(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년02월03일
(11) 등록번호 10-2761095
(24) 등록일자 2025년01월23일

(51) 국제특허분류(Int. Cl.)
H04N 19/117 (2014.01) *H04N 19/14* (2014.01)
H04N 19/176 (2014.01) *H04N 19/184* (2014.01)
H04N 19/70 (2014.01) *H04N 19/82* (2014.01)
(52) CPC특허분류
H04N 19/117 (2015.01)
H04N 19/14 (2015.01)
(21) 출원번호 10-2021-7030932
(22) 출원일자(국제) 2020년04월16일
심사청구일자 2023년03월27일
(85) 번역문제출일자 2021년09월27일
(65) 공개번호 10-2021-0145749
(43) 공개일자 2021년12월02일
(86) 국제출원번호 PCT/CN2020/085075
(87) 국제공개번호 WO 2020/211810
국제공개일자 2020년10월22일
(30) 우선권주장
PCT/CN2019/082855 2019년04월16일 중국(CN)
(56) 선행기술조사문헌
US20180192050 A1
US20130259118 A1

(73) 특허권자
두인 비전 컴퍼니 리미티드
중국, 베이징 100041, 스징산 디스트릭트, 스싱 로드, 넘버 30, 넘버 3빌딩, 2층, 룸 비-0035
바이트댄스 아이엔씨
미국, 90066 캘리포니아 로스엔젤레스 스위트룸 137호 6층 제퍼슨 블러바드 웨스트 12655
(72) 발명자
리우, 홍빈
중국 100080 베이징 하이디안 디스트릭트 지춘 로드 넘버 63 차이나 세털라이트 커뮤니케이션스 타워 진리토우티아오 포스트오피스
창, 리
미국 90066 캘리포니아 로스앤젤레스 슈트 넘버 137 6 플로어 웨스트 제퍼슨 블러바드 12655
(뒷면에 계속)
(74) 대리인
특허법인 무한

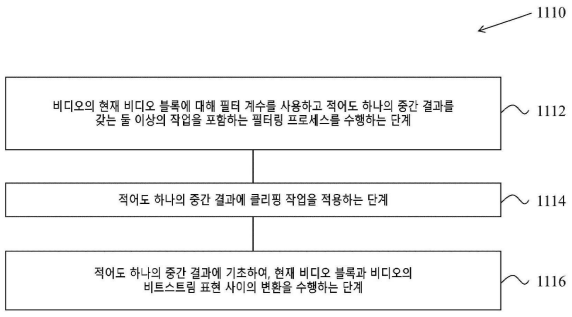
전체 청구항 수 : 총 9 항 심사관 : 김영태

(54) 발명의 명칭 **비디오 코딩을 위한 적응형 루프 필터링**

(57) 요 약

적응형 루프 필터링을 위한 장치, 시스템 및 방법이 설명된다. 예시적인 측면에서, 비디오 처리를 위한 방법은 비디오의 현재 비디오 블록에 대해, 필터 계수를 사용하고 적어도 하나의 중간 결과를 갖는 둘 이상의 작업을 포함하는 필터링 프로세스를 수행하는 단계, 적어도 하나의 중간 결과에 클리핑 작업을 적용하는 단계, 및 적어도 하나의 중간 결과에 기초하여, 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 수행하는 단계 - 적어도 하나의 중간 결과는 현재 비디오 블록의 현재 샘플과 현재 샘플의 이웃 샘플 간의 차이와 필터 계수의 가중합에 기초함 - 를 포함한다.

대 표 도



(52) CPC특허분류
H04N 19/176 (2015.01)
H04N 19/184 (2015.01)
H04N 19/70 (2015.01)
H04N 19/82 (2015.01)
(72) 발명자
창, 카이
미국 90066 캘리포니아 로스앤젤레스 슈트 넘버 137 6 플로어 웨스트 제퍼슨 블러바드 12655
추앙, 시아오 치앙
미국 90066 캘리포니아 로스앤젤레스 슈트 넘버 137 6 플로어 웨스트 제퍼슨 블러바드 12655
뎅, 지핀
중국 100080 베이징 하이디안 디스트릭트 지춘 로드 넘버 63 차이나 세털라이트 커뮤니케이션스 타워 진리토우티아오 포스트오피스

명 세 서

청구범위

청구항 1

비디오 데이터 처리 방법에 있어서,

비디오의 제1 비디오 영역과 상기 비디오의 비트스트림 사이의 변환을 위해, 상기 제1 비디오 영역에 대응하는 제1 신택스 요소가 상기 비트스트림에 존재하는지를 결정하는 단계 - 상기 제1 신택스 요소는 제1 비디오 영역이 참조하는 적응형 루프 필터(Adaptive Loop Filter; ALF) 파라미터 세트를 나타내고, 제1 비디오 영역은 비디오 픽처 또는 비디오 슬라이스임 -; 및

상기 결정하는 단계에 기초하여 상기 변환을 수행하는 단계

를 포함하고,

상기 적응형 루프 필터 파라미터 세트가 디스에이블된 경우, 상기 제1 신택스 요소는 상기 비트스트림에 존재하지 않고,

상기 제1 비디오 영역의 샘플에 대해, 필터 인덱스는 서로 다른 방향의 다수의 샘플 차이에 기초하여 유도되고, 특정 루마 필터는, 상기 필터 인덱스에 기초하여, 상기 적응형 루프 필터 파라미터 세트의 상기 루마 필터 및 상기 루마 필터 세트에 포함되지 않은 다른 루마 필터로부터 선택되고,

상기 비디오 영역은, 다중 비디오 코딩 트리 블록을 포함하고, 상기 비디오 코딩 트리 블록은, 다중 M*M 비디오 블록으로 분할되고, M은 2 또는 4이고,

동일한 필터 인덱스는, 동일한 M*M 비디오 블록의 샘플에 대해 적용되고,

서로 다른 방향의 상기 다수의 샘플 차이는, 1:2 서브 샘플링 속도(rate)에 기초하여 모든 M*M 비디오 영역에 대해 유도되는,

방법.

청구항 2

제1항에 있어서,

상기 적응형 루프 필터 파라미터 세트는, 적응형 루프 필터링 적응형 파라미터 세트를 포함하는,

방법.

청구항 3

제1항에 있어서,

상기 필터 인덱스의 최대값은, N으로 표시되고,

상기 루마 필터 세트의 루마 필터의 수는, N + 1보다 작거나 같은,

방법.

청구항 4

제3항에 있어서,

상기 루마 필터 세트의 루마 필터의 수가 N + 1보다 작은 것에 응답하여, 다른 필터 인덱스를 갖는 적어도 2개

의 루마 필터가 동일한 필터 계수를 갖는,

방법.

청구항 5

제1항에 있어서,

상기 변환은, 현재 비디오 블록을 상기 비트스트림으로 인코딩 하는 단계

를 포함하는,

방법.

청구항 6

제1항에 있어서,

상기 변환은, 상기 비트스트림으로 부터 현재 비디오 블록을 디코딩 하는 단계

를 포함하는,

방법.

청구항 7

프로세서 및 명령들이 있는 비일시적 메모리를 포함하는 비디오 데이터를 처리하기 위한 장치에 있어서,

상기 명령들은,

상기 프로세서에 의해 실행 시, 상기 프로세서로 하여금,

비디오의 제1 비디오 영역과 상기 비디오의 비트스트림 사이의 변환을 위해, 상기 제1 비디오 영역에 대응하는 제1 신택스 요소가 상기 비트스트림에 존재하는지를 결정하고 - 상기 제1 신택스 요소는 제1 비디오 영역이 참조하는 적응형 루프 필터(Adaptive Loop Filter; ALF) 파라미터 세트를 나타내고, 제1 비디오 영역은 비디오 픽처 또는 비디오 슬라이스임 -; 및

상기 결정하는 단계에 기초하여 상기 변환을 수행

하도록 하고,

상기 적응형 루프 필터 파라미터 세트가 디스에이블된 경우, 상기 제1 신택스 요소는 상기 비트스트림에 존재하지 않고,

상기 제1 비디오 영역의 샘플에 대해, 필터 인덱스는 서로 다른 방향의 다수의 샘플 차이에 기초하여 유도되고, 특정 루마 필터는, 상기 필터 인덱스에 기초하여, 상기 적응형 루프 필터 파라미터 세트의 상기 루마 필터 및 상기 루마 필터 세트에 포함되지 않은 다른 루마 필터로부터 선택되고,

상기 비디오 영역은, 다중 비디오 코딩 트리 블록을 포함하고, 상기 비디오 코딩 트리 블록은, 다중 M*M 비디오 블록으로 분할되고, M은 2 또는 4이고,

동일한 필터 인덱스는, 동일한 M*M 비디오 블록의 샘플에 대해 적용되고,

서로 다른 방향의 상기 다수의 샘플 차이는, 1:2 서브 샘플링 속도에 기초하여 모든 M*M 비디오 영역에 대해 유도되는,

장치.

청구항 8

명령들을 저장하는 비일시적 컴퓨터 판독가능 저장 매체에 있어서,

상기 명령들은, 프로세서로 하여금,

비디오의 제1 비디오 영역과 상기 비디오의 비트스트림 사이의 변환을 위해, 상기 제1 비디오 영역에 대응하는 제1 신택스 요소가 상기 비트스트림에 존재하는지를 결정하고 - 상기 제1 신택스 요소는 제1 비디오 영역이 참조하는 어떤 적응형 루프 필터(Adaptive Loop Filter; ALF) 파라미터 세트를 나타내고, 제1 비디오 영역은 비디오 픽처 또는 비디오 슬라이스임 -; 및

상기 결정하는 단계에 기초하여 상기 변환을 수행

하도록 하고,

상기 적응형 루프 필터 파라미터 세트가 디스에이블된 경우, 상기 제1 신택스 요소는 상기 비트스트림에 존재하지 않고,

상기 제1 비디오 영역의 샘플에 대해, 필터 인덱스는 서로 다른 방향의 다수의 샘플 차이에 기초하여 유도되고, 특정 루마 필터는, 상기 필터 인덱스에 기초하여, 상기 적응형 루프 필터 파라미터 세트의 상기 루마 필터 및 상기 루마 필터 세트에 포함되지 않은 다른 루마 필터로부터 선택되고,

상기 비디오 영역은, 다중 비디오 코딩 트리 블록을 포함하고, 상기 비디오 코딩 트리 블록은, 다중 M*M 비디오 블록으로 분할되고, M은 2 또는 4이고,

동일한 필터 인덱스는, 동일한 M*M 비디오 블록의 샘플에 대해 적용되고,

서로 다른 방향의 상기 다수의 샘플 차이는, 1:2 서브 샘플링 속도에 기초하여 모든 M*M 비디오 영역에 대해 유도되는,

비일시적 컴퓨터 판독가능 저장 매체.

청구항 9

비디오 처리 장치에 의해 수행된 방법에 의해 생성된 비디오의 비트스트림을 저장한 비일시적 컴퓨터 판독가능 기록 매체에 있어서,

상기 방법은,

비디오의 제1 비디오 영역과 상기 비디오의 비트스트림 사이의 변환을 위해, 상기 제1 비디오 영역에 대응하는 제1 신택스 요소가 상기 비트스트림에 존재하는지를 결정하는 단계 - 상기 제1 신택스 요소는 제1 비디오 영역이 참조하는 어떤 적응형 루프 필터(Adaptive Loop Filter; ALF) 파라미터 세트를 나타내고, 제1 비디오 영역은 비디오 픽처 또는 비디오 슬라이스임 -; 및

상기 결정하는 단계에 기초하여 상기 변환을 수행하는 단계

를 포함하고,

상기 적응형 루프 필터 파라미터 세트가 디스에이블된 경우, 상기 제1 신택스 요소는 상기 비트스트림에 존재하지 않고,

상기 제1 비디오 영역의 샘플에 대해, 필터 인덱스는 서로 다른 방향의 다수의 샘플 차이에 기초하여 유도되고, 특정 루마 필터는, 상기 필터 인덱스에 기초하여, 상기 적응형 루프 필터 파라미터 세트의 상기 루마 필터 및 상기 루마 필터 세트에 포함되지 않은 다른 루마 필터로부터 선택되고,

상기 비디오 영역은, 다중 비디오 코딩 트리 블록을 포함하고, 상기 비디오 코딩 트리 블록은, 다중 M*M 비디오 블록으로 분할되고, M은 2 또는 4이고,

동일한 필터 인덱스는, 동일한 M*M 비디오 블록의 샘플에 대해 적용되고,

서로 다른 방향의 상기 다수의 샘플 차이는, 1:2 서브 샘플링 속도에 기초하여 모든 M*M 비디오 영역에 대해 유

도되는,

비일시적 컴퓨터 판독가능 기록 매체.

청구항 10

삭제

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

청구항 51

삭제

청구항 52

삭제

청구항 53

삭제

청구항 54

삭제

청구항 55

삭제

청구항 56

삭제

발명의 설명

기 술 분 야

[0001] 이 특허 문서는 비디오 코딩 기술, 장치 및 시스템에 관한 것이다.

[0002] 본 출원은 2019년 4월 16일에 출원된 국제 특허 출원 번호 PCT/CN2019/082855의 우선권과 이익을 주장하는 2020년 4월 16일에 출원된 국제 특허 출원 PCT/CN2020/085075에 기초한다. 법에 따라 모든 목적을 위해, 상기 출원의 전체 공개는 이 출원의 공개의 일환으로 참조에 의해 통합된다.

배 경 기 술

[0003] 비디오 압축의 발전에도 불구하고 디지털 비디오는 여전히 인터넷 및 기타 디지털 통신 네트워크에서 가장 큰 대역폭 사용을 차지한다. 비디오를 수신하고 표시할 수 있는 연결된 사용자 장치의 수가 증가함에 따라 디지털 비디오 사용에 대한 대역폭 수요가 계속 증가할 것으로 예상된다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0004] 디지털 비디오 코딩, 특히 비디오 코딩을 위한 적응형 루프 필터링과 관련된 장치, 시스템 및 방법이 설명된다. 설명된 방법은 기존의 비디오 코딩 표준(예를 들어, HEVC(고효율 비디오 코딩(High Efficiency Video Coding)) 및 미래의 비디오 코딩 표준(예를 들어, VVC(범용 비디오 코딩(Versatile Video Coding)) 또는 코덱 모두에 적용될 수 있다.

[0005] 비디오 코딩 표준은 주로 잘 알려진 ITU-T 및 ISO/IEC 표준의 개발을 통해 발전해 왔다. ITU-T는 H.261과 H.263을, ISO/IEC는 MPEG-1과 MPEG-4 Visual을, 두 조직은 공동으로 H.262/MPEG-2 Video와 H.264/MPEG-4 Advanced 비디오 코딩(AVC) 및 H.265/HEVC 표준을 제작했다. H.262 이후로, 비디오 코딩 표준은 시간적 예측과 변환 코딩이 사용되는 하이브리드 비디오 코딩 구조에 기초한다. HEVC를 넘어 미래의 비디오 코딩 기술을 연구하기 위해 2015년 VCEG와 MPEG가 공동으로 연합 비디오 연구 팀 (Joint Video Exploration Team) (JVET)을 설립했다. 그 이후로 JVET는 많은 새로운 방법을 채택하고 연합 연구 모델 (Joint Exploration Model) (JEM)이라는 참조 소프트웨어에 적용했다. 2018년 4월, VCEG(Q6/16)와 ISO/IEC JTC1 SC29/WG11(MPEG) 간의 JVET가 HEVC에 비해 50% 비트 전송률 감소를 목표로 하는 VVC 표준 작업을 위해 만들어졌다.

[0006] 하나의 대표적인 측면에서, 개시된 기술은 비디오 처리(video processing)를 위한 방법을 제공하기 위해 사용될 수도 있다. 이 방법은, 비디오의 현재 비디오 블록(current video block)에 대해, 필터 계수(filter coefficient)를 사용하고 적어도 하나의 중간 결과(intermediate result)를 갖는 둘 이상의 작업(operation)을 포함하는 필터링 프로세스(filtering process)를 수행하는 단계, 적어도 하나의 중간 결과에 클리핑 작업(clipping operation)을 적용하는 단계, 및 적어도 하나의 중간 결과에 기초하여, 현재 비디오 블록과 비디오의 비트스트림 표현(bitstream representation) 사이의 변환을 수행하는 단계 - 적어도 하나의 중간 결과는 현재 비디오 블록의 현재 샘플과 현재 샘플의 이웃 샘플 간의 차이와 필터 계수의 가중 합(weighted sum)에 기초함 - 를 포함한다.

[0007] 다른 대표적인 측면에서, 개시된 기술은 비디오 처리를 위한 방법을 제공하기 위해 사용될 수도 있다. 이 방법은, 비디오의 현재 비디오 블록을 비디오의 비트스트림 표현으로 인코딩 하는 단계 - 현재 비디오 블록은 적응형 루프 필터(adaptive loop filter)(ALF)로 코딩됨 -, 및 하나 이상의 시간 적응형 필터(temporal adaptive filter) 세트의 사용 가능성 또는 사용에 기초하여, 비트스트림 표현에서 시간 적응형 필터의 하나 이상의 세트 내의 시간 적응형 필터 세트의 표시(indication)를 선택적으로 포함하는 단계를 포함한다.

[0008] 또 다른 대표적인 측면에서, 개시된 기술은 비디오 처리를 위한 방법을 제공하기 위해 사용될 수도 있다. 이 방법은 비디오의 비트스트림 표현에서 시간 적응형 필터 세트의 표시에 기초하여, 적응형 루프 필터(ALF)로 코딩된 비디오의 현재 비디오 블록에 적용 가능한 시간 적응형 필터의 세트를 포함하는 시간 적응형 필터의 하나 이상의 세트의 사용 가능성 또는 사용을 결정하는 단계, 및 결정하는 단계에 기초하여, 시간 적응형 필터의 세트를 선택적으로 적용함으로써 비트스트림 표현으로부터 디코딩 된 현재 비디오 블록을 생성하는 단계를

포함한다.

[0009] 또 다른 대표적인 측면에서, 개시된 기술은 비디오 처리를 위한 방법을 제공하기 위해 사용될 수도 있다. 이 방법은, 적응형 루프 필터로 코딩 된 현재 비디오 블록에 대해, 사용 가능한 시간적 ALF 계수 세트에 기초하여 시간 적응형 루프 필터링(ALF) 계수 세트의 수를 결정하는 단계 - 사용 가능한 시간적 ALF 계수 세트는 결정하는 단계 이전에 인코딩 또는 디코딩 되었고, 및 ALF 계수 세트의 수는 타일 그룹, 타일, 슬라이스, 픽처, 코딩 트리 블록(CTB), 또는 현재 비디오 블록을 포함하는 비디오 유닛에 대해 사용됨 -, 및 시간적 ALF 계수 세트의 수에 기초하여, 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다.

[0010] 또 다른 대표적인 측면에서, 개시된 기술은 비디오 처리를 위한 방법을 제공하기 위해 사용될 수도 있다. 이 방법은 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역의 헤더에 있는 적응형 루프 필터링(ALF)의 표시가 비트스트림 표현과 연관된 적응형 파라미터 세트(adaptive parameter set)(APS) 네트워크 추상 계층(network abstraction layer)(NAL) 유닛에서 ALF의 표시와 동일하다고 결정하는 단계, 및 변환을 수행하는 단계를 포함한다.

[0011] 또 다른 대표적인 측면에서, 개시된 기술은 비디오 처리를 위한 방법을 제공하기 위해 사용될 수도 있다. 이 방법은 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역에 의해 사용되는 적응 루프 필터의 유형에 기초한 비선형 적응 루프 필터링(ALF) 작업을 선택적으로 활성화하는 단계, 및 선택적으로 활성화 하는 단계에 후속하여, 변환을 수행하는 단계를 포함한다.

[0012] 또 다른 대표적인 측면에서, 상술한 방법은 프로세서 실행 가능 코드의 형태로 구현되어 컴퓨터 판독 가능 프로그램 매체에 저장된다.

[0013] 또 다른 대표적인 측면에서, 전술한 방법을 수행하도록 구성되거나 작업가능한 장치가 개시된다. 장치는 이 방법을 구현하도록 프로그래밍된 프로세서를 포함할 수 있다.

[0014] 또 다른 대표적인 측면에서, 비디오 디코더 장치는 여기에 설명된 바와 같은 방법을 구현할 수도 있다.

[0015] 개시된 기술의 상기 및 기타 측면 및 특징은 도면, 설명 및 청구범위에서 보다 상세하게 설명된다.

도면의 간단한 설명

[0016] 도 1은 비디오 코딩을 위한 인코더 블록 다이어그램의 예를 나타낸다.

도 2a, 2b 및 2c는 기하학 변환 기반 적응형 루프 필터(GALF) 필터 형태의 예를 나타낸다.

도 3은 GALF 인코더 결정에 대한 흐름 그래프의 예를 나타낸다.

도 4a 내지 4d는 적응형 루프 필터(ALF) 분류를 위한 서브 샘플링 된 라플라스 계산의 예를 나타낸다.

도 5는 루마 필터 형상의 예를 나타낸다.

도 6은 WVGA(Wide Video Graphic Array) 시퀀스의 영역 분할의 예를 나타낸다.

도 7은 재형성에 의한 디코딩 흐름의 예시적인 흐름도를 나타낸다.

도 8은 양방향 광학 흐름(BIO) 알고리즘에 의해 사용되는 광학 흐름 궤적의 예를 나타낸다.

도 9a 및 9b는 블록 확장 없이 양방향 광학 흐름(BIO) 알고리즘을 사용하는 예시적인 스냅샷을 나타낸다.

도 10은 광학 흐름(PROF)에 의한 예측 정제의 예를 나타낸다.

도 11a 내지 도 11f는 개시된 기술에 따른 적응형 루프 필터링을 위한 예시적인 방법의 흐름도를 나타낸다.

도 12는 본 문서에서 설명하는 비주얼 매체 디코딩 또는 비주얼 매체 인코딩 기술을 구현하기 위한 하드웨어 플랫폼의 일 예의 블록도이다.

도 13은 개시된 기술이 구현될 수 있는 예시적인 비디오 처리 시스템의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0017] 고해상도 비디오의 수요가 증가함에 따라 비디오 코딩 방법과 기술은 현대 기술에서 유비쿼터스이다. 비디오 코

덱에는 일반적으로 디지털 비디오를 압축하거나 압축 해제하는 전자 회로 또는 소프트웨어가 포함되며 코딩 효율성을 높이기 위해 지속적으로 개선되고 있다. 비디오 코덱은 압축되지 않은 비디오를 압축 형식으로 변환하거나 그 반대의 경우도 마찬가지이다. 비디오 품질, 비디오(비트 전송률에 의해 결정됨), 인코딩 및 디코딩 알고리즘의 복잡성, 데이터 손실 및 오류에 대한 민감도, 편집 용이성, 무작위 액세스 및 종단 간 지연(대기 시간) 사이에는 복잡한 관계가 있다. 압축 형식은 일반적으로 표준 비디오 압축 사양(예를 들어, HEVC) 표준(H.265 또는 MPEG-H 파트 2라고도 함), 범용 비디오 코딩(VVC) 표준또는 기타 현재 및/또는 미래의 비디오 코딩 표준을 준수한다.

[0018] 일부 실시예에서, 미래의 비디오 코딩 기술은 연합 연구 모델 (Joint Exploration Model)(JEM)로 알려진 참조 소프트웨어를 사용하여 연구된다. JEM에서 서브 블록 기반 예측은 몇몇의 코딩 툴에서 선택되며, 예를 들어, 아핀 예측, 대체 시간적 모션 벡터 예측(ATMVP), 공간-시간적 모션 벡터 예측(STMVP), 양방향 광학 흐름(bi-directional optical flow)(BIO), 프레임 레이트 업 변환 (Frame Rate Up Conversion) (FRUC), 로컬 적응형 모션 벡터 해상도 (Locally Adaptive Motion Vector Resolution) (LAMVR), 오버랩 블록 모션 보상 (Overlapped Block Motion Compensation) (OBMC), 로컬 조명 보상 (Local Illumination Compensation) (LIC) 및 디코더측 모션 벡터 정제 (Decoder-side Motion Vector Refinement) (DMVR)이 있다.

[0019] 개시된 기술의 실시예는 런타임 성능을 향상시키기 위해 기존 비디오 코딩 표준(예를 들어, HEVC, H.265)과 미래 표준에 적용될 수 있다. 섹션 제목은 설명의 가독성을 향상시키기 위해 본 문서에서 사용되며, 토론 또는 실시예(및/또는 구현)를 각 섹션으로만 제한하지 않는다.

[0020] **1 컬러 공간 및 크로마 서브 샘플링의 예**

[0021] 컬러 모델(또는 컬러 시스템)이라고도 하는 컬러 공간은 일반적으로 3또는 4값 또는 컬러 컴포넌트(예를 들어, RGB)로 컬러 범위를 숫자의 줄로 묘사하는 추상적인 수학적 모델이다. 기본적으로 컬러 공간은 좌표계와 서브 공간의 정교화이다.

[0022] 비디오 압축의 경우 가장 자주 사용되는 컬러 공간은 YCbCr 및 RGB이다.

[0023] YCbCr, Y'CbCr, 또는 Y'Pb/Cb Pr/Cr은 YCBCR 또는 Y'CBCR으로 기재된 컬러 공간의 제품군은 비디오 및 디지털 픽처 시스템의 컬러 이미지 파이프라이닝의 일부로 사용된다. Y'는 루마 컴포넌트이고 CB와 CR은 청색-차이 및 적색-차이 크로마 컴포넌트이다. Y'(프라임) Y에서 구별된다, Y'(프라임 포함)는 휘도인 Y와 구별되고, 즉, 감마 보정된 RGB 프라이머리에 기초하여 빛의 강도가 비선형적으로 인코딩 된다.

[0024] 크로마 서브 샘플링(Chroma subsampling)은 휘도보다 컬러 차이에 대한 인간 시각 시스템의 낮은 선명도를 사용하여 루마 정보보다 크로마 정보에 대해 더 낮은 해상도를 구현하여 이미지를 인코딩 하는 방식이다.

[0025] **1.1 4:4:4 컬러 형식**

[0026] 3개의 Y'CbCr 컴포넌트 각각은 동일한 샘플링 레이트를 가지므로 크로마 서브 샘플링이 없다. 이 방식은 때때로 고급 필름 스캐너와 시네마틱 포스트 프로덕션에 사용된다.

[0027] **1.2 4:2:2 컬러 형식**

[0028] 두 크로마 컴포넌트는 루마의 샘플 비율의 절반으로 샘플링되며, 예를 들어 수평 크로마 해상도가 반으로 줄어든다. 이렇게 하면 압축되지 않은 비디오 신호의 대역폭이 시각적 차이가 거의 또는 전혀 없는 3분의 1로 줄어든다.

[0029] **1.3 4:2:0 컬러 형식**

[0030] 4:2:0에서, 수평 샘플링은 4:1:1에 비해 두 배가되지만 Cb 및 Cr 채널은 이 구성표의 각 대체 줄에서만 샘플링되기 때문에 수직 해상도가 반으로 줄어든다. 따라서 데이터 속도(data rate)는 동일하다. Cb와 Cr은 각각 수평 및 수직 모두에서 2의 계수로 서브 샘플링 된다. 4:2:0 방식의 세 가지 변형이 있으며, 수평 및 수직 위치가 상이하다.

[0031] ○ MPEG-2에서는 Cb와 Cr이 수평으로 구비된다. Cb와 Cr은 수직 방향의 픽셀 간에 위치된다(중간에 위치됨).

[0032] ○ JPEG/JFIF, H.261 및 MPEG-1에서 Cb와 Cr은 대체 루마 샘플 사이의 중간에 교차하여 위치된다.

[0033] ○ 4:2:0 DV에서 Cb와 Cr은 수평 방향으로 공동 위치로 지정된다. 수직 방향으로는 교대선에 공동으로

구성된다.

[0034] 2 일반적인 비디오 코덱의 코딩 흐름의 예

[0035] 도 1은 3개의 인루프 필터링 블록을 포함하는 VVC의 인코더 블록 다이어그램의 예를 보여준다: 디블로킹 필터(deblocking filter)(DF), 샘플 적응 오프셋(sample adaptive offset)(SAO) 및 ALF. 미리 정의된 필터를 사용하는 DF와 달리, SAO 및 ALF는 오프셋을 추가하고 오프셋 및 필터 계수를 시그널링 하는 코딩 된 부가 정보와 함께 유한 임펄스 응답(finite impulse response)(FIR) 필터를 각각 적용하여 원래 샘플과 재구성된 샘플 간의 평균 제곱 오류를 줄이기 위해 현재 픽처의 원래 샘플을 활용한다. ALF는 각 픽처의 마지막 처리 단계에 위치하며 이전 단계에서 만든 아티팩트를 포착하고 수정하려는 툴(tool)로 간주될 수 있다.

