘텐츠를 포함하는 CTile들의 세트를 선택할 수 있다.

[0134] 프로세스는 단계(1101)에서 새로운 비디오 비트스트림에 병합될 때 CTile들의 새로운 위치를 결정한다.

[0135] 추출될 CTile들이 알려져 있다면, 추출될 결정된 CTile들 각각의 현재 CTile 식별자들을 획득함으로써 단계(1102)에서 그 새로운 식별자들이 결정된다. 이들 식별자들은 본 발명의 실시예들에 따라 프레임 배열 정보에서 시그널링된다. 전술한 바와 같이, 대안적인 실시예에서, 프레임 배열 정보는 입력 비트스트림들을 캡슐화하는 데 이용되는 파일 포맷으로 설명될 수 있다. 프레임 배열 정보는 XPS 및 파일 포맷으로 존재할 수 있다.

[0136] 하나보다 많은 CTile가 동일한 식별자를 갖는다는 것을 의미하는 식별자 충돌의 경우, 단계(1101)는 이러한 충돌들을 해결하기 위해 새로운 CTile 식별자들을 결정하는 것을 추가로 포함한다.

[0137] 그 후, 프로세스는 새로운 비디오 비트스트림의 병합된 비디오 시퀀스에 대한 프레임 부분 배열 정보를 생성한다(1103). 이것은 병합된 비트스트림 내의 CTile들의 새로운 위치를 그 새로운 CTile 식별자들과 연관시키는 XPS 중 하나에서 파라미터들을 생성하는 것을 포함한다.

[0138] 단계(1104)에서, 단계(1100)에서 결정된 CTile들의 세트의 인코딩된 프레임 부분 데이터가 추출되거나 획득된다. 이것은 CTile들의 인코딩된 프레임 부분 데이터를 포함하는 NAL 유닛들을 검색하는 것을 포함한다. 이것은 단계(1102)에서 결정된 CTile 식별자들을 갖는 것들을 추출하기 위해 입력 비트스트림들에서 모든 NAL 유닛들을 파싱함으로써 행해질 수 있다. 입력 비트스트림들이 파일 포맷 사양에 따를 때, 하나의 프레임 부분에 대응하는 모든 NAL 유닛들은 하나의 컨테이너, 예컨대 ISOBMFF를 위한 비디오 트랙 내에 캡슐화된다. 다음으로, 단계(1104)는 선택된 프레임 부분들의 트랙에 대응하는 데이터를 검색하는 것을 포함한다.

[0139] 마지막으로, 임의적 단계(1105)에서, XPS의 NAL 유닛들, 및 추출된 CTile 인코딩된 프레임 부분 데이터를 포함하는 NAL 유닛들을 새로운 비트스트림에 임베딩하고, 가능하게는 이 비트스트림을 더 높은 레벨의 설명 포맷으로 캡슐화함으로써 새로운 비트스트림이 생성된다.

[0140] CTile 식별자 충돌로 인해 새로운 CTile 식별자가 단계(1101)에서 결정된 CTile들의 경우, 단계(1105)는 원래의 CTile 식별자를 포함하는 NAL 유닛들에 포함된 헤더들을 수정하는 것을 더 포함한다. 이 헤더들은 원래의 CTile 식별자가 단계(1102)에서 결정된 CTile 식별자로 대체되도록 수정된다.

[0141] 일 예에서, 도 11의 병합 프로세스는 동일한 비트스트림으로부터 CTile들의 서브세트를 추출하는 것으로 구성된다. 이러한 경우, 식별자 충돌을 처리할 필요가 없다.

[0142] 도 13a, 도 13b 및 도 13c는 본 발명의 여러 실시예들에 따라 인코딩 프로세스에 의해 수행되는 프레임 부분 배열의 시그널링의 예들을 도시한다.

[0143] 도 13a는 본 발명의 실시예에 따른 비트스트림 내의 CTile들의 식별을 도시한다.

[0144] 여기서 ctile_unique_identifier(1301)이라고 명명된 CTile 식별자는 프레임 부분 인코딩된 데이터에 표시된다. 바람직하게는, 식별자는 프레임 부분 인코딩된 데이터에 속하는 각각의 데이터 시퀀스(즉, 슬라이스 세그먼트 헤더)에 표시된다. 따라서, 이는,

[0145] - 비트스트림의 어느 부분들이 CTile에 속하는지에 관한 용이한 식별, 및

[0146] - 이들 부분들의 신속한 액세스 또는 추출을 허용한다.

[0147] 보다 정확하게는, 도 13a에 도시된 실시예에서, CTile 식별자(1301)는 식별자(1301)를 갖는 CTile에 대응하는 각각의 슬라이스 세그먼트의 슬라이스 세그먼트 헤더(slice_segment_header)(1302)에서 시그널링된다.

[0148] 전술한 바와 같이, 디코더는 CTile 식별자를 파싱하여, 프레임 부분 배열 정보에 기반하여 CTile의 연관된 위치를 결정한다. 실시예들에 따르면, 프레임 부분 배열 정보는 도 13b 또는 도 13c를 참조하여 나중에 설명되는 바와 같이, 파라미터 세트, 예컨대 TPS에서 제공된다.

[0149] 이하에서 명시적으로 언급되거나 적용가능하지 않은 경우를 제외하면 간략함을 위해, CTile와 CTile 시퀀스 간의 구별이 없을 것이다. 또한, CTile 식별자는 CTile 시퀀스 식별자로도 볼 수 있다.

[0150] 실시예에서, 식별자가 반드시 요구되지는 않는 HEVC형 타일들을 CTile들과 구별하기 위해, 예를 들어, ctile_flag(1303)와 같은 정보가 (예를 들어, 슬라이스 세그먼트 헤더에서) 프레임 부분 인코딩된 데이터에 속하는 데이터 시퀀스에서 이용될 수 있다. ctile_flag가 비활성이면(예를 들어, '거짓'으로 설정되면), HEVC형 타일들에 대한 파라미터들(1304)이 제공된다. 그 파라미터들은 예를 들어, first_slice_in_pic_flag 또는 CTU 어드레스(slice_segment_adress)와 같은 타일 포지셔닝 정보, 또는 slice_pic_parameter_set_id와 같은 다른 비트스트림 요소들에 대한 참조들을 포함할 수 있다. 이러한 신택스 요소들은 프레임 파티셔닝에 의존하고, 비

디오 시퀀스마다 상이할 수 있다.

[0151] ctile_flag가 활성이면, 이들 파라미터들은 생략되고, CTile 고유 식별자(1301)를 포함하는 CTile 특정 정보가 대신 제공된다. CTile에서 복수의 슬라이스를 갖는 가능성을 허용하기 위한 하나의 솔루션은 여기서 명명된 정보(ctb_addr_offset_inside_tile(1305))를 제공하는 것이다. 이 정보 조각(1305)은 또한 고려된 프레임을 갖는 CTile 위치에 대한 슬라이스 세그먼트의 디코딩을 시작할 위치를 지정하는 데 이용된다. 예를 들어, 이 위치는 CTile의 시작과 (CTB에서의) 그 폭에 대해 상대적으로 코딩 블록들(예컨대, HEVC 표준 코딩 트리 블록인 CTB)의 원시 스캔 순서화된 수로 표현되므로, ctb_addr_offset_inside_tile 정보는 CTile 인코딩/디코딩 위치와 독립적이다.

[0152] 다른 실시예에서, 플래그 ctile_flag는 이용되지 않는다. 예를 들어, CTile 식별자는 모든 타일들, CTile들 및 다른 타일들(HEVC형 타일들)에 존재한다. 미리 결정된 값, 예를 들어 값 0은 HEVC형 타일들을 식별하는 데 이용될 수 있다.

[0153] 실시예에서, 공간적 프레임 부분이 CTile인지 여부를 식별하기 위한 정보가 제공된다.

[0154] 다른 실시예에서, 공간적 프레임 부분이 CTile인지 여부를 식별하기 위한 어떠한 정보도 제공되지 않으며, CTile만이 유일하게 이용된 프레임 부분들이라고 가정한다.

[0155] 바람직하게는, 주어진 프레임에서, 주어진 식별자를 갖는 하나 이하의 CTile가 존재한다. 동일한 CTile 식별자는 시간적으로 의존적인 모든 CTile들에서(예를 들어, CTile 시퀀스에서) 이용된다. 따라서, 연속적으로 인코딩된 픽처들에서 동일한 CTile 식별자를 갖는 CTile들이 추출되면, 이들은 적절히 디코딩될 것이다.

[0156] 다시 말해서, CTile 식별자는 인코딩된 비디오 시퀀스 내의 CTile를 식별하는 고유 식별자이다. 실시예에서, CTile 식별자는 CTile에 포함된 슬라이스 세그먼트들의 슬라이스 헤더들에 삽입된다. 이것은 비트스트림에서 CTile에 대응하는 NAL 유닛들(슬라이스 세그먼트들)이 CTile 식별자를 포함한다는 것을 의미한다. 따라서, 임의의 CTile는 이 CTile 식별자에 기반하여 비트스트림으로부터 쉽게 파싱되고 추출될 수 있다.

[0157] 비트스트림 내의 CTile 배열 정보를 시그널링하는 것이 유리하다. 예를 들어, CTile 배열 정보는 프레임 내의 CTile들의 수, 연관된 CTile 식별자들 및 CTile들의 위치에 의해 정의된다.

[0158] 도 13b는 본 발명의 실시예에 따른 CTile 배열을 도시한다.

[0159] 제1 실시예에서, 인코더는 디코딩될 픽처 내의 CTile들의 프레임 부분 배열 정보에 관련된 추가적인 시그널링 정보를 지정한다. 시그널링 정보는 파라미터 세트(XPS)에서, 바람직하게는 타일링 파라미터 세트(TPS)에서 제공된다. 바람직하게는, 추가 시그널링 정보는 여기서 num_ctiles로 불리는 픽처 내의 CTile들의 번호(1311)를 포함한다. 각각의 CTile에 대해, 이것은 CTile의 고유 식별자(1312)를, 여기서 tile_ctb_addr이라고 불리고, 픽처 내부의 디코딩 위치를 의미하는 CTile 위치(1313)와 연관시킨다. CTile 위치는 픽처 내의 디코딩 위치로서 제공된다. 이것은 (예를 들어 래스터 스캔 순서에 대해 상대적으로) CTB 인덱스 번호로 표현될 수 있다.

[0160] 다른 실시예에서, 1304에 의해 지정된 부분에서 도 13a에 나타낸 슬라이스 세그먼트 헤더들에서 또한 언급되는, slice_pic_parameter_set_id로 여기서 명명된 파라미터는 TPS를 나타내는 고유 식별자를 지칭한다. 변형예에서, 고유 식별자는 PPS를 나타낸다. 이 다른 실시예에서,

[0161] - 각각의 TPS는 TPS를 식별하는 tile_parameter_set_id(간략화를 위해 예시되지 않음) 파라미터를 포함하고, 예를 들어, 픽처에서 CTile 배열이 변경될 때마다, 인코더가 새로운 TPS를 생성할 수 있다는 것을 의미하고, 동일한 TPS(또는 PPS) 고유 식별자를 갖는 TPS를 생성하여 프레임의 각각의 슬라이스 헤더를 재기입하는 것을 피하도록 추천되고,

[0162] - 슬라이스 세그먼트 헤더의 slice_pic_parameter_set_id(1314)는 슬라이스에 적용되는 TPS의 tile_parameter_set_id와 동일하다. 이러한 경우, 슬라이스 세그먼트 헤더의 slice_pic_parameter_set_id의 명명은 slice_tile_parameter_set_id로 재명명될 수 있다.

[0163] 하나의 대안에서, TPS 식별자는 슬라이스 데이터에서 지정되지 않는다: 디코더는 슬라이스 NAL 유닛들에 선행하는 마지막 TPS NAL 유닛이 현재 CTile에 대한 프레임 부분 배열을 포함하는 것을 추론한다.

[0164] 도 13c는 본 발명의 다른 실시예에 따른 CTile 배열을 도시한다. 이 실시예에 따르면, TPS(1320)는 타일들의 수-1, 예를 들어 'num_tiles_minus1'(1321)을 나타내는 파라미터 값을 포함한다. 대안적으로, TPS는 프레임 내의 타일들의 수를 직접 또는 간접적으로 제공하는 예를 들어 'num_tiles'라는 명명된 파라미터 값을 포함한다.

[0165] 실시예에서, TPS가 하나의 프레임 부분만이 있다는 것을 나타내는 경우, 이것은 비디오 프레임들과 동일한 치수들을 갖고 그 원점에 위치하는 CTile인 것으로 가정된다. 그 밖의(TPS가 몇 개의 프레임 부분들을 설명함) 프레임 부분 위치들은 이전 실시예들에서와 같이 설명된다.

[0166] 다른 실시예에서, TPS가 없는 경우, 비디오 프레임들과 동일한 치수들을 갖는 하나의 CTile가 존재하는 것으로 가정된다. 이러한 하나의 CTile는 그 원점에 위치된다.

[0167] 다른 실시예에 따르면, TPS는 예를 들어, 'num_tile_rows_minus1', 'num_tile_cols_minus1' 및 'uniform_spacing_flag'를 지정하는 HEVC 그리드와 유사한 신택스를 갖는 공간적 프레임 부분 그리드를 설명할 수 있다. 'uniform_spacing_flag'가 설정되지 않으면, 각각의 열의 폭 및 각각의 행의 높이(추론될 수 있는 마지막 행 및 열 크기들을 제외함)가 또한 지정된다. 'uniform_spacing_flag'가 설정되면, 예를 들어 HEVC 사양에서와 같이, CTile 폭들 및 높이들이 픽처 폭 및 높이로부터 계산된다. 이러한 실시예들에서, CTile 위치는 (예를 들어, 타일들의 래스터 스캔 순서를 이용하여) 공간적 프레임 부분 그리드 인덱스에 대응하는 CTile 번호에 의해 표현될 수 있는데, 그 이유는 그리드 인덱스가 대응하는 CTile를 국소화할 수 있게 하기 때문이다.

[0168] 대안적인 실시예에 따르면, 'ctile_flag'는 여러 값들을 취할 수 있는 'ctile_level'로 대체되고, 각각의 값은 CTile에 적용되는 상이한 레벨의 인코딩 제약들을 나타낸다. 예를 들어, ctile_level이 0과 동일하다는 것은 CTile가 (HEVC형 타일들과 같이) 제약되지 않는다는 것을 나타낸다. ctile_level이 '1'과 동일하다는 것은, CTile가 추출되고 (그 원래의 이웃 없이) 단독으로 적절히 디코딩될 수 있거나 그 원래의 이웃과 함께 디코딩될 수 있지만, 다른 CTile들과 셔플링되는 경우 적절히 디코딩되지 않을 수 있도록 제약된다는 것을 나타낸다. ctile_level이 '2'와 동일하다는 것은, CTile가 임의의 곳에서 디코딩될 수 있고, 임의의 이웃(이전 실시예에서 1과 동일한 ctile_flag와 동등함)과 셔플링될 수 있도록 제약된다는 것을 나타낸다.

[0169] 다른 실시예에서, 'ctile_level'은 인코더가 제약 레벨을 만족시키기 위해 그 인코딩 결정들을 취한 정보를 단지 제공한다. 따라서 임의의 제약 레벨들을 갖는 CTile의 디코딩 프로세스는 HEVC형 타일들과 동일한 디코딩 프로세스에 의해 구현될 수 있다(예를 들어, CTile 경계들에 대해 경계 확장이 수행되지 않는다).

[0170] 다른 실시예에서, 인코딩 및 디코딩 프로세스들은 모든 제약 레벨에 대해 동일하지 않다. 예를 들어, ctile_level이 '1'과 동일한 CTile는 HEVC형 타일들(디코더에 영향을 주지 않고 인코더에서 이용되는 일부 제약들)과 동일한 디코딩 프로세스를 이용하는 반면, ctile_level이 '2'와 동일한 CTile는 CTile 경계들 상의 경계 확장을 이용하고, 움직임 벡터 예측자들의 리스트들의 특정 도출 프로세스를 이용하여 디코딩되어야 한다.

