



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0043164

(43) 공개일자 2015년04월22일

(51) 국제특허분류(Int. Cl.)

H04N 19/00 (2014.01) H04N 13/00 (2006.01)

(21) 출원번호 10-2013-0122265

(22) 출원일자 2013년10월14일

심사청구일자 없음

(71) 출원인

한국전자통신연구원

대전광역시 유성구 가정로 218 (가정동)

경희대학교 산학협력단

경기도 용인시 기흥구 덕영대로 1732 (서천동, 경희대학교 국제캠퍼스내)

(72) 발명자

방건

대전광역시 유성구 송강로 42번길 61, 310-1003

송강청솔아파트

이광순

대전광역시 유성구 진잠로149번길 30 한승미메이
드아파트 208/303

(뒷면에 계속)

(74) 대리인

에스앤아이퍼특허법인

전체 청구항 수 : 총 8 항

(54) 발명의 명칭 2D 비디오 기반 3D 비디오 코딩을 위한 병합 움직임 후보 리스트를 구성하는 방법 및 장치

(57) 요약

1. 청구범위에 기재된 발명이 속한 기술분야

본 발명은 비디오의 부호화/복호화 방법 및 장치에 관한 것임.

2. 발명이 해결하려고 하는 기술적 과제

본 발명에서는 비디오 코덱에서 구현 복잡도 및 계산 복잡도를 감소시키는 방법을 제안한다.

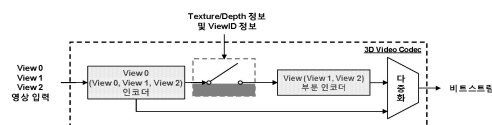
3. 발명의 해결방법의 요지

본 발명은 부호화/복호화 모듈의 재사용성을 증가시키는 방법을 포함함.

4. 발명의 중요한 용도

본 발명은 비디오 부호화에서 부호화 및 복호화의 구현 복잡도 및 계산 복잡도를 감소시킬 수 있다.

대표도 - 도10



(72) 발명자

허남호

대전광역시 유성구 은구비남로 34, 열매마을아파트
801동 1001호

박광훈

경기 성남시 분당구 예원로 7, B동 302호 (분당동,
동아빌라)

허영수

경기 수원시 영통구 봉영로1492번길 20, B01호 (영
통동)

김경용

경기 수원시 영통구 덕영대로 1725, 402호 (영
통동)

이윤진

경기 용인시 기흥구 흥덕2로 80, 309동 1502호 (영
덕동, 휴먼시아흥덕마을3단지아파트)

명세서

청구범위

청구항 1

독립적인 시점(Independent View; 시점 0(View 0))에 대한 일반 영상의 부호화/복호화 모듈을 재사용하여 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들을 부호화/복호화하는 방법 및 장치.

청구항 2

제 1항에 있어서, 독립적인 시점(Independent View; 시점 0(View 0))에 대한 일반 영상의 부호화에 사용되는 부호화/복호화 모듈을 그대로 사용하여, 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들을 부호화/복호화한 후, 추가적인 부호화/복호화 모듈을 이용하여 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들을 재부호화/재복호화하거나 혹은 재설정(혹은 변경)하는 방법 및 장치

청구항 3

제 2항에 있어서, 독립적인 시점(Independent View; 시점 0(View 0))에 대한 일반 영상의 부호화에 사용되는 “기본 병합 움직임 리스트 구성” 모듈을 그대로 사용하여, 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들에 대한 “기본 병합 움직임 후보 리스트”를 구성한 후, 추가적으로 “추가 병합 움직임 리스트 구성” 모듈을 이용하여 “기본 병합 움직임 후보 리스트”를 재설정(혹은 변경)하여 “확장 병합 움직임 후보 리스트”를 구성하는 방법 및 장치

청구항 4

제 2항과 3항에 있어서, 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)를 통해, 추가적인 부호화/복호화 모듈과 “추가 병합 움직임 리스트 구성” 모듈의 수행여부를 결정하는 방법 및 장치

청구항 5

제 3항에 있어서, “기본 병합 움직임 후보 리스트”를 이용하여 “확장 병합 움직임 후보 리스트”를 구성하는데 있어서, 추가적인 움직임 정보를 “기본 병합 움직임 후보 리스트”의 첫번째 항목(혹은 임의의 위치에 해당하는 항목)에 삽입하는 방법 및 장치

청구항 6

제 5항에 있어서, 추가적인 움직임 정보를 “기본 병합 움직임 후보 리스트”의 첫번째 항목(혹은 임의의 위치에 해당하는 항목)에 삽입하기 전에, 추가적인 움직임 정보와 첫번째 항목의 움직임 정보를 서로 비교하여 삽입 여부를 결정하는 방법 및 장치.

청구항 7

제 5항에 있어서, 추가적인 움직임 정보들을 “기본 병합 움직임 후보 리스트”의 임의의 위치에 각각 삽입하고, 움직임 후보들을 재설정 및 재배치하는 방법 및 장치.

청구항 8

제 5항에 있어서, 시점 간 병합 후보와 시차 병합 후보를 리스트에 삽입하기 전에 리스트 내의 일부 후보들만을 이용하여 비교하여 삽입 여부를 결정하는 방법 및 장치

발명의 설명

기술 분야

[0001] 3차원 비디오는 3차원 입체 디스플레이 장치에 대한 것이다.

배경 기술

[0002] 3차원 비디오는 3차원 입체 디스플레이 장치를 통해 사용자에게 실세계에서 보고 느끼는 것과 같은 입체감을 생생하게 제공한다. 이와 관련된 연구로써 ISO/IEC의 MPEG(Moving Picture Experts Group)과 ITU-T의 VCEG(Video Coding Experts Group)의 공동 표준화 그룹인 JCT-3V(The Joint Collaborative Team on 3D Video Coding Extension Development)에서 3차원 비디오 표준이 진행 중이다. 3차원 비디오 표준은 실제 영상과 그것의 깊이 정보 맵을 이용하여 스테레오스코픽 영상뿐만 아니라 오토스테레오스코픽 영상의 재생등을 지원할 수 있는 진보된 데이터 형식과 그에 관련된 기술에 대한 표준을 포함하고 있다.

[0003] 3차원 비디오 표준에서 고려하고 있는 기본적인 3차원 비디오 시스템은 도 1과 같다. 송신 측에서는 스테레오 카메라, 깊이정보 카메라, 다시점 카메라, 2차원 영상을 3차원 영상으로 변환 등을 이용하여 $N(N \geq 2)$ 시점의 영상 콘텐츠를 획득한다. 획득된 영상 콘텐츠에는 N 시점의 비디오 정보와 그것의 깊이정보 맵(Depth-map) 정보 그리고 카메라 관련 부가정보 등이 포함될 수 있다. N 시점의 영상 콘텐츠는 다시점 비디오 부호화 방법을 사용하여 압축되며, 압축된 비트스트림은 네트워크를 통해 단말로 전송된다. 수신 측에서는 전송받은 비트스트림을 다시점 비디오 복호화 방법을 사용하여 복호화하여 N 시점의 영상을 복원한다. 복원된 N 시점의 영상은 깊이정보 맵 기반 렌더링(DIBR; Depth-Image-Based Rendering) 과정에 의해 N 시점 이상의 가상시점 영상들을 생성한다. 생성된 N 시점 이상의 가상시점 영상들은 다양한 입체 디스플레이 장치에 맞게 재생되어 사용자에게 입체감이 있는 영상을 제공하게 된다.

[0004] 가상시점 영상을 생성하는데 사용되는 깊이정보 맵은 실세계에서 카메라와 실제 사물(object) 간의 거리(실사 영상과 동일한 해상도로 각 화소에 해당하는 깊이정보)를 일정한 비트수로 표현한 것이다. 깊이정보 맵의 예로써, 도 2는 국제 표준화 기구인 MPEG의 3차원 비디오 부호화 표준에서 사용 중인 “balloons” 영상(도 2 (a))과 그것의 깊이정보 맵(도 2 (b))을 보여주고 있다. 실제 도 2의 깊이정보 맵은 화면에 보이는 깊이 정보를 화소당 8비트로 표현한 것이다.

[0005] 실제 영상과 그것의 깊이정보 맵을 부호화하는 방법의 일례로, H.264/AVC(MPEG-4 Part 10 Advanced Video Coding)를 이용하여 부호화를 수행할 수 있으며, 또 다른 방법의 일 예로 MPEG(Moving Picture Experts Group)과 VCEG(Video Coding Experts Group)에서 공동으로 표준화를 진행한 HEVC(High Efficiency Video Coding) 국제 동영상 표준을 사용할 수 있다.

[0006] 실제 영상과 그것의 깊이정보맵은 카메라 하나뿐만아니라 여러 개의 카메라에서 획득한 영상일 수 있다. 여러 개의 카메라에서 획득한 영상은 독립적으로 부호화될 수 있으며, 일반적인 2차원 비디오 부호화 코덱을 사용할 수 있다. 또한, 여러 개의 카메라에서 획득한 영상은 시점 간의 상관성이 존재하므로, 부호화 효율을 높이기 위하여 서로 다른 시점간 예측을 사용하여 영상들을 부호화할 수 있다. 일 실시 예로, 도 3은 카메라 3대에서 획득한 영상들에 대한 시점간 예측 구조의 일례를 나타낸다.

[0007] 도 3에서, 시점 1(View 1)은 시점 0(View 0)을 기준으로 왼쪽에 위치한 카메라에서 획득한영상이고, 시점 2(View 2)는 시점 0(View 0)을 기준으로 오른쪽에 위치한 카메라에서 획득한 영상이다. 또한, 시점 1(View 1)과 시점 2(View 2)는 시점 0(View 0)을 참조 영상으로 사용하여 시점 간 예측을 수행하며, 부호화 순서는 시점 1(View 1)과 시점 2(View 2)보다 시점 0(View 0)이 먼저 부호화되어야 한다. 이때, 시점 0(View 0)은 다른 시점과 상관없이 독립적으로 부호화될 수 있으므로 독립적인 시점(Independent View)이라고 한다. 반면, 시점 1(View 1)과 시점 2(View 2)는 시점 0(View 0)을 참조 영상으로 사용하므로 의존적인 시점(Dependent View)라고 한다. 독립적인 시점 영상은 일반적인 2차원 비디오 코덱을 사용하여 부호화 될 수 있다. 반면, 의존적인 시점 영상은 시점간 예측을 수행하여야 하므로, 시점간 예측 과정이 포함된 3차원 비디오 코덱을 사용하여 부호화할 수 있다.

[0008] 또한, 시점 1(View 1)과 시점 2(View 2)의 부호화 효율을 증가시키기 위하여 깊이정보 맵을 이용하여 부호화 할 수 있다. 일례로, 실제 영상과 그것의 깊이정보 맵을 부호화할 때, 서로 독립적으로 부호화/복호화할 수 있다. 또한 실제 영상과 깊이정보 맵을 부호화할 때, 도 4와 같이 서로 의존적으로 부호화/복호화 할 수 있다. 일 실시 예로, 이미 부호화된/복호화된 깊이정보맵을 이용하여 실제 영상을 부호화/복호화할 수 있으며, 반대로 이미

부호화된/복호화된 실제 영상을 이용하여 깊이정보맵을 부호화/복호화할 수 있다.

