(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년10월30일
(11) 등록번호 10-2038783
(24) 등록일자 2019년10월24일

(51) 국제특허분류(Int. Cl.)
H04N 21/236 (2011.01) *H04N 19/17* (2014.01)
H04N 19/46 (2014.01) *H04N 21/84* (2011.01)
H04N 21/845 (2011.01) *H04N 21/854* (2011.01)
(52) CPC특허분류
H04N 21/23614 (2013.01)
H04N 19/17 (2015.01)
(21) 출원번호 10-2018-7000514
(22) 출원일자(국제) 2016년06월08일
심사청구일자 2018년01월10일
(85) 번역문제출일자 2018년01월08일
(65) 공개번호 10-2018-0016519
(43) 공개일자 2018년02월14일
(86) 국제출원번호 PCT/EP2016/063035
(87) 국제공개번호 WO 2016/202664
국제공개일자 2016년12월22일
(30) 우선권주장
1510608.1 2015년06월16일 영국(GB)
(56) 선행기술조사문헌
KR1020080048054 A*
KR1020110015438 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
캐논 가부시끼가이샤
일본 도쿄도 오오따꾸 시모마루꼬 3쪼메 30방 2고
(72) 발명자
마즈 프레데릭
프랑스, 35850 랑강, 뤼 데 띠옐 6
드누알 프랑크
프랑스, 35190 생 도미누크, 라 빌 에스 레 8
(뒷면에 계속)
(74) 대리인
장수길, 이중희

전체 청구항 수 : 총 15 항 심사관 : 옥윤철

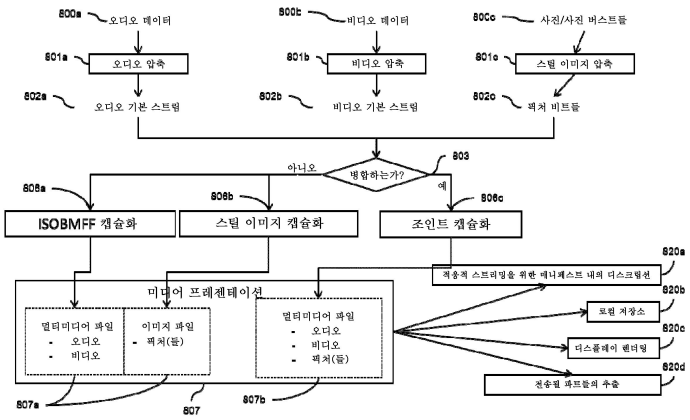
(54) 발명의 명칭 **이미지 데이터 캡슐화**

(57) 요 약

하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 방법이 제공되고, 캡슐화된 비트스트림은 데이터 파트 및 메타데이터 파트를 포함한다. 본 방법은: - 서브 이미지 또는 단일 이미지의 이미지 및/또는 단일 이미지들의 세트를 표현하는 데이터 파트의 한 부분을 식별해주는 이미지 아이템 정보를 제공하는 단계; - 하

(뒷면에 계속)

대 표 도



나 이상의 이미지들에 관련된 변환 연산자들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 디스크립션 정보를 제공하는 단계; 및 - 상기 비트스트림을 상기 제공된 정보와 함께 캡슐화된 데이터 파일로서 출력하는 단계를 포함한다. 상기 이미지 아이템 정보는 고려된 서브 이미지 또는 단일 이미지 또는 단일 이미지들의 세트에 전용된 이미지 디스크립션 정보의 적어도 일부를 포함하는 하나 이상의 속성들을 포함하고, 상기 이미지 디스크립션 정보는 하나 이상의 박스들에 정의된다.

(52) CPC특허분류
H04N 19/46 (2015.01)
H04N 21/84 (2013.01)
H04N 21/8455 (2013.01)
H04N 21/85406 (2013.01)

(72) 발명자
우드라오고 나엘
프랑스, 35330 모르 드 브르따뉴, 꾸뚜즈
르 훼브르 쟝
프랑스 94230 까샹 아브뉘 뒤 프레지덩 윌슨 7 에스끄 베1
꽁꼴라또 씨릴
프랑스 77380 꽁 라 빌 뤼 드 라 헤르스 18

명 세 서

청구범위

청구항 1

하나 이상의 이미지들에 기초하여 이미지 파일을 생성하는 방법으로서, 상기 이미지 파일은 ISOBMFF 표준에 따르는 비트스트림을 캡슐화하고, 상기 방법은,

- 상기 하나 이상의 이미지들을 입력하는 단계;

- 상기 하나 이상의 이미지들 각각에 관련된 속성을 식별하는 단계; 및

- 적어도 (i) 식별되는 하나 이상의 속성들 - 상기 하나 이상의 속성들은 동일한 메타데이터 구조에 리스트됨(listed) -; (ii) 상기 하나 이상의 속성들의 각각의 속성의 식별 정보와 상기 하나 이상의 이미지들의 각각의 이미지의 식별 정보를 연관시키기 위한 연관 정보; 및 (iii) 상기 이미지들을 표현하는 미디어 데이터를 포함하는, 상기 비트스트림을 캡슐화하는 이미지 파일을 생성하는 단계를 포함하는, 방법.

청구항 2

제1항에 있어서,

- 상기 이미지 파일은 상기 하나 이상의 이미지들의 각각의 이미지에 대한 정보를 표현하기 위한 "ItemInfoBox" 메타데이터 구조를 더 포함하고,

- 상기 하나 이상의 속성들의 각각의 속성의 상기 식별 정보와 상기 하나 이상의 이미지들의 각각의 이미지의 상기 식별 정보를 연관시키기 위한 상기 연관 정보는 상기 동일한 메타데이터 구조와 상이한 미리 결정된 박스에 기술되는, 방법.

청구항 3

제2항에 있어서, 상기 하나 이상의 이미지들의 폭 및 높이를 표시하는 "ispe"는 상기 "ItemInfoBox" 메타데이터 구조와 상이한 상기 동일한 메타데이터 구조에서 상기 속성으로서 기술되고, 상기 연관 정보는 상기 하나 이상의 이미지들의 각각의 이미지의 상기 식별 정보와 하나 이상의 "ispe"의 각각의 "ispe"의 식별 정보를 연관시키는, 방법.

청구항 4

제2항에 있어서, 상기 하나 이상의 이미지들의 디코더 구성을 표시하는 "hvcC"는 상기 "ItemInfoBox" 메타데이터 구조와 상이한 상기 동일한 메타데이터 구조에서 상기 속성으로서 기술되고, 상기 연관 정보는 상기 하나 이상의 이미지들의 각각의 이미지의 상기 식별 정보와 하나 이상의 "hvcC"의 각각의 "hvcC"의 식별 정보를 연관시키는, 방법.

청구항 5

제2항에 있어서, 상기 연관 정보는 식별되는 하나 이상의 속성들 중 하나의 속성의 식별 정보와 2개 이상의 이미지들에 대응하는 식별 정보 간의 대응 관계를 표현하는, 방법.

청구항 6

제3항에 있어서, 상기 이미지 파일은 복수의 단일 이미지들에 기초하여 생성되는, 방법.

청구항 7

제3항에 있어서, 상기 이미지 파일은 단일 이미지에 대응하는 복수의 서브 이미지(sub-image)들에 기초하여 생성되는, 방법.

청구항 8

제3항에 있어서, 상기 이미지 파일은 메타데이터 파트(metadata part) 및 미디어 데이터 파트(media data part)를 포함하고, 상기 하나 이상의 속성들 및 상기 연관 정보 양자 모두는 상기 메타데이터 파트에 기술되는, 방법.

청구항 9

제1항에 있어서, 상기 식별된 속성들이 순서화(order)되고, 상기 하나 이상의 속성들의 각각의 속성의 상기 식별 정보는 그것의 순서에 대응하는, 방법.

청구항 10

하나 이상의 이미지들에 기초하여 이미지 파일을 생성하기 위한 디바이스로서, 상기 이미지 파일은 ISOBMFF 표준에 따르는 비트스트림을 캡슐화하고, 상기 디바이스는,

- 상기 하나 이상의 이미지들을 입력하기 위한 입력 모듈;

- 상기 하나 이상의 이미지들 각각에 관련된 속성을 식별하기 위한 식별기 모듈; 및

- 적어도 (i) 식별되는 하나 이상의 속성들 - 상기 하나 이상의 속성들은 동일한 메타데이터 구조에 리스트됨(listed) -; (ii) 상기 하나 이상의 속성들의 각각의 속성의 식별 정보와 상기 하나 이상의 이미지들의 각각의 이미지의 식별 정보를 연관시키기 위한 연관 정보; 및 (iii) 상기 이미지들을 표현하는 미디어 데이터를 포함하는, 상기 이미지들의 비트스트림을 캡슐화하는 이미지 파일을 생성하기 위한 생성기 모듈을 포함하는, 디바이스.

청구항 11

제10항에 있어서,

- 상기 이미지 파일은 상기 하나 이상의 이미지들의 각각의 이미지에 대한 정보를 표현하기 위한 "ItemInfoBox" 메타데이터 구조를 더 포함하고,

- 상기 하나 이상의 속성들의 각각의 속성의 상기 식별 정보와 상기 하나 이상의 이미지들의 각각의 이미지의 상기 식별 정보를 연관시키기 위한 상기 연관 정보는 상기 동일한 메타데이터 구조와 상이한 미리 결정된 박스에 기술되는, 디바이스.

청구항 12

제11항에 있어서, 상기 하나 이상의 이미지들의 폭 및 높이를 표시하는 "ispe"는 상기 "ItemInfoBox" 메타데이터 구조와 상이한 상기 동일한 메타데이터 구조에서 상기 속성으로서 기술되고, 상기 연관 정보는 상기 하나 이상의 이미지들의 각각의 이미지의 상기 식별 정보와 하나 이상의 "ispe"의 각각의 "ispe"의 식별 정보를 연관시키는, 디바이스.

청구항 13

제11항에 있어서, 상기 하나 이상의 이미지들의 디코더 구성을 표시하는 "hvcC"는 상기 "ItemInfoBox" 메타데이터 구조와 상이한 상기 동일한 메타데이터 구조에서 상기 속성으로서 기술되고, 상기 연관 정보는 상기 하나 이상의 이미지들의 각각의 이미지의 상기 식별 정보와 하나 이상의 "hvcC"의 각각의 "hvcC"의 식별 정보를 연관시키는, 디바이스.

청구항 14

제11항에 있어서, 상기 연관 정보는 식별되는 하나 이상의 속성들 중 하나의 속성의 식별 정보와 2개 이상의 이미지들에 대응하는 식별 정보 간의 대응 관계를 표현하는, 디바이스.

청구항 15

제1항 내지 제9항 중 어느 한 항에 따른 방법을 구현하기 위한 컴퓨터 프로그램의 명령어들을 저장하는, 컴퓨터 판독가능 저장 매체.

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

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

발명의 설명

기 술 분 야

[0001] 본 발명은, 스틸 이미지들, 스틸 이미지들의 버스트들 또는 비디오 데이터와 같은, 이미지 데이터를 기술적 메타데이터(descriptive metadata)와 함께 미디어 컨테이너에 저장하는 것에 관한 것이다. 이러한 메타데이터는 일반적으로 이미지 데이터 및 이미지 데이터의 부분들에의 용이한 액세스를 제공한다.

배 경 기 술

[0002] 이 섹션에서 설명되는 접근법들 중 일부가 추구될 수는 있지만, 꼭 이전에 생각되었거나 추구되었던 접근법들인 것은 아니다. 따라서, 이 섹션에서 설명되는 접근법들이 꼭 본 출원에서의 청구항들에 대한 종래 기술인 것은 아니고, 이 섹션에 포함되어 있다는 것에 의해 종래 기술인 것으로 인정되지 않는다.

[0003] HEVC 표준은 스틸 이미지(still image)들의 인코딩을 위한 프로파일을 정의하고 단일 스틸 이미지들 또는 스틸 이미지들의 버스트(burst)들을 압축하기 위한 특정 도구들을 설명한다. 이러한 종류의 이미지 데이터에 사용되는 ISO 베이스 미디어 파일 포맷(ISO Base Media File Format)(ISOBMFF)의 확장이 ISO/IEC 23008 표준, Part 12에 "Image File Format"이라는 이름으로 포함시키기 위해 제안되었다. 이 표준은 상이한 사용 사례들에 대응하는 두 가지 형태의 저장을 다루고 있다:

[0004] - 디코더에서 임의로 사용되는 타이밍을 갖고 이미지들이 다른 이미지들에 종속적일 수 있는, 이미지 시퀀스들의 저장, 및

[0005] - 단일 이미지들, 및 독립적으로 코딩된 이미지들의 컬렉션들의 저장.

[0006] 첫 번째 경우에, 캡슐화는 비디오 트랙들을 ISO 베이스 미디어 파일 포맷으로 캡슐화에 하는 것에 가깝고(문서 << Information technology - Coding of audio-visual objects - Part 12: ISO base media file format>>, ISO/IEC 14496-12:2014, Fifth edition, Avril 2015를 참조), 'trak' 박스들 및 디스크립션(description)을 위한 샘플 그룹화(sample grouping)와 같은, 동일한 도구들 및 개념들이 사용된다. 'trak' 박스는 트랙, 즉 관련 샘플들의 시간 설정된 시퀀스(timed sequence)를 기술하기 위한 서브 박스(sub box)들을 포함하는 파일 포맷 박스(file format box)이다.

[0007] 두 번째 경우에, ISOBMFF 박스들의 세트인, 'meta' 박스들이 사용된다. 이 박스들 및 그들의 계층구조(hierarchy)는 'track' 박스보다 더 적은 디스크립션 도구(description tool)들을 제공하며, 관련 샘플들 대신에 "정보 아이템(information item)들" 또는 "아이템들"에 관련되어 있다.

[0008] 멀티미디어 파일들을 로컬적으로 디스플레이하기 위해 또는 멀티미디어 프레젠테이션(multimedia presentation)들을 스트리밍하기 위해 이미지 파일 포맷이 사용될 수 있다. HEVC 스틸 이미지들은 많은 문제들을 제기하는 많은 적용분야들을 갖는다.

[0009] 이미지 버스트들은 하나의 적용분야이다. 이미지 버스트들은 카메라에 의해 캡처되어 단일 표현으로서 저장되는 스틸 픽처(still picture)들의 시퀀스들이다(많은 픽처 아이템들이 데이터 블록을 참조함). 사용자들은 이 픽처들에 대해 몇 가지 유형들의 행동들: 하나의 픽처를 섬네일(thumbnail) 또는 커버(cover)로서 선택하는 것, 이 픽처들에 효과들을 적용하는 것 등을 수행하고자 할 수 있다.

[0010] 따라서, 픽처들의 리스트를 데이터 블록 내의 그들의 대응하는 바이트들을 사용해 식별하기 위한 기술적 메타데이터가 필요하다.

[0011] 컴퓨터 사진(computational photography)은 다른 적용분야이다. 컴퓨터 사진에서, 사용자들은 동일한 픽처의 상이한 해상도들(상이한 노출들, 상이한 초점들 등)에 액세스할 수 있다. 이 상이한 해상도들은, 해상도가 선택될 수 있고 대응하는 데이터 조각(piece of data)이 처리(렌더링, 편집, 전송 등)를 위해 위치확인(locate)되고 추출될 수 있도록, 메타데이터로서 저장되어야만 한다.

[0012] 크기의 면에서의 픽처 해상도의 증가에 따라, 따라서 이 큰 픽처들의 몇몇 공간 파트(spatial part)들만이 쉽게 식별되고 추출될 수 있도록 충분한 디스크립션을 제공할 필요가 있다.

[0013] 다른 종류의 적용분야는, 예를 들어, 비디오 요약을 위해, 비디오 시퀀스 중의 특정 픽처들에 액세스하는 것, 비디오 감시 데이터 내의 증거 이미지들 등에 액세스하는 것이다.

[0014] 이러한 종류의 적용분야들의 경우, 압축된 비디오 데이터 및 비디오 트랙 메타데이터 외에도, 키 이미지(key image)들에 쉽게 액세스할 수 있게 하는 이미지 메타데이터가 필요하다.

[0015] 그에 부가하여, 전문가용 카메라들은 높은 공간 해상도들에 도달했다. 4K2K 해상도의 비디오들 또는 이미지들이 이제 흔히 볼 수 있다. 8k4k 비디오들 또는 이미지들조차도 이제 흔히 볼 수 있게 되고 있다. 이와 동시에, 비디오 스트리밍 능력을 갖춘 연결된 모바일 디바이스들에서 비디오가 점점 더 많이 재생된다. 따라서, 모바일 디바이스의 사용자가 품질을 유지하거나 심지어 개선시키는 것에 의해 비디오의 서브 파트(sub-part)들을 디스플레이하고자 하거나 그 서브 파트들에 집중하고자 하는 경우 비디오들을 타일(tile)들로 분할(split)하는 것이 중요하게 된다. 타일들을 사용하는 것에 의해, 사용자는 따라서 비디오의 공간적 서브 파트(spatial sub-part)들을 대화식으로(interactively) 요청할 수 있다.

[0016] 따라서, 메타데이터 박스들을 단순히 파싱하는 것 이외에 부가의 처리 없이 액세스가능하기 위해, 비디오의 이 공간적 서브 파트들을 컴팩트한 방식으로 파일 포맷에 기술할 필요가 있다. 그렇게 기술된 비디오들에 대응하

는 이미지들에 대해, 사용자가 공간적 서브 파트들에 액세스하는 것이 또한 관심 대상이다.

[0017] 그에 부가하여, 사용자는 보통 새로운 파생 이미지(derived image)들을 생성하기 위해 이미지들을 변환(transform)하거나 합성(compose)한다. 이 파생 이미지들은, 회전 또는 클리핑과 같은, 하나 이상의 지정된 연산(operation)들을 다른 이미지들 또는 이미지들의 세트에 적용하여 획득된다.

[0018] 따라서, 원본 이미지들로부터 파생 이미지들을 검색하기 위해 하나 이상의 입력 이미지들에 적용될 연산들을 메타데이터로서 파일 포맷에 기술할 필요가 있다.

[0019] ISO/IEC 23008-12 표준은 최근 논의된 스틸 이미지들을 파일 포맷 내에 캡슐화하기 위한 두 가지 방식들을 다루고 있다.

[0020] 하나의 방식은 'track' 박스들, 및 연관된 디스크립션 도구들과 함께 관련 샘플들의 시간 설정된 시퀀스의 개념에 기반한 것이고, 다른 방식은 특히 관심 영역 디스크립션 및 타일링 지원을 위해, 보다 적은 디스크립션 도구들을 제공하는, 샘플들 대신에, 정보 아이템들에 기초한, 'meta' 박스들에 기초한다.

[0021] 따라서 새로운 이미지 파일 포맷(Image File Format)에서 타일링 지원을 제공할 필요가 있다.

[0022] 특히 압축 시에, 타일들을 사용하는 것이 종래 기술에 일반적으로 공지되어 있다. ISO 베이스 미디어 파일 포맷에서의 그들의 인덱스화(indexation)와 관련하여, ISO/IEC 14496 표준의 Part 15의 개정 초안 "Carriage of NAL unit structured video in the ISO Base Media File Format"에 타일링 디스크립터(tiling descriptor)들이 존재한다.

[0023] 그렇지만, 이 디스크립터들은 'track' 박스들 및 및 샘플 그룹화 도구들에 의존하며, 'meta' 기반 접근법을 사용할 때 스틸 이미지 파일 포맷(Still Image File Format)에서 사용될 수 없다. 이러한 디스크립터들이 없으면, 이 파일 포맷으로 저장되는 코딩된 픽처(coded picture)로부터 타일들을 선택하고 추출하는 것이 복잡해진다.

[0024] 도 1은, MPEG contribution m32254에 개시된 바와 같이, 타일들을 이용해 인코딩된 스틸 이미지를 ISO 베이스 미디어 파일 포맷의 'meta' 박스(100)에 기술하는 것을 예시하고 있다.

[0025] 각각의 타일 픽처에 대한 각자의 정보 아이템들(102, 103, 104 및 105)에 부가하여, 풀 픽처(full picture)에 대한 정보 아이템(101)이 정의된다. 그 정보 아이템들은 'ItemInfoBox'(iinf)라고 불리는 박스에 저장된다. ISO BMFF 표준으로부터의, 'ItemReferenceBox'라고 불리는, 박스(106)는 풀 픽처의 정보 아이템과 타일 픽처들에 대응하는 4개의 정보 아이템들 사이에(108) 'tile' 관계(107)가 존재한다는 것을 표시하기 위해 사용된다. 'ItemLocationBox'라고 불리는, 박스(109)가 각각의 정보 아이템을 표현하는 인코딩된 데이터(110)에서의 바이트 범위(들)를 제공하도록, 각각의 정보 아이템의 식별자들이 사용된다. 다른 박스 "ItemReferenceBox'"(112)는 EXIF 메타데이터(111)를 풀 픽처에 대한 정보 아이템(101)과 연관시키는 데 사용되고, 대응하는 데이터 블록(111)이 미디어 데이터 박스(110)에 생성된다. 또한, EXIF 메타데이터를 식별하기 위한 부가의 정보 아이템(113)이 생성된다.

[0026] 풀 픽처 및 그의 타일들이 정보 아이템들로서 도입되더라도, 여기에서 어떤 타일링 정보도 제공되지 않는다. 더욱이, 부가의 메타데이터를 (EXIF와 같은) 정보 아이템과 연관시킬 때, 부가의 ItemReferenceBox'를 사용하여 참조되는 데이터 블록이 생성되지 않는다.

[0027] EXIF로부터의 타일링에 관한 정보를 재사용하는 것 및 스틸 이미지 파일 포맷(Still Image File format) 초안에 정의된 메커니즘을 재사용하는 것이 비정규 격자(non-regular grid)를 기존의 EXIF 태그들을 사용해 기술하는 것을 가능하게 하지 않을 것이다.

[0028] 따라서, 스틸 이미지들, 특히 HEVC 스틸 이미지들에 대한 파일 포맷에서의 개선들이 여전히 필요하다. 상세하게는, 이 파일 포맷을 사용해 저장된 스틸 이미지들에서의 관심 영역을 추출하기 위한 방법들이 필요하다.

[0029] 본 발명은 상기한 바와 관련되어 있다.

발명의 내용

[0030] 본 발명의 제1 양태에 따르면, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 방법이 제공되고, 본 방법은:

[0031] 이미지 구역(image area)을 하나 이상의 타일들로 분할(divide)하기 위한 공간 파라미터들을 포함하는 타일 디

스크립션 정보(tile description information)를 제공하는 단계;

[0032] - 단일 이미지의 타일을 표현하는 비트스트림의 한 부분을 식별해주는 타일 픽처 아이템 정보(tile picture item information)를 제공하는 단계;

[0033] - 상기 타일 픽처 아이템을 상기 타일 디스크립션 정보에 링크시키는 참조 정보를 제공하는 단계, 및

[0034] - 상기 비트스트림을 상기 제공된 정보와 함께 캡슐화된 데이터 파일로서 출력하는 단계를 포함한다.

[0035] 출력은 정의된 표준에 따라 수행될 수 있고, 판독가능하며 디코딩가능하다.

[0036] 제1 양태에 따른 방법은, 신택스 엘리먼트(syntax element)들을 파싱하는 것에 의해 그리고 복잡한 계산 없이, 예를 들어, 초고해상도 이미지들(4K2K, 8K4K ...)로부터 타일들을 쉽게 식별, 선택 및 추출하는 것을 가능하게 한다.

[0037] ISO 베이스 미디어 파일 포맷의 메타데이터 박스들의 디스크립션 도구들이 확장될 수 있다. 상세하게는, 본 방법은 타일 디스크립션(tile description)을 정보 아이템들과 연관시키는 것을 가능하게 한다.

[0038] 부가의 디스크립션 도구들을 제공하기 위해 그리고 특히 스틸 이미지들 내에서의 타일 기반 액세스를 지원하기 위해 'meta' 박스 계층구조의 부분들이 확장될 수 있다.

[0039] 제1 양태에 따른 방법은 HEVC 타일들에 기초하여 관심 영역을, 인코딩된 HEVC 스틸 픽처로부터, 용이하게 추출하는 것을 가능하게 한다.

[0040] 본 발명의 실시예들은 HEVC 표준에 따라 인코딩된 스틸 이미지들에 대한 타일 디스크립션 지원(tile description support) 및 타일 액세스를 제공한다.

[0041] 이것은 스틸 이미지에 대한 비디오 트랙들에 대해 이용가능한 관심 영역 특징을 보존(preserve)하는 것을 가능하게 한다. 일반적으로, 미디어 플레이어들로의 렌더링 또는 전송을 위해 사용자 정의 관심 영역(user-defined region of interest)에 대응하는 스틸 픽처의 파트들이 식별되고 쉽게 추출될 수 있다.

[0042] 예를 들어, 상기 캡슐화된 인코딩된 비트스트림은 또한 비디오 시퀀스에 대응하는 상기 데이터 스트림의 시간 설정된 부분(timed portion)을 식별해주는 정보를 포함한다.

[0043] 따라서, 이 비디오의 일부인 몇몇 스틸 이미지들에서처럼 비디오에 대한 동일한 액세스 기능(access facility)들을 제공하는 단일 데이터 조각에 대한 이중 인덱싱(double indexing)이 제공될 수 있다.

[0044] 예를 들어, 타일 디스크립션 정보는 각각의 타일 픽처 아이템에 대한 공간 파라미터들의 세트를 포함한다.

[0045] 예를 들어, 타일 디스크립션 정보는 하나 초과의 타일 픽처 아이템에 공통인 공간 파라미터들을 포함한다.

[0046] 예를 들어, 타일 디스크립션 정보는 비트스트림에 임베딩(embed)된다.

[0047] 예를 들어, 타일 디스크립션 정보는 메타데이터로서 제공된다.

[0048] 예를 들어, 참조 정보는 참조 타입(reference type), 및 상기 타일 디스크립션 정보를 비롯한 부가의 기술적 메타데이터를 포함한다.

[0049] 예를 들어, 참조 정보는 참조 타입, 및 상기 타일 디스크립션 정보에 관련된 참조 파라미터를 포함한다.

[0050] 본 방법은 비트스트림 내의 상기 타일 디스크립션 정보를 참조하기 위한 메타데이터 아이템을 제공하는 단계를 추가로 포함할 수 있다.

[0051] 예를 들어, 타일 픽처 아이템들이 그룹화되고, 여기서 참조 정보는 타일 픽처 아이템들의 그룹을 상기 타일 디스크립션 정보에 링크시키기 위해 제공된다.

[0052] 예를 들어, 메타데이터 아이템들을 다른 아이템에 링크시키는 모든 참조들은 캡슐화된 데이터 파일 내의 단일 참조 박스에 포함된다.

[0053] 예를 들어, 임의의 타입의, 하나의 아이템으로부터의 모든 관계들이 단일 아이템 정보 디스크립터(single item information descriptor)에 저장된다.

[0054] 예를 들어, 여기서 상기 출력하는 단계는 적응적 스트리밍을 위해 서버 모듈에 의해 수행된다.

[0055] 예를 들어, 상기 출력하는 단계는 메모리에 저장하기 위해 수행된다.

[0056] 예를 들어, 상기 출력하는 단계는 디스플레이하기 위해 디스플레이 모듈에 대해 수행된다.

[0057] 예를 들어, 상기 출력하는 단계는 전송을 위해 통신 모듈에 의해 수행된다.

[0058] 예를 들어, 상기 캡슐화된 데이터 파일은 표준화된 파일 포맷에 대응한다.

[0059] 예를 들어, 상기 캡슐화된 데이터 파일은 디코딩가능하고 재생가능하다.

[0060] 본 발명의 제2 양태에 따르면, 하나 이상의 이미지들에 대응하는 인코딩된 비트스트림, 및 이미지 구역을 하나 이상의 타일들로 분할하기 위한 공간 파라미터들을 포함하는 타일 디스크립션 정보를 비롯한 정보를 포함하는 캡슐화된 데이터 파일을 처리하는 방법이 제공되며, 본 방법은:

[0061] - 이미지 관심 영역을 선택하는 단계,

[0062] - 상기 타일 디스크립션 정보에 기반하여, 선택된 관심 영역에 대응하는 타일들을 식별하는 단계,

[0063] - 상기 식별된 타일들에 링크된 하나 이상의 타일 픽처 아이템들을 선택하는 단계 - 각각의 타일 픽처 아이템은 단일 이미지의 타일을 표현하는 비트스트림의 한 부분을 식별해줌 -,

[0064] - 선택된 타일 픽처 아이템(들)에 의해 식별된 비트스트림의 한 부분을 추출하는 단계, 및

[0065] - 상기 추출된 비트스트림 부분을 출력하는 단계를 포함한다.

[0066] 예를 들어, 여기서 상기 출력하는 단계는 적응적 스트리밍을 위해 서버 모듈에 의해 수행된다.

[0067] 예를 들어, 상기 출력하는 단계는 메모리에 저장하기 위해 수행된다.

[0068] 예를 들어, 상기 출력하는 단계는 디스플레이하기 위해 디스플레이 모듈에 대해 수행된다.

[0069] 예를 들어, 상기 출력하는 단계는 전송을 위해 통신 모듈에 의해 수행된다.

[0070] 예를 들어, 상기 캡슐화된 데이터 파일은 표준화된 파일 포맷에 대응한다.

[0071] 예를 들어, 상기 캡슐화된 데이터 파일은 디코딩가능하고 재생가능하다.

[0072] 본 발명의 제3 양태에 따르면, 캡슐화 파일로 캡슐화하기 위해 적어도 하나의 이미지를 표현하는 이미지 데이터를 처리하는 방법이 제공되며, 본 방법은:

[0073] - 상기 적어도 하나의 이미지의, 복수의 이미지 부분들로의, 공간 세분(spatial subdivision)을 획득하는 단계 -

[0074] - 상기 복수의 이미지 부분들 중 한 이미지 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 부분 식별 데이터(portion identification data)를 결정하는 단계,

[0075] - 상기 이미지 데이터를 적어도

[0076] o 상기 적어도 하나의 이미지의 상기 세분을 표현하는 세분 디스크립션 데이터(subdivision description data),

[0077] o 상기 부분 식별 데이터, 및

[0078] o 상기 세분 디스크립션 데이터와 상기 부분 식별 데이터를 링크시키는 참조 데이터와 함께 상기 캡슐화 파일로 캡슐화하는 단계를 포함한다.

[0079] 예를 들어, 상기 이미지 데이터는 비디오 시퀀스의 복수의 이미지들을 표현하고, 본 방법은 상기 비디오 시퀀스의 한 시간 부분(time portion)을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 시간 식별 데이터(time identification data)를 결정하는 단계를 추가로 포함하고, 상기 이미지 데이터는 상기 시간 식별 데이터와 함께 캡슐화된다.

[0080] 예를 들어, 상기 비디오 시퀀스의 상기 시간 부분의 이미지들의 동일한 이미지 부분을, 각각, 표현하는 복수의 부분 식별 데이터가 결정된다.

[0081] 예를 들어, 적어도 상기 세분 디스크립션 데이터는 이미지 데이터에 대한 메타데이터로서 캡슐화된다.

[0082] 예를 들어, 상기 공간 세분은 상기 이미지 데이터를 포함하는 비트스트림에 임베딩된다.

[0083] 예를 들어, 각각의 이미지 부분에 대한 각자의 부분 식별 데이터가 결정된다.

[0084] 예를 들어, 복수의 이미지 부분들에 대한 공통의 부분 식별 데이터가 결정된다.