[0171] 다른 실시예에 따르면, 심지어 HEVC형 타일들이 (예를 들어, XPS에서 그 파라미터들을 연관시키기 위해) 식별자를 가질 필요가 있을 수 있고, 이 식별자는 CTile 식별자와 유사한 방식으로 슬라이스 세그먼트 헤더에서 지정된다. 주어진 프레임에서, HEVC형 타일은 CTile 또는 다른 HEVC형 타일과 동일한 식별자를 갖지 않는다.

[0172] 일 실시예에 따르면, 인코더는 파라미터 세트들, 예를 들어 PPS 또는 TPS 중 하나에서 각각의 공간적 프레임 부분의 'ctile_flag'의 값들을 시그널링함으로써 공간적 프레임 부분이 CTile라는 것을 표시한다. 예를 들어, 인코더는 프레임의 각각의 타일에 대한 고유 식별자를 생성한다. 프레임 부분 배열을 설명할 때, 인코더는 플래그(예컨대, ctile_flag)를 각각의 타일 고유 식별자에 연관시킨다. 이 플래그는 대응하는 타일(즉, 연관된 고유 식별자와 동일한 식별자를 갖는 타일)의 인코딩이 독립적인 디코딩을 보장하도록 제약일 때 참이다. 반대로, 타일의 인코딩이 독립적인 디코딩을 보장하도록 충분히 제약되지 않을 때 이 플래그는 거짓이다.

[0173] 제2 실시예에 따르면, 인코더는 다른 플래그(예컨대, all_ctile_flag)를 포함하는 프레임 부분 배열 정보를 생성한다. 이 플래그가 '1'로 설정된다면, 이는 프레임 부분 배열에서 설명된 모든 타일들이 CTile들이라는 것을 의미한다. 공간적 프레임 부분이 CTile인지 여부를 시그널링하는 플래그들(예컨대, 각각의 ctile_flag)은 생략되고, 참과 동일한 것으로 추론된다. 이 플래그가 0으로 설정되면, CTile들은 이전 실시예 중 하나를 이용하여 명시적으로 설명된다. 파라미터들이 HEVC형 타일들에 특정적인 경우, 이들은 슬라이스 세그먼트 헤더, 예컨대 TPS 대신 XPS에서 시그널링된다. 예를 들어, 1304에 의해 다른 실시예에 대해 참조로 포함되는 slice_segment_address는 HEVC형 타일들에 특정적이다. 실시예에서, TPS가 또한 공간적 프레임 부분이 CTile가 아닌 것을 나타내는 경우, 이것은 TPS에 표시된다. 이 실시예는 슬라이스 세그먼트 헤더의 신택스 및 파싱을 단순화하는 것을 가능하게 한다.

[0174] 다른 실시예에 따르면, 인코더는 슬라이스 세그먼트 헤더에서 'ctile_flag'를 이용하는 대신에 CTile에 대응하는 슬라이스 데이터에 대한 새로운 NAL 유닛 유형들을 정의한다. 예를 들어, 인코더는 CTile 내부에 있는 순간적인 디코딩 리프레시(IDR) 프레임으로부터 슬라이스 NAL 유닛에 대한 CTILE_IDR NAL 유닛을 정의한다. 인코더

는 인코딩 포맷이 보통의 슬라이스 데이터를 지정하는 만큼의 많은 새로운 NAL 유닛 유형들을 정의한다. 예를 들어, HEVC는, 다음과 같은 NAL 유닛 유형들, CRA(clean random access) 픽처의 슬라이스 세그먼트들에 대한 CRA_NUT; RADL(random access decodable leading) IDR 픽처의 슬라이스 세그먼트들에 대한 IDR_W_RADL; 연관된 선행 픽처들이 비트스트림에 존재하지 않는 IDR 픽처의 슬라이스 세그먼트들에 대한 IDR_N_LP; BLA(broken link access) 픽처의 슬라이스 세그먼트들에 대한 BLA_W_LP, BLA_W_RADL, BLA_N_LP; RASL(random access skipped leading) 픽처의 슬라이스 세그먼트들에 대한 RASL_N, RASL_R; RADL 픽처의 슬라이스 세그먼트들에 대한 RADL_N, RADL_R; STSA(step-wise temporal sub-layer access) 픽처의 슬라이스 세그먼트들에 대한 STSA_N, STSA_R; TSA(temporal sub-layer access) 픽처의 슬라이스 세그먼트들에 대한 TSA_N, TSA_R; 및 비-TSA, 비-STSA 후행 픽처의 슬라이스 세그먼트들에 대한 TRAIL_N, TRAIL_R을 정의한다.

[0175] W_LP: 연관된 RASL 또는 RADL 픽처들을 가질수 있고; W_RADL: 연관된 RASL 픽처가 없고; N_LP: 연관된 선행 픽처가 없고; *_N: 픽처가 SLNR(sub-layer non-reference) 픽처이며(그렇지 않으면 이것은 하위 계층 참조 픽처이며); *_R: 픽처가 단지 하위 계층 참조 픽처이다.

[0176] 이러한 HEVC NAL 유닛 유형들은 제약된 타일 데이터에 대한 동일한 목적을 갖는 새로운 대응하는 NAL 유닛 유형들 CTILE_BLA_*, CTILE_CRA_*, CTILE_IDR_*, CTILE_RASL_*, CTILE_RADL_*, CTILE_STSA_*, CTILE_TSA_*, CTILE_TRAIL_*으로 확장될 수 있다. 이러한 새로운 NAL 유닛 유형들 중 하나를 이용하는 것은 NAL 유닛이 CTile에 속한다는 것을 나타낸다.

[0177] 이 대안은 디코딩 프로세스를 단순화하는데, 그 이유는 슬라이스 데이터가 CTile 내부에 있는지 여부를 결정하기 위해 인코더가 각각의 NAL 유닛들의 제1 비트들만을 파싱해야 하기 때문이다.

[0178] 바람직하게는, CTile 식별자의 고유성은 주어진 비트스트림을 의미하는 주어진 시퀀스에 대한 인코딩에서의 구성에 의해 보장된다. 그러나, 상이한 비트스트림들로부터 오는 것을 의미하는, 상이한 시퀀스들로부터 CTile들을 셔플링할 때, 고유성은 보장되지 않는다. 실시예에 따르면, 다양한 시퀀스들로부터 잠재적으로 오는 CTile들과의 공간적 프레임 부분 셔플링을 용이하게 하기 위해, CTile 식별자는 제한된 비트 수에 대해 고유하다. 고유 값은 랜덤 값, 예를 들어, 해시 값, 또는 그 위치를 반드시 표현하지 않는 임의의 다른 값일 수 있다. 따라서, 상이한 비트스트림의 CTile를 취할 때 식별자 충돌을 가질 확률을 감소시킨다.

[0179] 실시예에서, 복수의 시퀀스로부터 CTile들의 셔플링을 수행할 때, 2개의 CTile 식별자 사이의 충돌의 경우에, 충돌된 CTile 식별자들을 대체하는 것으로 충분하다. 모든 슬라이스 세그먼트 헤더들을 재생성할 의무를 갖지 않고 이를 효율적으로 행하기 위해, 바람직한 실시예에서, CTile 식별자를 인코딩하는데 고정된 미리 결정된 비트 수가 이용된다. 예를 들어, 도 13a 및 도 13b에서, CTile 식별자는 8 비트로 인코딩된다.

[0180] 대안적인 실시예에서, 시퀀스 또는 픽처의 모든 CTile 식별자는 동일한 비트 수로 인코딩된다. 이러한 비트 수는 SPS, PPS 또는 RPS와 같은 하나의 파라미터 세트의 파라미터, 예컨대 'uid_num_bits'로 지정된다. 슬라이스 세그먼트 헤더에서, (8의 배수가 아닌 비트 수를 취할 때) CTile 식별자 후에 바이트 정렬 메커니즘을 갖는 것이 바람직하다. 대안적으로, 비트 수는, 예를 들어, 바이트(8 비트) 수: 'uid_num_bytes'로 표현될 수 있다. 다양한 시퀀스들로부터 CTile들을 함께 셔플링할 때, 이들이 모두 동일한 비트 수를 갖지 않을 때 CTile 식별자들을 변경하는 것이 필요할 수 있다. 이것은 몇 개의 슬라이스 세그먼트 헤더들을 변경하는 것을 요구할 것이지만, 바이트(들)만이 추가/제거되거나 대체될 필요가 있을 것이므로, 슬라이스 세그먼트 헤더를 업데이트하는 것보다 더 쉬울 것이다.

[0181] 또 다른 대안적인 실시예에서, 각각의 CTile 식별자들은 가변적인 비트 수로 인코딩될 수 있다. 그 비트 수는 슬라이스 세그먼트 헤더에서 지정된다. 대안적으로, 비트 수는 CTile 식별자에 이용되는 코드로부터 자동으로 결정될 수 있다: 가변 바이트 길이 코드들이 예를 들어, 지수 골롬 코딩(Exponential Golomb coding)(또는 바이트 정렬 비트들이 후속하는 등가적으로 가변 길이 코드)에 이용된다.

[0182] 실시예들에 따르면, CTile 식별자는 시그널링 크기를 감소시키기 위해 의존적인 슬라이스 세그먼트 헤더에서 시그널링되지 않는다. 이어서, 의존적인 슬라이스 세그먼트 헤더의 CTile 식별자가 이전의 독립적인 슬라이스 세그먼트 헤더로부터 추론된다. 대안적인 실시예에 따르면, CTile 식별자는 CTile를 포함하는 서브-비트스트림의 파싱 및 추출을 용이하게 하기 위해 의존적인 슬라이스 헤더에서 시그널링된다.

[0183] 'tile_ctb_addr[i]'(1313) 또는 'slice_segment_address'(1306) 코딩 유닛 어드레스로 CTile 위치를 시그널링하는 것에 대한 대안으로서, 미세 입도 포지셔닝을 이용하여 미세한(fine-grained) CTile들이 도입된다. 이러한 입도는 루마 샘플 위치까지 정밀화될 수 있지만, 다른 실시예에서는, (CTU 크기보다 작은) '2'의 거듭제곱에

대응하는 다수의 루마 샘플의 입도가 충분하다. 일부 실시예들에서, 입도는 미리 결정될 수 있다. 대안적인 실시예에서, 입도는 예를 들어 VPS, SPS 또는 PPS에서 시그널링된다. 미세한 CTile들이 이용될 때, CTile의 치수들은 반드시 CTU 크기의 배수인 것은 아니다.

[0184] CTile의 크기가 CTU 크기의 배수가 아닌 경우, CTile의 우측 및 하측 상의 코딩 유닛들은 도 4에 도시된 바와 같이 픽처의 우측 및 하측 상의 HEVC CTU들에 대해 이용되는 것과 유사한 자동 분할 메커니즘을 이용한다.

[0185] 대안적인 실시예에 따르면, 코딩 유닛들이 불완전한 경우에도, 신택스는 전체 코딩을 설명할 것이고, 분해 트리(예를 들어, 쿼트리 또는 QTBT)의 레이트 왜곡 최적화를 위한 일부 공간을 허용하고, 압축을 개선하기에 적합한 정보의 최종 패딩을 허용한다.

[0186] HEVC형 타일들에서, 치수들은 그리드를 이용하여 지정된다. 따라서, 모든 HEVC형 타일들은 행들 및 열들에 의해 정렬되고, 주어진 행의 모든 HEVC형은 동일한 높이를 갖고, 주어진 열의 모든 HEVC형 타일들은 동일한 폭을 갖는다. 각각의 열의 폭 및 각각의 행의 높이는 XPS에서 지정된다. 미세한 HEVC형 타일들에 의해, 예를 들어 복수의 ROI들의 더 효율적인 코딩을 가능하게 하기 위해 덜 엄격한 배열들을 허용하는 것이 편리할 수 있다.

[0187] 실시예에 따르면, CTile의 치수들은 CTile의 슬라이스 세그먼트의 슬라이스 세그먼트 헤더에서 지정될 수 있다. 결과적인 비트스트림의 크기를 감소시키기 위해, CTile의 치수들은 제1 슬라이스 세그먼트에서만 지정된다. 이어서, 슬라이스 세그먼트들이 동일한 CTile 치수들을 재이용한다. 대안으로서, 모든 CTile들의 치수들은 XPS에서, 예를 들어 CTile 위치들과 함께 제공된다.

[0188] 다른 대안으로서, 치수들은 CTile의 제1 슬라이스 세그먼트 헤더 및 XPS 모두에서 제공된다.

[0189] 다른 대안으로서, CTile들의 치수들은 제공되지 않지만, XPS에서 타일 정보(예를 들어, 위치 또는 'ctile_flag')를 제공하는 데 이용되는 순서로부터 그리고 CTile 위치로부터 추론되며, 예를 들어, CTile 위치들은 XPS에서 선언되고, CTile의 각각의 대응하는 하단 우측 코너가 증가하는 래스터 스캔 순서(예를 들어)로 순서화되도록 CTile 위치들이 순서화된다. 이하의 도 14는 이러한 순서화의 예를 제공한다.

[0190] 실시예에 따르면, 도 13a의 예에서 이용되는 dependent_slice_segment_enabled_flag는 HEVC에서와 동일한 의미를 갖는다: 이것은 의존적인 슬라이스 세그먼트들이 허용되는지 여부를 나타내는 데 이용된다. HEVC에서, dependent_slice_segment_enabled_flag는 PPS에서 시그널링된다. 우리의 바람직한 실시예들에 따르면, dependent_slice_segment_enabled_flag는 (동일한 비트스트림에서의 의존적인 슬라이스 세그먼트들 없이 인코딩된 CTile들과 함께 의존적인 슬라이스 세그먼트들로 인코딩된 CTile들의 이용을 허용하기 위해) 각각의 CTile에 대해 tiling_parameter_set(TPS)에서 시그널링된다. 모든 CTile들이 의존적인 슬라이스 세그먼트로 인코딩되거나 의존적인 슬라이스 세그먼트 없이 인코딩되는 공통 이용 사례들의 TPS에서의 신택스를 줄이기 위해, TPS 구조의 루트에서 다른 플래그 dependent_slice_segment_enabled_flag_for_all_ctiles가 이용된다. 이 플래그가 1로 설정될 때, dependent_slice_degment_enabled_flag는 각각의 CTile에 대해 시그널링되지 않는다. 대신에, ctile_dependent_slice_segment_enabled_flag가 또한 TPS 구조의 루트에서 시그널링되고, 각각의 CTile의 dependent_slice_degment_enabled_flag에 대해 추론될 값을 제공한다. HEVC형 타일들에 대해, dependent_slice_segment_enabled_flag가 PPS에서 여전히 시그널링될 수 있지만, 바람직한 실시예들에서는, TPS에서 시그널링된다.

[0191] 대안적인 실시예에 따르면, dependent_slice_segments_enabled_flag는 전혀 시그널링되지 않고, 신택스를 단순화하기 위해 항상 참으로서 추론된다.

[0192] 도 14는 비-그리드 기반 파티셔닝의 예를 도시한다. 프레임(1401)은, #1부터 #15까지 넘버링된 15개의 CTile로 분할된다. 이러한 넘버링은, 각각의 타일의 하단 우측 코너가 래스터 스캔 순서로 순서화되도록, XPS에서 타일 위치의 선언 순서를 제공한다. 이러한 순서화를 이용하여 각각의 CTile의 크기가 추론될 수 있다. 예를 들어, 마지막 CTile, CTile #15를 취하면, 그 치수들이 추론될 수 있는데, 그 이유는 마지막 타일이고, 그 하단 우측 코너가 래스터 스캔 순서에서 마지막이고, 따라서 이것은 프레임의 하단 우측 코너이기 때문이다. 이어서, 슬라이스 #15의 치수들은 프레임의 치수들 - 그 위치: h#15 = h_frame - y#15; w#15 = w_frame - x#15이다. 타일 #14는 CTile #15 이전의 마지막 하단 우측 코너를 그 자신의 하단 우측 코너로서 가져야 하고, 따라서 하단 우측 코너 위치는 최하단(프레임의 하단) 및 최우측(이전 타일의 좌측 바로 위)이다. 이어서, CTile #14 치수들은 h#14 = h_frame - y#14; w#14 = x#15 - x#14이다. 동일한 것이 타일들 #13 및 #12에 대해 반복된다. 그 다음, 타일 #11의 경우 최하단 위치들이 채워지기 때문에, 새로운 최하단 위치는 y#14이다. 그 다음, h#11 = y#14 - x#11이다. CTile #1까지 그런 식이다.

