



(19) 대한민국특허청(KR)  
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0010249  
(43) 공개일자 2015년01월28일

(51) 국제특허분류(Int. Cl.)  
H04N 19/50 (2014.01) H04N 13/00 (2006.01)  
(21) 출원번호 10-2013-0084981  
(22) 출원일자 2013년07월18일  
심사청구일자 없음

(71) 출원인  
경희대학교 산학협력단  
경기도 용인시 기흥구 덕영대로 1732, 국제캠퍼스  
내 (서천동, 경희대학교)  
(72) 발명자  
박광훈  
경기 성남시 분당구 예원로 7, B동 302호 (분당동, 동아빌라)  
이민성  
경기 평택시 평택5로114번길 13, 105동 1502호 (비전동, 현대이화아파트)  
(74) 대리인  
(뒷면에 계속)  
에스앤아이퍼특허법인

전체 청구항 수 : 총 3 항

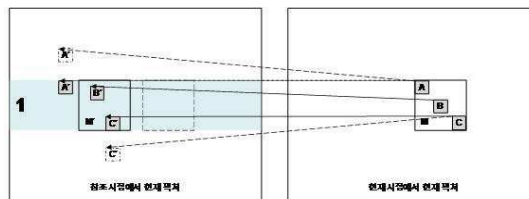
(54) 발명의 명칭 제한적인 움직임 정보 유도를 통한 메모리 효율성을 향상시키는 방법 및 장치

(57) 요약

제한적인 움직임 정보 유도를 통한 메모리 효율성을 향상시키는 방법 및 장치가 개시된다.

이 방법은 임의의 범위 내에서만 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하도록 제한하는 방법이다.

대표도 - 도12



(72) 발명자

**김경용**

경기 수원시 영통구 덕영대로 1725, 402호 (영통동)

**허영수**

경기 수원시 영통구 봉영로1492번길 20, B01호 (영통동)

**이윤진**

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

## 특허청구의 범위

### 청구항 1

임의의 범위 내에서만 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하도록 제한하는 방법.

### 청구항 2

제 1항에 있어서, “움직임 정보 (ex. DV)”와 “임의의 범위”를 입력받아, 임의의 범위로 변이 벡터의 수직 방향 성분을 클리핑하여 “변경된 DV”를 출력하는 단계; “변경된 DV”와 “참조시점 영상에서의 움직임 정보”를 입력받아, 변경된 변이 벡터를 이용하여 참조 시점 영상으로부터 움직임 정보를 유도하여 “유도된 움직임 정보”를 출력하는 단계;를 포함하는 움직임 유도 방법.

### 청구항 3

제 1항에 있어서, 수직 방향의 움직임 예측(ME) 범위를 메모리가 허용되는 범위내의 ‘임의의 값(T)’으로 제한하는 단계; “움직임 정보 (ex. DV)”와 “임의의 범위” 그리고 “임의의 값(T)를 입력받아, “임의의 범위와 임의의 값(T)”로 변이 벡터의 수직 방향 성분을 클리핑하여 “변경된 DV”를 출력하는 단계; “변경된 DV”와 “참조시점 영상에서의 움직임 정보”를 입력받아, 변경된 변이 벡터를 이용하여 참조 시점 영상으로부터 움직임 정보를 유도하여 “유도된 움직임 정보”를 출력하는 단계;를 포함하는 움직임 유도 방법.

## 명세서

### 기술 분야

[0001] 본 발명은 비디오의 부호화/복호화 방법 및 장치에 관한 것으로서, 제한적인 움직임 정보 유도를 통한 메모리 효율성을 향상시키는 방법 및 장치에 관한 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. 도 1을 참조하면, 3차원 비디오 표준은 실제 영상과 그것의 깊이정보 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  $N(N=2)$  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 맵을 생성하는 방법은 카메라를 이용하여 획득하는 방법과 실제 일반 영상(Texture image)을 이용하여 자동적으로 생성하는 방법으로 나눌 수 있다. 카메라를 이용하여 획득하는 방법의 경우, 깊이정보 카메라가 일정한 거리 내에서만 작동된다는 문제점이 있다. 반면, 실제 일반 영상을 이용하여 자동적으로 생성하는 방법의 경우, 두 일반 영상

간의 시점 차이(disparity)를 이용하여 깊이정보 맵을 생성한다. 다시 말하면, 현재 시점에서의 임의의 한 화소를 주변 시점에서의 화소들과 비교하여 매칭이 가장 잘되는 부분의 화소를 찾고, 그 화소들간의 거리를 깊이정보로 표현하는 방법이다. 실제 일반 영상을 이용하여 자동적으로 생성된 깊이정보 맵의 예로써, 도 2는 국제 표준화 기구인 MPEG의 3차원 비디오 부호화 표준에서 사용 중인 “balloons” 영상(도 2의 (a))과 그것의 깊이정보 맵(도 2의 (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 일례로 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] HEVC에서는 영상을 효율적으로 부호화하기 위해서 CU(Coding Unit: CU, 이하 ‘CU’라 함)로 부호화를 수행한다. 도 3는 영상을 부호화할 때 CTU(Coding Tree Unit; 이하 ‘CTU’라 함. 혹은 CTU(LCU)(Largest Coding Unit)라고도 할 수 있음) 내에서 CU를 분할하는 방법을 나타낸 그림이다.

[0007] 도 3와 같이 HEVC에서는 영상을 CTU(LCU) 단위로 순차적으로 분할한 후, CTU(LCU) 단위로 분할구조를 결정한다. 분할구조는 CTU(LCU) 내에서 영상을 효율적으로 부호화하기 위한 CU의 분포를 의미하는데, 이 분포는 CU를 크기가 가로 세로 절반씩 감소한 4개의 CU로 분할할지를 결정함으로써 결정할 수 있다. 분할된 CU는 같은 방식으로 가로 세로 절반씩 감소한 4개의 CU로 재귀적으로 분할할 수 있다. 이 때, CU의 분할은 미리 정의된 깊이까지 분할을 수행할 수 있는데, 깊이 정보(Depth)는 CU의 크기를 나타내는 정보로써 모든 CU에 저장되어 있다. 기본이 되는 CTU(LCU)의 깊이는 0이고, SCU의 깊이는 미리 정의된 최대 깊이이다. CTU(LCU)로부터 가로 세로 절반으로 분할을 수행할 때마다 CU의 깊이가 1씩 증가한다. 각각의 깊이 별로, 분할을 수행하지 않는 CU의 경우에는  $2N \times 2N$ 의 크기로, 분할을 수행할 경우에는  $N \times N$  크기의 CU 4개로 분할된다. N의 크기는 깊이가 1씩 증가할 때마다 절반으로 감소한다. 그림 5에서는 최소 깊이가 0인 CTU(LCU)의 크기가  $64 \times 64$  화소이고, 최대 깊이가 3인 SCU(Smallest Coding Unit)의 크기가  $8 \times 8$  화소인 예이다.  $64 \times 64$  화소 CU(CTU(LCU))는 깊이 ‘0’으로  $32 \times 32$  화소 CU는 깊이 ‘1’로  $16 \times 16$  화소 CU는 깊이 ‘2’로  $8 \times 8$  CU(SCU)는 깊이 ‘3’으로 표현된다. 또한 특정 CU를 분할할지에 대한 정보는 CU마다 1비트 정보인 분할 정보를 통해 표현한다. 이 분할 정보는 SCU를 제외한 모든 CU에 포함되어 있으며, CU를 분할하지 않을 경우에는 분할 정보에 ‘0’을 저장하고, 분할할 경우에는 ‘1’을 저장한다.