[0009] 일 실시 예로, 카메라 3대에서 획득한 실제 영상과 그것의 깊이정보맵을 부호화하기 위한 부호화 예측 구조를 도 5에 나타내었다. 도 5에서 3개의 실제 영상을 시점에 따라 T0, T1, T2로 나타내었고, 실제 영상과 동일한 위치의 3개의 깊이정보맵을 시점에 따라 D0, D1, D2로 나타내었다. 여기서 T0와 D0는 시점 0(View 0)에서 획득한 영상이며, T1와 D1는 시점 1(View 1)에서 획득한 영상이며, T2와 D2는 시점 2(View 2)에서 획득한 영상이다. 각 픽처는 I(Intra Picture), P(Uni-prediction Picture), B(Bi-prediction Picture)로 부호화될 수 있다. 도 5에서 화살표는 예측 방향을 나타낸다. 즉, 실제 영상과 그것의 깊이정보맵은 서로 의존적으로 부호화/복호화된다.

[0010] 실제 영상에서 현재 블록의 움직임 정보(움직임 벡터만을 의미할 수 있으며, 혹은 움직임벡터와 참조영상 번호와 단방향 예측인지 양방향 예측인지 그리고 시점간 예측인지 시간적 예측인지 또다른 예측인지를 의미할 수도 있다.)를 유추하기 위한 방법은 크게 시간적 예측과 시점간 예측으로 나뉜다. 시간적 예측은 동일한 시점 내에서 시간적 상관성을 이용한 예측 방법이고, 시점간 예측은 인접한 시점에서 시점간 상관성을 이용한 예측 방법이다. 이러한 시간적 예측과 시점간 예측은 한 픽처에서 서로 혼용되어 사용될 수 있다.

발명의 내용

해결하려는 과제

[0011] 본 발명에서는 비디오 코덱에서 구현 복잡도 및 계산 복잡도를 감소시키는 방법을 제안한다.

과제의 해결 수단

[0012] 본 발명은 부호화/복호화 모듈의 재사용성을 증가시키는 방법을 포함함.

발명의 효과

[0013] 본 발명은 비디오 부호화에서 부호화 및 복호화의 구현 복잡도 및 계산 복잡도를 감소시킬 수 있다.

[0014] 역방향 호환성을 제공하는 독립적인 시점(Independent View; 시점 0(View 0))에 대한 일반 영상의 부호화에 사용되는 모듈을 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들에 그대로 적용하여 구현 복잡도를 감소시키고, 그 다음으로 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들에 부분 인코더를 추가적으로 적용함으로써 부호화 효율을 향상시킬 수 있다.

도면의 간단한 설명

- [0015] 도 1. 3차원 비디오 시스템의 기본 구조와 데이터 형식
- 도 2. “balloons” 영상의 실제 영상과 깊이정보 맵 영상의 일 예: (a) 실제 영상, (b) 깊이정보 맵
- 도 3. 3D Video Codec에서 시점 간 예측 구조의 일 예
- 도 4. 3D Video Encoder/Decoder의 실시 일 예
- 도 5. 3D Video Codec의 예측 구조의 일 예
- 도 6. 3D Video Codec의 인코더 구조의 일 예
- 도 7. 3D-HEVC에서 병합 움직임 후보 리스트 생성 방법의 일 예
- 도 8. 병합 움직임 후보 목록으로 사용되는 현재 블록의 주변 블록들의 일 예
- 도 9. 병합 움직임 후보 리스트 생성 방법의 하드웨어 구현의 일 예
- 도 10. 제안 방법의 기본 개념도의 일 예
- 도 11. 제안하는 병합 움직임 후보 리스트 구성 방법의 일 예
- 도 12. 제안하는 병합 움직임 후보 리스트 구성 방법의 하드웨어 구현의 일 예
- 도 13. 제안하는 병합 움직임 후보 리스트 구성 방법의 실시 일 예

- 도 14. 확장 병합 움직임 후보 리스트를 구성하는 실시 일례
- 도 15. 확장 병합 움직임 후보 리스트를 구성하는 실시 일례
- 도 16. 병합 움직임 후보 리스트를 구성하는 장치 일례
- 도 17. 제안하는 확장 병합 움직임 후보 추가 장치 일례
- 도 17. 제안하는 시간적 병합 후보 유도과정 간소화 장치 일례
- 도 18. 확장 병합 움직임 후보 리스트를 구성하는 실시 일례
- 도 19. 주어진 CU(혹은 TU) 깊이가 2인 경우, 본 발명의 방법들을 적용하는 범위 결정 방식의 예. (O: 해당 깊이에 적용, X: 해당 깊이에 적용하지 않음.)

발명을 실시하기 위한 구체적인 내용

- [0016] 대용량의 3차원 영상 콘텐츠는 비트스트림의 양을 감소시키기 위하여 효율적으로 압축되어야 한다. 부호화 효율을 증가시키기 위하여 서로 다른 시점간 상관성을 이용할 수 있으며, 일반 영상(Texture View)과 깊이정보 맵(Depth View)간의 상관성을 이용할 수 있다. 하지만 이로 인해, 2차원 영상을 부호화할 때보다 더 많은 부호화 알고리즘이 요구되며, 이를 구현하기 위한 하드웨어 및 소프트웨어의 구현 복잡도가 증가되고 계산 복잡도가 증가되는 문제점이 존재한다. 실시 일례로, 도 6은 3차원 비디오 코덱(3D Video Codec)의 인코더 구조도를 나타낸다.
- [0017] 도 6의 3차원 비디오 코덱은 3개의 서로 다른 시점 영상들을 입력으로 받아 부호화하여통합된 하나의 비트스트림을 출력한다. 이때 영상들에는 일반 영상(Texture View)뿐만 아니라 깊이정보 맵(Depth View)도 포함될 수 있다. 도 6의 3차원 비디오 코덱은 시점 정보(View ID 정보)에 따라 서로 다른 인코더로 부호화된다. 즉, 도 6에서 시점 0(View 0)의 영상은 역방향 호환성(Backward compatibility)를 위하여 기존 2차원 비디오 코덱으로 부호화되어야 하므로, 기본 계층 인코더로 부호화 된다. 또한, 도 6에서 시점 1(View 1)과 시점 2(View 2)의 영상들은 시점간 예측 알고리즘과 일반 영상과 깊이정보 맵 간의 상관성을 이용한 알고리즘이 포함된 3차원 비디오 코덱으로 부호화되어야 하므로 향상 계층 인코더로 부호화된다. 또한, 일반 영상이 아닌 깊이정보 맵의 경우, 일반 영상의 부호화된 정보를 활용하여 부호화될 수 있으므로, 향상 계층 인코더로 부호화된다. 따라서, 시점 0(View 0)의 영상을 부호화할 때에 비해, 시점 1(View 1)과 시점 2(View 2)의 영상을 부호화할 때 더욱 복잡한 인코더가 요구되며, 또한 기본 계층의 일반 영상을 부호화할 때에 비해 깊이정보 맵을 부호화할 때 더욱 복잡한 인코더가 요구된다.
- [0018] HEVC에서는 영상 부호화/복호화 시 움직임 정보의 부호화 방법 중 하나로 병합 움직임(merge) 방법을 이용한다. 여기서 향상 계층에서의 부호화 효율을 증가시키기 위해서, 향상 계층에서는 기본 계층에서의 병합 움직임을 수정하여 향상된 병합 움직임을 사용한다. 실시 일례로, 도 7은 HEVC 기반 3차원 비디오 코덱인 3D-HEVC에서 병합 움직임 후보 리스트 생성 방법의 일례를 나타낸다.
- [0019] 도 7에서 확인할 수 있듯이, 3D-HEVC에서는 시점 0(View 0)에 대한 병합 움직임 구성 방법과 나머지 다른 시점들(시점 1(View 1)과 시점 2(View 2))에 대한 병합 움직임 구성 방법으로 서로 구분되어 있음을 알 수 있다. 도 7에서 현재 PU(Prediction Unit; 혹은 PB(Prediction Block) 혹은 임의 크기의 블록(Block)) 영상이 입력되면 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)를 통해, 시점 0(View 0)에 대한 병합 움직임 구성 방법과 나머지 다른 시점들(시점 1(View 1)과 시점 2(View 2))에 대한 병합 움직임 구성 방법 중에 하나가 선택되어 수행되며, 최종적으로 현재 PU 영상에 대한 병합 움직임 후보 리스트를 출력한다. 여기서 시점 0(View 0)에 대한 일반 영상은 역방향 호환성을 위해 기본 계층을 위한 병합 움직임 구성 방법을 이용하여 병합 움직임 후보 리스트를 구성한다. 반면, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵은 향상 계층을 위한 병합 움직임 구성 방법을 이용하여 병합 움직임 후보 리스트를 구성한다.
- [0020] 도 7에서 향상 계층을 위한 병합 움직임 구성 방법은 기본 계층을 위한 병합 움직임 구성 방법에 새로운 후보를 추가하거나 후보 리스트 순서를 수정하는 방법으로 수행된다. 즉, 향상 계층(나머지 다른 시점들(시점 1(View 1)과 시점 2(View 2))과 깊이정보 맵)을 위한 병합 움직임 구성 방법에는 기본 계층을 위한 병합 움직임 구성 방법이 포함되어 있다. 따라서, 향상 계층을 위한 병합 움직임 구성 방법은 기본 계층을 위한 병합 움직임 구성보다 복잡하게 구성되어 있으며, 계산 복잡도가 크다는 것을 알 수 있다. 또한, 하드웨어 혹은 소프트웨어로 구

현하는 측면에서 기본 계층을 위한 병합 움직임 구성 방법과 향상 계층을 위한 병합 움직임 구성 방법을 모두 구현해주어야하므로 구현 복잡도가 2배 이상 증가하는 것을 알 수 있다.