[0036] 3 JEM의 형상 변환 기반 적응형 루프 필터링의 예

[0037] JEM에서는 블록 기반 필터 어댑티브가 있는 형상 변환 기반 적응형 루프 필터링(GALF)가 적용된다. 루마 컴포넌트의 경우 로컬 그레디언트(local gradient)의 방향과 활동에 따라 각 2X2 블록에 대해 25개의 필터 중 하나가 선택된다.

[0038] 3.1 필터 형태의 예

[0039] JEM에서, 최대 3개의 다이아몬드 필터 형태 (5x5 다이아몬드, 7x7 다이아몬드 및 9x9 다이아몬드에 대해 각각 도 2a, 2b 및 2c에 도시됨)가 루마 컴포넌트에 대해 선택될 수 있다. 인덱스는 루마 컴포넌트에 사용되는 필터 형태를 나타내기 위해 픽처 레벨에서 시그널링 된다. 픽처의 크로마 컴포넌트의 경우 5X5 다이아몬드 형태가 항상 사용된다.

[0040] 3.1.1 블록 분류

[0041] 각 2×2 블록은 25개 클래스 중 하나로 분류된다. 분류 인덱스 C는 다음과 같이 방향성 D와 활동성 $\hat{A}$의 양자화된 값에 기초하여 유도된다:

[0042] $$C = 5D + \hat{A} \quad (1)$$

[0043] D와 $\hat{A}$를 계산하기 위해, 수평, 수직 및 두 개의 대각선 방향의 그레디언트는 먼저 1-D 라플라스(1-D Laplacian)를 사용하여 계산된다:

$$g_v = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} V_{k,l}, \; V_{k,l} = |2R(k,l) - R(k,l-1) - R(k,l+1)|, \quad (2)$$

$$g_h = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} H_{k,l}, \; H_{k,l} = |2R(k,l) - R(k-1,l) - R(k+1,l)|, \quad (3)$$

$$g_{d1} = \sum_{k=i-2}^{i+3} \sum_{l=j-3}^{j+3} D1_{k,l}, D1_{k,l} = |2R(k,l) - R(k-1,l-1) - R(k+1,l+1)| \quad (4)$$

$$g_{d2} = \sum_{k=i-2}^{i+3} \sum_{j=j-2}^{j+3} D2_{k,l}, \; D2_{k,l} = |2R(k,l) - R(k-1,l+1) - R(k+1,l-1)| \quad (5)$$

[0044]

[0045] 인덱스 i 및 j 블록에서 왼쪽 상부 샘플의 좌표를 2×2 참조하고 $R(i,j)$는 좌표 (i,j)에서 재구성된 샘플을 나타낸다.

[0046] 그런 다음 가로 및 수직 방향의 그레디언트의 D최대 값과 최소 값은 다음과 같이 설정된다:

[0047] $$g_{h,v}^{max} = max(g_h, g_v), \quad g_{h,v}^{min} = min(g_h, g_v) \quad (6)$$

[0048] 두 대각선 방향의 그레디언트의 최대 값과 최소 값은 다음과 같이 설정된다.

[0049] $$g_{d0,d1}^{max} = max(g_{d0}, g_{d1}), \quad g_{d0,d1}^{min} = min(g_{d0}, g_{d1}) \quad (7)$$

[0050] 방향성 D의 값을 유도하기 위해, 이러한 값은 서로 비교되고 두 임계값 t_1 및 t_2과 비교된다:

[0051] **1단계**. $g_{h,v}^{max} \le t_1 \cdot g_{h,v}^{min}$ 및 $g_{d0,d1}^{max} \le t_1 \cdot g_{d0,d1}^{min}$ 둘이 참이면, D는 0으로 설정된다.

[0052] **2단계**. $g_{h,v}^{max}/g_{h,v}^{min} > g_{d0,d1}^{max}/g_{d0,d1}^{min}$ 이면, 3단계에서 계속하고, 그렇지 않으면 4단계에서 계속된다.

[0053] **3단계**. $g_{h,v}^{max} > t_2 \cdot g_{h,v}^{min}$ 이면, D는 2로 설정되고, 그렇지 않으면 D는 1로 설정된다.

[0054] **4단계**. $g_{d0,d1}^{max} > t_2 \cdot g_{d0,d1}^{min}$ 이면, D는 4로 설정되고, 그렇지 않으면 D는 3으로 설정된다.

[0055] 활동 값(activity value) A는 다음과 같이 계산된다:

[0056] $$A = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} (V_{k,l} + H_{k,l}) \quad (8)$$

[0057] A는 0~4의 범위에 대해 더 양자화되고, 양자화된 값은 $\hat{A}$ 로 표시된다.픽처의 두 크로마 컴포넌트에 대해 분류 방법이 적용되지 않으며, 즉 단일 ALF 계수 세트가 각 크로마 컴포넌트에 적용된다.

[0058] ### 3.1.2 필터 계수의 기하학적 변환

[0059] 각 2X2 블록을 필터링 하기 전에 대응하는 블록에 대해 계산된 그레디언트 값에 따라 회전 또는 대각선 및 수직 뒤집기와 같은 기하학적 변환이 필터 계수 f(k,l)에 적용된다. 이는 필터 지원 영역의 샘플에 이러한 변환을 적용하는 것과 같다. 아이디어는 ALF가 적용되는 다른 블록을 방향성을 정렬하여 더 유사하게 만드는 것이다.

[0060] 대각선, 수직 뒤집기 및 회전을 포함한 세 가지 기하학적 변환이 도입된다:

[0061] $$\text{대각선: } f_D(k,l) = f(l,k),$$
$$\text{수직 뒤집기: } f_V(k,l) = f(k, K-l-1), \quad (9)$$
$$\text{회전: } f_R(k,l) = f(K-l-1, k).$$

[0062] K는 필터의 크기이고, 0≤k,l≤K-1은 위치 (0,0)이 왼쪽 상부 모서리에 있고 위치 (K-1,K-1)이 오른쪽 하부 모서리에 있도록 하는 계수 좌표이다. 변환은 대응하는 블록에 대해 계산된 그레디언트 값에 따라 필터 계수 f(k, l)에 적용된다. 변환과 네 방향의 네 가지 그레디언트 사이의 관계는 표 1에 요약되어 있다..

표 1

[0063] 한 블록및 변환에 대해 계산된 그레디언트 매핑

그레디언트 값	변형
$g_{d2} < g_{d1}$ 및 $g_h < g_v$	변환 없음
$g_{d2} < g_{d1}$ 및 $g_v < g_h$	대각선
$g_{d1} < g_{d2}$ 및 $g_h < g_v$	수직 뒤집기
$g_{d1} < g_{d2}$ 및 $g_v < g_h$	회전

[0064] ### 3.1.3 필터 파라미터의 시그널링

[0065] JEM에서, GALF 필터 파라미터는 제1 CTU, 즉 슬라이스 헤더 후 및 제1 CTU의 SAO 파라미터 전에 시그널링 된다. 최대 25세트의 루마 필터 계수가 시그널링 될 수 있다. 비트 오버헤드를 줄이기 위해, 서로 다른 분류의 필터 계수를 병합할 수 있다. 또한, 레퍼런스 픽처의 GALF 계수는 저장되어 현재 픽처의 GALF 계수로 재사용할 수 있다. 현재 픽처는 레퍼런스 픽처에 저장된 GALF 계수를 사용하고 GALF 계수 시그널링을 우회할 수 있다. 이 경우, 레퍼런스 픽처 중 하나에 대한 인덱스만 시그널링 되고, 표시된 레퍼런스 픽처의 저장된 GALF 계수는 현재 픽처에 대해 상속된다.

[0066] GALF 시간적 예측을 지원하려면 GALF 필터 세트의 후보 목록이 유지된다. 새 시퀀스를 디코딩할 때 후보 목록은 비어 있다. 한 장의 픽처를 디코딩 한 후 대응하는 필터 세트를 후보 목록에 추가할 수 있다. 후보 목록의 크기가 허용되는 최대 값(즉, 현재 JEM에서 6)에 도달하면 새 필터 세트가 가장 오래된 세트를 디코딩 순서로 덮어쓰고, 즉 선입선출(first-in-first-out)(FIFO) 규칙이 적용되어 후보 목록을 업데이트한다. 중복을 방지하기 위해, 대응하는 픽처가 GALF 시간적 예측을 사용하지 않는 경우에만 세트를 목록에 추가할 수 있다. 시간적 확장성을 지원하기 위해 필터 세트의 여러 후보 목록이 있으며 각 후보 목록은 시간적 계층과 연관된다. 보다 구체적으로, TempIdx(TempIdx)에 의해 할당된 각 어레이는 이전에 디코딩 된 픽처의 필터 세트를 하단 TempIdx와 동일하게 작성할 수 있다. 예를 들어, k 번째 어레이는 K와 동일한 TempIdx와 연관되도록 할당되고, 및 k보다 작거나 동일한 TempIdxfh 픽처의 필터 세트만 포함한다. 특정 픽처를 코딩 한 후, 픽처와 연관된 필터 세트가 동일하거나 더 높은 TempIdx와 연관된 어레이를 업데이트하는 데 사용된다.

[0067] GALF 계수의 시간적 예측은 시그널링 오버헤드를 최소화 하기 위해 인터 코딩 된 프레임에 사용된다. 인트라 프레임의 경우 시간적 예측을 사용할 수 없으며 각 클래스에 16개의 고정 필터 세트가 할당된다. 고정 필터의 사용을 나타내기 위해 각 클래스에 대한 플래그가 표시되고 필요한 경우 선택한 고정 필터의 인덱스가 표시된다. 주어진 클래스에 대해 고정 필터를 선택하더라도, 적응 필터 f(k,l)의 계수는 이 클래스에 대해 여전히 전송될 수 있으며, 이 경우 재구성된 이미지에 적용될 필터의 계수는 두 계수 세트의 합이다. 루마 컴포넌트의 필터링 프로세스는 CU 레벨에서 제어할 수 있다. 플레그는 GALF가 CU의 루마 컴포넌트에 적용되는지 여부를 나타내기 위해 시그널링 된다. 크로마 컴포넌트의 경우 GALF가 적용되었는지 여부에 관계없이 픽처 레벨에서만 표시된다.

[0068] **3.1.4 필터링 프로세스**

[0069]

[0070] 디코더 측에서, 블록에 대해 GALF가 활성화되면 블록 내의 각 샘플 R(i,j)이 필터링되어 아래와 같이 샘플 값 R'(i,j)가 생성되고, 여기서 L은 필터 길이, $f_{m,n}$은 필터 계수, $f_{k,l}$은 디코딩 된 필터 계수를 나타낸다.

[0071] $$R'(i,j) = \sum_{k=-L/2}^{L/2} \sum_{l=-L/2}^{L/2} f(k,l) \times R(i+k, j+l) \quad (10)$$

[0072] **3.1.5 인코더 측 필터 파라미터에 대한 측정 프로세스**

[0073] GALF에 대한 전반적인 인코더 의사 결정 과정은 도 3에 설명되어 있다. 각 CU의 루마 샘플의 경우 인코더는 GALF가 적용되고 적절한 신호 플래그가 슬라이스 헤더에 포함되어 있는지 여부를 결정한다. 크로마 샘플의 경우 필터를 적용하기로 한 결정은 CU 레벨이 아닌 픽처 레벨에 따라 수행된다. 또한, 픽처에 대한 크로마 GALF는 픽처에 대한 루마 GALF가 활성화된 경우에만 확인된다.

[0074] **4 VVC의 형상 변환 기반 적응형 루프 필터링의 예**

[0075] VVC에서 GALF의 현재 디자인은 JEM에 비해 다음과 같은 주요 변경 사항이 있다:

[0076] 1) 적응형 필터 형태가 제거된다. 루마 컴포넌트에는 7x7 필터 형태만 허용되고 크로마 컴포넌트에는 5x5 필터 형태가 허용된다.

[0077] 2) ALF 파라미터의 시간적 예측및 고정 필터의 예측은 모두 제거된다.

[0078] 3) 각 CTU에 대해 ALF가 활성화되어 있는지 비활성화되었는지 여부에 관계없이 하나의 비트 플래그가 시그널링 된다.

[0079] 4) 클래스 인덱스의 계산은 2x2 대신 4x4 레벨에서 수행된다. 또한, JVET-L0147에서 제안된 바와 같이, 도 4a 내지 도 4d에 도시된 바와 같이 ALF 분류를 위한 서브샘플링 된 라플라시안 계산 방법이 활용된다. 보다 구체적으로, 한 블록 내에서 각 샘플에 대해 수평/수직/45 대각선/135도 그레디언트를 계산할 필요가 없다. 대신 1:2 서브 샘플링이 활용된다.

[0080] **5 AVS2의 영역 기반 적응형 루프 필터의 예**

[0081] ALF는 인루프 필터링(in-loop filtering)의 마지막 단계이다. 이 프로세스에는 두 단계가 있다. 첫 번째 단계는 필터 계수 유도(filter coefficient derivation)이다. 필터 계수(filter coefficient)를 훈련시키기 위해, 인코더는 루마 컴포넌트의 재구성된 픽셀을 16개 영역으로 분류하고, 원래 프레임과 재구성된 프레임 사이의 평균 제곱 오류를 최소화하기 위해 위너-홉프 식(wiener-hopf equation)을 사용하여 각 범주에 대해 필터 계수의 한

세트를 훈련한다. 이러한 16개 필터 계수 세트 간의 중복성을 줄이기 위해, 인코더는 비율 왜곡 성능에 기초하여 이들을 적응적으로 병합한다. 최대 16개의 서로 다른 필터 세트를 루마 컴포넌트(luminance component)에 할당할 수 있으며 크로마 컴포넌트(chrominance component)에는 하나만 할당할 수 있다. 두 번째 단계는 프레임 레벨과 LCU 레벨을 모두 포함하는 필터 결정이다. 먼저 인코더는 프레임 레벨 적응형 루프 필터링을 수행할지 여부를 결정한다. 프레임 레벨 ALF가 켜져 있으면 인코더는 LCU 레벨 ALF를 수행할지 여부를 추가로 결정한다.

[0082] ### 5.1 필터 형태(Filter shape)

[0083] AVS-2에 채용된 필터 형상은 루마 및 크로마 컴포넌트 모두에 대해 도 5에 도시된 바와 같이 3X3 정사각형을 중첩한 7X7 십자형이다. 도 5의 각 사각형은 샘플에 해당한다. 따라서 C8 위치의 샘플에 대한 필터링 된 값을 유도하기 위해 총 17개의 샘플이 사용된다. 계수 전송의 오버헤드를 고려하여 점대칭 필터를 사용하여 계수 {C0, C1, . . . , C8}, 필터 계수의 수와 필터링의 곱셈 수를 절반으로 줄이다. 포인트 대칭 필터는 또한 하나의 필터링 된 샘플에 대한 계산의 절반을 줄일 수 있다.

[0084] ### 5.2 영역 기반 적응형 병합

[0085] 다른 코딩 오류를 적용하기 위해 AVS-2는 루마 컴포넌트에 대해 영역 기반 다중 적응형 루프 필터를 채택한다. 루마 컴포넌트는 도 6에 도시된 바와 같이 각각의 기본 영역이 최대 코딩 유닛(LCU) 경계와 정렬되는 16개의 대략 동일한 크기의 기본 영역으로 분할되고, 각 영역에 대해 하나의 위너 필터(Wiener filter)가 유도된다. 더 많은 필터를 사용할수록 더 많은 왜곡이 감소하지만 이러한 계수를 인코딩 하는 데 사용되는 비트는 필터 수와 함께 증가한다. 최고의 비율 왜곡 성능을 달성하기 위해 이러한 영역은 동일한 필터 계수를 공유하는 더 적은 수의 더 큰 영역으로 병합될 수 있다. 병합 프로세스를 단순화하기 위해 각 영역에는 이미지 사전 상관 관계에 기초하여 수정된 힐베르트 차수(Hilbert order)에 따라 인덱스가 할당된다. 연속적인 인덱스가 있는 두 영역은 비율 왜곡 비용에 기초하여 병합될 수 있다.

[0086] 영역 간의 매핑 정보는 디코더에 시그널링되어야 한다. AVS-2에서 기본 영역의 수는 병합 결과를 나타내는 데 사용되며 필터 계수는 영역 순서에 따라 순차적으로 압축된다. 예를 들어 {0, 1}, {2, 3, 4}, {5, 6, 7, 8, 9} 및 왼쪽 기본 영역이 각각 하나의 영역으로 병합될 때 이 병합 맵을 나타내기 위해 3개의 정수만, 즉 2, 3, 5, 코딩 된다.

[0087] ### 5.3 부가 정보의 신호

[0088] 다중 스위치 플래그도 사용된다. 시퀀스 전환 플래그인 adaptive_loop_filter_enable은 적응형 루프 필터가 전체 시퀀스에 적용되는지 여부를 제어하는 데 사용된다. 이미지 전환 플래그인 picture_alf_enble[i]은 해당 i번째 이미지 구성요소에 ALF가 적용되는지 여부를 제어한다. picture_alf_enble[i]이 활성화된 경우에만 해당 컬러 컴포넌트에 대한 해당 LCU 레벨 플래그 및 필터 계수가 전송된다. LCU 레벨 플래그인 lcu_alf_enable[k]는 해당 k번째 LCU에 대해 ALF가 활성화되는지 여부를 제어하고 슬라이스 데이터에 인터리브 된다. 다른 레벨의 규제 플래그(regulated fag)의 결정은 모두 비율 왜곡 비용에 기초한다. 높은 유연성은 ALF가 코딩 효율성을 훨씬 더 크게 향상시킨다.

[0089] 일부 실시예에서, 그리고 루마 컴포넌트에 대해 최대 16개의 필터 계수 세트가 있을 수 있다.

[0090] 일부 실시예에서, 그리고 각각의 크로마 컴포넌트(Cb 및 Cr)에 대해, 필터 계수의 한 세트가 전송될 수 있다.

[0091] ## 6 VTM-4의 GALF

[0092] VTM4.0에서는 적응형 루프 필터링의 필터링 프로세스가 다음과 같이 수행된다:

[0093] $$O(x,y) = \sum_{(i,j)} w(i,j).I(x+i,y+j) \quad (11)$$

[0094] 샘플 I(x+I,y+j)은 입력 샘플이고, O(x,y)는 필터링 된 출력 샘플(예를 들어, 필터 결과)이고, w(I,j)는 필터 계수를 나타낸다. 실제로, VTM4.0에서는 고정 점 정밀 컴퓨테이션(fixed point precision computation)을 위해 정수 산술을 사용하여 구현된다:

[0095] $$O(x,y) = \left(\sum_{i=-\frac{L}{2}}^{\frac{L}{2}}\sum_{j=-\frac{L}{2}}^{\frac{L}{2}} w(i,j).I(x+i,y+j) + 64\right) \gg 7 \quad (12)$$

[0096] 여기서 L은 필터 길이를 나타내고, $w(i,j)$ 는 고정 점 정밀도에서 필터 계수이다.

[0097] 7 JVET-N0242의 비선형 적응형 루프 필터링링(ALF)

[0098] 7.1 필터링 재형성

[0099] 식(11)은 코딩 효율성에 영향을 주지 않고 다음 식에서 다시 수식화할 수 있다:

[0100] $$O(x,y) = I(x,y) + \sum_{(i,j)\neq(0,0)} w(i,j).\big(I(x+i,y+j) - I(x,y)\big) \quad (13)$$

[0101] 여기서, w(i,j) 식(11)과 동일한 필터 계수이다[식 (11)에는 $1-\sum_{(i,\ j)\neq(0,0)}$w(i,j)과 동일하지만 식(13)에서 1과 동일한 w(0,0)을 제외].

[0102] 7.2 수정된 필터

[0103] 상기 필터 식(13)을 사용하여, 필터링되는 현재 샘플 값(I(x,y))과 너무 다른 이웃 샘플 값(I(x+i,y+j))의 영향을 줄이기 위해 간단한 클리핑 함수을 사용하여 ALF를 보다 효율적으로 만들기 위해 비선형성을 쉽게 도입할 수 있다.

[0104] 이 제안서에서 ALF 필터는 다음과 같이 수정된다:

[0105] $$O'(x,y) = I(x,y) + \sum_{(i,j)\neq(0,0)} w(i,j) \times K(I(x+i,y+j) - I(x,y), k(i,j)) \quad (14)$$

[0106] 여기서, K(d,b)=min(b, max(-b,d))는 클리핑 함수이고, k(i,j)는 (i,j) 필터 계수에 따라 달라지는 파라미터를 클리핑한다. 인코더는 최적의 k(i,j)를 찾기 위해 최적화를 수행한다. 정수 정밀도의 구현을 위해 반올림으로 $\sum_{(i,j)\neq(0,0)} w(i,j) \times K(I(x+i,y+j) - I(x,y), k(i,j))$ 시프트가 적용된다.

[0107] JVET-N0242 구현에서, 클리핑 파라미터 k(i,j)는 각 ALF 필터에 대해 지정되며, 필터 계수당 하나의 클리핑 값이 시그널링 된다. 이는 루마 필터당 비트스트림에서 최대 12개의 클리핑 값이 시그널링 될 수 있으며 크로마 필터의 클리핑 값은 최대 6개까지 시그널링 될 수 있음을 의미한다.

[0108] 시그널링 비용과 인코더 복잡성을 제한하기 위해 클리핑 값의 평가를 가능한 값의 작은 세트로 제한한다. 제안서에서는 인터(INTER) 및 인트라(INTRA) 타일 그룹에 대해 동일한 4개의 고정 값만 사용한다.

[0109] 루마의 경우 크로마보다 지역 별 차이의 변화가 더 높기 때문에 루마 필터와 크로마 필터에 두 개의 서로 다른 세트를 사용한다. 또한 각 세트에 최대 샘플 값(10비트 비트 깊이에 대해 1024)을 포함하므로 필요하지 않은 경우 클리핑을 비활성화할 수 있다.

[0110] JVET-N0242 테스트에 사용되는 클리핑 값 세트는 표 2에 제공된다. 4개의 값은 대수 영역에서 루마(Luma)의 경우 샘플 값의 전체 범위(10비트로 코딩됨)와 크로마(Chroma)의 경우 4에서 1024까지의 범위를 대략 균등하게 분할하여 선택되었다.

[0111] 더 정확하게, 클리핑 값의 루마 테이블은 다음과 같은 수식에 의해 얻어졌다:

[0112] AlfClipL = $\left\{\text{round}\left(\left((M)^{\frac{1}{N}}\right)^{N-n+1}\right) \text{ for } n \in 1..N\right\}$, M=$2^{10}$ 및 N=4.

[0113] 마찬가지로 클리핑 값의 크로마 테이블은 다음 수식에 따라 가져온다:

[0114] AlfClipC = $\left\{\text{round}\left(A.\left(\left(\frac{M}{A}\right)^{\frac{1}{N-1}}\right)^{N-n}\right) \text{ for } n \in 1..N\right\}$, M=$2^{10,}$, N=4 및 A=4.

표 2

[0115] 승인된 클리핑 값

	인트라/인터 타일 그룹
루마	{1024, 181, 32, 6}
크로마	{1024, 181, 25, 4}

[0116] 선택한 클리핑 값은 위의 표 2의 클리핑 값 인덱스에 대응하는하는 Golomb 인코딩 스키마를 사용하여

"alf_data" 신택스 요소에 코딩 된다. 이 인코딩 구성표는 필터 인덱스에 대한 인코딩 스키마와 동일하다.

[0117] 8 JVET-N0415의 CTU 기반 ALF

[0118] 슬라이스 레벨 시간적 필터. VTM4에는 적응형 파라미터 세트 (Adaptive Parameter Set) (APS)가 채택되었다. 각 APS에는 신호된 ALF 필터 세트가 포함되어 있으며 최대 32개의 APS가 지원된다. 제안서에서는 슬라이스 레벨의 시간적 필터를 테스트한다. 타일 그룹은 APS의 ALF 정보를 재사용하여 오버헤드를 줄일 수 있다. APS는 선입선출(FIFO) 버퍼로 업데이트된다.

[0119] CTB 기반 ALF. 루마 컴포넌트(luma component)의 경우 ALF가 루마 CTB에 적용될 때 16개의 고정된, 5개의 시간적 또는 1개의 시그널링 된 필터 세트(signaled filter set) 중에서 선택이 표시된다. 필터 세트 인덱스만 신호된다. 하나의 슬라이스에 대해 새로운 25개 필터 세트 하나만 신호를 보낼 수 있다. 슬라이스에 대해 새 세트가 신호되면 동일한 슬라이스의 모든 luma CTB가 해당 세트를 공유한다. 고정 필터 세트는 새로운 슬라이스 레벨 필터 세트를 예측하는 데 사용할 수 있으며 루마 CTB에 대한 후보 필터 세트로도 사용할 수 있다. 필터의 수는 총 64개이다.

[0120] 크로마 컴포넌트의 경우 크로마 CTB에 ALF가 적용될 때 슬라이스에 대해 새로운 필터가 시그널링되면 CTB는 새로운 필터를 사용하고, 그렇지 않으면 시간적 확장성 제약을 만족하는 가장 최근의 시간적 크로마 필터를 적용한다.

[0121] 슬라이스 레벨 시간적 필터로서 APS는 선입선출(FIFO) 버퍼로 업데이트된다.

[0122] 사양.

[0123] 다음 텍스트는 {{고정 필터}}, [[임시 필터]] 및 ((CTB 기반 필터 인덱스))가 있는 JVET-K1001-v6에 기초하여 수정되었고, 즉, 이중 중괄호, 이중 중괄호 및 이중 괄호 사용한다.

[0124] 7.3.3.2 적응형 루프 필터 데이터 신택스

[0125]

alf_data( ) {	기술자
alf_chroma_idc	tu(v)
((alf_signal_new_filter_luma	u(1)
if(alf_signal_new_filter_luma > 0) {))	
{{ **alf_luma_use_fixed_filter_flag**	u(1)
if(alf_luma_use_fixed_filter_flag){	
alf_luma_fixed_filter_set_index	tb(v)
alf_luma_fixed_filter_usage_pattern	u(1)
if (alf_luma_fixed_filter_usage_pattern > 0)	
for (i = 0; i < NumAlfFilters; i++)	
alf_luma_fixed_filter_usage[i]	u(1) }}
((**alf_num_available_temporal_filter_sets_luma**	tb(1)))
alf_luma_num_filters_signalled_minus1	tb(v)
alf_luma_type_flag	u(1)
if(alf_luma_num_filters_signalled_minus1 > 0) {	
for(filtIdx = 0; filtIdx < NumAlfFilters; filtIdx++)	
alf_luma_coeff_delta_idx[filtIdx]	tu(v)
}	
alf_luma_coeff_delta_flag	u(1)
if (!alf_luma_coeff_delta_flag && alf_luma_num_filters_signalled_minus1 > 0)	
alf_luma_coeff_delta_prediction_flag	u(1)
alf_luma_min_eg_order_minus1	tu(v)
for(i = 0; i < (alf_luma_type_flag == 1) ? 2 : 3; i++)	
alf_luma_eg_order_increase_flag[i]]	u(1)

if (alf_luma_coeff_delta_flag) {	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++)	
alf_luma_coeff_flag[sigFiltIdx]	u(1)
}	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++){	
if (alf_luma_coeff_flag[sigFiltIdx]) {	
for (j = 0; j < (alf_luma_type_flag == 1) ? 6 : 12; j++) {	
alf_luma_coeff_delta_abs[sigFiltIdx][j]	uek(v)
if(alf_luma_coeff_delta_abs[sigFiltIdx][j])	
alf_luma_coeff_delta_sign[sigFiltIdx][j]	u(1)
}	
}	
}	
((}))	
if (alf_chroma_idc > 0) {	
((**alf_signal_new_filter_chroma**	u(1)
if (alf_signal_new_filter_chroma){))	
alf_chroma_min_eg_order_minus1	tu(v)
for(i = 0; i < 2; i++)	
alf_chroma_eg_order_increase_flag[i]	u(1)
for(j = 0; j < 6; j++) {	
alf_chroma_coeff_abs[j]	uek(v)
if(alf_chroma_coeff_abs[j] > 0)	
alf_chroma_coeff_sign[j]	u(1)
}	
((}))	
}	
}	

[0126]

[0127] 7.3.4.2 코딩 트리 유닛 신택스

coding_tree_unit( ) {	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_alf_enable_flag){	
alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
if (alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] {	
((coding_alf_ctu_filter_set_index(CtbAddrInRs)))	
}	
if(alf_chroma_idc == 1 || alf_chroma_idc == 3)	
alf_ctb_flag[1][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
if(alf_chroma_idc == 2 || alf_chroma_idc == 3)	
alf_ctb_flag[2][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
}	
if(slice_type = = I && qtbtt_dual_tree_intra_flag) {	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbLog2SizeY, 0)	
else	
coding_quadtree(xCtb, yCtb, CtbLog2SizeY, 0, SINGLE_TREE)	
}	

[0128]

((coding_alf_ctu_filter_set_index(CtbAddrInRs, slice_type){))	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
alf_use_new_filter	ae(1)
if (alf_use_new_filter == 0){	
alf_use_fixed_filter	ae(1)
}	
if (alf_use_new_filter){	
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = 16	
}	
else if (alf_use_fixed_filter){	
alf_fixed_filter_index	tb(v)
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = alf_fixed_filter_index	
}	
else{	
alf_temporal_index	tb(v)
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = 16 + alf_temporal_index	
}	

[0129]

[0130] 7.4.4.2 적응형 루프 필터 데이터 의미론

[0131] 1과 동일한 ((alf_signal_new_filter_luma))는 새로운 루마 필터 세트가 시그널링됨을 지정한다. 0과 동일한 alf_signal_new_filter_luma는 새로운 루마 필터 세트가 시그널링되지 않음을 지정한다. 존재하지 않을 때는 0이다.