[0008] 이러한 CU는 부호화 단위로써, 하나의 CU단위로 부호화 모드를 가질 수 있다. 즉, CU 각각은 화면 내 부호화(MODE\_INTRA 혹은 INTRA라고 할 수도 있음) 모드 혹은 화면 간 부호화(MODE\_INTER 혹은 INTER라고 할 수도 있음) 모드로 나누어 질 수 있다. 화면 간 부호화(MODE\_INTER) 모드에는 MODE\_INTER 모드와 MODE\_SKIP(SKIP이라고 할 수도 있음) 모드로 구분될 수 있다.

[0009] PU(Prediction Unit)는 예측의 단위로서, 도 4과 같이 하나의 CU는 여러 개의 PU(Prediction Unit: PU, 이하 ‘PU’라 함)들로 분할되어 예측이 수행될 수 있다. 하나의 CU의 부호화 모드가 INTRA 라면, 그 CU내의 PU들은 모두 INTRA 로 부호화 되고, 하나의 CU의 부호화 모드가 INTER 라면, 그 CU내의 PU들은 모두 INTER 로 부호화된다. 실시 일례로, 하나의 CU가 INTRA 일 경우, PU의 분할 구조는  $PU\ 2N \times 2N$ 과  $PU\ N \times N$ 만 가질 수 있다. 실시 일례로, 하나의 CU가 INTER라면 PU는 도 4의 모든 분할 구조를 가질 수 있다.

[0010] 실제 영상과 그것의 깊이정보맵은 카메라 하나뿐만 아니라 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 5는 카메라 3대에서 획득한 영상들에 대한 시점간 예측 구조의 일례를 나타낸다.

[0011] 도 5에서, 시점 1(View 1)은 시점 0(View 0)을 기준으로 왼쪽에 위치한 카메라에서 획득한 영상이고, 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 시

점 영상은 시점간 예측을 수행하여야 하므로, 시점간 예측 과정이 포함된 3차원 비디오 코덱을 사용하여 부호화할 수 있다.

[0012] 또한, 시점 1(View 1)과 시점 2(View 2)의 부호화 효율을 증가시키기 위하여 깊이정보 맵을 이용하여 부호화할 수 있다. 일례로, 실제 영상과 그것의 깊이정보 맵을 부호화할 때, 서로 독립적으로 부호화/복호화할 수 있다. 또한 실제 영상과 깊이정보 맵을 부호화할 때, 도 6과 같이 서로 의존적으로 부호화/복호화할 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.

[0013] 일 실시 예로, 카메라 3대에서 획득한 실제 영상과 그것의 깊이정보맵을 부호화하기 위한 부호화 예측 구조를 도 7에 나타내었다. 도 7에서 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 7에서 화살표는 예측 방향을 나타낸다. 즉, 실제 영상과 그것의 깊이정보맵은 서로 의존적으로 부호화/복호화된다.

[0014] 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.

[0015] 영상 부호화/복호화 시 움직임 정보의 부호화 방법 중 하나로 병합 움직임(merge) 방법을 이용한다. 이때 움직임 정보란, 움직임 벡터, 참조영상에 대한 인덱스 그리고 예측 방향(단방향, 양방향, 등) 중에서 적어도 하나를 포함하는 정보다. 예측 방향은 참조픽처목록(Reference Picture List; RefPicList)의 사용에 따라 크게 단방향 예측과 양방향 예측으로 나누어 질 수 있다. 단방향 예측에는 순방향 참조픽처목록 (LIST 0)을 사용한 순방향 예측(Pred\_L0; Prediction L0)과 역방향 참조픽처목록 (LIST 1)을 사용한 역방향 예측(Pred\_L1; Prediction L1)으로 구분된다. 또한, 양방향 예측(Pred\_BI; Prediction BI)은 순방향 참조픽처목록 (LIST 0)과 역방향 참조픽처목록 (LIST 1)을 모두 사용하며, 순방향 예측과 역방향 예측이 모두 존재하는 것을 말할 수 있다. 또한, 순방향 참조픽처목록 (LIST 0)을 역방향 참조픽처목록 (LIST 1)에 복사하여 순방향 예측이 두개 존재하는 경우도 양방향 예측에 포함할 수 있다. 이러한 예측 방향 여부를 predFlagL0, predFlagL1를 사용하여 정의할 수 있다. 실시 일례로, 단방향 예측이고 순방향 예측일 경우 predFlagL0는 '1'이 되고 predFlagL1은 '0'이 될 수 있다. 또한, 단방향 예측이고 역방향 예측일 경우 predFlagL0는 '0'이 되고 predFlagL1은 '1'이 될 수 있다. 또한, 양방향 예측일 경우 predFlagL0는 '1'이 되고 predFlagL1은 '1'이 될 수 있다.

[0016] 병합 움직임은 부호화 유닛(Coding Unit: CU, 이하 'CU' 라 함) 단위의 병합 움직임과 예측 유닛(Prediction Unit: PU, 이하 'PU' 라 함) 단위의 병합 움직임이 가능하다. CU또는 PU(이하, 설명의 편의를 위해 '블록' 이라 함) 단위로 병합 움직임을 수행하는 경우에는, 블록 파티션(partition)별로 병합 움직임을 수행할지에 관한 정보와 현재 블록에 인접한 주변 블록(현재 블록의 좌측 인접 블록, 현재 블록의 상측 인접 블록, 현재 블록의 시간적(temporal) 인접 블록 등) 중 어떤 블록과 병합 움직임을 할 것인지에 대한 정보를 전송할 필요가 있다.