[0085] 본 방법은 서버 디바이스에 의해 적응적 스트리밍을 위해 상기 캡슐화 파일을 비트스트림 내로 출력하는 단계를 추가로 포함할 수 있다.

[0086] 본 방법은 상기 이미지 데이터를 디스플레이하기 위해 디스플레이 디바이스로 전송하기 위해 상기 캡슐화 파일을 비트스트림 내로 출력하는 단계를 추가로 포함할 수 있다.

[0087] 본 방법은 클라이언트 디바이스로 전송하기 위해 상기 캡슐화 파일을 비트스트림 내로 출력하는 단계를 추가로 포함할 수 있다.

[0088] 본 방법은 상기 캡슐화 파일을 저장 디바이스에 저장하는 단계를 추가로 포함할 수 있다.

[0089] 예를 들어, 참조 데이터는 참조 타입, 및 상기 세분 디스크립션 데이터를 비롯한 부가의 기술적 메타데이터를 포함한다.

[0090] 예를 들어, 참조 데이터는 참조 타입, 및 상기 세분 디스크립션 데이터에 관련된 참조 파라미터를 포함한다.

[0091] 예를 들어, 상기 세분 디스크립션 데이터는 메타데이터 아이템에서 참조된다.

[0092] 예를 들어, 부분 식별 데이터가 그룹화되고, 여기서 참조 데이터는 부분 식별 데이터의 그룹을 상기 부분 식별 데이터에 링크시킨다.

[0093] 예를 들어, 상기 캡슐화된 파일은 이미지 데이터에 대한 모든 참조 데이터를 포함하는 단일 참조 박스를 포함한다.

[0094] 예를 들어, 상기 캡슐화된 파일은 상기 세분 디스크립션 데이터, 부분 식별 데이터 및 참조 데이터 간의 관계들의 표현을 포함하는 디스크립션을 포함한다.

[0095] 본 발명의 제4 양태에 따르면, 캡슐화 파일을 처리하는 방법이 제공되고, 캡슐화 파일은:

[0096] - 적어도 하나의 이미지를 표현하는 이미지 데이터,

[0097] - 상기 적어도 하나의 이미지의, 복수의 이미지 부분들로의, 공간 세분을 표현하는 세분 디스크립션 데이터,

[0098] - 상기 복수의 이미지 부분들 중 한 이미지 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 부분 식별 데이터, 및

[0099] - 상기 세분 디스크립션 데이터와 상기 부분 정보를 링크시키는 참조 데이터를 포함하고,

[0100] 본 방법은:

[0101] - 상기 적어도 하나의 이미지에서의 관심 영역을 결정하는 단계,

[0102] - 상기 세분 디스크립션 데이터에 기초하여, 상기 관심 영역에 속하는 적어도 하나의 이미지 부분을 결정하는 단계,

[0103] - 상기 참조 데이터에 기초하여, 상기 관심 영역에 속하는 상기 적어도 하나의 이미지 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 부분 식별 데이터에 액세스하는 단계, 및

[0104] - 상기 이미지 데이터 내의 상기 데이터 부분을 추출하는 단계를 포함한다.

[0105] 예를 들어, 상기 이미지 데이터는 비디오 시퀀스의 복수의 이미지들을 포함하고, 상기 캡슐화 파일은, 상기 비디오 시퀀스의 한 시간 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 시간 식별 데이터를 추가로 포함하며, 상기 비디오 시퀀스의 상기 시간 부분의 이미지들에 대한 관심 영역이 결정되고, 상기 비디오 시퀀스의 상기 시간 부분의 복수의 이미지들에서의 상기 관심 영역에 대응하는 데이터 부분들이 추출된다.

[0106] 예를 들어, 복수의 부분 식별 데이터는, 각각, 상기 비디오 시퀀스의 상기 시간 부분의 이미지들의 동일한 이미지 부분을 표현한다.

[0107] 예를 들어, 적어도 상기 세분 디스크립션 데이터는 이미지 데이터에 대한 메타데이터로서 캡슐화된다.

[0108] 예를 들어, 각각의 이미지 부분에 대한 각자의 부분 식별 데이터가 결정된다.

[0109] 예를 들어, 복수의 이미지 부분들에 대한 공통의 부분 식별 데이터가 결정된다.

[0110] 본 방법은 상기 캡슐화 파일을 서버 디바이스에 의해 적응적으로 스트리밍되는 비트스트림으로서 수신하는 단계를 추가로 포함할 수 있다.

[0111] 본 방법은 상기 관심 영역을 디스플레이하는 단계를 추가로 포함할 수 있다.

[0112] 예를 들어, 참조 데이터는 참조 타입, 및 상기 세분 디스크립션 데이터를 비롯한 부가의 기술적 메타데이터를 포함한다.

[0113] 예를 들어, 참조 데이터는 참조 타입, 및 상기 세분 디스크립션 데이터에 관련된 참조 파라미터를 포함한다.

[0114] 예를 들어, 상기 세분 디스크립션 데이터는 메타데이터 아이템에서 참조된다.

[0115] 예를 들어, 부분 식별 데이터가 그룹화되고, 여기서 참조 데이터는 부분 식별 데이터의 그룹을 상기 부분 식별 데이터에 링크시킨다.

[0116] 예를 들어, 상기 캡슐화된 파일은 이미지 데이터에 대한 모든 참조 데이터를 포함하는 단일 참조 박스를 포함한다.

[0117] 예를 들어, 상기 캡슐화된 파일은 상기 세분 디스크립션 데이터, 부분 식별 데이터 및 참조 데이터 간의 관계들의 표현을 포함하는 디스크립션을 포함한다.

[0118] 본 발명의 제5 양태에 따르면, 제1 양태에 따른 방법을 구현하도록 구성된 디바이스가 제공된다.

[0119] 본 디바이스는:

[0120] - 이미지 구역을 하나 이상의 타일들로 분할하기 위한 공간 파라미터들을 포함하는 타일 디스크립션 정보를 제공하고; 단일 이미지의 타일을 표현하는 비트스트림의 한 부분을 식별해주는 타일 픽처 아이템 정보를 제공하며; 상기 타일 픽처 아이템을 상기 타일 디스크립션 정보에 링크시키는 참조 정보를 제공하도록 구성된 처리 유닛, 및

[0121] - 상기 비트스트림을 상기 제공된 정보와 함께 캡슐화된 데이터 파일로서 출력하도록 구성된 통신 유닛을 포함할 수 있다.

[0122] 본 발명의 제6 양태에 따르면, 제2 양태에 따른 방법을 구현하도록 구성된 디바이스가 제공된다.

[0123] 본 디바이스는 하나 이상의 이미지들에 대응하는 인코딩된 비트스트림, 및 이미지 구역을 하나 이상의 타일들로 분할하기 위한 공간 파라미터들을 포함하는 타일 디스크립션 정보를 비롯한 정보를 포함하는 캡슐화된 데이터 파일을 처리하도록 구성될 수 있다. 본 디바이스는 또한:

[0124] - 이미지 관심 영역을 선택하고, 상기 타일 디스크립션 정보에 기초하여, 선택된 관심 영역에 대응하는 타일들을 식별하며, 상기 식별된 타일들에 링크된 하나 이상의 타일 픽처 아이템들을 선택하고 - 각각의 타일 픽처 아이템은 단일 이미지의 타일을 표현하는 비트스트림의 한 부분을 식별해줌 -, 선택된 타일 픽처 아이템(들)에 의해 식별된 비트스트림의 한 부분을 추출하도록 구성된 처리 유닛, 및

[0125] - 상기 추출된 비트스트림 부분을 출력하도록 구성된 통신 유닛을 포함할 수 있다.

[0126] 본 발명의 제7 양태에 따르면, 제3 양태에 따른 방법을 구현하도록 구성된 디바이스가 제공된다.

[0127] 본 디바이스는 캡슐화 파일로 캡슐화하기 위해 적어도 하나의 이미지를 표현하는 이미지 데이터를 처리하도록 구성될 수 있고, 본 디바이스는 상기 적어도 하나의 이미지의, 복수의 이미지 부분들로의, 공간 세분을 획득하고, - 상기 복수의 이미지 부분들 중 한 이미지 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 부분 식별 데이터를 결정하며, 상기 이미지 데이터를 적어도

[0128] - 상기 적어도 하나의 이미지의 상기 세분을 표현하는 세분 디스크립션 데이터,

[0129] - 상기 부분 식별 데이터, 및

[0130] - 상기 세분 디스크립션 데이터와 상기 부분 식별 데이터를 링크시키는 참조 데이터와 함께 상기 캡슐화 파일로

캡슐화하도록 구성된 처리 유닛을 포함할 수 있다.

[0131] 본 발명의 제8 양태에 따르면, 제4 양태에 따른 방법을 구현하도록 구성된 디바이스가 제공된다.

[0132] 본 디바이스는 캡슐화 파일을 처리하도록 구성될 수 있고, 캡슐화 파일은:

[0133] - 적어도 하나의 이미지를 표현하는 이미지 데이터,

[0134] - 상기 적어도 하나의 이미지의, 복수의 이미지 부분들로의, 공간 세분을 표현하는 세분 디스크립션 데이터,

[0135] - 상기 복수의 이미지 부분들 중 한 이미지 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 부분 식별 데이터, 및

[0136] - 상기 세분 디스크립션 데이터와 상기 부분 정보를 링크시키는 참조 데이터를 포함한다.

[0137] 본 디바이스는 또한 상기 적어도 하나의 이미지에서의 관심 영역을 결정하고, 상기 세분 디스크립션 데이터에 기초하여, 상기 관심 영역에 속하는 적어도 하나의 이미지 부분을 결정하며, 상기 참조 데이터에 기초하여, 상기 관심 영역에 속하는 상기 적어도 하나의 이미지 부분을 표현하는, 상기 이미지 데이터 내의 한 데이터 부분을 식별해주는 적어도 하나의 부분 식별 데이터에 액세스하고, 상기 이미지 데이터 내의 상기 데이터 부분을 추출하도록 구성된 처리 유닛을 포함할 수 있다.

[0138] 본 발명의 제9 양태에 따르면, 시스템이 제공되고, 본 시스템은:

[0139] - 제5 또는 제7 양태에 따른 제1 디바이스, 및

[0140] - 상기 제1 디바이스로부터의 파일들을 처리하기 위한 제6 또는 제8 양태에 따른 제2 디바이스를 포함한다.

[0141] 본 발명의 제10 양태에 따르면, 프로그래밍가능 장치의 컴퓨터 수단에 로딩되어 실행될 때, 본 발명의 제1, 제2, 제3 및/또는 제4 양태(들)에 따른 방법들을 구현하기 위한 명령어들을 포함하는 컴퓨터 프로그램들 및 컴퓨터 프로그램 제품들이 제공된다.

[0142] 본 발명의 제11 양태에 따르면, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 방법이 제공되고, 캡슐화된 비트스트림은 데이터 파트(data part) 및 메타데이터 파트(metadata part)를 포함한다. 본 방법은:

[0143] - 서브 이미지(sub-image) 또는 단일 이미지의 이미지를 표현하는 데이터 파트의 한 부분을 식별해주는 이미지 아이템 정보(image item information)를 제공하는 단계;

[0144] - 하나 이상의 이미지들에 관련된 변환 연산자(transformation operator)들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 디스크립션 정보(image description information)를 제공하는 단계, 및

[0145] - 상기 비트스트림을 상기 제공된 정보와 함께 캡슐화된 데이터 파일로서 출력하는 단계를 포함하고;

[0146] - 여기서 이미지 디스크립션 정보는 메타데이터 파트에 저장된다.

[0147] 일 실시예에서, 이미지 디스크립션 정보에 포함된 각각의 파라미터는

[0148] - 타입 정보, 및/또는

[0149] - 이미지 아이템 정보를 상기 파라미터에 링크시키는 데 사용되는 식별자를 포함하는 부가의 데이터와 연관되어 있다.

[0150] 일 실시예에서, 메타데이터 파트는 ISOBMFF의 'meta' 데이터 박스에 포함된다.

[0151] 일 실시예에서, 부가의 데이터는 헤더이다.

[0152] 일 실시예에서, 부가의 데이터는 가상 아이템(Virtual item)이다.

[0153] 다른 실시예에서, 이미지 디스크립션 정보에 포함된 각각의 변환 연산자들은 변환된 아이템을 상기 변환 연산자들에 링크시키는 데 사용되는 식별자를 포함하는 부가의 데이터와 연관되어 있다.

[0154] 일 실시예에서, 메타데이터 파트에 저장된 박스는 적어도 하나의 변환 연산자를 포함한다.

[0155] 일 실시예에서, 캡슐화된 비트스트림의 데이터 파트는 하나 이상의 변환 연산자들과 연관되어 있는 변환된 아이템을 포함하고, 메타데이터 파트는:

[0156] - 변환 연산자가 적용되는 원본 이미지를 식별해주기 위한 정보, 및

[0157] - 데이터 파트 내의 변환된 아이템을 위치결정(localize)하기 위한 정보를 추가로 포함한다.

[0158] 일 실시예에서, 변환된 아이템은, 메타데이터 파트 내의 변환 연산자들 중 하나를 식별하는 것을 가능하게 하는 인덱스인, 적어도 하나의 변환 인덱스를 포함한다.

[0159] 본 발명의 제12 양태에 따르면, 데이터 파트 및 메타데이터 파트 - 하나 이상의 이미지들에 대응하는 인코딩된 비트스트림은 데이터 파트에 포함시키고, 메타데이터 파트 내의 정보는 하나 이상의 이미지들 또는 서브 이미지들에 관련된 변환 연산자들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 또는 서브 이미지 디스크립션 정보를 포함함 - 를 포함하는 캡슐화된 데이터 파일을 처리하는 방법이 제공된다. 본 방법은:

[0160] - 관심 이미지 또는 서브 이미지를 선택하는 단계,

[0161] - 상기 참조된 이미지 또는 서브 이미지 디스크립션 정보에 기초하여, 메타데이터 파트로부터 연관된 디스플레이 파라미터들 및/또는 변환 연산자들을 식별하는 단계;

[0162] - 변환 연산자들이 식별된 경우에, 이미지 또는 서브 이미지에 변환을 적용하고, 최종적으로 변환된 상기 이미지 또는 서브 이미지를, 상기 디스플레이 파라미터들에 따라, 디스플레이하는 단계를 포함한다.

[0163] 일 실시예에서, 본 방법은, 식별하는 단계 이전에, 상기 파라미터들에 포함된 부가의 데이터를 검색하는 단계를 추가로 포함하고, 상기 부가의 데이터는:

[0164] - 타입 정보, 및/또는

[0165] - 이미지 또는 서브 이미지 아이템 정보를 상기 파라미터에 링크시키는 데 사용되는 식별자를 포함한다.

[0166] 일 실시예에서, 메타데이터 파트는 ISOBMFF의 'meta' 데이터 박스에 포함된다.

[0167] 일 실시예에서, 부가의 데이터는 헤더이다.

[0168] 일 실시예에서, 부가의 데이터는 가상 아이템이다.

[0169] 다른 실시예에서, 이미지 디스크립션 정보에 포함된 각각의 변환 연산자들은 변환된 아이템을 상기 변환 연산자들에 링크시키는 데 사용되는 식별자를 포함하는 부가의 데이터와 연관되어 있다.

[0170] 일 실시예에서, 메타데이터 파트에 저장된 박스는 적어도 하나의 변환 연산자를 포함한다.

[0171] 일 실시예에서, 캡슐화된 비트스트림의 데이터 파트는 하나 이상의 변환 연산자들과 연관되어 있는 변환된 아이템을 포함하고, 메타데이터 파트는:

[0172] - 변환 연산자가 적용되는 원본 이미지를 식별해주기 위한 정보, 및

[0173] - 데이터 파트 내의 변환된 아이템을 위치결정하기 위한 정보를 추가로 포함한다.

[0174] 일 실시예에서, 변환된 아이템은, 메타데이터 파트 내의 변환 연산자들 중 하나를 식별하는 것을 가능하게 하는 인덱스인, 적어도 하나의 변환 인덱스를 포함한다.

[0175] 본 발명의 제13 양태에 따르면, 본 발명의 제11 양태에 따른 캡슐화 방법을 구현하도록 구성된, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 서버 디바이스가 제공된다.

[0176] 본 발명의 제14 양태에 따르면, 본 발명의 제12 양태에 따른 처리 방법을 구현하도록 구성된, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 클라이언트 디바이스가 제공된다.

[0177] 본 발명의 제15 양태에 따르면, 프로그래밍가능 장치의 컴퓨터 수단에 로딩되어 실행될 때, 본 발명의 제11 및 제12 양태에 따른 방법들을 구현하기 위한 명령어들을 포함하는 컴퓨터 프로그램들 및 컴퓨터 프로그램 제품들이 제공된다.

[0178] 본 발명의 제16 양태에 따르면, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 방법이 제공되고, 캡슐화된 비트스트림은 데이터 파트 및 메타데이터 파트를 포함하며, 본 방법은:

[0179] - 서브 이미지 또는 단일 이미지의 이미지 및/또는 단일 이미지들의 세트를 표현하는 데이터 파트의 한 부분을 식별해주는 이미지 아이템 정보를 제공하는 단계;

[0180] - 하나 이상의 이미지들에 관련된 변환 연산자들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 디스크립션 정보를 제공하는 단계, 및

[0181] - 상기 비트스트림을 상기 제공된 정보와 함께 캡슐화된 데이터 파일로서 출력하는 단계를 포함한다.

[0182] 상기 이미지 아이템 정보는 고려된 서브 이미지 또는 단일 이미지 또는 단일 이미지들의 세트에 전용된 이미지 디스크립션 정보의 적어도 일부를 포함하는 하나 이상의 속성들을 포함하고, 상기 이미지 디스크립션 정보는 하나 이상의 박스들에 정의된다.

[0183] 본 발명의 이 양태는 효율적인 참조 메커니즘을 위해 데이터와 메타데이터의 명확한 분리를 제공하는 것을 가능하게 한다.

[0184] 일 실시예에서, 이미지 아이템 정보는 박스이고 각각의 이미지 아이템 정보의 속성은 박스이며, 속성 박스들은 박스들의 테이블을 형성하기 위해 조직화(organize)된다.

[0185] 일 실시예에서, 각각의 속성은, 출현 순서를 따르는 것 또는 박스들의 테이블에서의 대응하는 박스에 의해, 서브 이미지 또는 이미지 및/또는 단일 이미지들의 세트에 적용된다.

[0186] 일 실시예에서, 서브 이미지 또는 단일 이미지 및/또는 단일 이미지들의 세트는 비디오 시퀀스에 관련되고, 이미지 아이템 정보 속성들 중 하나는 상기 비디오 시퀀스의 초기화 정보를 참조하기 위한 하나 이상의 초기화 파라미터들을 포함한다.

[0187] 일 실시예에서, 몇 개의 서브 이미지들 또는 단일 이미지들 및/또는 단일 이미지들의 세트 사이에서 공유되는 이미지 디스크립션 정보의 일부는 하나의 전용 공유 박스(dedicated shared box)에 정의되고, 각각의 이미지 디스크립션 정보는 고려된 이미지 아이템 정보를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 통해 검색가능하고, 상기 링크시키는 구조체는:

[0188] - 고려된 이미지 아이템 정보마다의 제1 식별자 - 상기 제1 식별자는 이미지 아이템 정보의 속성으로서 정의되고 전용 공유 박스 내의 동일한 값을 갖는 제2 식별자를 참조함 -,

[0189] - 전용 공유 박스에 포함된 하나 또는 몇 개의 제2 식별자들 - 각각의 제2 식별자는 이미지 디스크립션 정보를 참조함 - 을 포함한다.

[0190] 일 실시예에서, 몇 개의 서브 이미지들 또는 단일 이미지들 및/또는 단일 이미지들의 세트 사이에서 공유되는 이미지 디스크립션 정보의 일부는 2개의 전용 공유 박스들에 정의되고, 하나의 공유 박스는 디스플레이 파라미터들에 관련되고 하나의 다른 공유 박스는 변환 연산자들에 관련되며, 각각의 이미지 디스크립션 정보는 이미지 아이템 정보를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 통해 검색가능하다.

[0191] 일 실시예에서, 상기 링크시키는 구조체는 이미지 아이템 정보와 적어도 하나의 이미지 디스크립션 정보를 링크시키는 2개의 참조 타입 파라미터들을 포함하고, 각각의 참조 타입 파라미터는 전용 공유 박스들 중 하나에 특정적(specific)이다.

[0192] 일 실시예에서, 상기 링크시키는 구조체는:

[0193] -고려된 이미지 아이템 정보마다의 제1 식별자 및 제2 식별자들 - 상기 제1 식별자는 이미지 아이템 정보의 속성으로서 정의되고 디스플레이 파라미터들에 관련된 전용 공유 박스 내의 제3 식별자들을 참조하며, 상기 제2 식별자들은 이미지 아이템 정보의 속성으로서 정의되고 변환 연산자들에 관련된 전용 공유 박스 내의 제4 식별자들을 참조함 -,

[0194] - 디스플레이 파라미터들 및 변환 연산자들에 관련된 전용 공유 박스들에, 각각, 포함된 하나 또는 몇 개의 제3 및 제4 식별자들 - 각각의 제3 및 제4 식별자들은, 각각, 디스플레이 파라미터 및 변환 연산자를 참조함 - 을 포함한다.

[0195] 일 실시예에서, 디스플레이 파라미터들 중 하나는 단일 이미지의 일부들에 대응하는 단일 이미지들의 세트를 정의하기 위한 격자이다.

[0196] 일 실시예에서, 단일 이미지들의 세트 중의 이미지들은 동일한 단일 이미지에 관련되어 있다.

[0197] 본 발명의 제17 양태에 따르면, 하나 이상의 이미지들을 표현하는 캡슐화된 비트스트림을 획득하는 방법이 제공되고, 캡슐화된 비트스트림은 인코딩된 데이터 파트 및 메타데이터 파트를 포함하며, 본 방법은:

[0198] - 서브 이미지 또는 단일 이미지의 이미지 및/또는 단일 이미지들의 세트를 표현하는 데이터 파트의 한 부분을 식별해주는 이미지 아이템 정보를 획득하는 단계;

[0199] - 하나 이상의 이미지들에 관련된 변환 연산자들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 디스크립션 정보를 획득하는 단계, 및

[0200] - 상기 비트스트림을 상기 결정된 정보와 함께 캡슐화된 데이터 파일로서 추출하는 단계를 포함한다.

[0201] 상기 이미지 아이템 정보는 고려된 서브 이미지 또는 단일 이미지 또는 단일 이미지들의 세트에 전용된 이미지 디스크립션 정보의 적어도 일부를 포함하는 하나 이상의 속성들을 포함하고, 상기 이미지 디스크립션 정보는 하나 이상의 박스들에 정의된다.

[0202] 일 실시예에서, 이미지 아이템 정보가 박스이고 각각의 이미지 아이템 정보의 속성이 박스이며, 속성 박스들은 박스들의 테이블을 형성하기 위해 조직화된다.

[0203] 일 실시예에서, 각각의 속성은, 출현 순서를 따르는 것 또는 박스들의 테이블에서의 대응하는 박스에 의해, 서브 이미지 또는 이미지 및/또는 단일 이미지들의 세트에 적용된다.

[0204] 일 실시예에서, 서브 이미지 또는 단일 이미지 및/또는 단일 이미지들의 세트는 비디오 시퀀스에 관련되고, 이미지 아이템 정보 속성들 중 하나는 상기 비디오 시퀀스의 초기화 정보를 참조하기 위한 하나 이상의 초기화 파라미터들을 포함한다.

[0205] 일 실시예에서, 몇 개의 서브 이미지들 또는 단일 이미지들 및/또는 단일 이미지들의 세트 사이에서 공유되는 이미지 디스크립션 정보의 일부가 하나의 전용 공유 박스에 정의되고, 각각의 이미지 디스크립션 정보는 고려된 이미지 아이템 정보를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 통해 검색가능하고, 상기 링크시키는 구조체는:

[0206] - 고려된 이미지 아이템 정보에 대한 제1 식별자 - 상기 제1 식별자는 이미지 아이템 정보의 속성으로서 정의되고 전용 공유 박스 내의 동일한 값을 갖는 제2 식별자를 참조함 -,

[0207] - 전용 공유 박스에 포함된 하나 또는 몇 개의 제2 식별자들 - 각각의 제2 식별자는 이미지 디스크립션 정보를 참조함 - 을 포함한다.

[0208] 일 실시예에서, 몇 개의 서브 이미지들 또는 단일 이미지들 및/또는 단일 이미지들의 세트 사이에서 공유되는 이미지 디스크립션 정보의 일부가 2개의 전용 공유 박스들에 정의되고, 하나의 공유 박스는 디스플레이 파라미터들에 관련되고 하나의 다른 공유 박스는 변환 연산자들에 관련되며, 각각의 이미지 디스크립션 정보는 이미지 아이템 정보를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 통해 검색가능하다.

[0209] 일 실시예에서, 상기 링크시키는 구조체는 이미지 아이템 정보와 적어도 하나의 이미지 디스크립션 정보를 링크시키는 2개의 참조 타입 파라미터들을 포함하고, 각각의 참조 타입 파라미터는 전용 공유 박스들 중 하나에 특정적이다.

[0210] 일 실시예에서, 상기 링크시키는 구조체는:

[0211] -고려된 이미지 아이템 정보에 대한 제1 식별자 및 제2 식별자들 - 상기 제1 식별자는 이미지 아이템 정보의 속성으로서 정의되고 디스플레이 파라미터들에 관련된 전용 공유 박스 내의 제3 식별자들을 참조하며, 상기 제2 식별자들은 이미지 아이템 정보의 속성으로서 정의되고 변환 연산자들에 관련된 전용 공유 박스 내의 제4 식별자들을 참조함 -,

[0212] 디스플레이 파라미터들 및 변환 연산자들에 관련된 전용 공유 박스들에, 각각, 포함된 하나 또는 몇 개의 제3 및 제4 식별자들 - 각각의 제3 및 제4 식별자들은, 각각, 디스플레이 파라미터 및 변환 연산자를 참조함 - 을 포함한다.

[0213] 일 실시예에서, 디스플레이 파라미터들 중 하나는 단일 이미지의 일부들에 대응하는 단일 이미지들의 세트를 정의하기 위한 격자이다.

[0214] 일 실시예에서, 단일 이미지들의 세트 중의 이미지들은 동일한 단일 이미지에 관련되어 있다.

[0215] 본 발명의 제18 양태에 따르면, 본 발명의 제16 양태에 따른 방법을 구현하도록 구성된, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하기 위한 디바이스가 제공된다.

[0216] 본 발명의 제19 양태에 따르면, 본 발명의 제17 양태에 따른 방법을 구현하도록 구성된, 하나 이상의 이미지들을 표현하는 캡슐화된 비트스트림을 처리하기 위한 디바이스가 제공된다.

[0217] 본 발명의 제20 양태에 따르면, 시스템이 제공되고, 본 시스템은:

[0218] - 본 발명의 제18 양태에 따른 제1 디바이스, 및

[0219] - 상기 제1 디바이스로부터의 파일들을 처리하기 위한 본 발명의 제19 양태에 따른 제2 디바이스를 포함한다.

[0220] 본 발명의 제21 양태에 따르면, 프로그램이 프로그래밍가능 장치에 의해 로딩되어 실행될 때, 본 발명의 제16 또는 제17 양태에 따른 방법을 구현하기 위한 명령어들을 포함하는 컴퓨터 프로그램 제품이 제공된다.

[0221] 본 발명의 제22 양태에 따르면, 프로그램이 컴퓨터 또는 마이크로프로세서에 의해 로딩되어 실행될 때, 본 발명의 제16 또는 제17 양태에 따른 방법을 구현하기 위한 컴퓨터 프로그램의 명령어들을 저장하는, 컴퓨터 또는 마이크로프로세서에 의해 판독가능한 비일시적 정보 저장 수단이 제공된다.

[0222] 본 발명의 제23 양태에 따르면, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하는 방법이 제공되고, 캡슐화된 비트스트림은 데이터 파트 및 메타데이터 파트를 포함한다. 본 방법은:

[0223] - 서브 이미지 또는 단일 이미지의 이미지 및/또는 단일 이미지들의 세트를 표현하는 데이터 파트의 한 부분을 식별해주는 이미지 아이템 정보를 제공하는 단계;

[0224] - 하나 이상의 이미지들에 관련된 변환 연산자들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 디스크립션 정보를 제공하는 단계, 및

[0225] - 상기 비트스트림을 상기 제공된 정보와 함께 캡슐화된 데이터 파일로서 출력하는 단계를 포함한다.

[0226] 이미지 디스크립션 정보는 하나 또는 2개의 전용 박스들에 정의되고, 각각의 이미지 디스크립션 정보는 이미지 아이템 정보를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 통해 검색가능하다.

[0227] 일 실시예에서, 이미지 디스크립션 정보는 하나의 전용 박스에 정의되고, 상기 링크시키는 구조체는 이미지 아이템 정보와 적어도 하나의 이미지 디스크립션 정보를 링크시키는 참조 타입 파라미터를 포함한다.

[0228] 일 실시예에서, 이미지 디스크립션 정보는 하나 또는 2개의 전용 박스들에 정의되고, 상기 링크시키는 구조체는 이미지 아이템 정보와 적어도 하나의 이미지 디스크립션 정보를 링크시키기 위한 하나 또는 2개의 인덱스 세트를 포함하며, 각각의 세트는 전용 박스들 중 하나의 전용 박스에 연관되어 있다.

[0229] 일 실시예에서, 이미지 디스크립션 정보는 2개의 전용 박스들에 정의되며, 하나의 박스는 디스플레이 파라미터들에 관련되어 있고 하나의 다른 박스는 변환 연산자들에 관련되어 있다.

[0230] 일 실시예에서, 이미지 디스크립션 정보는 2개의 전용 박스들에 정의되고, 상기 링크시키는 구조체는 2개의 전용 박스들의 각각의 전용 박스에, 각각, 연관된 2개의 참조 타입 파라미터들을 포함하고, 각각의 참조 타입 파라미터는 연관된 전용 박스 내의 적어도 하나의 이미지 디스크립션 정보와 이미지 아이템 정보를 링크시킨다.

[0231] 본 발명의 제24 양태에 따르면, 하나 이상의 이미지들을 표현하는 캡슐화된 비트스트림을 획득하는 방법이 제공되고, 캡슐화된 비트스트림은 인코딩된 데이터 파트 및 메타데이터 파트를 포함하며, 본 방법은:

[0232] - 서브 이미지 또는 단일 이미지의 이미지 및/또는 단일 이미지들의 세트를 표현하는 데이터 파트의 한 부분을 식별해주는 이미지 아이템 정보를 획득하는 단계;

[0233] - 하나 이상의 이미지들에 관련된 변환 연산자들 및/또는 디스플레이 파라미터들을 비롯한 파라미터들을 포함하는 이미지 디스크립션 정보를 획득하는 단계, 및

[0234] - 상기 비트스트림을 상기 결정된 정보와 함께 캡슐화된 데이터 파일로서 추출하는 단계를 포함하고,

[0235] - 여기서 이미지 디스크립션 정보는 하나 또는 2개의 전용 박스들에 정의되고, 각각의 이미지 디스크립션 정보는 이미지 아이템 정보를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 통해 검색가능하다.

