(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2025년02월11일
(11) 등록번호 10-2764983
(24) 등록일자 2025년02월04일

(51) 국제특허분류(Int. Cl.)
H04N 19/117 (2014.01) *H04N 19/136* (2014.01)
H04N 19/14 (2014.01) *H04N 19/182* (2014.01)
H04N 19/82 (2014.01) *H04N 19/86* (2014.01)
(52) CPC특허분류
H04N 19/117 (2015.01)
H04N 19/136 (2015.01)
(21) 출원번호 10-2021-7021789
(22) 출원일자(국제) 2019년11월29일
심사청구일자 2021년07월12일
(85) 번역문제출일자 2021년07월12일
(65) 공개번호 10-2021-0099134
(43) 공개일자 2021년08월11일
(86) 국제출원번호 PCT/EP2019/083066
(87) 국제공개번호 WO 2020/126411
국제공개일자 2020년06월25일
(30) 우선권주장
1821156.5 2018년12월21일 영국(GB)
1901775.5 2019년02월08일 영국(GB)
(56) 선행기술조사문헌
EP02477403 A1*
EP03244611 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
캐논 가부시끼가이샤
일본 도쿄도 오오따꾸 시모마루꼬 3쪼메 30방 2고
(72) 발명자
타케 조나단
프랑스 35160 딸랑삭 라 프로아디에르
지스케 크리스토프
프랑스 35690 아시뉴 뤼 뒤 마르땡 베쉐르 43
(뒷면에 계속)
(74) 대리인
장수길, 이중희

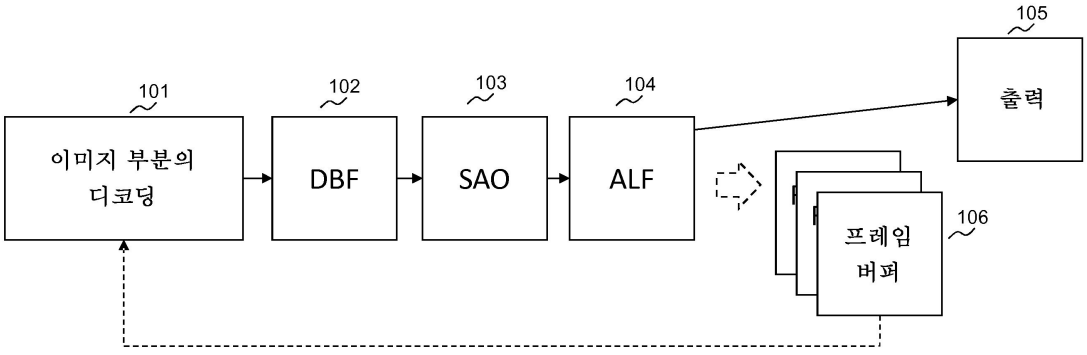
전체 청구항 수 : 총 13 항 심사관 : 김영태

(54) 발명의 명칭 **비선형 클리핑을 갖는 적응 루프 필터링(ALF)**

(57) 요 약

본 발명은 이미지의 하나 이상의 이미지 부분에 대한 적응 루프 필터를 제어하는 방법을 제공하고, 이 방법은 제1 샘플 값의 하나 이상의 이웃하는 샘플 값(들)에 기초하여 이미지 부분의 제1 샘플에 대한 필터링을 제어하는 단계를 포함하며, 여기서 제어하는 단계는 이웃하는 샘플 값(들) 중 하나 이상에 기초한 하나 이상의 변수를 갖는 비선형 함수를 사용한다.

대 표 도



(52) CPC특허분류
H04N 19/14 (2015.01)
H04N 19/182 (2015.01)
H04N 19/82 (2015.01)
H04N 19/86 (2015.01)
(72) 발명자
라로쉬 기욤
프랑스 35250 생또방 도비뉴 라 바스 로므래 5
오노 파트리스
프랑스 35700 렌느 뤼 메슬레 5

명 세 서

청구범위

청구항 1

적응 루프 필터링을 사용하여 이미지의 하나 이상의 부분을 인코딩하는 방법으로서,

비트스트림에 대한 이미지의 데이터를 획득하는 단계;

이미지의 이미지 부분의 현재 샘플을 필터링하는 단계

를 포함하고,

상기 필터링하는 단계는, 상기 현재 샘플의 이웃하는 샘플 값과 상기 현재 샘플 값 사이의 차이 값을 클리핑하기 위한 클리핑 함수로서, 각각이 클리핑 파라미터들을 갖는 클리핑 함수를 포함하는 비선형 방정식을 사용하여 필터링하고,

상기 클리핑 파라미터는 클리핑의 값 범위를 나타내며, 상기 클리핑 함수 각각의 상기 클리핑 파라미터는 상기 이웃하는 샘플의 위치에 의존하는, 방법.

청구항 2

제1항에 있어서,

상기 클리핑 함수들은 max(-b, min(b,d)), min(b, max(-b,d)), max(c-b, min(c+b,n)), 또는 min(c+b, max(c-b,n)) 중 하나를 반환하고;

c는 상기 현재 샘플 값이며, n은 상기 현재 샘플의 이웃하는 샘플 값이고, d=n-c이며, b는 상기 클리핑 파라미터인, 방법.

청구항 3

제2항에 있어서, 클리핑 파라미터는 복수의 클리핑 파라미터 값들로부터 클리핑 파라미터 값을 식별하기 위한 인덱스를 사용하여 결정되는, 방법.

청구항 4

제1항에 있어서,

상기 이미지를 수신하는 단계; 및

상기 수신된 이미지를 인코딩하고 비트스트림을 생성하는 단계

를 추가로 포함하는, 방법.

청구항 5

적응 루프 필터링을 사용하여 이미지의 하나 이상의 부분을 디코딩하는 방법으로서,

비트스트림으로부터 이미지의 데이터를 획득하는 단계;

이미지의 이미지 부분의 현재 샘플을 필터링하는 단계

를 포함하고,

상기 필터링하는 단계는, 상기 현재 샘플의 이웃하는 샘플 값과 상기 현재 샘플 값 사이의 차이 값을 클리핑하기 위한 클리핑 함수로서, 각각이 클리핑 파라미터들을 갖는 클리핑 함수를 포함하는 비선형 방정식을 사용하여 필터링하고,

상기 클리핑 파라미터는 상기 클리핑의 값 범위를 나타내며, 상기 클리핑 함수 각각의 상기 클리핑 파라미터는 상기 이웃하는 샘플의 위치에 의존하는, 방법.

청구항 6

제5항에 있어서,

상기 클리핑 함수들은 max(-b, min(b,d)), min(b, max(-b,d)), max(c-b, min(c+b,n)), 또는 min(c+b, max(c-b,n)) 중 하나를 반환하고;

c는 상기 현재 샘플 값이며, n은 상기 현재 샘플의 이웃하는 샘플 값이고, d=n-c이며, b는 상기 클리핑 파라미터인, 방법.

청구항 7

제5항에 있어서, 클리핑 파라미터는 복수의 클리핑 파라미터 값들로부터 클리핑 파라미터 값을 식별하기 위한 인덱스를 사용하여 결정되는, 방법.

청구항 8

제5항에 있어서,

비트스트림을 수신하는 단계; 및

상기 이미지를 획득하기 위해 상기 수신된 비트스트림으로부터 정보를 디코딩하는 단계

를 추가로 포함하는, 방법.

청구항 9

제8항에 있어서, 상기 비트스트림으로부터 상기 정보를 획득하는 단계

를 추가로 포함하는, 방법.

청구항 10

이미지를 인코딩하기 위한 디바이스로서, 상기 디바이스는 제1항 내지 제4항 중 어느 한 항의 방법을 수행하도록 구성되는, 디바이스.

청구항 11

이미지를 디코딩하기 위한 디바이스로서, 상기 디바이스는 제5항 내지 제9항 중 어느 한 항의 방법을 수행하도록 구성되는, 디바이스.

청구항 12

컴퓨터 또는 프로세서에서 실행될 때, 상기 컴퓨터 또는 프로세서로 하여금 제1항 내지 제4항 중 어느 한 항의 방법을 수행하게 하는 컴퓨터 프로그램을 저장하는 컴퓨터 판독 가능 저장 매체.

청구항 13

컴퓨터 또는 프로세서에서 실행될 때, 상기 컴퓨터 또는 프로세서로 하여금 제5항 내지 제9항 중 어느 한 항의 방법을 수행하게 하는 컴퓨터 프로그램을 저장하는 컴퓨터 판독 가능 저장 매체.

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

발명의 설명

기 술 분 야

[0001] 본 발명은 비디오 성분의 블록들의 인코딩 또는 디코딩에 관한 것이다. 본 발명의 실시예들은 그러한 성분의 샘플들을 필터링하기 위한 필터를 제어할 때 특별하지만 배타적이지는 않은 용도를 찾는다. 특별히 그러나 배타적이지 않게, 적응 루프 필터(adaptive loop filter)를 제어한다.

배 경 기 술

[0002] 비디오 코딩은 이미지 코딩을 포함한다(이미지는 비디오의 단일 프레임과 동등하다). 비디오 코딩에서, 변환 계수들의 양자화 또는 모션 보상(보간 필터들을 사용하여 종종 수행됨)과 같은 일부 코딩 도구들은 종종 왜곡 편향/효과(시스템적인 것으로 보이거나 적어도 주어진 콘텍스트에서 랜덤하지 않은 왜곡)를 도입한다. 해당 편향들/아티팩트들을 보상하고 코딩 효율을 개선시키기(또는 적어도 양호한 레벨의 코딩 효율을 유지하기) 위해, 포스트 필터(post-filter) 또는 인루프 필터(in-loop filter)라고 불리는 일부 특정 코딩 도구들이 사용된다. 디블로킹 필터(Deblocking filter, DBF), 샘플 적응 오프셋(sample adaptive offset, SAO) 필터 또는 적응 루프 필터(adaptive loop filter, ALF)는 그러한 코딩 도구들의 일부 예이다. 인루프 필터들이 현재 프레임에 제공하는 이미지 품질 향상이 현재 프레임에 기초하여 코딩되는 다음 프레임들에 대한 향상된 코딩 효율을 가능하게 하도록, 인루프 필터들이 코딩 루프 내에서 적용된다. 예를 들어, DCT 계수들의 양자화는 비디오 압축에 효율적이지만 종종 압축된 샘플 블록들의 경계들에 블로킹 아티팩트들(편향들)을 도입한다. 디블로킹 필터는 그러한 아티팩트들로 인해 발생하는 바람직하지 않은 효과들을 감소시킨다. (디코딩된 프레임이 다른 프레임에 대한 모션 보상을 위한 참조 프레임으로서 역할하기 전에) 코딩 루프 내에서 디코딩된 프레임을 (DBF를 사용하여) 디블로킹하는 것은 (예를 들면, 예를 들어, 프레임을 디스플레이하기 직전에) 코딩 루프 밖에서 프레임을 디블로킹하는 것과 비교하여 모션 보상의 코딩 효율을 크게 향상시킨다.

[0003] 본 발명은 특히 디코딩된 프레임들/이미지들에서의 바람직하지 않은 압축 아티팩트들을 감소시키기 위해 인루프 필터로서 또한 적용되는 적응 루프 필터(ALF)에 관한 것이다. ALF는 VCEG/MPEG(Video Coding Experts Group/Moving Picture Experts Group) 표준화 그룹들에 의해 연구되었으며, VVC(Versatile Video Coding) 표준에서, 예를 들어, VVC Test Model 소프트웨어의 세 번째 버전(VTM-3.0 또는 VVC Draft 버전 3)에서 사용하기 위해 고려되고 있다.

발명의 내용

[0004] 본 발명의 실시예들의 목적은 비디오 성분의 블록들의 전술한 인코딩 또는 디코딩의 하나 이상의 문제점 또는

단점을 해결하는 것이다.

[0005] ALF가 효율적인 코딩 도구이지만, 그의 선형 필터링은 이미지 품질을 개선시키기 위한 차선의 해결책이다. 본 발명은 ALF를 사용한 비선형 필터링이 그의 효율 및/또는 성능을 개선시키는 것을 가능하게 한다.

[0006] 본 발명의 양태들에 따르면, 첨부된 청구항들에 기재된 바와 같은 장치/디바이스, 방법, 프로그램, 컴퓨터 판독 가능 저장 매체 및 신호가 제공된다. 본 발명의 다른 특징들은 종속 청구항들 및 설명으로부터 명백해질 것이다.

[0007] 본 발명의 제1 양태에 따르면, 이미지의 하나 이상의 이미지 부분에 대한 적응 루프 필터를 제어하는 방법이 제공되며, 이 방법은 이미지 부분의 제1 샘플에 대한 필터링을 제1 샘플 값의 하나 이상의 이웃하는 샘플 값(들)에 기초하여 제어하는 단계를 포함하며, 여기서 제어하는 단계는 하나 이상의 이웃하는 샘플 값(들)을 변수들로서 갖는 비선형 함수를 사용한다. 적합하게는, 비선형 함수의 변수들은 2개 이상의 이웃하는 샘플 값을 포함한다. 적합하게는, 비선형 함수의 변수들은 제1 샘플 값 및 하나 이상의 이웃하는 샘플 값(들)의 위치(들)에 의존하는 제1 변수를 추가로 포함한다. 적합하게는, 제1 변수는 2개 이상의 이웃하는 샘플 값의 위치들에 의존한다. 적합하게는, 비선형 함수의 출력은 적응 루프 필터에 대한 입력(또는 입력 파라미터)으로서 사용된다.

[0008] 본 발명의 제2 양태에 따르면, 이미지의 하나 이상의 이미지 부분에 대한 필터를 제어하는 방법이 제공되며, 이 방법은 제1 샘플 값의 하나 이상의 이웃하는 샘플 값들에 기초하여 이미지 부분의 제1 샘플에 대한 필터링을 제어하는 단계를 포함하며, 여기서 제어하는 단계는 제1 샘플 값, 하나 이상의 이웃하는 샘플 값 및 제1 변수를 포함하는 복수의 변수들을 갖는 비선형 함수를 사용하고, 상기 제1 변수는 하나 이상의 이웃하는 샘플 값의 위치(들)에 의존한다. 적합하게는, 복수의 변수들은 2개 이상의 이웃하는 샘플 값을 포함한다. 적합하게는, 제1 변수는 2개 이상의 이웃하는 샘플 값의 위치들에 의존한다. 적합하게는, 필터는 적응 루프 필터이다.

[0009] 본 발명의 제3 양태에 따르면, 이미지의 하나 이상의 이미지 부분에 대한 필터를 제어하는 방법이 제공되며, 이 방법은 이미지 부분의 제1 샘플에 대한 필터링을 제1 샘플 값의 하나 이상의 이웃하는 샘플 값(들)에 기초하여 제어하는 단계를 포함하며, 여기서 제어하는 단계는 제1 샘플 값, 하나 이상의 이웃하는 샘플 값 및 클리핑 파라미터에 기초한 하나 이상의 제어 파라미터(들)를 갖는 하나 이상의 클리핑 함수(들)를 사용한다. 적합하게는, 하나 이상의 제어 파라미터(들)는 제1 샘플 값, 2개 이상의 이웃하는 샘플 값 및 클리핑 파라미터에 기초한다. 적합하게는, 필터는 적응 루프 필터이다. 적합하게는, 하나 이상의 클리핑 함수(들) 각각은 max(-b, min(b,d)), min(b, max(-b,d)), max(c-b, min(c+b,n)), 또는 min(c+b, max(c-b,n)) 중 하나이고; c는 제1 샘플 값이며, n은 이웃하는 샘플 값이고, d=n-c이며, b는 클리핑 파라미터이다.

[0010] 본 발명의 제4 양태에 따르면, 이미지의 하나 이상의 이미지 부분에 대한 적응 루프 필터를 제어하는 방법이 제공되며, 이 방법은 이미지 부분의 제1 샘플에 대한 필터링을 제1 샘플 값의 하나 이상의 이웃하는 샘플 값(들)에 기초하여 제어하는 단계를 포함하며, 여기서 제어하는 단계는 적응 루프 필터에 대한 입력 파라미터로서 제1 샘플 값과 하나 이상의 이웃하는 샘플 값(들)의 비선형 조합을 사용한다. 적합하게는, 적응 루프 필터는 VTM3.0에서 명시된 바와 같다.

[0011] 본 발명의 제5 양태에 따르면, 이미지의 하나 이상의 이미지 부분에 대한 적응 루프 필터를 제어하는 방법이 제공되며, 이 방법은 제1 샘플 값의 2개 이상의 이웃하는 샘플 값에 기초하여 이미지 부분의 제1 샘플에 대한 필터링을 제어하는 단계를 포함하며, 여기서 제어하는 단계는 적응 루프 필터에 대한 입력 파라미터로서 2개 이상의 이웃하는 샘플 값의 비선형 조합을 사용한다. 적합하게는, 적응 루프 필터는 VTM3.0에서 명시된 바와 같다.

[0012] 본 발명의 제4 양태 및 제5 양태에 대하여, 그의 실시예에 따라 이하의 특징들이 제공될 수 있다. 적합하게는, 비선형 조합은 비선형 함수의 일부이다. 적합하게는, 적응 루프 필터에 대한 입력 파라미터들은 하나 이상의 이웃하는 샘플 값(들)의 위치(들)에 의존하는 제1 변수를 추가로 포함한다. 적합하게는, 제1 변수는 2개 이상의 이웃하는 샘플 값의 위치들에 의존한다. 적합하게는, 비선형 조합은 제1 샘플 값, 하나의(또는 2개 이상의) 이웃하는 샘플 값(들) 및 제1 변수로 되어 있다.

[0013] 본 발명의 제6 양태에 따르면, 이미지의 하나 이상의 부분 - 이미지 부분은 그와 연관된 크로마 샘플들 및 루마 샘플들을 가짐 - 을 프로세싱하는 방법이 제공되며, 여기서 이 방법은, 비트스트림으로부터 획득되는 정보 또는 이미지 부분의 제1 샘플 값 및 그의 하나 이상의 이웃하는 샘플 값(들)에 기초하여, 제1 양태, 제2 양태, 제3 양태, 제4 양태 또는 제5 양태에 따른 방법을 사용하여 제어되는 필터를 사용할지 여부; 상기 필터의 사용을 인에이블 또는 디스에이블하는 것; 또는 제1 샘플 값에 대해 필터링할 때 상기 필터에 대해 사용하기 위한 필터링 파라미터 중 적어도 하나를 결정하는 단계를 포함한다. 적합하게는, 비트스트림으로부터 획득 가능한 정보는

플래그 또는 인덱스를 포함한다. 적합하게는, 비트스트림으로부터 획득 가능한 정보는 상기 필터를 식별하기 위한 정보; 사용 여부를 나타내기 위한 플래그; 인에이블 또는 디스에이블을 나타내기 위한 플래그; 상기 필터에 대해 사용하기 위한 제1 변수에 관한 정보; 또는 값 범위를 지정하기 위한 클리핑 파라미터에 관한 정보 중 하나 이상을 포함한다. 적합하게는, 제1 변수에 관한 정보는 값을 지정하거나 제1 함수를 제1 변수로서 식별하기 위한 것이다. 적합하게는, 제1 변수는 하나 이상의 이웃하는 샘플 값의 위치(들)에 의존한다(또는 그에 기초하여 변할 수 있다). 적합하게는, 제1 변수는 2개 이상의 이웃하는 샘플 값의 위치들에 의존한다.

[0014] 본 발명의 제7 양태에 따르면, 하나 이상의 이미지(들)를 인코딩하는 방법이 제공되며, 이 방법은, 이미지의 하나 이상의 부분에 대해, 제1 양태, 제2 양태, 제3 양태, 제4 양태, 또는 제5 양태에 따른 필터를 제어하는 단계, 또는 제6 양태에 따른 프로세싱하는 단계를 포함한다. 적합하게는, 이 방법은: 이미지를 수신하는 단계; 수신된 이미지를 인코딩하고 비트스트림을 생성하는 단계; 및 인코딩된 이미지를 프로세싱하는 단계를 추가로 포함하고, 여기서 프로세싱하는 단계는 제1 양태, 제2 양태, 제3 양태, 제4 양태, 또는 제5 양태에 따른 제어하는 단계, 또는 제6 양태에 따른 프로세싱하는 단계를 포함한다. 적합하게는, 제6 양태에 종속적일 때, 이 방법은 비트스트림에서 상기 정보를 제공하는 단계를 추가로 포함한다. 적합하게는, 이 방법은: 복수의 이용 가능한 함수들로부터 비선형 함수 또는 하나 이상의 클리핑 함수(들)를 선택하는 단계; 인코딩된 이미지를 프로세싱할 때 선택된 함수를 사용하는 단계; 및 선택된 함수를 식별하기 위한 정보를, 비트스트림에서, 제공하는 단계를 추가로 포함한다.

[0015] 본 발명의 제8 양태에 따르면, 하나 이상의 이미지(들)를 디코딩하는 방법이 제공되며, 이 방법은, 이미지의 하나 이상의 부분에 대해, 제1 양태, 제2 양태, 제3 양태, 제4 양태, 또는 제5 양태에 따른 필터를 제어하는 단계, 또는 제6 양태에 따른 프로세싱하는 단계를 포함한다. 적합하게는, 이 방법은: 비트스트림을 수신하는 단계; 이미지를 획득하기 위해 수신된 비트스트림으로부터 정보를 디코딩하는 단계; 및 획득된 이미지를 프로세싱하는 단계를 추가로 포함하고, 여기서 프로세싱하는 단계는 제1 양태, 제2 양태, 제3 양태, 제4 양태, 또는 제5 양태에 따른 제어하는 단계, 또는 제6 양태에 따른 프로세싱하는 단계를 포함한다. 적합하게는, 제6 양태에 종속적일 때, 이 방법은 비트스트림으로부터 상기 정보를 획득하는 단계를 추가로 포함한다. 적합하게는, 이 방법은: 복수의 이용 가능한 함수들로부터 비선형 함수 또는 하나 이상의 클리핑 함수(들)를 식별하기 위한 정보를, 비트스트림으로부터, 획득하는 단계; 및 획득된 이미지를 프로세싱할 때 식별된 함수를 사용하는 단계를 추가로 포함한다.

[0016] 본 발명의 제9 양태에 따르면, 이미지의 하나 이상의 부분에 대한 필터를 제어하기 위한 디바이스가 제공되며, 이 디바이스는 제1 양태, 제2 양태, 제3 양태, 제4 양태, 제5 양태, 또는 제6 양태에 따른 방법을 수행하도록 구성된 제어기를 포함한다.

[0017] 본 발명의 제10 양태에 따르면, 이미지를 인코딩하기 위한 디바이스가 제공되며, 이 디바이스는 제9 양태에 따른 제어 디바이스를 포함한다. 적합하게는, 이 디바이스는 제7 양태에 따른 방법을 수행하도록 구성된다.

[0018] 본 발명의 제11 양태에 따르면, 이미지를 디코딩하기 위한 디바이스가 제공되며, 이 디바이스는 제9 양태에 따른 제어 디바이스를 포함한다. 적합하게는, 이 디바이스는 제8 양태에 따른 방법을 수행하도록 구성된다.

[0019] 본 발명의 제12 양태에 따르면, 이미지의 하나 이상의 이미지 부분에 대한 적응 루프 필터를 제어하는 방법이 제공되며, 이 방법은 제1 샘플 값의 복수의 이웃하는 샘플 값들에 기초하여 이미지 부분의 제1 샘플의 필터링을 제어하는 단계를 포함하며, 여기서 제어하는 단계는 비선형 함수를 사용하는 단계 - 비선형 함수는 이웃하는 샘플 값(들) 중 하나 이상에 기초한 하나 이상의 변수를 가짐 - 를 포함한다. 이웃하는 샘플(값)이 인접한 샘플(값)로 제한되지 않고 제1 샘플(값) 주변 또는 근처의 샘플(값)을 또한 포함한다는 것이 이해된다. 적합하게는, 제어하는 단계는, 적응 루프 필터에 대한 입력 파라미터(들)로서, 결정하는 단계에 사용되지 않는 하나 이상의 다른 이웃하는 샘플의 하나 이상의 이웃하는 샘플 값(들)을 사용하거나 또는 비선형 함수에서의 변수로서 사용하는 단계를 포함한다. 적합하게는, 값들이 결정하는 단계에 사용되거나 또는 하나 이상의 비선형 함수(또는 비선형 함수)에서의 변수들로서 사용되는 이웃하는 샘플들은 교차점에 제1 샘플을 갖는 십자형(cross); 또는 평행사변형의 형상으로 배열된다. 적합하게는, 값들이 결정하는 단계에 사용되거나 또는 하나 이상의 비선형 함수(또는 비선형 함수)에서의 변수들로서 사용되는 이웃하는 샘플들은, 8개의 이웃하는 샘플의 값들이 결정하는 단계에 사용되거나 또는 비선형 함수에서의 변수들로서 사용될 때, 교차점에 제1 샘플을 갖는 5개 샘플 높이와 5개 샘플 폭의 십자형 또는 3개 샘플 길이의 각각의 변을 갖는 속이 빈 평행사변형; 또는 12개의 이웃하는 샘플의 값들이 결정하는 단계에 사용되거나 또는 하나 이상의 비선형 함수(또는 비선형 함수)에서의 변수들로서 사용될 때, 3개 샘플 길이의 각각의 변을 갖는 평행사변형의 형상으로 배열된다. 적합하게는,

교차점에(및/또는 중심에) 제1 샘플을 갖는 십자형은: 수직선과 수평선의 십자형"+"; 또는 사선들의 십자형 "X"(대각 십자형) 중 하나이다. 적합하게는, (속이 빈) 평행사변형은: 정사각형; 직사각형; 또는 다이아몬드 형상 중 하나이다. 적합하게는, (속이 빈) 평행사변형은 중심에 위치된 제1 샘플 주위에 있거나 이를 둘러싸고 있다. 대안적으로, 값들이 결정하는 단계에 사용되거나 또는 하나 이상의 비선형 함수(또는 비선형 함수)에서의 변수들로서 사용되는 이웃하는 샘플들은, 중심에 제1 샘플이 있는 상태로, 수직선 "|"; 수평선 "-"; 좌측 상단으로부터 우측 하단으로의 대각선/사선 "\"; 또는 우측 상단으로부터 좌측 하단으로의 대각선/사선 "/"의 형상으로 배열된다. 대안적으로, 값들이 결정하는 단계에 사용되거나 또는 하나 이상의 비선형 함수(또는 비선형 함수)에서의 변수들로서 사용되는 이웃하는 샘플들은 (속이 빈) 다각형의 형상으로 배열된다. 적합하게는, (속이 빈) 다각형은 중심에 위치된 제1 샘플 주위에 있거나 이를 둘러싸고 있다. 대안적으로, 값들이 결정하는 단계에 사용되거나 또는 하나 이상의 비선형 함수(또는 비선형 함수)에서의 변수들로서 사용되는 이웃하는 샘플들은 전술한 형상들의 임의의 조합의 형상으로 배열된다. 적합하게는, 필터 변수는 2개 이상의 이웃하는 샘플 간에 공유된다. 적합하게는, 이웃하는 샘플들의 배열 형상은 중심에 대해 대칭성을 갖는다. 적합하게는, 십자형 형상, 평행사변형 형상 또는 다각형 형상은 중심에 대해 대칭성을 갖는다. 대안적으로, 이웃하는 샘플의 배열 형상은 중심에 대해 대칭성을 갖지 않는다. 적합하게는, 십자형 형상 또는 평행사변형 형상은 중심에 대해 대칭성을 갖지 않는다. 적합하게는, 제1 샘플 및 복수의 이웃하는 샘플 값들의 이웃하는 샘플들은 교차점에 제1 샘플을 갖는 십자형; 또는 평행사변형 또는 속이 빈 평행사변형의 형상으로 배열된다. 적합하게는, 제1 샘플 및 복수의 이웃하는 샘플 값들의 이웃하는 샘플들은 복수의 이웃하는 샘플 값들이 24개의 이웃하는 샘플의 샘플 값들로 이루어져 있을 때 4개 샘플 길이의 각각의 변; 또는 복수의 이웃하는 샘플 값들이 12개의 이웃하는 샘플의 샘플 값들로 이루어져 있을 때 3개 샘플 길이의 각각의 변을 갖는 평행사변형의 형상으로 배열된다. 적합하게는, 교차점에(및/또는 중심에) 제1 샘플을 갖는 십자형은: 수직선과 수평선의 십자형"+"; 또는 사선들의 십자형 "X"(대각 십자형) 중 하나이다. 적합하게는, (속이 빈) 평행사변형은: 정사각형; 직사각형; 또는 다이아몬드 형상 중 하나이다. 적합하게는, (속이 빈) 평행사변형은 중심에 위치된 제1 샘플 주위에 있거나 이를 둘러싸고 있다. 대안적으로, 제1 샘플 및 복수의 이웃하는 샘플 값들의 이웃하는 샘플들은, 중심에 제1 샘플이 있는 상태로, 수직선 "|"; 수평선 "-"; 좌측 상단으로부터 우측 하단으로의 대각선/사선 "\"; 또는 우측 상단으로부터 좌측 하단으로의 대각선/사선 "/"의 형상으로 배열된다. 대안적으로, 제1 샘플 및 복수의 이웃하는 샘플 값들의 이웃하는 샘플들은 (속이 빈) 다각형의 형상으로 배열된다. 적합하게는, (속이 빈) 다각형은 중심에 위치된 제1 샘플 주위에 있거나 이를 둘러싸고 있다. 대안적으로, 제1 샘플 및 복수의 이웃하는 샘플 값들의 이웃하는 샘플들은 전술한 형상들의 임의의 조합의 형상으로 배열된다. 적합하게는, 필터 변수는 2개 이상의 이웃하는 샘플 간에 공유된다. 적합하게는, 이웃하는 샘플들의 배열 형상은 중심에 대해 대칭성을 갖는다. 적합하게는, 십자형 형상, 평행사변형 형상 또는 다각형 형상은 중심에 대해 대칭성을 갖는다. 대안적으로, 이웃하는 샘플들의 배열 형상은 중심에 대해 대칭성을 갖지 않는다. 적합하게는, 십자형 형상, 평행사변형 형상 또는 다각형 형상은 중심에 대해 대칭성을 갖지 않는다. 적합하게는, 제1 샘플 및 상기 이웃하는 샘플들은 루마 성분 샘플들이다. 대안적으로, 제1 샘플 및 상기 이웃하는 샘플은 크로마 성분 샘플들이다. 적합하게는, 비선형 함수(또는 하나 이상의 비선형 함수)의 변수들은 제1 샘플 값을 추가로 포함하고, 비선형 함수(또는 하나 이상의 비선형 함수)는 제1 샘플 값과 하나 이상의 이웃하는 샘플 값(들) 각각 사이의 차이에 적용된다. 적합하게는, 비선형 함수(또는 하나 이상의 비선형 함수)의 변수들은 제1 샘플 값 및 하나 이상의 이웃하는 샘플의 위치(들)에 의존하는 하나 이상의 필터 변수(들)를 추가로 포함하고, 각각의 필터 변수는 2개 이상의 이웃하는 샘플에 대해 동일하며; 비선형 함수(또는 하나 이상의 비선형 함수)는 제1 샘플 값과 동일한 필터 변수를 갖는 2개 이상의 이웃하는 샘플의 2개 이상의 이웃하는 샘플 값 각각 사이의 2개 이상의 차이의 합에 적용된다. 적합하게는, 비선형 함수(또는 하나 이상의 비선형 함수)의 출력은 적응 루프 필터에 대한 입력(또는 입력 파라미터)으로서 사용된다. 적합하게는, 비선형 함수는 하나 이상의 클리핑 함수(들)를 포함하고, 하나 이상의 클리핑 함수(들) 각각은 max(-b, min(b,d)), min(b, max(-b,d)), max(c-b, min(c+b,n)), min(c+b, max(c-b,n)), max(-b, min(b,d1+d2)), min(b, max(-b,d1+d2)), max(2*c-b, min(2*c+b,n1+n2)), 또는 min(2*c+b, max(2*c-b,n1+n2)) 중 하나이며, 여기서 c는 제1 샘플 값이고, n 또는 n1 또는 n2는 이웃하는 샘플 값이며, d=n-c이고, d1=n1-c이며, d2=n2-c이고, b는 클리핑 파라미터이다.

[0020] 본 발명의 제13 양태에 따르면, 이미지의 하나 이상의 부분 - 이미지 부분은 그와 연관된 크로마 샘플들 및 루마 샘플들을 가짐 - 을 프로세싱하는 방법이 제공되며, 여기서 이 방법은, 비트스트림으로부터 획득되는 정보 또는 이미지 부분의 제1 샘플 값 및 그의 하나 이상의 이웃하는 샘플 값(들)에 기초하여, 제12 양태의 방법을 사용하여 제어되는 필터를 사용할지 여부; 상기 필터의 사용을 인에이블 또는 디스에이블하는 것; 또는 제1 샘플 값을 필터링할 때 상기 필터에 대해 사용하기 위한 필터링 파라미터 또는 필터 변수 중 적어도 하나를 결정

하는 단계를 포함한다. 적합하게는, 비트스트림으로부터 획득되는 정보는 루마 또는 크로마 성분 중 하나에 대해 제공되는 플래그를 포함하고, 플래그는 해당 성분에 대해 제12 양태의 방법을 사용하여 제어되는 필터를 사용할지 여부; 또는 상기 필터의 사용을 인에이블 또는 디스에이블하는 것 중 적어도 하나를 나타낸다. 적합하게는, 플래그는 적응 파라미터 세트에서 제공된다. 적합하게는, 비트스트림으로부터 획득되는 정보는 하나 이상의 이미지 부분에 대해 제공되는 플래그를 포함하고, 플래그는 하나 이상의 이미지 부분에 대해 제12 양태의 방법을 사용하여 제어되는 필터를 사용할지 여부; 또는 상기 필터의 사용을 인에이블 또는 디스에이블하는 것 중 적어도 하나를 나타낸다. 적합하게는, 플래그는 적응 파라미터 세트에서 제공된다.

[0021] 본 발명의 제14 양태에 따르면, 하나 이상의 이미지(들)를 인코딩하는 방법이 제공되며, 이 방법은, 이미지의 하나 이상의 부분에 대해, 제12 양태의 방법에 따라 필터를 제어하는 단계, 또는 제13 양태의 방법에 따라 프로세싱하는 단계를 포함한다. 적합하게는, 이 방법은: 이미지를 수신하는 단계; 수신된 이미지를 인코딩하고 비트스트림을 생성하는 단계; 및 인코딩된 이미지를 프로세싱하는 단계를 추가로 포함하고, 여기서 프로세싱하는 단계는 제12 양태의 방법에 따라 제어하는 단계, 또는 제13 양태의 방법에 따라 프로세싱하는 단계를 포함한다. 적합하게는, 이 방법은 (제13 양태에 종속적일 때) 비트스트림에서 상기 정보를 제공하는 단계를 추가로 포함한다. 적합하게는, 비선형 함수는 하나 이상의 클리핑 함수(들)를 포함하고; 비선형 함수(또는 하나 이상의 비선형 함수)의 변수들은 값들이 비선형 함수(또는 하나 이상의 비선형 함수)에서 변수들로서 사용되는 하나 이상의 이웃하는 샘플의 위치(들)에 의존하는 하나 이상의 필터 변수(들)를 추가로 포함하며; 상기 정보를 제공하는 단계는 비트스트림에서 하나 이상의 클리핑 파라미터(들)를 제공하는 단계를 포함하고; 위치에 대한 필터 변수가 0일 때, 해당 위치에 있는 이웃하는 샘플의 이웃하는 샘플 값에 적용될 클리핑 함수에 대해 사용하기 위한 클리핑 파라미터는 비트스트림에서 제공되지 않는다. 적합하게는, 이 방법은: 복수의 이용 가능한 함수들로부터 비선형 함수 또는 하나 이상의 클리핑 함수(들)를 선택하는 단계; 인코딩된 이미지를 프로세싱할 때 선택된 함수를 사용하는 단계; 및 선택된 기능을 식별하기 위한 정보를, 비트스트림에서, 제공하는 단계를 추가로 포함한다.

[0022] 본 발명의 제15 양태에 따르면, 하나 이상의 이미지(들)를 디코딩하는 방법이 제공되며, 이 방법은, 이미지의 하나 이상의 부분에 대해, 제12 양태의 방법에 따라 필터를 제어하는 단계, 또는 제13 양태의 방법에 따라 프로세싱하는 단계를 포함한다. 적합하게는, 이 방법은: 비트스트림을 수신하는 단계; 이미지를 획득하기 위해 수신된 비트스트림으로부터 정보를 디코딩하는 단계; 및 획득된 이미지를 프로세싱하는 단계를 추가로 포함하고, 여기서 프로세싱하는 단계는 제12 양태의 방법에 따라 제어하는 단계, 또는 제13 양태의 방법에 따라 프로세싱하는 단계를 포함한다. 적합하게는, 이 방법은 (제13 양태에 종속적일 때) 비트스트림으로부터 상기 정보를 획득하는 단계를 추가로 포함한다. 적합하게는, 비선형 함수는 하나 이상의 클리핑 함수(들)를 포함하고; 비선형 함수(또는 하나 이상의 비선형 함수)의 변수들은 값들이 비선형 함수(또는 하나 이상의 비선형 함수)에서 변수들로서 사용되는 하나 이상의 이웃하는 샘플의 위치(들)에 의존하는 하나 이상의 필터 변수(들)를 추가로 포함하며; 위치에 대한 필터 변수가 0일 때, 해당 위치에 있는 이웃하는 샘플의 이웃하는 샘플 값에 클리핑 함수가 적용되지 않는다. 적합하게는, 이 방법은: 복수의 이용 가능한 함수들로부터 비선형 함수 또는 하나 이상의 클리핑 함수(들)를 식별하기 위한 정보를, 비트스트림으로부터, 획득하는 단계; 및 획득된 이미지를 프로세싱할 때 식별된 함수를 사용하는 단계를 추가로 포함한다.

[0023] 본 발명의 제16 양태에 따르면, 디바이스가 제공되고, 이 디바이스는: 제12 양태 또는 제13 양태의 방법을 수행하도록 구성된 제어기; 제14 양태의 방법을 수행하도록 구성된 인코더; 또는 제15 양태의 방법을 수행하도록 구성된 디코더를 포함한다.

[0024] 본 발명의 제17 양태에 따르면, 이미지의 하나 이상의 부분에 대한 필터를 제어하기 위한 디바이스가 제공되며, 이 디바이스는 제12 양태 또는 제13 양태의 방법을 수행하도록 구성된 제어기를 포함한다.

[0025] 본 발명의 제18 양태에 따르면, 이미지를 인코딩하기 위한 디바이스가 제공되며, 이 디바이스는 제17 양태의 제어 디바이스를 포함한다. 적합하게는, 이 디바이스는 제14 양태의 방법을 수행하도록 구성된다.

[0026] 본 발명의 제19 양태에 따르면, 이미지를 디코딩하기 위한 디바이스가 제공되며, 이 디바이스는 제17 양태의 제어 디바이스를 포함한다. 적합하게는, 이 디바이스는 제15 양태의 방법을 수행하도록 구성된다.

[0027] 본 발명의 제20 양태에 따르면, 컴퓨터 또는 프로세서 상에서 실행될 때, 컴퓨터 또는 프로세서로 하여금 제1 양태, 제2 양태, 제3 양태, 제4 양태, 제5 양태, 제6 양태, 제7 양태, 제8 양태, 제12 양태, 제13 양태, 제14 양태, 또는 제15 양태에 따른 방법을 수행하게 하는 프로그램이 제공된다.

[0028] 본 발명의 제21 양태에 따르면, 제20 양태에 따른 컴퓨터 프로그램을 저장하는 컴퓨터 판독 가능 저장 매체가 제공된다.

[0029] 본 발명의 제22 양태에 따르면, 제7 양태 또는 제14 양태에 따른 방법을 사용하여 인코딩되고 비트스트림에 의해 표현되는 이미지에 대한 정보 데이터세트를 운반하는 신호 - 이미지는 재구성 가능한 샘플들의 세트를 포함하고, 각각의 재구성 가능한 샘플은 샘플 값을 가짐 - 가 제공되며, 여기서 정보 데이터세트는 제1 재구성 가능한 샘플의 이웃하는 샘플들의 샘플 값들에 기초하여 제1 재구성 가능한 샘플에 대한 필터링을 제어하기 위한 제어 데이터를 포함한다.

[0030] 본 발명의 전술한 양태들에 대하여, 그의 실시예에 따라 이하의 특징들이 제공될 수 있다. 적합하게는, 비선형 함수는 클리핑 함수, 톱니 커널 함수, 삼각 커널 함수 또는 가우시안 커널 함수 중 하나 이상을 포함한다. 적합하게는, 비선형 함수는 하나 이상의 클리핑 함수(들)를 포함하고, 하나 이상의 클리핑 함수(들) 각각은 max(-b, min(b,d)), min(b, max(-b,d)), max(c-b, min(c+b,n)), min(c+b, max(c-b,n)), max(-b, min(b,d1+d2)), min(b, max(-b,d1+d2)), max(2*c-b, min(2*c+b,n1+n2)), 또는 min(2*c+b, max(2*c-b,n1+n2)) 중 하나이며, 여기서 c는 제1 샘플 값이고, n 또는 n1 또는 n2는 이웃하는 샘플 값이며, d=n-c이고, d1=n1-c이며, d2=n2-c이고, b는 클리핑 파라미터이다. 적합하게는, (복수의) 변수들은 (수학적) 비선형 함수의 독립 변수들이고, 비선형 함수의 출력은 그의 종속 변수이다. 적합하게는, 이웃하는 샘플의 위치는 제1 샘플의 위치로부터의 상대 거리 또는 상대 변위로서 정의된다. 적합하게는, 이웃하는 샘플의 위치는 (래스터) 스캔 순서에서의 그의 위치로서(적합하게는, (래스터) 스캔 순서에서의 제1 샘플의 위치와 관련하여) 정의된다. 적합하게는, 적응 루프 필터는 비선형 함수를 사용하여 필터링을 수행한다. 적합하게는, 비선형 함수의 출력은 적응 루프 필터에 대한 입력(또는 입력 파라미터)으로서 사용된다. 적합하게는, 하나 이상의 그러한 입력(파라미터(들))에 적응 루프 필터를 적용하는 것은 하나 이상의 입력(파라미터(들))을 그의 변수들로서 갖는 선형 함수를 사용하는 것을 포함한다. 적합하게는, 적응 루프 필터는 VTM3.0에서 명시된 바와 같다.