[0193] 대안적인 실시예들에 따르면, XPS에서 CTile 위치들을 지정하는 대신에, CTile 치수들만이 지정되고, CTile 위치들은 (예를 들어, 래스터 스캔 순서로 증가하는) 좌측 상단 위치에 따라 순서화된 CTile들을 이용하여 CTile 치수들로부터 계산된다. 치수들로부터 위치들을 계산하는 알고리즘은 위치들로부터 치수들을 계산하기 위한 전술한 알고리즘으로부터 용이하게 도출될 수 있다.

[0194] 실시예에 따르면, XPS에 설명된 CTile 파라미터들은 존재하지 않는 CTile들에 대한 CTile 위치(및/또는 CTile 치수들)를 제공할 수 있고, 이들 CTile에 대한 슬라이스 세그먼트가 존재하지 않을 것이다. 이 CTile 설명은, 위치들 또는 치수들만이 제공되고, 치수들 또는 위치들이 추론되는 실시예들에서, 적절한 추론을 허용하는데 필요하다.

[0195] 비디오 렌더링의 경우, 디폴트 샘플 값 또는 패딩 방법이 존재하지 않는 CTile들을 채우는 데 이용되거나, 또는 대안적으로 패딩 방법의 값 또는 인덱스가 XPS 파라미터에서 제공된다. 예컨대, 이것은 (예를 들어, 도 9의 단계(900) 전에) 디코딩 프로세스의 시작에서 예비 단계를 추가함으로써 구현될 수 있으며, 이 예비 단계는,

[0196] - 렌더링 버퍼에서 프레임의 콘텐츠를 적절한 디폴트 샘플 값으로 초기화하는 단계, 및/또는

[0197] - 모든 프레임 부분들이 디코딩된 후에, 예를 들어, 인-페인팅 방법을 이용하여, 임의의 타일 또는 CTile에 의해 커버되지 않는 모든 영역들을 패딩하는 것으로 구성되는 새로운 단계를 추가하는 단계로 구성된다.

[0198] 실시예에 따르면, 동일한 공간적 위치에서의 복수의 CTile들 또는 중첩하는 CTile들을 처리하는 것이 가능하다. 각각의 CTile 식별자의 경우, 연관된 디코딩된 CTile 버퍼(HEVC에서 디코딩된 픽처 버퍼(DPB)와 등가이지만 여기서는 디코딩된 CTile 데이터만을 포함함)가 존재한다. 주어진 프레임에 대해, 각각의 CTile는 연관된 디코딩된 CTile 버퍼에서 이용가능한 시간적 데이터를 이용하여 디코딩된다. 다음으로, 제1 대안에 따르면, CTile들의 렌더링 순서는 비트스트림에서의 CTile 순서와 동일한 순서이다. 제2 대안에서, CTile들은 XPS 데이터로부터 결정될 수 있는 렌더링 순서와 연관된다. 두 대안들의 경우, 각각의 CTile의 디코딩 결과들의 샘플들은 CTile들의 렌더링 순서로 렌더링 프레임 버퍼의 프레임 내에 놓여진다(이어서 가능하게는 이전 CTile에 의해 순서대로 이전에 놓인 샘플들을 삭제/마스킹한다).

[0199] 실시예에 따르면, CTile 샘플들은 렌더링 프레임 버퍼의 프레임에서 CTile를 렌더링할 때 적용될 투명도의 레벨을 나타내는 알파 채널을 더 포함한다. 대안적으로, 샘플은 CTile의 어느 샘플들이 렌더링 프레임 버퍼의 프레임에 렌더링되어야 하는지를 나타내는 이진 마스크 값을 더 포함한다.

[0200] 실시예에 따르면, 중첩하는 동일한 위치 또는 CTile들에서 복수의 CTile들을 처리하는 것이 가능한 경우, CTile 위치 및 CTile 크기 둘 다는 그 컨텍스트에서 다른 것으로부터 하나를 추론하는 것이 가능하지 않기 때문에 XPS에서 지정되어야 한다.

[0201] 임의의 주어진 후필터링 알고리즘(예컨대, 디블록킹 필터, 샘플 적응적 오프셋, 또는 적응적 루프 필터)에 대한 실시예에 따르면, CTile 경계 후필터링 플래그들은 후필터링 알고리즘이 CTile에 대해 이용가능한지 여부를 표시하기 위해 XPS에서 지정될 수 있다. CTile 경계 후필터링 플래그, 예를 들어, 'usable_for_post_filtering_flag'는, 주어진 후필터링 알고리즘이 (시간적 디코딩을 수정할 수 있으므로, 디코딩된 픽처 버퍼에서가 아닌) 렌더링 프레임 버퍼의 렌더링된 프레임에서의 CTile 경계에 적용될 수 있음을 나타낸다. 유리하게도, 이는 예를 들어 시각적 품질을 개선하는 것을 목표로 한다. 플래그는 CTile의 전체 프레임 레벨 및/또는 각각의 프레임 레벨에 대해 지정될 수 있다. 이 플래그는 후필터링될 때 아티팩트들을 도입하기 쉬운 것으로 알려진 일부 에지들의 필터링을 방지하는데 유용할 수 있다. 예를 들어, 플래그는 적응적 품질 스트리밍의 맥락에서 CTile 셔플링에 대해 참일 것이지만, CTile 경계가 360° 콘텐츠의 정육면체 투영의 2개의 면 사이에 있고, 그 면들이 그 에지 상에서 인접하지 않는 경우 거짓일 수 있다. 후필터링되는 CTile 경계들은 에지의 2개의 측이 후필터링이 적용될 수 있다는 것을 지정하고 있는 경계들 또는 그 에지가 후필터링을 허가하는 CTile와 HEVC형 타일 사이에 있을 때의 경계들이다.

[0202] 대안적인 실시예에 따르면, CTile 경계들은 디코딩된 픽처 버퍼(DPB)에서 후필터링될 수 있고, 그 실시예에서 디코딩이 임의의 디코딩 구성에서 정확하다는 것을 보장하기 위해, 인터 예측이 이용될 때, 그 예측에 이용되는 샘플들은 후필터링되지 않은 샘플들이다. 따라서, 경계 확장 메커니즘은 경계 정보를 이용하여 후필터링된 샘플 이전의 경계 상의 마지막 샘플들에 적용되며, 이는 경계 확장이 필터링되지 않은 샘플들에 대해 수행된다는 것을 의미한다.

[0203] 대안적인 실시예에 따르면, 하나보다 많은 CTile가 동일한 CTile 식별자를 가질 수 있다. 이 실시예에서,

CTile 식별자는 CTile 세트 식별자가 된다. CTile 세트를 형성하는 CTile들의 세트는 적절히 디코딩되기 위해 모두 함께 그리고 동일한 상대적 포지셔닝을 갖고 유지되어야 한다.

[0204] 이들 실시예들에서, CTile 세트의 위치 및 크기는 XPS로부터 추론된다. 이것은 CTile 세트에 속하는 CTile들의 세트의 경계 박스의 위치 및 크기에 대응한다. 따라서, XPS에서, CTile 세트 식별자는 하나 이상의 위치 및 크기(CTile 세트 내의 각각의 CTile에 대해 하나)와 연관된다.

[0205] 이들 실시예들에서, 슬라이스 세그먼트 헤더 'ctb_addr_offset_inside_tile'(1305) 정보는 'ctb_addr_offset_inside_tile_set' 정보로 대체될 수 있다. 'ctb_addr_offset_inside_tile_set'는 어느 CTile가 슬라이스 세그먼트에 속하고, 따라서 슬라이스 세그먼트를 디코딩하는 동안 이용될 기하학적 구조를 추론할 수 있게 한다.

[0206] 이러한 실시예들 중 하나에서, CTile의 세트의 임의의 샘플이 시간적 움직임 보상에 이용될 수 있다. 움직임 보상이 CTile 세트 밖의 샘플 값들을 이용하는 경우, 이 샘플 값들은 (경계 확장을 적용하는 것과 동등하지만, 2개의 CTile에 의해 공유되지 않는 CTile 경계 부분들에 대해서만의) CTile 세트들 중 임의의 하나의 공간적으로 가장 가까운 샘플의 값으로 설정된다. CTile 밖의 임의의 샘플이 하나보다 많은 가장 가까운 CTile 샘플을 갖는 경우, 간단한 규칙이 이용되어 어느 것을 이용할지, 예컨대 가장 작은 래스터 스캔 순서를 갖는 것을 이용하기로 결정한다.

[0207] 도 15는 CTile 식별자를 시그널링하기 위한, 도 13a, 도 13b 및 도 13c에 도시된 것들에 대한 대안적인 실시예들을 도시한다. 현재의 블록 기반 코덱, 전형적인 HEVC에서, NAL 유닛 헤더(1501)는 다음과 같은 필드들을 포함한다:

[0208] - 0: (거짓)으로 설정된 1 비트;

[0209] - NAL 유닛 유형: (유형)을 포함하는 6 비트;

[0210] - HEVC에서 항상 0과 동일하지만 예를 들어 규모조정가능한 HEVC(SHVC)에서의 규모조정가능한 계층 인덱스 또는 멀티-뷰 HEVC(MV-HEVC)에서의 뷰 인덱스에 대응하는 계층 식별자: (LayerId)를 포함하는 6 비트; 및

[0211] - HEVC에서 시간적 규모조정성에 대한 시간적 계층 인덱스에 대응하는 시간적 계층 식별자: (TID)를 나타내는 3 비트.

[0212] NAL 유닛 헤더(1501)에 기반한 실시예에서, 인코더는 비디오 시퀀스를 프레임 부분들로 분할한다. 인코더는 각각의 프레임 부분에 대해 하나의 인코딩 또는 규모조정성 계층을 이용한다. 이것은 공간적 영역에 기반한 계층 인코딩으로서 보여질 수 있다. 인코더는 다른 영역들과 독립적으로 각각의 공간적 영역 계층을 인코딩할 수 있다. 이러한 경우에, 각각의 공간적 영역 계층은 하나의 CTile에 대응한다. 이러한 특정 경우에, 인코딩할 때 공간적 영역 계층의 모든 슬라이스들은 참으로 설정된 ctile_flag를 갖는다. 주요 차이는 각각의 공간적 영역 계층이 HEVC형 타일들로 추가로 분할될 수 있다는 것이다.

[0213] 인코더는 상이한 공간적 영역 계층들을 LayerId로 시그널링한다. 이것은 LayerId의 값을 CTile의 식별자와 동일하게 설정한다. 그 결과, CTile 식별자는 슬라이스 세그먼트 헤더에서 요구되지 않는다. 이것이 고정된 비트 길이이므로, 비디오 스트림의 프레임 부분들을 셔플링할 때 CTile 식별자의 처리가 간단하게 유지된다.

[0214] 인코더는 파라미터 세트들 중 하나, 예를 들어 VPS에서 프레임 부분 배열을 시그널링한다. VPS는 공간적 영역 계층의 고유 식별자를 이전 실시예들에서 설명된 신택스에 대응할 수 있는 신택스를 갖는 디코딩 위치와 연관시킴으로써 각각의 공간적 영역 계층의 디코딩 위치를 표시한다.

[0215] 인코더는 또한 비디오 스트림의 상이한 계층들 사이의 의존성들을 설명한다. 그 다음, 디코더는, 파라미터 세트 NAL 유닛들에 설명된 계층들 간의 의존성들의 분석을 통해 다른 계층들과는 독립적으로 코딩된 공간적 영역 계층을 결정한다.

[0216] 인코더는 공간적 영역 계층들의 서브세트를, 다른 공간적 영역 계층들과는 독립적으로 CTile로서 압축한다. 하나의 공간적 영역 계층들이 이전 프레임(즉, 동일한 CTile 식별자를 가짐)에서의 다른 공간적 영역 계층(이 계층의 슬라이스들은 거짓으로 설정된 ctile_flag를 가짐)에 의존하는 경우, 인코더는 이 의존적인 계층으로부터의 참조 프레임들을 현재 계층의 디코딩된 픽처 버퍼에 추가한다. 업샘플링 또는 다운샘플링 필터는, 참조 프레임들이 현재 계층의 크기와 동일한 크기를 갖도록 2개의 계층의 크기들이 상이할 때 적용된다.

[0217] 실시예에 따르면, LayerId는 또한 ctile_flag를 추론하는 데 이용될 수 있다: LayerId가 0일 때, NAL 유닛은

HEVC형 타일에 속한다. LayerId가 0이 아닌 경우, NAL 유닛은 CTile 식별자가 LayerId와 동일한 CTile에 속한다. 대안적으로, LayerId의 1 비트는 ctile_flag를 시그널링하기 위해 예비된다. CTile 식별자를 전송하기 위해 LayerId를 이용하는 것의 이점은 이것이 CTile의 추출을 위해 비트스트림을 파싱하는 복잡성을 크게 감소시킨다는 점이다.

[0218] 다른 실시예에서, 공간적 영역 규모조정성은 HEVC에서의 시간적 규모조정성과 유사하게 정의되고, 즉 상이한 계층 식별자들이 다른 규모조정성 계층들(예컨대, SNR, 해상도, 멀티-뷰)로부터 시간적, 공간적 영역을 식별한다. 실제로, 이러한 접근법의 이점은 공간적 영역 규모조정성 및 SNR 또는 해상도 규모조정성 계층들 둘 다를 이용하는 것이 가능하다는 것이다.

[0219] 이제 프레임 부분 식별자를 나타내는 RAID(random access identifier)로 NAL 유닛 헤더(1502)가 확장된다. LayerId 시멘틱들은 HEVC에 대해서와 동일하게 유지되며, 즉 이는 멀티-뷰, SNR 또는 해상도 규모조정성 계층을 나타낸다.

[0220] 인코더는 그 RAID 값을 파라미터 세트들 중 하나, 예컨대 VPS에서의 디코딩 위치와 연관시킴으로써 각각의 공간적 영역 계층의 위치를 지정한다. 공간적 영역을 인코딩하는 각각의 NAL 유닛들(SPS, PPS 및 VCL NAL 유닛들을 포함함)은 공간적 영역에 대응하는 프레임 부분 식별자(CTile (세트) 식별자)와 동일한 RAID를 갖는다.

[0221] 그 결과, (비디오 비트스트림들의 세트로부터 CTile들을 추출하고 이들을 새로운 비디오 비트스트림으로 조합하는 것으로 구성되는) 전술한 병합 프로세스는 조합할 비디오 스트림들과 관련된 프레임 부분 배열로부터 병합하기 위해 공간적 영역 계층의 CTile 식별자들을 추출한다. 그 다음, 이것은 추출된 식별자들의 세트와 동일한 RAID 값을 갖는 모든 NAL 유닛들을 추출한다.

[0222] 2개의 비디오 시퀀스를 조합할 때 식별자의 충돌 위험을 제한하기 위해, 인코더는 RAID 값을 랜덤 값으로 설정한다. 이것은 비디오 시퀀스가 단일 프레임 부분을 포함할 때의 경우들을 포함한다.

[0223] 실시예에 따르면, RAID는 공간적 영역이 CTile인지를 지정한다(이것은 슬라이스 세그먼트 헤더들에서의 ctile_flag의 시그널링을 대체한다): RAID가 0일 때, NAL 유닛은 HEVC형 타일에 속한다. RAID가 0이 아닐 때, NAL 유닛은 RAID와 동일한 식별자를 갖는 CTile에 속한다. 대안적으로, RAID의 1 비트는 ctile_flag를 시그널링하기 위해 예비된다. 대안적인 실시예들에서, RAID 식별자는 더 많은 CTile들을 허용하기 위해 16 비트 또는 24 비트이다.