[0132] 1과 동일한 {{ alf_luma_use_fixed_filter_flag }}는 고정 필터 세트가 적응형 루프 필터를 시그널링 하는 데 사용됨을 지정한다. 0과 동일한 alf_luma_use_fixed_filter_flag는 고정 필터 세트가 적응형 루프 필터를 시그널링 하는 데 사용되지 않음을 지정한다.

[0133] {{ alf_luma_fixed_filter_set_index }}는 고정 필터 세트 인덱스를 지정한다. 0...15일 수 있다.

[0134] 0과 동일한 {{ alf_luma_fixed_filter_usage_pattern }}는 모든 새 필터가 고정 필터를 사용하도록 지정한다. 1과 동일한 alf_luma_fixed_filter_usage_pattern은 일부 새 필터는 고정 필터를 사용하고 나머지는 사용하지 않음을 지정한다.

[0135] 1과 동일한 {{ alf_luma_fixed_filter_usage[i] }}는 i번째 필터가 고정 필터를 사용하도록 지정한다. 0과 동일한 alf_luma_fixed_filter_usage[i]는 i번째 필터가 고정 필터를 사용하지 않음을 지정한다. 존재하지 않는 경우 1로 유추된다.

[0136] 1과 동일한 ((alf_signal_new_filter_chroma)) 새로운 크로마 필터가 시그널링되는 것을 지정한다. 0과 동일한 alf_signal_new_filter_chroma는 새로운 크로마 필터가 시그널링되지 않음을 지정한다.

[0137] ((alf_num_available_temporal_filter_sets_luma))는 현재 슬라이스에 사용할 수 있는 사용 가능한 시간적 필터 세트의 수를 지정하며 0..5부터 가능한다. 존재하지 않을 때는 0이다.

[0138] 변수 alf_num_available_filter_sets는 16 + alf_signal_new_filter_luma + alf_num_available_temporal_filter_sets_luma로 유도된다.

[0139] ((alf_signal_new_filter_luma 가 1 이면 다음 과정))

[0140] sigFiltIdx = 0..alf_luma_num_filters_signalled_minus1, j = 0..11인 변수 filterCoefficients[sigFiltIdx][j]는 다음과 같이 초기화된다:

$$\text{filterCoefficients}[\text{sigFiltIdx}][j] = \text{alf\_luma\_coeff\_delta\_abs}[\text{sigFiltIdx}][j] * (1 - 2 * \text{alf\_luma\_coeff\_delta\_sign}[\text{sigFiltIdx}][j]) \quad (7\text{-}50)$$

[0141]

[0142] alf_luma_coeff_delta_prediction_flag가 1일 때 sigFiltIdx = 1..alf_luma_num_filters_signalled_minus1 및

j = 0..11인 filterCoefficients[sigFiltIdx][j]는 다음과 같이 수정된다:

[0143] filterCoefficients[sigFiltIdx][j] += filterCoefficients[sigFiltIdx - 1][j] (7-51)

[0144] filtIdx = 0..NumAlfFilters - 1 및 j = 0..11인 요소 AlfCoeffL[filtIdx][j]가 있는 루마 필터 계수 AlfCoeffL은 다음과 같이 유도된다:

[0145] $AlfCoeff_L$[filtIdx][j] = filterCoefficients[alf_luma_coeff_delta_idx[filtIdx]][j] (7-52)

[0146] {{ alf_luma_use_fixed_filter_flag가 1이고 alf_luma_fixed_filter_usage[filtidx]가 1이면 다음이 적용된다:

[0147] AlfCoeffL[filtIdx][j] = AlfCoeffL[filtIdx][j] + AlfFixedFilterCoeff[AlfClassToFilterMapping[alf_luma_fixed_filter_index][filtidx]][j] }}

[0148] filtIdx = 0..NumAlfFilters - 1에 대한 마지막 필터 계수 AlfCoeffL[filtIdx][12]는 다음과 같이 유도된다:

[0149] AlfCoeffL[filtIdx][12] = 128 - k (AlfCoeffL[filtIdx][k] << 1), k = 0..11 (7-53)

[0150] filtIdx = 0..NumAlfFilters - 1, j = 0..11인 AlfCoeffL[filtIdx][j] 값이 -2^7 ~ 2^7 - 1 범위에 있어야 하고 AlfCoeffL[filtIdx][12]의 값은 0에서 2^8 - 1까지의 범위에 있어야 한다.

[0151] ((루마 필터 계수)) 요소 $AlfCoeff_{LumaAll}$[filtSetIdx][filtIdx][j] 를 갖는 $AlfCoeff_{LumaAll}$, filtSetIdx = 0..15, filtSetIdx = 0..NumAlfFilters - 1 및 j = 0..1은 다음과 같이 유도된다

[0152] $AlfCoeff_{LumaAll}$[filtSetIdx][filtIdx][j] = {{ AlfFixedFilterCoeff[AlfClassToFilterMapping[}} filtSetIdx {{][filtidx]][j] }}

[0153] ((루마 필터 계수)) 요소 $AlfCoeff_{LumaAll}$[filtSetIdx][filtIdx][j] 를 갖는 $AlfCoeff_{LumaAll}$, filtSetIdx = 16, filtSetIdx = 0..NumAlfFilters - 1 및 j = 0..12가 다음과 같이 유도된다:

[0154] 변수 Nearest_temporal_index는 -1로 초기화된다. Tid는 현재 슬라이스의 시간적 계층 인덱스이다.

[0155] ((if **alf_signal_new_filter_luma** is 1))

[0156] $AlfCoeff_{LumaAll}$[16][filtIdx][j] = $AlfCoeff_L$[filtIdx][j]

[0157] ((그렇지 않으면 다음 프로세스가 호출됨))

[0158] (i = Tid, i >= 0, i--)

[0159] {

[0160] for (k = 0; k < temp_size_L; k++)

[0161] {

[0162] if ($temp_{Tid\_L}$[k] == i)

[0163] {

[0164] closest_temporal_index is set as k;

[0165] break;

[0166] }

[0167] }

[0168] }

[0169] $AlfCoeff_{LumaAll}[\ 16\ ][\ filtIdx\ ][\ j\ ] = Temp_L[\ closest\_temporal\_index\ ][\ filtIdx\ ][\ j\ ]$

[0170] ((루마 필터 계수)) 요소 $AlfCoeff_{LumaAll}[\ filtSetIdx\ ][\ filtIdx\ ][\ j\ ]$을 갖는 $AlfCoeff_{LumaAll}$, filtSetIdx = 17.. alf_num_available_filter_sets-1, filtSetIdx = 0..NumAlf 및 j는 유도됨 따르다

[0171] i = 17;

[0172] for (k = 0; k < temp_size_L and i < alf_num_available_filter_sets; j++)

[0173] {

[0174] if ($tempT_{id\_L}[\ k\ ]$ <= Tid 및 k가 Nearest_temporal_index와 같지 않음)

[0175] {

[0176] $AlfCoeff_{LumaAll}[\ i\ ][\ filtIdx\ ][\ j\ ] = Temp_L[\ k\ ][\ filtIdx\ ][\ j\ ];$

[0177] i++;

[0178] }

[0179] }

[0180] {{ AlfFixedFilterCoeff }}[64][13] =

[0181] {

[0182] { 0, 0, 2, -3, 1, -4, 1, 7, -1, 1, -1, 5, 112 },

[0183] { 0, 0, 0, 0, 0, -1, 0, 1, 0, 0, -1, 2, 126 },

[0184] { 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 126 },

[0185] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1, 128 },

[0186] { 2, 2, -7, -3, 0, -5, 13, 22, 12, -3, -3, 17, 34 },

[0187] {-1, 0, 6, -8, 1, -5, 1, 23, 0, 2, -5, 10, 80 },

[0188] { 0, 0, -1, -1, 0, -1, 2, 1, 0, 0, -1, 4, 122 },

[0189] { 0, 0, 3, -11, 1, 0, -1, 35, 5, 2, -9, 9, 60 },

[0190] { 0, 0, 8, -8, -2, -7, 4, 4, 2, 1, -1, 25, 76 },

[0191] { 0, 0, 1, -1, 0, -3, 1, 3, -1, 1, -1, 3, 122 },

[0192] { 0, 0, 3, -3, 0, -6, 5, -1, 2, 1, -4, 21, 92 },

[0193] {-7, 1, 5, 4, -3, 5, 11, 13, 12, -8, 11, 12, 16 },

[0194] {-5, -3, 6, -2, -3, 8, 14, 15, 2, -7, 11, 16, 24 },

[0195] { 2, -1, -6, -5, -2, -2, 20, 14, -4, 0, -3, 25, 52 },

[0196] { 3, 1, -8, -4, 0, -8, 22, 5, -3, 2, -10, 29, 70 },

[0197] { 2, 1, -7, -1, 2, -11, 23, -5, 0, 2, -10, 29, 78 },

[0198] {-6, -3, 8, 9, -4, 8, 9, 7, 14, -2, 8, 9, 14 },

[0199] { 2, 1, -4, -7, 0, -8, 17, 22, 1, -1, -4, 23, 44 },

[0200] { 3, 0, -5, -7, 0, -7, 15, 18, -5, 0, -5, 27, 60 },

[0201] { 2, 0, 0, -7, 1, -10, 13, 13, -4, 2, -7, 24, 74 },

[0202] { 3, 3, -13, 4, -2, -5, 9, 21, 25, -2, -3, 12, 24 },

[0203] {-5, -2, 7, -3, -7, 9, 8, 9, 16, -2, 15, 12, 14 },

[0204] { 0, -1, 0, -7, -5, 4, 11, 11, 8, -6, 12, 21, 32 },

[0205] { 3, -2, -3, -8, -4, -1, 16, 15, -2, -3, 3, 26, 48 },

[0206] { 2, 1, -5, -4, -1, -8, 16, 4, -2, 1, -7, 33, 68 },

[0207] { 2, 1, -4, -2, 1, -10, 17, -2, 0, 2, -11, 33, 74 },

[0208] { 1, -2, 7, -15, -16, 10, 8, 8, 20, 11, 14, 11, 14 },

[0209] { 2, 2, 3, -13, -13, 4, 8, 12, 2, -3, 16, 24, 40 },

[0210] { 1, 4, 0, -7, -8, -4, 9, 9, -2, -2, 8, 29, 54 },

[0211] { 1, 1, 2, -4, -1, -6, 6, 3, -1, -1, -3, 30, 74 },

[0212] {-7, 3, 2, 10, -2, 3, 7, 11, 19, -7, 8, 10, 14 },

[0213] { 0, -2, -5, -3, -2, 4, 20, 15, -1, -3, -1, 22, 40 },

[0214] { 3, -1, -8, -4, -1, -4, 22, 8, -4, 2, -8, 28, 62 },

[0215] { 0, 3, -14, 3, 0, 1, 19, 17, 8, -3, -7, 20, 34 },

[0216] { 0, 2, -1, -8, 3, -6, 5, 21, 1, 1, -9, 13, 84 },

[0217] {-4, -2, 8, 20, -2, 2, 3, 5, 21, 4, 6, 1, 4 },

[0218] { 2, -2, -3, -9, -4, 2, 14, 16, 3, -6, 8, 24, 38 },

[0219] { 2, 1, 5, -16, -7, 2, 3, 11, 15, -3, 11, 22, 36 },

[0220] { 1, 2, 3, -11, -2, -5, 4, 8, 9, -3, -2, 26, 68 },

[0221] { 0, -1, 10, -9, -1, -8, 2, 3, 4, 0, 0, 29, 70 },

[0222] { 1, 2, 0, -5, 1, -9, 9, 3, 0, 1, -7, 20, 96 },

[0223] {-2, 8, -6, -4, 3, -9, -8, 45, 14, 2, -13, 7, 54 },

[0224] { 1, -1, 16, -19, -8, -4, -3, 2, 19, 0, 4, 30, 54 },

[0225] { 1, 1, -3, 0, 2, -11, 15, -5, 1, 2, -9, 24, 92 },

[0226] { 0, 1, -2, 0, 1, -4, 4, 0, 0, 1, -4, 7, 120 },

[0227] { 0, 1, 2, -5, 1, -6, 4, 10, -2, 1, -4, 10, 104 },

[0228] { 3, 0, -3, -6, -2, -6, 14, 8, -1, -1, -3, 31, 60 },

[0229] { 0, 1, 0, -2, 1, -6, 5, 1, 0, 1, -5, 13, 110 },

[0230] { 3, 1, 9, -19, -21, 9, 7, 6, 13, 5, 15, 21, 30 },

[0231] { 2, 4, 3, -12, -13, 1, 7, 8, 3, 0, 12, 26, 46 },

[0232] { 3, 1, -8, -2, 0, -6, 18, 2, -2, 3, -10, 23, 84 },

[0233] { 1, 1, -4, -1, 1, -5, 8, 1, -1, 2, -5, 10, 112 },

[0234] { 0, 1, -1, 0, 0, -2, 2, 0, 0, 1, -2, 3, 124 },

[0235] { 1, 1, -2, -7, 1, -7, 14, 18, 0, 0, -7, 21, 62 },

[0236] { 0, 1, 0, -2, 0, -7, 8, 1, -2, 0, -3, 24, 88 },

[0237] { 0, 1, 1, -2, 2, -10, 10, 0, -2, 1, -7, 23, 94 },

[0238] { 0, 2, 2, -11, 2, -4, -3, 39, 7, 1, -10, 9, 60 },

[0239] { 1, 0, 13, -16, -5, -6, -1, 8, 6, 0, 6, 29, 58 },

[0240] { 1, 3, 1, -6, -4, -7, 9, 6, -3, -2, 3, 33, 60 },

[0241] { 4, 0, -17, -1, -1, 5, 26, 8, -2, 3, -15, 30, 48 },

[0242] { 0, 1, -2, 0, 2, -8, 12, -6, 1, 1, -6, 16, 106 },

[0243] { 0, 0, 0, -1, 1, -4, 4, 0, 0, 0, -3, 11, 112 },

[0244] { 0, 1, 2, -8, 2, -6, 5, 15, 0, 2, -7, 9, 98 },

[0245] { 1, -1, 12, -15, -7, -2, 3, 6, 6, -1, 7, 30, 50 },

[0246] };

[0247] {{ AlfClassToFilterMapping }}[16][25] =

[0248] {

[0249] { 8, 2, 2, 2, 3, 4, 53, 9, 9, 52, 4, 4, 5, 9, 2, 8, 10, 9, 1, 3, 39, 39, 10, 9, 52 },

[0250] { 11, 12, 13, 14, 15, 30, 11, 17, 18, 19, 16, 20, 20, 4, 53, 21, 22, 23, 14, 25, 26, 26, 27, 28, 10 },

[0251] { 16, 12, 31, 32, 14, 16, 30, 33, 53, 34, 35, 16, 20, 4, 7, 16, 21, 36, 18, 19, 21, 26, 37, 38, 39 },

[0252] { 35, 11, 13, 14, 43, 35, 16, 4, 34, 62, 35, 35, 30, 56, 7, 35, 21, 38, 24, 40, 16, 21, 48, 57, 3 },

[0253] { 11, 31, 32, 43, 44, 16, 4, 17, 34, 45, 30, 20, 20, 7, 5, 21, 22, 46, 40, 47, 26, 48, 63, 58, 10 },

[0254] { 12, 13, 50, 51, 52, 11, 17, 53, 45, 9, 30, 4, 53, 19, 0, 22, 23, 25, 43, 44, 37, 27, 28, 10, 55 },

[0255] { 30, 33, 62, 51, 44, 20, 41, 56, 34, 45, 20, 41, 41, 56, 5, 30, 56, 38, 40, 47, 11, 37, 42, 57, },

[0256] { 35, 11, 23, 32, 14, 35, 20, 4, 17, 18, 21, 20, 20, 20, 4, 16, 21, 36, 46, 25, 41, 26, 48, 49, 58 },

[0257] { 12, 31, 59, 59, 3, 33, 33, 59, 59, 52, 4, 33, 17, 59, 55, 22, 36, 59, 59, 60, 22, 5 36, 59, 25, 5 },

[0258] { 31, 25, 15, 60, 60, 22, 17, 19, 55, 55, 20, 20, 53, 19, 55, 22, 46, 25, 43, 60, 37, 28, 15, 55, },

[0259] { 12, 31, 32, 50, 51, 11, 33, 53, 19, 45, 16, 4, 4, 53, 5, 22, 36, 18, 25, 43, 26, 27, 27, 28, 10 },

[0260] { 5, 2, 44, 52, 3, 4, 53, 45, 9, 3, 4, 56, 5, 0, 2, 5, 10, 47, 52, 3, 63, 39, 10, 9, 52 },

[0261] { 12, 34, 44, 44, 3, 56, 56, 62, 45, 9, 56, 56, 7, 5, 0, 22, 38, 40, 47, 52, 48, 57, 39, 10, 9 },

[0262] { 35, 11, 23, 14, 51, 35, 20, 41, 56, 62, 16, 20, 41, 56, 7, 16, 21, 38, 24, 40, 26, 26, 42, 57, },

[0263] { 33, 34, 51, 51, 52, 41, 41, 34, 62, 0, 41, 41, 56, 7, 5, 56, 38, 38, 40, 44, 37, 42, 57, 39, 10 },

[0264] { 16, 31, 32, 15, 60, 30, 4, 17, 19, 25, 22, 20, 4, 53, 19, 21, 22, 46, 25, 55, 26, 48, 63, 58, 55 }, };

[0265] ((alf_signal_new_filter_chroma 가 1 이면 다음 과정))

[0266] j = 0..5인 크로마 필터 계수 AlfCoeffC[j]는 다음과 같이 유도된다:

[0267] AlfCoeffC[j] = alf_chroma_coeff_abs[j] * (1 - 2 * alf_chroma_coeff_sign[j]) (7-57)

[0268] j = 6에 대한 마지막 필터 계수는 다음과 같이 유도된다:

[0269] AlfCoeffC[6] = 128 - k (AlfCoeffC[k] << 1), k = 0..5 (7-58)

[0270] j = 0..5인 AlfCoeffC[j] 값이 -27 - 1 ~ 27 - 1 범위에 있어야 하고 AlfCoeffC[6] 값이 0에서 28 - 1까지의 범위이다.

[0271] 그렇지 않으면 (((alf_signal_new_filter_chroma is 0))) 다음이 호출된다.

[0272] (i = Tid, i >= 0, i--)

[0273] {

[0274] for (k = 0; k < temp_size_C; k++)

[0275] {

[0276] if ($temp_{Tid\_C}$[k] == i)

[0277] {

[0278] closest_temporal_index is set as k;

[0279] break;

[0280] }

[0281] }

[0282] }

[0283] j = 0..6인 크로마 필터 계수 AlfCoeffC[j]는 다음과 같이 유도된다:

[0284] $AlfCoeff_C$[j] = $Temp_C$[closest_temporal_index][j]

[0285] 7.4.5.2 트리 유닛 시맨틱 코딩

[0286] **((alf_luma_ctb_filter_set_index**[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]))는 위치(xCtb, yCtb)에서 luma CTB의 필터 세트 인덱스를 지정한다.

[0287] **1과 동일한 ((alf_use_new_filter))는** alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]가 16임을 지정한다.

[0288] 0과 동일한 alf_use_fixed_filter는 alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]가 16이 아님을 지정한다.

[0289] ((alf_use_fixed_filter))는 1과 동일하며 고정 필터 세트 중 하나가 사용됨을 지정한다. 0과 동일한 alf_use_fixed_filter은 현재 루마 CTB가 고정 필터 세트를 사용하지 않음을 지정한다.

[0290] ((alf_fixed_filter_index)) 고정 필터 세트 인덱스를 지정하며 0에서 15 사이일 수 있다.

[0291] ((alf_temporal_index))는 0에서 alf_num_available_temporal_filter_sets_luma - 1까지일 수 있는 시간 필터 세트 인덱스를 지정한다.

[0292] [[8.5.1 일반]]

[0293] 1. sps_alf_enabled_flag가 1이면 다음이 적용된다:

[0294] - [[8.5.4.5절에 명시된 임시 필터 새로 고침 프로세스가 호출된다.]]

[0295] - 8.5.4.5 절에 지정된 적응형 루프 필터 프로세스는 재구성된 픽처 샘플 어레이 SL, SCb 및 SCr을 입력으로 사용하고 샘플 적응형 오프셋 후 수정된 재구성된 픽처 샘플 어레이 S'L, S'Cb 및 S'Cr을 출력으로 사용하여 호출된다.

[0296] - 어레이 S'L, S'Cb 및 S'Cr은 각각 어레이 SL, SCb 및 SCr(디코딩 된 화상을 나타냄)에 할당된다.

[0297] - [[8.5.4.6절에 명시된 임시 필터 업데이트 프로세스가 호출된다.]]

[0298] ((8.5.4.2 루마 샘플에 대한 코딩 트리 블록 필터링 프로세스))

[0299] filtIdx[x][y]로 지정된 필터에 해당하는 루마 필터 계수 f[j]의 배열은 j = 0..12일 때 다음과 같이 유도된다:

[0300] f[j] = ((AlfCoeffLumaAll))[alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]]][filtIdx[x][y]][j] (8-732)

[0301] [[8.5.4.5 임시 필터 새로 고침]]

[0302] 아래 조건 중 하나라도 참일 경우,

[0303] - 현재 픽처는 IDR 픽처다.

[0304] - 현재 픽처는 BLA 픽처다

[0305] - 디코딩 순서에서, 현재 픽처는 POC가 마지막으로 디코딩 된 IRAP 픽처의 POC보다 큰 제1 픽처, 즉 선행 픽처 후 및 후행 픽처 전이다.

[0306]

[0307] 그런 다음 temp_size_L 및 temp_size_C가 0으로 설정된다.

[0308] [[8.5.4.6 임시 필터 업데이트]]

[0309] slice_alf_enabled_flag가 1이고 alf_signal_new_filter_luma가 1이면 다음이 적용된다.

[0310] 루마 시간적 필터 버퍼 크기 temp_size_L < 5인 경우 temp_size_L = temp_size_L + 1이다.

[0311] TempL[i][j][k] i = temp_size_L - 1 ... 1, j = 0... NumAlfFilters-1 및 k = 0...12는 다음과 같이 업데이트된다.

[0312] TempL[i][j][k] = TempL[i-1][j][k]

[0313] TempL[0][j][k] j = 0... NumAlfFilters-1 및 k = 0..12는 다음과 같이 업데이트된다.

[0314] 온도[0][j][k] = AlfCoeffL[j][k]

[0315] TempTid_L[i] i = temp_size_L - 1...1은 다음과 같이 업데이트된다.

[0316] TempTid_L[i] = TempTid_L[i-1]

[0317] TempTid_L[0]은 현재 슬라이스의 시간적 레이어 인덱스 Tid로 설정된다.

[0318] alf_chroma_idx가 0이 아니고 alf_signal_new_filter_chroma가 1이면 다음이 적용된다.

[0319] Tempc[i][j] i = temp_size_c - 1 ... 1 및 j = 0...6인 경우 다음과 같이 업데이트된다.

[0320] Tempc[i][j] = Tempc[i-1][j]

[0321] j = 0... 6인 Tempc[0][j]는 다음과 같이 업데이트된다.

[0322] 온도[0][j] = AlfCoeffC [j]

[0323] TempTid_C[i] i = temp_size_C - 1...1은 다음과 같이 업데이트된다.

[0324] TempTid_C[i] = TempTid_C[i-1]

[0325] TempTid_C[0]은 현재 슬라이스의 Tid로 설정된다.

[0326] 표 9-4 신택스 요소 및 관련 이진화

신택스 구조	신택스 요소	이진화	
		프로세스	입력 파라미터
coding_tree_unit( )	alf_ctb_flag[][][]	FL	cMax = 1
	((alf_use_new_filter_set	FL	cMax = 1
	alf_used_fixed_filter	FL	cMax = 1
	alf_fixed_filter_index	TB	cMax = 15
	alf_temporal_index	TB	cMax = alf_num_available_temporal_filter_sets_luma - 1))

[0327]

[0328] 표 9-10 컨텍스트 코딩 된 빈이 있는 신택스 요소에 ctxInc 할당

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
((alf_use_new_filter_set	0	na	na	na	na	na
alf_used_fixed_filter	0	na	na	na	na	na))

[0329]

[0330] ## 9 JVET-M0427의 인루프 재형성(ILR)

[0331] [0001] 인루프 재형성(ILR)의 기본 아이디어는 원래(제1도메인에서) 신호(예측/재구성 신호)를 제2 도메인(재형성된 도메인)으로 변환하는 것이다.

[0332] [0002] 인루프 루마 재형성기(luma reshaper)는 한 쌍의 룩업 테이블(LUT)로 구현되지만 두 LUT 중 하나만 시그널링 된다면 다른 하나는 신호를 받은 LUT에서 컴퓨팅 될 수 있다. 각 LUT는 1차원 10비트 1024 항목 매핑 테이블(1D-LUT)이다. 하나의 LUT는 입력 루마 코드 값을 변경된 값 Y_r:Y_r=FwdLUT[Y_i]에 매핑 하는 순방향 LUT인 FwdLUT이다. 다른 LUT는 변경된 코드 값 Y_r을 $\hat{Y}_i$: $\hat{Y}_i$=InvLUT[Y_r]에 매핑 하는 역 LUT인 InvLUT이다. ($\hat{Y}_i$는 Y_i의 재구성 값을 나타냄).

[0333] ### 9.1 PWL 모델

[0334] 개념적으로 PWL(피스(piece) 별 선형)은 다음과 같은 방식으로 구현된다.

[0335] x1, x2를 두 개의 입력 피벗 포인트라고 하고 y1, y2를 한 피스에 대한 해당 출력 피벗 포인트라고 가정한다. x1과 x2 사이의 입력 값 x에 대한 출력 값 y는 다음 식으로 보간될 수 있다.

[0336] $$y = ((y2-y1)/(x2-x1)) * (x-x1) + y1$$

[0337] 고정 소수점 구현에서 식은 다음과 같이 다시 작성할 수 있다.

[0338] $$y = ((m * x + 2FP\_PREC-1) >> FP\_PREC) + c$$

[0339] 여기서 m은 스칼라, c는 오프셋, FP_PREC는 정밀도를 지정하기 위한 상수값이다.

[0340] CE-12 소프트웨어에서 PWL 모델은 1024 항목 FwdLUT 및 InvLUT 매핑 테이블을 미리 계산하는 데 사용된다; 그러나 PWL 모델은 구현이 LUT를 미리 계산하지 않고도 즉석에서 동일한 매핑 값을 계산할 수 있도록 한다.

