(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년03월13일
(11) 등록번호 10-1958055
(24) 등록일자 2019년03월07일

(51) 국제특허분류(Int. Cl.)
H04N 19/51 (2014.01) *H04N 19/533* (2014.01)
H04N 19/537 (2014.01) *H04N 19/597* (2014.01)
(52) CPC특허분류
H04N 19/51 (2015.01)
H04N 19/533 (2015.01)
(21) 출원번호 10-2015-7029927
(22) 출원일자(국제) 2014년02월05일
심사청구일자 2018년03월09일
(85) 번역문제출일자 2015년10월16일
(65) 공개번호 10-2015-0132536
(43) 공개일자 2015년11월25일
(86) 국제출원번호 PCT/US2014/014835
(87) 국제공개번호 WO 2014/149212
국제공개일자 2014년09월25일
(30) 우선권주장
61/804,583 2013년03월22일 미국(US)
14/172,410 2014년02월04일 미국(US)
(56) 선행기술조사문헌
Gerhard et. al., "3D-HEVC Test Model 3", JCT3V 3rd meeting, Geneva, 23, Jan., 2013, JCT3V-C1005_d0..*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
강 제원
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
천 잉
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인코리아나

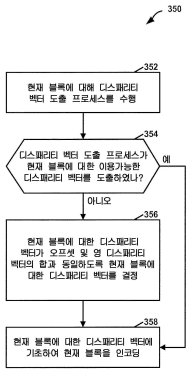
전체 청구항 수 : 총 27 항 심사관 : 박상철

(54) 발명의 명칭 **비디오 코딩에서의 디스패리티 벡터 리파인먼트**

(57) 요 약

비디오 코딩 디바이스가 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행한다. 현재 블록은 현재 뷰 내에 있다. 가용성 값은 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 나타낸다. 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 비디오 코딩 디바이스는 다른 방식으로 현재 블록에 대한 디스패리티 벡터를 생성한다.

대 표 도 - 도11



(52) CPC특허분류
H04N 19/537 (2015.01)
H04N 19/597 (2015.01)

명 세 서

청구범위

청구항 1

멀티-뷰 비디오 데이터를 디코딩하는 방법으로서,

상기 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계로서, 상기 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 단계;

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 상기 가용성 값을 설정하는 단계;

상기 디스패리티 벡터 도출 프로세스가 상기 디스패리티 벡터를 도출할 수 없는 것에 기초하여, 상기 디스패리티 벡터를 생성하는 단계로서, 상기 디스패리티 벡터를 생성하는 단계는,

상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 상기 가용성 값이 나타내는 것에 기초하여,

참조 뷰의 깊이 뷰 성분에 액세스하는 것으로서, 상기 참조 뷰는 상기 현재 뷰와는 상이한, 상기 깊이 뷰 성분에 액세스하는 것;

상기 참조 뷰의 상기 깊이 뷰 성분과 연관된 깊이 블록의 4 개의 코너 화소들과 연관된 개별의 깊이 값들로부터 깊이 값을 선택하는 것; 및

선택된 상기 깊이 값을 생성된 영 (zero) 디스패리티 벡터의 수평 성분으로 변환하여 리파인된 디스패리티 벡터를 형성하는 것

에 의해 적어도 부분적으로 영 수평 성분 및 영 수직 성분을 갖는 영 디스패리티 벡터에 디스패리티 벡터 리파인먼트 프로세스를 적용하는 단계를 포함하는, 상기 디스패리티 벡터를 생성하는 단계; 및

상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 디코딩하는 단계를 포함하는, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 2

제 1 항에 있어서,

상기 현재 블록에 대한 상기 디스패리티 벡터가 이용가능함을 나타내도록 상기 가용성 값을 설정하는 단계를 더 포함하는, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 3

제 2 항에 있어서,

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 상기 디스패리티 벡터를 원래 도출하였는지의 여부를 나타내기 위해 변수를 유지하는 단계; 및

상기 변수를 하나 이상의 코딩 도구들에 의한 사용을 위해 제공하는 단계를 더 포함하는, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 4

제 3 항에 있어서,

상기 변수의 값에 기초하여 상기 코딩 도구들 중 하나 이상을 인에이블하는 단계를 더 포함하는, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 5

제 1 항에 있어서,

상기 디스패리티 벡터 도출 프로세스는 이웃 블록 디스패리티 벡터 (neighboring block disparity vector; NBDV) 도출 프로세스이며; 그리고

상기 디스패리티 벡터 리파인먼트 프로세스는 이웃 블록 디스패리티 벡터 리파인먼트 (neighboring block disparity vector refinement; NBDV-R) 프로세스인, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 6

제 1 항에 있어서,

상기 현재 블록에 대한 상기 디스패리티 벡터에 기초하여 상기 현재 블록을 디코딩하는 단계는 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 상기 디스패리티 벡터를 사용하는 단계를 포함하는, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 7

제 1 항에 있어서,

상기 멀티-뷰 비디오 데이터는 3D-HEVC (High Efficiency Video Coding) 비디오 데이터인, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 8

제 1 항에 있어서,

상기 깊이 블록 및 상기 현재 블록은 동일한 사이즈인, 멀티-뷰 비디오 데이터를 디코딩하는 방법.

청구항 9

멀티-뷰 비디오 데이터를 인코딩하는 방법으로서,

상기 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계로서, 상기 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 단계;

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 상기 가용성 값을 설정하는 단계;

상기 디스패리티 벡터 도출 프로세스가 상기 디스패리티 벡터를 도출할 수 없는 것에 기초하여, 상기 디스패리티 벡터를 생성하는 단계로서, 상기 디스패리티 벡터를 생성하는 단계는,

상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 상기 가용성 값이 나타내는 것에 기초하여,

참조 뷰의 깊이 뷰 성분에 액세스하는 것으로서, 상기 참조 뷰는 상기 현재 뷰와는 상이한, 상기 깊이 뷰 성분에 액세스하는 것;

상기 참조 뷰의 상기 깊이 뷰 성분과 연관된 깊이 블록의 4 개의 코너 화소들과 연관된 개별의 깊이 값들로부터 깊이 값을 선택하는 것; 및

선택된 상기 깊이 값을 생성된 영 디스패리티 벡터의 수평 성분으로 변환하여 리파인된 디스패리티 벡터를 형성하는 것

에 의해 적어도 부분적으로 영 수평 성분 및 영 수직 성분을 갖는 영 디스패리티 벡터에 디스패리티 벡터 리파인먼트 프로세스를 적용하는 단계를 포함하는, 상기 디스패리티 벡터를 생성하는 단계; 및

상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 인코딩하는 단계를 포함하는, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 10

제 9 항에 있어서,

상기 현재 블록에 대한 상기 디스패리티 벡터가 이용가능함을 나타내도록 상기 가용성 값을 설정하는 단계를 더 포함하는, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 11

제 10 항에 있어서,

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 상기 디스패리티 벡터를 원래 도출하였는지의 여부를 나타내기 위해 변수를 유지하는 단계; 및

상기 변수를 하나 이상의 코딩 도구들에 의한 사용을 위해 제공하는 단계를 더 포함하는, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 12

제 11 항에 있어서,

상기 변수의 값에 기초하여 상기 코딩 도구들 중 하나 이상을 인에이블하는 단계를 더 포함하는, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 13

제 9 항에 있어서,

상기 디스패리티 벡터 도출 프로세스는 이웃 블록 디스패리티 벡터 (neighboring block disparity vector; NBDV) 도출 프로세스이며; 그리고

상기 디스패리티 벡터 리파인먼트 프로세스는 이웃 블록 디스패리티 벡터 리파인먼트 (neighboring block disparity vector refinement; NBDV-R) 프로세스인, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 14

제 9 항에 있어서,

상기 현재 블록에 대한 상기 디스패리티 벡터에 기초하여 상기 현재 블록을 인코딩하는 단계는 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 상기 디스패리티 벡터를 사용하는 단계를 포함하는, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 15

제 9 항에 있어서,

상기 멀티-뷰 비디오 데이터는 3D-HEVC (High Efficiency Video Coding) 비디오 데이터인, 멀티-뷰 비디오 데이터를 인코딩하는 방법.

청구항 16

비디오 디코딩 디바이스로서,

멀티-뷰 비디오 데이터를 저장하도록 구성된 메모리; 및

상기 메모리에 커플링된 하나 이상의 프로세서들을 포함하며,

상기 하나 이상의 프로세서들은,

상기 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 것으로서, 상기 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하고;

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 상기 현

재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 상기 가용성 값을 설정하고;

상기 디스패리티 벡터 도출 프로세스가 상기 디스패리티 벡터를 도출할 수 없는 것에 기초하여, 상기 디스패리티 벡터를 생성하는 것으로서, 상기 디스패리티 벡터를 생성하기 위해, 상기 하나 이상의 프로세서들은,

상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 상기 가용성 값이 나타내는 것에 기초하여, 영 수평 성분 및 영 수직 성분을 갖는 영 디스패리티 벡터에 디스패리티 벡터 리파인먼트 프로세스를 적용하도록 구성되고, 상기 디스패리티 벡터 리파인먼트 프로세스를 적용하기 위해, 상기 하나 이상의 프로세서들은,

상기 현재 블록에 대한 상기 디스패리티 벡터를 생성하기 위해 참조 뷰의 깊이 뷰 성분에 액세스하는 것으로서, 상기 참조 뷰는 상기 현재 뷰와는 상이한, 상기 깊이 뷰 성분에 액세스하고;

상기 참조 뷰의 상기 깊이 뷰 성분과 연관된 깊이 블록의 4 개의 코너 화소들과 연관된 개별의 깊이 값들로부터 깊이 값을 선택하고; 그리고

선택된 상기 깊이 값을 생성된 영 디스패리티 벡터의 수평 성분으로 변환하여 리파인된 디스패리티 벡터를 형성하도록

구성되는, 상기 디스패리티 벡터를 생성하고; 그리고

상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 디코딩하도록 구성되는, 비디오 디코딩 디바이스.

청구항 17

제 16 항에 있어서,

상기 비디오 디코딩 디바이스는,

하나 이상의 집적 회로들;

하나 이상의 마이크로프로세서들;

하나 이상의 디지털 신호 프로세서들 (DSP들);

하나 이상의 필드 프로그램가능 게이트 어레이들 (FPGA들);

데스크톱 컴퓨터;

랩톱 컴퓨터;

태블릿 컴퓨터;

폰;

텔레비전;

카메라;

디스플레이 디바이스;

디지털 미디어 플레이어;

비디오 게임 콘솔;

비디오 게임 디바이스;

비디오 스트리밍 디바이스; 또는

무선 통신 디바이스

중 적어도 하나를 포함하는, 비디오 디코딩 디바이스.

청구항 18

제 16 항에 있어서,

상기 하나 이상의 프로세서들은 상기 현재 블록에 대한 상기 디스패리티 벡터가 이용가능함을 나타내도록 상기 가용성 값을 설정하도록 구성되는, 비디오 디코딩 디바이스.

청구항 19

제 18 항에 있어서,

상기 하나 이상의 프로세서들은,

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 상기 디스패리티 벡터를 원래 도출하였는지의 여부를 나타내기 위해 변수를 유지하며; 그리고

상기 변수를 하나 이상의 코딩 도구들에 의한 사용을 위해 제공하도록 구성되는, 비디오 디코딩 디바이스.

청구항 20

제 19 항에 있어서,

상기 하나 이상의 프로세서들은 상기 변수의 값에 기초하여 상기 코딩 도구들 중 하나 이상을 인에이블하도록 구성되는, 비디오 디코딩 디바이스.

청구항 21

제 16 항에 있어서,

상기 디스패리티 벡터 도출 프로세스는 이웃 블록 디스패리티 벡터 (neighboring block disparity vector; NBDV) 도출 프로세스이며; 그리고

상기 디스패리티 벡터 리파인먼트 프로세스는 이웃 블록 디스패리티 벡터 리파인먼트 (neighboring block disparity vector refinement; NBDV-R) 프로세스인, 비디오 디코딩 디바이스.

청구항 22

제 16 항에 있어서,

상기 하나 이상의 프로세서들은 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 상기 디스패리티 벡터를 사용하도록 구성되는, 비디오 디코딩 디바이스.

청구항 23

제 16 항에 있어서,

상기 멀티-뷰 비디오 데이터는 3D-HEVC (High Efficiency Video Coding) 비디오 데이터인, 비디오 디코딩 디바이스.

청구항 24

제 16 항에 있어서,

상기 하나 이상의 프로세서들은 상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 디코딩하도록 구성되는, 비디오 디코딩 디바이스.

청구항 25

제 16 항에 있어서,

상기 하나 이상의 프로세서들은, 상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 코딩하는 부분으로서, 상기 하나 이상의 프로세서들이 상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 인코딩하도록 구성되는, 비디오 디코딩 디바이스.

청구항 26

비디오 디코딩 디바이스로서,

멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 수단으로서, 상기 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 수단;

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 상기 가용성 값을 설정하는 수단;

상기 디스패리티 벡터 도출 프로세스가 상기 디스패리티 벡터를 도출할 수 없는 것에 기초하여, 상기 디스패리티 벡터를 생성하는 수단으로서, 상기 디스패리티 벡터를 생성하는 수단은,

상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 상기 가용성 값이 나타내는 것에 기초하여, 영 수평 성분 및 영 수직 성분을 갖는 영 디스패리티 벡터에 디스패리티 벡터 리파인먼트 프로세스를 적용하는 수단을 포함하고, 상기 디스패리티 벡터 리파인먼트 프로세스를 적용하는 수단은,

참조 뷰의 깊이 뷰 성분에 액세스하는 수단으로서, 상기 참조 뷰는 상기 현재 뷰와는 상이한, 상기 깊이 뷰 성분에 액세스하는 수단;

상기 참조 뷰의 상기 깊이 뷰 성분과 연관된 깊이 블록의 4 개의 코너 화소들과 연관된 개별의 깊이 값들로부터 깊이 값을 선택하는 수단; 및

선택된 상기 깊이 값을 생성된 영 디스패리티 벡터의 수평 성분으로 변환하여 리파인된 디스패리티 벡터를 형성하는 수단

을 포함하는, 상기 디스패리티 벡터를 생성하는 수단; 및

상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 디코딩하는 수단을 포함하는, 비디오 디코딩 디바이스.

청구항 27

명령들을 저장하고 있는 비일시적 컴퓨터 판독가능 데이터 저장 매체로서,

상기 명령들은, 실행되는 경우, 비디오 디코딩 디바이스의 하나 이상의 프로세서들을,

멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 것으로서, 상기 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하고;

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 상기 현재 블록에 대한 상기 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 상기 가용성 값을 설정하고;

상기 디스패리티 벡터 도출 프로세스가 상기 디스패리티 벡터를 도출할 수 없는 것에 기초하여, 상기 디스패리티 벡터를 생성하는 것으로서, 상기 디스패리티 벡터를 생성하기 위한 상기 명령들은, 실행되는 경우, 상기 하나 이상의 프로세서들로 하여금,

상기 디스패리티 벡터 도출 프로세스가 상기 현재 블록에 대한 상기 디스패리티 벡터를 도출하지 않았음을 상기 가용성 값이 나타내는 것에 기초하여, 영 수평 성분 및 영 수직 성분을 갖는 영 디스패리티 벡터에 디스패리티 벡터 리파인먼트 프로세스를 적용하게 하는 명령들을 포함하고, 상기 디스패리티 벡터 리파인먼트 프로세스를 적용하기 위한 상기 명령들은, 실행되는 경우, 상기 하나 이상의 프로세서들로 하여금,

참조 뷰의 깊이 뷰 성분에 액세스하게 하는 것으로서, 상기 참조 뷰는 상기 현재 뷰와는 상이한, 상기 깊이 뷰 성분에 액세스하게 하고;

상기 참조 뷰의 상기 깊이 뷰 성분과 연관된 깊이 블록의 4 개의 코너 화소들과 연관된 개별의 깊이 값들로부터 깊이 값을 선택하게 하고; 그리고

선택된 상기 깊이 값을 생성된 영 디스패리티 벡터의 수평 성분으로 변환하여 리파인된 디스패리티 벡터를 형성하게 하는

명령들을 포함하는, 상기 디스패리티 벡터를 생성하고; 그리고

상기 현재 블록에 대한 상기 리파인된 디스패리티 벡터에 기초하여 상기 현재 블록을 디코딩하도록 구성하는, 비일시적 컴퓨터 판독가능 데이터 저장 매체.

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

청구항 45

삭제

발명의 설명

기 술 분 야

[0001] 본 출원은 2013년 3월 22일자로 출원된 미국 특허 가출원 제61/804,583호의 이익을 주장하며, 그 전체 내용은 참조로 본원에 통합된다.

[0002] 기술 분야

[0003] 본 개시물은 비디오 인코딩 및 디코딩에 관한 것이다.

배 경 기 술

[0004] 디지털 비디오 능력들은 디지털 텔레비전들, 디지털 직접 브로드캐스트 시스템들, 무선 브로드캐스트 시스템들, 개인 정보 단말기들 (PDA들), 랩톱 또는 데스크톱 컴퓨터들, 태블릿 컴퓨터들, e-북 리더들, 디지털 카메라들, 디지털 레코딩 디바이스들, 디지털 미디어 플레이어들, 비디오 게이밍 디바이스들, 비디오 게임 콘솔들, 셀룰러 또는 위성 무선 전화기들, 이른바 "스마트 폰들", 비디오 원격회의 디바이스들, 비디오 스트리밍 디바이스들 등을 포함한 넓은 범위의 디바이스들에 통합될 수 있다. 디지털 비디오 디바이스들은 MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, 파트 10, 고급 비디오 코딩 (Advanced Video Coding; AVC) 에 의해 규정된 표준들, 현재 개발중인 고 효율 비디오 코딩 (High Efficiency Video Coding; HEVC) 표준, 및 이러한 표준들의 확장본들에 기재된 것들과 같은 비디오 압축 기법들을 구현한다. 비디오 디바이스들은 이러한 비디오 압축 기법들을 구현하는 것에 의해 디지털 비디오 정보를 더 효율적으로 송신, 수신, 인코딩, 디코딩, 및/또는 저장할 수도 있다.

[0005] 비디오 압축 기법들은 공간적 (화상 내) 예측 및/또는 시간적 (화상 간) 예측을 수행하여 비디오 시퀀스들에 내재하는 리던던시를 감소시키거나 또는 제거한다. 블록 기반 비디오 코딩의 경우, 비디오 슬라이스 (즉, 비디오 프레임 또는 비디오 프레임의 부분) 가 비디오 블록들로 파티셔닝될 수도 있다. 화상의 인트라 코딩식 (intra-coded; I) 슬라이스에서의 비디오 블록들은 동일한 화상의 이웃 블록들에서의 참조 샘플들에 관한 공간적 예측을 이용하여 인코딩된다. 화상의 인터 코딩식 (inter-coded; P 또는 B) 슬라이스에서의 비디오 블록들은 동일한 화상의 이웃 블록들에서의 참조 샘플들에 관한 공간적 예측 또는 다른 참조 화상들에서의 참조 샘플들에 관한 시간적 예측을 이용할 수도 있다. 화상들은 프레임들이라고 지칭될 수도 있고, 참조 화상들은 참조 프레임들이라고 지칭될 수도 있다.

[0006] 공간적 또는 시간적 예측은 코딩될 블록에 대한 예측 블록이 생겨나게 한다. 잔차 데이터는 코딩될 원본 블록과 예측 블록 사이의 화소 차이들을 나타낸다. 인터 코딩식 블록이 예측 블록을 형성하는 참조 샘플들의 블록을 가리키는 모션 벡터에 따라 인코딩되고, 잔차 데이터는 코딩된 블록 및 예측 블록 사이의 차이를 나타낸다. 인트라 코딩식 블록이 인트라 코딩 모드 및 잔차 데이터에 따라 인코딩된다. 추가 압축을 위해, 잔차 데이터는 화소 도메인으로부터 변환 도메인으로 변환될 수도 있으며, 결과적으로 잔차 계수들이 생겨나며, 그 계수들은 그 다음에 양자화될 수도 있다. 처음에는 2차원 어레이로 배열된 양자화된 계수들은, 계수들의 1차원 벡터를 생성하기 위하여 스캐닝될 수도 있고, 엔트로피 코딩이 더 많은 압축을 달성하기 위해 적용될 수도 있다.

[0007] 멀티뷰 코딩 비트스트림이, 예컨대, 다수의 관점들에서 뷰들을 인코딩함으로써 생성될 수도 있다. 멀티뷰 코딩 양태들을 사용하는 몇몇 3차원 (3D) 비디오 표준들이 개발되어 있다. 예를 들어, 상이한 뷰들이 3D 비디오를 지원하기 위해 좌안 및 우안 뷰들을 송신할 수도 있다. 대안으로, 몇몇 3D 비디오 코딩 프로세스들이 이른바 멀티뷰 플러스 깊이 (multi-view plus depth) 코딩을 적용할 수도 있다. 멀티뷰 플러스 깊이 코딩에서, 3D 비디오 비트스트림이 텍스처 뷰 성분들뿐만 아니라, 깊이 뷰 성분들도 포함할 수도 있다. 예를 들어, 각각의 뷰는 하나의 텍스처 뷰 성분과 하나의 깊이 뷰 성분을 포함할 수도 있다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0008] 대체로, 본 개시물은 멀티-뷰 비디오 코딩에 관련된다. 더 구체적으로는, 비디오 코더가 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다. 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 비디오 코더는 현재 블록에 대한 리파인 (refine) 된 디스패리티 벡터를 생성하기 위해 디스패리티 벡터 리파인먼트 프로세스를 수행할 수도 있다. 본 개시물의 기법들은 백워드 뷰 합성 예측이 지원되는 경우 디스패리티 벡터 도출에 적용가능할 수도 있다.

[0009] 하나의 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 디코딩하는 방법을 설명하며, 그 방법은, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 단계; 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정하는 단계; 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내는 경우, 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하는 단계; 및 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 디코딩하는 단계를 포함한다.

[0010] 다른 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 인코딩하는 방법을 설명하며, 그 방법은, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 단계; 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정하는 단계; 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내는 경우, 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하는 단계; 및 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 인코딩하는 단계를 포함한다.

[0011] 다른 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 저장하는 메모리 및 하나 이상의 프로세서들을 포함하는 비디오 코딩 디바이스를 설명하며, 하나 이상의 프로세서들은, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 것으로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하고; 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정하며; 그리고 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내는 경우, 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하기 위해 디스패리티 벡터 리파인먼트 프로세스를 수행하도록 구성된다.

[0012] 다른 예에서, 본 개시물은 비디오 코딩 디바이스를 설명하며, 그 비디오 코딩 디바이스는, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 수단으로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 수단; 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정하는 수단; 및 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내는 경우, 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하는 수단을 포함한다.

[0013] 다른 예에서, 본 개시물은 명령들을 저장하고 있는 비일시적 컴퓨터 판독가능 데이터 저장 매체를 설명하며, 명령들은, 실행되는 경우, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 것으로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하고; 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정하며; 그리고 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터

를 생성하도록 하나 이상의 프로세서들을 구성한다.

[0014] 다른 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 디코딩하는 방법을 설명하며, 그 방법은, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 단계; 현재 블록에 대한 디스패리티 벡터가 이용불가능하다고 디스패리티 벡터 도출 프로세스가 결정하는 경우, 오프셋을 영 (zero) 디스패리티 벡터에 더함으로써 현재 블록에 대한 디스패리티 벡터를 결정하는 단계; 및 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 디코딩하는 단계를 포함한다.

[0015] 다른 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 인코딩하는 방법을 설명하며, 그 방법은, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 단계; 현재 블록에 대한 디스패리티 벡터가 이용불가능하다고 디스패리티 벡터 도출 프로세스가 결정하는 경우, 오프셋을 영 디스패리티 벡터에 더함으로써 현재 블록에 대한 디스패리티 벡터를 결정하는 단계; 및 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 인코딩하는 단계를 포함한다.

[0016] 다른 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 저장하는 메모리 및 하나 이상의 프로세서들을 포함하는, 멀티-뷰 비디오 데이터를 코딩하는 디바이스를 설명하며, 하나 이상의 프로세서들은, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 것으로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하고; 현재 블록에 대한 디스패리티 벡터가 이용불가능하다고 디스패리티 벡터 도출 프로세스가 결정하는 경우, 오프셋을 영 디스패리티 벡터에 더함으로써 현재 블록에 대한 디스패리티 벡터를 결정하며; 그리고 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 코딩하도록 구성된다.

[0017] 다른 예에서, 본 개시물은 멀티-뷰 비디오 데이터를 코딩하는 디바이스를 설명하며, 그 디바이스는, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 수단으로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하는 수단; 현재 블록에 대한 디스패리티 벡터가 이용불가능하다고 디스패리티 벡터 도출 프로세스가 결정하는 경우, 오프셋을 영 디스패리티 벡터에 더함으로써 현재 블록에 대한 디스패리티 벡터를 결정하는 수단; 및 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 코딩하는 수단을 포함한다.

[0018] 다른 예에서, 본 개시물은 명령들을 저장하고 있는 비일시적 컴퓨터 판독가능 데이터 저장 매체를 설명하며, 명령들은, 멀티-뷰 비디오 데이터를 코딩하는 디바이스의 하나 이상의 프로세서들에 의해 실행되는 경우, 상기 디바이스로 하여금, 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하게 하는 것으로서, 현재 블록은 현재 뷰 내에 있는, 상기 디스패리티 벡터 도출 프로세스를 수행하게 하고; 현재 블록에 대한 디스패리티 벡터가 이용불가능하다고 디스패리티 벡터 도출 프로세스가 결정하는 경우, 오프셋을 영 디스패리티 벡터에 더함으로써 현재 블록에 대한 디스패리티 벡터를 결정하게 하며; 그리고 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 코딩하게 한다.

[0019] 본 개시물의 하나 이상의 예들의 상세는 첨부 도면들 및 다음의 설명에서 언급된다. 다른 특징들, 목적들, 및 장점들은 상세한 설명, 도면들, 및 청구항들로부터 명확하게 될 것이다.

도면의 간단한 설명

[0020] 도 1은 본 개시물에서 설명되는 기법들을 이용할 수도 있는 일 예의 비디오 코딩 시스템을 도시하는 블록도이다.

도 2는 현재 PU에 관하여 예의 공간적으로 이웃하는 예측 유닛 (prediction unit; PU) 들을 도시하는 개념도이다.

도 3은 일 예의 멀티-뷰 디코딩 순서를 도시하는 개념도이다.

도 4는 멀티-뷰 코딩을 위한 일 예의 예측 구조를 도시하는 개념도이다.

도 5는 백워드 뷰 합성 예측 (backward view synthesis prediction; BVSP) 을 수행하기 위한 참조 뷰로부터의 깊이 블록 도출을 예시하는 개념도이다.

도 6은 본 개시물에서 설명되는 기법들을 구현할 수도 있는 일 예의 비디오 인코더를 도시하는 블록도이다.

도 7은 본 개시물에서 설명되는 기법들을 구현할 수도 있는 일 예의 비디오 디코더를 도시하는 블록도이다.

도 8은 본 개시물의 하나 이상의 기법들에 따른, 멀티-뷰 비디오 데이터를 디코딩하는 비디오 디코더의 일 예의 동작을 도시하는 흐름도이다.

도 9는 본 개시물의 하나 이상의 기법들에 따른, 멀티-뷰 비디오 데이터를 인코딩하는 비디오 인코더의 일 예의 동작을 도시하는 흐름도이다.

도 10은 본 개시물의 하나 이상의 부가적인 기법들에 따른, 멀티-뷰 비디오 데이터를 디코딩하는 비디오 디코더의 일 예의 동작을 도시하는 흐름도이다.

도 11은 본 개시물의 하나 이상의 부가적인 기법들에 따른, 멀티-뷰 비디오 데이터를 인코딩하는 비디오 인코더의 일 예의 동작을 도시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0021] 고 효율 비디오 코딩 (HEVC) 이 새로 개발된 비디오 코딩 표준이다. 3D-HEVC는 3차원 비디오 데이터에 대한 HEVC의 확장본이다. 3D-HEVC는 상이한 관점들로부터 동일한 장면의 다수의 뷰들을 제공한다. 3D-HEVC에 대한 표준화 노력들은 HEVC에 기초한 멀티-뷰 비디오 코덱의 표준화를 포함한다. 3D-HEVC에서, 상이한 뷰들로부터의 복원된 뷰 성분들에 기초한 뷰 간 예측이 가능하게 된다. 코딩 효율을 더욱 개선하기 위해, 2 개의 새로운 코딩 도구들, 즉 뷰 간 모션 예측 및 뷰 간 잔차 예측이, 3D-HEVC에 대한 참조 소프트웨어의 일부 버전들에서 채택되었다.

[0022] 3D-HEVC에서, 뷰 간 모션 예측은 표준 HEVC에서 사용되는 모션 보상과 유사하고, 동일한 또는 유사한 신택스 엘리먼트들을 이용할 수도 있다. 병합 모드, 스킵 모드, 및 고급 모션 벡터 예측 (Advanced Motion Vector Prediction; AMVP) 모드가 모션 예측의 예의 유형들이다. 비디오 코더가 예측 유닛 (PU) 에 대해 뷰 간 모션 예측을 수행하는 경우, 비디오 코더는, 모션 정보의 소스로서, PU와 동일한 액세스 유닛에 있지만 상이한 뷰에 있는 화상을 사용할 수도 있다. 그 반면, 기존의 모션 보상은 상이한 액세스 유닛들에서의 화상들만을 참조 화상들로서 사용한다. 따라서, 3D-HEVC에서, 의존성 뷰에서의 블록의 모션 파라미터들은 동일한 액세스 유닛의 다른 뷰들에서의 이미 코딩된 모션 파라미터들에 기초하여 예측 또는 유추될 수도 있다.

[0023] 비디오 코더가 모션 예측을 수행하는 경우, 현재 PU의 모션 정보가 병합 모드, 스킵 모드, 또는 AMVP 모드를 사용하여 시그널링될 때, 비디오 코더는 후보 리스트 (예컨대, 병합하는 후보 리스트 또는 AMVP 후보 리스트) 를 생성할 수도 있다. 3D-HEVC에서의 뷰 간 모션 예측을 구현하기 위해, 후보 리스트는 뷰 간 예측된 모션 벡터 후보들을 포함할 수도 있다. 비디오 코더는 후보 리스트에서의 다른 후보들과 동일한 방식으로 뷰 간 예측된 모션 벡터 후보를 사용할 수도 있다. 뷰 간 예측된 모션 벡터 후보가 디스패리티 참조 화상의 PU (즉, 참조 PU) 의 모션 정보를 특정할 수도 있다. 디스패리티 참조 화상은 현재 PU와 동일한 액세스 유닛에 있지만 현재 PU와는 상이한 뷰에 있을 수도 있다. 디스패리티 참조 화상에서의 참조 PU를 결정하기 위해, 비디오 코더는 현재 PU에 대한 디스패리티 벡터를 결정하기 위해 디스패리티 벡터 구축 프로세스를 수행할 수도 있다. 현재 PU에 대한 디스패리티 벡터는 현재 PU의 예측 블록과 디스패리티 참조 화상 내의 로케이션 사이의 공간적 변위를 나타낼 수도 있다. 참조 PU는 디스패리티 벡터에 의해 나타내어진 로케이션을 커버하는 디스패리티 참조 화상의 PU일 수도 있다.