[0031] 본 발명의 또 다른 양태들은, 컴퓨터 또는 프로세서에 의해 실행될 때, 컴퓨터 또는 프로세서로 하여금 전술한 양태들의 방법들 중 임의의 것을 수행하게 하는 프로그램들에 관한 것이다. 프로그램은 그 자체로 제공될 수 있거나 또는 캐리어 매체 상에, 캐리어 매체에 의해 또는 캐리어 매체 내에 담겨 있을 수 있다. 캐리어 매체는 비일시적일 수 있으며, 예를 들어, 저장 매체, 상세하게는 컴퓨터 판독 가능 저장 매체일 수 있다. 캐리어 매체는 또한 일시적일 수 있으며, 예를 들어, 신호 또는 다른 전송 매체일 수 있다. 신호는, 인터넷을 포함한, 임의의 적합한 네트워크를 통해 전송될 수 있다.

[0032] 본 발명의 다른 추가의 양태들은 전술한 디바이스 양태들 중 임의의 것에 따른 디바이스를 포함하는 카메라에 관한 것이다. 본 발명의 또 다른 양태에 따르면, 전술한 디바이스 양태들 중 임의의 것에 따른 디바이스 및/또는 상기 카메라 양태를 구체화하는 카메라를 포함하는 모바일 디바이스가 제공된다.

[0033] 본 발명의 일 양태에서의 임의의 특징은, 임의의 적절한 조합으로, 본 발명의 다른 양태들에 적용될 수 있다. 상세하게는, 방법 양태들이 장치 양태들에 적용될 수 있으며, 그 반대의 경우도 마찬가지이다. 게다가, 하드웨어로 구현되는 특징들이 소프트웨어로 구현될 수 있으며, 그 반대의 경우도 마찬가지이다. 본 명세서에서의 소프트웨어 및 하드웨어 특징들에 대한 임의의 언급은 그에 따라 해석되어야 한다. 본 명세서에서 기술된 바와 같은 임의의 장치 특징은 또한 방법 특징으로서 제공될 수 있으며, 그 반대의 경우도 마찬가지이다. 본 명세서에서 사용되는 바와 같이, 기능식(means plus function) 특징들은 대안적으로, 그들의 대응하는 구조, 예컨대, 적합하게 프로그래밍된 프로세서 및 연관된 메모리의 관점에서 표현될 수 있다. 또한, 본 발명의 임의의 양태들에서 기술되고 정의되는 다양한 특징들의 특정 조합들이 독립적으로 구현 및/또는 공급 및/또는 사용될 수 있음을 이해해야 한다.

도면의 간단한 설명

[0034] 본 발명의 실시예들이 이제 이하의 도면들을 참조하여, 단지 예로서, 기술될 것이다.

도 1은 VTM-3.0의 전형적인 디코딩 루프 내의 어디에서 ALF가 이루어지는지를 도시한다;

도 2는 ALF에 대한 VTM-3.0에 존재하는 신택스 요소들의 개관을 갖는 플로차트이다;

도 3a는 본 발명의 실시예에 따른 크로마 성분을 필터링하기 위한 단계들을 예시하는 플로차트이다;

도 3b는 본 발명의 실시예에 따른 크로마 필터에 대한 필터 형상 및 계수 배열을 제공한다;

도 4a는 본 발명의 실시예에 따른 루마 성분을 필터링하기 위한 단계들을 예시하는 플로차트이다;

도 4b는 본 발명의 실시예에 따른 루마 필터에 대한 필터 형상 및 4개의 가능한 계수 배열을 제공한다;

도 5는 본 발명의 실시예에 따른 수정된 신택스 요소들의 개관을 갖는 플로차트이다;

도 6은 VTM-3.0에서의 ALF 내의 선형 필터의 블록 다이어그램이다;

도 7은 본 발명의 실시예에 따른 비선형 필터의 블록 다이어그램이다;

도 8은 본 발명의 실시예에 따른 도 7에서의 필터의 비선형 부분이 디스에이블될 때 획득되는 선형 필터의 블록 다이어그램이다;

도 9는 본 발명의 실시예에 따른 ALF 인코딩 프로세스를 예시하는 플로차트이다;

도 10은 본 발명의 실시예들에 따른 인코딩 방법의 단계들을 예시하는 플로차트이다;

도 11은 본 발명의 실시예들에 따른 디코딩 방법의 단계들을 예시하는 플로차트이다;

도 12는 본 발명의 하나 이상의 실시예가 구현될 수 있는 데이터 통신 시스템을 개략적으로 예시하는 블록 다이어그램이다;

도 13은 본 발명의 하나 이상의 실시예가 구현될 수 있는 프로세싱 디바이스의 컴포넌트들을 예시하는 블록 다이어그램이다;

도 14는 본 발명의 하나 이상의 실시예가 구현될 수 있는 네트워크 카메라 시스템을 예시하는 다이어그램이다;

도 15는 본 발명의 하나 이상의 실시예가 구현될 수 있는 스마트 폰을 예시하는 다이어그램이다;

도 16a는 본 발명의 실시예들에 따른 감소된 수의 클리핑되는 위치들을 갖는 7x7 다이아몬드 필터 형상들을 제공한다;

도 16b는 본 발명의 실시예들에 따른 감소된 수의 클리핑되는 위치들을 갖는 5x5 다이아몬드 필터 형상들을 제공한다.

발명을 실시하기 위한 구체적인 내용

[0035] 아래에서 기술되는 본 발명의 실시예들은 이미지들의 인코딩 및 디코딩을 개선시키는 것에 관한 것이다.

[0036] 본 명세서에서, "시그널링"은 필터를 제어하기 위한 하나 이상의 파라미터에 관한 정보, 예를 들어, 모드/스킴의 사용, 비사용, 인에이블 또는 디스에이블 또는 다른 필터 제어 관련 정보를 비트스트림에 삽입(제공/포함/인코딩)하는 것 또는 비트스트림으로부터 추출/획득(디코딩)하는 것을 지칭할 수 있다.

[0037] 본 명세서에서, 용어 "슬라이스"는 이미지 부분의 예로서 사용된다(그러한 이미지 부분의 다른 예는 타일 또는 타일(들)의 그룹/세트인 타일 그룹일 것이다). 본 발명의 실시예들이 또한, 슬라이스 대신에, 이미지 부분(예를 들면, 타일 또는 타일 그룹), 및 (슬라이스 헤더 대신에) 이미지 부분/타일/타일 그룹에 대한 헤더, (슬라이스 유형 대신에) 이미지 부분/타일/타일 그룹의 유형, 및 (슬라이스 통계 대신에) 이미지 부분/타일/타일 그룹에 대한 통계와 같은 적절하게 수정된 파라미터/값/신택스에 기초하여 구현될 수 있음이 이해된다. 본 발명의 실시예들에서, 슬라이스 헤더 또는 시퀀스 파라미터 세트(SPS) 대신에, 적응 파라미터 세트(APS) 또는 타일(그룹) 헤더가 또한 ALF 파라미터들(또는 ALF 필터링을 사용하는 것에 대한 정보)를 시그널링하는 데 사용될 수 있음이 또한 이해된다. ALF 파라미터들(또는 ALF 필터링을 사용하는 것에 대한 정보)를 시그널링하기 위해 APS가 사용될 때, 슬라이스 헤더 또는 타일 그룹 헤더는 ALF 파라미터들(또는 ALF 필터링을 사용하는 것에 대한 정보)를 획득하기 위해 어느 APS가 사용되어야 하는지를, 예를 들어, 적응 세트 식별자(aps_id)를 표시하는 것에 의해, 나타내기 위해 사용될 수 있다. 슬라이스, 타일 그룹, 타일, 코딩 트리 유닛(CTU)/최대 코딩 유닛(LCU), 코딩 트리 블록(CTB), 코딩 유닛(CU), 예측 유닛(PU), 변환 유닛(TU), 또는 픽셀들/샘플들의 블록 중 임의의 것이 이미지 부분이라고 지칭될 수 있음이 또한 이해된다.

[0038] 필터 또는 도구가 "활성"으로서 기술될 때는, 필터/도구가 "인에이블되거나" 또는 "사용 가능하거나" 또는 "사용되고"; "비활성"으로서 기술될 때는, 필터/도구가 "디스에이블되거나" 또는 "사용 가능하지 않거나" 또는 "사용되지 않으며"; "클래스"가 하나 이상의 요소의 그룹, 그룹화한 것(grouping), 카테고리 또는 분류를 지칭한다는 것이 또한 이해된다. 게다가, 플래그가 "활성"으로서 기술될 때, 이는 플래그가 관련 필터/도구가 "활성"임

을 나타낸다는 것을 의미한다는 것이 또한 이해된다.

[0039] 적응 루프 필터(ALF)

[0040] 도 1은 VTM-3.0의 전형적인 디코딩 루프 내의 어디에서 ALF가 이루어지는지를 도시한다. 101에서, 이미지 부분(예를 들면, 슬라이스)이 코딩 트리 유닛(CTU: VVC에서의 최대 코딩 유닛, 전형적으로 128x128 샘플/픽셀 크기임)의 단위로 디코딩된다. CTU는 직사각형 블록들 또는 코딩 유닛들(CU)로 분할되며, 이들은 특정 예측 스킴/모드를 사용하여 인코딩되며, 종종 잔차 블록의 손실 인코딩이다. 블록 기반 인코딩의 사용으로 인해, 인코딩된 블록들 사이의 경계들에서 블로킹 아티팩트들이 보일 수 있다. 102에서, 디코딩된 이미지 부분은 이어서 해당 아티팩트들을 감소/제거하기 위해 DBF에 의해 프로세싱된다. 전형적으로, 블록 예측을 위한 잔차(블록)를 인코딩하기 위해, (몇몇 계수들에서의 잔차 에너지를 압축하기 위해) 잔차 값들이 DCT 유사 변환을 사용하여 변환되고, 변환된 계수들이 인코딩 비용을 감소시키기 위해 양자화된다. 이러한 양자화는 종종 재구성된 블록들(즉, 프레임 버퍼(106)에 저장되는 참조 프레임들에서의 블록들)에 얼마간의 링잉(ringing) 아티팩트들을 도입한다. 103에서, DBF의 출력 이미지 부분은 이어서 SAO 필터에 의해 프로세싱되는데, 이는 낮은 계산 비용으로 이러한 아티팩트들을 중 일부를 감소시키는 데 유용하다. 104에서, SAO 필터의 출력 이미지 부분은 이어서 ALF에 의해 프로세싱된다. ALF는, 예를 들어, "링잉"과 같은 아티팩트들을 추가로 감소시킬 수 있다. ALF는 상위 차수 에러 모델링 능력을 가지고 있지만 계산 비용이 더 높다. ALF의 출력 이미지 부분은 이어서 출력(예를 들면, 디스플레이 또는 디스플레이와 통신하기 위한 통신 인터페이스)(105)으로 보내진다. 이는 또한, (시간 예측 도구들이 사용될 때) 시간 예측을 위해 사용될 수 있도록, 프레임 버퍼(106)에 (그 안에 저장된 참조 프레임의 일 부분으로서) 넣어질 수 있다. 이것이 DBF, SAO 필터 및 ALF가 "인루프" 필터라고 불리는 이유이다. 인코더는 인루프 필터들 중 일부를 디스에이블시킴으로써 디코딩 시에 이들이 바이패스될 수 있도록(즉, 필터링이 수행되지 않고 디스에이블된 도구에 대응하는 단계의 출력이 그의 입력과 동일하도록) 할 수 있다. 또한, 일부 경우에, 프로세싱된 이미지 부분은 슬라이스로 제한되지 않고 하나 또는 다수의 슬라이스를 포함하는 전체 프레임(full frame)일 수 있으며, (하나 초과가 존재하는 경우) 슬라이스 경계들에 걸쳐 필터들을 적용하여 해당 경계들에서 아티팩트를 감소시킬 수 있다. 다중 성분 이미지(예를 들면, YCrCb 포맷으로 된 이미지)의 경우, DBF, SAO 필터 또는 ALF 프로세싱이 각각의 성분에 개별적으로 그리고 어쩌면 상이하게(예를 들면, 다른 성분에 대해 상이한 필터링 파라미터를 사용하여) 적용된다.

[0041] 도 2는 ALF에 대한 VTM-3.0에 존재하는 신택스 요소들의 개관을 제공한다. 시퀀스 파라미터 세트(SPS)는, 플래그를 사용하여, 비디오 시퀀스에 대해 ALF 도구가 활성인지(즉, 인에이블되어 있는지)(201)를 나타내고, 만약 그렇다면, 슬라이스 헤더는 ALF가 슬라이스에 대해 활성인지(202)를 나타내며 ALF를 제어하기 위한 필터 파라미터들을 제공한다(203 내지 207). 202에서 ALF가 활성일 때, ALF는 적어도 루마 성분에 대해 활성이고, 슬라이스 헤더는 게다가 ALF가 크로마 성분들 각각에 대해 활성인지(203)를 나타낸다.

[0042] ALF는 루마 성분에 대해 하나 초과의 필터를 사용할 수 있다. 즉, 상이한 필터 구성들이 동일한 필터 계수들을 공유할 수 있기 때문에 하나 초과의 필터 계수 테이블이 사용될 수 있다(추후에 기술됨). VTM-3.0 소프트웨어의 신택스 요소 스킴에서, 이러한 상이한 필터 구성들이 개별적으로 구별 가능하지 않으며, 상이한 구성들을 갖는 필터들이 동일한 필터로서 간주된다. 이하의 설명에서, 상이한 계수 테이블들에 대한 명시적인 언급들이 이루어질 때를 제외하고는 단일 필터에 대한 동일한 언급이 사용된다. 신택스 요소 스킴에서의 단일 필터의 이러한 공유는 필터링된 이미지에 대한 통계가 회전된 직교 및/또는 미러 구성들에 대해 동일하다는 점을 고려하는 것에 의해 필터들에 할당되는 비트 수를 감소시키는 효과적인 방식이며, 즉, 기준 샘플 구성을 위해 설계된 하나의 기준 필터가 직교 및/또는 미러 샘플 구성들을 필터링하기 위해 회전 및/또는 미러링되는 것으로 지정된다. 슬라이스 헤더는 (하나 이상인) 인코딩된 루마 필터들의 수(204)를 포함한다.

[0043] 루마 샘플들을 필터링할 때, ALF는 샘플들을 (이웃하는 샘플들의 구성에 따라) 25개의 가능한 클래스(카테고리/분류/그룹) 중 하나로 로컬로 분류(카테고리화)하여 해당 샘플들에 로컬로 적용할 (해당 특정 클래스/카테고리/분류/그룹과 연관되고/되거나 그에 할당되는) 필터를 선택한다. 여기서, "로컬로" 분류하다 및 "로컬로" 적용하다라는 용어들이 사용되는데 그 이유는 샘플들이 블록(전형적으로, 4x4 샘플들, 예를 들면, VTM-3.0에서) 단위로 또는 CU 단위로 프로세싱되기 때문이다. 선정된/선택된 루마 필터는 이어서, 예를 들면, 선정된 루마 필터를 식별하기 위한 인덱스/정보를 사용하여, 비트스트림에서 시그널링된다. 인코딩 프로세스 동안 사용되는 루마 필터들의 수가 1개 초과일 때, 슬라이스 헤더는 해당 루마 필터들을 식별/선택하기 위한 하나 이상의 인덱스(들)(예를 들면, 25개의 클래스에 대해 사용되는 최대 25개의 필터에 대한 인덱스들, 각각의 인덱스는 해당 클래스들 중 하나에 사용되는 루마 필터에 대응함)(205)를 또한 포함한다. 인코딩 프로세스에서 사용되는 루마

필터들의 수가 1개인 경우에, 이 단일 루마 필터가 모든 클래스들에 적용된다.

[0044] 이어서, 슬라이스 헤더는 루마 필터들 각각에 대한 모든 필터 계수들(또는 필터 파라미터들)(206)(즉, 인코딩 프로세스 동안 사용되는 루마 필터들 각각에 대한 계수 테이블)과 뒤이어서 크로마 필터에 대한 필터 계수들(207)을 포함한다. VTM-3.0에서 ALF가 크로마 성분들 둘 모두에 대해 활성일 때(즉, 인에이블될 때) 2개의 크로마 성분은 동일한 크로마 필터를 공유한다는 점에 유의한다.

[0045] ALF가 활성일 때, ALF가 활성인 성분들 각각에 대해, 필터링이 CTU별로 인에이블될 수 있다. 인코딩된 비트스트림 내부에서, 각각의 CTU에 대해, 해당 CTU의 성분에 대해 ALF가 인에이블되어 있는지 여부(208, 209 및 210) 그리고 따라서 해당 CTU의 해당 성분의 샘플들이 ALF를 사용하여 필터링되어야 하는지 여부를 나타내기 위해, 인코딩된 슬라이스 데이터는, ALF가 활성인 각각의 성분마다, 하나의 엔트로피 코딩된 플래그를 포함한다. 이 플래그는 콘텍스트 적응 이진 산술 코딩(context-adaptive binary arithmetic coding, CABAC)을 사용하여 인코딩된다.

[0046] VTM-3.0에서, 루마 필터들에 대한 계수들(206)의 시그널링은 다음과 같이 수행된다:

[0047] [1] 'alf coefficients delta flag'가 먼저 시그널링되고 - 이 플래그는 일부 필터들이 디스에이블될 수 있는지 여부를 나타냄 -, 'alf coefficients delta flag'가 0이고 루마 필터들의 수가 1개 초과인 경우, 'coeff delta pred mode flag'가 시그널링되어, 필터 계수 인코딩이 예측을 사용할 것임을 나타낸다(추후에 더 상세히 기술됨).

[0048] [2] VTM-3.0에서, (지수-)골롬((exp-)Golomb) 코드들을 사용하여 루마 계수들을 인코딩하기 위해, 3개(크로마의 경우 2개)의 (지수-)골롬 구성을 사용한다. (지수-)골롬 인코딩의 유일한 파라미터는 (지수-)골롬 차수(종종 'k'로 표기됨)이다. 각각의 구성은 (지수-)골롬 인덱스를 갖는다. 필터 계수의 (지수-)골롬 인코딩에 대한 파라미터들은 (지수-)골롬 코드들의 '최소 차수', 및 이어서, 각각의 (지수-)골롬 인덱스에 대해, (첫 번째 인덱스에 대해 '최소 차수'로 시작하여) (지수-)골롬 차수가 해당 인덱스와 다음 인덱스들에 대해 증가되어야 하는지를 시그널링하는 플래그에 대해 가변 길이 코드(VLC)를 사용하여 시그널링된다.

[0049] [3] 이어서, 'alf coefficients delta flag'가 일부 필터들이 인에이블되어 있음을 나타내는 경우, 각각의 필터에 대해, 해당 필터가 디스에이블되는지(따라서 코딩되지 않는지) 여부를 나타내는 플래그가 시그널링된다.

[0050] [4] 이어서, 각각의 (디스에이블되지 않은) 필터에 대한 필터 계수들이 (부호 있는 정수에 대한) (지수-)골롬 코드들을 사용하여 시그널링되고, (지수-)골롬 차수는 (지수-)골롬 파라미터들을 저장하는 테이블로부터의 (고정된 테이블 내의) 필터 계수 인덱스와 연관된 (지수-)골롬 인덱스를 사용하여 취해진다.

[0051] 'coeff delta pred mode flag'가 시그널링되는 경우에, 이는 필터 계수 인코딩이 예측을 사용한다는 것을 나타낸다. 이는 필터 인덱스(즉, 각각의 필터를 식별해 주는 인덱스)가 1 이상인 '현재' 필터의 필터 계수들이 "현재" 필터의 필터 계수들과 이전에 프로세싱된 필터의 필터 계수들 간의 차이들로서 인코딩된다(예를 들면, 이전에 프로세싱된 필터의 필터 계수들을 필터 계수 예측자들로서 사용하여 - 즉 예측을 사용하여 - 필터 계수 잔차값으로서 인코딩된다)는 것을 의미한다. 첫 번째 필터(필터 인덱스가 0임)는 예측을 사용하지 않고 인코딩된다.

[0052] 크로마 필터의 계수(207)의 시그널링은, 'alf coefficients delta flag'가 없고, 'coeff delta pred mode flag'가 없으며, 3개 대신에 2개의(지수-)골롬 인덱스가 있다는 점을 제외하고는, 루마 필터들에 대해서와 유사하게 수행된다.

[0053] 도 3a는 본 발명의 실시예에 따른 크로마 성분을 필터링하기 위한 단계들을 예시하는 플로차트이고, 도 3b는 본 발명의 실시예에 따른 크로마 필터에 대한 필터 형상 및 계수 배열을 제공한다.

[0054] 도 3a에서, 루마 성분과 크로마 성분 간에 ALF 필터링이 상이하게 적용된다. 간단한 사례로 시작하면, 도 3a는 크로마 성분을 필터링하기 위한 주요 단계들을 제공한다. 입력 이미지 부분(301)(예를 들면, 타일 또는 타일 그룹)은 필터링될 크로마 샘플들을 포함한다. 입력 필터링 파라미터들(302)은 도 2에서의 ALF 신택스 요소들을 참조하여 기술된 ALF 파라미터들(예를 들면, 'alf coefficients delta flag', 'coeff delta pred mode flag', 필터 계수들, 또는 ALF에 대한 임의의 다른 플래그들 또는 필터 파라미터들)을 포함한다. 크로마 필터에 대한 인코딩된 필터 계수들을 사용하여, 303에서 크로마 필터가 도출/획득된다.

[0055] 변형례에서, 크로마 필터는 5x5 크기, 즉, 5개 샘플 높이와 5개 샘플 폭(도 3b에서의 306 참조)의 다이아몬드 형상/패턴/마스크/서포트(support)를 갖는다. 크로마 필터의 필터 계수들은 크로마 필터가 중심에 대해 대칭성

을 갖도록 편성된다. 이러한 크로마 필터에 대해, 도면에 도시된 대응하는 계수 위치들에 배치되는, 0부터 5까지 번호가 부여된 인덱스를 갖는, 6개의 인코딩된 필터 계수가 있다. 동일한 필터 계수를 공유하는 상이한 계수 위치들이 구분될 수 있도록, 인덱스 번호에 대한 위 첨자 부호("+" 또는 "-")가 제공되며, 즉, 2개의 대칭적인(동일한 필터 계수를 공유한다는 의미에서) 이웃을 지칭할 때: i^+는 (래스터) 스캔 순서에서 (즉, 필터 형상/패턴/마스크/서포트에서 중심에 있는) 필터링될 샘플 이후에 인코딩/디코딩/프로세싱/저장/액세스되는 인덱스 i를 갖는 이웃 샘플에 대응하고; i^-는 (래스터) 스캔 순서에서 필터링될 샘플 이전에 인코딩/디코딩/프로세싱/저장/액세스되는 인덱스 i를 갖는 이웃 샘플에 대응한다. 크로마 필터 형상의 중심에 대한 7 번째 계수(인덱스 번호 6을 갖는 계수 위치)는 다른 필터 계수들로부터 추론/도출된다. 계수 위치 6에 있는 이 7 번째 계수의 값은, 고정 소수점 계산으로 구해지는, $1-2.\sum_{i<6} w_i$ (w_i = 인덱스 i에 대한 필터 계수)과 동일하다. 즉, 1에서 계수 위치 0 내지 5에 대한 모든 필터 계수들의 합의 2배(대칭으로 인해)를 뺀 것이다. 따라서, 7 번째 계수를 포함한, 다이아몬드 형상의 모든 계수들의 합은 1이고 이 1에 대해 좌측 7 비트 시프트가 적용되며(1 << 7), 여기서 7은 고정 소수점 계산을 위해 인코딩된 필터 계수들의 비트 정밀도에서 1을 뺀 것이다. 이것이 "필터 계수들의 총수에서 1을 뺀 것"의 절반만이 비트스트림에 인코딩되는 필터 계수들의 수인 이유이다.

[0056] ALF가 활성인(즉, 적용되는) 각각의 크로마 성분 샘플에 대해, 출력 이미지 부분(305)을 획득하기 위해 단계(304)에서의 필터링이 다음과 같이 수행된다: 입력 이미지 부분의 각각의 현재 샘플에 대해,

[0057] - 현재 샘플이 ALF가 인에이블된 CTU에 속하고 크로마 필터의 다이아몬드 형상에서의 계수 위치들에 대한 필터 계수들을 획득하는 데 필요한 현재 샘플(현재 샘플은 다이아몬드 형상의 중심에 위치됨)의 이웃하는 샘플들이(예를 들어, 픽처의 경계들에서 테두리 확장(border extension)을 사용하여 또는 슬라이스의 테두리에서 이용 가능한 경우 이웃 슬라이스 샘플들을 사용하여) 이용 가능한 경우; (x, y) 위치에 있는 현재 샘플 I(x, y)와 동일한 위치에 있는 출력 필터링된 샘플은 수학식 1과 동일하고:

수학식 1

[0058] $$O(x,y) = \left((1 \ll (N-1)) + \sum_{(i,j)} w(i,j).I(x+i,y+j)\right) \gg N$$

[0059] 여기서 i와 j는 필터 형상의 중심(즉, (x, y)에 있는 현재 샘플의 위치)에 상대적인 2개의 정수 오프셋(수평 및 수직)이고, w(i, j)는 오프셋 (i, j)에 있는 필터 계수이며, N=7은 오프셋 (i, j)에 있는 필터 계수 w(i, j)의 표현에서 사용되는 실수의 소수 부분(고정 소수점 표현)의 정수 근사(integer approximation)를 위한 비트 수이고, I(x + i, y + j)는 현재 샘플 위치 (x, y)에 상대적인 오프셋 (i, j)에 있는 입력 샘플 값이며, O(x, y)는 위치 (x, y)에 대한 출력 필터링된 샘플 값이다. a << N은 a의 정수 값에 N 비트의 좌측 비트 시프트가 적용된다는 것을 의미한다. 이는 2의 N제곱으로 정수 곱셈을 수행하는 것과 동등하다. a >> N은 a의 정수 값에 N 비트의 우측 비트 시프트가 적용된다는 것을 의미한다. 여기서, 수학식 1의 괄호 안의 합계의 결과가 대부분의 경우 양(positive)이기 때문에, 따라서 이는 2의 N제곱으로 정수 나눗셈을 수행하는 것과 동등하다. 음수의 경우, 우측 시프트할 때 부호가 전파되며, 따라서 음수는 음(negative)(적어도 -1)으로 유지될 것이다. ALF의 출력이 일반적으로 0과 2의 비트 깊이 거듭제곱에서 1을 뺀 것 사이로 클리핑되기 때문에 정수가 아닌 부분이 없다. N=7은 ALF 계산을 위해 VVC에서 고정된 십진 정밀도를 제공하지만 다른 실시예들에서 다른 값들이 사용될 수 있다. 우측 시프트 >> N을 수행하기 전에 (1 << (N - 1)) 가산하는 것의 효과는 스칼라 곱의 고정 소수점 결과를 반올림하는 것이다.

[0060] - 그렇지 않은 경우, 동일한 위치에 있는 현재 샘플 값을 그대로(즉, 이 ALF를 적용하지 않고) 출력한다.

[0061] 이 실시예의 변형례에 따르면, 필터를 구현하는 동안 수행되는 곱셈 연산들의 수를 감소시키고 표기법을 단순화하기 위해, 중심에 대해 대칭인 필터 형상을 갖는 ALF 필터에 대해, 수학식 1은 다음과 같이 재구성될 수 있으며:

수학식 2

[0062] $$O_n = \left((1 \ll (N-1)) + w_c . I_n^c + \sum_{i=0}^{i<c} w_i . \left(I_n^{i^-} + I_n^{i^+}\right)\right) \gg N$$

[0063] 여기서 O_n은 래스터 스캔 순서 인덱스/위치 n에 있는 출력 샘플이다. 래스터 스캔 순서 인덱스 n은 샘플들의 행에서 좌측에서 우측으로 증가한 다음에 위에서 아래로 샘플들의 각각의 행을 따라 증가하는 샘플 인덱스인 인덱스 n을 의미한다. I_n^c은 출력 샘플과 동일한 위치 n에 있는 입력 샘플이고(필터의 중심에 있는 입력 샘플의 위치에 대응함), $I_n^{i^-}$은 n보다 낮은 래스터 스캔 순서를 갖는 필터의 필터 형상/패턴/마스크/서포트에서 I_n^c의 i 번째(래스터 스캔 순서로) 이웃하는 입력 샘플이며, $I_n^{i^+}$은 I_n^c의 중심 위치에 대해 $I_n^{i^-}$의 미러링된 공간 위치에 있는 이웃하는 입력 샘플이다. 따라서, 형상이 중심에 대해 대칭이라는 것은, $I_n^{i^-}$이 필터의 형상/패턴/마스크/서포트에 있을 때, $I_n^{i^+}$이 또한 필터의 동일한 필터 형상/패턴/마스크/서포트에 있다는 것을 의미한다. w_i는 이웃하는 입력 샘플들 $I_n^{i^-}$ 및 $I_n^{i^+}$과 연관된 필터 계수이고, w_c는 중심 입력 샘플 I_n^c에 대한 필터 계수이며, c는 인코딩된 필터 계수들의 수이다(이는, 중심 필터 계수의 이웃하는 필터 계수들로부터 구해질 수 있기 때문에 인코딩되지 않는, 중심 필터 계수에 대한 인덱스 값과 동일하다). i의 값들 및 $I_n^{i^-}$과 $I_n^{i^+}$의 연관된 위치들은 도 3b의 필터 형상(306)에서의 인덱스 값들 및 위 첨자 부호("+" 또는 "-")를 갖는 인덱스 값들에 대응하며, 예를 들어, 필터 형상의 중심에 있는 필터 계수의 인덱스인 c에 대해, i =c="6"이다.

[0064] 도 4a는 본 발명의 실시예에 따른 루마 성분을 필터링하기 위한 단계들을 예시하는 플로차트이고, 도 4b는 본 발명의 실시예에 따른 루마 필터에 대한 필터 형상 및 4개의 가능한 계수 배열을 제공한다.

[0065] 도 4a는 루마 성분에 대한 필터링 프로세스의 주요 단계들을 예시한다. 입력 이미지 부분(401)은 필터링될 루마 샘플들을 포함한다. 입력 필터링 파라미터들(402)은 도 2에서의 ALF 신택스 요소들을 참조하여 기술된 ALF 파라미터들(예를 들면, 'alf coefficients delta flag', 'coeff delta pred mode flag', 필터 계수들, 또는 ALF에 대한 임의의 다른 플래그들 또는 필터 파라미터들)을 포함한다. 필터링하기 전에, 403에서 이미지 부분의 콘텐츠가 분석된다. 이 분석의 주요 목표는 로컬 콘텐츠 배향 및 활동 레벨의 결정을 할 수 있게/가능하게 하는 것이다(단계(405) 참조). 이는 콘텐츠가 동질적인지 또는 임의의 급격한 변동들(대체로 콘텐츠의 강도 또는 콘트라스트)을 갖는지 및 콘텐츠가 (예를 들면, 에지들 또는 배향된 텍스처들에 기초하여) 우세한 배향을 갖는지 여부 및 우세한 배향이 어느 배향인지에 대한 로컬 추정/평가를 할 수 있게/가능하게 한다. 예를 들어, VTM-3.0에서, 분석은 수평으로 및 수직으로 2개의 샘플마다(즉, 샘플들의 1/4에 대해) 4개의 배향(수평, 수직 및 2개의 대각선)에 대해 계산되는 라플라시안 값들을 사용하는 로컬 구배(local gradient) 분석을 포함한다. 입력 이미지 부분의 샘플들을 블록들(예를 들면, VTM-3.0에서 4x4 샘플)로 분할(404)하고 분석의 결과들을 사용하는 것에 의해, 단계(405)에서, 각각의 블록은 25개의 가능한 클래스 중 하나로 분류되며, 각각의 클래스는 블록 내의 샘플들에 대해 계산되는 라플라시안 값들에 따른 인덱스(즉, 블록은 25개의 카테고리/분류/그룹 중 하나로 카테고리화됨)를 사용하여 식별 가능하다. 예를 들어, VTM-3.0에서, 이는 16개의 라플라시안 값(4개의 샘플에 대한 4개의 배향)을 사용하는 것에 대응한다. 분류는 활동, 방향성의 강도의 파티셔닝을 달성하고 수평 및 수직 배향들과 대각선 배향들을 분리한다. 또한, 단계(405)에서, 각각의 블록은 전치 인덱스(transposition index)와 연관된다. 이 전치 인덱스(예를 들면, 'transposeIdx')는 콘텐츠의 배향을 완전히 표현/표시하기 위한 분류에 대한 보충/추가 정보로 볼 수 있다. 4개의 가능한 전치 인덱스가 있다. 블록의 클래스가 블록이 수평 또는 수직임을 나타낼 때, 전치 인덱스는 배향이 북쪽에서 남쪽, 동쪽에서 서쪽, 서쪽에서 동쪽, 또는 남쪽에서 북쪽인지를 추가로 나타낸다. 블록의 클래스가 블록이 대각선임을 나타낼 때, 전치 인덱스는 배향이 북서에서 남동, 북동에서 남서, 남서에서 북동, 또는 남동에서 북서인지를 추가로 나타낸다.

[0066] 클래스 인덱스와 전치 인덱스는 주어진 샘플 블록에 대한 적응 루프 필터 파라미터들로 볼 수 있다. 단계(406)는 이러한 파라미터들을 받아서 블록의 샘플들 각각을 필터링하는 데 사용될 루마 필터를 도출한다. 도 2를

참조하여 이전에 기술된 바와 같이, 402에서, 각각의 클래스는 루마 필터들의 계수 테이블의 인덱스와 연관된다. 4x4 블록에 대한 루마 필터를 도출하기 위해, 전치 인덱스는 도 4b에 도시된 4개의 형상/패턴(409, 410, 411 또는 412) 중 하나를 선택하는 것을 할 수 있게/가능하게 한다. 패턴은 루마 필터를 구축하기 위해 204 내지 206에 대한 설명에서 설명된 바와 같이 블록의 클래스와 연관되는 인코딩된 필터 계수들을 (예를 들면, 스캔(닝) 순서에 기초하여) 어떻게 편성할지를 나타낸다. 루마 필터는 7x7 크기의 다이아몬드 형상을 갖는다. 각각의 루마 필터(인덱스 번호 0 내지 11)에 대해 12개의 인코딩된 필터 계수가 있다. 필터 형상의 중심(인덱스 번호 12)에 대한 13 번째 계수는 위에서 기술된 크로마 필터의 중심 계수와 동일한 방식으로 다른 필터 계수들로부터 추론/도출된다.

[0067] 출력 이미지 부분(408)을 획득하기 위한 단계(407)에서의 필터링은 크로마 필터에 의해 필터링이 수행된 방법과 동일한 방식으로 수행되며, 즉 각각의 현재 샘플 블록에 대해, 현재 블록에 대해 406에서 도출된/획득된 루마 필터가 현재 블록의 각각의 현재 샘플에 대해 적용된다.

[0068] 루마 샘플들을 필터링할 때, 배향 기반 분류(클래스) 및 연관된 전치(transposeIdx)로 인해, 각각의 클래스에 대한 최적의 ALF 필터들(즉, 로컬 배향에 따라, 예를 들면, 로컬 구배에 기초하여 선택되는 "배향 기반 필터들"), 특히 높은 활동(예를 들면, 높은 로컬 구배들)을 갖는 것들이 배향된 필터들인 경향이 있다. 가장 낮은 활동을 갖는 영역, 즉 로컬 구배들이 상대적으로 작은 클래스(들)에 속하는 블록들을 갖는 영역의 경우, 그러한 배향은 덜 두드러지고 그러한 블록에 대한 배향 기반 필터들은 배향된 필터가 아닌 경향이 있다.

[0069] 배향 기반 필터들은 에지들의 선명도에 너무 많은 영향을 주지 않으면서 에지들 주위를 필터링하는 데 적합하다. 아래에서 기술되는 바와 같이, 필터들의 시그널링 비용을 감소시키기 위해 클래스들이 함께 그룹화될 수 있다. 클래스들의 이러한 그룹화를 사용하더라도, 필터 최적화는 일반적으로 배향된 필터들을 획득하게 할 것이다. 그러한 배향 기반(ALF) 필터들을 사용하면, 로컬 구배가 수평 방향을 따라 있을 때, 필터 배향은 일반적으로 수직이다(즉, (절댓값에서) 가장 높은 계수들은 일반적으로 중심 위와 아래의 위치들에 있는 반면, 중심의 좌우에 있는 계수들은 일반적으로 0에 더 가까울 것이다). 유사하게, 로컬 구배가 수직 방향을 따라 있을 때, 필터 배향은 일반적으로 수평이고, 로컬 구배가 대각선 방향을 따라 있을 때, 필터 배향은 일반적으로 로컬 구배의 해당 대각선 방향에 수직인 방향을 따라 있다.

[0070] VTM-3.0에서 크로마 샘플들을 필터링할 때, 루마 샘플 필터링과 달리, 분류가 사용되지 않기 때문에(그리고 크로마가 종종 매끄럽기 때문에), 사용되는 필터들은 일반적으로 모든 배향들에 더 잘 반응하고 배향된 필터가 아니다(또는 배향 기반 필터가 아니다).

[0071] 도 3b와 도 4b 둘 모두에서, 필터 형상들은 중심 픽셀에 대해 대칭이다. 이 대칭성은 VTM 소프트웨어에서 ALF의 설계에서 선택되었지만, 실시예들의 변형례에서, 비대칭 형상도 사용될 수 있다. 이어서, 13개의 입력 샘플에 대해 7개의 계수를 공유하는 대신에, 도 3b에서와 같이 동일한 필터 서포트/마스크에 대해 최대 13개의 계수가 사용될 수 있다. 그리고 25개의 입력 샘플에 대해 13개의 계수를 공유하는 대신에, 도 4b에서와 같이 동일한 필터 서포트/마스크에 대해 최대 25개의 계수가 사용될 수 있다. 다른 변형례들에서는, 필터 서포트/마스크 조차도 비대칭이다. 변형례들에서, 필터의 형상/패턴/서포트/마스크는 ALF 파라미터들과 함께 전송된다.

[0072] 다른 변형례들에서, 필터 형상/패턴/서포트/마스크는 다이아몬드 형상과 상이하다. 예를 들어, 변형례에서, 이는 정사각형이고, 다른 변형례에서, 이는 직사각형이며, 또 다른 변형례에서, 이는 육각형이고, 다른 변형례에서는, 이는 팔각형이다.

[0073] 일부 변형례들에서, 필터 형상/패턴/서포트/마스크가 모든 클래스 배향들에 대해 동일한 것은 아니다. 일 변형례에서, 형상은 수평 클래스/transposeIdx 배향 구성들의 경우 수평 직사각형(예를 들면, "-"), 수직 클래스/transposeIdx 배향 구성들의 경우 수직 직사각형(예를 들면, "|"), NW-SE 및 SE-NW 클래스/transposeIdx 배향 구성들의 경우 북서 남동(NW-SE, 예를 들면, "\") 직사각형, NE-SW 및 SW-NE 클래스/transposeIdx 배향 구성들의 경우 북동 남서(NE-SW, 예를 들면, "/") 직사각형이다. 다른 변형례들에서, 필터 형상/패턴/서포트/마스크는 수평-수직 십자형("+"), 대각선 십자형("X"), 수직 세그먼트("|"), 수평 세그먼트("-"), 좌측 상단에서 우측 하단으로의 대각선 세그먼트("\"), 우측 상단에서 좌측 하단으로의 대각선 세그먼트("/") 또는 전술한 필터 형상들/패턴들/서포트들/마스크들의 임의의 조합이다.

[0074] 도 6은 ALF 내의 선형 필터의 블록 다이어그램을 제공한다. 이는 필터링이 기능적으로 수학식 1에 대응하는 ALF의 예시적인 구현에 대응하며, 'C'(601)는 현재 샘플의 샘플 값인 I(x, y)에 대응하고, 'S0'(602) 내지 'Sn'(603)은 이웃하는 샘플 값들인 각각의 (i, j) ≠ (0, 0)에 대한 I(x + i, y + j)에 대응하며, 'wC'(604)는

현재 샘플에 대한 필터 계수인 w(0,0)에 대응하고, 'w0'(605) 내지 'wn'(606)은 위치 (x + i, y + j)에 있는 이웃하는 샘플들에 대한 필터 계수들인 각각의 (i, j) ≠ (0, 0)에 대한 w(i, j)에 대응하며, 'O'(607)는 ALF 필터의 출력, 즉, 현재 샘플의 필터링된 샘플 값인 O(x, y)에 대응한다. 이 프로세스는, 고정 소수점 정밀도를 갖는 실수의 정수 표현으로 된 정수 입력 샘플 값들(즉, 'C'및 'S0' 내지 'Sn')의 테이블 및 입력 가중치들, 즉 'wC' 및 'w0' 내지 'wn'의 테이블(또는 필터 계수들 - 이 테이블은 정수 입력 샘플 값들의 테이블과 동일한 수의 요소들을 가진다는 점에 유의함)을, 입력으로서, 받는다. 2개의 테이블은 동일한 수의 요소들을 가지며, 요소별로 곱해지고, 해당 곱셈들의 결과는 함께 합산된다(즉, 해당 2개의 테이블의 요소들의 선형 결합이 획득된다). 이 연산의 결과는 실수의 고정 소수점 정밀도 근사이다. 이 고정 소수점 값은 정수 값으로 반올림되며, 그 결과는 하나의 출력 샘플 값 'O'이다.

[0075] 인코더의 관점에서, ALF는 위너(Wiener) 필터에서 영감을 받았다. 위너 필터는 1) 유한한 수의 관찰된 프로세스들/변수들(그의 입력들)의 선형 결합인, 추정된 랜덤 프로세스/변수(그의 출력)와, 2) 원하는 프로세스/변수(그의 타깃, 즉 아티팩트들이 발생하기 전의 원래 이미지) 사이의 평균 제곱 오차를 최소화하는 선형 필터(신호/이미지 프로세싱에서 선형 콘볼루션 필터로서 종종 적용됨)이다. 신호/이미지 프로세싱에서, FIR(finite impulse response) 위너 필터는, 예를 들어, 소스 분리 또는 잡음 제거(denoising)에 적용된다. 이미지 코딩의 경우에, 타깃은 (압축/양자화에 의해 변경되기 이전의) 원래 이미지인 반면, 입력들은 필터를 적용하는 것에 의해 개선시키기를 원하는 압축 이미지로부터의 샘플들이다.

[0076] X(관찰된 랜덤 프로세스들의 실현들의 입력 행렬임, 각각의 열은 랜덤 프로세스들 각각에 대한 하나의 실현을 포함함) 및 y(동일한 열 인덱스에서의 관찰된 랜덤 프로세스에 대한 원하는 프로세스의 실현을 포함하는 출력 행 벡터임)에 대한 최소 제곱 해는 다음과 같다.