[0017] 병합 움직임(merge) 후보 리스트(List)은 움직임 정보들이 저장된 리스트를 나타내며, 병합 움직임이 수행되기 전에 생성된다. 여기서 병합 움직임 후보 리스트에 저장되는 움직임 정보는 현재 블록에 인접한 주변 블록의 움직임 정보이거나 참조 영상에서 현재 블록에 대응되는(collocated) 블록의 움직임 정보일 수 있다. 또한 병합 움직임 후보 리스트에 저장되는 움직임 정보는 이미 병합 움직임 후보 리스트에 존재하는 움직임 정보들을 조합하여 만든 새로운 움직임 정보일 수 있다.

[0018] 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. 또한, 각 주변 블록은 서로 동일한 움직임 정보를 가지는지 확인하여 동일한 움직임 정보일 경우, 해당 주변 블록의 움직임 정보는 병합 움직임 후보 리스트에 포함되지 않는다. 실시 일례로, 도 8에서 X 블록에 대한 병합 움직임 후보 리스트를 생성할 때, 주변 블록 A가 사용가능하여 병합 움직임 후보 리스트에 포함된 후, 주변 블록 B는 주변 블록 A와 동일한 움직임 정보가 아닐 경우에만, 병합 움직임 후보 리스트에 포함될 수 있다. 동일한 방법으로 주변 블록 C는 주변 블록 B와 동일한 움직임 정보가 아닐 경우에만 병합 움직임 후보 리스트에 포함될 수 있다. 동일한 방법으로 주변

블록 D와 주변 블록 E에 적용될 수 있다. 여기서 동일한 움직임 정보라는 것은 움직임 벡터가 동일하고 동일한 참조픽처를 사용하고 동일한 예측 방향(단방향(정방향, 역방향), 양방향)을 사용함을 의미할 수 있다. 마지막으로 도 8에서 X 블록에 대한 병합 움직임 후보 리스트는 소정의 순서, 예컨대, A→B→C→D→E→H(혹은 M) 블록 순서로 리스트에 추가될 수 있다.

## 발명의 내용

### 해결하려는 과제

- [0019] 3차원 비디오 부호화에서는 움직임 정보를 효율적으로 부호화하기 위하여, 인접한 시점의 움직임 정보를 이용할 수 있다. 도 9는 인접한 시점의 움직임 정보를 이용하여 현재 블록의 움직임 정보를 유도하는 방법의 일예를 나타낸다.
- [0020] 도 9에서 현재 블록(현재 위치 X에 대한 블록)에 대한 움직임 정보를 인접한 시점에서 유도하기 위하여, 인접한 시점에서 현재 블록의 움직임 정보와 가장 유사할 것 같은 대상 블록(참조 위치 XR에 대한 블록)을 찾는다. 현재 시점에서의 현재 픽처와 인접한 참조 시점에서 현재 픽처는 카메라 거리만의 차이가 있으므로, 인접한 시점에서 현재 블록과 가장 유사한 대상 블록은 변이 벡터(DV; Disparity Vector)를 통해 그 위치를 유도할 수 있다.
- [0021] 변이 벡터를 유도하기 위해서 현재 블록의 주변 블록들을 이용하는 방법과 현재 블록과 연관된 깊이정보 맵을 이용하는 방법을 각각 이용할 수 있으며, 혼합하여 이용할 수 있다. 먼저, 시점간 예측 블록의 움직임 정보는 변이 벡터를 의미하므로, 현재 블록의 주변 블록들에서 시점간 예측을 수행한 블록을 찾아서 그 블록의 움직임 정보를 변이 벡터로 사용할 수 있다. 다음으로 깊이정보 맵이 두 시점간의 시점간 차이를 통해 생성되었으므로, 현재 블록과 연관된 깊이정보 맵 블록의 깊이정보 값을 변이 벡터로 변환하여 사용할 수 있다.
- [0022] 이러한 찾은 변이 벡터를 이용하여 참조 시점의 현재 픽처에서 현재 블록의 현재 위치 X와 연관된 참조 위치 XR을 포함하는 대상 블록을 찾은 다음, 그 대상 블록의 움직임 정보를 현재 블록의 움직임 정보로 사용할 수 있다.
- [0023] 아래의 내용은 상술한 방법에 따른 움직임 정보 유도 과정을 3D-HEVC Test Model 3를 기반으로 하여 설명한 것이다.
- [0024] 3D-HEVC Test Model 3에서 사용되는 용어 및 계산식
- [0025] - PredMode; 현재 PU(Prediction Unit) 블록의 부호화 모드를 나타내며, MODE\_SKIP, MODE\_INTRA, MODE\_INTER 가 존재한다.
- [0026] - RefPicListX; 참조픽처리스트로서, 여기서 X는 '0' 또는 '1'이 될 수 있다. 일례로, X가 0이면(즉, RefPicList0 이라면), L0 참조픽처 리스트를 사용한다.
- [0027] - PicWidthInSamplesL ; 현재 픽처에서 너비
- [0028] - PicHeightInSamplesL ; 현재 픽처에서 높이
- [0029] - CtbSizeY; 현재 CTU(혹은 LCU)의 높이
- [0030] [수학식 1]
- [0031] 
$$\text{Clip3}(x, y, z) = \begin{cases} x & ; z < x \\ y & ; z > y \\ z & ; \text{otherwise} \end{cases}$$
- [0032] 시간적인 시점간 움직임 벡터를 유도하는 과정