[0236] 일 실시예에서, 이미지 디스크립션 정보는 하나의 전용 박스에 정의되고, 상기 링크시키는 구조체는 이미지 아이템 정보와 적어도 하나의 이미지 디스크립션 정보를 링크시키는 참조 타입 파라미터를 포함한다.

[0237] 일 실시예에서, 이미지 디스크립션 정보는 하나 또는 2개의 전용 박스들에 정의되고, 상기 링크시키는 구조체는

이미지 아이템 정보와 적어도 하나의 이미지 디스크립션 정보를 링크시키기 위한 하나 또는 2개의 인덱스 세트를 포함하며, 각각의 세트는 전용 박스들 중 하나의 전용 박스에 연관되어 있다.

[0238] 일 실시예에서, 이미지 디스크립션 정보는 2개의 전용 박스들에 정의되며, 하나의 박스는 디스플레이 파라미터들에 관련되어 있고 하나의 다른 박스는 변환 연산자들에 관련되어 있다.

[0239] 일 실시예에서, 이미지 디스크립션 정보는 2개의 전용 박스들에 정의되고, 상기 링크시키는 구조체는 2개의 전용 박스들의 각각의 전용 박스에, 각각, 연관된 2개의 참조 타입 파라미터들을 포함하고, 각각의 참조 타입 파라미터는 연관된 전용 박스 내의 적어도 하나의 이미지 디스크립션 정보와 이미지 아이템 정보를 링크시킨다.

[0240] 본 발명의 제25 양태에 따르면, 본 발명의 제23 양태에 따른 방법을 구현하도록 구성된, 하나 이상의 이미지들을 표현하는 인코딩된 비트스트림을 캡슐화하기 위한 디바이스가 제공된다.

[0241] 본 발명의 제26 양태에 따르면, 본 발명의 제24 양태에 따른 방법을 구현하도록 구성된, 하나 이상의 이미지들을 표현하는 캡슐화된 비트스트림을 처리하기 위한 디바이스가 제공된다.

[0242] 본 발명의 제27 양태에 따르면, 시스템이 제공되고, 본 시스템은:

[0243] - 본 발명의 제25 양태에 따른 제1 디바이스, 및

[0244] - 상기 제1 디바이스로부터의 파일들을 처리하기 위한 본 발명의 제26 양태에 따른 제2 디바이스를 포함한다.

[0245] 본 발명의 제28 양태에 따르면, 프로그램이 프로그래밍가능 장치에 의해 로딩되어 실행될 때, 본 발명의 제23 또는 제24 양태에 따른 방법을 구현하기 위한 명령어들을 포함하는 컴퓨터 프로그램 제품이 제공된다.

[0246] 본 발명의 제28 양태에 따르면, 프로그램이 컴퓨터 또는 마이크로프로세서에 의해 로딩되어 실행될 때, 본 발명의 제23 또는 제24 양태에 따른 방법을 구현하기 위한 컴퓨터 프로그램의 명령어들을 저장하는, 컴퓨터 또는 마이크로프로세서에 의해 판독가능한 비일시적 정보 저장 수단이 제공된다.

도면의 간단한 설명

[0247] 본 발명의 다른 특징들 및 장점들은, 도 1에 부가하여, 첨부된 도면들을 참조하여 비제한적인 예시적인 실시예들에 대한 하기의 설명으로부터 명백해질 것이다.

도 2는 타일링된 비디오의 일 예를 예시한 도면;

도 3은 HEVC에서의 다양한 타일/슬라이스 구성들을 예시한 도면;

도 4는 'track' 박스들을 갖는 ISO 베이스 미디어 파일 포맷에 따른 타일 캡슐화를 예시한 도면;

도 5는 정보 아이템들을 ISOBMFF의 'meta' 박스들에 기술하기 위한 표준 메타데이터를 예시한 도면;

도 6은 정보 아이템 디스크립션에 대한 예시적인 확장을 예시한 도면;

도 7은 정보 아이템들 간의 참조 메커니즘들을 예시한 도면;

도 8은 본 발명의 실시예들의 구현의 상황(context)을 예시한 도면;

도 9는 본 발명의 하나 이상의 실시예들을 구현하기 위한 컴퓨팅 디바이스의 개략 블록도.

발명을 실시하기 위한 구체적인 내용

[0248] 이하에서, 본 발명의 실시예들이 설명된다.

[0249] 기술적 상황(technical context)을 보다 잘 이해하기 위해, 연속적인 시간 프레임들을 갖는 비디오(200)를 도시하는 도 2를 참조하여 비디오 타일링(video tiling)이 설명된다. 각각의 프레임(201)은 "타일들" T1 내지 T8이라고 지칭되는 8개의 부분들(여기서 직사각형 부분들)로 분할된다. 타일들의 수 및 형상은 상이할 수 있다. 이하에서, 비디오 프레임의 인덱스가 무엇이든 간에 타일링이 동일한 것으로 간주된다.

[0250] 이 타일링의 결과는 8개의 독립적인 서브 비디오(sub-video)들(202)이다. 이 서브 비디오들은 전체 전역 비디오(whole global video)의 파티션(partition)을 표현한다. 각각의 독립적인 서브 비디오는, 예를 들어, AVC 또는 HEVC 표준들에 따라, 독립적인 비트스트림으로서 인코딩될 수 있다. 서브 비디오는 또한, 예를 들어, HEVC 표준의 타일들 또는 AVC 표준의 슬라이스들과 같은, 하나의 단일 비디오 비트스트림의 일부일 수 있다.

[0251] HEVC 표준은 픽처들의 상이한 공간 세분: 타일들, 슬라이스들 및 슬라이스 세그먼트들을 정의한다. 이 상이한 세분들(또는 파티션들)은 상이한 목적들을 위해 도입되었으며: 슬라이스들은 스트리밍 문제들에 관련되어 있는 반면, 타일들 및 슬라이스 세그먼트들은 병렬 처리를 위해 정의되었다.

[0252] 타일은 정수 개의 CTU(Coding Tree Unit)들을 포함하는 픽처의 직사각형 영역을 정의한다. 도 3은 행 및 열 경계들(301, 302)에 의해 정의된 이미지(300)의 타일링을 도시하고 있다. 이것은 타일들을 위치 및 크기의 면에서 관심 영역 디스크립션을 위한 좋은 후보자들로 만든다. 그렇지만, HEVC 표준 비트스트림 조직화는 신택스 및 NAL(Network Abstract Layer) 단위들로의 그의 캡슐화의 면에서 오히려 (AVC 표준에서와 같이) 슬라이스들에 기초한다.

[0253] HEVC 표준에 따르면, 슬라이스는 슬라이스 세그먼트들의 세트이고, 적어도 제1 슬라이스 세그먼트는 독립 슬라이스 세그먼트(independent slice segment)이고, 다른 슬라이스 세그먼트들은, 있는 경우, 종속 슬라이스 세그먼트(dependent slice segment)들이다. 슬라이스 세그먼트는 정수 개의 연속적인 CTU들을 (래스터 스캔 순서로) 포함한다. 슬라이스 세그먼트는 꼭 직사각형 형상을 갖는 것은 아니다(따라서 관심 영역 표현에 타일들보다 덜 적절함). 슬라이스 세그먼트는 "slice_segment_header"라고 불리는 헤더 및 그에 뒤따르는 "slice_segment_data"라고 불리는 데이터로서 HEVC 비트스트림에 인코딩된다. 독립 슬라이스 세그먼트들과 종속 슬라이스 세그먼트들은 그들의 헤더가 상이하며: 종속 슬라이스 세그먼트들은, 독립 슬라이스 세그먼트의 헤더로부터의 정보를 재사용하기 때문에, 보다 짧은 헤더를 갖는다. 독립 슬라이스 세그먼트들 및 종속 슬라이스 세그먼트들 둘 다는, 타일들에의 또는 엔트로피 디코딩 동기화 지점(entropy decoding synchronization point)들에의, 비트스트림에서의 진입 지점들의 리스트를 포함한다.

[0254] 도 3은 슬라이스, 슬라이스 세그먼트들 및 타일들의 이미지들(310 및 320)의 상이한 구성들을 도시하고 있다. 이 구성들은 하나의 타일이 하나의 슬라이스(하나의 독립 슬라이스 세그먼트만을 포함함)를 갖는 이미지(300)의 구성과 상이하다. 이미지(310)는 2개의 수직 타일들(311, 312) 및 하나의 슬라이스(5개의 슬라이스 세그먼트들을 가짐)로 파티셔닝된다. 이미지(320)는 2개의 타일들(321, 322)로 분할되고, 좌측 타일(321)은 2개의 슬라이스들(각각이 2개의 슬라이스 세그먼트들을 가짐)를 가지며, 우측 타일(322)은 하나의 슬라이스(2개의 슬라이스 세그먼트들을 가짐)를 갖는다. HEVC 표준은 다음과 같이 요약될 수 있는 타일들과 슬라이스 세그먼트들 사이의 조직화 규칙(organization rule)들을 정의한다(조건들 중 하나 또는 둘 다가 충족되어야만 함):

[0255] - 슬라이스 세그먼트 내의 모든 CTU들이 동일한 타일에 속하는 것, 및

[0256] - 타일 내의 모든 CTU들이 동일한 슬라이스 세그먼트에 속하는 것.

[0257] 매칭하는 관심 영역 지원 및 전달을 갖기 위해, 하나의 타일이 하나의 독립 세그먼트를 갖는 하나의 슬라이스를 포함하는 구성(300)이 선호된다. 그렇지만, 캡슐화 해결책이 다른 구성들(310 또는 320)에 대해 작동할 것이다.

[0258] 타일이 관심 영역들에 대한 적절한 지원이지만, 슬라이스 세그먼트가 네트워크를 통한 전달을 위해 실제로 NAL 단위(NAL unit)들로 바뀌고(put into) 액세스 단위(access unit)(파일 포맷 레벨의 코딩된 픽처 또는 샘플)를 형성하기 위해 합쳐지는(aggregate) 엔터티이다. HEVC 표준에 따르면, NAL 단위의 타입은 NAL 단위 헤더에 지정되어 있다. "코딩된 슬라이스 세그먼트(coded slice segment)" 타입의 NAL 단위들의 경우, slice_segment_header는 "slice_segment_address" 신택스 엘리먼트를 통해 슬라이스 세그먼트에서의 첫 번째 코딩 트리 블록의 주소를 표시한다. 타일링 정보는 PPS(Picture Parameter Set) NAL 단위로 제공된다. 슬라이스 세그먼트와 타일 간의 관계가 이어서 이 파라미터들로부터 추론할 수 있다.

[0259] 정의에 의해, 타일 경계들에서, 공간 예측들이 리셋된다. 그렇지만, 타일이 참조 프레임(들) 내의 상이한 타일로부터의 시간 예측자(temporal predictor)들을 사용하는 것을 방지하지 못한다. 독립 타일(independent tile)들을 구축(build)하기 위해, 인코딩 시에, 타일 내의 예측 단위들에 대한 움직임 벡터들이 참조 프레임(들)에서의 동일 위치의 타일(co-located tile)에 여전히 있도록 제약된다. 그에 부가하여, 하나의 타일만을 디코딩할 때 오차 드리프트(error drift)가 유입되지 않도록 인-루프 필터(in-loop filter)(디블록킹(deblocking) 및 SAO)들이 타일 경계들에서 비활성화되어야만 한다. 인-루프 필터들의 이 제어는 HEVC 표준에서 이미 이용가능하고, "loop_filter_across_tiles_enabled_flag"라고 불리는 플래그를 이용해 슬라이스 세그먼트 헤더들에 설정(set)되어 있다. 이 플래그를 0으로 명시적으로 설정하는 것에 의해, 타일 경계들에 있는 픽셀들은 이웃 타일들의 경계에 있는 픽셀들에 의존하지 않는다. 움직임 벡터들에 관한 그리고 인-루프 필터들에 관한 2개의 조건들이 충족될 때, 타일들은 "독립적으로 디코딩가능한 타일들(independently decodable)" 또

는 "독립 타일들(independent)"이라고 말해진다.

[0260] 비디오 시퀀스가 독립 타일들의 세트로서 인코딩될 때, 비디오 시퀀스가, 참조 데이터의 누락 또는 재구성 오차(reconstruction error)들의 전파의 위험을 감수하는 일 없이, 타일 기반 디코딩을 사용하여 하나의 프레임으로부터 다른 프레임으로 디코딩될 수 있다. 이 구성은, 예를 들어, 관심 영역에 대응하는 원본 비디오의 공간 파트만을 재구성하는 것을 가능하게 한다.

[0261] 이하에서는, 독립 타일들이 고려된다.

[0262] 도 4를 참조하여, 타일들을 ISOBMFF 파일 포맷으로 캡슐화하는 것이 설명된다. 예를 들어, 각각의 타일이 전용 트랙 내에 캡슐화된다. 모든 타일들에 공통인 셋업(setup) 및 초기화(initialization) 정보는, 예를 들어, "타일 베이스 트랙(tile base track)"이라고 불리는, 특정 트랙 내에 캡슐화된다. 풀 비디오(full video)는 따라서 이 트랙들 전부, 즉 타일 베이스 트랙과 타일 트랙(tile track)들의 세트의 합성(composition)으로서 캡슐화된다.

[0263] 도 4는 예시적인 캡슐화를 예시하고 있다. ISOBMFF 표준에 따라 타일링된 비디오를 캡슐화하는 하나의 방법은 모든 타일들에 공통인 셋업 및 초기화 정보를, 예를 들어, "타일 베이스 트랙"이라고 불리는, 특정 트랙에 캡슐화하기 위해 그리고 풀 비디오를 이 트랙들 전부: 타일 베이스 트랙 플러스 타일 트랙들의 세트의 합성으로서 캡슐화하기 위해, 각각의 타일을 전용 트랙으로 분할하는 것이다. 캡슐화는 따라서 "다중 트랙 타일 캡슐화(multi-track tile encapsulation)"라고 지칭된다. 다중 트랙 타일 캡슐화의 일 예가 도 4에 제공되어 있다.

[0264] 박스(401)는 메인 ISOBMFF 박스(main ISOBMFF box) 'moov'를 나타내고, 트랙들의 전체 리스트를 트랙들의 식별자들과 함께 포함한다. 예를 들어, 박스들(411 내지 414)은 타일 트랙들(본 예에서의 4개의 타일들)을 나타내고, 박스(420)는 타일 베이스 트랙을 나타낸다. 오디오 또는 텍스트 트랙들과 같은 부가의 트랙들이 사용되고 동일한 파일에 캡슐화될 수 있다. 그렇지만, 간결함을 위해, 이러한 부가의 트랙들은 여기서 논의되지 않는다.

[0265] 도 4에 나타낸 바와 같이, 타일 트랙(들)의 임의의 조합이 디코딩 및 디스플레이를 위해 타일 트랙들을 참조하여 타일 베이스 트랙으로부터 쉽게 재구성될 수 있도록, 타일 데이터가 독립적이고 어드레싱가능한(addressable) 트랙들로 분할된다. 타일 베이스 트랙은, 임의의 타일들: 하나의 타일, 다수의 타일들 또는 모든 타일들의 조합을 가능하게 하도록 설계되어 있기 때문에, "합성 트랙(composite track)" 또는 "참조 트랙(reference track)"이라고도 지칭될 수 있다. 타일 베이스 트랙(420)은 모든 타일 트랙들에 대한 공통 정보 및 "mdat" 박스 내의 샘플들(450)의 리스트(첫 번째 샘플만이 도 4에 나타내어져 있음)를 포함한다. 타일 베이스 트랙(420)의 각각의 샘플(450)은, 추출기(extractor)들(451 내지 454, 각각은 각각의 타일에 대한 하나의 추출기를 나타냄)의 사용을 통해, 각각의 타일 트랙을 참조하여 구축된다. 각각의 타일 트랙(411 내지 414)은 전체 비디오(whole video) 또는 풀 프레임 비디오(full-frame video)의 공간 파트를 나타낸다. 타일 디스크립션(위치, 크기, 대역폭 등)은 각각의 타일 트랙(411 내지 414)의 트랙 헤더 박스들(도시되지 않음)에 저장된다. 타일 베이스 트랙 및 각각의 타일 트랙은 각각의 트랙에서의 "TrackReferenceBox" 박스를 사용하여 상호 참조된다(405). 각각의 타일 트랙(411 내지 414)은 타일 베이스 트랙(420)을 'tbas' 트랙이라고 지칭한다('tbas'는 각각의 타일 트랙과 타일 베이스 트랙 간의 코딩 종속성(coding dependency), 특히 파일 포맷 파싱의 결과, 기본 스트림(elementary stream)을 처리할 비디오 디코더를 셋업하는 것을 가능하게 하는 파라미터 "HEVCDecoderConfigurationRecord"를 어디에서 찾아야 하는지를 표시하는 특정 코드임). 이와 달리, 풀 비디오 재구성(full-video reconstruction)을 가능하게 하기 위해, 타일 베이스 트랙(420)은 각각의 타일 트랙(405)에 대한 'scal' 타입의 종속성을 표시한다. 이것은 코딩 종속성을 표시하고 타일 베이스 트랙의 샘플(450)을 추출기들로서 정의한 것을 타일 트랙 데이터(tile tracks data)에 반영하기 위한 것이다. 이 추출기들은, 파싱 시에, 데이터의 부재를 지원할 수 있는 특정 추출기들이다. 도 4에서, 파일의 스트리밍가능 버전을 제공하기 위해, 각각의 트랙이 미디어 세그먼트들(타일 트랙들에 대한 431 내지 434, 타일 베이스 트랙에 대한 460)로 분해된다. 각각의 미디어 세그먼트는, 'moof' 박스 플러스 데이터로 표시된, 하나 이상의 무비 프래그먼트(movie fragment)들을 포함한다. 타일 트랙들에 대해, 데이터 파트는 비디오의 공간적 서브 파트에 대응하는 반면, 타일 베이스 트랙에 대해, 데이터 파트는 파라미터 세트들, 존재하는 경우 SEI 메시지들 및 추출기들의 리스트를 포함한다. 스트리밍 적용분야의 경우에 "moov" 박스(401)는 초기화 세그먼트에 적합(fit in)할 것이다. 도 4는 하나의 세그먼트만을 예시하고 있지만, 트랙들이 임의의 수의 세그먼트들로 분해될 수 있으며, 제약조건은 타일 트랙들에 대한 그리고 타일 베이스 트랙에 대한 세그먼트들이 동일한 시간 분해(temporal decomposition)를 따라야 한다(즉, 세그먼트들이 시간적으로 정렬(temporally align)됨)는 것이고, 이것은 풀 비디오와 타일 또는 타일들의 세트 간의 전환(switching)을 가능하게 하기 위한 것이다. 이 시간 분해의 입도

(granularity)는, 간결함을 위해, 여기서 설명되지 않는다.

[0266] 하나의 타일, 타일들의 조합 또는 모든 타일들에 대응하는 데이터가 기술적 메타데이터를 파싱하는 것에 의해 쉽게 식별될 수 있도록, 파일 포맷은 트랙들 사이의 관계들을 기술하는 기술적 메타데이터(예를 들어, "VisualSampleGroupEntries", 또는 'tref' 박스들 내의 트랙 참조 타입들 등)를 갖는다.

[0267] 이하에서는, 스틸 이미지들이 동일한 레벨에서 기술된다. 따라서, 픽처의 임의의 타일들, 타일들의 조합 또는 모든 타일들의 사용자 선택 시에, 식별 및 추출이 용이하게 된다. 픽처들이 비디오 데이터와 혼합되어 있는 경우에, 디스크립션이 비디오에 대한 기술적 메타데이터와 동시에 나온다. 따라서, 동일한 데이터 세트에 대해, (비디오에 대한 그리고 오디오에 대한 인덱스화 레이어(indexation layer)들에 부가하여) 픽처들에 대한 부가의 인덱스화 레이어가 제공된다.

[0268] 'meta' 박스들을 사용하는 스틸 이미지 파일 포맷들에서, 픽처들이 관련 정보와 함께 정보 아이템들로서 기술된다. 도 5에 예시된 바와 같이, 정보 아이템들은 'meta' 박스의 전용 서브 박스 "ItemInfoBox"(500)에 열거된다. 이 서브 박스는 파일에 존재하는 정보 아이템들의 수를 제공한다. 서브 박스는 또한, 각각의 아이템에 대해, "ItemInfoEntry"(501)로서 표현된 기술적 메타데이터를 제공한다. ISO BMFF 표준 진화에 따라, 이 박스의 몇 개의 버전들(502)(0, 1, 2, 3)이 존재한다.

[0269] "meta" 아이템들이 연속적으로(contiguously) 파일에 저장되지 않을 수 있다. 또한, 아이템 데이터의 인터리빙(interleaving)에 관한 특별한 제한이 없다. 따라서, 동일한 파일 내의 2개의 아이템들이 하나 또는 몇 개의 데이터 블록들을 공유할 수 있다. 이것은, 독립적으로 디코딩가능한 타일마다 하나의 아이템을 갖는 것을 간단하도록 만들어줄 수 있기 때문에, HEVC 타일들(타일들이 연속적으로 저장될 수 있거나 그렇지 않을 수 있음)에 특히 유용하다. 이 아이템은 메인 HEVC 픽처(main HEVC picture)에서의 데이터 오프셋 및 타일에 대해 사용된 슬라이스(들)의 길이를 ItemLocationBox를 통해 표시한다.

[0270] 실시예들에 따르면, 예를 들어, "hvct" 또는 "tile"이라는 이름의, 타일 픽처를 기술하기 위한 새로운 아이템 타입이 추가될 수 있거나 ISO/IEC 14496-15: 'hvt1'로부터 재사용될 수 있다. 타일 픽처를 표현하는 각각의 아이템은 (선택된 4 문자 코드(four character code)가 무엇이든 간에) 그것이 추출된 'hvc1' 아이템에 대한 "tbas" 타입의 참조를 가질 수 있다. 각각의 아이템은 식별자 "item_ID"(503)를 가지며, 픽처들에 대한 압축된 데이터를 포함하는 미디어 데이터 박스에서의 바이트 위치 및 크기로서 "ItemLocationBox" 박스에 추가로 기술된다.

[0271] 이러한 신택스는 파일 포맷 판독기(또는 "파서")가, 정보 아이템들의 리스트를 통해, 몇 개의 정보 아이템들이 이용가능한지를 그들의 타입(504) - 예를 들어, 정보 아이템이 풀 픽처의 타일 픽처라는 것을 표시하기 위한 'tile' - 에 관한 정보를 이용해 결정하는 것을 가능하게 한다.

[0272] 따라서, 다른 타일들을 스킵(skip)하면서, 이미지의 하나의 타일 및 연관된 디코더 구성만을 다운로드하기 위해, 파일 내의 정보 아이템들의 서브세트, 이들의 조합, 또는 정보 아이템들의 풀 세트(full set)를 선택하는 것이 가능하게 된다.

[0273] HEVC 타일이 디코딩을 위해 다른 HEVC 타일에 의존하는 경우에, 문서 w14123, WD of ISO/IEC 14496-15:2013 AMD 1, "Enhanced carriage of HEVC and support of MVC with depth information", MPEG 107 San Jose January 2014에 설명된 바와 같이, 종속성이 'dpnd' 타입의 아이템 참조(또는 코딩 종속성을 표시하는 임의의 특정 4 문자 코드)에 의해 표시되어야 한다.

[0274] 이 문서는 HEVC 타일 NALU들을 ("TileRegionGroupEntry" 디스크립터를 사용하여) 타일의 공간 위치를 표시하는 샘플 그룹 디스크립션(sample group description)들과 연관시키기 위한 도구들을 정의한다. 그렇지만, 이 디스크립터들의 재사용을 가능하게 할 수 있는, 메타데이터 정보 아이템들에 대한 샘플 그룹화의 직접적인 등가물이 없다.

[0275] 따라서, 실시예들에 따르면, 타일 디스크립션 아이템이 타일마다 정의되고, 이하에서 설명되는 바와 같은 "ItemReferenceBox" 박스의 수정된 버전을 사용하여 타일이 그의 디스크립션에 링크된다.

[0276] 다른 실시예들에 따르면, 하나의 타일링 디스크립션만이, 바람직하게는 일반적 방식으로(in a generic way), 제공된다. 따라서, 아이템 리스트가 별로 길지 않게 된다.

[0277] 설계는 다음과 같을 수 있다:

[0278] - 일부 아이템들이, 샘플 그룹들과 유사하지만 각각의 아이템 타입에 특정적인, 메타데이터의 세트를 기술하는 것을 가능하게 하는 것,

[0279] - 임의의 아이템에 대해, 주어진 타입의 아이템 참조에 대한 하나의 파라미터를 기술하는 기능을 추가하는 것. 파라미터는 그러면 참조된 아이템의 타입에 의존하여 해석될 것이다(그룹화 타입과 유사함).

[0280] 도 6을 참조하여 이하에서 설명되는 바와 같이, 정보 아이템에 대한 기술적 메타데이터의 업그레이드가 필요할 수 있다.

[0281] ISOBMFF 표준에 따르면, 샘플 그룹화 메커니즘은 다음과 같이 "grouping_type" 파라미터를 갖는 2개의 메인 박스들에 기초한다:

[0282] - "SampleGroupDescriptionBox" 박스는 속성들의 리스트("SampleGroupEntry" 리스트)를 정의하는 'sgpd' 파라미터를 갖는다.

[0283] - "SampleToGroupBox" 박스는 샘플 그룹을 속성에 매핑하는 것에 의해 샘플 그룹의 리스트를 정의하는 'sbgp' 파라미터를 갖는다.

[0284] "grouping_type" 파라미터는 샘플 그룹들의 리스트를 속성들의 리스트에 링크시키고, 샘플 그룹을 리스트 내의 하나의 속성에 매핑하는 것은 "SampleToGroupBox" 박스에 지정되어 있다.

[0285] 정보 아이템들에 대해 동일한 기능을 제공하기 위해, 정보 아이템 그룹들의 리스트 및 속성들의 리스트가 기술되어야만 한다. 또한, 정보 아이템들의 각각의 그룹을 속성에 매핑하는 것이 가능해야만 한다.

[0286] 이하에서, 이러한 기술적 메타데이터가 스틸 이미지 파일 포맷에 임베딩되는 것을 어떻게 가능하게 할지가 설명된다. 환언하면, 디스크립터를 이미지 아이템에 어떻게 링크시킬지가 설명된다. HEVC 스틸 이미지 파일 포맷에 대한 사용 사례들이 설명되었지만, ISO/IEC 14496-12와 같은 다른 표준들에서 임의의 종류의 정보 아이템을 부가의 기술적 메타데이터와 연관시키기 위해 하기의 특징들이 사용될 수 있다.

[0287] 실시예들에 따르면, 도 6에 도시된 바와 같이 "iref_type"(604)이라고 불리는 새로운 파라미터를 통해 각각의 아이템을 속성에 링크시키기 위해 새 버전 번호(new version number)(602 및 603)를 이용해 'infe' 파라미터를 갖는 기존의 "ItemInformationEntry" 박스(601)가 확장된다. 이것은 새로운 박스들의 생성을 피하는 것을 가능하게 하고, 디스크립션을 짧게 유지하면서 디스크립션을 개선시킨다.

[0288] ItemInformationEntry 박스의 원래의 정의는 하기에 의해 주어진다:

[0289]
```
if (version >= 2) {
```

[0290]
```
        if (version == 2) {
```

[0291]
```
            unsigned int(16)        item_ID;
```

[0292]
```
        } else if (version == 3) {
```

[0293]
```
            unsigned int(32)        item_ID;
```

[0294]
```
        }
```

[0295]
```
        unsigned int(16)        item_protection_index;
```

[0296]
```
        unsigned int(32)        item_type;
```

[0297]
```
        string                  item_name;
```

[0298]
```
        if (item_type=='mime') {
```

[0299]
```
            string              content_type;
```

[0300]
```
            string              content_encoding;        //optional
```

[0301]
```
        } else if (item_type == 'uri ') {
```

[0302]
```
            string              item_uri_type;
```

[0303]
```
        }
```

[0304] }

[0305] 타일 픽처를 그의 디스크립션에 링크시키는 새 버전 만들기(new version making)는 다음과 같을 수 있다:

[0306]
```
if (version >= 2) {
        if (version == 2) {
                unsigned int(16)        item_ID;
        } else if (version >= 3) {
                unsigned int(32)        item_ID;
        }
        unsigned int(16)        item_protection_index;
        unsigned int(32)        item_type;
        string                          item_name;
        if (item_type=='mime') {
                string                  content_type;
                string                  content_encoding;       //optional
        } else if (item_type == 'uri ') {
                string                  item_uri_type;
        }
        if (version == 4) {
                unsigned int(32) item_iref_parameter_count;
                for (i=0 ; i< item_iref_parameter_count ; i++) {
                        unsigned int(32) iref_type;
                        unsigned int(32) iref_parameter;
                }
        }
}
```

[0329] 다른 실시예들에 따르면, "SampleToGroupBox" 박스에 보다 가깝게, 4 문자 코드 'iinf'를 갖는 "ItemInformationBox" 박스의 정의가, 예를 들어, 이 박스의 새 버전을 도입하는 것에 의해 다음과 같이 변경된다:

[0330] 현재 버전:

[0331]
```
aligned(8) class ItemInfoBox
                extends FullBox('iinf', version, 0) {
        if (version == 0) {
                unsigned int(16)        entry_count;
        } else {
                unsigned int(32) entry_count;
        }
```

[0338] ItemInfoEntry[entry_count] item_infos;

[0339] }

[0340] 이 다음으로 변경된다:

[0341] aligned(8) class ItemInfoBox extends FullBox('iinf', **version = 2**, 0) {

[0342] unsigned int(16)group_entry_count;

[0343] for (int g=0; g< group_entry_count;g++){

[0344] unsigned int(16) item_run;

[0345] unsigned int(16) grouping_type;

[0346] unsigned int(16) property_index;

[0347] unsigned int(32) entry_count;

[0348] ItemInfoEntry[entry_count] item_infos;

[0349] }

[0350] unsigned int(16) remaining_entry_count;

[0351] ItemInfoEntry[remaining_entry_count] item_infos;

[0352] }

[0353] 대안적으로, 그룹이 사용 중인지 여부를 신호하기 위해, 현재 버전이 다음으로 변경된다:

[0354] aligned(8) class ItemInfoBox extends FullBox('iinf', **version = 2**, 0) {

[0355] unsigned int(1)group_is_used;

[0356] if (group_is_used == 0){ // standard iinf box but with 1 additional byte overhead

[0357] unsigned int(7)reserved; // for byte alignment

[0358] unsigned int(32) entry_count;

[0359] ItemInfoEntry[entry_count] item_infos;

[0360] } else {

[0361] unsigned int(15)group_entry_count;

[0362] for (int g=0; g< group_entry_count;g++){

[0363] unsigned int(16) item_run;

[0364] unsigned int(16) grouping_type;

[0365] unsigned int(16) property_index;

[0366] unsigned int(32) entry_count;

[0367] ItemInfoEntry[entry_count] item_infos;

[0368] }

[0369] unsigned int(16) remaining_entry_count;

[0370] ItemInfoEntry[remaining_entry_count] item_infos;

[0371] }

[0372] }

[0373] "group_entry_count" 파라미터는 미디어 파일 내의 정보 아이템 그룹들의 수를 정의한다. 각각의 정보 아이템

그룹에 대해, item_ID = 0부터 시작하여, 정보 아이템들의 수가 표시된다. 정보 아이템들이, 샘플들과는 달리, 시간 제약조건들 및 관계들을 갖지 않기 때문에, 캡슐화 모듈은 임의의 순서로 정보 아이템 식별자들을 할당할 수 있다. 아이템 그룹 이후에 증가하는 식별자 번호들을 할당하는 것에 의해, 정보 그룹의 리스트가 그룹에서의 연속적인 정보 아이템 식별자들의 런(run)들을 식별해주는 item_run 파라미터를 사용하여 보다 효율적으로 표현될 수 있다.