수학식 3

[0077] $$\hat{w} = (XX^T)^{-1}Xy^T$$

[0078] 위너 필터 계수들은 $\hat{w}$에 대응한다.

[0079] '실현'이 관찰 또는 랜덤 변수의 관찰된 값, 즉 실제로는 실제로 관찰되는 값임이 이해된다.

[0080] VTM-3.0 ALF 인코더에서, FIR 위너 필터(또는 기능적으로 동등한 최소 제곱 해)는 ALF 파라미터들의 코딩 비용(FIR 필터 계수들을 인코딩하기 위한 비용에 의해 주로 좌우됨)과 인코딩된 ALF 파라미터들을 사용하여(즉, FIR 필터들을 사용하여) 픽처를 필터링하는 것에 의해 획득되는 왜곡 이득 간의 레이트/왜곡(Rate/Distortion, R/D) 절충을 최적화하는 데 사용된다. ALF 파라미터들을 인코딩하기 위한 레이트(즉, 코딩 비용)가 문제가 되지 않고 주어진 프레임의 PSNR(Peak Signal to Noise Ratio)을 최대화하는 것이 유일한 목표인 경우(시간적 영향을 고려하지 않음), 위너 필터는 VTM-3.0에서의 ALF 필터링 설계를 위한 최적의 솔루션을 달성하는 것을 가능하게 한다. 따라서 본 발명의 실시예에 따른 ALF 인코더는 그와 함께 제공되는 선형 ALF 필터와 동일하거나 유사한 필터를 사용할 수 있다.

[0081] 도 9는 본 발명의 실시예에 따른 ALF 인코딩 프로세스를 예시하는 플로차트를 도시하며, 이 실시예는 도 6을 참조하여 기술된 VTM-3.0의 ALF를 수정하는 것에 의해 구현된다. 본 발명의 다른 실시예들에 따르면, 다른 ALF가 상기 다른 실시예들을 구현하기 위해 동일한 방식으로 수정될 수 있음이 이해된다.

[0082] ALF 인코딩 프로세스는 각각의 4x4 루마 샘플 블록에 대한 클래스 인덱스 및 전치 인덱스를 결정하는 것(901)으로 시작된다.

[0083] 이어서 902에서 위너 필터를 도출하는 데 사용될 통계가 추출/획득된다. 이러한 통계는 수학식 3에서의 XX^T에 대응하는 (자기) 공분산 통계, 및 수학식 3에서의 Xy^T에 대응하는 교차 공분산 통계이다. 이들은, XX^T 및 Xy^T를 N(X에서의 열들의 수, 즉 필터링할 샘플들의 수)으로 제각기 나누는 것에 의해, (자기) 공분산 행렬 및 교차 공분산 행렬을 구축/획득/추정하는 데 사용된다. 각각의 클래스에 대해 그리고 루마 성분 샘플들의 각각의 CTU에 대해, 그리고 각각의 크로마 성분 샘플들의 각각의 CTU에 대해 이러한 (자기) 공분산 행렬 및 교차 공분산 행렬이 구축/획득/추정된다(902). 이하의 설명에서, '(교차) 공분산 행렬 통계'와 '(교차) 공분산 행렬'이라는 용

어들은 동일한 것을 지칭하기 위해 상호 교환 가능하게 사용된다. 이 둘 사이의 차이점이 '(교차) 공분산 행렬 통계'가 값들을 누적(또는 합산)하는 것에 의해 획득되는 반면 '(교차) 공분산 행렬'이 또한 값들을 누적(또는 합산)하는 것에 의해 획득되지만, 이어서 그것이 (기댓값을 추정하기 위해) 누적 횟수에 의해 정규화된다는 것임이 이해된다.

[0084] 주어진 CTU/클래스/성분에 대해, X는 다음과 같이 획득된다. 필터들의 형상이 대칭인 것으로 간주되기 때문에, X에서의 행들의 수는 필터의 필터 계수들의 수에 대응한다. X의 하나의 행(마지막 행)은 필터 형상에서의 중심 샘플(필터 형상의 중심은 주어진 CTU/클래스/성분에 속함)의 실현들을 포함하는 반면, 인덱스 i를 갖는 각각의 다른 행은 필터 형상에서의 인덱스 i를 갖는 2개의 대칭적인 샘플(대칭적인 샘플들은 주어진 CTU/클래스/성분에 속하는 필터 형상의 중심의 이웃들임)의 합을 포함한다. 루마 성분 샘플들의 경우에, X의 각각의 행 i가 인덱스 i를 갖는 필터의 형상에 속하는 샘플들에 대한 샘플 통계를 포함하도록, 전치 인덱스가 또한 도 4b의 상이한 형상들에 따라 필터 형상의 샘플 위치들을 전치하는 데 사용된다. 예를 들어, 도 4b의 형상들에 대해, $X_{i,j}$는: i <12인 경우, 형상에서의 인덱스 i를 갖는 (j 번째 필터링되는 샘플의) 2개의 대칭적인 이웃의 합, 및 i = 12인 경우, j 번째 필터링되는 샘플을 포함하며; 여기서 $X_{i,j}$는 행렬 X에서 i 행과 j 열에 있는 값이다.

[0085] 벡터 y는 모든 타깃 샘플 값들을 포함한다(즉, y_j는 압축 이전의 소스/원래 이미지의 j 번째 샘플의 값이다).

[0086] 행렬 X는 실제로는 구축/계산되지 않는다. 그 대신에, XX^T는 $X_iX_i^T$의 결과를 반복적으로 합산하는 것에 의해 계산되고, 여기서 X_i는 주어진 샘플 위치에 대해 획득되는 X의 i 번째 열이다.

[0087] 벡터 y도 실제로는 구축/계산되지 않는다. 그 대신에, Xy^T는 X_iy_i의 결과를 반복적으로 합산하는 것에 의해 계산되며, 여기서 y_i는 y의 i 번째 요소이고 입력 X_i를 사용하여 i 번째 입력 샘플을 필터링할 때의 타깃 샘플 값에 대응한다.

[0088] VTM-3.0에서, ALF 인코더는 레이트-왜곡 절충의 라그랑지 최적화를 위한 D + λR과 동일한 R/D 비용을 감소시키려고 한다. 여기서, D는 왜곡(2차 오차(quadratic error))이고, R은 레이트이며, λ는 VTM-3.0 인코더 양자화 파라미터(QP), 슬라이스 유형(인트라(intra) 또는 인터(inter)) 및 압축된 성분의 유형(루마 또는 크로마)에 주로 기초하여 인코더에 의해 결정된다. ALF 인코더는 먼저 이 라그랑지 R/D 비용을 최소화하는 것에 의해 ALF 파라미터들을 최적화하려고 한다.

[0089] ALF가 루마 성분에 대한 비용을 감소시키는 경우, 즉 ALF가 활성이 아닐 때의 출력 슬라이스의 왜곡이 ALF가 활성일 때의 왜곡 + ALF 파라미터들을 시그널링하는 데 레이트의 λ배보다 큰 경우, 인코더는 ALF를 루마 성분에 대해 활성인 것으로 결정한다. 이어서 루마 성분에 대해 활성인 경우, 인코더는, 크로마 성분들을 시그널링하는 R/D 비용을 개선시킬 수 있는지를 확인하기 위해, 해당 크로마 성분들에 대한 ALF 파라미터들을 최적화하려고 한다. 이에 기초하여, 인코더는 해당 성분들 각각에 대해 ALF를 활성화/인에이블시키는 것이 더 나은지 여부를 결정할 수 있다.

[0090] 본 발명의 실시예에 따르면, ALF 인코더는 그의 ALF 파라미터들에 대해 동일하거나 기능적으로 동등한 최적화 프로세스를 수행한다. 그러한 실시예의 변형례에 따르면, ALF 파라미터들 최적화 프로세스의 시작에서, ALF는 모든 CTU들에 대해 활성으로 설정된다(903). 공분산 행렬 및 교차 공분산 행렬을 구축하기 위한 슬라이스 레벨에서의 통계는 ALF가 활성인/인에이블되어 있는 각각의 CTU에 대한 통계를 집계하는 것에 의해 획득된다. 하나의 행렬은, 루마 성분에 대한 각각의 클래스에 대해, 해당 클래스에 속하는 것으로 분류된 4x4 샘플 블록들의 모든 샘플들에 대해 획득되는 통계를 사용하여 계산되고(904); 하나의 행렬은 크로마에 대한 2개의 크로마 성분의 통계를 집계(합산)하는 것에 의해 계산된다(912).

[0091] 루마에 대한 필터 최적화 프로세스는 클래스들을 함께 결합/병합시키기 위한 25개의 필터 그룹, 즉 25개의 필터의 첫 번째 그룹, 24개의 필터의 두 번째 그룹 ... 1개 필터의 마지막 그룹(각각의 가능한 수의 필터들에 대해 하나의 그룹)을 찾는 것(905)으로 시작된다. 인코더는 각각의 클래스에 대해 하나의 필터로 시작하며(따라서 총 25개의 필터), 이 필터는 해당 클래스에 있는 블록들의 샘플들의 공분산 및 교차 공분산으로부터 계산되는 위너 필터이다. 이것은 25개의 필터의 첫 번째 그룹이다. 인코더는 필터들을 함께 병합시키는 것, 즉, 더 정밀하도록 필터들의 연관된 클래스들을 병합시키는 것에 의해 (모든 원하는 그룹들을 획득하기 위해, 하나씩 그리고 단지 하나가 남아 있을 때까지) 필터들의 수를 반복적으로 감소시키려고 한다. 인코더는 처음에 2개의 상이한 필터와 연관된 클래스들을 공통 필터를 공유하도록 만든다(즉, 인코더는 하나의 연관된 필터를 공유하도록

2개의 클래스를 병합시킨다). 어느 필터들을 병합시킬지(즉, 어느 클래스들을 병합시킬지)를 결정하기 위해, 각각의 필터에 대해 공분산 및 교차 공분산 통계가 결정된다. 인코더는, 하나 이상의 클래스(들)(인덱스(들))와 연관된 각각의 필터에 대해, 필터를 사용하여 해당 클래스(들)와 연관된 모든 샘플 블록들에 대해 필터링을 수행한 후에 획득되는 총 잔차 오차를 추정한다/구한다. 이어서 각각의 필터 쌍(및 그와 연관된 클래스들)에 대해, 인코더는, 병합된 클래스들에 대한 위너 필터를 결정하고 결정된 위너 필터를 사용하여 클래스 인덱스와 연관된 모든 샘플 블록들을 필터링한 후에 획득되는 총 잔차 오차를 결정하기 위해, 병합된 공분산 및 교차 공분산 통계를 계산한다. 이어서 인코더는 (2개의 필터와 연관되어 있는 클래스(들)와 연관된 통계로부터 도출되는) 병합된 필터의 총 잔차 오차와 (하나의 필터의 총 잔차 오차를 다른 필터의 총 잔차 오차와 가산하는 것에 의한) 해당 2개의 필터의 총 잔차 오차의 합 사이의 차이가 가장 작은 2개의 필터를 결정하고, 해당 2개의 필터를 병합시킨다(따라서 이 병합된 필터는 다음 필터 병합 반복들을 위해 준비가 되어 있게 된다). 요컨대, 인코더는 $\arg\min_{(a,b)}(Err(a+b)-Err(a)-Err(b))$ 를 최소화하는 2개의 상이한 필터 통계인 (a, b)를 병합시키며, 여기서 Err(x)는 필터 통계 x에 대한 오차를 반환하고, a + b는 2개의 필터 통계 a와 b의 병합된 통계이다.

[0092] 지금까지는, 통계, 필터 및 오차는 배정도 부동 소수점 값을 사용하여 추정된다. 이제 인코더는 루마 필터들을 인코딩하기 위한 최상의 R/D 비용 절충을 찾으려고 한다(906). (25개의 필터에서 시작하여 1개의 필터까지) 25개의 필터/클래스 그룹이 일단 결정되면, 각각의 그룹에 대해, ALF 인코더는 각각의 필터에 대한 (정수 값 인코딩 및 고정 소수점 정밀도 계산을 위한) 정수 필터 계수들을 도출한다. 이어서 인코더는 상이한 대안적인 코딩 스킴들이 사용될 때 R/D 비용 절충 면에서 최상의 필터를 탐색한다. 제1 대안은 (지수-)골롬 인코딩을 사용하여 모든 필터들에 대한 모든 계수들을 인코딩하는 것이다. 제2 대안은 필터들의 델타 인코딩을 사용하는 것이며, 여기서 각각의 필터의 필터 계수들은 ((지수-)골롬 인코딩을 사용하여) 이전 필터의 필터 계수들과의 차이로서 인코딩된다. 제3 대안(최적화된 R/D)은 하나의 플래그를 사용하여 일부 필터들을 디스에이블시키고 디스에이블되지 않은 필터들의 모든 필터 계수들을 ((지수-)골롬 인코딩을 사용하여) 인코딩하는 것을 가능하게 한다. 처음 2개의 대안은 비트레이트의 감소를 가져오는 반면, 제3 대안은 더 작은 비트레이트의 대가로 더 많은 왜곡을 가져올 수 있다.

[0093] 인코더는 R/D 비용을 최소화하는 필터 그룹 및 인코딩 절충을 취한다/선택/선정한다.

[0094] 각각의 CTU에 대해, 사용할 루마 필터들이 인코더에 의해 일단 결정/선택되면, 인코더는 CTU의 루마 샘플들을 필터링하는 R/D 비용이 동일한 CTU의 루마 샘플들을 필터링하지 않는 R/D 비용보다 나은지를 확인하기 위해 CTU 통계를 사용한다. 더 낫지 않은 경우, 해당 CTU의 루마 샘플들에 대해 ALF가 디스에이블된다(907). 인코더는 이어서, 908에서, (ALF가 인에이블되어 있는) CTU의 공분산 및 교차 공분산 통계에 대한 슬라이스 통계를 업데이트하면서 루마에 대한 루마 필터 최적화 단계(904)로 루프백할 수 있다. 예를 들어, VTM-3.0에서, 인코더는 4번 더 루프백한다.

[0095] 루마 성분 샘플에 ALF를 적용하는 것과 그렇지 않은 것 사이의 R/D 비용 차이에 따라, 인코더는, 909에서, ALF가 루마 성분에 대해 활성이어야/인에이블되어야 하는지 여부를 결정한다.

[0096] ALF가 루마 성분에 대해 활성인/인에이블된 경우, 인코더는, 910에서, 크로마 성분(들)을 프로세싱하는 것으로 진행한다. 루마 성분에 대해 ALF가 비활성인/디스에이블된 경우, 911에서, 활성이 아닌/디스에이블된 것으로 ALF에 시그널링되고 ALF 인코딩 프로세스가 종료된다.

[0097] 크로마 성분에 대한 ALF 인코딩 프로세스는, ALF가 슬라이스의 모든 CTU들에 대해 활성인/인에이블된 경우, 2개의 크로마 성분의 통계를 결합시키는 것(912)으로 시작된다.

[0098] 인코더는 이어서 크로마 필터를 결정한다(913). 인코더는 먼저 크로마 성분들 둘 모두의 CTU 통계를 사용하여 (부동 소수점) 위너 필터를 결정한다. 인코더는 정수 필터 계수들을 도출한다. 이어서 각각의 크로마 성분에 대해, 인코더는 CTU의 크로마 성분을 필터링하는 R/D 비용이 CTU의 크로마 성분을 필터링하지 않는 R/D 비용보다 나은지를 확인하기 위해 CTU 통계를 사용한다. 더 낫지 않은 경우, 해당 CTU의 크로마 성분 샘플들에 대해 ALF가 디스에이블된다(914). 인코더가 주어진 크로마 성분의 모든 CTU들에 대해 ALF가 활성이 아니어야 한다(즉, 디스에이블되어야 한다)고 결정하는 경우, 인코더는 해당 크로마 성분에 대해 ALF를 디스에이블시키고, 따라서 각각의 CTU에 대해 0인 'enable flag'를 코딩할 필요가 없다(915).

[0099] 인코더는 이어서, 916에서, (ALF가 인에이블되어 있는) CTU에 대한 크로마 성분들의 공분산 및 교차 공분산 통

계에 대한 슬라이스 통계를 업데이트하면서 크로마 필터 최적화 단계로 루프백할 수 있다. 예를 들어, VTM-3.0에서, 인코더는 2번 더 루프백한다.

[0100] 인코더는 이어서 결정된 ALF 파라미터들을 사용하여(즉, 단계(913)로부터의 결정된 크로마 필터를 사용하여), 917에서 본 발명의 실시예에 따른 ALF를 적용한다. 인코더 구성에 따라, 결과적인 이미지가 출력되고/되거나 참조 프레임 버퍼에 넣어질 수 있다. 인코더는 최종적으로 최상의 R/D 비용 파라미터들, 즉 ALF enable flag 및, ALF enable flag 플래그가 활성/인에이블됨을 나타내는 경우, 결정된 ALF 파라미터들을 인코딩한다(918).

[0101] 변형례들에 따르면, 다른 ALF 파라미터들에 대한 최적화 프로세스가 ALF의 해당 파라미터들을 최적화하기 위해 수행될 수 있다는 것이 이해된다.

[0102] 비선형 필터링 능력을 갖는 ALF

[0103] 단계들(304 및 407)에서 사용되는 선형 필터링이 본 발명의 실시예들에 따라 수정될 수 있으며, 이는 비선형성을 도입하고 필터링 결과들을 개선시킨다(필터링 품질과 코딩 효율 사이의 더 나은 절충을 달성한다). ALF 필터링의 목표는 코딩 도구들에 의해 도입되는 어떤 "잡음"(예를 들면, 양자화 잡음/오차)을 제거하는 것이다. 그러한 잡음을 제거하기 위해, 저역 통과 선형 필터들이 신호를 평활화하고 작은 로컬 변동들을 감소시키기 위해 종종 사용된다. 그러한 종류의 필터들은, 특히 콘트라스트가 강한 영역들, 예를 들어, 에지들 근처에서, 필터링된 출력에 블러링(blurring)을 도입할 수 있다. 예를 들어, 양방향 필터(bilateral filter)와 같은 비선형 필터들이, 에지들 주변에서도 더 적은 블러링 또는 링잉 효과들을 도입하면서, 더 효율적인 잡음 제거를 가능하게 하기 위해 개발되었다. 그렇게 하기 위해, 이러한 비선형 필터들은 종종, 선형 필터들과 같이, 그의 로컬 이웃(즉, 이웃하는 샘플들)에 기초하여 샘플을 필터링하는 것에 의존하지만, 매우 상이한 값들을 가지는 샘플들보다 필터링될 샘플과 유사한 값들을 가지는 샘플들에 더 많은 주의(또는 가중치)를 부여한다. 이웃 값들에 대한 가중은 종종 비선형 함수들(즉, 비선형 매핑들)을 사용하여 수행된다. 이러한 종류의 비선형 필터들은 종종 선형 필터들보다 더 복잡하며, 그들의 파라미터들을 최적화하는 것이 어려울 수 있고/있거나, 비선형 필터들은 새로운 ALF 설계에서 그러한 종류의 필터들을 사용하기를 원하는 경우 선형 필터들보다 적은 유연성을 가질 수 있다.

[0104] 본 발명의 실시예에 따르면, VTM-3.0(또는 전술한 실시예들 또는 그 변형례들 중 임의의 것)의 ALF는 ALF의 병렬화 가능한 설계를 유지하는 연산들을 포함하는 상대적으로 낮은 복잡성을 갖는 비선형성을 도입하는 것에 의해 수정된다.

[0105] 수학식 1을 단순화하기 위해 실수를 사용하여 수학식 1을 재구성하는 것에 의해 VTM-3.0에서의 ALF를 살펴보면, (고정 소수점 표현 및 정수 반올림에 관련된 연산들의 제거)가 이루어져 있다:

수학식 4

[0106] $$O(x,y) = \sum_{(i,j)} w(i,j).I(x+i,y+j)$$

[0107] 이 ALF의 경우, 조건

수학식 5

[0108] $$\sum_{(i,j)} w(i,j) = 1$$

[0109] 이 충족된다. 이는 ALF의 모든 필터 계수들의 합이 1임을 의미한다. 이어서 수학식 4가 다음과 같이 재구성될 수 있음을 알 수 있다:

수학식 6

[0110] $$O(x,y) = I(x,y) + \sum_{(i,j)\neq(0,0)} w(i,j).\big(I(x+i,y+j) - I(x,y)\big)$$

[0111] 출력 샘플 O(x, y)는 그러면 (동일한 위치에 있는) 입력 샘플 I(x, y)를 필터 계수 벡터와 로컬 구배 벡터(입력 샘플의 이웃하는 샘플들과 입력 샘플 자체 사이의 차이들로서 계산되는 로컬 구배들의 벡터) 사이의 스칼라 곱에 가산한 결과이다. 환언하면, 출력 샘플(즉, 필터링된 샘플 값)은 입력 샘플을 필터 계수들과 로컬 구배들의 선형 결합에 가산한 결과이다.

[0112] 필터에 대해 이러한 통상적인 선형 수식화(linear formulation)를 사용하는 대신에, 본 발명의 실시예에 따르면, ALF 필터링 프로세스는 ALF 필터들에 비선형성을 도입하도록 수정된다. 이러한 비선형성은 오프셋 (i, j)에서의 로컬 구배 d를 제1 파라미터로서 취하고 제2 파라미터 b = k(i, j)에 따라 그의 값이 변하는 다변량 함수 K(d, b)를 사용하여 달성된다. 수학식 6에서의 스칼라 곱에서 오프셋들 (i, j)에 대한 로컬 구배들을 사용하는 대신에, 스칼라 곱에서 K(d, b)가 사용되어, 로컬 구배에 따라 비선형 방식으로 변하는 출력 샘플 O(x, y)를 결과하며:

수학식 7

[0113] $$O(x,y) = I(x,y) + \sum_{(i,j)\neq(0,0)} w(i,j).K\big(I(x+i,y+j) - I(x,y), k(i,j)\big)$$

[0114] 여기서 K(d, b)는 그의 제1 파라미터/변수로서 d = I(x + i, y + j) -I(x, y)(위치 (x + i, y + j)에 있는 이웃하는 샘플 값과 위치 (x, y)에 있는 현재 샘플 값 사이의 차이로서 계산되는 오프셋 (i, j)에서의 로컬 구배)를 취하고, 제2 파라미터/변수로서 b = k(i, j)(추가 필터링 파라미터)를 취하는 함수이다. 추가 필터링 파라미터 k(i, j)는 w(i, j)와 동시에 결정된다. 구현에서, k(i, j) 및 w(i, j)의 값들이 (예를 들어, 왜곡을 최소화하기 위해) 필터링 및 시그널링 프로세스를 최적화하도록 결정된다. 이 최적화 프로세스는 ALF를 사용하고 있는 인코더에 의해 수행된다. 그러한 최적화 프로세스의 예는 설명에서 나중에 제공될 것이다.

[0115] 따라서 본 발명의 실시예들에 따르면, 입력 샘플 값이 입력 샘플 자체와 입력 샘플의 이웃들인 입력 샘플들의 적응 비선형 변환의 선형 결합을 사용하여 필터링된다. 적응 비선형 변환은 필터링되고 있는 입력 샘플의 위치와 관련하여 이웃하는 입력 샘플들의 상대 위치들에 의존한다.

[0116] 그러한 수정된 ALF를 사용하면, K(d, b)가 모든 d에 대해 a와 d를 곱한 것과 동일하도록, b가 존재하고 0이 아닌 a가 존재한다는 조건을 K가 충족시킬 때의 특정 K에 대해, 선형 필터링을 달성하는 것이 여전히 가능하다, 즉:

수학식 8

[0117] $$K(d,b) = \alpha d, \forall\, d \text{ 이도록 } \exists b,\ \exists \alpha \neq 0$$

[0118] 따라서, b의 일부 값들에 대해 선형 함수(즉, 선형 사상(linear mapping))처럼 거동하는 함수 K(d, b)(즉, 함수 K(d, b)가 조건(수학식 8)을 충족시킴)를 선택하는 것은, 실시예의 수정된 ALF 필터링 프로세스가 여전히 적어도 표준 선형 ALF를 사용하는 것만큼 효율적일 수 있도록(즉, 최악의 경우에, 수정된 ALF가 VTM-3.0에서의 ALF와 동일한 레벨의 효율을 달성할 수 있도록) 보장한다. 예를 들어, 클리핑 함수가 사용될 때, 파라미터 b는 클리핑 함수를 선형 함수처럼 거동하도록 만들기 위해 최대 가능 정수 값으로 설정될 수 있다(이상적으로는 무한대로 설정되어야 하지만, 제한된 정수 정밀도 스킴에서 최대 가능 정수 값을 사용하여 동일한 효과를 달성할 수 있다).

[0119] 본 발명의 일부 실시예들에서, ALF 필터링이 대안적인 수식화/필터링 수식을 사용한다는 점에 유의한다. 로컬 구배와 추가 파라미터/변수를 취하는 다변량 함수 K(d, b)는 3개의 파라미터/변수, 즉 이웃하는 샘플 값, 필터

링될 샘플 값, 및 추가 파라미터/변수를 취하는 다른 다변량 함수로 대체된다. 이는 수학식 4와 유사한 표기법을 사용하고 조건(수학식 5)을 충족시키면서 수학식 7을 사용하여 재구성되며:

수학식 9

[0120] $$O(x,y) = w(0,0).I(x,y) + \sum_{(i,j)\neq(0,0)} w(i,j).K'(I(x+i,y+j),I(x,y),k(i,j))$$

[0121] 여기서 K'(n, c, b)는, 그의 파라미터들/변수들로서, 이웃하는 샘플 값(n), 필터링될 현재 샘플 값(c) 및 추가 필터링 파라미터/변수(b)를 취하는 함수이다.

[0122] 본 발명의 실시예에 따르면, 크로마 성분들(304) 및 루마 성분(407) 둘 모두에 대한 필터링 단계들이 (수학식 1을 참조하여 기술된 바와 같이) 정수 산술을 사용하여 구현되는 (바람직하게는 수학식 7을 사용하지만 또한 대안적으로 수학식 9를 사용하는) 비선형 수식화를 갖는 이 필터링을 사용하도록 수정된다.

[0123] 예를 들어, 임의의 선택이 복잡도에 기초하는 대안적인 실시예에 따르면, 2개의 필터링 단계(304 또는 407) 중 하나만이 비선형 수식화를 갖는 필터를 사용하도록 수정된다.

[0124] 일부 실시예들에 따르면, 파라미터/변수 b = k(i, j)는 하나 초과의 차원을 가지며, 예를 들어, 이는, 예를 들어, (i, j)에 따라 달라질 수 있는 S_b 및 D_b 값들을 갖는 수학식 10을 사용하는 $K(d, b = (S_b, D_b)) = f(d, S_b, D_b)$일 수 있다:

수학식 10

[0125] $$f(d,S_b,D_b) = \begin{cases} \min\left(d, \max\left(0, S_b - \left\lfloor \frac{d}{2^{D_b - \lfloor \log_2 S_b \rfloor}} \right\rfloor\right)\right), d \geq 0 \\ \max\left(d, \min\left(0, \left\lceil \frac{-d}{2^{D_b - \lfloor \log_2 S_b \rfloor}} \right\rceil - S_b\right)\right), d < 0 \end{cases}$$

[0126] 하나 초과의 차원을 갖는 파라미터들/변수들/함수들을 사용하는 것은 종종 필터링을 개선시키는 것을 가능하게 하지만, 필터 파라미터들을 시그널링하는 데 더 많은 비용이 든다. 추가적으로, 이는 인코더에서 더 높은 차원의 공간에서 필터 파라미터들을 최적화할 때 더 많은 복잡도를 도입하며, 디코더에서 함수를 계산하는 것이 종종 더 복잡하다(더 많은 필터 파라미터들은 종종 이들을 사용하기 위한 더 많은 연산들을 암시한다). 실시예에서, b는 단일 차원을 가지며, 이는 많은 ALF 응용들에 대해 양호한 절충을 달성할 수 있다.

[0127] 일부 실시예들에 따르면, 함수 K는 또한 상이한 오프셋 (i, j)에 대한 하나 초과의 상이한 함수들을 포함하는 함수일 수 있다. 일부 실시예들에서, 필터 형상/패턴에서의 함수들의 구성/배열은 인코더 및 디코더 둘 모두에서 미리 정의된다. 대안적으로, 함수들의 구성이 미리 정의된 구성들의 세트로부터 선정/선택되고, 선정된 구성에 대한 인덱스가 전송/시그널링된다. 다른 실시예들에서, 함수는 (중심 위치를 제외한) 필터 형상/패턴에서의 각각의 계수 인덱스에 대한 미리 정의된 함수들의 세트로부터 선택되고, 그의 인덱스가 전송/시그널링된다. 그러한 실시예들의 변형례에서, 각각의 함수는 단일 변수 함수이고(즉, K 함수는 더 이상 추가 파라미터/변수 b를 갖지 않으며), 따라서 파라미터 b를 시그널링하는 대신에, 함수 인덱스가 시그널링된다. 그러한 변형례는 다변량 수식화, 예를 들어, K(d, b) = K_b(d) 수식화를 사용하는 것으로 간주될 수 있다. 예를 들어, b는 함수들의 세트로부터 단일 변수 함수 K_b를 선택하기 위한 정수 인덱스이다.

[0128] 일부 실시예들에 따르면, 함수 K는 루마 필터(들) 및 크로마 필터(들)에 대해 동일하지 않다. 일부 실시예들에 따르면, 각각의 루마 필터에 대해 함수 K가 동일하지 않다. 이러한 실시예들 중 일부에서, 각각의 필터에 대해, 선택된 함수의 인덱스가 슬라이스 헤더에서 제공된다.

[0129] 본 발명의 실시예에 따르면, K는 (너무 많은 디코딩 복잡도를 도입하지 않도록) 계산하기 간단하도록 선택된다. 예를 들어, 그러한 실시예에서, K는 단순히 다음과 같은 클리핑 함수이다:

수학식 11

[0130] $$K(d,b) = \max(-b, \min(b,d))$$

[0131] 또는 등가적으로:

수학식 12

[0132] $$K(d,b) = \min(b, \max(-b,d))$$

[0133] 아래에서 기술되는 대안적인 함수들 중 일부와 달리, 높은 로컬 구배 값들에 대해 클리핑 함수가 사라지지 않는다(환언하면, x가 무한대에 가까워짐에 따라 클리핑 함수 f(x)가 0으로 수렴하지 않는다). 그러나 그러한 간단한 함수를 사용한 압축 결과가 종종 더 복잡한 함수들을 사용하는 것만큼 효율적이며 심지어 그보다 나을 수 있다는 것이 실험적으로 관찰되었다. 사실, 높은 로컬 구배 값들에 대해 클리핑 함수가 사라지지 않고, 이들을 단순히 클리핑하는 것은 계속하여 급격한 에지 전환을 고려하면서 해당 급격한 에지 전환 영역들 주변에 존재하는 높은 분산의 영향을 제한하는 것을 가능하게 할 수 있다. 아티팩트들이 일반적으로 급격한 전환들 주변에서 더 강하기 때문에, 이것이 관심 대상이다.

[0134] 수학식 9의 필터링 수식화를 사용하는 이 클리핑 함수 K의 등가 함수 K'은 다음과 같다.

수학식 13

[0135] $$K'(n,c,b) = \max(c-b, \min(c+b, n))$$

[0136] 또는 등가적으로:

[0137] $$K'(n,c,b) = \min(c+b, \max(c-b, n))$$

[0138] b가 최대 가능 샘플 값(예를 들어, 2의 이미지 비트 깊이 거듭제곱) 이상이면, 클리핑 함수는 수학식 8을 충족시킨다.

[0139] 이하의 설명에서, '클리핑 범위' 또는 '클리핑 파라미터'는 K 또는 K'의 파라미터/변수 b를 지칭하는 데 사용된다. 그러한 용어들이 비선형 함수 파라미터들을 지칭하기 위한 일반 용어로 간주될 수 있음이 이해된다. 유사하게, '클리핑' 또는 '클리핑 함수'는 위에서 기술된 K 또는 K' 또는 그의 기능적으로 등가 함수를 지칭하기 위해 사용될 수 있다.

[0140] 대안적인 실시예에서, K는 다른 비선형 함수일 수 있다. 변형례에서, K는 $f(d,p) = \begin{cases} 0 & if\ |d| > p \\ d & else \end{cases}$에 대응하는 반대칭 함수(anti-symmetric function), 예를 들어, 하나의 톱니 주기(sawtooth period)이거나; 또는 $D_d = \lfloor \log_2 S_d \rfloor$일 때의 수학식 10의 특별한 경우에 대응하는 일부 삼각 주기들이거나; 또는 양방향 필터와 유사한 방식으로 가우시안 커널을 사용할 수도 있다: 예를 들어, $K(d,b) = de^{-\frac{d^2}{b}}$. 예를 들어, K는 상기 변형례의 반대칭 함수들과 같이 특정 임계 값 초과(또는 미만)에서 0으로 설정되는 임의의 함수일 수 있다.

[0141] 실시예에 따르면, ALF 필터링을 수행하는 데 필요한 샘플 라인 버퍼들의 수를 감소시키기 위해(즉, 예를 들면, 디코더에서 필터링을 수행할 때 메모리에서 프로세싱/액세스/유지될 필요가 있는 입력 이미지 성분의 샘플들의

수를 감소시키기 위해) 선형 ALF 대신 비선형 ALF가 사용된다.

[0142] 변형례에 따르면, 트레이드오프/절충으로서, 루마 필터들에 대한 필터 형상/패턴의 크기가 7x7 다이아몬드 형상으로부터 더 작은 필터 형상/패턴으로 감소된다. 예를 들어, 루마 필터(들)에 대해 5x5 다이아몬드 형상이 사용된다(예를 들면, 도 3b에서의 크로마 필터와 동일한 형상이지만 여전히 전치 인덱스 변형들을 사용한다). 이는 여전히 7x7 다이아몬드 형상 루마 필터(들)(예를 들면, 도 4b에 도시된 것)로 그러나 감소된 수의 프로세싱/액세스/저장할 샘플들(예를 들면, 감소된 수의 샘플 라인 버퍼들) 및 또한 감소된 계산 복잡도로 선형 전용 ALF와 유사한 코딩 이득을 달성할 수 있다: 즉, ALF 필터를 프로세싱하는 데 필요한 곱셈들의 수가 각각의 필터링된 입력 샘플당 6씩 감소되면서 양호한 코딩 이득을 달성한다.

[0143] 변형례에 따르면, 샘플에 대해 ALF 필터링을 수행할 때, 선형 ALF에 사용되었을 모든 이웃하는 샘플들에 기초하여 비선형 ALF가 사용된다. 다른 변형례에 따르면, 샘플에 대해 ALF 필터링을 수행할 때, 선형 ALF에 대해 사용되었을 이웃하는 샘플들 중 일부만이 비선형 ALF에 대해 사용되며, 나머지 이웃하는 샘플들은 선형 ALF에 대해 사용된다. 또 다른 변형례에 따르면, 샘플에 대해 ALF 필터링을 수행할 때, 선형 ALF에 대해 사용되었을 이웃하는 샘플들 중 일부만이 비선형 ALF에 대해 사용되며, 나머지 이웃하는 샘플들은 선형 ALF에 대해 사용되지 않는다.

[0144] 변형례들에 따르면, 이러한 필터 형상들/패턴들에서 대칭성을 활용하는 것에 의해, 선형 및/또는 비선형 ALF 필터 구현을 단순화하기 위해 수학식 2와 동일한 표기법을 사용하여, 수학식 4에서의 선형 함수는 다음과 같이 재구성될 수 있으며:

수학식 14

[0145] $$O_n = w_c . I_n^c + \sum_{i=0}^{i<c} w_i . \left(I_n^{i^-} + I_n^{i^+} \right)$$

[0146] 이는, ALF의 경우, 다음 조건을 충족시킨다:

수학식 15

[0147] $$w_c + 2 . \sum_{i=0}^{i<c} w_i = 1$$

[0148] 수학식 6에서의 선형 함수가 또한 다음과 같이 재구성될 수 있으며:

수학식 16

[0149] $$O_n = I_n^c + \sum_{i=0}^{i<c} w_i . \left(I_n^{i^-} + I_n^{i^+} - 2 . I_n^c \right)$$

[0150] 수학식 7에서의 비선형 함수는 다음과 같이 되고:

수학식 17

[0151] $$O_n = I_n^c + \sum_{i=0}^{i<c} w_i . \left(K\left(I_n^{i^-} - I_n^c, k_i \right) + K\left(I_n^{i^+} - I_n^c, k_i \right) \right)$$

[0152] 여기서 k_i는 필터 계수 w_i와 연관된 필터 클리핑 파라미터이다.

[0153] 마지막으로, 수학식 9의 비선형 함수는 다음과 같이 재구성될 수 있다:

수학식 18

[0154] $$O_n = w_c . I_n^c + \sum_{i=0}^{i<c} w_i . \left(K'(I_n^{i-}, I_n^c, k_i) + K'(I_n^{i+}, I_n^c, k_i) \right)$$

[0155] 이 실시예의 변형례에 따르면, 함수 K' 또는 함수 K는 클리핑 함수이다.

[0156] 실시예에 따르면, 수학식 17에서 함수 K를 프로세싱하기 위한 계산들의 수와 비교하여 비선형 함수를 프로세싱하는 데 수반되는 계산들의 수를 감소시키기 위해, 그 대신에 적어도 2개의 이웃 차이의 합에 대해 비선형성이 도입된다(즉, 2개 이상의 로컬 구배의 합을 그의 변수들로서 갖는 비선형 함수가 사용될 수 있다):

수학식 19

[0157] $$O_n = I_n^c + \sum_{i=0}^{i<c} w_i . \left(K\left(I_n^{i-} + I_n^{i+} - 2 . I_n^c, k_i\right) \right)$$

[0158] 수학식 19가 항상 수학식 17과 등가인 것은 아니며, 수학식 19를 사용하는 필터는 덜 효율적일 수 있지만 계산 복잡도를 감소시킨다. 변형례에 따르면, 예를 들어, K가 클리핑 함수일 때, 시그널링된/인코딩된 클리핑 파라미터들/값들의 수는 수학식 17의 것과 변함이 없다.

[0159] 다른 변형례에 따르면 감소된 복잡도를 갖는 수학식 18에 기초하여 유사하게 도출된 수학식이 사용될 수 있으며, 함수 K'은 2개 이상의 이웃 차이 값(로컬 구배들)의 합을 그의 변수들로서 갖는다는 것이 이해된다:

수학식 20

[0160] $$O_n = w_c . I_n^c + \sum_{i=0}^{i<c} w_i . \left(K'\left(I_n^{i-} + I_n^{i+}, 2 . I_n^c, k_i\right) \right)$$

[0161] 도 7은, VTM-3.0에서의 ALF 내의 선형 필터에 대한 대체물일 수 있는(또는 ALF 내의 선형 필터 이외의 추가 비선형 필터일 수 있는), 본 발명의 실시예에 따른 비선형 필터의 블록 다이어그램를 제공한다. 이는 수학식 11의 클리핑 함수를 사용하는, 정수 고정 소수점 산술을 갖는 수학식 7의 구현에 대응한다. 'C'(701)는 I(x, y)에 대응하고, 'S0'(702) 내지 'Sn'(703)은 각각의 (i, j) ≠ (0, 0)에 대한 I(x + i, y + j)에 대응하며, 'k0'(704) 내지 'kn'(705)은 각각의 (i, j) ≠ (0, 0)에 대한 k(i, j)에 대응하고, 'w0'(706) 내지 'wn'(707)은 각각의 (i, j) ≠ (0, 0)에 대한 w(i, j)에 대응하며, 'O'(708)는 O(x, y)에 대응한다.

[0162] 도 5는, 전술한 실시예들/변형례들 및 이들과 관련된 실시예들/변형례들의 비선형 함수(및 그의 파라미터들)를 구현하는 데 사용될 수 있는 신택스 요소들의 예를 제공하는, 본 발명의 실시예에 따른 수정된 신택스 요소들의 개관을 갖는 플로차트이다. 이 실시예에서, 수학식 7(또는 수학식 9)의 각각의 필터에 대한 인코딩된 필터 계수는 그 자신의 클리핑 범위와 연관되어 있으며, 따라서 k(i, j)는 오프셋 (i, j)에 따라 변하는 상이한 값들을 가질 수 있다. 대부분의 신택스 요소들은 VTM-3.0에서 이미 사용되고 도 2를 참조하여 설명된 것과 동일하다: 도 5에서의 501, 502, 503, 504, 505, 506, 507, 508, 509, 510은 도 2에서의 201, 202, 203, 204, 205, 206, 207, 208, 209, 210과 동일한 시그널링 및 시맨틱스를 갖는다. 새로운 신택스 요소들은 각각의 루마 필터에 대한 클리핑 파라미터들(511), 루마 필터들에 대한 모든 필터 계수들(506), 및 루마 필터들 각각에 대한 모든 클리핑 파라미터(511)이며, 이들은, 예를 들어, 슬라이스 헤더에서 시그널링될 수 있다. 이 시그널링 이후에 크로마 필터에 대한 모든 필터 계수들(507) 및 크로마 필터에 대한 모든 클리핑 파라미터들(512)의 시그널링이 뒤따른다.

[0163] 실시예에 따르면, 임의의 필터에 대해, 시그널링된 클리핑 파라미터들의 수는 시그널링된 필터 계수들의 수와 동일하다. 오프셋 (i, j)에 대한 클리핑 파라미터들은 동일한 위치 (x + i, y + j)에 있는 필터 계수들과 동일한 방식으로 획득된다. 크로마의 경우, 이들은 도 3a의 단계(303)에서의 필터 계수 도출 프로세스에 대해 기술된 것과 동일한 방식으로 (그러나 크로마 클리핑 파라미터들을 사용하여) 프로세싱되고; 루마의 경우, 이들은

도 4a의 단계(406)에서의 필터 계수 도출 프로세스에 대해 기술된 것과 동일한 방식으로 (그러나 루마 필터 클리핑 파라미터들을 사용하여) 프로세싱된다.

[0164] 대안적인 실시예에서, 필터당 단지 하나의 클리핑 파라미터가 있으며, 이 클리핑 파라미터는 모든 필터 (i, j) ≠ (0, 0) 위치들에 대해 사용된다.

