(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2009-0079932
(43) 공개일자 2009년07월22일

(51) Int. Cl.
H04N 7/24 (2006.01)
(21) 출원번호 10-2009-7009623
(22) 출원일자 2007년10월15일
심사청구일자 2009년05월14일
(85) 번역문제출일자 2009년05월11일
(86) 국제출원번호 PCT/IB2007/054200
(87) 국제공개번호 WO 2008/047303
국제공개일자 2008년04월24일
(30) 우선권주장
60/852,223 2006년10월16일 미국(US)

(71) 출원인
노키아 코포레이션
핀랜드핀-02150 에스푸 카일알라덴티에 4
(72) 발명자
천 잉
핀랜드 핀-33720 탐페레 이시누우린카투 60 데 226
왕 예-쿠이
핀랜드 핀-33710 탐페레 네일리카쿠야 2 베 5
한눅셀라 미스카
핀랜드 핀-36110 루우타나 린눈라울룬쿠야 9 베 5
(74) 대리인
리앤목특허법인

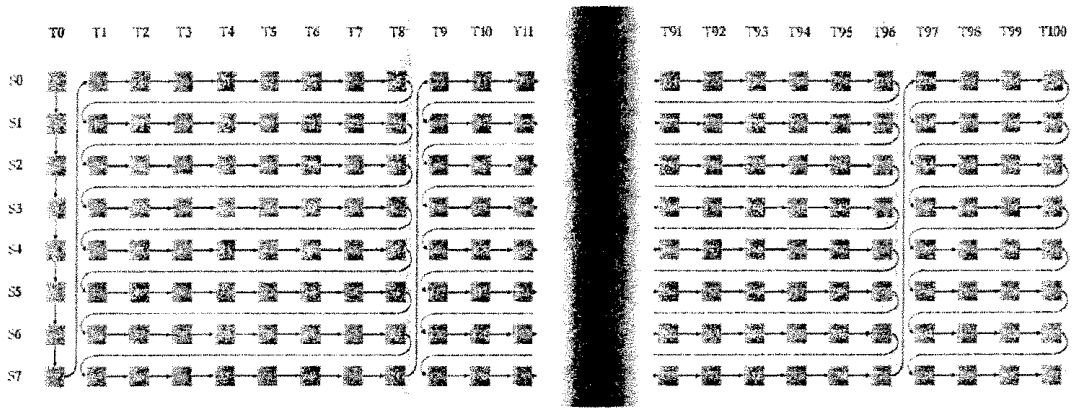
전체 청구항 수 : 총 26 항

(54) 멀티뷰 비디오 코딩에서 효율적인 디코딩된 버퍼 관리를 구현하기 위한 시스템 및 방법

(57) 요 약

제1 픽쳐 시퀀스 및 제2 픽쳐 시퀀스를 코드된 픽쳐들로 인코딩하는 시스템 및 방법. 제1 픽쳐 시퀀스와 제2 픽쳐 시퀀스는 서로 다르고, 제2 픽쳐 시퀀스의 적어도 하나의 코딩된 픽쳐는 제1 픽쳐 시퀀스의 적어도 하나의 픽쳐로부터 예측된다. 본 발명의 다양한 실시 예들에 따라, 신호 엘리먼트는 제2 픽쳐 시퀀스의 코드된 픽쳐로 인코딩된다. 신호 엘리먼트는, 제1 픽쳐 시퀀스 내의 픽쳐가 제2 픽쳐 시퀀스의 코딩된 픽쳐의 예측을 위해 사용되는지 여부를 표시한다.

대 표 도



특허청구의 범위

청구항 1

한 장면(scene)의 복수의 뷰(view)들을 나타내는 복수의 장면 신호들을 인코딩하는 방법으로서,

한 뷰의 픽쳐(picture)에 대응하는 시그널링(signalling) 엘리먼트를 제공하는 것을 포함하고,

상기 시그널링 엘리먼트는, 상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되는지 여부를 나타내는, 방법.

청구항 2

제1항에 있어서,

상기 시그널링 엘리먼트는 플래그이고 상기 픽쳐에 대응하는 NAL(network abstraction layer) 유닛 헤더 내에서 시그널링되는, 방법.

청구항 3

인코딩된 비디오 비트스트림을 디코딩하는 방법으로서, 복수의 장면 신호들의 코딩된 표현(representation)은 한 장면의 복수의 뷰들을 나타내고, 상기 방법은,

상기 인코딩된 비디오 비트스트림으로부터 한 뷰의 픽쳐에 대응하는 시그널링 엘리먼트를 검색하는 것을 포함하고,

상기 시그널링 엘리먼트는, 상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되는지 여부를 나타내는, 방법.

청구항 4

제3항에 있어서, 상기 방법은

상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되지 않는다는 것을 상기 시그널링 엘리먼트가 표시한다면, 상기 픽쳐에 대응하는 상기 인코딩된 비트스트림의 일부의 전송을 생략하는 것을 더 포함하는, 방법.

청구항 5

제3항에 있어서, 상기 방법은

상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되지 않는다는 것을 상기 시그널링 엘리먼트가 표시한다면, 상기 픽쳐에 대응하는 상기 인코딩된 비트스트림의 일부의 디코딩을 생략하는 것을 더 포함하는, 방법.

청구항 6

제3항에 있어서,

상기 시그널링 엘리먼트는 플래그이고 상기 픽쳐에 대응하는 NAL(network abstraction layer) 유닛 헤더로부터 검색되는, 방법.

청구항 7

프로세서; 및

상기 프로세서에 연결되어 통신하는 메모리 유닛을 포함하는 장치로서,

상기 메모리 유닛은,

한 뷰의 픽쳐에 대응하는 시그널링(signalling) 엘리먼트를 제공하는 컴퓨터 코드를 포함하고, 상기 시그널링 엘리먼트는, 상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되는지

여부를 나타내는, 장치.

청구항 8

제7항에 있어서,

상기 시그널링 엘리먼트는 플래그이고 상기 픽쳐에 대응하는 NAL(network abstraction layer) 유닛 헤더 내에서 시그널링되는, 장치.

청구항 9

프로세서; 및

상기 프로세서에 연결되어 통신하는 메모리 유닛을 포함하는 장치로서,

상기 메모리 유닛은,

인코딩된 비디오 비트스트림으로부터 한 뷰의 픽쳐에 대응하는 시그널링 엘리먼트를 검색하는 컴퓨터 코드를 포함하고,

상기 시그널링 엘리먼트는, 상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되는지 여부를 나타내는, 장치.

청구항 10

제9항에 있어서, 상기 장치는,

상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 하나의 픽쳐에 대한 레퍼런스(reference)로서 사용되지 않는다는 것을 상기 시그널링 엘리먼트가 표시한다면, 상기 픽쳐에 대응하는 상기 인코딩된 비트스트림의 일부의 전송을 생략하는 컴퓨터 코드를 더 포함하는, 장치.

청구항 11

제9항에 있어서, 상기 장치는,

상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되지 않는다는 것을 상기 시그널링 엘리먼트가 표시한다면, 상기 픽쳐에 대응하는 상기 인코딩된 비트스트림의 일부의 디코딩을 생략하는 컴퓨터 코드를 더 포함하는, 장치.

청구항 12

제9항에 있어서,

상기 시그널링 엘리먼트는 플래그이고 상기 픽쳐에 대응하는 NAL(network abstraction layer) 유닛 헤더로부터 검색되는, 장치.

청구항 13

인코딩된 비디오 비트스트림을 디코딩하기 위한, 컴퓨터 판독가능 매체 내에 구현된 컴퓨터 프로그램 제품으로서, 복수의 장면 신호들의 코딩된 표현(representation)은 한 장면의 복수의 뷰들을 나타내고, 상기 컴퓨터 프로그램 제품은,

상기 인코딩된 비디오 비트스트림으로부터 한 뷰의 픽쳐에 대응하는 시그널링 엘리먼트를 검색하는 컴퓨터 코드를 포함하고,

상기 시그널링 엘리먼트는, 상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되는지 여부를 나타내는, 컴퓨터 프로그램 제품.

청구항 14

제13항에 있어서, 상기 컴퓨터 프로그램 제품은,

상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 하나의 픽쳐에 대한 레퍼런스(reference)로서 사용되지 않는다

는 것을 상기 시그널링 엘리먼트가 표시한다면, 상기 픽쳐에 대응하는 상기 인코딩된 비트스트림의 일부의 전송을 생략하는 컴퓨터 코드를 더 포함하는, 컴퓨터 프로그램 제품.

청구항 15

제13항에 있어서, 상기 컴퓨터 프로그램 제품은,

상기 뷰의 픽쳐가 다른 뷰에 속하는 임의의 다른 픽쳐에 대한 레퍼런스(reference)로서 사용되지 않는다는 것을 상기 시그널링 엘리먼트가 표시한다면, 상기 픽쳐에 대응하는 상기 인코딩된 비트스트림의 일부의 디코딩을 생략하는 컴퓨터 코드를 더 포함하는, 컴퓨터 프로그램 제품.

청구항 16

제13항에 있어서,

상기 시그널링 엘리먼트는 플래그이고 상기 픽쳐에 대응하는 NAL(network abstraction layer) 유닛 헤더로부터 검색되는, 컴퓨터 프로그램 제품.

청구항 17

한 장면의 복수의 뷰들을 나타내는 복수의 장면(scene) 신호들을 인코딩하는 방법으로서,

인트라-뷰(intra-view) 레퍼런스 픽쳐들 및 인터-뷰(inter-view) 레퍼런스 픽쳐들을 기초로 초기 레퍼런스 픽쳐 리스트를 구성하고, 그리고

상기 초기 레퍼런스 픽쳐 리스트에 대하여 인터-뷰 레퍼런스 픽쳐들의 재배열을 위해 시그널링 엘리먼트를 제공하는 것[상기 시그널링 엘리먼트는 뷰 식별자(identifier) 값에 기초해서 획득됨]을 포함하는, 방법.

청구항 18

제17항에 있어서,

상기 시그널링 엘리먼트는, 상기 레퍼런스 픽쳐 리스트 내의 현재 인덱스에 배치된 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 차를 나타내는, 방법.

청구항 19

인코딩된 비디오 비트스트림을 디코딩하는 방법으로서, 복수의 장면 신호들의 코딩된 표현(representation)은 한 장면의 복수의 뷰들을 나타내고, 상기 방법은,

인트라-뷰(intra-view) 레퍼런스 픽쳐들 및 인터-뷰(inter-view) 레퍼런스 픽쳐들을 기초로 초기 레퍼런스 픽쳐 리스트를 구성하고, 그리고

상기 인코딩된 비트스트림 엘리먼트 및 뷰 식별자 값으로부터 검색된 시그널링에 기초하여 상기 초기 레퍼런스 픽쳐 리스트에 대하여 인터-뷰 레퍼런스 픽쳐들을 재배열하는 것을 포함하는 방법.

청구항 20

제19항에 있어서,

상기 시그널링 엘리먼트는, 상기 레퍼런스 픽쳐 리스트 내의 현재 인덱스에 배치된 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 차를 나타내는, 방법.

청구항 21

프로세서; 및

상기 프로세서에 연결되어 통신하는 메모리 유닛을 포함하는 장치로서,

상기 메모리 유닛은,

인트라-뷰(intra-view) 레퍼런스 픽쳐들 및 인터-뷰(inter-view) 레퍼런스 픽쳐들을 기초로 초기 레퍼런스 픽쳐 리스트를 구성하는 컴퓨터 코드, 및

상기 초기 레퍼런스 픽쳐 리스트에 대하여 인터-뷰 레퍼런스 픽쳐들의 재배열을 위해 시그널링 엘리먼트를 제공하는 컴퓨터 코드[상기 시그널링 엘리먼트는 뷰 식별자(identifier) 값에 기초해서 획득됨]을 포함하는, 장치.

청구항 22

제21항에 있어서,

상기 시그널링 엘리먼트는, 상기 레퍼런스 픽쳐 리스트 내의 현재 인덱스에 배치된 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 차를 나타내는, 장치.

청구항 23

프로세서; 및

상기 프로세서에 연결되어 통신하는 메모리 유닛을 포함하는 장치로서,

상기 메모리 유닛은,

인트라-뷰(intra-view) 레퍼런스 픽쳐들 및 인터-뷰(inter-view) 레퍼런스 픽쳐들을 기초로 초기 레퍼런스 픽쳐 리스트를 구성하는 컴퓨터 코드, 및

상기 인코딩된 비트스트림 엘리먼트 및 뷰 식별자 값으로부터 검색된 시그널링에 기초하여 상기 초기 레퍼런스 픽쳐 리스트에 대하여 인터-뷰 레퍼런스 픽쳐들을 재배열하는 컴퓨터 코드를 포함하는 장치.

청구항 24

제23항에 있어서,

상기 시그널링 엘리먼트는, 상기 레퍼런스 픽쳐 리스트 내의 현재 인덱스에 배치된 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 차를 나타내는, 장치.

청구항 25

인코딩된 비디오 비트스트림을 디코딩하기 위한, 컴퓨터 판독가능 매체 내에 구현된 컴퓨터 프로그램 제품으로서, 복수의 장면 신호들의 코딩된 표현(representation)은 한 장면의 복수의 뷰들을 나타내고, 상기 컴퓨터 프로그램 제품은,

인트라-뷰(intra-view) 레퍼런스 픽쳐들 및 인터-뷰(inter-view) 레퍼런스 픽쳐들을 기초로 초기 레퍼런스 픽쳐 리스트를 구성하는 컴퓨터 코드, 및

상기 인코딩된 비트스트림 엘리먼트 및 뷰 식별자 값으로부터 검색된 시그널링에 기초하여 상기 초기 레퍼런스 픽쳐 리스트에 대하여 인터-뷰 레퍼런스 픽쳐들의 재배열하는 컴퓨터 코드를 포함하는 컴퓨터 프로그램 제품.

청구항 26

제25항에 있어서,

상기 시그널링 엘리먼트는, 상기 레퍼런스 픽쳐 리스트 내의 현재 인덱스에 배치된 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 차를 나타내는, 컴퓨터 프로그램 제품.

명 세 서

기 술 분 야

<1> 본 발명은 일반적으로 비디오 코딩에 관련된다. 특히 본 발명은 멀티뷰 비디오 코딩에서 코딩된 픽쳐 버퍼 관리와 관련된다.

배 경 기 술

<2> 이 섹션은 청구항들에 기재된 본 발명의 배경기술 또는 콘텍스트를 제공할 의도이다. 여기의 설명은 추구될 수

있는 개념들을 포함할 수 있지만, 반드시 이전에 착상되거나 추구되었던 것들은 아니다. 따라서 여기에서 다르게 지시되지 않는한 이 섹션에서 설명되는 것은 이 출원의 설명 및 청구항들에 대한 종래 기술이 아니고 이 섹션의 삽입으로 인해 종래 기술로 인정되어서는 안 된다.

<3> 멀티뷰 비디오 코딩에서 서로 다른 카메라들로부터 나온 비디오 시퀀스들(각각은 장면(scene)의 서로 다른 뷰들에 대응)은 하나의 비트스트림으로 인코딩된다. 디코딩 이후에, 일정 뷰를 디스플레이하기 위해, 그 뷰에 속하는 디코딩된 픽처(picture)들은 재구성되고(reconstruct) 디스플레이된다. 하나 이상의 뷰가 재구성되고 디스플레이되는 것이 가능하다.

<4> 멀티뷰 비디오 코딩은, 프리-뷰포인트 비디오/텔레비전, 3차원(3D) TV 및 감시(surveillance) 애플리케이션들을 포함하는 다양한 종류의 애플리케이션들을 포함한다. 현재, ISO(International Organization for Standardization)의 JVT(Joint Video Team)/IEC(International Engineering Consotium) MPEG(Motion Picture Expert Group) 및 ITU(International Telecommunication Union)-T VCEG(Video Coding Expert Group)은, ISO/IEC MPEG-4 Part- 10로 또한 알려진 ITU-T H.264의 익스텐션(extension)이되는 MVC(multiview video coding) 표준을 개발하기 위해 작업하고 있다. 이 초안(draft) 표준들은 여기서 MVC 및 AVC로 각각 지칭된다. MVC 표준의 가장 최근 초안은, JVT-T208, "Joint Multiview Video Model (JMVM) 1.0", 20th JVT meeting, Klagenfurt, Austria에서 서술되고, ftp3.itu.ch/av-arch/jvt-site/2006 07 Klagenfurt/JVT-T208.zip에서 찾아질 수 있고, 여기에 전체로서 참조병합된다.