- [0021] 도 8은 병합 움직임 후보 리스트를 구성하는데 사용되는 주변 블록들을 나타낸다.
- [0022] 병합 움직임 후보 리스트는 도 8의 주변 블록(A, B, C, D, E)과 그리고 동일 위치의 후보 블록(H(혹은 M))에 대하여 해당 블록의 움직임 정보가 현재 블록의 병합 움직임에 이용될 수 있는지를 판단하여, 이용 가능한 경우에는 해당 블록의 움직임 정보를 병합 움직임 후보 리스트에 입력할 수 있다.
- [0023] 도 9는 실제 병합 움직임 후보 리스트 생성 방법의 하드웨어 구현의 일예를 나타낸다.
- [0024] 도 9에서 시점0(View 0)에 대한 일반 영상에서 적용되는 병합 움직임 리스트 구성 방법의 입력 파라미터와 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵에서 적용되는 병합 움직임 리스트 구성 방법의 입력 파라미터가 동일함을 알 수 있다. 단지 다른 점은 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵을 위해 “추가 움직임 F”와 “추가 움직임 G”가 추가된 부분이다. 또한 추가 움직임 정보들을 통해 병합 움직임 후보 리스트를 구성하는 부분이 변경된다. 즉, 추가 움직임 정보들을 병합 움직임 후보 리스트에 포함시키기 위해서(부호화 효율을 증가시키기 위해서), 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵을 위한 병합 움직임 리스트 구성 모듈을 새로 구현하여야한다. 이는 하드웨어의 구현 복잡도를 증가시킬 수 있다.
- [0025] 따라서, 본 발명에서는 향상 계층(실시 일예로, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵)을 위한 부호화 알고리즘 및 비디오 코덱의 구현 복잡도 및 계산 복잡도를 감소시키는 방법을 제안한다. 본 발명의 궁극적인 목적은 이미 하드웨어 칩 형태로 구현이 완료된 기본 계층(시점0(View 0)에 대한 일반 영상)을 위한 “병합 움직임 후보 리스트 구성” 모듈을 그대로 재사용하여 향상 계층(실시 일예로, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵)에 적용함으로써, 하드웨어 구현 복잡도를 감소시키는 것이다. 즉, 2차원 영상 서비스에 사용되는 기본 계층을 위한 부호화/복호화기(구체적인 실시 일예로, “병합 움직임 후보 리스트 구성” 모듈)를 가지고 있는 소비자가 3차원 영상 서비스를 받고자 한다면, 단지 추가적인 모듈(구체적인 실시 일예로, 향상 계층을 위한 “병합 움직임 후보 리스트 구성” 모듈)만 부착하면 손쉽게 3차원 영상 서비스를 받을 수 있게 된다.
- [0026] 본 발명에서는 비디오 코덱의 구현 복잡도 및 계산 복잡도를 감소시켜서 데이터 처리량을 향상시키는 방법을 제안한다.
- [0027] [기본 방법]
- [0028] 도 10의 3차원 비디오 코덱은 3개의 서로 다른 시점 영상들을 입력으로 받아 부호화하여통합된 하나의 비트스트림을 출력한다. 이때 영상들에는 일반 영상(Texture View)뿐만 아니라 깊이정보 맵(Depth View)도 포함될 수 있다. 도 10의 3차원 비디오 코덱은 모든 시점에 대한 일반 영상 및 깊이정보 맵이 동일한 인코더(부호화기)로 부호화를 수행한다. 실시 일예로, 여기서 사용되는 인코더는 MPEG-1, MPEG-2, MPEG-4 Part 2 Visual, H.264/AVC, VC-1, AVS, KTA, HEVC(H.265/HEVC) 등이 될 수 있다. 이때, 일반 영상의 시점 0(View 0)이 아닌 다른 시점(시점 1(View 1)과 시점 2(View 2))들과 깊이정보 맵에 대하여, 부호화 효율을 증가시키기 위하여 부분 인코더가 추가적으로 수행된다. 마지막으로 일반 영상의 시점 0(View 0)에 대한 비트스트림과 그리고 다른 시점(시점 1(View 1)과 시점 2(View 2))들과 깊이정보 맵들에 대한 비트스트림들은 하나의 비트스트림으로 다중화된다. 즉, 역방향 호환성을 제공하는 독립적인 시점(Independent View; 시점 0(View 0))에 대한 일반 영상의 부호화에 사용되는 모듈을 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들에 그대로 적용하여 구현 복잡도를 감소시키고, 그 다음으로 의존적인 시점(dependent view; 시점 1(View 1)과 시점 2(View 2))에 대한 일반 영상과 깊이정보 맵들에 부분 인코더를 추가적으로 적용함으로써 부호화 효율을 향상시킬 수 있다.
- [0029] [세부 방법]
- [0030] 도 10의 방법을 부호화/복호화 전체에 대하여 적용할 수 있으며, 부호화/복호화의 각 모듈마다 각각 적용할 수

있다. 실시 일례로, HEVC 기반 3차원 비디오 코덱인 3D-HEVC에서 병합 움직임 후보 리스트를 구성(혹은 생성)하는 방법의 일례를 도 11에 나타내었다.

[0031] 도 11의 제안하는 병합 움직임 후보 리스트 구성 방법에서 입력은 현재 PU 정보(혹은 영상)와 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)이며, 출력은 현재 PU에 대한 병합 움직임 후보 리스트이다. 현재 PU에 대하여 “기본 병합 움직임 리스트 구성” 단계가 기본적으로 수행되어 “기본 병합 움직임 후보 리스트”를 출력한다. 실시 일례로, 여기서 “기본 병합 움직임 리스트 구성”은 HEVC에서의 병합 움직임 후보 리스트 구성 방법을 그대로 사용할 수 있다. 다음으로, 현재 PU 영상이 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)에 따라 “추가 병합 움직임 리스트 구성” 단계가 추가적으로 수행된다. 이때, “추가 병합 움직임 리스트 구성” 단계에서 입력은 “기본 병합 움직임 리스트 구성” 단계에서의 “기본 병합 움직임 후보 리스트”이며, 출력은 “확장 병합 움직임 후보 리스트”이다. “추가 병합 움직임 리스트 구성” 단계는 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵들에 대하여 수행된다.

[0032] 도 12는 도 11의 제안하는 병합 움직임 후보 리스트 구성 방법을 하드웨어로 구현한 일례를 나타낸다.

[0033] 도 12에서 입력은 도 8의 주변 블록(A, B, C, D, E)과 그리고 동일 위치의 후보 블록(H(혹은 M))과 그리고 추가 움직임(F, G)이다. 출력은 시점0(View 0)에 대한 일반 영상의 경우에는 “기본 병합 움직임 후보 리스트”이고, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵의 경우에는 “확장 병합 움직임 후보 리스트”이다. 도 12에서는 먼저 도 8의 주변 블록(A, B, C, D, E)과 그리고 동일 위치의 후보 블록(H(혹은 M))을 이용하여 “기본 병합 움직임 리스트 구성” 단계를 수행하여 “기본 병합 움직임 후보 리스트”를 출력한다.

[0034] 다음으로, 현재 PU 영상이 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)에 따라 “추가 병합 움직임 리스트 구성” 단계가 추가적으로 수행된다. 즉, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵일 경우에는 “추가 병합 움직임 리스트 구성” 단계가 추가적으로 수행된다. “추가 병합 움직임 리스트 구성” 단계에서 입력은 “기본 병합 움직임 리스트 구성” 단계에서의 “기본 병합 움직임 후보 리스트”와 “추가 움직임(F, G)”이며, 출력은 “확장 병합 움직임 후보 리스트”이다.

[0035] 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵을 위한 병합 움직임 후보 리스트를 구성하기 위해서, 새로운 모듈을 구현하지 않고 단지 추가적인 부분 모듈만을 구현함으로써 하드웨어 구현 복잡도를 감소시킬 수 있다. 즉, 이미 하드웨어 칩 형태로 구현이 완료된 기본 계층(시점0(View 0)에 대한 일반 영상)을 위한 “병합 움직임 후보 리스트 구성” 모듈을 그대로 재사용하여 항상 계층(실시 일례로, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵)에 적용함으로써, 하드웨어 구현 복잡도를 감소시킬 수 있다.

[0036] 도 13는 도 11과 도 12에서 제안하는 병합 움직임 후보 리스트 구성 방법의 실시 일례를 나타낸다.

[0037] 도 13에서 입력은 현재 PU 정보(혹은 영상)와 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)이며, 출력은 현재 PU에 대한 병합 움직임 후보 리스트이다. 우선, 현재 PU에 대하여 기본 병합 움직임 후보 리스트가 구성된다. 실시 일례로, 여기서 기본 병합 움직임 후보 리스트는 HEVC에서의 병합 움직임 후보 리스트 구성 방법을 그대로 사용할 수 있다. 다음으로, 현재 PU 영상이 일반 영상인지 혹은 깊이정보 맵 영상인지에 대한 정보(Texture/Depth 정보)와 시점 정보(ViewID 정보)에 따라 확장 병합 움직임 후보 리스트가 구성된다. 이때, 확장 병합 움직임 후보 리스트는 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵들에 대하여 수행되며, 추가적인 움직임 정보인 “움직임 정보 F”가 추가될 수 있다. 만일, 현재 PU가 일반 영상의 시점 0(View 0)일 경우, 도 13의 출력은 “기본 병합 움직임 후보 리스트”가 될 수 있다. 그렇지 않고 현재 PU가 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵들일 경우, 도 13의 출력은 “확장 병합 움직임 후보 리스트”가 될 수 있다. 이때 “확장 병합 움직임 후보 리스트”의 후보 개수는 “기본 병합 움직임 후보 리스트”의 후보 개수보다 많을 수 있다.

[0038] 이 때, “확장 병합 움직임 후보 리스트”를 구성하는 방법은 다양하게 적용될 수 있다.

[0039] 실시 일례로, 도 14와 같이, 추가적인 움직임 정보인 “움직임 정보 F”를 움직임 후보 리스트의 첫번째 항목

(혹은 임의의 위치에 해당하는 항목)에 삽입할 수 있다. 이때, 추가적인 움직임 후보를 삽입하기 전에 “기본 병합 움직임 후보 리스트”의 첫번째 움직임 후보인 “움직임 정보 A”와 추가적인 움직임 후보인 “움직임 후보 F”를 서로 비교하여 동일하지 않을 경우에 리스트의 첫번째 항목에 삽입할 수 있으며, 그 역도 가능하다. 여기서 두 움직임 정보를 비교할 때, 두 움직임 벡터의 차이가 임의의 임계값 이내일 경우에는 삽입하지 않을 수 있으며, 그 역도 가능하다. 또한, 여기서 두 움직임 정보의 참조 영상이 동일하지 않을 경우에 삽입할 수 있으며, 그 역도 가능하다.

[0040] 또 다른 실시 일례로, 도 15와 같이, 추가적인 움직임 정보인 “움직임 정보 F”를 움직임 후보 리스트의 첫번째 항목(혹은 임의의 위치에 해당하는 항목)에 삽입하고, 또 다른 추가적인 움직임 정보인 “움직임 정보 G”를 움직임 후보 리스트의 세번째 항목(혹은 임의의 위치에 해당하는 항목)에 삽입할 수 있다. 이때, 추가적인 움직임 후보를 삽입하기 전에 “기본 병합 움직임 후보 리스트”에 있던 원래 항목과 추가적인 움직임 후보를 서로 비교하여 동일하지 않을 경우에 리스트의 첫번째 항목에 삽입할 수 있으며, 그 역도 가능하다. 여기서 두 움직임 정보를 비교할 때, 두 움직임 벡터의 차이가 임의의 임계값 이내일 경우에는 삽입하지 않을 수 있으며, 그 역도 가능하다. 또한, 여기서 두 움직임 정보의 참조 영상이 동일하지 않을 경우에 삽입할 수 있으며, 그 역도 가능하다.

[0041] [추가 방법들]

[0042] 도 10과 도 11과 도 12와 그리고 도 13의 방법들은 다음과 같이 다양하게 적용될 수 있다.

[0043] 1. 실시 일례로, 기본 인코더(혹은 기본 모듈)를 시점 0(View 0)에 대한 일반 영상뿐만 아니라 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵들에 적용할 수 있다.

[0044] 2. 또 다른 실시 일례로, 기본 인코더(혹은 기본 모듈)를 복잡도가 높은 작은 블록단위(일례로, 8x8 단위 혹은 임의의 블록 크기)에만 적용할 수 있다. 이때, 시점 1(View 1)과 시점 2(View 2)에 대한 일반 영상과 깊이정보 맵들에 대하여, 작은 블록 크기 이하에서는 기본 인코더(혹은 기본 모듈)를 사용하여 부호화되고, 작은 블록 크기보다 클 경우에는 기본 인코더(혹은 기본 모듈)와 부분 인코더(혹은 확장 모듈)를 사용하여 부호화될 수 있다. 여기서 기본 인코더(혹은 기본 모듈)는 도 11와 도 13에서 “기본 병합 움직임 리스트 구성” 단계를 의미할 수 있으며, 또한 부분 인코더(혹은 확장 모듈)은 도 11와 도 13에서 “추가 병합 움직임 리스트 구성” 단계를 의미할 수 있다.