[0165] 대안적인 실시예에서, 클리핑 파라미터들의 수는 필터 계수들의 수보다 낮다. 이 실시예의 변형례에서, 이러한 클리핑 파라미터들은 오프셋 (i, j)을 갖는 필터 위치들의 (미리 정의되거나 결정될 수 있는) 서브세트에 사용된다. 다른 필터 위치들에 대해, 통상적인 선형 필터링이 수행되거나(또는 환언하면, 해당 다른 필터 위치들에서, K가 항등 함수인 것으로 간주됨) 또는 대안적으로 미리 정의된 클리핑 파라미터 값들이 사용된다.

[0166] 그러한 실시예의 변형례들은 도 16a 및 도 16b를 참조하여 기술된다. 이러한 변형례들에서, 클리핑 함수(들)는 필터 위치들의 서브세트에 적용된다, 즉 필터링 프로세스 동안 이웃하는 샘플 값들/위치들의 서브세트만이 클리핑되도록(예를 들면, 비선형 함수가 해당 서브세트로부터의 샘플 값들에만 적용되도록) 클리핑 패턴/형상/서포트/마스크는 필터 형상/패턴에서의 필터 위치들의 서브세트만을 포함한다. 클리핑 연산들은 계산 비용이 많이 들 수 있기 때문에, 이러한 방식으로 필터링에 관여된 클리핑 연산들의 수를 감소시키는 것은 필터링 프로세스의 복잡도와 계산 비용을 감소시킬 수 있다.

[0167] 도 16a는 감소된 수의 클리핑되는/클리핑 위치들을 갖는 7x7 다이아몬드 필터 형상들을 제공하며, 7x7 필터 형상으로, 예를 들어, 루마에 대한 ALF로 필터링을 수행할 때 필요한 클리핑 연산들의 수를 감소시키기 위한 클리핑되는/클리핑 위치들의 세 가지 가능한 배열/구성을 예시한다. 도 16a에 도시된 클리핑 패턴들을 사용하는 것에 의해, 필터링을 수행하기 위한 클리핑 연산들의 수가 감소되는데 그 이유는 클리핑 연산들이 모든 필터 (입력) 위치들에 적용되지 않기 때문이다, 즉, 필터 위치들의 (미리 정의되거나 시그널링되거나 추론될 수 있는) 서브세트만이 클리핑 연산들에 사용된다(즉, 비선형 함수에 대해 사용된다). 도 16a에서, 클리핑되는 샘플 위치들은 클리핑 패턴들(1601, 1602 및 1603)에서 'X'로 표시되어 있다. 클리핑 패턴들(1601 및 1602)은 클리핑 연산들의 수를 2/3만큼 감소시키는 반면, 클리핑 패턴(1603)은 이 수를 절반으로 감소시킨다. 이러한 클리핑 패턴들(1601, 1602 및 1603)은 7x7 다이아몬드 필터 형상을 사용할 때 출력의 정확도와 필터링 프로세스에 관여된 복잡도 사이의 양호한 절충에 도달한다. 클리핑 패턴들(1601, 1602 및 1603)은 배향 기반 ALF(즉, 로컬 콘텐츠 배향 및 활동 레벨에 기초한 분류를 위해 구축된 필터, 예를 들면, VTM-3.0에서의 루마 필터(들))에 기초하는 비선형 ALF에 대해 잘 작동한다. 그러나 분류가 사용되지 않는 경우, 패턴들(1602 또는 1603)을 사용하는 것이 바람직하다.

[0168] 다른 변형례들에 따르면, 더 많은 또는 더 적은 클리핑되는 위치들이 사용될 수 있음이 이해된다. 예를 들어, 클리핑 패턴(1601)(즉, 중심 위치를 제외한, 위에서 아래로 그리고 좌에서 우로의 십자형) 대신에 (중심 위치를 제외한) 전체 열과 전체 행이 클리핑될 수 있거나, 또는 클리핑 패턴(1602) 대신에 외부 에지 상의 위치들(즉, 더 큰 다이아몬드/평행사변형 클리핑 패턴)이 클리핑될 수 있다. 클리핑되는 위치들의 다른 변형례들에서, 클리핑 패턴/형상/서포트/마스크는 (중심 위치를 제외한) 대각 십자형 "X", (중심 위치를 제외한) 수직 세그먼트 "|", (중심 위치를 제외한) 수평 세그먼트 "-", (중심 위치를 제외한) 좌측 상단에서 우측 하단으로의 대각 세그먼트 "\", (중심 위치를 제외한) 우측 상단에서 좌측 하단으로의 대각 세그먼트 "/", 또는 전술한 클리핑 패턴들/형상들/서포트들/마스크들의 임의의 조합을 형성한다.

[0169] 도 16b는 감소된 수의 클리핑되는/클리핑 위치들을 갖는 5x5 다이아몬드 필터 형상들을 제공하며, 5x5 필터 형상으로 필터링을 수행할 때 필요한 클리핑 연산들의 수를 감소시키기 위한 클리핑되는/클리핑 위치들의 두 가지 가능한 배열/구성을 예시한다. 도 16a에서와 같이, 도 16b에 도시된 클리핑 패턴, 즉 클리핑 패턴들(1604 및 1605)을 사용하는 것에 의해, 5x5 다이아몬드 필터 형상으로 필터링할 때, 이 필터링 프로세스 동안 관여되는 클리핑 연산들의 수를 감소시킬 수 있다. 클리핑 패턴들(1604 및 1605)은 배향 기반 ALF(즉, 로컬 콘텐츠 배향 및 활동 레벨에 기초한 분류를 위해 구축된 필터, 예를 들면, VTM-3.0에서의 루마 필터(들))에 기초하는 비선형 ALF에 대해 잘 작동한다. 그러나 분류가 사용되지 않는 경우, 패턴(1605)을 사용하는 것이 바람직하다.

[0170] 수평 및 수직 방향 이웃들이 (더 큰 유클리드 거리로 인해) 대각 이웃들보다 통계적으로 더 신뢰할 수 있기 때문에, 클리핑 패턴들(1602 및 1605)은 배향에 기초하는 분류를 사용하지 않는 필터들에 대해 더 잘 작동한다(예를 들면, VTM-3.0에서 수평, 수직 또는 대각 배향에서의 분류에 의존하는 루마 필터들과 달리 크로마 필터에 더 양호하다). 따라서, 더 신뢰할 수 있는 수평 및 수직 방향의 이웃하는 샘플들의 필터 위치들보다는 덜 신뢰할 수 있는 대각의 이웃하는 샘플들의 필터 위치들에서만 클리핑 연산들을 사용하는 것이 더 유리하다.

[0171] 배향 기반 필터들(즉, 배향 기반 분류를 위해 획득된 필터들)의 경우, 클리핑 연산들은 일반적으로 필터 배향들에 수직인 필터 위치들에 있는 샘플들에 적용된다. 예를 들어, 1601에서의 클리핑 패턴은 대각 배향을 갖는 클래스들에 대한 필터들보다 수평 또는 수직 배향을 갖는 클래스들에 대한 필터들에 대해 더 잘 작동할 것이다. 클리핑 패턴(1602)은 4개의 클래스 배향에 대해서도 대체로 작동될 것이지만, 수평 및 수직 배향들에 대해서는 패턴(1601)보다 덜 효율적일 것이다. 따라서 1601 및 1602에서의 클리핑 패턴은 7x7 필터 패턴을 사용하는 루마 필터들에 대해 평균적으로 거의 동등하게 작동되고 1604 및 1605에서의 클리핑 패턴은 5x5 필터 패턴을 사용하는 루마 필터들에 대해 평균적으로 거의 동등하게 작동될 것이다.

[0172] 변형례에 따르면, 배향 기반 필터를 개선시키기 위해, 클리핑 패턴이 필터 배향에 기초하여 미리 정의된 패턴들 중에서 선택된다. 예를 들어, 클리핑 패턴(1601)은 수평 또는 수직 필터들에 대해 사용되는 반면 클리핑 패턴(1602)은 대각 필터들에 사용된다. 또 다른 변형례에서, 수직 필터에 대해 선택되는 클리핑 패턴은 수평 필터에 대해 선택되는 클리핑 패턴과 상이하고(예를 들어, 각각이 다른 것이 90° 회전된 버전임), 2개의 대각 필터 배향에 대한 클리핑 패턴이 또한 상이하다(예를 들어, 각각이 다른 것의 90° 회전된 버전임). 또 다른 변형례에서, 클리핑 패턴은 각각의 4x4 샘플 값 블록에 대한 전치 인덱스(tranposeIdx)에 따라 선택된다.

[0173] 변형례에 따르면, 각각의 필터에 대해 하나의 미리 결정된 클리핑 패턴이 있다. 다른 변형례에 따르면, 사용 가능한 복수(예를 들면, 미리 결정된 수)의 클리핑 패턴들이 있고, 각각의 필터 인덱스에 대해, 선택된 클리핑 패턴들에 대한 (클리핑 패턴) 인덱스가 ALF 파라미터들과 함께, 예를 들어, APS 또는 타일 그룹 헤더에, 인코딩/시그널링된다. 또 다른 변형례에 따르면, 클리핑 패턴(들)이 ALF 파라미터들과 함께 시그널링된다. 예를 들어, 클리핑 패턴의 시그널링은, 필터(패턴) 위치 인덱스(예를 들면, 인코딩된/디코딩된 필터 계수 인덱스와 동등함)당 하나의 플래그씩, 플래그들의 시퀀스를 인코딩/디코딩하는 것에 의해 수행될 수 있으며, 각각의 플래그는 대응하는 인덱스에 대해 클리핑이 적용되는지 또는 실제로 적용되지 않는지를 나타낸다. 대안적으로, 클리핑 패턴의 시그널링은 클리핑되는 위치 인덱스들 자체를 인코딩/디코딩하는 것에 의해 또는 클리핑되지 않을 위치들의 필터(패턴) 위치 인덱스를 인코딩/디코딩하는 것(바람직하게는, 더 적은 수의 비트들이 시그널링되는 것을 필요로 하는 쪽)에 의해 수행될 수 있다.

[0174] 변형례에 따르면, (미리 결정되어 있는 또는 그렇지 않은) 클리핑 패턴이 사용될 때, 클리핑 연산이 적용되는 필터(패턴) 위치들에 대해서만 클리핑 파라미터들이 시그널링된다.

[0175] 일부 변형례들에서, SIMD(Single instruction, multiple data) 병렬 구현을 망가뜨리지 않기 위해, 클리핑이 모든 필터 위치들에 적용되지만, 클리핑 파라미터가 제공되지 않은 위치들에 대한 클리핑 파라미터는 선형 출력(예를 들면, 변형례에서, 항등 함수의 출력)을 갖는 것으로 간주된다.

[0176] 대안적인 실시예에서, 클리핑 파라미터들의 수는 필터 계수들의 수보다 낮은데 그 이유는 하나 이상의 클리핑 파라미터(들)가 공유되기 때문이다. 예를 들어, 도 3b 및 도 4b의 필터 형상들에서, 일부 계수 인덱스들은 동일한 클리핑 파라미터들을 공유할 수 있다. 변형례에 따르면, 클리핑 파라미터의 인덱스는 다수의 요소들을 포함하는 테이블로 표시되며, 개수는 (대칭성으로 인해) 형상 크기의 절반에서 1을 뺀 것(중심 계수 위치에 대한 클리핑 파라미터가 없기 때문임)과 동일하다. 이 테이블은 클리핑 파라미터 인덱스를 각각의 필터 계수와 연관시키는 것을 가능하게 한다. 변형례에 따르면, 이 테이블은 코덱에 의해 고정/미리 설정된다. 변형례에 따르면, 다수의 고정된/미리 설정된 테이블이 정의되고, 테이블에서 클리핑 파라미터를 식별하기 위한 인덱스는, 예를 들어, 슬라이스 헤더에서 시그널링된다. 대안적인 변형례에서, 그러한 테이블의 내용은, 예를 들어, 슬라이스 헤더에서 시그널링되고, 모든 루마 필터들에 의해 공유된다.

[0177] 실시예에서, 클리핑 파라미터들이 취할 수 있는 값들의 수는 (작은 품질 이점들과 대비하여 인코더 복잡도 및 인코딩 비용을 감소시키기 위해) 작도록 제한된다. 클리핑 파라미터에 대해 허가되는 값들은 정수 값 인덱스로, 바람직하게는 증가 또는 감소하는 순서로 인덱싱된다. 이어서, 이러한 인덱스들은 클리핑 파라미터들의 테이블의 각각의 요소에 매핑될 수 있다. 이어서, 클리핑 파라미터 값들을 시그널링하는 대신에, 테이블에 있는 관련 클리핑 파라미터 값의 인덱스(p)가 시그널링된다. p가 테이블에서의 필터 인덱스인 실시예에서, p에 대한 가능한 값들의 수를 감소시키기 위해 중간 테이블을 사용할 필요가 없다. 함수들의 테이블에서 이용 가능한 함수들의 수를 직접 감소시키고 따라서 그의 크기를 감소시키는 것이 가능하다.

[0178] 일 실시예에서, 클리핑 파라미터 값들은 2의 거듭제곱, 즉 2^p로 제한된다. 인코딩되는 것은 그러면 p이다. p의 최댓값은 입력 이미지의 비트 깊이 B_d이다(더 높은 값들은 동일한 결과들을 제공할 것이므로 필요하지 않다). 대안적인 실시예에서, p 대신에, 인코딩되는 것은 $B_d - p$이다. 다른 실시예에서, p의 범위는 p_{min}과 p_{max} 사이로

제한된다. 예를 들어, p_{min} = 3이고 p_{max} = B_d - 1이다. 이어서 p - p_{min} 또는 p_{max} - p가 시그널링될 수 있다.

[0179] 실시예에서, 크로마 필터에 대한 허가된/허용 가능한/이용 가능한 클리핑 파라미터 값들은 루마 필터들에 대한 것들과 동일하지 않다.

[0180] 실시예에 따르면, 가능한 클리핑 파라미터 값들을 시그널링할 때 이들의 수가 한정/제한될 수 있도록, 슬라이스에서의 클리핑 파라미터들에 대해 사용되는 테이블에서의 최소 인덱스 p_{min} 및 테이블에서의 최대 인덱스 p_{max}가 슬라이스 헤더에서 제공된다. 실시예에 따르면, p_{min} 및 p_{max}가 루마 및 크로마 필터들에 의해 공유된다. 대안적인 실시예에서, p_{min} 및 p_{max}가 루마에 대해서만 제공되며, 크로마 인덱스들은 제한되지 않는다. 다른 대안적인 실시예에서, 루마와 크로마에 대해 p_{min} 및 p_{max}가 슬라이스 헤더에서 제공된다.

[0181] 대안적인 실시예에 따르면, 루마 및 크로마 성분들 둘 모두에 대해 허가된 클리핑 파라미터 값들의 하나의 테이블이 슬라이스 헤더에서 시그널링되거나, 또는 대안적으로 2개의 테이블, 즉 루마에 대한 테이블과 크로마에 대한 테이블이 시그널링된다.

[0182] 본 발명의 실시예에서, 클리핑 파라미터들(511 및 512)은 슬라이스 헤더에서 시그널링되지 않는다. 그 대신에, 클리핑 파라미터들은 필터링되는 샘플을 시그널링하는 데 사용된 양자화 파라미터(QP)를 사용하여 그리고 필터의 클래스 인덱스에 기초하여 인코더/디코더에서 결정된다.

[0183] 대안적인 실시예에서, 클리핑 파라미터들(511 및 512)은 슬라이스 헤더에서 시그널링된다. 그러나 필터링 프로세스에서, 이들이 직접 사용되지 않는다. 그 대신에, 필터링되는 샘플을 시그널링하는 데 사용된 양자화 파라미터(QP)에 따라 클리핑 파라미터들이 스케일링된다.

[0184] 실시예에서, 각각의 필터에 대해 루마 필터에 대한 클리핑 파라미터들이 제공되지 않는다. 클리핑 파라미터들의 2개의 테이블만이 모든 루마 필터들에 의해 공유되고 슬라이스 헤더에서 시그널링된다. 하나의 테이블은 수평/수직 클래스들에 대한 것인 반면 다른 테이블은 대각 클래스들에 대한 것이다. 각각의 클래스의 클리핑 파라미터들은 클래스 배향에 따라, 도 4b의 계수 배열들을 사용하는 것 그리고 클래스의 활동 레벨로부터 결정되는 고정 값에 따라 클리핑 파라미터들을 스케일링하는 것(즉, 곱하는 것)에 의해, 해당 2개의 테이블로부터 도출된다. 스케일링은 4x4 블록들을 분류할 때 활동 레벨을 고려한다. 예를 들어, (예를 들면, 동질적인 영역들에서의) 낮은 활동 레벨들의 영역들과 비교할 때, (예를 들면, 에지들 근처에서의) 높은 활동 레벨의 영역에 대해, 더 높은 클리핑 값들이 사용될 수 있다.

[0185] 실시예에서, 슬라이스 헤더에서의 하나의 비트는 각각의 필터에 대한 각각의 필터 클리핑 인덱스에 대해 클리핑이 인에이블/디스에이블되어 있음을 시그널링한다. 클리핑 파라미터당 단지 하나의 허용된 클리핑 값이 있는 실시예에서, 비트가 클리핑이 활성임을 나타내면, 해당 클리핑 값이 사용된다(즉, 클리핑 파라미터 값에 대해 다른 어떤 것도 시그널링되지 않는다). 대안적으로, 하나 초과의 클리핑 값들이 허용될 때, 하나의 클리핑 인덱스에 대한 비트가 해당 필터 클리핑 인덱스에 대해 클리핑이 디스에이블되어 있음을 나타내는 경우, 클리핑 값이 시그널링되지 않는 반면, 다른 경우에는 클리핑 값이 시그널링된다.

[0186] 여기서 필터 클리핑 인덱스는 필터 형상들에서의 필터 계수 인덱스와 연관된 클리핑 파라미터의 인덱스에 대응한다.

[0187] 실시예에서, 허용된 클리핑 값들은 (예를 들어, INTRA, B 또는 P일 수 있는) 슬라이스 유형에 의존한다.

[0188] 실시예의 변형례들에서, 허용된 클리핑 값들은: B 또는 P 슬라이스에서의 루마에 대해 {6, 32, 181, 1024}이고; B 또는 P 슬라이스에서의 크로마에 대해 {4, 25, 161, 1024}이며; INTRA 슬라이스에서의 루마에 대해 {10, 102, 1024}이고/이거나 INTRA 슬라이스에서의 크로마에 대해 {4, 24, 1024}이다. 따라서 임의의 클리핑 파라미터는 (슬라이스 유형 및 필터링되는 성분에 따라) 해당 세트들 중 하나에 속하는 값을 취할 수 있다. 그리고, 각각의 클리핑 파라미터에 대해, 세트에 있는 해당 값의 인덱스가 각각의 필터에 대한 슬라이스 헤더에 인코딩된다.

[0189] 변형례에서, 허용된 클리핑 값들의 테이블은 다음과 같이 정의되며:

[0190] $$\left\{\text{round}\left(\left((M)^{\frac{1}{N}}\right)^{n}\right) \text{ for } n \in 1..N\right\}$$

[0191] 여기서 N은 테이블에 있는 클리핑 값들의 수(즉, 테이블의 크기)이고, M은 최대 클리핑 값(테이블에서의 마지막 엔트리임, 예를 들어, $M = 2^D$ 이거나 $M = 2^D - 1$이고 여기서 D는 테이블이 정의되는 성분의 샘플 비트 깊이임)이며, 여기서 'round'는 (예를 들어, 가장 가까운 정수로의) 반올림 연산자이다.

[0192] 변형례에서, 허용된 클리핑 값들의 테이블은 다음과 같이 정의되며:

[0193] $$\left\{A.\mathrm{round}\left(\left(\left(\frac{M}{A}\right)^{\frac{1}{N-1}}\right)^{n-1}\right)\ \text{for}\ n \in 1..N\right]\right\}$$

[0194] 여기서 N은 테이블에 있는 클리핑 값들의 수(즉, 테이블의 크기)이고, M은 최대 클리핑 값(테이블에서의 마지막 엔트리임, 예를 들어, $M = 2^D$ 이거나 $M = 2^D - 1$이고 여기서 D는 샘플 비트 깊이임)이며, A는 가장 작은 클리핑 값(테이블에서의 첫 번째 엔트리임)이고, 여기서 'round'는 (예를 들어, 가장 가까운 정수로의) 반올림 연산자이다.

[0195] 실시예에서, 각각의 필터 클리핑 인덱스에 대해, 허용된 클리핑 값들이 동일하지 않다.

[0196] 차이 값에 대해 클리핑 함수를 사용하는, 즉 이웃 샘플 값과 중심 값 사이의 차이에 대해 수학식 11 또는 수학식 12의 함수 K를 적용하는 실시예들의 변형례들에서, 클리핑의 출력에서의 비트들의 수가 감소되도록 최대 허용 클리핑 값이 정의된다. 이는 해당 변형례들에 따라 필터링을 수행하는 하드웨어 구현에 대해 처리될 필요가 있는 비트들의 수를 제한하는 것을 가능하게 한다. 따라서, 이는, 예를 들어, 칩 내의 논리 게이트들/트랜지스터들의 수를 감소시키는 것을 가능하게 한다. 곱셈이 많은 논리 게이트들을 필요로 하기 때문에 곱셈 연산자의 입력단에서의 비트들의 수를 감소시키는 것이 특히 관심 대상이다. 예를 들어, 최대 허용 클리핑 값을 2의 (샘플 비트 깊이 -1) 거듭제곱에서 1을 뺀 것으로(즉, 최대 클리핑을 2^(비트 깊이 - 1) - 1과 동일하게) 설정하는 것은 클리핑의 출력단에서 비트들의 최대 수를 1만큼 감소시키고 곱셈 연산자의 입력단에서 그렇게 하는 것을 가능하게 한다. 그리고 샘플 비트 깊이에서 2를 뺀 것을 사용하는 것(즉, 최대 클리핑이 2^(bit depth - 2) - 1과 동일한 것)은 비트들의 최대 수를 2만큼 감소시키는 것을 가능하게 한다.

[0197] APS를 사용하는 실시예에서, APS가 INTRA 및 INTER 슬라이스들/픽처들/타일 그룹들/이미지 부분들에 의해 공유되기 때문에, 클리핑 값 도출 동안 INTRA 슬라이스와 INTER 슬라이스(또는 INTRA 픽처/이미지 부분/타일 그룹과 INTER 픽처/이미지 부분/타일 그룹) 간의 구별이 가능하지 않을 수 있다. 그러면, 허용된 클리핑 값들의 미리 정의된 테이블이 더 이상 슬라이스/이미지 부분/타일 그룹 유형에 의존할 수 없다. 따라서, 하나의 기본 클리핑 값 테이블이 모든 슬라이스/픽처/타일 그룹/이미지 부분 유형들에 의해 공유된다. 변형례에서, 예를 들어, 복잡도 감소 고려사항들을 위해, 클리핑 값들이 기본 클리핑 값 테이블로부터의 클리핑 값들의 서브세트로 제한되는지 여부를 결정하는 것(및 이어서 디코더에 시그널링하는 것)은 그러면 인코더에 달려 있다.

[0198] 변형례에서, 사용 가능한 클리핑 값들의 클리핑 값 테이블은 ALF 파라미터들과 함께 APS에서 시그널링된다. 따라서 인코더(및 디코더)는 모든 클리핑 값들을 사용할 수 있고, 사용되는 클리핑 값들을 자체 클리핑 값 테이블을 사용하여 시그널링할 수 있다.

[0199] 변형례에서, 하나 초과의 기본 클리핑 값 테이블이 있으며, 예를 들면, 하나는 INTER 슬라이스들에 대해 사용되고 다른 하나는 INTRA 슬라이스들에 대해 사용된다. 사용되는 테이블을 식별하기 위한 정보, 예를 들면, 사용되는 테이블의 인덱스가 그러면 ALF 파라미터들과 함께 APS에서 제공된다. 변형례에서, 루마와 크로마에 대해 기본 클리핑 값 테이블들이 상이하다.

[0200] 실시예에서, 필터의 출력 값이 또한 입력 샘플 값 및 (클리핑 파라미터들의 테이블에 추가될 수 있는) 추가 클리핑 파라미터에 따라 클리핑된다.

[0201] 일 실시예에 따르면, ALF 활성 플래그 외에도, 비선형 ALF가 활성/사용됨/인에이블됨을 나타내기 위해, NLALF(NonLinear ALF) 활성 플래그가 시퀀스 레벨에, 예를 들어, SPS에, 또는 대안적으로 프레임 레벨에, 예를 들어, 픽처 헤더에 놓인다. 이 플래그가 활성인 경우(즉, 이 플래그가 비선형 ALF가 활성임을 나타내는 경우), ALF에 대한 신택스 요소들은 비선형 필터들에 대한 필터 파라미터들, 예를 들어, 도 5의 것을 포함한다. 플래그가 비활성인 경우(즉, 플래그가 비선형 ALF가 활성임을 나타내지 않는 경우), ALF에 대한 신택스 요소들은 선형 필터에 대한 필터 파라미터들, 예를 들어, 도 2의 것만을 포함한다.

[0202] 실시예에 따르면, 슬라이스 헤더에서 ALF 플래그가 활성인 경우, NLALF 활성 플래그가 슬라이스 헤더에 놓인다.

이 NLALF 플래그가 활성인 경우, 각각의 루마 필터에 대한 클리핑 파라미터들 신택스 요소(511) 및 크로마 필터에 대한 클리핑 파라미터들(512)이 슬라이스 헤더에 존재한다. 이 NLALF 플래그가 비활성인 경우, 신택스 요소들(511 및 512)이 슬라이스 헤더에 존재하지 않으며, ALF는 통상적인 선형 필터링을 사용한다.

[0203] 대안적인 실시예에 따르면, 각각의 성분 유형 루마 및 크로마에 대해 별도의 NLALF 활성 플래그가 제공되고, 이러한 플래그들 중 하나 또는 둘 모두가 특정 성분의 샘플에 대해 비선형 ALF가 사용될지 여부를 나타내는 데 사용된다(예를 들면, 이들이 시퀀스 레벨에서 또는 슬라이스 헤더에서와 같이, NLALF 활성 플래그와 동일한 위치에서 제공된다). 변형례에서, 비선형 ALF는 루마에 대해 사용되고 선형 ALF는 크로마에 대해 사용된다. 대안적인 변형례에서, 비선형 ALF는 크로마에 대해 사용되고 선형 ALF는 루마에 대해 사용된다.

[0204] 변형례에서, NLALF 루마 활성 플래그 및 NLALF 크로마 활성 플래그가 슬라이스 헤더에서 제공된다. NLALF 루마 활성 플래그는 비선형 적응 루프 필터가 루마 성분에 대해 사용되는지 여부 및 따라서 비선형 ALF의 클리핑 파라미터들이 각각의 루마 필터에 대해 시그널링되는지 여부를 나타낸다. NLALF 크로마 활성 플래그는 비선형 적응 루프 필터가 크로마 성분에 대해 사용되는지 여부 및 따라서 비선형 ALF의 클리핑 파라미터들이 크로마 필터에 대해 시그널링되는지 여부를 나타낸다.

[0205] 변형례에서, NLALF 크로마 활성 플래그가 시그널링되지 않지만 NLALF 루마 활성 플래그가 0과 동일한 경우 0과 같은 것으로 추론되며, 따라서 비선형 ALF는 크로마에 대해서만 사용될 수 없다.

[0206] 변형례에서, 비선형 ALF는 항상 루마에 대해 사용되고(기본적으로, 따라서 NLALF 루마 활성 플래그는 사용되지 않을 수 있거나 또는 대응하는 기본 값인 것으로 가정될 수 있음), NLALF 크로마 활성 플래그는 크로마에 대해 비선형 ALF 또는 선형 ALF가 사용되는지를 나타내는 데 사용된다. 다른 변형례에서, 선형 ALF는 항상 그렇듯이 루마에 대해 사용되고(기본적으로, 따라서 NLALF 루마 활성 플래그는 사용될 수 없거나 또는 대응하는 기본 값인 것으로 가정될 수 있음), NLALF 크로마 활성 플래그는 크로마에 대해 비선형 ALF 또는 선형 ALF가 사용되는지를 나타내는 데 사용된다. 변형례에서, NLALF 루마 활성 플래그는 루마에 대해 비선형 ALF 또는 선형 ALF가 사용되는지를 나타내는 데 사용되고, 비선형 ALF는 항상 크로마에 대해 사용된다(기본적으로, 따라서 NLALF 크로마 활성 플래그는 사용되지 않을 수 있거나 또는 대응하는 기본 값으로 가정될 수 있음). 변형례에서, NLALF 루마 활성 플래그는 루마에 대해 비선형 ALF 또는 선형 ALF가 사용되는지를 나타내는 데 사용되고, 선형 ALF는 항상 크로마에 대해 사용된다(기본적으로, 따라서 NLALF 크로마 활성 플래그는 사용되지 않을 수 있거나 또는 대응하는 기본 값으로 가정될 수 있음).

[0207] 일 실시예에서, NLALF 활성 플래그(또는 NLALF 루마 활성 플래그 또는 NLALF 크로마 활성 플래그)는 NLALF 플래그가 SPS에서 활성인 경우에만 슬라이스 헤더에 놓인다. NLALF 활성 플래그(또는 NLALF 루마 활성 플래그 또는 NLALF 크로마 활성 플래그)가 존재하지 않는 경우에, 이는 기본적으로 비활성으로 간주된다.

[0208] 일 실시예에서, NLALF 플래그는 SPS 및 슬라이스 헤더에서 ALF 플래그를 대체한다. 클리핑 함수가 허용된 클리핑 파라미터 값들에 대한 선형 출력을 가능하게 하는 실시예들를 제외하고는, 고전적인(선형 전용) ALF가 더 이상 사용될 수 없다.

[0209] 실시예에 따르면, 클리핑 파라미터들의 시그널링은 필터 계수들에 대해서와 유사한 방식으로 수행된다. 클리핑 파라미터들의 (지수-)골롬 인코딩에 대한 파라미터들이 먼저 시그널링되고: VLC 코드가 (지수-)골롬 코드들의 최소 차수를 시그널링하는 데 사용되고, 이어서 각각의 (지수-)골롬 인덱스(예를 들어, 루마 필터들에 대한 3개의 인덱스 및 크로마에 대한 2개의 인덱스)에 대해, 현재 인덱스 및 (최소 차수부터 시작하여) 다음 인덱스들에 대해 (지수-)골롬 차수가 증가되어야 하는지 여부를 시그널링하는 데 플래그가 사용된다.

[0210] 이어서, 각각의 (디스에이블되지 않은) 필터에 대한 클리핑 파라미터들이 (부호 있는 정수에 대한) (지수-)골롬 코드를 사용하여 시그널링되고, (지수-)골롬 차수는 (지수-)골롬 파라미터들을 저장하는 테이블로부터의 (예를 들어, 필터 계수들에 대해서와 동일하게, 고정된 테이블 내의) 계수 인덱스와 연관된 (지수-)골롬 인덱스를 사용하여 취해진다.

[0211] 대안적인 실시예에 따르면, 각각의 인코딩된 필터에 대해, (지수-)골롬 인코딩 파라미터 값은, 예를 들어, 슬라이스 헤더에서 시그널링되고, 각각의 클리핑 파라미터 값은 부호 없는 정수 (지수-)골롬 인코딩에 대한 제공된 (지수-)골롬 인코딩 파라미터 값을 사용하여 인코딩된다.

[0212] 실시예에 따르면, (지수-)골롬 인코딩 파라미터 값은 이용 가능한 (지수-)골롬 인코딩 파라미터 값들의 테이블에서의 인덱스로서 시그널링된다.

[0213] 실시예에 따르면, 플래그는, 예를 들어, 슬라이스 헤더에서 시그널링되고, 플래그는 루마 필터들에 대한 클리핑 파라미터들이 델타 모드를 사용하여 시그널링되는지 여부를 나타낸다. 델타 모드가 활성일 때, 첫 번째 필터 클리핑 파라미터들은 이전에 기술된 바와 같이 인코딩되지만, 각각의 후속 필터 클리핑 파라미터들은 후속 필터 클리핑 파라미터와 이전에 인코딩된 필터 클리핑 파라미터 간의 차이로서 인코딩된다. 그러한 경우에, 차이는 부호 있는 정수 (지수-)골롬 인코딩을 사용하여 인코딩된다.

[0214] 실시예에 따르면, 플래그는, 예를 들어, 슬라이스 헤더에서 시그널링되고, 플래그는 필터들에 대한 클리핑 파라미터가 델타 모드를 사용할 수 있는지 여부를 나타낸다. 실시예에 따르면, 델타 모드가 활성일 때, 각각의 필터에 대해 플래그가 시그널링되고, 플래그는 필터에 대한 클리핑 파라미터가 델타 모드를 사용하는지 여부를 나타낸다. 실시예에 따르면, 주어진 필터의 클리핑 파라미터들을 인코딩하기 위해 델타 모드가 활성일 때, 클리핑 파라미터들은 하나씩 인코딩되고, 첫 번째 것은 부호 없는(지수-)골롬 인코딩을 사용하고, 후속하는 것들은 클리핑 파라미터와 이전에 인코딩된 클리핑 파라미터의 값 사이의 차이의 부호 있는 (지수-)골롬 인코딩을 사용한다.

[0215] 대안적인 실시예에 따르면, 클리핑 파라미터가 취할 수 있는 값들의 수는 2개의 상이한 값으로 제한된다. 그러면 어느 클리핑 파라미터 값이 사용되어야 하는지를 시그널링하기 위해 단일 비트가 사용된다((지수-)골롬 파라미터들을 제공할 필요가 없다).

[0216] 실시예에 따르면, (예를 들면, 필터 계수가 0이거나 디코더 측에 있는 필터에 알려진 값과 동일한 경우) 불필요한 데이터의 시그널링이 최소화되도록 클리핑 파라미터의 조건부 시그널링이 수행된다. 변형례에서, 클리핑 파라미터들이 시그널링되기 전에, 필터 계수들이, 예를 들어, 슬라이스 헤더에서 시그널링되고, 0과 동일한 필터의 각각의 i 번째 필터 계수에 대해, 그의 대응하는 i 번째 클리핑 파라미터는 시그널링되지 않는다. 이러한 이유는 필터에 대한 i 번째 클리핑이 유용하지 않을 것이기 때문인데, 왜냐하면 그의 결과가 제로 계수와 곱해지고 따라서 필터링된 출력에 대한 i 번째 클리핑으로부터 어떠한 효과도 없을 것이기 때문이다. 일 변형례에서, 해당 클리핑은 기본 값(예를 들면, 샘플 비트 깊이에 따른 최대 가능 값)으로 설정된다. 다른 변형례에서, 필터링 프로세스는 나중에 0 계수와 곱해질 것으로 예상되는 입력(들) 샘플(들)에 클리핑을 적용하지 않는다.

[0217] 실시예에 따르면, 비선형 필터가 해당 루마 필터에 대해 사용되는지 여부를 나타내기 위해 각각의 루마 필터에 대해 플래그가 시그널링된다. 변형례에서, 루마 필터의 클리핑 파라미터들/값들은 해당 루마 필터에 대해 비선형 필터가 사용되는 경우(즉, 플래그가 활성인 경우)에만 시그널링된다.

[0218] 실시예에 따르면, 비선형 필터가 해당 크로마 필터에 대해 사용되는지 여부를 나타내기 위해 각각의 크로마 필터에 대해 플래그가 시그널링된다. 변형례에서, 크로마 필터의 클리핑 파라미터들/값들은 해당 크로마 필터에 대해 비선형 필터가 사용되는 경우(즉, 플래그가 활성인 경우)에만 시그널링된다.

[0219] 실시예에 따르면, ALF 파라미터들 및 플래그들과 같은 ALF 관련 정보를 시그널링하는 것은: VVC 초안 버전 3에서와 동일한 신택스 명명 규칙을 사용하는, 아래에 나와 있는 APS 신택스(표 1); 타일 그룹 헤더 신택스(표 2); 코딩 트리 유닛 신택스(표 3); 및/또는 비선형 ALF 데이터 신택스(표 4) 중 하나 이상을 사용하여 수행된다. 변형례에 따르면, 네 가지 신택스들(예를 들면, 표 1 내지 표 4) 모두가 ALF 관련 정보를 시그널링하는 데 사용된다. 대안적인 변형례에 따르면, ALF 관련 정보를 시그널링하기 위해 네 가지 신택스들의 서브세트가 사용된다.

[0220] 표 1과 표 2는 ALF 파라미터(들)를 제공/시그널링하기 위한 상위 레벨 신택스 요소들, 예를 들면, alf_data()를 제공한다.

표 1

adaptation_parameter_set_rbsp( ) {	기술자
adaptation_parameter_set_id	ue(v)
alf_data( )	
}	

[0221] 표 1 - 적응 파라미터 세트(APS) 신택스

표 2

[0222]

tile_group_header( ) {	기술자
...	
if(sps_alf_enabled_flag) {	
tile_group_alf_enabled_flag	u(1)
if(tile_group_alf_enabled_flag)	
tile_group_aps_id	ue(v)
}	
...	u(1)
}	

[0223] 표 2 - 타일 그룹 헤더 신택스

[0224] 다른 실시예들(또는 그 변형례들)과 관련하여 기술되는 슬라이스(헤더)는 표 2의 타일 그룹 헤더 신택스에 나와 있는 바와 같이 타일 그룹(헤더)로 대체된다. 따라서 'tile_group_alf_enabled_flag'는 도 2에서의 202, 도 5에서의 502에 대한 신택스 요소에 해당하며, 이는 타일 그룹에 대해 ALF가 활성인지 여부를 나타낸다. 게다가, ALF 데이터 신택스 요소는 타일 그룹 헤더에서 제공되지 않는다(슬라이스에 대한 다른 실시예들에서, 이는 슬라이스 헤더에서 제공될 수 있다). 그 대신에, ALF 데이터 신택스 요소는 표 1에 나와 있는 바와 같이 적응 파라미터 세트(APS)라고 불리는 특정 파라미터 세트에서 제공된다. ALF 데이터는 ALF 신택스 요소들을 포함하고, ALF 데이터 신택스 요소를 APS에서 제공하는 것은 하나 초과의 타일 그룹들 사이에서, 예를 들면, 동일한 및/또는 상이한 디코딩되는 픽처들 내의 복수의 타일 그룹들 사이에서 ALF 파라미터들을 공유하는 것을 가능하게 한다.

[0225] 표 1의 적응 파라미터 세트(APS) 신택스는 임의의 적응 파라미터 세트의 시그널링을 정의하는 데 사용된다. 변형례에서, APS는 비-'비디오 코딩 계층'(VCL) '네트워크 추상화 계층'(NAL) 유닛(예를 들면, "APS_NUT" 또는 APS NAL 유닛 유형이라고 함)에 포함된다.

[0226] APS에 대한 각각의 신택스 요소들의 시맨틱스는 다음과 같다:

[0227] **적응 파라미터 세트 시맨틱스**

[0228] **adaptation_parameter_set_id**는 다른 신택스 요소들에 의한 참조를 위해 APS를 식별해 준다. adaptation_parameter_set_id의 값은 0 내지 63(경계 포함)의 범위에 있어야 한다.

[0229] 표 2의 타일 그룹 헤더 신택스는 각각의 타일 그룹 헤더를 정의하는 데 사용된다. 각각의 타일 그룹에 대한 타일 그룹 헤더가 제공된다. 타일 그룹은 타일(들)의 세트를 포함하고, 각각의 타일은 CTU(들)의 세트를 포함한다.

[0230] 타일 그룹 헤더에 대한 신택스 요소들의 시맨틱스는 다음과 같다:

[0231] **타일 그룹 헤더 시맨틱스**

[0232] **tile_group_aps_id**는 타일 그룹이 참조하는 APS의 adaptation_parameter_set_id를 명시한다. tile_group_aps_id의 값은 0 내지 63(경계 포함)의 범위에 있어야 한다. tile_group_aps_id와 동일한 adaptation_parameter_set_id를 갖는 APS NAL 유닛의 TemporalId는 코딩되는 타일 그룹 NAL 유닛의 TemporalId보다 작거나 같아야 한다.

[0233] 표 3은, CTU 레벨에서, ALF가 활성인 각각의 성분에 대해 사용되는 ALF(인에이블) 플래그들을 시그널링하기 위한 신택스 요소들을 제공한다. 이는 도 2에서의 208, 209 및 210 및 도 5에서의 508, 509 및 510에서 사용되는 신택스 요소들에 대응한다.

표 3

[0234]

coding_tree_unit( ) {	기술자
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
...	
if(tile_group_alf_enabled_flag){	
alf_ctb_flag[0][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]	ae(v)
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)
}	
...	

[0235] 표 3 - 코딩 트리 유닛 신택스

[0236] 표 3의 코딩 트리 유닛 신택스는 (인코딩되는) 코딩 트리 유닛을 정의하는 데 사용된다. 코딩 트리 유닛에 대한 각각의 신택스 요소들의 시맨틱스는 다음과 같다:

[0237] **코딩 트리 유닛 시맨틱스**

[0238] CTU는 코딩 쿼드트리 구조의 루트 노드이다.

[0239] **alf_ctb_flag**[cIdx][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]가 1과 동일한 것은 적응 루프 필터가 루마 위치 (xCtb, yCtb)에 있는 코딩 트리 유닛의 cIdx로 표시되는 색상 성분의 코딩 트리 블록에 적용되는 것을 명시한다. alf_ctb_flag[cIdx][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]가 0과 동일한 것은 적응 루프 필터가 루마 위치 (xCtb, yCtb)에 있는 코딩 트리 유닛의 cIdx로 표시되는 색상 성분의 코딩 트리 블록에 적용되지 않는 것을 명시한다.

[0240] alf_ctb_flag[cIdx][xCtb >> Log2CtbSize][yCtb >> Log2CtbSize]가 존재하지 않을 때, 이는 0과 동일한 것으로 추론된다.

[0241] 표 4는, VVC 초안 버전 3의 ALF 데이터 신택스 요소들을 기반으로 하는, 비선형 ALF 파라미터들(즉, 비선형 ALF 데이터 신택스)를 시그널링하기 위한 신택스 요소들을 제공한다. 이러한 신택스 요소들은 도 2의 신택스 요소들을 기반으로 하는 도 5의 비선형 ALF 신택스 요소들을 참조하여 본 명세서에서 기술된 변형례에서 사용되는 것들을 기반으로 한다.