[0341] ### 9.2 4차 VVC 회의에서 CE12-2 테스트

[0342] #### 9.2.1 루마 재형성(Luma reshaping)

[0343] 인루프 루마 재형성(in-loop luma reshaping)의 테스트 2(즉, 제안의 CE12-2)는 슬라이스 간 재구성에서 블록 유닛 인트라 예측을 위한 디코딩 대기 시간을 제거하는 더 낮은 복잡성 파이프라인을 제공한다. 인트라 예측은 인터 및 인트라 슬라이스 모두에 대해 재형성된 도메인에서 수행된다.

[0344] 인트라 예측은 슬라이스 유형에 관계없이 항상 재형성된 영역에서 수행된다. 이와 같이 구성하면 이전 TU 재구성이 완료된 직후 인트라 예측을 시작할 수 있다. 이러한 배열은 또한 슬라이스 종속 대신 인트라 모드에 대한

통합 프로세스를 제공할 수 있다. 도 7은 모드에 기반한 CE12-2 디코딩 프로세스의 블록도를 나타낸다.

[0345] CE12-2는 또한 CE12-1의 32 피스 PWL 모델 대신 루마 및 크로마 잔차 스케일링에 대해 16피스 PWL(피스별 선형) 모델을 테스트한다.

[0346] CE12-2에서 인루프 루마 재형성기(luma reshaper)를 사용한 슬라이스 간 재구성(밝은 음영 블록은 재형성된 도메인의 신호를 나타낸다: 루마 잔차(luma residue), 인트라 루마 예측(intra luma predicted), 인트라 루마 재구성(intra luma reconstructed))

[0347] ### 9.2.2 루마 의존 크로마 잔차 스케일링

[0348] 루마-종속 크로마 잔차 스케일링은 고정 소수점 정수 작업으로 구현된 승법 프로세스이다. 크로마 잔차 스케일링은 크로마 신호와 루마 신호의 상호 작용을 보상한다. 크로마 잔차 스케일링(Chroma Residual Scaling)은 TU 레벨에서 적용된다. 보다 구체적으로, 해당 루마 예측 블록의 평균값이 활용된다.

[0349] 평균은 PWL 모델에서 인덱스를 식별하는 데 사용된다. 인덱스는 스케일링 인자 cScaleInv를 식별한다. 크로마 잔차에 해당 숫자를 곱한다.

[0350] 크로마 스케일링 인자(chroma scaling factor)는 재구성된 루마 값보다는 순방향 매핑 된 예측 루마 값으로부터 계산된다는 점에 유의한다.

[0351] ### 9.2.3 ILR 부가 정보 신호

[0352] 파라미터는 (현재) 타일 그룹 헤더(ALF와 유사)로 전송된다. 이들은 40-100비트를 필요로 하는 것으로 알려져 있다.

[0353] *다음 사양은 JVET-L1001 버전 9를 기준으로 한다. 추가된 신택스는 노란색으로 강조 표시된다.*

[0354] *7.3.2.1 시퀀스 파라미터 세트 RBSP 신택스에서*

[0355]

seq_parameter_set_rbsp( ) {	기술자
sps_seq_parameter_set_id	ue(v)
...	
sps_triangle_enabled_flag	u(1)
sps_ladf_enabled_flag	u(1)
if (sps_ladf_enabled_flag) {	
sps_num_ladf_intervals_minus2	u(2)
sps_ladf_lowest_interval_qp_offset	se(v)
for(i = 0; i < sps_num_ladf_intervals_minus2 + 1; i++) {	
sps_ladf_qp_offset[i]	se(v)
sps_ladf_delta_threshold_minus1[i]	ue(v)
}	
}	
{{ **sps_reshaper_enabled_flag**	u(1) }}
rbsp_trailing_bits( )	
}	

[0356] *7.3.3.1 일반 타일 그룹 헤더 신택스에서*

tile_group_header( ) {	기술자
...	
if(num_tiles_in_tile_group_minus1 > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_tiles_in_tile_group_minus1; i++)	
entry_point_offset_minus1[i]	u(v)
}	
{{ if (sps_reshaper_enabled_flag) {	
tile_group_reshaper_model_present_flag	u(1)
if (tile_group_reshaper_model_present_flag)	
tile_group_reshaper_model ( )	
tile_group_reshaper_enable_flag	u(1)
if (tile_group_reshaper_enable_flag && (!(qtbtt_dual_tree_intra_flag && tile_group_type == I)))	
tile_group_reshaper_chr oma_residual_scale_flag	u(1)
} }}	
byte_alignment( )	
}	

[0357]

[0358] *새 신택스 테이블 타일 그룹 재형성 모델을 추가한다.*

{{ tile_group_reshaper_model ( ) {	기술자
reshaper_model_min_bin_idx	ue(v)
reshaper_model_delta_max_bin_idx	ue(v)
reshaper_model_bin_delta_abs_cw_prec_minus1	ue(v)
for (i = reshaper_model_min_bin_idx; i <= reshaper_model_max_bin_idx; i++) {	
reshape_model_bin_delta_abs_CW [i]	u(v)
if (reshaper_model_bin_delta_abs_CW[i]) > 0)	
reshaper_model_bin_delta_sign_CW_flag[i]	u(1)
}	
} }}	

[0359]

[0360] *{{ 일반 시퀀스 파라미터 세트 RBSP 의미 체계에서 다음 의미 체계를 추가한다. }}*

[0361] 1과 동일한 sps_reshaper_enabled_flag는 reshaper가 코딩 된 비디오 시퀀스(CVS)에서 사용됨을 지정한다. sps_reshaper_enabled_flag가 0이면 reshaper가 CVS에서 사용되지 않음을 지정한다.

[0362] *{{ 타일 그룹 헤더 신택스에 다음 의미를 추가한다. }}*

[0363] 1과 동일한 **tile_group_reshaper_model_present_flag**는 tile_group_reshaper_model()이 타일 그룹 헤더에 존재함을 지정한다. 0과 동일한 tile_group_reshaper_model_present_flag는 tile_group_reshaper_model()이 타일 그룹 헤더에 존재하지 않음을 지정한다. tile_group_reshaper_model_present_flag가 존재하지 않는 경우 0과 동일한 것으로 유추된다.

[0364] 1과 동일한 **tile_group_reshaper_enabled_flag**는 현재 타일 그룹에 대해 reshaper가 활성화됨을 지정한다. tile_group_reshaper_enabled_flag가 0이면 현재 타일 그룹에 대해 reshaper가 활성화되지 않음을 지정한다. tile_group_reshaper_enable_flag가 존재하지 않는 경우 0과 동일한 것으로 유추된다.

[0365] 1과 동일한 **tile_group_reshaper_chroma_residual_scale_flag**는 현재 타일 그룹에 대해 크로마 잔차 스케일링이 활성화됨을 지정한다. 0과 동일한 tile_group_reshaper_chroma_residual_scale_flag는 크로마 잔차 스케일링이 현재 타일 그룹에 대해 활성화되지 않음을 지정한다. tile_group_reshaper_chroma_residual_scale_flag가 존재하지 않는 경우, 0과 동일한 것으로 유추된다.

[0366] {{ *Add tile_group_reshaper_model( ) syntax* }}

[0367] **reshape_model_min_bin_idx**는 reshaper 구성 프로세스에서 사용할 최소 빈(또는 피스) 인덱스를 지정한다. reshape_model_min_bin_idx의 값은 0에서 MaxBinIdx까지의 범위에 있어야 한다. MaxBinIdx의 값은 15와 같아야 한다.

[0368] **reshape_model_delta_max_bin_idx**는 최대 허용 bin(또는 피스) 인덱스 MaxBinIdx에서 reshaper 구성 프로세스에 사용할 최대 bin 인덱스를 뺀 값을 지정한다. reshape_model_max_bin_idx의 값은 MaxBinIdx - reshape_model_delta_max_bin_idx와 동일하게 설정된다.

[0369] **reshaper_model_bin_delta_abs_cw_prec_minus1** 더하기 1은 신택스 reshape_model_bin_delta_abs_CW[i]의 표현에 사용되는 비트 수를 지정한다.

[0370] reshape_model_bin_delta_abs_CW[i]는 i번째 빈에 대한 절대 델타 코드워드 값을 지정한다.

[0371] **reshaper_model_bin_delta_sign_CW_flag[i]**는 다음과 같이 reshape_model_bin_delta_abs_CW[i]의 부호를 지정한다.

[0372] - reshape_model_bin_delta_sign_CW_flag[i]가 0이면 해당 변수 RspDeltaCW[i]는 양수 값이다.

[0373] - 그렇지 않으면(reshape_model_bin_delta_sign_CW_flag[i] 가 0 이 아님), 해당 변수 RspDeltaCW[i]는 음수 값이다.

[0374] reshape_model_bin_delta_sign_CW_flag[i]가 존재하지 않는 경우, 0과 동일한 것으로 유추된다.

[0375] 변수 RspDeltaCW[i] = (1 2*reshape_model_bin_delta_sign_CW [i]) * reshape_model_bin_delta_abs_CW [i];

[0376] 변수 RspCW[i]는 다음 단계에 따라 유도된다.

[0377] 변수 OrgCW는 (1 << BitDepthY) / (MaxBinIdx + 1)과 동일하게 설정된다.

[0378] - If reshaper_model_min_bin_idx < = i <= reshaper_model_max_bin_idx

[0379] RspCW[i] = OrgCW + RspDeltaCW[i].

[0380] - 그렇지 않으면 RspCW[i] = 0이다.

[0381] RspCW [i]의 값은 BitDepthY의 값이 10과 같으면 32에서 2 * OrgCW - 1의 범위에 있어야 한다.

[0382] 0에서 MaxBinIdx + 1(포함)의 범위에 있는 i가 있는 변수 InputPivot[i]는 다음과 같이 유도된다.

[0383] InputPivot[i] = i * OrgCW

[0384] i가 0에서 MaxBinIdx + 1(포함)의 범위에 있는 변수 ReshapePivot[i], 변수 ScaleCoef[i] 및 InvScaleCoeff[i](0에서 MaxBinIdx 포함 포함) 범위에 i가 있는 변수는 다음과 같이 유도된다.

[0385] shiftY = 14

[0386] ReshapePivot[0] = 0;

[0387] for(i = 0; i <= MaxBinIdx ; i++) {

[0388] ReshapePivot[i + 1] = ReshapePivot[i] + RspCW[i]

[0389] ScaleCoef[i] = (RspCW[i] * (1 << shiftY) + (1 << (Log2(OrgCW) - 1))) >> (Log2(OrgCW))

[0390] if (RspCW[i] == 0)

[0391] InvScaleCoeff[i] = 0

[0392] else

[0393] InvScaleCoeff[i] = OrgCW * (1 << shiftY) / RspCW[i]

[0394] }

[0395] 변수 ChromaScaleCoef[i]는 0에서 MaxBinIdx(포함)의 범위에 있으며 다음과 같이 유도된다.

[0396] ChromaResidualScaleLut [64] = {16384, 16384, 16384, 16384, 16384, 16384, 16384, 8192, 8192, 8192, 8192, 5461, 5461, 5461, 5461, 4096, 4096, 4096, 4096, 3277, 3277, 3277, 3,277, 2,731, 2,731, 2,731, 2,731, 2,341, 2,341, 2,341, 2,048, 2,048, 2,048, 1,820, 1,820, 1,820, 1,638, 1,638, 1,638, 1,638, 1,489, 1,489, 1,489, 1,489, 1,365, 1,365, 1,365, 1365, 1260, 1260, 1260, 1260, 1170, 1170, 1170, 1170,

1092, 1092, 1092, 1092, 1024, 1024, 11024},

[0397] shiftC = 11

[0398] - if (RspCW[i] == 0)

[0399] ChromaScaleCoef [i] = (1 << shiftC)

[0400] - Otherwise (RspCW[i] != 0), ChromaScaleCoef[i] = ChromaResidualScaleLut[RspCW[i] >> 1]

[0401] 9.2.4 ILR의 사용

[0402] 인코더 측에서, 각 픽처(또는 타일 그룹)은 먼저 재형성된 도메인으로 변환된다. 그리고 모든 코딩 프로세스는 재형성된 도메인에서 수행된다. 인트라 예측의 경우 이웃 블록은 재형성된 도메인에 있다; 인터 예측을 위해, (디코딩 된 픽처 버퍼로부터 원래 도메인으로부터 생성된) 레퍼런스 블록이 먼저 재형성된 도메인으로 변환된다. 그런 다음 잔차가 생성되어 비트스트림으로 코딩 된다.

[0403] 전체 픽처(또는 타일 그룹)가 인코딩/디코딩을 완료한 후 재형성된 도메인의 샘플을 원래 도메인으로 변환한 다음 디블로킹 필터 및 기타 필터를 적용한다.

[0404] 예측 신호에 대한 순방향 재형성은 다음과 같은 경우 비활성화된다.

[0405] Û 현재 블록은 인트라 코딩됨

[0406] Û 현재 블록은 CPR(현재 픽처 레퍼런스, 일명 인트라 블록 복사, IBC)로 코딩 된다.

[0407] Û 현재 블록이 결합된 인터-인트라 모드 (Combined Inter-Intra Mode) (CIIP)로 코딩 되고 인트라 예측 블록에 대해 순방향 재형성이 비활성화됨.

[0408] 10 양방향 광학 흐름(BDOF)

[0409] 10.1 BIO 개요 및 분석

[0410] BIO에서, 모션 보상은 현재 블록의 제1 예측(각 예측 방향)을 생성하기 위해 먼저 수행된다. 제1 예측은 공간적 구배, 시간적 구배 및 블록 내의 각 서브블록 또는 픽셀의 광학적 흐름을 유도하는 데 사용되며, 이는 제2 예측, 예를 들어 서브블록 또는 픽셀의 최종 예측을 생성하는 데 사용된다. 자세한 내용은 다음과 같다.

[0411] 양방향 광학 흐름(BIO) 방법은 이중 예측을 위해 블록 당 모션 보상(block-wise motion compensation) 위에 수행되는 샘플 당 모션 정제(sample-wise motion refinement)이다. 일부 구현에서, 샘플 레벨 모션 정제는 시그널링을 사용하지 않는다.

[0412] $I^{(k)}$를 블록 모션 보상 후 레퍼런스 k(k=0, 1)의 루마 값이라고 하고, $I^{(k)}$ 그레디언트의 수평 및 수직 컴포넌트로 $\partial I^{(k)}/\partial x$ 및 $\partial I^{(k)}/\partial y$를 각각 표시한다. 광학 흐름이 유효하다고 가정하면 모션 벡터 필드는 다음과 같이 주어진다.

[0413] $$\partial I^{(k)}/\partial t + v_x \partial I^{(k)}/\partial x + v_y \partial I^{(k)}/\partial y = 0. \quad \text{식(11)}$$

[0414] 각 샘플의 모션 궤적(motion trajectory)에 대한 이 광학 흐름 식을 헤르밋(Hermite) 보간법과 결합하면 함수 값 $I^{(k)}$와 일치하고 끝에서 $\partial I^{(k)}/\partial x$ 및 $\partial I^{(k)}/\partial y$을 유도하는 고유한 3차 다항식이 생성된다. t=0에서 이 다항식의 값은 BIO 예측이다:

[0415] $$pred_{BIO} = 1/2 \cdot \left(I^{(0)} + I^{(1)} + v_x/2 \cdot \left(\tau_1 \partial I^{(1)}/\partial x - \tau_0 \partial I^{(0)}/\partial x\right) + v_y/2 \cdot \left(\tau_1 \partial I^{(1)}/\partial y - \tau_0 \partial I^{(0)}/\partial y\right)\right) \quad \text{식(12)}$$

[0416] 도 8은 양방향 광학 흐름(BIO) 방법에 있어서 예시적인 광학 흐름 궤적을 나타낸다. 여기서, τ_0 및 τ_1은 기준 좌표계까지의 거리를 나타낸다. 거리 τ_0 및 τ_1은 Ref_0 및 Ref_1에 대한 POC에 기초하여 계산된다. τ_0=POC(현재) - POC(Ref_0), τ_1= POC(현재) - POC(Ref_1). 두 예측이 동일한 시간 방향(과거 또는 둘 다 미래에

서 온 것)에서 나온 경우 기호가 다르다(예를 들어, $\tau_0 \cdot \tau_1 < 0$). 이 경우 예측이 같은 시점(예를 들어, $\tau_0 \neq \tau_1$)이 아닌 경우 BIO가 적용된다.

[0417] 모션 벡터 필드(v_x, v_y)는 A 지점과 B 지점의 값의 차이 Δ를 최소화하여 결정된다. 도 8은 모션 궤적과 기준 프레임 평면이 교차하는 예를 나타낸다. 모델은 Δ에 대한 로컬 테일러 전개의 첫 번째 선형 항만 사용한다:

[0418] $$\Delta = \left(I^{(0)} - I^{(1)}_0 + v_x\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right) + v_y\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)\right) \quad \text{식(13)}$$

[0419] 위 식의 모든 값은 (i', j')로 표시된 샘플 위치에 따라 다르다. Δ는 현재 예측된 지점 (i,j)을 중심으로 하는 (2M+1)x(2M+1) 정사각형 윈도우 Ω 내부에서 최소화할 수 있고, 여기서 M은 2와 같다:

[0420] $$(v_x, v_y) = \underset{v_x, v_y}{\operatorname{argmin}} \sum_{[i',j] \in \Omega} \Delta^2[i', j'] \quad \text{식(14)}$$

[0421] 이 최적화 문제에 대해 JEM은 먼저 수직 방향으로 최소화한 다음 수평 방향으로 최소화하는 단순화된 접근 방식을 사용한다. 결과는 다음과 같다.

[0422] $$v_x = (s_1 + r) > m ? clip3\left(-thBIO, thBIO, -\frac{s_3}{(s_1+r)}\right) : 0 \quad \text{식(15)}$$

[0423] $$v_y = (s_5 + r) > m ? clip3\left(-thBIO, thBIO, -\frac{s_6 - v_x s_2/2}{(s_5+r)}\right) : 0 \quad \text{식(16)}$$

[0424] 여기서,

[0425] $$s_1 = \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)^2; s_3 = \sum_{[i',j] \in \Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)$$
$$s_2 = \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)$$
$$s_5 = \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)^2; s_6 = \sum_{[i',j] \in \Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right) \quad \text{식(17)}$$

[0426] 0 또는 매우 작은 값으로 나누는 것을 피하기 위해 정규화 파라미터 r 및 m이 식(15) 및 (16)에 도입될 수 있다. 여기서:

[0427] $r = 500 * 4^{d-8}$ 식(18)

[0428] $m = 700 * 4^{d-8}$ 식(19)

[0429] 여기서, d는 비디오 샘플의 비트 심도(bit depth)이다.

[0430] BIO에 대한 메모리 액세스를 일반 이중 예측 모션 보상과 동일하게 유지하기 위해, 모든 예측 및 그레디언트 값, $I^{(k)}$, $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$는 현재 블록 내부의 위치에 대해 계산된다. 도 9a는 블록(900) 외부의 액세스 위치의 예를 나타낸다. 도 9a에 도시된 바와 같이, 식(17)에서, (2M+1)x(2M+1) 정사각형 윈도우 Ω는 현재 예측된 블록 경계의 경계에 위치하여 블록 외부의 위치에 액세스해야 한다. JEM에서 $I^{(k)}$, $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$ 블록 외부값은 블록 내에서 가장 가까운 사용 가능한 값과 동일하게 설정된다. 예를 들어, 이는 도 9b에 도시된 바와 같이 패딩 영역(padding area)(901)으로 구현될 수 있다.

[0431] BIO를 사용하면 각 샘플에 대해 모션 필드를 정제할 수 있다. 계산 복잡성을 줄이기 위해 BIO의 블록 기반 설계가 JEM에서 사용된다. 모션 정제는 4x4 블록에 기초하여 계산할 수 있다. 블록 기반 BIO에서, 4x4 블록의 모든 샘플 중 식(17)의 s_n 값은 4x4 블록에 대한 유도된 BIO 모션 벡터 오프셋에 사용된다. 보다 구체적으로 블록 기반 BIO 유도에 다음 공식을 사용할 수 있다.

$$s_{1,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j']\in\Omega(x,y)} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)^2; s_{3,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j']\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)$$

$$s_{2,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j']\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)$$

$$s_{5,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j']\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)^2; s_{6,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j']\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)$$

[0432] 식(20)

[0433] 여기서, b_k는 예측 블록의 k번째 4x4 블록에 속하는 샘플들의 세트를 의미한다. 식 (15) 및 식 (16)의 s_n은 연관된 모션 벡터 오프셋을 유도하기 위해 (($s_{n,bk}$) >> 4)로 대체된다.

[0434] 일부 시나리오에서는 BIO의 MV 연대가 소음이나 불규칙한 작업으로 인해 신뢰할 수 없을 수 있다. 따라서 BIO에서는 MV 연대의 규모가 임계값으로 잘린다. 임계값은 현재 픽처의 레퍼런스 픽처가 모두 한 방향인지 여부에 따라 결정된다. 예를 들어, 현재 픽처의 모든 레퍼런스 픽처가 한 방향에서 온 경우 임계값 값은 $12x12^{14-d}$로 설정되고; 그렇지 않으면 $12x12^{13-d}$ 로 설정된다.

[0435] BIO에 대한 그레디언트는 HEVC 모션 보상 프로세스(예를 들어, 2D 분리 가능한 유한 임펄스 응답(Finite Impulse Response)(FIR))과 일치하는 작업을 사용하여 모션 보상 보간과 동시에 계산될 수 있다. 일부 실시예에서, 2D 분리 가능한 FIR에 대한 입력은 블록 모션 벡터의 분수 부분에 따른 모션 보상 프로세스 및 분수 위치(fracX, fracY)에 대한 것과 동일한 레퍼런스 프레임 샘플이다. 수평 그레디언트 ∂I/∂y의 경우, 신호는 먼저 스케일링 시프트 d-8을 제거하는 분수 위치 fracY에 해당하는 BIO*filterS*를 사용하여 수직으로 보간된다. 그런 다음 그레디언트 필터 BIO*filterG*가 18-d만큼 스케일링 제거 시프트를 사용하여 분수 위치 *frac*X에 해당하는 수평 방향으로 적용된다. 수직 그레디언트 ∂I/∂x의 경우 디스케일링 시프트가 d-8인 분수 위치 *frac*Y에 해당하는 BIO*filterG*를 사용하여 그레디언트 필터가 수직으로 적용된다. 그런 다음 18-d만큼 디스케일링 시프트를 사용하여 분수 위치 *fracX*에 해당하는 수평 방향으로 BIO*filterS*를 사용하여 신호 변위(signal displacement)를 수행한다. 그레디언트 계산 BIO*filterG* 및 신호 변위 BIO*filterF*를 위한 보간 필터의 길이는 합리적인 복잡성을 유지하기 위해 더 짧을 수 있다(예를 들어, 6-탭). 표 2는 BIO에서 블록 모션 벡터의 서로 다른 분수 위치의 그레디언트 계산에 사용할 수 있는 예제 필터를 보여준다. 표 3은 BIO에서 예측 신호 생성에 사용할 수 있는 보간 필터의 예를 보여준다.

[0436] 표 2 : BIO의 그레디언트 계산을 위한 예시 필터

분수 펠 위치(Fractional pel position)	그레디언트 보간 필터(Interpolation filter for gradient(BIOfilterG))
0	{8, −39, −3, 46, −17, 5}
1/16	{8, −32, −13, 50, −18, 5}
1/8	{7, −27, −20, 54, −19, 5}
3/16	{6, −21, −29, 57, −18, 5}
1/4	{4, −17, −36, 60, −15, 4}
5/16	{3, −9, −44, 61, −15, 4}
3/8	{1, −4, −48, 61, −13, 3}
7/16	{0, 1, −54, 60, −9, 2}
1/2	{−1, 4, −57, 57, −4, 1}

[0437]

[0438] 표 3 : BIO에서 예측 신호 생성을 위한 예시적인 보간 필터

분수 펠 위치(Fractional pel position)	그레디언트 보간 필터(Interpolation filter for gradient(BIOfilterG))
0	{ 0, 0, 64, 0, 0, 0}
1/16	{ 1, −3, 64, 4, −2, 0}
1/8	{ 1, −6, 62, 9, −3, 1}
3/16	{ 2, −8, 60, 14, −5, 1}
1/4	{ 2, −9, 57, 19, −7, 2}
5/16	{ 3, −10, 53, 24, −8, 2}
3/8	{ 3, −11, 50, 29, −9, 2}
7/16	{ 3, −11, 44, 35, −10, 3}
1/2	{ 3, −10, 35, 44, −11, 3}

[0439]

[0440] JEM에서 BIO는 두 예측이 서로 다른 레퍼런스 픽처에서 나온 경우 모든 이중 예측 블록에 적용될 수 있다. CU에 대해 LIC(Local Illumination Compensation)가 활성화되면 BIO가 비활성화될 수 있다.

[0441] 일부 실시예에서, OBMC는 정상 MC 프로세스 이후의 블록에 적용된다. 계산 복잡도를 줄이기 위해 OBMC 과정에서 BIO를 적용하지 않을 수 있다. 즉, 자신의 MV를 사용하는 경우 블록에 대한 MC 프로세스에 BIO를 적용하고 OBMC 과정에서 이웃 블록의 MV를 사용하는 경우 MC 프로세스에 BIO를 적용하지 않는다.

[0442] **11 JVET-N0236의 광학 흐름(PROF)을 통한 예측 정제**

[0443] 이 기여는 광학 흐름을 사용하여 서브 블록 기반 아핀 모션 보상 예측을 정제하는 방법을 제안한다. 서브 블록 기반 아핀 모션 보상이 수행된 후, 광학 흐름 식에 의해 유도된 차이를 추가하여 예측 샘플이 정제되며, 이를 광학 흐름을 통한 예측 정제(PROF)라고 한다. 제안하는 방법은 메모리 액세스 대역폭을 증가시키지 않고 픽셀 레벨 입도(pixel level granularity)에서 인터 예측을 달성할 수 있다.

[0444] 모션 보상의 더 미세한 입도를 달성하기 위해, 이 기여는 광학 흐름을 사용하여 서브 블록 기반 아핀 모션 보상 예측을 정제하는 방법을 제안한다. 서브 블록 기반 아핀 모션 보상이 수행된 후, 광학 흐름 식에 의해 유도된 차이를 추가함으로써 루마 예측 샘플이 정제된다. 제안된 PROF는 다음 4단계로 설명된다.

[0445] 단계 1) 서브 블록 기반 아핀 모션 보상을 수행하여 서브 블록 예측을 생성한다.

[0446] 2단계) 3-탭 필터[-1, 0, 1]를 사용하여 각 샘플 위치에서 공간적 그레디언트 및 서브블록 예측을 계산한다.

[0447] $$g_x(i,j) = I(i+1,j) - I(i-1,j)$$

[0448] $$g_y(i,j) = I(i,j+1) - I(i,j-1)$$

[0449] 서브 블록 예측은 그레디언트 계산을 위해 각 측면에서 한 픽셀씩 확장된다. 메모리 대역폭과 복잡성을 줄이기 위해 확장된 경계의 픽셀은 레퍼런스 픽처에서 가장 가까운 정수 픽셀 위치에서 복사된다. 따라서 패딩 영역에 대한 추가 보간을 피한다.

[0450] 단계 3) 광학 흐름 식에 의해 루마 예측 정제가 계산된다.

[0451] $$\Delta I(i,j) = g_x(i,j) * \Delta v_x(i,j) + g_y(i,j) * \Delta v_y(i,j)$$

[0452] 여기서, Δv(i,j)는 도 10에 도시된 바와 같이 샘플 위치에 대해 계산된 픽셀 MV(v(i,j)로 표시됨)와 픽셀 (i,j)이 속한 서브 블록의 서브 블록 MV의 차이이다.