[0374] 관련 정보 아이템들은, 예를 들어, "property_index"라고 불리는 인덱스를 갖는다. "grouping_type" 파라미터와 연관된 이 "property_index" 파라미터는 파일 포맷 파서(또는 "판독기")가 기술적 메타데이터에 대한 참조 또는 기술적 메타데이터 자체를 식별하는 것을 가능하게 한다. 도 7은 2개의 예시적인 실시예들을 예시하고 있다.

[0375] "SingleItemTypeReferenceBox" 박스(701)의 그룹 특징(group feature)이, from_item_ID 파라미터의 값에 대해 보통 사용되는 정보 아이템 ID(information item identification)(item_ID) 대신에, 그룹 ID(group identification) "group_ID"와 함께 사용될 수 있다. 설계에 의해, "SingleItemTypeReferenceBox" 박스는 특정 종류의 또는 특정 아이템으로부터의 모든 참조들을 찾아내는 것을 보다 쉽도록 만든다. 이 박스를 "item_ID" 대신에 "group_ID"와 함께 사용하는 것은 특정 타입의 모든 참조들을 쉽게 식별해주는 아이템들의 그룹을 찾아내는 것을 가능하게 한다. 유리하게도, 캡슐화된 파일마다 최대 하나의 "ItemInformationBox" 박스가 있기 때문에, 그룹 ID(group identification)들을 정의할 필요가 없다. 캡슐화 모듈(인코딩 동안) 및 파싱 모듈(디코딩 동안)은, 정보 아이템 그룹들이 생성되거나 판독될 때, 정보 아이템 그룹들의 리스트에 대해("ItemInformationBox" 박스에서의 "g" 변수인) 각자의 카운터를 실행할 수 있다. 대안으로서, 파서는, "group_used_flag" 플래그를 사용하여, 그룹 ID 카운터(group identification counter)를 유지할지 여부를 알 수 있다.

[0376] 타일 픽처들에 대응하는 하나의 정보 아이템 그룹을 갖는 예로 돌아가서, 하나의 그룹은 4개의 엔트리들을 포함할 수 있고, "SingleItemTypeReference" 참조(700)는 4개의 타일 픽처 정보 아이템들이 의존하는 정보 아이템들(704)의 리스트를 표시할 수 있으며, 특정의 참조 타입(703)에 대해 마찬가지이다.

[0377] 다른 예시적인 실시예들에 따르면, 정보 아이템은, 하나의 아이템(722)으로부터, 다양한 다른 정보 아이템들(724)에 대한 다수의 참조 타입들(723)을 열거하는 것을 가능하게 하는, 이후부터 설명되는 바와 같은, 새로운 종류의 박스 "ItemReferenceBox"에서 사용된다.

[0378] 후자의 경우에, 특정 박스 "ItemReferenceBox"(721)는 다음과 같이 구현될 수 있다:

[0379] aligned(8) class MultipleItemTypeReferenceBox(void) extends

[0380] Box(void) {

[0381] unsigned int(16) from_item_ID;

[0382] unsigned int(16) reference_count;

[0383] for (j=0; j<reference_count; j++) {

[0384] **unsigned int(32) reference_type; // new parameter to allow multiple types**

[0385] unsigned int(16) to_item_ID;

[0386] }

[0387] }

[0388] 표준의 박스 "ItemInformationBox"에 관해서는, 아이템 엔트리들의 리스트가 기술되지만, 이번에는 그룹화에 의존하는 상이한 순서를 갖는다. 타일 예에서, 이것은 'tile'이라고 명명될 수 있는 파라미터를 이용해 그룹을 이루어 모여 있는 타일 픽처들에 대응하는 4개의 정보 아이템들의 제1 그룹 및 그에 뒤이어서 구성 정보에 대한, 풀 픽처 정보 아이템에 대한 그리고 임의로 EXIF 메타데이터에 대한 비-그룹화된 정보 아이템들을 가져올 수 있다.

[0389] 따라서, 하나의 박스가 수정되고 특정 종류의 ItemReferenceBox인 하나의 박스가 생성된다. 이하에서, 이 새로운 종류의 ItemReferenceBox가 설명된다.

[0390] "ItemReferenceBox" 박스는 또한 다음과 같이 ItemReferenceBox의 일부인 "FullBox" 박스의 플래그 파라미터들

을 사용하여 다양한 종류들의 ItemReferenceBox를 구별하는 것에 의해 확장될 수 있다:

[0391] aligned(8) class ItemReferenceBox extends FullBox('iref', 0, flags) {

[0392] switch (flags) {

[0393] case 0:

[0394] SingleItemTypeReferenceBox references[];

[0395] break;

[0396] case 1:

[0397] *MultipleItemTypeReferenceBox* references[];

[0398] break;

[0399] case 2:

[0400] SharedItemTypeReferenceBox references[];

[0401] break;

[0402] }

[0403] }

[0404] "MultipleItemTypeReferenceBox" 박스(721)를 사용하여, 4개의 타일들을 갖는 하나의 픽처가 다음과 같이 기술될 수 있다:

[0405] Item Reference Box (version=1 or flags=1):

[0406] fromID=2, ref_count=1, type='cdsc', toID=1;

[0407] fromID=1, ref_count=1, type='init', toID=3;

[0408] fromID=4, ref_count=2, type='tbas', toID=1, type='tile' toID=8;

[0409] fromID=5, ref_count=2, type='tbas', toID=1, type='tile' toID=8;

[0410] fromID=6, ref_count=2, type='tbas', toID=1, type='tile' toID=8;

[0411] fromID=7, ref_count=2, type='tbas', toID=1, type='tile' toID=8;

[0412] 이 설계는 특정 아이템으로부터의 임의의 종류들의 모든 참조들을 찾아내는 것을 상당히 더 쉽도록 만든다.

[0413] 주어진 타입(713)을 갖는 동일한 아이템(714)을 참조하는 아이템들(712)의 리스트에 대한 디스크립션 지원(description support)(711)은 다음과 같을 수 있다:

[0414] aligned(8) class SharedItemTypeReferenceBox(ref_type) extends

[0415] Box(referenceType) {

[0416] unsigned int(16) reference_count;

[0417] for (j=0; j<reference_count; j++) {

[0418] unsigned int(16) from_item_ID;

[0419] }

[0420] unsigned int(16) to_item_ID;

[0421] }

[0422] }

[0423] 4개의 타일들을 갖는 픽처의 예에서, 다음과 같을 수 있다:

[0424] type='cdsc', ref_count=1, fromID=2, toID=1;

[0425] type='init', ref_count=1, fromID=1, toID=3;

[0426] type='tbas', ref_count=4, fromID=4, fromID=5, fromID=6, fromID=7, toID=1;

[0427] type='tile', ref_count=4, fromID=4, fromID=5, fromID=6, fromID=7, toID=8;

[0428] "SharedItemTypeReferenceBox" 박스의 설계는 특정 아이템을 가리키는 특정 타입의 모든 참조들을 찾아내는 것을 보다 쉽도록 만든다. 이것은 "SingleItemTypeReferenceBox" 박스와 대조를 이룬다. 그러나 트랙 참조들에 대해 정의된 "reference_type"의 대부분이 양방향이 아니기 때문에, "SingleItemTypeReferenceBox" 박스는 다른 아이템들에 대해 이 참조 타입을 갖는 모든 노드들을 신호하기 위해 일부 단방향 참조 타입과 함께 사용될 수 없다. 대안적으로, 직접 참조(direct reference)인지 역참조(reverse reference)인지를 표시하기 위한 플래그가 "SingleItemTypeReference"에 제공될 수 있고, 그로써 새로운 SharedItemTypeReferenceBox의 필요성을 완화시킬 수 있다.

[0429] 이상의 내용을 고려하여, 정보 아이템이 타일링 정보와 연관될 수 있다. 이 타일링 정보에 대한 설명이 이제 제공될 필요가 있다.

[0430] 예를 들어, 각각의 타일이, 확장된 "ItemInfoEntry"(601)의 "iref_parameter"(605)와 같은, 타일 디스크립터(tile descriptor)를 사용하여 기술될 수 있다. 특정 디스크립터는 다음과 같을 수 있다:

[0431] aligned(8) class TileInfoDataBlock() {

[0432] unsigned int(8) version;

[0433] unsigned int(32) reference_width; // full image sizes

[0434] unsigned int(32) reference_height;

[0435] unsigned int(32) horizontal_offset; // tile positions

[0436] unsigned int(32) vertical_offset;

[0437] unsigned int(32) region_width; // tile sizes

[0438] unsigned int(32) region_height;

[0439] }

[0440] 실시예들에 따르면, 저장될 하나 이상의 픽처들에 적용할 타일들의 격자에 대한 디스크립터가 사용될 수 있다.

[0441] 이러한 디스크립터는 다음과 같을 수 있다:

[0442] aligned(8) class TileInfoDataItem () {

[0443] unsigned int(8) version;

[0444] unsigned int(1) regular_spacing; // regular grid or not

[0445] unsigned int(7) reserved = 0;

[0446] unsigned int(32) reference_width; // full-frame sizes

[0447] unsigned int(32) reference_height;

[0448] unsigned int(32) nb_cell_horiz;

[0449] unsigned int(32) nb_cell_vert;

[0450] if (!regular_spacing) {

[0451] for (i=0; i<nb_cell_width; i++)

[0452] unsigned int(16) cell_width;

[0453] for (i=0; i<nb_cell_height; i++)

[0454] unsigned int(16) cell_height;

[0455] }

[0456] }

[0457] }

[0458] 이 디스크립터 "TileInfoDataItem"은 타일링 격자(정규(regular) 또는 비정규(irregular))를 기술하는 것을 가능하게 한다. 격자는 좌측 상단으로부터 시작하여 행 단위로(rows by rows) 기술된다.

[0459] 디스크립터는 'tile' 타입의 아이템으로서 저장되어야 한다. 다른 아이템이 이 아이템을 참조할 때, 다른 아이템은 이 디스크립션에 대한 "tile" 타입의 참조를 사용해야 하고 다른 아이템은 "iref_parameter" 파라미터를 지정해야 하며, 이 파라미터의 값은 디스크립터에 의해 정의된 격자 내의 셀의 0 기반 인덱스(0-based index)이고, 여기서 0은 좌측 상단 아이템이고 1은 셀 0의 바로 오른쪽에 있는 셀이며, 이하 마찬가지이다.

[0460] 디스크립터에서:

[0461] - "version"은 TileInfoDataItem에 대한 신택스의 버전을 표시한다. 값 0만이 정의된다.

[0462] - "regular_spacing"은 격자 내의 모든 타일들이 동일한 폭 및 동일한 높이를 갖는지를 표시한다.

[0463] - "reference_width, reference_height"는 격자가 기술되는 단위들을 표시한다. 이 단위들은 이 아이템을 참조하는 이미지의 픽셀 해상도와 매칭할 수 있거나 그렇지 않을 수 있다. 격자가 정규인 경우, "reference_width"(각각, "reference_height")는 "nb_cell_horiz"(각각, "nb_cell_vert")의 배수여야 한다.

[0464] - "cell_width"는, 좌측으로부터 시작하는, 비정규 타일(non-regular tile)들에서의 격자의 수평 분할(horizontal division)을 제공한다.

[0465] - "cell_height"는, 상단으로부터 시작하는, 비정규 타일들에서의 격자의 수직 분할(vertical division)을 제공한다.

[0466] 이상의 접근법은 모든 타일들에 대한 타일링 정보를 공유하는 것을 가능하게 한다.

[0467] 더욱이, 다수의 픽처들이 동일한 타일링을 공유하는 경우에, 타일들의 격자 내의 셀을 단순히 참조하는 것에 의해 훨씬 더 많은 디스크립션이 공유될 수 있다.

[0468] 타일링 구성은 미디어 데이터 박스에 또는 타일 정보 아이템들 간에 (참조에 의해) 공유되는 전용 박스에 넣어질 수 있다.

[0469] 이상의 디스크립터들은 보다 큰 이미지 내의 서브 이미지(들)에 대한 공간 위치(spatial location)들 및 크기들만을 제공한다는 의미에서 순수 공간 디스크립터들이다. 일부 사용 사례들에서, 예를 들어, 이미지 컬렉션들 또는 이미지 합성에서, 전형적으로 이미지들이 오버랩할 때, 이미지를 기술하는 데 공간 위치로는 충분하지 않다. 이것이 이상의 TileInfoDataBlock 디스크립터의 하나의 한계이다. 이미지 합성을 가능하게 하기 위해, 이미지가 무엇이든 간에, 즉 타일 또는 독립/전체 이미지든 간에, 한편으로는 이미지의 위치들 및 크기들(공간적 관계들)을 그리고 다른 한편으로는 그 픽처에 대한 디스플레이 정보(색상, 크로핑(cropping) ...)를 정의하는 것이 유용할 수 있다. 예를 들어, 디스플레이하기 위해 서브 이미지를 하나의 색 공간으로부터 다른 색 공간으로 변환하기 위해 색상 정보가 제공될 수 있다. 이 종류의 정보는 ISOBMFF의 ColorInformationBox 'colr'에서 전달될 수 있다. 2개의 상이한 그렇게 변환된 픽처들을 전달하는 것보다는 단지 적용할 변환 파라미터들을 제공하는 것에 의해 상이한 종류의 디스플레이를 위해 동일한 데이터를 준비하는 것이, 컴팩트성(compacity)을 위해, 유용할 수 있다. 또한, 각각의 픽처의 인코딩된 폭 및 높이와 상이할 수 있는 폭 및 높이를 재정의하기 위해 ISOBMFF Part-12에 정의된 PixelAspectRatio 박스 'pasp'와 같은 픽셀 종횡비(pixel aspect ratio)가 이 디스크립터에 넣어질 수 있다. 이것은 이미지의 디코딩 이후에 디스플레이에 의해 적용할 스케일 비율(scale ratio)을 표시할 것이다. 그러면 비디오 샘플 엔트리들(예를 들어, 'stsd' 박스)에 저장된 코딩된 크기들 및 'pasp' 박스로부터 추론된 디스플레이 크기들이 얻어질 것이다. 디스플레이에 대한 다른 가능한 정보는 ISOBMFF에 역시 정의된 클린 애퍼처(clean aperture) 정보 박스 'clap'일 수 있다. SMPTE 274M 표준에 따르면, 클린 애퍼처는 픽처 정보가 주관적으로 모든 에지 과도 왜곡(edge transient distortion)들(아날로그-디지털 변환들 이후에 이미지들의 경계들에서 있을 수 있는 링잉 효과(ringing effect)들)에 의해 오염되지 않는 구역(area)을 정의한다. 디스플레이에 대해 유용한 이 파라미터들의 리스트는 제한적인 것이 아니며, 우리

는 임의의 다른 기술적 메타데이터 박스를 임의적인 컴포넌트들로서 서브 이미지 디스크립터에 넣을 수 있다. 이 파라미터들은 명확히 언급될 수 있는데, 그 이유는 그들이 이미 표준의 일부이고 이미지 크로핑, 샘플 종횡비 수정 및 색상 조절들을 표시하기 위한 제네릭 도구(generic tool)들을 제공하기 때문이다. 안타깝게도, 그들의 사용은, 'meta' 박스들에 의존하는 이미지 파일 포맷이 아니라, 미디어 트랙들에 대해서만 가능하였다. 우리는 이어서 클린 애퍼처 또는 샘플 종횡비와 같은 다른 속성들과 함께, 이미지 아이템들의 공간 디스크립션(spatial description)을 지원하기 위해, 예를 들어, "SimpleImageMetaData"라고 불리는 새로운 디스크립터를 제안한다. 이것은 보다 큰 이미지에 합성되거나 거꾸로 보다 큰 이미지로부터 추출되도록 의도되어 있는 임의의 서브 이미지(타일 또는 독립 이미지)에 적용된다.

[0470] aligned(8) class SimpleImageMetaData {

[0471] CleanApertureBox clap; // optional

[0472] PixelAspectRatioBox pasp; // optional

[0473] ColourInformationBox colour; // optional

[0474] ImageSpatialRelationBox location; // optional

[0475] }

[0476] 또는 (예를 들어, extra_boxes를 통해) 디스플레이 프로세스를 돕기 위해 확장 파라미터들을 고려할 때의 그의 변형:

[0477] aligned(8) class SimpleImageMetaData {

[0478] CleanApertureBox clap; // optional

[0479] PixelAspectRatioBox pasp; // optional

[0480] ColourInformationBox colour; // optional

[0481] ImageSpatialRelationBox location; // optional

[0482] extra_boxes boxes; // optional

[0483] }

[0484] 여기서 ImageSpatialRelationBox는 이하에서 설명되는 바와 같이 TileInfoDataBlock의 확장이다. 고려해야 할 다른 유용한 파라미터는 이미지들을 레이어(layer)들로서 합성할 가능성이다. 우리는 이어서 이 레이어 합성(layered composition)에서 이미지에 연관된 레벨을 표시하기 위한 파라미터를 삽입하는 것을 제안한다. 이것은 전형적으로 이미지들이 오버랩할 때 유용하다. 이것은, 예를 들어, 레이어 정보 표시를 갖는 'layer'라고 불리울 수 있다. 이러한 디스크립터에 대한 예시적인 신택스가 제공된다:

[0485] 정의:

[0486] Box Type: 'isre'

[0487] Container: Simple image meta-data item ('simd')

[0488] Mandatory: No

[0489] Quantity: Zero or one per item

[0490] 신택스:

[0491] aligned(8) class ImageSpatialRelationBox

[0492] extends FullBox('isre, version = 0, 0) {

[0493] unsigned int(32) horizontal_display_offset;

[0494] unsigned int(32) vertical_display_offset;

[0495] unsigned int(32) display_width;

[0496] unsigned int(32) display_height;

[0497] int(16) layer;

[0498] }

[0499] 연관된 시맨틱스:

[0500] horizontal_display_offset은 이미지의 수평 오프셋을 지정한다.

[0501] vertical_display_offset은 이미지의 수직 오프셋을 지정한다.

[0502] display_width는 이미지의 폭을 지정한다.

[0503] display_height는 이미지의 높이를 지정한다.

[0504] layer는 이미지의 전후 순서(front-to-back ordering)를 지정하고; 보다 낮은 번호들을 갖는 이미지들이 보는 사람(viewer)에 보다 가깝다. 0은 정상 값(normal value)이고, -1은 레이어 0의 전방에 있을 것이며, 이하 마찬가지이다.

[0505] 이 새로운 'isre' 박스 타입은 이미지 컬렉션에서의 다른 이미지들에 대한 이미지의 상대 위치(relative position)를 기술할 수 있다. 이는 미디어 파일의 무비 또는 트랙 헤더 박스에서 보통 발견되는 변환 행렬의 기능들의 서브세트를 제공한다. ImageSpatialRelationBox에서의 좌표들은 컬렉션의 저작자(author)의 의도된 디스플레이 크기를 제공하는 정사각형 격자 상에 표현되고; 이 단위들은 이미지의 코딩된 크기와 매칭하거나 그렇지 않을 수 있다. 의도된 디스플레이 크기는 다음과 같이 정의된다:

[0506] - 수평(Horizontally): 모든 'isre' 박스들에 대한 최댓값(horizontal_display_offset + display_width)

[0507] - 수직(Vertically): 모든 'isre' 박스들에 대한 최댓값(vertical_display_offset + display_height)

[0508] 일부 이미지들은 연관된 어떤 'isre'도 갖지 않는 반면 파일 내의 다른 이미지들은 연관된 'isre'를 가질 때, 어떤 'isre'도 갖지 않는 디폴트 이미지(default image)들은 그들의 수평 및 수직 오프셋들이 0인 것처럼 취급되어야 하고, 그들의 디스플레이 크기는 의도된 디스플레이 크기이며, 그들의 레이어는 0이다.

[0509] ImageSpatialRelationBox는 임의의 크로핑 또는 샘플 종횡비가 이미지들에 적용된 후의 이미지들의 상대 공간 위치(relative spatial position)를 표시한다. 이것은, SimpleImageMetaData에서 'isre'가 'pasp' 등과 조합될 때, 이미지가 디코딩되고, 'pasp', 'clap', 'colr'가, 존재하는 경우, 적용되며, 이어서 이미지가 'isre' 박스에 선언된 오프셋 및 크기에 따라 이동되고 스케일링된다는 것을 의미한다.

[0510] 이 새로운 디스크립터는 이미지를 표현하는 아이템 정보와 디스크립터를 표현하는 아이템 정보 사이의 연관(association)을 정의하는 것에 의해 이미지(타일 또는 단일 이미지)의 디스크립션으로서 사용될 수 있다(SimpleImageMetadata 정의에 대해 'simd' 타입을 부여하기로 하면, mp4 파서가 현재 처리 중인 메타데이터의 종류를 쉽게 식별하는 데 임의의 예비된 4 문자 코드로도 만족스러울 것이다). 이 연관은 ItemRefererenceBox를 이용해 그리고 새로운 참조 타입; "공간 이미지 관계"를 표시하기 위한 'simr'을 이용해 행해진다. 이하의 예시적인 디스크립션은 합성 자체가 어떤 연관된 아이템도 갖지 않는 4개의 이미지들의 합성의 경우를 예시하고 있다. 각각의 이미지 아이템은 'simr' 타입의 아이템 참조를 통해 SimpleImageMetaData 아이템에 연관되고, 전용 'hvcC' 아이템에서의 DecoderConfigurationRecord 정보를 공유한다.

[0511] ftyp box: major-brand = 'hevc', compatible-brands = 'hevc'

[0512] meta box: (container)

[0513] handler box: hdlr = 'hvc1' // no primary item provided

[0514] 아이템 정보 엔트리들:

[0515] item_type = 'hvc1', itemID=1, item_protection_index = 0

[0516] item_type = 'hvc1', itemID=2, item_protection_index = 0

[0517] item_type = 'hvc1', itemID=3, item_protection_index = 0

[0518] item_type = 'hvc1', itemID=4, item_protection_index = 0

[0519] item_type='simd' itemID=5 (sub-image descriptor)

[0520] item_type='simd' itemID=6 (sub-image descriptor)

[0521] item_type='simd' itemID=7 (sub-image descriptor)

[0522] item_type='simd' itemID=8 (sub-image descriptor)

[0523] item_type = 'hvcC', item_ID=9, item_protection_index = 0 ...

[0524] 아이템 참조:

[0525] type='simr' fromID=1, toID=5

[0526] type='simr' fromID=2, toID=6

[0527] type='simr' fromID=3, toID=7

[0528] type='simr' fromID=4, toID=8

[0529] type='init', fromID=1, toID=9;

[0530] type='init', fromID=3, toID=9;

[0531] type='init', fromID=4, toID=9;

[0532] type='init', fromID=5, toID=9;

[0533] 아이템 위치:

[0534] itemID = 1, extent_count = 1, extent_offset = P1, extent_length = L1;

[0535] itemID = 2, extent_count = 1, extent_offset = P2, extent_length = L2;

[0536] itemID = 3, extent_count = 1, extent_offset = P3, extent_length = L3;

[0537] itemID = 4, extent_count = 1, extent_offset = P4, extent_length = L4;

[0538] itemID = 5, extent_count = 1, extent_offset = P5, extent_length = L5;

[0539] itemID = 6, extent_count = 1, extent_offset = P6, extent_length = L6;

[0540] itemID = 7, extent_count = 1, extent_offset = P7, extent_length = L7;

[0541] itemID = 8, extent_count = 1, extent_offset = P8, extent_length = L8;

[0542] itemID = 9, extent_count = 1, extent_offset = P0, extent_length = L0;

[0543] 미디어 데이터 박스:

[0544] 1 HEVC Decoder Configuration Record ('hvcC' at offset P0)

[0545] 4 HEVC Images (at file offsets P1, P2, P3, P4)

[0546] 4 simple image metadata (at file offsets P5, P6, P7, P8)

[0547] 이상에서의 데이터의 조직화는 일 예로서 제공되고: 예를 들어, 이미지 플러스 그의 메타데이터가 단일 바이트 범위로서 어드레싱가능하도록 이미지와 메타데이터가 미디어 데이터 박스에서 인터레이싱될 수 있다. 이 디스크립션을 수신할 때, 파서는, 'simd' 아이템들 내의 정보들을 파싱하는 것에 의해, 서브 이미지가 풀 픽처로부터 크로핑된 것인지 또는 이와 달리 풀 픽처가 서브 이미지들로부터의 합성인지를 알게 된다. 크로핑의 경우에, 풀 픽처 아이템과 크로핑된 이미지는 이하의 예에서와 같이 동일한 데이터 범위 및 동일한 디코더 구성 정보를 공유한다. 서브 이미지는 그러면 'clap' 정보만을 갖고 위치(positioning)를 갖지 않는, 따라서 'isre'를 갖지 않는 'simd' 아이템에 연관될 것이다.

[0548] 합성의 경우에: 이러한 경우에, 풀 픽처 아이템은 'isre' 정보만을 포함하는 'simd' 아이템에 연관되고, 서브 이미지는 풀 이미지(full image)에서의 그의 위치를 반영하는 'simd' 아이템에 연관될 것이다.

[0549] 이하의 예는 4개의 이미지들이 보다 큰 이미지로 합성되는 경우를 예시하고 있다. 합성된 이미지를 비롯한 모

든 이미지들이 제안된 디스크립터를 사용하여 재생가능 아이템으로서 노출(expose)된다.

[0550] ftyp box: major-brand = 'hevc', compatible-brands = 'hevc'

[0551] meta box: (container)

[0552] handler box: hdlr = 'hvc1' primary item: itemID = 1;

[0553] 아이템 정보 엔트리들:

[0554] item_type = 'hvc1', itemID=1, item_protection_index = 0... // full-image

[0555] item_type = 'hvc1', itemID=2, item_protection_index = 0... // sub-image

[0556] item_type = 'hvc1', itemID=3, item_protection_index = 0... // sub-image

[0557] item_type = 'hvc1', itemID=4, item_protection_index = 0... // sub-image

[0558] item_type = 'hvc1', itemID=5, item_protection_index = 0... // sub-image

[0559] item_type = 'simd' itemID=6 (sub-image descriptor)...

[0560] item_type = 'simd' itemID=7 (sub-image descriptor)...

[0561] item_type = 'simd' itemID=8 (sub-image descriptor)...

[0562] item_type = 'simd' itemID=9 (sub-image descriptor)...

[0563] item_type = 'hvcC', item_ID=10 (decoder config record)

[0564] item_type = 'simd', item_ID=11 (sub-image descriptor)

[0565] 아이템 참조 엔트리들:

[0566] type= 'simr', fromID=1, toID=11

[0567] type= 'simr', fromID=2, toID=6

[0568] type= 'simr', fromID=3, toID=7

[0569] type= 'simr', fromID=4, toID=8

[0570] type= 'simr', fromID=5, toID=9

[0571] type= 'init', fromID=1, toID=10...

[0572] type= 'init', fromID=2, toID=10...

[0573] type= 'init', fromID=3, toID=10...

[0574] type= 'init', fromID=4, toID=10...

[0575] type= 'init', fromID=5, toID=10...

[0576] 아이템 위치:

[0577] itemID = 1, extent_count = 4,// full image is composed of 4 sub-images

[0578] extent_offset = P2, extent_length = L2;

[0579] extent_offset = P3, extent_length = L3;

[0580] extent_offset = P4, extent_length = L4;

[0581] extent_offset = P5, extent_length = L5;

[0582] itemID = 2, extent_count = 1, extent_offset = P2, extent_length = L2;

[0583] itemID = 3, extent_count = 1, extent_offset = P3, extent_length = L3;

[0584] itemID = 4, extent_count = 1, extent_offset = P4, extent_length = L4;

[0585] itemID = 5, extent_count = 1, extent_offset = P5, extent_length = L5;

[0586] itemID = 6, extent_count = 1, extent_offset = P6, extent_length = L6;

[0587] itemID = 7, extent_count = 1, extent_offset = P7, extent_length = L7;

[0588] itemID = 8, extent_count = 1, extent_offset = P8, extent_length = L8;

[0589] itemID = 9, extent_count = 1, extent_offset = P9, extent_length = L9;

[0590] itemID = 10, extent_count = 1, extent_offset = P0, extent_length = L0;

[0591] itemID = 11, extent_count = 1, extent_offset = P10, extent_length = L10;

[0592] 미디어 데이터 박스:

[0593] 1 HEVC Decoder Configuration Record ('hvcC' at offset P0)

[0594] 4 HEVC (sub) Images (at file offsets P2, P3, P4, P5)

[0595] 5 simple image metadata (at file offsets P6, P7, P8, P9, P10)

[0596] 이 다른 예는 풀 픽처가 실제로는 타일링된 HEVC 픽처(4개의 타일들)인 경우를 예시하고 있다.

[0597] ftyp box: major-brand = 'hevc', compatible-brands = 'hevc'

[0598] meta box: (container)

[0599] handler box: hdlr = 'hvc1' primary item: itemID = 1;

[0600] 아이템 정보 엔트리들:

[0601] item_type = 'hvc1', itemID=1, item_protection_index = 0... // full-image

[0602] item_type = 'hvt1', itemID=2, item_protection_index = 0... // sub-image

[0603] item_type = 'hvt1', itemID=3, item_protection_index = 0... // sub-image

[0604] item_type = 'hvt1', itemID=4, item_protection_index = 0... // sub-image

[0605] item_type = 'hvt1', itemID=5, item_protection_index = 0... // sub-image

[0606] item_type = 'simd' itemID=6 (sub-image descriptor)...

[0607] item_type = 'simd' itemID=7 (sub-image descriptor)...

[0608] item_type = 'simd' itemID=8 (sub-image descriptor)...

[0609] item_type = 'simd' itemID=9 (sub-image descriptor)...

[0610] item_type = 'hvcC', item_ID=10 (decoder config record)

[0611] 아이템 참조 엔트리들:

[0612] type= 'init', fromID=1, toID=10...

[0613] // declare sub-images as tiles of the full image

[0614] type= 'tbas', fromID=2, toID=1...

[0615] type= 'tbas', fromID=3, toID=1...

[0616] type= 'tbas', fromID=4, toID=1...

[0617] type= 'tbas', fromID=5, toID=1...