표 4

alf_data( ) {	기술자
alf_chroma_idc	tu(v)
alf_luma_clip	u(1)
if(alf_choma_idc)	
alf_chroma_clip	u(1)
if(alf_luma_clip) {	
alf_luma_clip_default_table	u(1)
if(!alf_luma_clip_default_table) {	
alf_luma_num_clipping_values_minus1	ue(v)
for(clipIdx = 0; clipIdx <= alf_luma_num_clipping_values_minus1; clipIdx++)	
alf_luma_clipping_value[clipIdx]	u(bitDepthY)
}	
}	
if(alf_chroma_clip) {	
alf_chroma_clip_default_table	u(1)
if(!alf_chroma_clip_default_table) {	
alf_chroma_num_clipping_values_minus1	ue(v)
for(clipIdx = 0; clipIdx <= alf_chroma_num_clipping_values_minus1; clipIdx++)	
alf_chroma_clipping_value[clipIdx]	u(bitDepthC)
}	
}	
alf_luma_num_filters_signalled_minus1	
if(alf_luma_num_filters_signalled_minus1 > 0) {	
for(filtIdx = 0; filtIdx < NumAlfFilters; filtIdx++)	
alf_luma_coeff_delta_idx[filtIdx]	tb(v)
if(alf_luma_clip) {	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++)	
alf_luma_filter_clip[sigFiltIdx]	u(1)
}	
}	
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	ue(v)
for(i = 0; i < 3; i++)	

[0242]

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 < 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_chroma_min_eg_order_minus1	ue(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)
}	
}	
if(alf_luma_clip) {	
alf_luma_clip_min_eg_order_minus1	ue(v)
for(i = 0; i < 3; i++)	
alf_luma_clip_eg_order_increase_flag[i]	u(1)

[0243]

for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++) {	
if (alf_luma_coeff_flag[sigFiltIdx] && alf_luma_filter_clip[sigFiltIdx]) {	
for (j = 0; j < 12; j++) {	
if(AlfCoeff$_L$[filtIdx][j])	
alf_luma_clip_idx[sigFiltIdx][j]	uek(v)
}	
}	
}	
}	
if (alf_chroma_idc > 0 && alf_chroma_clip) {	
alf_chroma_clip_min_eg_order_minus1	ue(v)
for(i = 0; i < 2; i++)	
alf_chroma_clip_eg_order_increase_flag[i]	u(1)
for(j = 0; j < 6; j++) { if(AlfCoeff$_C$[j])	
alf_chroma_clip_idx[j]	uek(v)
}	
}	
}	

[0244]

[0245] 표 4 - 비선형 ALF 데이터 신택스

[0246] 표 4의 비선형 적응 루프 필터(ALF) 데이터 신택스는 (표 1에 나와 있는) 적응 파라미터 세트를 정의하는 데 사

용된다. 비선형 ALF 데이터 신택스 요소들의 시맨틱스는 다음과 같다:

[0247] **적응 루프 필터 데이터 시맨틱스**

[0248] **alf_chroma_idc**가 0과 동일한 것은 적응 루프 필터가 Cb 및 Cr 색상 성분들에 적용되지 않는다는 것을 명시한다. alf_chroma_idc가 1과 동일한 것은 적응 루프 필터가 Cb 색상 성분에 적용된다는 것을 나타낸다. alf_chroma_idc가 2와 동일한 것은 적응 루프 필터가 Cr 색상 성분에 적용된다는 것을 나타낸다. alf_chroma_idc가 3과 동일한 것은 적응 루프 필터가 Cb 및 Cr 색상 성분들에 적용된다는 것을 나타낸다.

[0249] 절사된 단항 이진화(truncated unary binarization) tu(v)의 최댓값 maxVal은 3과 동일하도록 설정된다.

[0250] 상이한 적응 루프 필터들의 수를 명시하는 변수 NumAlfFilters는 25와 동일하도록 설정된다.

[0251] **alf_luma_num_filters_signalled_minus1** + 1은 루마 계수들이 시그널링될 수 있는 적응 루프 필터 클래스들의 수를 명시한다. alf_luma_num_filters_signalled_minus1의 값은 0 내지 NumAlfFilters-1(경계 포함)의 범위에 있어야 한다.

[0252] 절사된 이항 이진화(truncated binary binarization) tb(v)의 최댓값 maxVal은 NumAlfFilters - 1과 동일하도록 설정된다.

[0253] **alf_luma_coeff_delta_idx**[filtIdx]는 0 내지 NumAlfFilters - 1의 범위에 있는 filtIdx로 표시되는 필터 클래스에 대한 시그널링된 적응 루프 필터 루마 계수 델타들의 인덱스들을 명시한다. alf_luma_coeff_delta_idx[filtIdx]가 존재하지 않을 때, 이는 0과 동일한 것으로 추론된다.

[0254] 절사된 이항 이진화 tb(v)의 최댓값 maxVal은 alf_luma_num_filters_signalled_minus1과 동일하도록 설정된다.

[0255] **alf_luma_coeff_delta_flag**가 1과 동일한 것은 alf_luma_coeff_delta_prediction_flag가 시그널링되지 않는다는 것을 나타낸다. alf_luma_coeff_delta_flag가 0과 동일한 것은 alf_luma_coeff_delta_prediction_flag가 시그널링될 수 있다는 것을 나타낸다.

[0256] **alf_luma_coeff_delta_prediction_flag**가 1과 동일한 것은 시그널링되는 루마 필터 계수 델타들이 이전 루마 계수들의 델타들로부터 예측된다는 것을 명시한다. alf_luma_coeff_delta_prediction_flag가 0과 동일한 것은 시그널링되는 루마 필터 계수 델타들이 이전 루마 계수들의 델타들로부터 예측되지 않는다는 것을 명시한다. 존재하지 않을 때, alf_luma_coeff_delta_prediction_flag는 0과 동일한 것으로 추론된다.

[0257] **alf_luma_min_eg_order_minus1** + 1은 루마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수를 명시한다. alf_luma_min_eg_order_minus1의 값은 0 내지 6(경계 포함)의 범위에 있어야 한다.

[0258] **alf_luma_eg_order_increase_flag**[i]가 1과 동일한 것은 루마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분된다는 것을 명시한다. alf_luma_eg_order_increase_flag[i]가 0과 동일한 것은 루마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분되지 않는다는 것을 명시한다.

[0259] alf_luma_coeff_delta_abs[sigFiltIdx][j]의 값들을 디코딩하는 데 사용되는 지수-골롬 코드의 차수 expGoOrderY[i]는 다음과 같이 도출된다:

[0260] expGoOrderY[i] = alf_luma_min_eg_order_minus1 + 1+ alf_luma_eg_order_increase_flag[i]

[0261] **alf_luma_coeff_flag**[sigFiltIdx]가 1과 동일한 것은 sigFiltIdx로 표시되는 루마 필터의 계수들이 시그널링된다는 것을 명시한다. alf_luma_coeff_flag[sigFiltIdx]가 0과 동일한 것은 sigFiltIdx로 표시되는 루마 필터의 모든 필터 계수들이 0과 동일하도록 설정된다는 것을 명시한다. 존재하지 않을 때, alf_luma_coeff_flag[sigFiltIdx]는 1과 동일하도록 설정된다.

[0262] **alf_luma_coeff_delta_abs**[sigFiltIdx][j]는 sigFiltIdx로 표시되는 시그널링되는 루마 필터의 j 번째 계수 델타의 절댓값을 명시한다. alf_luma_coeff_delta_abs[sigFiltIdx][j]가 존재하지 않을 때, 이는 0과 동일한 것으로 추론된다.

[0263] 지수-골롬 이진화 uek(v)의 차수 k는 다음과 같이 도출된다:

[0264] golombOrderIdxY[ ] = { 0, 0, 1, 0, 0, 1, 2, 1, 0, 0, 1, 2 }

[0265] k = expGoOrderY[golombOrderIdxY[j]]

[0266] **alf_luma_coeff_delta_sign**[sigFiltIdx][j]는 다음과 같이 sigFiltIdx로 표시되는 필터의 j 번째 루마 계수의 부호를 명시한다.

[0267] - alf_luma_coeff_delta_sign[sigFiltIdx][j]가 0과 동일한 경우, 대응하는 루마 필터 계수는 양의 값을 갖는다.

[0268] - 그렇지 않은 경우(alf_luma_coeff_delta_sign[sigFiltIdx][j]가 1과 동일한 경우), 대응하는 루마 필터 계수는 음의 값을 갖는다.

[0269] alf_luma_coeff_delta_sign[sigFiltIdx][j]가 존재하지 않을 때, 이는 0과 동일한 것으로 추론된다.

[0270] 변수 filterCoefficients[sigFiltIdx][j], 단 sigFiltIdx = 0..alf_luma_num_filters_signalled_minus1, j = 0..11은 다음과 같이 초기화된다:

[0271] filterCoefficients[sigFiltIdx][j] = alf_luma_coeff_delta_abs[sigFiltIdx][j] * (1 - 2 * alf_luma_coeff_delta_sign[sigFiltIdx][j])

[0272] alf_luma_coeff_delta_prediction_flag가 1과 동일할 때, filterCoefficients[sigFiltIdx][j], 단 sigFiltIdx = 1..alf_luma_num_filters_signalled_minus1, j = 0..11은 다음과 같이 수정된다:

[0273] filterCoefficients[sigFiltIdx][j] += filterCoefficients[sigFiltIdx - 1][j]

[0274] 요소들 AlfCoeffL[filtIdx][j]를 갖는 루마 필터 계수들 AlfCoeffL, 단, filtIdx = 0..NumAlfFilters - 1, j = 0..11은 다음과 같이 도출된다:

[0275] AlfCoeffL[filtIdx][j] = filterCoefficients[alf_luma_coeff_delta_idx[filtIdx]][j]

[0276] filtIdx = 0..NumAlfFilters - 1에 대한 마지막 필터 계수들 AlfCoeffL[filtIdx][12]는 다음과 같이 도출된다:

[0277] AlfCoeffL[filtIdx][12] = 128

[0278] AlfCoeffL[filtIdx][j], 단, filtIdx = 0..NumAlfFilters-1, j = 0..11의 값들이 -2^7 내지 2^7 - 1(경계 포함)의 범위에 있어야 한다는 것과 AlfCoeffL[filtIdx][12]의 값들이 0 내지 2^8 - 1(경계 포함)의 범위에 있어야 한다는 것이 비트스트림 적합성의 요구사항이다.

[0279] **alf_chroma_min_eg_order_minus1** + 1은 크로마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수를 명시한다. alf_chroma_min_eg_order_minus1의 값은 0 내지 6(경계 포함)의 범위에 있어야 한다.

[0280] **alf_chroma_eg_order_increase_flag**[i]가 1과 동일한 것은 크로마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분된다는 것을 명시한다. alf_chroma_eg_order_increase_flag[i]가 0과 동일한 것은 크로마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분되지 않는다는 것을 명시한다.

[0281] alf_chroma_coeff_abs[j]의 값들을 디코딩하는 데 사용되는 지수-골롬 코드의 차수 expGoOrderC[i]는 다음과 같이 도출된다:

[0282] expGoOrderC[i] = alf_chroma_min_eg_order_minus1 + 1+ alf_chroma_eg_order_increase_flag[i]

[0283] **alf_chroma_coeff_abs**[j]는 j 번째 크로마 필터 계수의 절댓값을 명시한다. alf_chroma_coeff_abs[j]가 존재하지 않을 때, 이는 0과 동일한 것으로 추론된다. alf_chroma_coeff_abs[j]의 값들이 0 내지 2^7 - 1(경계 포함)의 범위에 있어야 한다는 것이 비트스트림 적합성의 요구사항이다.

[0284] 지수-골롬 이진화 uek(v)의 차수 k는 다음과 같이 도출된다:

[0285] golombOrderIdxC[ ] = { 0, 0, 1, 0, 0, 1 }

[0286] k = expGoOrderC[golombOrderIdxC[j]]

[0287] **alf_chroma_coeff_sign**[j]는 다음과 같이 j 번째 크로마 필터 계수의 부호를 명시한다:

[0288] - alf_chroma_coeff_sign[j]가 0과 동일한 경우, 대응하는 크로마 필터 계수는 양의 값을 갖는다.

[0289] - 그렇지 않은 경우(alf_chroma_coeff_sign[j]가 1과 동일한 경우), 대응하는 크로마 필터 계수는 음의 값을 갖는다.

[0290] alf_chroma_coeff_sign[j]가 존재하지 않을 때, 이는 0과 동일한 것으로 추론된다.

[0291] 요소들 cC[j], 단, j = 0..5를 갖는 크로마 필터 계수들 AlfCoeffC는 다음과 같이 도출된다:

[0292] AlfCoeffC[j] = alf_chroma_coeff_abs[j] * (1 - 2 * alf_chroma_coeff_sign[j])

[0293] j = 6에 대한 마지막 필터 계수는 다음과 같이 도출된다:

[0294] AlfCoeffC[6] = 128

[0295] AlfCoeffC[j], 단, j = 0..5의 값들이 -2^7 - 1 내지 2^7 - 1(경계 포함)의 범위에 있어야 한다는 것과 AlfCoeffC[6]의 값들이 0 내지 2^8 - 1(경계 포함)의 범위에 있어야 한다는 것이 비트스트림 적합성의 요구사항이다.

[0296] **alf_luma_clip**이 0과 동일한 것은 선형 적응 루프 필터가 루마 성분에 적용된다는 것을 명시한다. alf_luma_clip이 1과 동일한 것은 비선형 적응 루프 필터가 루마 성분에 적용될 수 있다는 것을 명시한다.

[0297] **alf_chroma_clip**이 0과 동일한 것은 선형 적응 루프 필터가 크로마 성분들에 적용된다는 것을 명시하고; alf_chroma_clip이 1과 동일한 것은 비선형 적응 루프 필터가 크로마 성분에 적용된다는 것을 명시한다. 존재하지 않는 경우, alf_chroma_clip은 0으로 추론된다.

[0298] **alf_luma_clip_default_table**이 1과 동일한 것은 기본 alf_luma_clipping_value[ ] 테이블이 클리핑 인덱스를 클리핑 값으로 변환하는 데 사용된다는 것을 명시한다. Alf_luma_clip_default_table이 0과 동일한 것은 alf_luma_clipping_value[ ] 테이블이 alf_data()에 존재한다는 것을 나타낸다.

[0299] **alf_luma_num_clipping_values_minus1** + 1은 alf_luma_clipping_value[ ] 테이블의 크기를 나타낸다. 존재하지 않는 경우, 이는 bitDepthY와 동일한 것으로 추론된다.

[0300] **alf_luma_clipping_value**[clipIdx]는 클리핑 인덱스 clipIdx가 alf_luma_clip_idx[ ][ ] 테이블에서 시그널링될 때 사용할 클리핑 값을 명시한다. 존재하지 않을 때, 이는 alf_luma_clipping_value[ ] = { 1 << bitDepthY, 1 << (bitDepthY-1), ..., 8, 4, 2 }인 것으로 추론된다.

[0301] 편의상, alf_luma_clipping_value[alf_luma_num_clipping_values_minus1 + 1]은 1 << bitDepthY와 동일한 것으로 추론된다.

[0302] **alf_chroma_clip_default_table**이 1과 동일한 것은 기본 alf_chroma_clipping_value[ ] 테이블이 클리핑 인덱스를 클리핑 값으로 변환하는 데 사용된다는 것을 명시하고; alf_chroma_clip_default_table이 0과 동일한 것은 alf_chroma_clipping_value[ ] 테이블이 alf_data()에 존재한다는 것을 나타낸다.

[0303] **alf_chroma_num_clipping_values_minus1** + 1은 alf_chroma_clipping_value[ ] 테이블의 크기를 나타낸다. 존재하지 않는 경우, 이는 bitDepthC와 동일한 것으로 추론된다.

[0304] **alf_chroma_clipping_value**[clipIdx]는 클리핑 인덱스 clipIdx가 alf_chroma_clip_idx[ ][ ] 테이블에서 시그널링될 때 사용할 클리핑 값을 명시한다. 존재하지 않을 때, 이는 alf_chroma_clipping_value[ ] = { 1 << bitDepthC, 1 << (bitDepthC-1), ..., 8, 4, 2 }인 것으로 추론된다.

[0305] 편의상, alf_chroma_clipping_value[alf_chroma_num_clipping_values_minus1 + 1]은 1 << bitDepthC와 동일한 것으로 추론된다.

[0306] **alf_luma_filter_clip**[sigFiltIdx]가 0과 동일한 것은 선형 적응 루프 필터가 sigFiltIdx로 표시되는 루마 필터에 대해 적용된다는 것을 명시한다. alf_luma_filter_clip[sigFiltIdx]가 1과 동일한 것은 비선형 적응 루프 필터가 sigFiltIdx로 표시되는 루마 필터에 대해 적용된다는 것을 명시한다.

[0307] **alf_luma_clip_min_eg_order_minus1** + 1은 루마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수를 명시한다. alf_luma_clip_min_eg_order_minus1의 값은 0 내지 6(경계 포함)의 범위에 있어야 한다.

[0308] **alf_luma_clip_eg_order_increase_flag**[i]가 1과 동일한 것은 루마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분된다는 것을 명시한다. alf_luma_clip_eg_order_increase_flag[i]가 0과 동일한 것은 루마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분되지 않는다는 것을 명시한다.

[0309] alf_luma_clip_idx[sigFiltIdx][j] 의 값들을 디코딩하는 데 사용되는 지수-골롬 코드의 차수

expGoOrderYClip[i]는 다음과 같이 도출된다:

[0310] expGoOrderYClip[i] = alf_luma_clip_min_eg_order_minus1 + 1+ alf_luma_clip_eg_order_increase_flag[i]

[0311] **alf_luma_clip_idx**[sigFiltIdx][j]는 sigFiltIdx로 표시되는 시그널링되는 루마 필터의 j 번째 계수와 곱하기 전에 사용할 클리핑의 클리핑 인덱스를 명시한다. alf_luma_clip_idx[sigFiltIdx][j]가 존재하지 않을 때, 이는 alf_luma_num_clipping_values_minus1 + 1(클리핑 없음)과 동일한 것으로 추론된다.

[0312] 지수-골롬 이진화 uek(v)의 차수 k는 다음과 같이 도출된다:

[0313] golombOrderIdxYClip[ ] = { 0, 0, 1, 0, 0, 1, 2, 1, 0, 0, 1, 2 }

[0314] k = expGoOrderYClip[golombOrderIdxYClip[j]]

[0315] 변수 filterClips[sigFiltIdx][j], 단 sigFiltIdx = 0..alf_luma_num_filters_signalled_minus1, j = 0..11은 다음과 같이 초기화된다:

[0316] filterClips[sigFiltIdx][j] = alf_luma_clipping_value[alf_luma_clip_idx[sigFiltIdx][j]]

[0317] 요소들 AlfClipL[filtIdx][j]를 갖는 루마 필터 클리핑 값들 AlfClipL, 단, filtIdx = 0..NumAlfFilters - 1, j = 0..11은 다음과 같이 도출된다:

[0318] AlfClipL[filtIdx][j] = filterClips[alf_luma_coeff_delta_idx[filtIdx]][j]

[0319] **alf_chroma_clip_min_eg_order_minus1** + 1은 크로마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수를 명시한다. alf_chroma_clip_min_eg_order_minus1의 값은 0 내지 6(경계 포함)의 범위에 있어야 한다.

[0320] **alf_chroma_clip_eg_order_increase_flag**[i]가 1과 동일한 것은 크로마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분된다는 것을 명시한다. alf_chroma_clip_eg_order_increase_flag[i]가 0과 동일한 것은 크로마 필터 계수 시그널링에 대한 지수-골롬 코드의 최소 차수가 1씩 증분되지 않는다는 것을 명시한다.

[0321] alf_chroma_coeff_abs[j]의 값들을 디코딩하는 데 사용되는 지수-골롬 코드의 차수 expGoOrderC[i]는 다음과 같이 도출된다:

[0322] expGoOrderC[i] = alf_chroma_clip_min_eg_order_minus1 + 1+ alf_chroma_clip_eg_order_increase_flag[i]

[0323] **alf_chroma_clip_idx**[j]는 크로마 필터의 j 번째 계수와 곱하기 전에 사용할 클리핑의 클리핑 인덱스를 명시한다. alf_chroma_clip_idx[j]가 존재하지 않을 때, 이는 alf_chroma_num_clipping_values_minus1 + 1(클리핑 없음)과 동일한 것으로 추론된다.

[0324] 지수-골롬 이진화 uek(v)의 차수 k는 다음과 같이 도출된다:

[0325] golombOrderIdxC[ ] = { 0, 0, 1, 0, 0, 1 }

[0326] k = expGoOrderC[golombOrderIdxC[j]]

[0327] 요소들 AlfClipC [j], 단, j = 0..5를 갖는 크로마 필터 클리핑 값들 AlfClipC는 다음과 같이 도출된다:

[0328] AlfClipC[j] = alf_chroma_clipping_value[alf_chroma_clip_idx[j]]

[0329] 변형례에 따르면, alf_luma_num_clipping_values_minus1은 alf_luma_num_clipping_values_minus2로 대체되고 클리핑 값들은 index=1 내지 alf_luma_num_clipping_values_minus2+1로부터 제공되며, alf_luma_clipping_value[ ]의 첫 번째 값은 alf_luma_clipping_value[0] = 1<<bitDepthY로 추론된다. alf_chroma_num_clipping_values_minus1은 alf_chroma_num_clipping_values_minus2로 대체되고 클리핑 값들은 index=1 내지 alf_chroma_num_clipping_values_minus2+1로부터 제공되며, alf_chroma_clipping_value[ ]의 첫 번째 값은 alf_chroma_clipping_value[0] = 1<<bitDepthC로 추론된다.

[0330] 변형례에 따르면, 적응 루프 필터링 프로세스는 상기 신택스 요소들을 사용한다. 그러한 변형례는 아래에서 VVC 초안 규격의 표기법 규칙을 사용하여 기술된다.

[0331] **적응 루프 필터 프로세스**

[0332] 일반

[0333] 이 프로세스의 입력들은 적응 루프 필터 이전의 재구성된 픽처 샘플 어레이들 recPictureL, recPictureCb 및 recPictureCr이다.

[0334] 이 프로세스의 출력들은 적응 루프 필터 이후의 수정된 재구성된 픽처 샘플 어레이들 alfPictureL, alfPictureCb 및 alfPictureCr이다.

[0335] 적응 루프 필터 이후의 수정된 재구성된 픽처 샘플 어레이들 alfPictureL, alfPictureCb 및 alfPictureCr에서의 샘플 값들은 초기에, 제각기, 적응 루프 필터 이전의 재구성된 픽처 샘플 어레이들 recPictureL, recPictureCb 및 recPictureCr에서의 샘플 값들과 동일하도록 설정된다.

[0336] tile_group_alf_enabled_flag가 1과 동일할 때, 루마 코딩 트리 블록 위치 (rx, ry), 단, rx = 0..PicWidthInCtbs - 1이고 ry = 0..PicHeightInCtbs - 1를 갖는 모든 코딩 트리 유닛에 대해, 이하가 적용된다:

[0337] alf_ctb_flag[0][rx][ry]가 1일 때, '루마 샘플들에 대한 코딩 트리 블록 필터링 프로세스' 절에 명시된 바와 같은 루마 샘플들에 대한 코딩 트리 블록 필터링 프로세스가 recPictureL, alfPictureL, 및 (rx << CtbLog2SizeY, ry << CtbLog2SizeY)와 동일하도록 설정된 루마 코딩 트리 블록 위치 (xCtb, yCtb)를 입력들로서 사용하여 호출되고, 출력은 수정된 필터링된 픽처 alfPictureL이다.

[0338] alf_ctb_flag[1][rx][ry]가 1과 동일할 때, '크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스' 절에 명시된 바와 같은 크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스는 recPictureCb와 동일하도록 설정된 recPicture, alfPictureCb와 동일하도록 설정된 alfPicture, 및 (rx << (CtbLog2SizeY - 1), ry << (CtbLog2SizeY - 1))와 동일하도록 설정된 크로마 코딩 트리 블록 위치 (xCtbC, yCtbC)를 입력들로서 사용하여 호출되고, 출력은 수정된 필터링된 픽처 alfPictureCb이다.

[0339] alf_ctb_flag[2][rx][ry]가 1과 동일할 때, '크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스' 절에 명시된 바와 같은 크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스는 recPictureCr과 동일하도록 설정된 recPicture, alfPictureCr과 동일하도록 설정된 alfPicture, 및 (rx << (CtbLog2SizeY - 1), ry << (CtbLog2SizeY - 1))와 동일하도록 설정된 크로마 코딩 트리 블록 위치 (xCtbC, yCtbC)를 입력들로서 사용하여 호출되고, 출력은 수정된 필터링된 픽처 alfPictureCr이다.

[0340] 루마 샘플들에 대한 코딩 트리 블록 필터링 프로세스

[0341] 이 프로세스의 입력들은: 적응 루프 필터링 프로세스 이전의 재구성된 루마 픽처 샘플 어레이 recPictureL, 필터링된 재구성된 루마 픽처 샘플 어레이 alfPictureL, 현재 픽처의 좌측 상단 샘플에 상대적인 현재 루마 코딩 트리 블록의 좌측 상단 샘플을 명시하는 루마 위치 (xCtb, yCtb)이다.

[0342] 이 프로세스의 출력은 수정된 필터링된 재구성된 루마 픽처 샘플 어레이 alfPictureL이다.

[0343] '루마 샘플들에 대한 ALF 전치 및 필터 인덱스에 대한 도출 프로세스' 절에서의 필터 인덱스에 대한 도출 프로세스는 위치 (xCtb, yCtb) 및 재구성된 루마 픽처 샘플 어레이 recPictureL을 입력들로서 사용하여 호출되고, filtIdx[x][y] 및 transposeIdx[x][y], 단 x, y = 0..CtbSizeY - 1이 출력들로서 얻어진다.

[0344] 필터링된 재구성된 루마 샘플들 alfPictureL[x][y]의 도출을 위해, 현재 루마 코딩 트리 블록 recPictureL[x][y] 내의 각각의 재구성된 루마 샘플은 다음과 같이 필터링되며, 단, x, y = 0..CtbSizeY - 1이다:

[0345] 루마 필터 계수들의 어레이 f[j] 및 filtIdx[x][y]에 의해 명시되는 필터에 대응하는 루마 필터 클리핑의 어레이 c[j]는 다음과 같이 도출되며, 단, j = 0..12이다:

[0346] f[j] = AlfCoeffL[filtIdx[x][y]][j]

[0347] c[j] = AlfClipL[filtIdx[x][y]][j]

[0348] 루마 필터 계수들 filterCoeff 및 필터 클리핑 값들 filterClip은 다음과 같이 transposeIdx[x][y]에

따라 도출된다:

[0349] transposeIndex[x][y] = = 1인 경우,

[0350] filterCoeff[ ] = { f[9], f[4], f[10], f[8], f[1], f[5], f[11], f[7], f[3], f[0], f[2], f[6], f[12] }

[0351] filterClip[ ] = { c[9], c[4], c[10], c[8], c[1], c[5], c[11], c[7], c[3], c[0], c[2], c[6], c[12] }

[0352] 그렇지 않고, transposeIndex[x][y] = = 2인 경우,

[0353] filterCoeff[ ] = { f[0], f[3], f[2], f[1], f[8], f[7], f[6], f[5], f[4], f[9], f[10], f[11], f[12] }

[0354] filterClip[ ] = { c[0], c[3], c[2], c[1], c[8], c[7], c[6], c[5], c[4], c[9], c[10], c[11], c[12] }

[0355] 그렇지 않고, transposeIndex[x][y] = = 3인 경우,

[0356] filterCoeff[ ] = { f[9], f[8], f[10], f[4], f[3], f[7], f[11], f[5], f[1], f[0], f[2], f[6], f[12] }

[0357] filterClip[ ] = { c[9], c[8], c[10], c[4], c[3], c[7], c[11], c[5], c[1], c[0], c[2], c[6], c[12] }

[0358] 그렇지 않은 경우,

[0359] filterCoeff[ ] = { f[0], f[1], f[2], f[3], f[4], f[5], f[6], f[7], f[8], f[9], f[10], f[11], f[12] }

[0360] filterClip[ ] = { c[0], c[1], c[2], c[3], c[4], c[5], c[6], c[7], c[8], c[9], c[10], c[11], c[12] }

[0361] 루마 샘플들의 주어진 어레이 recPicture 내의 대응하는 루마 샘플들 (x, y) 각각에 대한 위치들 (hx, vy)는 다음과 같이 도출된다:

[0362] hx = Clip3(0, pic_width_in_luma_samples - 1, xCtb + x)

[0363] vy = Clip3(0, pic_height_in_luma_samples - 1, yCtb + y)

[0364] 변수 curr은 다음과 같이 도출된다: curr = recPictureL[hx, vy]

[0365] 변수 sum은 다음과 같이 도출된다:

[0366] sum = filterCoeff[0] * (Clip3(-filterClip[0], filterClip[0], recPictureL[hx, vy + 3] - curr) + Clip3(-filterClip[0], filterClip[0], recPictureL[hx, vy - 3] - curr))

[0367] + filterCoeff[1] * (Clip3(-filterClip[1], filterClip[1], recPictureL[hx + 1, vy + 2] - curr) + Clip3(-filterClip[1], filterClip[1], recPictureL[hx - 1, vy - 2] - curr))

[0368] + filterCoeff[2] * (Clip3(-filterClip[2], filterClip[2], recPictureL[hx, vy + 2] - curr) + Clip3(-filterClip[2], filterClip[2], recPictureL[hx, vy - 2] - curr))

[0369] + filterCoeff[3] * (Clip3(-filterClip[3], filterClip[3], recPictureL[hx - 1, vy + 2] - curr) + Clip3(-filterClip[3], filterClip[3], recPictureL[hx + 1, vy - 2] - curr))

[0370] + filterCoeff[4] * (Clip3(-filterClip[4], filterClip[4], recPictureL[hx + 2, vy + 1] - curr) + Clip3(-filterClip[4], filterClip[4], recPictureL[hx - 2, vy - 1] - curr))

[0371] + filterCoeff[5] * (Clip3(-filterClip[5], filterClip[5], recPictureL[hx + 1, vy + 1] - curr) + Clip3(-filterClip[5], filterClip[5], recPictureL[hx - 1, vy - 1] - curr))

[0372] + filterCoeff[6] * (Clip3(-filterClip[6], filterClip[6], recPictureL[hx, vy + 1] -

curr) + Clip3(-filterClip[6], filterClip[6], recPictureL[hx, vy - 1] - curr))

[0373] + filterCoeff[7] * (Clip3(-filterClip[7], filterClip[7], recPictureL[hx - 1, vy + 1] - curr) + Clip3(-filterClip[7], filterClip[7], recPictureL[hx + 1, vy - 1] - curr))

[0374] + filterCoeff[8] * (Clip3(-filterClip[8], filterClip[8], recPictureL[hx - 2, vy + 1] - curr) + Clip3(-filterClip[8], filterClip[8], recPictureL[hx + 2, vy - 1] - curr))

[0375] + filterCoeff[9] * (Clip3(-filterClip[9], filterClip[9], recPictureL[hx + 3, vy] - curr) + Clip3(-filterClip[9], filterClip[9], recPictureL[hx - 3, vy] - curr))

[0376] + filterCoeff[10] * (Clip3(-filterClip[10], filterClip[10], recPictureL[hx + 2, vy] - curr) + Clip3(-filterClip[10], filterClip[10], recPictureL[hx - 2, vy] - curr))

[0377] + filterCoeff[11] * (Clip3(-filterClip[11], filterClip[11], recPictureL[hx + 1, vy] - curr) + Clip3(-filterClip[11], filterClip[11], recPictureL[hx - 1, vy] - curr))

[0378] + filterCoeff[12] * recPictureL[hx, vy]

[0379] sum = (sum + 64) >> 7

[0380] 수정된 필터링된 재구성된 루마 픽처 샘플 alfPictureL[xCtb + x][yCtb + y]는 다음과 같이 도출된다:

[0381] alfPictureL[xCtb + x][yCtb + y] = Clip3(0, (1 << BitDepthY) - 1, sum)

[0382] 루마 샘플들에 대한 ALF 전치 및 필터 인덱스에 대한 도출 프로세스

[0383] 이 프로세스의 입력들은: 현재 픽처의 좌측 상단 샘플에 상대적인 현재 루마 코딩 트리 블록의 좌측 상단 샘플을 명시하는 루마 위치(xCtb, yCtb), 적응 루프 필터링 프로세스 이전의 재구성된 루마 픽처 샘플 어레이 recPictureL이다.

[0384] 이 프로세스의 출력들은 다음과 같다.

[0385] 분류 필터 인덱스 어레이 filtIdx[x][y], 단, x, y = 0..CtbSizeY - 1,

[0386] 전치 인덱스 어레이 transposeIdx[x][y], 단, x, y = 0..CtbSizeY - 1.

[0387] 루마 샘플들의 주어진 어레이 recPicture 내의 대응하는 루마 샘플들 (x, y) 각각에 대한 위치들 (hx, vy)는 다음과 같이 도출된다:

[0388] hx = Clip3(0, pic_width_in_luma_samples - 1, x)

[0389] vy = Clip3(0, pic_height_in_luma_samples - 1, y)

[0390] 분류 필터 인덱스 어레이 filtIdx 및 전치 인덱스 어레이 transposeIdx는 다음과 같은 순서의 단계들에 의해 도출된다:

[0391] 1) 변수들 filtH[x][y], filtV[x][y], filtD0[x][y] 및 filtD1[x][y], 단, x, y = -2..CtbSizeY + 1이 다음과 같이 도출된다:

[0392] x와 y 둘 모두가 짝수이거나 또는 x와 y 둘 모두가 홀수인 경우, 이하가 적용된다:

[0393] filtH[x][y] = Abs((recPicture[hxCtb+x, vyCtb+y] << 1) - recPicture[hxCtb+x-1, vyCtb+y] - recPicture[hxCtb+x+1, vyCtb+y])

[0394] filtV[x][y] = Abs((recPicture[hxCtb+x, vyCtb+y] << 1) - recPicture[hxCtb+x, vyCtb+y-1] - recPicture[hxCtb+x, vyCtb+y+1])

[0395] filtD0[x][y] = Abs((recPicture[hxCtb+x, vyCtb+y] << 1) - recPicture[hxCtb+x-1, vyCtb+y-1] - recPicture[hxCtb+x+1, vyCtb+y+1])

[0396] filtD1[x][y] = Abs((recPicture[hxCtb+x, vyCtb+y] << 1) - recPicture[

hxCtb+x+1, vyCtb+y-1] - recPicture[hxCtb+x-1, vyCtb+y+1])

[0397] 그렇지 않은 경우, filtH[x][y], filtV[x][y], filtD0[x][y] 및 filtD1[x][y]가 0과 동일하도록 설정된다.

[0398] 2) 변수들 varTempH1[x][y], varTempV1[x][y], varTempD01[x][y], varTempD11[x][y] 및 varTemp[x][y], 단, x, y = 0..(CtbSizeY - 1) >> 2가 다음과 같이 도출된다:

[0399] sumH[x][y] = ∑i∑j filtH[(x << 2) + i][(y << 2) + j], 단, i, j = -2..5

[0400] sumV[x][y] = ∑i∑j filtV[(x << 2) + i][(y << 2) + j], 단, i, j = -2..5

[0401] sumD0[x][y] = ∑i∑j filtD0[(x << 2) + i][(y << 2) + j], 단, i, j = -2..5

[0402] sumD1 [x] [y] = ∑i∑j filtD1 [(x << 2) i] [(y << 2) j], 단, i, j = -2..5

[0403] sumOfHV[x][y] = sumH[x][y] + sumV[x][y]

[0404] 3) 변수들 dir1[x][y], dir2[x][y] 및 dirS[x][y], 단, x, y = 0..CtbSizeY - 1은 다음과 같이 도출된다:

[0405] 변수들 hv1, hv0 및 dirHV는 다음과 같이 도출된다:

[0406] sumV[x >> 2][y >> 2]가 sumH[x >> 2][y >> 2]보다 큰 경우, 이하가 적용된다:

[0407] hv1 = sumV[x >> 2][y >> 2]

[0408] hv0 = sumH[x >> 2][y >> 2]

[0409] dirHV = 1

[0410] 그렇지 않은 경우, 이하가 적용된다:

[0411] hv1 = sumH[x >> 2][y >> 2]

[0412] hv0 = sumV[x >> 2][y >> 2]

[0413] dirHV = 3

[0414] 변수들 d1, d0 및 dirD는 다음과 같이 도출된다:

[0415] sumD0[x >> 2][y >> 2]가 sumD1[x >> 2][y >> 2]보다 큰 경우, 이하가 적용된다:

[0416] d1 = sumD0[x >> 2][y >> 2]

[0417] d0 = sumD1[x >> 2][y >> 2]

[0418] dirD = 0

[0419] 그렇지 않은 경우, 이하가 적용된다:

[0420] d1 = sumD1[x >> 2][y >> 2]

[0421] d0 = sumD0[x >> 2][y >> 2]

[0422] dirD = 2

[0423] 변수들 hvd1, hvd0은 다음과 같이 도출된다:

[0424] hvd1 = (d1 * hv0 > hv1 * d0) ? d1 : hv1

[0425] hvd0 = (d1 * hv0 > hv1 * d0) ? d0 : hv0

[0426] 변수들 dirS[x][y], dir1[x][y] 및 dir2[x][y]는 다음과 같이 도출된다:

[0427] dir1[x][y] = (d1 * hv0 > hv1 * d0) ? dirD : dirHV

[0428] dir2[x][y] = (d1 * hv0 > hv1 * d0) ? dirHV : dirD

[0429] dirS[x][y] = (hvd1 > 2 * hvd0) ? 1 : ((hvd1 * 2 > 9 * hvd0) ? 2 : 0)

[0430] 4) 변수들 avgVar[x][y], 단, x, y = 0..CtbSizeY - 1은 다음과 같이 도출된다:

[0431] varTab[ ] = { 0, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4 }

[0432] avgVar[x][y] = varTab[Clip3(0, 15, (sumOfHV[x >> 2][y >> 2] * 64) >> (3 + BitDepthY))]

[0433] 5) 분류 필터 인덱스 어레이 filtIdx[x][y] 및 전치 인덱스 어레이 transposeIdx[x][y], 단, x = y = 0..CtbSizeY - 1은 다음과 같이 도출된다:

[0434] transposeTable[ ] = { 0, 1, 0, 2, 2, 3, 1, 3 }

[0435] transposeIdx[x][y] = transposeTable[dir1[x][y] * 2 + (dir2[x][y] >> 1)]

[0436] filtIdx[x][y] = avgVar[x][y]

[0437] dirS[x][y]가 0과 동일하지 않을 때, filtIdx[x][y]는 다음과 같이 수정된다:

[0438] filtIdx[x][y] += (((dir1[x][y] & 0x1) << 1) + dirS[x][y]) * 5

[0439] 크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스

[0440] 이 프로세스의 입력들은: 적응 루프 필터링 프로세스 이전의 재구성된 크로마 픽처 샘플 어레이 recPicture, 필터링된 재구성된 크로마 픽처 샘플 어레이 alfPicture, 현재 픽처의 좌측 상단 샘플에 상대적인 현재 크로마 코딩 트리 블록의 좌측 상단 샘플을 명시하는 크로마 위치 (xCtbC, yCtbC)이다.

[0441] 이 프로세스의 출력은 수정된 필터링된 재구성된 크로마 픽처 샘플 어레이 alfPicture이다.

[0442] 현재 크로마 코딩 트리 블록 ctbSizeC의 크기는 다음과 같이 도출된다:

[0443] ctbSizeC = CtbSizeY / SubWidthC

[0444] 필터링된 재구성된 크로마 샘플들 alfPicture[x][y]의 도출을 위해, 현재 크로마 코딩 트리 블록 recPicture[x][y] 내의 각각의 재구성된 크로마 샘플은 다음과 같이 필터링되며, 단, x, y = 0..ctbSizeC - 1이다:

[0445] 크로마 샘플들의 주어진 어레이 recPicture 내의 대응하는 크로마 샘플들 (x, y) 각각에 대한 위치들 (hx, vy)는 다음과 같이 도출된다:

[0446] hx = Clip3(0, pic_width_in_luma_samples / SubWidthC - 1, xCtbC + x)

[0447] vy = Clip3(0, pic_height_in_luma_samples / SubHeightC - 1, yCtbC + y)

[0448] 변수 curr은 다음과 같이 도출된다:

[0449] curr = recPicture[hx, vy]

[0450] 변수 sum은 다음과 같이 도출된다:

[0451] sum = AlfCoeffC[0] * (Clip3(-AlfClipC[0], AlfClipC[0], recPicture[hx, vy + 2]- curr) + Clip3(-AlfClipC[0], AlfClipC[0], recPicture[hx, vy - 2] - curr))

[0452] + AlfCoeffC[1] * (Clip3(-AlfClipC[1], AlfClipC[1], recPicture[hx + 1, vy + 1] - curr) + Clip3(-AlfClipC[1], AlfClipC[1], recPicture[hx - 1, vy - 1] - curr))

[0453] + AlfCoeffC[2] * (Clip3(-AlfClipC[2], AlfClipC[2], recPicture[hx, vy + 1] - curr) + Clip3(-AlfClipC[2], AlfClipC[2], recPicture[hx, vy - 1] - curr))

[0454] + AlfCoeffC[3] * (Clip3(-AlfClipC[3], AlfClipC[3], recPicture[hx - 1, vy + 1] - curr) + Clip3(-AlfClipC[3], AlfClipC[3], recPicture[hx + 1, vy - 1] - curr))

[0455] +
AlfCoeffC[4] * (Clip3(-AlfClipC[4], AlfClipC[4], recPicture[hx + 2, vy] - curr)
+ Clip3(-AlfClipC[4], AlfClipC[4], recPicture[hx - 2, vy] - curr))

[0456] +
AlfCoeffC[5] * (Clip3(-AlfClipC[5], AlfClipC[5], recPicture[hx + 1, vy] - curr)
+ Clip3(-AlfClipC[5], AlfClipC[5], recPicture[hx - 1, vy] - curr))

[0457] + AlfCoeffC[6] * recPicture[hx, vy]

[0458] sum = (sum + 64) >> 7

[0459] 수정된 필터링된 재구성된 크로마 픽처 샘플 alfPicture[xCtbC + x][yCtbC + y]는 다음과 같이 도출된다:

[0460] alfPicture[xCtbC + x][yCtbC + y] = Clip3(0, (1 << BitDepthC) - 1, sum)

[0461] 상기 변형례에서, 클리핑 연산이 이웃 샘플과 필터링되는 샘플 간의 차이에 적용된다(즉, 수학식 7이 사용된다). 수학식 9를 사용하는 다른 변형례에서, 그의 시맨틱스와 필터링 프로세스는 다음과 같이 수정된다(상기 변형의 시맨틱스 및 필터링 프로세스에 기초하여 수정된다).