[0045] 도 12의 방법을 3D-HEVC에 적용하여 병합 움직임 후보 리스트를 구성하는 장치는 도 16과 같이 작동 한다.

[0046] 먼저, 리스트에 기본 병합 움직임 후보들을 추가 한다. 그 후, 현재 PU가 위치하는 픽처가깊이정보 맵 또는 의존적인 시점이라면, 기존 리스트에 확장 병합 움직임 후보들을 추가한다.

[0047] 표 1~의 기술 형식은 현재 MPEG(Moving Picture Experts Group)과 VCEG(Video Coding Experts Group)에서 공동으로 표준화를 진행 중인 JCT-3V(Joint Collaborative Team on 3D Video Coding Extensions of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11)의 기술 형식이다.

[0048] 표 1은 JCT-3V 표준화에서 확장 병합 움직임 후보 추가 과정을 포함하는 프로세스의 입출력의 일례이다. 확장 병합 움직임 후보 추가 모듈은 입력으로서, 현재 블록과 예측 블록의 위치, 크기, 파티션 정보, 병합 움직임 후보 리스트(mergCandList), 기본 병합 움직임 후보가 추가 되었는지를 지시하는 플래그(availableFlagN)가 사용된다. 이 때, N은 좌(left), 상(above), 우상(above-right), 좌하(bottom-left), 좌상(above-left) 위치의 후보임을 의미하는 A0, A1, B0, B1, B2로 대체될 수 있다. 입력 받은 병합 움직임 후보 리스트(mergCandList)에는 기본 병합 움직임 후보들이 임의의 우선순위대로 저장되어 있다. 일 예로, 좌 후보, 상 후보, 우상 후보, 좌하 후보, 좌상 후보, 시간 적 예측 후보, 합성 양방향 예측 후보, 움직임이 0인 후보 순서로 저장 되어 있을 수 있다. 출력은 확장 병합 움직임 후보들에 대한 추가작업이 완료 된 병합 움직임 후보 리스트이다.

표 1. 확장 병합 움직임 후보 추가 모듈의 입력 및 출력 일예

H.8.5.3.2.1 Derivation process for luma motion vectors for merge mode Inputs to this process are: - a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, - a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture, - a variable nCbS specifying the size of the current luma coding block, - two variables nPbW and nPbH specifying the width and the height of the luma prediction block, - a variable partIdx specifying the index of the current prediction unit within the current coding unit, - the merge candidate lists mergeCandList, - the availability flags availableFlagN of the neighbouring prediction units are (with N being replaced by A₀, A₁, B₀, B₁ or B₂). Outputs of this process are: - the luma motion vectors mvL0 and mvL1, - the reference indices refIdxL0 and refIdxL1, - the prediction list utilization flags predFlagL0 and predFlagL1, - the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1, the flag vspModeFlag

도 17은 제안하는 확장 병합 움직임 후보 추가 장치의 일 예이다. 장치의 구동 과정은 다음과 같다.

1. 플래그 `iv_mv_pred_flag[nuh_layer_id]`는 현재 PU가 시점 간 예측을 시도 할 수 있는가를 의미한다. 플래그가 1이라면, 시점 간 병합 후보(IvMC), 시차 병합 후보(IvDC), 이동 된 시점 간 병합 후보(IvMCShift)의 사용 가능 여부를 확인하여 각각 플래그 `availableFlagIvMC`, `availableIvMCShift`, `availableFlagIvDC`에 저장하고, 사용 가능한 후보들의 움직임 정보를 유도한다.
2. 플래그 `view_synthesis_pred_flag[nuh_layer_id]`는 현재 PU가 시점 합성 예측을 시도 할 수 있는가를 의미한다. 플래그가 1이라면, 시점 간 합성 병합 후보의 사용 가능 여부를 플래그 `availableFlagVSP`에 저장하고, 사용 가능하다면 움직임 정보를 유도한다.
3. 플래그 `DepthFlag`는 현재 PU가 깊이정보 맵이며, 텍스처 블록으로부터 움직임 예측을 시도 할 수 있는가를 의미한다. 플래그가 1이면, 텍스처 병합 후보의 사용 가능 여부를 플래그 `availableFlagT`에 저장하고, 사용 가능하다면 움직임 정보를 유도한다.
4. 현재까지 기본 병합 움직임 후보들로만 구성 된 병합 움직임 후보 리스트(`mergeCandList`)와 각 후보에 대한 시점 간 예측 플래그(`mergeCandIsVspFlag`)는 다음과 같은 과정으로 재구성 된다.
 - a. `numMergeCand`은 병합 움직임 후보의 총 개수, `numMergeCand3DV`는 확장 병합 움직임 후보들의 총 개수, `numA1B1B0`는 기본 병합 움직임 후보 중 좌(left), 상(above), 우상(above-right) 위치에 해당하는 후보들의 개수, `numA0B2`는 기본 병합 움직임 후보 중 좌하(bottom-left), 좌상(above-left)의 위치에 해당하는 후보들의 개수이다. `numMergeCand`, `numMergeCand3DV`, `numA1B1B0`, `numA0B2`을 0으로 초기화 한다.
 - b. 좌 후보가 사용 가능하다면, `numA1B1B0`을 1 증가시킨다. 또한, 좌 후보가 시점 간 합성 예측(View Synthesis Prediction, 이하 VSP)을 사용하였는지를 플래그로 저장한다.
 - c. 상 후보가 사용 가능하다면, `numA1B1B0`을 1 증가시킨다. 또한, 상 후보가 VSP를 사용하였는지를 플래그로 저장한다.
 - d. 우상 후보가 사용 가능하다면, `numA1B1B0`을 1 증가시킨다. 또한, 우상 후보가 VSP를 사용하였는지를 플래그로 저장한다.
 - e. 좌하 후보가 사용 가능하다면, `numA0B2`을 1 증가시킨다. 또한, 좌하 후보가 VSP 사용하였는지를 플래그로 저장한다.
 - f. 좌상 후보가 사용 가능하다면, `numA0B2`을 1 증가시킨다. 또한, 좌상 후보가 VSP를 사용하였는지를 플래그로 저장한다.
 - g. 플래그 `availableFlagT`가 1이면 다음 과정을 수행한다.

- [0063] - 텍스처 후보의 움직임 정보와 사용 가능한 좌 후보 및 상 후보를 비교한다. 그 결과, 좌 후보 및 상 후보의 움직임 정보와 모두 다르다면 병합 움직임 후보 리스트(이하 리스트)의 가장 앞에 새로운 공간을 하나 만든 후 텍스처 후보를 추가한다. 그렇지 않다면 리스트의 첫 번째 위치와 두 번째 위치의 후보의 위치를 서로 바꾸고 numA1B1B0을 1 감소시킨다.
- [0064] - numMergeCand3DV, numMergeCand을 1 증가시킨다.
- [0065] h. 플래그 availableFlagIvMC가 1이면 다음 과정을 수행한다.
- [0066] - 시점 간 병합 후보(IvMC)의 움직임 정보와 사용 가능한 좌 후보 및 상 후보를 비교한다. 그 결과, 좌 후보 및 상 후보의 움직임 정보와 모두 다르다면 리스트의 가장 앞에 새로운 공간을 하나 만든 후 시점 간 병합 후보(IvMC)를 추가한다. 그렇지 않다면 리스트의 첫 번째 위치와 두 번째 위치의 후보의 위치를 서로 바꾸고 numA1B1B0을 1 감소시킨다.
- [0067] - numMergeCand3DV, numMergeCand을 1 증가시킨다.
- [0068] i. numMergeCand에 numA1B1B0을 더한다.
- [0069] j. 플래그 availableFlagIvDC가 1이면 다음 과정을 수행한다.
- [0070] - 시차 병합 후보(IvDC)의 움직임 정보와 사용 가능한 좌 후보 및 상 후보를 비교한다. 그 결과, 좌 후보 및 상 후보의 움직임 정보와 모두 다르다면 병합 리스트의 numMergeCand 위치에 새로운 공간을 하나 만든 후 시차 병합 후보(IvDC)를 추가한다.
- [0071] - umMergeCand3DV, numMergeCand을 1 증가시킨다.
- [0072] k. 플래그 availableFlagVSP가 1이면 시점 합성 병합 후보를 리스트에 추가하고 numMergeCand3DV, numMergeCand를 1 증가시킨다.
- [0073] l. numMergeCand에 numA0B2를 더한다.
- [0074] m. 플래그 availableFlagIvMCShift가 1이고 numMergeCand가 리스트의 최대 길이(일예로 6)보다 작으면, 다음 과정을 수행한다.
- [0075] - 시점 간 병합 후보(IvMC)가 사용 가능하다면, 시점 간 병합 후보와 이동 된 시점 간 병합 후보(IvMCShift)를 비교하여 서로 다르다면 리스트의 numMergeCand 위치에 새로운 공간을 하나 만든 후 이동 된 시점 간 병합 후보를 추가한다.
- [0076] - numMergeCand3DV, numMergeCand을 1 증가시킨다.
- [0077] n. 플래그 availableFlagIvMCShift가 0이고, 현재 PU가 깊이정보 맵 상에 위치하지 않고, numMergeCand가 리스트의 최대 길이(일예로 6)보다 작으면, 다음 과정을 수행한다.
- [0078] - 이동 된 시차 예측 후보(IvDCShift)가 사용 가능하다면, 리스트의 numMergeCand 위치에 새로운 공간을 하나 만든 후 이동 된 시점 간 병합 후보를 추가한다.
- [0079] - numMergeCand3DV, numMergeCand을 1 증가시킨다.
- [0080] o. 리스트의 공간이 모두 채워졌는지 확인하기 위하여, numMergeCand를 다음과 같이 수정한다.
- [0081] - $(5 + \text{numMergeCand3DV})$ 가 MaxNumMergeCand보다 크다면 리스트의 최대 길이(일 예로 6)으로 수정한다.
- [0082] - 그렇지 않다면 $(\text{numMergeCandBase} + \text{numMergeCand3DV})$ 로 수정한다.
- [0083] p. 리스트의 공간이 후보들로 모두 채워지지 않았다면 남아있는 공간에, 움직임이 0인 후보를 추가한다.
- [0084] 과정 h, j에서 시점 간 병합 후보와 시차 병합 후보를 리스트의 기존 후보들과 비교 시, 리스트 내의 모든 후보들과 비교하지 않고 그 중 일부와 비교하는 방법을 사용하여 복잡도를 낮출 수 있다. 일 예로 좌 후보와 상 후보만을 비교에 사용할 수 있다.