[0224] 실시예에 따르면, CTile들의 시퀀스는 독립 비트스트림으로서 고려된다. 예를 들어, 일부 실시예들에서, 동일한 식별자를 갖는 CTile들의 시퀀스 순서는 다른 식별자를 갖는 CTile들의 시퀀스 순서와 상이할 수 있다(즉, GOP 구조는 2개의 CTile 사이에서 상이할 수 있다). 따라서, 동일한 프레임 내의 2개의 CTile는 상이한 NAL 유닛 유형 또는 TID를 가질 수 있다.

[0225] 다른 실시예에서, XPS는 디코더가 어떠한 에러들도 없이 이들을 처리할 수 있는 방식으로, CTile들 사이의 일부 의존성들을 설명하는 추가적인 정보를 포함한다. 본 발명자들은 CTile들의 독립성이 각각의 CTile들의 레벨 대신에 CTile들의 세트의 레벨에서 고려될 수 있다는 것을 알았다. 이 구성에서, CTile 세트 내의 일부 CTile들은 일부 의존성들을 가질 수 있다.

[0226] 예를 들어, 도 16a는 CTile마다의 의존성 리스트를 포함하는 XPS를 나타낸다. 의존성 리스트는 식별자(1600)를 갖는 주어진 CTile에 의존하는 CTile 식별자들(1601)을 제공한다. 주어진 CTile가 다른 CTile에 대한 의존성들을 갖는 것으로 표시되는 경우, 이는 주어진 CTile가 다른 CTile 없이 추출될 수 없다는 것을 의미한다.

[0227] 도 16b는 CTile 의존성들의 제1 예를 도시한다. 현재 프레임(1602)에서의 CTile #1은 이전의 인코딩된 프레임(1603)으로부터 움직임 보상을 수행할 때 CTile #2로부터의 샘플 값들을 이용하고, CTile #2는 CTile #1로부터의 샘플 값들을 이용한다. 이러한 예에서, XPS는 식별자 #1을 갖는 CTile가 식별자 #2를 갖는 CTile에 대한 의존성들을 갖고, 식별자 #2를 갖는 CTile가 식별자 #1을 갖는 CTile에 대한 의존성들을 갖는다는 것을 나타낸다. 이러한 상호 의존성들의 시그널링은 CTile 세트를 시그널링하는 것에 대한 대안이다.

[0228] 도 16c는 CTile 의존성들의 제2 예를 도시한다. 이러한 예에서, CTile #3은 프레임들(1604 및 1607)에서 CTile들 #1 및 #2에 대한 의존성들로서 프레임들(1605 및 1607)에 존재한다. 이러한 예에서, CTile #3은 또한 CTile들 #1 및 #2를 추출하지 않고 추출될 수 없다. 그러나, CTile들 #1 및 #2는 의존성들을 갖지 않고 단독으로 추출될 수 있다. 실시예들에 따르면, 이 시나리오는, 예를 들어 프레임들(1604 내지 1606)이 시간적으로 순서화되는 경우, 다양한 프레임 레이트들에서 CTile들의 추출을 용이하도록 적용될 수 있다. 대안적으로, 이것

은 상이한 품질 계층들에 대한 CTile들의 추출을 용이하게 하기 위해, 예를 들어 프레임(1605)이 프레임(1604)의 정밀화 계층이고, 프레임(1607)이 프레임(1606)의 정밀화 계층인 경우 규모조정가능한 코딩에서 이용될 수 있다.

[0229] 일부 실시예들에 따르면, CTile는 연속적인 프레임들 사이에서 공간적 위치 또는 크기를 변경할 수 있다.

[0230] 실시예들에 따르면, '타일 파라미터 세트', TilePS가 또한 도입된다. TPS는 CTile들의 서브세트의 CTile 파라미터들(하나의 이동하는 및/또는 변경하는 크기만)을 업데이트하는 것을 허용하고, 예를 들어 'num_updated_tiles' 값을 포함하고, 이 후 타일 식별자를 'num_updated_tiles' 수정된 CTile들에 대한 그 새로운 특성들에 연관시킨다.

[0231] 고전적으로, 움직임 벡터들은 인코딩할 블록과 병치된 블록에 대한 참조 이미지에서 예측자 블록의 위치를 제공한다. 인코딩할 주어진 블록에 대해, 제1 단계는 참조 이미지에서 병치된 블록을 식별하는 것이다. 병치된 블록은 인코딩할 블록과 동일한 원점(좌측 상단 위치) 및 동일한 크기를 의미하는, 동일한 위치를 갖는 참조 이미지 내의 블록으로 정의된다. 그 후, 움직임 벡터는 예측자 블록의 원점을 결정하기 위해 병치된 블록의 원점에 적용된다.

[0232] CTile들을 고려할 때, 병치된 블록의 결정은 CTile 내에서 동일한 위치를 갖고 프레임 내에 더 이상 존재하지 않는 블록을 고려하도록 적응된다. CTile가 셔플링되었을 때, 이것은 프레임 내의 그 위치가 인코딩에서의 프레임 내의 그 위치에 대해 디코딩에서의 프레임 내에서 수정되었음을 의미한다. 그러나, 독립적인 디코딩을 보장하기 위해 CTile 내에서 예측이 억제되는 것을 고려하면, 우측 예측자 블록은 CTile 내의 인코딩 블록에 병치된 블록에 움직임 벡터를 적용함으로써 여전히 결정될 수 있다. 이것은 CTile가 그 크기 및 위치를 프레임별로 유지하는 한 그러하다. CTile가 프레임 내의 그 위치 및/또는 2개의 연속적인 프레임 사이의 그 크기를 변경할 때 어려움이 발생한다. 이 경우, 인코더 및 디코더는 예측자 블록을 정확하게 결정하기 위해 움직임 벡터가 적용될 참조 프레임 내의 병치된 블록의 위치를 결정하는 방식을 수용한다.

[0233] CTile가 연속적으로 인코딩된 프레임들 사이의 위치 또는 크기를 변경할 수 있는 실시예에 따르면, 2개의 연속적인 프레임에서의 CTile의 상대 위치는 2개의 상이한 비트스트림에서 동일하지 않을 수 있다. 도 17은 이러한 실시예들의 예를 제공하며, 여기서 제1 비트스트림은 비디오 감시(1700)의 프레임들을 포함한다. 제1 프레임(1701)에서, 주어진 ctile_id를 갖는 이동하는 관심 영역에 대한 CTile(1702)를 포함하는 몇 개의 공간적 프레임 부분들이 있다. 다른 프레임(1703)에서, 주어진 ctile_id를 갖는 CTile는 이동하였고, 크기(1704)를 변경했다. 제2 비트스트림은 1700으로부터 추출된 CTile들을 조립함으로써 생성된 비디오(1705)를 포함하고, 생성된 CTile들은 균일한 컬러(예를 들어, 흑색)를 포함한다. 제1 프레임(1706)에서, 주어진 ctile_id(1702)를 갖는 CTile는 제1 비트스트림으로부터 추출되어 프레임(1707)의 중심에 넣어졌다. 다른 프레임(1708)에서, 주어진 ctile_id(1704)를 갖는 CTile는 제1 비트스트림으로부터 추출되어 프레임(1709)의 중심에 넣어졌다. 제1 비트스트림에서, CTile(1704)는 CTile(1701)에 대한 시간적 참조와 함께 인터 예측을 이용한다. 따라서, 생성된 비디오(1705)에서, CTile(1709)는 CTile(1707)에 대한 시간적 참조와 함께 인터 예측을 이용한다. CTile(1702 및 1704) 사이의 상대적인 공간적 위치는 1707과 1709 사이에서와 동일하지 않다. 따라서, 디코딩 상대 위치가 무엇이든 간에 적절히 디코딩되기 위해, 인터 예측 모드들이 이용될 때, 인코딩된 움직임 벡터들은 CTile 위치 변경(즉, 연속적인 프레임들 사이의 상대적인 공간적 위치)을 고려하지 않는다.

[0234] 제1 대안에 따르면, 움직임 벡터들은 2개의 연속적으로 인코딩된 프레임 내의 CTile의 미리 결정된 참조 포인트가 동일한 공간적 위치(예컨대, 좌측 상단, 우측 상단, 좌측 하단, 우측 하단, 중간 상단, 중간 하단, 좌측 중간, 우측 중간 또는 중심)에 있는 것처럼 계산된다. 따라서, 인코딩된 움직임 벡터는 참조 프레임에서의 블록에 대한 움직임 벡터에서, 그 참조 프레임에서의 CTile의 참조 포인트 사이의 움직임 벡터를 뺀 것에 또한 대응할 것이므로, CTile에 대한 참조에서의 움직임 벡터가 획득된다. 이것은 CTile가 이후에 나중의 공간적 변경과 독립적으로 디코딩가능하다는 결과를 낳는다.

[0235] 도 17b는 참조 프레임(1713)에서 동일한 ctile_id를 갖는 참조 CTile(1712)를 이용하여 프레임(1711)에서 인코딩된 CTile(1710)에 대한 것을 도시한다. 블록(1714)은 프레임(1716) 내의 움직임 벡터와 미리 결정된 참조 포인트들(1717)(이 예에서 미리 결정된 참조 포인트들은 CTile의 좌측 상단 코너에 있음) 간의 움직임 벡터 사이의 차이에 대응하는 움직임 벡터(1715)를 이용하여 인코딩된다. 도 17b는 또한 CTile의 상대적인 시간적 디코딩 위치가 인코딩 위치와 동일하지 않더라도, 인코딩된 벡터(1715)는 디코딩된 프레임(1720)에서 움직임 벡터를 획득하기 위해 디코딩된 프레임(1719) 내의 미리 결정된 참조 포인트들 사이의 움직임 벡터에 이를 추가함으로써 블록(1718)을 디코딩할 때 여전히 유효하다는 것을 도시한다.

[0236] 제2 대안에 따르면, 움직임 벡터들은 CTile의 주어진 포인트가 2개의 연속적으로 인코딩된 프레임에서 동일한 공간적 위치에 있는 것처럼 계산된다. 주어진 포인트는 미리 결정된 포인트들(예컨대, 좌측 상단, 우측 상단, 좌측 하단, 우측 하단, 중간 상단, 중간 하단, 좌측 중간, 우측 중간 또는 중심)의 리스트에서의 인덱스로서 CTile 인코딩된 데이터에서 시그널링된다.

[0237] 제3 대안에 따르면, 고정된 포인트(또는 대안적으로 시그널링된 포인트)가 고려되고, 움직임 벡터가 CTile 인코딩된 데이터에서 인코딩된다. CTile 인코딩된 데이터에서 인코딩된 움직임 벡터는, 2개의 연속적으로 인코딩된 프레임 내의 CTile의 고정된(또는 시그널링된) 포인트가 동일한 공간적 위치에 있다고 고려할 때 CTile들 내의 시간적 예측들에 연관된 인터 움직임 벡터들 각각에 추가되는 움직임 벡터를 제공한다. 이것은 인코더가 움직임 벡터들의 코딩 비용을 감소시키게 한다. 예를 들어, 인코더는 CTile의 움직임 보상된 블록들의 평균 움직임 벡터를 선택할 수 있다. 도 17b를 예로 들면, 이 평균 움직임 벡터는 벡터(1715)로 감산될 것이다. 그 감산의 결과는 인코딩될 움직임 벡터로서 고려된다. 결과의 관점에서 동등한 대안은, 고정된 포인트 또는 고정된 포인트 인덱스 및 움직임 벡터 대신에(또는 이에 추가하여), 인코딩된 CTile에서 참조 포인트의 (서브) 픽셀 위치를 제공하는 것이다.

[0238] 제4 대안에 따르면, 고정된 포인트(또는 대안적으로 시그널링된 포인트)가 고려된다. 움직임 벡터 필드의 파라미터들은 CTile 인코딩된 데이터에서 인코딩된다. 움직임 필드는, 2개의 연속적으로 인코딩된 프레임 내의 CTile의 고정된(또는 시그널링된) 포인트가 동일한 공간적 위치에 있다고 고려할 때 CTile들에서의 시간적 예측들에 연관된 인터 움직임 벡터들 각각에 추가되는 움직임 벡터를 결정하는 것을 허용한다. 예를 들어, 인코더는 CTile 내의 블록들의 움직임 벡터들을 추정할 수 있고, 그 잔차를 최소화하여 그 인코딩의 비용을 감소시키기 위해 그 예측을 최소화하는 움직임 벡터 필드를 추정할 수 있다. 인터 인코딩된 블록의 각각의 움직임 보상 벡터는 이어서 움직임 벡터 필드 파라미터들로부터 계산된 움직임 벡터를 움직임 보상 벡터(예를 들어, 1715)로 감산한 결과이다.

[0239] 실시예들에 따르면, 인터 예측 모드들은 2개 이상의 이전에 인코딩된 참조 프레임을 지칭할 수 있다. 그 실시예에서, 전술한 실시예들은 CTile의 고정된 포인트(또는 시그널링된 포인트)가 인코딩된 프레임 및 모든 참조 프레임들에서 정렬되는 것을 고려하도록 확장될 수 있다.

[0240] 움직임 벡터 또는 움직임 벡터 필드가 시그널링되는 실시예들에서, 복수의 프레임에 대한 확장은 2개의 대안적인 방식, 즉

[0241] - 참조된 프레임들의 수만큼 많은 움직임 벡터들(또는 움직임 벡터 필드 파라미터들)을 시그널링함으로써, 또는

[0242] - 인코딩된 프레임에 대한 그 시간적 차이에 따라 각각의 참조 프레임에 대한 하나의 움직임 벡터(또는 움직임 벡터 필드)를 도출하는 데 이용되는 단지 하나의 움직임 벡터(또는 움직임 벡터 필드) 'x'를 시그널링함으로써 수행될 수 있다.

[0243] 예를 들어, 선형 규모조정이 이용된다: 참조 프레임의 시간적 위치가 't-s'(여기서 's'는 프레임들 사이의 일정한 시간적 샘플링 기간이고, 't'는 시간임)이고, 인코딩된 프레임의 시간적 위치가 't'인 경우, 이용된 규모조정 계수는 '(t)/s - (t-s)/s = 1'이지만, 참조 프레임의 시간적 위치가 't+2s'인 경우, 이용된 규모조정 계수는 '(t)/s - (t+2s)/s = -2'이다. 규모조정 계수는 각각의 참조 프레임에 대한 움직임 벡터를 계산하기 위해 적용된다. 예를 들어, 도 17b를 보면, 참조 프레임(1713)이 't+2s'인 경우, 움직임 벡터(1715)는 -2*'x'로 감산된다. 감산의 결과 'y'는 인코딩되는 움직임 벡터의 값이다(예를 들어, 인코딩 모드가 HEVC의 인터 예측 모드이면, 'y'는 움직임 벡터 예측자 인덱스를 이용하여 예측되는 움직임 벡터이다). 다시 말해서, 디코더측에서, 움직임 벡터 'y'는 움직임 보상된 블록에 대해 디코딩되고, 이어서 -2*'x'를 추가하여 벡터(1715)를 획득한다. 움직임 벡터(1717)는 또한 프레임 레벨에서 움직임 벡터를 획득하기 위해 추가된다.

[0244] 도 12는, 이전의 설명에서 언급된 바와 같이, 비트스트림이 상위 레벨 설명 포맷으로 캡슐화될 때, 인코더측에서의 캡슐화 단계(906 또는 1105)에 대한 상세들을 제공한다.

[0245] 바람직한 실시예에서, CTile들을 갖는 비디오 비트스트림은 ISO 베이스 미디어 파일 포맷(ISOBMFF, ISO/IEC 14496-12 및 14496-15)에 따라 캡슐화된다. 도 12에 관련된 다음의 설명에서, 단어 "샘플"은 "프레임", 즉 ISOBMFF에 대해 정의된 바와 같이, 코딩된 픽처에 대응하는 비디오 비트스트림으로부터의 NAL 유닛들의 세트에 대응한다.