- [0033] 이 단계에서 입력은 아래와 같다
- [0034] - 현재 PU 블록의 좌상단 위치; (xP, yP)
- [0035] - 현재 PU 블록의 너비와 높이를 가르키는 변수; (nPSW, nPSH)
- [0036] - 참조 픽처 리스트 구분자; X (일예로, X가 0이면, L0 참조픽처 리스트를 사용한다.)
- [0037] - 참조 시점 인덱스; refViewIdx
- [0038] - 변이 벡터; mvDisp
- [0039] - Merge\_flag; 현재 PU블록이 merge 모드로 부호화되었는지를 나타낸다. 현재 PU블록이 merge 모드로 부호화되었다면, Merge\_flag가 '1'로 설정하며, 그 반대로도 할 수 있다.
- [0040] - RefPicListX에서 참조 픽처를 지시하는 인덱스; refIdxLX (여기서 X는 '0' 또는 '1'이 될수 있다.)
- [0041] 이 단계에서 출력은 아래와 같다.
- [0042] - 시간적인 시점간 움직임 벡터가 유효한지 여부를 나타내는 플래그; availableFlagLXInterView
- [0043] - 시간적인 시점간 움직임 벡터 후보; mvLXInterView (만일, availableFlagLXInterView가 참일 경우) (여기서 X는 '0' 또는 '1'이 될수 있다.)
- [0044] - RefPicListX에서 참조 픽처를 지시하는 인덱스; refIdxLX
- [0045] availableFlagLXInterView 를 '0'으로 초기화한다.
- [0046] mvL0InterView와 mvL1InterView를 '0'으로 초기화한다.
- [0047] 변이 벡터를 이용하여, 참조 시점에서 현재 블록에 대한 참조 블록의 위치를 아래와 같이 유도한다.
- [0048] [수학식 2]
- $$xRef = Clip3(0, PicWidthInSamples_L - 1, xP + ((nPSW - 1) >> 1) + ((mvDisp[0] + 2) >> 2))$$
- $$yRef = Clip3(0, PicHeightInSamples_L - 1, yP + ((nPSH - 1) >> 1) + ((mvDisp[1] + 2) >> 2))$$
- [0049]
- [0050] refViewIdx 값으로 설정된 ViewIdx에 대한 참조 시점 영상에서, 위에서 유도된 (xRef, yRef) 위치를 포함하는 부호화 코딩 블록을 refCU 라고 하자.
- [0051] refCU의 PredMode가 MODE\_SKIP이거나 혹은 MODE\_INTER일 경우, 아래의 과정을 수행한다. (여기서 변수 Y는 X에서 (1-X)의 범위를 갖는다.)
- [0052] 1. refPredFlagLY에 refPU의 refIdxLX를 설정한다.
- [0053] 2. refMvLY에 refPU의 mvLY를 설정한다.
- [0054] 3. refRefPicListLY에 refPU의 RefPicListLY를 설정한다.
- [0055] 4. refPredFlagLY가 '참'이면, 다음의 과정을 i 가 (mergeFlag ? 0 : refIdxLX) 부터 (mergeFlag ? num\_ref\_idx\_lX\_active\_minus1 : refIdxLX) 까지 수행한다.
- [0056] A. availableFlagLXInterView 가 '0'이고 refRefPicListLY[refRefIdxLY]의 POC(Picture Order Count)가

RefPicListLX[i]의 POC와 같으면, availableFlagLXInterView를 '1'로 설정하고 다음을 수행한다.

i. mvLXInterView 는 다음과 같이 유도된다.

mvLXInterView[0] = refMvLY[0]

mvLXInterView[1] = refMvLY[1]

ii. mergeFlag가 '1'이면, refIdxLX는 다음과 같이 유도된다.

refIdxLX = i

o 수직 방향 유도에서의 문제점

수평 방향으로 설치된 카메라들은 그 위치와 방향에 오류가 있을 수 있다. 이러한 비 정렬된 다시점 영상에 대한 오류를 최소화하기 위하여, 다시점 영상들에 수직시차보정 즉, 편위수정(rectification)을 수행한다. 편위수정(rectification)된 다시점 영상이라 하더라도, 오차가 있을 수 있다. 또한, 카메라 내에서 수직 방향의 움직임을 가진 객체들이 존재할 수 있으며, 이는 카메라의 촬영 시간 오차로 인해 시점 간의 수직 방향에서의 움직임 차이가 생길 수 있다. 따라서, 도 9의 과정에서 주변 블록으로부터 유도된 변이 벡터(DV; Disparity Vector)는 수평 방향 성분과 수직 방향 성분을 모두 가질 수 있다.

도 10은 현재 LCU (혹은 CTU) M의 임의의 블록 A, B, C에 대하여, 주변 블록으로부터 유도된 변이 벡터(DV)를 이용하여 찾은 참조 블록들의 일례를 나타낸다.

도 10에서 도시된 것과 같이, 주변 블록으로부터 유도된 변이 벡터(DV)의 수직 방향 성분으로 인해, 참조 블록이 원래 위치보다 위 아래로 이동한 것을 확인할 수 있다. 이때, 참조 시점의 참조 블록들에 대한 움직임 정보를 빠르고 쉽게 접근하기 위해서는 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)가 내부 메모리상에 존재하여야 한다. 그렇지 않고 참조 블록들의 움직임 정보가 외부 메모리에 존재할 경우, 필요할 때마다 외부 메모리에서 내부 메모리로 읽어와야 하므로, 메모리 대역폭을 증가시키는 원인이 된다.

도 11은 현재 LCU (혹은 CTU) M의 임의의 블록 A, B, C에 대하여, 부호화 및 복호화하는 단계에서 움직임 정보를 유도하기 위해 요구되는 메모리 사용량의 일례를 나타낸다.

도 11을 참고하면, 현재 LCU (혹은 CTU) M의 임의의 블록 A, B, C에 대한 움직임 정보를 유도하기 위해서는 현재 블록의 수평 위치에 대한 움직임 정보(1번 부분)뿐만아니라 위쪽 블록(2번 부분)과 아래쪽 블록(3번 부분)의 움직임 정보들이 필요하다. 즉, 최소 3개 LCU에 대한 참조 블록들의 움직임 정보가 필요하다. 또한, 현재 블록에 대한 부호화 및 복호화 속도를 감소시키기 위해서, 이러한 참조 블록들의 움직임 정보는 내부 메모리 상에 존재하여야 한다.

하지만, 참조 블록들의 움직임 정보를 내부 메모리에 가지고 있을 경우, 제한된 내부 메모리의 용량 때문에 한계가 따른다. 더구나, 현재 부호화하고자 하는 영상의 크기가 FHD (1920x1088) 이상의 크기일 경우, 내부 메모리에 저장되는 참조 블록들의 움직임 정보의 양이 상당히 많다. 따라서, 저장해야할 참조 블록들의 움직임 정보를 줄이기 위한 방안이 필요하다.

### 과제의 해결 수단

본 발명의 특징은 임의의 범위 내에서만 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하도록 제한하는 방법을 제공한다.

### 발명의 효과

임의의 범위 내에서만 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하도록 제한하는 방법을 통해, 메모리 효율성을 향상시킨다.