[0085]

표 2. 확장 병합 움직임 후보 추가 과정 일예

H.8.5.3.2.1 Derivation process for luma motion vectors for merge mode

1. Depending on $iv_mv_pred_flag[nuh_layer_id]$, the following applies.
 If $iv_mv_pred_flag[nuh_layer_id]$ is equal to 0, the flags $availableFlagIvMC$, $availableIvMCShift$ and $availableFlagIvDC$ are set equal to 0.
 Otherwise ($iv_mv_pred_flag[nuh_layer_id]$ is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause H.8.5.3.2.10 is invoked with the luma location (xPb, yPb) , the variables $nPbW$ and $nPbH$, as the inputs and the output is assigned to the availability flags $availableFlagIvMC$, $availableIvMCShift$ and $availableFlagIvDC$, the reference indices $refIdxLXivMC$, $refIdxLXivMCShift$ and $refIdxLXivDC$, the prediction list utilization flags $predFlagLXivMC$, $predFlagLXivMCShift$ and $predFlagLXivDC$, and the motion vectors $mvLXivMC$, $mvLXivMCShift$ and $mvLXivDC$ (with X being 0 or 1, respectively).
2. Depending on $view_synthesis_pred_flag[nuh_layer_id]$, the following applies:
 If $view_synthesis_pred_flag[nuh_layer_id]$ is equal to 0, the flag $availableFlagVSP$ is set equal to 0.
 Otherwise ($view_synthesis_pred_flag[nuh_layer_id]$ is equal to 1), the derivation process for a view synthesis prediction merge candidate as specified in subclause H.8.5.3.2.13 is invoked with the luma locations (xCb, yCb) as input and the outputs are the availability flag $availableFlagVSP$, the reference indices $refIdxL0VSP$ and $refIdxL1VSP$, the prediction list utilization flags $predFlagL0VSP$ and $predFlagL1VSP$, and the motion vectors $mvL0VSP$ and $mvL1VSP$.
3. Depending on $DepthFlag$, the following applies:
 If $DepthFlag$ is equal to 0, the variable $availableFlagT$ is set equal to 0.
 Otherwise ($DepthFlag$ is equal to 1), the derivation process for the texture merging candidate as specified in subclause H.8.5.3.2.14 is invoked with the luma location (xPb, yPb) , the variables $nPbW$ and $nPbH$ as the inputs and the outputs are the flag $availableFlagT$, the prediction utilization flags $predFlagL0T$ and $predFlagL1T$, the reference indices $refIdxL0T$ and $refIdxL1T$, and the motion vectors $mvL0T$ and $mvL1T$.
4. The merge candidate lists $mergeCandList$ and $mergeCandIsVspFlag$ are constructed as specified by the following ordered steps:
 - a. The variable $numMergeCand$ is set equal to 0, **the variable $numMergeCand3DV$, $numA1B1B0$, $numA0B2$ are set equal to 0.**
 - b. When $availableFlagA_1$ is equal to 1, the variable $numA1B1B0$ is increased by 1, and the following applies:
 - If $VspModeFlag[xPb - 1][yPb + nPbH - 1]$ is equal to 1, the entry $mergeCandIsVspFlag[numA1B1B0]$ is set equal to $VspModeFlag[xPb - 1][yPb + nPbH - 1]$.
 - c. When $availableFlagB_1$ is equal to 1, the variable $numA1B1B0$ is increased by 1, and the following applies:
 - If $VspModeFlag[xPb + nPbW - 1][yPb - 1]$ is equal to 1, the entry $mergeCandIsVspFlag[numA1B1B0]$ is set equal to $VspModeFlag[xPb + nPbW - 1][yPb - 1]$.
 - d. When $availableFlagB_0$ is equal to 1, the variable $numA1B1B0$ is increased by 1, and the following applies:
 - **If $VspModeFlag[xPb + nPbW][yPb - 1]$ is equal to 1, the entry $mergeCandIsVspFlag[numA1B1B0]$ is set equal to $VspModeFlag[xPb + nPbW][yPb - 1]$.**

[0086]

- e. When availableFlagA₀ is equal to 1, the variable numA0B2 is increased by 1, and the following applies:

== If VspModeFlag[xPb - 1][yPb + nPbH] is equal to 1, the entry mergeCandIsVspFlag[numA1B1B0] is set equal to VspModeFlag[xPb - 1][yPb + nPbH].

- f. When availableFlagB₂ is equal to 1, the variable numA0B2 is increased by 1, and the following applies:

== If VspModeFlag[xPb - 1][yPb - 1] is equal to 1, the entry mergeCandIsVspFlag[numA1B1B0 + numA0B2] is set equal to VspModeFlag[xPb - 1][yPb - 1].

- g. When availableFlagT is equal to 1, the following applies:

The variable pruneA1, pruneB1 is set equal to 0.

If all of the following conditions is true,

== availableFlagA1 == 1,

== predFlagLXA1 == predFlagLXT, (with X being replaced by 0 and 1),

== mvLXA1 == mvLXT (with X being replaced by 0 and 1),

== refIdxLXA1 == refIdxLXT (with X being replaced by 0 and 1),

pruneA1 is set equal to 1.

If all of the following conditions is true,

== availableFlagB1 == 1,

== predFlagLXB1 == predFlagLXT, (with X being replaced by 0 and 1),

== mvLXB1 == mvLXT (with X being replaced by 0 and 1),

== refIdxLXB1 != refIdxLXT (with X being replaced by 0 and 1),

pruneB1 is set equal to 1.

If pruneA1 is equal to 0 and pruneB1 is equal to 0, k ranges from 5 to numMergeCand + 1, inclusive, and mergeCandList[k] is set equal to mergeCandList[k - 1], mergeCandIsVspFlag[k] is set equal to mergeCandIsVspFlag[k - 1], the entry mergeCandList[numMergeCand] is set equal to T, the entry mergeCandIsVspFlag[numMergeCand] is set equal to 0 and the variable numMergeCand3DV is increased by 1.

Otherwise, if pruneB1 is equal to 1 and entry mergeCandList[1] is equal to B_i, the following applies:

== (mergeCandList[0], mergeCandList[1]) = Swap(mergeCandList[1], mergeCandList[0])

== The variable numA1B1B0 is decreased by 1.

== The variable numMergeCand3DV is increased by 1.

The variable numMergeCand is increased by 1.

- h. When availableFlagIvMC is equal to 1, the following applies:

The variable pruneA1, pruneB1 is set equal to 0.

If all of the following conditions is true,

== availableFlagA1 == 1,

== predFlagLXA1 == predFlagLXIvMC, (with X being replaced by 0 and 1),

```

mvLXA1 == mvLXivMC (with X being replaced by 0 and 1),
refldxLXA1 == refldxLXivMC (with X being replaced by 0 and 1),
pruneA1 is set equal to 1.
If all of the following conditions is true,
availableFlagB1 == 1,
predFlagLXB1 == predFlagLXivMC, (with X being replaced by 0 and 1),
mvLXB1 == mvLXivMC (with X being replaced by 0 and 1),
refldxLXB1 == refldxLXivMC (with X being replaced by 0 and 1),
pruneB1 is set equal to 1.
If pruneA1 is equal to 0 and pruneB1 is equal to 0, k ranges from 5 to numMergeCand
+ 1, inclusive, and mergeCandList[ k ] is set equal to mergeCandList[ k - 1 ],
mergeCandIsVspFlag[ k ] is set equal to mergeCandIsVspFlag[ k - 1 ], the entry
mergeCandList[ numMergeCand ] is set equal to ivMC, the entry
mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable
numMergeCand3DV is increased by 1.
Otherwise, if pruneB1 is equal to 1 and entry mergeCandList[ 1 ] is equal to B1, the
following applies:
- ( mergeCandList[ 0 ], mergeCandList[ 1 ] ) = Swap( mergeCandList[ 1 ],
mergeCandList[ 0 ] )
- The variable numA1B1B0 is decreased by 1.
- The variable numMergeCand3DV is increased by 1.
The variable numMergeCand is increased by 1.
i. The variable numMergeCand is set equal to numMergeCand + numA1B1B0.
j. When availableFlagivDC is equal to 1, and one or more of the following conditions is true,
- availableFlagA1 == 0,
- predFlagLXA1 != predFlagLXivDC, (with X being replaced by 0 and 1),
- mvLXA1 != mvLXivDC (with X being replaced by 0 and 1),
- refldxLXA1 != refldxLXivDC (with X being replaced by 0 and 1),
and one or more of the following conditions is true,
- availableFlagB1 == 0,
- predFlagLXB1 != predFlagLXivDC, (with X being replaced by 0 and 1),
- mvLXB1 != mvLXivDC (with X being replaced by 0 and 1),
- refldxLXB1 != refldxLXivDC (with X being replaced by 0 and 1),
k ranges from 5 to numMergeCand + 1, inclusive, and mergeCandList[ k ] is set equal
to mergeCandList[ k - 1 ], mergeCandIsVspFlag[ k ] is set equal to
mergeCandIsVspFlag[ k - 1 ], the entry mergeCandList[ numMergeCand ] is set equal to
ivDC, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0, the variable
numMergeCand3DV is increased by 1 and the variable numMergeCand is increased by 1.
k. When availableFlagVSP is equal to 1, k ranges from 5 to numMergeCand + 1, inclusive,
and mergeCandList[ k ] is set equal to mergeCandList[ k - 1 ], mergeCandIsVspFlag[ k ] is
set equal to mergeCandIsVspFlag[ k - 1 ], the entry mergeCandList[ numMergeCand ] is
set equal to VSP, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal 1, the

```

[0088]

	variable numMergeCand3DV is increased by 1 and the variable numMergeCand is increased by 1.
l.	The variable numMergeCand is set equal to numMergeCand + numA0B2.
m.	When availableFlagIvMCShift is equal to 1 and numMergeCand is less than 6, and one or more of the following conditions are true, - availableFlagIvMC == 0, - predFlagLXMC != predFlagLXMCShift (with X being replaced by 0 and 1), - mvLXMC != mvLXivMCShift (with X being replaced by 0 and 1), - refIdxLXMC != refIdxLXMCShift (with X being replaced by 0 and 1), k ranges from 5 to numMergeCand + 1, inclusive, and mergeCandList[k] is set equal to mergeCandList[k - 1], mergeCandIsVspFlag[k] is set equal to mergeCandIsVspFlag[k - 1], the entry mergeCandList[numMergeCand] is set equal to IvMCShift, the entry mergeCandIsVspFlag[numMergeCand] is set equal to 0, the variable numMergeCand3DV is increased by 1 and the variable numMergeCand is increased by 1.
n.	A variable availableFlagIvDCShift is set to 0 and when all of the following conditions are true - DepthFlag is equal to 0, availableFlagIvMCShift is equal to 0, - numMergeCand is less than 6, the derivation process for the shifted disparity merging candidate as specified in subclause H.8.5.3.2.15 is invoked with the availability flags availableFlagN, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N, of every candidate N being in mergeCandList, mergeCandList, mergeCandIsVspFlag, and numMergeCand as the inputs and the outputs are the flag availableFlagIvDCShift, the prediction utilization flags predFlagL0IvDCShift and predFlagL1IvDCShift, the reference indices refIdxL0IvDCShift and refIdxL1IvDCShift, and the motion vectors mvL0IvDCShift and mvL1IvDCShift. When availableFlagIvDCShift is equal to 1, k ranges from 5 to numMergeCand + 1, inclusive, and mergeCandList[k] is set equal to mergeCandList[k - 1], mergeCandIsVspFlag[k] is set equal to mergeCandIsVspFlag[k - 1], the entry mergeCandList[numMergeCand] is set equal to IvDCShift, the entry mergeCandIsVspFlag[numMergeCand] is set equal to 0, the variable numMergeCand3DV is increased by 1 and the variable numMergeCand is increased by 1.
5.	The variable numMergeCand is set to (5 + numMergeCand3DV > MaxNumMergeCand ? MaxNumMergeCand : numMergeCandBase + numMergeCand3DV), and the variable numCurrMergeCand is set equal to numMergeCand.
6.	The derivation process for zero motion vector merging candidates specified in subclause 8.5.3.2.4 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, and numCurrMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0zeroCand _m and refIdxL1zeroCand _m , the prediction list utilization flags predFlagL0zeroCand _m and predFlagL1zeroCand _m , and the motion vectors mvL0zeroCand _m and mvL1zeroCand _m of every new candidate zeroCand _m being added into mergeCandList. The number of candidates being added, numZeroMergeCand, is set equal to (dded, numZeroMergeCand, is set equal to When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand to d is greater, and mergeCandIsVspFlag[d] is greater than d is greater than 0. r.
	