[0453] 아핀 모델 파라미터와 서브 블록 중심에 대한 픽셀 위치는 서브 블록에서 서브 블록으로 변경되지 않으므로, Δv(i,j)는 제1 서브 블록에 대해 계산할 수 있고, 동일한 CU의 다른 서브 블록에 재사용된다. x 및 y를 픽셀 위치에서 서브 블록의 중심까지의 수평 및 수직 오프셋이라고 하면 Δv(x,y)는 다음 식에 의해 유도될 수 있다:

[0454] $$\begin{cases}\Delta v_x(x,y) = c*x + d*y \\ \Delta v_y(x,y) = e*x + f*y\end{cases}$$

[0455] 4-파라미터 아핀(affine) 모델의 경우,

[0456] $$\begin{cases} c = f = \frac{v_{1x}-v_{0x}}{w} \\ e = -d = \frac{v_{1y}-v_{0y}}{w}\end{cases}$$

[0457] 6-파라미터 아핀 모델의 경우,

[0458] $$\begin{cases} c = \frac{v_{1x}-v_{0x}}{w} \\ d = \frac{v_{2x}-v_{0x}}{h} \\ e = \frac{v_{1y}-v_{0y}}{w} \\ f = \frac{v_{2y}-v_{0y}}{h}\end{cases}$$

[0459] 여기서, (v_{0x},v_{0y}), (v_{1x},v_{1y}), (v_{2x},v_{2y})는 좌상단, 우상단, 좌하단 제어점 모션 벡터이고, w와 h는 CU의 너비와 높이이다.

[0460] 단계 4) 마지막으로, 루마 예측 정제가 서브블록 예측에 추가된다. 최종 예측 I'는 다음 식으로 생성된다.

[0461] $$I'(i,j) = I(i,j) + \Delta I(i,j)$$

[0462]

[0463] **12 기존 구현의 단점**

[0464] JVET-N0242 설계의 비선형 ALF(NLALF)에는 다음과 같은 문제가 있다.

[0465] (1) NLALF에서는 많은 클리핑 작업이 필요한다.

[0466] (2) CTU 기반 ALF에서 alf_num_available_temporal_filter_sets_luma가 0과 같을 때 사용 가능한 시간적 루마 필터가 없다. 그러나 lf_temporal_index는 여전히 시그널링될 수 있다.

[0467] (3) CTU 기반 ALF에서 alf_signal_new_filter_chroma가 0일 때, 크로마 컴포넌트에 대한 새로운 필터는 시그널링되지 않고, 시간적 크로마 필터가 사용된다고 가정한다. 그러나 시간적 크로마 필터를 사용할 수 있다는 보장은 없다.

[0468] (4) CTU 기반 ALF에서 alf_num_available_temporal_filter_sets_luma는 사용 가능한 시간적 필터 세트보다 클 수 있다.

[0469] **13 비디오 코딩을 위한 적응형 루프 필터링을 위한 예시적인 방법**

[0470] 현재 개시된 기술의 실시예는 기존 구현의 결점을 극복함으로써, 더 높은 코딩 효율을 갖는 비디오 코딩을 제공한다. 개시된 기술에 기초한 적응형 루프 필터링을 위한 기술은 기존 및 미래의 비디오 코딩 표준 모두를 향상시킬 수 있으며, 다양한 구현에 대해 설명되는 다음 예에서 설명된다. 아래에 제공된 개시된 기술의 예는 일반적인 개념을 설명하며 제한하는 것으로 해석되지 않는다. 예에서, 반대로 명시적으로 나타내지 않는 한, 이러하나의 실시예에서 설명된 다양한 특징이 결합될 수 있다.

[0471] 1. 샘플 차이를 클리핑 하는 대신 필터링 프로세스 동안 중간 결과에 클리핑 작업을 적용하는 것이 좋다. 필터링 과정에서 사용되는 현재 샘플의 인접 샘플(인접 또는 비인접)을 N(N >= 1)개의 그룹으로 분류할 수 있다고 가정한다.

[0472] a. 하나의 실시예에서, 그룹에 대해 하나 또는 다수의 중간 결과가 계산되고, 하나 또는 다수의 중간 결과에 대해 클리핑이 수행될 수 있다.

[0473] i. 예를 들어, 그룹에 대해, 각각의 이웃 픽셀과 현재 픽셀 간의 차이가 먼저 계산될 수 있고, 그런 다음 이러한 차이가 대응하는 ALF 계수를 사용하여 가중 평균될 수 있다(wAvgDiff로 표시됨). 그룹

에 대해 wAvgDiff에서 클리핑을 한 번 수행할 수 있다.

[0474] b. 다른 그룹에 대해 다른 클리핑 파라미터를 사용할 수 있다.

[0475] c. 하나의 실시예에서, 샘플 차이로 곱해진 필터 계수의 최종 가중 합에 클리핑이 적용된다.

[0476] i. 예를 들어, N = 1이고 클리핑은 다음과 같이 수행될 수 있고, 여기서 K(d,b)=min(b,max(-b,d))는 클리핑 함수이고 k는 클리핑 파라미터다. $O(x,y) = I(x,y) + K\left(\sum_{(i,j)\neq(0,0)} w(i,j).\left(I(x+i,y+j) - I(x,y)\right), k\right)$

[0477] 1) 또는, 가중 합계

[0478] $\sum_{(i,j)\neq(0,0)} w(i,j).\left(I(x+i,y+j) - I(x,y)\right)$ 반올림 여부에 관계없이 시프팅을 통해 정수 값으로 더 반올림될 수 있다.

[0479] 2. 하나의 샘플을 필터링할 때, N(N > 1)개의 이웃 샘플이 하나의 필터 계수를 공유하는 경우, 클리핑(예를 들어, 비선형 ALF에 의해 요구됨)은 N개의 모든 이웃 픽셀에 대해 한 번 수행될 수 있다.

[0480] a. 예를 들어, I(x+i1,y+j1)과 I(x+i2,y+j2)가 하나의 필터 계수(또는/및 하나의 클리핑 파라미터 k(i1,j1))를 공유하는 경우, 클리핑은 다음과 같이 한 번 수행될 수 있다.

[0481] clipValue(i1,j1)=K(I(x+i1,y+j1)+I(x+i2,y+j2)-2*I(x,y),k(i1,j1)), 및 w(i1,j1)*clipValue(i1,j1)는 식(14)에서 w(i1,j1)* K(I(x+i1,y+j1)-I(x,y),k(i1, j1))+w(i2,j2)*K(I(x+i2,y+j2)-I(x,y),k(i2,j2))를 대체하기 위해 사용될 수 있다.

[0482] i. 하나의 실시예에서, i1 및 i2는 대칭적인 위치에 있을 수 있다. 또한, j1과 j2는 대칭적인 위치에 있을 수 있다.

[0483] 1. 하나의 실시예에서, i1은 (-i2)와 같고 j1은 (-j2)와 같다.

[0484] ii. 하나의 실시예에서, (x+i1, y+j1)과 (x, y) 사이의 거리와 (x+i2, j+j2)와 (x, y) 사이의 거리는 동일할 수 있다.

[0485] iii. 글머리 기호 2에 개시된 방법은 필터 형태가 대칭 모드일 때 활성화된다.

[0486] iv. 대안적으로, 또한, I(x+i1,y+j1)와 연관된 클리핑 파라미터는 비트스트림으로부터 시그널링/유도될 수 있으며, ClipParam으로 표시되고 위에서 언급된 k(i1,j1)는 2* ClipParam과 같은 시그널링 된 클리핑 파라미터에서 유도된다.

[0487] b. 예를 들어, (i,j)∈가 하나의 필터 계수 w1(또는/및 하나의 클리핑 파라미터 k1)을 공유하고, N개의 요소를 포함하는 경우 클리핑은 다음과 같이 한 번 수행될 수 있다.

[0488] $$clipValue = K\left(\left(\sum_{(i,j)\in C} I(x+i,y+j)\right) - N * I(x,y), k1\right)$$

[0489] 여기서 k1은 와 연관된 클리핑 파라미터고, clipValue * w1은 식(14)에서 다음 항목을 대체하는 데 사용될 수 있다.

[0490] $$\sum_{(i,j)\in C} w(i,j) * K(I(x+i,y+j) - I(x,y), k(i,j))$$

[0491] i. 대안적으로, 또한, I(x+i,y+j)와 연관된 클리핑 파라미터는 ClipParam으로 표시되는 비트스트림으로부터 시그널링/유도될 수 있고, k1은 N* ClipParam과 같은 시그널링 된 클리핑 파라미터로부터 유래된다.

[0492] ii. 대안적으로, $\sum_{(i,j)\in C} I(x+i,y+j)$ 또는 $\left(\sum_{(i,j)\in C} I(x+i,y+j)\right) - N * I(x,y)$ 는 클리핑 되기 전에 오른쪽으로 이동한다.

[0493] c. 하나의 실시예에서, N개의 이웃 샘플 중 M1(M1 <= N)에 대해 클리핑이 한 번 수행될 수 있다.

[0494] d. 하나의 실시예에서, N개의 인접 샘플을 M2개의 그룹으로 분류하고, 각 그룹에 대해 클리핑을 한 번 수행할 수 있다.

[0495] e. 하나의 실시예에서, 이 방법은 특정 또는 모든 컬러 구성요소에 적용될 수 있다.

[0496] i. 예를 들어, 루마 컴포넌트에 적용될 수 있다.

[0497] ii. 예를 들어, Cb 또는/및 Cr 컴포넌트에 적용될 수 있다.

[0498] 3. 본 명세서에서는 min, max를 포함하는 [min, max] 범위로 입력을 클리핑 하는 클리핑 함수 K(min, max, input)를 사용할 수 있다.

[0499] a. 하나의 실시예에서, 위의 글머리 기호에서는 min, max를 제외한 범위(min, max)로 입력을 클리핑 하는 클리핑 함수 K(min, max, input)가 사용될 수 있다.

[0500] b. 하나의 실시예에서, 위의 글머리 기호에서는 max를 포함하지만 min을 제외한 범위(min, max]에 입력을 클리핑 하는 클리핑 함수 K(min, max, input)가 사용될 수 있다.

[0501] c. 하나의 실시예에서, 위의 글머리 기호에서는 min을 포함하지만 max를 제외한 입력을 [min, max) 범위로 클리핑 하는 클리핑 함수 K(min, max, input)가 사용될 수 있다.

[0502] 4. 시간적 ALF 계수 세트를 사용할 수 없는 경우(예를 들어, ALF 계수가 이전에 인코딩/디코딩 된 적이 없거나 인코딩/디코딩 된 ALF 계수가 "사용 불가"로 표시됨), 어떤 시간적 ALF 계수 세트가 사용되는지에 대한 표시의 시그널링은 스킵될 수 있다.

a. 하나의 실시예에서, 시간적 ALF 계수 세트를 사용할 수 없을 때, 새로운 ALF 계수나 고정된 ALF 계수가 CTB/블록/타일 그룹/타일/슬라이스/픽처에 대해 사용되지 않는다면, ALF는 CTB/블록/타일 그룹/타일/슬라이스/픽처 에 대해 허용되지 않는 것으로 유추된다.

[0503] 삭제

[0504] i. 대안적으로, 또한, 이 경우, CTB/블록/타일 그룹/타일/슬라이스/픽처에 대해 ALF가 적용된다고 지시(예를 들어, **alf_ctb_flag**가 CTU/블록에 대해 참)하더라도, ALF는 최종적으로 다음과 같이 유추될 수 있고, CTB/블록/타일 그룹/타일/슬라이스/픽처에 대해 허용되지 않는다.

[0505] b. 하나의 실시예에서, 시간적 ALF 계수 세트를 사용할 수 없는 경우, 새로운 ALF 계수 또는 고정된 ALF 계수 등만이 적합성 비트스트림에서 CTB/블록/타일 그룹/타일/슬라이스/픽처에 사용되도록 지시될 수 있다.

[0506] i. 예를 들어, **alf_use_new_filter** 또는 **alf_use_fixed_filter**는 참이여야 한다.

[0507] c. 하나의 실시예에서, 비트스트림은 다음 조건이 충족되는 경우 비준수 비트스트림으로 간주되고, 시간적 ALF 계수 세트를 사용할 수 없는 경우, ALF가 사용되는 것으로 표시된 CTB/블록/타일 그룹/타일/슬라이스/픽처에 대해, 새로운 ALF 계수나 고정된 ALF 계수가 사용되도록 표시되지 않는다.

[0508] i. 예를 들어, *alf_use_new_filter*와 *alf_use_fixed_filter*가 모두 거짓인 비트스트림은 부적합 비트스트림으로 간주된다.

[0509] d. 하나의 실시예에서, **alf_num_available_temporal_filter_sets_luma**가 0인 경우, **alf_temporal_index**는 시그널링되지 않을 수 있다.

[0510] e. 제안하는 방법은 컬러 컴포넌트에 따라 다르게 적용될 수 있다.

[0511] 5. 타일 그룹/타일/슬라이스/픽처/CTB/블록/비디오 유닛에 대해 얼마나 많은 시간적 ALF 계수 세트가 사용될 수 있는지는, 예를 들어, "사용 가능"으로 표시된 이전에 인코딩/디코딩 된 ALF 계수 세트와 같은, 사용 가능한 시간적 ALF 계수 세트(ALFavai로 표시됨)에 의존할 수 있고, 예를 들어 이전에 인코딩/디코딩 된 ALF 계수 세트에 따라 달라질 수 있다.

[0512] a. 하나의 실시예에서, 타일 그룹/타일/슬라이스/픽처/CTB/블록에 대해 ALFavai 시간적 ALF 계수

세트 이하가 사용될 수 있다.

[0513] b. 타일 그룹/타일/슬라이스/픽처/CTB/블록에 대해 min(N, ALFavai) 이하의 시간적 ALF 계수 세트가 사용될 수 있으며, 여기서 N >= 0이다. 예를 들어 N = 5이다.

[0514] 6. 새로운 ALF 계수 세트는 인코딩/디코딩 된 후 "사용 가능"으로 표시될 수 있다. 한편, 모든 "사용 가능" ALF 계수 세트는, IRAP(인트라 랜덤 액세스 포인트) 액세스 유닛 또는/및 IRAP 픽처 또는/및 IDR(순간 디코딩 리프레시) 액세스 유닛 또는/및 IDR 픽처를 접할 때 모두 "사용할 수 없음"으로 표시될 수 있다.

[0515] a. "사용 가능" ALF 계수 세트는 다음의 코딩 된 픽처/타일/타일 그룹/슬라이스/CTB/블록에 대한 시간적 ALF 계수 세트로서 사용될 수 있다.

[0516] b. "사용 가능" ALF 계수 세트는 N(N > 0)과 동일한 최대 크기의 하나의 ALF 계수 세트 목록에서 유지될 수 있다.

[0517] i. ALF 계수 세트 목록은 선입선출 순서로 유지될 수 있다.

[0518] c. "사용할 수 없음"으로 표시되면 관련 ALF APS 정보가 비트스트림에서 제거되거나 다른 ALF APS 정보로 대체된다.

[0519] 7. 시간적 계층마다 하나의 ALF 계수 세트 목록이 유지될 수 있다.

[0520] 8. K개의 이웃하는 시간적 계층에 대해 하나의 ALF 계수 세트 목록이 유지될 수 있다.

[0521] 9. 픽처가 이전 픽처로부터(표시 순서대로) 예측되는지 여부에 따라 상이한 픽처에 대해 상이한 ALF 계수 세트 목록이 유지될 수 있다.

[0522] a. 예를 들어, 선행 픽처로부터 예측된 픽처에 대해서만 하나의 ALF 계수 세트 목록이 유지될 수 있다.

[0523] b. 예를 들어, 선행 픽처와 후속 픽처 모두로부터 예측된 픽처에 대해 하나의 ALF 계수 세트 목록이 유지될 수 있다.

[0524] 10. ALF 계수 세트 목록은 IRAP 액세스 유닛 또는/및 IRAP 픽처 또는/및 IDR 액세스 유닛 또는/및 IDR 픽처를 접한 후에 비워질 수 있다.

[0525] 11. 다른 컬러 컴포넌트에 대해 다른 ALF 계수 세트 목록이 유지될 수 있다.

[0526] a. 하나의 실시예에서, 하나의 ALF 계수 세트 목록이 루마 컴포넌트에 대해 유지된다.

[0527] b. 하나의 실시예에서, Cb 또는/및 Cr 컴포넌트에 대해 하나의 ALF 계수 세트 목록이 유지된다.

[0528] 12. 하나의 ALF 계수 세트 목록이 유지될 수 있지만, 목록의 항목들은 상이한 픽처/타일 그룹/타일/슬라이스/CTU에 대해 상이한 인덱스(또는 우선순위)로 할당될 수 있다.

[0529] a. 하나의 실시예에서, ALF 계수 세트는 그것과 현재 픽처/타일 그룹/타일/슬라이스/CTU 사이의 절대 시간적 계층 차이를 오름차순으로 하기 위한 오름차순 인덱스로 할당될 수 있다.

[0530] b. 하나의 실시예에서, ALF 계수 세트는 그것과 현재 픽처/타일 그룹/타일/슬라이스/CTU 사이의 절대 POC(픽처 순서 카운트) 차이를 오름차순으로 하기 위한 오름차순 인덱스로 할당될 수 있다.

[0531] c. 하나의 실시예에서, 현재 픽처/타일 그룹/타일/슬라이스/CTU에 의해 허용되는 K ALF 계수 세트가 있다고 가정하면, 이들은 가장 작은 인덱스를 갖는 K ALF 계수 세트일 수 있다.

[0532] d. 하나의 실시예에서, 현재 픽처/타일 그룹/타일/슬라이스/CTU에 의해 사용되는 시간적 ALF 계수 세트의 표시는 또한 목록의 원래 항목 인덱스 대신에 할당된 인덱스에 의존할 수 있다.

[0533] 13. ALF에서 사용되는 인접 샘플들은 K(K >= 1)개의 그룹으로 분류될 수 있고, 각 그룹에 대해 하나의 클리핑 파라미터 세트가 시그널링될 수 있다.

[0534] 14. 클리핑 파라미터는 특정 또는 모든 고정 ALF 필터 세트에 대해 미리 정의될 수 있다.

[0535] a. 대안적으로, 현재 타일 그룹/슬라이스/픽처/타일에 의해 사용되는 특정 또는 모든 고정 필터 세트에 대해 클리핑 파라미터가 시그널링될 수 있다.

[0536] i. 하나의 실시예에서, 클리핑 파라미터는 특정 컬러 컴포넌트(예를 들어, 루마 컴포넌트)에 대해서만 시그널링될 수 있다.

[0537] b. 대안적으로, 고정 ALF 필터 세트를 사용하는 경우 클리핑이 수행되지 않을 수 있다.

[0538] i. 하나의 실시예에서, 클리핑은 특정 컬러 구성요소에 대해 수행되고 다른 컬러 구성요소에 대해서는 수행되지 않을 수 있다.

[0539] 15. 클리핑 파라미터는 ALF 계수와 함께 저장될 수 있으며, 코딩 된 CTU/CU/타일/타일 그룹/슬라이스/픽처에 따라 상속될 수 있다.

[0540] a. 하나의 실시예에서, CTU/CU/타일/타일 그룹/슬라이스/픽처에 의해 시간적 ALF 계수 세트가 사용되는 경우, 해당 ALF 클리핑 파라미터도 사용될 수 있다.

[0541] i. 하나의 실시예에서, 클리핑 파라미터는 특정 컬러 구성요소(예를 들어, 루마 구성요소)에 대해서만 상속될 수 있다.

[0542] b. 또는 CTU/CU/타일/타일 그룹/슬라이스/픽처에서 시간적 ALF 계수 세트를 사용하는 경우 클리핑 파라미터가 시그널링될 수 있다.

[0543] i. 하나의 실시예에서, 클리핑 파라미터는 특정 컬러 컴포넌트(예를 들어, 루마 컴포넌트)에 대해서만 시그널링될 수 있다.

[0544] c. 하나의 실시예에서, 클리핑 파라미터는 특정 컬러 구성요소에 대해 상속될 수 있고 다른 컬러 구성요소에 대해 시그널링될 수 있다.

[0545] d. 하나의 실시예에서, 시간적 ALF 계수 세트가 사용되는 경우, 클리핑이 수행되지 않는다.

[0546] i. 하나의 실시예에서, 클리핑은 특정 컬러 구성요소에 대해 수행되고 다른 컬러 구성요소에 대해서는 수행되지 않을 수 있다.

[0547] 16. 비선형 ALF의 사용 여부는 ALF 필터 세트 유형(예를 들어, 고정 ALF 필터 세트, 시간적 ALF 필터 세트 또는 시그널링 된 ALF 계수 세트)에 따라 달라질 수 있다.

[0548] a. 하나의 실시예에서, 현재 CTU가 고정 ALF 필터 세트 또는 시간적 ALF 필터 세트(일명, 이전에 시그널링 된 필터 세트가 사용됨)를 사용하는 경우, 비선형 ALF는 현재 CTU에 대해 사용되지 않을 수 있다.

[0549] b. 하나의 실시예에서, alf_luma_use_fixed_filter_flag가 1과 같을 때, 비선형 ALF는 현재 슬라이스/타일 그룹/타일/CTU에 대해 사용될 수 있다.

[0550] 17. 비선형 ALF 클리핑 파라미터는 ALF 필터 세트 유형(예를 들어, 고정 ALF 필터 세트, 시간적 ALF 필터 세트 또는 시그널링 된 ALF 계수 세트)에 따라 조건부로 시그널링될 수 있다.

[0551] a. 하나의 실시예에서, 비선형 ALF 클리핑 파라미터는 모든 ALF 필터 세트에 대해 시그널링될 수 있다.

[0552] b. 하나의 실시예에서, 비선형 ALF 클리핑 파라미터들은 시그널링 된 ALF 필터 계수 세트들에 대해서만 시그널링될 수 있다.

[0553] c. 하나의 실시예에서, 비선형 ALF 클리핑 파라미터는 고정된 ALF 필터 계수 세트에 대해서만 시그널링될 수 있다.

[0554] 위에서 설명된 예들은 비디오 디코더 및/또는 비디오 인코더에서 구현될 수도 있는 방법들, 예를 들어, 방법들(1110, 1120, 1130, 1140, 1150 및 1160)과 같이 아래에서 설명되는 방법과 관련하여 통합될 수도 있다.

[0555] 도 11a는 비디오 처리를 위한 예시적인 방법의 흐름도를 나타낸다. 방법(1110)은, 작업(1112)에서, 비디오의 현재 비디오 블록에 대해 필터 계수를 사용하고 적어도 하나의 중간 결과를 갖는 둘 이상의 작업을 포함하는 필터링 프로세스를 수행하는 단계를 포함한다.

[0556] 방법(1110)은, 작업(1114)에서, 적어도 하나의 중간 결과에 클리핑 작업을 적용하는 단계를 포함한다.

[0557] 방법(1110)은, 작업(1116)에서, 적어도 하나의 중간 결과에 기초하여, 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다. 일부 실시예에서, 적어도 하나의 중간 결과는 필터 계수의 가중

합 및 현재 비디오 블록의 현재 샘플과 현재 샘플의 이웃 샘플 간의 차이에 기초한다.

[0558] 도 11b는 비디오 처리를 위한 예시적인 방법의 흐름도를 나타낸다. 방법(1120)은, 작업(1122)에서, 비디오의 현재 비디오 블록을 비디오의 비트스트림 표현으로 인코딩 하는 단계를 포함하고, 현재 비디오 블록은 적응형 루프 필터(ALF)로 코딩 된다.

[0559] 방법(1120)은, 작업(1124)에서, 하나 이상의 시간 적응형 필터 세트의 사용 가능성 또는 사용에 기초하여, 비트스트림 표현에서 하나 이상의 시간 적응형 필터 세트 내의 시간 적응형 필터 세트의 표시를 선택적으로 포함하는 단계를 포함한다.

[0560] 도 11c는 비디오 처리를 위한 예시적인 방법의 흐름도를 나타낸다. 방법(1130)은, 작업(1132)에서, 비디오의 비트스트림 표현에서 시간 적응형 필터의 세트의 표시에 기초하여, 적응형 루프 필터(ALF)로 코딩 된 비디오의 현재 비디오 블록에 적용가능한 시간 적응형 필터의 세트를 포함하는 시간 적응형 필터의 하나 이상의 세트의 사용 가능성 또는 사용을 결정하는 단계를 포함한다.

[0561] 방법(1130)은, 작업(1134)에서, 결정하는 단계에 기초하여, 시간 적응형 필터의 세트를 선택적으로 적용함으로써 비트스트림 표현으로부터 디코딩 된 현재 비디오 블록을 생성하는 단계를 포함한다.

[0562] 도 11d는 비디오 처리를 위한 예시적인 방법의 흐름도를 나타낸다. 방법(1140)은, 작업(1142)에서, 적응형 루프 필터로 코딩 된 현재 비디오 블록에 대해, 사용 가능한 시간적 ALF 계수 세트에 기초하여 시간 적응형 루프 필터링(ALF) 계수 세트의 수를 결정하는 단계를 포함하고, 여기서 사용 가능한 시간적 ALF는 계수 세트는 결정하는 단계 이전에 인코딩 또는 디코딩 되었으며, ALF 계수 세트의 수는 타일 그룹, 타일, 슬라이스, 픽처, 코딩 트리 블록(CTB), 또는 현재 비디오 블록을 포함하는 비디오 유닛에 사용된다.

[0563] 방법(1140)은, 작업(1144)에서, 시간적 ALF 계수 세트의 수에 기초하여, 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다.

[0564] 도 11e는 비디오 처리를 위한 예시적인 방법의 흐름도를 나타낸다. 방법(1150)은, 작업(1152)에서, 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역의 헤더에서 적응형 루프 필터링(ALF)의 표시가 비트스트림 표현과 연관된 적응형 파라미터 세트(APS) 네트워크 추상 계층(NAL) 유닛에서 ALF의 표시와 동일하다고 결정하는 단계를 포함한다.

[0565] 방법(1150)은, 작업(1154)에서, 변환을 수행하는 단계를 포함한다.

[0566] 도 11f는 비디오 처리를 위한 예시적인 방법의 흐름도를 나타낸다. 방법(1160)은, 작업(1162)에서, 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역에 의해 사용되는 적응형 루프 필터의 유형에 기초하여 비선형 적응형 루프 필터링(ALF) 동작을 선택적으로 가능하게 하는 단계를 포함한다.

[0567] 방법(1160)은, 작업(1164)에서, 선택적으로 활성화 하는 단계에 후속하여, 변환을 수행하는 단계를 포함한다.

[0568] ## 10 개시된 기술의 예시적인 구현

[0569] ### 10.1 실시예 #1

[0570] 루마 및 크로마에 대해 각각 하나의 ALF 계수 세트 목록이 유지되고 두 목록의 크기가 각각 lumaALFSetSize 및 chromaALFSetSize라고 가정한다. ALF 계수 세트 목록의 최대 크기는 각각 lumaALFSetMax(예를 들어, lumaALFSetMax가 5와 동일) 및 chromaALFSetMax(예를 들어, chromaALFSetMax가 5와 동일)이다.

[0571] 새로 추가된 부분은 이중 굵은 중괄호로 묶고, 즉, "a"로 표시된 {{a}}가 추가되고 삭제된 부분은 이중 대괄호로 묶이고, 즉, [[a]]는 "a"가 삭제됨을 나타낸다.