[0462] - AlfCoeffL[filtIdx][12]는 다음과 같이 도출될 것이다:

[0463] AlfCoeffL[filtIdx][12] = 128- $\sum_k$ (AlfCoeffL[filtIdx][k] << 1), 단, k = 0..11

[0464] - AlfCoeffC[6]은 다음과 같이 도출될 것이다:

[0465] AlfCoeffC[6] = 128 - $\sum_k$ (AlfCoeffC[k] << 1), 단, k = 0..5

[0466] - 0 내지 2^8 - 1(경계 포함)의 범위에 있어야 한다는, AlfCoeffL[filtIdx][12] 및 AlfCoeffC[6]의 비트 깊이에 대한 제한이 더 나은 필터링 성능을 달성하기 위해 0 내지 2^9 - 1의 범위로 또는 심지어 0 내지 2^10 - 1의 범위로 완화된다.

[0467] 이러한 이유는 클리핑 연산들에서, (필터 입력 샘플들이 더 신뢰할 수 있기 때문에, 즉, 일단 비선형 함수가 적용된 경우, 더 나은 필터링을 제공할 가능성이 더 많기 때문에) 더 나은 효율을 위해 이웃 샘플 위치들과 연관된 계수들이 (절댓값에서) 더 높아야 하는 일이 종종 발생하기 때문이다. 그 결과, 중심 계수가 (고정 소수점 표현으로) 256보다 높아야 하도록, 해당 계수들(따라서, 중심 계수를 포함하지 않음)의 합이 (고정 소수점 표현으로) -128보다 낮은 일이 자주 발생한다. 따라서, 중심 계수를 0 내지 2^8 - 1의 기준 범위(normative range)에 있도록 제한하는 것은 인코더가, 계수들의 다이내믹(dynamic)/진폭을 감소시키는 것에 의해, 최적이 아닌 계수들을 찾아야 하므로, 덜 효율적인 필터링을 결과한다는 것을 암시한다. 중심 계수에 대해 10 비트 정밀도의 기준 범위(즉, 0 내지 2^10-1의 범위)를 사용하는 것에 의해, 인코더가 최적이 아닌 필터 계수들을 사용해야 하고 따라서 8 비트 정밀도에 비해 전체 필터링 품질이 향상되는 일이 훨씬 덜 빈번하다. ALF 선형 필터링의 경우, 이웃 필터 입력 샘플들이 종종 더 작은 가중치(즉, 더 작은 계수들)를 갖기 때문에 중심 계수에 대한 8 비트로의 제한은 필터 효율에 그렇게 많은 영향을 주지 않는다. 반면에 클리핑, 즉, ALF 비선형 필터링의 경우, 최적의 필터들은, 일단 클리핑되면, 더 신뢰할 수 있는, 이웃 샘플들에 더 많은 가중치를 부여할 수 있다.

[0468] - '루마 샘플들에 대한 코딩 트리 블록 필터링 프로세스'에서의 변수 sum은 다음과 같이 도출된다:

[0469] sum = filterCoeff[0] * (Clip3(curr-filterClip[0], curr+filterClip[0], recPictureL[hx, vy + 3]) + Clip3(curr-filterClip[0], curr+filterClip[0], recPictureL[hx, vy - 3]))

[0470] + filterCoeff[1] * (Clip3(curr-filterClip[1], curr+filterClip[1], recPictureL[hx + 1, vy + 2]) + Clip3(curr-filterClip[1], curr+filterClip[1], recPictureL[hx - 1, vy - 2]))

[0471] + filterCoeff[2] * (Clip3(curr-filterClip[2], curr+filterClip[2], recPictureL[hx, vy + 2]) + Clip3(curr-filterClip[2], curr+filterClip[2], recPictureL[hx, vy - 2]))

[0472] + filterCoeff[3] * (Clip3(curr-filterClip[3], curr+filterClip[3], recPictureL[hx - 1, vy

+ 2]) + Clip3(curr-filterClip[3], curr+filterClip[3], recPictureL[hx + 1, vy - 2]))

[0473] + filterCoeff[4] * (Clip3(curr-filterClip[4], curr+filterClip[4], recPictureL[hx + 2, vy + 1]) + Clip3(curr-filterClip[4], curr+filterClip[4], recPictureL[hx - 2, vy - 1]))

[0474] + filterCoeff[5] * (Clip3(curr-filterClip[5], curr+filterClip[5], recPictureL[hx + 1, vy + 1]) + Clip3(curr-filterClip[5], curr+filterClip[5], recPictureL[hx - 1, vy - 1]))

[0475] + filterCoeff[6] * (Clip3(curr-filterClip[6], curr+filterClip[6], recPictureL[hx, vy + 1]) + Clip3(curr-filterClip[6], curr+filterClip[6], recPictureL[hx, vy - 1]))

[0476] + filterCoeff[7] * (Clip3(curr-filterClip[7], curr+filterClip[7], recPictureL[hx - 1, vy + 1]) + Clip3(curr-filterClip[7], curr+filterClip[7], recPictureL[hx + 1, vy - 1]))

[0477] + filterCoeff[8] * (Clip3(curr-filterClip[8], curr+filterClip[8], recPictureL[hx - 2, vy + 1]) + Clip3(curr-filterClip[8], curr+filterClip[8], recPictureL[hx + 2, vy - 1]))

[0478] + filterCoeff[9] * (Clip3(curr-filterClip[9], curr+filterClip[9], recPictureL[hx + 3, vy]) + Clip3(curr-filterClip[9], curr+filterClip[9], recPictureL[hx - 3, vy]))

[0479] + filterCoeff[10] * (Clip3(curr-filterClip[10], curr+filterClip[10], recPictureL[hx + 2, vy]) + Clip3(curr-filterClip[10], curr+filterClip[10], recPictureL[hx - 2, vy]))

[0480] + filterCoeff[11] * (Clip3(curr-filterClip[11], curr+filterClip[11], recPictureL[hx + 1, vy]) + Clip3(curr-filterClip[11], curr+filterClip[11], recPictureL[hx - 1, vy]))

[0481] + filterCoeff[12] * recPictureL[hx, vy]

[0482] sum = (sum + 64) >> 7

[0483] - '크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스'에서의 변수 sum은 다음과 같이 도출된다:

[0484] sum = AlfCoeffC[0] * (Clip3(curr-AlfClipC[0], curr+AlfClipC[0], recPicture[hx, vy + 2]) + Clip3(curr-AlfClipC[0], AlfClipC[0], recPicture[hx, vy - 2]))

[0485] + AlfCoeffC[1] * (Clip3(curr-AlfClipC[1], curr+AlfClipC[1], recPicture[hx + 1, vy + 1]) + Clip3(curr-AlfClipC[1], curr+AlfClipC[1], recPicture[hx - 1, vy - 1]))

[0486] + AlfCoeffC[2] * (Clip3(curr-AlfClipC[2], curr+AlfClipC[2], recPicture[hx, vy + 1]) + Clip3(curr-AlfClipC[2], curr+AlfClipC[2], recPicture[hx, vy - 1]))

[0487] + AlfCoeffC[3] * (Clip3(curr-AlfClipC[3], curr+AlfClipC[3], recPicture[hx - 1, vy + 1]) + Clip3(curr-AlfClipC[3], curr+AlfClipC[3], recPicture[hx + 1, vy - 1]))

[0488] + AlfCoeffC[4] * (Clip3(curr-AlfClipC[4], curr+AlfClipC[4], recPicture[hx + 2, vy]) + Clip3(curr-AlfClipC[4], curr+AlfClipC[4], recPicture[hx - 2, vy]))

[0489] + AlfCoeffC[5] * (Clip3(curr-AlfClipC[5], curr+AlfClipC[5], recPicture[hx + 1, vy]) + Clip3(curr-AlfClipC[5], curr+AlfClipC[5], recPicture[hx - 1, vy]))

[0490] + AlfCoeffC[6] * recPicture[hx, vy]

[0491] sum = (sum + 64) >> 7

[0492] 이웃 샘플 값과 중심 샘플 값 사이의 차이에 적용되는 클리핑을 또한 사용하는, (ALF를 적용하는 데 필요한 버퍼 라인들의 수를 감소시키기 위해) 루마 필터들에 대해 5x5 다이아몬드 필터 형상을 사용하는 다른

변형례에서, 표 4의 비선형 ALF 데이터 신택스는 다음과 같이 수정된다(이전에 제시된 ALF 데이터 신택스에 대해 이루어진 변경들은 밑줄도 그어져 있고 굵게도 표시되어 있으며, 반복되는 부분들은 가능한 한 생략된다).

표 5

Alf_data( ) {	
...	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++) {	
if (alf_luma_coeff_flag[sigFiltIdx]) {	
for (j = 0; j < 6; 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)
}	
}	
}	
...	
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++) {	
if (alf_luma_coeff_flag[sigFiltIdx] && alf_luma_filter_clip[sigFiltIdx]) {	
for (j = 0; j < 6; j++) {	
if(AlfCoeffL[filtIdx][j])	
alf_luma_clip_idx[sigFiltIdx][j]	uek(v)
}	
...	

[0493]

[0494] 표 5 - 비선형 ALF 데이터 신택스의 수정된 부분

[0495] 루마 필터들에 대한 ALF 계수들의 수는 13개가 되며, 따라서 마지막 계수 필터 초기화 프로세스는 다음과 같이 된다:

[0496] filtIdx = 0..NumAlfFilters - 1에 대한 마지막 필터 계수들 AlfCoeffL[filtIdx][6]는 다음과 같이 도출된다:

[0497] AlfCoeffL[filtIdx][6] = 128

[0498] AlfCoeffL[filtIdx][j], 단, filtIdx = 0..NumAlfFilters-1, j = 0..11의 값들이 -2^7 내지 2^7 - 1(경계 포함)의 범위에 있어야 한다는 것과 AlfCoeffL[filtIdx][6]의 값들이 0 내지 2^8 - 1(경계 포함)의 범위에 있어야 한다는 것이 비트스트림 적합성의 요구사항이다.

[0499] "루마 샘플들에 대한 코딩 트리 블록 필터링 프로세스"의 경우, '루마 샘플들에 대한 ALF 전치 및 필터 인덱스에 대한 도출 프로세스' 절에서의 필터 인덱스에 대한 도출 프로세스가 다음과 같이 수정된다(이전에 제시된 ALF 데이터 신택스에 대해 이루어진 변경들은 밑줄도 그어져 있고 굵게도 표시되어 있으며, 반복되는 부분들은 가능한 한 생략된다):

[0500] '루마 샘플들에 대한 ALF 전치 및 필터 인덱스에 대한 도출 프로세스' 절에서의 필터 인덱스에 대한 도출 프로세스는 위치 (xCtb, yCtb) 및 재구성된 루마 픽처 샘플 어레이 recPictureL을 입력들로서 사용하여 호출되고, filtIdx[x][y] 및 transposeIdx[x][y], 단 x, y = 0..CtbSizeY - 1이 출력들로서 얻어진다.

[0501] 필터링된 재구성된 루마 샘플들 alfPictureL[x][y]의 도출을 위해, 현재 루마 코딩 트리 블록 recPictureL[x][y] 내의 각각의 재구성된 루마 샘플은 다음과 같이 필터링되며, 단, x, y = 0..CtbSizeY - 1이다:

[0502] 루마 필터 계수들의 어레이 f[j] 및 filtIdx[x][y]에 의해 명시되는 필터에 대응하는 루마 필터 클리핑의 어레이 c[j]는 다음과 같이 도출되며, 단, j = 0..6이다:

[0503] f[j] = AlfCoeffL[filtIdx[x][y]][j]

[0504] c[j] = AlfClipL[filtIdx[x][y]][j]

[0505] 루마 필터 계수들 filterCoeff 및 필터 클리핑 값들 filterClip은 다음과 같이 transposeIdx[x][y]에 따라 도출된다:

[0506] transposeIndex[x][y] = = 1인 경우,

[0507] filterCoeff[ ] = { f[4], f[1], f[5], f[3], f[0], f[2], f[6] }

[0508] filterClip[ ] = { c[4], c[1], c[5], c[3], c[0], c[2], c[6] }

[0509] 그렇지 않고, transposeIndex[x][y] = = 2인 경우,

[0510] filterCoeff[ ] = { f[0], f[3], f[2], f[1], f[4], f[5], f[6] }

[0511] filterClip[ ] = { c[0], c[3], c[2], c[1], c[4], c[5], c[6] }

[0512] 그렇지 않고, transposeIndex[x][y] = = 3인 경우,

[0513] filterCoeff[ ] = { f[4], f[3], f[5], f[1], f[0], f[2], f[6] }

[0514] filterClip[ ] = { c[4], c[3], c[5], c[1], c[0], c[2], c[6] }

[0515] 그렇지 않은 경우,

[0516] filterCoeff[ ] = { f[0], f[1], f[2], f[3], f[4], f[5], f[6] }

[0517] filterClip[ ] = { c[0], c[1], c[2], c[3], c[4], c[5], c[6] }

[0518] 루마 샘플들의 주어진 어레이 recPicture 내의 대응하는 루마 샘플들 (x, y) 각각에 대한 위치들 (hx, vy)는 다음과 같이 도출된다:

[0519] hx = Clip3(0, pic_width_in_luma_samples - 1, xCtb + x)

[0520] vy = Clip3(0, pic_height_in_luma_samples - 1, yCtb + y)

[0521] 변수 curr은 다음과 같이 도출된다: curr = recPictureL[hx, vy]

[0522] 변수 sum은 다음과 같이 도출된다:

[0523] sum = filterCoeff[0] * (Clip3(-filterClip[0], filterClip[0], recPictureL[hx, vy + 2] - curr) + Clip3(-filterClip[0], filterClip[0], recPictureL[hx, vy - 2] - curr))

[0524] + filterCoeff[1] * (Clip3(- filterClip[1], filterClip[1], recPictureL[hx + 1, vy + 1] - curr) + Clip3(- filterClip[1], filterClip[1], recPictureL[hx - 1, vy - 1] - curr))

[0525] + filterCoeff[2] * (Clip3(-filterClip[2], filterClip[2], recPictureL[hx, vy + 1] - curr) + Clip3(-filterClip[2], filterClip[2], recPictureL[hx, vy - 1] - curr))

[0526] + filterCoeff[3] * (Clip3(-filterClip[3], filterClip[3], recPictureL[hx - 1, vy + 1] - curr) + Clip3(-filterClip[3], filterClip[3], recPictureL[hx + 1, vy - 1] - curr))

[0527] + filterCoeff[4] * (Clip3(-filterClip[4], filterClip[4], recPictureL[hx + 2, vy] - curr) + Clip3(-filterClip[4], filterClip[4], recPictureL[hx - 2, vy] - curr))

[0528] + filterCoeff[5] * (Clip3(-filterClip[5], filterClip[5], recPictureL[hx + 1, vy] - curr) + Clip3(-filterClip[5], filterClip[5], recPictureL[hx - 1, vy] - curr))

[0529] + filterCoeff[6] * recPictureL[hx, vy]

[0530] sum = (sum + 64) >> 7

[0531] 수정된 필터링된 재구성된 루마 픽처 샘플 alfPictureL[xCtb + x][yCtb + y]는 다음과 같이 도출된

다:

[0532] alfPictureL[xCtb + x][yCtb + y] = Clip3(0, (1 << BitDepthY) - 1, sum)

[0533] 이웃들과 중심 간의 차이를 사용하지 않는 변형례들에서, AlfCoeffL[filtIdx][6] = 128- $\sum_k$ (AlfCoeffL[filtIdx][k] << 1), 단, k = 0..5가 사용된다. 필터링의 경우, 필터링에서의 클리핑 파라미터들에 대한 유사한 변경들이 또한 이루어진다.

[0534] 다른 변형례에서, 필터에 대해 7x7 다이아몬드 형상을 사용하지만 루마 필터들에 대해 도 16a의 클리핑 패턴(1601)을 사용하고 크로마 필터에 대해 도 16b의 클리핑 패턴(1605)을 사용하여, 표 4의 비선형 ALF 데이터 신택스가 다음과 같이 수정된다(표 4의 이전에 제시된 ALF 데이터 신택스에 대해 이루어진 변경들은 밑줄도 그어져 있고 굵게도 표시되어 있으며, 반복되는 부분들은 가능한 한 생략된다).

표 6

[0535]

Alf_data( ) {	
...	
if(alf_luma_clip) {	
alf_luma_clip_min_eg_order_minus1	ue(v)
for(i = 0; i < 3; i++)	
alf_luma_clip_eg_order_increase_flag[i]	u(1)
for(sigFiltIdx = 0; sigFiltIdx <= alf_luma_num_filters_signalled_minus1; sigFiltIdx++) {	
if (alf_luma_coeff_flag[sigFiltIdx] && alf_luma_filter_clip[sigFiltIdx]) {	
for (j = 0; j < 12; j++) {	
if(AlfCoeff$_L$[filtIdx][j] **&& AlfClipPatL[j]**)	
alf_luma_clip_idx[sigFiltIdx][j]	uek(v)
}	
}	
}	
}	
if (alf_chroma_idc > 0 && alf_chroma_clip) {	
alf_chroma_clip_min_eg_order_minus1	ue(v)
for(i = 0; i < 2; i++)	
alf_chroma_clip_eg_order_increase_flag[i]	u(1)
for(j = 0; j < 6; j++) { if(AlfCoeff$_C$[j] **&& AlfClipPatC[j]**)	
alf_chroma_clip_idx[j]	uek(v)
}	
}	
}	

[0536] 표 6 - 비선형 ALF 데이터 신택스의 수정된 부분

[0537] AlfClipPatL[]은 루마 필터들에 대한 상수 클리핑 패턴 테이블이며 다음과 같이 설정된다:

[0538] AlfClipPatL[] = { 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1 }

[0539] AlfClipPatC[]는 크로마 필터에 대한 상수 클리핑 패턴 테이블이며 다음과 같이 설정된다:

[0540] AlfClipPatC[] = { 1, 1, 0, 1, 1, 0 }

[0541] 시맨틱스의 일부가 다음과 같이 수정된다:

[0542] - '루마 샘플들에 대한 코딩 트리 블록 필터링 프로세스'에서의 변수 sum은 다음과 같이 도출된다:

[0543] sum = filterCoeff[0] * (recPictureL[hx, vy + 3] - curr + recPictureL[hx, vy - 3] - curr)

[0544] + filterCoeff[1] * (recPictureL[hx + 1, vy + 2] - curr) + recPictureL[hx - 1, vy - 2] - curr)

[0545] + filterCoeff[2] * (Clip3(-filterClip[2], filterClip[2], recPictureL[hx, vy + 2] - curr) + Clip3(-filterClip[2], filterClip[2], recPictureL[hx, vy - 2] - curr))

[0546] + filterCoeff[3] * (recPictureL[hx - 1, vy + 2] - curr + recPictureL[hx + 1, vy - 2] - curr)

[0547] + filterCoeff[4] * (recPictureL[hx + 2, vy + 1] - curr + recPictureL[hx - 2, vy - 1] - curr)

[0548] + filterCoeff[5] * (recPictureL[hx + 1, vy + 1] - curr + recPictureL[hx - 1, vy - 1] - curr)

[0549] + filterCoeff[6] * (Clip3(-filterClip[6], filterClip[6], recPictureL[hx, vy + 1] - curr) + Clip3(-filterClip[6], filterClip[6], recPictureL[hx, vy - 1] - curr))

[0550] + filterCoeff[7] * (recPictureL[hx - 1, vy + 1] - curr + recPictureL[hx + 1, vy - 1] - curr)

[0551] + filterCoeff[8] * (recPictureL[hx - 2, vy + 1] - curr + recPictureL[hx + 2, vy - 1] - curr)

[0552] + filterCoeff[9] * (recPictureL[hx + 3, vy] - curr) + recPictureL[hx - 3, vy] - curr)

[0553] + filterCoeff[10] * (Clip3(-filterClip[10], filterClip[10], recPictureL[hx + 2, vy] - curr) + Clip3(-filterClip[10], filterClip[10], recPictureL[hx - 2, vy] - curr))

[0554] + filterCoeff[11] * (Clip3(-filterClip[11], filterClip[11], recPictureL[hx + 1, vy] - curr) + Clip3(-filterClip[11], filterClip[11], recPictureL[hx - 1, vy] - curr))

[0555] + filterCoeff[12] * recPictureL[hx, vy]

[0556] sum = (sum + 64) >> 7

[0557] - '크로마 샘플들에 대한 코딩 트리 블록 필터링 프로세스'에서의 변수 sum은 다음과 같이 도출된다:

[0558] sum = AlfCoeffC[0] * (Clip3(-AlfClipC[0], AlfClipC[0], recPicture[hx, vy + 2]- curr) + Clip3(-AlfClipC[0], AlfClipC[0], recPicture[hx, vy - 2] - curr))

[0559] + AlfCoeffC[1] * (Clip3(-AlfClipC[1], AlfClipC[1], recPicture[hx + 1, vy + 1] - curr) + Clip3(-AlfClipC[1], AlfClipC[1], recPicture[hx - 1, vy - 1] - curr))

[0560] + AlfCoeffC[2] * (recPicture[hx, vy + 1] - curr + recPicture[hx, vy - 1] - curr)

[0561] + AlfCoeffC[3] * (Clip3(-AlfClipC[3], AlfClipC[3], recPicture[hx - 1, vy + 1] - curr) + Clip3(-AlfClipC[3], AlfClipC[3], recPicture[hx + 1, vy - 1] - curr))

[0562] + AlfCoeffC[4] * (Clip3(-AlfClipC[4], AlfClipC[4], recPicture[hx + 2, vy] - curr) + Clip3(-AlfClipC[4], AlfClipC[4], recPicture[hx - 2, vy] - curr))

[0563] + AlfCoeffC[5] * (recPicture[hx + 1, vy] - curr + recPicture[hx - 1, vy] - curr)

[0564] + AlfCoeffC[6] * recPicture[hx, vy]

[0565] 다른 클리핑 패턴들을 사용하는 다른 변형례들이 감소된 수의 클리핑(즉, 감소된 수의 클리핑되는 위치들, 예를 들면, 루마 성분의 경우, 클리핑이 현재(중심) 샘플 값 "curr"에 적용되지 않으며 따라서 수학식 17 및 도 7에서와 같이 "f[12]", "c[12]", "filterCoeff[12]"이 없고, "sum = curr + ((sum+64)>>7임)"을 사용하여 그리고/또는 이웃 샘플들과 중심 샘플 간의 차이를 사용하지 않고, 전술한 변형례는 물론 다른 변형례들로부터 용이하

게 도출될 수 있음이 이해된다.

[0566] 도 8은 본 발명의 실시예에 따른 도 7에서의 필터의 비선형 부분이 디스에이블될 때 획득되는 선형 필터의 블록 다이어그램이다. 실시예에 따르면, ALF가 활성이고 비선형 ALF가 비활성일 때, 선형 ALF가 필터링에 사용된다. 그러한 실시예의 변형례에서, 수학식 4로 정의되고 도 6에 예시된 바와 같은 선형 ALF를 사용하는 대신에, 수학식 7로 정의되고 도 7에 예시된 바와 같은 비선형 ALF이 사용되고, 수학식 6으로 정의되고 도 8에 예시된 바와 같은 선형 필터링이 수행된다. 하드웨어 설계에서, 이는 비선형 필터와 선형 필터 간에 공통 회로 부분을 공유하는 것을 가능하게 한다. 도 7과 도 8을 비교할 때, min/max 연산들이 단지 우회되어, 샘플 값으로 직접 진행된다. 이 접근법을 사용하는 하나의 다른 장점은 인코더가 그의 필터들을 설계할 때, 인코더가 각각의 필터의 모든 필터 계수들의 합이 1이도록 보장할 필요가 없다는 것이다. 이 설계의 결과는, 현재 샘플을 제외하고, 계수당 하나의 감산을 추가하고, 현재 샘플에 대해 하나의 곱셈을 제거하는 것이다.

[0567] 도 7의 설계는 또한, 클리핑 파라미터가 취할 수 있는 최댓값이 충분히 낮을 때, 곱셈을 구현하기 위한 하드웨어 비트들을 절감하는 장점이 있다.

[0568] 실시예에 따르면, 도 5에서의 슬라이스 헤더에서 시그널링되는 ALF의 신택스 요소들은 더 이상 슬라이스 헤더에서 시그널링되지 않고, 그 대신에 프레임 레벨에서, 예를 들어, 픽처 헤더에서 시그널링된다. 그러면, 슬라이스 헤더에는 프레임 레벨에서 제공되는 파라미터들이 슬라이스에서 직접 사용된다는 것; 또는 프레임 레벨에서 제공되는 파라미터들이 슬라이스 레벨에서 "리프레시(refresh)"(즉, 업데이트)되어야 한다는 것을 시그널링하기 위한 하나 이상의 플래그/신택스 요소가 있다. 슬라이스 레벨에서의 리프레시는 슬라이스 헤더에서 행해진다. 필터 계수들 및/또는 클리핑 파라미터들의 리프레시는 프레임 레벨에서 표시되는 값들과 슬라이스에서 사용할 값들 사이의 차이를 인코딩하는 것에 의해 행해진다. 실시예에 따르면, 슬라이스 헤더에서의 추가 신택스 요소는 필터 계수들 및/또는 클리핑 파라미터들의 값들이 차이로서 슬라이스 헤더에 인코딩되는지 또는 그들의 값들이 인코딩되는지를 나타낸다.

[0569] 실시예에 따르면, 슬라이스 레벨에서(슬라이스 헤더에) 정의되는 필터 계수들 및/또는 클리핑 파라미터들은 (예를 들어, 각각의 CTU에 대해) 더 미세한 입도 레벨로 업데이트될 수 있다. 신택스 요소가 업데이트가 수행되어야 함을 나타낼 때 업데이트가 수행되어야 하는지를 나타내는 신택스 요소가 슬라이스 데이터는 물론 업데이트 데이터에 엔트로피 코딩된다. 일 실시예에서, 인코딩된 비트스트림에서, 각각의 CTU에 대해, 인코딩된 슬라이스 데이터는 루마 필터들/클리핑 파라미터들이 업데이트되어야 하는지 여부를 나타내는 하나의 엔트로피 코딩된 플래그를 포함한다. 플래그가 업데이트가 행해져야 함을 나타내는 경우, 필터 계수들의 각각의 테이블에 대해, 엔트로피 코딩된 플래그는 필터가 업데이트되어야 하는지를 나타낸다. 필터가 업데이트되어야 하는 경우, 계수들에 대한 오프셋들(슬라이스에서의 기준 값들과 차이 또는 대안적으로 동일한 슬라이스에 속하는 이전에 인코딩된 CTU에서의 기준 값들과의 차이) 및/또는 클리핑 파라미터들에 대한 오프셋들이 엔트로피 코딩된다.

[0570] 실시예에 따르면, 인코더 관점에서, 도 9를 참조하여 기술된 VTM-3.0의 ALF 인코딩 프로세스의 주요 논리적 단계들이 유지된다. 주요 변경들은 필터들을 구축하기 위해 추출되는 통계의 특성 및 필터가 결정되는 방식이다. 간단한 위너 필터 계산 대신에, 최상의 클리핑 파라미터들 및 연관된 비선형 위너 필터 계수들을 찾기 위해 클리핑 파라미터들이 반복적으로 최적화된다.

[0571] 실시예에 따르면, 하나의 클리핑 파라미터가 취할 수 있는 값들의 수에 대한 제한인 N_K를 정의한다. N_K는 작을 수 있으며, 예를 들어, N_K = 12 또는 훨씬 더 작을 수 있다. 이어서, 공분산 및 교차 공분산 행렬 통계를 계산하는 대신에, 단계(902)에서, 인코더는 (하나의 공분산 행렬 통계 및 하나의 교차 공분산 행렬 통계 대신에) N_K x N_K 클리핑-공분산 행렬들 통계의 2-엔트리 테이블(two-entry table) E 및 N_K 클리핑-교차 공분산 행렬들 통계의 1-엔트리 테이블(one-entry table) y를 구축한다.

[0572] 클리핑-공분산 행렬 $E_c[a][b]$는 $\frac{1}{N}X^a X^{bT}$ 로부터 추정되는 공분산 행렬이며, 여기서 X^m은 C_m 번째 클리핑 값을 사용할 때 입력들의 N개의 실현들(관찰된 값들)을 포함한다. 예를 들어, 도 4b의 형상들을 사용하면, X_i^m는 주어진 샘플 위치에 대해 획득되는 X^m의 i 번째 열이고, $X_{i,j}^m$는: i < N_c(도 4b의 루마 필터에 대한 N_c = 12는 필터 계수들의 수에서 1을 뺀 것임)인 경우, 형상에서의 인덱스 i를 갖는 2개의 대칭적인 이웃에 대해, m 번째

클리핑 값을 사용하여 (j 번째 필터링되는 샘플의) 이웃 샘플과 j 번째 필터링되는 샘플 사이의 차이를 클리핑한 결과들의 합; 및 i = 12인 경우, j 번째 필터링된 샘플을 포함하고; 여기서 $X_{i,j}^{m}$는 행렬 X^{m}에서 i 행과 j 열에 있는 값이다.

[0573] 클리핑-교차 공분산 행렬 $y_c[c]$는 원하는 필터 출력(즉, 원래 샘플들)의 실현들과 X^{c}의 실현들 간의 교차 공분산 행렬이다.

[0574] 따라서 $c_0, \ldots, c_{N_c-1}$과 동일한 클리핑 파라미터들을 사용하여 필터를 구축/테스트해야 할 때, $C = [c_0, \ldots, c_{N_c-1}, 0]$을 정의한다(예를 들어, 루마 필터에 대한 N_c = 12는 필터 계수들의 수에서 1을 뺀 것이고, 마지막 값이 0과 동일하면 편리하며, 다른 값이 사용될 수 있고 중심 계수 12가 클리핑되지 않기 때문에 동일한 결과들을 가질 것이다). 각각의 $i < N_c$, $j < N_c$에 대해 $E[i,j] = E_c[c[i]][c[j]][i,j]$이고, 각각의 $k < N_c$에 대해 $y[k] = y_c[c[k]][k]$이도록 이러한 클리핑 파라미터들로부터 공분산 행렬 E와 교차 공분산 행렬 y를 계산한다. 이어서, 선형 ALF에 대해 이전에 설명된 바와 같이, 이러한 2개의 행렬을 사용하여 위너 필터를 구축한다.

[0575] 단계(904)에서, 공분산과 교차 공분산 행렬들의 통계를 결합시키는 대신에, 인코더는 각각의 클리핑-공분산과 클리핑-교차 공분산 행렬들의 통계를 결합시킨다(여기서도, 결합은 통계를 합산하는 것에 의해 행해진다).

[0576] 단계(905)에서, 최적화된 필터 그룹들(즉, 최적화된 병합된 필터)을 찾는 데 사용되는 방법은 이전과 유사하다. 주요 차이점은, 각각의 테스트된/비교된 필터에 대해, 단지 하나의 위너 필터를 계산하는 대신에, (통계 행렬들에 대해 계산되는) 필터의 출력 오차를 최소화하기 위해, 클리핑 파라미터들이 반복적으로 최적화되고, 각각의 단계가 특정 클리핑 파라미터들에 대한 위너 필터를 계산한다는 것이다. 간단한 최적화 알고리즘은 모든 클리핑 파라미터 값들(c_i)이 0(또는 대안적으로 중앙 클리핑 값(median clipping value) 또는 임의의 기본 값)으로 설정되는 것으로 시작되고, 이어서 필터를 개선시키면서(즉, 출력 오차를 감소시키면서) 루프들에서 이하의 단계들을 따라간다: 각각의 클리핑 파라미터 값에 대해, 이용 가능한/허용된 경우(즉, 인덱스 < 최대 인덱스 값인 경우), 이 클리핑 파라미터 값으로 계산되는 위너 필터가 더 나은 경우, 다음 클리핑 파라미터 값을 취하거나, 또는 이용 가능한/허용된 경우(즉, 인덱스 > 0인 경우), 이 클리핑 파라미터 값으로 계산되는 위너 필터가 더 나은 경우, 이전 클리핑 파라미터 값을 취한다. 2개의 필터를 그룹화할 때, 한 가지 전략은 각각의 클리핑 파라미터 값을 2개의 필터의 각각의 클리핑 파라미터 값의 평균으로 설정하는 것으로 최적화를 시작하는 것일 수 있다.

[0577] 단계(906, 907, 908, 909, 910 및 911)에 대한 실제 변경은 없다.

[0578] 단계(912)에서, 결합된 통계 변경은 단계(904)에 대해 기술된 것과 동일하다. 단계(913)에서의 크로마 필터에 대한 결정은 단계(905)에 대해 기술된 필터 최적화에 대응한다. ALF 파라미터들이 각각의 필터에 대해 결정되는 클리핑 파라미터들을 포함하는 단계(918)를 제외하고는 다른 단계들은 변경되지 않는다.

[0579] "정상(normal)" ALF를 사용하는 VTM-3.0에서, 인코더가 부동 소수점 값들로부터 정수 필터 계수들을 도출할 때, 인코더는 먼저 부동 소수점 값을 고정 소수점 값으로 양자화하는 것을 수행한다: 각각의 필터 계수에 대해, 고정 소수점 정밀도에 의한 필터 계수 곱셈을 가장 가까운 정수 값으로 반올림한다는 점에 유의한다. 이어서, 인코더는 (반올림 오차를 보상하기 위해) 필터 계수들을 반복적으로 조정하는 것에 의해 필터 효율을 개선시키려고 한다. 인코더는 또한 (곱셈에서의 중심 샘플의 사용이 결국 너무 많은 비트들을 사용하지 않도록 보장하기 위해) 모든 필터 계수들의 합이 (고정 소수점 정밀도로 곱해지는) 1이고 중심 샘플에 대한 필터 계수가 최댓값을 초과하지 않도록 보장해야 한다.

[0580] 이웃하는 샘플들과의 차이가 필터에 대한 입력으로서 사용되는 본 발명의 실시예들에서, 중심 계수가 항상 1이기 때문에, 중심 계수가 최댓값을 초과하지 않도록 보장할 필요가 없다. 또한, 모든 필터 계수들의 합의 값을 종종 추적할 필요가 없다.

[0581] 이러한 실시예들의 장점은 코딩 효율 개선이다.

[0582] 이러한 실시예들의 설명은 루마 및 크로마 성분을 언급하지만 단일 루마 성분 또는 RGB 성분들과 같은 다른 성분들에 쉽게 적응될 수 있다.

[0583] 전술한 실시예 또는 변형례에서, 본 발명은 비선형 필터링 효과를 달성하기 위해 필터 입력들에 대해 비선형 함수들을 사용하도록 도 3a 및 도 4a에서의 304 및 407에서의 필터링을 수정하는 것을 결과한다. 도 2는 또한 비선형 함수들에 대한 추가 파라미터들을 시그널링하기 위한 새로운 신택스 요소들을 추가하도록 수정된다(실시예의 예는 도 5를 참조하여 기술되었다). 암시적으로, 302 및 402에서의 입력 필터링 파라미터들은 비선형 함수들에 대한 추가 파라미터들을 추가로 포함하도록 수정된다. 마지막으로, 303 및 406에서의 필터 도출은 필터들에 대한 비선형 함수들을 도출하도록(즉, "비선형 필터들"을 도출하도록) 수정된다. 따라서, 대부분의 상황에서, 출력 이미지 부분(305 및 408)은 VTM-3.0에서 생성된 것들과 동일하지 않으며, 동등한/비교 가능한 입력 샘플들에 대해 더 높은 품질을 갖고/갖거나 출력 이미지 품질과 코딩 효율 사이의 더 나은 절충을 달성한다. 다른 실시예/변형례에서, ALF 자체가 전술한 비선형 함수들 중 임의의 것을 사용하여 필터링하게 동작하도록 수정된다는 것이 이해된다.

[0584] 본 발명의 실시예들의 구현

[0585] 전술한 실시예들 중 하나 이상은 하나 이상의 전술한 실시예의 방법 단계들을 수행하는 인코더 또는 디코더의 형태로 구현될 수 있다. 이하의 실시예들은 그러한 구현들을 예시한다.

[0586] 예를 들어, 전술한 실시예들 중 임의의 것에 따른 적응 루프 필터는 도 10에서의 인코더에 의해 수행되는 포스트 필터링(9415) 또는 도 11에서의 디코더에 의해 수행되는 포스트 필터링(9567)에서 사용될 수 있다.

[0587] 도 10은 본 발명의 실시예에 따른 인코더의 블록 다이어그램를 예시한다. 인코더는 연결된 모듈들에 의해 표현되며, 각각의 모듈은 본 발명의 하나 이상의 실시예에 따른 이미지들의 시퀀스의 이미지를 인코딩하는 적어도 하나의 실시예를 구현하는 방법의 적어도 하나의 대응하는 단계를, 예를 들어, 디바이스의 중앙 프로세싱 유닛(CPU)에 의해 실행되는 프로그래밍 명령어들의 형태로, 구현되도록 구성된다.

[0588] 원래 디지털 이미지들(i0 내지 in)의 시퀀스(9401)는 인코더(9400)에 의해 입력으로서 수신된다. 각각의 디지털 이미지는, 때때로 픽셀들이라고도 지칭되는(이후부터, 픽셀들이라고 지칭됨), 샘플들의 세트에 의해 표현된다. 인코딩 프로세스의 구현 이후에 인코더(9400)에 의해 비트스트림(9410)이 출력된다. 비트스트림(9410)은 슬라이스들과 같은 복수의 인코딩 유닛들 또는 이미지 부분들에 대한 데이터를 포함하고, 각각의 슬라이스는 슬라이스를 인코딩하는 데 사용되는 인코딩 파라미터들의 인코딩 값들을 전송하기 위한 슬라이스 헤더, 및 인코딩된 비디오 데이터를 포함하는 슬라이스 보디(slice body)를 포함한다.

[0589] 입력 디지털 이미지들(i0 내지 in)(9401)은 모듈(9402)에 의해 픽셀 블록들로 분할된다. 블록들은 이미지 부분들에 대응하고 가변 크기들일 수 있다(예를 들면, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128 픽셀들 및 여러 직사각형 블록 크기들이 또한 고려될 수 있다). 각각의 입력 블록에 대해 코딩 모드가 선택된다. 두 가지 계열의 코딩 모드들, 즉 공간 예측 코딩(인트라 예측)에 기초한 코딩 모드들과 시간 예측에 기초한 코딩 모드들(인터 코딩, MERGE, SKIP)이 제공된다. 가능한 코딩 모드들이 테스트된다.

[0590] 모듈(9403)은, 인코딩될 주어진 블록이 인코딩될 상기 블록의 이웃의 픽셀들로부터 계산되는 예측자에 의해 예측되는, 인트라 예측 프로세스를 구현한다. 인트라 코딩이 선택되는 경우 잔차를 제공하기 위해, 선택된 인트라 예측자의 표시 및 주어진 블록과 그의 예측자 간의 차이가 인코딩된다.

[0591] 시간 예측은 모션 추정 모듈(9404) 및 모션 보상 모듈(9405)에 의해 구현된다. 먼저, 참조 이미지들(9416)의 세트 중에서 참조 이미지가 선택되고, 인코딩될 주어진 블록에 (픽셀 값 유사도 면에서 가장 가까운) 최근접 영역인, 참조 영역 또는 이미지 부분이라고도 불리는, 참조 이미지의 일 부분이 모션 추정 모듈(9404)에 의해 선택된다. 이어서, 모션 보상 모듈(9405)은 선택된 영역을 사용하여 인코딩될 블록을 예측한다. 잔차 블록/데이터라고도 불리는, 선택된 참조 영역과 주어진 블록 간의 차이가 모션 보상 모듈(9405)에 의해 계산된다. 선택된 참조 영역은 모션 정보(예를 들면, 모션 벡터)를 사용하여 표시된다.

[0592] 따라서, 이들 경우(공간 및 시간 예측) 둘 모두에서, SKIP 모드에 있지 않을 때 원래 블록으로부터 예측자를 감산하는 것에 의해 잔차가 계산된다.

[0593] 모듈(9403)에 의해 구현되는 INTRA 예측에서, 예측 방향이 인코딩된다. 모듈들(9404, 9405, 9416, 9418, 9417)에 의해 구현되는 인터 예측에서, 시간 예측을 위해 그러한 모션 벡터를 식별해 주기 위한 적어도 하나의 모션 벡터 또는 정보(데이터)가 인코딩된다. 인터 예측이 선택되는 경우 모션 벡터 및 잔차 블록에 관련된 정

보가 인코딩된다. 비트레이트를 더욱 감소시키기 위해, 모션이 동질적이라고 가정하면, 모션 벡터 예측자에 대한 차이에 의해 모션 벡터가 인코딩된다. 모션 정보 예측자 후보들의 세트로부터의 모션 벡터 예측자들은 모션 벡터 예측 및 코딩 모듈(9417)에 의해 모션 벡터들 필드(9418)로부터 획득된다.

[0594] 인코더(9400)는 레이트-왜곡 기준과 같은 인코딩 비용 기준을 적용하는 것에 의해 코딩 모드를 선택하기 위한 선택 모듈(9406)을 추가로 포함한다. 중복성을 더욱 감소시키기 위해, (DCT와 같은) 변환이 변환 모듈(9407)에 의해 잔차 블록에 적용되고, 획득되는 변환된 데이터는 이어서 양자화 모듈(9408)에 의해 양자화되고 엔트로피 인코딩 모듈(9409)에 의해 엔트로피 인코딩된다. 마지막으로, SKIP 모드에 있지 않고 선택된 코딩 모드가 잔차 블록의 인코딩을 필요로 할 때, 인코딩되고 있는 현재 블록의 인코딩된 잔차 블록은 비트스트림(9410)에 삽입된다.

[0595] 인코더(9400)는 또한 후속 이미지들의 모션 추정을 위한 참조 이미지(예를 들면, 참조 이미지들/픽처들(9416) 내의 참조 이미지들)를 생성하기 위해 인코딩된 이미지의 디코딩을 수행한다. 이것은 인코더 및 비트스트림을 수신하는 디코더가 동일한 참조 프레임들(예를 들면, 재구성된 이미지들 또는 재구성된 이미지 부분들이 사용됨)을 가질 수 있게 한다. 역 양자화(inverse quantization)("역양자화(dequantization)") 모듈(9411)은 양자화된 데이터의 역 양자화("역양자화")를 수행하며, 이어서 역변환 모듈(9412)에 의해 수행되는 역변환이 뒤따른다. 인트라 예측 모듈(9413)은 예측 정보를 사용하여 주어진 블록에 대해 어느 예측자를 사용할지를 결정하고, 모션 보상 모듈(9414)은 모듈(9412)에 의해 획득되는 잔차를 참조 이미지들(9416)의 세트로부터 획득되는 참조 영역에 실제로 가산한다. 이어서, 픽셀들의 재구성된 프레임(이미지 또는 이미지 부분들)을 필터링하여 참조 이미지들(9416)의 세트에 대한 다른 참조 이미지를 획득하기 위해 모듈(9415)에 의해 포스트 필터링이 적용된다.