[0089]

[0090]

제안하는 방법에서는 혼합 양방향 예측 후보를 유도할 시, 확장 움직임 병합 후보들을 추가로 이용하는 기존 방법을 사용하지 않고, HEVC 표준에서 사용하는 기본 움직임 병합 후보들만을 혼합하여 유도하는 방법을 사용하여 기존 방법보다 간단하면서 거의 동일한 부호화 효율을 얻을 수 있다. 표 3은 혼합 양방향 예측 후보를 유도하는 과정의 일예이다. 의존적인 시점이나 깊이정보 맵 만을 위한 새로운 모듈을 제작할 필요 없이, 기존의 모듈을 재사용 한다.

[0091] 표 3. 제안하는 HEVC 혼합 양방향 예측 후보 유도 과정 재사용 일예

H.8.5.3.2.3 Derivation process for combined bi-predictive merging candidates The specifications in subclause 8.5.2.1.3 apply. Inputs to this process are: a merging candidate list <code>mergeCandList</code>; a list <code>mergeCandIsVspFlag</code>; the reference indices <code>refIdxL0N</code> and <code>refIdxL1N</code> of every candidate <code>N</code> in <code>mergeCandList</code>; the prediction list utilization flags <code>predFlagL0N</code> and <code>predFlagL1N</code> of every candidate <code>N</code> in <code>mergeCandList</code>; the motion vectors <code>mvL0N</code> and <code>mvL1N</code> of every candidate <code>N</code> in <code>mergeCandList</code>; the number of elements <code>numCurrMergeCand</code> within <code>mergeCandList</code>; the number of elements <code>numOrigMergeCand</code> within the <code>mergeCandList</code> after the spatial and temporal merge candidate derivation process; Outputs of this process are: the merging candidate list <code>mergeCandList</code>; the number of elements <code>numCurrMergeCand</code> within <code>mergeCandList</code>; the reference indices <code>refIdxL0combCandk</code> and <code>refIdxL1combCandk</code> of every new candidate <code>combCandk</code> added into <code>mergeCandList</code> during the invocation of this process; the prediction list utilization flags <code>predFlagL0combCandk</code> and <code>predFlagL1combCandk</code> of every new candidate <code>combCandk</code> added into <code>mergeCandList</code> during the invocation of this process; the motion vectors <code>mvL0combCandk</code> and <code>mvL1combCandk</code> of every new candidate <code>combCandk</code> added into <code>mergeCandList</code> during the invocation of this process; When <code>numOrigMergeCand</code> is greater than 1 and less than <code>MaxNumMergeCand</code>, the variable <code>numInputMergeCand</code> is set equal to <code>numCurrMergeCand</code>, the variable <code>combIdx</code> is set equal to 0, the variable <code>combStop</code> is set equal to FALSE, and the following steps are repeated until <code>combStop</code> is equal to TRUE: 1. The variables <code>l0CandIdx</code> and <code>l1CandIdx</code> are derived using <code>combIdx</code> as specified in Table 8-6. 2. The following assignments are made, with <code>l0Cand</code> being the candidate at position <code>l0CandIdx</code> and <code>l1Cand</code> being the candidate at position <code>l1CandIdx</code> in the merging candidate list <code>mergeCandList</code>: <code>l0Cand = mergeCandList[l0CandIdx]</code> <code>l1Cand = mergeCandList[l1CandIdx]</code> 3. When all of the following conditions are true: <code>mergeCandIsVspFlag[l0CandIdx] == 0</code>; <code>mergeCandIsVspFlag[l1CandIdx] == 0</code>; <code>predFlagL0l0Cand == 1</code> <code>predFlagL1l1Cand == 1</code> <code>(DiffPicOrderCnt(RefPicList0[refIdxL0l0Cand], RefPicList1[refIdxL1l1Cand]) != 0) </code> <code>(mvL0l0Cand != mvL1l1Cand)</code> the candidate <code>combCand_k</code>, with <code>k</code> equal to <code>(numCurrMergeCand - numInputMergeCand)</code> is added at the end of <code>mergeCandList</code>, i.e. <code>mergeCandList[numCurrMergeCand]</code> is set equal to <code>combCand_k</code>, and the reference indices, the prediction list utilization flags, and the motion vectors of <code>combCand_k</code>, are derived as follows and <code>numCurrMergeCand</code> is incremented by 1: <code>refIdxL0combCand_k = refIdxL0l0Cand</code>
--

[0092]

<code>refIdxL1combCand_k = refIdxL1l1Cand</code> <code>predFlagL0combCand_k = 1</code> <code>predFlagL1combCand_k = 1</code> <code>mvL0combCand_k[0] = mvL0l0Cand[0]</code> <code>mvL0combCand_k[1] = mvL0l0Cand[1]</code> <code>mvL1combCand_k[0] = mvL1l1Cand[0]</code> <code>mvL1combCand_k[1] = mvL1l1Cand[1]</code> <code>numCurrMergeCand = numCurrMergeCand + 1</code> 4. The variable <code>combIdx</code> is incremented by 1. When <code>combIdx</code> is equal to <code>(numOrigMergeCand * (numOrigMergeCand - 1))</code> or <code>numCurrMergeCand</code> is equal to <code>MaxNumMergeCand</code>, <code>combStop</code> is set equal to TRUE.

[0093]

[0094]

제안하는 방법에서는 시간적 병합 후보 유도과정을 간소화 하면서 기존과 거의 동일한 부호화 효율을 얻을 수 있다. 일 예로, 표 4에서와 같이 현재 픽처의 시점과 시간적 병합 후보 유도에 사용 되는 collocated 픽처의 시점이 동일한지 초기에 비교하여 두 픽처의 시점이 다르다면 유도과정을 종료한다.

- [0095] 도 18은 제안하는 시간적 병합 후보 유도과정 간소화 장치의 일예이다. 장치의 구동 과정은 다음과 같다.
- [0096] - 현재 PU, collocated PU, collocated 픽처를 입력으로 받아서 현재 픽처의 시점과 collocated 픽처의 시점이 동일한지 비교한다.
- [0097] - 두 픽처의 시점이 동일하다면 시간적 병합 후보의 움직임 정보를 기존 방법을 사용하여 유도한다.
- [0098] - 그렇지 않다면, 과정을 종료한다.

[0099] 표 4. 제안하는 시간적 병합 후보 유도과정 간소화 방법 일예

H.8.5.3.2.8 Derivation process for collocated motion vectors Inputs to this process are: - a variable currPb specifying the current prediction block, - a variable colPic specifying the collocated picture, - a variable colPb specifying the collocated prediction block inside the collocated picture specified by colPic, - a luma location (xColPb, yColPb) specifying the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by colPic, - a reference index refIdxLX, with X being 0 or 1. Outputs of this process are: - the motion vector prediction mvLXCol, - the availability flag availableFlagLXCol. The variable currPic specifies the current picture. The arrays predFlagLXCol[x][y], mvLXCol[x][y], and refIdxLXCol[x][y] are set equal to the corresponding arrays of the collocated picture specified by colPic, PredFlagLX[x][y], MvLX[x][y], and RefIdxLX[x][y], respectively, with X being the value of X this process is invoked for. The variables mvLXCol and availableFlagLXCol are derived as follows: If (ViewIdx != colViewIdx), both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0. - If colPb is coded in an intra prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0. - Otherwise, the motion vector mvCol, the reference index refIdxCol, and the reference list identifier listCol are derived as follows: - If predFlagL0Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol, and listCol are set equal to mvL1Col[xColPb][yColPb], refIdxL1Col[xColPb][yColPb], and L1, respectively. - Otherwise, if predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol, and listCol are set equal to mvL0Col[xColPb][yColPb], refIdxL0Col[xColPb][yColPb], and L0, respectively. - Otherwise (predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 1), the following assignments are made: - If DiffPicOrderCnt(aPic, currPic) is less than or equal to 0 for every picture aPic in every reference picture list of the current slice, mvCol, refIdxCol, and listCol are set equal to mvLXCol[xColPb][yColPb], refIdxLXCol[xColPb][yColPb] and LX, respectively. - Otherwise, mvCol, refIdxCol, and listCol are set equal to mvLNCol[xColPb][yColPb], refIdxLNCol[xColPb][yColPb], and LN, respectively, with N being the value of collocated_from_l0_flag. and mvLXCol and availableFlagLXCol are derived as follows: - The variables curlvFlag and collvFlag, specifying whether inter-view prediction is utilized for the current and collocated PU are derived as: curlvFlag = LongTermRefPic(currPic, currPb, refIdxLX, LX) collvFlag = LongTermRefPic(colPic, colPb, refIdxCol, listCol) - When MvHevcCompatibilityFlag is equal to 0, curlvFlag is not equal to collvFlag, and AltRefIdxLX is not equal to -1, the variables refIdxCol and collvFlag are modified as follows: refIdxLX = AltRefIdxLX

[0100]

```

        curlyFlag = LongTermRefPic( currPic, currPb, refIdxLX, LX )
    - The motion vector mvLXCol is modified as follows.
    - If curlyFlag is not equal to collyFlag, both components of mvLXCol are set equal to 0 and
      availableFlagLXCol is set equal to 0.
    - Otherwise, the variable availableFlagLXCol is set equal to 1, refPicListCol[ refIdxCol ] is set to be
      the picture with reference index refIdxCol in the reference picture list listCol of the slice
      containing prediction block currPb in the picture colPic, and the following applies:
      - The variables colDiff and currDiff specifying a POC or ViewId difference are derived as
        follows.
      - If curlyFlag is equal to 0 or ( ( ViewIdx != 0 ) && iv_mv_scaling_flag ) is equal to
        0, the following applies.
          colDiff = DiffPicOrderCnt( colPic, refPicListCol[ refIdxCol ] )
          currDiff = DiffPicOrderCnt( currPic, RefPicListX[ refIdxLX ] )
      - Otherwise, ( curlyFlag is equal to 1 and ( ( ViewIdx != 0 ) && iv_mv_scaling_flag )
        is equal to 1), the following applies.
          colDiff = DiffViewId( colPic, refPicListCol[ refIdxCol ] )
          currDiff = DiffViewId( currPic, RefPicListX[ refIdxLX ] )
      - If colDiff is equal to currDiff, mvLXCol is derived as follows:
          mvLXCol = mvCol
      - Otherwise, mvLXCol is derived as a scaled version of the motion vector mvCol as
        follows:
          tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td
          distScaleFactor = Clip3( -4096, 4095, ( tb * tx + 32 ) >> 6 )
          mvLXCol = Clip3( -32768, 32767, Sign( distScaleFactor * mvCol ) *
            ( ( Abs( distScaleFactor * mvCol ) + 127 ) >> 8 ) )
          where td and tb are derived as follows:
          td = Clip3( -128, 127, colDiff )
          tb2Q Equation \* AR27, CurrDiff currDH-123)

```

[0101]

[0102]

제안하는 방법을 구현하면 표 5와 같은 비교 결과를 얻을 수 있다. 실험에 사용 된 비디오 시퀀스들은 JCT-3V 표준화에서 공식적으로 사용하는 테스트용 비디오 시퀀스들이다.