[0618] // providing positions and sizes

[0619] type= 'simr', fromID=2, toID=6

[0620] type= 'simr', fromID=3, toID=7

[0621] type= 'simr', fromID=4, toID=8

[0622] type= 'simr', fromID=5, toID=9

[0623] 아이템 위치:

[0624] itemID = 1, extent_count = 4,// full image is composed of 4 tiles

[0625] extent_offset = P2, extent_length = L2... // data for tile 1

[0626] extent_offset = P3, extent_length = L3... // data for tile 2

[0627] extent_offset = P4, extent_length = L4... // data for tile 3

[0628] extent_offset = P5, extent_length = L5... // data for tile 4

[0629] itemID = 2, extent_count = 1, extent_offset = P2, extent_length = L2;

[0630] itemID = 3, extent_count = 1, extent_offset = P3, extent_length = L3;

[0631] itemID = 4, extent_count = 1, extent_offset = P4, extent_length = L4;

[0632] itemID = 5, extent_count = 1, extent_offset = P5, extent_length = L5;

[0633] itemID = 6, extent_count = 1, extent_offset = P6, extent_length = L6;

[0634] itemID = 7, extent_count = 1, extent_offset = P7, extent_length = L7;

[0635] itemID = 8, extent_count = 1, extent_offset = P8, extent_length = L8;

[0636] itemID = 9, extent_count = 1, extent_offset = P9, extent_length = L9;

[0637] itemID = 10, extent_count = 1, extent_offset = P0, extent_length = L0;

[0638] 미디어 데이터 박스:

[0639] 1 HEVC Decoder Configuration Record ('hvcC' at offset P0)

[0640] 1 HEVC Image (with 4 tiles at file offsets P2, P3, P4, P5)

[0641] 4 simple image metadata (at file offsets P6, P7, P8, P9)

[0642] 사용 사례들에 따라, 예를 들어, 모든 이미지들에 동일한 크로핑이 적용되어야 할 때, 동일한 메타데이터를 공유하는 몇 개의 이미지 아이템들을 갖는 것이 가능할 것이다. 예를 들어, 크로핑은 이미지들 간에 공유되지만 공간 정보는 공유되지 않을 때, 이미지 아이템이 상이한 SimpleImageMetaData에 대한 다수의 'simr' 참조들을 갖는 것이 또한 가능할 것이다.

[0643] (도 6에 예시된 바와 같은) ItemInfoEntry의 새 버전에 대한 대안의 실시예는 정보 아이템 엔트리 및 참조마다 하나 초과의 파라미터(605)를 정의하는 것이다. 도 6의 실시예에서, iref_parameter는 타일 인덱스의 경우에 타일링 격자 내의 셀을 참조하는 데 유용한 4 바이트 코드이다. 그러나 보다 풍부한 디스크립션을 갖기 위해 그리고 링크된 디스크립션(linked description)을, (mdat 박스에) 데이터와 함께보다는, 아이템 정보 엔트리(item info entry) 자체 내에 임베딩할 수 있기 위해, 하기의 확장이 유용할 수 있다:

[0644] if (version == 4) {

[0645] unsigned int(32) item_iref_parameter_count;

[0646] for (i=0 ; i< item_iref_parameter_count ; i++) {

[0647] unsigned int(32) iref_type;

[0648] ItemReferenceParameterEntry parameter;

[0649] }

[0650] aligned(8) abstract class ItemReferenceParameterEntry (unsigned int(32) format)

[0651] extends Box(format){

[0652] }

[0653] // Example to reference a tile index

[0654] aligned(8) abstract class TileIndexItemReferenceParameterEntry

[0655] extends ItemReferenceParameterEntry('tile'){

[0656] unsigned int(32) tile_index;

[0657] }

[0658] // Example to inline the tile description

[0659] aligned(8) abstract class TileIndexItemReferenceParameterEntry

[0660] extends ItemReferenceParameterEntry('tile'){

[0661] unsigned int(32) tile_index;

[0662] }

[0663] 이상의 확장에서:

[0664] - item_iref_parameter_count는 참조 타입들의 수 - 이를 위해 파라미터가 제공됨 - 를 제공한다. 이것은 도 6에서의 아이템(605)과 비교하여 변화가 없다.

[0665] - iref_type은, 'iref' 박스에 표시된 바와 같은, 참조 타입 - 이를 위해 이 아이템에 대해 파라미터가 적용됨 - 을 제공한다. 이것은 도 6에서의 아이템(605)과 비교하여 변화가 없다.

[0666] - parameter는 여기서 iref_parameter(도 6에서의 아이템(605))와 상이한데, 그 이유는 그것이 새로운 박스 ItemReferenceParameterEntry를 통해 확장 수단을 제공하기 때문이다. (타일링 구성에서 타일 인덱스에 대해 TileIndexItemReferenceParameterEntry를 이용하여 앞서 행해진 바와 같이) 이 새로운 박스를 특화(specialize)하는 것에 의해, 캡슐화 및 파싱 모듈들이 이 특화된 박스의 구조체를 인식하기만 한다면 임의의 종류의 부가 메타데이터가 정보 아이템 엔트리와 연관될 수 있다. 이것은 표준 타입들의 ItemReferenceParameterEntry에 의해 또는 파라미터 엔트리의 구조체를 구성(construction)에 의해 또는 협상 단계에서 제공하는 것에 의해 행해질 수있다. 파라미터의 시맨틱스는 iref_type 타입을 갖는 아이템의 시맨틱스에 의해 제공된다.

[0667] 이하에서, 4개의 타일들을 갖는 픽처를 기술하는 정보 아이템들에 대한 예시적인 기술적 메타데이터 및 풀 픽처의 EXIF 메타데이터가 제공된다.

[0668] 종래 기술에서, 타일 픽처들은 본원에서 이하에서 보여지는 바와 같이 제공된 임의의 대응하는 디스크립션을 갖지 않는 정보 아이템들로서 열거되었다. 더욱이, 'hvcC' 타입으로 표시된 셋업 정보(setup information)가 아이템으로서 기술되지 않았다. 이것은 모든 타일 픽처들에 그리고 풀 픽처에 적용되는 HEVC 파라미터 세트들 및 SEI 메시지들에 관련된 공통 데이터를 인수분해(factorize)하는 것을 가능하게 한다.

[0669] ftyp box: major-brand = 'hevc', compatible-brands = 'hevc'

[0670] meta box: (container)

[0671] handler box: hdlr = 'hvc1' primary item: itemID = 1;

[0672] 아이템 정보:

[0673] item_type = 'hvc1', itemID=1, item_protection_index = 0 (unused) => Full pict.

[0674] item_type = 'Exif', itemID=2, item_protection_index = 0 (unused)

[0675] item_type = 'hvcC', itemID=3, item_protection_index = 0 (unused)

[0676] item_type = 'hvct', itemID=4, item_protection_index = 0 (unused) => Tile pict.

[0677] item_type = 'hvct', itemID=5, item_protection_index = 0 (unused) => Tile pict.

[0678] item_type = 'hvct', itemID=6, item_protection_index = 0 (unused) => Tile pict.

[0679] item_type = 'hvct', itemID=7, item_protection_index = 0 (unused) => Tile pict.

[0680] 아이템 위치:

[0681] itemID = 1, extent_count = 1, extent_offset = X, extent_length = Y;

[0682] itemID = 2, extent_count = 1, extent_offset = P, extent_length = Q;

[0683] itemID = 3, extent_count = 1, extent_offset = R, extent_length = S;

[0684] itemID = 4, extent_count = 1, extent_offset = X, extent_length = ET1;

[0685] itemID = 5, extent_count = 1, extent_offset = X+ET1, extent_length = ET2;

[0686] itemID = 6, extent_count = 1, extent_offset = X+ET2, extent_length = ET3;

[0687] itemID = 7, extent_count = 1, extent_offset = X+ET3, extent_length = ET4;

[0688] 아이템 참조:

[0689] type='cdsc', fromID=2, toID=1;

[0690] type='init', fromID=1, toID=3;

[0691] type='tbas', fromID=4, toID=1;

[0692] type='tbas', fromID=5, toID=1;

[0693] type='tbas', fromID=6, toID=1;

[0694] type='tbas', fromID=7, toID=1;

[0695] 미디어 데이터 박스:

[0696] HEVC Image (at file offset X, with length Y)

[0697] Exif data block (at file offset P, with length Q)

[0698] HEVC Config Record (at file offset R, with length S)

[0699] // No Tile description

[0700] 실시예들에 따르면, ItemInfoEntry 박스(601)의 버전 4(도 6, 602, 603를 참조)를 갖는 확장을 사용하여: 타일 픽처 정보는 정보 아이템(ID = 8)으로서 역시 기술되는 타일링 구성의 파트들에 대한 연관된 참조들을 사용해 열거된다.

[0701] ftyp box: major-brand = 'hevc', compatible-brands = 'hevc'

[0702] meta box: (container)

[0703] handler box: hdlr = 'hvc1' primary item: itemID = 1;

[0704] 아이템 정보:

[0705] item_type = 'hvc1', itemID=1, item_protection_index = 0 (unused)

[0706] item_type = 'Exif', itemID=2, item_protection_index = 0 (unused)

[0707] item_type = 'hvcC', itemID=3, item_protection_index = 0 (unused)

[0708] item_type = 'hvct', itemID=4, parameter for **ireftype==tile: tile_index=0**

[0709] item_type = 'hvct', itemID=5, parameter for **ireftype==tile: tile_index=1**

[0710] item_type = 'hvct', itemID=6, parameter for **ireftype==tile: tile_index=2**

[0711] item_type = 'hvct', itemID=7, parameter for **ireftype==tile: tile_index=3** item_type = 'tile',

itemID=8, (tiling configuration)

[0712] 아이템 위치:

[0713] itemID = 1, extent_count = 1, extent_offset = X, extent_length = Y;

[0714] itemID = 2, extent_count = 1, extent_offset = P, extent_length = Q;

[0715] itemID = 3, extent_count = 1, extent_offset = R, extent_length = S;

[0716] itemID = 4, extent_count = 1, extent_offset = X, extent_length = ET1;

[0717] itemID = 5, extent_count = 1, extent_offset = X+ET1, extent_length = ET2;

[0718] itemID = 6, extent_count = 1, extent_offset = X+ET2, extent_length = ET3;

[0719] itemID = 7, extent_count = 1, extent_offset = X+ET3, extent_length = ET4;

[0720] itemID = 8, extent_count = 1, extent_offset = i, extent_length = I;

[0721] 아이템 참조:

[0722] type='cdsc', fromID=2, toID=1;

[0723] type='init', fromID=1, toID=3;

[0724] type='tbas', fromID=4, toID=1;

[0725] type='tbas', fromID=5, toID=1;

[0726] type='tbas', fromID=6, toID=1;

[0727] type='tbas', fromID=7, toID=1;

[0728] type='tile', fromID=4, toID=8; //

[0729] type='tile', fromID=5, toID=8; // link each tile pict.

[0730] type='tile', fromID=6, toID=8; // to the tiling config item

[0731] type='tile', fromID=7, toID=8; //

[0732] 미디어 데이터 박스:

[0733] HEVC Image (at file offset X, with length Y)

[0734] Exif data block (at file offset P, with length Q)

[0735] HEVC Config Record (at file offset R, with length S)

[0736] Tile description data block (at file offset i, with length I)

[0737] 도 8은 본 발명의 실시예들의 구현의 상황을 예시하고 있다. 제1 상이한 미디어들이 레코딩된다: 예를 들어, 오디오는 단계(800a) 동안, 비디오는 단계(800b) 동안 그리고 하나 이상의 픽처들은 단계(800c) 동안 레코딩된다. 각각의 미디어가 각자의 단계들(801a, 801b, 801c) 동안 압축된다. 이 압축 단계들 동안, 기본 스트림들(802a, 802b 및 802c)이 생성된다. 다음에, 어플리케이션 레벨(그래픽 사용자 인터페이스로부터의 사용자 선택; 멀티미디어 생성 시스템의 구성 등)에서, 이 기본 스트림들 전부가 병합(merge)되어야 하는지 여부를 결정하기 위해 캡슐화 모드가 선택된다. "병합" 모드가 활성화될 때(테스트(803), "예"), 오디오, 비디오 및 스틸 이미지들에 대한 데이터가 앞서 설명된 바와 같이 단계(806c) 동안 동일한 파일에 캡슐화된다. "병합" 모드가 활성화되지 않으면(테스트(803), "아니오"), 2개의 캡슐화된 파일들이 단계들(806a 및 806b) 동안 연속적으로 또는 병렬로 생성되고, 그로써, 각각, 단계(807a) 동안 동기화된 시간 미디어 데이터에 대한 하나의 파일이 생성되고 스틸 이미지들만을 갖는 부가의 파일이 생성된다(907b). 타일 디스크립션 및 관심 영역 특징들을 제공하기 위해 본원에서 앞서 설명된 바와 같이, 단계(806a) 동안, 오디오 및 비디오 기본 스트림들이 ISOBMFF 표준에 따라 캡슐화되고, 단계(806b) 동안 스틸 픽처들이 캡슐화된다. 마지막으로, 미디어 프레젠테이션(media presentation)(807)이 획득되고, 전체적으로 또는 기술적 메타데이터를 파싱하는 것에 의해 (타일들과 같은) 일부 파트들이 추출된 이후에, 미디어 프레젠테이션을 스트리밍할 준비를 하기 위해 DASH 생성기에 제공될 수 있

거나(단계(820a)), 메모리에 저장될 수 있거나(단계(820b)), 디스플레이 유닛 상에 렌더링될 수 있거나(단계(820c)), 원격 엔티티에게 전송될 수 있다(단계(820d)).

[0738] 실시예들의 이전의 설명들에 따르면, 예를 들어, SimpleImageMetadata('simd') 박스(스틸 이미지 파일 포맷 규격의 마지막 버전에서 ISOBMFFMetaData라고도 불리움)와 같은 기술적 메타데이터가 본격적인 아이템(full-blown item)들로서 기술된다는 점에 유의해야 한다. 부가의 기술적 메타데이터(descriptive meta-data) 또는 규범적 메타데이터(prescriptive meta-data)가 또한 문서 w14878, committee draft study of ISO/IEC 23008-12:2013 1st edition, "Information technology — MPEG systems technologies — Part 12: Image File Format", MPEG 110 Strasbourg October 2014에 설명된 바와 같은 스틸 이미지 파일 포맷 규격에 의해 정의된다. 기술적 또는 규범적 메타데이터의 예들은 CleanApertureBox('clap'), ImageRotation('irot'), ExifDataBlock('exif'), 또는 ImageOverlay('iovl')이다. 보다 일반적으로, 기술적 메타데이터는 이미지 또는 서브 이미지와 같은 아이템에 대한 부가의 정보 또는 디스크립션을 제공하는 메타데이터(예컨대, Exif 메타데이터)이고, 규범적 메타데이터는 아이템에 적용될 연산들 또는 변환(예컨대, 회전, 크로핑 또는 변환 연산자들을 형성하는 몇 개의 아이템들의 조합)이다.

[0739] 그렇지만, 규격에서의 이러한 기술적 또는 규범적 메타데이터를 본격적인 아이템들로서 저장해야만 하는 것은 상당히 귀찮은 것처럼 보일 수 있고; 이들은, 기술적 또는 규범적 메타데이터가 인코딩된 데이터와 함께 mdat 박스(110)에 저장될 것을 요구하고 itemLocationBox(iloc)(109), itemInfoBox(iinf) 및 itemProtectionBox(ipro)에 엔트리들을 정의할 것을 요구하는, 의사 아이템(pseudo-item)들에 불과하다. 이것을 위해 iloc, iinf 및 ipro 내의 그 엔트리들을 요구하는 것은 상당한 오버헤드이다. 예를 들어, itemInfoBox 내의 엔트리는 적어도 12-바이트 헤더를 갖는 풀 박스(full box)의 사용을 요구하고, 그에 부가하여, itemInfoBox(iinf) 내의 엔트리마다 총 15 바이트의 추가 비용에 대해 item_protection_index(16 비트) 플러스 비어 있는 item_name(8 비트)이 정의되어야만 한다. itemLocationBox(iloc) 내의 엔트리는 또한 보다 나은 경우들에서 적어도 9 바이트를 요구한다(base_offset_size = offset_size = length_size = 1, 1 extent). 실제로, itemLocationBox 엔트리가 base_offset_size = offset_size = length_size = 2 또는 4와 함께 사용되며, 이는 12 또는 18 바이트의 추가 비용을 의미한다. 게다가, 이 메타데이터는 보통 작으며 다른 아이템들의 효율적인 판독을 가능하게 한다. 그들을 전용 아이템들로서 저장하는 것은 파일 파싱(file parsing), 특히 파일의 부분 페칭(partial fetching)을 복잡하게 할 수 있다(예를 들어, HTTP 요청들의 증가).

[0740] 대안의 실시예에서, 모든 기술적 및 규범적 메타데이터가, mdat 박스(110)에보다는 다른 박스들의 일부로서 meta 박스(100)에 저장될 수 있고 따라서 itemInfoBox 및 itemLocationBox 엔트리들을 정의하는 추가 비용을 회피할 수 있는, 임베딩된 아이템들로서 정의될 수 있다.

[0741] 기술적 및 규범적 메타데이터를 meta 박스에 저장하기 위해, 'VirtualItemBox'라고 불리는 가상 아이템 박스가 정의된다. 이 실시예에 따르면, 모든 기술적 및 규범적 메타데이터 박스는 이 가상 아이템 클래스로부터 상속받는다.

[0742] 가상 아이템은, 박스들의 세트와 함께, 그에 할당된 item_ID 및 item_type을 갖는다. 가상 아이템들은, 전형적으로 다른 아이템들과 연관될 메타데이터를 기술하는 데 사용되는, 부가의 데이터이다. 예를 들어, 가상 아이템은 아이템(이미지 또는 서브 이미지)을 식별해주는 itemInfoBox의 엔트리와 이 아이템에 적용될 연산 또는 변환을 연관시키는 것을 가능하게 한다. 전형적으로, 이미지의 item_ID와 메타데이터 연산 또는 변환 디스크립션 박스의 item_ID 간의 이 연관은 itemReferenceBox에 'simr' 타입의 엔트리를 정의하는 것에 의해 기술될 수 있다. 가상 아이템들은 아이템 참조 박스(item reference box)들 및 기본 아이템 박스(primary item box)들에서만 참조될 수 있고, 임의의 다른 박스(예컨대, itemLocationBox(iloc), itemInfoBox(iinf), itemProtectionBox(ipro))에서 선언 또는 참조되어서는 안된다. 'VirtualItemBox'는 다음과 같이 정의되고:

[0743] aligned(8) class VirtualItemBox(unsigned int(32) item_type)

[0744] extends FullBox('vite', version, 0) {

[0745] if (version == 0) {

[0746] unsigned int(16) item_ID;

[0747] } else {

[0748] unsigned int(32) item_ID;

[0749] }

[0750] unsigned int(32) item_type;

[0751] }

[0752] 그의 파라미터들에 대한 시맨틱스는 다음과 같다:

[0753] item_ID: 이 아이템의 ID(또는 식별자). iinf, iloc 또는 ipro에 동일한 item_ID 값을 가진 엔트리들을 갖는 것은 불법이다.

[0754] item_type은, 'mime'과 같은, 정의된 유효한 아이템 타입 지시자인, 전형적으로 4개의 인쇄가능 문자들인, 32 비트 값이다.

[0755] 임의로, 일 변형에서, 'VirtualItemBox'는 또한 "descriptor_family"라고 불리는 부가의 파라미터를 포함할 수 있다. 디스크립터 패밀리(Descriptor family)는 메타데이터 박스가 기술적 메타데이터인지 규범적 메타데이터인지를 표시한다. 일 변형에서, 디스크립터 패밀리는 미리 정의된 값들의 리스트로부터 메타데이터 박스의 타입을 표시한다. 예를 들어: transfo_operator, composed_image, descriptive_metadata ...

[0756] 이 가상 아이템 박스로부터 상속받는 것에 의해, 모든 기술적 및 규범적 메타데이터들이, 연관된 아이템들을 itemInfoBox(iinf) 및 itemLocationBox(iloc)에 정의할 필요 없이, meta 박스에 저장될 수 있지만 아이템 참조 박스들에 의해 어드레싱가능하다는 장점을 여전히 유지한다.

[0757] 이 실시예에 따르면, ImageOverlay(iovl), SubSampleItemData(subs), AuxiliaryConfiguration(auxC), ExifDataBlock(exif), SimpleImageMetadata(simd) 및 파생된 이미지 아이템은 가상 아이템 클래스로부터 상속받고 있다.

[0758] 여전히 이 실시예에 따르면, 'simd' 타입의 아이템들에 대한 'simr' 타입의 아이템 참조들을 갖는, 'dimg'라고 불리는 단일 제네릭 아이템 타입(single, generic item type)이 도입된다. 이 접근법은, 적절한 경우, 속성들의 재사용을 가능하게 하고 아이템들 및 아이템 참조들의 수를 감소시킨다. ImageRotationBox가 SimpleImageMetadata(simd)에 추가된다. 'simr' 참조 타입은, 이미지 기술적 메타데이터에의 직접 액세스를 제공하기 위해, 이미지 아이템으로부터 'simd' 아이템 쪽으로의 링크를 정의한다.

[0759] 그에 부가하여, ImageOverlay(iovl) 메타데이터 박스는 참조 순서에 더 이상 의존하지 않도록 다음과 같이 재설계된다.

[0760] aligned(8) class ImageOverlay {

[0761] unsigned int(8) version = 0;

[0762] unsigned int(8) flags;

[0763] for (j=0; j<3; j++) {

[0764] unsigned int(16) canvas_fill_value;

[0765] }

[0766] FieldLength = ((flags & 1) + 1) * 16;

[0767] unsigned int(FieldLength) output_width;

[0768] unsigned int(FieldLength) output_height;

[0769] for (i=0; i<reference_count; i++) {

[0770] **unsigned int(16) item_id;**

[0771] signed int(FieldLength) horizontal_offset;

[0772] signed int(FieldLength) vertical_offset;

[0773] }

[0774] }

[0775] 합성되는 아이템을 명시적으로 식별해주기 위해 루프 내의 각각의 엔트리에 대한 명시적 item_id가 추가된다.

[0776] 대안의 실시예에서, SimpleImageMetadata(simd)에 포함된 모든 박스들은 가상 아이템 박스로부터 상속받은 독립 메타데이터 박스들로서 정의된다.

[0777] 대안의 실시예에서, 간단한 이미지 회전은 다음과 같이 회전 연산을 이미지 메타데이터 디스크립터 SimpleImageMetadata('simd') 박스(또는 스틸 이미지 파일 포맷 규격의 마지막 버전에서 ISOBMFFMetaData라고도 불리움)에 직접 통합시키는 것에 의해 선언될 수 있다.

[0778] aligned(8) class ISOBMFFMetaData {

[0779] CleanApertureBox clap; // optional

[0780] PixelAspectRatioBox pasp; // optional

[0781] ColourInformationBox colour; // optional

[0782] ImageSpatialRelationBox location; // optional

[0783] ImageRotationBox rotation; // optional

[0784] Box extra_boxes[]; // optional

[0785] }

[0786] aligned(8) class ImageRotationBox

[0787] extends FullBox('irot', version = 0, flags = 0) { // 12 extra-bytes

[0788] unsigned int (6) reserved = 0;

[0789] unsigned int (2) angle;

[0790] }

[0791] rotation 박스가 'irot' 아이템들(12 바이트)보다 약간 더 크지만, rotation 및 CleanApperture와 같은, 변환들을 조합할 때 이 접근법을 사용하는 것의 이점은 명확한데, 그 이유는, 파생된 아이템들의 캐스케이드(cascade)가 아니라, 하나의 'simd'만이 필요하기 때문이다.

[0792] 이러한 경우에, 이미지 아이템 및 메타데이터 디스크립션 둘 다를 참조하기 위해 제네릭 파생 아이템(generic derived item) 'dimg'(앞서 설명됨)이 사용될 수 있다. 이러한 아이템은 그러면 PrimaryItemBox('pitm')에 기본 아이템으로서 열거될 수 있다.

[0793] 이 접근법의 다른 이점은 저작자가 회전된 아이템만이 디스플레이되기를 원한다는 것을 명확하게 표시할 수 있다는 것이다.

[0794] 하기의 단락들은 앞서 설명된 실시예에 대한 대안을 제안한다. 이 대안은 유리하게도 변환들(또는 "효과들")이 ISO 스틸 이미지 파일 포맷의 이미지들에 어떻게 적용될 수 있는지에 관해 간단하다. 상세하게는, 하기의 문제들이 이 대안의 실시예로 해결된다:

[0795] - 아이템 참조들의 수가 많음;

[0796] - 효과들을 캐스케이딩할 때 아이템들의 수가 증가함; 및

[0797] - 이미지들의 세트 또는, 관심 영역과 같은, 이미지들의 부분들을 의미하는, 주어진 아이템들의 세트에 대한 효과들을 상호관련(mutualize)시킬 수 없음.

[0798] 기존의 해결책들은 효과들을 아이템의 상이한 익스텐트(extent)들(데이터 파트에서의 바이트 오프셋들을 의미함)로서 상호관련시키는 것을 제안하였다. 보다 상세하게는, 익스텐트는 파생 이미지가 itemLocationBox("iloc")에서 익스텐트들의 리스트로서 기술될 것임을 의미하고, 각각의 익스텐트는 데이터 파트('mdat')의 프래그먼트를 식별해주며, 각각의 프래그먼트는 하나 이상의 기술적 또는 규범적 또는 변환 메타데이터에 대응한다.

[0799] 그러나 이 해결책에는 몇 가지 단점들이 내재해 있다:

[0800] - 캡슐화된 이미지 파일의 저작(authoring)이 아주 복잡하게 된다: 하나의 파생 이미지 아이템에서 하나의 효과를 터치(touch)하는 것은, 모든 파생 이미지들이 동일한 익스텐트를 공유하고 그의 일부를 재기입(rewrite)할 가능성이 있는지를 체크하기 위해, 모든 파생 이미지들을 검사하는 것을 암시한다;

[0801] - 이미지 파일 판독기가 상기 파일 내의 다른 아이템들에 대한 변환들/효과들의 체인이 동일한지 여부를 알아낼 필요가 있기 때문에(직접 시그널링이 없음), 파싱이 그다지 간단하지 않다.

[0802] - 각각의 변환/효과에 대해, 새로운 변환/효과가 적용할 변환들/효과들의 체인에서의 변환/효과와 함께 연속적으로 저장되지 않을 때마다 itemLocationBox("iloc")에 새 익스텐트가 필요할 것이다. 더욱이, 효과들의 조합 또는 캐스케이딩은 데이터 파트에서의 인접한 익스텐트들에 저장되지 않을 때 비용이 많이 들 수 있다.

[0803] 더욱이, 이 해결책들은 구현 저장을 요구했고, 이는, 효과의 타입을 이해하기 위해(지금까지는, 효과의 타입이 item_type에 의해 제공되었음), 효과를 저장하기 위한 박스의 생성을 암시한다. 효과에 대한 새로운 박스 포맷을 정의하는 것에 의해, 보다 간단한 해결책은 효과들을 아이템들과 별도로 정의하고 어떤 부가의 비용도 없이 아이템들과 효과들 간의 직접 매핑(direct mapping)을 갖는 것이다.

[0804] 대안의 실시예는 파일 포맷들에서의 깔끔한 분리를 갖는 것에 의해 효과 핸들링의 단순화를 제안한다:

[0805] - (앞서 제안된 바와 같이: 'init' 또는 'simr' 참조 타입 또는 기술적 메타데이터를 기술하는 임의의 참조 타입을 통해) 정규 아이템(regular item)들(이미지들 또는 이미지들의 부분들)(예컨대, hvc1, ...)이 그들의 기술적 메타데이터와 링크됨;

[0806] - "파생 이미지" 아이템으로부터의 소스 아이템에 대한 'dimg' 아이템 참조를 통해 식별된 하나 이상의 소스 아이템들(이미지 또는 이미지의 부분)에 적용된 효과들(또는 변환들)의 컬렉션인 "파생 이미지들"; 및

[0807] - 몇 개의 상이한 효과들의 컬렉션을 비롯한, 변환들/효과들을 표현하는 구조체.

[0808] 이 대안의 실시예의 장점들은 다음과 같다:

[0809] - 효과들의 재사용성(reusability): 한 번 선언되고 여러 번 참조될 가능성이 있음;

[0810] - 효과들의 컬렉션을 정의하는 것에 의한 보다 컴팩트한 디스크립션들(그에 관한 추가 내용은 이하를 참조);

[0811] - itemLocationBox의 새로운 익스텐트들이 필요하지 않다는 것을 비롯한 전반적 가독성(overall readability); 및

[0812] - 아이템 참조들의 수를 적게 유지하는 것.

[0813] 이 대안의 실시예에 따르면, 새로운 단일 파생 아이템이 'dimg' 아이템 유형으로 정의된다. 이 단일 파생 아이템은 구체적으로 다음과 같이 표현된다:

[0814]
```
aligned(8) class DerivedImage  {
```

[0815]
```
        bit(2) index_mode;
```

[0816]
```
    bit (6) reserved;
```

[0817]
```
        if (index_mode==0) nb_bits_effect = 8;
```

[0818]
```
        else if (index_mode==1) nb_bits_effect = 16;
```

[0819]
```
        else if (index_mode==2) nb_bits_effect = 32;
```

[0820]
```
        unsigned int(nb_bits_effect) nb_effects;
```

[0821]
```
        for (i=0; i<nb_effects; i++) {
```

[0822]
```
        unsigned int(nb_bits_effect) effect_id;
```

[0823]
```
}
```

[0824]
```
}
```

[0825] 여기서 nb_effects는 파생 이미지를 합성하기 위해 소스 이미지에 적용될 효과들의 수를 나타내고, effect_id는 적용할 효과의 캡슐화된 파일에서의 고유 식별자이다. 효과들은 효과들의 리스트에서의 효과들의 출현의 역순

으로 적용된다.

[0826] "DerivedImage"라고 명명된 파생 이미지 또는 변환된 아이템은 이미지를, 예를 들어, 이미지를 사용자 또는 디스플레이 화면에 제시하기 전에 소스 이미지에 적용될 효과들의 세트로서, 정의한다. 소스 이미지는 파생 아이템으로부터 소스 이미지로의 'dimg' 타입(또는 임의의 예비된 참조 타입)의 아이템 참조에 의해 식별된다. 소스 이미지 자체는 ISO 스틸 이미지 파일 포맷 규격에 정의된 임의의 이미지 아이템(이미지들 또는 이미지들의 부분들, 이미지 오버레이, 파생 이미지)일 수 있다. 동일한 아이템으로부터의 'dimg' 아이템 참조가 하나 초과 있어서는 안된다(그러나 이 아이템이 다양한 합성들을 위해 여러 번 재사용되는 경우, 동일한 아이템에 대해 다수 있을 수 있다).

[0827] 파생 이미지는 파일의 데이터 파트에 저장된다.

[0828] 캡슐화된 파일을 편집할 때, 예를 들어, 이미지 파일로부터 효과를 제거할 때, 이 효과에 대한 모든 참조들이 파생 이미지들로부터 제거되어야만 한다.

[0829] 효과들은 DerivedImage 아이템들을 통해 이미지들, 이미지들의 부분, 합성 이미지 또는 파생 이미지에 적용될 수 있다. 각각의 효과는 이하에서 예시되는 BaseEffectBox 구조체로부터 파생되는 박스에 의해 기술된다.

[0830] class BaseEffetBox(effect_type) extends FullBox(effect_type, version, flags){

[0831] if (version==0) nb_bits_effect = 8;

[0832] else if (version ==1) nb_bits_effect = 16;

[0833] else if (version ==2) nb_bits_effect = 32;

[0834] unsigned int(nb_bits_effect) effect_id;

[0835] }

[0836] 시맨틱스는 다음과 같다:

[0837] effect_type은 이 클래스로부터 파생되는 효과들의 박스 타입이고, 고유의 4 문자 코드는 박스의 종류를 식별해 준다;

[0838] effect_id는 주어진 효과 또는 변환에 대한 고유 식별자이다. 이 식별자는 'meta' 박스 내에서 고유해야 한다.

[0839] nb_bits_effect는 버전 값으로부터 파생되고, effect_id를 표현하는 데 사용되는 비트들의 수를 표시한다.