<5> JMVM 1.0에서, 각각의 GOP(group of pictures) 마다 임의의 뷰의 픽처들이 디코딩 순서로 연속된다. 이것은 도 1에 도시되고, 수평 방향은 시간(각각의 시간 인스턴트(instant)가 T_m으로 표현됨)이고, 수직 방향은 뷰(각각의 뷰가 S_n으로 표현됨)이다. 각 뷰의 픽처들은 GOP(group of pictures)들로 그룹화된다. 예컨대 각각의 뷰마다 도 1의 픽처들 T_1 내지 T_8이 GOP(group of pictures)를 형성한다. 이 디코딩 순서 배열은 뷰-우선(view-first) 코딩으로 지칭된다. 하나의 뷰 내 픽처들 및 하나의 GOP 내의 픽처들용, 픽처들 중 어떤 두 개의 픽처 사이에 삽입되는 어떤 다른 픽처들 없이 그들의 디코딩 순서가 연속적이지만, 내부적으로 그들의 디코딩 순서는 변할 수 있다는 것을 주목해야 한다.

<6> 뷰-우선(view-first) 코딩을 위해 논의되는 디코딩 순서와 다른 디코딩 순서를 갖는 것 또한 가능하다. 예컨대 임의의 템포럴 위치의 픽처들이 디코딩 순서로 연속적이도록 픽처들이 배열될 수 있다. 이 배열은 도 2에 보여진다. 이 디코딩 순서 배열은 시간-우선(time-first) 코딩으로 지칭된다. 액세스 유닛들의 디코딩 순서가 템포럴 순서와 동일하지 않을 수 있다는 것을 또한 주목해야 한다.

<7> (각 뷰 내의 인터-픽처(inter-picture) 예측과 인터-뷰(inter-view) 예측을 포함하는) 멀티뷰 코딩을 위한 전형적인 예측 구조가 도 2에 보여진다. 도 2에서 예측들은 화살표로 표시되고, 예측 레퍼런스를 위해 도달점 지시(pointed-to object) 객체(object)가 출발점 지시(pointed- from) 객체를 사용한다. 하나의 뷰 내의 인터-픽처 예측은 템포럴 예측, 인트라-뷰 예측, 또는 단순히 인터 예측으로서 또한 지칭된다.

<8> IDR(Instantaneous Decoding Refresh) 픽처는, IDR 픽처를 즉각적으로 디코딩한 이후에 "레퍼런스를 위해 사용되지 않음"으로 모든 레퍼런스 픽처들을 표시하도록 하는 디코딩 프로세스를 일으키는 인트라-코딩된 픽처이다. IDR 픽처의 디코딩 이후에, 디코딩 순서로 모든 후속 코딩된 픽처들은 IDR 픽처 이전에 임의의 디코딩된 픽처로부터의 인터-예측 없이 디코딩될 수 있다.

<9> AVC 및 MVC에서, 코딩된 비디오 시퀀스를 통해 변하지 않은 채로 있는 코딩 파라미터들이 시퀀스 파라미터 세트 내에 포함된다. 디코딩 프로세스에 필수적인 파라미터들에 추가하여, 시퀀스 파라미터 세트가 VUI(video usability information)를 선택적으로 담고 있을 수 있다. VUI는 버퍼링, 픽처 출력 타이밍, 렌더링, 및 리소스 예약(reservation)을 위해 중요한 파라미터들을 포함한다. 시퀀스 파라미터 세트들을 전달하도록 특정된 2개의 구조들 --시퀀스에서 AVC 픽처들을 위한 모든 데이터를 담고 있는 시퀀스 파라미터 세트 NAL, 및 MVC를 위한 시퀀스 파라미터 세트 익스텐션-- 이 존재한다. 픽처 피라미터 세트는 다수의 코딩되는 픽처들에서 변하지 않기 쉬운 이런 파라미터들을 담고 있다. 종종 픽처-레벨 데이터변경이 각 슬라이스 헤더에서 반복되고, 픽처 파라미터 세트들은 잔여 픽처-레벨 파라미터들을 전달한다. H.264/AVC 신택스는 시퀀스 및 픽처 파라미터 세트들의 많은 인스턴스들을 허용하고, 각 인스턴스는 고유 식별자로 식별된다. 각 슬라이스 헤더는, 슬라이스를 담고 있는 픽처의 디코딩을 위해 액티브 픽처 파라미터 세트의 식별자를 포함하고, 각 픽처 파라미터 세트는 액티브 시퀀스 파라미터 세트의 식별자를 담고 있다. 결론적으로 픽처 및 시퀀스 파라미터 세트들의 전송은 슬라이스들의 전송과 정확히 동기화될 필요 없다. 대신에, 액티브 시퀀스 및 픽처 파라미터 세트들이 그것들이 레퍼런스되기

전의 임의의 순간에 수신되는 것으로 충분하고, 이것은 슬라이스 데이터를 위해 사용되는 프로토콜들과 비교해서 더 신뢰적인 전송 메커니즘을 사용하는 파라미터 세트들의 전송을 허용한다. 예컨대 파라미터 세트들은 H.264/AVC RTP(Real-Time Protocol) 세션들을 위한 세션 디스크립션에서 MIME 파라미터로서 포함될 수 있다. 사용되고 있는 애플리케이션에서 가능할 때마다 아웃오브밴드(out-of-band) 신뢰 전송 메커니즘을 사용하는 것이 추천된다. 파라미터 세트들이 인밴드에서 전송되고 있다면, 그 파라미터 세트들은 오류 강건성(error robustness)을 개선시키기 위해 반복될 수 있다.

<10> 여기서 설명되는 것과 같이, 앵커 픽쳐는 코딩된 픽쳐이고, 여기서 모든 슬라이스 레퍼런스가 동일한 템포럴 인덱스로만 슬라이스한다. 즉 다른 뷰들 내에서만 슬라이스하고 현재 뷰의 초기 픽쳐들 내에서 슬라이스하지 않는다. 앵커 픽쳐가 anchor_pic_flag 를 1로 설정함으로써 시그널링된다. 앵커 픽쳐를 디코딩한 이후에, 디스플레이 순서로 모든 시퀀스 코딩된 픽쳐들은, 앵커 픽쳐 이전에 디코딩되는 임의의 픽쳐로부터의 인터-예측 없이 디코딩될 수 있다. 하나의 뷰 내의 픽쳐가 앵커 픽쳐라면, 다른 뷰들 내의 동일한 템포럴 인덱스를 가진 모든 픽쳐들은 또한 앵커 픽쳐들이다. 결론적으로, 임의의 뷰의 디코딩은 앵커 픽쳐들과 대응하는 템포럴 인덱스로부터 개시될 수 있다.

<11> 출력 타임스탬핑과 같은, 픽쳐 출력 타이밍은, AVC 또는 MVC 비트스트림들의 중요 부분(integral part) 내에 포함되지 않는다. 그러나 POC(picture order count)의 값이, 각 픽쳐를 위해 유도되고, 모든 픽쳐들을 "레퍼런스를 위해 사용되지 않음"으로 마크하는 메모리 관리 제어 오퍼레이션을 담고 있는 픽쳐 또는 이전 IDR 픽쳐에 상대적인 출력 순서로 증가하는 픽쳐 위치와 동시에 감소되지 않는다. 따라서 POC는 픽쳐들의 출력 순서를 나타낸다. B(bi-predictive) 슬라이스들의 직접 모드에서 모션벡터의 암시적(implicit) 스케일링을 위해, 가중 예측에서의 암시적으로 유도된 가중치들을 위해, 그리고 B 슬라이스들의 레퍼런스 픽쳐 리스트 초기화(initialization)를 위해, POC가 디코딩 프로세스에서 또한 사용된다. 출력 순서 일치의 검증에서 POC가 또한 사용된다.

<12> POC의 값들은 액티브 시퀀스 파라미터 세트에서 시그널링되는 3개의 모드들 중 하나로 코딩될 수 있다. 제1 모드에서, POC 값의 lsb(least significant bit)의 선택된 수가 각 슬라이스 헤더 내에 포함된다. 제2 모드에서, 코딩된 비디오 시퀀스의 디코딩 순서로 픽쳐 위치의 함수와 같은 POC의 상대적 증분(increment)들이, 시퀀스 파라미터 세트 내에서 코딩된다. 또한 시퀀스 파라미터 세트로부터 유도된 POC 값으로부터의 편차(deviation)들이 슬라이스 헤더들 내에서 포함될 수 있다. 제3 모드에서, POC의 값은, 디코딩 순서로부터 유도되는데, 그 방식은 디코딩 및 출력 순서가 동일하다고 가정하는 것이다. 또한, 제3 모드가 사용될 때 오직 하나의 비-레퍼런스 픽쳐가 연속적으로 발생할 수 있다.

<13> nal_ref_idc는 NAL 유닛 헤더 내의 2-비트 신택스 엘리먼트이다. nal_ref_idc의 값은 샘플 값들의 재구성을 위해 NAL 유닛의 관련성(relevance)을 나타낸다. nal_ref_idc의 비-제로 값들은, 파라미터 세트 NAL 유닛들을 위해서는 물론, 레퍼런스 픽쳐들의 코딩된 슬라이스 및 슬라이스 데이터 파티션 NAL 유닛들을 위해 사용되어야 한다. nal_ref_idc의 값은, 비-레퍼런스 픽쳐들의 슬라이스들 및 슬라이스 데이터 파티션들에 대해서, 그리고 보충 인핸스먼트 정보 NAL 유닛들과 같은, 샘플 값들의 재구성에 영향을 주지 않는 NAL 유닛들에 대해서 0과 동일해야 한다. H.264/AVC 하이-레벨 설계에서, 외부 명세(specification)들(즉 H.264/AVC를 참조하거나 사용하는 어떤 시스템 또는 명세)이 nal_ref_idc의 비-제로 값들로 해석(interpretation)을 명시하도록 허가되었다. H.264/AVC, RFC(Request for Comments) 3984 (www.ietf.org/rfc/rfc3984.txt에서 찾아질 수 있고 여기에 참조 병합됨) 은 nal_ref_idc의 사용에 대해 강한 추천들을 명시했다. 환언하면 몇몇 시스템들은 비제로 nal_ref_idc 값들을 설정하고 해석하는 실행(practice)들을 확립했다. 예컨대 RTP 믹서는 nal_ref_idc를 NAL 유닛 유형에 따라 nal_ref_idc를 설정할 수 있다. 예컨대, nal_ref_idc는 IDR NAL 유닛들을 위해 3으로 설정된다. MVC가 백워드-호환성 H.264/AVC 표준의 익스텐션이기 때문에, 기존의 H.264/AVC 인식 시스템 엘리먼트들이 MVC 스트림들을 또한 핸들링할 수 있는 것이 바람직하다. 따라서 nal_ref_idc의 특정 비제로 값의 시맨틱스가, nal_ref_idc의 어떤 다른 비-제로 값과 비교하여 MVC 명세에서 다르게 명시되도록 하는 것은 바람직하지 않다.

<14> 연속하는 코딩된 픽쳐들을 예측하고 장래 출력을 위해 사용되는 디코딩된 픽쳐들이, DPB(decoded picture buffer)에서 버퍼된다. 버퍼 메모리를 효율적으로 이용하기 위해서, DPB(decoded picture buffer)로의 디코딩된 픽쳐들의 저장 프로세스를 포함하는, DPB 관리 프로세스들, 레퍼런스 픽쳐들의 마킹 프로세스들, 및 DPB로부터의 디코딩된 픽쳐들의 출력 및 제거 프로세스들이 상술되어야 한다.

<15> AVC에서의 레퍼런스 픽쳐 마킹 프로세스는, 다음과 같은 것이 일반적이다. M으로 나타나는, 인터 예측을 위해 사용되는 레퍼런스 픽쳐들의 최대 수는, 액티브 시퀀스 파라미터 세트 내에서 표시된다. 레퍼런스 픽쳐가 디코

딩될 때, 그것은 "레퍼런스를 위해 사용되는 것"으로 마크된다. 레퍼런스 픽쳐의 디코딩으로 M개 이상의 픽쳐들이 "레퍼런스를 위해 사용되는 것"으로 마크된다면, 적어도 하나의 픽쳐가 "레퍼런스를 위해 사용되지 않음"으로 마크되어야 한다. 그 다음에 DPB 제거 프로세스는 DPB로부터 "레퍼런스를 위해 사용되지 않음"으로 마크되는 픽쳐들을 , 그것들이 출력을 위해 또한 필요하지 않다면 삭제할 것이다.

<16> 레퍼런스 픽쳐 마킹을 위한 오퍼레이션들의 두 가지 유형들이 있다: 적응식(adaptive) 메모리 제어 및 슬라이딩 윈도우. 레퍼런스 픽쳐 마킹을 위한 오퍼레이션 모드는 픽쳐 기반으로 선택된다. 적응식 메모리 제어는, 비트스트림 내의 MMCO(memory management control operation) 명령어들의 존재를 요구한다. 메모리 관리 제어 오퍼레이션들은, 어떤 픽쳐들이 "레퍼런스를 위해 사용되지 않음"으로 마크되는지의 명백한 시그널링, 장기(long-term) 인덱스들(indice)을 단기(short-term) 레퍼런스 픽쳐들로 할당, 현재 픽쳐를 장기 픽쳐로의 저장, 그리고 최대 허용되는 장기 인덱스(MaxLongTermFrameIdx: maximum allowed long-term index)의 할당을 가능케 한다. 슬라이딩 윈도우 오퍼레이션 모드가 사용되고 있고, M개의 픽쳐들이 "레퍼런스를 위해 사용되는 것"으로 마크된다면, "레퍼런스를 위해 사용되는 것"으로 마크되는 단기 레퍼런스 픽쳐들 중에서 첫 번째로 디코딩되는 픽쳐였던 단기 레퍼런스 픽쳐가 "레퍼런스를 위해 사용되지 않는 것"으로 마크된다. 환언하면, 슬라이딩 윈도우 오퍼레이션 모드는, 단기 레퍼런스 픽쳐들 중에서 선입선출(first-in/first-out) 버퍼링 연산을 낳는다.

<17> 각각의 단기 픽쳐는, frame_num 신택스 엘리먼트로부터 유도된 변수 PicNum와 연관된다. 각각의 장기 픽쳐는, MMCO 명령어가 시그널링하는 long_term_frame_idx 신택스 엘리먼트로부터 유도된 가변 LongTermPicNum와 연관된다. PicNum은, 프레임 또는 필드가 코딩되는지 또는 디코딩되는지 여부에 따라, FrameNumWrap 신택스 엘리먼트로부터 유도된다. PicNum이 FrameNumWrap와 동일한 프레임들의 경우에, FrameNumWrap가 FrameNum로부터 유도되고, 그리고 FrameNum가 frame_num으로부터 직접적으로 유도된다. 예컨대 AVC 디코딩에서, FrameNum은 frame_num과 동일한 값을 할당받고, FrameNumWrap은 다음과 같이 정의된다. :

<18> if(FrameNum > frame_num)

<19> FrameNumWrap = FrameNum - MaxFrameNum

<20> else

<21> FrameNumWrap = FrameNum

<22> LongTermPicNum은 픽쳐를 위해 할당된 LongTermFrameIdx(long-term frame index)로부터 유도된다. 프레임들에 대해서, LongTermPicNum은 LongTermFrameIdx와 동일하다. frame_num은 각 슬라이스 헤더 내의 신택스 엘리먼트이다. 하나의 프레임 또는 하나의 보충(complementary) 필드 쌍을 위한 frame_num의 값이, 이전 레퍼런스 프레임 또는 레퍼런스 보충 필드 쌍의 frame_num 값에 비교해서, 모듈로 산술(modulo arithmetic)에서 본질적으로 하나씩 증가한다. 모든 픽쳐들을 "레퍼런스를 위해 사용되지 않음"으로 마킹하는 메모리 제어 오퍼레이션 마킹을 담고 있는 픽쳐들을 위해서, 픽쳐의 디코딩 이후에 frame_num의 값이 0으로 간주된다.

<23> MMCO 명령어들은, 다음과 같은 명령을 위한 타깃 픽쳐를 표시하기 위해 PicNum 및 LongTermPicNum을 사용한다. 단기 픽쳐를 "레퍼런스를 위해 사용되지 않음" 으로 마크하기 위해, 현재 픽쳐 p와 목적지 픽쳐 r 간의 PicNum 차이가 MMCO 명령어 내에서 시그널링된다. 장기 픽쳐를 "레퍼런스를 위해 사용되지 않음"으로 마킹하기 위해서, 제거될(to-be-removed) 픽쳐 r 의 LongTermPicNum이 MMCO 명령어 내에서 시그널링된다. 현재 픽쳐 p를 장기 픽쳐로 저장하기 위해서, long_term_frame_idx가 MMCO 명령어와 함께 시그널링된다. 이 인덱스는 LongTermPicNum의 값으로서 새롭게 저장되는 장기 픽쳐에 할당된다. 단기 픽쳐로부터 장기 픽쳐로 픽쳐 r 변경하기 위해서 현재 픽쳐 p와 픽쳐 r 간의 PicNum 차이가 MMCO 명령어 내에서 시그널링되고, long_term_frame_idx가 MMCO 명령어 내에서 시그널링되고, 그리고 인덱스가 이 장기 픽쳐로 할당된다.