[0024] 대체로, 예측 목적으로 이웃 뷰포인트 (viewpoint) 로부터의 화상을 현재 뷰포인트로 워핑 (warping) 하는 기법이 뷰 합성 예측 (VSP) 이다. 깊이 정보가 워핑을 수행하는데 사용된다. 백워드 워핑 VSP (BVSP) 가 백워드 워핑 동작을 수행하기 위해 이웃 블록들을 사용하여 깊이 블록을 도출한다. 백워드 워핑에서, 워핑을 위해 사용되는 깊이는 현재 화상의 동일한 뷰포인트로부터이며, 이는 의존성 뷰에서의 깊이 우선 코딩을 통상 요구한다. BVSP에서의 블록에 대한 깊이 정보를 추정하기 위하여, 비디오 코더가 이웃 블록들로부터 디스패리티 벡터를 도출할 수도 있다. 비디오 코더는 그 다음에 참조 뷰로부터 깊이 블록을 획득하기 위해 도출된 디스패리티 벡터를 사용할 수도 있다.

[0025] 비디오 코더는 디스패리티 벡터를 도출하기 위해 이웃 블록 기반 디스패리티 벡터 (Neighboring Blocks based Disparity Vector; NBDV) 라고 지칭되는 방법을 수행할 수도 있다. 비디오 코더가 현재 PU에 대한 디스패리티 벡터를 도출하기 위해 NBDV 도출을 사용하는 경우, 비디오 코더는 디스패리티 벡터를 도출하기 위해 공간적 및 시간적 이웃 블록들로부터의 디스패리티 모션 벡터들을 사용할 수도 있다. 본 개시물에서, PU의 디스패리티 모션 벡터가 디스패리티 참조 화상 (즉, PU와는 상이한 뷰에 있는 참조 화상) 에서의 포지션을 나타내는

모션 벡터를 지칭한다. 더욱이, 설명의 편의를 위해, 본 개시물은 공간적으로 이웃하는 블록들 또는 시간적으로 이웃하는 블록들 중 어느 한쪽을 이웃 블록들이라고 지칭할 수도 있다.

[0026] 비디오 코더가 참조 뷰의 깊이 뷰 성분을 사용하여 디스패리티 벡터를 리파인할 수 있다. 비디오 코더는 백워드 뷰 합성 예측에서의 사용을 위한 디스패리티 모션 벡터를 리파인하기 위해 동일한 리파인먼트 프로세스를 사용할 수도 있다. 특히, 비디오 코더는 현재 PU에 대한 디스패리티 벡터를 결정하기 위해 NBDV 프로세스를 사용할 수도 있다. 비디오 코더가 NBDV 도출을 사용하여 이용가능한 디스패리티 벡터를 결정하는 경우 (예컨대, 비디오 코더가 이웃 블록들 중에서 디스패리티 모션 벡터를 찾는 경우), 비디오 코더는 참조 뷰의 깊이 맵으로부터 깊이 데이터를 취출함으로써 디스패리티 벡터를 추가로 리파인할 수도 있다. 리파인먼트 프로세스는 2 개의 단계들을 포함한다. 첫째, 비디오 코더는 이전에 코딩된 참조 깊이 뷰, 이를테면 기본 뷰 (base view) 에서의 도출된 디스패리티 벡터에 의해 대응하는 깊이 블록을 위치결정한다. 대응하는 깊이 블록의 사이즈는 현재 PU의 사이즈와 동일하다. 둘째, 비디오 코더는 하나의 깊이 값을 대응하는 깊이 블록의 4 개의 코너 화소들로부터 선택하고 선택된 깊이 값을 리파인된 디스패리티 벡터의 수평 성분으로 변환한다. 디스패리티 벡터의 수직 성분은 변경되지 않는다.

[0027] 위의 프로세스는 NBDV 리파인먼트 (NBDV-R) 또는 깊이 지향 NBDV (Do-NBDV) 라고 또한 지칭된다. 비디오 코더는 뷰 간 모션 예측을 위해 리파인된 디스패리티 벡터를 사용할 수도 있는 한편 비디오 코더는 뷰 간 잔차 예측을 위해 미리파인된 (unrefined) 디스패리티 벡터를 사용할 수도 있다. 더욱이, 비디오 코더는 PU가 백워드 VSP 모드로 코딩되면 리파인된 디스패리티 벡터를 PU의 모션 벡터로서 저장할 수도 있다. NBDV 프로세스가 이용가능한 디스패리티 벡터를 제공하지 않는 경우, 위의 NBDV-R 프로세스는 스킵되고 영 디스패리티 벡터가 직접 반환된다.

[0028] 3D-HEVC에서의 현재 디스패리티 벡터 도출 방법은 여러 문제들을 갖는다. 예를 들어, NBDV 프로세스가 이용불가능한 디스패리티 벡터를 제공하는 경우, 비디오 코더는 그 디스패리티 벡터를 리파인하는 NBDV-R 프로세스를 스킵할 수도 있다. 이는 코딩 성능 저하로 이어질 수도 있다.

[0029] 본 개시물의 기법들은 이전에 언급된 문제들 또는 단점들을 해결할 수도 있다. 다시 말하면, 본 개시물은 디스패리티 벡터 도출 방법 (예컨대, 3D-HEVC에서의 NBDV) 이 이용가능한 디스패리티 벡터를 생성할 수 없는 경우 참조 뷰의 깊이 뷰 성분에 액세스함으로써 양호한 리파인된 디스패리티 벡터를 제공할 수도 있다.

[0030] 몇몇 예들에서, 비디오 디코더는 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다. 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 비디오 디코더는 그럼에도 불구하고 여전히 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하기 위해 디스패리티 벡터 리파인먼트 프로세스를 수행할 수도 있다. 이러한 몇몇 예들에서, 디스패리티 벡터 리파인먼트 프로세스는 영 디스패리티 벡터를 사용한다. 다르게 말하면, 디스패리티 벡터 리파인먼트 프로세스는 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하는 경우들로 제한되지 않는다. 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않더라도, 본 개시물의 기법들은 일부 디폴트 디스패리티 벡터 (이를테면 영과 동일한 수평 및 수직 성분들을 갖는 디폴트 디스패리티 벡터) 에 대해 디스패리티 벡터 리파인먼트를 허용한다.

[0031] 몇몇 예들에서, 디스패리티 벡터 도출 프로세스 (예컨대, NBDV 도출) 가 이용가능한 디스패리티 벡터를 원래 반환하였는지의 여부를 식별하기 위해 변수가 유지된다. 이 변수는 특정한 조건들에서의 다른 코딩 도구들을 위해 사용될 수 있다. 예를 들면, 0과 동일한 이 플래그는 현재 블록에 대한 뷰 간 잔차 예측의 불능화 (disabling) 로 이어질 수도 있다.

[0032] 도 1은 본 개시물의 기법들을 이용할 수도 있는 일 예의 비디오 코딩 시스템 (10) 을 도시하는 블록도이다. 본원에서 사용되는 바와 같이, "비디오 코더"라는 용어는 비디오 인코더들 및 비디오 디코더들 양쪽 모두를 일반적으로 지칭한다. 본 개시물에서, "비디오 코딩" 또는 "코딩"이란 용어들은 비디오 인코딩 또는 비디오 디코딩을 일반적으로 지칭할 수도 있다.

[0033] 도 1에 도시된 바와 같이, 비디오 코딩 시스템 (10) 은 소스 디바이스 (12) 및 목적지 디바이스 (14) 를 구비한다. 소스 디바이스 (12) 는 인코딩된 비디오 데이터를 생성한다. 따라서, 소스 디바이스 (12) 는 비디오 인코딩 디바이스 또는 비디오 인코딩 장치라고 지칭될 수도 있다. 목적지 디바이스 (14) 는 소스 디바이스 (12) 에 의해 생성된 인코딩된 비디오 데이터를 디코딩할 수도 있다. 따라서, 목적지 디바이스 (14) 는 비디오 디코딩 디바이스 또는 비디오 디코딩 장치라고 지칭될 수도 있다. 소스 디바이스 (12) 및 목적지 디

바이스 (14) 는 비디오 코딩 디바이스들 또는 비디오 코딩 장치들의 예들일 수도 있다.

[0034] 소스 디바이스 (12) 및 목적지 디바이스 (14) 는 데스크톱 컴퓨터들, 모바일 컴퓨팅 디바이스들, 노트북 (예컨대, 랩톱) 컴퓨터들, 태블릿 컴퓨터들, 셋톱 박스들, 이른바 "스마트" 폰들과 같은 전화기 핸드셋들, 텔레비전들, 카메라들, 디스플레이 디바이스들, 디지털 미디어 플레이어들, 비디오 게이밍 콘솔들, 차량내 컴퓨터들 등을 포함한 다양한 범위의 디바이스들을 포함할 수도 있다.

[0035] 목적지 디바이스 (14) 는 소스 디바이스 (12) 로부터의 인코딩된 비디오 데이터를 채널 (16) 을 통해 수신할 수도 있다. 채널 (16) 은 소스 디바이스 (12) 로부터 목적지 디바이스 (14) 로 인코딩된 비디오 데이터를 이동시킬 수 있는 하나 이상의 매체들 또는 디바이스들을 포함할 수도 있다. 하나의 예에서, 채널 (16) 은 소스 디바이스 (12) 가 인코딩된 비디오 데이터를 직접 목적지 디바이스 (14) 로 실시간으로 송신하는 것을 가능하게 하는 하나 이상의 통신 매체들을 포함할 수도 있다. 이 예에서, 소스 디바이스 (12) 는 인코딩된 비디오 데이터를 통신 표준, 이를테면 무선 통신 프로토콜에 따라 변조할 수도 있고, 변조된 비디오 데이터를 목적지 디바이스 (14) 로 송신할 수도 있다. 하나 이상의 통신 매체들은 무선 및/또는 유선 통신 매체들, 이를테면 무선 주파수 (RF) 스펙트럼 또는 하나 이상의 물리적 송신 라인들을 포함할 수도 있다. 하나 이상의 통신 매체들은 패킷 기반 네트워크, 이를테면 로컬 영역 네트워크, 광역 네트워크, 또는 글로벌 네트워크 (예컨대, 인터넷) 의 일부를 형성할 수도 있다. 하나 이상의 통신 매체들은 라우터들, 스위치들, 기지국들, 또는 소스 디바이스 (12) 로부터 목적지 디바이스 (14) 로의 통신을 용이하게 하는 다른 장비를 포함할 수도 있다.

[0036] 다른 예에서, 채널 (16) 은 소스 디바이스 (12) 에 의해 생성된 인코딩된 비디오 데이터를 저장하는 저장 매체를 포함할 수도 있다. 이 예에서, 목적지 디바이스 (14) 는 예컨대, 디스크 액세스 또는 카드 액세스를 통해 저장 매체에 액세스할 수도 있다. 저장 매체는 블루 레이 디스크들, DVD들, CD-ROM들, 플래시 메모리, 또는 인코딩된 비디오 데이터를 저장하기 위한 다른 적합한 디지털 저장 매체들과 같은 다양한 국소적으로 액세스되는 데이터 저장 매체들을 포함할 수도 있다.

[0037] 추가의 예에서, 채널 (16) 은 소스 디바이스 (12) 에 의해 생성된 인코딩된 비디오 데이터를 저장하는 파일 서버 또는 다른 중간 저장 디바이스를 포함할 수도 있다. 이 예에서, 목적지 디바이스 (14) 는 파일 서버 또는 다른 중간 저장 디바이스에 저장된 인코딩된 비디오 데이터를 스트리밍 또는 다운로드를 통해 액세스할 수도 있다. 파일 서버는 인코딩된 비디오 데이터를 저장하고 그 인코딩된 비디오 데이터를 목적지 디바이스 (14) 로 송신할 수 있는 유형의 서버일 수도 있다. 예의 파일 서버들은 웹 서버들 (예컨대, 웹사이트용), 파일 전송 프로토콜 (FTP) 서버들, 네트워크 부속 스토리지 (network attached storage; NAS) 디바이스들, 및 로컬 디스크 드라이브들을 포함한다.

[0038] 목적지 디바이스 (14) 는 표준 데이터 접속, 이를테면 인터넷 접속을 통해, 인코딩된 비디오 데이터에 액세스할 수도 있다. 예의 유형들의 데이터 접속들은 파일 서버 상에 저장된 인코딩된 비디오 데이터에 액세스하기에 적합한 무선 채널들 (예컨대, Wi-Fi 접속들), 유선 접속들 (예컨대, 디지털 가입자 회선 (digital subscriber line; DSL), 케이블 모뎀 등), 또는 양쪽 모두의 조합들을 포함할 수도 있다. 파일 서버로부터의 인코딩된 비디오 데이터의 송신은 스트리밍 송신, 다운로드 송신, 또는 양쪽 모두의 조합일 수도 있다.

[0039] 본 개시물의 기법들은 무선 애플리케이션들 또는 설정 (setting) 들로 제한되지 않는다. 그 기법들은, 다양한 멀티미디어 애플리케이션들, 이를테면 OTA (over-the-air) 텔레비전 브로드캐스트들, 케이블 텔레비전 송신들, 위성 텔레비전 송신들, 예컨대, 인터넷을 통한 스트리밍 비디오 송신들의 지원 하의 비디오 코딩, 데이터 저장 매체 상의 저장을 위한 비디오 데이터의 인코딩, 데이터 저장 매체 상에 저장된 비디오 데이터의 디코딩, 또는 다른 애플리케이션들에 적용될 수도 있다. 몇몇 예들에서, 비디오 코딩 시스템 (10) 은 비디오 스트리밍, 비디오 플레이백, 비디오 브로드캐스팅, 및/또는 화상 통화와 같은 애플리케이션들을 지원하기 위해 단방향 또는 양방향 비디오 송신을 지원하도록 구성될 수도 있다.

[0040] 도 1은 단지 일 예이고 본 개시물의 기법들은 인코딩 및 디코딩 디바이스들 간에 임의의 데이터 통신을 반드시 포함하지는 않는 비디오 코딩 설정들 (예컨대, 비디오 인코딩 또는 비디오 디코딩) 에 적용될 수도 있다. 다른 예들에서, 데이터는 로컬 메모리로부터 취출되며, 네트워크를 통해 스트리밍되는 등등이 된다. 비디오 인코딩 디바이스가 데이터를 인코딩하고 메모리에 저장할 수도 있으며, 그리고/또는 비디오 디코딩 디바이스가 메모리로부터 데이터를 취출하고 디코딩할 수도 있다. 많은 예들에서, 인코딩 및 디코딩은, 서로 통신하지 않지만 단순히 데이터를 메모리에 인코딩하고 및/또는 메모리로부터 데이터를 취출하고 디코딩하는 디바이스들에 의해 수행된다. 그 데이터는 비디오 데이터, 이를테면 멀티-뷰 비디오 데이터를 포함할 수도 있다.

[0041] 도 1의 예에서, 소스 디바이스 (12) 는 비디오 소스 (18), 비디오 인코더 (20), 및 출력 인터페이스 (22) 를 구비한다. 몇몇 예들에서, 출력 인터페이스 (22) 는 변조기/복조기 (모뎀) 및/또는 송신기를 구비할 수도 있다. 비디오 소스 (18) 는 비디오 캡처 디바이스, 예컨대, 비디오 카메라, 이전에 캡처된 비디오 데이터를 포함한 비디오 아카이브, 비디오 콘텐츠 제공자로부터 비디오 데이터를 수신하는 비디오 피드 인터페이스, 및/또는 비디오 데이터를 생성하는 컴퓨터 그래픽 시스템, 또는 비디오 데이터의 이러한 소스들의 조합을 포함할 수도 있다.

[0042] 비디오 인코더 (20) 는 비디오 소스 (18) 로부터의 비디오 데이터를 인코딩할 수도 있다. 몇몇 예들에서, 소스 디바이스 (12) 는 출력 인터페이스 (22) 를 통해 목적지 디바이스 (14) 로 인코딩된 비디오 데이터를 직접 송신할 수도 있다. 다른 예들에서, 인코딩된 비디오 데이터는 디코딩 및/또는 플레이백을 위한 목적지 디바이스 (14) 에 의한 나중의 액세스를 위해 저장 매체 또는 파일 서버 상에 또한 저장될 수도 있다.

[0043] 도 1의 예에서, 목적지 디바이스 (14) 는 입력 인터페이스 (28), 비디오 디코더 (30), 및 디스플레이 디바이스 (32) 를 구비한다. 몇몇 예들에서, 입력 인터페이스 (28) 는 수신기 및/또는 모뎀을 구비한다. 입력 인터페이스 (28) 는 채널 (16) 을 통해 인코딩된 비디오 데이터를 수신할 수도 있다. 비디오 디코더 (30) 는 인코딩된 비디오 데이터를 디코딩할 수도 있다. 디스플레이 디바이스 (32) 는 디코딩된 비디오 데이터를 디스플레이할 수도 있다. 디스플레이 디바이스 (32) 는 목적지 디바이스 (14) 와 통합될 수도 있거나 또는 그것 외부에 있을 수도 있다. 디스플레이 디바이스 (32) 는 액정 디스플레이 (LCD), 플라즈마 디스플레이, 유기 발광 다이오드 (OLED) 디스플레이, 또는 다른 유형의 디스플레이 디바이스와 같은 다양한 디스플레이 디바이스들을 포함할 수도 있다.

[0044] 비디오 인코더 (20) 와 비디오 디코더 (30) 각각은 다양한 적합한 회로부, 이를테면 하나 이상의 마이크로프로세서들, 디지털 신호 프로세서들 (DSP들), 주문형 집적회로들 (ASIC들), 필드 프로그램가능 게이트 어레이들 (FPGA들), 개별 로직, 하드웨어, 또는 그것들의 임의의 조합 중 임의의 것으로서 구현될 수도 있다. 그 기법들이 부분적으로 소프트웨어에서 구현되면, 디바이스가 적합한 비일시적 컴퓨터 판독가능 저장 매체 내에 소프트웨어에 대한 명령을 저장할 수도 있고 하나 이상의 프로세서들을 사용하여 하드웨어에서 그 명령들을 실행하여 본 개시물의 기법들을 수행할 수도 있다. 전술한 바 (하드웨어, 소프트웨어, 하드웨어 및 소프트웨어의 조합 등을 포함) 중 임의의 것은 하나 이상의 프로세서들인 것으로 간주될 수도 있다. 비디오 인코더 (20) 및 비디오 디코더 (30) 의 각각은 하나 이상의 인코더들 또는 디코더들 내에 구비될 수도 있고, 그것들 중 어느 하나는 결합형 인코더/디코더 (CODEC) 의 일부로서 개별 디바이스 내에 통합될 수도 있다.

[0045] 본 개시물은 다른 디바이스, 이를테면 비디오 디코더 (30) 에 특정한 정보를 "시그널링하는" 비디오 인코더 (20) 에 일반적으로 관련이 있을 수도 있다. "시그널링"이란 용어는 일반적으로는 신택스 엘리먼트들 및/또는 압축된 비디오 데이터를 디코딩하는데 사용된 다른 데이터의 통신을 말할 수도 있다. 이러한 통신은 실시간 또는 거의 실시간으로 일어날 수도 있다. 대안으로, 이러한 통신은, 인코딩 시에 신택스 엘리먼트들을 인코딩된 비트스트림으로 컴퓨터 판독가능 저장 매체에 저장하고 그 신택스 엘리먼트들이 이 매체에 저장된 후의 임의의 시간에 디코딩 디바이스에 의해 취출될 수도 있는 경우에 일어날 바와 같이 어떤 기간 (span of time) 에 걸쳐 일어날 수도 있다.

[0046] 몇몇 예들에서, 비디오 인코더 (20) 와 비디오 디코더 (30) 는, ISO/IEC MPEG-4 비주얼 그리고 SVC (Scalable Video Coding) 확장본, MVC (Multi-view Video Coding) 확장본, MVC 기반 3DV 확장본을 포함한 ITU-T H.264 (또한 ISO/IEC MPEG-4 AVC로 알려짐) 와 같은 비디오 압축 표준에 따라 동작한다. 몇몇 경우들에서, H.264/AVC의 MVC 기반 3DV 확장본을 준수하는 임의의 비트스트림은 H.264/AVC의 MVC 확장본을 따르는 서브-비트스트림을 항상 포함한다. 더욱이, H.264/AVC에 대한 3차원 비디오 (3DV) 코딩 확장본, 즉 AVC 기반 3DV를 생성하려는 지속적인 노력이 있다. 다른 예들에서, 비디오 인코더 (20) 및 비디오 디코더 (30) 는 ITU-T H.261, ISO/IEC MPEG-1 비주얼, ITU-T H.262 또는 ISO/IEC MPEG-2 비주얼, 및 ITU-T H.264, ISO/IEC 비주얼에 따라 동작할 수도 있다.

[0047] 다른 예들에서, 비디오 인코더 (20) 및 비디오 인코더 (30) 는, ITU-T 비디오 코딩 전문가 그룹 (Video Coding Experts Group; VCEG) 및 ISO/IEC 동 화상 전문가 그룹 (Motion Picture Experts Group; MPEG) 의 JCT-VC (Joint Collaboration Team on Video Coding) 에 의해 개발된 고 효율 비디오 코딩 (HEVC) 표준에 따라 동작할 수도 있다. "HEVC 규격 초안 8"이라고 지칭되는 HEVC 표준의 초안이, 『Bross et al., "High Efficiency Video Coding (HEVC) text specification draft 8", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 10th Meeting, Stockholm, Sweden, July 2012』에 기재되어

있다. "HEVC 규격 초안 9"라고 지칭되는 HEVC 표준의 다른 초안이, 『Bross et al., "High Efficiency Video Coding (HEVC) text specification draft 9", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 11th Meeting, Shanghai, China, October 2012』에 기재되어 있다. 더욱이, HEVC에 대한 스케일러블 비디오 코딩, 멀티-뷰 코딩, 및 3DV 확장본들을 생성하려는 지속적인 노력이 있다. HEVC의 스케일러블 비디오 코딩 확장본은 SHEVC라고 지칭될 수도 있다.

[0048] 현재, VCEG 및 MPEG의 JCT-3C (Joint Collaboration Team on 3D video Coding) 가 HEVC에 기초하여 3DV 표준을 개발하고 있는데, 표준화 노력의 부분이 HEVC에 기초한 멀티-뷰 비디오 코덱 (MV-HEVC) 의 표준화를 포함하고 다른 부분이 HEVC에 기초한 3D 비디오 코딩 (3D-HEVC) 의 표준화를 포함한다. 3D-HEVC의 경우, 코딩 유닛 및/또는 예측 유닛 레벨에서의 코딩 도구들을 포함한 새로운 코딩 도구들이, 텍스처 뷰 및 깊이 뷰 양쪽 모두에 대해, 포함되고 지원될 수도 있다. 3D-HEVC에 대한 소프트웨어 3D-HTM이 다음의 링크로부터 다운로드될 수 있다: [3D-HTM 버전 6.0]: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-6.0/.

[0049] 3D-HEVC의 규격 초안 뿐만 아니라 참조 소프트웨어 설명이 다음 : 2013년 12월 24일 현재 다음의 링크, 즉, http://phenix.it-sudparis.eu/jct2/doc_end_user/documents/2_Shanghai/wg11/JCT3V-B1005-v1.zip으로부터 다운로드될 수 있는 『Gerhard Tech, Krzysztof Wegner, Ying Chen, Sehoon Yea, "3D-HEVC Test Model Description draft 2", JCT3V-B1005, Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 2nd Meeting: Shanghai, CN, Oct. 2012』로서 입수가능하다. 3D-HEVC의 참조 소프트웨어 설명의 다른 버전이 http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=706 으로부터 입수가능할 수도 있다. "3D-HEVC 테스트 모델 설명 초안 3"이라고 지칭되는 3D-HEVC의 다른 초안이, 『Tech et al., "3D-HEVC Test Model 3", Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 3rd Meeting: Geneva, CH, 17-23 Jan. 2013, document no. JCT3V-C1005_spec_d1』에 기재되어 있다. 비디오 인코더 (20) 및 비디오 디코더 (30) 는 SHEVC, MV-HEVC, 및/또는 3D-HEVC에 따라 동작할 수도 있다.

[0050] HEVC 및 다른 비디오 코딩 사양들에서, 비디오 시퀀스가 일련의 화상들을 통상 포함한다. 화상들은 "프레임들"이라고 또한 지칭될 수도 있다. 화상이 S_L, S_{Cb} 및 S_{Cr}로 표시되는 3 개의 샘플 어레이들을 포함할 수도 있다. S_L은 루마 샘플들의 2차원 어레이 (즉, 블록) 이다. S_{Cb}는 Cb 크로미넌스 샘플들의 2차원 어레이이다. S_{Cr}은 Cr 크로미넌스 샘플들의 2차원 어레이이다. 크로미넌스 샘플들은 본원에서 "크로마" 샘플들이라고 또한 지칭될 수도 있다. 다른 경우들에서, 화상이 모노크롬일 수도 있고 루마 샘플들의 어레이만을 포함할 수도 있다.

[0051] 화상의 인코딩된 표현을 생성하기 위해, 비디오 인코더 (20) 는 코딩 트리 유닛들 (CTU들) 의 세트를 생성할 수도 있다. CTU들의 각각은, 루마 샘플들의 코딩 트리 블록, 크로마 샘플들의 2 개의 대응 코딩 트리 블록들, 및 코딩 트리 블록들의 샘플들을 코딩하는데 사용된 신택스 구조들을 포함할 수도 있다. 모노크롬 화상들 또는 3 개의 별개의 컬러 평면들을 갖는 화상들에서, CTU가 단일 코딩 트리 블록과 그 코딩 트리 블록의 샘플들을 코딩하는데 사용된 신택스 구조들을 포함할 수도 있다. 코딩 트리 블록이 샘플들의 NxN 블록일 수도 있다. CTU가 "트리 블록" 또는 "최대 코딩 유닛 (largest coding unit; LCU)"이라고 또한 지칭될 수도 있다. HEVC의 CTU들은 다른 표준들, 이를테면 H.264/AVC의 매크로블록들과 대체로 유사할 수도 있다. 그러나, CTU가 특정 사이즈로 반드시 제한되는 것은 아니고 하나 이상의 코딩 유닛들 (CU들) 을 포함할 수도 있다. 슬라이스가 래스터 스캔 순서로 연속하여 순서화된 정수 수의 CTU들을 포함할 수도 있다.

[0052] 코딩된 슬라이스가 슬라이스 헤더와 슬라이스 데이터를 포함할 수도 있다. 슬라이스의 슬라이스 헤더는 슬라이스에 관한 정보를 제공하는 신택스 엘리먼트들을 포함하는 신택스 구조일 수도 있다. 슬라이스 데이터는 슬라이스의 코딩된 CTU들을 포함할 수도 있다.

[0053] 본 개시물은 하나 이상의 샘플 블록들 및 그 하나 이상의 샘플 블록들을 코딩하는데 사용된 신택스 구조들을 지칭하기 위해 "비디오 유닛" 또는 "비디오 블록" 또는 "블록"이란 용어를 사용할 수도 있다. 비디오 유닛들의 예의 유형들은 CTU들, CU들, PU들, 변환 유닛들 (TU들), 매크로블록들, 매크로블록 파티션들 등을 포함할 수도 있다. 대체로, 블록의 사이즈는 블록과 연관된 샘플 블록의 사이즈를 지칭할 수도 있다.

[0054] 코딩된 CTU를 생성하기 위해, 비디오 인코더 (20) 는 CTU의 코딩 트리 블록들에 대해 쿼드트리 파티셔닝을 재귀적으로 수행하여 코딩 트리 블록들을 코딩 블록들로 분할할 수도 있으며, 이런 이유로 그 이름이 "코딩 트리 유

닛들"이다. 코딩 블록이 샘플들의 NxN 블록이다. CU는 루마 샘플들의 코딩 블록 및 루마 샘플 어레이, Cb 샘플 어레이 및 Cr 샘플 어레이를 갖는 화상의 크로마 샘플들의 2 개의 대응 코딩 블록들, 및 그 코딩 블록들의 샘플들을 코딩하는데 사용된 신택스 구조들을 포함할 수도 있다. 모노크롬 화상들 또는 3 개의 별개의 컬러 평면들을 갖는 화상들에서, CU가 단일 코딩 블록과 그 코딩 블록의 샘플들을 코딩하는데 사용된 신택스 구조들을 포함할 수도 있다.

[0055] 비디오 인코더 (20) 는 CU의 코딩 블록을 하나 이상의 예측 블록들로 파티셔닝할 수도 있다. 예측 블록은 동일한 예측이 적용되는 샘플들의 직사각형 (즉, 정사각형이거나 또는 정사각형이 아닌) 블록이다. CU의 예측 유닛 (PU) 이 루마 샘플들의 예측 블록, 크로마 샘플들의 2 개의 대응 예측 블록들, 및 그 예측 블록들을 예측하는데 사용된 신택스 구조들을 포함할 수도 있다. 모노크롬 화상들 또는 3 개의 별개의 컬러 평면들을 갖는 화상들에서, PU가 단일 예측 블록과 그 예측 블록을 예측하는데 사용된 신택스 구조들을 포함할 수도 있다. 비디오 인코더 (20) 는 CU의 각각의 PU의 루마, Cb 및 Cr 예측 블록들에 대한 예측 루마, Cb 및 Cr 블록들을 생성할 수도 있다.

[0056] 비디오 인코더 (20) 는 PU에 대한 예측 블록들을 생성하기 위해 인트라 예측 또는 인터 예측을 사용할 수도 있다. 비디오 인코더 (20) 가 PU의 예측 블록들을 생성하기 위해 인트라 예측을 사용하면, 비디오 인코더 (20) 는 그 PU와 연관된 화상의 샘플들에 기초하여 그 PU의 예측 블록들을 생성할 수도 있다. 본 개시물에서, "에 기초하여"라는 어구는 "에 적어도 부분적으로 기초하여"를 나타낼 수도 있다.