[0840] 효과들은, 'meta' 박스에 포함된, 임의적인 EffectDeclarationBox에서 선언될 수 있다.

[0841] Box Type: 'effd'

[0842] Container: meta

[0843] Mandatory: No

[0844] Quantity: Zero or One

[0845] class EffectDeclarationBox extends Box('effd'){

[0846] //one or more effect boxes

[0847] }

[0848] 예를 들어, 다음과 같은 효과들이 정의될 수 있다(제한적인 리스트가 아님):

[0849] - 회전 효과(Rotation Effect): 회전 효과는 소스 이미지를 반시계 방향으로 90도의 단위로 변환한다.

[0850] Box Type: 'erot'

[0851] Container: effd

[0852] Mandatory: No

[0853] Quantity: Zero or More

[0854] class RotationEffectBox extends BaseEffectBox('erot'){

[0855] unsigned int (6) reserved = 0;

[0856] unsigned int (2) angle;

[0857] }

[0858] 시맨틱스는 다음과 같다:

[0859] angle * 90: 이는 각도를 (반시계 방향으로) 도의 단위로 지정한다.

[0860] - 클린 애퍼처 효과 : 클린 애퍼처 효과는 소스 이미지의 가시 파트(visible part)를 수정한다.

[0861] Box Type: 'ecla'

[0862] Container: effd

[0863] Mandatory: No

[0864] Quantity: Zero or More

[0865] class CleanApertureEffectBox extends BaseEffectBox('ecla'){

[0866] unsigned int(nb_bits_effect) cleanApertureWidthN;

[0867] unsigned int(nb_bits_effect) cleanApertureWidthD;

[0868] unsigned int(nb_bits_effect) cleanApertureHeightN;

[0869] unsigned int(nb_bits_effect) cleanApertureHeightD;

[0870] unsigned int(nb_bits_effect) horizOffN;

[0871] unsigned int(nb_bits_effect) horizOffD;

[0872] unsigned int(nb_bits_effect) vertOffN;

[0873] unsigned int(nb_bits_effect) vertOffD;

[0874] }

[0875] 시맨틱스는 다음과 같다:

[0876] nb_bits_effect는 부모 클래스 BaseEffectBox로부터 파생되고, CleanApertureEffectBox의 상이한 필드들을 표현하는 데 사용되는 비트들의 수를 표시한다.

[0877] hSpacing, vSpacing: 픽셀의 상대 폭 및 높이를 정의한다.

[0878] cleanApertureWidthN, cleanApertureWidthD: 이미지의 정확한 클린 애퍼처 폭을, 카운트된 픽셀들로, 정의하는 분수(fractional number);

[0879] cleanApertureHeightN, cleanApertureHeightD: 이미지의 정확한 클린 애퍼처 높이를, 카운트된 픽셀들로, 정의하는 분수;

[0880] horizOffN, horizOffD: 클린 애퍼처 중심의 수평 오프셋에서 (폭-1)/2(전형적으로 0)을 뺀 것을 정의하는 분수;

[0881] vertOffN, vertOffD: 클린 애퍼처 중심의 수직 오프셋에서 (높이-1)/2(전형적으로 0)을 뺀 것을 정의하는 분수.

[0882] 효과 컬렉션(Effect Collection): 효과 컬렉션 박스는 몇 개의 효과들의 세트를 단일 효과로서 정의하고 이를 몇 개의 이미지들에 대해 재사용하는 것 및 따라서 바이트들의 면에서 디스크립션 비용(description cost)을 감소시키는 것을 가능하게 한다.

[0883] Box Type: 'ecol'

[0884] Container: effd

[0885] Mandatory: No

[0886] Quantity: Zero or More.

[0887] class EffectCollectionBox extends BaseEffectBox('ecol') {

[0888] unsigned int(nb_bits_effect) nb_effects;

[0889] for (i=0; i<nb_effects; i++) {

[0890] unsigned int(nb_bits_effect) apply_effect_id;

[0891] }

[0892] }

[0893] 시맨틱스는 다음과 같다:

[0894] nb_bits_effect는 부모 클래스 BaseEffectBox로부터 파생되고, EffectCollectionBox의 상이한 필드들을 표현하는 데 사용되는 비트들의 수를 표시한다.

[0895] apply_effect_id: 소스 이미지에 적용할 효과의 ID를 표시한다.

[0896] 효과 컬렉션 내의 효과들은 DerivedImaged 아이템 내의 효과들과 동일한 순서로 적용되고; 예컨대, 각각의 효과는 효과들의 리스트에서의 효과들의 출현의 역순으로 입력에 적용된다.

[0897] OverlayEffectBox는 이미지들의 합성을 오버레이로서 선언한다. 이 특수 효과의 경우, 결과 파생 이미지가 임의의 소스 이미지에 대한 어떤 참조도 갖지 않는데, 그 이유는 이 효과가 합성의 일부인 소스 이미지들의 리스트를 선언하기 때문이다.

[0898] class OverlayEffectBox extends BaseEffectBox('eovl'){

[0899] bit(1) fill_required;

[0900] bit(7) reserved;

[0901] if (fill_required) {

[0902] for (j=0; j<3; j++) {

[0903] unsigned int(nb_bits_effects) canvas_fill_value;

[0904] }

[0905] }

[0906] unsigned int(nb_bits_effects) output_width;

[0907] unsigned int(nb_bits_effects) output_height;

[0908] unsigned int(nb_bits_effects) nb_images;

[0909] for (i=0; i<nb_images; i++) {

[0910] unsigned int (nb_bits_effects) image_item_ID;

[0911] signed int(nb_bits_effects) horizontal_offset;

[0912] signed int(nb_bits_effects) vertical_offset;

[0913] }

[0914] }

[0915] 시맨틱스는 다음과 같다:

[0916] nb_bits_effects는 부모 클래스 BaseEffectBox로부터 파생되고, OverlayEffectBox의 상이한 필드들을 표현하는 데 사용되는 비트들의 수를 표시한다.

[0917] fill_required는 결과 합성 이미지에 배경 값으로 채울 구멍(hole)들이 있는지 여부를 표시한다.

[0918] canvas_fill_value: 임의의 입력 이미지의 어떤 픽셀도 특정의 픽셀 위치에 위치되지 않는 경우 사용되는 채널들마다의 픽셀 값을 표시한다. 입력 이미지들이 3개 미만의 채널들을 포함하면, 입력 이미지들에 존재하지 않는 채널들에 대응하는 canvas_fill_value의 시맨틱스는 지정되지 않는다(unspecified);

[0919] nb_images는 합성할 이미지들의 수를 표시하고, 각각은 image_item_ID 파라미터에 의해 표시된 바와 같은 그들의 item_ID에 의해 식별된다.

[0920] output_width, output_height: 입력 이미지들이 놓이게 되는 출력 이미지의 폭 및 높이를, 각각, 지정한다. 출력 이미지의 픽처 구역(picture area)은 캔버스(canvas)라고 지칭된다.

[0921] horizontal_offset, vertical_offset: 캔버스의 좌측 상단 코너로부터, 입력 이미지가 위치되는 곳까지의 오프셋을 지정한다. 네거티브 오프셋 값을 갖는 픽셀 위치들은 출력 이미지에 포함되지 않는다. output_width보다 크거나 같은 수평 픽셀 위치들은 출력 이미지에 포함되지 않는다. output_height보다 크거나 같은 수직 픽셀 위치들은 출력 이미지에 포함되지 않는다.

[0922] 본 발명의 다른 양태에 따르면, 모든 기술적 및 규범적 메타데이터의 저장이, 이상의 실시예들과 비교하여, 기술적 및/또는 규범적 메타데이터가 특정의 이미지 아이템에 특정적이거나 몇 개의 이미지 아이템들 간에 공유되는지에 따라 추가로 최적화될 수 있다. 바이트 범위들의 공유를 사용하지 않거나 이상의 실시예들에 의해 요구되는 바와 같은 광범위한 아이템 참조들의 리스트를 정의하지 않으면서, 이러한 공유가 가능하게 된다. 이 대안의 실시예에 따르면, 모든 기술적 및 규범적 메타데이터는 여전히 'meta' 박스(100)에서의 박스 계층구조 내에만 저장되어, ISOBMFF 판독기들이 'idat' 또는 'mdat' 박스를 페치할 필요 없이 모든 시스템 정보를 파싱할 수 있게 한다. 따라서, 미디어 데이터를 어드레싱하거나 몇 개의 이미지 아이템들 간의 관계를 표현하기 위해서만 ('iinf'박스에 있는) 이미지 아이템들 및 ('iref' 박스에 있는) 아이템 참조들의 수가 제한된다. 이러한 설계는 파일의 파싱을 보다 간단하게 만들고 파일 포맷에 대한 높은 수준의 이해를 용이하게 한다.

[0923] 이 실시예의 주된 양태는 모든 시스템-레벨 아이템 정보가 어떤 'mdat' 또는 'idat' 박스도 페치하는 일 없이 파서에 의해 액세스가능한 (ISOBMFF fullbox를 사용하는) 전용 박스들에 박싱(box)되고, 직접 아이템 정보 엔트리에 포함되거나 정보 아이템 엔트리에 의해 참조된다는 것이다.

[0924] 이 실시예는 다음과 같은 변화들을 도입한다:

[0925] - SharedItemPropertiesBox('sitp')라고 불리는 새로운 전용 박스가 아이템들 간에 공유되는 박스 구조의(box-structured) 기술적 및 규범적 메타데이터를 포함하도록 정의된다.

[0926] - 박스 구조의 기술적 및 규범적 메타데이터를 아이템과 연관시키도록 아이템 정보 엔트리('infe')를 수정. 그 메타데이터는, 메타데이터가 이 아이템에만 관련되거나 'sitp' 박스에 저장되는 경우, 'infe' 박스에 직접 저장될 수 있고, 메타데이터가 몇 개의 아이템들 간에 공유되는 경우, 'infe' 박스로부터 참조될 수 있다.

[0927] - 이미지 아이템과 트랙 내의 샘플 간의 동일한 초기화 데이터의 공유를 가능하게 하는 새로운 박스(초기화 파라미터를 표현하는 SampleDescriptionEntryReference 'sder').

[0928] SharedItemPropertiesBox('sitp')라고 불리는 새로운 박스는 다음과 같이 정의된다:

[0929] Box Type: 'sitp'

[0930] Container: MetaBox ('meta')

[0931] Mandatory: No

[0932] Quantity: Zero or One

[0933] SharedItemProperties 박스(전용 공유 박스)는 부모 'meta' 박스에 선언된 몇 개의 아이템들에 적용가능할 수 있는 기술적(디스플레이 파라미터들) 및 규범적(변환 연산자들) 메타데이터(속성들이라고도 지칭됨)를 정의하는 박스들의 리스트를 포함한다. 이 박스들은 ItemInfoEntry 박스로부터 0 기반 인덱스에 의해 참조된다. 이 박스는 다음과 같은 신택스를 갖는다:

[0934]
```
aligned(8) class SharedItemPropertiesBox extends Box('sitp') {
```

[0935]
```
    // one or more boxes
```

[0936]
```
}
```

[0937] 아이템 정보 엔트리의 수정에 관해서는, 새 버전(4)이 하기의 시맨틱스로 정의된다: ItemInfoEntry 박스는 이 아이템에 대한 속성들을 제공하는 부가의 박스들을 아이템 정보 엔트리에 포함시키거나 부가의 박스들을 참조할 가능성을 제공한다. 포함된 특성들과 참조된 특성들의 합집합(union)에 주어진 타입의 특성이 최대 하나 있어야 한다. 속성들은 순서 의존적(order-dependent)일 수 있고, 이 경우에 ItemInfoEntry 박스에서 주어진 순서가 사용되어야 하며, 즉 첫 번째의 포함된 속성이 먼저 적용되고, 순서상 그에 뒤따라서 다른 모든 포함된 속성들이 적용되며, 그에 뒤따라서 이어서 모든 참조된 속성들이 적용된다.

[0938] 부가의 신택스는 다음과 같이 지정된다:

[0939] if (version == 4) {

[0940] unsigned int(16) included_prop_count;

[0941] Box item_properties[included_prop_count];

[0942] unsigned int(16) indexed_prop_count;

[0943] unsigned int(16) box_prop_idx[indexed_prop_count];

[0944] }

[0945] 연관된 시맨틱스는 다음과 같다:

[0946] included_prop_count: item_properties 어레이에 포함된 속성들(기술적 또는 규범적 메타데이터)의 수.

[0947] item_properties: 이 아이템에 대한 부가의 정보를 제공하는 박스들의 어레이 또는 박스들의 테이블(아이템 정보의 속성들). 허용된 박스들은 SharedItemProperties 박스에서와 동일하다.

[0948] indexed_prop_count: SharedItemProperties 박스 내의 속성들에 대한 참조들의 수.

[0949] box_prop_idx : 'meta' 박스의 SharedItemProperties 박스에 저장된 박스들의 리스트에 대한 0 기반 인덱스들.

[0950] 이 실시예에 따르면, 모든 기술적 및 규범적 메타데이터는 SharedItemProperties 박스에 또는 ItemInfoEntry 박스 내의 item_properties 어레이에 저장될 ISOBMFF 풀 박스들이다.

[0951] 예를 들어, 이미지 회전에 대한 규범적 메타데이터는 다음과 같이 정의된다:

[0952] Box Type: 'irot'

[0953] Container: SharedItemProperties

[0954] Mandatory: No

[0955] Quantity: Zero or more.

[0956] 이미지 회전 박스(Image Rotation Box)는 반시계 방향으로 90도의 단위로 회전 각도를 제공한다. 하나의 이러한 박스만이 이미지 아이템의 속성으로서 할당되어야 한다. 이 박스의 신택스는 다음과 같이 정의된다:

[0957] aligned(8) class ImageRotationBox extends FullBox('irot', version, flags) {

[0958] unsigned int (6) reserved = 0;

[0959] unsigned int (2) angle;

[0960] }

[0961] 애트리뷰트 시맨틱스(attribute semantics)는 다음과 같다:

[0962] version은 0이어야 한다.

[0963] flags는 0이어야 한다.

[0964] angle * 90은 각도를 (반시계 방향으로) 도의 단위로 지정한다.

[0965] 이미지 오버레이에 대한 규범적 메타데이터는 다음과 같이 정의된다:

[0966] Box Type: 'iovl'

[0967] Container: SharedItemProperties

[0968] Mandatory: No

[0969] Quantity: Zero or more.

[0970] 이미지 오버레이 박스는 하나 이상의 입력 이미지들을 보다 큰 캔버스 내에 주어진 레이어링 순서(layering order)로 위치시킨다. 입력 이미지들은 이 박스를 속성으로서 포함하거나 참조하는 파생 이미지 아이템에 대한 'dimg' 타입의 SingleItemTypeReferenceBox에서 그들이 레이어링되는 순서로, 즉 최하단 입력 이미지가 첫 번째로 그리고 최상단 입력 이미지가 마지막으로 열거된다. 하나의 이러한 박스만이 이미지 아이템의 속성으로서 할당되어야 한다.

[0971] 이 박스의 신택스는 다음과 같이 정의된다:

[0972]
```
aligned(8) class ImageOverlay extends FullBox('iovl', version, flags){
```

[0973]
```
        for (j=0; j<4; j++) {
```

[0974]
```
                unsigned int(16) canvas_fill_value;
```

[0975]
```
        }
```

[0976]
```
        FieldLength = ((flags & 1) + 1) * 16;
```

[0977]
```
        unsigned int(FieldLength) output_width;
```

[0978]
```
        unsigned int(FieldLength) output_height;
```

[0979]
```
        for (i=0; i<reference_count; i++) {
```

[0980]
```
                signed int(FieldLength) horizontal_offset;
```

[0981]
```
                signed int(FieldLength) vertical_offset;
```

[0982]
```
        }
```

[0983]
```
}
```

[0984] 애트리뷰트 시맨틱스는 다음과 같다:

[0985] version은 0이어야 한다.

[0986] (flags & 1) = 0은 output_width, output_height, horizontal_offset, 및 vertical_offset 필드들의 길이가 16 비트라는 것을 지정한다. (flags & 1) = 1은 output_width, output_height, horizontal_offset, 및 vertical_offset 필드들의 길이가 32 비트라는 것을 지정한다. 1보다 큰 flags의 값들이 예비되어 있다.

[0987] canvas_fill_value: 임의의 입력 이미지의 어떤 픽셀도 특정의 픽셀 위치에 위치되지 않는 경우 사용되는 채널들마다의 픽셀 값을 표시한다. 채우기 값(fill value)들은 RGBA(R, G, B, 및 A는, 각각, 루프 카운터 j가 0, 1, 2 및 3인 것에 대응함)로서 지정된다. RGB 값들은 IEC 61966-2-1에 정의된 바와 같은 sRGB 색 공간에 있다. A 값은 0(완전 투명) 내지 65535(완전 불투명)의 범위에 있는 선형 불투명도 값(linear opacity value)이다.

[0988] output_width, output_height: 입력 이미지들이 놓이게 되는 출력 이미지의 폭 및 높이를, 각각, 지정한다. 출력 이미지의 이미지 구역은 캔버스라고 지칭된다.

[0989] reference_count는 이 박스를 사용하는 아이템이 from_item_ID 필드에 의해 식별되는 'dimg' 타입의 SingleItemTypeReferenceBox로부터 획득된다.

[0990] horizontal_offset, vertical_offset: 캔버스의 좌측 상단 코너로부터, 입력 이미지가 위치되는 곳까지의 오프셋을 지정한다. 네거티브 오프셋 값을 갖는 픽셀 위치들은 출력 이미지에 포함되지 않는다. output_width보다 크거나 같은 수평 픽셀 위치들은 출력 이미지에 포함되지 않는다. output_height보다 크거나 같은 수직 픽셀 위치들은 출력 이미지에 포함되지 않는다.

[0991] 이미지 격자에 대한 규범적 메타데이터는 다음과 같이 정의된다:

[0992] Box Type: 'grid'

[0993] Container: SharedItemProperties

[0994] Mandatory: No

[0995] Quantity: Zero or more.

[0996] 이미지 격자 박스는 하나 이상의 입력 이미지들로부터의 출력 이미지를 보다 큰 캔버스 내에 주어진 격자 순서로 형성한다. 하나의 이러한 박스만이 이미지 아이템의 속성으로서 할당되어야 한다. 입력 이미지들은 ItemReferenceBox 내의 이 박스를 사용하여 파생 이미지 아이템에 대해 'dimg' 타입의 SingleItemTypeReferenceBox의 순서에서 행 우선 순서(row-major order)로, 상단 행을 첫 번째로, 좌에서 우로, 삽입된다. 이 아이템으로부터의 입력 이미지들에 대한 rows*columns개의 아이템 참조들이 있어야 한다. 모든 입력 이미지들은 완전히 동일한 폭 및 높이를 가져야 한다; 그 tile_width 및 tile_height를 상기한다. 타일링된 입력 이미지들은, tile_width*columns가 output_width보다 크거나 같고 tile_height*rows가 output_height보다 크거나 같은, 출력 이미지 격자 캔버스를 완전히 "커버"해야 한다. 출력 이미지는 입력 이미지들을, 갭 또는 오버랩 없이, tile_width와 같은 열 폭(column width)(어쩌면 최우측 열을 제외함) 및 tile_height와 같은 행 높이(row height)(어쩌면 최하단 행을 제외함)를 갖는 격자 내로 타일링하고 이어서, 우측 및 하단에서, 표시된 output_width 및 output_height로 트리밍(trimming)하는 것에 의해 형성된다.

[0997] 이 박스의 신택스는 다음과 같이 정의된다:

[0998] aligned(8) class ImageGridBox extends FullBox('grid', version, flags) {

[0999] FieldLength = ((flags & 1) + 1) * 16;

[1000] unsigned int(8) rows;

[1001] unsigned int(8) columns;

[1002] unsigned int(FieldLength) output_width;

[1003] unsigned int(FieldLength) output_height;

[1004] }

[1005] 애트리뷰트 시맨틱스는 다음과 같다:

[1006] version은 0이어야 한다.

[1007] (flags & 1) = 0은 output_width, output_height 필드들의 길이가 16 비트라는 것을 지정한다. (flags & 1) = 1은 output_width, output_height 필드들의 길이가 32 비트라는 것을 지정한다. 1보다 큰 flags의 값들이 예비되어 있다.

[1008] output_width, output_height: 입력 이미지들이 놓이게 되는 출력 이미지의 폭 및 높이를, 각각, 지정한다. 출력 이미지의 이미지 구역은 캔버스라고 지칭된다.

[1009] rows, columns: 입력 이미지들의 행들의 수 및 행마다의 입력 이미지의 수를 지정한다. 입력 이미지들은 상단 행을 첫 번째로 채우고 뒤이어서 두 번째 행 및 후속 행들을 아이템 참조들의 순서로 채운다.

[1010] 이와 유사하게, 보조 구성 박스(auxiliary configuration box)('auxC'), 이미지 공간 익스텐트 박스(image spatial extents box)('ispe'), 픽셀 정보 박스('pixi'), 상대 위치 박스('rloc'), 클린 애퍼처 박스('clap')(제한적인 리스트가 아님)와 같은, 모든 다른 규범적 및 기술적 메타데이터 모두는 ISOBMFF fullbox로부터 상속된다.

[1011] 이 실시예에 따르면, 아이템은, 파생에의 입력들인, 하나 이상의 다른 이미지 아이템들에 대한 'dimg' 아이템 참조를 포함할 때, 파생 이미지이다. 파생 이미지는 지정된 입력 이미지들에 대해, 회전과 같은, 지정된 연산을 수행하는 것에 의해 획득된다. 파생 이미지를 획득하기 위해 수행되는 연산은 아이템의 item_type에 의해 식별된다. 파생 이미지에의 입력으로서 사용되는 이미지 아이템들은 코딩된 이미지들일 수 있거나 다른 파생 이미지 아이템들일 수 있다.

[1012] 예를 들어, 클린 애퍼처 파생 이미지 아이템은 item_type 값 'clap'에 의해 식별된다. 이는 어떤 데이터도 저장하지 않으며, 'iloc' 테이블에 연관된 엔트리를 갖지 않아야 한다. 이는 ISO/IEC 14496-12에 정의된 바와 같은 CleanApertureBox 타입의 아이템 속성을 포함하거나 참조해야 한다. 이는 이미지 아이템에 대한 'dimg' 타

입의 아이템 참조를 가져야 한다. 다른 예로서, 이미지 회전 파생 이미지 아이템은 item_type 값 'irot'에 의해 식별된다. 이는 어떤 데이터도 갖지 않으며, 'iloc' 테이블에 연관된 엔트리를 갖지 않아야 한다. 이는 앞서 정의된 바와 같은 ImageRotationBox 타입의 아이템 속성을 포함하거나 참조해야 한다. 이는 이미지 아이템에 대한 'dimg' 타입의 아이템 참조를 가져야 한다.

[1013] 이와 유사하게, 이미지 오버레이 파생 이미지 아이템은 item_type 'iovl'에 의해 식별된다. 이는 어떤 데이터도 갖지 않으며, 'iloc' 테이블에 연관된 엔트리를 갖지 않아야 한다. 이는 앞서 정의된 바와 같은 ImageOverlayBox 타입의 아이템 속성을 포함하거나 참조해야 한다. 이는 이미지 아이템들의 세트에 대한 'dimg' 타입의 아이템 참조를 가져야 한다. 이미지 격자 파생 이미지 아이템은 item_type 값 'grid'에 의해 식별된다. 이는 어떤 데이터도 갖지 않으며, 'iloc' 테이블에 연관된 엔트리를 갖지 않아야 한다. 이는 앞서 정의된 바와 같은 ImageGridBox 타입의 아이템 속성을 포함하거나 참조해야 한다. 이는 이미지 아이템들의 세트에 대한 'dimg' 타입의 아이템 참조를 가져야 한다.

[1014] 기술적 및 규범적 메타데이터(또는 속성들)를 이미지들에 할당하기 위해 SharedItemProperties 박스 및 확장된 ItemInfoEntry 박스를 사용하는 것을 보여주는 일부 예들이 이하에 있다.

[1015] 이하의 예에서, 2개의 속성 박스들('hvcC' 및 'ispe')이 연관된 itemInfoEntry 내에서 직접 item_properties 어레이에서 이미지 아이템에 할당된다.

[1016] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1017] MetaBox: (container)

[1018] HandlerBox: hdlr = 'pict'

[1019] PrimaryItemBox: itemID = 1;

[1020] ItemInfoBox:

[1021] 1) item_type = 'hvc1', itemID=1, item_protection_index = 0 (unused), item properties: 'hvcC', 'ispe'

[1022] ItemLocationBox:

[1023] itemID = 1, extent_count = 1, extent_offset = X, extent_length = Y; MediaDataBox:

[1024] HEVC Image (at file offset X, with length Y)

[1025] 이하의 예에서, 이전의 예에 부가하여, 이미지 회전 연산자(image rotation operator)('irot')가 유사한 방식으로 이미지 아이템에 할당된다.

[1026] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1027] MetaBox: (container)

[1028] HandlerBox: hdlr = 'pict'

[1029] PrimaryItemBox: itemID = 1;

[1030] ItemInfoBox:

[1031] 1) item_type = 'hvc1', itemID=1, item_protection_index = 0 (unused), item properties: 'hvcC', 'ispe', 'irot'

[1032] ItemLocationBox:

[1033] itemID = 1, extent_count = 1, extent_offset = X, extent_length = Y; MediaDataBox:

[1034] HEVC Image (at file offset X, with length Y)

[1035] 이하의 예에서, 상이한 HEVC 구성들을 갖는 다수의 이미지들은 SharedItemProperty 박스('sitp')에 저장된 공통의 이미지 공간 익스텐트 박스('ispe')에 기술된 것과 동일한 차원들을 공유한다. 각각의 이미지 itemInfoEntry 박스는 그 자신의 HEVC 구성 박스('hvcC')를 포함하고, 공통의 이미지 공간 익스텐트 박스('ispe')를 참조하기 위해, SharedItemProperty 박스에 대한 인덱스(아이템 속성 인덱스들)을 사용한다.

[1036] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1037] MetaBox: (container)

[1038] HandlerBox: hdlr = 'pict'

[1039] PrimaryItemBox: itemID = 1;

[1040] ItemInfoBox:

[1041] 1) item_type = 'hvc1', itemID=1,

[1042] item_protection_index = 0 (unused),

[1043] item properties: 'hvcC', item properties indices: 0

[1044] 2) item_type = 'hvc1', itemID=2,

[1045] item_protection_index = 0 (unused)

[1046] item properties: 'hvcC', item properties indices: 0

[1047] 3) item_type = 'hvc1', itemID=3,

[1048] item_protection_index = 0 (unused)

[1049] item properties: 'hvcC', item properties indices: 0

[1050] 4) item_type = 'hvc1', itemID=4,

[1051] item_protection_index = 0 (unused)

[1052] item properties: 'hvcC', item properties indices: 0

[1053] SharedItemPropertiesBox:

[1054] 0) 'ispe'

[1055] ItemLocationBox:

[1056] itemID = 1, extent_count = 1, extent_offset = X,

[1057] extent_length = Y;

[1058] itemID = 2, extent_count = 1, extent_offset = P0,

[1059] extent_length = Q0;

[1060] itemID = 3, extent_count = 1, extent_offset = P1,

[1061] extent_length = Q1;

[1062] itemID = 4, extent_count = 1, extent_offset = P2,

[1063] extent_length = Q2;

[1064] MediaDataBox:

[1065] HEVC Image (at file offset X, with length Y)

[1066] HEVC Image (at file offset P1, with length Q1)

[1067] HEVC Image (at file offset P2, with length Q2)

[1068] HEVC Image (at file offset P3, with length Q3)

[1069] item properties indices 테이블의 엔트리들은 식별자들의 세트를 형성한다. 다른 식별자들의 세트는 전용 공유 박스 [SharedItemPropertiesBox] 내의 이미지 디스크립션 정보(여기서 'ispe')의 랭크(rank)(여기서 "0")에 의해 형성된다.

[1070] 다른 실시예에서, 다른 식별자는 전용 공유 박스 내의 이미지 디스크립션 정보에 할당된 다른 ID에 의해 형성될 수 있다. 예를 들어, 이미지 디스크립션 정보에 할당된 이 다른 ID는 ISOBMFF "fullbox" 대신에 (앞서 설명된) "VirtualItemBox"로부터 상속하는 것에 의해 정의될 수 있다. 이 실시예는 유리하게도 식별자들의 세트에 영향을 주지 않으면서 전용 공유 박스 내의 이미지 디스크립션 정보를 재순서화(re-order)할 수 있게 한다.

[1071] 양쪽 식별자들의 세트는 이미지 아이템 정보[ItemInfoBox의 아이템으로 표현됨]를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 형성한다.

[1072] 하기의 예는 회전된 격자에 있는 다수의 이미지들로 구성된 파생 이미지를 설명한다. 격자를 구성하는 모든 이미지들은 SharedItemProperty 박스 내에 위치되고 박스 속성 인덱스를 통해 참조되는 'hvcC' 및 'ispe' 박스들을 통해 동일한 HEVC 구성 및 동일한 이미지 차원들을 공유한다. 격자를 표현하는 파생 이미지는 이미지 격자 박스를 포함하는 itemInfoEntry를 통해 기술된다. 적용할 회전은 파생 이미지에 연관된 이미지 회전 박스로 기술된다. 파생 이미지를 구성할 입력 이미지들은 아이템 참조 박스('iref') 박스 내의 아이템 참조 엔트리를 통해 참조된다.