[0572] 7.3.3.2 적응형 루프 필터 데이터 신택스

alf_data( ) {	기술자
alf_chroma_idc	tu(v)
alf_signal_new_filter_luma	u(1)
if(**alf_signal_new_filter_luma** > 0) {	
alf_luma_use_fixed_filter_flag	u(1)
if(**alf_luma_use_fixed_filter_flag**){	
alf_luma_fixed_filter_set_index	tb(v)
alf_luma_fixed_filter_usage_pattern	u(1)
if (**alf_luma_fixed_filter_usage_pattern** > 0)	
for (i = 0; i < NumAlfFilters; i++)	
alf_luma_fixed_filter_usage[i]	u(1)
{{ if (lumaALFSetSize > 0) }}	
alf_num_available_temporal_filter_sets_luma	tb(1)
alf_luma_num_filters_signalled_minus1	tb(v)
alf_luma_type_flag	u(1)
if(**alf_luma_num_filters_signalled_minus1** > 0) {	
for(filtIdx = 0; filtIdx < NumAlfFilters; **filtIdx**++)	
alf_luma_coeff_delta_idx[filtIdx]	tu(v)
}	
alf_luma_coeff_delta_flag	u(1)
if (!**alf_luma_coeff_delta_flag** && alf_luma_num_filters_signalled_minus1 > 0)	
alf_luma_coeff_delta_prediction_flag	u(1)
alf_luma_min_eg_order_minus1	tu(v)
for(i = 0; i < (**alf_luma_type_flag** == 1) ? 2 : 3; i++)	
alf_luma_eg_order_increase_flag[i]	u(1)

[0573]

if (alf_luma_coeff_delta_flag) {	
for(sigFiltIdx = 0; **sigFiltIdx** <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++)	
alf_luma_coeff_flag[sigFiltIdx]	u(1)
}	
for(sigFiltIdx = 0; **sigFiltIdx** <= **alf_luma_num_filters_signalled_minus1**; sigFiltIdx++){	
if (alf_luma_coeff_flag[sigFiltIdx]) {	
for (j = 0; j < (alf_luma_type_flag == 1) ? 6 : 12; j++) {	
alf_luma_coeff_delta_abs[sigFiltIdx][j]	uek(v)
if(alf_luma_coeff_delta_abs[sig**FiltIdx**][j])	
alf_luma_coeff_delta_sign[sigFiltIdx][j]	u(1)
}	
}	
}	
}	
if (alf_chroma_idc > 0) {	
alf_signal_new_filter_chroma	u(1)
if (alf_signal_new_filter_chroma){	
alf_chroma_min_eg_order_minus1	tu(v)
for(i = 0; i < 2; i++)	
alf_chroma_eg_order_increase_flag[i]	u(1)
for(j = 0; j < 6; j++) {	
alf_chroma_coeff_abs[j]	uek(v)
if(alf_chroma_coeff_abs[j] > 0)	
alf_chroma_coeff_sign[j]	u(1)
}	
}	
}	
}	

[0574]

[0575] 7.3.4.2 코딩 트리 유닛 신택스

coding_tree_unit( ) {	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_alf_enable_flag){	
alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
if (alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] {	
coding_alf_ctu_filter_set_index(CtbAddrInRs)	
}	
if(alf_chroma_idc == 1 || alf_chroma_idc == 3)	
alf_ctb_flag[1][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
if(alf_chroma_idc == 2 || alf_chroma_idc == 3)	
alf_ctb_flag[2][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
}	
if(slice_type == I && qtbtt_dual_tree_intra_flag) {	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbLog2SizeY, 0)	
else	
coding_quadtree(xCtb, yCtb, CtbLog2SizeY, 0, SINGLE_TREE)	
}	

[0576]

coding_alf_ctu_filter_set_index(CtbAddrInRs, slice_type){	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
alf_use_new_filter	ae(1)
if (alf_use_new_filter == 0){	
alf_use_fixed_filter	ae(1)
}	
if (alf_use_new_filter){	
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = 16	
}	
else if (alf_use_fixed_filter){	
alf_fixed_filter_index	tb(v)
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = alf_fixed_filter_index	
}	
Else {{ if (alf_num_available_temporal_filter_sets_luma > 0) }} {	
alf_temporal_index	tb(v)
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = 16 + alf_temporal_index	
}	

[0577]

[0578] 1과 동일한 **alf_signal_new_filter_luma**는 새로운 루마 필터 세트가 시그널링됨을 지정한다. 0과 동일한 alf_signal_new_filter_luma는 새로운 루마 필터 세트가 시그널링되지 않음을 지정한다. 존재하지 않을 때는 0이다.

[0579] 1과 동일한 **alf_luma_use_fixed_filter_flag**는 고정 필터 세트가 적응형 루프 필터를 신호하는 데 사용됨을 지정한다. 0과 동일한 alf_luma_use_fixed_filter_flag는 고정 필터 세트가 적응형 루프 필터를 신호하는 데 사용되지 않음을 지정한다.

[0580] **alf_num_available_temporal_filter_sets_luma**는 현재 슬라이스에 사용할 수 있는 사용 가능한 시간적 필터 세트의 수를 지정하며, 0..[[5]] {{ lumaALFSetSize }}일 수 있다. 존재하지 않을 때는 0이다.

[0581] {{ alf_num_available_temporal_filter_sets_luma가 0과 같을 때 alf_signal_new_filter_luma 또는 alf_luma_use_fixed_filter_flag가 1과 같아야 하는 것이 제한된다. }}

[0582] 1과 동일한 **alf_signal_new_filter_chroma**는 새로운 크로마 필터가 시그널링되는 것을 지정한다. 0과 동일한

alf_signal_new_filter_chroma는 새로운 크로마 필터가 시그널링되지 않음을 지정한다.

[0583] {{ chromaALFSetSize가 0일 때 alf_signal_new_filter_chroma가 1이어야 한다는 제약이 있다. }}

[0584] ## 10.2 실시예 #2

[0585] 루마 및 크로마에 대해 각각 하나의 ALF 계수 세트 목록이 유지되고 두 목록의 크기가 각각 lumaALFSetSize 및 chromaALFSetSize라고 가정한다. ALF 계수 세트 목록의 최대 크기는 각각 lumaALFSetMax(예를 들어, lumaALFSetMax가 5와 동일) 및 chromaALFSetMax(예를 들어, chromaALFSetMax가 5와 동일)이다.

[0586] 이중 굵은 중괄호로 묶인 새로 추가된 부분, 즉 "a"로 표시된 {{a}}가 추가되고 삭제된 부분이 이중 대괄호로 묶이고, 즉, [[a]]가 "a"가 삭제됨을 나타낸다.

[0587] 7.3.3.2 적응형 루프 필터 데이터 신택스

alf_data( ) {	기술자
alf_chroma_idc	tu(v)
alf_signal_new_filter_luma	u(1)
if(alf_signal_new_filter_luma > 0) {	
alf_luma_use_fixed_filter_flag	u(1)
if(alf_luma_use_fixed_filter_flag){	
alf_luma_fixed_filter_set_index	tb(v)
alf_luma_fixed_filter_usage_pattern	u(1)
if (alf_luma_fixed_filter_usage_pattern > 0)	
for (i = 0; i < NumAlfFilters; i++)	
alf_luma_fixed_filter_usage[i]	u(1)
{{ if (lumaALFSetSize > 0) }}	
alf_num_available_temporal_filter_sets_luma	tb(1)
alf_luma_num_filters_signalled_minus1	tb(v)
alf_luma_type_flag	u(1)
if(alf_luma_num_filters_signalled_minus1 > 0) {	
for(filtIdx = 0; filtIdx < NumAlfFilters; filtIdx++)	
alf_luma_coeff_delta_idx[filtIdx]	tu(v)
}	
alf_luma_coeff_delta_flag	u(1)
if(!alf_luma_coeff_delta_flag && alf_luma_num_filters_signalled_minus1 > 0)	
alf_luma_coeff_delta_prediction_flag	u(1)
alf_luma_min_eg_order_minus1	tu(v)
for(i = 0; i < (alf_luma_type_flag == 1) ? 2 : 3; i++)	
alf_luma_eg_order_increase_flag[i]	u(1)

[0588]

if (alf_luma_coeff_delta_flag) {	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++)	
alf_luma_coeff_flag[sigFiltIdx]	u(1)
}	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++){	
if (alf_luma_coeff_flag[sigFiltIdx]) {	
for (j = 0; j < (alf_luma_type_flag == 1) ? 6 : 12; j++) {	
alf_luma_coeff_delta_abs[sigFiltIdx][j]	uek(v)
if(alf_luma_coeff_delta_abs[sigFiltIdx][j])	
alf_luma_coeff_delta_sign[sigFiltIdx][j]	u(1)
}	
}	
}	
}	
if (alf_chroma_idc > 0) {	
alf_signal_new_filter_chroma	u(1)
if (alf_signal_new_filter_chroma){	
alf_chroma_min_eg_order_minus1	tu(v)
for(i = 0; i < 2; i++)	
alf_chroma_eg_order_increase_flag[i]	u(1)
for(j = 0; j < 6; j++) {	
alf_chroma_coeff_abs[j]	uek(v)
if(alf_chroma_coeff_abs[j] > 0)	
alf_chroma_coeff_sign[j]	u(1)
}	
}	
}	
}	

[0589]

[0590] 7.3.4.2 코딩 트리 유닛 신택스

coding_tree_unit( ) {	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_alf_enable_flag){	
alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
if (alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] {	
coding_alf_ctu_filter_set_index(CtbAddrInRs)	
}	
if(alf_chroma_idc == 1 || alf_chroma_idc == 3)	
alf_ctb_flag[1][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
if(alf_chroma_idc == 2 || alf_chroma_idc == 3)	
alf_ctb_flag[2][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
}	
if(slice_type = = I && qtbtt_dual_tree_intra_flag) {	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbLog2SizeY, 0)	
else	
coding_quadtree(xCtb, yCtb, CtbLog2SizeY, 0, SINGLE_TREE)	
}	

[0591]

coding_alf_ctu_filter_set_index(CtbAddrInRs, slice_type){	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
alf_use_new_filter	ae(1)
if (alf_use_new_filter == 0){	
alf_use_fixed_filter	ae(1)
}	
if (alf_use_new_filter){	
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = 16	
}	
else if (alf_use_fixed_filter){	
alf_fixed_filter_index	tb(v)
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = alf_fixed_filter_index	
}	
Else {{ if (alf_num_available_temporal_filter_sets_luma > 0) }} {	
alf_temporal_index	tb(v)
alf_luma_ctb_filter_set_index[xCtb >> Log2CtbSize][yCtb >> Log2CtbSize] = 16 + alf_temporal_index	
}	

[0592]

[0593] 1과 동일한 **alf_signal_new_filter_luma**는 새로운 루마 필터 세트가 시그널링됨을 지정한다. 0과 동일한 alf_signal_new_filter_luma는 새로운 루마 필터 세트가 시그널링되지 않음을 지정한다. 존재하지 않을 때는 0이다.

[0594] 1과 동일한 **alf_luma_use_fixed_filter_flag**는 고정 필터 세트가 적응형 루프 필터를 신호하는 데 사용됨을 지정한다. 0과 동일한 alf_luma_use_fixed_filter_flag는 고정 필터 세트가 적응형 루프 필터를 신호하는 데 사용되지 않음을 지정한다.

[0595] **alf_num_available_temporal_filter_sets_luma**는 현재 슬라이스에 사용할 수 있는 사용 가능한 시간적 필터 세트의 수를 지정하며, 0..[[5]] {{ lumaALFSetSize }}일 수 있다. 존재하지 않을 때는 0이다.

[0596] {{ alf_num_available_temporal_filter_sets_luma가 0과 같을 때 alf_signal_new_filter_luma 또는 alf_luma_use_fixed_filter_flag가 1과 같아야 하는 것이 제한된다. }}

[0597] 1과 동일한 **alf_signal_new_filter_chroma**는 새로운 크로마 필터가 시그널링되는 것을 지정한다. 0과 동일한 alf_signal_new_filter_chroma는 새로운 크로마 필터가 시그널링되지 않음을 지정한다.

[0598] {{ chromaALFSetSize가 0일 때 alf_signal_new_filter_chroma가 1이어야 한다는 제약이 있다. }}

[0599] 일부 실시예에서 다음과 같은 기술적 솔루션이 구현될 수 있다.

[0600] A1. 비디오 처리를 위한 방법으로서, 비디오의 현재 비디오 블록에 대해, 필터 계수를 사용하고 적어도 하나의 중간 결과를 갖는 둘 이상의 작업을 포함하는 필터링 프로세스를 수행하는 단계; 적어도 하나의 중간 결과에 클리핑 작업을 적용하는 단계; 및 적어도 하나의 중간 결과에 기초하여, 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함하고, 적어도 하나의 중간 결과는 필터 계수의 가중 합 및 현재 비디오 블록의 현재 샘플과 현재 샘플의 이웃 샘플 간의 차이에 기초한다.

[0601] A2. A1의 방법에 있어서, 현재 샘플에 대해, 현재 샘플의 이웃 샘플들을 복수의 그룹으로 분류하는 단계를 더 포함하고, 클리핑 작업은 복수의 그룹 각각의 중간 결과에 상이한 파라미터로 적용된다.

[0602] A3. A2의 방법에 있어서, 적어도 하나의 중간 결과는 현재 샘플과 복수의 그룹 각각의 이웃 샘플 간의 차이의 가중 평균을 포함한다.

[0603] A4. A1의 방법에 있어서, 현재 비디오 블록의 샘플의 복수의 이웃 샘플은 필터 계수를 공유하고, 클리핑 작업은 복수의 이웃 샘플 각각에 한 번 적용된다.

[0604] A5. A4의 방법에 있어서, 복수의 이웃 샘플 중 적어도 2개의 위치는 현재 비디오 블록의 샘플에 대해 대칭이다.

[0605] A6. A4 또는 A5의 방법에 있어서, 필터링 프로세스와 연관된 필터 형태는 대칭 모드이다.

[0606] A7. A4 내지 A6 중 어느 하나의 방법에 있어서, 클리핑 작업의 하나 이상의 파라미터는 비트스트림 표현에서 시그널링 된다.

[0607] A8. A1의 방법에 있어서, 현재 비디오 블록의 샘플은 N개의 이웃 샘플을 포함하고, 여기서 클리핑 작업은 N개의 이웃 샘플 중 M1개의 이웃 샘플에 한 번 적용되고, M1 및 N은 양의 정수이고, M1 ≤ N이다.

[0608] A9. A1의 방법에 있어서, 현재 비디오 블록의 샘플에 대해, 샘플의 N개의 이웃 샘플을 M2개의 그룹으로 분류하는 단계를 더 포함하고, 여기서 클리핑 작업은 M2개의 그룹 각각에 한 번 적용되고, M2 및 N은 양의 정수이다.

[0609] A10. A1의 방법에 있어서, 클리핑 작업은 현재 비디오 블록과 관련된 루마 구성 요소에 적용된다.

[0610] A11. A1의 방법에 있어서, 클리핑 작업은 현재 비디오 블록과 관련된 Cb 컴포넌트 또는 Cr 컴포넌트에 적용된다.

[0611] A12. A1 내지 A11 중 어느 하나의 방법에 있어서, 클리핑 작업은 K(min, max, input)로 정의되고, 여기서 입력은 클리핑 작업에 대한 입력이고, min은 클리핑 작업의 출력의 공칭 최소값이고, max는 클리핑 작업의 출력의 공칭 최대값이다.

[0612] A13. A12의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값보다 작고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값보다 크다.

[0613] A14. A12의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값과 동일하고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값보다 크다.

[0614] A15. A12의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값보다 작고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값과 동일하다.

[0615] A16. A12의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값과 동일하고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값과 동일하다.

[0616] A17. A1의 방법에 있어서, 필터링 프로세스는 복수의 ALF 필터 계수 세트로 구성된 적응형 루프 필터링(ALF) 프로세스를 포함한다.

[0617] A18. A17의 방법에 있어서, 클리핑 작업을 위한 적어도 하나의 파라미터는 복수의 ALF 필터 계수 세트 중 하나 이상에 대해 미리 정의된다.

[0618] A19. A17의 방법에 있어서, 클리핑 작업을 위한 적어도 하나의 파라미터는 타일 그룹, 슬라이스, 픽처, 또는 현재 비디오 블록을 포함하는 타일에 대한 비트스트림 표현에서 시그널링 된다.

[0619] A20. A19의 방법에 있어서, 적어도 하나의 파라미터는 현재 비디오 블록과 연관된 하나 이상의 컬러 컴포넌트에 대해서만 시그널링 된다.

[0620] A21. A17의 방법에 있어서, 복수의 ALF 필터 계수 세트 중 적어도 하나 및 클리핑 작업을 위한 하나 이상의 파라미터는 동일한 메모리 위치에 저장되고, 적어도 하나의 복수의 ALF 필터 계수 세트 또는 하나 이상의 파라미터는 코딩 된 코딩 트리 유닛(CTU), 코딩 유닛(CU), 타일, 타일 그룹, 슬라이스, 또는 현재 비디오 블록을 포함하는 픽처에 의해 상속된다.

[0621] A22. A21의 방법에 있어서, 클리핑 작업은, 시간적 ALF 계수 세트가 CTU, CU, 타일, 타일 그룹, 슬라이스, 또는 현재 비디오 블록을 포함하는 픽처에 대한 필터링 프로세스에서 사용된다는 결정에 따라, 복수의 ALF 필터 계수 세트의 시간적 ALF 계수 세트에 대응하는 하나 이상의 파라미터를 사용하도록 구성된다.

[0622] A23. A22의 방법에 있어서, 시간적 ALF 계수 세트에 대응하는 하나 이상의 파라미터는 현재 비디오 블록과 연관된 하나 이상의 컬러 컴포넌트에 대해서만 사용된다.

[0623] A24. A21의 방법에 있어서, 복수의 ALF 필터 계수 세트 중 시간적 ALF 계수 세트에 대응하는 하나 이상의 파라미터는, 시간적 ALF 계수 세트가 CTU, CU, 타일, 타일 그룹, 슬라이스, 또는 현재 비디오 블록을 포함하는 픽처에 대한 필터링 프로세스에서 사용된다는 결정에 따라, 비트스트림 표현으로 시그널링 된다.

[0624] A25. A24의 방법에 있어서, 시간적 ALF 계수 세트에 대응하는 하나 이상의 파라미터는 현재 비디오 블록과 연관

된 하나 이상의 컬러 컴포넌트에 대해서만 시그널링 된다.

[0625] A26. A21의 방법에 있어서, 현재 비디오 블록과 연관된 제1 컬러 컴포넌트에 대한 하나 이상의 파라미터의 파라미터의 제1 세트가 시그널링되고, 현재 비디오 블록과 연관된 제2 컬러 컴포넌트에 대한 하나 이상의 파라미터의 제2 파라미터 세트가 상속된다.

[0626] A27. A1 내지 A26 중 어느 하나의 방법에 있어서, 변환은 비트스트림 표현으로부터 현재 비디오 블록을 생성한다.

[0627] A28. A1 내지 A26 중 어느 하나의 방법에 있어서, 변환은 현재 비디오 블록으로부터 비트스트림 표현을 생성한다.

[0628] A29. 프로세서 및 그 위에 명령이 있는 비일시적 메모리를 포함하는 비디오 시스템의 장치에 있어서, 프로세서에 의한 실행 시 명령은 프로세서로 하여금 A1 내지 A28 중 어느 하나의 방법을 구현하게 하는 장치.

[0629] A30. 비일시적 컴퓨터 판독 가능 매체에 저장된 컴퓨터 프로그램에 있어서, A1 내지 A28 중 어느 하나의 방법을 수행하기 위한 프로그램 코드를 포함하는 컴퓨터 프로그램.

[0630] 일부 실시예에서 다음과 같은 기술적 솔루션이 구현될 수 있다.

[0631] B1. 비디오 처리 방법에 있어서, 비디오의 현재 비디오 블록을 비디오의 비트스트림 표현으로 인코딩 하는 단계 -현재 비디오 블록은 적응형 루프 필터(ALF)로 코딩됨 -; 및 하나 이상의 시간 적응형 필터 세트의 가용성 또는 사용에 기초하여, 비트스트림 표현에서 시간 적응형 필터의 하나 이상의 세트 내에 시간 적응형 필터 세트의 표시를 선택적으로 포함하는 단계를 포함한다.

[0632] B2. B1의 방법에 있어서, 시간 적응형 필터의 세트가 사용할 수 없는 경우, 세트의 표시는 비트스트림 표현에서 제외된다.

[0633] B3. B1 또는 B2의 방법에 있어서, 시간 적응형 필터의 세트를 사용할 수 없는 경우, 그 이후 세트의 표시가 비트스트림 표현에 포함되고, 4. 제1항 내지 제3항 중 어느 한 항에 있어서, 하나 이상의 시간 적응형 필터 세트 중 어느 것도 사용할 수 없는 경우, 표시는 비트스트림 표현에서 제외된다.

[0634] B4. B1 내지 B3 중 어느 하나의 방법에 있어서, 시간 적응형 필터의 하나 이상의 세트 각각은 필터 인덱스와 연관된다.

[0635] B5. B1 내지 B3 중 어느 하나의 방법에 있어서, 하나 이상의 시간 적응형 필터 세트 중 어느 것도 사용할 수 없는 경우 고정 필터를 사용하는 표시는 참과 동일해야 한다.

[0636] B6. B1 내지 B3 중 어느 하나의 방법에 있어서, 하나 이상의 시간 적응형 필터 세트 중 어느 것도 사용할 수 없는 경우 시간 적응형 필터를 사용하는 표시는 거짓과 동일해야 한다.

[0637] B7. B1 내지 B3 중 어느 하나의 방법에 있어서, 하나 이상의 시간 적응형 필터 세트 중 어느 것도 사용 가능하지 않은 경우, 고정 필터의 인덱스의 표시가 비트스트림 표현에 포함된다.

[0638] B8. 비디오 처리를 위한 방법으로서, 비디오의 비트스트림 표현에서 시간 적응형 필터 세트의 표시에 기초하여, 적응형 루프 필터(ALF)로 코딩 된 비디오의 현재 비디오 블록에 적용 가능한 시간 적응형 필터의 세트를 포함하는 시간 적응형 필터의 하나 이상의 세트의 가용성 또는 사용을 결정하는 단계; 및 결정하는 단계에 기초하여, 시간 적응형 필터의 세트를 선택적으로 적용함으로써 비트스트림 표현으로부터 디코딩 된 현재 비디오 블록을 생성하는 단계를 포함한다.

[0639] B9. B8의 방법에 있어서, 시간 적응형 필터의 세트를 사용할 수 없는 경우, 그 이후 생성하는 단계는 시간 적응형 필터의 세트를 적용하지 않고 수행된다.

[0640] B10. B8 또는 B9의 방법에 있어서, 시간 적응형 필터의 세트를 사용할 수 없는 경우, 그 이후 생성하는 단계를 수행하는 단계는 시간 적응형 필터의 세트를 적용하는 단계를 포함한다.

[0641] B11. B1 내지 B10 중 어느 하나의 방법에 있어서, 시간 적응형 필터의 하나 이상의 세트는 적응형 파라미터 세트(APS)에 포함되고, 표시는 APS 인덱스이다.

[0642] B12. B1 내지 B10 중 어느 하나의 방법에 있어서, 상이한 방향의 그레디언트 계산에 기초하여, 하나 이상의 시간 적응형 필터 세트 중 적어도 하나에 대한 필터 인덱스를 결정하는 단계를 더 포함한다.

[0643] B13. B1 내지 B11 중 어느 하나의 방법에 있어서, 시간 적응형 필터의 하나 이상의 세트 중 어느 것도 사용할 수 없고 새로운 ALF 계수 세트 및 고정 ALF 계수 세트가 현재 비디오 블록을 포함하는 코딩 트리 블록(CTB), 블록, 타일 그룹, 타일, 슬라이스 또는 픽처에 사용되지 않는다고 결정하는 단계; 및 결정하는 단계에 기초하여, 적응형 루프 필터링이 비활성화 되어 있음을 유추하는 단계를 포함한다.

[0644] B14. B1 내지 B11 중 어느 하나의 방법에 있어서, 비트스트림 표현은 새로운 ALF 계수 세트의 사용에 대한 제1 표시 및 시간 적응형 필터의 하나 이상의 세트 중 적어도 하나가 사용 불가능하다는 것에 응답하여 고정 ALF 계수 세트의 사용의 제2 표시를 포함하고, 및 제1 표시 및 제2 표시 중 정확히 하나가 비트스트림 표현에서 참이다.

[0645] B15. B14의 방법에 있어서, 비트스트림 표현은 ALF의 작업과 연관된 포맷 규칙을 따른다.

[0646] B16. B1 내지 B11 중 어느 하나의 방법에 있어서, 시간 적응형 필터의 하나 이상의 세트 중 어느 것도 사용 가능하지 않은 것에 응답하여, 비트스트림 표현은 ALF가 활성화되고 새로운 ALF 계수 세트 및 고정된 ALF 계수 세트가 현재 비디오 블록을 포함하는 코딩 트리 블록(CTB), 블록, 타일 그룹, 타일, 슬라이스 또는 픽처에서 사용되지 않는다는 표시를 포함한다.

[0647] B17. B16의 방법에 있어서, 비트스트림 표현이 ALF의 작업과 연관된 포맷 규칙을 따르지 않는다.

[0648] B18. B1 내지 B17 중 어느 하나의 방법에 있어서, ALF가 현재 비디오 블록과 연관된 하나 이상의 컬러 컴포넌트에 적용된다.

[0649] B19. 비디오 처리 방법에 있어서, 적응형 루프 필터로 코딩 된 현재 비디오 블록에 대해, 사용 가능한 시간적 ALF 계수 세트에 기초하여 시간 적응형 루프 필터링(ALF) 계수 세트의 수를 결정하는 단계 - 사용 가능한 시간적 ALF 계수 세트는 결정하는 단계 이 전에 인코딩 또는 디코딩 되었고, ALF 계수 세트의 수는 현재 비디오 블록을 포함하는 타일 그룹, 타일, 슬라이스, 픽처, 코딩 트리 블록(CTB), 또는 비디오 유닛을 위해 사용됨 -; 및 시간적 ALF 계수 세트의 수에 기초하여, 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다.

[0650] B20. B19의 방법에 있어서, 시간적 ALF 계수 세트의 수의 최대 수가 사용 가능한 시간적 ALF 계수 세트의 수와 동일하게 설정된다.

[0651] B21. B20의 방법에 있어서, 시간적 ALF 계수 세트의 수는 사용 가능한 시간적 ALF 계수 세트의 수 및 미리 정의된 수 N 중 더 작은 것과 동일하게 설정되고, N은 정수이고, $N \geq 0$이다.

[0652] B22. B21의 방법에 있어서, $N = 5$이다.

[0653] B23. 비디오 처리 방법에 있어서, 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환의 일부로, 하나 이상의 새로운 적응형 루프 필터링(ALF) 계수 세트를 처리하는 단계 - 현재 비디오 블록은 적응형 루프 필터로 코딩됨 -; 및 처리에 후속하여, 하나 이상의 새로운 ALF 계수 세트를 사용 가능한 ALF 계수 세트로 지정하는 단계를 포함한다.

[0654] B24. B23의 방법에 있어서, 인트라 랜덤 액세스 포인트(IRAP) 액세스 유닛, IRAP 픽처, 인스턴트 디코딩 리프레시(IDR) 액세스 유닛 또는 IDR 픽처를 접하는 단계; 및 접하는 단계에 기초하여, 사용 가능한 ALF 계수 세트를 사용 불가능한 ALF 계수 세트로 지정하는 단계를 포함한다.

[0655] B25. B23 또는 B24의 방법에 있어서, 사용 가능한 ALF 계수 세트 중 적어도 하나는 현재 비디오 블록에 후속하는 비디오 블록에 대한 시간적 ALF 계수 세트이다.

[0656] B26. B23 내지 B25 중 어느 하나의 방법에 있어서, 사용 가능한 ALF 계수 세트는 최대 크기가 N인 ALF 계수 세트 목록에서 유지되며, 여기서 N은 정수이다.

[0657] B27. B26의 방법에 있어서, ALF 계수 세트 목록은 선입선출(FIFO) 순서로 유지된다.

[0658] B28. B1 내지 B27 중 어느 하나의 방법에 있어서, 현재 비디오 블록과 연관된 각 시간 계층에 대해 하나의 ALF 계수 세트 목록이 유지된다.

[0659] B29. B1 내지 B27 중 어느 하나의 방법에 있어서, 현재 비디오 블록과 연관된 K개의 이웃 시간 계층에 대해 하나의 ALF 계수 세트 목록이 유지된다.

[0660] B30. B1 내지 B27 중 어느 하나의 방법에 있어서, 제1 ALF 계수 세트 목록은 현재 비디오 블록을 포함하는 현재 픽처에 대해 유지되고, 제2 ALF 계수 세트 목록은 현재 픽처에 후속하는 픽처에 대해 유지된다.

[0661] B31. B30의 방법에 있어서, 현재 픽처에 기초하여 현재 픽처에 후속하는 픽처가 예측되고, 제1 ALF 계수 세트 목록은 제2 ALF 계수 세트 목록과 동일하다.

[0662] B32. B30의 방법에 있어서, 현재 픽처는 현재 픽처에 후속하는 픽처 및 현재 픽처 이전의 픽처에 기초하여 예측되고, 제1 ALF 계수 세트 목록은 제2 ALF 계수 세트 목록과 동일하다.

[0663] B33. B23의 방법에 있어서, 인트라 랜덤 액세스 포인트(IRAP) 액세스 유닛, IRAP 픽처, 인스턴트 디코딩 리프레시(IDR) 액세스 유닛 또는 IDR 픽처를 접하는 단계; 및 접하는 단계에 후속하여, 하나 이상의 ALF 계수 세트 목록을 비우는 단계를 포함한다.

[0664] B34. B23의 방법에 있어서, 상이한 ALF 계수 세트 목록은 현재 비디오 블록과 연관된 다른 컬러 컴포넌트에 대해 유지된다.