[0103]

표 5. 기존 방법과 제안하는 방법의 부호화 효율 및 부호화 시간 비교 실험 결과

	시점 0	시점 1	시점 2	video PSNR / video bitrate	video PSNR / total bitrate	synth PSNR / total bitrate
Balloons	0.0%	0.0%	0.2%	0.0%	0.0%	0.0%
Kendo	0.0%	0.1%	0.0%	0.0%	0.0%	0.1%
Newspaper_CC	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%
GT_Fly	0.0%	0.0%	-0.1%	0.0%	0.0%	0.0%
Poznan_Hall2	0.0%	0.1%	0.2%	0.0%	0.0%	-0.1%
Poznan_Street	0.0%	0.0%	-0.2%	0.0%	0.0%	0.0%
Undo_Dancer	0.0%	0.0%	-0.2%	0.0%	0.0%	0.1%
1024x768	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%
1920x1088	0.0%	0.0%	-0.1%	0.0%	0.0%	0.0%
average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Shark	0.0%	-0.2%	0.1%	0.0%	0.0%	0.0%
-------	------	-------	------	------	------	------

[0104]

[0105]

비교 결과 기존 방법에 비하여 PSNR 대비 비트율(bitrate) 증가가 0.1% 미만으로 거의 동일한 부호화 효율을 보인다.

[0106]

상술한 방법은 현재 MPEG(Moving Picture Experts Group)과 VCEG(Video Coding Experts Group)에서 공동으로 표준화를 진행 중인 HEVC(High Efficiency Video Coding)를 사용할 수 있다. 따라서 상술한 방법은 도 19의 예와 같이 블록 크기 혹은 CU(Coding Unit) 깊이 혹은 TU(Transform Unit) 깊이 등에 따라 적용 범위를 달리할 수 있다. 이렇게 적용 범위를 결정하는 변수(즉, 크기 혹은 깊이 정보)는 부호화기 및 복호화기가 미리 정해진

값을 사용하도록 설정할 수도 있고, 프로파일 또는 레벨에 따라 정해진 값을 사용하도록 할 수도 있고, 부호화기가 변수 값을 비트스트림에 기재하면 복호화기는 비트스트림으로부터 이 값을 구하여 사용할 수도 있다. CU 깊이에 따라 적용 범위를 달리하는 할 때는 도 19에 예시한 바와 같이, 방식A) 주어진 깊이 이상의 깊이에만 적용하는 방식, 방식B) 주어진 깊이 이하에만 적용하는 방식, 방식C) 주어진 깊이에만 적용하는 방식이 있을 수 있다.

[0107]

모든 깊이에 대하여 본 발명의 방법들을 적용하지 않는 경우는 임의의 지시자(flag)를 사용하여 나타낼 수도 있고, CU 깊이의 최대값보다 하나 더 큰 값을 적용범위를 나타내는 CU 깊이 값으로 시그널링 함으로써 표현할 수도 있다.

[0108]

본 발명의 추가적인 특징으로, 상술한 방법의 적용 여부는 비트스트림에 포함할 수 있으며, SPS(Sequence Parameter Set), PPS(Picture Parameter Set), Slice Header Syntax에 아래와 같은 Semantics로 적용될 수 있다.

[0109]

표 6. SPS 에 적용된 일예

seq_parameter_set_rbsp() {	Descriptor
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
...	ue(v)
reuse_enabled_flag	u(1)
if(reuse_enabled_flag)	
reuse_disabled_info	ue(v)
...	

[0110]

[0111]

표 7. PPS 에 적용된 일예

pic_parameter_set_rbsp() {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag	u(1)
...	
reuse_enabled_flag	u(1)
if(reuse_enabled_flag)	
reuse_disabled_info	ue(v)
...	

[0112]

[0113] 표 8. Slice Header Syntax 에 적용된 일 예

slice_header() {	Descriptor
slice_type	ue(v)
pic_parameter_set_id	ue(v)
frame_num	u(v)
...	
reuse_enabled_flag	u(1)
if(reuse_enabled_flag)	
reuse_disabled_info	ue(v)
...	

[0114]

[0115] 표 9. Slice Header Syntax 에 적용된 또 다른 일 예

slice_header() {	Descriptor
lightweight_slice_flag	u(1)
if(!lightweight_slice_flag) {	
slice_type	ue(v)
pic_parameter_set_id	ue(v)
frame_num	u(v)
...	
}	
if(entropy_coding_mode_flag && slice_type != I)	
cabac_init_idc	ue(v)
first_slice_in_pic_flag	u(1)
...	
reuse_enabled_flag	u(1)
if(reuse_enabled_flag)	
reuse_disabled_info	ue(v)
...	

[0116]

[0117] 여기서 “reuse_enabled_flag” 는 제안하는 방법의 적용 여부를 알려준다. 여기서 제안하는 방법이 적용되었을 경우 “reuse_enabled_flag” 가 ‘1’ 이 되며, 적용되지 않았을 경우 “reuse_enabled_flag” 가 ‘0’ 이 된다. 그 반대도 가능하다.

[0118] 또한, “reuse_disabled_info” 는 제안 방법이 적용되었을 경우(혹은 “reuse_enabled_flag” 이 참일 경우) 활성화되는 syntax로써, 이는 CU의 깊이(depth) (혹은 CU의 크기(size) 혹은 매크로 블록의 크기 혹은 서브 매크로 블록의 크기 혹은 블록의 크기)에 따라 제안 방법의 적용 여부를 알려준다. 하나의 일 예로, “reuse_disabled_info” 가 ‘0’ 일 경우에는 제안 방법이 모든 블록 크기에 적용될 수 있다. 또한 “reuse_disabled_info” 가 ‘1’ 일 경우에는 제안 방법이 4x4 블록 크기보다 큰 단위에만 적용할 수 있다. 또 다른 일 예로, “reuse_disabled_info” 가 ‘2’ 일 경우에는 제안 방법이 8x8 블록 크기보다 큰 단위에만 적용할 수 있다. 혹은 그 역도 가능하다. 예를 들어 “reuse_disabled_info” 가 ‘1’ 일 경우에는 제안 방법이 4x4 블록 크기보다 작은 단위에만 적용할 수 있다. 이러한 “reuse_disabled_info” syntax의 사용 방법은 다양하게 적용할 수 있다.

[0119] 위의 방법을 이용하여 픽처(혹은 프레임) 단위로 적용여부를 결정할 수 있다. 또한 P 픽처(혹은 프레임) 에만 본 명세서의 방법이 적용될 수 있으며, B 픽처(혹은 프레임) 에만 본 명세서의 방법이 적용될 수 있다.

[0120] 위의 방법들은 3차원 비디오 코덱뿐만아니라 스케일러블 비디오 코덱에도 적용할 수 있다. 실시 일례로, 스케일러블 비디오 코덱의 기본 계층에서 사용하는 부호화/복호화 모듈을 그대로 향상 계층에 적용한 후, 추가적으로

부분 부호화/복호화 모듈을 이용하여 향상 계층을 부호화/복호화할 수 있다. 실시 일례로, 스케일러블 비디오 코덱의 기본 계층에서 사용하는 “기본 병합 움직임 리스트 구성” 모듈을 그대로 향상 계층에 적용하여 “기본 병합 움직임 후보 리스트”를 구성한 후, 추가적으로 “추가 병합 움직임 리스트 구성” 모듈을 이용하여 “기본 병합 움직임 후보 리스트”를 재설정(변경)하여 향상 계층을 위한 “확장 병합 움직임 후보 리스트”를 구성할 수 있다.

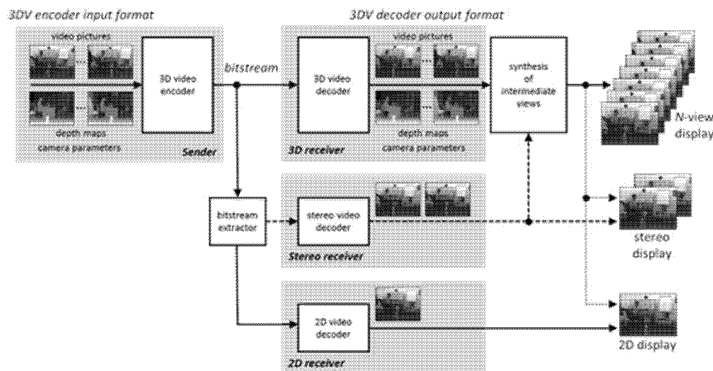
부호의 설명

[0121]

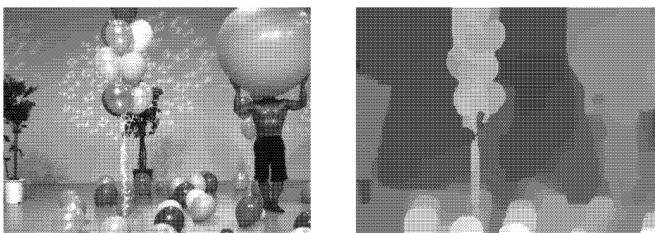
MPEG(Moving Picture Experts Group), 3DV(3D Video Coding), JCT-VC(Joint Collaborative Team on Video Coding), JCT-3V(The Joint Collaborative Team on 3D Video Coding Extension Development), HEVC(High Efficiency Video Coding), 3D-HEVC(3D High Efficiency Video Coding), HTM(3D-HEVC Test Model), SHVC(Scalable High Efficiency Video Coding), SHM(SHVC Test Model), Merge