[0057] 비디오 인코더 (20) 가 PU의 예측 블록들을 생성하기 위해 인터 예측을 사용하면, 비디오 인코더 (20) 는 그 PU와 연관된 화상 이외의 하나 이상의 화상들의 디코딩된 샘플들에 기초하여 그 PU의 예측 블록들을 생성할 수도 있다. 인터 예측은 단방향 (즉, 단예측 (uni-prediction)) 또는 양방향 (즉, 양예측 (bi-prediction)) 일 수도 있다. 인터 예측을 수행하기 위해, 비디오 인코더 (20) 는 현재 화상에 대한 제 1 참조 화상 리스트 (RefPicList0) 를 생성할 수도 있고 그 현재 화상에 대한 제 2 참조 화상 리스트 (RefPicList1) 를 또한 생성할 수도 있다. 참조 화상 리스트들의 각각은 하나 이상의 참조 화상들을 포함할 수도 있다. 참조 화상 리스트 (즉, 이용가능하면, RefPicList0 및 RefPicList1) 가 구성된 후, 참조 화상 리스트에 대한 참조 인덱스가 참조 화상 리스트에 포함된 임의의 참조 화상을 식별하는데 사용될 수 있다.

[0058] 단예측을 사용하는 경우, 비디오 인코더 (20) 는 참조 화상 내의 참조 로케이션을 결정하기 위해 RefPicList0 및 RefPicList1 중 어느 하나 또는 양쪽 모두에서 참조 화상들을 검색할 수도 있다. 더욱이, 단예측을 사용하는 경우, 비디오 인코더 (20) 는, 참조 로케이션에 대응하는 샘플들에 적어도 부분적으로 기초하여, PU에 대한 예측 블록들을 생성할 수도 있다. 더구나, 단예측을 사용하는 경우, 비디오 인코더 (20) 는 PU의 예측 블록 및 참조 로케이션 사이의 공간적 변위를 나타내는 단일 모션 벡터를 생성할 수도 있다. 그 모션 벡터는 PU의 예측 블록과 참조 로케이션 사이의 수평 변위를 특정하는 수평 성분을 포함할 수도 있고, PU의 예측 블록과 참조 로케이션 사이의 수직 변위를 특정하는 수직 성분을 포함할 수도 있다.

[0059] 양예측을 사용하여 PU를 인코딩하는 경우, 비디오 인코더 (20) 는 RefPicList0에서의 참조 화상의 제 1 참조 로케이션 및 RefPicList1에서의 참조 화상의 제 2 참조 로케이션을 결정할 수도 있다. 비디오 인코더 (20) 는 그 다음에, 제 1 및 제 2 참조 로케이션들에 대응하는 샘플들에 적어도 부분적으로 기초하여, PU에 대한 예측 블록들을 생성할 수도 있다. 더구나, 양예측을 사용하여 PU를 인코딩하는 경우, 비디오 인코더 (20) 는, PU의 예측 블록 및 제 1 참조 로케이션 사이의 공간적 변위를 나타내는 제 1 모션 벡터와 PU의 예측 블록 및 제 2 참조 로케이션 사이의 공간적 변위를 나타내는 제 2 모션 벡터를 생성할 수도 있다.

[0060] 비디오 인코더 (20) 가 CU의 하나 이상의 PU들에 대해 하나 이상의 예측 블록들 (예컨대, 루마, Cb 및 Cr 예측 블록들) 을 생성한 후, 비디오 인코더 (20) 는 그 CU에 대해 하나 이상의 잔차 블록들을 생성할 수도 있다. 예를 들면, 비디오 인코더 (20) 는 CU에 대해 루마 잔차 블록을 생성할 수도 있다. CU의 루마 잔차 블록에서의 각각의 샘플은 CU의 예측 루마 블록들 중 하나의 예측 루마 블록에서의 루마 샘플과 CU의 원래의 루마 코딩 블록에서의 대응하는 샘플 사이의 차이를 나타낸다. 덧붙여서, 비디오 인코더 (20) 는 CU에 대한 Cb 잔차 블록을 생성할 수도 있다. CU의 Cb 잔차 블록에서의 각각의 샘플은 CU의 예측 Cb 블록들 중 하나의 예측 Cb 블록에서의 Cb 샘플과 CU의 원래의 Cb 코딩 블록에서의 대응하는 샘플 사이의 차이를 나타낼 수도 있다. 비디오 인코더 (20) 는 CU에 대한 Cr 잔차 블록을 또한 생성할 수도 있다. CU의 Cr 잔차 블록에서의 각각의 샘플은 CU의 예측 Cr 블록들 중 하나의 예측 Cr 블록에서의 Cr 샘플과 CU의 원래의 Cr 코딩 블록에서의 대응하는 샘플 사이의 차이를 나타낼 수도 있다.

[0061] 더욱이, 비디오 인코더 (20) 는 쿼드트리 파티셔닝을 사용하여 CU의 잔차 블록들 (예컨대, 그 CU의 루마, Cb 및

Cr 잔차 블록들) 을 하나 이상의 변환 블록들 (예컨대, 루마, Cb 및 Cr 변환 블록들) 로 분해할 수도 있다. 동일한 변환이 적용되는 샘플들의 직사각형 (예컨대, 정사각형이거나 또는 정사각형이 아닌) 블록이 변환 블록이다. CU의 변환 유닛 (TU) 이 루마 샘플들의 변환 블록, 크로마 샘플들의 2 개의 대응 변환 블록들, 및 그 변환 블록 샘플들을 변환하는데 사용된 신택스 구조들을 포함할 수도 있다. 따라서, CU의 각각의 TU는 루마 변환 블록, Cb 변환 블록, 및 Cr 변환 블록과 연관될 수도 있다. TU와 연관된 루마 변환 블록은 CU의 루마 잔차 블록의 서브-블록일 수도 있다. Cb 변환 블록은 CU의 Cb 잔차 블록의 서브-블록일 수도 있다. Cr 변환 블록은 CU의 Cr 잔차 블록의 서브-블록일 수도 있다. 모노크롬 화상들 또는 3 개의 별개의 컬러 평면들을 갖는 화상들에서, TU가 단일 변환 블록과 그 변환 블록의 샘플들을 변환하는데 사용된 신택스 구조들을 포함할 수도 있다.

[0062] 비디오 인코더 (20) 는 하나 이상의 변환들을 TU의 변환 블록에 적용하여 TU에 대한 계수 블록을 생성할 수도 있다. 예를 들면, 비디오 인코더 (20) 는 하나 이상의 변환들을 TU의 루마 변환 블록에 적용하여 그 TU에 대한 루마 계수 블록을 생성할 수도 있다. 계수 블록이 변환 계수들의 2차원 어레이일 수도 있다. 변환 계수가 스칼라 양일 수도 있다. 비디오 인코더 (20) 는 하나 이상의 변환들을 TU의 Cb 변환 블록에 적용하여 TU에 대한 Cb 계수 블록을 생성할 수도 있다. 비디오 인코더 (20) 는 하나 이상의 변환들을 TU의 Cr 변환 블록에 적용하여 TU에 대한 Cr 계수 블록을 생성할 수도 있다.

[0063] 계수 블록 (예컨대, 루마 계수 블록, Cb 계수 블록 또는 Cr 계수 블록) 을 생성한 후, 비디오 인코더 (20) 는 그 계수 블록을 양자화할 수도 있다. 양자화는 변환 계수들이 그 변환 계수들을 표현하는데 사용된 데이터의 양을 가능한 한 줄이도록 양자화되어서, 추가의 압축을 제공하는 프로세스를 일반적으로 지칭한다. 비디오 인코더 (20) 가 계수 블록을 양자화한 후, 비디오 인코더 (20) 는 양자화된 변환 계수들을 나타내는 신택스 엘리먼트들을 엔트로피 인코딩할 수도 있다. 예를 들어, 비디오 인코더 (20) 는 양자화된 변환 계수들을 나타내는 신택스 엘리먼트들에 대해 콘텍스트 적응 이진 산술 코딩 (Context-Adaptive Binary Arithmetic Coding; CABAC) 을 수행할 수도 있다.

[0064] 비디오 인코더 (20) 는 코딩된 화상들의 표현 및 연관된 데이터를 형성하는 비트들의 시퀀스를 포함하는 비트스트림을 출력할 수도 있다. 그 비트스트림은 네트워크 추상화 계층 (network abstraction layer; NAL) 유닛들의 시퀀스를 포함할 수도 있다. NAL 유닛이, NAL 유닛에서의 데이터의 유형의 표시 (indication) 와 에뮬레이션 방지 바이트들이 필요에 따라 점재된 (interspersed) RBSP (raw byte sequence payload) 형태로 당해 데이터를 포함한 바이트들을 포함하는 신택스 구조이다. NAL 유닛들의 각각은 NAL 유닛 헤더를 포함하고 RBSP를 캡슐화한다. NAL 유닛 헤더는 NAL 유닛 유형 코드를 나타내는 신택스 엘리먼트를 포함할 수도 있다. NAL 유닛의 NAL 유닛 헤더에 의해 특정된 NAL 유닛 유형 코드는 NAL 유닛의 유형을 나타낸다. RBSP가 NAL 유닛 내에 캡슐화되는 정수 수의 바이트들을 포함하는 신택스 구조일 수도 있다. 일부 경우들에서, RBSP가 0 비트들을 포함한다.

[0065] 상이한 유형들의 NAL 유닛들이 상이한 유형들의 RBSP들을 캡슐화할 수도 있다. 예를 들어, NAL 유닛의 상이한 유형들은 비디오 파라미터 세트들 (VPS들), 시퀀스 파라미터 세트들 (SPS들), 화상 파라미터 세트들 (PPS들), 코딩된 슬라이스들, 추가 향상 정보 (supplemental enhancement information; SEI) 등에 대해 상이한 RBSP들을 캡슐화할 수도 있다. 비디오 코딩 데이터에 대한 RBSP들 (파라미터 세트들 및 SEI 메시지들에 대한 RBSP과는 대조적임) 을 캡슐화하는 NAL 유닛들은, 비디오 코딩 계층 (video coding layer; VCL) NAL 유닛들이라고 지칭될 수도 있다.

[0066] HEVC에서, SPS들은 코딩된 비디오 시퀀스 (coded video sequence; CVS) 의 모든 슬라이스들에 적용되는 정보를 포함할 수도 있다. HEVC에서, CVS가, 순간적 디코딩 리프레시 (instantaneous decoding refresh; IDR) 화상, 또는 깨진 링크 액세스 (broken link access; BLA) 화상, 또는 비트스트림에서의 첫 번째 화상인 깨끗한 랜덤 액세스 (clean random access; CRA) 화상으로부터 시작하여, IDR 또는 BLA 화상이 아닌 모든 후속 화상들을 포함할 수도 있다. 다시 말하면, HEVC에서, CVS가, 디코딩 순서로, 비트스트림에서의 첫 번째 액세스 유닛인 CRA 액세스 유닛, IDR 액세스 유닛 또는 BLA 액세스 유닛, 후속하여 임의의 후속 IDR 또는 BLA 액세스 유닛 전까지를 포함하는 모든 후속 액세스 유닛들을 포함하는 0 이상의 비-IDR 및 비-BLA 액세스 유닛들로 이루어질 수도 있는 액세스 유닛들의 시퀀스를 포함할 수도 있다.

[0067] 0 이상의 전체 CVS들에 적용되는 신택스 엘리먼트들을 포함하는 신택스 구조가 VPS이다. SPS는 그 SPS가 액티브인 경우 액티브가 되는 VPS를 식별하는 신택스 엘리먼트를 포함할 수도 있다. 따라서, VPS의 신택스 엘리먼트들은 SPS의 신택스 엘리먼트들보다 더 일반적으로 적용가능할 수도 있다. 0 이상의 코딩된 화상들에

적용되는 신택스 엘리먼트들을 포함하는 신택스 구조가 PPS이다. PPS는 그 PPS가 액티브인 경우 액티브가 되는 SPS를 식별하는 신택스 엘리먼트를 포함할 수도 있다. 슬라이스의 슬라이스 헤더가 그 슬라이스가 코딩되고 있는 경우 액티브인 PPS를 나타내는 신택스 엘리먼트를 포함할 수도 있다.

[0068] 비디오 디코더 (30) 는 비디오 인코더 (20) 에 의해 생성된 비트스트림을 수신할 수도 있다. 덧붙여서, 비디오 디코더 (30) 는 비트스트림으로부터 신택스 엘리먼트들을 획득하기 위해 그 비트스트림을 파싱할 수도 있다. 비디오 디코더 (30) 는 비트스트림으로부터 획득된 신택스 엘리먼트들에 적어도 부분적으로 기초하여 비디오 데이터의 화상들을 복원할 수도 있다. 비디오 데이터를 복원하는 프로세스는 비디오 인코더 (20) 에 의해 수행된 프로세스에 일반적으로 역일 수도 있다. 예를 들면, 비디오 디코더 (30) 는 현재 CU의 PU들에 대한 예측 블록들을 결정하기 위해 그 PU들의 모션 벡터들을 사용할 수도 있다. 덧붙여서, 비디오 디코더 (30) 는 현재 CU의 TU들과 연관된 계수 블록들을 역 양자화할 수도 있다. 비디오 디코더 (30) 는 현재 CU의 TU들과 연관된 변환 블록들을 복원하기 위해 계수 블록들에 대해 역 변환들을 수행할 수도 있다. 비디오 디코더 (30) 는 현재 CU의 PU들에 대한 예측 블록들의 샘플들을 현재 CU의 TU들의 변환 블록들의 대응하는 샘플들에 더함으로써 현재 CU의 코딩 블록들을 복원할 수도 있다. 화상의 각각의 CU에 대한 코딩 블록들을 복원함으로써, 비디오 디코더 (30) 는 그 화상을 복원할 수도 있다.

[0069] 위에서 나타낸 바와 같이, CU가 하나 이상의 PU들로 파티셔닝될 수도 있다. "part_mode"라는 용어는 현재 CU의 파티셔닝 모드를 나타낼 수도 있다. 다르게 말하면, part_mode는 CU가 PU들로 파티셔닝되는 방식을 나타낼 수도 있다. HEVC에서, part_mode의 값은 다음과 같이 제한될 수도 있다:

[0070] • 하나의 CU의 코딩 모드가 MODE_INTRA와 동일하면, part_mode는 0 또는 1과 동일할 것이다. CU의 코딩 모드는 CU가 인트라 예측을 사용하여 코딩되는지 (즉, MODE_INTRA인지) 또는 인터 예측을 사용하여 코딩되는지 (즉, MODE_INTER인지) 를 나타낼 수도 있다.

[0071] • 그렇지 않으면 (하나의 CU의 코딩 모드가 MODE_INTER와 동일하면), 다음이 적용될 수도 있다:

[0072] • 현재 CU의 사이즈가 최소 허용가능 CU의 사이즈보다 더 크고 비대칭 모션 파티셔닝이 가능하게 되면, part_mode는 0 내지 2의 범위 내에 그리고 4 내지 7의 범위 내에 있을 것이다. 비대칭 모션 파티셔닝이 가능하게 되는 경우, 현재 CU는 동일한 사이즈를 각각 갖지 않는 PU들로 파티셔닝될 수도 있다.

[0073] • 그렇지 않고, 현재 CU의 사이즈가 최소 CU의 사이즈보다 더 크고 비대칭 모션 파티션이 불가능하게 되면, part_mode는 0 내지 2의 범위 내에 있을 것이다. CU의 사이즈는 CU의 코딩 블록 (예컨대, 루마 코딩 블록) 의 사이즈를 지칭할 수도 있다.

[0074] • 그렇지 않고, 현재 CU의 사이즈가 8과 동일하면, part_mode의 값은 0 내지 2의 범위 내에 있을 것이다.

[0075] • 그렇지 않으면 (현재 CU의 사이즈가 8보다 더 크면), part_mode의 값은 0 내지 3의 범위 내에 있을 것이다.

[0076] part_mode와 변수 PartMode의 연관된 값 사이의 관계는 아래의 표 1에 정의되어 있다. 변수 PartMode는 파티셔닝 모드의 인간 판독가능 이름을 제공한다. part_mode라는 용어는 표 1과 일관되는, 코딩된 비트스트림에서의 인덱스 값을 정의할 수도 있는데, 그 인덱스 값은 표 1에서 변수 PartMode에 의해 규정된 정의에 매핑된다. 몇몇 예들에서, part_mode가 존재하지 않는 경우, 현재 CU의 파티션 모드는 PART_2Nx2N과 동일한 것으로 유추된다.

표 1

[0077] **예측 모드 및 파티셔닝 유형에 대한 이름 연관성**

하나의 CU의 코딩 모드	part_mode	PartMode
MODE_INTRA	0	PART_2Nx2N
	1	PART_NxN

MODE_INTER	0	PART_2Nx2N
	1	PART_2NxN
	2	PART_Nx2N
	3	PART_NxN
	4	PART_2NxnU
	5	PART_2NxnD
	6	PART_nLx2N
	7	PART_nRx2N

[0078] H.264/AVC에서, 비디오 시퀀스가 일련의 비디오 프레임들을 통상 포함한다. 더욱이, H.264/AVC에서, 화상들의 그룹 (GOP) 이 일련의 하나 이상의 비디오 프레임들을 일반적으로 포함한다. GOP가 신택스 데이터를 GOP의 헤더, GOP의 하나 이상의 프레임들의 헤더, 또는 GOP에 포함된 프레임들의 수를 기술하는 다른 곳에 포함할 수도 있다. 각각의 프레임은 개별 프레임에 대한 인코딩 모드를 기술하는 프레임 신택스 데이터를 포함할 수도 있다. 비디오 인코더 (20) 는 비디오 데이터를 인코딩하기 위하여 개개의 비디오 프레임들 내의 블록들에 대해 통상 동작한다. H.264/AVC에서, 블록이 매크로블록 (MB) 또는 매크로블록의 파티션에 대응할 수도 있다. MB가 3 개의 샘플 어레이들을 갖는 화상의 루마 샘플들의 16x16 블록 및 크로마 샘플들의 2 개의 대응하는 블록들, 또는 모노크롬 화상 또는 3 개의 별개의 컬러 평면들을 사용하여 코딩된 화상의 샘플들의 16x16 블록이다. MB 파티션이, 3 개의 샘플 어레이들을 갖는 화상에 대한 인터 예측을 위한 매크로블록의 파티셔닝의 결과로 생긴 루마 샘플들의 블록 및 크로마 샘플들의 2 개의 대응하는 블록들, 또는 모노크롬 화상 또는 3 개의 별개의 컬러 평면들을 사용하여 코딩된 화상의 인터 예측을 위한 매크로블록의 파티셔닝의 결과로 생긴 루마 샘플들의 블록이다. HEVC에서, 블록이 PU에 해당할 수도 있다. 블록들 (예컨대, 비디오 유닛들) 과 연관된 샘플 블록들은 고정된 또는 가변하는 사이즈들을 가질 수도 있고, 특정된 코딩 표준에 따라 사이즈가 상이할 수도 있다.

[0079] H.264/AVC에서, 비디오 인코더 (20) 는 하나 이상의 참조 화상들에 기초하여 인터 MB에 대한 예측 블록을 생성한다. 각각의 인터 MB는 4 개의 상이한 방도들로 파티셔닝될 수도 있다:

[0080] • 하나의 16x16 MB 파티션

[0081] • 2 개의 16x8 MB 파티션들

[0082] • 2 개의 8x16 MB 파티션들

[0083] • 4 개의 8x8 MB 파티션들

[0084] 하나의 MB에서의 상이한 MB 파티션들은 각각의 방향 (즉, RefPicList0 또는 RefPicList1) 에 대해 상이한 참조 인덱스들을 가질 수도 있다. 따라서, 비디오 코더가, 상이한 참조 화상들에 기초하여, 하나의 인터 MB의 상이한 MB 파티션들에 대한 예측 블록들을 생성할 수도 있다. 인터 MB가 4 개의 8x8 MB 파티션들로 파티셔닝되지 않은 경우, 그 인터 MB는 각각의 방향에서 전체 MB 파티션에 대해 단지 하나의 모션 벡터를 가질 수도 있다. 다르게 말하면, 인터 MB가 4 개의 8x8 MB 파티션들로 파티셔닝되지 않은 경우, 인터 MB에 대한 단일 RefPicList0 모션 벡터 및 인터 MB에 대한 단일 RefPicList1 모션 벡터만이 있을 수도 있다.

[0085] MB가 4 개의 8x8 MB 파티션들로 파티셔닝되는 경우, 각각의 8x8 MB 파티션은 서브-블록들로 추가로 파티셔닝될 수 있다. 8x8 MB 파티션을 서브-블록들로 파티셔닝하는 4 개의 상이한 방도들이 있다:

[0086] • 하나의 8x8 서브-블록

[0087] • 2 개의 8x4 서브-블록들

[0088] • 2 개의 4x8 서브-블록들

[0089] • 4 개의 4x4 서브-블록들

[0090] 각각의 서브블록은 각각의 방향에서 상이한 모션 벡터를 가질 수 있다. 다르게 말하면, B 슬라이스에서의 각각의 서브-블록이 RefPicList0 모션 벡터와 RefPicList1 모션 벡터를 가질 수도 있다. "서브-블록

파티션"은 8x8 MB 파티션이 서브-블록들로 파티셔닝되는 방법을 나타내는데 사용되는 용어이다.

[0091] HEVC에서, 비디오 인코더 (20) 는 병합 모드 또는 고급 모션 벡터 예측 (AMVP) 모드를 사용하여 PU의 모션 정보를 시그널링할 수도 있다. 다르게 말하면, HEVC에서는, 모션 파라미터들 (즉, 모션 정보) 의 예측을 위해 하나는 병합 모드이고 다른 하나는 AMVP인 2 개의 모드들이 있다. 모션 예측은 하나 이상의 다른 블록들의 모션 정보에 기초한 블록 (예컨대, PU) 의 모션 정보의 결정을 포함할 수도 있다. PU의 모션 정보는 PU의 모션 벡터 (들), PU의 참조 인덱스 (들), 및 하나 이상의 예측 방향 표시자들을 포함할 수도 있다.

[0092] 비디오 인코더 (20) 가 병합 모드를 사용하여 현재 PU의 모션 정보를 시그널링하는 경우, 비디오 인코더 (20) 는 병합 후보 리스트를 생성한다. 다르게 말하면, 비디오 인코더 (20) 는 모션 벡터 예측자 (predictor) 리스트 구축 프로세스를 수행할 수도 있다. 병합 후보 리스트는 현재 PU에 공간적으로 또는 시간적으로 이웃하는 PU들의 모션 정보를 나타내는 병합 후보들의 세트를 포함한다. 다시 말하면, 병합 모드에서, 후보가 공간적 및 시간적 이웃 블록들로부터 형성될 수 있는 모션 파라미터들 (예컨대, 참조 인덱스들, 모션 벡터들 등) 의 후보 리스트가 구축된다.

[0093] 더욱이, 병합 모드에서, 비디오 인코더 (20) 는 병합 후보 리스트로부터 병합 후보를 선택할 수도 있고 선택된 병합 후보에 의해 나타내어진 모션 정보를 현재 PU의 모션 정보로서 사용할 수도 있다. 비디오 인코더 (20) 는 선택된 병합 후보의 병합 후보 리스트에서의 포지션을 시그널링할 수도 있다. 예를 들면, 비디오 인코더 (20) 는 인덱스를 후보 리스트로 송신함으로써 선택된 모션 벡터 파라미터들을 시그널링할 수도 있다. 비디오 디코더 (30) 는, 비트스트림으로부터, 후보 리스트에 대한 인덱스 (즉, 후보 리스트 인덱스) 를 획득할 수도 있다. 덧붙여서, 비디오 디코더 (30) 는 동일한 병합 후보 리스트를 생성할 수도 있고 선택된 병합 후보의 포지션의 표시에 기초하여, 선택된 병합 후보를 결정할 수도 있다. 비디오 디코더 (30) 는 그 다음에 선택된 병합 후보의 모션 정보를 사용하여 현재 PU에 대한 예측 블록들을 생성할 수도 있다. 다시 말하면, 비디오 디코더 (30) 는, 후보 리스트 인덱스에 적어도 부분적으로 기초하여, 후보 리스트에서 선택된 후보를 결정할 수도 있는데, 선택된 후보는 현재 PU에 대한 모션 벡터를 특정한다. 이런 식으로, 디코더 측에서, 일단 인덱스가 디코딩되면, 인덱스가 가리키는 대응 블록의 모든 모션 파라미터들은 현재 PU에 의해 인계될 수도 있다.

[0094] 스킵 모드는 병합 모드와 유사하다. 스킵 모드에서, 비디오 인코더 (20) 와 비디오 디코더 (30) 는 비디오 인코더 (20) 와 비디오 디코더 (30) 가 병합 모드에서 병합 후보 리스트를 사용하는 것과 동일한 방식으로 병합 후보 리스트를 생성하고 사용한다. 그러나, 비디오 인코더 (20) 가 스킵 모드를 사용하여 현재 PU의 모션 정보를 시그널링하는 경우, 비디오 인코더 (20) 는 현재 PU에 대해 임의의 잔차 데이터를 시그널링하지 않는다. 따라서, 비디오 디코더 (30) 는, 잔차 데이터의 사용 없이, 병합 후보 리스트에서의 선택된 후보의 모션 정보에 의해 나타내어진 참조 블록에 기초하여 PU에 대한 예측 블록을 결정할 수도 있다.

[0095] AMVP 모드는 비디오 인코더 (20) 가 후보 리스트를 생성할 수도 있고 그 후보 리스트로부터 후보를 선택할 수도 있다는 점에서 병합 모드와 유사하다. 그러나, 비디오 인코더 (20) 가 AMVP 모드를 사용하여 현재 PU의 RefPicListX (여기서 X는 0 또는 1) 모션 정보를 시그널링하는 경우, 비디오 인코더 (20) 는 현재 PU에 대한 RefPicListX 모션 벡터 예측자 (motion vector predictor; MVP) 플래그를 시그널링하는 것에 더하여 현재 PU에 대한 RefPicListX 모션 벡터 차이 (motion vector difference; MVD) 와 현재 PU에 대한 RefPicListX 참조 인덱스를 시그널링할 수도 있다. 현재 PU에 대한 RefPicListX MVP 플래그는 AMVP 후보 리스트에서의 선택된 AMVP 후보의 포지션을 나타낼 수도 있다. 현재 PU에 대한 RefPicListX MVD는 현재 PU의 RefPicListX 모션 벡터와 선택된 AMVP 후보의 모션 벡터 사이의 차이를 나타낼 수도 있다. 이런 식으로, 비디오 인코더 (20) 는 RefPicListX MVP 플래그, RefPicListX 참조 인덱스 값, 및 RefPicListX MVD를 시그널링함으로써 현재 PU의 RefPicListX 모션 정보를 시그널링할 수도 있다. 다르게 말하면, 현재 PU에 대한 모션 벡터를 나타내는 비트스트림에서의 데이터는 참조 인덱스, 후보 리스트에 대한 인덱스, 및 MVD를 표현하는 데이터를 포함할 수도 있다.

[0096] 더욱이, 현재 PU의 모션 정보가 AMVP 모드를 사용하여 시그널링되는 경우, 비디오 디코더 (30) 는, 비트스트림으로부터, 현재 PU에 대한 MVD와 MVP 플래그를 획득할 수도 있다. 비디오 디코더 (30) 는 동일한 AMVP 후보 리스트를 생성할 수도 있고, MVP 플래그에 기초하여, 선택된 AMVP 후보를 결정할 수도 있다. 비디오 디코더 (30) 는 MVD를 선택된 AMVP 후보에 의해 나타내어진 모션 벡터에 더함으로써 현재 PU의 모션 벡터를 복구할 수도 있다. 다시 말하면, 비디오 디코더 (30) 는, 선택된 AMVP 후보 및 MVD에 의해 나타내어진 모션 벡터에 기초하여, 현재 PU의 모션 벡터를 결정할 수도 있다. 비디오 디코더 (30) 는 그 다음에 현재 PU의 복구된 모션 벡터 또는 모션 벡터들을 사용하여 현재 PU에 대한 예측 블록들을 생성할 수도 있다.

[0097] 비디오 디코더 (30) 가 현재 PU에 대한 AMVP 후보 리스트를 생성하는 경우, 비디오 디코더 (30) 는 현재 PU에 공간적으로 이웃하는 로케이션들을 커버하는 PU들 (즉, 공간적으로 이웃하는 PU들) 의 모션 정보에 기초하여 하나 이상의 AMVP 후보들을 도출할 수도 있다. 도 2는 현재 PU (40) 에 관하여 예의 공간적으로 이웃하는 PU들을 도시하는 개념도이다. 도 2의 예에서, 공간적으로 이웃하는 PU들은 A_0, A_1, B_0, B_1, 및 B_2로서 나타내어진 로케이션들을 커버하는 PU들일 수도 있다. PU의 예측 블록이 한 로케이션을 포함하는 경우 그 PU는 그 로케이션을 커버할 수도 있다.

[0098] 현재 PU에 시간적으로 이웃하는 PU (즉, 현재 PU와는 상이한 시간 인스턴스에 있는 PU) 의 모션 정보에 기초하는 병합 후보 리스트 또는 AMVP 후보 리스트에서의 후보가 시간적 모션 벡터 예측자 (temporal motion vector predictor; TMVP) 라고 지칭될 수도 있다. TMVP들은 HEVC의 코딩 효율을 개선하는데 사용될 수도 있고, 다른 코딩 도구들과는 달리, TMVP들은 디코딩된 화상 버퍼에서의, 더 구체적으로는 참조 화상 리스트에서의 프레임의 모션 벡터에 액세스할 필요가 있을 수도 있다.

[0099] TMVP들의 사용은 CVS 단위 기반으로, 슬라이스 단위 기반으로, 또는 다른 기반으로 가능 또는 불능이 될 수도 있다. SPS에서의 신택스 엘리먼트 (예컨대, sps_temporal_mvp_enable_flag) 가 TMVP들의 사용이 CVS에 대해 가능한지의 여부를 나타낼 수도 있다. 더욱이, TMVP들의 사용이 CVS에 대해 가능하게 되는 경우, TMVP들의 사용은 CVS 내의 특정 슬라이스들에 대해 가능 또는 불능이 될 수도 있다. 예를 들면, 슬라이스 헤더에서의 신택스 엘리먼트 (예컨대, slice_temporal_mvp_enable_flag) 가 TMVP들의 사용이 슬라이스에 대해 가능한지의 여부를 나타낼 수도 있다. 따라서, 인터 예측된 슬라이스에서, TMVP가 전체 CVS에 대해 가능하게 되는 (예컨대, SPS에서의 sps_temporal_mvp_enable_flag가 1로 설정되는) 경우, slice_temporal_mvp_enable_flag는 TMVP들의 사용이 현재 슬라이스에 대해 가능한지의 여부를 나타내기 위해 슬라이스 헤더에서 시그널링된다.