[1073] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1074] MetaBox: (container)

[1075] HandlerBox: hdlr = 'pict'

[1076] PrimaryItemBox: itemID = 5;

[1077] ItemInfoBox:

[1078] 1) item_type = 'hvc1', itemID=1,

[1079] item_protection_index = 0 (unused),

[1080] item properties indices: 0, 1

[1081] 2) item_type = 'hvc1', itemID=2,

[1082] item_protection_index = 0 (unused),

[1083] item properties indices: 0, 1

[1084] 3) item_type = 'hvc1', itemID=3,

[1085] item_protection_index = 0 (unused),

[1086] item properties indices: 0, 1

[1087] 4) item_type = 'hvc1', itemID=4,

[1088] item_protection_index = 0 (unused),

[1089] item properties indices: 0, 1

[1090] 5) item_type = 'grid', itemID=5,

[1091] item_protection_index = 0 (unused),

[1092] item properties: 'grid', 'irot'

[1093] SharedItemPropertiesBox:

[1094] 0) 'hvcC'

[1095] 1) 'ispe'

[1096] ItemLocationBox:

[1097] itemID = 1, extent_count = 1, extent_offset = X,

[1098] extent_length = Y;

[1099] itemID = 2, extent_count = 1, extent_offset = P0,

[1100] extent_length = Q0;

[1101] itemID = 3, extent_count = 1, extent_offset = P1,

[1102] extent_length = Q1;

[1103] itemID = 4, extent_count = 1, extent_offset = P2,

[1104] extent_length = Q2;

[1105] ItemReferenceBox:

[1106] type='dimg', fromID=5, toID=1,2,3,4;

[1107] MediaDataBox:

[1108] HEVC Image (at file offset X, with length Y)

[1109] HEVC Image (at file offset P1, with length Q1)

[1110] HEVC Image (at file offset P2, with length Q2)

[1111] HEVC Image (at file offset P3, with length Q3)

[1112] 하기의 예는 HEVC 타일링된 이미지를 설명한다. 이 예에서, SharedItemPropertiesBox를 통해, 모든 아이템들(풀 이미지(itemID = 1) 및 타일들(itemID = 2,3,4,5))은 동일한 HEVC 구성 박스를 공유하고 모든 타일들은 타일 크기(Wt, Ht)를 정의하는 동일한 이미지 공간 익스텐트 박스를 공유한다. 그에 부가하여, 모든 타일 아이템들은 각각의 타일의 x, y 좌표들을 제공하는 그 자신의 상대 위치 박스('rloc')를 포함한다:

[1113] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1114] MetaBox: (container)

[1115] HandlerBox: hdlr = 'pict'

[1116] PrimaryItemBox: itemID = 1;

[1117] ItemInfoBox:

[1118] 1) item_type = 'hvc1', itemID=1,

[1119] item_protection_index = 0 (unused),

[1120] item properties indices: 0

[1121] item properties: 'ispe' (W,H)

[1122] 2) item_type = 'hvt1', itemID=2,

[1123] item_protection_index = 0 (unused)

[1124] item properties indices: 0, 1

[1125] item properties: 'rloc'

[1126] 3) item_type = 'hvt1', itemID=3,

[1127] item_protection_index = 0 (unused)

[1128] item properties indices: 0, 1

[1129] item properties: 'rloc'

[1130] 4) item_type = 'hvt1', itemID=4,

[1131] item_protection_index = 0 (unused)

[1132] item properties indices: 0, 1

[1133] item properties: 'rloc'

[1134] 5) item_type = 'hvt1', itemID=5,

[1135] item_protection_index = 0 (unused)

[1136] item properties indices: 0, 1

[1137] item properties: 'rloc'

[1138] SharedItemPropertiesBox:

[1139] 0) 'hvcC'

[1140] 1) 'ispe' (Wt, Ht)

[1141] ItemLocationBox:

[1142] itemID = 1, extent_count=1, extent_offset=X,

[1143] extent_length=Q0+Q1+Q2+Q3;

[1144] itemID = 2, extent_count=1, extent_offset=X,

[1145] extent_length=Q0;

[1146] itemID = 3, extent_count=1, extent_offset=X+Q0,

[1147] extent_length=Q1;

[1148] itemID = 4, extent_count=1, extent_offset=X+Q0+Q1,

[1149] extent_length=Q2;

[1150] itemID = 5, extent_count=1, extent_offset=X+Q0+Q1+Q2,

[1151] extent_length=Q3;

[1152] ItemReferenceBox:

[1153] type='tbas', fromID=2, toID=1;

[1154] type='tbas', fromID=3, toID=1;

[1155] type='tbas', fromID=4, toID=1;

[1156] type='tbas', fromID=5, toID=1;

[1157] MediaDataBox:

[1158] HEVC Image (at file offset X, with length Q0+Q1+Q2+Q3)

[1159] 그에 부가하여, 일부 이미지 포맷들은 이미지 아이템 데이터를 디코딩하기 위한 초기화 데이터를 요구한다. 초기화 데이터는 코덱 특정적(codec-specific)이며, 비디오 트랙들에 대해 지정된 디코더 구성 레코드와 같거나 그와 유사할 수 있다. 이러한 경우에, 파일 포맷에서 초기화 데이터를 반복하지 않고 초기화 데이터를 공유하는 것이 유용하다. 이러한 초기화 데이터가 필요한 경우, 그 초기화 데이터는 아이템 정보에서 특정 타입의 기술적 메타데이터(속성)에 의해 제공된다. 몇 개의 이미지 아이템들이 동일한 이러한 속성을 공유할 수 있다. 이미지 아이템과 트랙의 일부 샘플들 간에 동일한 초기화 데이터를 공유하는 것을 가능하게 하기 위해, SampleDescriptionEntryReference('sder')라고 불리는 새로운 기술적 메타데이터 박스가 다음과 같이 정의된다:

[1160] Box Type: 'sder'

[1161] Container: SharedItemProperties

[1162] Mandatory: No

[1163] Quantity: Zero or more.

[1164] SampleDescriptionEntryReferenceBox는 이미지 아이템이 트랙의 일부 샘플들과 동일한 초기화 데이터를 재사용한다는 것을 표시하는 것을 가능하게 한다. 이는 트랙 및 그 트랙의 그 샘플들의 샘플 디스크립션 엔트리(sample description entry)를 식별해준다. 이 박스는 다음과 같은 신택스를 갖는다:

[1165] aligned(8) class SampleDescriptionEntryReferenceBox

[1166] extends FullBox('sder', 0, flags) {

[1167] unsigned int(32) track_ID;

[1168] unsigned int(32) sample_description_index;

[1169] }

[1170] 그의 파라미터들에 대한 시맨틱스는 다음과 같다:

[1171] track_ID: 초기화가 재사용되는 트랙의 식별자.

[1172] sample_description_index: 이 아이템에서의 데이터를 기술하는 연관된 트랙에서의 샘플 아이템의 1 기반 인덱스(1-based index)이다.

[1173] 하기의 예는 이미지 itemInfoEntry에 연관된 SampleDescriptionEntryReference 박스('sder')를 통해 트랙과 이미지 아이템 간에 HEVC 구성을 공유하는 것을 보여준다.

[1174] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic, mp41'

[1175] MetaBox: (container)

[1176] HandlerBox: hdlr = 'pict'

[1177] PrimaryItemBox: itemID = 1;

[1178] ItemInfoBox:

[1179] 1) item_type = 'hvc1', itemID=1, item_protection_index = 0 (unused), item properties: 'sder' (track: 1, sample_desc_index: 1), 'ispe'

[1180] ItemLocationBox:

[1181] itemID = 1, extent_count = 1, extent_offset = X,

[1182] extent_length = Y;

[1183] Movie Box: (container)

[1184] MP4에 의해 요구되는 바와 같은 무비 헤더, 트랙들 (적어도 하나의 샘플 디스크립션을 갖는 트랙 1을 포함함) 등

[1185] MediaDataBox:

[1186] HEVC Image (at file offset X, with length Y)

[1187] 무비에 의해 필요로 하게 되는 바와 같은 미디어 데이터

[1188] (일부는 이미지 데이터와 공유될 수 있음)

[1189] 이미지 아이템 데이터가 HEVC 타일들을 표현할 때, 각각의 HEVC 타일 아이템은 HEVC 타일 아이템에 존재하는 타일들을 디코딩하는 데 요구된 모든 파라미터 세트들을 갖는 HEVCConfigurationBox 타입의 속성을 포함하거나 참조해야 한다. 몇 개의 HEVC 타일 아이템들이 동일한 HEVCConfigurationBox 속성을 공유할 수 있다. HEVC 타일 아이템은 또한 각자의 HEVC 이미지 아이템 내의 HEVC 타일 아이템의 위치를 표시하는 RelativeLocationBox 속성('rloc')을 포함하거나 참조해야 한다. 상이한 HEVC 이미지들에 속하는 타일들에 대응하는 몇 개의 HEVC 타일 아이템들은 동일한 RelativeLocationBox를 공유할 수 있다. 각각의 HEVC 타일 아이템에 대해 ImageSpatialExtentsBox 속성('ispe')이 사용되어야 한다. ImageSpatialExtentsBox의 display_width 및

display_height는 HEVC 타일 아이템의 폭 및 높이로 설정되어야 한다.

[1190] 이상의 대안의 실시예에 대한 일 변형에서, 모든 공유된 기술적 및 규범적 메타데이터를 하나의 단일 컨테이너 SharedItemPropertiesBox로 그룹화하지 않고, 2개의 상이한 컨테이너 박스들 - 하나는 기술적 메타데이터에 전용되고 다른 하나는 규범적 메타데이터에 전용됨 - 이 정의될 수 있다. 이러한 경우에, 확장된 ItemInfoEntry는 2개의 상이한 속성 인덱스 어레이들(box_prop_idx 및 box_ope_idx)을 포함하거나, 메타데이터의 타입(기술적 또는 규범적)이, 연관된 컨테이너를 검색하기 위해, 속성 인덱스 어레이(box_prop_idx)의 각각의 엔트리에 연관된다.

[1191] box_prop_idx 및 box_ope_idx의 엔트리들은 식별자들의 세트를 형성한다. 다른 식별자들의 세트는 2개의 전용 공유 박스들 내의 이미지 디스크립션 정보의 랭크에 의해 형성된다.

[1192] 다른 실시예에서, 다른 식별자들의 세트는 각각의 전용 공유 박스들 내의 이미지 디스크립션 정보에 할당된 다른 ID들에 의해 형성될 수 있다. 이 실시예는 유리하게도 식별자들의 세트에 영향을 주지 않으면서 전용 공유 박스 내의 이미지 디스크립션 정보를 순서조정할 수 있게 한다.

[1193] 양쪽 식별자들의 세트는 이미지 아이템 정보[ItemInfoBox의 아이템으로 표현됨]를 적어도 하나의 이미지 디스크립션 정보에 링크시키기 위한 구조체를 형성한다.

[1194] 본 발명의 이 마지막 양태의 추가의 예들이 부록에 설명되어 있다.

[1195] 본 발명의 다른 양태에서, 모든 기술적 및 규범적 메타데이터가 여전히 SharedItemPropertiesBox와 유사한 하나 또는 2개의 박스들로 그룹화될 수 있지만, itemInfoEntry 박스를 수정하는 것보다는, 이미지 아이템들을 그들의 기술적 및 규범적 메타데이터와 연관시키기 위해 아이템 참조 박스가 사용될 수 있다. 이 대안의 실시예에서, 2개의 상이한 컨테이너 박스들 - 하나는 기술적 속성(descriptive property)들에 대한 것(예컨대, SharedItemProperties)이고 다른 하나는 규범적 속성(prescriptive property)들에 대한 것(예컨대, SharedItemOperators)임 - 이 정의된다.

[1196] aligned(8) class SharedItemPropertiesBox extends Box('sitp') {

[1197] // one or more boxes

[1198] }

[1199] aligned(8) class SharedItemOperatorsBox extends Box('sito') {

[1200] // one or more boxes

[1201] }

[1202] 'infe' 박스를 수정하는 대신에, 이미지 및 파생 이미지 아이템들을 그들의 기술적 메타데이터 및 규범적 메타데이터(연산자들이라고도 지칭됨)에 연관시키는 데 itemReferenceBox 'iref' 박스가 사용된다.

[1203] 2개의 새로운 참조 타입들: 예를 들어, 기술적 메타데이터에 대한 'sipr' 및 규범적 메타데이터에 대한 'sior'이 정의된다.

[1204] 관계 타입(relation type)('sipr' 또는 'sior')에 따라, 아이템 참조 박스 내의 'to_item_ID' 파라미터는, 각각, SharedItemPropertiesBox 또는 SharedItemOperatorsBox에 대한 인덱스인 것으로서 해석된다. 'to_item_ID'에 연관된 참조 타입들(여기서 'sipr' 또는 'sior')은 이미지 아이템 정보(ItemInfoBox 내의 아이템에 의해 표현됨)를 이미지 디스크립션 정보(기술적 메타데이터 및 규범적 메타데이터)에 링크시키기 위한 구조체를 형성한다.

[1205] 임의의 다른 기존의 참조 타입들의 경우, 'to_item_ID' 애트리뷰트는 여전히 ItemInfoBox 내의 itemID를 가리키는 것으로서 해석된다.

[1206] 이하는 회전된 격자에 있는 다수의 이미지들을 기술하기 위해 'sipr' 및 'sior' 관계 타입들을 사용하는 일 예이다:

[1207] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1208] MetaBox: (container)

[1209] HandlerBox: hdlr = 'pict'

[1210] PrimaryItemBox: itemID = 5;

[1211] ItemInfoBox:

[1212] 1) item_type = 'hvc1', itemID=1,

[1213] item_protection_index = 0 (unused)

[1214] 2) item_type = 'hvc1', itemID=2,

[1215] item_protection_index = 0 (unused)

[1216] 3) item_type = 'hvc1', itemID=3,

[1217] item_protection_index = 0 (unused)

[1218] 4) item_type = 'hvc1', itemID=4,

[1219] item_protection_index = 0 (unused)

[1220] 5) item_type = 'grid', itemID=5,

[1221] item_protection_index = 0 (unused)

[1222] SharedItemPropertiesBox:

[1223] 0) 'hvcC'

[1224] 1) 'ispe'

[1225] SharedItemOperatorsBox:

[1226] 0) 'irot'

[1227] 1) 'grid'

[1228] ItemLocationBox:

[1229] itemID = 1, extent_count = 1,

[1230] extent_offset = X, extent_length = Y;

[1231] itemID = 2, extent_count = 1,

[1232] extent_offset = P0, extent_length = Q0;

[1233] itemID = 3, extent_count = 1,

[1234] extent_offset = P1, extent_length = Q1;

[1235] itemID = 4, extent_count = 1,

[1236] extent_offset = P2, extent_length = Q2;

[1237] ItemReferenceBox:

[1238] type='sipr', fromID=1, toID=0,1;

[1239] type='sipr', fromID=2, toID=0,1;

[1240] type='sipr', fromID=3, toID=0,1;

[1241] type='sipr', fromID=4, toID=0,1;

[1242] type='dimg', fromID=5, toID=1,2,3,4;

[1243] type='sior', fromID=5, toID=1,0;

[1244] MediaDataBox:

[1245] HEVC Image (at file offset X, with length Y)

[1246] HEVC Image (at file offset P1, with length Q1)

[1247] HEVC Image (at file offset P2, with length Q2)

[1248] HEVC Image (at file offset P3, with length Q3)

[1249] 일 변형으로서, 공유 박스들 내의 각각의 이미지 디스크립션 정보는 적절한 ID에 연관된다. 이 실시예는 유리하게도 식별자에 영향을 주지 않으면서 전용 공유 박스 내의 이미지 디스크립션 정보를 순서조정할 수 있게 한다.

[1250] 일 변형에서, 각각의 기존의 참조 타입은 itemInfoBox, SharedItemProperties 박스 또는 SharedItemOperators 박스에 암시 적으로 연관된다. 예를 들어, 'ispe', 'rloc', 'clap' 또는 'hvcC'와 같은, 기술적 메타데이터의 참조 타입들은 SharedItemProperties 박스와 연관되고, 'irot', 'iovl', 'grid'와 같은, 규범적 메타데이터의 참조 타입들은 SharedItemOperators 박스와 연관된다.

[1251] 도 9는 본 발명의 하나 이상의 실시예들을 구현하기 위한 컴퓨팅 디바이스(900)의 개략 블록도이다. 컴퓨팅 디바이스(900)는 마이크로 컴퓨터(micro-computer), 워크스테이션 또는 경량 휴대용 디바이스(light portable device)와 같은 디바이스일 수 있다. 컴퓨팅 디바이스(900)는 하기의 것들에 연결된 통신 버스를 포함한다:

[1252] - CPU로 표시된, 마이크로프로세서와 같은, 중앙 처리 유닛(901);

[1253] - 본 발명의 실시예들의 방법의 실행가능 코드를 저장하기 위한, RAM으로 표시된, 랜덤 액세스 메모리(902)는 물론, 매니페스트(manifest)들을 판독 및 기입하기 위한 그리고/또는 비디오를 인코딩하기 위한 그리고/또는 주어진 파일 포맷에 따른 데이터를 판독 또는 생성하기 위한 방법을 구현하는 데 필요한 변수들 및 파라미터들을 기록하도록 적합화된 레지스터들, 그의 메모리 용량은, 예를 들어, 확장 포트에 연결된 임의적인 RAM에 의해 확장될 수 있음;

[1254] - 본 발명의 실시예들을 구현하기 위한 컴퓨터 프로그램들을 저장하기 위한, ROM으로 표시된, 판독 전용 메모리(903);

[1255] - 네트워크 인터페이스(904)는 전형적으로 통신 네트워크 - 처리될 디지털 데이터가 이를 통해 전송 또는 수신됨 - 에 연결된다. 네트워크 인터페이스(904)는 단일 네트워크 인터페이스일 수 있거나, 상이한 네트워크 인터페이스들의 세트(예를 들어, 유선 및 무선 인터페이스들, 또는 상이한 종류의 유선 또는 무선 인터페이스들)로 구성될 수 있다. 데이터는, CPU(901)에서 실행 중인 소프트웨어 애플리케이션의 제어 하에서, 전송을 위해 네트워크 인터페이스에 기입되거나 수신을 위해 네트워크 인터페이스로부터 판독된다;

[1256] - 사용자로부터 입력들을 수신하거나 사용자에게 정보를 디스플레이하기 위한 사용자 인터페이스(9805);

[1257] - HD로 표시된 하드 디스크(906);

[1258] - 비디오 소스 또는 디스플레이와 같은 외부 디바이스들로/로부터 데이터를 수신/송신하기 위한 I/O 모듈(907)

[1259] 실행가능 코드는 판독 전용 메모리(903)에, 하드 디스크(906) 상에 또는, 예를 들어, 디스크와 같은 이동식 디지털 매체에 저장될 수 있다. 일 변형에 따르면, 프로그램들의 실행가능 코드는, 실행되기 전에, 하드 디스크(906)와 같은, 통신 디바이스(900)의 저장 수단들 중 하나에 저장되기 위해, 네트워크 인터페이스(904)를 통해, 통신 네트워크에 의해 수신될 수 있다.

[1260] 중앙 처리 유닛(901)은 본 발명의 실시예들에 따른 프로그램 또는 프로그램들의 소프트웨어 코드의 명령어들 또는 부분들의 실행을 제어 및 지시하도록 적합화되어 있고, 이 명령어들은 전술한 저장 수단들 중 하나에 저장된다. 전원을 켠 후에, CPU(901)는 소프트웨어 애플리케이션에 관련된 명령어들을, 그 명령어들이, 예를 들어, 프로그램 ROM(903) 또는 하드 디스크(HD)(906)로부터 로딩된 후에, 메인 RAM 메모리(902)로부터 실행할 수 있다. 이러한 소프트웨어 애플리케이션은, CPU(901)에 의해 실행될 때, 실시예들에 따른 방법의 단계들이 수행되게 한다.

[1261] 대안적으로, 본 발명은 하드웨어로(예를 들어, ASIC(Application Specific Integrated Circuit)의 형태로) 구현될 수 있다.

[1262] 본 발명은, 예를 들어, 특정의 관심 영역 상으로 줌인하기 위해, 카메라, 스마트폰 또는 TV에 대한 리모트 컨트

롤러(remote controller)로서 기능하는 태블릿과 같은 디바이스에 임베딩될 수 있다. 본 발명은 또한 특정 관심 영역들을 선택하는 것에 의해 동일한 디바이스들로부터 TV 프로그램의 개인화된 브라우징 경험을 갖는 데 사용될 수 있다. 사용자에 의한 이 디바이스들의 다른 용도는 그의 선호하는 비디오의 일부 선택된 서브 파트들을 다른 연결된 디바이스들과 공유하는 것이다. 본 발명이 또한 감시 카메라가 본 발명의 생성 파트(generation part)를 지원한다면 감시 하에 놓여 있는 건물의 특정 구역에서 무슨 일이 일어났는지를 모니터링하기 위해 스마트폰이나 태블릿에서 사용될 수 있다.

[1263] 본 발명은 도면들 및 전술한 설명에서 상세하게 예시되고 설명되었지만, 이러한 예시 및 설명은 제한적인 것이 아니라 설명에 도움이 되거나 예시적인 것으로 간주되어야 하며, 본 발명이 개시된 실시예로 제한되지 않는다. 도면들, 개시내용, 및 첨부된 청구항들을 살펴보는 것으로부터, 개시된 실시예에 대한 다른 변형들이 본 기술분야의 통상의 기술자에 의해 이해되고 청구된 발명을 실시하는 데 행해질 수 있다.

[1264] 청구항들에서, "포함하는(comprising)"이라는 단어는 다른 요소들 또는 단계들을 배제하지 않으며, 단수 관형사 "한" 또는 "어떤"은 복수를 배제하지 않는다. 단일 프로세서 또는 다른 유닛이 청구항들에 인용된 몇 개의 아이템들의 기능들을 수행할 수 있다. 상이한 특징들이 상호 상이한 종속 청구항들에서 인용되고 있다는 단순한 사실은 이 특징들의 조합이 유리하게 사용될 수 없다는 것을 나타내지 않는다. 청구항들에서의 임의의 참조 부호(reference sign)들이 본 발명의 범주를 제한하는 것으로 해석되어서는 안된다.

[1265] 부록

[1266] 예 1: 단일 이미지

[1267] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1268] MetaBox: (container)

[1269] HandlerBox: hdlr = 'pict'

[1270] PrimaryItemBox: itemID = 1;

[1271] ItemInfoBox:

[1272] 1) item_type = 'hvc1', itemID=1,

[1273] item_protection_index = 0 (unused),

[1274] item properties: 'hvcC', 'ispe'

[1275] ItemLocationBox:

[1276] itemID = 1, extent_count = 1, extent_offset = X,

[1277] extent_length = Y; MediaDataBox:

[1278] HEVC Image (at file offset X, with length Y)

[1279] 예 2: 회전을 갖는 단일 이미지

[1280] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1281] MetaBox: (container)

[1282] HandlerBox: hdlr = 'pict'

[1283] PrimaryItemBox: itemID = 1;

[1284] ItemInfoBox:

[1285] 1) item_type = 'hvc1', itemID=1,

[1286] item_protection_index = 0 (unused),

[1287] item properties: 'hvcC', 'ispe', 'irot'

[1288] ItemLocationBox:

[1289] itemID = 1, extent_count = 1, extent_offset = X,

[1290] extent_length = Y; MediaDataBox:

[1291] HEVC Image (at file offset X, with length Y)

[1292] 예 3: 회전 및 클린 애퍼처를 갖는 단일 이미지

[1293] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1294] MetaBox: (container)

[1295] HandlerBox: hdlr = 'pict'

[1296] PrimaryItemBox: itemID = 1;

[1297] ItemInfoBox:

[1298] 1) item_type = 'hvc1', itemID=1, item_protection_index = 0 (unused),

[1299] item properties: 'hvcC', 'ispe', 'clap', 'irot'

[1300] ItemLocationBox:

[1301] itemID = 1, extent_count = 1, extent_offset = X, extent_length = Y; MediaDataBox:

[1302] HEVC Image (at file offset X, with length Y)

[1303] 예 4: 동일한 차원들을 갖지만 다른 HEVC 구성을 갖는 다수의 이미지들

[1304] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1305] MetaBox: (container)

[1306] HandlerBox: hdlr = 'pict'

[1307] PrimaryItemBox: itemID = 1;

[1308] ItemInfoBox:

[1309] 1) item_type = 'hvc1', itemID=1,

[1310] item_protection_index = 0 (unused),

[1311] item properties: 'hvcC'

[1312] item properties indices: 0

[1313] 2) item_type = 'hvc1', itemID=2,

[1314] item_protection_index = 0 (unused)

[1315] item properties: 'hvcC'

[1316] item properties indices: 0

[1317] 3) item_type = 'hvc1', itemID=3,

[1318] item_protection_index = 0 (unused)

[1319] item properties: 'hvcC'

[1320] item properties indices: 0

[1321] 4) item_type = 'hvc1', itemID=4,

[1322] item_protection_index = 0 (unused)

[1323] item properties: 'hvcC'

[1324] item properties indices: 0

[1325] SharedItemPropertiesBox:

[1326] 0) 'ispe'

[1327] ItemLocationBox:

[1328] itemID = 1, extent_count = 1, extent_offset = X,

[1329] extent_length = Y;

[1330] itemID = 2, extent_count = 1, extent_offset = P0,

[1331] extent_length = Q0;

[1332] itemID = 3, extent_count = 1, extent_offset = P1,

[1333] extent_length = Q1;

[1334] itemID = 4, extent_count = 1, extent_offset = P2,

[1335] extent_length = Q2; MediaDataBox:

[1336] HEVC Image (at file offset X, with length Y)

[1337] HEVC Image (at file offset P1, with length Q1)

[1338] HEVC Image (at file offset P2, with length Q2)

[1339] HEVC Image (at file offset P3, with length Q3)

[1340] 예 5: 동일한 HEVC 구성 및 차원들을 갖는 다수의 이미지들

[1341] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1342] MetaBox: (container)

[1343] HandlerBox: hdlr = 'pict'

[1344] PrimaryItemBox: itemID = 1;

[1345] ItemInfoBox:

[1346] 1) item_type = 'hvc1', itemID=1,

[1347] item_protection_index = 0 (unused),

[1348] item properties indices: 0, 1

[1349] 2) item_type = 'hvc1', itemID=2,

[1350] item_protection_index = 0 (unused)

[1351] item properties indices: 0, 1

[1352] 3) item_type = 'hvc1', itemID=3,

[1353] item_protection_index = 0 (unused)

[1354] item properties indices: 0, 1

[1355] 4) item_type = 'hvc1', itemID=4,

[1356] item_protection_index = 0 (unused)

[1357] item properties indices: 0, 1

[1358] SharedItemPropertiesBox:

[1359] 0) 'hvcC'

[1360] 1) 'ispe'

[1361] ItemLocationBox:

[1362] itemID = 1, extent_count = 1, extent_offset = X,

[1363] extent_length = Y;

[1364] itemID = 2, extent_count = 1, extent_offset = P0,

[1365] extent_length = Q0;

[1366] itemID = 3, extent_count = 1, extent_offset = P1,

[1367] extent_length = Q1;

[1368] itemID = 4, extent_count = 1, extent_offset = P2,

[1369] extent_length = Q2; MediaDataBox:

[1370] HEVC Image (at file offset X, with length Y)

[1371] HEVC Image (at file offset P1, with length Q1)

[1372] HEVC Image (at file offset P2, with length Q2)

[1373] HEVC Image (at file offset P3, with length Q3)

[1374] 예 6: 동일한 HEVC 구성 및 차원들을 갖지만 상이한 회전들을 갖는 다수의 이미지들

[1375] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1376] MetaBox: (container)

[1377] HandlerBox: hdlr = 'pict'

[1378] PrimaryItemBox: itemID = 1;

[1379] ItemInfoBox:

[1380] 1) item_type = 'hvc1', itemID=1,

[1381] item_protection_index = 0 (unused),

[1382] item properties: 'irot'

[1383] item properties indices: 0, 1

[1384] 2) item_type = 'hvc1', itemID=2,

[1385] item_protection_index = 0 (unused)

[1386] item properties: 'irot'

[1387] item properties indices: 0, 1

[1388] 3) item_type = 'hvc1', itemID=3,

[1389] item_protection_index = 0 (unused)

[1390] item properties: 'irot'

[1391] item properties indices: 0, 1

[1392] 4) item_type = 'hvc1', itemID=4,

[1393] item_protection_index = 0 (unused)

[1394] item properties: 'irot'

[1395] item properties indices: 0, 1

[1396] SharedItemPropertiesBox:

[1397] 0) 'hvcC'

[1398] 1) 'ispe'

[1399] ItemLocationBox:

[1400] itemID = 1, extent_count = 1, extent_offset = X,

[1401] extent_length = Y;

[1402] itemID = 2, extent_count = 1, extent_offset = P0,

[1403] extent_length = Q0;

[1404] itemID = 3, extent_count = 1, extent_offset = P1,

[1405] extent_length = Q1;

[1406] itemID = 4, extent_count = 1, extent_offset = P2,

[1407] extent_length = Q2; MediaDataBox:

[1408] HEVC Image (at file offset X, with length Y)

[1409] HEVC Image (at file offset P1, with length Q1)

[1410] HEVC Image (at file offset P2, with length Q2)

[1411] HEVC Image (at file offset P3, with length Q3)

[1412] 예 7: 격자에 있는 다수의 이미지들

[1413] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1414] MetaBox: (container)

[1415] HandlerBox: hdlr = 'pict'

[1416] PrimaryItemBox: itemID = 5;

[1417] ItemInfoBox:

[1418] 1) item_type = 'hvc1', itemID=1,

[1419] item_protection_index = 0 (unused),

[1420] item properties indices: 0, 1

[1421] 2) item_type = 'hvc1', itemID=2,

[1422] item_protection_index = 0 (unused)

[1423] item properties indices: 0, 1

[1424] 3) item_type = 'hvc1', itemID=3,

[1425] item_protection_index = 0 (unused)

[1426] item properties indices: 0, 1

[1427] 4) item_type = 'hvc1', itemID=4,

[1428] item_protection_index = 0 (unused)

[1429] item properties indices: 0, 1

[1430] 5) item_type = 'grid', itemID=5,

[1431] item_protection_index = 0 (unused)

[1432] item properties: 'grid'

[1433] SharedItemPropertiesBox:

[1434] 0) 'hvcC'

[1435] 1) 'ispe'

[1436] ItemLocationBox:

[1437] itemID = 1, extent_count = 1, extent_offset = X,

[1438] extent_length = Y;

[1439] itemID = 2, extent_count = 1, extent_offset = P0,

[1440] extent_length = Q0;

[1441] itemID = 3, extent_count = 1, extent_offset = P1,

[1442] extent_length = Q1;

[1443] itemID = 4, extent_count = 1, extent_offset = P2,

[1444] extent_length = Q2;

[1445] ItemReferenceBox:

[1446] type='dimg', fromID=5, toID=1,2,3,4;

[1447] MediaDataBox:

[1448] HEVC Image (at file offset X, with length Y)

[1449] HEVC Image (at file offset P1, with length Q1)

[1450] HEVC Image (at file offset P2, with length Q2)

[1451] HEVC Image (at file offset P3, with length Q3)

[1452] 예 8: 회전된 격자에 있는 다수의 이미지들

[1453] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1454] MetaBox: (container)

[1455] HandlerBox: hdlr = 'pict'

[1456] PrimaryItemBox: itemID = 5;

[1457] ItemInfoBox:

[1458] 1) item_type = 'hvc1', itemID=1,

[1459] item_protection_index = 0 (unused),

[1460] item properties indices: 0, 1

[1461] 2) item_type = 'hvc1', itemID=2,

[1462] item_protection_index = 0 (unused)

[1463] item properties indices: 0, 1

[1464] 3) item_type = 'hvc1', itemID=3,

[1465] item_protection_index = 0 (unused)

[1466] item properties indices: 0, 1

[1467] 4) item_type = 'hvc1', itemID=4,

[1468] item_protection_index = 0 (unused)

item properties indices: 0, 1

5) item_type = 'grid', itemID=5,

item_protection_index = 0 (unused)

item properties: 'grid', 'irot'

SharedItemPropertiesBox:

0) 'hvcC'

1) 'ispe'

ItemLocationBox:

itemID = 1, extent_count = 1, extent_offset = X,

extent_length = Y;

itemID = 2, extent_count = 1, extent_offset = P0,

extent_length = Q0;

itemID = 3, extent_count = 1, extent_offset = P1,

extent_length = Q1;

itemID = 4, extent_count = 1, extent_offset = P2,

extent_length = Q2;

ItemReferenceBox:

type='dimg', fromID=5, toID=1,2,3,4;

MediaDataBox:

HEVC Image (at file offset X, with length Y)

HEVC Image (at file offset P1, with length Q1)

HEVC Image (at file offset P2, with length Q2)

HEVC Image (at file offset P3, with length Q3)

예 9: 오버레이를 갖는 다수의 이미지들

FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

MetaBox: (container)

HandlerBox: hdlr = 'pict'

PrimaryItemBox: itemID = 3;

ItemInfoBox:

1) item_type = 'hvc1', itemID=1,

item_protection_index = 0 (unused),

item properties indices: 0, 1

2) item_type = 'hvc1', itemID=2,

item_protection_index = 0 (unused)

item properties indices: 0, 1

3) item_type = 'iovl', itemID=3,

[1505] item_protection_index = 0 (unused)