도면

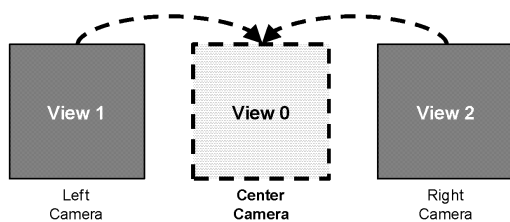
도면1



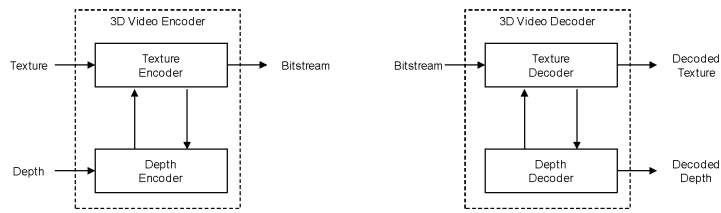
도면2



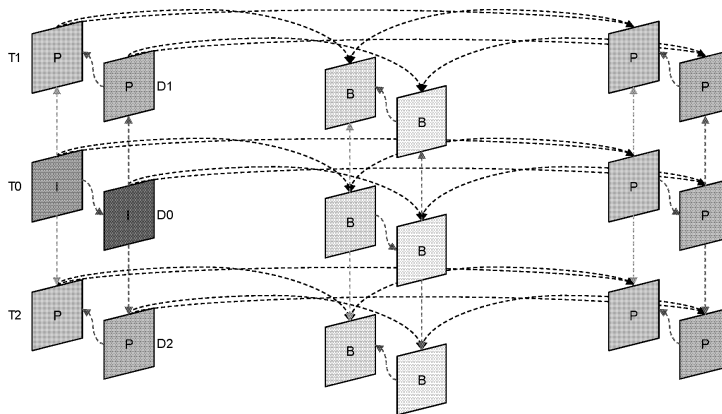
도면3



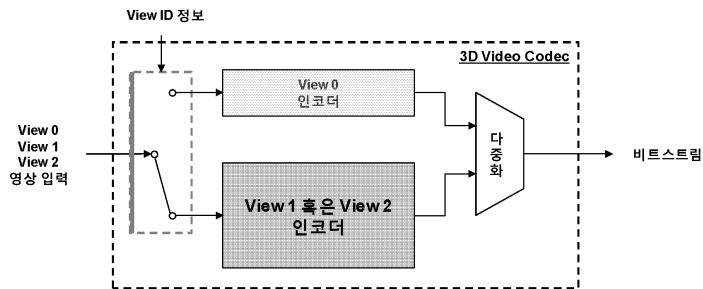
도면4



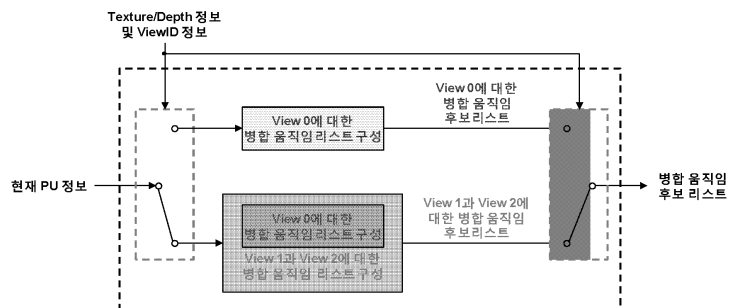
도면5



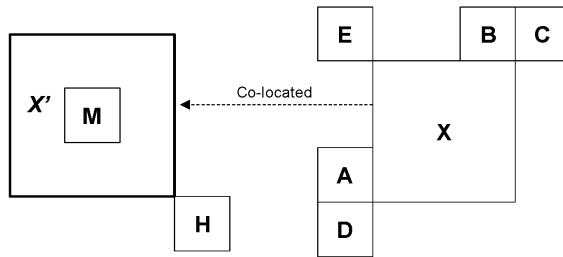
도면6



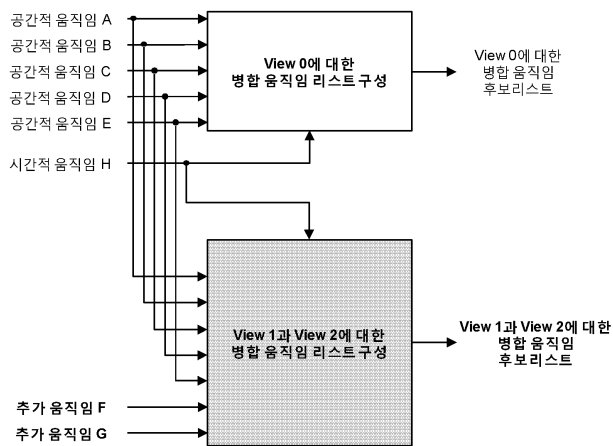
도면7



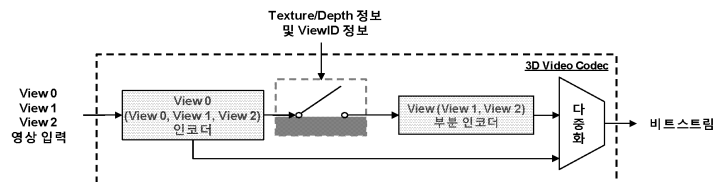
도면8



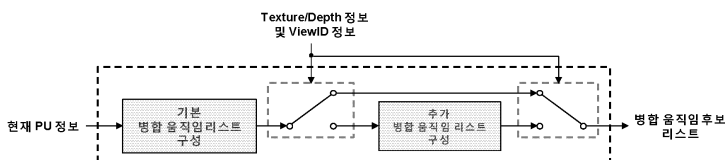
도면9



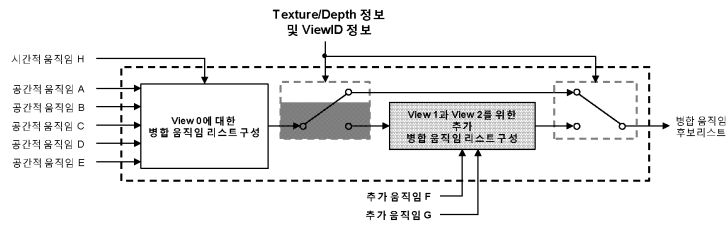
도면10



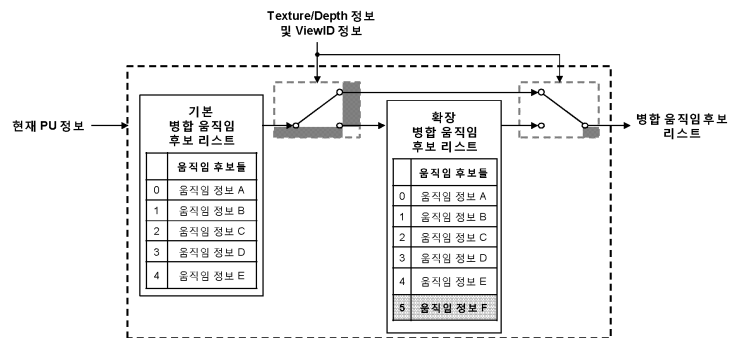
도면11



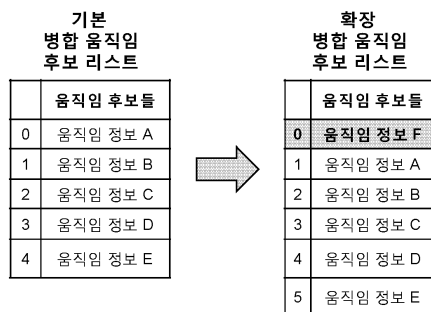
도면12



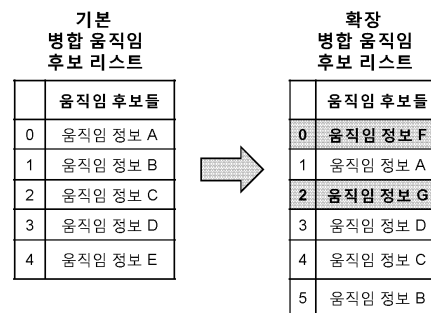
도면13



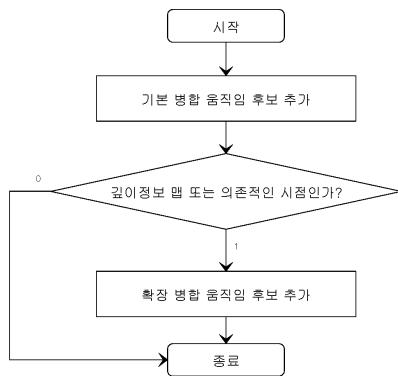
도면14



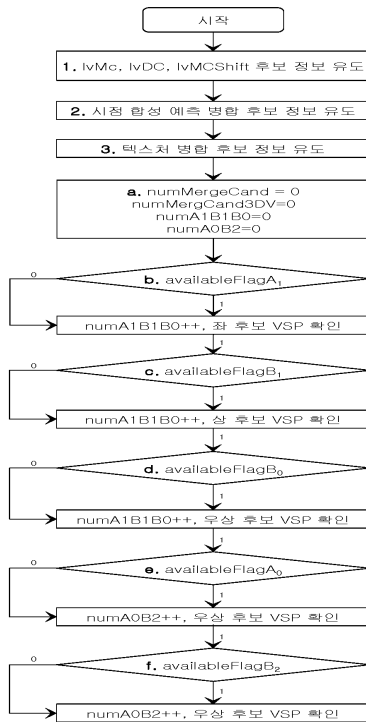
도면15



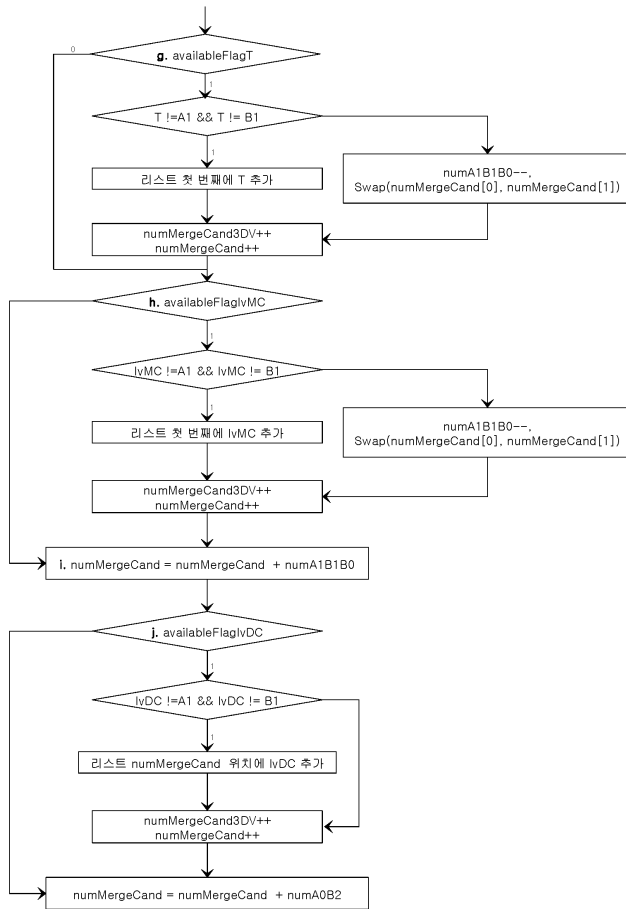
도면16



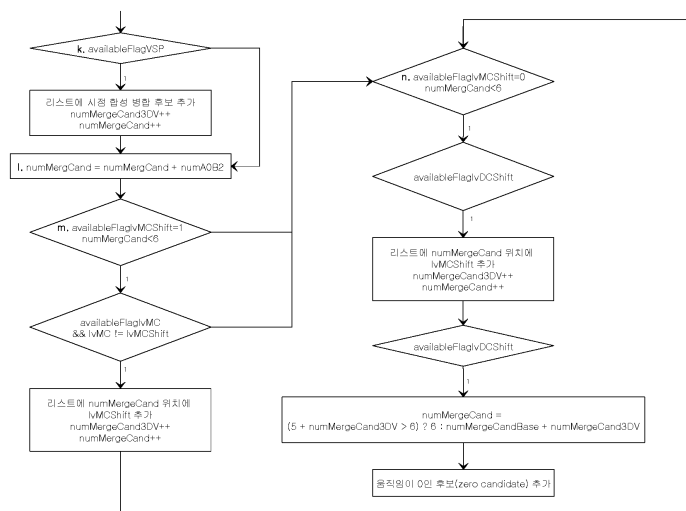
도면17a



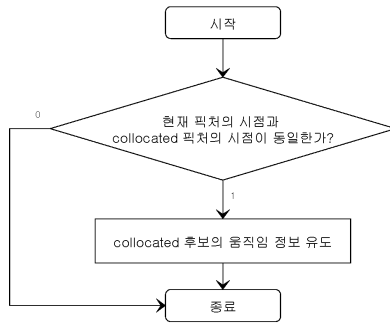
도면17b



도면17c



도면18



도면19

적용범위를 나타내는 CU(혹은 PU 혹은 TU) 깊이	방법A	방법B	방법C
0	X	O	X
1	X	O	X
2	O	O	O
3	O	X	X
4 이상	O	X	X