[0100] TMVP를 결정하기 위해, 비디오 코더가 현재 PU와 병치되는 PU를 포함하는 참조 화상을 맨 먼저 식별할 수도 있다. 다르게 말하면, 비디오 코더는 이른바 병치된 화상을 식별할 수도 있다. 현재 화상의 현재 슬라이스가 B 슬라이스 (즉, 양 방향으로 인터 예측된 PU들을 포함하는 것이 허용된 슬라이스) 이면, 비디오 인코더 (20) 는, 슬라이스 헤더에서, 병치된 화상이 RefPicList0으로부터 유래하는지 또는 RefPicList1로부터 유래하는지를 나타내는 신택스 엘리먼트 (예컨대, collocated_from_l0_flag) 를 시그널링할 수도 있다. 다르게 말하면, TMVP들의 사용이 현재 슬라이스에 대해 가능하게 되고 현재 슬라이스가 B 슬라이스 (예컨대, 양 방향으로 인터 예측된 PU들을 포함하는 것이 허용된 슬라이스) 인 경우, 비디오 인코더 (20) 는 병치된 화상이 RefPicList0에 있는지 또는 RefPicList1에 있는지를 나타내는 신택스 엘리먼트 (예컨대, collocated_from_l0_flag) 를 슬라이스 헤더에서 시그널링할 수도 있다. 비디오 디코더 (30) 가 병치된 화상을 포함하는 참조 화상 리스트를 식별한 후, 비디오 디코더 (30) 는 슬라이스 헤더에서 시그널링될 수도 있는 다른 신택스 엘리먼트 (예컨대, collocated_ref_idx) 를 사용하여, 식별된 참조 화상 리스트에서 화상 (즉, 병치된 화상) 을 식별할 수도 있다. 다시 말하면, 참조 화상 리스트가 식별된 후, 슬라이스 헤더에서 시그널링된 collocated_ref_idx는 참조 화상 리스트에서 화상을 식별하는데 사용될 수도 있다.

[0101] 비디오 코더는 병치된 화상을 체킹함으로써 병치된 PU를 식별할 수도 있다. 비디오 코더가 병치된 PU를 체킹하는 경우, 비디오 코더는 병치된 화상의 우측하단 PU와 병치된 화상의 중앙 PU를 체킹할 수도 있다. 우측하단 PU는 현재 PU의 예측 블록의 우측하부 코너의 바로 우측 아래의 로케이션과 병치된 로케이션을 커버할 수도 있다. 중앙 PU는 현재 PU의 예측 블록의 중앙과 병치된 로케이션을 커버할 수도 있다. TMVP는 우측하단 PU 또는 중앙 PU의 모션 정보를 나타낼 수도 있다.

[0102] 위의 프로세스에 의해 식별된 모션 벡터들 (즉, TMVP의 모션 벡터들) 이 병합 모드 또는 AMVP 모드에 대한 모션 후보를 생성하는데 사용되는 경우, 비디오 코더는 (POC 값에 의해 반영된) 시간적 로케이션에 기초하여 모션 벡터들을 스케일링할 수도 있다. 예를 들면, 비디오 코더가, 현재 화상 및 참조 화상의 POC 값들 간의 차이가 작은 경우보다는 현재 화상 및 참조 화상의 POC 값들 간의 차이가 더 큰 경우에, 더 큰 양만큼 모션 벡터의 크기를 증가시킬 수도 있다.

[0103] TMVP로부터 도출된 시간적 병합 후보에 대한 모든 가능한 참조 화상 리스트들의 타겟 참조 인덱스는 항상 0으로 설정될 수도 있다. 그러나, AMVP의 경우, 모든 가능한 참조 화상들의 타겟 참조 인덱스는 디코딩된 참조 인덱스와 동일하게 설정된다. HEVC에서, SPS가 플래그 (예컨대, sps_temporal_mvp_enable_flag) 를 포함할 수도 있고 슬라이스 헤더는 sps_temporal_mvp_enable_flag가 1과 동일한 경우 플래그 (예컨대, pic_temporal_mvp_enable_flag) 를 포함할 수도 있다. pic_temporal_mvp_enable_flag 및 temporal_id 양쪽 모두가 특정 화상에 대해 0과 동일한 경우, 디코딩 순서에서 그 특정 화상 전의 화상들로부터의 모션 벡터는 특

정 화상의 또는 디코딩 순서에서 특정 화상 후의 화상의 디코딩에 있어서 TMVP로서 사용되지 않는다.

[0104] 멀티-뷰 코딩에서는, 동일한 장면의 상이한 관점들로부터의 다수의 뷰들이 있을 수도 있다. "액세스 유닛"이라는 용어는 동일한 시간 인스턴스에 대응하는 화상들의 세트를 지칭하는데 사용된다. 따라서, 비디오 데이터는 시간이 지남에 따라 발생하는 일련의 액세스 유닛들로서 개념화될 수도 있다. "뷰 성분"이 단일 액세스 유닛에서의 뷰의 코딩된 표현일 수도 있다. 뷰 성분은 텍스처 뷰 성분과 깊이 뷰 성분을 포함할 수도 있다. 본 개시물에서, "뷰"는 동일한 뷰 식별자와 연관된 뷰 성분들의 시퀀스를 지칭할 수도 있다.

[0105] 텍스처 뷰 성분 (즉, 텍스처 화상) 이 단일 액세스 유닛에서의 뷰의 텍스처의 코딩된 표현일 수도 있다. 텍스처 뷰가 뷰 순서 인덱스의 동일한 값과 연관된 텍스처 뷰 성분들의 시퀀스일 수도 있다. 뷰의 뷰 순서 인덱스가 그 뷰의 다른 뷰들에 대한 카메라 포지션을 나타낼 수도 있다. 깊이 뷰 성분 (즉, 깊이 화상) 이 단일 액세스 유닛에서의 뷰의 깊이의 코딩된 표현일 수도 있다. 깊이 뷰가 뷰 순서 인덱스의 동일한 값과 연관된 깊이 뷰 성분들의 시퀀스일 수도 있다.

[0106] 멀티-뷰 코딩은 뷰 간 예측을 지원한다. 뷰 간 예측은 HEVC에서 사용된 인터 예측과 유사하고, 동일한 신택스 엘리먼트들을 사용할 수도 있다. 그러나, 비디오 코더가 현재 블록 (이를테면 PU) 에 대해 뷰 간 예측을 수행하는 경우, 비디오 인코더 (20) 는, 참조 화상으로서, 현재 블록과 동일한 액세스 유닛에 있지만 상이한 뷰에 있는 화상을 사용할 수도 있다. 그 반면, 기존의 인터 예측은 상이한 액세스 유닛들에서의 화상들만을 참조 화상들로서 사용한다.

[0107] 텍스처 뷰 성분은 디스플레이되는 실제 이미지 콘텐츠를 포함한다. 예를 들어, 텍스처 뷰 성분은 루마 (예컨대, Y) 및 크로마 (예컨대, Cb 및 Cr) 성분들을 포함할 수도 있다. 깊이 뷰 성분은 그것의 대응하는 텍스처 뷰 성분에서의 화소들의 상대 깊이들을 나타낼 수도 있다. 하나의 예로서, 깊이 뷰 성분은 루마 값들만을 포함하는 그레이 스케일 이미지이다. 다르게 말하면, 깊이 뷰 성분은 임의의 이미지 콘텐츠를 전달하지 않고, 오히려 텍스처 뷰 성분에서의 화소들의 상대 깊이들의 측정치를 제공할 수도 있다.

[0108] 멀티-뷰 코딩에서, 비트스트림이 복수의 계층들을 가질 수도 있다. 그 계층들의 각각은 상이한 뷰에 대응할 수도 있다. 멀티-뷰 코딩에서, 비디오 디코더 (예컨대, 비디오 디코더 (30)) 가 뷰에서의 화상들을 임의의 다른 뷰에서의 화상들에 대한 참조 없이 디코딩할 수 있다면, 그 뷰는 "기본 뷰"라고 지칭될 수도 있다. 뷰의 디코딩이 하나 이상의 다른 뷰들에서의 화상들의 디코딩에 의존한다면 그 뷰는 비-기본 뷰 (non-base view) 라고 지칭될 수도 있다.

[0109] 예를 들면, NAL 유닛들이 헤더들 (즉, NAL 유닛 헤더들) 과 페이로드들 (예컨대, RBSP들) 을 포함할 수도 있다. NAL 유닛 헤더들은 nuh_reserved_zero_6bits 신택스 엘리먼트들을 포함할 수도 있다. 상이한 값들을 특정하는 nuh_reserved_zero_6bit 신택스 엘리먼트들을 갖는 NAL 유닛들은 비트스트림의 상이한 "계층들"에 속한다. 따라서, 멀티-뷰 코딩, 3DV, 또는 SVC에서, NAL 유닛의 nuh_reserved_zero_6bits 신택스 엘리먼트는 NAL 유닛의 계층 식별자 (즉, 계층 ID) 를 특정한다. 몇몇 예들에서, NAL 유닛이 멀티-뷰 코딩, 3DV 코딩, 또는 SVC에서의 기본 계층에 관련되면, 그 NAL 유닛의 nuh_reserved_zero_6bits 신택스 엘리먼트는 0과 동일하다. 비트스트림의 기본 계층에서의 데이터는 그 비트스트림의 임의의 다른 계층에서의 데이터에 대한 참조 없이 디코딩될 수도 있다. NAL 유닛이 멀티-뷰 코딩, 3DV, 또는 SVC에서의 기본 계층에 관련되지 않으면, nuh_reserved_zero_6bits 신택스 엘리먼트는 0이 아닌 값을 가질 수도 있다. 위에서 나타낸 바와 같이, 멀티-뷰 코딩 및 3DV 코딩에서, 비트스트림의 상이한 계층들은 상이한 뷰들에 대응할 수도 있다. SVC에서, 기본 계층 이외의 계층들은 "향상 계층 (enhancement layer) 들"이라고 지칭될 수도 있고 비트스트림으로부터 디코딩된 비디오 데이터의 시각적 품질을 향상시키는 정보를 제공할 수도 있다.

[0110] 더욱이, 계층 내의 일부 화상들이 동일한 계층 내의 다른 화상들에 대한 참조 없이 디코딩될 수도 있다. 따라서, 계층의 특정한 화상들의 데이터를 캡슐화하는 NAL 유닛들은 그 계층에서의 다른 화상들의 디코딩능력 (decodability) 에 영향을 미치는 일 없이 비트스트림으로부터 제거될 수도 있다. 이러한 화상들의 데이터를 캡슐화하는 NAL 유닛들을 제거하는 것은 비트스트림의 프레임 레이트를 감소시킬 수도 있다. 계층 내의 다른 화상들에 대한 참조 없이 디코딩될 수도 있는 그 계층 내의 화상들의 서브세트가 "서브-계층" 또는 "시간적 서브-계층"이라고 지칭될 수도 있다.

[0111] 비-기본 뷰들 중 하나의 비-기본 뷰에서의 화상을 코딩하는 경우, 그 화상이 비디오 코더가 현재 코딩하고 있는 화상과는 상이한 뷰에 있지만 동일한 시간 인스턴스 (즉, 액세스 유닛) 내에 있다면, 비디오 코더 (이를테면 비디오 인코더 (20) 또는 비디오 디코더 (30)) 는 그 화상을 참조 화상 리스트에 부가할 수도 있다. 다른 인

터 예측 참조 화상들처럼, 비디오 코더는 뷰 간 예측 참조 화상을 참조 화상 리스트의 임의의 포지션에 삽입할 수도 있다.

[0112] 도 3은 일 예의 멀티-뷰 디코딩 순서를 도시하는 개념도이다. 멀티-뷰 디코딩 순서는 비트스트림 순서일 수도 있다. 도 3의 예에서, 각각의 정사각형은 뷰 성분에 대응한다. 정사각형들의 열들은 액세스 유닛들에 대응한다. 각각의 액세스 유닛은 시간 인스턴스의 모든 뷰들의 코딩된 화상들을 포함하도록 정의될 수도 있다. 정사각형들의 행들은 뷰들에 대응한다. 도 3의 예에서, 액세스 유닛들은 T0...T8로 라벨 표시되고 뷰들은 S0...S7로 라벨 표시된다. 액세스 유닛의 각각의 뷰 성분이 다음 액세스 유닛의 임의의 뷰 성분 전에 디코딩되기 때문에, 도 3의 디코딩 순서는 시간 우선 코딩 (time-first coding) 이라고 지칭될 수도 있다. 액세스 유닛들의 디코딩 순서는 뷰들의 출력 또는 디스플레이 순서와 동일하지 않을 수도 있다.

[0113] 멀티-뷰 코딩은 뷰 간 예측을 지원할 수도 있다. 뷰 간 예측은 H.264/AVC, HEVC, 또는 다른 비디오 코딩 사양들에서 사용된 인터 예측과 유사하고, 동일한 신택스 엘리먼트들을 사용할 수도 있다. 그러나, 비디오 코더가 현재 블록 (이를테면 매크로블록 또는 PU) 에 대해 뷰 간 예측을 수행하는 경우, 비디오 코더는, 참조 화상으로서, 현재 블록과 동일한 액세스 유닛에 있지만 상이한 뷰에 있는 화상을 사용할 수도 있다. 본원에서 설명된 뷰 간 예측과는 대조적으로, 기존의 인터 예측은 상이한 액세스 유닛들에서의 화상들만을 참조 화상들로서 사용한다.

[0114] 도 4는 멀티-뷰 코딩을 위한 일 예의 예측 구조를 도시하는 개념도이다. 도 4의 멀티-뷰 예측 구조는 시간적 및 뷰 간 예측을 포함한다. 도 4의 예에서, 각각의 정사각형은 뷰 성분에 대응한다. 도 4의 예에서, 액세스 유닛들은 T0...T11로 라벨 표시되고 뷰들은 S0...S7로 라벨 표시된다. "I"로 라벨 표시된 정사각형들은 인트라 예측된 뷰 성분들이다. "P"로 라벨 표시된 정사각형들은 단방향으로 인터 예측된 뷰 성분들이다. "B" 및 "b"로 라벨 표시된 정사각형들은 양방향으로 인터 예측된 뷰 성분들이다. "b"로 라벨 표시된 정사각형들은 "B" 라벨 표시된 정사각형들을 참조 화상들로서 사용할 수도 있다. 첫 번째 정사각형에서부터 두 번째 정사각형을 가리키는 화살표가, 첫 번째 정사각형이 인터 예측에서 두 번째 정사각형에 대한 참조 화상으로서 이용가능하다는 것을 나타낸다. 도 4에서 수직 화살표들에 의해 나타낸 바와 같이, 동일한 액세스 유닛의 상이한 뷰들에서의 뷰 성분들은 참조 화상들로서 이용가능할 수도 있다. 하나의 액세스 유닛의 하나의 뷰 성분의 동일한 액세스 유닛의 다른 뷰 성분에 대한 참조 화상으로서의 사용은 뷰 간 예측이라고 지칭될 수도 있다.

[0115] H.264/AVC의 MVC 확장본에서, 뷰 간 예측은 디스패리티 모션 보상에 의해 지원되는데, 이 디스패리티 모션 보상은 H.264/AVC 모션 보상의 신택스를 사용하지만, 상이한 뷰에서의 화상이 참조 화상으로서 사용되는 것을 허용한다. 2 개의 뷰들의 코딩은 H.264/AVC의 MVC 확장본에 의해 또한 지원될 수도 있다. H.264/AVC의 MVC 확장본의 장점들 중 하나는, MVC 인코더가 2 개를 초과하는 뷰들을 3D 비디오 입력으로서 취할 수도 있고 MVC 디코더는 이러한 멀티뷰 표현을 디코딩할 수도 있다는 것이다. 결과적으로, MVC 디코더를 갖는 임의의 렌더러 (renderer) 는 2 개를 초과하는 뷰들을 갖는 3D 비디오 콘텐츠를 기대할 수도 있다.

[0116] 멀티-뷰 코딩에서, 뷰들 간의 상관을 제거하기 위해 동일한 액세스 유닛의 (즉, 동일한 타임 인스턴스를 갖는) 상이한 뷰들에서의 화상들 중에서 뷰 간 예측이 수행될 수도 있다. 비-기본 뷰들 중 하나에서의 화상을 코딩하는 경우, 화상이 상이한 뷰에 있지만 동일한 시간 인스턴스를 가지면, 그 화상은 참조 화상 리스트에 부가될 수도 있다. 뷰 간 예측 참조 화상이, 임의의 인터 예측 참조 화상처럼, 참조 화상 리스트의 임의의 포지션에 놓일 수 있다.

[0117] 멀티-뷰 비디오 코딩의 맥락에서, 2 종류의 모션 벡터들이 있다. 한 종류의 모션 벡터는 시간적 참조 화상을 가리키는 정상적인 모션 벡터이다. 정상적인 시간적 모션 벡터에 대응하는 인터 예측의 유형은 "모션 보상 예측 (motion-compensated prediction) "또는 "MCP"라고 지칭될 수도 있다. 뷰 간 예측 참조 화상이 모션 보상을 위해 사용되는 경우, 대응하는 모션 벡터는 "디스패리티 모션 벡터"라고 지칭된다. 다르게 말하면, 디스패리티 모션 벡터가 상이한 뷰에서의 화상 (즉, 뷰 간 참조 화상) 을 가리킨다. 디스패리티 모션 벡터에 대응하는 인터 예측의 유형은 "디스패리티 보상 예측 (disparity-compensated prediction)" 또는 "DCP"라고 지칭될 수도 있다.

[0118] 위에서 언급했듯이, HEVC의 3DV 확장본 (즉, 3D-HEVC) 이 개발중에 있다. 3D-HEVC는 뷰 간 모션 예측 및 뷰 간 잔차 예측을 사용하여 코딩 효율을 개선할 수도 있다. 뷰 간 모션 예측에서, 비디오 코더가 현재 PU와는 상이한 뷰에서의 PU의 모션 정보에 기초하여 현재 PU의 모션 정보를 결정 (즉, 예측) 할 수도 있다. 뷰 간 잔차 예측에서, 비디오 코더가 현재 CU와는 상이한 뷰에서의 잔차 데이터에 기초하여 현재 CU의 잔차 블

록들을 결정할 수도 있다.

[0119] 뷰 간 모션 예측과 뷰 간 잔차 예측을 가능하게 하기 위해, 비디오 코더가 블록들 (예컨대, PU들, CU들 등) 에 대한 디스패리티 벡터들을 결정할 수도 있다. 대체로, 디스패리티 벡터가 2 개의 뷰들 간의 변위의 추정자 (estimator) 로서 사용된다. 비디오 코더가 뷰 간 모션 또는 잔차 예측을 위해 다른 뷰에서 참조 블록을 위치결정하기 위해 블록에 대한 디스패리티 벡터를 사용할 수도 있거나, 또는 비디오 코더는 뷰 간 모션 예측을 위해 디스패리티 벡터를 디스패리티 모션 벡터로 변환할 수도 있다. 예를 들면, 블록이 뷰 간 모션 예측으로 코딩되는 경우, 디스패리티 벡터가 상이한 뷰에서 대응 블록을 선택하기 위해 도출될 필요가 있을 수도 있다.

[0120] 더욱이, 비디오 코더가 현재 블록에 대한 디스패리티 벡터를 도출할 수도 있다. 몇몇 예들에서, 비디오 코더는 이웃 블록 기반 디스패리티 벡터 (NBDV) 법을 사용하여 현재 블록에 대한 디스패리티 벡터를 도출할 수도 있다. 다시 말하면, 현재 블록에 대한 디스패리티 벡터를 도출하기 위해, NBDV라 지칭되는 프로세스가 3D-HEVC를 위한 테스트 모델 (즉, 3D-HTM) 에서 사용될 수도 있다. 3D-HEVC는 『Zhang et al., "3D-CE5.h: Disparity vector generation results", Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 1st Meeting: Stockholm, SE, 16-20 July 2012, document JCT3V-A0097』에서 제안된 NBDV 도출 프로세스를 먼저 채택했다.

[0121] NBDV 프로세스는 공간적 및 시간적 이웃 블록들로부터의 디스패리티 모션 벡터들을 사용하여 현재 블록에 대한 디스패리티 벡터를 도출한다. 이웃 블록들 (예컨대, 현재 블록에 공간적으로 또는 시간적으로 이웃하는 블록들) 이 비디오 코딩에서 거의 동일한 모션 및 디스패리티 정보를 공유할 가능성이 있기 때문에, 현재 블록은 이웃 블록들에서의 모션 벡터 정보를 현재 블록의 디스패리티 벡터의 예측자로서 사용할 수 있다. 따라서, NBDV 프로세스는 상이한 뷰들에서의 디스패리티 벡터를 추정하기 위해 이웃 디스패리티 정보를 사용한다.

[0122] 비디오 코더가 NBDV 프로세스를 수행하는 경우, 비디오 코더는, 고정된 체킹 순서로, 공간적으로 이웃하는 및 시간적으로 이웃하는 PU들의 모션 벡터들을 체킹할 수도 있다. 다시 말하면, 여러 공간적 및 시간적 이웃 블록들이 먼저 정의되며, 그것들의 각각은 그 다음에 현재 블록과 후보 블록 (즉, 공간적 또는 시간적 이웃 블록) 간의 상관의 우선순위에 의해 결정될 수도 있는 미리 정의된 순서로 체킹된다. 따라서, 이웃 블록들의 2 개의 세트들이 이용된다. 하나의 세트는 공간적 이웃 블록들로부터이고 다른 세트는 시간적 이웃 블록들로부터이다.

[0123] 비디오 코더가 공간적으로 이웃하는 또는 시간적으로 이웃하는 PU의 모션 벡터(들)를 체킹하는 경우, 비디오 코더는 모션 벡터(들)가 디스패리티 모션 벡터들인지의 여부를 결정할 수도 있다. 화상의 PU의 디스패리티 모션 벡터가 화상의 뷰 간 참조 화상 내의 로케이션을 가리키는 모션 벡터이다. 화상의 뷰 간 참조 화상이 그 화상과 동일한 액세스 유닛에 있지만 상이한 뷰에 있는 화상일 수도 있다. 일단 디스패리티 모션 벡터 (즉, 그 모션 벡터는 뷰 간 참조 화상을 가리킨다) 가 후보들에서 찾아지면, 비디오 코더는 디스패리티 모션 벡터를 디스패리티 벡터로 변환할 수도 있다. 예를 들어, 비디오 코더는 현재 블록에 대한 디스패리티 벡터의 수평 성분을 디스패리티 모션 벡터의 수평 성분과 동일하게 설정할 수도 있고 디스패리티 벡터의 수직 성분을 0으로 설정할 수도 있다.

[0124] 3D-HEVC의 몇몇 설계들 (예컨대, 3D-HTM 6.0) 에서, 비디오 코더가 NBDV 프로세스를 수행하는 경우, 비디오 코더는 시간적 이웃 블록들에서의 디스패리티 모션 벡터들, 공간적 이웃 블록들에서의 디스패리티 모션 벡터들, 그리고 그 다음에 암시적 디스패리티 벡터 (implicit disparity vector; IDV) 들을 그 순서대로 체킹한다. IDV가 뷰간 예측을 사용하여 코딩되는 공간적으로 또는 시간적으로 이웃하는 PU의 디스패리티 벡터일 수도 있다. IDV들은 도출된 디스패리티 벡터들이라고 또한 지칭될 수도 있다. PU가 뷰 간 모션 벡터 예측을 채용하는 경우, 즉, AMVP 또는 병합 모드들에 대한 후보가 디스패리티 벡터의 도움으로 다른 뷰에서의 참조 블록으로부터 도출되는 경우 IDV가 생성될 수도 있다. 이러한 디스패리티 벡터는 IDV라고 지칭된다. IDV가 디스패리티 벡터 도출의 목적으로 PU에 저장될 수도 있다. 예를 들면, 블록이 모션 예측으로 코딩되더라도, 그 블록에 대한 도출된 디스패리티 벡터는 다음 블록을 코딩할 목적으로 버려지지 않는다. 따라서, 비디오 코더가 디스패리티 모션 벡터 또는 IDV를 식별하는 경우, 비디오 코더는 식별된 디스패리티 모션 벡터 또는 IDV를 반환할 수도 있다. IDV들은 『Sung et al., "3D-CE5.h: Simplification of disparity vector derivation for HEVC-based 3D video coding, document JCTV3-A0126"』에서의 NBDV의 단순화된 버전에 포함되었다. NBDV 프로세스에서의 IDV들의 사용은 디코딩된 화상 버퍼에 저장된 IDV들을 제거함으로써 『Kang et al., "3D-CE5.h related: improvements for disparity vector derivation", document JCT3V-B0047』에서 더욱

단순화되었고, 랜덤 액세스 포인트 (random access point; RAP) 화상 선택으로 코딩 이득을 또한 개선하였다.

비디오 코더는 반환된 디스패리티 모션 벡터 또는 IDV를 디스패리티 벡터로 변환할 수도 있고 그 디스패리티 벡터를 뷰 간 모션 예측 및 뷰 간 잔차 예측을 위해 사용할 수도 있다. 랜덤 액세스는 비트스트림에서 처음 코딩된 화상이 아닌 코딩된 화상으로부터 시작하는 비트스트림의 디코딩을 지칭한다. 랜덤 액세스 화상들 또는 랜덤 액세스 포인트들의 비트스트림 속으로의 규칙적인 간격들로의 삽입은 랜덤 액세스를 가능하게 할 수도 있다. 랜덤 액세스 화상들의 예의 유형들은 순간적 디코더 리프레시 (IDR) 화상들, 클린 랜덤 액세스 (clean random access; CRA) 화상들, 및 깨진 링크 액세스 (broken link access; BLA) 화상들을 포함한다. 이런 이유로, IDR 화상들, CRA 화상들 및 BLA 화상들은 RAP 화상들이라고 총칭된다. 몇몇 예들에서, RAP 화상들은 BLA_W_LP, BLA_W_RADL, BLA_N_LP, IDR_W_RADL, IDR_N_LP, RSV_IRAP_VCL22, RSV_IRAP_VCL23, 또는 CRA_NUT와 동일한 NAL 유닛 유형들을 가질 수도 있다.

[0125] 비디오 코더가 디스패리티 모션 벡터 또는 IDV를 식별하는 경우, 비디오 코더는 체킹 프로세스를 종료할 수도 있다. 따라서, 일단 비디오 코더가 현재 블록에 대한 디스패리티 벡터를 찾으면, 비디오 코더는 NBDV 프로세스를 종료할 수도 있다. 비디오 코더가 NBDV 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 결정할 수 없는 경우 (즉, NBDV 프로세스 동안 찾아진 디스패리티 모션 벡터 또는 IDV가 없는 경우), NBDV는 이용불가능으로 마킹된다. 다르게 말하면, NBDV 프로세스는 이용불가능한 디스패리티 벡터를 반환한다고 간주될 수 있다.

[0126] 비디오 코더가 NBDV 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 없다면 (즉, 디스패리티 벡터가 찾아지지 않는다면), 비디오 코더는 영 디스패리티 벡터를 현재 PU에 대한 디스패리티 벡터로서 사용할 수도 있다. 영 디스패리티 벡터는 0과 동일한 수평 및 수직 성분들 양자를 갖는 디스패리티 벡터이다. 따라서, NBDV 프로세스가 이용불가능한 결과를 반환하는 경우라도, 디스패리티 벡터를 필요로 하는 비디오 코더의 다른 코딩 프로세스들은 현재 블록에 대해 영 디스패리티 벡터를 사용할 수도 있다. 몇몇 예들에서, 비디오 코더가 NBDV 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 없다면, 비디오 코더는 현재 블록에 대한 뷰 간 잔차 예측을 불가능하게 할 수도 있다. 그러나, 비디오 코더가 NBDV 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 있는지의 여부에 상관 없이, 비디오 코더는 현재 블록에 대해 뷰 간 모션 예측을 사용할 수도 있다. 다시 말하면, 모든 미리 정의된 이웃 블록들을 체킹한 후 디스패리티 벡터가 찾아지지 않으면, 영 디스패리티 벡터가 뷰 간 모션 예측을 위해 사용될 수도 있는 한편 뷰 간 잔차 예측은 대응하는 CU에 대해 불가능하게 될 수도 있다.

[0127] 위에서 언급했듯이, 비디오 코더는 현재 PU에 대한 디스패리티 벡터를 결정하는 프로세스의 부분으로서 공간적으로 이웃하는 PU들을 체킹할 수도 있다. 일부 예들에서, 비디오 코더는 다음의 공간적으로 이웃하는 블록들, 즉, 좌하측의 공간적으로 이웃하는 블록, 좌측의 공간적으로 이웃하는 블록, 우상측의 공간적으로 이웃하는 블록, 상측의 공간적으로 이웃하는 블록, 및 좌상측의 공간적으로 이웃하는 블록을 체킹할 수도 있다. 예를 들면, NBDV 프로세스의 몇몇 버전들에서, 5 개의 공간적 이웃 블록들이 디스패리티 벡터 도출을 위해 사용된다.

5 개의 공간적으로 이웃하는 블록들은 도 2에 나타낸 바와 같이 로케이션들 (A_0, A_1, B_0, B_1, 및 B_2) 을 각각 커버할 수도 있다. 비디오 코더는 5 개의 공간적으로 이웃하는 블록들을 A_1, B_1, B_0, A_0, 및 B_2의 순서로 체킹할 수도 있다. 동일한 5 개의 공간적으로 이웃하는 블록이 HEVC에 대한 병합 모드들에서 사용될 수도 있다. 그러므로, 몇몇 예들에서, 부가적인 메모리 액세스가 요구되지 않는다. 공간적으로 이웃하는 블록들 중 하나가 디스패리티 모션 벡터를 갖는다면, 비디오 코더는 체킹 프로세스를 종료할 수도 있고 비디오 코더는 디스패리티 모션 벡터를 현재 블록에 대한 최종 디스패리티 벡터로서 사용할 수도 있다.

[0128] 더욱이, 위에서 언급했듯이, 비디오 코더는 현재 블록에 대한 디스패리티 벡터를 결정하는 프로세스의 부분으로서 시간적으로 이웃하는 PU들을 체킹할 수도 있다. 시간적 이웃 블록들 (예컨대, PU들) 을 체킹하기 위해, 후보 화상 리스트의 구축 프로세스가 먼저 수행될 수도 있다. 몇몇 예들에서, 비디오 코더는 디스패리티 모션 벡터들에 대해 현재 뷰로부터 2 개까지의 참조 화상들을 체킹할 수도 있다. 첫 번째 참조 화상은 병치된 화상일 수도 있다. 따라서, 병치된 화상 (즉, 병치된 참조 화상) 은 후보 화상 리스트에 먼저 삽입될 수도 있으며, 그 뒤로 다른 후보 화상들이 참조 인덱스의 오름 차순으로 삽입될 수도 있다. 양자의 참조 화상 리스트들에서 동일한 참조 인덱스를 갖는 참조 화상들이 이용가능한 경우, 병치된 화상과는 동일한 참조 화상 리스트에서의 참조 화상은, 후보 리스트에서, 병치된 화상과는 동일한 참조 화상 리스트에 있지 않은 참조 화상에 선행한다.