[1506] item properties: 'iovl'

[1507] SharedItemPropertiesBox:

[1508] 0) 'hvcC'

[1509] 1) 'ispe'

[1510] ItemLocationBox:

[1511] itemID = 1, extent_count = 1, extent_offset = X,

[1512] extent_length = Y;

[1513] itemID = 2, extent_count = 1, extent_offset = P0,

[1514] extent_length = Q0;

[1515] ItemReferenceBox:

[1516] type='dimg', fromID=3, toID=1,2;

[1517] MediaDataBox:

[1518] HEVC Image (at file offset X, with length Y)

[1519] HEVC Image (at file offset P1, with length Q1)

[1520] 예 10: 하나의 이미지 및 그의 회전된 버전

[1521] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1522] MetaBox: (container)

[1523] HandlerBox: hdlr = 'pict'

[1524] PrimaryItemBox: itemID = 3;

[1525] ItemInfoBox:

[1526] 1) item_type = 'hvc1', itemID=1,

[1527] item_protection_index = 0 (unused),

[1528] item properties: 'hvcC', 'ispe'

[1529] 2) item_type = 'irot', itemID=2,

[1530] item_protection_index = 0 (unused)

[1531] item properties: 'irot'

[1532] ItemLocationBox:

[1533] itemID = 1, extent_count = 1, extent_offset = X,

[1534] extent_length = Y;

[1535] ItemReferenceBox:

[1536] type='dimg', fromID=2, toID=1;

[1537] MediaDataBox:

[1538] HEVC Image (at file offset X, with length Y)

[1539] 예 11: 타일링된 이미지들

[1540] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

```
[1541]      MetaBox:        (container)
[1542]              HandlerBox:     hdlr = 'pict'
[1543]              PrimaryItemBox: itemID = 1;
[1544]              ItemInfoBox:
[1545]                      1)      item_type = 'hvc1', itemID=1,
[1546]                              item_protection_index = 0 (unused),
[1547]                              item properties indices: 0
[1548]                              item properties: 'ispe' (W,H)
[1549]                      2)      item_type = 'hvt1', itemID=2,
[1550]                              item_protection_index = 0 (unused)
[1551]               item properties indices: 0, 1
[1552]                              item properties: 'rloc'
[1553]                      3)      item_type = 'hvt1', itemID=3,
[1554]                item_protection_index = 0 (unused)
[1555]               item properties indices: 0, 1
[1556]                              item properties: 'rloc'
[1557]                      4)      item_type = 'hvt1', itemID=4,
[1558]                              item_protection_index = 0 (unused)
[1559]               item properties indices: 0, 1
[1560]                              item properties: 'rloc'
[1561]                      5)      item_type = 'hvt1', itemID=5,
[1562]                              item_protection_index = 0 (unused)
[1563]               item properties indices: 0, 1
[1564]                              item properties: 'rloc'
[1565]              SharedItemPropertiesBox:
[1566]                      0) 'hvcC'
[1567]                      1) 'ispe' (Wt, Ht)
[1568]              ItemLocationBox:
[1569]                      itemID = 1, extent_count=1, extent_offset=X,
[1570]                              extent_length=Q0+Q1+Q2+Q3;
[1571]                      itemID = 2, extent_count=1, extent_offset=X,
[1572]                              extent_length=Q0;
[1573]                      itemID = 3, extent_count=1, extent_offset=X+Q0,
[1574]                              extent_length=Q1;
[1575]                      itemID = 4, extent_count=1, extent_offset=X+Q0+Q1,
[1576]                                                      extent_length=Q2;
```

[1577] itemID = 4, extent_count=1, extent_offset=X+Q0+Q1+Q2,

[1578] extent_length=Q3;

[1579] ItemReferenceBox:

[1580] type='tbas', fromID=2, toID=1;

[1581] type='tbas', fromID=3, toID=1;

[1582] type='tbas', fromID=4, toID=1;

[1583] type='tbas', fromID=5, toID=1;

[1584] MediaDataBox:

[1585] HEVC Image (at file offset X, with length Q0+Q1+Q2+Q3)

[1586] 예 12: 마스터 이미지와 동일한 HEVC 구성 및 차원들을 갖는 보조 이미지

[1587] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1588] MetaBox: (container)

[1589] HandlerBox: hdlr = 'pict'

[1590] PrimaryItemBox: itemID = 1;

[1591] ItemInfoBox:

[1592] 1) item_type = 'hvc1', itemID=1,

[1593] item_protection_index = 0 (unused),

[1594] item properties indices: 0, 1

[1595] 2) item_type = 'hvc1', itemID=2,

[1596] item_protection_index = 0 (unused)

[1597] item properties indices: 0, 1

[1598] item properties: 'auxC'

[1599] SharedItemPropertiesBox:

[1600] 0) 'hvcC'

[1601] 1) 'ispe'

[1602] ItemLocationBox:

[1603] itemID = 1, extent_count = 1, extent_offset = X,

[1604] extent_length = Y;

[1605] itemID = 2, extent_count = 1, extent_offset = P,

[1606] extent_length = Q;

[1607] ItemReferenceBox:

[1608] type='auxl', fromID=2, toID=1;

[1609] MediaDataBox:

[1610] HEVC Image (at file offset X, with length Y)

[1611] HEVC Image (at file offset P, with length Q)

[1612] 예 13: 서브 샘플 디스크립션을 갖는 이미지

[1613] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic'

[1614] MetaBox: (container)

[1615] HandlerBox: hdlr = 'pict'

[1616] PrimaryItemBox: itemID = 1;

[1617] ItemInfoBox:

[1618] 1) item_type = 'hvc1', itemID=1,

[1619] item_protection_index = 0 (unused),

[1620] item properties: 'hvcC', 'ispe', 'subs'

[1621] ItemLocationBox:

[1622] itemID = 1, extent_count = 1, extent_offset = X,

[1623] extent_length = Y;

[1624] MediaDataBox:

[1625] HEVC Image (at file offset X, with length Y)

[1626] 예 14: 트랙과 아이템 간의 공유 HEVC 구성

[1627] FileTypeBox: major-brand = 'heic', compatible-brands = 'heic, mp41'

[1628] MetaBox: (container)

[1629] HandlerBox: hdlr = 'pict'

[1630] PrimaryItemBox: itemID = 1;

[1631] ItemInfoBox:

[1632] 1) item_type = 'hvc1', itemID=1,

[1633] item_protection_index = 0 (unused),

[1634] item properties: 'sder' (track: 1, sample_desc_index: 1), 'ispe'

[1635] ItemLocationBox:

[1636] itemID = 1, extent_count = 1, extent_offset = X,

[1637] extent_length = Y; Movie Box: (container)

[1638] MP4에 의해 요구되는 바와 같은, 무비 헤더, 트랙들(적어도 하나의 샘플 디스크립션을 갖는 트랙 1을 포함함) 등

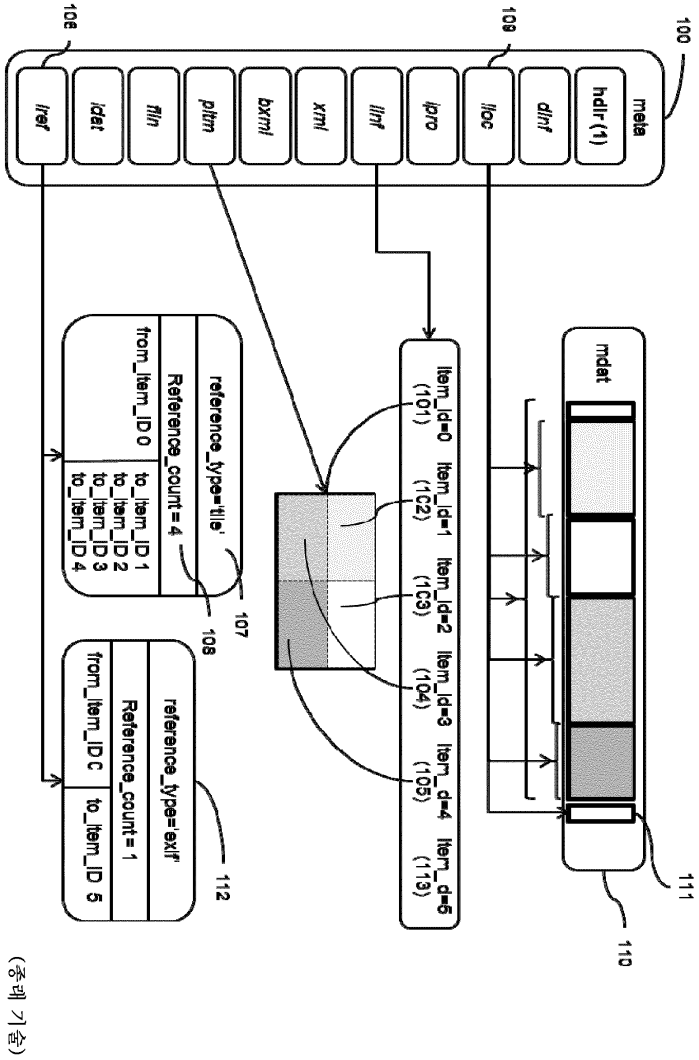
[1639] MediaDataBox:

[1640] HEVC Image (at file offset X, with length Y)

[1641] Media data as needed by the movie (some may be shared with the image data)

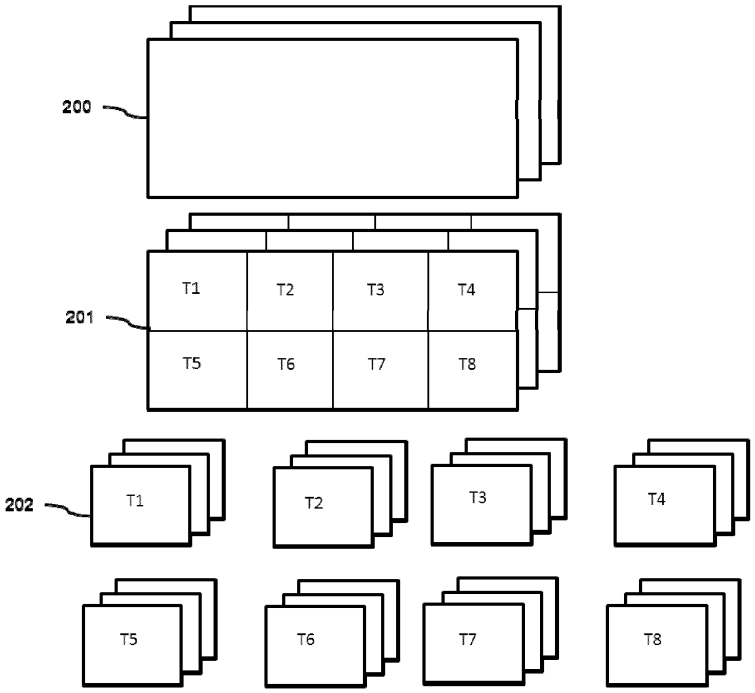
도면

도면1

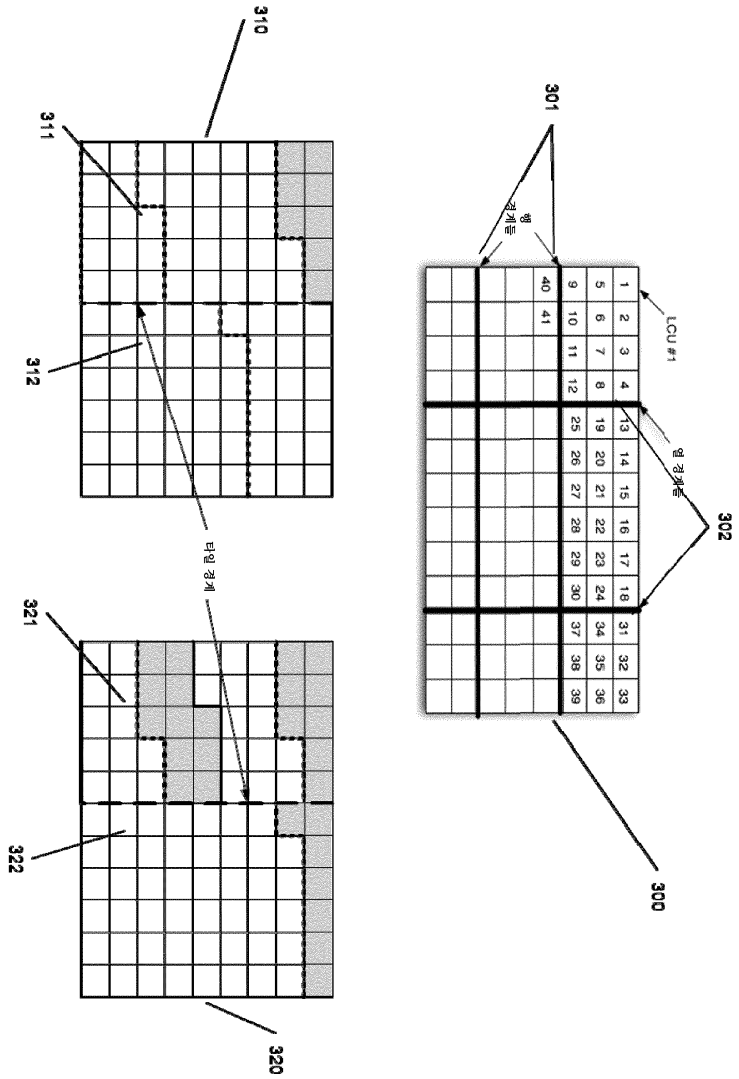

meta
hdlr (1)
dinf
iloc
ipro
iinf
xml
bxml
pitm
ffin
idat
iref
100
109
106
mdat
111
110
Item_id=0 (101)
Item_id=1 (102)
Item_id=2 (103)
Item_id=3 (104)
Item_id=4 (105)
Item_id=5 (113)
reference_type='tile'
Reference_count=4
from_Item_ID 0
to_Item_ID 1
to_Item_ID 2
to_Item_ID 3
to_Item_ID 4
107
108
reference_type='exif'
Reference_count=1
from_Item_ID C
to_Item_ID 5
112

(종래 기술)

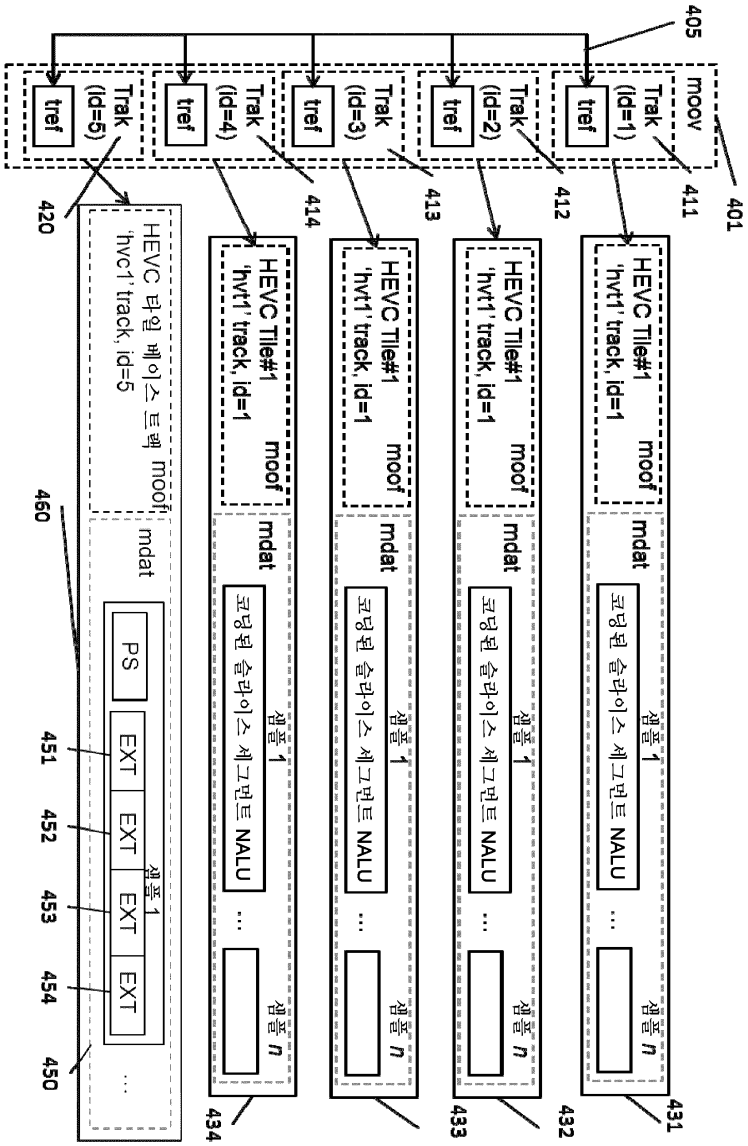
도면2



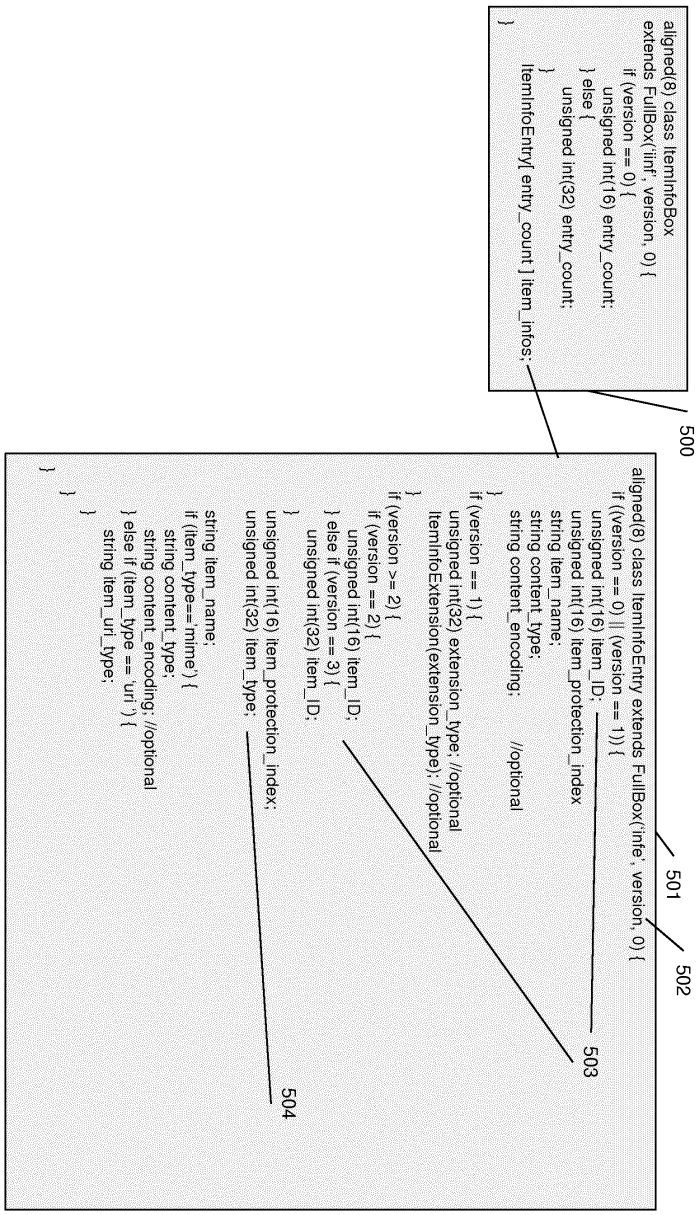
도면3



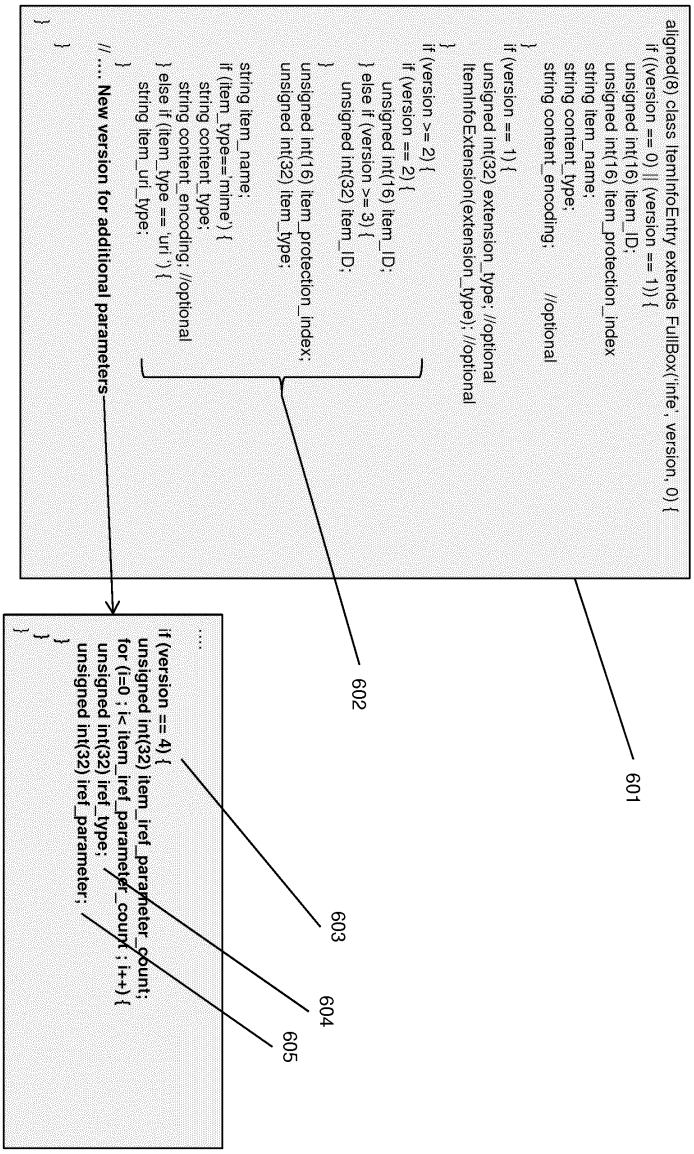
도면4


405
moov
401
Trak (id=1)
411
Trak (id=2)
412
Trak (id=3)
413
Trak (id=4)
414
Trak (id=5)
420
tref
HEVC Tile#1 'hvt1' track, id=1
moof
mdat
코딩된 슬라이스 세그먼트 NALU
샘플 1
샘플 n
431
432
433
434
HEVC 타일 베이스 트랙 'hvc1' track, id=5
PS
EXT
샘플 1
450
451
452
453
454
460

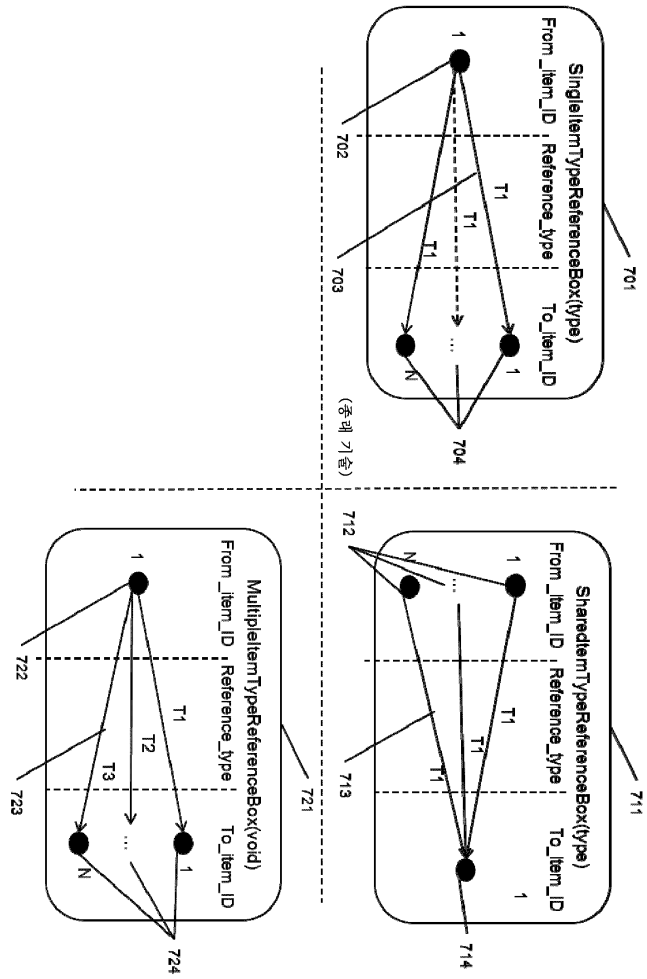
도면5


aligned(8) class ItemInfoBox
extends FullBox('iinf', version, 0) {
if (version == 0) {
unsigned int(16) entry_count;
} else {
unsigned int(32) entry_count;
}
ItemInfoEntry[entry_count] item_infos;
}
500
aligned(8) class ItemInfoEntry extends FullBox('infe', version, 0) {
if ((version == 0) || (version == 1)) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index
string item_name;
string content_type;
string content_encoding; //optional
}
if (version == 1) {
unsigned int(32) extension_type; //optional
ItemInfoExtension(extension_type); //optional
}
if (version >= 2) {
if (version == 2) {
unsigned int(16) item_ID;
} else if (version == 3) {
unsigned int(32) item_ID;
}
unsigned int(16) item_protection_index;
unsigned int(32) item_type;
string item_name;
if (item_type=='mime') {
string content_type;
string content_encoding; //optional
} else if (item_type == 'uri') {
string item_uri_type;
}
}
}
501
502
503
504

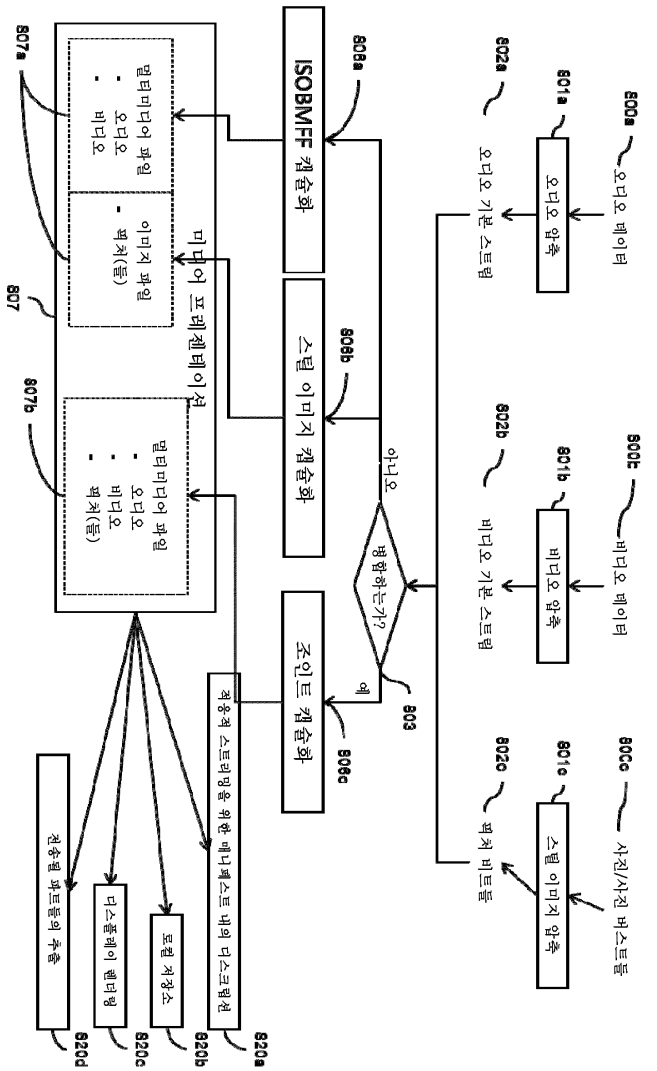
도면6


aligned(8) class ItemInfoEntry extends FullBox('infe', version, 0) {
if ((version == 0) || (version == 1)) {
unsigned int(16) item_ID;
unsigned int(16) item_protection_index
string item_name;
string content_type;
string content_encoding; //optional
}
if (version == 1) {
unsigned int(32) extension_type; //optional
ItemInfoExtension(extension_type); //optional
}
if (version >= 2) {
if (version == 2) {
unsigned int(16) item_ID;
} else if (version >= 3) {
unsigned int(32) item_ID;
}
unsigned int(16) item_protection_index;
unsigned int(32) item_type;
string item_name;
if (item_type=='mime') {
string content_type;
string content_encoding; //optional
} else if (item_type == 'uri ') {
string item_uri_type;
}
// .... New version for additional parameters
}
}
601
602
....
if (version == 4) {
unsigned int(32) item_iref_parameter_count;
for (i=0 ; i< item_iref_parameter_count ; i++) {
unsigned int(32) iref_type;
unsigned int(32) iref_parameter;
}
}
}
603
604
605

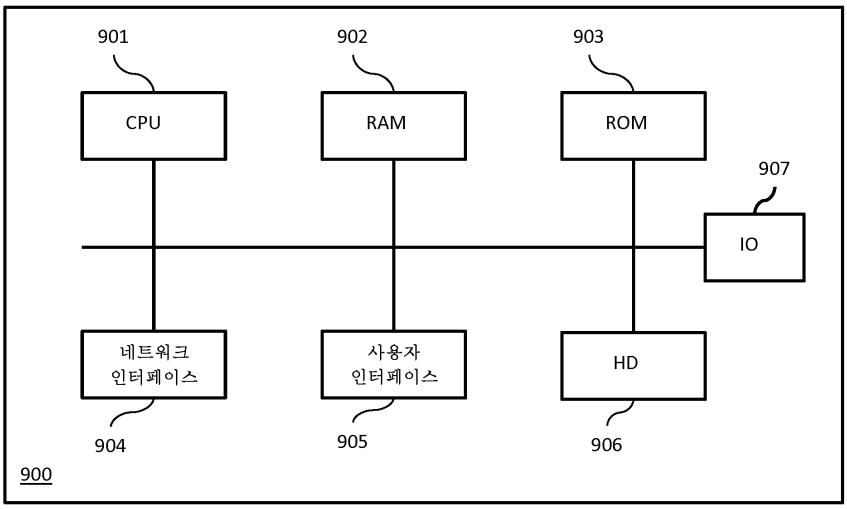
도면7


SingleItemTypeReferenceBox(type)
From_item_ID
Reference_type
To_item_ID
701
702
703
704
T1
1
N
(중략 기술)
SharedItemTypeReferenceBox(type)
711
712
713
714
MultipleItemTypeReferenceBox(void)
721
722
723
724
T2
T3

도면8


807a
멀티미디어 파일
- 오디오
- 비디오
807
이미지 파일
- 썸처(들)
미디어 프레젠테이션
807b
멀티미디어 파일
- 오디오
- 비디오
- 썸처(들)
806a
ISOBMFF 캡슐화
806b
스틸 이미지 캡슐화
806c
조인트 캡슐화
병합하는가?
아니오
예
803
802a
오디오 기본 스트림
801a
오디오 압축
800a
오디오 데이터
802b
비디오 기본 스트림
801b
비디오 압축
800b
비디오 데이터
802c
썸처 비트들
801c
스틸 이미지 압축
800c
사진/사진 버스트들
적응적 스트리밍을 위한 매니페스트 내의 디스크립션
820a
로컬 저장소
820b
디스플레이 렌더링
820c
전송될 파트들의 추출
820d

도면9


901
CPU
902
RAM
903
ROM
907
IO
네트워크
인터페이스
904
사용자
인터페이스
905
HD
906
900