[0665] B35. B34의 방법에 있어서, 상이한 컬러 컴포넌트가 루마 컴포넌트, Cr 컴포넌트, 및 Cb 컴포넌트 중 하나 이상을 포함한다.

[0666] B36. B23의 방법에 있어서, 하나의 ALF 계수 세트 목록은 복수의 픽처, 타일 그룹, 타일, 슬라이스, 또는 코딩 트리 유닛(CTU)에 대해 유지되고, 및 하나의 ALF 계수 세트 목록의 인덱싱은 복수의 픽처, 타일 그룹, 타일, 슬라이스, 또는 코딩 트리 유닛(CTU) 각각에 대해 상이하다.

[0667] B37. B36의 방법에 있어서, 인덱싱은 오름차순이고, 및 현재 비디오 블록과 연관된 제1 시간 계층 인덱스 및 현재 비디오 블록을 포함하는 현재 픽처, 타일 그룹, 타일, 슬라이스, 또는 코딩 트리 유닛(CTU)과 연관된 제2 시간 계층 인덱스에 기초한다.

[0668] B38. B36의 방법에 있어서, 인덱싱은 오름차순이고, 및 현재 비디오 블록과 연관된 픽처 순서 카운트(POC) 및 현재 비디오 블록을 포함하는 현재 픽처, 타일 그룹, 타일, 슬라이스, 또는 코딩 트리 유닛(CTU)과 연관된 제2 POC에 기초한다.

[0669] B39. B36의 방법에 있어서, 인덱싱은 사용 가능한 ALF 계수 세트에 할당된 가장 작은 인덱스를 포함한다.

[0670] B40. B23의 방법에 있어서, 변환은 클리핑 작업을 포함하고, 및 방법은: 현재 비디오 블록의 샘플에 대해, 샘플의 이웃 샘플을 복수의 그룹으로 분류하는 단계; 및 복수의 그룹 각각에 대한 클리핑 작업을 위해, 비트스트림 표현에서 시그널링 된, 단일 세트의 파라미터를 사용하는 단계를 포함한다.

[0671] B41. B23의 방법에 있어서, 변환은 클리핑 작업을 포함하고, 클리핑 작업을 위한 파라미터 세트는 하나 이상의 새로운 ALF 계수 세트에 대해 미리 정의된다.

[0672] B42. B23의 방법에 있어서, 변환은 클리핑 작업을 포함하고, 클리핑 작업을 위한 파라미터 세트는 하나 이상의 새로운 ALF 계수 세트에 대한 비트스트림 표현에서 시그널링 된다.

[0673] B43. 비디오 처리 방법에 있어서, 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역의 헤더에서 적응형 루프 필터링(ALF)의 표시가 비트스트림 표현과 연관된 적응형 파라미터 세트(APS) 네트워크 추상 계층(NAL) 유닛에서 ALF의 표시와 동일하다고 결정하는 단계; 및 변환을 수행하는 단계를 포함한다.

[0674] B44. B43의 방법에 있어서, 비디오 영역은 픽처이다.

[0675] B45. B43의 방법에 있어서, 비디오 영역은 슬라이스이다.

[0676] B46. 비디오 처리 방법에 있어서, 비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역에 의해 사용되는 적응형 루프 필터의 유형에 기초하여 비선형 적응형 루프 필터링(ALF) 작업을 선택적으로 활성화 하는 단계; 및 선택적으로 활성화 단계에 후속하여, 변환을 수행하는 단계를 포함한다.

[0677] B47. B46의 방법에 있어서, 비디오 영역은 코딩 트리 유닛(CTU)이고, 및 비선형 ALF 작업은 적응형 루프 필터의 유형이 고정 ALF 세트 또는 시간적 ALF 세트를 포함한다고 결정되면 비활성화 된다.

[0678] B48. B46의 방법에 있어서, 비디오 영역은 슬라이스, 타일 그룹, 타일 또는 코딩 트리 유닛(CTU)이고, 및 비선

형 ALF 작업은 적응형 루프 필터의 유형이 고정 ALF 세트를 포함한다고 결정되면 활성화 된다.

[0679] B49. B46의 방법에 있어서, 비선형 ALF 작업을 위한 하나 이상의 클리핑 파라미터를 비트스트림 표현에서 선택적으로 시그널링 하는 단계를 더 포함한다.

[0680] B50. B49의 방법에 있어서, 하나 이상의 클리핑 파라미터가 시그널링 된다.

[0681] B51. B49의 방법에 있어서, 하나 이상의 클리핑 파라미터가 비트스트림 표현에서 시그널링되는 ALF 필터 계수 세트에 대해 시그널링 된다.

[0682] B52. B49의 방법에 있어서, 적응형 루프 필터의 유형이 고정 ALF 세트를 포함한다고 결정되면 하나 이상의 클리핑 파라미터가 시그널링 된다.

[0683] B53. B19 내지 B52 중 어느 하나의 방법에 있어서, 변환은 비트스트림 표현으로부터 현재 비디오 블록을 생성한다.

[0684] B54. B19 내지 B52 중 어느 하나의 방법에 있어서, 변환은 현재 비디오 블록으로부터 비트스트림 표현을 생성한다.

[0685] B55. 프로세서 및 명령이 포함된 비일시적 메모리를 포함하는 비디오 시스템의 장치에 있어서, 프로세서에 의해 실행하는 명령으로 하여금, 제1항 내지 제54항 중 어느 한 항에 기재된 방법을 구현하게 하는 장치.

[0686] B56. 비일시적 컴퓨터 판독 가능 매체에 저장된 컴퓨터 프로그램에 있어서, 제1항 내지 제54항 중 어느 한 항에 기재된 방법을 수행하기 위한 프로그램 코드를 포함하는 컴퓨터 프로그램.

[0687] 일부 실시예에서 다음과 같은 기술적 솔루션이 구현될 수 있다.

[0688] C1. 비디오 처리 방법에 있어서, 현재 비디오 블록에 대해, 적어도 하나의 중간 결과를 갖는 둘 이상의 작업을 포함하는 필터링 프로세스를 수행하는 단계; 적어도 하나의 중간 결과에 클리핑 작업을 적용하는 단계; 및 필터링 작업에 기초하여, 현재 비디오 블록을 현재 비디오 블록의 비트스트림 표현으로 변환하는 것을 수행하는 단계를 포함한다.

[0689] C2. C1의 방법에 있어서, 현재 비디오 블록의 샘플에 대해, 샘플의 이웃 샘플을 복수의 그룹으로 분류하는 단계를 더 포함하고, 클리핑 작업은 복수의 그룹 각각의 중간 결과에 상이한 파라미터와 함께 적용된다.

[0690] C3. C2의 방법에 있어서, 적어도 하나의 중간 결과는 현재 샘플과 복수의 그룹 각각의 이웃 샘플 간의 차이의 가중 평균을 포함한다.

[0691] C4. C2의 방법에 있어서, 필터링 프로세스는 필터 계수를 사용하고, 적어도 하나의 중간 결과는 필터 계수의 가중 합 및 현재 샘플과 이웃 샘플 간의 차이를 포함한다.

[0692] C5. C1의 방법에 있어서, 현재 비디오 블록의 샘플의 복수의 이웃 샘플은 필터 계수를 공유하고, 클리핑 작업은 복수의 이웃 샘플 각각에 한 번 적용된다.

[0693] C6. C5의 방법에 있어서, 필터링 작업과 연관된 필터 형태가 대칭 모드에 있다.

[0694] C7. C5 또는 C6의 방법에 있어서, 클리핑 작업의 하나 이상의 파라미터는 비트스트림 표현에서 시그널링 된다.

[0695] C8. C1 내지 C7 중 어느 하나의 방법에 있어서, 클리핑 작업은 K(min, max, input)로 정의되고, 여기서 입력은 클리핑 작업에 대한 입력이고, min은 클리핑 작업의 출력의 공칭 최소값, max는 클리핑 작업의 출력의 공칭 최대값이다.

[0696] C9. C8의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값보다 작고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값보다 크다.

[0697] C10. C8의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값과 동일하고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값보다 크다.

[0698] C11. C8의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값보다 작고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값과 동일하다.

[0699] C12. C8의 방법에 있어서, 클리핑 작업의 출력의 실제 최대값은 공칭 최대값과 동일하고, 클리핑 작업의 출력의 실제 최소값은 공칭 최소값과 동일하다.

[0700] C13. 비디오 처리 방법에 있어서, 시간 적응형 루프 필터링 계수 세트의 비가용성에 기초하여, 비트스트림 표현이 시간 적응형 루프 필터링 계수 세트의 표시를 생략하도록 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다.

[0701] C14. C13의 방법에 있어서, 새로운 적응형 루프 필터링(ALF) 계수 및 고정 ALF 계수가 현재 비디오 블록을 포함하는 코딩 트리 블록(CTB), 블록, 타일 그룹, 타일, 슬라이스 또는 픽처에서 사용되지 않는다고 결정하는 단계; 및 적응형 루프 필터링이 비활성화되었음을 추론하는 단계를 더 포함한다.

[0702] C15. C13의 방법에 있어서, 적합성 비트스트림은 새로운 적응형 루프 필터링(ALF) 계수의 표시 또는 고정 ALF 계수의 표시를 포함한다.

[0703] C16. 비디오 처리 방법에 있어서, 현재 비디오 블록에 대해, 사용 가능한 시간 ALF 계수 세트에 기초하여 하나 이상의 시간 적응형 루프 필터링(ALF) 계수 세트를 결정하는 단계를 포함하고, 사용 가능한 시간적 ALF 계수 세트는 결정하는 단계 이전에 인코딩 또는 디코딩 되고; 및 하나 이상의 시간적 ALF 계수 세트에 기초하여, 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다.

[0704] C17. C16의 방법에 있어서, 하나 이상의 시간적 ALF 계수 세트의 최대 수는 ALFavailable이다.

[0705] C18. C17의 방법에 있어서, 하나 이상의 시간적 ALF 계수 세트의 수는 min(N, ALFavailable)이고, N은 정수이고, $N \geq 0$이다.

[0706] C19. C18의 방법에 있어서, $N = 5$이다.

[0707] C20. 비디오 처리 방법에 있어서, 현재 비디오 블록에 대해, 하나 이상의 새로운 적응형 루프 필터링(ALF) 계수 세트를 처리하는 단계; 처리하는 단계에 후속하여, 하나 이상의 새로운 ALF 계수 세트를 사용 가능한 ALF 계수 세트로서 지정하는 단계; 및 사용 가능한 ALF 계수 세트에 기초하여, 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계를 포함한다.

[0708] C21. C20의 방법에 있어서, 인트라 랜덤 액세스 포인트(IRAP) 액세스 유닛, IRAP 픽처, 인스턴트 디코딩 리프레시(IDR) 액세스 유닛 또는 IDR 픽처를 접하는 단계; 및 사용 가능한 ALF 계수 세트를 사용 불가능한 ALF 계수 세트로 지정하는 단계를 포함한다.

[0709] C22. C20 또는 C21의 방법에 있어서, 사용 가능한 ALF 계수 세트는 현재 비디오 블록에 후속하는 비디오 블록에 대한 시간적 ALF 계수 세트이다.

[0710] C23. C20 내지 C22 중 어느 하나의 방법에 있어서, 사용 가능한 ALF 계수 세트는 최대 크기가 N인 ALF 계수 세트 목록에서 유지되고, 여기서 N은 정수이다.

[0711] C24. C23의 방법에 있어서, ALF 계수 세트 목록은 FIFO(선입 선출) 순서로 유지된다.

[0712] C25. C13 내지 C24 중 어느 하나의 방법에 있어서, 현재 비디오 블록과 연관된 각 시간 계층에 대해 하나의 ALF 계수 세트 목록이 유지된다.

[0713] C26. C13 내지 C24 중 어느 하나의 방법에 있어서, 현재 비디오 블록과 연관된 K개의 이웃하는 시간 계층에 대해 하나의 ALF 계수 세트 목록이 유지된다.

[0714] C27. C13 내지 C24 중 어느 하나의 방법에 있어서, 제1 ALF 계수 세트 목록은 현재 비디오 블록을 포함하는 현재 픽처에 대해 유지되고, 제2 ALF 계수 세트 목록은 현재 픽처에 후속하는 픽처에 대해 유지된다.

[0715] C28. C27의 방법에 있어서, 현재 픽처에 후속하는 픽처는 현재 픽처에 기초하여 예측되고, 제1 ALF 계수 세트 목록은 제2 ALF 계수 세트 목록과 동일하다.

[0716] C29. C20의 방법에 있어서, 인트라 랜덤 액세스 포인트(IRAP) 액세스 유닛, IRAP 픽처, 인스턴트 디코딩 리프레시(IDR) 액세스 유닛 또는 IDR 픽처를 접하는 단계; 및 접하는 단계에 후속하여, 하나 이상의 ALF 계수 세트 목록을 비우는 단계를 포함한다.

[0717] C30. C20의 방법에 있어서, 현재 비디오 블록의 다른 색상 구성 요소에 대해 다른 ALF 계수 세트 목록이 유지된다.

[0718] C31. 프로세서 및 명령어가 포함된 비일시적 메모리를 포함하는 비디오 시스템의 장치로서, 프로세서에 의한 실행 시 명령은 프로세서로 하여금 C1 내지 C30 중 어느 하나의 방법을 구현하게 하는 장치.

[0719] C32. 비일시적 컴퓨터 판독 가능 매체에 저장된 컴퓨터 프로그램에 있어서, C1 내지 C30 중 어느 하나의 방법을 수행하기 위한 프로그램 코드를 포함하는 컴퓨터 프로그램.

[0720] 도 12는 비디오 처리 장치(1200)의 블록도이다. 장치(1200)는 여기에 설명된 방법들 중 하나 이상을 구현하는 데 사용될 수 있다. 장치(1200)는 스마트폰, 태블릿, 컴퓨터, 사물인터넷(IoT) 수신기 등으로 구현될 수 있다. 장치(1200)는 하나 이상의 프로세서(1202), 하나 이상의 메모리(1204) 및 비디오 처리 하드웨어(1206)를 포함할 수 있다. 프로세서(들)(1202)는 본 문서에 설명된 하나 이상의 방법(방법(1100 및 1150)을 포함하지만 이에 제한되지 않음)을 구현하도록 구성될 수 있다. 메모리(메모리)(1204)는 여기에 설명된 방법 및 기술을 구현하는 데 사용되는 데이터 및 코드를 저장하는 데 사용될 수 있다. 비디오 처리 하드웨어(1206)는 하드웨어 회로에서 본 문서에 설명된 일부 기술을 구현하는 데 사용될 수 있다.

[0721] 일부 실시예에서, 비디오 코딩 방법은 도 12와 관련하여 설명된 바와 같이 하드웨어 플랫폼 상에서 구현되는 장치를 사용하여 구현될 수 있다.

[0722] 개시된 기술의 일부 실시예는 비디오 처리 도구 또는 모드를 가능하게 하기 위한 결정 또는 결정을 내리는 단계를 포함한다. 예에서 비디오 처리 도구 또는 모드가 활성화되면 인코더는 비디오 블록의 처리에서 도구 또는 모드를 사용하거나 구현하지만 도구 또는 모드의 사용에 따라 결과 비트스트림을 반드시 수정할 필요는 없다. 즉, 비디오 블록에서 비디오의 비트스트림 표현으로의 변환은 결정 또는 결정에 따라 활성화될 때 비디오 처리 도구 또는 모드를 사용한다. 다른 예에서, 비디오 처리 툴 또는 모드가 활성화 될 때, 디코더는 비디오 처리 툴 또는 모드에 기초하여 비트스트림이 수정되었다는 지식으로 비트스트림을 처리할 것이다. 즉, 비디오의 비트스트림 표현에서 비디오 블록으로의 변환은 판정(decision) 또는 결정에 기초하여 활성화된 비디오 처리 도구 또는 모드를 사용하여 수행될 것이다.

[0723] 개시된 기술의 일부 실시예는 비디오 처리 도구 또는 모드를 비활성화하기 위한 판정 또는 결정을 내리는 단계를 포함한다. 예에서, 비디오 처리 도구 또는 모드가 비활성화되면 인코더는 비디오 블록을 비디오의 비트스트림 표현으로 변환하는 데 도구 또는 모드를 사용하지 않는다. 다른 예에서, 비디오 처리 툴 또는 모드가 비활성화될 때, 디코더는 판정 또는 결정에 기초하여 활성화된 비디오 처리 툴 또는 모드를 사용하여 비트스트림이 수정되지 않았다는 지식으로 비트스트림을 처리할 것이다.

[0724] 도 13은 여기에 개시된 다양한 기술들이 구현될 수 있는 예시적인 비디오 처리 시스템(video processing system)(1300)을 도시하는 블록도이다. 다양한 구현은 시스템(1300)의 구성요소의 일부 또는 전부를 포함할 수 있다. 시스템(1300)은 비디오 콘텐츠를 수신하기 위한 입력(1302)을 포함할 수 있다. 비디오 콘텐츠는 원시 또는 압축되지 않은 형식, 예를 들어 8 또는 10비트 다중 구성요소 픽셀 값으로 수신될 수 있거나 압축 또는 인코딩 된 형식일 수 있다. 입력(1302)은 네트워크 인터페이스, 주변 버스 인터페이스, 또는 저장 인터페이스를 나타낼 수 있다. 네트워크 인터페이스의 예로는 이더넷, 수동 광 네트워크(PON) 등과 같은 유선 인터페이스와 Wi-Fi 또는 셀룰러 인터페이스와 같은 무선 인터페이스가 있다.

[0725] 시스템(1300)은 본 문서에 설명된 다양한 코딩 또는 인코딩 방법을 구현할 수 있는 코딩 컴포넌트(1304)를 포함할 수 있다. 코딩 컴포넌트(1304)는 비디오의 코딩 된 표현을 생성하기 위해 입력(1302)으로부터 코딩 컴포넌트(1304)의 출력으로 비디오의 평균 비트레이트를 감소시킬 수 있다. 따라서 코딩 기술은 비디오 압축 또는 비디오 트랜스코딩 기술이라고도 한다. 코딩 컴포넌트(1304)의 출력은 컴포넌트(1306)에 의해 표현되는 바와 같이, 연결된 통신을 통해 저장되거나 전송될 수 있다. 입력(1302)에서 수신된 비디오의 저장되거나 통신된 비트스트림(또는 코딩 된) 표현은 디스플레이 인터페이스(1310)로 전송되는 픽셀 값 또는 디스플레이 가능한 비디오를 생성하기 위해 컴포넌트(1308)에 의해 사용될 수 있다. 비트스트림 표현에서 사용자가 볼 수 있는 비디오를 생성하는 프로세스를 비디오 압축 해제라고도 한다. 또한, 특정 비디오 처리 작업은 "코딩" 작업 또는 도구로 지칭되지만, 코딩 툴 또는 작업은 인코더에서 사용되며, 코딩의 결과를 역전시키는 대응하는 디코딩 툴 또는 작업은 디코더에 의해 수행될 것임을 이해할 것이다.

[0726] 주변 기기 버스 인터페이스 또는 디스플레이 인터페이스의 예는 범용 직렬 버스 (Universal Serial Bus) (USB) 또는 고해상도 멀티미디어 인트페이스(High Definition Multimedia Interface) (HDMI) 또는 디스플레이포트 (Displayport) 등을 포함할 수 있다. 저장 인터페이스의 예는 직렬 고급 기술 어테치먼트 (Serial Advanced Technology Attachment) (SATA), PCI, IDE 인터페이스 등을 포함한다. 본 문서에서 설명된 기술은 디지털 데이터 처리 및/또는 비디오 디스플레이를 수행할 수 있는 휴대폰, 랩톱, 스마트폰 또는 기타 장치와 같은 다양한 전자 장치에서 구현될 수 있다.

[0727] 전술한 내용으로부터, 현재 개시된 기술의 특정 실시예가 예시의 목적으로 여기에서 설명되었지만, 본 발명의 범위를 벗어나지 않고 다양한 수정이 이루어질 수 있다는 것이 이해될 것이다. 따라서, 현재 개시된 기술은 첨부된 청구범위에 의한 경우를 제외하고는 제한되지 않는다.

[0728] 본 특허 문서에 기재된 주제의 구현 및 기능적 작업은 본 사양 및 구조상 등가물 또는 그 중 하나 이상의 조합으로 다양한 시스템, 디지털 전자 회로 또는 컴퓨터 소프트웨어, 펌웨어 또는 하드웨어에서 구현될 수 있다. 본 명세서에 기재된 주제의 구현은 하나 이상의 컴퓨터 프로그램 제품, 즉, 데이터 처리 장치의 작동을 제어하기 위해 실물 및 비일시적 컴퓨터 판독 가능한 매체에 인코딩 된 컴퓨터 프로그램 지침의 하나 이상의 모듈로 구현될 수 있다. 컴퓨터가 읽을 수 있는 매체는 기계가 읽을 수 있는 저장 장치, 기계가 읽을 수 있는 스토리지 기판, 메모리 장치, 기계가 읽을 수 있는 전파 신호에 영향을 미치는 물질의 조성 또는 하나 이상의 조합일 수 있다. 용어 "데이터 처리 장치" 또는 "데이터 처리 장치"는 예를 들어 프로그래밍 가능한 프로세서, 컴퓨터 또는 다중 프로세서 또는 컴퓨터를 포함하여 데이터를 처리하기 위한 모든 장치, 장치 및 기계를 포함한다. 이 장치에는 대응하는 컴퓨터 프로그램에 대한 실행 환경을 만드는 코드(예를 들어, 프로세서 펌웨어, 프로토콜 스택, 데이터베이스 관리 시스템, 운영 체제 또는 하나 이상의 조합)가 포함될 수 있다.

[0729] 컴퓨터 프로그램(프로그램, 소프트웨어, 소프트웨어 응용 프로그램, 스크립트 또는 코드라고도 함)은 컴파일되거나 해석된 언어를 포함하며 독립 실행형 프로그램 또는 모듈, 구성 요소, 서브루틴 또는 컴퓨팅 환경에서 사용하기에 적합한 기타 유닛을 포함하여 모든 형태로 배포될 수 있는 모든 형태의 프로그래밍 언어로 작성할 수 있다. 컴퓨터 프로그램이 파일 시스템의 파일에 반드시 대응하는되는 것은 아니다. 프로그램은 다른 프로그램이나 데이터(예를 들어, 마크업 언어 문서에 저장된 하나 이상의 스크립트), 대응하는 프로그램에 전념하는 단일 파일 또는 여러 조정된 파일(예를 들어, 하나 이상의 모듈, 서브 프로그램 또는 코드 일부를 저장하는 파일)에 저장할 수 있다. 컴퓨터 프로그램은 한 컴퓨터 또는 한 위치에 위치하거나 여러 위치에 분산되고 통신 네트워크에 의해 상호 연결된 여러 컴퓨터에서 실행하도록 배포할 수 있다.

[0730] 이 사양에 설명된 프로세스 및 논리 흐름은 하나 이상의 프로그래밍 가능한 프로세서가 하나 이상의 프로그래밍 가능한 프로세서에서 수행하여 입력 데이터에서 작동하고 출력을 생성하여 기능을 수행할 수 있다. 프로세스 및 로직 흐름도 수행될 수 있으며, 장치는 특수 목적 논리 회로, 예를 들어 FPGA(필드 프로그래밍 가능한 게이트 어레이) 또는 ASIC(응용 프로그램 별 집적 회로)로 구현될 수 있다.

[0731] 컴퓨터 프로그램의 실행에 적합한 프로세서에는 예를 들어 일반 및 특수 목적 마이크로프로세서와 모든 종류의 디지털 컴퓨터의 하나 이상의 프로세서가 포함된다. 일반적으로 프로세서는 읽기 전용 메모리 또는 임의 액세스 메모리 또는 둘 다에서 지침과 데이터를 수신한다. 컴퓨터의 필수 요소는 명령을 수행하기 위한 프로세서와 명령 및 데이터를 저장하기 위한 하나 이상의 메모리 장치이다. 일반적으로, 컴퓨터는 또한 데이터를 저장하기 위한 하나 이상의 대용량 저장 장치, 예를 들어 자기, 광자기 디스크 또는 광 디스크로부터 데이터를 수신하거나 이들로 데이터를 전송하거나 둘 모두를 포함하거나 작동 가능하게 결합된다. 그러나 컴퓨터에는 그러한 장치가 필요하지 않는다. 컴퓨터 프로그램 지침 및 데이터를 저장하는 데 적합한 컴퓨터 판독 가능한 매체에는 반도체 메모리 장치(예를 들어, EPROM, EEPROM 및 플래시 메모리 장치)를 비롯한 모든 형태의 비휘발성 메모리, 매체 및 메모리 장치가 포함된다. 프로세서와 메모리는 특수 목적 논리 회로에 의해 보충되거나 통합될 수 있다.

[0732] 본 특허 문서에는 여러 가지 구체적인 내용이 포함되어 있지만, 이는 발명의 범위나 주장될 수 있는 것의 한계로 해석되어서는 안 되며, 오히려 특정 발명의 특정 실시예에 특정할 수 있는 특징에 대한 설명으로 해석되어야 한다. 이 특허 문서에 기재된 특정 특징은 별도의 실시예의 맥락에서 또한 단일 실시예에서 조합하여 구현될 수 있다. 반대로, 단일 실시예의 문맥상에 기재되는 다양한 특징은 또한 다중 실시예에서 개별적으로 또는 어느 적합한 서브 조합에서 구현될 수 있다. 또한, 특징들이 특정 조합으로 작용하는 것으로 위에서 설명될 수 있고 심지어 초기에 그러한 것으로 청구될 수 있지만, 청구된 조합의 하나 이상의 특징이 경우에 따라 조합에서 제거될 수 있고, 청구된 조합은 서브 조합 또는 서브 조합의 변형에 관한 것일 수 있다.

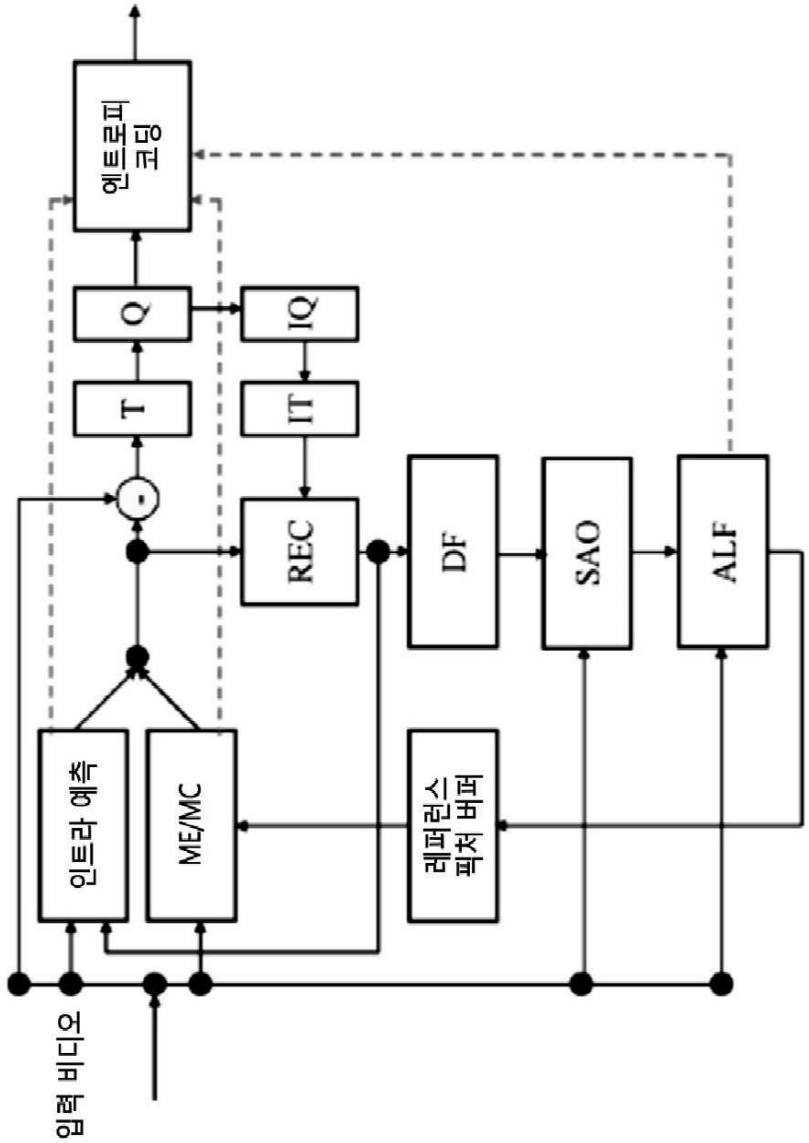
[0733] 마찬가지로, 작업은 특정 순서로 도면에 묘사되는 동안, 이러한 작업이 표시된 특정 순서 또는 순차순서로 수행되거나, 모든 일러스트레이션 작업을 수행하여 바람직한 결과를 달성하도록 요구하는 것으로 이해되어서는 안된다. 또한, 본 특허 문서에 기재된 실시예에서 다양한 시스템 컴포넌트의 분리는 모든 실시예에서 이러한 분리를 요구하는 것으로 이해되어서는 안된다.