[0129] 후보 화상 리스트에서의 각각의 후보 화상 (즉, 랜덤-액세스 화상과 병치된 화상) 에 대해, 비디오 코더는 3 개

의 후보 지역들을 체킹할 수도 있다. 3 개의 후보 지역들은 다음과 같이 정의될 수도 있다:

[0130] • CPU: 현재 PU 또는 현재 CU의 병치된 지역.

[0131] • CLCU: 현재 블록의 병치된 지역을 커버하는 최대 코딩 유닛 (LCU).

[0132] • BR: CPU의 우측하단 4x4 블록.

[0133] 후보 지역을 커버하는 PU가 디스패리티 모션 벡터를 특정하면, 비디오 코더는 PU의 디스패리티 모션 벡터에 기초하여 현재 비디오 유닛의 디스패리티 벡터를 결정할 수도 있다.

[0134] 비디오 코더가 이웃 PU (예컨대, 공간적으로 또는 시간적으로 이웃하는 PU) 를 체킹하는 경우, 비디오 코더는 그 이웃 PU가 디스패리티 모션 벡터를 갖는지의 여부를 먼저 체킹할 수도 있다. 이웃 PU들 중 어느 것도 디스패리티 모션 벡터를 갖지 않는다면, 비디오 코더는 공간적으로 이웃하는 PU들 중 어떤 것이 IDV를 갖는지의 여부를 결정할 수도 있다. 비디오 코더는 공간적으로 이웃하는 PU들을 A_0, A_1, B_0, B_1, 및 B_2의 순서로 체킹할 수도 있다. 공간적으로 이웃하는 PU들 중 하나가 IDV를 갖고 IDV가 병합/스킵 모드로서 코딩되면, 비디오 코더는 체킹 프로세스를 종료할 수도 있고 IDV를 현재 블록에 대한 최종 디스패리티 벡터로서 사용할 수도 있다.

[0135] 위에서 나타낸 바와 같이, 비디오 코더가 NBDV 프로세스를 적용하여 현재 블록 (예컨대, CU, PU 등) 에 대한 디스패리티 벡터를 도출할 수도 있다. 현재 블록에 대한 디스패리티 벡터는 참조 뷰에서 참조 화상에서의 로케이션 (즉, 참조 성분) 을 나타낼 수도 있다. 몇몇 3D-HEVC 설계들에서, 비디오 코더는 참조 뷰에 대한 깊이 정보에 액세스하는 것이 허용된다. 이러한 몇몇 3D-HEVC 설계들에서, 비디오 코더가 현재 블록에 대한 디스패리티 벡터를 도출하기 위해 NBDV 프로세스를 사용하는 경우, 비디오 코더는 리파인먼트 프로세스를 적용하여 현재 블록에 대한 디스패리티 벡터를 더욱 리파인할 수도 있다. 비디오 코더는 참조 화상의 깊이 맵에 기초하여 현재 블록에 대한 디스패리티 벡터를 리파인할 수도 있다. 비디오 코더는 백워드 뷰 합성 예측을 위한 디스패리티 모션 벡터를 리파인하기 위해 유사한 리파인먼트 프로세스를 사용할 수도 있다. 이런 식으로, 깊이는 백워드 뷰 합성 예측을 위해 사용될 디스패리티 벡터 또는 디스패리티 모션 벡터를 리파인하는데 사용될 수 있다. 이 리파인먼트 프로세스는 본원에서 NBDV 리파인먼트 ("NBDV-R"), NBDV 리파인먼트 프로세스, 또는 깊이 지향 NBDV (Do-NBDV) 라고 지칭될 수도 있다.

[0136] NBDV 프로세스가 이용가능한 디스패리티 벡터를 반환하는 경우 (예컨대, NBDV 프로세스가 현재 블록에 대한 디스패리티 벡터를 디스패리티 모션 벡터 또는 이웃 블록의 IDV에 기초하여 도출할 수 있었음을 나타내는 변수를 NBDV 프로세스가 반환하는 경우), 비디오 코더는 참조 뷰의 깊이 맵으로부터 깊이 데이터를 취출함으로써 디스패리티 벡터를 추가로 리파인할 수도 있다. 몇몇 예들에서, 리파인먼트 프로세스는 다음 2 개의 단계들을 포함한다:

[0137] 1. 현재 블록의 디스패리티 벡터를 사용하여 참조 뷰의 깊이 맵에서 블록을 위치결정함. 다르게 말하면, 이전에 코딩된 참조 깊이 뷰, 이를테면 기본 뷰에서의 도출된 디스패리티 벡터에 의해 대응하는 깊이 블록을 위치결정한다. 이 예에서, 깊이에서의 대응 블록의 사이즈는 현재 블록의 사이즈와 동일할 수도 있다.

[0138] 2. 하나의 깊이 값을 대응하는 깊이 블록의 4 개의 코너 화소들로부터 선택하고 깊이 값을 리파인된 디스패리티 벡터의 수평 성분으로 변환. 이 예에서, 비디오 코더는 디스패리티 벡터의 수직 성분을 변경하지 않는다.

[0139] NBDV 프로세스가 이용가능한 디스패리티 벡터를 반환하지 않는 경우 (예컨대, NBDV 프로세스가 디스패리티 모션 벡터 또는 이웃 블록의 IDV에 기초하여 현재 블록에 대한 디스패리티 벡터를 도출할 수 없었음을 나타내는 변수를 NBDV 프로세스가 반환하는 경우), 비디오 코더는 NBDV 리파인먼트 프로세스를 수행하지 않고 비디오 코더는, 현재 블록에 대한 디스패리티 벡터로서, 영 디스패리티 벡터를 사용한다. 다르게 말하면, NBDV가 이용가능한 디스패리티 벡터를 제공하지 않고 따라서 NBDV의 결과가 이용가능하지 않은 경우, 위의 NBDV-R 프로세스는 스킵되고 영 디스패리티 벡터가 직접적으로 반환된다.

[0140] 몇몇 3D-HEVC 설계들에서, 비디오 코더는 뷰 간 모션 예측을 위해 현재 블록에 대한 리파인된 디스패리티 벡터를 사용하는 반면 비디오 코더는 뷰 간 잔차 예측을 위해 현재 블록에 대한 미리파인된 디스패리티 벡터를 사용한다. 예를 들어, 비디오 코더는 NBDV 프로세스를 사용하여 현재 블록에 대한 미리파인된 디스패리티 벡터를 도출할 수도 있다. 비디오 코더는 그 다음에 현재 블록에 대한 리파인된 디스패리티 벡터를 도출하기 위

해 NBDV 리파인먼트 프로세스를 적용할 수도 있다. 비디오 코더는 현재 블록의 모션 정보를 결정할 목적으로 현재 블록에 대해 리파인된 디스패리티 벡터를 사용할 수도 있다. 더구나, 비디오 코더는 현재 블록의 잔차 블록을 결정할 목적으로 현재 블록에 대한 미리파인된 디스패리티 벡터를 사용할 수도 있다. 몇몇 예들에서, 비디오 코더는 PU가 BVSP 모드로 코딩되면 리파인된 디스패리티 벡터를 하나의 PU의 모션 벡터로서 저장할 수도 있다.

[0141] 백워드 뷰 합성 예측 (BVSP) 접근법이『Tian et al., "CE1.h: Backward View Synthesis Prediction Using Neighboring Blocks", document JCT3V-C0152』 (이후로는, "JCT3V-C0152") 에서 제안되었고 3차 JCT-3V 미팅에서 채택되었다. 비디오 코더가 BVSP를 수행하여 뷰 성분을 합성할 수도 있다. 뷰 성분이 합성될 수 있기 때문에, 비트스트림이 뷰 성분의 코딩된 표현을 포함하는 것이 불필요할 수도 있다. 적어도 이런 이유로, BVSP의 사용은 비트스트림의 사이즈를 감소시킬 수도 있다.

[0142] BVSP는 3D-AVC에서의 블록 기반 VSP와 개념적으로 유사하다. 다르게 말하면, 백워드-워핑 VSP의 기본 아이디어는 3D-AVC에서의 블록 기반 VSP와 동일하다. BVSP 및 3D-AVC에서의 블록 기반 VSP 양쪽 모두는 모션 벡터 차이들의 송신을 피하기 위해 그리고 더욱 정확한 모션 벡터들을 사용하기 위해 백워드 워핑과 블록 기반 VSP를 사용한다. 그러나, 구현예의 세부사항들은 상이한 플랫폼들로 인해 상이하다.

[0143] 대체로, 비디오 코더가 참조 텍스처 화상을 합성하기 위해 BVSP를 수행하는 경우, 비디오 코더는 의존성 텍스처 화상에서의 블록들 (예컨대, 비디오 유닛들) 을 프로세싱한다. 의존성 텍스처 화상과 합성된 텍스처 화상은 동일한 액세스 유닛에 있지만, 상이한 뷰들에 있다. 비디오 코더가 의존성 텍스처 화상의 블록 (즉, 현재 블록) 을 프로세싱하는 경우, 비디오 코더는 현재 블록의 디스패리티 벡터를 식별하기 위해 NBDV 프로세스를 수행할 수도 있다. 다시 말하면, 블록에 대한 깊이 정보를 추정하기 위하여, 비디오 코더가 이웃 블록들로부터 디스패리티 벡터를 먼저 도출할 수도 있다.

[0144] 더욱이, 비디오 코더가 참조 텍스처 화상을 합성하기 위해 BVSP를 수행하는 경우, 비디오 코더는 참조 깊이 화상에서의 참조 블록을 식별하기 위해 현재 블록의 디스패리티 벡터를 사용할 수도 있다. 다르게 말하면, 비디오 코더는 그 다음에 참조 뷰로부터 깊이 블록을 획득하기 위해 도출된 디스패리티 벡터를 사용할 수도 있다. 예를 들면, NBDV 프로세스에 의해 식별된 디스패리티 벡터는 (dv_x, dv_y) 로서 표시될 수도 있고 현재 블록 포지션은 ($block_x$, $block_y$) 로서 표시될 수도 있다. 더욱이, 이 예에서, 비디오 코더는 참조 뷰의 깊이 이미지에서의 ($block_x+dv_x$, $block_y+dv_y$) 에서 깊이 블록을 페치할 수도 있다. 이 예에서, 페치된 깊이 블록은 현재 PU의 동일한 사이즈를 갖는다. 의존성 텍스처 화상과 참조 깊이 화상은 동일한 액세스 유닛에 있지만, 상이한 뷰들에 있다. 비디오 코더는 그 다음에, 현재 블록의 샘플 값들 및 참조 화상의 식별된 참조 블록의 샘플 값들에 기초하여, 합성된 화상의 샘플 값들을 결정하기 위해 백워드 워핑 프로세스를 수행할 수도 있다. 다르게 말하면, 비디오 코더는, 이 예에서, 페치된 깊이 블록을 사용하여 현재 PU에 대한 백워드 워핑을 수행할 수도 있다.

[0145] 위에서 나타낸 바와 같이, 비디오 코더가 BVSP를 수행하는 경우, 비디오 코더는 현재 블록에 대한 디스패리티 벡터를 식별하기 위해 NBDV 프로세스를 수행할 수도 있다. 더욱이, 비디오 코더가 BVSP를 수행하는 경우, 비디오 코더는 NBDV 프로세스를 사용하여 도출된 디스패리티 모션 벡터를 리파인하기 위해 본 개시물의 다른 곳에서 설명된 것과 유사한 리파인먼트 프로세스를 사용할 수도 있다. 비디오 코더가 디스패리티 벡터 리파인먼트 프로세스를 수행하는 경우, 비디오 코더는 참조 뷰에서의 깊이 맵의 깊이 값들에 기초하여 디스패리티 벡터를 리파인할 수도 있다. 다르게 말하면, 깊이는 BVSP를 위해 사용될 디스패리티 벡터 또는 디스패리티 모션 벡터를 리파인하는데 사용될 수 있다. 리파인된 디스패리티 벡터가 BVSP 모드로 코딩되면 리파인된 디스패리티 벡터는 하나의 PU의 모션 벡터로서 저장될 수도 있다.

[0146] 3D-HEVC의 몇몇 버전들에서, 텍스처 우선 코딩이 적용된다. 텍스처 우선 코딩에서, 비디오 코더는 텍스처 뷰 성분을 대응하는 깊이 뷰 성분 (즉, 텍스처 뷰 성분과 동일한 POC 값 및 뷰 식별자를 갖는 깊이 뷰 성분) 을 코딩하는 것보다 먼저 코딩 (예컨대, 인코딩 또는 디코딩) 한다. 그러므로, 비-기본 뷰 깊이 뷰 성분이 대응하는 비-기본 뷰 텍스처 뷰 성분을 코딩함에 있어서의 사용에 이용불가능하다. 다르게 말하면, 비디오 코더가 비-기본 텍스처 뷰 성분을 코딩하는 경우, 대응하는 비-기본 깊이 뷰 성분은 이용불가능하다. 그러므로, 깊이 정보는 BVSP를 수행하기 위해 추정되고 사용될 수도 있다.

[0147] 도 5는 BVSP 예측을 수행하기 위한 참조 뷰로부터의 깊이 블록 도출을 예시하는 개념도이다. 도 5의 예에서, 비디오 코더가 현재 텍스처 화상 (60) 을 코딩하고 있다. 현재 텍스처 화상 (60) 은 "의존성 텍스

처 화상"으로 라벨 표시되는데 현재 텍스처 화상 (60) 이 합성된 참조 텍스처 화상 (62) 에 의존하기 때문이다. 다르게 말하면, 비디오 코더는 현재 텍스처 화상 (60) 을 디코딩하기 위하여 참조 텍스처 화상 (62) 을 합성할 필요가 있을 수도 있다. 참조 텍스처 화상 (62) 과 현재 텍스처 화상 (60) 은 동일한 액세스 유닛에 있지만 상이한 뷰들에 있다.

[0148] 참조 텍스처 화상 (62) 을 합성하기 위하여, 비디오 코더는 현재 텍스처 화상 (60) 의 블록들 (즉, 비디오 유닛들) 을 프로세싱할 수도 있다. 도 5의 예에서, 비디오 코더는 현재 블록 (64) 을 프로세싱하고 있다. 비디오 코더가 현재 블록 (64) 을 프로세싱하는 경우, 비디오 코더는 현재 블록 (64) 에 대한 디스패리티 벡터를 도출하기 위해 NBDV 프로세스를 수행할 수도 있다. 예를 들면, 도 5의 예에서, 비디오 코더는 현재 블록 (64) 에 이웃하는 블록 (68) 의 디스패리티 벡터 (66) 를 식별한다. 디스패리티 벡터 (66) 의 식별은 도 5의 스텝 1로서 예시된다. 더욱이, 도 5의 예에서, 비디오 코더는, 디스패리티 벡터 (66) 에 기초하여, 현재 블록 (64) 의 디스패리티 벡터 (68) 를 결정한다. 예를 들면, 디스패리티 벡터 (68) 는 디스패리티 벡터 (66) 의 카피일 수도 있다. 디스패리티 벡터 (66) 를 카피하는 것이 도 5의 스텝 2로서 도시되어 있다.

[0149] 비디오 코더는, 현재 블록 (64) 의 디스패리티 벡터 (68) 에 기초하여, 참조 깊이 화상 (72) 에서의 참조 블록 (70) 을 식별할 수도 있다. 참조 깊이 화상 (72), 현재 텍스처 화상 (60), 및 참조 텍스처 화상 (62) 은 각각이 동일한 액세스 유닛에 있을 수도 있다. 참조 깊이 화상 (72) 과 참조 텍스처 화상 (62) 은 동일한 뷰에 있을 수도 있다. 비디오 코더는, 현재 블록 (64) 의 텍스처 샘플 값들 및 참조 블록 (70) 의 깊이 샘플 값들에 기초하여, 참조 텍스처 화상 (62) 의 텍스처 샘플 값들을 결정할 수도 있다. 텍스처 샘플 값들을 결정하는 프로세스는 백워드 워핑이라고 지칭될 수도 있다. 3D-HEVC 테스트 모델 3의 섹션 H.8.5.2.2.7은 백워드 워핑의 프로세스를 설명한다. 백워드 워핑은 도 5의 스텝 3으로서 도시되어 있다. 이런 식으로, 도 5는 참조 뷰로부터의 깊이 블록이 위치결정된 후 BVSP 예측을 위해 사용되는 방법의 3 개의 스텝들을 예시한다.

[0150] 몇몇 예들에서, BVSP는 몇몇 CVS들에 대해 가능하지만, 다른 것들에 대해서는 아니다. BVSP이 가능하게 된 CVS들에서, 비디오 코더가 뷰 간 모션 예측에 대해 NBDV 프로세스와는 상이한 NBDV 프로세스를 수행할 수도 있다. 다시 말하면, BVSP가 CVS에서 가능하게 되면, 뷰 간 모션 예측을 위한 NBDV 프로세스는 변경될 수도 있다. 비디오 코더가 BVSP가 가능하게 된 CVS에서의 현재 블록에 대하여 NBDV 프로세스를 수행하는 경우, 비디오 코더는 시간적 이웃 블록들이 디스패리티 모션 벡터들을 갖는지의 여부를 결정할 수도 있다. 시간적 이웃 블록이 디스패리티 모션 벡터를 갖는다면, 비디오 코더는 시간적 이웃 블록의 디스패리티 모션 벡터에 기초하여 현재 블록의 디스패리티 벡터를 결정할 수도 있다. 비디오 코더는 그 다음에 디스패리티 벡터를 리파인할 수도 있다. 다르게 말하면, 시간적 이웃 블록들의 각각에 대해, 그 블록이 디스패리티 모션 벡터를 사용한다면, 디스패리티 모션 벡터는 디스패리티 벡터로서 반환되고 본 개시물의 다른 곳에서 설명된 방법으로 추가로 리파인된다.

[0151] 더욱이, 비디오 코더가 현재 블록에 대하여 NBDV 프로세스를 수행하는 경우, 비디오 코더는 공간적 이웃 블록들을 평가할 수도 있다. 공간적 이웃 블록들의 각각에 대해, 다음이 적용된다. RefPicList0과 RefPicList1의 순서의 경우, 공간적 이웃 블록이 디스패리티 모션 벡터를 사용하면, 비디오 코더는 디스패리티 모션 벡터를 현재 블록의 디스패리티 벡터로서 반환하고 비디오 코더는 본 개시물의 다른 곳에서 설명된 바와 같이 디스패리티 벡터를 추가로 리파인할 수도 있다. 그렇지 않고, 공간적 이웃 블록이 BVSP 모드를 사용하면, 비디오 코더는 연관된 모션 벡터를 현재 블록의 디스패리티 벡터로서 반환하고 본 개시물의 다른 곳에서 설명된 것과 유사한 방식으로 디스패리티 벡터를 리파인할 수도 있다. 그러나, 공간적 이웃 블록이 BVSP 모드를 사용하면, 비디오 코더는 4 개의 코너 화소들이 아니라 대응하는 깊이 블록의 모든 화소들로부터 최대 깊이 값을 선택할 수도 있고 비디오 코더는 리파인된 디스패리티 벡터의 수직 성분을 0으로 설정할 수도 있다. 공간적 이웃 블록이 IDV를 사용하면, 비디오 코더는 IDV를 디스패리티 벡터로서 반환할 수도 있고 비디오 코더는 본 개시물의 다른 곳에서 설명된 방법으로 디스패리티 벡터를 추가로 리파인할 수도 있다. 이용가능한 디스패리티 모션 벡터가 없다면, 비디오 디코더는 리파이닝 프로세스를 적용하지 않고, 비디오 코더는 영 벡터인 디스패리티 벡터를 도출할 수도 있다.

[0152] 비디오 코더가 현재 블록에 대한 디스패리티 벡터를 결정하고 현재 블록에 대한 디스패리티 벡터를 리파인한 후, 비디오 코더는 현재 블록에 대한 디스패리티 모션 벡터를 도출할 수도 있다. 현재 블록 (즉, BVSP 모드로 코딩된 하나의 PU) 내의 각각의 서브-지역 (4x4 블록) 에 대해, 비디오 코더는, 리파인된 디스패리티 벡터에 기초하여, 참조 깊이 뷰에서의 대응하는 4x4 깊이 블록을 위치결정할 수도 있다. 둘째로, 비디오 코더는 대응하는 깊이 블록에서의 16 개 깊이 화소들의 최대 값을 선택할 수도 있다. 셋째로, 비디오 코더는 최대 값

을 디스패리티 모션 벡터의 수평 성분으로 변환할 수도 있다. 비디오 코더는 디스패리티 모션 벡터의 수직 성분을 0으로서 설정할 수도 있다.

[0153] 비디오 코더가 특정한 유형들의 블록들 (예컨대, PU들, CU들 등) 에 대해 NBDV를 사용하여 디스패리티 벡터들을 도출할 수 있을 가능성은 없다. 예를 들어, 이용불가능한 디스패리티 벡터들이 블록들에 대한 NBDV로부터 슬라이스, 타일, 또는 화상 경계들을 따라 발생할 가능성이 있다. 다른 예에서, 이용불가능한 디스패리티 벡터들은, 이웃 블록들이 모두가 인트라 예측으로 코딩된다면, NBDV로부터 발생할 가능성이 있다.

[0154] 대체로, 타일이, 타일의 코딩 트리 블록 래스터 스캔으로 연속하여 순서화되는, 하나의 열 및 하나의 행에서 함께 발생하는 정수 수의 코딩 트리 블록들이다. 행이 정수 수의 코딩 트리 블록들일 수도 있다. 열들은 화상의 상단 경계로부터 하단 경계까지 연장하는 수직 경계들에 의해 서로로부터 윤곽이 그려지고, 그 화상에서 좌측부터 우측으로 연속하여 순서화된다. 행들은 화상의 좌측 경계로부터 우측 경계까지 연장하는 수평 경계들에 의해 서로로부터 윤곽이 그려지고, 그 화상에서 상단부터 하단까지 연속하여 순서화된다. 열이 정수 수의 코딩 트리 블록들일 수도 있다.

[0155] 3D-HEVC에서의 현재 디스패리티 벡터 도출 방법은 여러 문제들을 갖는다. 예를 들어, NBDV 프로세스가 이용불가능한 디스패리티 벡터를 제공하는 경우, 비디오 코더는 그 디스패리티 벡터를 리파인하는 NBDV-R 프로세스를 스킵할 수도 있다. 이는 코딩 성능 저하로 이어질 수도 있다.

[0156] 본 개시물의 기법들은 이전의 비디오 코딩 기법들을 넘어서는 장점들을 가질 수도 있다. 예를 들면, 본 개시물의 기법들은 디스패리티 벡터 도출 방법 (예컨대, 3D-HEVC에서의 NBDV) 이 이용가능한 디스패리티 벡터를 생성할 수 없는 경우 참조 뷰의 깊이 뷰 성분에 액세스함으로써 양호한 리파인된 디스패리티 벡터를 제공할 수도 있다.

[0157] 몇몇 예들에서, 비디오 코더는 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다. 비디오 코더가 디스패리티 벡터 도출 프로세스를 수행하는 경우, 비디오 코더는 현재 블록에 대한 디스패리티 벡터를 성공적으로 도출하는 것이 가능할 수도 있거나 또는 가능하지 않을 수도 있다. 예를 들면, 비디오 코더는 이웃 블록들 중 어느 것도 디스패리티 모션 벡터 또는 IDV를 갖지 않는 경우 디스패리티 벡터 도출 프로세스를 수행하는 것에 의해 현재 블록에 대한 디스패리티 벡터를 성공적으로 도출하는 것이 가능하지 않을 수도 있다. 이런 이유로, 비디오 코더가 디스패리티 벡터 도출 프로세스를 수행하는 경우, 비디오 코더는 현재 블록에 대한 디스패리티 벡터와 가용성 값 (예컨대, availableDV) 을 생성할 수도 있다. 가용성 값은 비디오 코더가 디스패리티 벡터 도출 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 있었는지의 여부를 나타낸다.

[0158] 본 개시물에서, 비디오 코더가 디스패리티 벡터 도출 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 있었음을 가용성 값이 나타내면, 현재 블록에 대한 디스패리티 벡터는 "이용가능"하다. 비슷하게, 비디오 코더가 디스패리티 벡터 도출 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 없었음을 가용성 값이 나타내면, 현재 블록에 대한 디스패리티 벡터는 "이용불가능"하다. 이런 이유로, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 가용성 값은 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 나타낸다. 더욱이, 본 개시물에서, 디스패리티 벡터 도출 프로세스는 비디오 코더가 디스패리티 벡터 도출 프로세스를 수행함으로써 현재 블록에 대한 디스패리티 벡터를 도출할 수 있었는지의 여부에 의존하여 이용가능한 또는 이용불가능한 디스패리티 벡터를 반환한다고 말해진다.

[0159] 본 개시물의 하나 이상의 기법들에 따라, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 비디오 코더는 그럼에도 불구하고 여전히, 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하기 위해 디스패리티 벡터 리파인먼트 프로세스를 수행할 수도 있다. 따라서, 디스패리티 벡터 리파인먼트 프로세스는 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하는 경우들로 제한되지 않는다. 이런 이유로, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않더라도, 본 개시물의 기법들은 디폴트 디스패리티 벡터 (이를테면 영과 동일한 수평 및 수직 성분들을 갖는 디폴트 디스패리티 벡터) 에 대해 디스패리티 벡터 리파인먼트를 허용할 수도 있다.

[0160] 예를 들어, 비디오 디코더 (30) 는 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스 (예컨대, NBDV 프로세스) 를 수행할 수도 있다. 현재 블록은 현재 뷰에 있을 수도 있다. 더욱이, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스

패리티 벡터가 이용불가능함을 가용성 값이 나타낼 수도 있다. 이 예에서, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 비디오 디코더 (30) 는 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스 (예컨대, NBDV-R 프로세스) 를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다. 비디오 디코더 (30) 는 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 디코딩할 수도 있다. 예를 들면, 비디오 디코더 (30) 는 리파인된 디스패리티 벡터를 사용하여 현재 블록에 대한 뷰 간 모션 예측, 뷰 간 잔차 예측 및/또는 백워드 워핑 뷰 합성 예측을 수행할 수도 있다.

[0161] 마찬가지로, 비디오 인코더 (20) 는 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스 (예컨대, NBDV 프로세스) 를 수행할 수도 있다. 현재 블록은 현재 뷰에 있을 수도 있다. 더욱이, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타낼 수도 있다. 이 예에서, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 비디오 인코더 (20) 는 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스 (예컨대, NBDV-R 프로세스) 를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다. 비디오 인코더 (20) 는 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 인코딩할 수도 있다. 예를 들면, 비디오 인코더 (20) 는 리파인된 디스패리티 벡터를 사용하여 현재 블록에 대한 뷰 간 모션 예측, 뷰 간 잔차 예측 및/또는 백워드 워핑 뷰 합성 예측을 수행할 수도 있다.

[0162] 몇몇 예들에서, NBDV 프로세스가 디스패리티 모션 벡터와 이웃 블록들에서의 IDV들을 체킹한 후 이용불가능한 결과를 반환하는 경우 (즉, 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내는 경우), 직접적으로 영 디스패리티 벡터를 사용하고 NBDV-R에서의 리파인먼트 프로세스를 스킵하는 대신, 비디오 코더가 NBDV-R 프로세스에서의 영 디스패리티 모션 벡터를 사용할 수도 있다. 그러므로, NBDV-R에서의 리파인먼트 프로세스는 깊이 뷰 성분에 액세스함으로써 디스패리티 벡터를 리파인하는데 사용될 수 있다.

[0163] 대안으로, 몇몇 예들에서, NBDV 프로세스가 이용불가능한 디스패리티 벡터를 반환하는 경우 (즉, 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내는 경우), 비디오 코더는 현재 블록의 가용성을 이용불가능으로 설정할 수도 있고 현재 블록의 디스패리티 벡터를 영으로 설정할 수도 있다. 다르게 말하면, 비디오 코더는 현재 블록에 대한 디스패리티 벡터가 이용가능함을 나타내기 위해 가용성 값을 수정할 수도 있다. 비디오 디코더는 그 다음에 디스패리티 벡터 리파인먼트 프로세스를 현재 블록에 대한 디스패리티 벡터에 적용할 수도 있다.

[0164] 위에서 나타낸 바와 같이, NBDV 프로세스는 가용성 값 (예컨대, availableDV) 을 반환할 수도 있다. 하나의 예에서, 0과 동일한 availableDV는 NBDV 프로세스가 이용불가능한 디스패리티 벡터를 반환하였음을 나타낼 수도 있다. 다르게 말하면, 0과 동일한 availableDV는 NBDV 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없었음을 나타낼 수도 있다. 1과 동일한 availableDV는 NBDV 프로세스가 이용가능한 디스패리티 벡터를 반환하였음을 나타낼 수도 있다. 다르게 말하면, 1과 동일한 availableDV는 NBDV 프로세스가 현재 블록에 대해 디스패리티 벡터를 도출할 수 있었음을 나타낼 수도 있다. 더욱이, 이 예에서, mvDisp가 현재 블록의 디스패리티 벡터를 나타내며, mvDisp[0]이 현재 블록의 디스패리티 벡터의 수평 성분을 나타내고, mvDisp[1]이 현재 블록의 디스패리티 벡터의 수직 성분을 나타낸다. 따라서, 이 예에서, availableDV가 0과 동일한 경우, 비디오 코더는 mvDisp[0]을 0으로 설정하며, mvDisp[1]을 0으로 설정하고, availableDV를 1로 설정할 수도 있다. 비디오 코더는 그 다음에 디스패리티 벡터 리파인먼트 프로세스를 mvDisp에 적용할 수도 있다.

[0165] 이런 식으로, 가용성 값 (예컨대, availableDV) 이 디스패리티 벡터 도출 프로세스 (예컨대, NBDV 프로세스) 가 현재 블록에 대한 디스패리티 벡터를 도출할 수 있었는지의 여부를 나타낼 수도 있다. 이 예에서, 디스패리티 벡터 도출 프로세스가 현재 블록에 대해 디스패리티 벡터를 도출할 수 없었음을 가용성 값이 나타내는 경우, 비디오 코더는 현재 블록에 대한 디스패리티 벡터가 이용가능함을 나타내는 가용성 값을 설정할 수도 있고 현재 블록에 대한 디스패리티 벡터를 영 디스패리티 벡터로 설정할 수도 있다.