### 도면의 간단한 설명

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. CTU(CTU(LCU))를 CU단위로 분할하는 방법의 일 예
- 도 4. PU의 분할 구조의 일 예
- 도 5. 3D Video Codec에서 시점 간 예측 구조의 일 예
- 도 6. 3D Video Encoder/Decoder의 실시 일 예
- 도 7. 3D Video Codec의 예측 구조의 일 예
- 도 8. 병합 움직임 후보 리스트로 사용되는 현재 블록의 주변 블록들의 일 예
- 도 9. 인접한 시점의 움직임 정보를 이용하여 현재 블록의 움직임 정보를 유도하는 방법의 일 예
- 도 10. 주변 블록으로부터 유도된 변이 벡터(DV)를 이용하여 찾은 참조 블록들의 일 예
- 도 11. 움직임 정보를 유도하기 위해 요구되는 메모리 사용량의 일 예
- 도 12. 제안 방법의 개념도의 일 예
- 도 13. 제안 방법의 기본 구조도의 일 예

### 발명을 실시하기 위한 구체적인 내용

- [0072] 본 발명에서는 내부 메모리를 효율적으로 사용하기 위하여, 임의의 범위 내에서만 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하도록 제한하는 방법을 제안한다.
- [0073] 도 12는 제안 방법의 개념도의 일 예를 나타낸다.
- [0074] 도 12에서 현재 LCU (혹은 CTU) M의 임의의 블록 A와 C의 참조 블록(A'와 C')은 임의의 범위(1번 부분, 일 예로 1개 LCU 크기)내에 존재하지 않는다. 따라서, 임의의 범위내에 존재하도록 하기 위하여, 참조 블록(A'와 C')을 유도하기 위해 변이 벡터(DV)의 수직 방향 성분을 임의의 범위 내로 클리핑(clipping)을 수행한다. 그 후 클리핑된 변이 벡터(DV)를 이용하여 현재 블록에 대한 참조 블록을 유도한다.
- [0075] 도 13은 제안 방법의 기본 구조도의 일 예를 나타낸다.
- [0076] 도 13의 “클리핑 과정”에서는 현재 블록의 주변 블록들로부터 유도된 변이 벡터(도 13에서는 “움직임 정보(ex. DV)”)와 “임의의 범위”를 입력받아, 임의의 범위로 변이 벡터의 수직 방향 성분을 클리핑하여 “변경된 DV”를 출력한다. 다음으로 “움직임 정보 유도” 과정에서는 “변경된 DV”와 “참조시점 영상에서의 움직임 정보”를 입력받아, 변경된 변이 벡터를 이용하여 참조 시점 영상으로부터 움직임 정보를 유도하여 “유도된 움직임 정보”를 출력한다.
- [0077] 아래의 내용은 상술한 제안 방법에 따른 움직임 정보 유도 과정을 3D-HEVC Test Model 3를 기반으로 하여 설명한 것이다.
- [0078] 3D-HEVC Test Model 3에서 사용되는 용어 및 계산식
- [0079] - PredMode; 현재 PU(Prediction Unit) 블록의 부호화 모드를 나타내며, MODE\_SKIP, MODE\_INTRA, MODE\_INTER 가 존재한다.
- [0080] - RefPicListX; 참조픽처리스트로서, 여기서 X는 '0' 또는 '1'이 될 수 있다. 일 예로, X가 0이면(즉, RefPicList0 이라면), L0 참조픽처 리스트를 사용한다.
- [0081] - PicWidthInSamplesL ; 현재 픽처에서 너비
- [0082] - PicHeightInSamplesL ; 현재 픽처에서 높이
- [0083] - CtbSizeY; 현재 CTU(혹은 LCU)의 높이
- [0084] Clip3(x,y,z)는 수학식 1과 같다.

- [0085] 시간적인 시점간 움직임 벡터를 유도하는 과정
- [0086] 이 단계에서 입력은 아래와 같다
- [0087] - 현재 PU 블록의 좌상단 위치; (xP, yP)
- [0088] - 현재 PU 블록의 너비와 높이를 가르키는 변수; (nPSW, nPSH)
- [0089] - 참조 픽처 리스트 구분자; X (일예로, X가 0이면, L0 참조픽처 리스트를 사용한다.)
- [0090] - 참조 시점 인덱스; refViewIdx
- [0091] - 변이 벡터; mvDisp
- [0092] - Merge\_flag; 현재 PU블록이 merge 모드로 부호화되었는지를 나타낸다. 현재 PU블록이 merge 모드로 부호화되었다면, Merge\_flag가 '1' 로 설정하며, 그 반대로도 할 수 있다.
- [0093] - RefPicListX에서 참조 픽처를 지시하는 인덱스; refIdxLX (여기서 X는 '0' 또는 '1'이 될수 있다.)
- [0094] 이 단계에서 출력은 아래와 같다.
- [0095] - 시간적인 시점간 움직임 벡터가 유효한지 여부를 나타내는 플래그; availableFlagLXInterView
- [0096] - 시간적인 시점간 움직임 벡터 후보; mvLXInterView (만일, availableFlagLXInterView가 참일 경우) (여기서 X는 '0' 또는 '1'이 될수 있다.)
- [0097] - RefPicListX에서 참조 픽처를 지시하는 인덱스; refIdxLX
- [0098] availableFlagLXInterView 를 '0'으로 초기화한다.
- [0099] mvL0InterView와 mvL1InterView을 '0'으로 초기화한다.
- [0100] 현재 부호화하는 CTU의 좌상단 위치(xCtb, yCtb)를 유도한다.
- [0101] 변이 벡터를 이용하여, 참조 시점에서 현재 블록에 대한 참조 블록의 위치를 아래와 같이 유도한다.
- [0102] [수학식 3]
- $$xRef = Clip3( 0, PicWidthInSamples_L - 1, xP + ( (nPSW - 1) >> 1 ) + ( ( mvDisp [0] + 2 ) >> 2 ) )$$
- $$yRef = Clip3( yCtb, yCtb + CtbSizeY - 1, xP + ( (nPSH - 1) >> 1 ) + ( ( mvDisp [1] + 2 ) >> 2 ) )$$
- [0103]
- [0104] refViewIdx 값으로 설정된 ViewIdx에 대한 참조 시점 영상에서, 위에서 유도된 (xRef, yRef) 위치를 포함하는 부호화 코딩 블록을 refCU 라고 하자.
- [0105] refCU의 PredMode가 MODE\_SKIP이거나 혹은 MODE\_INTER일 경우, 아래의 과정을 수행한다. (여기서 변수 Y는 X에서 (1-X)의 범위를 갖는다.)
- [0106] 1. refPredFlagLY에 refPU의 refIdxLX를 설정한다.
- [0107] 2. refMvLY에 refPU의 mvLY를 설정한다.
- [0108] 3. refRefPicListLY에 refPU의 RefPicListLY를 설정한다.
- [0109] 4. refPredFlagLY가 '참'이면, 다음의 과정을 i 가 (mergeFlag ? 0 : refIdxLX) 부터 (mergeFlag ? num\_ref\_idx\_lx\_active\_minus1 : refIdxLX) 까지 수행한다.
- [0110] A. availableFlagLXInterView 가 '0'이고 refRefPicListLY[refRefIdxLY]의 POC(Picture Order Count)가 RefPicListLX[i]의 POC와 같으면, availableFlagLXInterView를 '1'로 설정하고 다음을 수행한다.