<24> 다중 레퍼런스 픽쳐들이 사용될 수 있을 때, 각 레퍼런스 픽쳐가 식별되어야 한다. AVC에서, 코딩된 블록을 위해 사용되는 레퍼런스 픽쳐의 ID(identification)는 다음과 같다. 첫째로 장래 픽쳐들의 예측 레퍼런스를 위해 DPB 내에 저장되는 모든 레퍼런스 픽쳐들이 "단기 레퍼런스를 위해 사용되는 것"(단기 픽쳐들) 또는 "장기 레퍼런스를 위해 사용되는 것"(장기 픽쳐들) 중 어느 하나로 마크된다. 코딩된 슬라이스를 디코딩할 때, 레퍼런스 픽쳐 리스트가 구성된다. 코딩된 슬라이스가 B(bi- predicted) 슬라이스라면, 제2 레퍼런스 픽쳐 리스트가 또한

구성된다. 코딩된 블록을 위해 사용되는 레퍼런스 픽쳐가, 레퍼런스 픽쳐 리스트 내의 사용되는 레퍼런스 픽쳐의 인덱스로 식별된다. 하나 이상의 레퍼런스 픽쳐가 사용될 수 있을 때 인덱스가 비트스트림 내에서 코딩된다.

<25> 레퍼런스 픽쳐 리스트 구성 프로세스가 다음과 같다. 단순화를 위해 오직 하나의 레퍼런스 픽쳐 리스트가 필요하다고 가정한다. 우선 초기 레퍼런스 픽쳐 리스트는, 단기 픽쳐 및 장기 픽쳐 모두를 포함하는 것으로 구성된다. 그 다음에, 슬라이스 헤더가 RPLR(Reference picture list reordering) 명령어들을 담고 있을 때, RPLR(reference picture list reordering)이 수행된다. RPLR 프로세스는, 초기 리스트에서의 순서와 다른 순서로 레퍼런스 픽쳐들을 재배열할(reorder) 수 있다. 최종적으로 가능하게는 재배열된 리스트의 시작에서 다수개의 픽쳐들만을 보유함으로써 최종 리스트가 구성되고, 슬라이스가 나타내는 픽쳐 파라미터 세트 또는 슬라이스 헤더 내의 다른 하나의 신택스 엘리먼트에 의해 그 개수가 표시된다.

<26> 초기화 프로세스 동안에, 단기 및 장기 픽쳐들의 모두가, 현재 픽쳐용 레퍼런스 픽쳐 리스트를 위한 후보들로서 간주된다. 현재 픽쳐가 B 픽쳐 또는 P 픽쳐이든지 여부에 불구하고, 장기 픽쳐들이 RefPicList0(B 슬라이스들을 위해 이용가능한 RefPicList1)내의 단기 픽쳐들 이후에 위치된다. P 픽쳐들을 위해, RefPicList0용 초기 레퍼런스 픽쳐 리스트가 PicNum의 내림차순으로 배열된 모든 단기 레퍼런스 픽쳐들을 담고 있다. B 픽쳐들을 위해, 모든 단기 픽쳐들로부터 획득된 그 레퍼런스 픽쳐들이, 현재 POC 개수와 레퍼런스 픽쳐의 POC 개수와 관련된 규칙--RefPicList0를 위해, (현재 POC와 비교해서) 더 작은 POC를 갖는 레퍼런스 픽쳐들이 첫번째로 간주되고, POC의 내림차순으로 RefPictList0로 삽입--으로 배열된다. 그 다음에 더 큰 POC를 갖는 픽쳐들은, POC의 올림차순으로 첨부된다. RefPicList1이 이용가능하다면, (현재 POC와 비교해서) 더 큰 POC를 갖는 레퍼런스 픽쳐들이 첫 번째로 간주되고 POC의 올림 차순으로 RefPicList1로 삽입된다. 모든 단기 레퍼런스 픽쳐들을 고려한 이후에, 장기 레퍼런스 픽쳐들은 P 픽쳐 및 B 픽쳐 모두를 위해 LongTermPicNum의 올림 차순으로 첨부된다.

<27> 렌더링 프로세스는 4개의 유형들을 포함하는, 연속 RPLR 명령어들에 의해 야기된다. 제1 유형은 이동될 (시간적으로(temporally) 예측되는 PicNum과 비교해서) 더 작은 PicNum를 갖는 단기 픽쳐를 특정하는 명령어이다. 제2 유형은 이동될 더 큰 PicNum를 갖는 단기 픽쳐를 특정하는 명령어이다. 제3 유형은 이동될 일정 LongTermPicNum를 갖는 장기 픽쳐와 RPLR 루프의 끝을 특정하는 명령어이다. 현재 픽쳐가 B(bi-predicted)이면, 2개의 루프들--하나는 포워드 레퍼런스 리스트용이고 다른 하나는 백워드 레퍼런스 리스트를 위한 것--이 있다.

<28> picNumLXPred로 지칭되는 예측된 PicNum는, 현재 코딩된 픽쳐의 PicNum로 초기화된다. 이것은 단기 픽쳐용 각각의 재배열 프로세스 이후에 방금 이동된 픽쳐의 PicNum로 설정된다. 재배열되는 현재 픽쳐의 PicNum와 picNumLXPred 사이의 차이가 RPLR 명령어 내에서 시그널링된다. 재배열되는 것으로 표시된 픽쳐가 레퍼런스 픽쳐 리스트의 시작으로 이동한다. 재배열 프로세스가 완료된 이후에, 전체 레퍼런스 픽쳐 리스트가, num_ref_idx_lX_active_minus1+1 (0 또는 1과 동일한 X는, 각각 RefPicList0 와 RefPicList1에 대응)인 액티브 레퍼런스 픽쳐 리스트 크기에 기초해서 절단된다(truncated).

<29> H.264/AVC 표준의 Annex C에서 상술된, HRD(hypothetical reference decoder)는 비트스트림 및 디오더 적합성(conformance)을 체크하기 위해 사용된다. HRD는 CPB(coded picture buffer), IDP(instantaneous decoding process), DPB(decoded picture buffer), 및 출력 픽쳐 크롭핑 블록(output picture cropping block)을 담고 있다. CPB 및 IDP(instantaneous decoding process)는 임의의 다른 비디오 코딩 표준과 유사하게 특정되고, 그리고 출력 픽쳐 크롭핑 블록은, 시그널된 출력 픽쳐 범위(extent)의 외부에 있는 디코딩된 픽쳐로부터 그 샘플들을 단순히 크롭한다. DPB는 적합 비트스트림들의 디코딩을 위해 요구되는 메모리 리소스들을 제어하기 위해 H.264/ AVC 내에서 도입되었다 .

<30> 인터 예측에서의 레퍼런스들을 위해서 및 출력 순서로의 디코딩된 픽쳐들의 재배열을 위해 디코딩된 픽쳐들을 버퍼링하기 위한 2가지 이유가 있다. H.264/AVC 표준이 레퍼런스 픽쳐 마킹 및 출력 재배열 모두를 위해 상당한 정도의 유연성(flexibility)을 제공하기 때문에, 레퍼런스 픽쳐 버퍼링 및 출력 픽쳐 버퍼링을 위한 별개의 버퍼들이 메모리 리소스들의 낭비일 수 있다. 따라서 DPB는 레퍼런스 픽쳐들 및 출력 재배열을 위한 통합된 디코딩된 픽쳐 버퍼링 프로세스를 포함한다. 디코딩된 픽쳐는 더 이상 레퍼런스로서 사용되지 않거나 출력을 위해 필요하지 않을 때 DPB로부터 제거된다. 비트스트림들이 사용하도록 허용된 DPB의 최대 크기는, H.264/ AVC 표준의 레벨 정의들(아넥스 A)에서 상술된다.

<31> 디코더들을 위한 적합성(conformance)의 두 가지 유형이 있다 : 출력 타이밍 적합성 및 출력 순서 적합성. 출력 타이밍 적합성의 경우에, 디코더가 HRD와 비교해서 동일한 시간대들에서 픽쳐들을 출력해야 한다. 출력 순서 적합성의 경우에, 출력 픽쳐의 정확한 순서만이 고려되어야 한다. 출력 순서 DPB는, 프레임 버퍼들의 최대 허용되는 개수를 담고 있는 것으로 간주된다. 프레임은 레퍼런스로서 더 이상 사용되지 않거나 출력을 위해 필요하지

않을 때 DPB로부터 제거된다. DPB가 가득 찼을 때, 적어도 하나의 프레임 버퍼가 덜 차게 될 때까지 출력 순서에서 가장 이른 프레임이 출력이다.

<32> 템포럴 스케일러빌러티(scalability)가 오직 AVC 툴들만을 사용해서 계층적 B 픽쳐 GOP 구조에 의해 실현된다. 전형적인 템포럴 스케일러빌러티 GOP는 I 프레임 또는 P 프레임으로서 코딩되는 키(key) 픽쳐, 및 B 픽쳐들로서 코딩되는 다른 픽쳐들을 포함하는 것이 보통이다. 그 B 픽쳐들은 POC에 기초해서 계층적으로 코딩된다. GOP의 코딩은, GOP 내의 키 픽쳐들 이외에 이전 GOP의 키 픽쳐들만이 필요하다. 상대적 POC의 개수(POP에서 이전 앵커 픽쳐 POC 뺌)는 구현에서 POCIdInGOP로 나타난다. 모든 POCIdInGOP는 POCIdInGOP=$2^x y$ (여기서 y는 홀수이다)의 형태를 가질 수 있다. x와 동일한 값을 갖는 픽쳐들은, L-x (여기서 L = *log2(GOP_length)*)로서 표기되는 동일한 템포럴 레벨에 속한다. 가장 높은 템포럴 레벨 L을 갖는 픽쳐들만이 레퍼런스 픽쳐들로서 저장된다. 일반적으로 템포럴 레벨 내의 픽쳐들은, 템포럴 스케일러빌러티를 지원하는 레퍼런스들로서 더 낮은 템포럴 레벨들 내의 픽쳐들을 사용한다. 즉 더 높은 템포럴 레벨 픽쳐들이 더 낮은 템포럴 레벨 픽쳐들의 디코딩에 영향을 줌이 없이 드롭될 수 있다. 유사하게, 동일한 계층적 구조가 뷰 스케일러빌러티를 위한 뷰 디멘션(dimension)에 적용될 수 있다.

<33> 현재 JMVM에서, frame_num은 각각의 뷰를 위해 별개로 코딩되고 시그널링된다. 즉 frame_num의 값이 현재 픽쳐와 동일한 뷰 내에서 이전 레퍼런스 프레임 또는 레퍼런스 보충(complementary) 필드 쌍에 상대적으로 증분된다. 또한 모든 뷰들 내의 픽쳐들이 동일한 DBP 버퍼를 공유한다. 레퍼런스 픽쳐 리스트 구성 및 레퍼런스 픽쳐 관리를 글로벌하게 핸들링하기 위해, FrameNum및 POC 생성이 다음과 같이 정의된다:

<34> FrameNum=frame__num * (1 + num_views_minus_1) + view_id

<35> PicOrderCnt() = PicOrderCnt() * (1 + num_views_minus_1) + view_id;

<36> JMVM는 AVC를 위해 사용되는 것과 동일한 레퍼런스 픽쳐 마킹을 따른다. 유일한 차이는, FrameNum Wrap이 다음과 같이 재정의되도록 JMVM에서 FrameNum가 재정의된다는 것이다.

<37> if(FrameNum > frame_num * (1 + num_views_minus_1) + view_id)

<38> FrameNum Wrap = FrameNum - MaxFrameNum * (1 + num_views_minus_1) + view_id

<39> else

<40> FrameNumWrap = FrameNum

<41> 현재 JMVM 표준에서, 인터-뷰 레퍼런스 픽쳐들은 SPS (Sequence Parameter Set) 익스텐션(extension) 내에 암시적으로 상술되고, 여기서 인터뷰 레퍼런스 리스트들의 액티브 개수 및 그 픽쳐들의 view_id 가 상술된다. 이 정보는 동일 SPS로 나타내지는 모든 픽쳐들에 의해 공유된다. 래퍼런스 픽쳐 리스트 구성 프로세스는, AVC에서와 동일한 방식이지만, DPB에 저장된 레퍼런스 픽쳐들 모두를 고려해서 우선 레퍼런스 픽쳐 리스트 초기화, 재배열, 및 절단(truncation)을 처리한다. SPS에서 상술되고 동일한 템포럴 축 내(즉 동일한 캡쳐/출력 시간을 갖음)에 있는 view_id 들을 갖는 픽쳐들은, SPS에서 리스트된 순서로 레퍼런스 리스트에 첨부된다.

<42> 안타깝게도 위의 JMVM 표준은 다수의 문제가 생기게 한다. (디코더에 의해) 디코딩되고, (송신기에 의해) 송신되고, (미디어 게이트웨이 또는 MANE에 의해) 포워드되는 뷰들의 스위칭은, 앵커 픽쳐들에 대응되는 시간 인덱스 이외의 시간 인덱스에서 생길 수 있는 것이 때때로 바람직할 수 있다. 예컨대 베이스 뷰는 최대 코딩 효율성을 위해 압축될 수 있고(템포럴 예측이 심하게 사용됨), 앵커 픽쳐들이 드물게 코딩된다. 결론적으로 다른 뷰들을 위한 앵커 픽쳐들이 모든 뷰들을 통해 동기화되기 때문에 또한 드물게 생긴다. 현재 JMVM 신택스는, (그 시간 인덱스의 모든 뷰들이 앵커 픽쳐를 담고 있을 때까지) 일정 뷰의 디코딩이 시작될 수 있는 픽쳐의 시그널링을 포함하지 않는다.

<43> 두 번째로, 인터-뷰 예측을 위해 허용되는 레퍼런스 뷰들이, 각 뷰에 대해(그리고 앵커 픽쳐 및 비-앵커 픽쳐에 대해 개별적으로) 상술된다. 그러나, 코딩되고 있는 픽쳐와 동일 템포럴 축(axis) 내 및 잠재적 레퍼런스 뷰 내의 잠재적(potential) 픽쳐 간의 유사성에 의존하여, 인터-뷰 예측이 인코더 내에서 수행될 수도 수행되지 않을 수도 있다. 현재 JMVM 표준은, 픽쳐가 인트라-뷰 예측 또는 인터-뷰 예측을 위해 사용되는지를 표시하기 위해 nal_ref_idc를 사용하지만, 그것은 픽쳐가 인트라-뷰 예측 및/또는 인터-뷰 예측을 위해 사용되는지를 별도로 표시할 수 없다. 또한 JMVM 1 ,0에 따르면, AVC 호환성(compatible) 뷰를 위해, 픽쳐가 템포럴 예측을 위해 사용되지 않고 이때 그것이 인터 뷰 예측 레퍼런스를 위해서만 사용될지라도 nal_ref_idc는 0과 동일하지 않게 설

정되어야 한다. 결론적으로 그 뷰만이 디코딩되고 출력된다면, 이런 픽쳐들이 디코딩 되자마자 출력될 수 있을 때 이런 픽쳐들의 저장을 위해 추가의 DPB 사이즈가 필요하다.

<44> 세 번째로, JMVM 1.0에 상술된 레퍼런스 픽쳐 마킹 프로세스가, FrameNum, FrameNumWrap 및 당연히 PicNum의 재정의를 제외하고, AVC 프로세스와 기본적으로 동일하다. 따라서 다수의 특별한 문제들이 생긴다. 인터-뷰 예측을 위해 버퍼될 필요가 있는 디코딩되는 픽쳐들(특히 그 픽쳐들이 템포럴 예측 레퍼런스를 위해 사용될 때)의 관리를 이 프로세스는 효율적으로 처리할 수 없다. 그 이유는, AVC 표준에서 상술된 DPB 관리 프로세스가 단일-뷰 코딩을 위해 의도되었기 때문이다. AVC 표준에 있는 것과 같은 단일-뷰 코딩에서, 템포럴 예측 레퍼런스 또는 장래 출력을 위해 버퍼될 필요가 있는 디코딩되는 픽쳐들이, 그것들이 템포럴 예측 레퍼런스 및 장래 출력을 위해 더 이상 필요하지 않을 때 버퍼로부터 제거될 수 있다. 템포럴 예측 레퍼런스 및 장래 출력을 위해 더 이상 필요하게 되지 않을 때 레퍼런스 픽쳐의 제거를 가능하게 하기 위해서, 레퍼런스 픽쳐가 템포럴 예측 레퍼런스를 위해 더 이상 필요로 되지 않은 이후에 즉각적으로 알려질 수 있도록 레퍼런스 픽쳐 마킹 프로세스가 상술된다. 그러나 그것이 인터-뷰 예측 레퍼런스를 위한 픽쳐들이 되었을 때, 픽쳐가 인터-뷰 예측 레퍼런스를 위해 더 이상 필요로되지 않은 이후에 즉각적으로 알 방법이 없다. 결론적으로 인터-뷰 예측 레퍼런스를 위한 픽쳐들이 DPB에서 불필요하게 버퍼될 수 있고, 이것은 버퍼 메모리 사용의 효율성을 감소시킨다.