[0734] 몇 가지 구현 및 예제만 설명되며 이 특허 문서에 설명되고 설명된 내용에 따라 다른 구현, 개선 및 변형을 만

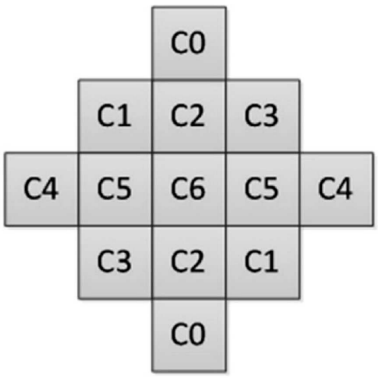
들 수 있다.

도면

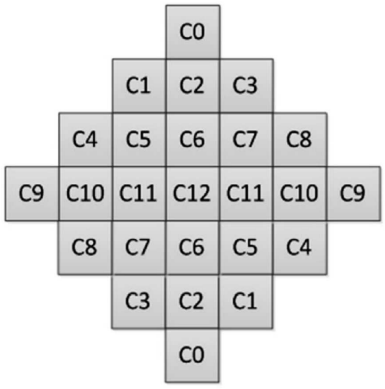
도면1



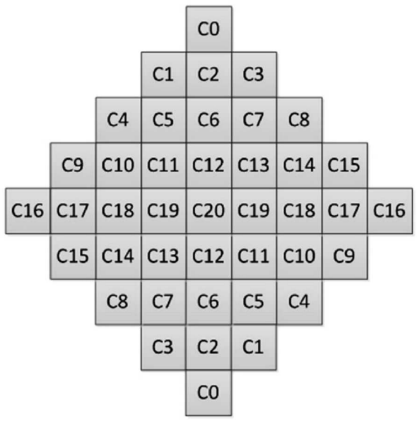
도면2a



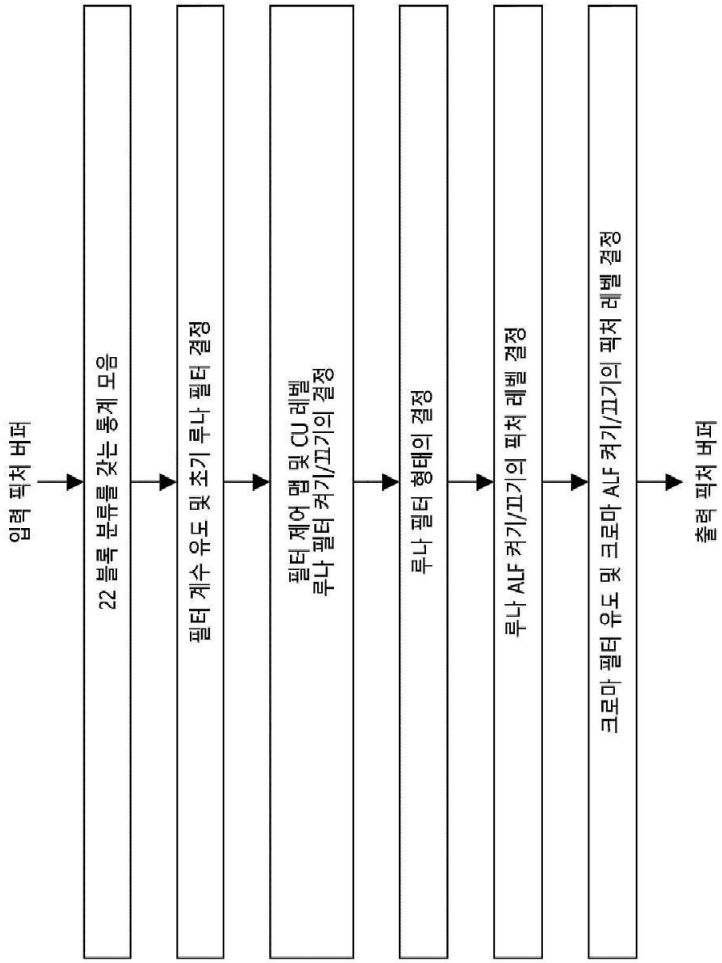
도면2b


C0
C1 C2 C3
C4 C5 C6 C7 C8
C9 C10 C11 C12 C11 C10 C9
C8 C7 C6 C5 C4
C3 C2 C1
C0

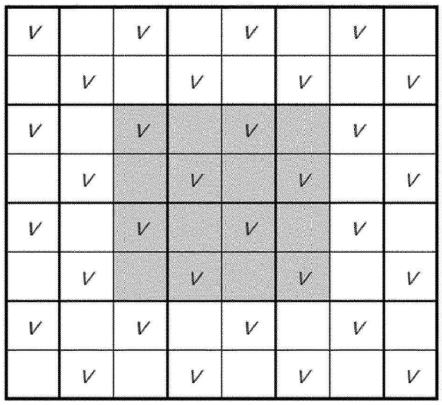
도면2c


C0
C1 C2 C3
C4 C5 C6 C7 C8
C9 C10 C11 C12 C13 C14 C15
C16 C17 C18 C19 C20 C19 C18 C17 C16
C15 C14 C13 C12 C11 C10 C9
C8 C7 C6 C5 C4
C3 C2 C1
C0

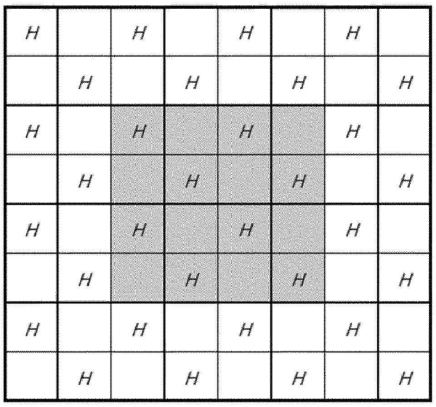
도면3


입력 픽처 버퍼
22 블록 분류를 갖는 통계 모음
필터 계수 유도 및 초기 루나 필터 결정
필터 제어 맵 및 CU 레벨
루나 필터 켜기/끄기의 결정
루나 필터 형태의 결정
루나 ALF 켜기/끄기의 픽처 레벨 결정
크로마 필터 유도 및 크로마 ALF 켜기/끄기의 픽처 레벨 결정
출력 픽처 버퍼

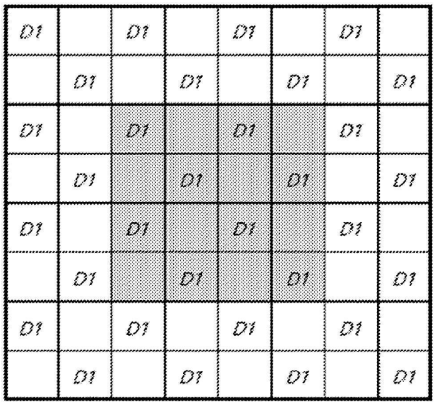
도면4a



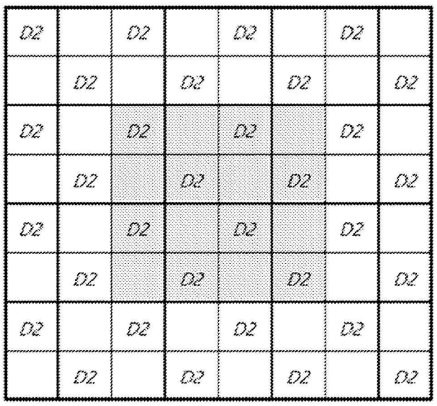
도면4b



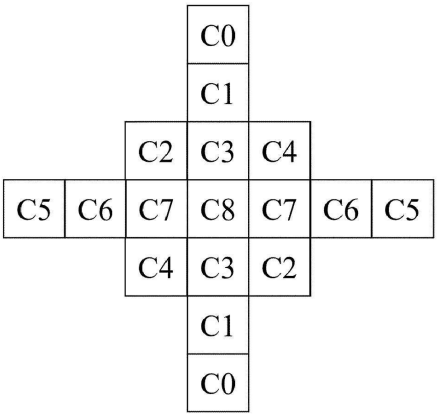
도면4c



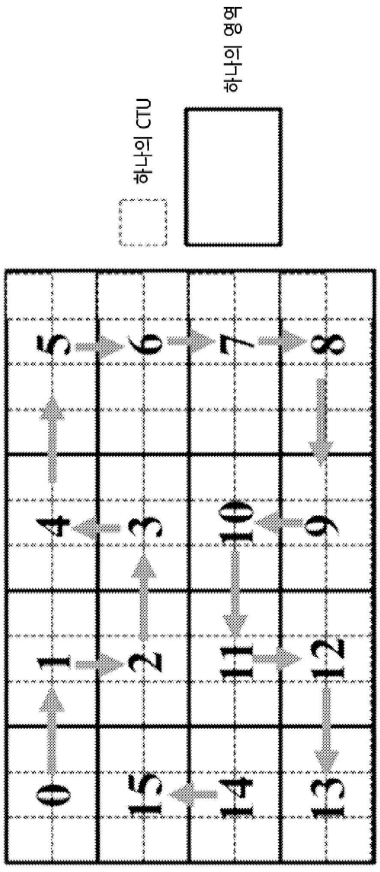
도면4d



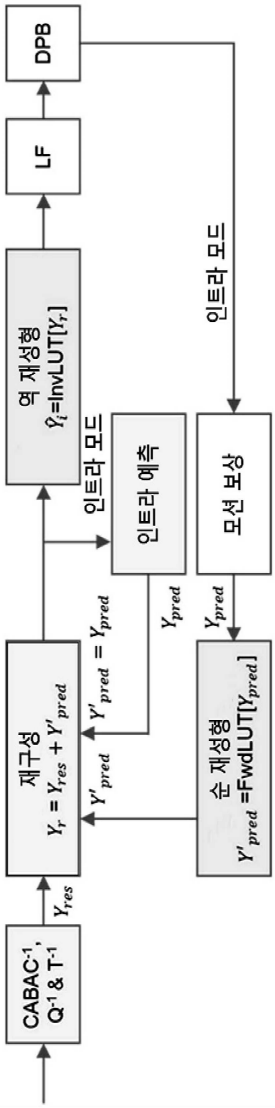
도면5


C0
C1
C2
C3
C4
C5
C6
C7
C8
C7
C6
C5
C4
C3
C2
C1
C0

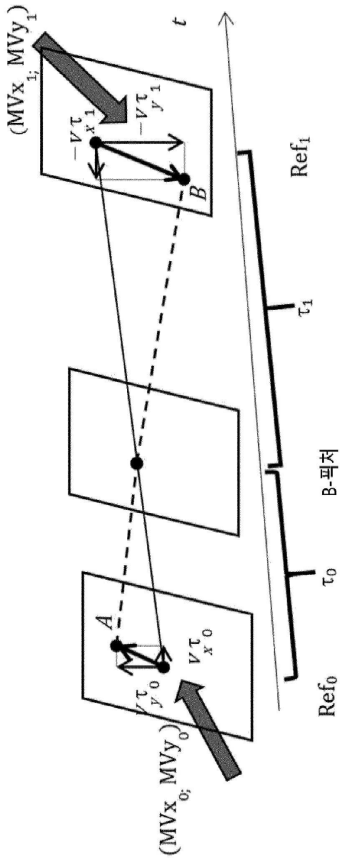
도면6


하위 영역
하나의 CTU
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

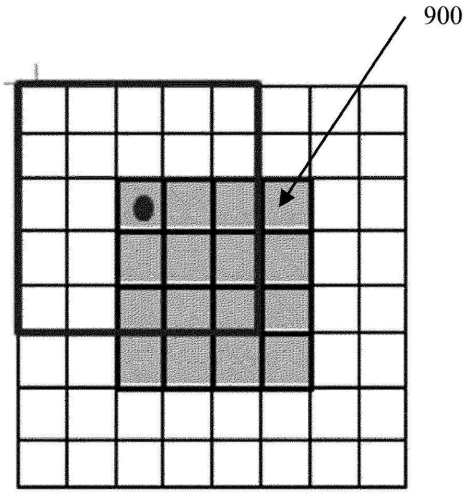
도면7


DPB
LF
역 재성형
$\hat{Y}_r$=InvLUT[Y_r]
인트라 모드
인트라 예측
모션 보상
인트라 모드
재구성
$Y_r = Y_{res} + Y'_{pred}$
$Y'_{pred} = Y_{pred}$
Y'_{pred}
Y_{pred}
Y_{pred}
순 재성형
Y'_{pred} =FwdLUT[Y_{pred}]
Y_{res}
CABAC^{-1}, Q^{-1} & T^{-1}

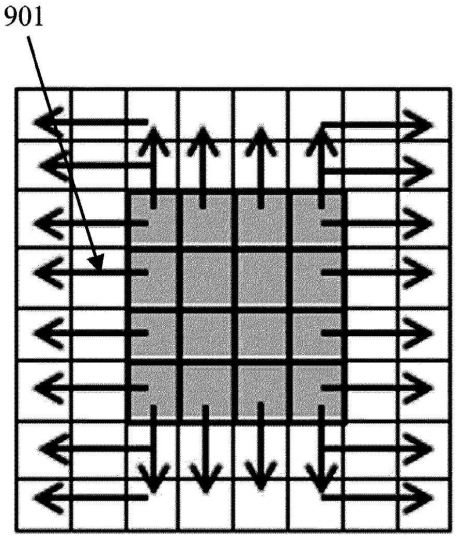
도면8



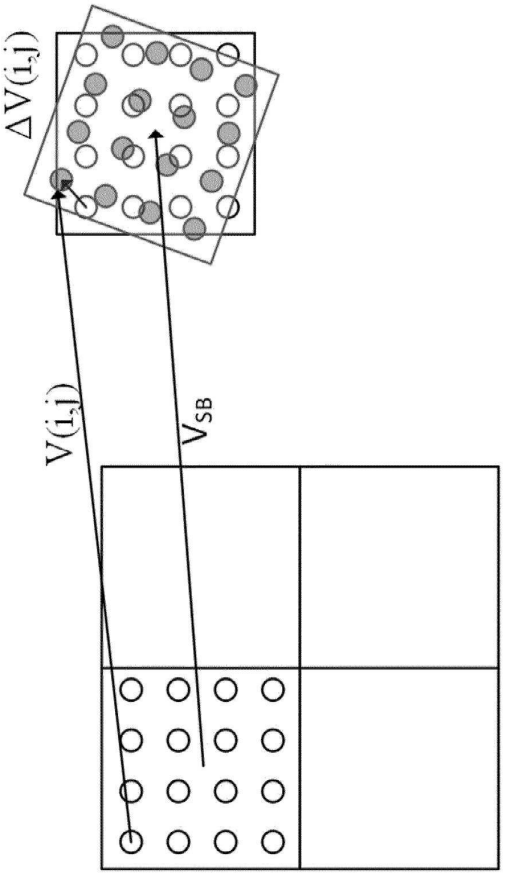
도면9a



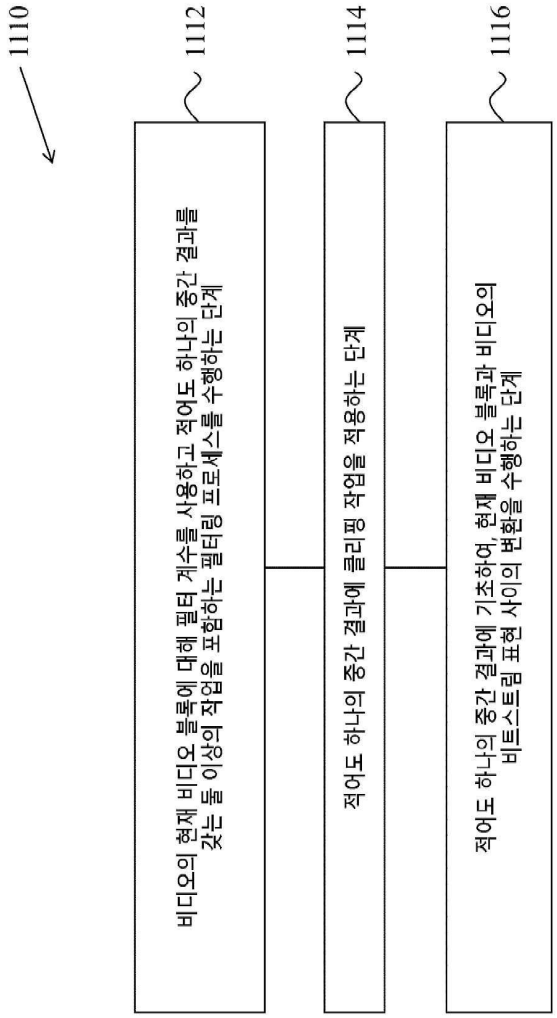
도면9b



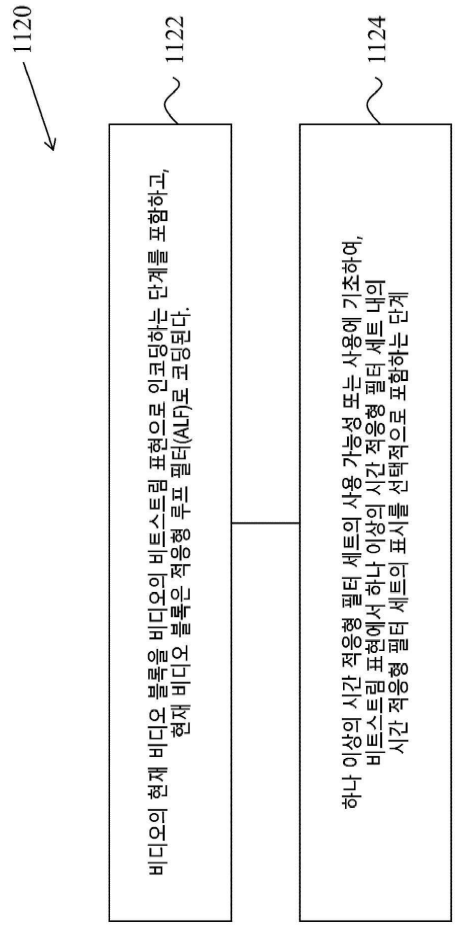
도면10



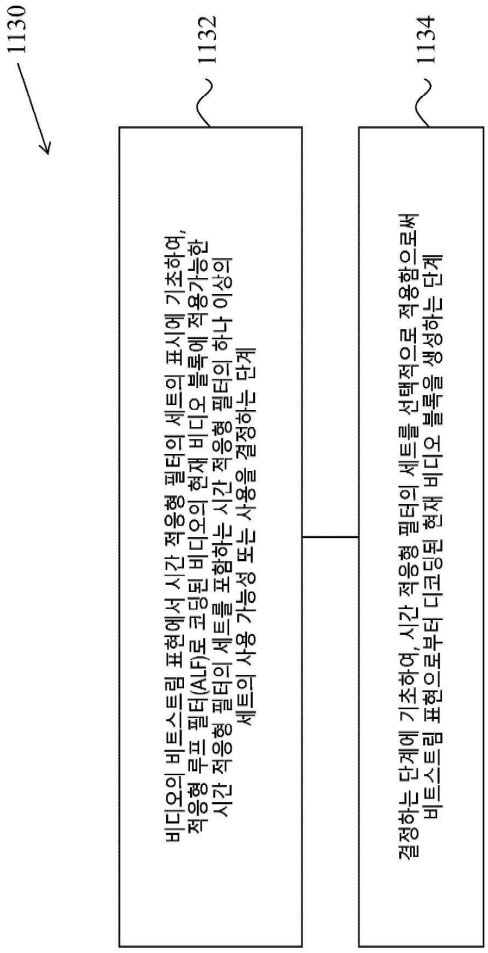
도면11a


1110
1112
비디오의 현재 비디오 블록에 대해 필터 계수를 사용하고 적어도 하나의 중간 결과를 갖는 둘 이상의 작업을 포함하는 필터링 프로세스를 수행하는 단계
1114
적어도 하나의 중간 결과에 클리핑 작업을 적용하는 단계
1116
적어도 하나의 중간 결과에 기초하여, 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 수행하는 단계

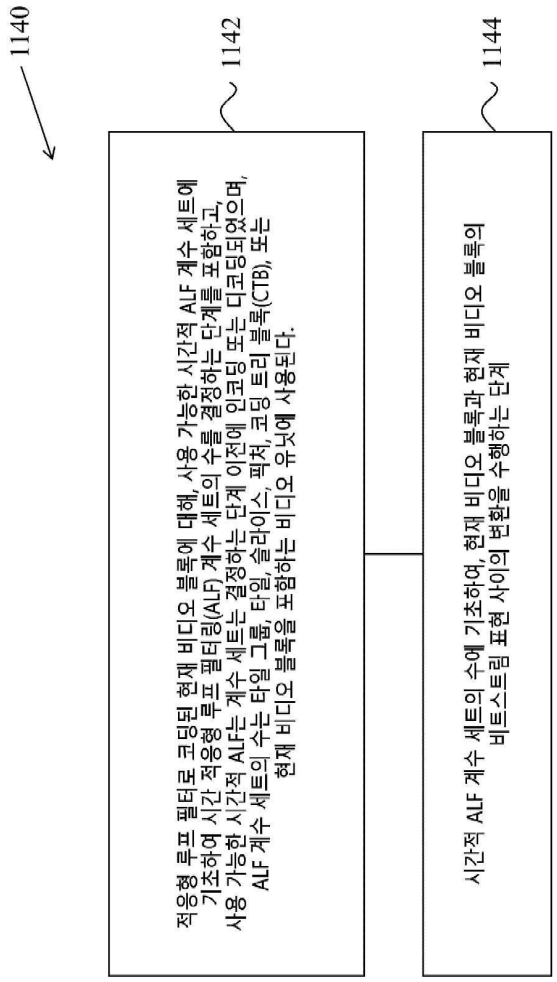
도면11b


1120
1122
비디오의 현재 비디오 블록을 비디오의 비트스트림 표현으로 인코딩하는 단계를 포함하고,
현재 비디오 블록은 적응형 루프 필터(ALF)로 코딩된다.
1124
하나 이상의 시간 적응형 필터 세트의 사용 가능성 또는 사용에 기초하여,
비트스트림 표현에서 하나 이상의 시간 적응형 필터 세트 내의
시간 적응형 필터 세트의 표시를 선택적으로 포함하는 단계

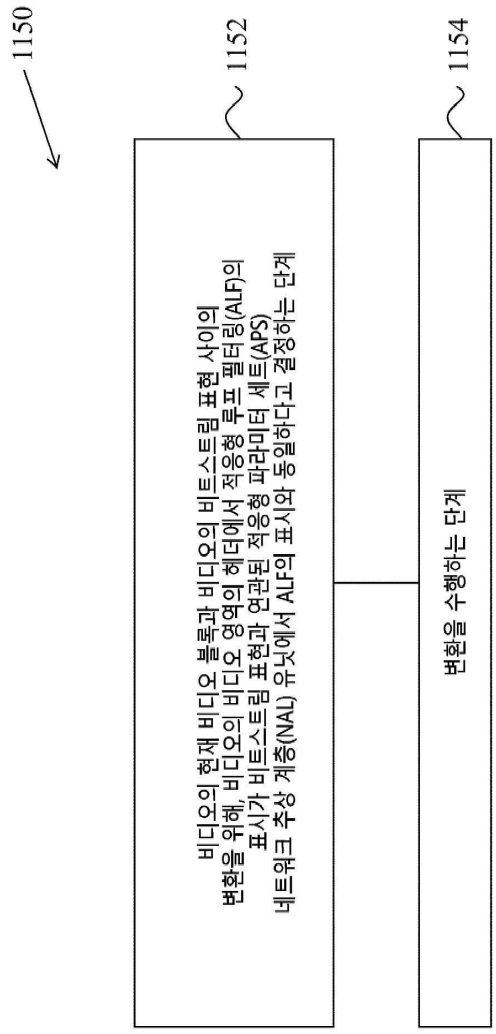
도면11c


1130
1132
비디오의 비트스트림 표현에서 시간 적응형 필터의 세트의 표시에 기초하여, 적응형 루프 필터(ALF)로 코딩된 비디오의 현재 비디오 블록에 적용가능한 시간 적응형 필터의 세트를 포함하는 시간 적응형 필터의 하나 이상의 세트의 사용 가능성 또는 사용을 결정하는 단계
1134
결정하는 단계에 기초하여, 시간 적응형 필터의 세트를 선택적으로 적용함으로써 비트스트림 표현으로부터 디코딩된 현재 비디오 블록을 생성하는 단계

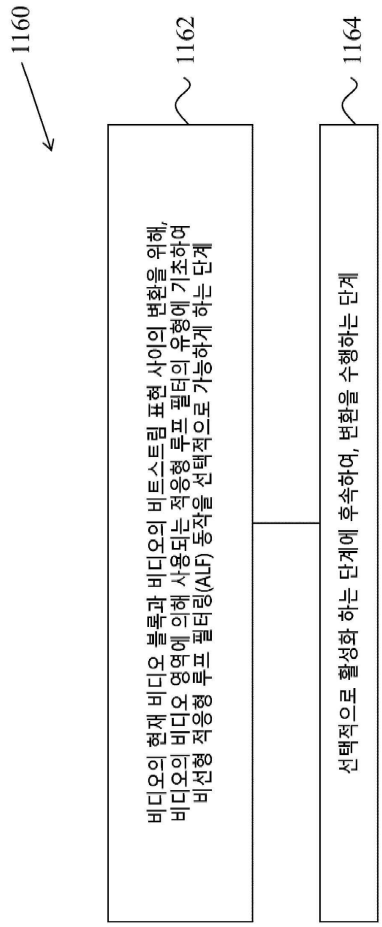
도면11d


1140
1142
적응형 루프 필터로 코딩된 현재 비디오 블록에 대해, 사용 가능한 시간적 ALF 계수 세트에 기초하여 시간 적응형 루프 필터링(ALF) 계수 세트의 수를 결정하는 단계를 포함하고, 사용 가능한 시간적 ALF는 계수 세트는 결정하는 단계 이전에 인코딩 또는 디코딩되었으며, ALF 계수 세트의 수는 타일 그룹, 타일, 슬라이스, 픽처, 코딩 트리 블록(CTB), 또는 현재 비디오 블록을 포함하는 비디오 유닛에 사용된다.
1144
시간적 ALF 계수 세트의 수에 기초하여, 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 사이의 변환을 수행하는 단계

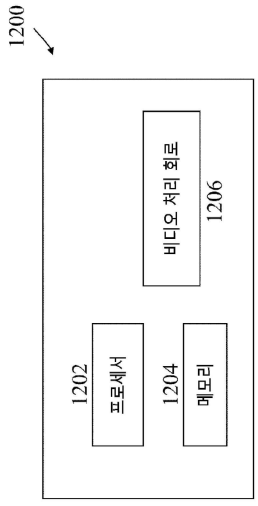
도면11e


1150
1152
1154
비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의
변환을 위해, 비디오의 비디오 영역의 헤더에서 적응형 루프 필터링(ALF)의
표시가 비트스트림 표현과 연관된 적응형 파라미터 세트(APS)
네트워크 추상 계층(NAL) 유닛에서 ALF의 표시와 동일하다고 결정하는 단계
변환을 수행하는 단계

도면11f


1160
1162
1164
비디오의 현재 비디오 블록과 비디오의 비트스트림 표현 사이의 변환을 위해, 비디오의 비디오 영역에 의해 사용되는 적응형 루프 필터의 유형에 기초하여 비선형 적응형 루프 필터링(ALF) 동작을 선택적으로 가능하게 하는 단계
선택적으로 활성화 하는 단계에 후속하여, 변환을 수행하는 단계

도면12


1200
1202
프로세서
1204
메모리
비디오 처리 회로
1206

도면13