i. mvLXInterView 는 다음과 같이 유도된다.

mvLXInterView[0] = refMvLY[0]

mvLXInterView[1] = refMvLY[1]

ii. mergeFlag가 '1'이면, refIdxLX는 다음과 같이 유도된다.

refIdxLX = i

본 발명에서 제안하는 방법의 우수성을 검증하기 위해, 3D-HEVC의 검증모델인 HTM7.0에 실제 구현을 하여 실험을 수행하였다. 실험 영상은 JCT3V의 실험 영상들을 사용하였다.

[표 1] 본 발명에서 제안하는 방법의 실험 결과

|                | video 0     | video 1       | video 2       | video only    |
|----------------|-------------|---------------|---------------|---------------|
| Balloons       | 0.0%        | -0.06%        | -0.02%        | -0.01%        |
| Kendo          | 0.0%        | 0.01%         | -0.08%        | -0.01%        |
| Newspapercc    | 0.0%        | -0.02%        | 0.00%         | 0.00%         |
| GhostTownFly   | 0.0%        | 0.02%         | 0.02%         | 0.00%         |
| PoznanHall2    | 0.0%        | -0.16%        | -0.10%        | -0.06%        |
| PoznanStreet   | 0.0%        | -0.01%        | 0.00%         | -0.01%        |
| UndoDancer     | 0.0%        | -0.11%        | -0.15%        | -0.03%        |
| 1024x768       | 0.0%        | -0.02%        | -0.03%        | -0.01%        |
| 1920x1088      | 0.0%        | -0.06%        | -0.06%        | -0.02%        |
| <b>average</b> | <b>0.0%</b> | <b>-0.05%</b> | <b>-0.05%</b> | <b>-0.02%</b> |

표 1은 본 발명에서 제안하는 방법과 기존 방법을 비교한 것이다. 본 발명에서 제안하는 방법은 기존 방법에 비해, 최소 1/3의 메모리를 감소시키고 최대 1/(전체 픽처)의 메모리를 감소시키면서도 성능에는 거의 영향이 없음을 확인할 수 있었다.

또 다른 방법으로, 현재 블록의 움직임 정보를 유도하기 위해, 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하는 데 사용되는 변이 벡터(DV)에서 수직 방향 성분을 사용하지 않고, 수평방향 성분만을 사용할 수 있다. 실시 일례로, 변이 벡터(DV)에서 수직 방향 성분에 '0'을 대입하여, 참조 시점의 참조 블록들에 대한 움직임 정보(움직임 벡터, 참조픽처 번호, 예측 방향 정보)를 유도하는 데 사용할 수 있다. 수평 방향으로의 이동만 존재하므로 메모리 효율성을 증가시킬 수 있다.

실시 일례로 제안 방법은 다음의 식을 통해 참조 시점의 참조 블록의 위치(xRef, yRef)를 계산할 수 있다.

[수학식 4]

$$xRef = Clip3(0, PicWidthInSamples_L - 1, xP + ((nPSW - 1) >> 1) + ((mvDisp[0] + 2) >> 2))$$

$$yRef = Clip3(0, PicHeightInSamples_L - 1, xP + ((nPSH - 1) >> 1) + 0)$$

또 다른 방법으로, 시점간 예측을 위해 사용되는 움직임 예측(ME; Motion Estimation)의 수직 방향의 범위를 제

한함으로써 해결할 수 있다. 실시 일례로, 수직 방향의 움직임 예측(ME) 범위를 '0'으로 제한할 수 있으며, 이 경우 변이 벡터의 수직 방향 성분이 자동적으로 '0'으로 설정되게 된다.

움직임 예측(Motion Estimation)의 수직 방향의 범위를 '0'으로 제한하는 방법을 검증하기 위해, 3D-HEVC의 검증모델인 HTM7.0에 실제 구현을 하여 실험을 수행하였다. 실험 영상은 JCT3V의 실험 영상들을 사용하였다.

[표 2] 움직임 예측의 수직 방향의 범위를 제한하는 방법의 실험 결과

|                | video 0     | video 1     | video 2     | video only  |
|----------------|-------------|-------------|-------------|-------------|
| Balloons       | 0.0%        | 0.2%        | 0.0%        | 0.1%        |
| Kendo          | 0.0%        | -0.1%       | -0.3%       | -0.1%       |
| Newspapercc    | 0.0%        | 0.0%        | 0.5%        | 0.1%        |
| GhostTownFly   | 0.0%        | 0.9%        | 0.6%        | 0.2%        |
| PoznanHall2    | 0.0%        | 2.3%        | 1.6%        | 1.0%        |
| PoznanStreet   | 0.0%        | 1.4%        | 1.7%        | 0.5%        |
| UndoDancer     | 0.0%        | 0.8%        | 0.7%        | 0.3%        |
| 1024x768       | 0.0%        | 0.0%        | 0.1%        | 0.0%        |
| 1920x1088      | 0.0%        | 1.3%        | 1.1%        | 0.5%        |
| <b>average</b> | <b>0.0%</b> | <b>0.8%</b> | <b>0.7%</b> | <b>0.3%</b> |

표 2는 본 발명에서 움직임 예측(Motion Estimation)의 수직 방향의 범위를 제한하는 방법과 기존 방법을 비교한 것이다. 움직임 예측(Motion Estimation)의 수직 방향의 범위를 제한하는 방법은 기존 방법에 비해, 메모리를 크게 감소시키는 장점이 있는 반면에, 비트량이 평균적으로 0.3% 증가되었고 특정 영상에서 2.3%까지 증가되어 부호화 성능을 크게 감소시키는 단점이 있다는 것을 확인할 수 있었다.