<45> 다른 하나 예에서, PicNum을 재계산하는 방법이 주어질 때, 슬라이딩 윈도우 오퍼레이션 모드가 사용되고 있고 장기 및 단기 픽쳐들의 개수가 최대와 동일하다면,최소 FrameNumWrap을 갖는 단기 레퍼런스 픽쳐가 "레퍼런스를 위해 사용되지 않음"으로 마크된다. 그러나 현재 JMVM에서 FrameNum 순서가 디코딩 순서를 따르지 않기 때문에 이 픽쳐가 반드시 가장 초기에 코딩된 픽쳐가 아니라는 사실 때문에, 슬라이딩 윈도유 레퍼런스 픽쳐 마킹은 현재 JMVM에서 최상으로 동작하지 않는다. 또한 PicNum가 재정의되고 스케일된 FrameNumWrap로부터 유도되었다는 사실 때문에, 2개의 코딩된 픽쳐들의 PicNum 값들 사이의 차이가 평균으로 스케일될 것이다. 예컨대 동일한 뷰내에 그리고 3과 5와 동일한 frame_num를 갖는 픽쳐들이 존재한다고 가정한다는 것이 유용하다. 오직 하나의 뷰만이 존재할 때, 즉 비트스트림이 AVC 스트림일 때, 2개의 PicNum 값들의 차이는 2가 될 것이다. 코딩 픽쳐가 5와 동일한 frame_num를 가질 때, MMCO 명령어가 3과 동일한 PicNum를 갖는 픽쳐를 "레퍼런스를 위해 사용되지 않음"으로 마크하기 위해 필요하다면, 그 2개의 값들의 차이에서 1을 뺀 것은 1과 동일하고, 이것이 MMCO로 시그널링된다. 이 값은 3비트가 필요하다. 그러나 256개의 뷰들이 존재한다면, 2개의 PicNum 값들의 차이에서 1을 뺀 것은 511이 될 것이다. 이 경우에, 19 비트가 그 값을 시그널링하기 위해 필요하다. 결론적으로 MMCO 명령어는 덜 효율적으로 디코딩된다. 전형적으로 비트들의 증가된 개수는, H.264/ AVC의 단일-뷰 코딩과 비교했을 때 현재 JMVM의 MMCO 명령어용으로 2*log2(뷰들의 개수)와 동일하다.

<46> 문제들의 4번째 세트는, JMVM 1.0에서 상술된 레퍼런스 픽쳐 리스트 구성 프로세스를 둘러싼 것이다. 레퍼런스 픽쳐 리스트 초기화 프로세스는 재배열 프로세스 이전에 모든 뷰들로부터의 레퍼런스 픽쳐들을 고려한다. 그러나 인터-뷰 예측을 위해 사용되는 다른 뷰들로부터의 픽쳐들이 리스트를 절단한(truncating) 이후에 리스트에 첨부된다는 사실 때문에, 다른 뷰들로부터의 레퍼런스 픽쳐들은 재배열 및 절단 이후에 레퍼런스 픽쳐 리스트 내에 어쨌든 나타나지 않는다. 따라서 초기화 프로세스에서 그 픽쳐들에 대한 고려가 필요하지 않다. 또한 불법적인 레퍼런스 픽쳐들(이런 픽쳐들은 현재 픽쳐와는 다른 view_id 를 가지고 있고 현재 픽쳐와 시간적으로(temporally) 정렬되지 않음) 및 반복되는 인터-뷰 레퍼런스 픽쳐들이 최종적으로 구성된 레퍼런스 픽쳐 리스트 내에 나타날 수 있다.

<47> 레퍼런스 픽쳐 리스트 초기화 프로세스는 다음 단계들에서 리스트된 것과 같이 동작한다. :레퍼런스 픽쳐들의 모두가, 그것들의 view_id 와 상관없이 그리고 그것들이 현재 픽쳐와 시간적으로 정렬되어 있는지 여부와 관계없이 초기 리스트내에 포함된다. 환언하면 초기 레퍼런스 픽쳐 리스트는 불법 레퍼런스 픽쳐들(이런 픽쳐들은 현재 픽쳐와는 다른 view_id 를 가지고 있고 현재 픽쳐와 템포럴하게 정렬되지 않음)을 포함할 수 있다. 그러나 뷰-우선 코딩에서, 초기 리스트의 시작은 현재 픽쳐와 동일한 뷰로부터의 레퍼런스 픽쳐들을 담고 있다. (2) 인트라-뷰 레퍼런스 픽쳐들 및 인터-뷰 레퍼런스 픽쳐들 모두가 재배열될 수 있다. 재배열 이후에, 리스트의 시작은 여전히 불법 레퍼런스 픽쳐들을 담고 있다. (3) 리스트가 절단되지만, 절단된 리스트는 여전히 불법 레퍼런스 픽쳐들을 담고 있을 수 있다. (4) 인터-뷰 레퍼런스 픽쳐들은, 그것들이 SPS의 MVC 익스텐션에서 나타나는 순서로 리스트 내에 첨부된다.

<48> 또한 JMVM 1.0에서 상술된 레퍼런스 픽쳐 리스트 재배열 프로세스는, 인터-뷰 프레임들 [그것들은 SPS의 MVC 익스텐션에서 나타나는 때의 순서로 리스트의 끝에 항상 놓임] 의 재배열을 고려하지 않는다.이것은 레퍼런스 픽쳐 리스트 구성에서 유동성을 적게하고, 그 결과 압축 효율성이 감소되고, 이때 인터-뷰 레퍼런스 프레임들의 디폴트 순서가 최적이 아니거나 일정 인터-뷰 레퍼런스 프레임들이 일정 인트라-뷰 레퍼런스 프레임들의 예측을

위해서 더 많이 사용될 경향이 있다. 또한 MMCO 명령어들과 유사하게, 재정의되고 스케일된 FrameNumWrap으로부터 PicNum가 유도되었다는 사실 때문에, PicNum 값들의 차이에서 1을 뺀 값의 시그널링을 포함하는 RPLR 명령어들의 코딩을 위해 H.264/ AVC 표준의 단일-뷰 코딩과 비교해서 더 긴 VLC 코딩워드들이 요청된다.

발명의 상세한 설명

<49> 본 발명은 멀티뷰 비디오 코딩에서 효율적인 디코딩된 픽쳐 버퍼 관리를 구현하기 위한 개선된 시스템 및 방법을 제공한다. 하나의 실시 예에서, 한 뷰의 디코딩이 일정 픽쳐에서 시작될 수 있는지 여부를 표시하기 위해 새로운 플래그가 사용된다. 더 자세한 실시 예에서, 이 플래그는 NAL(network abstraction layer) 유닛 헤더 내에서 시그널링된다. 다른 하나의 실시 예에서, 한 픽쳐가 인터-뷰(inter -view) 예측 레퍼런스를 위해 사용되는지 여부를 표시하기 위해 새로운 플래그가 사용되고, 반면에 신택스 엘리먼트 nal_ref_idc는 한 픽쳐가 템포럴(temporal) 예측 레퍼런스를 위해 사용되는지 여부만을 표시한다. 이 플래그는 NAL(network abstraction layer) 유닛 헤더 내에서 또한 시그널링될 수 있다. 세 번째 실시 예에서, 새로운 레퍼런스 픽쳐 마킹 방법들의 하나의 세트가, 디코딩된 픽쳐들을 효율적으로 관리하기 위해 사용된다. 이런 방법들은 슬라이딩 윈도우 및 적응식(adaptive) 메모리 제어 메커니즘들을 모두 포함한다. 네 번째 실시 예에서, 새로운 레퍼런스 픽쳐 리스트 구성(construction) 방법들의 하나의 세트가 사용되고, 레퍼런스 리스트 초기화 및 재배열(reordering) 모두를 포함한다.

<50> 본 발명의 이런 그리고 다른 이점들 및 특징들이, 그 동작 구성 및 방법과 더불어, 첨부된 도면들과 연관해서 다음의 상세할 설명으로부터 명백해질 것이다. 첨부된 도면들에서 동일한 엘리먼트들은 아래에서 설명되는 다수의 도면들을 통해서 동일한 참조번호들을 갖는다.

실 시 예

<57> 도 4은 본 발명에 사용되는 일반적 멀티미디어 통신 시스템을 보여준다. 도 4에 도시된 바와 같이, 데이터 소스(100)는 아날로그, 압축해제된 디지털이거나, 압축된 디지털 포맷 또는 이런 포맷들의 임의의 조합으로 소스 신호를 제공한다. 인코더(110)는 소스 신호를 코딩된 미디어 비트스트림으로 인코딩한다. 인코더(110)는 오디오 및 비디오와 같은 하나 이상의 미디어 타입을 인코딩할 수 있거나, 하나 이상의 인코더(110)는 소스 신호의 다른 미디어 유형들을 코딩하도록 요구될 수 있다. 인코더(110)는 그래픽스와 텍스트와 같은, 합성적으로(synthetically) 산출된 입력을 또한 얻을 수 있고, 또는 인코더(110)는 합성(synthetic) 미디어의 코딩된 비트스트림들을 산출할 수 있다. 다음에서, 하나의 미디어 타입의 하나의 코딩된 미디어 비트스트림의 처리(processing)만이 설명(description)을 단순화하기 위해 고려된다. 그러나 일반적으로 실시간 방송(broadcast) 서비스가 여러 스트림을(일반적으로 적어도 하나의 오디오, 비디오 및 텍스트 자막 처리(sub-titling) 스트림) 포함하는 것이 주목되어야 한다. 시스템이 많은 인코더를 포함할 수 있지만, 다음에서 범용성을 잃지 않고 설명을 단순화시키기 위해 단지 하나의 인코더(110)만이 간주되는 것을 또한 주의되어야 한다.

<58> 코딩된 미디어 비트스트림은 저장소(120)로 이동된다. 저장소(120)는 코딩된 미디어 비트스트림을 저장하기 위한 임의 유형의 대용량 메모리(mass memory)를 포함할 수 있다. 저장소(120)에 있는 코딩된 미디어 비트스트림의 포맷은 엘리멘터리(elementary) 자급식(self-contained) 비트스트림 포맷일 수 있거나, 한 개 이상의 코딩된 미디어 비트스트림이 컨테이너 파일 안으로 캡슐화될 수 있다. 몇몇 시스템은 "라이브"로서 동작한다 - 즉 저장을 생략하고 인코더(110)로부터 송신기(130)로 직접적으로 코딩된 미디어 비트스트림을 전송한다. 그런 다음에 코딩된 미디어 비트스트림은 필요하다면, 서버로서 또한 언급된 송신기(130)로 이동된다. 전송에서 사용된 포맷이 엘리멘터리 자급식 비트스트림 포맷, 패킷 스트림 포맷일 수 있거나, 1개 이상의 코딩된 미디어 비트스트림들이 컨테이너 파일 안으로 캡슐화될 수 있다. 인코더(110), 저장소(120)와 송신기(130)는 동일한 물리적 기기에서 상주할 수 있거나 그것들은 분리된 기기들에 포함될 수 있다. 인코더(110)와 송신기(130)는 라이브 실시간 콘텐츠에 의해서 동작할 수 있고, 그런 경우에 코딩된 미디어 비트스트림은 처리(processing) 지연, 전달 지연과 코딩된 미디어 비트레이트에 있는 변동(variation)을 매끄럽게하기 위하여 콘텐츠 인코더(110)에서 그리고/또는 송신기(130)에서 영구히 저장되지 않는 것이 통상적이고, 오히려 작은 시간 구간들 동안 버퍼링된다. 코딩된 미디어 비트스트림의 다른 부분들용으로 다른 통신 프로토콜들을 사용하는 것이 또한 가능하다. 예를 들어, 파라미터 세트 NAL 유닛들은 세션 디스크립션 프로토콜 (SDP)를 사용하여 전달될 수 있고, 반면에 남아있는 데이터는 RTP를 사용하여 운반된다.

<59> 송신기(130)는 통신 프로토콜 스택을 사용하여 코딩된 미디어 비트스트림을 송신한다. 스택은 실시간 트랜스포트 프로토콜(RTP : Real-time Transport Protocol), 사용자 데이터그램 프로토콜(UDP : User Datagram

Protocol)과 인터넷 프로토콜(IP : Internet Protocol)을 포함할 수 있지만, 이에 제한되지 않는다. 통신 프로토콜 스택이 패킷 지향 방식(packet-oriented)일 때, 송신기(130)는 패킷들 안으로 코딩된 미디어 비트스트림을 캡슐화한다. 예를 들어, RTP가 사용될 때, 송신기(130)는 RTP 페이로드 포맷에 따라서 RTP 패킷들 안으로 코딩된 미디어 비트스트림을 캡슐화한다. 일반적으로, 각각의 미디어 유형은 전용 RTP 페이로드 포맷을 가진다. 시스템이 하나 이상의 송신기(130)를 포함할 수 있지만, 간이화 목적을 위해서 아래의 설명은 하나의 송신기(130)만을 고려하는 것을 주목해야 한다.

<60> 송신기(130)는 통신 네트워크를 통해 게이트웨이(140)에 연결될 수도 연결되지 않을 수도 있다. 게이트웨이(140)는 하나의 통신 프로토콜 스택으로부터 다른 하나의 통신 프로토콜 스택에 따라 패킷 스트림의 번역(translation), 데이터 스트림의 통합과 포킹(forking), 및 다운링크 그리고/또는 수신기 성능에 따라 데이터 스트림의 조작(manipulation)[예컨데 우세한(prevailing) 다운링크 네트워크 상태에 따라서 포워드된 스트림의 비트 레이트를 제어하는 것]과 같은 기능의 서로 다른 타입들을 수행할 수 있다. 게이트웨이(140)의 예가 MCUs(multipoint conference control unit), 회선-교환과 패킷-교환 비디오 텔레포니(telephony) 사이의 게이트웨이, PoC(Push-to-talk over Cellular) 서버들, DVB-H(digital video broadcasting-handheld) 시스템들에서의 IP 인캡슐레이터들, 또는 브로드캐스트 전송들을 로컬하게 홈 무선 네트워크들로 포워드하는 셋톱 박스들을 포함한다, RTP가 사용될 때, 게이트웨이(140)는 RTP 믹서로 지칭되고, RTP 커넥션의 엔드포인트로 동작한다.

<61> 시스템은, 코딩된 미디어 비트스트림 안으로 전송된 신호를 수신하고, 복조하고 캡슐화 취소(de-capsulating)할 수 있는 것이 일반적인 한 개 이상의 수신기(150)를 포함한다. 코딩된 미디어 비트스트림은 일반적으로 디코더(160)에 의해 추가 프로세스되며, 디코더(160)의 출력은 하나 이상의 압축해제된(uncompressed) 미디어 스트림이다. 디코딩될 비트스트림은 가상적으로 임의 유형의 네트워크 내에 위치된 원격 디바이스로부터 수신될 수 있다. 또한 비트스트림은 로컬 하드웨어 또는 소프트웨어로부터 수신될 수 있다. 마침내, 렌더러(renderer)(170)는 예를 들어, 확성기 또는 디스플레이를 사용하여 압축해제된 미디어 스트림을 재생할 수 있다. 수신기(150), 디코더(160)와, 및 렌더러(renderer)(170)는 동일한 물리 기기에 상주할 수 있거나 그것들은 분리된 기기들에 포함될 수 있다.

<62> 비트 레이트, 디코딩 복잡성과 픽쳐 크기에 관한 스케일러빌러티는 이질적 이고 오류 경향이 있는 환경을 위한 바람직한 속성이다. 이 속성은 수신 기기에서 비트 레이트, 디스플레이 해상도, 네트워크 효율(throughput), 및 연산 능력에 대한 제약과 같은 제한에 상대하기 위하여 바람직하다.

<63> 여기에 포함된 텍스트 및 예들이 특히 인코딩 프로세스를 설명하였지만, 이 기술분야에서 통상의 지식을 가진자는, 동일한 개념들 및 원리들이 대응하는 디코딩 프로세스에 또한 적용될 수 있고 그 반대도 가능하다는 것을 쉽게 알 것이다.디코딩될 비트스트림이 가상적으로 네트워크의 임의의 유형 내에 위치된 원격 디바이스로부터 수신될 수 있다. 또한 비트스트림이 로컬 하드웨어 또는 소프트웨어로부터 수신될 수 있다.

<64> 본 발명의 통신 디바이스들은 다양한 전송 기술들을 사용하여 통신할 수 있고, 이때 전송 기술은 CDMA(Code Division Multiple Access), GSM(Global System for Mobile Communications), UMTS(Universal Mobile Telecommunications System), TDMA(Time Division Multiple Access), FDMA(Frequency Division Multiple Access), TCP/IP(Transmission Control Protocol/Internet Protocol), SMA(Short Messaging Service), MMS(Multimedia Messaging Service), 이메일, IMS(Instant Messaging Service), 블루투스(Bluetooth), IEEE 802.11, 기타 등등을 포함하지만 이에 제한되지는 않는다. 통신 디바이스는 무선(radio), 적외선(infrared), 레이저, 케이블 연결 기타 등등을 사용하여 통신할 수 있지만 이에 제한되는 것은 아니다.