[0246] 캡슐화는 ISOBMFF 또는 mp4 기입기에 의해 처리된다. 이 기입기는 NAL 유닛 헤더들의 파서를 포함한다. 이것

은 NALU 유형들, 식별자들 및 대응하는 압축된 데이터를 추출할 수 있다. 전형적으로, 추출된 NALU 데이터는 캡슐화된 파일의 미디어 데이터 컨테이너: 'mdat' 박스에 배치된다. NAL 유닛들의 설명에 대한 메타데이터는 메인 'moov' 박스 아래의 박스들의 구조화된 계층구조에 배치된다. 하나의 비디오 비트스트림은, 그 서브 박스들을 갖는 'trak' 박스에 의해 설명된 비디오 트랙으로 캡슐화된다.

[0247] 파티셔닝된 비디오 프레임들에 대해, 예측된 비디오의 이용에 따라 상이한 가능한 캡슐화들이 존재한다. 이 이용은 mp4 기입기 애플리케이션에서 하드 코딩되거나, 예를 들어 초기화 단계(1200)에서, 하나의 사용자 또는 다른 프로그램에 의해 입력 파라미터로서 제공될 수 있다. 실시예에서, 하나의 프레임 부분 또는 주어진 세트의 프레임 부분들을 하나의 비디오 트랙에 캡슐화하는 것이 편리할 수 있고, 따라서 멀티-트랙 캡슐화로 이어진다.

[0248] ISOBMFF 기입기의 초기화가 수행되었으면, 인코더는 NALU 유형들, 특히 파라미터 세트들(XPS)에 대응하는 것들을 판독함으로써 단계(1201)에서 비디오 비트스트림을 파싱하기 시작한다. 전술한 바와 같이, 파라미터 세트들은 비디오 비트스트림의 코딩 구성 등에 관한 상위 레벨 및 일반 정보를 제공하는 특정 NAL 유닛들이다.

[0249] 이들 파라미터 세트들의 파싱으로부터, mp4 기입기는 테스트(1202)에서 비디오 비트스트림이 프레임 부분들(예를 들어, TPS의 존재 또는 파라미터 세트들 중 하나에서의 특정 파티셔닝 구성)을 포함하는지를 결정할 수 있다. 프레임 부분들이 존재하는 경우, mp4 기입기는 동일한 테스트(1202)에서 이들이 "제약된 타일들", 즉 CTile들인지를 결정한다. 비트스트림이 프레임 부분들을 포함하지 않거나 CTile들을 포함하지 않는 경우, 테스트(1202)는 거짓이고, 비디오 비트스트림은 단계(1203)에서 하나의 비디오 트랙으로서 캡슐화된다.

[0250] TPS(타일링 파라미터 세트)는 파라미터 세트 정보에 대한 하나의 NALU로서 고려되고, 일반적으로 일부 코덱-특정 샘플 엔트리에서, 예를 들어, 'stsd' 박스와 같이 샘플 설명에 전용인 박스들 중 하나에서 발견될 수 있는 DecoderConfigurationRecord 박스들과 같이 디코더 구성 또는 셋업 정보를 제공하는 메타데이터에 임베딩될 수 있다.

[0251] 대안적으로, 본 발명의 실시예에 따르면, TPS는 비디오 데이터에 대한 NAL 유닛(VCL NALU)으로서 처리될 수 있고, 'mdat' 박스에 하나의 샘플 데이터로서 저장될 수 있다. 이것은 또한 샘플 엔트리 및 샘플 데이터 모두에 존재할 수 있다. 프레임 부분 구성이 비디오 시퀀스를 따라 변경하는 경우, 샘플 설명 레벨(샘플 엔트리)에서보다는 샘플 레벨(샘플 데이터)에서 이를 저장하는 것이 더 편리하다.

[0252] 프레임 부분 파티셔닝 구성에서의 변경이 수신측에서 디코더를 리셋할 것을 요구할 때, ISOBMFF 기입기는 바람직하게는 샘플 엔트리 내의 비디오 비트스트림으로부터의 TPS 및 CTile 관련 정보를 저장한다. 이 디코더 리셋은 디바이스가 파일을 수신하거나 소비하는 것을 허용하여 새로운 파티셔닝 구성을 고려하게 한다. 새로운 파티셔닝 구성은 예를 들어 처리할 데이터(즉, 레벨들)의 양 또는 지원할 코딩 도구들(즉, 프로파일들)의 표시를 포함할 수 있다. 파티셔닝 구성을 위한 프로파일 및 레벨 값들 또는 다른 파라미터들에 따라, 디바이스는 새로운 파티셔닝 구성을 지원하거나 그렇지 않을 수 있다. 지원되지 않을 때, 디바이스는 전송을 적응시킬 수 있거나, 이용가능할 때 비디오의 대안 버전들을 선택할 수 있다. 새로운 구성이 지원될 때, 디바이스는 파일의 디코딩 및 렌더링을 계속한다.

[0253] ROI에 대한 공간적 액세스가 요구되는 경우, 테스트(1204) 후의 분지 "예"에서, ISOBMFF 기입기는 이용 사례에 따라 상이한 캡슐화 전략을 가질 수 있다. 공간적 액세스는 예를 들어 ROI 또는 파티션 기반 디스플레이(즉, ROI 또는 부분 또는 프레임 부분 또는 프레임 부분들의 세트에 대응하는 데이터의 일부만의 추출 및 디코딩) 또는 ROI 또는 부분 기반 스트리밍(즉, ROI 또는 부분 또는 프레임 부분들의 세트에 대응하는 데이터 및 메타데이터의 일부만을 전송함)을 의미한다. 예측된 이용 사례가 테스트(1205)가 참인 것(분지 "예")에 대응하는 로컬 디스플레이를 위한 저장인 경우, 이것은 하나의 트랙에 파티셔닝된 비디오 비트스트림을 저장하는 것이 편리할 수 있지만 공간적 액세스가 요구되는 ROI 또는 프레임 부분 또는 프레임 부분들의 세트에 대한 NALU 매핑을 포함한다. 단계(1206)에서 NALU 매핑이 생성된다. 이는, ISOBMFF 기입기의 경우, 비디오 비트스트림의 각각의 NALU에 대해 주어진 CTile에 관련된 NAL 유닛들을 리스팅하거나 동일한 랜덤 액세스 식별자(도 15에서의 RAID 참조(1502))(즉, 선택가능하고 디코딩가능한 프레임 부분 또는 공간적 영역에 대응함)를 갖는 것으로 구성된다. 리스팅을 수행할 수 있기 위해, ISOBMFF 기입기의 NALU 파서 모듈은, 비트스트림 생성에 대한 실시예에 따라, 단계(903)에서 공간적 액세스가 요구되는, ROI 또는 프레임 부분 또는 프레임 부분들의 세트에 할당된 식별자의 값을 체크한다.

[0254] 비트스트림이 공간적 액세스가 요구되는 ROI 또는 프레임 부분 또는 프레임 부분들의 세트에 대한 NALU 헤더 특정 식별자를 포함하지 않는 경우, ISOBMFF 기입기는 슬라이스 헤더 파서에게 903에서 할당된 CTile의 식별자의

값, 예를 들어, 도 13a에서 1301로 참조되는 ctile_unique_identifier를 얻도록 요구한다.

[0255] 그 후, NALUMapEntry 구조 'nalm'은 NALU의 리스트 및 프레임 부분들 또는 프레임 부분들의 세트에 대한 그 매핑을 저장하기 위한 'trak' 박스 계층구조 아래의 박스로서 단계(1206)에서 생성된다. 각각의 프레임 부분 또는 프레임 부분들의 세트에 대해, 유형 'trif'의 SampleGroupDescriptionBox는, 각각의 프레임 부분 또는 프레임 부분들의 세트에 대해, 예를 들어 ISO/IEC 14496-15로부터의 TileRegionGroupEntry의 파라미터들을 제공하는 프레임 부분 또는 프레임 부분들의 세트의 설명을 제공한다. 프레임 부분 설명자 'trif'의 group_ID 값은 캡슐화할 CTile의 식별자의 값으로 설정된다.

[0256] 이어서, 단계(1203)에서, 모든 프레임 부분 또는 프레임 부분들의 세트가 단일 트랙으로서 캡슐화된다. 테스트(1208)가 참인 것(분지 "예")에 대응하는 이용 사례가 스트리밍 중일 때, 비디오에서의 공간적 액세스 레벨에 대응하는 각각의 프레임 부분 또는 프레임 부분들의 세트를 전용 트랙으로 분할하는 것이 편리할 수 있고, 테스트(1208)가 참일 때, 단일 트랙 캡슐화가 수행된다.

[0257] 스트리밍 이용 사례의 경우, 각각의 프레임 부분 또는 프레임 부분들의 세트에 대해, 프레임 부분 설명이 NALU 매핑에 대한 단계(1209)에서 생성된다. 프레임 부분들 또는 프레임 부분들의 세트의 수는 TPS를 파싱함으로써 결정될 수 있다. 'trif' 샘플 그룹이 이용되며, 심지어 디폴트 샘플 그룹이 이용될 수 있는데, 그 이유는 단계(1210)에서 생성된 프레임 부분 트랙(하나의 프레임 부분 또는 프레임 부분들의 세트에 관련된 데이터를 캡슐화하기 위한 트랙)마다 하나의 프레임 부분 또는 프레임 부분들의 하나의 세트가 존재하기 때문이다. 그 후 모든 샘플들은 ISO/IEC 14496-15에 따라 프레임 부분 설명자 'trif'인 동일한 샘플 그룹 설명에 매핑된다. 프레임 부분 설명자 'trif'의 group_ID 값은 캡슐화할 CTile(또는 만약 있다면 RAID)의 식별자의 값으로 설정된다.

[0258] 그 다음, 단계(1210)에서, 각각의 프레임 부분 또는 프레임 부분들의 세트가 그 자신의 트랙, 프레임 부분 트랙에 삽입된다. 프레임 부분 트랙은 샘플들이 실제로 비디오의 공간적 부분들임을 나타내고, 테스트(1211)에서 캡슐화할 더 이상의 프레임 부분 또는 프레임 부분 세트가 남아 있지 않을 때 단계(1212)에서 생성된 프레임 부분 베이스 트랙을 참조하는 특정 샘플 엔트리들을 포함한다. 이 프레임 부분 베이스 트랙은 타일링 파라미터 세트(TPS)를 포함하는, 파라미터 세트들에 대응하는 특정 NAL 유닛들을 포함한다. 프레임 부분 베이스 트랙은 각각의 프레임 부분 트랙들을 특정 트랙 참조 유형으로 순서대로 참조하여, 프레임 부분 또는 프레임 부분들의 세트의 임의의 선택의 암시적 재구성을 허용한다. 단계(1212)는 복합 트랙으로 대체될 수 있으며, 추출기들이라고 불리는 NAL 유닛들은 하나 이상의 프레임 부분 트랙으로부터 명시적 재구성을 제공한다.

[0259] 이어서, 추출기들은 복합 트랙의 각각의 샘플에 대해, 간단히 추출기가 프레임 부분 또는 프레임 부분들의 세트의 주어진 식별자를 가리키게 하고, 전형적으로 대응하는 CTile(또는 만약 있다면 RAID)의 식별자를 참조함으로써 원래의 것과 상이하더라도, 프레임 부분들 또는 프레임 부분들의 세트의 임의의 배열을 허용한다.

[0260] 단계(1212)에서 복합 트랙을 이용할 때, 단계(1210)에서의 프레임 부분 트랙들은 실제로 디코딩가능한 프레임 부분 트랙들일 수 있으며, 이는 각각이 프레임 부분 설명(1209에서 생성됨) 및 파라미터 세트들을 포함한다는 것을 의미한다. 추출기가 상이하게 재조합할 수 있기 때문에, 각각의 프레임 부분 트랙에서의 TPS의 존재는 임의적이다. 그 다음, 샘플 설명은 이용 중인 코덱에 따르는 샘플 엔트리, 예를 들어, HEVC가 이용 중인 경우 'hvc1' 또는 'hvc2' 또는 AVC(Advanced Video Coding)가 이용 중인 경우 'avc1' 또는 'avc2', 또는 이용 중인 비디오 코더를 명확히 식별하는 임의의 예비된 4개의 문자 코더를 나타낼 수 있다.

[0261] 도 18은 본 발명의 하나 이상의 실시예의 구현을 위한 컴퓨팅 디바이스(1800)의 개략적인 블록도이다. 컴퓨팅 디바이스(1800)는, 마이크로컴퓨터, 워크스테이션 또는 경량의 휴대형 디바이스 등의 디바이스일 수 있다. 컴퓨팅 디바이스(1800)는 통신 버스를 포함하며, 이 통신 버스는,

[0262] - CPU로 표시된, 마이크로 프로세서 등의 중앙 처리 유닛(1801);

[0263] - 본 발명의 실시예들의 방법의 실행가능한 코드를 저장하기 위한, RAM으로 표시된, 랜덤 액세스 메모리(1802) 뿐만 아니라, 본 발명의 실시예들에 따른 방법을 구현하는데 필요한 변수들 및 파라미터들을 기록하도록 적응된 레지스터들 - 그 메모리 용량은 예를 들어 확장 포트에 접속된 임의적 RAM에 의해 확장될 수 있음 -;

[0264] - 본 발명의 실시예들을 구현하기 위한 컴퓨터 프로그램들을 저장하기 위한, ROM으로 표시된, 판독 전용 메모리(1803);

[0265] - 네트워크 인터페이스(1804) - 이 네트워크 인터페이스(1804)는 전형적으로, 처리될 디지털 데이터가 전송되거나 수신되는 통신 네트워크에 접속된다. 네트워크 인터페이스(1804)는 단일 네트워크 인터페이스이거나, 한 세

트의 상이한 네트워크 인터페이스들(예를 들어, 유선 및 무선 인터페이스들, 또는 상이한 종류들의 유선 또는 무선 인터페이스들)로 구성될 수 있다. 데이터 패킷들은 전송을 위해 네트워크 인터페이스에 기입되거나, CPU(1801)에서 실행 중인 소프트웨어 애플리케이션의 제어 하의 수신을 위해 네트워크 인터페이스로부터 판독됨 -;

[0266] - 사용자로부터 입력들을 수신하거나 사용자에게 정보를 표시하는 데 이용될 수 있는 사용자 인터페이스(1805);

[0267] - 대용량 저장 디바이스로서 제공될 수 있는, HD로 표시된 하드 디스크(1806);

[0268] - 비디오 소스 또는 디스플레이 등의 외부 디바이스로부터/로 데이터를 수신/전송하는 데 이용될 수 있는 I/O 모듈(1807)에 접속된다.

[0269] 실행가능한 코드는, 판독 전용 메모리(1803), 하드 디스크(1806) 또는 예를 들어 디스크 등의 이동식 디지털 매체에 저장될 수 있다. 변형예에 따르면, 프로그램들의 실행가능한 코드는, 네트워크 인터페이스(1804)를 통해, 통신 네트워크에 의해 수신되어, 실행되기 전에, 하드 디스크(1806) 등의 통신 디바이스(1800)의 저장 수단들 중 하나에 저장될 수 있다.

[0270] 중앙 처리 유닛(1801)은 본 발명의 실시예들에 따른 프로그램 또는 프로그램들의 소프트웨어 코드의 명령어들 또는 부분들의 실행을 제어하고 지시하도록 적응되고, 이러한 명령어들은 전술된 저장 수단들 중 하나에 저장된다. 전원이 켜진 후, CPU(1801)는, 예를 들어 프로그램 ROM(1803) 또는 하드 디스크(HD)(1806)로부터 이들 명령어들이 로딩된 후 소프트웨어 애플리케이션에 관련된 메인 RAM 메모리(1802)로부터의 명령어들을 실행할 수 있다. 이러한 소프트웨어 애플리케이션은, CPU(1801)에 의해 실행될 때, 본 발명의 순서도들의 단계들이 수행되게 한다.

[0271] 본 발명의 알고리즘들의 임의의 단계는, PC("Personal Computer"), DSP("Digital Signal Processor") 또는 마이크로제어기 등의 프로그래밍가능한 컴퓨팅 머신에 의한 한 세트의 명령어들 또는 프로그램의 실행에 의해 소프트웨어로 구현되거나, 아니면 FPGA("Field-Programmable Gate Array") 또는 ASIC("Application-Specific Integrated Circuit") 등의 전용 구성요소 또는 머신에 의해 하드웨어로 구현될 수 있다.