[0166] 더욱이, 비디오 코더는 현재 블록의 디스패리티 벡터가 이용가능함을 나타내는 가용성 값을 설정하고 NBDV 프로세스가 이용불가능한 디스패리티 벡터를 반환하는 경우 현재 블록의 디스패리티 벡터를 영으로 설정하는 몇몇 예들에서, 비디오 코더는 NBDV 프로세스가 이용가능한 디스패리티 벡터를 원래 반환하였는지의 여부를 식별하기 위해 변수 (예컨대, availableDVRes) 를 유지할 수도 있다. 비디오 코더는 특정한 조건들에서 다른 코딩 도

구들에 대한 변수를 사용할 수도 있다. 예를 들면, 0과 동일한 이 변수는 비디오 코더가 현재 블록에 대한 뷰 간 잔차 예측을 불가능하게 하는 것으로 이어질 수도 있다. 따라서, 이 예에서, 비디오 코더는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 원래 도출하였는지의 여부를 나타내기 위해 변수를 유지할 수도 있다. 비디오 코더는 그 변수를 하나 이상의 코딩 도구들에 의한 사용을 위해 제공할 수도 있다. 더구나, 비디오 코더는 그 변수의 값에 기초하여 코딩 도구들 중 하나 이상을 인에이블할 수도 있다.

[0167] 비디오 코더가 현재 블록의 디스패리티 벡터의 가용성 값을 이용가능으로 설정하고 NBDV 프로세스가 이용불가능한 디스패리티 벡터를 반환하는 경우 현재 블록의 디스패리티 벡터를 영으로 설정하는 일부 예들에서, 비디오 코더는 오프셋을 현재 블록의 디스패리티 벡터 (즉, 영 디스패리티 벡터) 에 더할 수도 있다. 이 예에서, 비디오 코더는 디스패리티 벡터 리파인먼트 프로세스를 현재 블록에 대한 디스패리티 벡터에 적용하지 않는다. 다르게 말하면, 이 예에서는, 깊이 정보를 사용하는 리파인먼트 프로세스가 없다. 이런 이유로, 이 예에서, 디스패리티 벡터 도출 프로세스가 이용불가능한 디스패리티 벡터를 반환하는 경우, 비디오 코더는 오프셋을 영 디스패리티 벡터에 더하여 수정된 디스패리티 벡터를 생성할 수도 있다.

[0168] 이전 단락의 예에서, 비디오 코더는 다양한 방도들로 오프셋을 설정할 수도 있다. 예를 들어, 비디오 코더는 오프셋을 현재 블록의 디스패리티 벡터의 수평 성분에만 더할 수도 있다. 더욱이, 몇몇 예들에서, 비디오 코더 (또는 다른 디바이스) 는 카메라 파라미터들 및 디폴트 깊이 화소 값 (예컨대, 128) 으로 오프셋을 컴퓨팅할 수도 있다. 이런 이유로, 이 예에서, 비디오 코더는 하나 이상의 카메라 파라미터들 및 디폴트 깊이 화소 값에 기초하여 오프셋을 결정할 수도 있다. 몇몇 예들에서, 비디오 인코더 (20) 는 현재 블록을 포함하는 슬라이스의 슬라이스 헤더에서 오프셋을 시그널링한다. 이런 이유로, 그런 예들에서, 비디오 코더는 코딩된 비디오 비트스트림에서 수신된 시그널링된 값에 기초하여 오프셋을 결정할 수도 있다. 이러한 예들에서, 그 값은 슬라이스 헤더에서 시그널링될 수도 있다.

[0169] 본 개시물의 몇몇 기법들에 따라, 본 개시물에서 제공된 예들 중 하나 이상이 조합될 수 있다.

[0170] 도 6은 본 개시물의 기법들을 구현할 수도 있는 일 예의 비디오 인코더 (20) 를 도시하는 블록도이다. 도 6은 설명의 목적으로 제공되고 본 개시물에서 폭넓게 예시되고 설명된 바와 같은 기법들의 제한으로서 간주되지 않아야 한다. 설명의 목적으로, 본 개시물은 HEVC 코딩의 맥락에서 비디오 인코더 (20) 를 설명한다. 그러나, 본 개시물의 기법들은 다른 코딩 표준들 또는 방법들에 적용가능할 수도 있다.

[0171] 도 6의 예에서, 비디오 인코더 (20) 는 예측 프로세싱 유닛 (100), 잔차 생성 유닛 (102), 변환 프로세싱 유닛 (104), 양자화 유닛 (106), 역 양자화 유닛 (108), 역 변환 프로세싱 유닛 (110), 복원 유닛 (112), 필터 유닛 (114), 디코딩된 화상 버퍼 (116), 및 엔트로피 인코딩 유닛 (118) 을 포함한다. 예측 프로세싱 유닛 (100) 은 인터 예측 프로세싱 유닛 (120) 과 인트라 예측 프로세싱 유닛 (126) 을 구비한다. 인터 예측 프로세싱 유닛 (120) 은 모션 추정 유닛 (122) 과 모션 보상 유닛 (124) 을 구비한다. 다른 예들에서, 비디오 인코더 (20) 는 더 많거나, 더 적거나, 또는 상이한 기능성 컴포넌트들을 포함할 수도 있다.

[0172] 비디오 인코더 (20) 는 비디오 데이터를 수신할 수도 있다. 비디오 인코더 (20) 는 비디오 데이터의 화상의 슬라이스에서의 각각의 CTU를 인코딩할 수도 있다. CTU들의 각각은 화상의 동일 사이즈로 된 루마 코딩 트리 블록 (coding tree block; CTB) 들 및 대응하는 CTB들과 연관될 수도 있다. CTU를 인코딩하는 부분으로서, 예측 프로세싱 유닛 (100) 은 쿼드트리 파티셔닝을 수행하여 CTU의 CTB들을 점차적으로 더 작은 블록들로 분할할 수도 있다. 더 작은 블록들은 CU들의 코딩 블록들일 수도 있다. 예를 들어, 예측 프로세싱 유닛 (100) 은 CTU와 연관된 CTB를 4 개의 동일 사이즈로 된 서브-블록들로 파티셔닝하며, 그 서브-블록들 중 하나 이상을 4 개의 동일 사이즈로 된 서브 서브-블록들로 파티셔닝하는 등등을 수행할 수도 있다.

[0173] 비디오 인코더 (20) 는 CTU의 CU들을 인코딩하여 CU들의 인코딩된 표현들 (즉, 코딩된 CU들) 을 생성할 수도 있다. CU를 인코딩하는 부분으로서, 예측 프로세싱 유닛 (100) 은 CU의 하나 이상의 PU들 중에서 CU와 연관된 코딩 블록들을 파티셔닝할 수도 있다. 따라서, 각각의 PU는 루마 예측 블록 및 대응하는 크로마 예측 블록들과 연관될 수도 있다. 비디오 인코더 (20) 와 비디오 디코더 (30) 는 다양한 사이즈들을 갖는 PU들을 지원할 수도 있다. 위에서 나타낸 바와 같이, CU의 사이즈는 CU의 루마 코딩 블록의 사이즈를 말할 수도 있고 PU의 사이즈는 PU의 루마 예측 블록의 사이즈를 말할 수도 있다. 특정 CU의 사이즈가 2Nx2N이라고 가정하면, 비디오 인코더 (20) 와 비디오 디코더 (30) 는 인트라 예측을 위한 2Nx2N 또는 NxN의 PU 사이즈들과, 인터 예측을 위한 2Nx2N, 2NxN, Nx2N, NxN, 또는 유사한 것의 대칭적 PU 사이즈들을 지원할 수도 있다. 비디오 인코더 (20) 와 비디오 디코더 (30) 는 인터 예측을 위해 2NxnU, 2NxnD, nLx2N, 및 nRx2N의 PU 사이즈들에 대한 비대칭 파티셔닝을 또한 지원할 수도 있다.

[0174] 인터 예측 프로세싱 유닛 (120) 은 CU의 각각의 PU에 대해 인터 예측을 수행함으로써 PU에 대한 예측 데이터를 생성할 수도 있다. PU에 대한 예측 데이터는 PU의 예측 블록들 및 그 PU에 대한 모션 정보를 포함할 수도 있다. 인터 예측 프로세싱 유닛 (120) 은 PU가 I 슬라이스, P 슬라이스, 또는 B 슬라이스 중 어느 것에 있는지에 의존하여 CU의 PU에 대해 상이한 동작들을 수행할 수도 있다. I 슬라이스에서, 모든 PU들이 인트라 예측된다. 이런 이유로, PU가 I 슬라이스에 있으면, 인터 예측 프로세싱 유닛 (120) 은 PU에 대해 인터 예측을 수행하지 않는다.

[0175] PU가 P 슬라이스에 있으면, 모션 추정 유닛 (122) 은 PU에 대한 참조 지역을 찾아 참조 화상들의 리스트 (예컨대, "RefPicList0") 에서의 참조 화상들을 검색할 수도 있다. PU에 대한 참조 지역은, 참조 화상 내의, PU의 예측 블록들에 가장 밀접하게 대응하는 샘플들을 포함하는 지역일 수도 있다. 모션 추정 유닛 (122) 은 PU에 대한 참조 지역을 포함하는 참조 화상의 RefPicList0에서의 포지션을 나타내는 참조 인덱스를 생성할 수도 있다. 덧붙여서, 모션 추정 유닛 (122) 은 PU의 예측 블록 및 참조 지역과 연관된 참조 로케이션 사이의 공간적 변위를 나타내는 모션 벡터를 생성할 수도 있다. 예를 들면, 모션 벡터는 현재 화상에서의 좌표들로부터 참조 화상에서의 좌표들로의 오프셋을 제공하는 2차원 벡터일 수도 있다. 모션 추정 유닛 (122) 은 참조 인덱스 및 모션 벡터를 PU의 모션 정보로서 출력할 수도 있다. 모션 보상 유닛 (124) 은 PU의 모션 벡터에 의해 나타내어진 참조 로케이션에 있는 실제 또는 보간된 샘플들에 기초하여 PU의 예측 블록들을 생성할 수도 있다.

[0176] PU가 B 슬라이스에 있다면, 모션 추정 유닛 (122) 은 PU에 대해 단예측 또는 양예측을 수행할 수도 있다. PU에 대해 단예측을 수행하기 위해, 모션 추정 유닛 (122) 은 PU에 대한 참조 지역을 찾아 RefPicList0 또는 제 2 참조 화상 리스트 (예컨대, "RefPicList1") 의 참조 화상들을 검색할 수도 있다. 모션 추정 유닛 (122) 은, PU의 모션 정보로서, 참조 지역을 포함하는 참조 화상의 RefPicList0 또는 RefPicList1에서의 포지션을 나타내는 참조 인덱스, PU의 예측 블록 및 참조 지역과 연관된 참조 로케이션 사이의 공간적 변위를 나타내는 모션 벡터, 및 참조 화상이 RefPicList0에 있는지 또는 RefPicList1에 있는지를 나타내는 하나 이상의 예측 방향 표시자들을 출력할 수도 있다. 모션 보상 유닛 (124) 은 PU의 모션 벡터에 의해 나타내어진 참조 로케이션에 있는 실제 또는 보간된 샘플들에 적어도 부분적으로 기초하여 PU의 예측 블록들을 생성할 수도 있다.

[0177] PU에 대해 양방향 인터 예측을 수행하기 위해, 모션 추정 유닛 (122) 은 그 PU에 대한 참조 지역을 찾아 RefPicList0에서의 참조 화상들을 검색할 수도 있고, 또한 그 PU에 대한 다른 참조 지역을 찾아 RefPicList1에서의 참조 화상들에서 검색할 수도 있다. 모션 추정 유닛 (122) 은 참조 지역들을 포함하는 참조 화상들의 RefPicList0 및 RefPicList1에서의 포지션들을 나타내는 참조 인덱스들을 생성할 수도 있다. 덧붙여서, 모션 추정 유닛 (122) 은 그 참조 지역들과 연관된 참조 로케이션들 및 PU의 예측 블록 사이의 공간적 변위들을 나타내는 모션 벡터들을 생성할 수도 있다. PU의 모션 정보는 PU의 참조 인덱스들 및 모션 벡터들을 포함할 수도 있다. 모션 보상 유닛 (124) 은 PU의 모션 벡터들에 의해 나타내어진 참조 로케이션들에 있는 실제 또는 보간된 샘플들에 적어도 부분적으로 기초하여 PU의 예측 블록들을 생성할 수도 있다.

[0178] 인트라 예측 프로세싱 유닛 (126) 은 PU에 대해 인트라 예측을 수행함으로써 그 PU에 대한 예측 데이터를 생성할 수도 있다. PU에 대한 예측 데이터는 PU에 대한 예측 블록들과 다양한 신택스 엘리먼트들을 포함할 수도 있다. 인트라 예측 프로세싱 유닛 (126) 은 I 슬라이스들, P 슬라이스들, 및 B 슬라이스들에서의 PU들에 대해 인트라 예측을 수행할 수도 있다.

[0179] PU에 대해 인트라 예측을 수행하기 위해, 인트라 예측 프로세싱 유닛 (126) 은 PU에 대한 예측 블록들의 다수의 세트들을 생성하기 위해 다수의 인트라 예측 모드들을 사용할 수도 있다. 특정 인트라 예측 모드를 사용하여 인트라 예측을 수행하는 경우, 인트라 예측 프로세싱 유닛 (126) 은 이웃하는 블록들로부터의 샘플들의 특정 세트를 사용하여 PU에 대한 예측 블록들을 생성할 수도 있다. PU들, CU들, 및 CTU 들에 대한 좌측에서 우측으로, 상단에서 하단으로의 인코딩 순서를 가정하면, 이웃하는 블록들은 PU의 상측, 우상측, 좌상측, 또는 좌측에 있을 수도 있다. 인트라 예측 프로세싱 유닛 (126) 은 다양한 수들의 인트라 예측 모드들, 예컨대, 33 개의 방향성 인트라 예측 모드들을 사용할 수도 있다. 몇몇 예들에서, 인트라 예측 모드들의 수는 PU의 예측 블록들의 사이즈에 의존할 수도 있다.

[0180] 몇몇 예들에서, 인터 예측 프로세싱 유닛 (120) 은 현재 블록 (예컨대, CU, PU 등) 에 대한 디스패리티 벡터를 도출할 수도 있다. 디스패리티 벡터는 뷰 간 모션 예측, 뷰 간 잔차 예측, 백워드 워핑 뷰 합성 예측 등을 지원할 수도 있다. 본 개시물의 하나 이상의 기법들에 따라, 인터 예측 프로세싱 유닛 (120) 은 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다. 디스패리티 벡터 도

출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타낸다. 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 인터 예측 프로세싱 유닛 (120) 은 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다.

[0181] 예측 프로세싱 유닛 (100) 은 CU의 PU들에 대한 예측 데이터를, 그 PU들에 대해 인터 예측 프로세싱 유닛 (120) 에 의해 생성된 예측 데이터 또는 그 PU들에 대해 인트라 예측 프로세싱 유닛 (126) 에 의해 생성된 예측 데이터 중에서 선택할 수도 있다. 몇몇 예들에서, 예측 프로세싱 유닛 (100) 은 예측 데이터의 세트들의 레이트/왜곡 메트릭들에 기초하여 CU의 PU들에 대한 예측 데이터를 선택한다. 선택된 예측 데이터의 예측 블록들은 본원에서는 선택된 예측 블록들이라고 지칭될 수도 있다.

[0182] 잔차 생성 유닛 (102) 은, CU의 루마, Cb 및 Cr 코딩 블록들 및 그 CU의 PU들의 선택된 예측 루마, Cb 및 Cr 블록들에 기초하여, CU의 루마, Cb 및 Cr 잔차 블록들을 생성할 수도 있다. 예를 들면, 잔차 생성 유닛 (102) 은 CU의 잔차 블록들에서의 각각의 샘플이 그 CU의 코딩 블록에서의 샘플 및 그 CU의 PU의 대응하는 선택된 예측 블록에서의 대응하는 샘플 사이의 차이와 동일한 값을 가지도록 CU의 잔차 블록들을 생성할 수도 있다.

[0183] 변환 프로세싱 유닛 (104) 은 쿼드트리 파티셔닝을 수행하여 CU의 잔차 블록들을 그 CU의 TU들과 연관된 변환 블록들로 파티셔닝할 수도 있다. 따라서, TU가 루마 변환 블록 및 2 개의 대응하는 크로마 변환 블록들과 연관될 수도 있다. CU의 TU들의 루마 및 크로마 변환 블록들의 사이즈들 및 포지션들은 그 CU의 PU들의 예측 블록들의 사이즈들 및 포지션들에 기초할 수도 있거나 또는 기초하지 않을 수도 있다.

[0184] 변환 프로세싱 유닛 (104) 은 CU의 각각의 TU에 대한 변환 계수 블록들을, 하나 이상의 변환들을 TU의 변환 블록들에 적용함으로써 생성할 수도 있다. 변환 프로세싱 유닛 (104) 은 다양한 변환들을 TU와 연관된 변환 블록에 적용할 수도 있다. 예를 들어, 변환 프로세싱 유닛 (104) 은 이산 코사인 변환 (DCT), 방향성 변환, 또는 개념적으로 유사한 변환을 변환 블록에 적용할 수도 있다. 몇몇 예들에서, 변환 프로세싱 유닛 (104) 은 변환들을 변환 블록에 적용하지 않는다. 그런 예들에서, 변환 블록은 변환 계수 블록으로서 다루어질 수도 있다.

[0185] 양자화 유닛 (106) 은 계수 블록에서의 변환 계수들을 양자화할 수도 있다. 양자화 프로세스는 변환 계수들의 일부 또는 전부와 연관된 비트 깊이를 감소시킬 수도 있다. 예를 들어, n-비트 변환 계수가 양자화 동안에 m-비트 변환 계수로 버림될 (rounded down) 수도 있으며, 여기서 n은 m보다 크다. 양자화 유닛 (106) 은 CU와 연관된 양자화 파라미터 (QP) 값에 기초하여 그 CU의 TU와 연관된 계수 블록을 양자화할 수도 있다. 비디오 인코더 (20) 는 CU와 연관된 QP 값을 조정함으로써 그 CU와 연관된 계수 블록들에 적용되는 양자화 정도를 조정할 수도 있다. 양자화는 정보의 손실을 도입할 수도 있고, 그러므로, 양자화된 변환 계수들은 원래의 것들보다 낮은 정밀도를 가질 수도 있다.

[0186] 역 양자화 유닛 (108) 과 역 변환 프로세싱 유닛 (110) 은 계수 블록으로부터 잔차 블록을 복원하기 위해 역 양자화 및 역 변환들을 계수 블록에 각각 적용할 수도 있다. 복원 유닛 (112) 은 TU와 연관된 복원된 변환 블록을 생성하기 위해 복원된 잔차 블록을 예측 프로세싱 유닛 (100) 에 의해 생성된 하나 이상의 예측 블록들로부터의 대응하는 샘플들에 부가할 수도 있다. 복원 유닛 (112) 은 합산기라고 또한 지칭될 수도 있다. CU의 각각의 TU에 대한 변환 블록들을 이런 식으로 복원함으로써, 비디오 인코더 (20) 는 CU의 코딩 블록들을 복원할 수도 있다.

[0187] 필터 유닛 (114) 은 하나 이상의 블록화제거 동작들을 수행하여 CU와 연관된 코딩 블록들에서의 블록화 아티팩트들을 감소시킬 수도 있다. 디코딩된 화상 버퍼 (116) 는, 필터 유닛 (114) 이 복원된 코딩 블록들에 대해 하나 이상의 블록화제거 동작들을 수행한 후에 복원된 코딩 블록들을 저장할 수도 있다. 인터 예측 프로세싱 유닛 (120) 은 다른 화상들의 PU들에 대해 인터 예측을 수행하기 위해 복원된 코딩 블록들을 포함하는 참조 화상을 사용할 수도 있다. 덧붙여서, 인트라 예측 프로세싱 유닛 (126) 은 CU와 동일한 화상에서의 다른 PU들에 대해 인트라 예측을 수행하기 위해 디코딩된 화상 버퍼 (116) 에서의 복원된 코딩 블록들을 사용할 수도 있다. 디코딩된 화상 버퍼 (116) 는 참조 화상 메모리라고 또한 지칭될 수도 있다. 이런 이유로, 디코딩된 화상 버퍼 (116) 는 비디오 데이터, 이를테면 멀티-뷰 비디오 데이터를 저장하는 메모리를 포함할 수도 있다.

[0188] 엔트로피 인코딩 유닛 (118) 은 비디오 인코더 (20) 의 다른 기능성 컴포넌트들로부터 데이터를 수신할 수도 있

다. 예를 들어, 엔트로피 인코딩 유닛 (118) 은 양자화 유닛 (106) 으로부터 계수 블록들을 수신할 수도 있고 예측 프로세싱 유닛 (100) 으로부터 신택스 엘리먼트들을 수신할 수도 있다. 엔트로피 인코딩 유닛 (118) 은 데이터에 대해 하나 이상의 엔트로피 인코딩 동작들을 수행하여 엔트로피 인코딩된 데이터를 생성할 수도 있다. 예를 들어, 엔트로피 인코딩 유닛 (118) 은 그 데이터에 대해 콘텍스트 적응 가변 길이 코딩 (context-adaptive variable length coding; CAVLC) 동작, CABAC 동작, 가변 대 가변 (variable-to-variable; V2V) 길이 코딩 동작, 신택스 기반 콘텍스트 적응 이진 산술 코딩 (syntax-based context-adaptive binary arithmetic coding; SBAC) 동작, 확률 간격 파티셔닝 엔트로피 (Probability Interval Partitioning Entropy; PIPE) 코딩 동작, 지수-골롬 (Exponential-Golomb) 인코딩 동작, 또는 다른 유형의 엔트로피 인코딩 동작을 수행할 수도 있다. 비디오 인코더 (20) 는 엔트로피 인코딩 유닛 (118) 에 의해 생성된 엔트로피 인코딩된 데이터를 포함하는 비트스트림을 출력할 수도 있다.

[0189] 도 7은 본 개시물의 기법들을 구현하도록 구성되는 일 예의 비디오 디코더 (30) 를 도시하는 블록도이다. 도 7은 설명의 목적으로 제공되고 본 개시물에서 폭넓게 예시되고 설명된 바와 같은 기법들로 제한하고 있지는 않다. 설명의 목적으로, 본 개시물은 HEVC 코딩의 맥락에서 비디오 디코더 (30) 를 설명한다. 그러나, 본 개시물의 기법들은 다른 코딩 표준들 또는 방법들에 적용가능할 수도 있다.

[0190] 도 7의 예에서, 비디오 디코더 (30) 는 엔트로피 디코딩 유닛 (150), 예측 프로세싱 유닛 (152), 역 양자화 유닛 (154), 역 변환 프로세싱 유닛 (156), 복원 유닛 (158), 필터 유닛 (160), 및 디코딩된 화상 버퍼 (162) 를 구비한다. 예측 프로세싱 유닛 (152) 은 모션 보상 유닛 (164) 과 인트라 예측 프로세싱 유닛 (166) 을 구비한다. 다른 예들에서, 비디오 디코더 (30) 는 더 많거나, 더 적거나, 또는 상이한 기능성 컴포넌트들을 포함할 수도 있다.

[0191] 코딩된 화상 버퍼 (coded picture buffer; CPB) (151) 가 비트스트림의 인코딩된 비디오 데이터 (예컨대, NAL 유닛들) 를 수신하고 저장할 수도 있다. 엔트로피 디코딩 유닛 (150) 은 CPB (151) 로부터 NAL 유닛들을 수신하고 그 NAL 유닛들을 파싱하여 비트스트림으로부터 신택스 엘리먼트들을 획득할 수도 있다. 엔트로피 디코딩 유닛 (150) 은 NAL 유닛들에서의 엔트로피 인코딩된 신택스 엘리먼트들을 엔트로피 디코딩할 수도 있다. 예측 프로세싱 유닛 (152), 역 양자화 유닛 (154), 역 변환 프로세싱 유닛 (156), 복원 유닛 (158), 및 필터 유닛 (160) 은 비트스트림으로부터 추출된 신택스 엘리먼트들에 기초하여 디코딩된 비디오 데이터를 생성할 수도 있다.

[0192] 비트스트림의 NAL 유닛들은 코딩된 슬라이스 NAL 유닛들을 포함할 수도 있다. 비트스트림을 디코딩하는 부분으로서, 엔트로피 디코딩 유닛 (150) 은 코딩된 슬라이스 NAL 유닛들로부터 신택스 엘리먼트들을 추출하고 엔트로피 디코딩할 수도 있다. 코딩된 슬라이스들의 각각은 슬라이스 헤더 및 슬라이스 데이터를 포함할 수도 있다. 슬라이스 헤더는 슬라이스에 관계된 신택스 엘리먼트들을 포함할 수도 있다.

[0193] 비트스트림으로부터 신택스 엘리먼트들을 획득하는 것에 더하여, 비디오 디코더 (30) 는 CU에 대해 디코딩 동작을 수행할 수도 있다. CU에 대해 디코딩 동작을 수행함으로써, 비디오 디코더 (30) 는 그 CU의 코딩 블록들을 복원할 수도 있다.

[0194] CU에 대해 디코딩 동작을 수행하는 부분으로서, 역 양자화 유닛 (154) 은 그 CU의 TU들과 연관된 계수 블록들을 역 양자화, 즉, 탈양자화 (de-quantization) 할 수도 있다. 역 양자화 유닛 (154) 은 TU의 CU와 연관된 QP 값을 사용하여 양자화 정도를 결정하고, 비슷하게, 역 양자화 유닛 (154) 에 대해 적용할 역 양자화 정도를 결정할 수도 있다. 다시 말하면, 압축 비율, 즉, 원래의 시퀀스 및 압축된 시퀀스를 표현하는데 사용된 비트들의 수의 비율은, 변환 계수들을 양자화하는 경우에 사용된 QP의 값을 조정함으로써 제어될 수도 있다. 압축 비율은 채용된 엔트로피 코딩하는 방법에 또한 의존할 수도 있다.

[0195] 역 양자화 유닛 (154) 이 계수 블록을 역 양자화한 후, 역 변환 프로세싱 유닛 (156) 은 TU와 연관된 잔차 블록을 생성하기 위하여 하나 이상의 역 변환들을 계수 블록에 적용할 수도 있다. 예를 들어, 역 변환 프로세싱 유닛 (156) 은 역 DCT, 역 정수 변환, 역 카루넨-뢰베 변환 (Karhunen-Loeve transform; KLT), 역 회전 변환, 역 방향성 변환, 또는 다른 역 변환을 계수 블록에 적용할 수도 있다.

[0196] PU가 인트라 예측을 사용하여 인코딩되면, 인트라 예측 프로세싱 유닛 (166) 은 PU에 대한 예측 블록들을 생성하기 위해 인트라 예측을 수행할 수도 있다. 인트라 예측 프로세싱 유닛 (166) 은 인트라 예측 모드를 사용하여 공간적으로 이웃하는 PU들의 예측 블록들에 기초하여 PU에 대한 예측 루마, Cb 및 Cr 블록들을 생성할 수도 있다. 인트라 예측 프로세싱 유닛 (166) 은 비트스트림으로부터 디코딩된 하나 이상의 신택스 엘리먼트

들에 기초하여 PU에 대한 인트라 예측 모드를 결정할 수도 있다.

[0197] 예측 프로세싱 유닛 (152) 은 비트스트림으로부터 추출된 신택스 엘리먼트들에 기초하여 제 1 참조 화상 리스트 (RefPicList0) 및 제 2 참조 화상 리스트 (RefPicList1) 를 구축할 수도 있다. 더욱이, PU가 인터 예측을 사용하여 인코딩되면, 엔트로피 디코딩 유닛 (150) 은 그 PU에 대한 모션 정보를 획득할 수도 있다. 모션 보상 유닛 (164) 은, PU의 모션 정보에 기초하여, PU에 대한 하나 이상의 참조 지역들을 결정할 수도 있다. 모션 보상 유닛 (164) 은, PU에 대한 하나 이상의 참조 블록들에서의 샘플들에 기초하여, PU에 대한 예측 루마, Cb 및 Cr 블록들을 생성할 수도 있다.

[0198] 몇몇 예들에서, 예측 프로세싱 유닛 (152) 은 현재 블록 (예컨대, CU, PU 등) 에 대한 디스패리티 벡터를 도출할 수도 있다. 디스패리티 벡터는 뷰 간 모션 예측, 뷰 간 잔차 예측, 백워드 워핑 뷰 합성 예측 등을 지원할 수도 있다. 본 개시물의 하나 이상의 기법들에 따라, 예측 프로세싱 유닛 (152) 은 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다. 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타낸다. 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우, 예측 프로세싱 유닛 (152) 은 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다.

[0199] 복원 유닛 (158) 은, 적용가능한 것으로서, CU의 TU들과 연관된 루마, Cb 및 Cr 변환 블록들과 그 CU의 PU들의 예측 루마, Cb 및 Cr 블록들로부터의 잔차 값들, 즉 인트라 예측 데이터 또는 인터 예측 데이터 중 어느 하나를 사용하여 그 CU의 루마, Cb 및 Cr 코딩 블록들을 복원할 수도 있다. 예를 들어, 복원 유닛 (158) 은 루마, Cb 및 Cr 변환 블록들의 샘플들을 예측 루마, Cb 및 Cr 블록들의 대응하는 샘플들에 부가하여 CU의 루마, Cb 및 Cr 코딩 블록들을 복원할 수도 있다. 복원 유닛 (158) 은 합산기라고 또한 지칭될 수도 있다.

[0200] 필터 유닛 (160) 은 블록화제거 동작을 수행하여 CU의 루마, Cb 및 Cr 코딩 블록들과 연관된 블록화 아티팩트들을 감소시킬 수도 있다. 비디오 디코더 (30) 는 CU의 루마, Cb 및 Cr 코딩 블록들을 디코딩된 화상 버퍼 (162) 에 저장할 수도 있다. 디코딩된 화상 버퍼 (162) 는 참조 화상 메모리라고 또한 지칭될 수도 있다. 이런 이유로, 디코딩된 화상 버퍼 (162) 는 비디오 데이터, 이를테면 멀티-뷰 비디오 데이터를 저장하는 메모리를 포함할 수도 있다. 디코딩된 화상 버퍼 (162) 는 후속하는 모션 보상, 인트라 예측, 및 디스플레이 디바이스, 이를테면 도 1의 디스플레이 디바이스 (32) 상의 프레젠테이션을 위해 참조 화상들을 제공할 수도 있다. 예를 들면, 비디오 디코더 (30) 는, 디코딩된 화상 버퍼 (162) 에서의 루마, Cb 및 Cr 블록들에 기초하여, 다른 CU들의 PU들에 대해 인트라 예측 또는 인터 예측 동작들을 수행할 수도 있다. 이런 식으로, 비디오 디코더 (30) 는, 비트스트림으로부터, 유효 (significant) 루마 계수 블록의 변환 계수 레벨들을 추출하며, 그 변환 계수 레벨들을 역 양자화하며, 그 변환 계수 레벨들에 변환을 적용하여 변환 블록을 생성하며, 그 변환 블록에 적어도 부분적으로 기초하여, 코딩 블록을 생성하고, 그 코딩 블록을 디스플레이를 위해 출력할 수도 있다.