[0596] 도 11은 본 발명의 실시예에 따른, 인코더로부터 데이터를 수신하는 데 사용될 수 있는 디코더(9560)의 블록 다이어그램을 예시한다. 디코더는 연결된 모듈들에 의해 표현되며, 각각의 모듈은 디코더(9560)에 의해 구현되는 방법의 대응하는 단계를, 예를 들어, 디바이스의 CPU에 의해 실행될 프로그래밍 명령어들의 형태로, 구현되도록 구성된다.

[0597] 디코더(9560)는 인코딩된 유닛들(예를 들면, 이미지 부분, 블록 또는 코딩 유닛에 대응하는 데이터)을 포함하는 비트스트림(9561)을 수신하고, 각각의 인코딩된 유닛은 인코딩 파라미터들에 대한 정보를 포함하는 헤더 및 인코딩된 비디오 데이터를 포함하는 보디로 구성된다. 도 10과 관련하여 설명된 바와 같이, 인코딩된 비디오 데이터는 엔트로피 인코딩되고, 모션 정보(예를 들면, 모션 벡터 예측자들의 인덱스들)는, 주어진 이미지 부분(예를 들면, 블록 또는 CU)에 대해, 미리 결정된 수의 비트들에 인코딩된다. 수신된 인코딩된 비디오 데이터는 모듈(9562)에 의해 엔트로피 디코딩된다. 잔차 데이터는 이어서 모듈(9563)에 의해 역양자화되고, 이어서 픽셀 값들을 획득하기 위해 모듈(9564)에 의해 역변환이 적용된다.

[0598] 코딩 모드를 나타내는 모드 데이터가 또한 엔트로피 디코딩되고, 그 모드에 기초하여, 이미지 데이터의 인코딩된 블록들(유닛들/세트들/그룹들)에 대해 INTRA 유형 디코딩 또는 INTER 유형 디코딩이 수행된다. INTRA 모드의 경우에, INTRA 예측자는 비트스트림에 명시되는 인트라 예측 모드에 기초하여 인트라 예측 모듈(9565)에 의해 결정된다(예를 들면, 인트라 예측 모드는 비트스트림에 제공되는 데이터를 사용하여 결정 가능하다). 모드가 INTER 모드인 경우, 인코더에 의해 사용되는 참조 영역을 찾기(식별하기) 위해 비트스트림으로부터 모션 예측 정보가 추출/획득된다. 모션 예측 정보는, 예를 들어, 참조 프레임 인덱스 및 모션 벡터 잔차를 포함한다. 모션 벡터를 획득하기 위해 모션 벡터 디코딩 모듈(9570)에 의해 모션 벡터 예측자가 모션 벡터 잔차에 가산된다.

[0599] 모션 벡터 디코딩 모듈(9570)은 모션 예측에 의해 인코딩되는 각각의 이미지 부분(예를 들면, 현재 블록 또는 CU)에 대해 모션 벡터 디코딩을 적용한다. 일단 현재 블록에 대한 모션 벡터 예측자의 인덱스가 획득되면, 이미지 부분(예를 들면, 현재 블록 또는 CU)과 연관된 모션 벡터의 실제 값이 디코딩되어 모듈(9666)에 의해 모션 보상을 적용하는 데 사용될 수 있다. 모듈(9566)이 모션 보상을 수행할 수 있도록, 디코딩된 모션 벡터에 의해 표시되는 참조 이미지 부분이 참조 이미지들(9568)의 세트로부터 추출/획득된다. 후속적으로 디코딩되는 모션 벡터들의 예측에 사용되기 위해 모션 벡터 필드 데이터(9571)가 디코딩된 모션 벡터로 업데이트된다.

[0600] 마지막으로, 디코딩된 블록이 획득된다. 적절한 경우, 포스트 필터링 모듈(9567)에 의해 포스트 필터링이 적용된다. 디코딩된 비디오 신호(9569)가 최종적으로 획득되고 디코더(9560)에 의해 제공된다.

[0601] 도 12는 본 발명의 하나 이상의 실시예가 구현될 수 있는 데이터 통신 시스템을 예시한다. 데이터 통신 시스템

은 데이터 통신 네트워크(9200)를 통해 데이터 스트림(9204)의 데이터 패킷들을 수신 디바이스, 이 경우에 클라이언트 단말(9202)로 전송하도록 동작 가능한 전송 디바이스, 이 경우에 서버(9201)를 포함한다. 데이터 통신 네트워크(9200)는 WAN(Wide Area Network) 또는 LAN(Local Area Network)일 수 있다. 그러한 네트워크는, 예를 들어, 무선 네트워크(Wifi/802.11a 또는 b 또는 g), 이더넷 네트워크, 인터넷 네트워크 또는 여러 상이한 네트워크들로 구성된 혼합 네트워크일 수 있다. 본 발명의 특정 실시예에서, 데이터 통신 시스템은 서버(9201)가 동일한 데이터 콘텐츠를 다수의 클라이언트들로 송신하는 디지털 텔레비전 방송 시스템일 수 있다.

[0602] 서버(9201)에 의해 제공되는 데이터 스트림(9204)은 비디오 및 오디오 데이터를 나타내는 멀티미디어 데이터로 구성될 수 있다. 오디오 및 비디오 데이터 스트림들은, 본 발명의 일부 실시예들에서, 마이크로폰 및 카메라를, 제각기, 사용하여 서버(9201)에 의해 캡처될 수 있다. 일부 실시예들에서, 데이터 스트림들은 서버(9201)에 저장되거나 또는 서버(9201)에 의해 다른 데이터 제공자로부터 수신되거나 또는 서버(9201)에서 생성될 수 있다. 서버(9201)는, 특히 인코더에 대한 입력으로서 제시되는 데이터의 더 간결한 표현인 전송을 위한 압축된 비트스트림을 제공하기 위해, 비디오 및 오디오 스트림들을 인코딩하기 위한 인코더를 구비하고 있다. 전송되는 데이터의 양에 대한 전송되는 데이터의 품질의 더 양호한 비를 달성하기 위해, 비디오 데이터의 압축은, 예를 들어, HEVC(High Efficiency Video Coding) 포맷 또는 H.264/AVC(Advanced Video Coding) 포맷 또는 VVC(Versatile Video Coding) 포맷을 따를 수 있다. 클라이언트(9202)는, 디스플레이 디바이스에서 비디오 이미지를 재생하고 스피커에 의해 오디오 데이터를 재생하기 위해, 전송된 비트스트림을 수신하고 재구성된 비트스트림을 디코딩한다.

[0603] 이 실시예에서 스트리밍 시나리오가 고려되지만, 본 발명의 일부 실시예들에서 인코더와 디코더 사이의 데이터 통신이, 예를 들어, 광학 디스크와 같은, 미디어 저장 디바이스를 사용하여 수행될 수 있다는 것이 이해될 것이다. 본 발명의 하나 이상의 실시예에서, 최종 이미지에서 필터링된 픽셀들을 제공하기 위해 이미지의 재구성된 픽셀들에 적용하기 위한 보상 오프셋들을 나타내는 데이터와 함께 비디오 이미지가 전송될 수 있다.

[0604] 도 13은 본 발명의 적어도 하나의 실시예를 구현하도록 구성된 프로세싱 디바이스(9300)를 개략적으로 예시한다. 프로세싱 디바이스(9300)는 마이크로컴퓨터, 워크스테이션, 사용자 단말 또는 경량 휴대용 디바이스와 같은 디바이스일 수 있다. 디바이스/장치(9300)는 다음과 같은 것들에 연결되는 통신 버스(9313)를 포함한다:

[0605] - CPU로 표기된, 마이크로프로세서와 같은, 중앙 프로세싱 유닛(9311);

[0606] - 디바이스(9300)를 동작시키고/시키거나 본 발명을 구현하기 위한 컴퓨터 프로그램들/명령어들을 저장하기 위한, ROM으로 표기된, 판독 전용 메모리(9307);

[0607] - 본 발명의 실시예들의 방법의 실행 가능 코드는 물론, 디지털 이미지들의 시퀀스를 인코딩하는 방법 및/또는 본 발명의 실시예들에 따른 비트스트림을 디코딩하는 방법을 구현하는 데 필요한 변수들 및 파라미터들을 기록하도록 구성된 레지스터들을 저장하기 위한, RAM으로 표기된, 랜덤 액세스 메모리(9312); 및

[0608] - 프로세싱될 디지털 데이터가 그를 통해 전송되거나 수신되는 통신 네트워크(9303)에 연결된 통신 인터페이스(9302).

[0609] 임의로, 장치(9300)는 다음과 같은 컴포넌트들을 또한 포함할 수 있다:

[0610] - 본 발명의 하나 이상의 실시예의 방법들을 구현하기 위한 컴퓨터 프로그램들 및 본 발명의 하나 이상의 실시예의 구현 동안 사용되거나 생성되는 데이터를 저장하기 위한, 하드 디스크와 같은, 데이터 저장 수단(9304);

[0611] - 디스크(9306)(예를 들면, 저장 매체)에 대한 디스크 드라이브(9305) - 디스크 드라이브(9305)는 디스크(9306)로부터 데이터를 판독하거나 상기 디스크(9306)에 데이터를 기입하도록 구성됨 -; 또는

[0612] - 키보드(9310), 터치스크린 또는 임의의 다른 포인팅/입력 수단을 통해, 데이터를 디스플레이하고/하거나 사용자와의 그래픽 인터페이스로서 역할하기 위한 스크린(9309).

[0613] 장치(9300)는, 예를 들어, 디지털 카메라(9320) 또는 마이크로폰(9308)과 같은, 다양한 주변기기들에 연결될 수 있으며, 각각의 주변기기는 멀티미디어 데이터를 장치(9300)에 공급하기 위해 입출력 카드(도시되지 않음)에 연결된다.

[0614] 통신 버스(9313)는 장치(9300)에 포함되거나 이에 연결되는 다양한 요소들 간의 통신 및 상호운용성을 제공한다. 버스의 표현은 제한적이지 않으며, 특히 중앙 프로세싱 유닛(9311)은 장치(9300)의 임의의 요소에

직접 또는 장치(9300)의 다른 요소를 통해 명령어들을 통신하도록 동작 가능하다.

[0615] 디스크(9306)는, 예를 들어, 재기입 가능(rewritable)하거나 그렇지 않은 콤팩트 디스크(CD-ROM), ZIP 디스크 또는 메모리 카드와 같은 임의의 정보 매체로 대체될 수 있으며, 일반적으로, 장치에 통합되거나 통합되지 않은 마이크로컴퓨터 또는 프로세서에 의해 판독되고, 어쩌면 이동식이며, 실행이 구현될 본 발명에 따른 디지털 이미지들의 시퀀스를 인코딩하는 방법 및/또는 비트스트림을 디코딩하는 방법을 가능하게 하는 하나 이상의 프로그램을 저장하도록 구성된 정보 저장 수단으로 대체될 수 있다.

[0616] 실행 가능 코드는 판독 전용 메모리(9307)에, 하드 디스크(9304)에 또는, 예를 들어, 이전에 기술된 바와 같은 디스크(9306)와 같은 이동식 디지털 매체에 저장될 수 있다. 변형례에 따르면, 프로그램들의 실행 가능 코드는, 실행되기 전에 장치(9300)의 저장 수단들 중 하나, 예를 들어, 하드 디스크(9304)에 저장되기 위해, 통신 네트워크(9303)를 통해 인터페이스(9302)를 거쳐 수신될 수 있다.

[0617] 중앙 프로세싱 유닛(9311)은 전술한 저장 수단들 중 하나에 저장되는 명령어들인, 본 발명에 따른 프로그램 또는 프로그램들의 소프트웨어 코드의 명령어들 또는 부분들의 실행을 제어하고 지시하도록 구성된다. 전원을 켤 때, 비휘발성 메모리에, 예를 들어, 하드 디스크(9304), 디스크(9306)에 또는 판독 전용 메모리(9307)에 저장되어 있는 프로그램 또는 프로그램들은 랜덤 액세스 메모리(9312)로 전송되고, 그러면 랜덤 액세스 메모리(9312)는 프로그램 또는 프로그램들의 실행 가능 코드는 물론 본 발명을 구현하는 데 필요한 변수들 및 파라미터들을 저장하기 위한 레지스터들을 포함한다.

[0618] 이 실시예에서, 장치는 본 발명을 구현하기 위해 소프트웨어를 사용하는 프로그램 가능한 장치이다. 그렇지만, 대안적으로, 본 발명은 하드웨어로(예를 들어, 주문형 집적 회로 또는 ASIC의 형태로) 구현될 수 있다.

[0619] 본 발명의 실시예들의 구현

[0620] 또한, 본 발명의 다른 실시예들에 따르면, 전술한 실시예에 따른 디코더가 컴퓨터, 모바일 폰(셀룰러 폰), 태블릿 또는 사용자에게 콘텐츠를 제공/디스플레이할 수 있는 임의의 다른 유형의 디바이스(예를 들면, 디스플레이 장치)와 같은 사용자 단말에 제공되어 있는 것이 이해된다. 또 다른 실시예에 따르면, 전술한 실시예에 따른 인코더는 인코더가 인코딩할 콘텐츠를 캡처하여 제공하는 카메라, 비디오 카메라 또는 네트워크 카메라(예를 들면, 폐쇄 회로 텔레비전 또는 비디오 감시 카메라)를 또한 포함하는 이미지 캡처 장치에 제공되어 있다. 2개의 그러한 실시예가 도 14 및 도 15를 참조하여 아래에서 제공된다.

[0621] 도 14는 네트워크 카메라(9452) 및 클라이언트 장치(9454)를 포함하는 네트워크 카메라 시스템(9450)을 예시하는 다이어그램이다.

[0622] 네트워크 카메라(9452)는 이미징 유닛(9456), 인코딩 유닛(9458), 통신 유닛(9460) 및 제어 유닛(9462)을 포함한다. 네트워크 카메라(9452)와 클라이언트 장치(9454)는 네트워크(9200)를 통해 서로 통신할 수 있도록 상호 연결된다. 이미징 유닛(9456)은 렌즈와 이미지 센서(예를 들면, CCD(charge coupled device) 또는 CMOS(complementary metal oxide semiconductor))를 포함하며, 물체의 이미지를 캡처하고 이미지에 기초하여 이미지 데이터를 생성한다. 이 이미지는 정지 이미지 또는 비디오 이미지일 수 있다. 이미징 유닛은 (광학적으로 또는 디지털적으로) 줌잉 또는 패닝하도록 제각기 구성된 줌잉 수단 및/또는 패닝 수단을 또한 포함할 수 있다. 인코딩 유닛(9458)은 전술한 실시예들 중 하나 이상에서 설명된 상기 인코딩 방법들을 사용하여 이미지 데이터를 인코딩한다. 인코딩 유닛(9458)은 전술한 실시예들에서 설명된 인코딩 방법들 중 적어도 하나를 사용한다. 다른 경우에, 인코딩 유닛(9458)은 전술한 실시예들에서 설명된 인코딩 방법들의 조합을 사용할 수 있다.

[0623] 네트워크 카메라(9452)의 통신 유닛(9460)은 인코딩 유닛(9458)에 의해 인코딩되는 인코딩된 이미지 데이터를 클라이언트 장치(9454)로 전송한다. 게다가, 통신 유닛(9460)은 또한 클라이언트 장치(9454)로부터 커맨드들을 수신할 수 있다. 커맨드들은 인코딩 유닛(9458)에 의한 인코딩에 대한 파라미터들을 설정하는 커맨드들을 포함한다. 제어 유닛(9462)은 통신 유닛(9460)에 의해 수신되는 커맨드들 또는 사용자 입력에 따라 네트워크 카메라(9452) 내의 다른 유닛들을 제어한다.

[0624] 클라이언트 장치(9454)는 통신 유닛(9464), 디코딩 유닛(9466) 및 제어 유닛(9468)을 포함한다. 클라이언트 장치(9454)의 통신 유닛(9464)은 커맨드들을 네트워크 카메라(9452)로 전송할 수 있다. 게다가, 클라이언트 장치(9454)의 통신 유닛(9464)은 네트워크 카메라(9452)로부터 인코딩된 이미지 데이터를 수신한다. 디코딩 유닛(9466)은 전술한 실시예들 중 하나 이상에서 설명된 상기 디코딩 방법들을 사용하여 인코딩된 이미지 데이터를 디코딩한다. 다른 경우에, 디코딩 유닛(9466)은 전술한 실시예들에서 설명된 디코딩 방법들의 조합을 사용할

수 있다. 클라이언트 장치(9454)의 제어 유닛(9468)은 통신 유닛(9464)에 의해 수신되는 사용자 조작 또는 커맨드들에 따라 클라이언트 장치(9454) 내의 다른 유닛들을 제어한다. 클라이언트 장치(9454)의 제어 유닛(9468)은 또한 디코딩 유닛(9466)에 의해 디코딩되는 이미지를 디스플레이하도록 디스플레이 장치(9470)를 제어할 수 있다.

[0625] 클라이언트 장치(9454)의 제어 유닛(9468)은 또한 네트워크 카메라(9452)에 대한 파라미터들, 예를 들어, 인코딩 유닛(9458)에 의한 인코딩에 대한 파라미터들의 값들을 지정하기 위해 GUI(Graphical User Interface)를 디스플레이하도록 디스플레이 장치(9470)를 제어할 수 있다. 클라이언트 장치(9454)의 제어 유닛(9468)은 또한 디스플레이 장치(9470)에 의해 디스플레이되는 GUI에 입력되는 사용자 조작에 따라 클라이언트 장치(9454) 내의 다른 유닛들을 제어할 수 있다. 클라이언트 장치(9454)의 제어 유닛(9468)은 또한 디스플레이 장치(9470)에 의해 디스플레이되는 GUI에 입력되는 사용자 조작에 따라 네트워크 카메라(9452)에 대한 파라미터들의 값들을 지정하는 커맨드들을 네트워크 카메라(9452)에 전송하도록 클라이언트 장치(9454)의 통신 유닛(9464)을 제어할 수 있다.

[0626] 도 15는 스마트 폰(9500)을 예시하는 다이어그램이다. 스마트 폰(9500)은 통신 유닛(9502), 디코딩/인코딩 유닛(9504), 제어 유닛(9506) 및 디스플레이 유닛(9508)을 포함한다.

[0627] 통신 유닛(9502)은 네트워크(9200)를 통해 인코딩된 이미지 데이터를 수신한다. 디코딩/인코딩 유닛(9504)은 통신 유닛(9502)에 의해 수신되는 인코딩된 이미지 데이터를 디코딩한다. 디코딩/인코딩 유닛(9504)은 전술한 실시예들 중 하나 이상에서 설명된 상기 디코딩 방법들을 사용하여 인코딩된 이미지 데이터를 디코딩한다. 디코딩/인코딩 유닛(9504)은 또한 전술한 실시예들에서 설명된 인코딩 또는 디코딩 방법들 중 적어도 하나를 사용할 수 있다. 다른 경우에, 디코딩/인코딩 유닛(9504)은 전술한 실시예들에서 설명된 디코딩 또는 인코딩 방법들의 조합을 사용할 수 있다. 제어 유닛(9506)은 통신 유닛(9502)에 의해 수신되는 사용자 조작 또는 커맨드들에 따라 스마트 폰(9500) 내의 다른 유닛들을 제어한다. 예를 들어, 제어 유닛(9506)은 디코딩/인코딩 유닛(9504)에 의해 디코딩되는 이미지를 디스플레이하도록 디스플레이 유닛(9508)을 제어한다.

[0628] 스마트 폰은 이미지들 또는 비디오들을 레코딩하기 위한 이미지 레코딩 디바이스(9510)(예를 들어, 디지털 카메라 및 연관된 회로)를 추가로 포함할 수 있다. 그러한 레코딩된 이미지들 또는 비디오들은 제어 유닛(9506)의 지시에 따라 디코딩/인코딩 유닛(9504)에 의해 인코딩될 수 있다. 스마트 폰은 모바일 디바이스의 배향을 감지하도록 구성된 센서들(9512)을 추가로 포함할 수 있다. 그러한 센서들은 가속도계, 자이로스코프, 나침반, GPS(global positioning) 유닛 또는 유사한 위치 센서들을 포함할 수 있다. 그러한 센서들(9512)은 스마트 폰이 배향을 변경하는지를 결정할 수 있고 그러한 정보는 비디오 스트림을 인코딩할 때 사용될 수 있다.

[0629] 본 발명이 실시예들을 참조하여 기술되었지만, 본 발명이 개시된 실시예들로 제한되지 않는다는 것이 이해되어야 한다. 첨부된 청구항들에 한정된 바와 같은, 본 발명의 범위를 벗어나지 않으면서 다양한 변경들 및 수정이 이루어질 수 있음이 본 기술 분야의 통상의 기술자에 의해 이해될 것이다. 본 명세서(임의의 첨부된 청구항들, 요약서 및 도면들을 포함함)에 개시된 특징들 전부, 및/또는 그렇게 개시된 임의의 방법 또는 프로세스의 단계들 전부는, 그러한 특징들 및/또는 단계들 중 적어도 일부가 상호 배타적인 조합들을 제외한, 임의의 조합으로 조합될 수 있다. 본 명세서(임의의 첨부된 청구항들, 요약서 및 도면들을 포함함)에 개시된 각각의 특징은, 달리 명시적으로 언급되지 않는 한, 동일한, 동등한 또는 유사한 목적을 달성하는 대안적인 특징들로 대체될 수 있다. 따라서, 달리 명시적으로 언급되지 않는 한, 개시된 각각의 특징은 일반적인 일련의 동등하거나 유사한 특징들의 일 예에 불과하다.

[0630] 또한, 예를 들면, 디코딩 프로세스 동안, 비교, 결정, 평가, 선택, 실행, 수행 또는 고려를 실제로 수행하는 대신에, 표시된 또는 결정된/추론된 결과가 프로세싱에서 사용될 수 있도록, 위에서 기술된 비교, 결정, 평가, 선택, 실행, 수행 또는 고려, 예를 들어, 인코딩 또는 필터링 프로세스 동안 이루어진 선택의 임의의 결과가 비트스트림에서의 데이터, 예를 들면, 결과를 나타내는 플래그 또는 정보에 표시되거나 그로부터 결정 가능/추론 가능할 수 있음이 이해된다.

[0631] 청구항들에서, "포함하는(comprising)"이라는 단어는 다른 요소들 또는 단계들을 배제하지 않으며, 단수 관사("a" 또는 "an")는 복수를 배제하지 않는다. 상이한 특징들이 상호 상이한 종속 청구항들에서 인용되고 있다는 단순한 사실은 이러한 특징들의 조합이 유리하게 사용될 수 없다는 것을 나타내지 않는다. 청구항들에 나오는 참조 번호들은 예시에 불과하고, 청구항들의 범위를 제한하는 효과를 갖지 않아야 한다.

[0632] 선행 실시예들에서, 기술되는 기능들은 하드웨어, 소프트웨어, 펌웨어, 또는 이들의 임의의 조합으로 구현될 수

있다. 소프트웨어로 구현되는 경우, 기능들은, 하나 이상의 명령어 또는 코드로서, 컴퓨터 판독 가능 매체 상에 저장되거나 그를 통해 전송될 수 있고 하드웨어 기반 프로세싱 유닛에 의해 실행될 수 있다.

[0633] 컴퓨터 판독 가능 매체는 데이터 저장 매체와 같은 유형적 매체에 대응하는 컴퓨터 판독 가능 저장 매체, 또는, 예를 들면, 통신 프로토콜에 따라, 한 장소로부터 다른 장소로의 컴퓨터 프로그램의 전송을 용이하게 하는 임의의 매체를 포함한 통신 매체를 포함할 수 있다. 이러한 방식으로, 컴퓨터 판독 가능 매체는 일반적으로 (1) 비일시적인 유형적 컴퓨터 판독 가능 저장 매체 또는 (2) 신호 또는 반송파(carrier wave)와 같은 통신 매체에 대응할 수 있다. 데이터 저장 매체는 이 개시에서 설명되는 기술들의 구현을 위한 명령어들, 코드 및/또는 데이터 구조들을 검색하기 위해 하나 이상의 컴퓨터 또는 하나 이상의 프로세서에 의해 액세스될 수 있는 임의의 이용 가능한 매체일 수 있다. 컴퓨터 프로그램 제품은 컴퓨터 판독 가능 매체를 포함할 수 있다. 제한이 아닌 예로서, 그러한 컴퓨터 판독 가능 저장 매체는 RAM, ROM, EEPROM, CD-ROM 또는 다른 광학 디스크 스토리지, 자기 디스크 스토리지 또는 다른 자기 저장 디바이스들, 플래시 메모리, 또는 원하는 프로그램 코드를 명령어들 또는 데이터 구조들의 형태로 저장하는 데 사용될 수 있고 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 또한, 임의의 연결(connection)이 컴퓨터 판독 가능 매체라고 적절히 지칭된다. 예를 들어, 명령어들이 동축 케이블, 광섬유 케이블, 연선(twisted pair), DSL(digital subscriber line), 또는 적외선, 전파(radio), 및 마이크로파와 같은 무선 기술들을 사용하여 웹사이트, 서버, 또는 다른 원격 소스로부터 전송되는 경우, 동축 케이블, 광섬유 케이블, 연선, DSL, 또는 적외선, 전파, 및 마이크로파와 같은 무선 기술들은 매체의 정의에 포함된다. 그렇지만, 컴퓨터 판독 가능 저장 매체 및 데이터 저장 매체가 연결, 반송파, 신호, 또는 다른 일시적 매체를 포함하지 않고, 그 대신에 비일시적, 유형적 저장 매체에 관한 것임이 이해되어야 한다. 디스크(disk) 및 디스크(disc)는, 본 명세서에서 사용되는 바와 같이, CD(compact disc), 레이저 디스크, 광학 디스크, DVD(digital versatile disc), 플로피 디스크 및 블루레이 디스크를 포함하고, 여기서 디스크(disk)는 보통 데이터를 자기적으로 재생하는 반면, 디스크(disc)는 데이터를 레이저를 사용하여 광학적으로 재생한다. 상기한 것들의 조합들이 컴퓨터 판독 가능 매체의 범위 내에 또한 포함되어야 한다.

[0634] 명령어들은, 하나 이상의 디지털 신호 프로세서(DSP), 범용 마이크로프로세서, ASIC(application specific integrated circuit), FPGA(field programmable gate/logic array), 또는 다른 동등한 집적 또는 개별 로직 회로와 같은, 하나 이상의 프로세서에 의해 실행될 수 있다. 그에 따라, "프로세서"라는 용어는, 본 명세서에서 사용되는 바와 같이, 전술한 구조물 또는 본 명세서에 설명된 기술들의 구현에 적당한 임의의 다른 구조물 중 임의의 것을 지칭할 수 있다. 추가적으로, 일부 양태들에서, 본 명세서에 기술된 기능성은 인코딩 및 디코딩을 위해 구성된 전용 하드웨어 및/또는 소프트웨어 모듈들 내에 제공될 수 있거나, 또는 결합된 코덱(combined codec)에 통합될 수 있다. 또한, 기술들은 하나 이상의 회로 또는 로직 요소에서 완전히 구현될 수 있다.

[0635] 본 발명이 실시예들을 참조하여 기술되었지만, 본 발명이 개시된 실시예들로 제한되지 않는다는 것이 이해되어야 한다. 의심의 여지를 없애기 위해, 이하의 서술문들은 설명의 일부를 형성한다. 청구항들은 서술문들에 뒤따르며 이에 따라 표시되어 있다.

[0636] 서술문 1. 이미지의 하나 이상의 이미지 부분에 대한 적응 루프 필터를 제어하는 방법으로서, 이 방법은 제1 샘플 값의 복수의 이웃하는 샘플 값들에 기초하여 이미지 부분의 제1 샘플의 필터링을 제어하는 단계를 포함하며, 여기서 제어하는 단계는 비선형 함수를 사용하는 단계를 포함하고, 비선형 함수는 이웃하는 샘플 값(들) 중 하나 이상에 기초한 하나 이상의 변수(들) 및 적응 파라미터 세트에서 시그널링되는 하나 이상의 필터 변수(들)를 갖는다.

[0637] 서술문 2. 서술문 1의 방법으로서, 여기서 비선형 함수는 하나 이상의 클리핑 함수(들)를 포함하고, 하나 이상의 필터 변수(들)는 적응 파라미터 세트에서 제공된 정보를 사용하여 결정되는 하나 이상의 클리핑 파라미터(들)를 포함한다.

[0638] 서술문 3. 서술문 2의 방법으로서, 여기서 2개 이상의 클리핑 함수는 적응 파라미터 세트에서 시그널링되는 정보를 사용하여 결정되는 동일한 클리핑 파라미터를 공유한다.

[0639] 서술문 4. 서술문 2 또는 서술문 3의 방법으로서, 여기서 클리핑 함수에 대해 사용하기 위한 클리핑 파라미터는 복수의 클리핑 파라미터 값들로부터 클리핑 파라미터 값을 식별하기 위한 인덱스를 사용하여 결정 가능하고, 복수의 클리핑 파라미터 값들 중 하나는 비트 깊이에 기초한 최대 샘플 값에 대응한다. 적합하게는, 인덱스는 적응 파라미터 세트에서 제공된다.

[0640] 서술문 5. 서술문 2 내지 서술문 4 중 어느 한 서술문의 방법으로서, 여기서 클리핑 함수에 대해 사용하

기 위한 클리핑 파라미터는 복수의 클리핑 파라미터 값들로부터 클리핑 파라미터 값을 식별하기 위한 인덱스를 사용하여 결정 가능하고, 복수의 클리핑 파라미터 값들은 제1 샘플 값이 루마 샘플 값일 때와 제1 샘플 값이 크로마 샘플 값일 때 둘 모두에 대해 동일한 수의 값들을 포함한다. 적합하게는, 복수의 클리핑 파라미터 값들은 4개의(허용 가능한/이용 가능한/가능한) 값을 포함한다. 적합하게는, 복수의 클리핑 파라미터 값들은 이미지의 하나 이상의 이미지 부분 중 2개 이상에 대해 동일한 수의 (허용 가능한/이용 가능한/가능한) 값들을 포함한다. 적합하게는, 복수의 클리핑 파라미터 값들은 이미지의 하나 이상의 이미지 부분 중 2개 이상에 대해 4개의 (허용 가능한/이용 가능한/가능한) 값을 포함한다. 적합하게는, 복수의 클리핑 파라미터 값들은 이미지의 모든 이미지 부분들 또는 이미지들의 시퀀스의 모든 이미지 부분들에 대해 동일한 수의 (허용 가능한/이용 가능한/가능한) 값들을 포함한다. 적합하게는, 복수의 클리핑 파라미터 값들은 이미지의 모든 이미지 부분들 또는 이미지들의 시퀀스의 모든 이미지 부분들에 대해 4개의 (허용 가능한/이용 가능한/가능한) 값을 포함한다.

[0641] 서술문 6. 임의의 선행 서술문의 방법으로서, 여기서 시그널링되는 하나 이상의 필터 변수(들)의 수는 이웃하는 샘플 값(들) 중 하나 이상에 기초한 하나 이상의 변수(들)의 수보다 적다. 적합하게는, 시그널링되는 하나 이상의 필터 변수(들)의 수는 이웃하는 샘플 값(들) 중 하나 이상에 기초한 하나 이상의 변수(들)의 수의 절반이다.

[0642] 서술문 7. 임의의 선행 서술문의 방법으로서, 여기서 이웃하는 샘플 값(들) 중 하나 이상에 기초한 하나 이상의 변수는 제1 샘플 값과 하나 이상의 이웃하는 샘플 값(들) 각각 사이의 차이(들)를 포함한다.

[0643] 서술문 8. 임의의 선행 서술문의 방법으로서, 여기서 비선형 함수의 출력은 적응 루프 필터에 대한 입력 파라미터로서 사용된다. 적합하게는, 2개 이상의 비선형 함수(또는 클리핑 함수)로부터의 출력들이 적응 루프 필터의 입력 파라미터들로서 사용된다.

[0644] 서술문 9. 서술문 8의 방법으로서, 여기서 적응 루프 필터는 하나 이상의 필터 계수(들)를 사용하고; 필터 계수 및 그와 연관된 필터 변수는 이웃하는 샘플 값에 대한 인덱스를 사용하여 결정 가능하다.

[0645] 서술문 10. 서술문 9의 방법으로서, 여기서 이웃하는 샘플 값에 대한 인덱스는 스캔 순서에서의 그의 위치와 연관되고, 상기 연관은 전치 인덱스를 사용하여 도출 가능하다. 적합하게는, 상기 도출된 연관은 도 4b에서의 4개의 가능한 배열: 409의 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; 410의 {9, 4, 10, 8, 1, 5, 11, 7, 3, 0, 2, 6}; 411의 {0, 3, 2, 1, 8, 7, 6, 5, 4, 9, 10, 11}; 또는 412의 {9, 8, 10, 4, 3, 7, 11, 5, 1, 0, 2, 6} 중 하나에 대응한다.

[0646] 서술문 11. 임의의 선행 서술문의 방법으로서, 여기서 비선형 함수는 하나 이상의 클리핑 함수(들)를 포함하고, 하나 이상의 클리핑 함수(들) 각각은 max(-b, min(b,d)), min(b, max(-b,d)), max(c-b, min(c+b,n)), min(c+b, max(c-b,n)), max(-b, min(b,d1+d2)), min(b, max(-b,d1+d2)), max(2*c-b, min(2*c+b,n1+n2)), 또는 min(2*c+b, max(2*c-b,n1+n2)) 중 하나를 반환하며, 여기서 c는 제1 샘플 값이고, n 또는 n1 또는 n2는 이웃하는 샘플 값이며, d=n-c이고, d1=n1-c이며, d2=n2-c이고, b는 클리핑 파라미터이다.

[0647] 서술문 12. 이미지의 하나 이상의 부분 - 이미지 부분은 그와 연관된 크로마 샘플들 및 루마 샘플들을 가짐 - 을 프로세싱하는 방법으로서, 여기서 이 방법은, 비트스트림으로부터 획득되는 정보 또는 이미지 부분의 제1 샘플 값 및 그의 하나 이상의 이웃하는 샘플 값(들)에 기초하여, 임의의 선행 서술문의 방법을 사용하여 제어되는 필터를 사용할지 여부; 상기 필터의 사용을 인에이블 또는 디스에이블하는 것; 또는 제1 샘플 값을 필터링할 때 상기 필터에 대해 사용하기 위한 필터링 파라미터, 필터 계수 또는 필터 변수 중 적어도 하나를 결정하는 단계를 포함한다.

[0648] 서술문 13. 서술문 12의 방법으로서, 여기서 비트스트림으로부터 획득되는 정보는 루마 또는 크로마 성분 중 하나에 대해 제공되는 플래그를 포함하고, 플래그는 해당 성분에 대해 서술문 1 내지 서술문 10 중 어느 한 서술문의 방법을 사용하여 제어되는 필터를 사용할지 여부; 또는 상기 필터의 사용을 인에이블 또는 디스에이블하는 것 중 적어도 하나를 나타낸다.

[0649] 서술문 14. 서술문 12 또는 서술문 13의 방법으로서, 여기서 비트스트림으로부터 획득되는 정보는 하나 이상의 이미지 부분에 대해 제공되는 플래그를 포함하고, 플래그는 하나 이상의 이미지 부분에 대해 서술문 1 내지 서술문 10 중 어느 한 서술문의 방법을 사용하여 제어되는 필터를 사용할지 여부; 또는 상기 필터의 사용을 인에이블 또는 디스에이블하는 것 중 적어도 하나를 나타낸다.

[0650] 서술문 15. 하나 이상의 이미지(들)를 인코딩하는 방법으로서, 이 방법은, 이미지의 하나 이상의 부분에 대해, 서술문 1 내지 서술문 10 중 어느 한 서술문의 방법에 따라 필터를 제어하는 단계, 또는 서술문 12 내지

서술문 14 중 어느 한 서술문의 방법에 따라 프로세싱하는 단계를 포함한다.

[0651] 서술문 16. 서술문 15의 방법으로서, 이미지를 수신하는 단계; 수신된 이미지를 인코딩하고 비트스트림을 생성하는 단계; 및 인코딩된 이미지를 프로세싱하는 단계를 추가로 포함하고, 여기서 프로세싱하는 단계는 서술문 1 내지 서술문 11 중 어느 한 서술문의 방법에 따라 제어하는 단계 또는 서술문 12 내지 서술문 14 중 어느 한 서술문의 방법에 따라 프로세싱하는 단계를 포함한다.

[0652] 서술문 17. 서술문 12 내지 서술문 14 중 어느 한 서술문에 종속적일 때의 서술문 15의 방법으로서, 상기 정보를 비트스트림에서 제공하는 단계를 추가로 포함한다.

[0653] 서술문 18. 서술문 11에 종속적일 때의 서술문 17의 방법으로서, 여기서 비선형 함수의 변수들은 하나 이상의 이웃하는 샘플 값(들)의 인덱스/인덱스들에 의존하는 하나 이상의 필터 계수(들)를 추가로 포함하고; 상기 정보를 제공하는 단계는 하나 이상의 클리핑 파라미터(들)를 결정하기 위한 인덱스/인덱스들 및 이웃하는 샘플 값의 인덱스와 스캔 순서에서의 그의 위치 간의 연관을 도출하기 위한 하나 이상의 전치 인덱스/인덱스들을, 비트스트림에서, 제공하는 단계를 포함한다. 적합하게는, 도출된 연관은 도 4b에서의 4개의 가능한 배열: 409의 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; 410의 {9, 4, 10, 8, 1, 5, 11, 7, 3, 0, 2, 6}; 411의 {0, 3, 2, 1, 8, 7, 6, 5, 4, 9, 10, 11}; 또는 412의 {9, 8, 10, 4, 3, 7, 11, 5, 1, 0, 2, 6} 중 하나에 대응한다.

[0654] 서술문 19. 서술문 16 내지 서술문 18 중 어느 한 서술문의 방법으로서, 복수의 이용 가능한 함수들로부터 비선형 함수 또는 하나 이상의 클리핑 함수(들)를 선택하는 단계; 인코딩된 이미지를 프로세싱할 때 선택된 함수를 사용하는 단계; 및 선택된 기능을 식별하기 위한 정보를, 비트스트림에서, 제공하는 단계를 추가로 포함한다.

[0655] 서술문 20. 하나 이상의 이미지(들)를 디코딩하는 방법으로서, 이 방법은, 이미지의 하나 이상의 부분에 대해, 서술문 1 내지 서술문 11 중 어느 한 서술문의 방법에 따라 필터를 제어하는 단계, 또는 서술문 12 내지 서술문 14 중 어느 한 서술문의 방법에 따라 프로세싱하는 단계를 포함한다.

[0656] 서술문 21. 서술문 20의 방법으로서, 비트스트림을 수신하는 단계; 이미지를 획득하기 위해 수신된 비트스트림으로부터 정보를 디코딩하는 단계; 및 획득된 이미지를 프로세싱하는 단계를 추가로 포함하고, 여기서 프로세싱하는 단계는 서술문 1 내지 서술문 11 중 어느 한 서술문의 방법에 따라 제어하는 단계 또는 서술문 12 내지 서술문 14 중 어느 한 서술문의 방법에 따라 프로세싱하는 단계를 포함한다.

[0657] 서술문 22. 서술문 12 내지 서술문 14 중 어느 한 서술문에 종속적일 때의 서술문 21의 방법으로서, 비트스트림으로부터 상기 정보를 획득하는 단계를 추가로 포함한다.

[0658] 서술문 23. 서술문 11에 종속적일 때의 서술문 21의 방법으로서, 여기서 비선형 함수의 변수들은 하나 이상의 이웃하는 샘플 값(들)의 인덱스/인덱스들에 의존하는 하나 이상의 필터 계수(들)를 추가로 포함하고; 이 방법은 하나 이상의 클리핑 파라미터(들)를 결정하기 위한 인덱스/인덱스들 및 이웃하는 샘플 값의 인덱스와 스캔 순서에서의 그의 위치 간의 연관을 도출하기 위한 하나 이상의 전치 인덱스/인덱스들을, 비트스트림으로부터, 획득하는 단계를 추가로 포함한다. 적합하게는, 도출된 연관은 도 4b에서의 4개의 가능한 배열: 409의 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; 410의 {9, 4, 10, 8, 1, 5, 11, 7, 3, 0, 2, 6}; 411의 {0, 3, 2, 1, 8, 7, 6, 5, 4, 9, 10, 11}; 또는 412의 {9, 8, 10, 4, 3, 7, 11, 5, 1, 0, 2, 6} 중 하나에 대응한다.

[0659] 서술문 24. 서술문 21 내지 서술문 23 중 어느 한 서술문의 방법으로서, 복수의 이용 가능한 함수들로부터 비선형 함수 또는 하나 이상의 클리핑 함수(들)를 식별하기 위한 정보를, 비트스트림으로부터, 획득하는 단계; 및 획득된 이미지를 프로세싱할 때 식별된 함수를 사용하는 단계를 추가로 포함한다.

[0660] 서술문 25. 디바이스로서, 이 디바이스는: 서술문 1 내지 서술문 11 또는 서술문 12 내지 서술문 14 중 어느 한 서술문의 방법을 수행하도록 구성된 제어기; 서술문 15 내지 서술문 19 중 어느 한 서술문의 방법을 수행하도록 구성된 인코더; 또는 서술문 20 내지 서술문 24 중 어느 한 서술문의 방법을 수행하도록 구성된 디코더 중 하나 이상을 포함한다.