[0272] 본 발명이 위에서 특정 실시예들을 참조하여 설명되었지만, 본 발명은 특정 실시예들로 제한되지 않으며, 본 발명의 범위 내에 속하는 수정들이 관련 기술분야의 통상의 기술자에게 명백할 것이다.

[0273] 관련 기술분야의 통상의 기술자라면 예시 목적으로만 제공될 뿐, 첨부된 청구항들에 의해서만 결정되는 본 발명의 범위를 제한하도록 의도되지 않은 이러한 예시적인 실시예들을 참조하여 이들에 대해 많은 추가적인 수정들 및 변형들을 제안할 것이다. 특히, 상이한 실시예들로부터의 상이한 특징들이 적절한 경우에 상호교환될 수 있다.

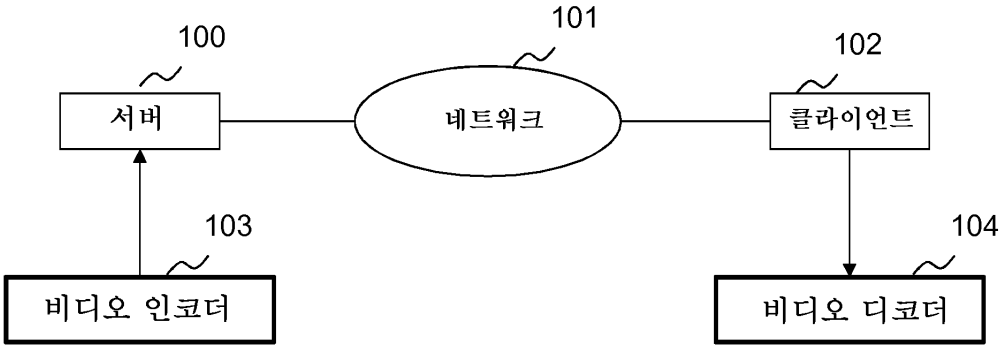
[0274] 전술한 본 발명의 실시예들 각각은 단독으로 또는 복수의 실시예들의 조합으로서 구현될 수 있다. 또한, 상이한 실시예들로부터의 특징들은 필요한 경우 또는 단일 실시예에서 개별 실시예들로부터의 요소들 또는 특징들의 조합이 유익한 경우에 조합될 수 있다.

[0275] (임의의 첨부된 청구항들, 요약서 및 도면들을 포함하여) 본 명세서에 개시된 각각의 특징은, 달리 명백히 언급되지 않는 한, 동일하거나 등가이거나 유사한 목적을 제공하는 대안적인 특징들로 대체될 수 있다. 따라서, 달리 명백히 언급되지 않는 한, 개시된 각각의 특징은 일반적인 일련의 등가이거나 유사한 특징들의 하나의 예일 뿐이다.

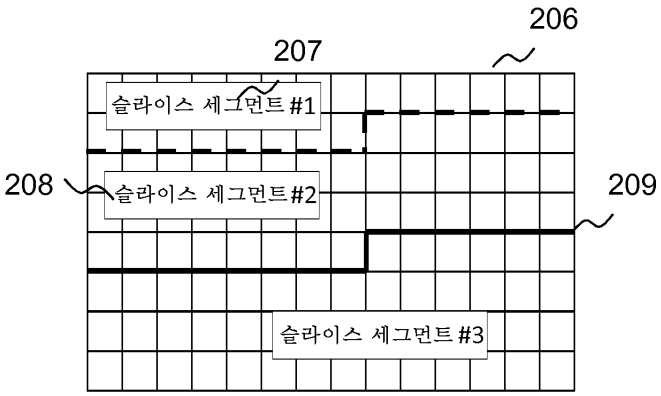
[0276] 청구항들에서, 단어 "포함하는"은 다른 요소들 또는 단계들을 배제하지 않고, 단수형은 복수를 배제하지 않는다. 상이한 특징들이 서로 상이한 종속항들에 인용되어 있다는 사실만으로는 이들 특징들의 조합이 유리하게 이용될 수 없다는 것을 나타내지 않는다.

도면

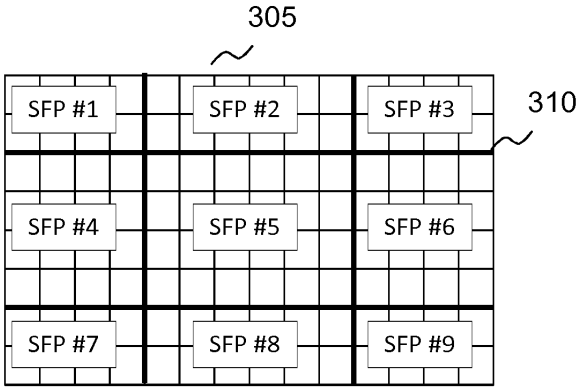
도면1



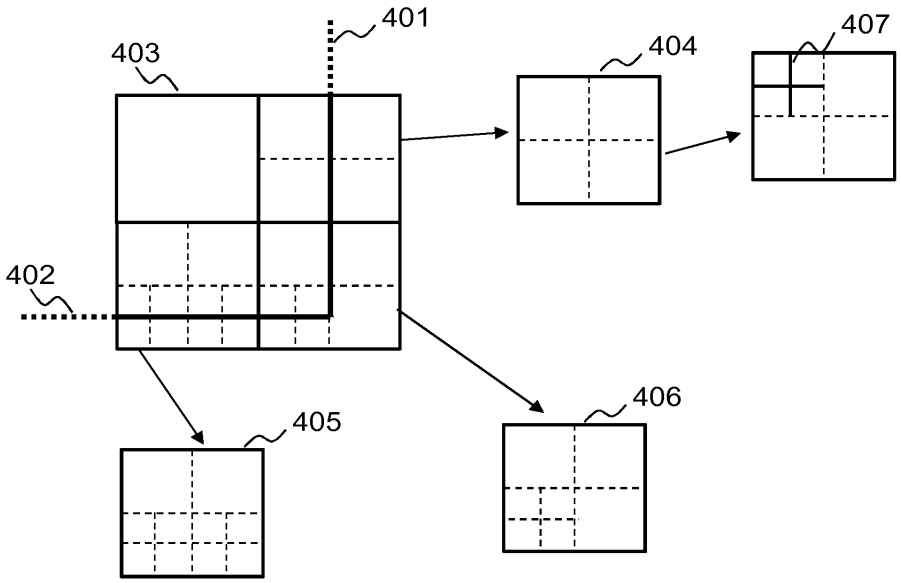
도면2



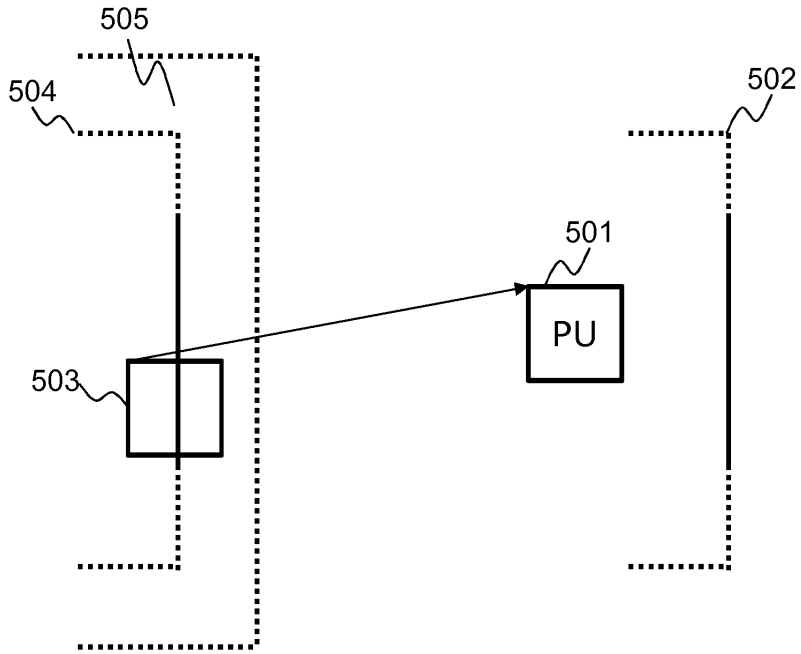
도면3



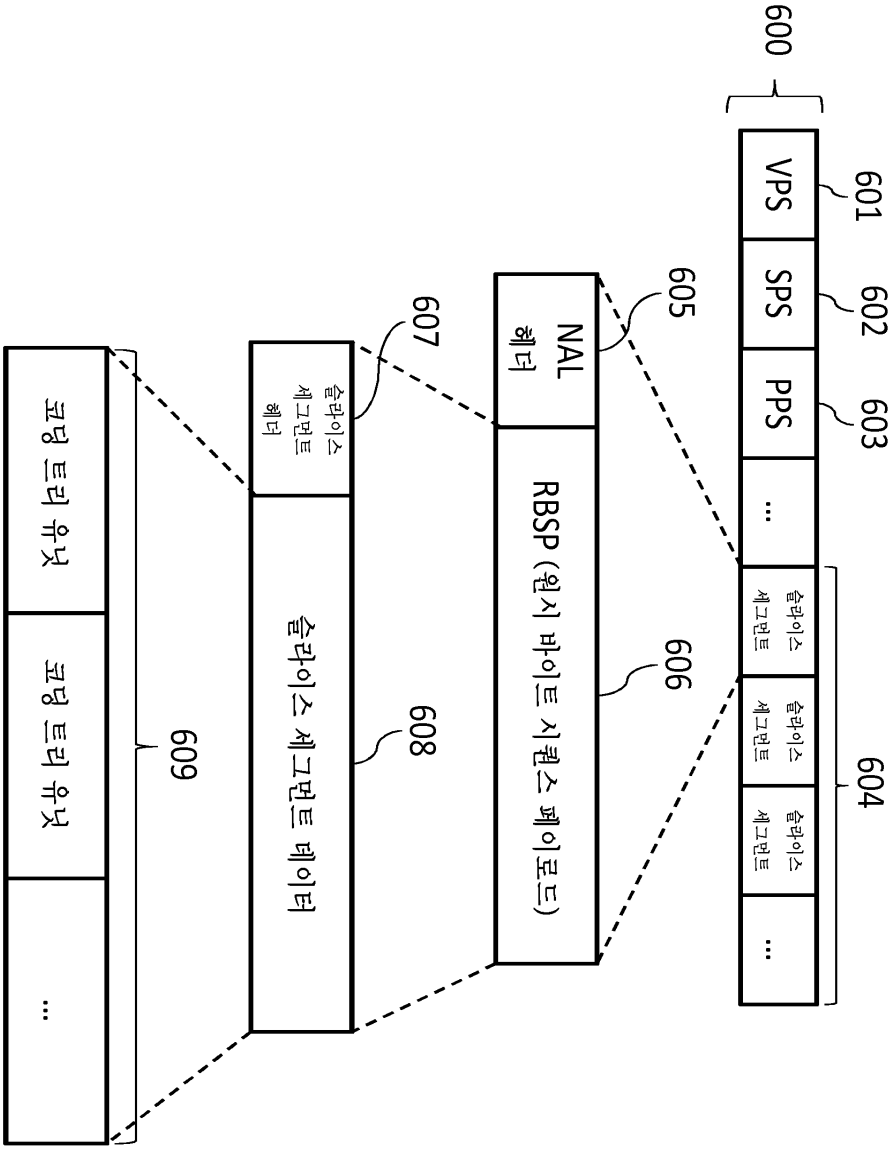
도면4



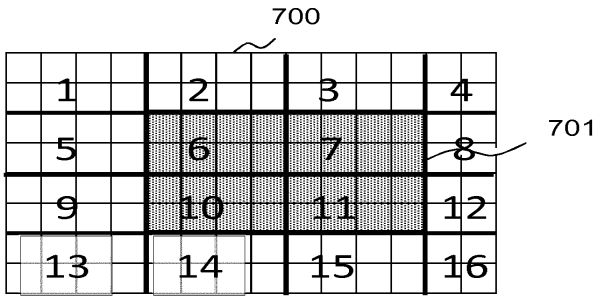
도면5



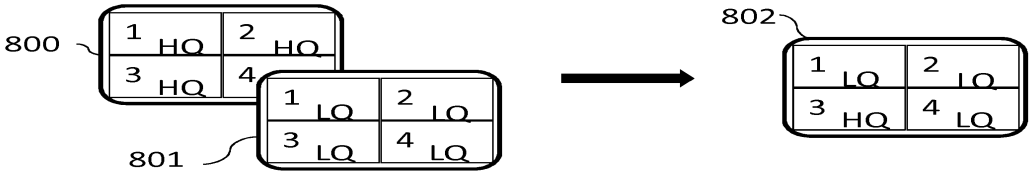
도면6


600
VPS
601
SPS
602
PPS
603
...
슬라이스 세그먼트
슬라이스 세그먼트
슬라이스 세그먼트
...
604
605
NAL 헤더
RBSP (원시 바이트 시퀀스 페이로드)
606
607
슬라이스 세그먼트 헤더
슬라이스 세그먼트 데이터
608
609
코딩 트리 유닛
코딩 트리 유닛
...