[0201] 위에서 설명된 바와 같이, 비디오 코더 (예컨대, 비디오 인코더 (20) 또는 비디오 디코더 (30)) 가 디스패리티 벡터를 결정하기 위해 NBDV 프로세스를 사용할 수도 있다. 더욱이, 비디오 코더는 디스패리티 벡터에 대해 리파인먼트 프로세스를 수행할 수도 있다. 본 개시물의 몇몇 기법들에 따라, 디스패리티 벡터가 이용불가능하다고 NDBV 프로세스가 결정하는 경우 비디오 코더는 영 디스패리티 벡터에 대해 리파인먼트 프로세스를 수행할 수도 있다. 영 디스패리티 벡터의 수평 성분 및 수직 성분이 0과 동일하다. 이러한 기법들에 따라, 3D HEVC 테스트 모델 설명 초안 3의 하위절 H.8.5.4는 다음과 같이 수정될 수도 있다. 다음의 텍스트에서, 추가된 내용은 밑줄이 그어져 있다.

[0202] **H.8.5.4 디스패리티 벡터에 대한 도출 프로세스**

[0203] ...

[0204] availableDV가 1과 동일하고 deriveFromDepthFlag가 1과 동일한 경우, 다음 순서의 단계들이 적용된다:

[0205] 1. 하위절 H.8.5.4.3에서 특정된 바와 같은 디스패리티 샘플 어레이에 대한 도출 프로세스는 루마 로케이션들 (xP, yP), 디스패리티 벡터 (mvDisp), 뷰 식별자 (refViewIdx), 변수들 (nPSW, nPSH), nPSW와 동일한 변수 nSubBlkW, nPSH와 동일한 변수 nSubBlkH, 및 1과 동일한 플래그 restMaxSearchFlag를 입력들로 하여 호출되고, 출력은 사이즈 (nPSWDs)x(nPSHDs)의 어레이 disparitySamples이다.

[0206] 디스패리티 벡터 mvDisp[0]의 수평 성분은 disparitySamples[0][0]과 동일하게 설정된다.

[0207] 그렇지 않고, availableDV가 0과 동일하고 deriveFromDepthFlag가 1과 동일한 경우, 다음 순서의 단계들이 적용된다:

[0208] 1. 하위절 H.8.5.4.3에서 특정된 바와 같은 디스패리티 샘플 어레이에 대한 도출 프로세스는 루마 로케이션들 (xP, yP), 0과 동일한 디스패리티 벡터 (mvDisp), 뷰 식별자 (refViewIdx), 변수들 (nPSW, nPSH), nPSW와 동일한 변수 nSubBlkW, nPSH와 동일한 변수 nSubBlkH, 및 1과 동일한 플래그 restMaxSearchFlag를 입력들로 하여 호출되고, 출력은 사이즈 (nPSWDs)x(nPSHDs)의 어레이 disparitySamples이다.

[0209] 2. 디스패리티 벡터 mvDisp[0]의 수평 성분은 disparitySamples[0][0]과 동일하게 설정된다.

[0210] 위에서 나타낸 바와 같이, 하위절 H.8.5.4.3 "디스패리티 샘플 어레이에 대한 도출 프로세스"는, 하위절 H.8.5.4 "디스패리티 벡터에 대한 도출 프로세스"에서 찾아진 이용가능한 디스패리티 모션 벡터가 없는 경우, 영 디스패리티 벡터를 연관된 깊이 정보를 사용하여 리파인하기 위해 호출된다. 3D HEVC 테스트 모델 설명 초안 3의 하위절 H.8.5.4.3이 아래와 같이 재현된다.

[0211] **H.8.5.4.3 디스패리티 샘플 어레이에 대한 도출 프로세스**

[0212] 이 프로세스에 대한 입력들은 다음이 된다:

[0213] - 현재 화상의 좌측상단 루마 샘플에 대하여 현재 예측 유닛의 좌측상단 루마 샘플의 루마 로케이션 (xP, yP),

[0214] - 디스패리티 벡터 mvDisp,

[0215] - 참조 뷰를 특정하는 뷰 식별자 refViewIdx,

[0216] - 현재 예측 유닛의 폭 및 높이를 각각 특정하는 변수들 (nPSW 및 nPSH),

[0217] - 대응 깊이 샘플들의 변환 정밀도를 특정하는 변수 nSubBlkW와 nSubBlkW,

[0218] - 최대 디스패리티에 대한 검색이 제한되는지의 여부를 특정하는 플래그 restMaxSearchFlag.

[0219] 이 프로세스의 출력들은 다음이 된다:

[0220] - 디스패리티 값들의 (nPSW)x(nPSH) 어레이 disparitySamples.

[0221] refDepPels를 refViewIdx와 동일한 ViewIdx를 갖는 깊이 뷰 성분의 복원된 깊이 샘플들의 어레이라고 하자. refDepPels의 좌측상단 루마 샘플의 루마 로케이션 (x_{TL}, y_{TL}) 이,

[0222] x_{TL} = Clip3(0, pic_width_in_luma_samples - nPSW - 1, xP +

[0223] (mvDisp[0] >> 2)) (H-291)

[0224] y_{TL} = Clip3(0, pic_height_in_luma_samples - nPSH - 1, yP +

[0225] (mvDisp[1] >> 2)) (H-292)

[0226] 에 의해 도출된다.

[0227] 사이즈 (nPSW)x(nPSH)의 어레이 disparitySamples는 다음에서 특정된 바와 같이 도출된다:

[0228] - 0 내지 ((nPSH / nSubBlkH) -1) 의 범위에서의 sBy에 대해, 다음이 적용된다:

[0229] - 0 내지 ((nPSW / nSubBlkW) -1) 의 범위에서의 sBx에 대해, 다음이 적용된다:

[0230] - 변수 maxDep는 -1과 동일하게 설정되고 다음에서 특정된 바와 같이 도출된다.

[0231] - restMaxSearchFlag가 0과 동일하면, 다음이 적용된다:

[0232] for (yOff = 0; yOff < nSubBlkH; yOff++)

[0233] for (xOff = 0; xOff < nSubBlkW; xOff++) {

[0234] x = x_{TL} + sBx * nSubBlkW + xOff

[0235] y = y_{TL} + sBy * nSubBlkH + yOff

[0236] maxDep = Max (maxDep, refDepPels[x][y])

[0237] }

[0238] - 그렇지 않고 (restMaxSearchFlag가 1과 동일하면), 다음이 적용된다:

[0239] x = x_{TL} + sBx * nSubBlkW

[0240] y = y_{TL} + sBy * nSubBlkH

[0241] maxDep = Max(maxDep, refDepPels[x][y])

[0242] maxDep = Max(maxDep, refDepPels[x][y + nSubBlkH - 1])

[0243] maxDep = Max(maxDep, refDepPels[x + nSubBlkW - 1][y])

[0244] maxDep = Max(maxDep, refDepPels[x + nSubBlkW - 1][y + nSubBlkH - 1])

[0245] maxDep = Max(maxDep, refDepPels[x + nSubBlkW/2][y + nSubBlkH/2])

[0246] - 어레이 depthSamples의 값들은 다음에서 특정된 바와 같이 도출된다:

[0247] for (yOff = 0; yOff < nSubBlkH; yOff++)

[0248] for (xOff = 0; xOff < nSubBlkW; xOff++) {

[0249] x = sBx * nSubBlkW + xOff

[0250] y = sBy * nSubBlkH + yOff

[0251] disparitySamples[x][y] =

[0252] DepthToDisparityB[refViewIdx][maxDep]

[0253] }

[0254] 본 개시물의 다른 기법들에서, NBDV 프로세스가 이용가능한 디스패리티 모션 벡터 또는 임의의 이웃 블록에서의 IDV를 찾지 못하는 경우에도 비디오 코더는 NBDV 결과에 대한 가용성 값을 1로 항상 설정할 수도 있다. 이러한 몇몇 기법들에 따라, 3D HEVC 테스트 모델 설명 초안 3의 하위절 H.8.5.4는 다음과 같이 수정될 수도 있다. 다음의 텍스트에서, 추가된 내용은 밑줄이 그어져 있다. 아래에서 도시되지 않은 3D HEVC 테스트 모델 설명 초안 3의 하위절 H.8.5.4의 부분들은 3D HEVC 테스트 모델 설명 초안 3에서와 동일할 수도 있다.

[0255] **H.8.5.4 디스패리티 벡터에 대한 도출 프로세스**

[0256] ...

[0257] availableDV가 0과 동일한 경우, mvDisp[0]은 0으로 설정되며, mvDisp[1]은 0으로 설정되고, availableDV는 1로 설정된다.

[0258] availableDV가 1과 동일하고 deriveFromDepthFlag가 1과 동일한 경우, 다음 순서의 단계들이 적용된다:

[0259] 1. 하위절 H.8.5.4.3에서 특정된 바와 같은 디스패리티 샘플 어레이에 대한 도출 프로세스는 루마 로케이션들 (xP, yP), 디스패리티 벡터 (mvDisp), 뷰 식별자 (refViewIdx), 변수들 (nPSW, nPSH), nPSW와 동일한 변수 nSubBlkW, nPSH와 동일한 변수 nSubBlkH, 및 1과 동일한 플래그 restMaxSearchFlag를 입력들로 하여 호출되고, 출력은 사이즈 (nPSWDs)x(nPSHDs)의 어레이 disparitySamples이다.

[0260] 디스패리티 벡터 mvDisp[0]의 수평 성분은 disparitySamples[0][0]과 동일하게 설정된다.

[0261] 대안적인 예에서, 변수 availableDVRes가 뷰 간 잔차 예측에 대한 이용불가능한 NBDV 결과를 사용하여 불가능하도록 도입될 수도 있다. 이 예에서, 변수 availableDVRes는 뷰 간 잔차 예측을 수행할지의 여부를 제어하기 위해 추가로 사용될 수도 있다. 이 예에서, availableDV를 3D HEVC 테스트 모델 설명 초안 3의 하위절

H.8.5.4의 출력으로서 반환하는 대신, availableDVRes가 출력이 되고 뷰 간 잔차 예측을 위해 추가로 사용된다.

이 예에 따라, 3D HEVC 테스트 모델 설명 초안 3의 하위절들 H.8.5.4 및 H.8.5.2.2.6은 다음과 같이 수정될 수도 있다. 다음의 텍스트에서, 추가된 내용은 밑줄이 그어져 있고 삭제된 내용은 이탤릭체와 이중 대괄호로 묶여 있다. 아래에서 도시되지 않은 3D HEVC 테스트 모델 설명 초안 3의 하위절들 H.8.5.4 및 H.8.5.2.2.6의 부분들은 3D HEVC 테스트 모델 설명 초안 3에서와 동일할 수도 있다.

[0262] **H.8.5.4 디스패리티 벡터에 대한 도출 프로세스**

[0263] ...

[0264] availableDV가 0과 동일한 경우, mvDisp[0]은 0으로 설정되고, mvDisp[1]은 0으로 설정되며, availableDVRes는 availableDV로 설정된다.

[0265] *[[availableDV가 1과 동일하고]]* deriveFromDepthFlag가 1과 동일한 경우, 다음 순서의 단계들이 적용된다:

[0266] 2. 하위절 H.8.5.4.3에서 특정된 바와 같은 디스패리티 샘플 어레이에 대한 도출 프로세스는 루마 로케이션들 (xP, yP), 디스패리티 벡터 (mvDisp), 뷰 식별자 (refViewIdx), 변수들 (nPSW, nPSH), nPSW와 동일한 변수 nSubBlkW, nPSH와 동일한 변수 nSubBlkH, 및 1과 동일한 플래그 restMaxSearchFlag를 입력들로 하여 호출되고, 출력은 사이즈 (nPSWDs)x(nPSHDs)의 어레이 disparitySamples이다.

[0267] 디스패리티 벡터 mvDisp[0]의 수평 성분은 disparitySamples[0][0]과 동일하게 설정된다.

[0268] **H.8.5.2.2.6 뷰 간 잔차 예측 프로세스**

[0269] 그 프로세스는 res_pred_flag가 1과 동일한 경우에만 호출된다.

[0270] 이 프로세스에 대한 입력들은 다음이 된다:

[0271] - 현재 화상의 좌측상단 루마 샘플에 대하여 현재 루마 코딩 블록의 좌측상단 샘플을 특정하는 루마 로케이션 (xC, yC),

[0272] - 현재 화상의 좌측상단 루마 샘플에 대하여 현재 예측 유닛의 좌측상단 루마 샘플의 루마 로케이션 (xP, yP),

[0273] - 현재 루마 코딩 블록의 사이즈를 특정하는 변수 nCS,

[0274] - 현재 예측 유닛의 폭 및 높이를 각각 특정하는 변수들 (nPSW 및 nPSH), 예측 리스트 이용 플래그들 predFlagL0 및 predFlagL1,

[0275] - 루마 예측 샘플들의 (nPSW)x(nPSH) 어레이 $predSamples_L$,

[0276] - 크로마 예측 샘플들의 2 개의 (nPSW/2)x(nPSH/2) 어레이들 $predSamples_{Cb}$ 및 $predSamples_{Cr}$.

[0277] 이 프로세스의 출력은 다음이 된다:

[0278] - 수정된 버전의 (nPSW)x(nPSH) 어레이 $predSamples_L$,

[0279] - 수정된 버전들의 (nPSW/2)x(nPSH/2) 어레이들 $predSamples_{Cb}$ 및 $predSamples_{Cr}$.

[0280] 하위절 H.8.5.4에서 특정된 바와 같은 디스패리티 벡터에 대한 도출 프로세스는 루마 로케이션들 (xC, yC) 및 (xP, yP), 코딩 블록 사이즈 nCS, 변수들 (nPSW 및 nPSH), 파티션 인덱스 partIdx 및 0과 동일한 변수 deriveFromDepthFlag를 입력들로 하여 호출되고 출력들은 뷰 순서 인덱스 refViewIdx, 플래그 *[[availableDV]]* availableDVRes 및 디스패리티 벡터 mvDisp이다.

[0281] $refResSamples_L$을 refViewIdx와 동일한 ViewIdx를 갖는 뷰 성분의 루마 잔차 샘플들 $ResSamples_L$의 ($PicWidthInSamples_L$) x ($PicHeightInSamples_L$) 어레이라고 하자. $refResSamples_{Cb}$ 및 $refResSamples_{Cr}$이 refViewIdx와 동일한 ViewIdx를 갖는 뷰 성분에 대한 인터 코딩된 코딩 유닛들에 대해, 각각 Cb 및 Cr 잔차 샘플들 $ResSamples_{Cb}$ 및 $ResSamples_{Cr}$의 ($PicWidthInSamples_L$/2)x($PicHeightInSamples_L$/2) 어레이들이라고 하자.

[0282] 플래그 *[[availableDV]]* availableDVRes가 0과 동일한 경우 이 하위절의 전체 디코딩 프로세스는 종료된다.

[0283] 값들 0..(nPSH - 1) 에 걸쳐 진행하는 y와 값들 0..(nPSW - 1) 에 걸쳐 진행하는 x에 대해, 다음 순서의 단계들

이 적용된다.

[0284] ...

[0285] 도 8은 본 개시물의 하나 이상의 기법들에 따른, 멀티-뷰 비디오 데이터를 디코딩하는 비디오 디코더 (30) 의 일 예의 동작 (200) 을 도시하는 흐름도이다. 멀티-뷰 비디오 데이터는 3D-HEVC 비디오 데이터일 수도 있다. 도 8의 예에서, 비디오 디코더 (30) 는 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다 (202). 현재 블록은 현재 뷰 내에 있다. 비디오 디코더 (30) 는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정할 수도 있다 (204). 예를 들면, 가용성 값은 디스패리티 벡터 도출 프로세스가 현재 블록에 이웃하는 블록의 디스패리티 모션 벡터 또는 암시적 디스패리티 벡터로부터 현재 블록에 대한 디스패리티 벡터를 도출할 수 있는 경우 현재 블록에 대한 디스패리티 벡터가 이용가능함을 나타낼 수도 있다. 몇몇 예들에서, 비디오 디코더 (30) 는 가용성 값을 디스패리티 벡터 도출 프로세스를 수행하는 부분으로서 설정할 수도 있다.

[0286] 현재 블록에 대한 디스패리티 벡터가 이용가능함을 가용성 값이 나타내는지 (예컨대, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우 또는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하였음을 가용성 값이 나타내는 경우) 에 무관하게, 비디오 디코더 (30) 는 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다 (206). 몇몇 예들에서, 비디오 인코더 (20) 는 디스패리티 벡터 리파인먼트 프로세스를 영 디스패리티 벡터에 적용함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다.

[0287] 도 8의 예에서, 비디오 디코더 (30) 는 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 디코딩한다 (208). 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 디코딩하는 부분으로서, 비디오 디코더 (30) 는 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 리파인된 디스패리티 벡터를 사용할 수도 있다. 뷰 간 모션 예측, 뷰 간 잔차 예측, 및/또는 백워드 뷰 합성 예측을 사용하는 것에 의해, 비디오 디코더 (30) 는 현재 블록에 대응하는 샘플 블록을 복원가능할 수도 있다.

[0288] 도 9는 본 개시물의 하나 이상의 기법들에 따른, 멀티-뷰 비디오 데이터를 인코딩하는 비디오 인코더 (20) 의 일 예의 동작 (250) 을 도시하는 흐름도이다. 멀티-뷰 비디오 데이터는 3D-HEVC 비디오 데이터일 수도 있다. 도 9의 예에서, 비디오 인코더 (20) 는 멀티-뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다 (252). 현재 블록은 현재 뷰 내에 있다. 비디오 인코더 (20) 는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출할 수 없는 경우 현재 블록에 대한 디스패리티 벡터가 이용불가능함을 가용성 값이 나타내도록 가용성 값을 설정할 수도 있다 (254). 예를 들면, 가용성 값은 디스패리티 벡터 도출 프로세스가 현재 블록에 이웃하는 블록의 디스패리티 모션 벡터 또는 암시적 디스패리티 벡터로부터 현재 블록에 대한 디스패리티 벡터를 도출할 수 있는 경우 현재 블록에 대한 디스패리티 벡터가 이용가능함을 나타낼 수도 있다. 몇몇 예들에서, 비디오 인코더 (20) 는 가용성 값을 디스패리티 벡터 도출 프로세스를 수행하는 부분으로서 설정할 수도 있다.

[0289] 현재 블록에 대한 디스패리티 벡터가 이용가능함을 가용성 값이 나타내는지 (예컨대, 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하지 않았음을 가용성 값이 나타내는 경우 또는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 디스패리티 벡터를 도출하였음을 가용성 값이 나타내는 경우) 에 무관하게, 비디오 인코더 (20) 는 참조 뷰의 깊이 뷰 성분에 액세스하는 디스패리티 벡터 리파인먼트 프로세스를 수행함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다 (256). 몇몇 예들에서, 비디오 인코더 (20) 는 디스패리티 벡터 리파인먼트 프로세스를 영 디스패리티 벡터에 적용함으로써 현재 블록에 대한 리파인된 디스패리티 벡터를 생성할 수도 있다. 참조 뷰는 현재 뷰와 상이하다.

[0290] 도 9의 예에서, 비디오 인코더 (20) 는 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 인코딩한다 (258). 다르게 말하면, 비디오 인코더 (20) 는 현재 블록에 대한 리파인된 디스패리티 벡터를 사용하여 현재 블록의 인코딩된 표현을 생성할 수도 있다. 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 인코딩하는 부분으로서, 비디오 인코더 (20) 는 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 리파인된 디스패리티 벡터를 사용할 수도 있다.

[0291] 도 10은 본 개시물의 하나 이상의 부가적인 기법들에 따른, 멀티-뷰 비디오 데이터를 디코딩하는 비디오 디코더 (30) 의 일 예의 동작 (300) 을 도시하는 흐름도이다. 도 10의 예에서, 비디오 디코더 (30) 는 현재 블록에 대한 디스패리티 벡터를 도출하기 위해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다 (302). 예를 들면, 비디오 디코더 (30) 는 현재 블록에 대한 디스패리티 벡터를 도출하기 위해 NBDV 프로세스를 수행할 수도 있다. 그 뒤에, 비디오 디코더 (30) 는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 디스패리티 벡터를 도출하였는지의 여부를 결정할 수도 있다 (304).

[0292] 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 디스패리티 벡터를 도출하지 않았다는 결정에 응답하여 (304의 "아니오"), 비디오 디코더 (30) 는 현재 블록에 대한 디스패리티 벡터가 오프셋 및 영 디스패리티 벡터의 합과 동일하도록 현재 블록에 대한 디스패리티 벡터를 결정할 수도 있다 (306). 몇몇 예들에서, 비디오 디코더 (30) 는 오프셋을 현재 블록의 디스패리티 벡터의 수평 성분에만 더할 수도 있다. 더욱이, 몇몇 예들에서, 비디오 디코더 (30) 는 카메라 파라미터들 및 디폴트 깊이 화소 값 (예컨대, 128) 으로 오프셋을 컴퓨팅할 수도 있다. 몇몇 예들에서, 비디오 인코더 (20) 는 현재 블록을 포함하는 슬라이스의 슬라이스 헤더에서 오프셋을 시그널링한다.

[0293] 306에서 현재 블록에 대한 디스패리티 벡터를 결정한 후 또는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 벡터를 도출하였다는 결정에 응답하여 (304의 "예"), 비디오 디코더 (30) 는 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 디코딩할 수도 있다 (308). 현재 블록에 대한 리파인된 디스패리티 벡터에 기초하여 현재 블록을 디코딩하는 부분으로서, 비디오 디코더 (30) 는 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 리파인된 디스패리티 벡터를 사용할 수도 있다. 뷰 간 모션 예측, 뷰 간 잔차 예측, 및/또는 백워드 뷰 합성 예측을 사용하는 것에 의해, 비디오 디코더 (30) 는 현재 블록에 대응하는 샘플 블록을 복원가능할 수도 있다.

[0294] 도 11은 본 개시물의 하나 이상의 부가적인 기법들에 따른, 멀티-뷰 비디오 데이터를 인코딩하는 비디오 인코더 (20) 의 일 예의 동작 (350) 을 도시하는 흐름도이다. 도 11의 예에서, 비디오 인코더 (20) 는 현재 블록에 대한 디스패리티 벡터를 도출하기 위해 디스패리티 벡터 도출 프로세스를 수행할 수도 있다 (352). 예를 들면, 비디오 인코더 (20) 는 현재 블록에 대한 디스패리티 벡터를 도출하기 위해 NBDV 프로세스를 수행할 수도 있다. 그 뒤에, 비디오 인코더 (20) 는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 디스패리티 벡터를 도출하였는지의 여부를 결정할 수도 있다 (354).

[0295] 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 디스패리티 벡터를 도출하지 않았다는 결정에 응답하여 (354의 "아니오"), 비디오 인코더 (20) 는 현재 블록에 대한 디스패리티 벡터가 오프셋 및 영 디스패리티 벡터의 합과 동일하도록 현재 블록에 대한 디스패리티 벡터를 결정할 수도 있다 (356). 몇몇 예들에서, 비디오 인코더 (20) 는 오프셋을 현재 블록의 디스패리티 벡터의 수평 성분에만 더할 수도 있다. 더욱이, 몇몇 예들에서, 비디오 인코더 (20) 는 카메라 파라미터들 및 디폴트 깊이 화소 값 (예컨대, 128) 으로 오프셋을 컴퓨팅할 수도 있다. 몇몇 예들에서, 비디오 인코더 (20) 는 현재 블록을 포함하는 슬라이스의 슬라이스 헤더에서 오프셋을 시그널링한다.

[0296] 356에서 현재 블록에 대한 디스패리티 벡터를 결정한 후 또는 디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 벡터를 도출하였다는 결정에 응답하여 (354의 "예"), 비디오 인코더 (20) 는 현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 인코딩할 수도 있다 (358).

[0297] 다음은 본 개시물의 기법들에 따른 부가적인 실시예들이다.

[0298] 실시예 1. 멀티뷰 비디오 데이터를 디코딩하는 방법으로서, 멀티뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계; 및 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하기 위해 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계를 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0299] 실시예 2. 실시예 1에 있어서, 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계는, 영 디스패리티 벡터를 사용하여 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계를 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0300] 실시예 3. 실시예 2에 있어서, 디스패리티 벡터 도출이 이용가능한 디스패리티 벡터를 생성하지 않는 경우 디스패리티 벡터 리파인먼트 프로세스에 대해 영 디스패리티 벡터를 생성하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0301] 실시예 4. 실시예 2에 있어서, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 디스패리티 벡터가 이용가능함을 나타내도록 스테이터스를 설정하고 디스패리티 벡터의 값을 영으로 설정하는 단계, 및 영의 값을 갖는 디스패리티 벡터를 사용하여 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0302] 실시예 5. 실시예 1 내지 실시예 4 중 임의의 실시예, 또는 그 조합들에 있어서, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하였는지의 여부를 나타내기 위해 변수를 유지하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0303] 실시예 6. 실시예 5에 있어서, 상기 변수를 하나 이상의 코딩 도구들에 의한 사용을 위해 제공하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0304] 실시예 7. 실시예 6에 있어서, 상기 변수의 값에 기초하여 코딩 도구들 중 하나 이상을 인에이블하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0305] 실시예 8. 실시예 1에 있어서, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 수정된 디스패리티 벡터를 생성하기 위해 오프셋을 영 디스패리티 벡터에 더하는 단계를 더 포함하는, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0306] 실시예 9. 실시예 8에 있어서, 하나 이상의 카메라 파라미터들 및 디폴트 깊이 화소 값에 기초하여 오프셋을 결정하는 단계를 더 포함하는, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0307] 실시예 10. 실시예 8에 있어서, 코딩된 비디오 비트스트림에서 수신된 시그널링된 값에 기초하여 오프셋을 결정하는 단계를 더 포함하는, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0308] 실시예 11. 실시예 10에 있어서, 상기 값은 슬라이스 헤더에서 시그널링되는, 멀티뷰 비디오 데이터 디코딩 방법.

[0309] 실시예 12. 실시예 1 내지 실시예 11 중 임의의 실시예, 또는 그 조합들에 있어서, 디스패리티 벡터 도출 프로세스는 이웃 블록 디스패리티 벡터 (NBDV) 도출 프로세스인, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0310] 실시예 13. 실시예 1 내지 실시예 12 중 임의의 실시예, 또는 그 조합들에 있어서, 디스패리티 벡터 리파인먼트 프로세스는 이웃 블록 디스패리티 벡터 리파인먼트 (NBDV-R) 프로세스인, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0311] 실시예 14. 실시예 1 내지 실시예 13 중 임의의 실시예, 또는 그 조합들에 있어서, 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 리파인된 디스패리티 벡터를 사용하는 단계를 더 포함하는, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0312] 실시예 15. 실시예 1 내지 실시예 14 중 임의의 실시예, 또는 그 조합들에 있어서, 멀티뷰 비디오 데이터는 3D-HEVC 비디오 데이터인, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0313] 실시예 16. 실시예 1 내지 실시예 15 중 임의의 실시예, 또는 그 조합들에 있어서, 리파인된 디스패리티 벡터에 기초하여 비디오 데이터를 디코딩하는 단계를 더 포함하는, 오프셋을 멀티뷰 비디오 데이터를 디코딩하는 방법.

[0314] 실시예 17. 멀티뷰 비디오 데이터를 인코딩하는 방법으로서, 멀티뷰 비디오 데이터의 현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행하는 단계; 및 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 현재 블록에 대한 리파인된 디스패리티 벡터를 생성하기 위해 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계를 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0315] 실시예 18. 실시예 17에 있어서, 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계는, 영 디스패리티 벡터를 사용하여 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계를 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0316] 실시예 19. 실시예 18에 있어서, 디스패리티 벡터 도출이 이용가능한 디스패리티 벡터를 생성하지 않는 경우 디스패리티 벡터 리파인먼트 프로세스에 대해 영 디스패리티 벡터를 생성하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0317] 실시예 20. 실시예 18에 있어서, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 디스패리티 벡터가 이용가능함을 나타내도록 스테이터스를 설정하고 디스패리티 벡터의 값을 영으로 설정하는 단계, 및 영의 값을 갖는 디스패리티 벡터를 사용하여 디스패리티 벡터 리파인먼트 프로세스를 수행하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0318] 실시예 21. 실시예 17 내지 실시예 20 중 임의의 실시예, 또는 그 조합들에 있어서, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하였는지의 여부를 나타내기 위해 변수를 유지하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0319] 실시예 22. 실시예 21에 있어서, 상기 변수를 하나 이상의 코딩 도구들에 의한 사용을 위해 제공하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0320] 실시예 23. 실시예 22에 있어서, 상기 변수의 값에 기초하여 코딩 도구들 중 하나 이상을 인에이블하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0321] 실시예 24. 실시예 17에 있어서, 디스패리티 벡터 도출 프로세스가 이용가능한 디스패리티 벡터를 생성하지 않는 경우, 수정된 디스패리티 벡터를 생성하기 위해 오프셋을 영 디스패리티 벡터에 더하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0322] 실시예 25. 실시예 24에 있어서, 하나 이상의 카메라 파라미터들 및 디폴트 깊이 화소 값에 기초하여 오프셋을 결정하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0323] 실시예 26. 실시예 24에 있어서, 코딩된 비디오 비트스트림에서 수신된 시그널링된 값에 기초하여 오프셋을 결정하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법 .

[0324] 실시예 27. 실시예 26에 있어서, 상기 값은 슬라이스 헤더에서 시그널링되는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0325] 실시예 28. 실시예 17 내지 실시예 27 중 임의의 실시예, 또는 그 조합들에 있어서, 디스패리티 벡터 도출 프로세스는 이웃 블록 디스패리티 벡터 (NBDV) 도출 프로세스인, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0326] 실시예 29. 실시예 17 내지 실시예 28 중 임의의 실시예, 또는 그 조합들에 있어서, 디스패리티 벡터 리파인먼트 프로세스는 이웃 블록 디스패리티 벡터 리파인먼트 (NBDV-R) 프로세스인, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0327] 실시예 30. 실시예 17 내지 실시예 29 중 임의의 실시예, 또는 그 조합들에 있어서, 뷰 간 모션 예측, 뷰 간 잔차 예측, 또는 백워드 뷰 합성 예측 중 적어도 하나에 대해 리파인된 디스패리티 벡터를 사용하는 단계를 더 포함하는, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0328] 실시예 31. 실시예 17 내지 실시예 30 중 임의의 실시예, 또는 그 조합들에 있어서, 멀티뷰 비디오 데이터는 3D-HEVC 비디오 데이터인, 멀티뷰 비디오 데이터를 인코딩하는 방법.