<65> 도 5 및 도 6은 본 발명이 구현될 수 있는 하나의 대표적인 이동 디바이스(12)를 도시한다. 그러나 본 발명은 이동 디바이스(12) 또는 다른 전자 디바이스의 하나의 특정 타입에 제한될 의도가 아님을 알아야 한다. 도 5 및 도 6에 도시된 특징들의 일부 또는 모두는 도 4에 표현된 기기들의 일부 또는 모두에 포함될 수 있다.

<66> 도 5 및 도 6의 이동 디바이스(12)는 하우징(30), LCD(liquid crystal display) 형태의 디스플레이(32), 키패드 (34), 마이크로 폰(36), 이어피스(ear-piece)(38), 배터리(40), 적외선 포트(42), 안테나(44), 본 발명의 하나의 실시 예에 따른 UICC 형태의 스마트 카드(46), 카드 리더(48), 무선 인터페이스 회로소자(52), 코덱 회로소자(54), 제어부(56), 및 메모리(58)를 포함한다. 개별 회로들 및 엘리먼트들들은 본 기술분야, 예를 들면 노키아 범위의 이동 전화들에서 모두 잘 알려진 형태이다.

<67> 본 발명은 멀티-뷰(multi-view) 비디오 코딩에서 효율적인 디코딩된 픽쳐 버퍼 관리를 구현하기 위한 개선된 시스템 및 방법을 제공한다. 현재 JMVM 신택스가 (시간 인덱스의 모든 뷰들이 앵커 픽쳐를 담고 있지 않는한) 일

정 뷰의 디코딩이 시작될 수 있는 픽쳐의 시그널링을 포함하지 않는다는 사실을 둘러싼 문제를 해결하기 위해서, 한 뷰가 일정 픽쳐로부터 액세스될 수 있는지 여부 -- 즉 일 예로서 한 뷰의 디코딩이 일정 픽쳐에서 시작될 수 있는지 여부-- 를 표시하기 위해 새로운 플래그가 시그널링된다. 본 발명의 하나의 실시 예에서, 이 플래그는 NAL(network abstraction layer) 유닛 헤더 내에서 시그널링된다. 다음은 하나의 특정 실시 예에 따른 플래그의 신택스 및 시맨틱스의 하나의 예이다. 그러나 새로운 신택스 엘리먼트를 추가하는 대신에 신택스 엘리먼트 anchor_pic_lag의 시맨틱스를 유사하게 변경하는 것이 또한 가능하다.

nal_unit_header_svc_mvc_extension( ) {	C	Descriptor
svc_mvc_flag	All	u(1)
if (!svc_mvc_flag) {		
priority_id	All	u(6)
discardable_flag	All	u(1)
temporal_level	All	u(3)
dependency_id	All	u(3)
quality_level	All	u(2)
layer_base_flag	All	u(1)
use_base_prediction_flag	All	u(1)
fragmented_flag	All	u(1)
last_fragment_flag	All	u(1)
fragment_order	All	u(2)
reserved_zero_two_bits	All	u(2)
} else {		
view_refresh_flag	All	u(1)
view_subset_id	All	u(2)
view_level	All	u(3)
anchor_pic_flag	All	u(1)
view_id	All	u(10)
reserved_zero_five_bits	All	u(6)
}		
nalUnitHeaderBytes += 3		
}		

<68>

<69> 한 뷰 내의 일정 픽쳐에 대해서, 인터-뷰 예측을 위해 사용되는 다른 뷰들로부터 동일한 템포럴 위치에 있는 픽쳐들의 모두는 "직접적 의존형 뷰 픽쳐들(directly depend-on view pictures)"로 지칭되고, 현재 픽쳐의 디코딩을 위해 요구되는 다른 뷰들로부터 동일한 템포럴 위치에 있는 모든 픽쳐들은 "의존형 뷰 픽쳐들(the depend-on view pictures)"로서 지칭된다.

<70> view_refresh_flag의 시맨틱스는 하나의 실시 예에서 4가지 방식으로 상술된다. view_refresh_flag 의 시맨틱스를 상술하는 첫 번째 방법은, 동일 뷰 또는 다른 뷰들 내의 임의의 이전(preceding) 픽쳐를 디코딩함이 없이 동일 뷰 내의 출력 순서로 현재 픽쳐 및 모든 연속 픽쳐들의 직접적 의존형 뷰 픽쳐들 모두가 또한 (가능하게는 부분적으로) 디코딩될 때, 동일한 뷰 내의 출력 순서로 현재 픽쳐 및 모든 연속 픽쳐들이 정확하게 디코딩될 수 있다고 view_refresh_flag가 표시하도록 하는 것을 포함한다. 이것은 (1) 의존형 뷰 픽쳐들의 어느 것도 임의의 뷰 내의 디코딩 순서로 임의의 이전 픽쳐에 의존하지 않거나, 또는 (2) 의존형 픽쳐들 중 어느 것이 임의의 뷰 내의 디코딩 순서로 임의의 이전 픽쳐들에 의존한다면, 동일한 뷰 내의 현재 픽쳐 및 모든 연속 픽쳐들의 직접적 의존형 뷰 픽쳐들의 강제적으로(constrainedly) 인트라 코딩된 영역들만이 인터-뷰 예측을 위해 사용된다는 것을 의미한다. 강제적으로(constrainedly) 인트라 코딩된 영역은, 인트라 예측을 위해 인터-코딩된 이웃 영역들로부터의 어떤 데이터도 사용하지 않는다.

<71> view_refresh_flag 의 시맨틱스를 상술하는 두 번째 방법은, 동일 뷰 내의 현재 픽쳐 및 연속 픽쳐들의 모든 직접적 의존형 뷰가, 임의의 이전 픽쳐를 디코딩함이 없이 완전히, 또는 일실시 예에서 부분적으로 디코딩되는 때, 동일 뷰 내의 *디코딩 순서로* 현재 픽쳐 및 모든 연속 픽쳐들이 정확하게 디코딩될 수 있다는 것을 view_refresh_flag 가 표시하도록 하는 것을 포함한다.

<72> view_refresh_flag의 시맨틱스를 상술하는 세 번째 방법은, 동일 뷰 내의 현재 픽쳐 및 연속 픽쳐들의 모든 의

존형 뷰 픽쳐들이 완전히, 또는 일실시 예에서 부분적으로 디코딩되는 때, 동일 뷰 내의 *출력 순서*로 현재 픽쳐 및 모든 연속 픽쳐들이 정확하게 디코딩될 수 있다는 것을 view_refresh_flag가 표시하도록 하는 것을 포함한다. 이 정의(definition)는 단일-뷰 코딩에서 열린(open) GOP(group of pictures)를 시작하는 인트라 픽쳐와 유사하다. 명세서 본문의 표현으로 이 옵션은 다음과 같이 쓰여질 수 있다 : 1과 같은 view_refresh_flag는, 출력 순서로 현재 픽쳐 및 다음 픽쳐와 같은 뷰 내의 디코딩 순서로 현재 픽쳐 및 임의의 연속 픽쳐가, 인터-예측 프로세스에서 디코딩 순서로 현재 픽쳐 이전의 픽쳐를 참조하지 않는다는 것을 표시한다. 0과 같은 view_refresh_flag는, 출력 순서로 현재 픽쳐 및 다음 픽쳐와 같은 뷰 내의 디코딩 순서로 현재 픽쳐 및 임의의 연속 픽쳐가, 인터-예측 프로세스에서 디코딩 순서로 현재 픽쳐 이전의 픽쳐를 참조할 수 있다는 것을 표시한다.

<73> view_refresh_flag의 시맨틱스를 상술하는 네 번째 방법은, 동일 뷰 내의 현재 픽쳐 및 연속 픽쳐들의 모든 의존형 뷰 픽쳐들이 완전히, 또는 일실시 예에서 부분적으로 디코딩되는 때, 동일 뷰 내의 *디코딩 순서*로 현재 픽쳐 및 모든 연속 픽쳐들이 정확하게 디코딩될 수 있다는 것을 view_refresh_flag가 표시하도록 하는 것을 포함한다. 이 정의(definition)는 단일-뷰 코딩에서 닫힌(closed) GOP(group of pictures)를 시작하는 인트라 픽쳐와 유사하다.

<74> view_refresh_flag는 도 4에 도시된 시스템과 같은 시스템에서 사용될 수 있다. 이 상황에서, 수신기(150)가 모든 가용 N개의 뷰들 중 뷰 A를 제외한 일정 서브셋 M만을 수신했거나, 디코더가(160) 디코딩했다. 유저 액션 때문에, 예컨대 수신기(150) 또는 디코더(160)가 앞으로는 뷰 A를 각각 수신하거나 디코더하고자 한다. 디코더는, 뷰 A 내에서 1과 같은 view_refresh_flag를 갖는 제1 픽쳐로부터 뷰 A의 디코딩을 시작할 수 있다. 뷰 A가 수신되지 않았다면, 수신기(150)는 게이트웨이(140) 또는 송신기(130)에게 뷰 A의 코딩된 픽쳐들을 전송된 비트스트림으로 삽입하라고 표시한다. 게이트웨이(140) 또는 송신기(130)는, 디코더(160)가 성공적으로 디코딩할 수 없는 뷰 A의 불필요한 픽쳐들의 송신을 회피하기 위해 뷰 A의 임의의 픽쳐들을 송신하기 전에 뷰 A 내에서 1과 같은 view_refresh_flag를 갖는 다음 픽쳐들 기다릴 수 있다.

<75> 이전에 논의된 두 번째 이슈를 해결하기 위해서, 한 뷰가 인터-뷰 예측 레퍼런스를 위해 사용되는지 여부를 표시하기 위해 새로운 플래그가 시그널링되고, 신택스 엘리먼트 nal_ref_idc 만이, 한 픽쳐가 템포럴 예측 레퍼런스를 위해 사용되는지 여부를 표시한다. 하나의 특정 실시 예에서, 이 플래그가 NAL 유닛 헤더 내에서 시그널링된다.

<76> 다음은 플래그의 신택스 및 시맨틱스의 한 예이다.

<77>

nal_unit_header_svc_mvc_extension( ) {	C	Descriptor
svc_mvc_flag	All	u(1)
if (!svc_mvc_flag) {		
priority_id	All	u(6)
discardable_flag	All	u(1)
temporal_level	All	u(3)
dependency_id	All	u(3)
quality_level	All	u(2)
layer_base_flag	All	u(1)
use_base_prediction_flag	All	u(1)
fragmented_flag	All	u(1)
last_fragment_flag	All	u(1)
fragment_order	All	u(2)
reserved_zero_two_bits	All	u(2)
} else {		
inter_view_reference_flag	All	u(1)
view_subset_id	All	u(2)
view_level	All	u(3)
anchor_pic_flag	All	u(1)
view_id	All	u(10)
reserved_zero_five_bits	All	u(5)
}		
nalUnitHeaderBytes += 3		
}		

<78> 0과 같은 inter_view_reference_flag 가, 현재 픽쳐가 인터-뷰 레퍼런스 픽쳐로서 사용되지 않는다고 표시한다. 1과 같은 inter_view_reference_flag가, 현재 픽쳐가 인터-뷰 레퍼런스 픽쳐로서 사용된다고 표시한다. profile_idc가 MVC profile 및 view_id가 0이라고 표시할 때, inter_view_reference_flag의 값이 1과 같다고 추론된다. 픽쳐를 디코딩할 때, 현재 픽쳐와 동일한 템포럴 축을 갖고 1과 같은 inter_view_reference_flag를 갖는 모든 픽쳐들이 현재 픽쳐의 인터-뷰 픽쳐들로서 지칭된다.

<79> inter_view_reference_flag는 MANE(media-aware network element)으로서 또한 지칭되는 게이트웨이(140)에서 사용될 수 있다. 픽쳐가 인터-뷰 레퍼런스 및 인트라-뷰 레퍼런스로서 사용되지 않을 때(inter_view_reference_fiag가 0과 같고 nal_ref_idc 가 0과 같음), MANE은 잔여 비트스트림의 디코딩에서 영향을 미치지 않고 그 픽쳐를 포워드하지 않기로 선택할 수 있다. 픽쳐가 인터-뷰 레퍼런스로서 사용되지 않지만 인트라-뷰 레퍼런스로서 사용될 때, MANE은 종속(dependent) 뷰들의 전송을 또한 드롭하는 경우에만 픽쳐를 드롭해야 한다. 픽쳐가 인터-뷰 레퍼런스로서 사용되지 않지만 인트라-뷰 레퍼런스로서 사용될 때, MANE은 그 픽쳐가 존재하는 뷰를 디코딩하기는 것이 요구되지 않거나 바람직하지 않을 때만 픽쳐를 드롭해야 한다.

<80> 인터-뷰 예측을 위해 버퍼되어야 하는 디코딩되는 픽쳐들의 관리를 효율적으로 처리할 수 없는, JMVM 1.0 에 상술된 레퍼런스 픽쳐 마킹 프로세스의 문제에 관하여, 플래그 inter_view_reference_flag가 재사용된다. 1과 같은 inter_view_reference_flag 를 지닌 픽쳐들이 3가지 방법들 중 일부를 사용하여 마크될 수 있다.

<81> 1과 같은 inter_view_reference_flag로 픽쳐들을 마킹하는 첫 번째 방법은, 장기 픽쳐들로서 인터-뷰 레퍼런스 픽쳐들을 일시적으로(temporally) 저장하는 것을 포함한다. 인코딩 프로세스에서, 인터-뷰 예측을 위해 사용되는 각각의 픽쳐는 비트스트림 내에서 "장기 레퍼런스를 위해 사용됨"으로 마크되는 것으로 표시된다. "장기 레퍼런스를 위해 사용됨"으로 마킹을 나타내는 한 방법은 inter_view_reference_flag이다. "장기 레퍼런스를 위해 사용됨" 및 "일시적인(temporary) 멀티-뷰 장기 레퍼런스"로서 픽쳐를 마크한 표시(indication)에 디코더가 응답한다. "장기 레퍼런스를 위해 사용됨" 및 "일시적인(temporary) 멀티-뷰 장기 레퍼런스"로서 마크된 픽쳐를 대상으로 하는 임의의 메모리 제어 오퍼레이션은 일시적으로 버퍼된다. 템포럴 축 내의 모든 픽쳐들이 인코딩되거나 디코딩될 때, "장기 레퍼런스를 위해 사용됨" 및 "일시적인(temporary) 멀티-뷰 장기 레퍼런스"로서 마크된 모든 픽쳐들은, 더 이상 "장기 레퍼런스를 위해 사용됨" 및 "시간적인(temporary) 멀티-뷰 장기 레퍼런스"로서 마크되지 않고, 슬라이딩 윈도우 오퍼레이션 또는 버퍼되는 메모리 관리 제어 오퍼레이션 중 하나(어느 것이든 특정 픽쳐에 적용가능한 것)를 사용해서 그 픽쳐들에 대하여 그 디코딩 순서로 레퍼런스 픽쳐 마킹이 다시 행해진다. 예컨대 픽쳐가 인터 예측을 위해 사용된다면(즉 nal_ref_idc의 값이 0보다 큼), 그 픽쳐는 "장기 레퍼런스를 위해 사용됨"으로 다시 마크된다. 픽쳐가 인터 예측을 위해 사용되지 않는다면(즉 nal_ref_idc의 값이 0과 같음), 그 픽쳐는 "장기 레퍼런스를 위해 사용되지 않음"으로 마크된다. 일정 템포럴 축 내의 픽쳐에 대해 2개의 경우만이 있는 것이 보통이다 : 모든 픽쳐들이 인터 예측을 위한 레퍼런스 픽쳐들이거나, 어떤 픽쳐들도 인터 예측을 위한 레퍼런스 픽쳐들이 아님. 템포럴 축 내의 최종 VCL NAL 유닛이 디코딩된 이후에, 또는 연속 템포럴 축 내의 다음 액세스 유닛 또는 다음 픽쳐가 디코딩되기 이전에, 최종 오퍼레이션이 수행될 수 있다. 디코딩 프로세스에서, 이 단계의 그 오퍼레이션은 템포럴 축의 변화에 의해 암시적으로 트리거될 수 있거나, 예컨대 MMCO 명령과 같이 명시적으로 시그널링될 수 있다. 이와 같은 방법으로, 시간적인 다이렉트 모드(temporal direct mode)에서 그리고 가중된 예측(weighted prediction)을 위해서 인터-뷰 레퍼런스 픽쳐들이 장기 레퍼런스 픽쳐들과 동일한 영향력을 갖는다.