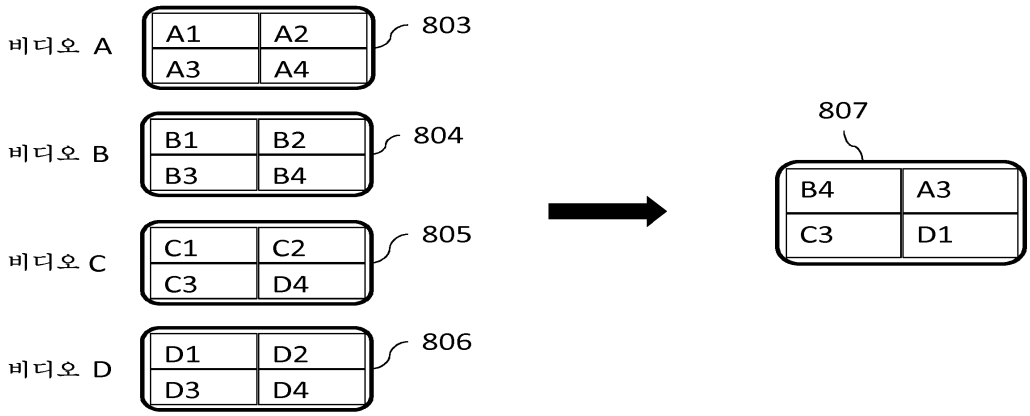
도면7


700
701
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

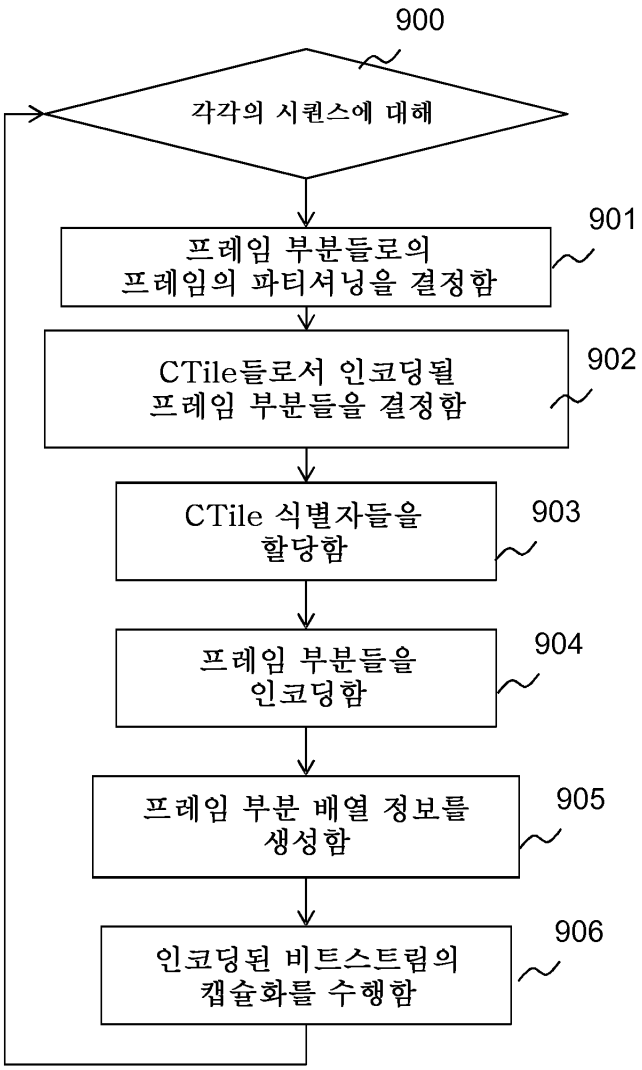
도면8a


800
1 HQ
2 HQ
3 HQ
4
1 LQ
2 LQ
3 LQ
4 LQ
801
802
1 LQ
2 LQ
3 HQ
4 LQ

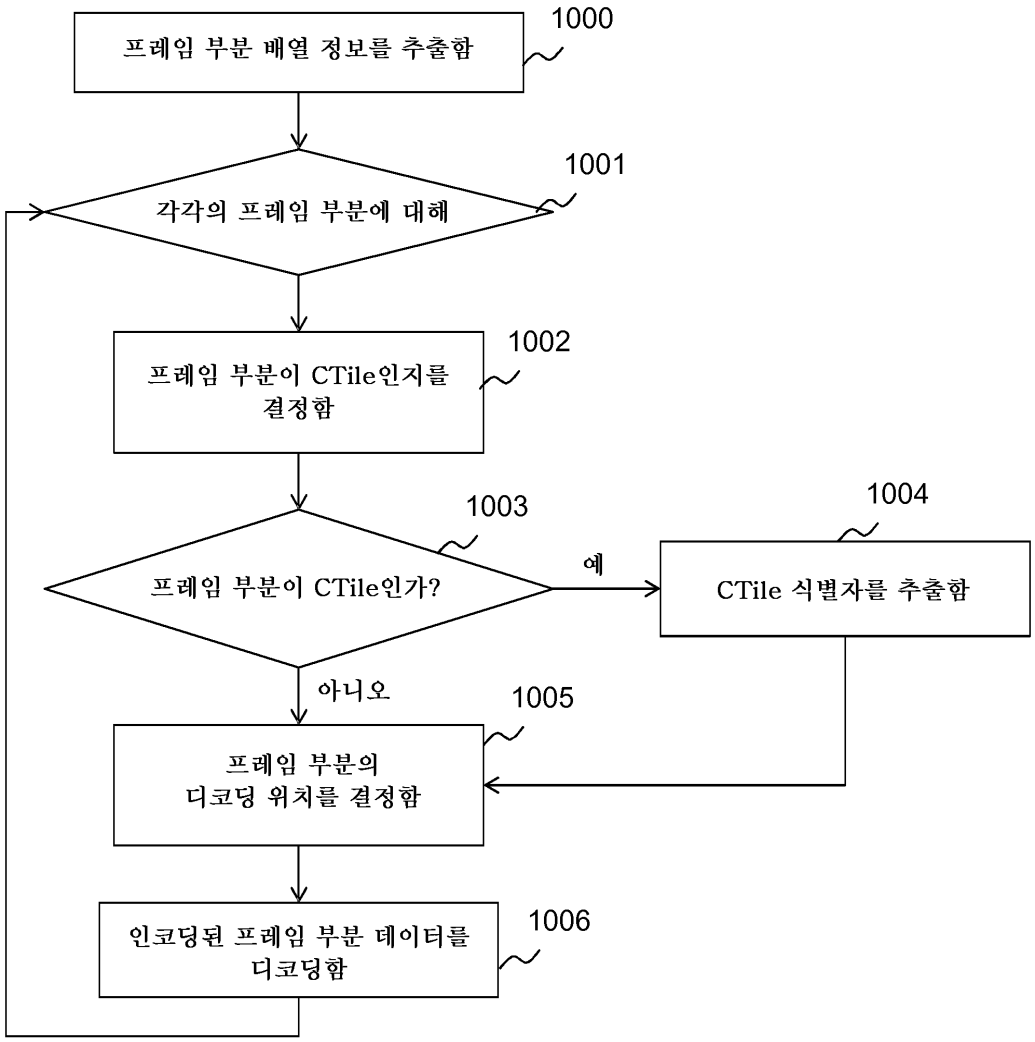
도면8b


비디오 A
A1
A2
A3
A4
803
비디오 B
B1
B2
B3
B4
804
비디오 C
C1
C2
C3
D4
805
비디오 D
D1
D2
D3
D4
806
807
B4
A3
C3
D1

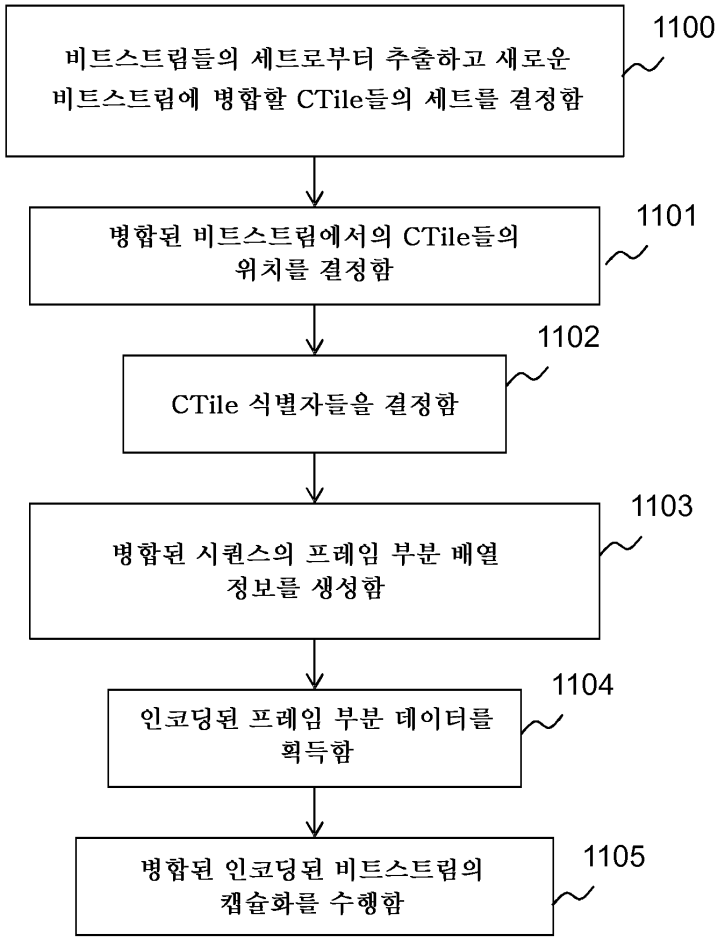
도면9


900
각각의 시퀀스에 대해
901
프레임 부분들로의 프레임의 파티셔닝을 결정함
902
CTile들로서 인코딩될 프레임 부분들을 결정함
903
CTile 식별자들을 할당함
904
프레임 부분들을 인코딩함
905
프레임 부분 배열 정보를 생성함
906
인코딩된 비트스트림의 캡슐화를 수행함

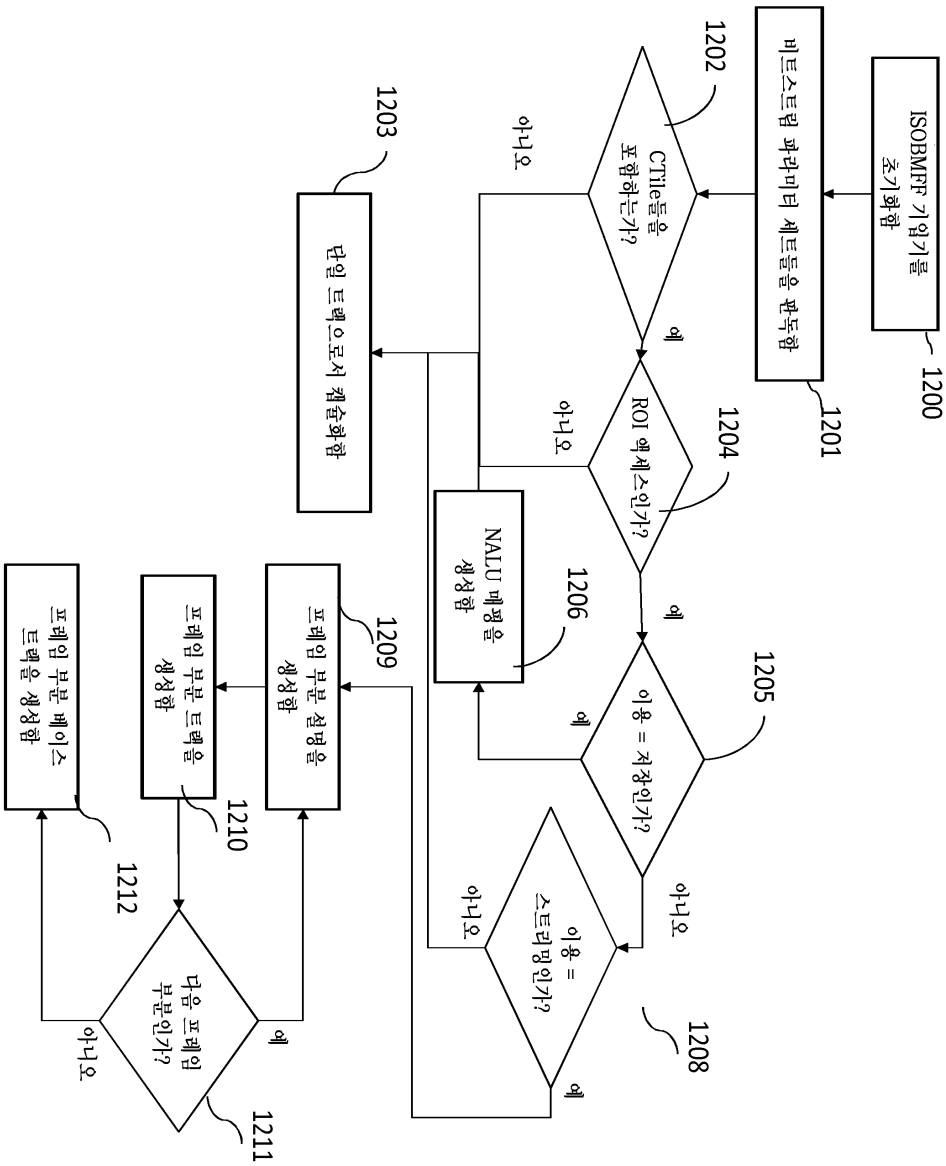
도면10


프레임 부분 배열 정보를 추출함
1000
각각의 프레임 부분에 대해
1001
프레임 부분이 CTile인지를 결정함
1002
프레임 부분이 CTile인가?
1003
예
CTile 식별자를 추출함
1004
아니오
프레임 부분의 디코딩 위치를 결정함
1005
인코딩된 프레임 부분 데이터를 디코딩함
1006

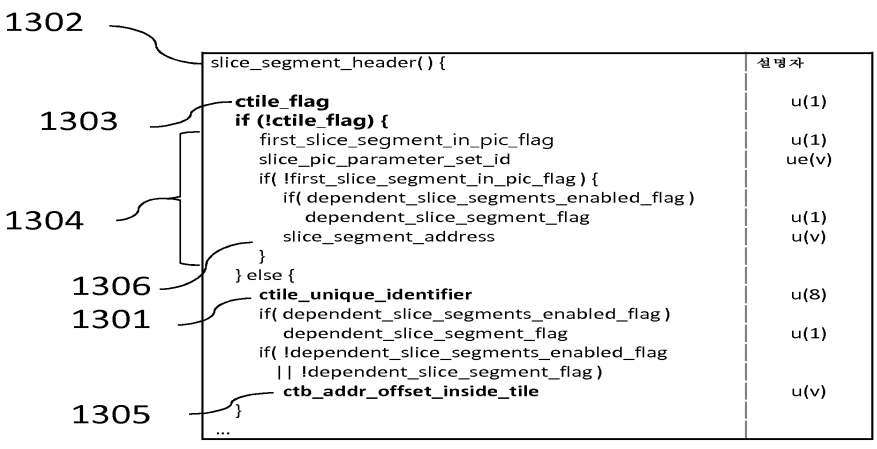
도면11


비트스트림들의 세트로부터 추출하고 새로운 비트스트림에 병합할 CTile들의 세트를 결정함
1100
병합된 비트스트림에서의 CTile들의 위치를 결정함
1101
CTile 식별자들을 결정함
1102
병합된 시퀀스의 프레임 부분 배열 정보를 생성함
1103
인코딩된 프레임 부분 데이터를 획득함
1104
병합된 인코딩된 비트스트림의 캡슐화를 수행함
1105

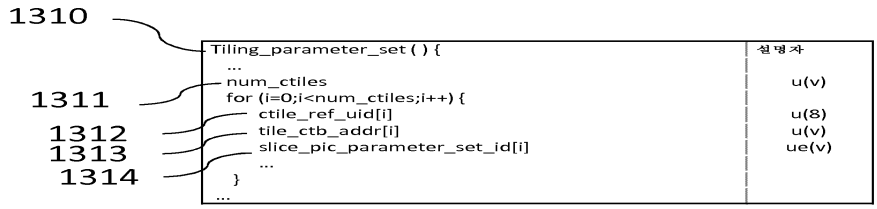
도면12


ISOBMFF 기입기를 초기화함
1200
비트스트림 파라미터 세트들을 판독함
1201
CTile들을 포함하는가?
1202
예
아니오
단일 트랙으로서 캡슐화함
1203
ROI 액세스인가?
1204
예
아니오
NALU 매핑을 생성함
1206
이용 = 저장인가?
1205
예
아니오
이용 = 스트리밍인가?
1208
예
아니오
프레임 부분 설명을 생성함
1209
프레임 부분 트랙을 생성함
1210
프레임 부분 베이스 트랙을 생성함
1212
다음 프레임 부분인가?
1211
예
아니오

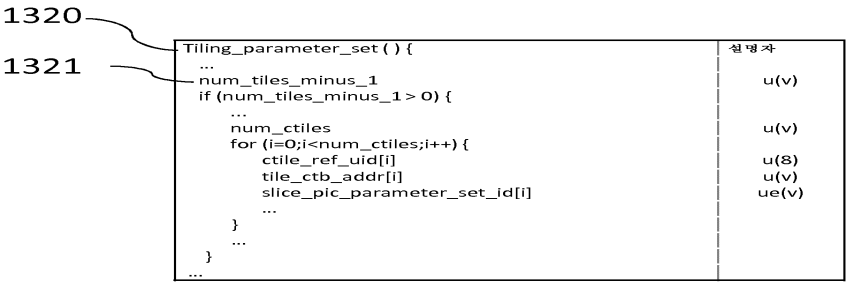
도면13a


1302
1303
1304
1306
1301
1305
slice_segment_header( ) {
ctile_flag
if (!ctile_flag) {
first_slice_segment_in_pic_flag
slice_pic_parameter_set_id
if(!first_slice_segment_in_pic_flag) {
if(dependent_slice_segments_enabled_flag)
dependent_slice_segment_flag
slice_segment_address
}
} else {
ctile_unique_identifier
if(dependent_slice_segments_enabled_flag)
dependent_slice_segment_flag
if(!dependent_slice_segments_enabled_flag
|| !dependent_slice_segment_flag)
ctb_addr_offset_inside_tile
}
...
설명자
u(1)
u(1)
ue(v)
u(1)
u(v)
u(8)
u(1)
u(v)