또 다른 실시 일례로, 수직 방향의 움직임 예측(ME) 범위를 메모리가 허용되는 범위내의 '임의의 값들(A, B)'으로 제한할 수 있다. 변이 벡터의 수직 방향 성분을 메모리가 허용되는 범위 이내로 수렴시키기 위해서, 실시 일례로 다음의 식을 통해 참조 시점의 참조 블록의 위치(xRef, yRef)를 계산할 수 있다. 여기서 A, B는 임의의 값을 나타낸다. 일례로, A는 "CtbSizeY"가 될 수 있으며, B는 "2\*CtbSizeY - 1"가 될 수 있다. 여기서, CtbSizeY는 현재 CTU(혹은 LCU)의 높이를 나타낸다.

[수학식 5]

$$xRef = Clip3(0, PicWidthInSamples_L - 1, xP + ((nPSW - 1) >> 1) + ((mvDisp[0] + 2) >> 2))$$

$$yRef = Clip3(A, B, xP + ((nPSH - 1) >> 1) + ((mvDisp[1] + 2) >> 2))$$

또 다른 실시 일례로, 수직 방향의 움직임 예측(ME) 범위를 메모리가 허용되는 범위내의 '임의의 값(T)'으로 제한할 수 있으며, 이 경우 변이 벡터의 수직 방향 성분은 임의의 값(T)의 범위를 넘길 수 있어서, 이 방법의 상기 제안한 방법과 혼합되어 수행될 수 있다. 따라서, 변이 벡터의 수직 방향 성분을 메모리가 허용되는 범위 내의 임의의 값(T) 범위 이내로 수렴시키기 위해서, 실시 일례로 다음의 식을 통해 참조 시점의 참조 블록의 위치(xRef, yRef)를 계산할 수 있다.

[0133] [수학식 6]

$$xRef = Clip3( 0, PicWidthInSamples_L - 1, xP + ( (nPSW - 1) >> 1 ) + ( ( mvDisp [0] + 2 ) >> 2 ) )$$

$$yRef = Clip3( yCtb - T, yCtb + CtbSizeY + T - 1, xP + ( (nPSH - 1) >> 1 ) + ( ( mvDisp [1] + 2 ) >> 2 ) )$$

[0134]

[0135]

상술한 방법은 현재 MPEG(Moving Picture Experts Group)과 VCEG(Video Coding Experts Group)에서 공동으로 표준화를 진행 중인 3D-HEVC(High Efficiency Video Coding)에서 사용될 수 있다. 따라서 상술한 방법은 도 16의 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 16에 예시한 바와 같이, 방식A) 주어진 깊이 이상의 깊이에만 적용하는 방식, 방식B) 주어진 깊이 이하에만 적용하는 방식, 방식C) 주어진 깊이에만 적용하는 방식이 있을 수 있다.

[0136]

[표 3] 주어진 CU(혹은 TU) 깊이가 2인 경우, 본 발명의 방법들을 적용하는 범위 결정 방식의 예. (0: 해당 깊이에 적용, X: 해당 깊이에 적용하지 않음.)

| 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   |

[0137]

[0138]

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.

[0139]

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

[0140] [표 4] SPS에 적용된 일예

| seq_parameter_set_rbsp( ) {                 | Descriptor |
|---------------------------------------------|------------|
| <b>profile_idc</b>                          | u(8)       |
| <b>reserved_zero_8bits</b> /* equal to 0 */ | u(8)       |
| <b>level_idc</b>                            | u(8)       |
| ...                                         | ue(v)      |
| <b>restricted_dv_enable_flag</b>            | u(1)       |
| if(restricted_dv_enable_flag)               |            |
| <b>restricted_dv_info</b>                   | ue(v)      |
| ...                                         |            |

[0141]

[0142] [표 5] PPS에 적용된 일예

| pic_parameter_set_rbsp( ) {      | Descriptor |
|----------------------------------|------------|
| <b>pic_parameter_set_id</b>      | ue(v)      |
| <b>seq_parameter_set_id</b>      | ue(v)      |
| <b>entropy_coding_mode_flag</b>  | u(1)       |
| ...                              |            |
| <b>restricted_dv_enable_flag</b> | u(1)       |
| if(restricted_dv_enable_flag)    |            |
| <b>restricted_dv_info</b>        | ue(v)      |
| ...                              |            |

[0143]

[0144] [표 6] Slice Header Syntax에 적용된 일예

| slice_header( ) {                | Descriptor |
|----------------------------------|------------|
| <b>slice_type</b>                | ue(v)      |
| <b>pic_parameter_set_id</b>      | ue(v)      |
| <b>frame_num</b>                 | u(v)       |
| ...                              |            |
| <b>restricted_dv_enable_flag</b> | u(1)       |
| if(restricted_dv_enable_flag)    |            |
| <b>restricted_dv_info</b>        | ue(v)      |
| ...                              |            |

[0145]

[0146] [표 7] Slice Header Syntax에 적용된 또 다른 일예

| slice_header() {                                 | Descriptor |
|--------------------------------------------------|------------|
| <b>lightweight_slice_flag</b>                    | u(1)       |
| if( !lightweight_slice_flag ) {                  |            |
| <b>slice_type</b>                                | ue(v)      |
| <b>pic_parameter_set_id</b>                      | ue(v)      |
| <b>frame_num</b>                                 | u(v)       |
| ...                                              |            |
| }                                                |            |
| if( entropy_coding_mode_flag && slice_type != I) |            |
| <b>cabac_init_idc</b>                            | ue(v)      |
| <b>first_slice_in_pic_flag</b>                   | u(1)       |
| ...                                              |            |
| <b>restricted_dv_enable_flag</b>                 | u(1)       |
| if(restricted_dv_enable_flag)                    |            |
| <b>restricted_dv_info</b>                        | ue(v)      |
| ...                                              |            |

[0147]

[0148]

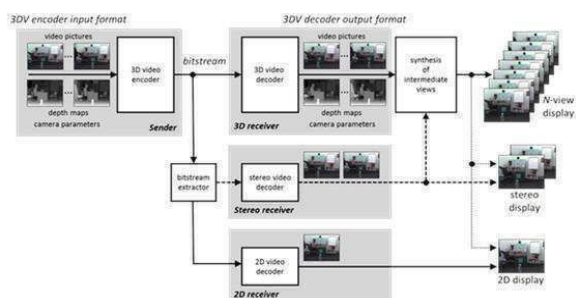
여기서 “restricted\_dv\_enable\_flag” 는 제안하는 방법의 적용 여부를 알려준다. 여기서 제안하는 방법이 적용되었을 경우 “restricted\_dv\_enable\_flag” 가 '1'이되며, 적용되지 않았을 경우 “restricted\_dv\_enable\_flag” 가 '0'이 된다. 그 반대도 가능하다.