<82> 1과 같은 inter_view_reference_flag로 픽쳐들을 마킹하는 두 번째 방법은, 인터-뷰 레퍼런스 픽쳐들을 "인터-뷰 레퍼런스를 위해 사용됨"으로 마킹하는 것을 포함한다. 이 방법을 써서, 인터 예측용 레퍼런스 픽쳐 마킹("단기 레퍼런스를 위해 사용됨" 및 "장기 레퍼런스를 위해 사용됨"으로 마킹)은 AVC 표준과 비교시 변하지 않았다. 시간적인 다이렉트 모드(temporal direct mode) 및 가중된 예측에 관련된 프로세스들을 위해, "인터-뷰 레퍼런스를 위해 사용됨"으로 마킹되는 픽쳐들--현재 픽쳐와 동일한 템포럴 축을 공유하는 인터-뷰 레퍼런스 픽쳐들--은 장기 레퍼런스 픽쳐들과 동일하게 다뤄진다. 템포럴 축 내의 모든 픽쳐들이 인코딩되거나 디코딩될 때, "인터-뷰 레퍼런스를 위해 사용됨"으로 마크된 모든 픽쳐들은, 더 이상 "인터-뷰 레퍼런스를 위해 사용됨"으로 마크되지 않는다.

<83> 템포럴 축 내의 모든 픽쳐들이 처리된 이후에 "인터-뷰 레퍼런스를 위해 사용됨" 마킹의 제거가 본 발명의 일 예인 것을 주목해야 한다. "인터-뷰 레퍼런스를 위해 사용됨"과 같은 마킹은 디코딩 프로세스의 다른 인스턴트들에서 또한 제거될 수 있다. 예컨대 특정 픽쳐의 "인터-뷰 레퍼런스를 위해 사용됨"과 같은 마킹은, 현재 픽쳐 또는 연속 픽쳐가, SPS의 MVC 익스텐션에 포함된 뷰 의존성(dependency) 시그널링에 따라 픽쳐에 직접적으로 또

는 간접적으로 더 이상 의존하지 않자마자 제거될 수 있다.

<84> 템포럴 축 내의 최종 VCL NAL 유닛이 디코딩되거나 연속 템포럴 축 내의 다음 액세스 유닛 또는 다음 픽쳐가 디코딩된 이후에, 적절한 픽쳐들이 "인터-뷰 레퍼런스를 위해 사용됨"으로 더 이상 마크되지 않게 하는 오퍼레이션이 행해질 수 이다. 디코딩 프로세스에서, 이것은 템포럴 축의 변화에 의해 암시적으로 트리거될 수 있거나, 예컨대 MMCO 명령과 같이 명시적으로 시그널링될 수 있다.

<85> 이 특정 방법에서, 시간적인 다이렉트 모드(temporal direct mode) 내에서 그리고 가중된 예측(weighted prediction)에 대해 인터-뷰 레퍼런스 픽쳐들이 장기 레퍼런스 픽쳐들과 동일한 영향력을 갖는다. 환언하면, 이 방법은, 시간적인 다이렉트 모드(temporal direct mode)에서 그리고 가중된 예측(weighted prediction)에 대해서, 위에서 논의된 첫 번째 방법과 동일한 효과를 가진다.

<86> 이 방법에서 인터-뷰 예측만을 위해 사용되는 픽쳐들 --즉 0과 같은 nal_ref_idc를 갖고 "인터-뷰 레퍼런스를 위해 사용됨"으로 마크된 픽쳐들--의 "인터-뷰 레퍼런스를 위해 사용됨" 마킹을 제거하기 위해 개선된 슬라이딩 메커니즘이 적용될 수 있다. 이 개선된 슬라이딩 메커니즘은, "인터-뷰 레퍼런스를 위해 사용됨"으로 마크되고 0과 같은 nal_ref_idc를 갖는 픽쳐들의 개수가 num_inter_view_ref_frames과 같을 때, 최초 디코딩된 픽쳐가 "인터-뷰 레퍼런스를 위해 사용됨"으로 마크되지 않도록,예컨대 MVC용 SPS 익스텐션에서 시그널링되는 것이 바람직한, num_inter_view_ref_frames로 지칭되는 변수를 사용한다. 결론적으로 픽쳐가 출력을 위해 필요하지 않거나, 계획적으로 출력하지 않기로 하면, 디코더는 새롭게 디코딩된 픽쳐가 DPB에 저장될 수 있도록 DPB로부터 픽쳐를 제거하는 프로세스를 불러일으킨다.

<87> 1과 같은 inter_view_reference_flag로 픽쳐들을 마킹하는 세 번째 방법은, 동일 축/시간 인덱스의 모든 픽쳐들을 디코딩한 후에 픽쳐들을 마킹하는 것을 포함한다. 픽쳐의 디코딩 이후에 즉각적으로 픽쳐를 마킹하는 대신에, 이 방법은 동일 템포럴 축(즉 동일 시간 인덱스)의 모든 픽쳐들의 디코딩 이후에 픽쳐들이 마킹되는 아이디어에 기초한다. 코딩되는 픽쳐들의 각각에서 나타난 적응식 레퍼런스 픽쳐 마킹 또는 슬라이딩 윈도우는, 픽쳐들이 디코딩되는 순서로 수행된다. 시간적인 다이렉트 모드(temporal direct mode)와 가중된 예측에 관련된 프로세스들에 대해서, 현재 픽쳐와 동일한 템포럴 축의 마킹 픽쳐들이, 장기 레퍼런스 픽쳐들과 동일하게 다뤄진다. 현재 픽쳐와 동일한 템포럴 축의 인터-뷰 레퍼런스 픽쳐들이, 초기 레퍼런스 픽쳐 리스트 구성 내에 포함되고, 그 픽쳐들의 view_id 에 기초해서 재배열되거나 장기 레퍼런스 인덱스들을 우선 할당받고 그 다음에 장기 레퍼런스 인덱스에 기초해서 재맵핑될 수 있다.

<88> 이전에 논의된 것과 같이 PicNum를 재계산하는 방법이 주어질 때, 슬라이딩 윈도우 오퍼레이션 모드가 사용되고 있고 단기 및 장기 픽쳐들의 개수가 최대와 같을 때, 최소 FrameNumWrap를 갖는 단기 레퍼런스 픽쳐가 "레퍼런스를 위해 사용되지 않음"으로 마킹된다. 그러나 현재 JMVM 내의 FrameNum 순서가 디코딩 순서를 따르지 않기 때문에 이 픽쳐가 꼭 최초 코딩된 픽쳐일 필요는 없다는 사실 때문에, 슬라이딩 윈도우 레퍼런스 픽쳐 마킹은 현재 JMVM 내에서 최적으로 동작하지 않는다. 이 문제를 해결하기 위해서, JMVM 표준과 비교해 볼 때 변수들 FrameNum 및 FrameNumWrap 이 재정의/스케일되지 않는다 - 즉 변수들의 정의가 AVC 표준과 비교해서 변하지 않은채로 있다. 단기 픽쳐들이 슬라이딩 윈도우의 선입, 선출 메커니즘에의해 자동적으로 다뤄질 수 있게 설계되어있다. JMVM 1.0과 비교해서 슬라이딩 윈도우 메커니즘의 오직 미약한 수정만이 요구된다. 이탤릭체로 표현된 새로운 원문으로된 수정들은 다음과 같다. :

<89> G. 8.2.5.3 슬라이딩 윈도우의 디코딩되는 레퍼런스 픽쳐 마킹 프로세스.

<90> 이 프로세스는 adaptive_ref_pic_marking_mode_flag 가 0과 같을 때 호출된다.

<91> *현재 슬라이스와 동일한 view_id를 갖는 레퍼런스 픽쳐들만이, numShortTerm 과 numLongTerm, 및 적용되는 num_ref_frames의 계산을 포함하는, 프로세스에서 고려된다.*

<92> 위의 방법에서, 전체 MVC 비트스트림의 레퍼런스 프레임들의 총 개수 [전체 MVC 비트스트림의 인트라-뷰 또는 인터-뷰 레퍼런스를 위해 사용되는 픽쳐들의 저장을 위한 버퍼 크기] 가, MVC 비트스트림을 디코딩하기 위한 인터-뷰 레퍼런스 프레임들의 최대 개수에 MVC 비트스트림에 담긴 모든 뷰들에 적용되는 num_ref_frame 값들의 합과 동일해야 한다. 대안적으로 모든 뷰들 내의 픽쳐들 모두에 대해 글로벌하게 슬라이딩 윈도우가 수행될 수 있다.

<93> 시간-우선 코딩의 경우에, 슬라이딩 윈도우 프로세스가, 이탤릭체로 표현된 JMVM 1.0 에 대한 새 원문으로써 아래에 정의된다.

<94> G. 8.2.5.3 슬라이딩 윈도우의 디코딩된 레퍼런스 픽쳐 마킹 프로세스.

<95> ...

<96> ...

<97> numShortTerm + numLongTerm이 Max(num_ref_frames, 1)와 같을 때, numShortTerm이 0보다 큰 조건이 충족될 것이고, *다음 규칙에 의해 선택된* 단기 레퍼런스 프레임, 보충(complementary) 레퍼런스 필드 쌍 또는 쌍이아닌(non-paired) 레퍼런스 필드가 "레퍼런스를 위해 사용되지 않음"으로 마크된다. 그것이 프레임 또는 보충 필드 쌍일 때, 그 필드 두 개 모두가 "레퍼런스를 위해 사용되지 않음"으로 또한 마크된다.

<98> *"선택 규칙 : FrameNumWrap의 최소 값을 갖는 모든 픽쳐들로부터, 디코딩 순서로 첫 번째 픽쳐가 선택된다. 그 픽쳐들의 디코딩 순서는 view_id 값에 의해, 또는 SPS fro MVC 익스텐션에서 시그널링되는 뷰 의존성 정보(*view dependency information *)에 의해 표시될 수 있다.*

<99> 시간-우선 코딩의 경우에, 슬라이딩 윈도우 프로세스가, 이탤릭체로 표현된 JMVM 1.0 에 대한 새 원문으로써 아래에 정의된다.

<100> G. 8.2.5.3 슬라이딩 윈도우의 디코딩된 레퍼런스 픽쳐 마킹 프로세스.

<101> ...

<102> ...

<103> numShortTerm + numLongTerm이 Max(num_ref_frames, 1)와 같을 때, numShortTerm이 0보다 큰 조건이 충족될 것이고, *다음 규칙에 의해 선택된* 단기 레퍼런스 프레임, 보충(complementary) 레퍼런스 필드 쌍 또는 쌍이 아닌 레퍼런스 필드가 "레퍼런스를 위해 사용되지 않음"으로 마크된다. 그것이 프레임 또는 보충 필드 쌍일 때, 그 필드 두 개 모두가 "레퍼런스를 위해 사용되지 않음"으로 또한 마크된다.

<104> *"선택 규칙 : 가장 이르게 디코딩되는 모든 픽쳐들로부터, 최소 FrameNumWrap를 갖는 하나의 픽쳐가 선택된다. 그 뷰의 디코딩 순서는 view_id 값에 의해, 또는 SPS fro MVC 익스텐션에서 시그널링되는 뷰 의존성 정보(*view dependency information*)에 의해 표시될 수 있다.*

<105> 이전에 논의되는 것과 같이, PicNum가 재정의되고 스케일된 FrameNumWrap로부터 유도된다는 사실 때문에, 2개의 코드된 픽쳐들의 PicNum 값들의 차이가 평균으로 스케일링될 것이다. 예컨대 동일한 뷰 내에 그리고 각각 3과 5와 같은 frame_num를 갖는 2개의 픽쳐들이 존재하는다는 것을 가정하는 것이 도움이 될 것이다. 오직 하나의 뷰만이 존재할 때 --즉 비트스트림이 AVC 스트림일 때--, 2개의 PicNum 값들의 차이가 2일 것이다. 5와 같은 frame_num를 갖는 픽쳐를 코딩할 때, MMCO 명령어가 3과 같은 PicNum를 갖는 픽쳐를 " 레퍼런스를 위해 사용되지 않음"으로 마크할 필요가 있다면, 2개의 값들의 차에서 1을 뺀 것은 1과 같을 것이고, 이것이 MMCO에서 시그널링된다. 이 값은 3 비트가 필요하다. 그러나 256개의 뷰들이 존재한다면, 2개의 PicNum 값들의 차이에서 1을 뺀 것은 511이 될 것이다. 이 경우에, 19 비트가 그 값을 시그널링하기 위해 필요하다. 결론적으로 MMCO 명령어는 훨씬 덜 효율적으로 코딩된다. 전형적으로 비트들의 증가된 개수는, H.264/ AVC의 단일-뷰 코딩과 비교했을 때 현재 JMVM의 MMCO 명령어용으로 2*log2(뷰들의 개수)와 같다.

<106> 이 문제를 해결하기 위해서 JMVM 표준과 대조적으로, 변수들 FrameNum 및 FrameNumWrap이 재정의되고/스케일링되지 않고, AVC 표준에서와 같다. 대부분의 경우들에서, 픽쳐가, 현재 픽쳐와 동일한 뷰나 동일한 템포럴 축 중 어느 것에도 속하지 않는 픽쳐를 제거하도록 하는 MMCO 명령어를 담고 있는 것이 DPB 크기 입장에서 요구되지 않는다. 오히려 그 픽쳐들 중 일부가 레퍼런스를 위해 더 이상 필요로되지 않고, 따라서 "레퍼런스를 위해 사용되지 않음"으로 마크될 수 있다. 이 경우에, 마킹은 슬라이딩 윈도우 프로세스를 사용해서 수행될 수 있거나, 동일한 view_id를 갖는 다음 코딩된 픽쳐까지 지연될 수 있다. 따라서 DPB(decoded picture buffer)가 다른 뷰들 또는 다른 템포럴 축의 픽쳐들을 담고 있을 수 있더라도, MMCO 명령어들은 동일한 뷰 또는 동일한 템포럴 축에 속하는 픽쳐들에 대해서 "레퍼런스를 위해 사용되지 않음"으로 픽쳐들을 마크할 제약만을 받는다.

<107> 인트라-뷰 레퍼런스 픽쳐 마킹을 위한 JMVM 1.0의 수정은 다음과 같고, 수정사항은 이탤릭체로 보여진다.

<108> G.8.2.5.4.1 단기 레퍼런스 픽쳐를 "레퍼런스를 위해 사용되지 않음"으로 마킹하는 프로세스

<109> adaptive_ref_pic_marking_mode_flag가 1과 동일할 때 이 프로세스가 호출된다.

<110> *현재 슬라이스와 동일한 view_id를 갖는 레퍼런스 픽쳐만이 프로세스에서 고려된다.*

<111> 인터-뷰 레퍼런스 픽쳐 마킹을 위한 신택스 및 시맨틱스가 다음과 같을 수 있다:

slice_header( ) {	C	Descriptor
...		
if(nal_ref_idc != 0)		
dec_ref_pic_marking( )	2	
if(inter_view_reference_flag)		
dec_view_ref_pic_marking_mvc( )	2	
}		

<112>

dec_view_ref_pic_marking_mvc( ) {	C	Descriptor
adaptive_view _ref_pic_marking_mode_flag	2	u(1)
if(adaptive_view _ref_pic_marking_mode_flag)		
do {		
view_memory_management_control_operation	2	ue(v)
if(view_memory_management_control_operation == 1 || view_memory_management_control_operation == 2)		
abs_difference_of_view_id_minus1	2	ue(v)
} while(view_memory_management_control_operation != 0)		
}		
}		

<113>

<114> 메모리 관리 제어 오퍼레이션(view_tnemory_management_control_operation) 값들은 다음과 같다.

view_memory_management_control_operation	Memory Management Control Operation
0	End view memory_management_control_operation loop
1	Remove the marking of "used for inter-view reference" or mark a picture as "unused for reference", abs_difference_of_view_id_minus1 is present and corresponds to a difference to subtract from the current view id
2	Remove the marking of "used for inter-view reference" or mark a picture as "unused for reference", abs_difference_of_view_id_minus1 is present and corresponds to a difference to add to the current view id

<115>

<116> adaptive_view_ref_pic_marking_mode_flag는 슬라이딩 윈도우 메커니즘(0과 동일할 때) 또는 적응식 레퍼런스 픽쳐 마킹 프로세스(1과 동일할 때)가 사용되고 있는지 여부를 상술한다.

<117> 인터-뷰 레퍼런스 픽쳐 마킹을 위해 수정된 디코딩 프로세스는 다음과 같다.

<118> 8.2.5.5.2 인터-뷰 픽쳐들의 마킹

<119> view_memory_management_control_operation이 1과 같을 때 이 프로세스가 호출된다.

<120> viewIDX가 다음과 같이 상술된다고 하자.

<121> if(view_memory_management_control_operation==1)

<122> viewIDX = CurrViewId - (difference_of_view_id_minusl + 1)

<123> else if(view_memory_management_control_operation==2)

<124> viewIDX = CurrViewId + (difference_of_view_id_minusl + 1)

<125> 뷰 스케일러빌러티--즉 어떤 뷰들이 전송, 포워드, 또는 디코딩되는지를 선택의 가능성(possibility)--를 허용하기 위해서, 메모리 관리 제어 오퍼레이션들이 다음과 같이 제약될 수 있다. currTemporal Level가 현재 픽처의 temporal_level과 같고 dependentViews이 현재뷰에 의존하는 뷰들의 세트라면, MMCO 명령어가 currTemporalLevel과 같거나 더 큰 temporal_level를 갖고 dependentView들 범위 내에 있는 픽처만을 목표로 할 수 있다. 이것을 허용하기 위해서, MMCO 명령어들은 view_id 의 표시와 함께 첨부되거나, view_id의 표시를 갖는 새로운 MMCO 명령어들이 특정된다.