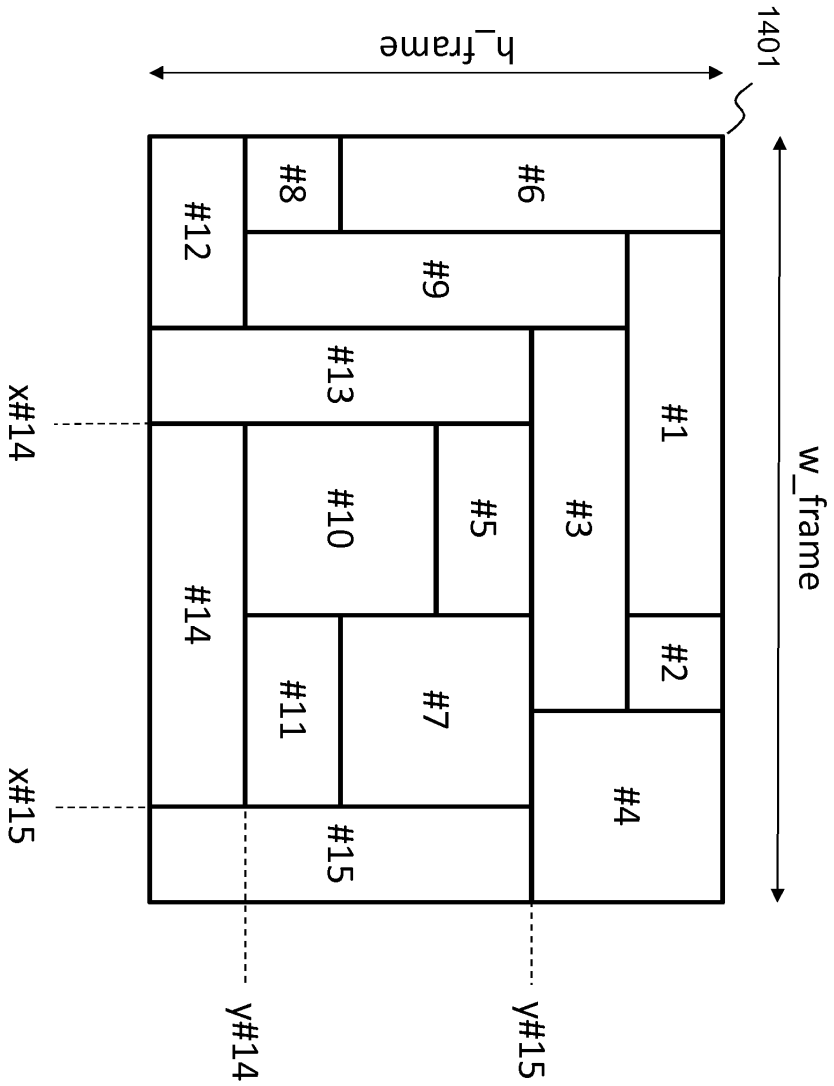
도면13b


1310
1311
1312
1313
1314
Tiling_parameter_set ( ) {
...
num_ctiles
for (i=0;i<num_ctiles;i++) {
ctile_ref_uid[i]
tile_ctb_addr[i]
slice_pic_parameter_set_id[i]
...
}
...
설명자
u(v)
u(8)
u(v)
ue(v)

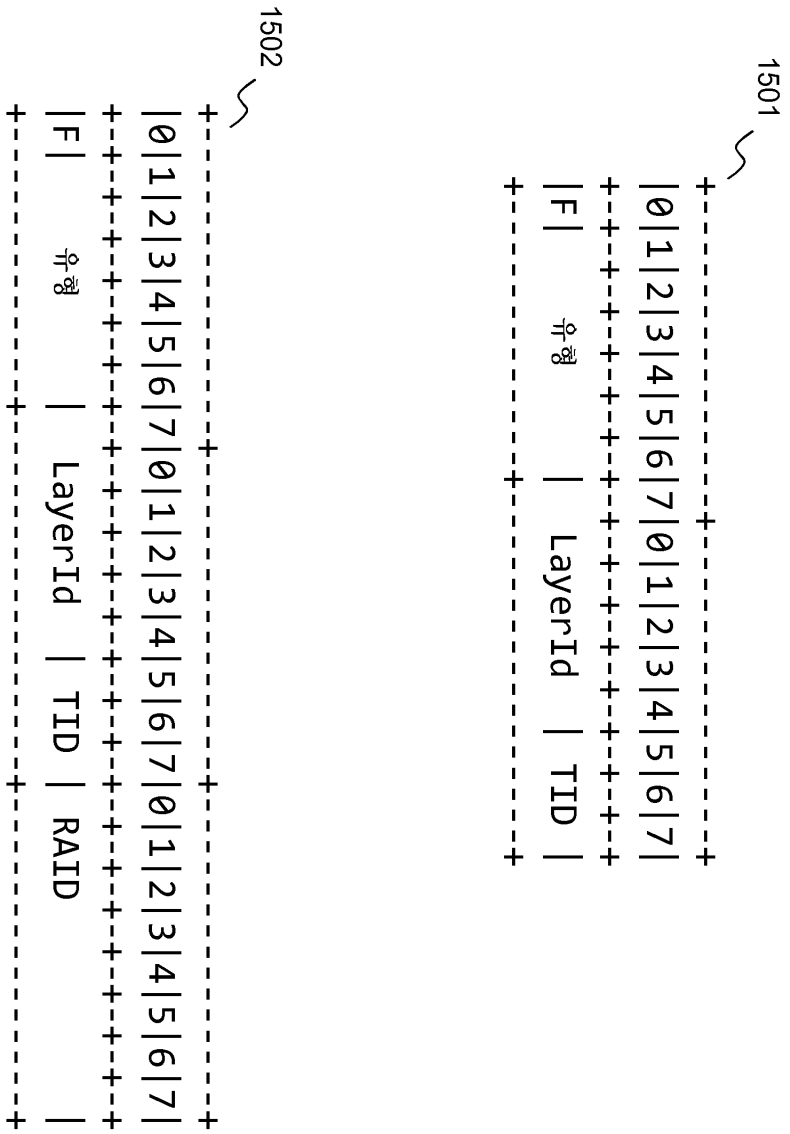
도면13c


1320
1321
Tiling_parameter_set ( ) {
...
num_tiles_minus_1
if (num_tiles_minus_1 > 0) {
...
num_ctiles
for (i=0;i<num_ctiles;i++) {
ctile_ref_uid[i]
tile_ctb_addr[i]
slice_pic_parameter_set_id[i]
...
}
...
}
...
설명자
u(v)
u(v)
u(8)
u(v)
ue(v)

도면14



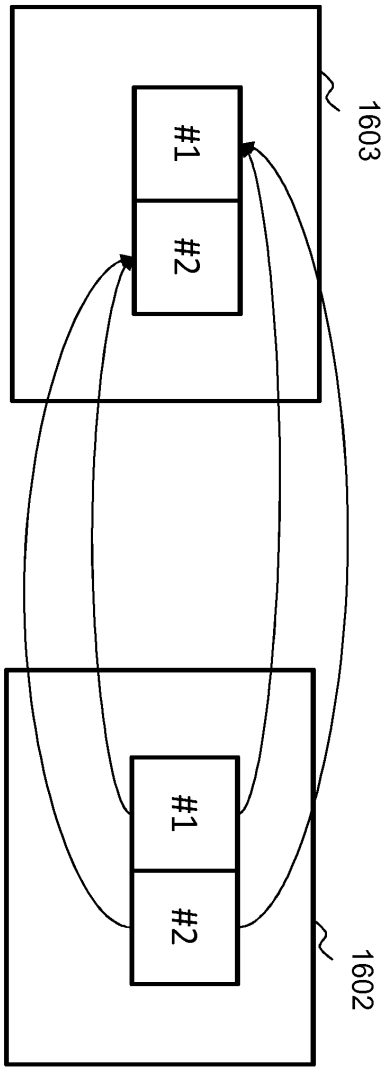
도면15



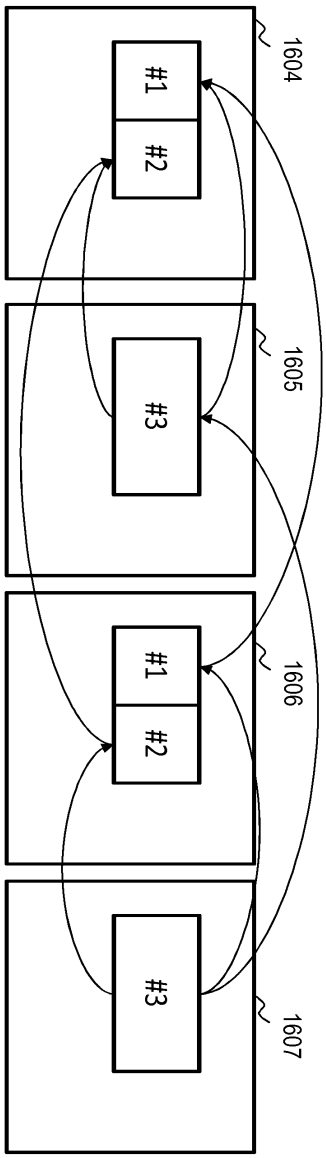
도면16a

Tiling_parameter_set ( ) {	설명자
...	
num_tiles_minus_1	u(v)
if (num_tiles_minus_1 > 0) {	
...	
num_ctiles	u(v)
for (i=0;i<num_ctiles;i++) {	
1600 ctile_ref_uid[i]	u(8)
...	
num_ctile_dependencies	u(v)
for (j=0;j<num_ctile_dependencies;j++) {	
1601 ctile_dep_uid[j]	u(8)
}	
...	
}	
...	
}	
...	

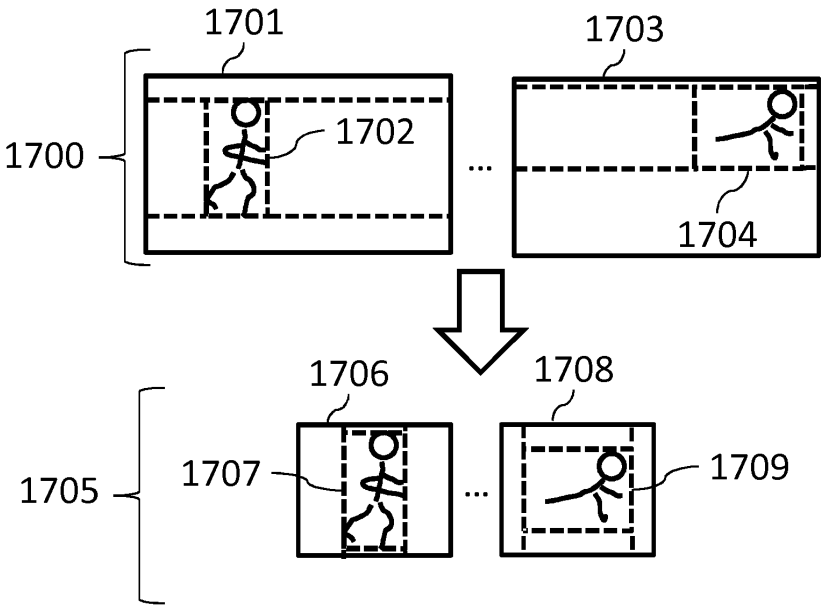
도면16b



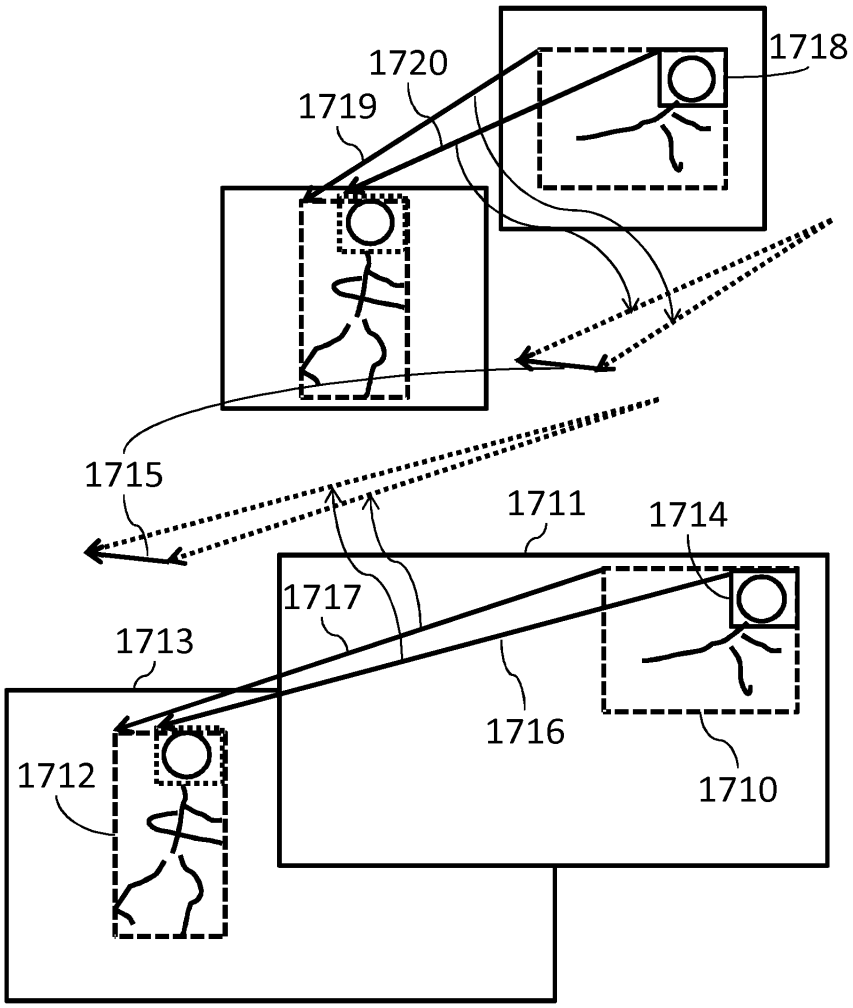
도면16c



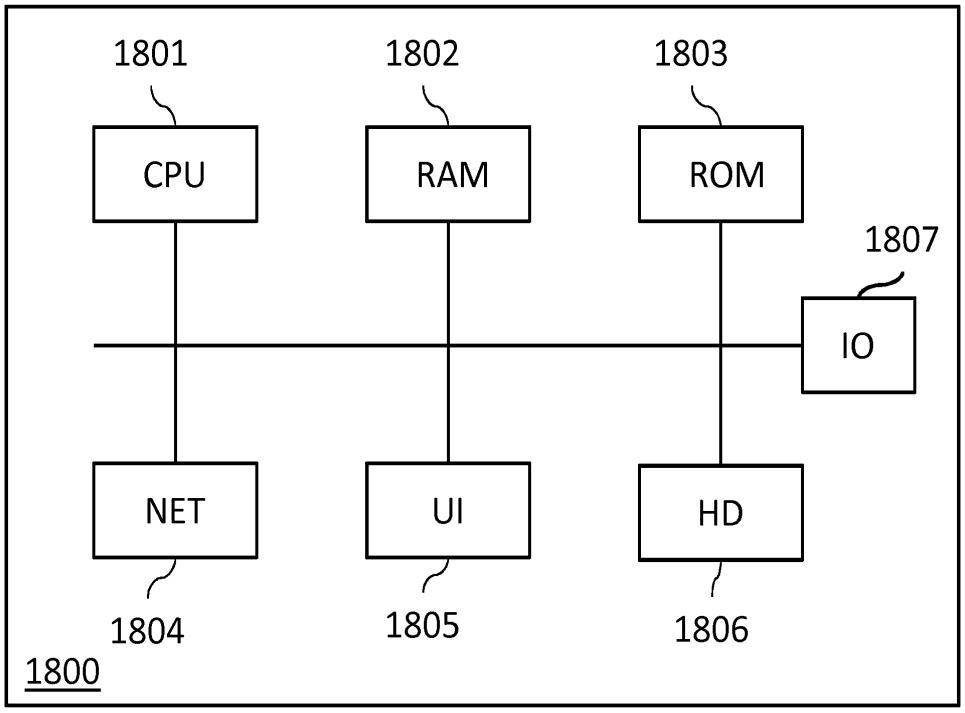
도면17a


1701
1703
1700
1702
1704
1706
1708
1705
1707
1709

도면17b


1718
1720
1719
1715
1711
1714
1717
1713
1716
1712
1710

도면18


1801
1802
1803
CPU
RAM
ROM
1807
IO
NET
UI
HD
1804
1805
1806
1800