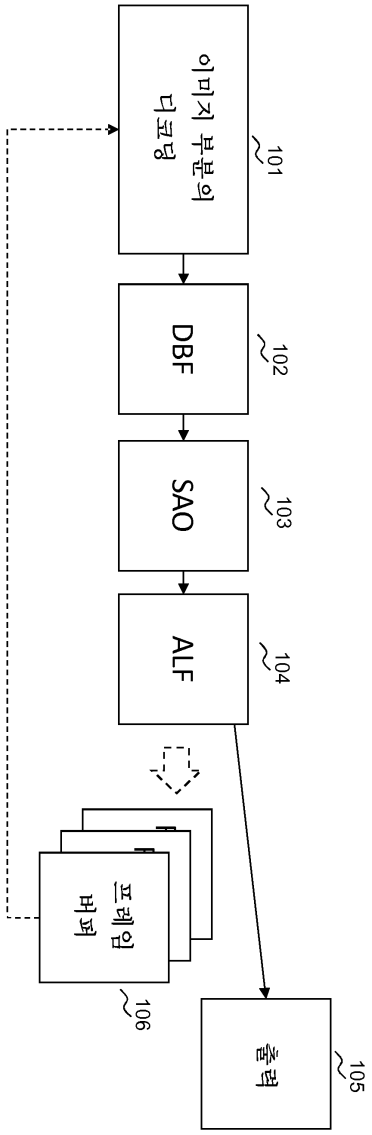
[0661] 서술문 26. 컴퓨터 또는 프로세서에서 실행될 때, 컴퓨터 또는 프로세서로 하여금 서술문 1 내지 서술문 11, 서술문 12 내지 서술문 14, 서술문 15 내지 서술문 19 또는 서술문 20 내지 서술문 24 중 한 서술문의 방법을 수행하게 하는 프로그램.

[0662] 서술문 27. 서술문 26의 컴퓨터 프로그램을 저장하는 컴퓨터 판독 가능 저장 매체.

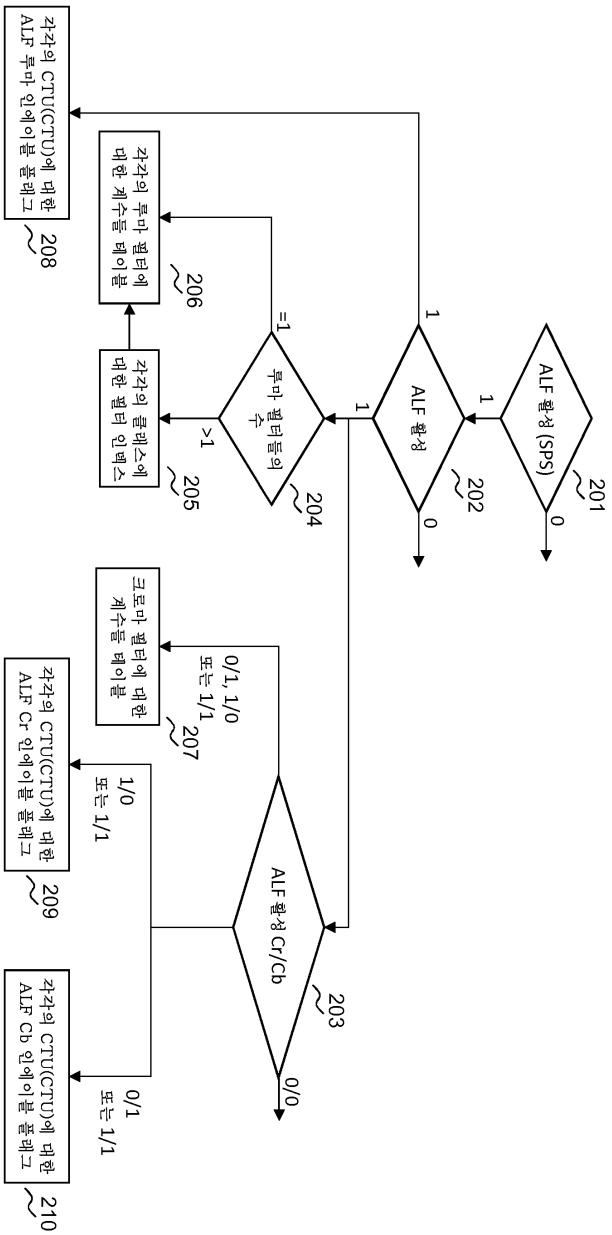
[0663] 서술문 28. 서술문 15 내지 서술문 19 중 어느 한 서술문의 방법을 사용하여 인코딩되고 비트스트림에 의해 표현되는 이미지에 대한 정보 데이터세트를 운반하는 신호로서, 이미지는 재구성 가능한 샘플들의 세트를 포함하고, 각각의 재구성 가능한 샘플은 샘플 값을 가지며, 여기서 정보 데이터세트는 제1 재구성 가능한 샘플의 이웃하는 샘플들의 샘플 값들에 기초하여 제1 재구성 가능한 샘플에 대한 필터링을 제어하기 위한 제어 데이터를 포함한다.

도면

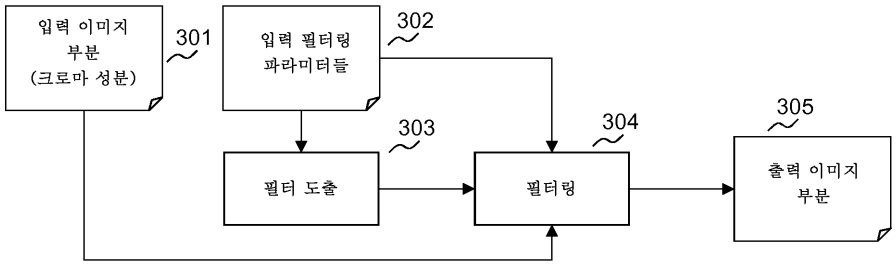
도면1



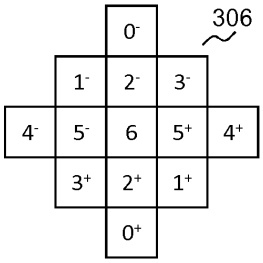
도면2


ALF 활성 (SPS)
201
ALF 활성
202
루마 필터들의 수
204
각각의 클래스에 대한 필터 인덱스
205
각각의 루마 필터에 대한 계수들 테이블
206
각각의 CTU(CTU)에 대한 ALF 루마 인에이블 플래그
208
ALF 활성 Cr/Cb
203
크로마 필터에 대한 계수들 테이블
207
각각의 CTU(CTU)에 대한 ALF Cr 인에이블 플래그
209
각각의 CTU(CTU)에 대한 ALF Cb 인에이블 플래그
210
1
0
>1
=1
0/1, 1/0 또는 1/1
1/0 또는 1/1
0/1 또는 1/1
0/0

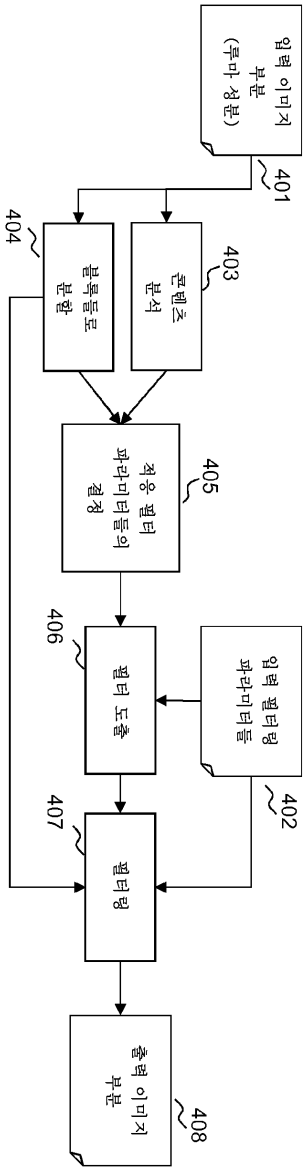
도면3a


입력 이미지 부분 (크로마 성분)
301
입력 필터링 파라미터들
302
필터 도출
303
필터링
304
출력 이미지 부분
305

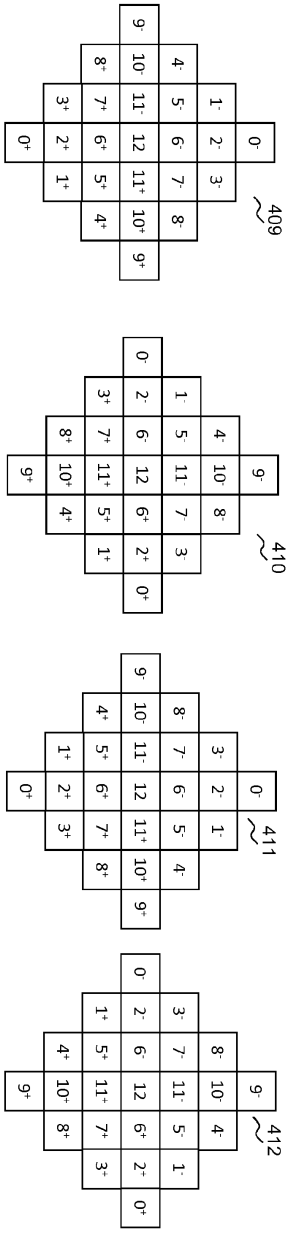
도면3b


306
0-
1-
2-
3-
4-
5-
6
5+
4+
3+
2+
1+
0+

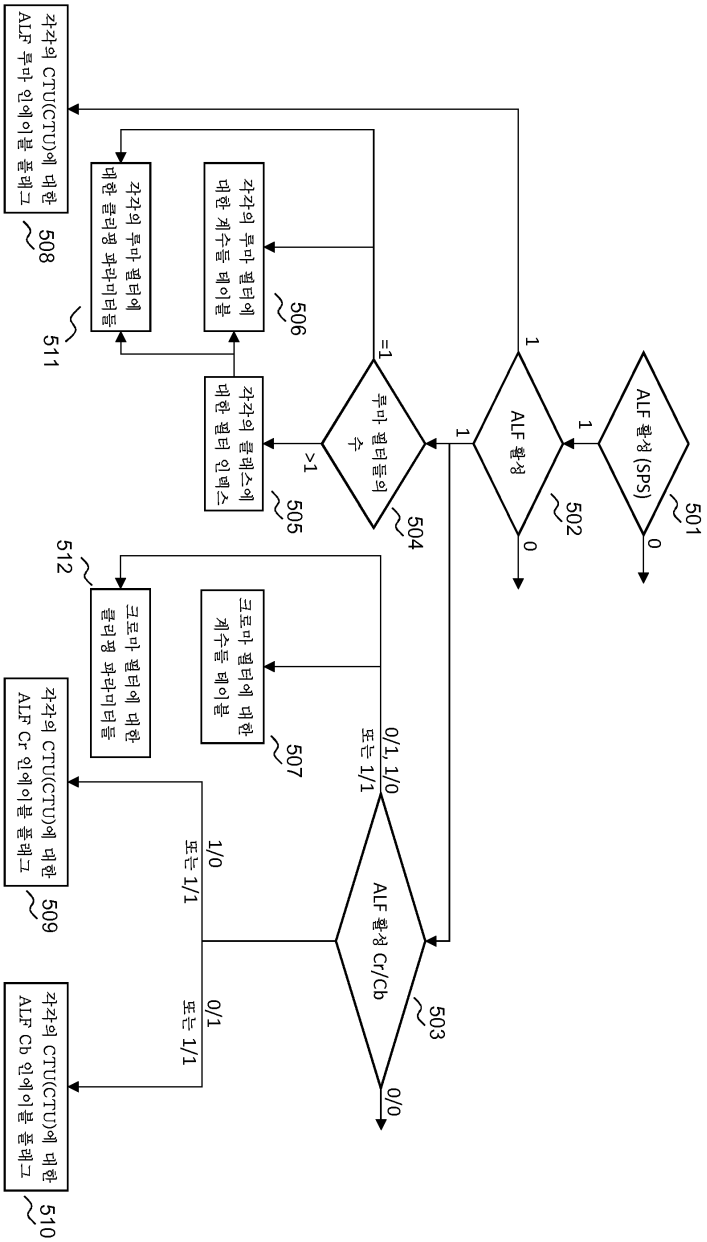
도면4a


입력 이미지 부분 (루마 성분)
401
404
블록들로 분할
콘텐츠 분석
403
적응 필터 파라미터들의 결정
405
406
필터 도출
입력 필터링 파라미터들
402
407
필터링
출력 이미지 부분
408

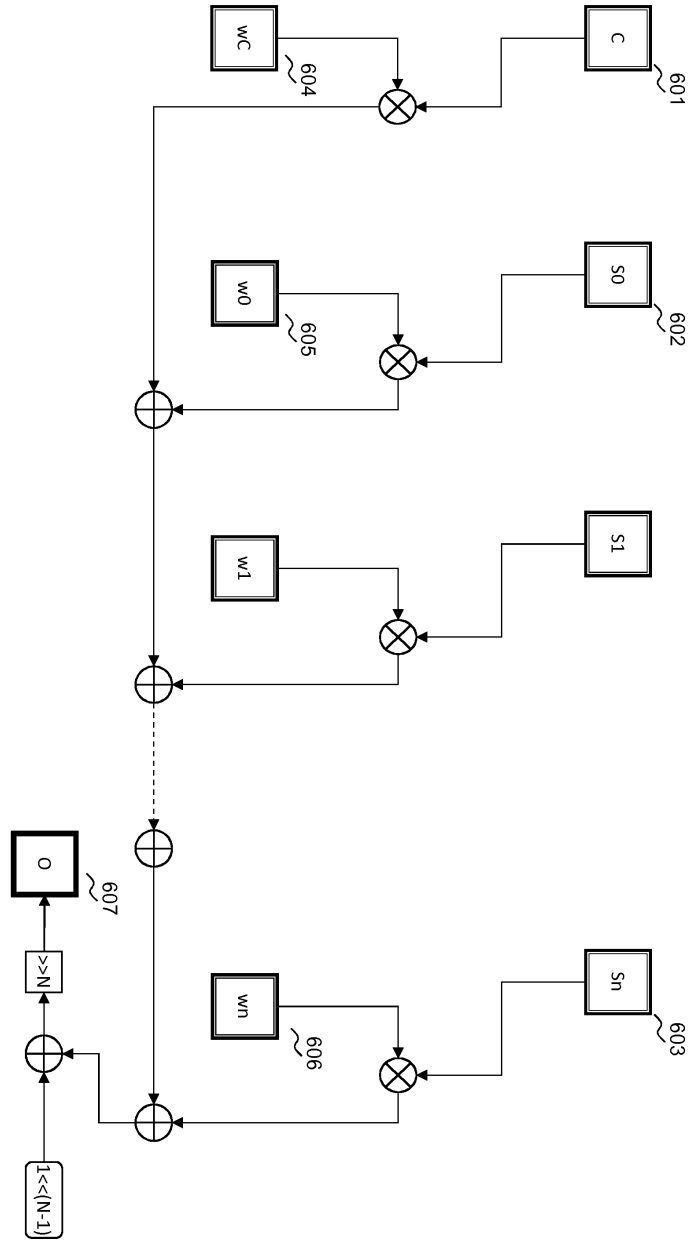
도면4b



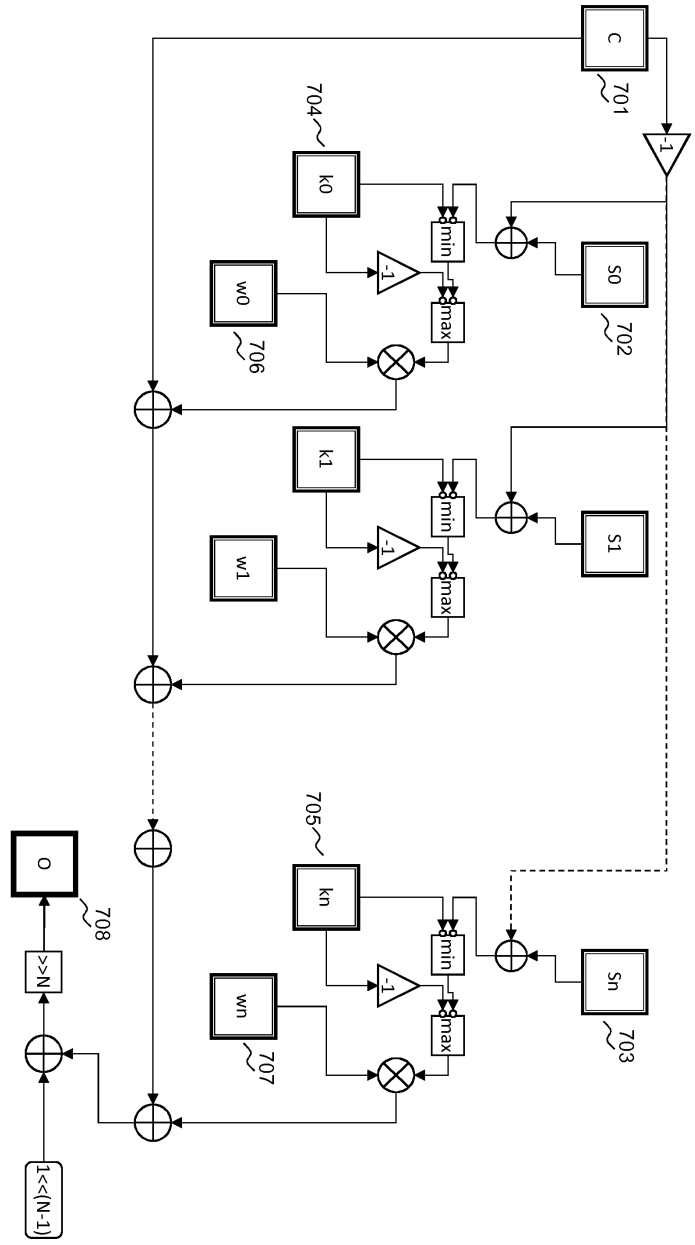
도면5


ALF 활성 (SPS)
501
ALF 활성
502
루마 필터들의 수
504
ALF 활성 Cr/Cb
503
각각의 루마 필터에 대한 클리핑 파라미터들
511
각각의 루마 필터에 대한 계수들 테이블
506
각각의 클래스에 대한 필터 인덱스
505
크로마 필터에 대한 클리핑 파라미터들
512
크로마 필터에 대한 계수들 테이블
507
각각의 CTU(CTU)에 대한 ALF 루마 인에이블 플래그
508
각각의 CTU(CTU)에 대한 ALF Cr 인에이블 플래그
509
각각의 CTU(CTU)에 대한 ALF Cb 인에이블 플래그
510
1
0
=1
>1
0/1, 1/0 또는 1/1
1/0 또는 1/1
0/1 또는 1/1
0/0

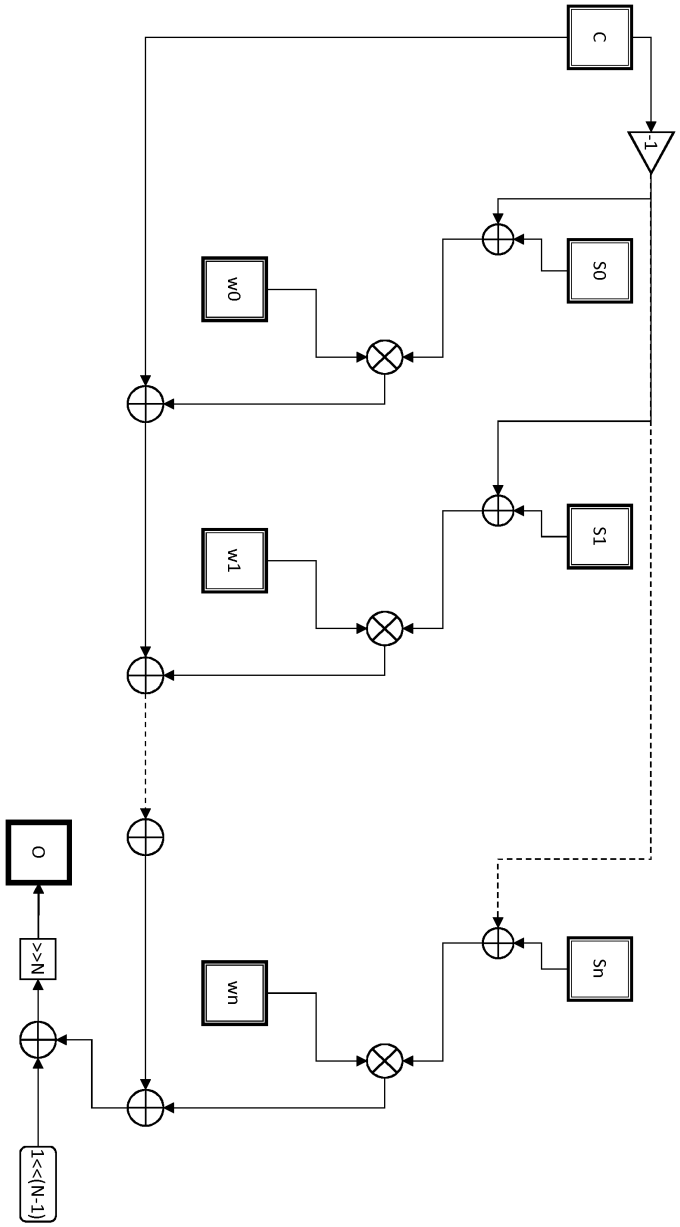
도면6



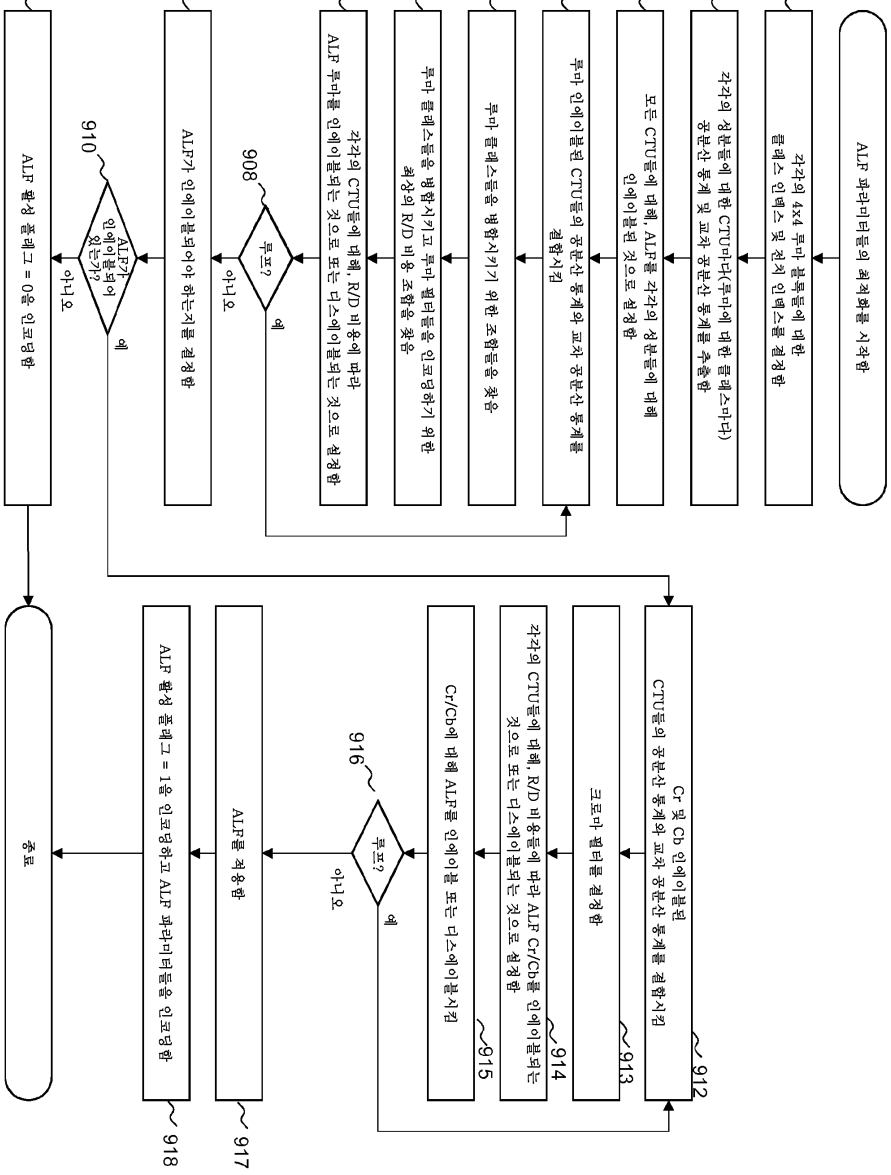
도면7


c
701
-1
704
k0
min
max
-1
s0
702
w0
706
k1
min
max
-1
s1
w1
705
kn
min
max
-1
Sn
703
wn
707
O
708
>>N
1<<(N-1)

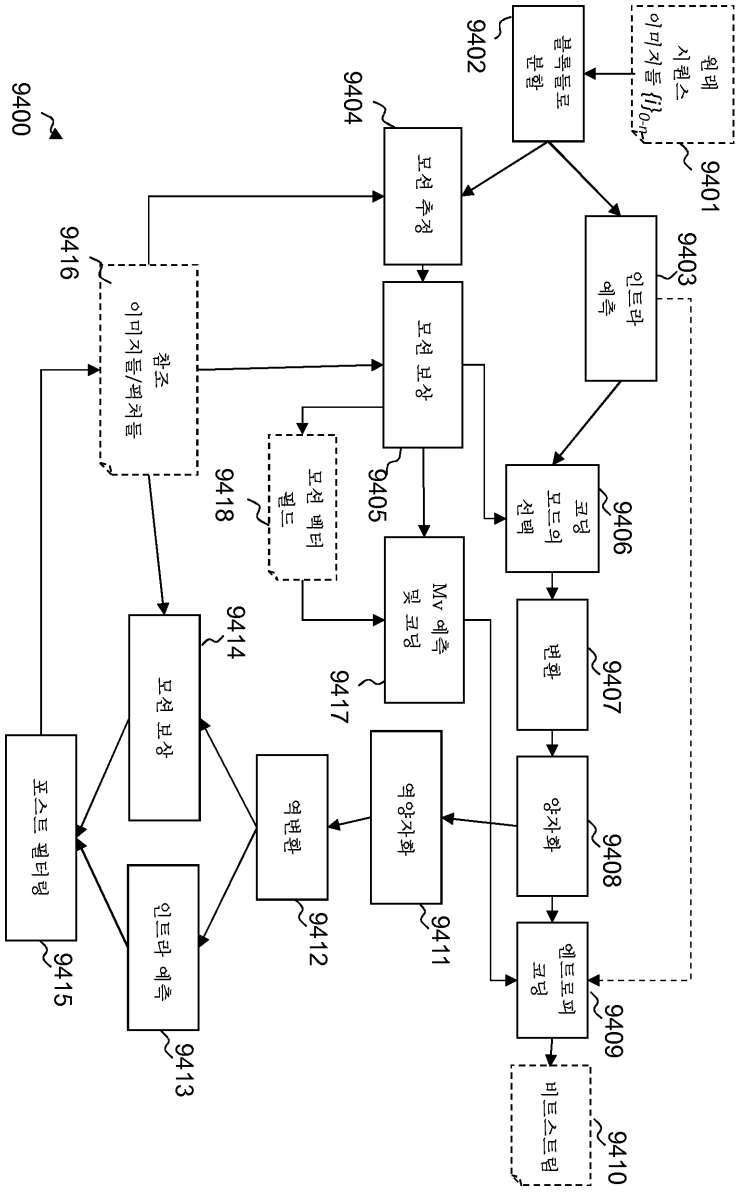
도면8



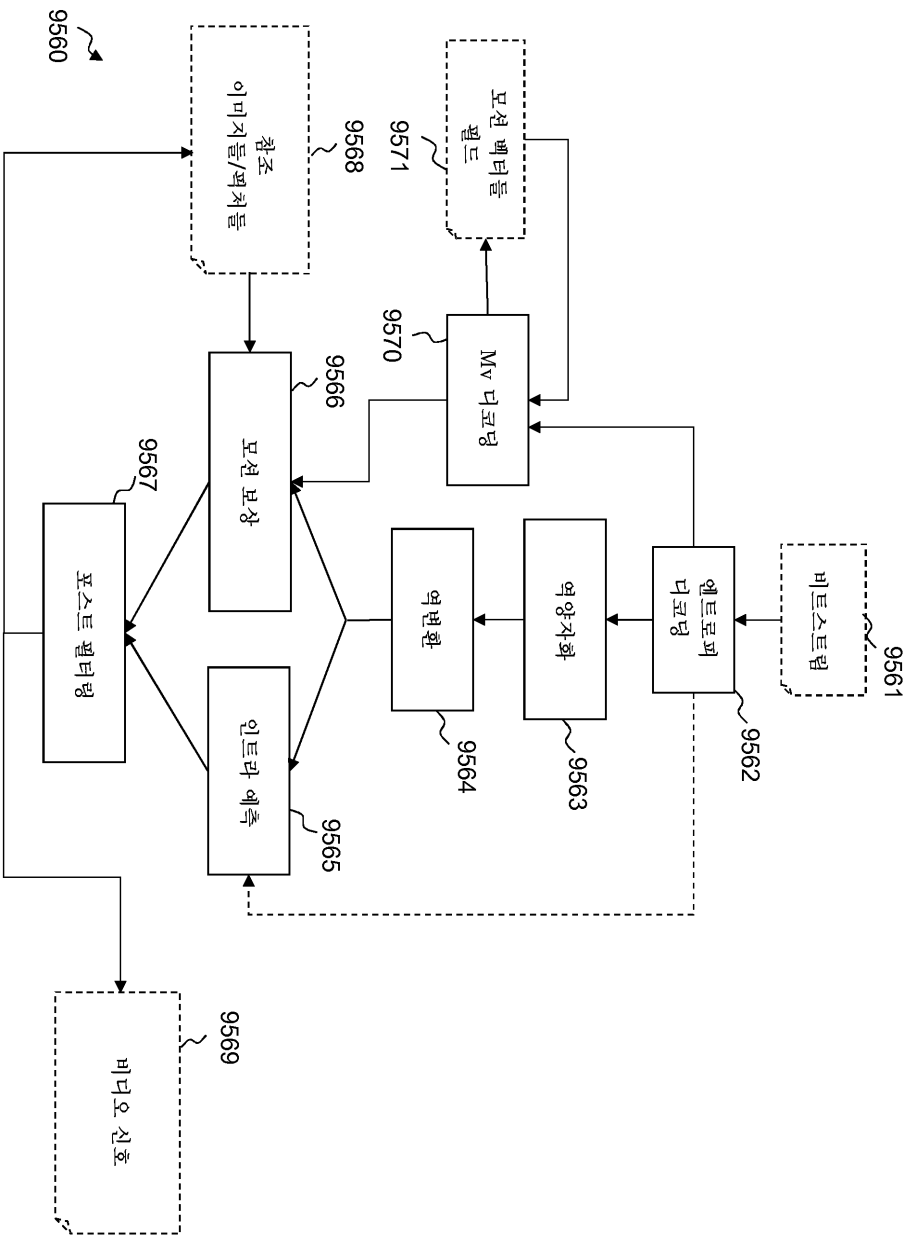
도면9


ALF 파라미터들의 최적화를 시작함
901
각각의 4x4 루마 블록들에 대한 클래스 인덱스 및 전치 인덱스를 결정함
902
각각의 성분들에 대한 CTU마다(루마에 대한 클래스마다) 공분산 통계 및 교차 공분산 통계를 추출함
903
모든 CTU들에 대해, ALF를 각각의 성분들에 대해 인에이블될 것으로 설정함
904
루마 인에이블된 CTU들의 공분산 통계와 교차 공분산 통계를 결합시킴
905
루마 클래스들을 병합시키기 위한 조합들을 찾음
906
루마 클래스들을 병합시키고 루마 필터들을 인코딩하기 위한 최상의 R/D 비용 조합을 찾음
907
ALF 루마를 인에이블되는 것으로 또는 디스에이블되는 것으로 설정함
908
루프?
예
아니오
909
ALF가 인에이블되어야 하는지를 결정함
910
ALF가 인에이블되어 있는가?
예
아니오
911
ALF 활성 플래그 = 0을 인코딩함
912
Cr 및 Cb 인에이블된 CTU들의 공분산 통계와 교차 공분산 통계를 결합시킴
913
크로마 필터를 결정함
914
각각의 CTU들에 대해, R/D 비용들에 따라 ALF Cr/Cb를 인에이블되는 것으로 또는 디스에이블되는 것으로 설정함
915
Cr/Cb에 대해 ALF를 인에이블 또는 디스에이블시킴
916
루프?
예
아니오
917
ALF를 적용함
918
ALF 활성 플래그 = 1을 인코딩하고 ALF 파라미터들을 인코딩함
종료

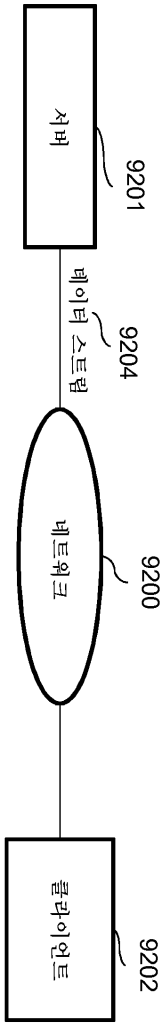
도면10


9400
원래 시퀀스 이미지들 $\{I_i\}_{i=0}^{N-1}$
9401
블록들로 분할
9402
모션 추정
9404
인트라 예측
9403
참조 이미지들/픽처들
9416
모션 보상
9405
모션 벡터 필드
9418
코딩 모드의 선택
9406
Mv 예측 및 코딩
9417
변환
9407
모션 보상
9414
양자화
9408
역양자화
9411
엔트로피 코딩
9409
역변환
9412
포스트 필터링
9415
인트라 예측
9413
비트스트림
9410

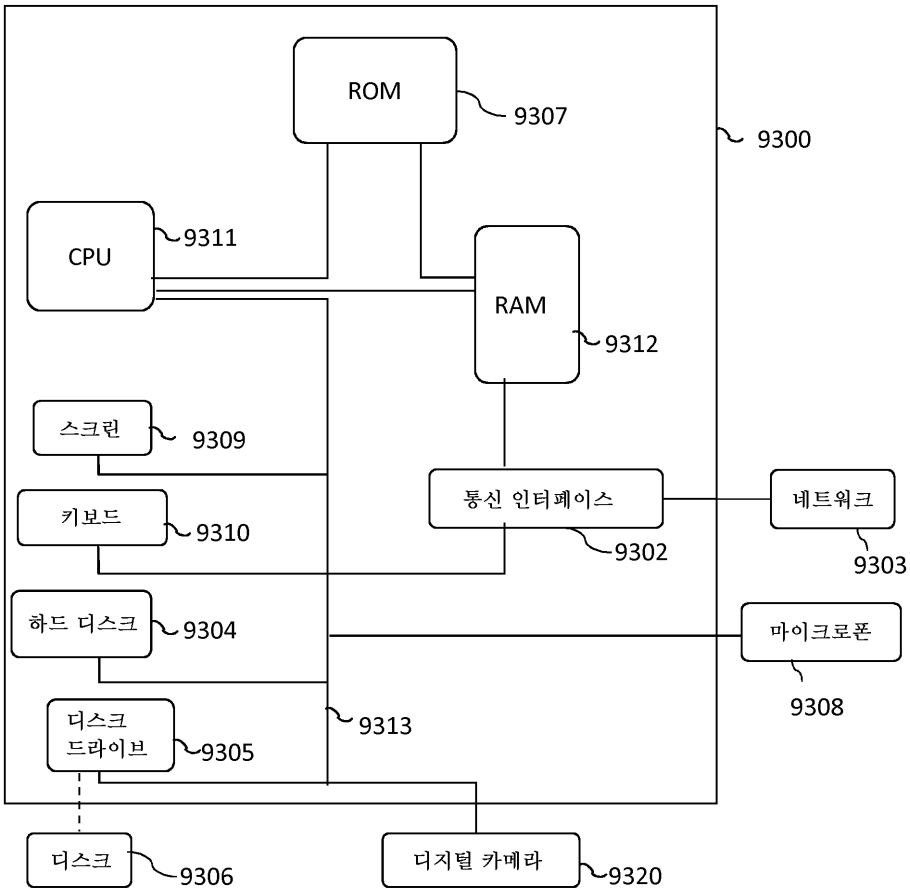
도면11


9560
참조 이미지들/픽처들
9568
모션 벡터들 필드
9571
Mv 디코딩
9570
모션 보상
9566
포스트 필터링
9567
인트라 예측
9565
역변환
9564
역양자화
9563
엔트로피 디코딩
9562
비트스트림
9561
비디오 신호
9569

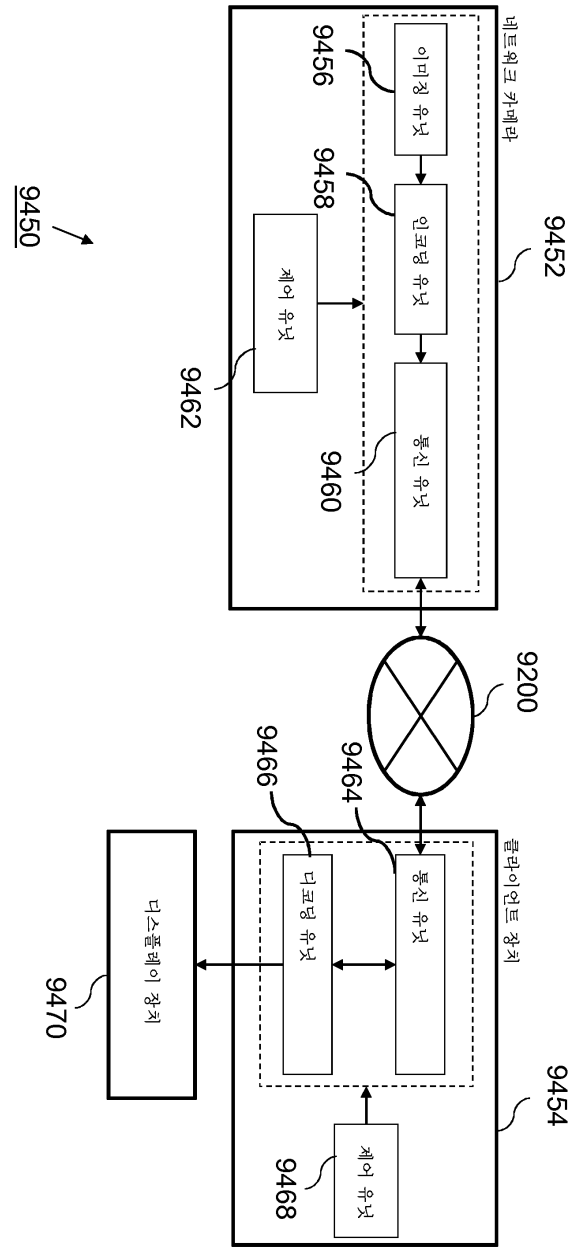
도면12



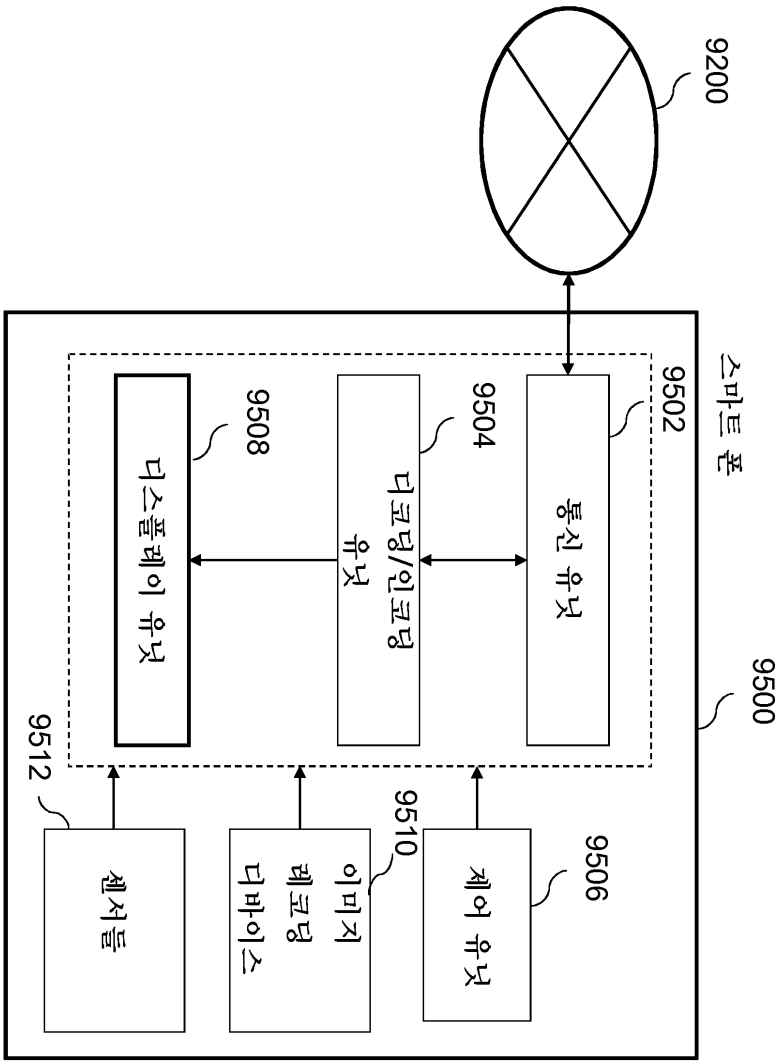
도면13


ROM
9307
9300
CPU
9311
RAM
9312
스크린
9309
키보드
9310
통신 인터페이스
9302
네트워크
9303
하드 디스크
9304
마이크로폰
9308
디스크
드라이브
9305
9313
디스크
9306
디지털 카메라
9320

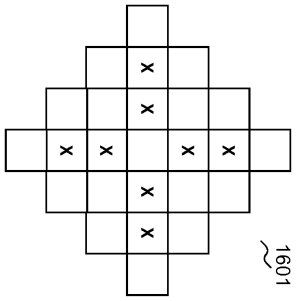
도면14

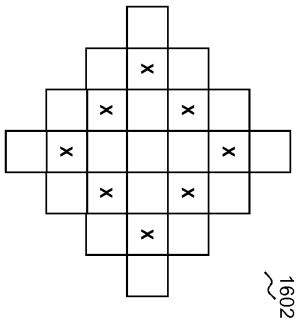

9450
네트워크 카메라
9452
이미징 유닛
9456
인코딩 유닛
9458
제어 유닛
9462
통신 유닛
9460
9200
9466
9464
클라이언트 장치
9454
디코딩 유닛
통신 유닛
제어 유닛
9468
디스플레이 장치
9470

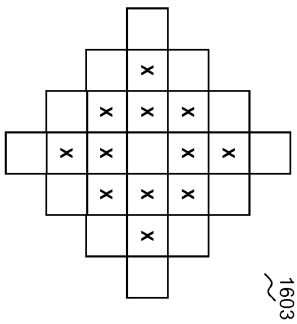
도면15


9200
스마트 폰
9500
9502
통신 유닛
9504
디코딩/인코딩 유닛
9508
디스플레이 유닛
9506
제어 유닛
9510
이미지 캡쳐 디바이스
9512
센서들

도면16a







도면16b