<126> 이전에 설명된 레퍼런스 픽처 리스트 구성 프로세스와 관련된 문제들을 해결하기 위해서, 변수들 FrameNum 및 FrameNumWrap가 재정의되고/스케일링되지 않는다. 이것은 AVC 표준에서 발생하는 것과 동일한 동작이고 JMVM 표준에 대조적이고, 여기서 변수들이 재정의되고/재스케일링된다. JMVM 1.0의 수정은 아래와 같고, 이탤릭체로 수정들이 표시된다:

<127> 8.2.4.3.1 단기 레퍼런스 픽처들을 위한 레퍼런스 픽처 리스트들의 재배열 프로세스에서, 8-38은 다음과 같이 변경될 것이다. :

<128> for(cIdx = num_ref_idx_lX_active_minusl + 1 ; cIdx > refIdxLX; cIdx--)

<129> RefPicListX[cIdx] = RefPicListX[cIdx - 1]

<130> RefPicListX[refIdxLX++] = short-term reference picture with PicNum equal to picNumLX and view_id equal to CurrViewID

<131> nIdx = refIdxLX

<132> for(cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minusl + 1 ; cIdx++) (8-38)

<133> *//if(PicNumF RefPicListX[cIdx]) != picNumLX)*

<134> *if(PicNumF(RefPicListX[cIdx]) != picNumLX ‖ ViewID(RefPicListX[cIdx]) != CurrViewID)*

<135> RefPicListX[nIdx++] = RefPicListX[cIdx]

<136> CurrViewID는 현재 디코딩 픽처의 view_id이다.

<137> 이전에 논의된 레퍼런스 픽처 리스트 초기화 프로세스와 연관된 문제들과 관련하여, 현재 슬라이스와 동일한 뷰에 속하는 프레임들, 필드들, 또는 필드 쌍들만이 초기화 프로세스에서 고려될 수 있다는 것을 주목함으로써 해결될 수 있다. JMVM 1.0의 용어로, 이 언어는 하위 조항들 8.2.4.2.1 "Initialisation process for the reference picture list for P and SP slices in frames" 내지 8.2.4.2.5 "Initialisation process for reference picture lists in fields."의 각각의 시작에 추가될 수 있다.

<138> 레퍼런스 픽처 리스트 구성 프로세스와 관련된 다른 문제들과 관련하여, 다수의 문제들이 인터-뷰 픽처들 및 인트라-예측에 사용되는 픽처들 모두를 효율적으로 재배열하기 위해 사용될 수 있다. 그러한 첫 번째 방법은, 인터-뷰 픽처들 및 인트라-뷰 예측을 위한 픽처들을 위해 별개의 RPLR 프로세스들을 특정하는 것뿐만 아니라, 리스트 내의 인트라-뷰 레퍼런스 픽처들의 앞에 인터-뷰 레퍼런스 픽처들을 놓는 것을 포함한다. 인트라-뷰 예측을 위해 사용되는 픽처들은 인트라-뷰 픽처들로서 또한 나타난다. 이 방법에서, 위에서 설명된 것과 같이 인트라-뷰 픽처들용 레퍼런스 픽처 리스트 초기화 프로세스가 수행되고, 그 다음에 RPLR 재배열 프로세스 및 인트라-뷰 픽처들 에 대한 리스트 절단 프로세스가 따라온다. 다음에, 인터-뷰 픽처들은 인트라-뷰 픽처들 이후에 리스트에 첨부된다. 마지막으로, 각각의 인터-뷰 픽처가 추가로 선택될 수 있고 JMVM 1.0로부터 수정된, 다음의 신택스, 시맨틱스, 및 디코딩 프로세스를 사용해서 레퍼런스 픽처 리스트의 특정된 엔트리에 놓일 수 있다. 방법은 refPicList0 및 refPiclist1가 존재한다면 그 모두에 적용가능할 수 있다.

ref_pic_list_reordering( ) {	C	Descriptor
if(slice_type != I && slice_type != SI) {		
...		
}		
if (svc_mvc_flag)		
{		
view_ref_pic_list_reordering_flag_l0	2	u(1)
if(view_ref_pic_list_reordering_flag_l0)		
do {		
view_reordering_idc	2	ue(v)
if(view_reordering_idc == 0 || view_reordering_idc == 1)		
abs_diff_view_idx_minus1	**2**	**ue(v)**
ref_idx	**2**	**ue(v)**
} while(view_reordering_idc!= 2)		
view_ref_pic_list_reordering_flag_l1	2	u(1)
if(view_ref_pic_list_reordering_flag_l1)		
do {		
view_reordering_idc	2	ue(v)
if(view_reordering_idc == 0 || view_reordering_idc == 1)		
abs_diff_view_idx_minus1	**2**	**ue(v)**
ref_idx	**2**	**ue(v)**
} while(view_reordering_idc != 2)		
}		

<139>

<140> 신택스와 관련해서, 1과 같은 view_ref_pic_list_reordering_flag_lX (X는 0 또는 1)이, 신택스 엘리먼트 view_reordering_idc 가 refPicListX를 위해 있다는 것을 상술한다. 0과 같은 view_ref_pic_list_reordering_flag_lX (X는 0 또는 1)은, 신택스 엘리먼트 view_reordering_idc가 refPicListX를 위해 없다는 것을 상술한다. ref_idx는 레퍼런스 픽쳐 리스트에 인터-뷰 픽쳐가 놓일 엔트리를 표시한다.

<141> abs_diff_view_idx_minusl plus 1는, ref_idx가 표시한 레퍼런스 픽쳐 리스트의 엔트리에 놓을 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 절대차(absolute difference)를 상술한다. abs_diff_view_idx_minus1은 0 내지 num_muitview_refs_for_listX[view_id]- 1의 범위이다. num_multiview_refs_for_listX[]는 앵커 픽쳐용 anchor_reference_view_for_list_X 및 비앵커 픽쳐용 non_anchor_reference_view_for_list_X[curr_view_id][] 를 나타내고, 여기서 curr_view_id는 현재 슬라이스를 담고 있는 뷰의 view_id 와 같다. inter- view 픽쳐의 뷰 인덱스는, MVC SPS 익스텍션에서 생기는 인터-뷰 픽쳐의 view_id 순서를 나타낸다. view_id와 같은 뷰 인덱스를 갖는 픽쳐의 경우, view_id 는 num_multiview_refs_for_listX[view_index]와 같다.

<142> abs_diff_view_idx_minusl plus 1는, 리스트 내의 현재 인덱스로 이동되고 있는 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 절대 차를 상술한다. abs_diff_view_idx_minusl은 0 내지 num_multiview_refs_for_listX[view_id]- 1의 범위이다. num_multiview_refs_for_listX[]는 앵커 픽쳐용 anchor_reference_view_for_list_X 및 비앵커 픽쳐용 non_anchor_reference_view_for_list_X[curr_view_id][] 를 나타내고, 여기서 curr_view_id는 현재 슬라이스를 담고 있는 뷰의 view_id 와 같다. inter- view 픽쳐의 뷰 인덱스는, MVC SPS 익스텍션에서 생기는 인터-뷰 픽쳐의 view_id 순서를 나타낸다. view_id 와 같은 뷰 인덱스를 갖는 픽쳐의 경우, view_id 는 num_multiview_refs_for_listX[view_index]와 같다.

<143> 디코딩 프로세스는 다음과 같다. :

<144> 인트라-뷰 픽쳐들의 절단(truncation) 이후에 NumRefIdxLXActive의 정의(definition)가 행해진다. :

<145> NumRefIdxLXActive = num_ref_idx_lX_active_minus1 +1 + num_multiview_refs_for_listX[view_id]

<146> G.8.2.4.3.3 인터-뷰 픽쳐들 용 레퍼런스 픽쳐 리스트들의 재배열 프로세스

<147> 이 프로세스에 대한 입력들은 레퍼런스 픽쳐 리스트 RefPicListX(X가 0 또는 1)이다.

<148> 이 프로세스의 출력들은 가능하게는 수정된 레퍼런스 픽쳐 리스트 RefPicListX((X가 0 또는 1)이다.

<149> 변수 picViewIdxLX는 다음과 같이 유도된다.

<150> If view_reordering_idc is equal to 0

<151> picViewIdxLX = picViewIdxLXPred - (abs_diff_view_idx_minus1 + 1)

<152> Otherwise (view_reordering_idc is equal to 1),

<153> picViewIdxLX = picViewIdxLXPred + (abs_diff_view_idx_minus1+ 1)

<154> picViewIdxLXPred는 변수 picViewIdxLX용 예측 값이다. 이 종속절에서 상술된 프로세스는 한 슬라이스에 처음으로 호출될 때(즉 ref_pic_list_reordering() 신택스 내에서 0 또는 1과 같은 view_reordering_idc 의 첫 번째 발생), picViewIdxL0Pred 및 picViewIdxL1Pred가 0으로 초기에 설정된다. picViewIdxLX의 각 할당 이후에, picViewIdxLX의 값이 picViewIdxLXPred로 할당된다.

<155> 다음 프로시저는 다음과 같이, picViewIdxLX와 같은 뷰 인덱스를 갖는 인터-뷰 픽쳐를 인덱스 위치 ref_Idx로 배치시키고, 임의의 다른 잔여 픽쳐들을 리스트의 뒤로 시프트한다.

```
for( cIdx = NumRefIdxLXActive; cIdx > ref_Idx; cIdx-- )
        RefPicListX[ cIdx ] = RefPicListX[ cIdx – 1]
RefPicListX[ref_Idx ] = inter-view reference picture with view id equal to
                        reference_view_for_list_X[picViewIdxLX]
nIdx = ref_Idx+1;
for( cIdx = refIdxLX; cIdx <= NumRefIdxLXActive; cIdx++ )
        if( ViewID(RefPicListX[ cIdx ] ) != TargetViewID||Time(RefPicListX[ cIdx
])!=TargetTime)
                RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
preView_id=PicViewIDLX
```

<156>

<157> Target ViewID 및 TargetTime 는 재배열될 타깃 레퍼런스 픽쳐의 view_id 또는 템포럴 축을 나타내고, Time(pic) 은 픽쳐 pic의 템포럴 축 값을 반환한다.

<158> 인터-뷰 픽쳐들과 인트라-예측을 위해 사용되는 픽쳐들 모두를 효율적으로 재배열하는 두 번째 방법에 따라, 위에서 상술된 것과 같은 인트라-뷰 픽쳐들용 레퍼런스 픽쳐 리스트 초기화 프로세스가 수행되고, 그 다음에 인터-뷰 픽쳐들이 MVC SPS 익스텐션에서 생기는 순서로 리스트의 끝에 첨부된다. 연속적으로 인트라-뷰 및 인터-뷰 픽쳐들 모두를 위한 RPLR 재배열 프로세스가 적용되고, 그 이후에 리스트 절단 프로세스가 적용된다. JMVM 1.0에 기초하여 수정된, 샘플 신택스, 시맨틱스 및 디코딩 프로세스는 다음과 같다.

<159> 레퍼런스 픽쳐 리스트 재배열 신택스

ref_pic_list_reordering( ) {	C	Descriptor
if(slice_type != I && slice_type !=) {		
ref_pic_list_reordering_flag_l0	2	u(1)
if(ref_pic_list_reordering_flag_l0)		
do {		
reordering_of_pic_nums_idc	2	ue(v)
if(reordering_of_pic_nums_idc == 0 || reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
if (reordering_of_pic_nums_idc == 4 || reordering_of_pic_nums_idc == 5)		
abs_diff_view_idx_minus1	2	ue(v)
} while(reordering_of_pic_nums_idc != 3)		
}		
if(slice_type == B || slice_type == EB) {		
ref_pic_list_reordering_flag_l1	2	u(1)
if(ref_pic_list_reordering_flag_l1)		
do {		
reordering_of_pic_nums_idc	2	ue(v)
if(reordering_of_pic_nums_idc == 0 || reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
if (reordering_of_pic_nums_idc == 4 || reordering_of_pic_nums_idc == 5)		
abs_diff_view_idx_minus1	2	ue(v)
} while(reordering_of_pic_nums_idc != 3)		
}		
}		

<160>

<161> G 7.4.3.1 레퍼런스 픽쳐 리스트 재배열 신택스

<162> 표

<163> 레퍼런스 픽쳐 리스트들의 재배열을 위한 Reordering_of_pic_nums_idc 오퍼레이션들

reordering_of_pic_nums_idc	Reordering specified
0	abs_diff_pic_num_minus1 is present and corresponds to a difference to subtract from a picture number prediction value
1	abs_diff_pic_num_minus1 is present and corresponds to a difference to add to a picture number prediction value
2	long_term_pic_num is present and specifies the long-term picture number for a reference picture
3	End loop for reordering of the initial reference picture list

<164>

4	abs_diff_view_idx_minus1 is present and corresponds to a difference to subtract from a view index prediction value
5	abs_diff_view_idx_minus1 is present and corresponds to a difference to add to a view index prediction value

<165>

<166> abs_diff_pic_num minus 1 또는 long_term_pic_num과 함께, reordering_of_pic_nums_idc는 리퍼런스 픽쳐들 중 어느 것이 재-맵핑될 것인지를 상술한다. abs_diff_view_idx_minus1과 함께 reordering_of__pic__nums_idc는 레퍼런스 픽쳐들 중 어느 것이 재-맵핑될 것인지를 상술한다. reordering_of_pic_nums_idc 의 값들은 상기 표에서 상술된다. ref_pic_list_reordering_flag_l0 또는 ref_pic_list_reordering__flag__l1 이후에 즉각적으로 따르는 첫 번째 reordering_of_pic_nums_idc는 3과 같지 않다.

<167> abs_diff_view_idx_minus1 plus 1 은, 레퍼런스 픽쳐 리스트 내의 현재 인덱스에 놓일 픽쳐의 뷰 인덱스와 뷰 인덱스 예측 값 사이의 절대 차이를 상술한다. abs_diff_view_idx_minus1 는, 0 내지 num_multiview_refs_for_listX[view_id]- 1사이의 범위이다. num_multiview_refs_for_listX[]는 앵커 픽쳐용 anchor_reference_view_for_list_X[curr_view_id][] 및 비앵커 픽쳐용 non_anchor_reference_view_for_list_X[curr_view_id][] 를 나타내고, 여기서 curr_view_id는 현재 슬라이스를 담고 있는 뷰의 view_id 와 같다. inter- view 픽쳐의 뷰 인덱스는, MVC SPS 익스텍션에서 생기는 인터-뷰 픽쳐의 view_id 순서를 나타낸다. view_index 와 같은 뷰 인덱스를 갖는 픽쳐의 경우, view_id 는 num_multiview_refs_for_listX[view_index]와 같다.

<168> 재배열 프로세스는 다음에서 설명되는 것과 같다.

<169> G. 8.2.4.3.3 인터-뷰 레퍼런스 픽쳐들을 위한 레퍼런스 픽쳐 리스트들의 재배열 프로세스

<170> 이 프로세스에 대한 입력은 index refIdxLX (X가 0 또는 1)이다.

<171> 이 프로세스의 출력은 증분된 index refIdxLX이다.

<172> 변수 picViewIdxLX는 다음과 같이 유도된다.

<173> If reordering_of_pic_nums_idc is equal to 4

<174> picViewIdxLX = picViewIdxLX Pred- (abs_diff_view_idx_minus1 + 1)

<175> Otherwise (reordering_of__pic_nums_idc is equal to 5),

<176> picViewIdxLX = picViewIdxLX Pred + (abs_diff_view_idx_minus1+ 1)

<177> picViewIdxLXPred는 변수 picViewIdxLX의 예측 값이다. 이 종속절에서 상술된 프로세스가 한 슬라이스에 대해 처음으로 호출될 때(즉 ref_pic_list_reordennng() 신택스 내에서 4 또는 5와 같은 reordering_of_pic_nums_idc의 첫 번째 발생), picViewIdxL0Pred 및 picViewIdxL1Pred가 0으로 초기에 설정된다. picViewIdxLX의 각 할당 이후에, picViewIdxLX의 값이 picViewIdxLXPred로 할당된다.