[0149]

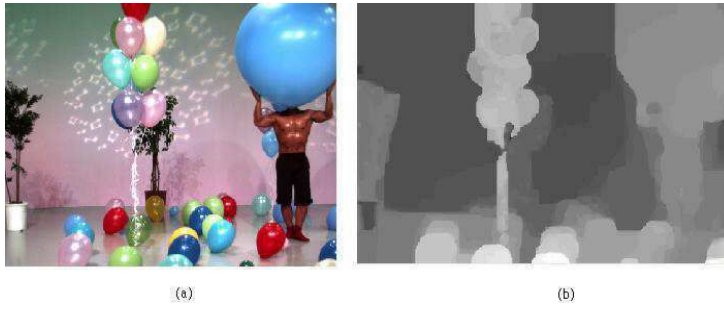
또한, “restricted\_dv\_info” 는 제안 방법이 적용되었을 경우(혹은 “restricted\_dv\_enable\_flag” 이 참일 경우) 활성화되는 syntax로써, 시퀀스, 픽처, 슬라이스 단위로 “수직 방향 성분” 을 클리핑하기 위한 임의의 범위를 시그널링하는 경우에 사용하는 syntax 이다. 여기서 이 값들은 양수 혹은 음수 부호를 가지는 형태로 부호화(se(v))될 수 있다. 혹은 이 값들은 0과 양수 부호를 가지는 형태로 부호화(ue(v))될 수 있다.

## 도면

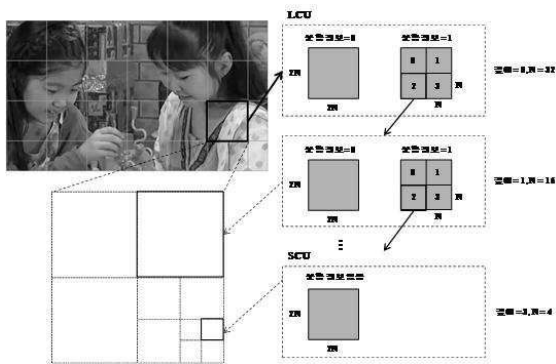
### 도면1



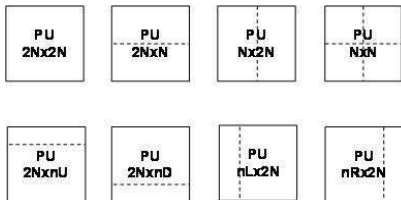
도면2



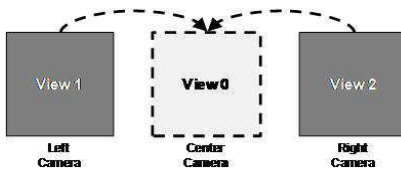
도면3



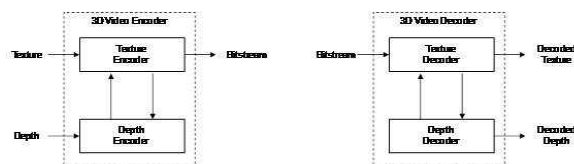
도면4



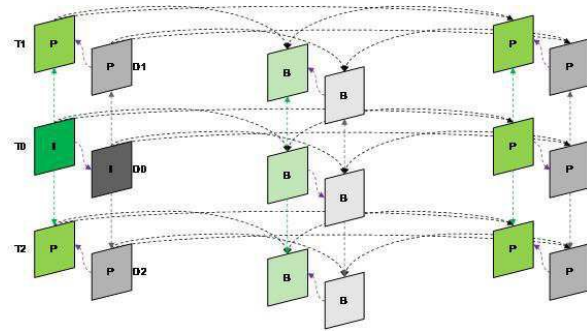
도면5



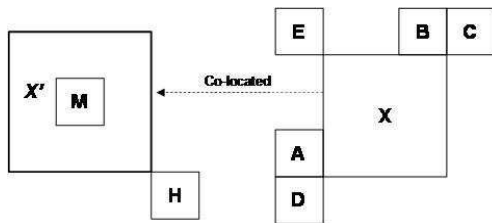
도면6



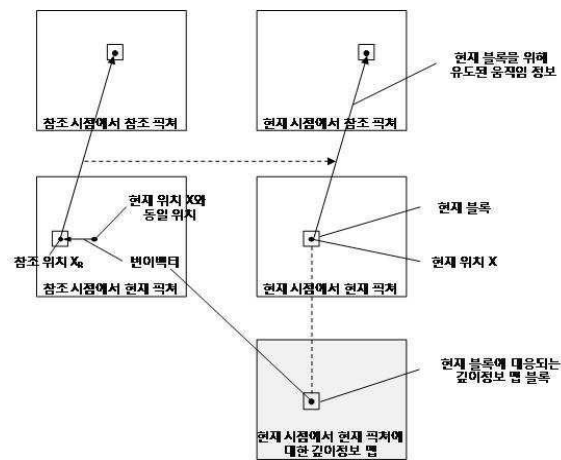
도면7



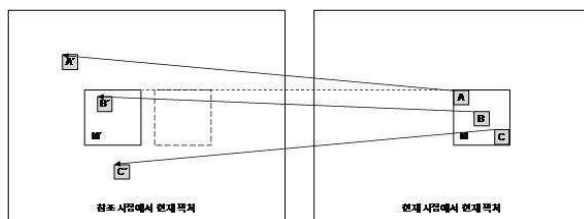
도면8



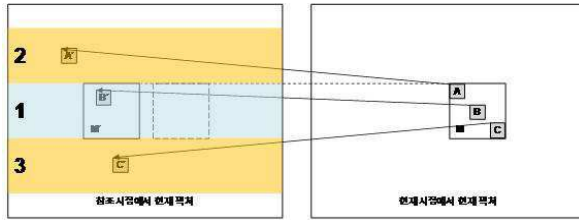
도면9



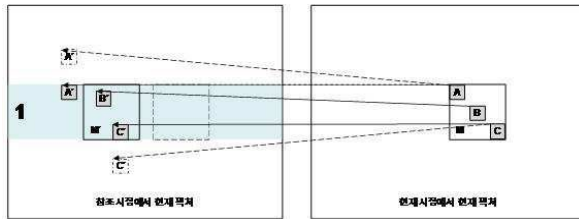
도면10



도면11



도면12



도면13