[0329] 실시예 32. 실시예 1 내지 실시예 16 중 임의의 실시예 또는 그 조합들의 방법을 수행하도록 구성된 비디오 디코딩 장치.

[0330] 실시예 33. 실시예 17 내지 실시예 31 중 임의의 실시예 또는 그 조합들의 방법을 수행하도록 구성된 비디오 인코딩 장치.

[0331] 실시예 34. 실시예 1 내지 실시예 16 중 임의의 실시예 또는 그 조합들의 방법을 수행하는 수단을 포함하는 비디오 디코딩 장치.

[0332] 실시예 35. 실시예 17 내지 실시예 31 중 임의의 실시예 또는 그 조합들의 방법을 수행하는 수단을 포함하는 비디오 인코딩 장치.

[0333] 실시예 36. 하나 이상의 프로세서들로 하여금, 실시예 1 내지 실시예 31 중 임의의 실시예 또는 그 조합들의 방법을 수행하게 하는 명령들을 포함하는 컴퓨터 판독가능 매체.

[0334] 실시예 37. 본원에 개시된 기법들 중 임의의 것에 따른 비디오 데이터를 인코딩하는 방법.

[0335] 실시예 38. 본원에 개시된 기법들 중 임의의 것에 따른 비디오 데이터를 디코딩하는 방법.

[0336] 실시예 39. 본원에 개시된 기법들 중 임의의 것을 수행하도록 구성된 디바이스.

[0337] 하나 이상의 예들에서, 설명된 기능들은 하드웨어, 소프트웨어, 펌웨어, 또는 이들의 임의의 조합에서 구현될 수도 있다. 소프트웨어에서 구현된다면, 그 기능들은 하나 이상의 명령들 또는 코드로서 컴퓨터 판독가능 매체 상에 저장되거나 또는 그것을 통해 송신될 수도 있고 하드웨어 기반 프로세싱 유닛에 의해 실행될 수도 있다. 컴퓨터 판독가능 매체들은, 데이터 저장 매체들과 같은 유형의 (tangible) 매체에 대응하는 컴퓨터 판독가능 저장 매체들, 또는 예컨대 통신 프로토콜에 따라 한 장소에서 다른 장소로 컴퓨터 프로그램의 전달을 용이하게 하는 임의의 매체를 포함하는 통신 매체들을 포함할 수도 있다. 이런 방식으로, 컴퓨터 판독가능 매체들은 일반적으로 (1) 비일시적 (non-transitory) 인 유형의 (tangible) 컴퓨터 판독가능 저장 매체들 또는 (2) 신호 또는 반송파와 같은 통신 매체에 해당할 수도 있다. 데이터 저장 매체들은 본 개시물에서 설명된 기법들의 구현을 위한 명령들, 코드 및/또는 데이터 구조들을 취출하기 위해 하나 이상의 컴퓨터들 또는 하나 이상의 프로세서들에 의해 액세스될 수 있는 임의의 이용가능 매체들일 수도 있다. 컴퓨터 프로그램 제품은 컴퓨터 판독가능 매체를 포함할 수도 있다.

[0338] 비제한적인 예로, 이러한 컴퓨터 판독가능 저장 매체들은 RAM, ROM, EEPROM, CD-ROM 또는 다른 광 디스크 스토리지, 자기 디스크 스토리지, 또는 다른 자기 저장 디바이스들, 플래시 메모리, 또는 소망의 프로그램 코드를 명령들 또는 데이터 구조들의 형태로 저장하는데 사용될 수 있고 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 또한, 임의의 접속이 컴퓨터 판독가능 매체로 적절히 칭해진다. 예를 들어, 명령들이 웹사이트, 서버, 또는 다른 원격 소스로부터 동축 케이블, 광섬유 케이블, 연선 (twisted pair), 디지털 가입자 회선 (DSL), 또는 무선 기술들 이를테면 적외선, 라디오, 및/또는 마이크로파를 이용하여 송신된다면, 동축 케이블, 광섬유 케이블, 연선, DSL, 또는 적외선, 라디오, 및 마이크로파와 같은 무선 기술은 매체의 정의에 포함된다. 그러나, 컴퓨터 판독가능 저장 매체들 및 데이터 저장 매체들은 커넥션들, 반송파들, 신호들, 또는 다른 일시적인 매체들을 포함하지 않지만, 대신 비일시적, 유형의 저장 매체들을 지향하고 있음이 이해되어야 한다. 디스크 (disk) 및 디스크 (disc) 는 본원에서 사용되는 바와 같이, 콤팩트 디스크 (compact disc; CD), 레이저 디스크, 광 디스크, 디지털 다기능 디스크 (DVD), 플로피 디스크 (floppy disk) 및 블루레이 디스크를 포함하는데, 디스크 (disk) 들은 보통 데이터를 자기적으로 재생하는 한편, 디스크 (disc) 들은 레이저들로 데이터를 광학적으로 재생한다. 상기한 것들의 조합들은 또한 컴퓨터 판독가능 매체들의 범위 내에 포함되어야 한다.

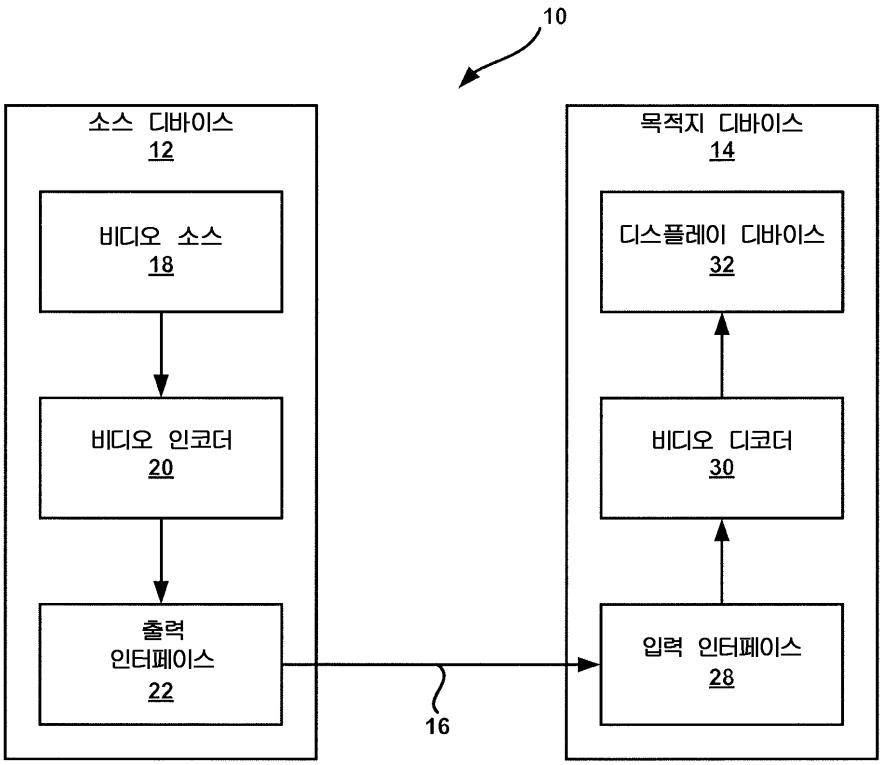
[0339] 명령들은 하나 이상의 프로세서들, 이를테면 하나 이상의 디지털 신호 프로세서들 (DSP들), 범용 마이크로프로세서들, 주문형 집적회로들 (ASIC들), 필드 프로그램가능 로직 어레이들 (FPGA들), 또는 다른 동등한 집적 또는 개별 로직 회로에 의해 실행될 수도 있다. 따라서, 본원에서 사용되는 바와 같은 용어 "프로세서"는 앞서의 구조 또는 본원에서 설명된 기법들의 구현에 적합한 임의의 다른 구조 중 임의의 것을 말할 수도 있다. 덧붙여서, 일부 양태들에서, 본원에서 설명된 기능성은 인코딩 및 디코딩을 위해 구성되는, 또는 결합형 코덱 (codec) 에 통합되는 전용 하드웨어 및/또는 소프트웨어 모듈들 내에 제공될 수도 있다. 또한, 기법들은 하나 이상의 회로들 또는 로직 엘리먼트들 내에 완전히 구현될 수 있다.

[0340] 본 개시물의 기법들은 무선 핸드셋, 집적회로 (IC) 또는 IC들의 세트 (예컨대, 칩 세트) 를 포함한 매우 다양한 디바이스들 또는 장치들로 구현될 수도 있다. 다양한 컴포넌트들, 모듈들, 또는 유닛들은 개시된 기법들을 수행하도록 구성된 디바이스들의 기능적 양태들을 강조하기 위해 본 개시물에서 설명되지만, 상이한 하드웨어 유닛들에 의한 실현을 반드시 요구하지는 않는다. 대신에, 위에서 설명된 바와 같이, 다양한 유닛들은 코덱 하드웨어 유닛에 결합되거나 또는 적합한 소프트웨어 및/또는 펌웨어와 함께, 위에서 설명된 바와 같은 하나 이상의 프로세서들을 포함하는, 상호운용적 하드웨어 유닛들의 컬렉션에 의해 제공될 수도 있다.

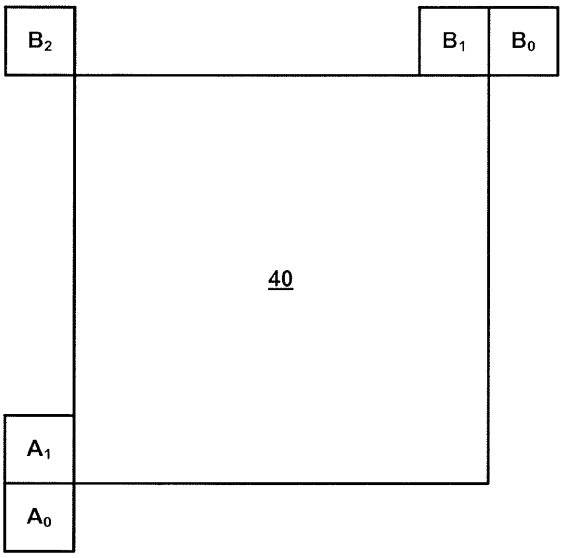
[0341] 다양한 예들이 설명되어 있다. 이들 및 다른 예들은 다음의 청구항들의 범위 내에 있다.

도면

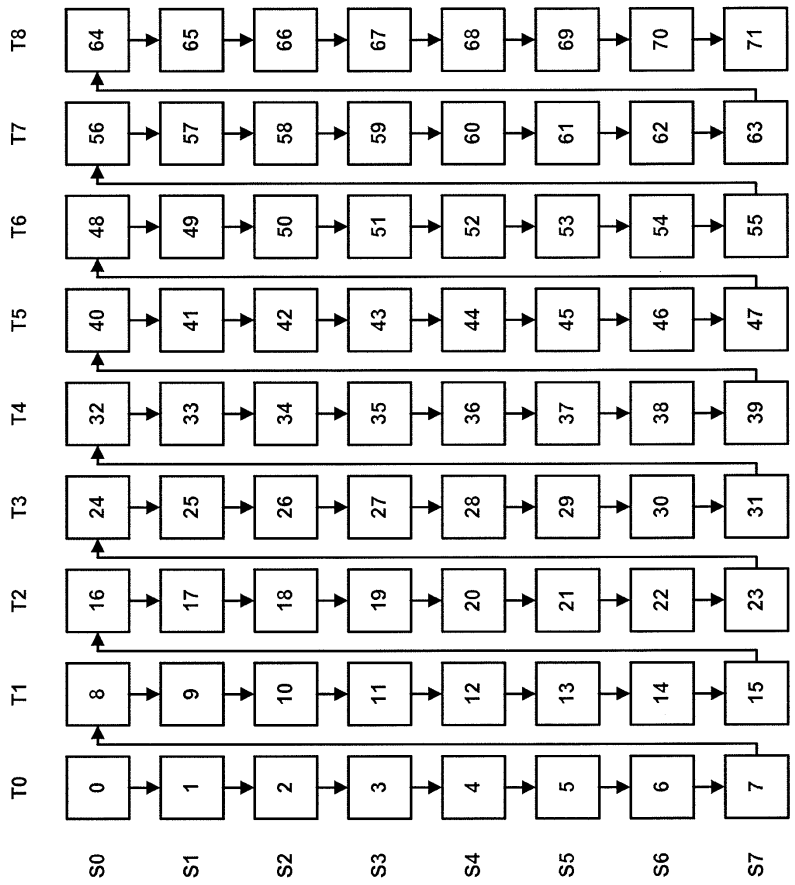
도면1


10
소스 디바이스
12
비디오 소스
18
비디오 인코더
20
출력
인터페이스
22
16
목적지 디바이스
14
디스플레이 디바이스
32
비디오 디코더
30
입력 인터페이스
28

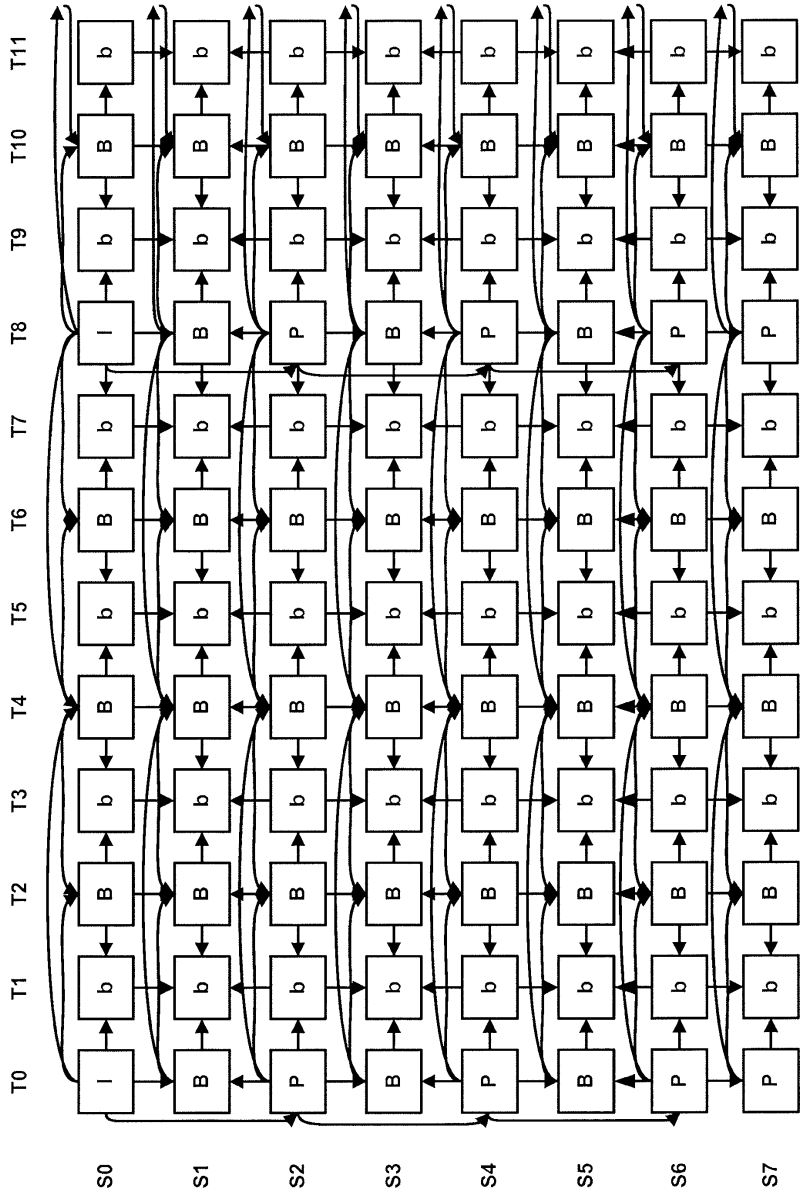
도면2


B2
B1
B0
40
A1
A0

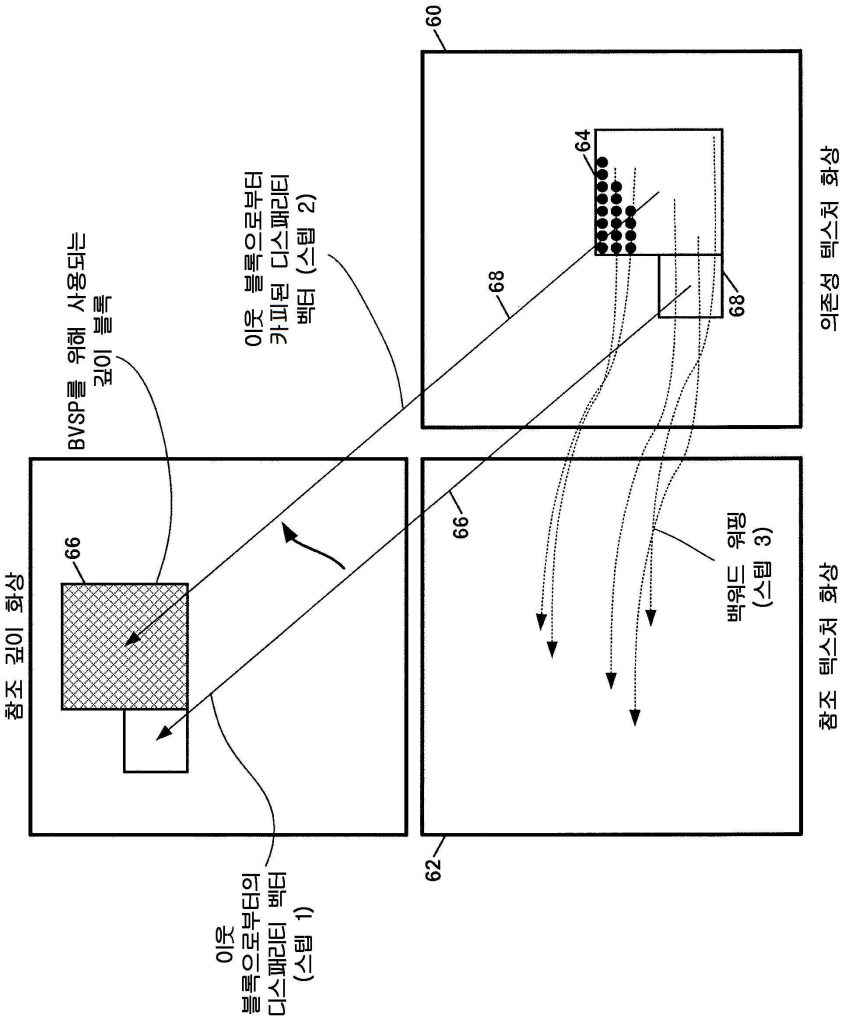
도면3


T8
T7
T6
T5
T4
T3
T2
T1
T0
64 65 66 67 68 69 70 71
56 57 58 59 60 61 62 63
48 49 50 51 52 53 54 55
40 41 42 43 44 45 46 47
32 33 34 35 36 37 38 39
24 25 26 27 28 29 30 31
16 17 18 19 20 21 22 23
8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7
S0 S1 S2 S3 S4 S5 S6 S7

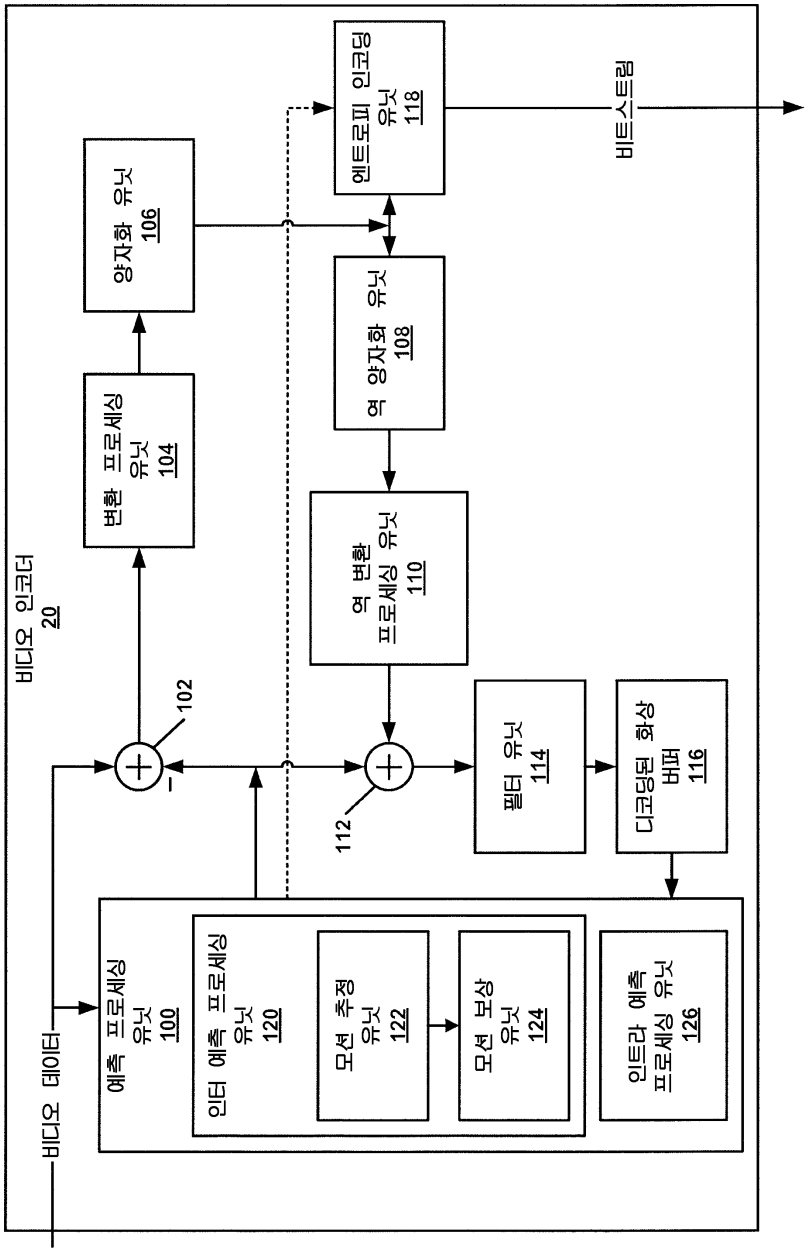
도면4



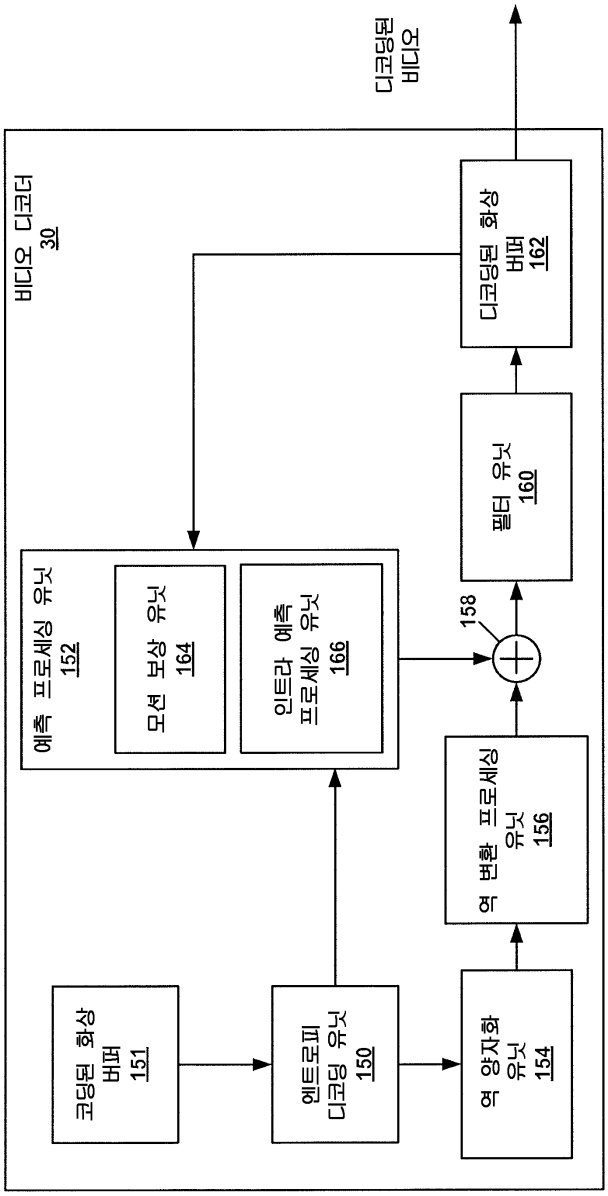
도면5


60
64
68
68
의존성 텍스처 화상
이웃 블록으로부터
카피된 디스패리티
벡터 (스텝 2)
BVSP를 위해 사용되는
깊이 블록
66
66
백워드 워핑
(스텝 3)
참조 깊이 화상
참조 텍스처 화상
62
이웃
블록으로부터의
디스패리티 벡터
(스텝 1)

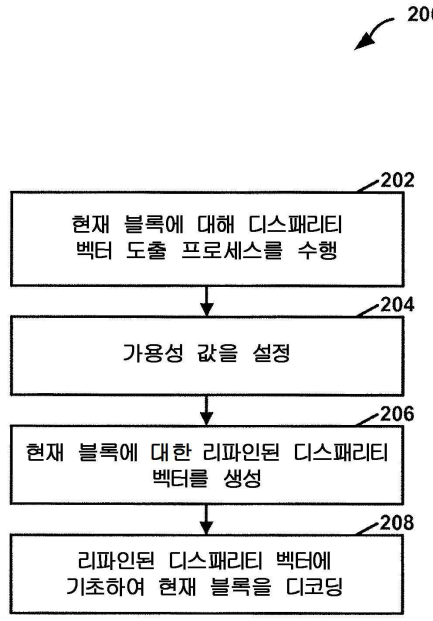
도면6



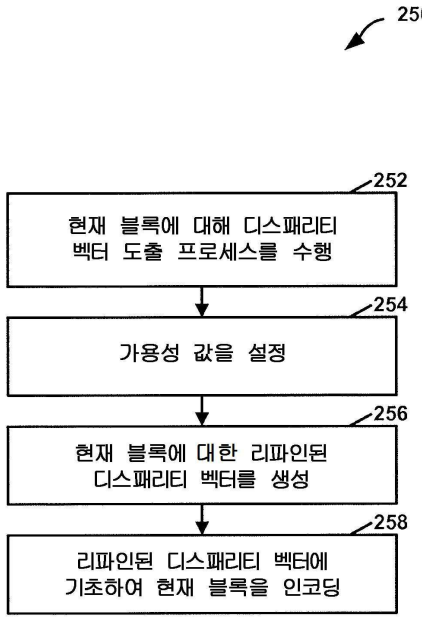
도면7


디코딩된
비디오
비디오 디코더
30
디코딩된 화상
버퍼
162
필터 유닛
160
158
예측 프로세싱 유닛
152
모션 보상 유닛
164
인트라 예측
프로세싱 유닛
166
역 변환 프로세싱
유닛
156
역 양자화
유닛
154
엔트로피
디코딩 유닛
150
코딩된 화상
버퍼
151

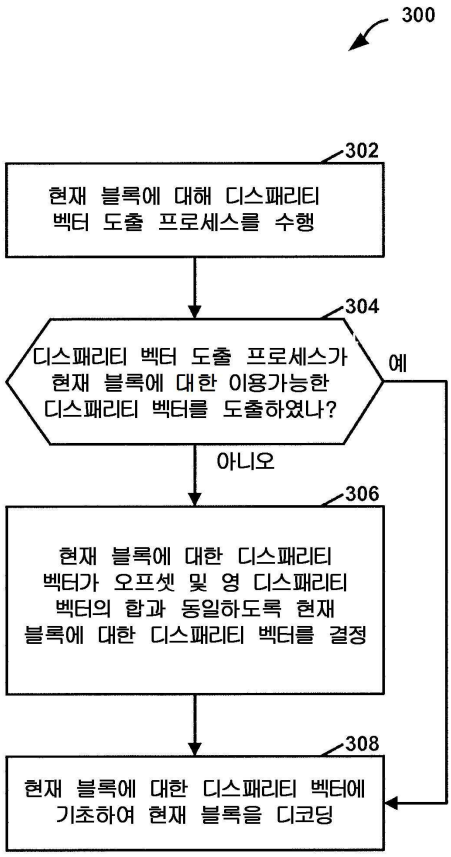
도면8


200
202
현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행
204
가용성 값을 설정
206
현재 블록에 대한 리파인된 디스패리티 벡터를 생성
208
리파인된 디스패리티 벡터에 기초하여 현재 블록을 디코딩

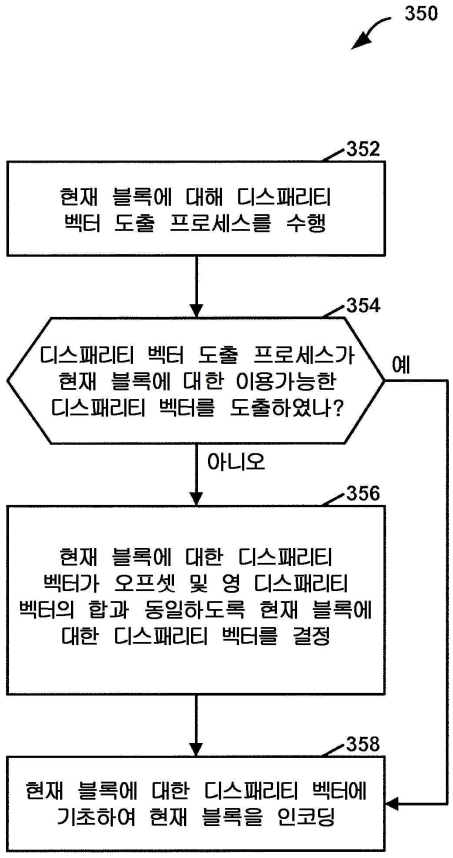
도면9


250
252
현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행
254
가용성 값을 설정
256
현재 블록에 대한 리파인된 디스패리티 벡터를 생성
258
리파인된 디스패리티 벡터에 기초하여 현재 블록을 인코딩

도면10


300
302
현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행
304
디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 디스패리티 벡터를 도출하였나?
예
아니오
306
현재 블록에 대한 디스패리티 벡터가 오프셋 및 영 디스패리티 벡터의 합과 동일하도록 현재 블록에 대한 디스패리티 벡터를 결정
308
현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 디코딩

도면11


350
352
현재 블록에 대해 디스패리티 벡터 도출 프로세스를 수행
354
디스패리티 벡터 도출 프로세스가 현재 블록에 대한 이용가능한 디스패리티 벡터를 도출하였나?
예
아니오
356
현재 블록에 대한 디스패리티 벡터가 오프셋 및 영 디스패리티 벡터의 합과 동일하도록 현재 블록에 대한 디스패리티 벡터를 결정
358
현재 블록에 대한 디스패리티 벡터에 기초하여 현재 블록을 인코딩