<178> 다음 프로시저는, picViewIdxLX와 같은 뷰 인덱스를 갖는 인터-뷰 픽쳐를 인덱스 위치 refldxLX로 배치시키고, 임의의 다른 잔여 픽쳐들의 위치를 리스트의 뒤로 시프트하고, 그리고 refldxLX의 값을 증분한다.

```
for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
        RefPicListX[ cIdx ] = RefPicListX[ cIdx – 1]
RefPicListX[ refIdxLX++ ] = inter-view reference picture with view id equal to
                            reference_view_for_list_X[picViewIdxLX]

nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )
        if( ViewID(RefPicListX[ cIdx ] ) != TargetViewID||Time(RefPicListX[ cIdx ])!=
TargetTime)
                RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

<179>

<180> Target ViewID 및 TargetTime 는 재배열될 타깃 레퍼런스 픽쳐의 view_id 또는 템포럴 축 값을 나타내고, Time(pic)은 픽쳐 pic의 템포럴 축 값을 반환한다.

<181> 인터-뷰 픽쳐들과 인트라-예측을 위해 사용되는 픽쳐들 모두를 효율적으로 재배열하는 세 번째 방법에 따라, 초기 레퍼런스 픽쳐 리스트는 "단기 레퍼런스를 위해 사용됨" 또는 "장기 레퍼런스를 위해 사용됨"으로 마크되고 현재 픽쳐와 동일한 view_id를 갖는 픽쳐들을 담고 있다. 또한 초기 레퍼런스 픽쳐 리스트는 인터-뷰 예측을 위해 사용될 수 있는 픽쳐들을 담고 있다. 인터-뷰 예측을 위해 사용되는 픽쳐들은 MVC용 시퀀스 파라미터 세트 익스텐션으로부터 추정되고, inter_view_reference_flag로부터 또한 예측될 수 있다. 인터-뷰 예측용 픽쳐들은, 이 픽쳐의 디코딩 프로세스용 일정 장기 레퍼런스 인덱스들을 할당받는다. 인터-뷰 레퍼런스 픽쳐들용 할당된 장기 레퍼런스 인덱스들, 예컨대 N개의 레퍼런스 인덱스들일 수 있고, 인트라-뷰 장기 픽쳐들용 인덱스들은 이 픽쳐의 디코딩 프로세스용 그 픽쳐들의 이전 값 + N과 같도록 수정될 수 있고, N은 인터-뷰 레퍼런스 픽쳐들의 개수를 나타낸다. 대안적으로, 할당된 장기 레퍼런스 인덱스들은, MaxLongTermFrameIdx + 1 이상 MaxLongTermFrameIdx + N 이하까지의 범위 내에 있을 수 있다. 대안적으로 MVC용 시퀀스 파라미터 세트 익스텐션은, 여기서 star_lt_index_for_rplr로 지칭되는 신택스 엘리먼트를 담고 있고, 할당된 장기 인덱스들은 start_lt_index_for_rplr이상 start_lt_index_for_rplr + N 미만 범위를 할당한다. 인터-뷰 레퍼런스 픽쳐들용 가용 장기 인덱스들은, view_id의 순서, 카메라 순서, 또는 뷰 종속성(dependency)들이 MVC용 시퀀스 파라미터 세트 익스텐션에 리스트되는 순서로 할당될 수 있다. RPLR 명령어들(신택스 및 시맨틱스)는 H.264/AVC 표준과 비교해서 변하지 않은 채 남아있다.

<182> 시간적 다이렉트- 관련(direct-related) 프로세싱을 위해, 예컨대 모션 벡터 스케일링을 위해, 양 레퍼런스 픽쳐들이 인터 예측(인트라-뷰 예측) 픽쳐들이라면(즉 레퍼런스 픽쳐들이 "인터-뷰 레퍼런스를 위해 사용됨"으로 마크되지 않았다면), AVC 디코딩 프로세스가 따라온다. 2개의 레퍼런스 픽쳐들 중 하나가 인터 예측 픽쳐이고 나머지 하나가 인터_뷰 예측 픽쳐라면,인터-뷰 예측 픽쳐가 장기 레퍼런스 픽쳐로서 다뤄진다. 만약 그렇지 않으면(즉 양 레퍼런스 픽쳐들이 인터-뷰 픽쳐들이라면), view_id 또는 카메라 순서 표시자(indicator) 값들이, 모션 벡터 스케일링을 위해 POC 값들 대신에 사용된다.

<183> 암시적(implicit) 가중 예측을 위한 예측 가중치들의 유도를 위해, 다음 프로세스가 수행된다. 양 레퍼런스 픽쳐들이 인터 예측(인트라-뷰 예측) 픽쳐들이라면(즉 "인터-뷰 레퍼런스를 위해 사용됨"으로 마크되지 않았다면), AVC 디코딩 프로세스가 따라온다. 2개의 레퍼런스 픽쳐들 중 하나가 인터 예측 픽쳐이고 나머지 하나가 인터_뷰 예측 픽쳐라면, 인터-뷰 예측 픽쳐가 장기 레퍼런스 픽쳐로서 다뤄진다. 만약 그렇지 않으면(즉 그 픽쳐들 모두가 인터-뷰 예측 픽쳐들이라면), view_id 또는 카메라 순서 표시자(indicator) 값들이, 가중 예측 파라미터들의 유도를 위해 POC 값들 대신에 사용된다.

<184> 본 발명은 방법 단계들의 일반적인 정황에서 설명되고, 네트워크 환경에서 컴퓨터에 의해 실행될 수 있고 컴퓨터 판독가능 매체 내에 구체화된 프로그램 코딩와 같은 컴퓨터 실행 가능한 명령어들을 포함하는 프로그램 제품에 의하여 일 실시예에서 그 단계들이 구현될 수 있다. 컴퓨터 판독가능 매체들의 예들은, 전자 디바이스 메모리 유닛들, RAM(random access memory), ROM(read only memory), CD(compact disc)들, DVD(digital versatile disc)들, 및 다른 내부 또는 외부 저장 디바이스들을 포함하는 다양한 유형의 저장 매체들을 포함할 수 있지만 이에 제한되지 않는다. 일반적으로 프로그램 모듈들은 특정 작업들을 수행하거나 특정 추출(abstract) 데이터형을 구현하는 루틴들, 프로그램들, 객체들, 컴포넌트들, 데이터 구조들 등을 포함한다. 데이터 구조들과 관련된 컴퓨터 실행 가능한 명령어들 및 프로그램 모듈들은, 여기에 개시된 방법들의 단계들을 실행하기 위한 프로그램 코딩의 예들을 나타낸다. 이와 같은 실행 가능한 명령어들 또는 관련된 데이터 구조들의 특정 시퀀스는 이와 같

은 단계들에서 묘사된 기능들을 구성하기 위한 상응하는 동작의 예들을 나타낸다.

<185> 다양한 데이터베이스 검색 단계, 상호연관(correlation) 단계, 비교 단계 및 결정 단계를 달성하기 위한 규칙 기반 로직 및 다른 로직을 갖는 표준 프로그래밍 테크닉들을 써서 본 발명의 소프트웨어 및 웹 구현들이 달성될 수 있다. 본 발명의 상세한 설명 및 청구항에서 사용되는 "컴포넌트(component)" 및 "모듈"이라는 용어는 하나 이상의 소프트웨어 코딩 라인들 및/또는 하드웨어 구현들, 및/또는 수신하는 수동 입력들을 위한 장치를 이용하여 구현되는 예들을 모두 포함하는 의도라는 것을 주목해야 한다.

<186> 본 발명의 실시 예들의 전술한 설명은 도해 및 설명 목적들을 위해 제시되었다. 본 발명을 개시된 정확한 형태로 제한하거나 한정시키려는 의도가 아니고, 수정들 및 변경들이 위의 교시들에 비추어 가능하거나 본 발명의 실행으로부터 얻어질 수 있다. 발명의 원리들 및 다양한 실시 예들과 고려되는 특정 용도에 적합한 다양한 수정들을 써서 당업자가 본 발명을 이용할 수 있도록 하는 그것의 실제적인 애플리케이션들을 설명하기 위해 실시 예들이 선택되고 설명되었다.

산업상 이용 가능성

<187> 본 발명은 멀티뷰 비디오 코딩에서 코딩된 픽쳐 버퍼 관리와 관련된다.

도면의 간단한 설명

<51> 도 1은 뷰-우선(first-view) 코딩 배열 내의 픽쳐들의 배열이다.

<52> 도 2는 시간-우선(time-first) 코딩 배열 내의 픽쳐들의 배열이다.

<53> 도 3은 예시적인 MVC 템포럴 및 인터-뷰 예측 구조의 묘사이다.

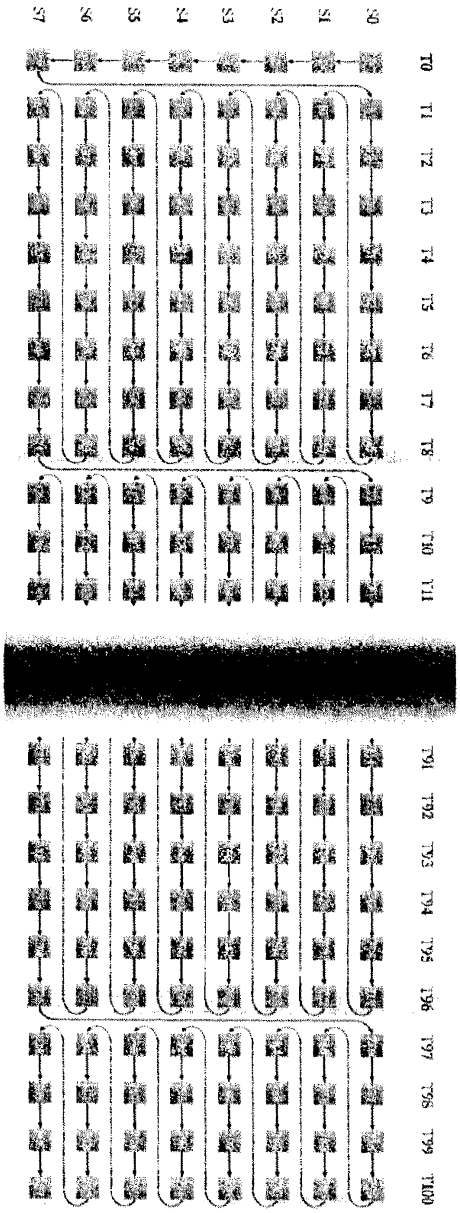
<54> 도 4는 본 발명이 구현될 수 있는 시스템의 개략도이다.

<55> 도 5는 본 발명의 구현에서 사용될 수 있는 이동 디바이스의 투시도이다.

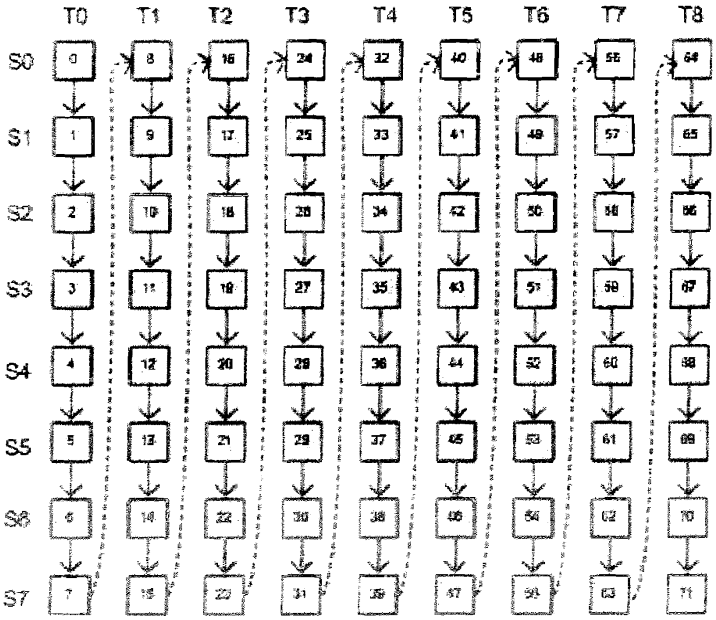
<56> 도 6은 도 5의 이동 디바이스의 회로소자의 개략적인 표현이다.

도면

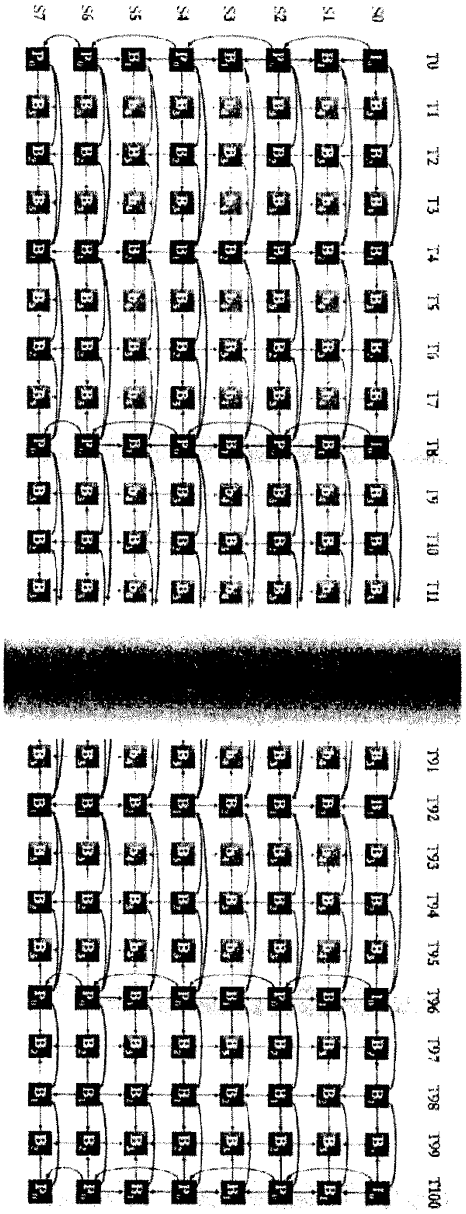
도면1


S0
S1
S2
S3
S4
S5
S6
S7
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T91
T92
T93
T94
T95
T96
T97
T98
T99
T100

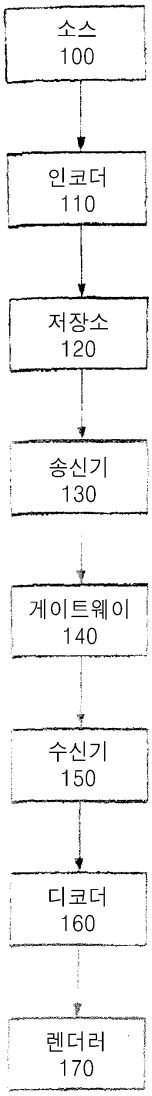
도면2



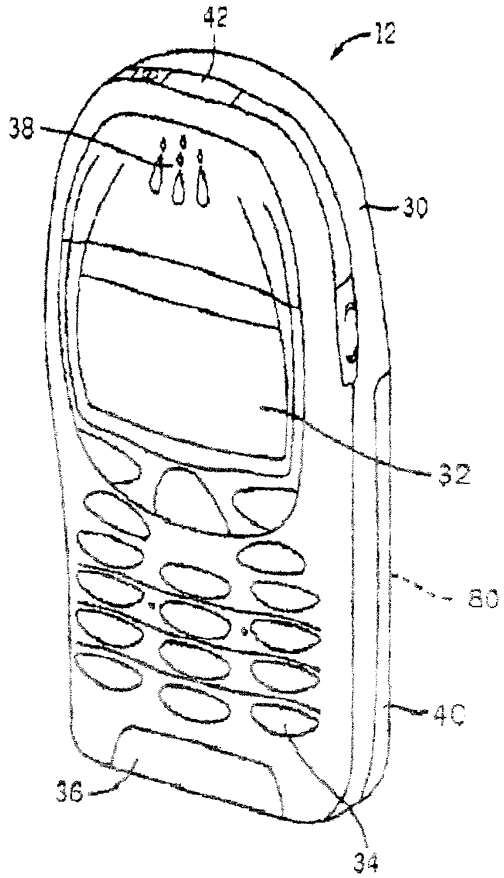
도면3



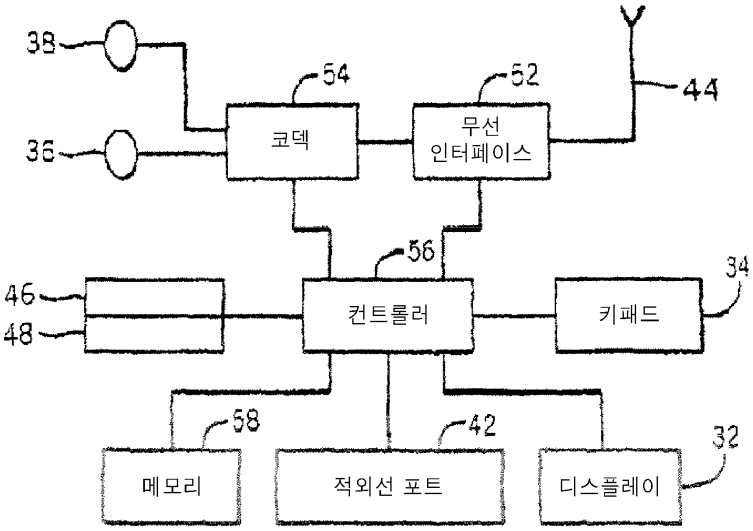
도면4



도면5


42
12
38
30
32
80
40
36
34

도면6


38
36
54
코덱
52
무선
인터페이스
44
56
46
48
컨트롤러
키패드
34
58
메모리
42
적외선 포트
디스플레이
32