(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년11월07일
(11) 등록번호 10-2598746
(24) 등록일자 2023년11월01일

(51) 국제특허분류(Int. Cl.)
H04N 19/82 (2014.01) *H04N 19/117* (2014.01)
H04N 19/124 (2014.01) *H04N 19/129* (2014.01)
H04N 19/18 (2014.01) *H04N 19/423* (2014.01)
H04N 19/70 (2014.01)
(52) CPC특허분류
H04N 19/82 (2015.01)
H04N 19/117 (2015.01)
(21) 출원번호 10-2021-7002321
(22) 출원일자(국제) 2019년07월02일
심사청구일자 2021년01월22일
(85) 번역문제출일자 2021년01월22일
(65) 공개번호 10-2021-0021581
(43) 공개일자 2021년02월26일
(86) 국제출원번호 PCT/RU2019/050101
(87) 국제공개번호 WO 2020/009618
국제공개일자 2020년01월09일
(30) 우선권주장
62/693,441 2018년07월02일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
WO2018117896 A1*
WO2018117938 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
후아웨이 테크놀러지 컴퍼니 리미티드
중국 518129 광동성 센젠 롱강 디스트릭트 반티안 후아웨이 어드미니스트레이션 빌딩
(72) 발명자
이코닌, 세르게이 유리에비치
중국 518129 광둥 선전 룽강 디스트릭트 반톈 화웨이 어드미니스트레이션 빌딩
스테핀, 빅터 알렉세예비치
중국 518129 광둥 선전 룽강 디스트릭트 반톈 화웨이 어드미니스트레이션 빌딩
(뒷면에 계속)
(74) 대리인
양영준, 김성운, 백만기

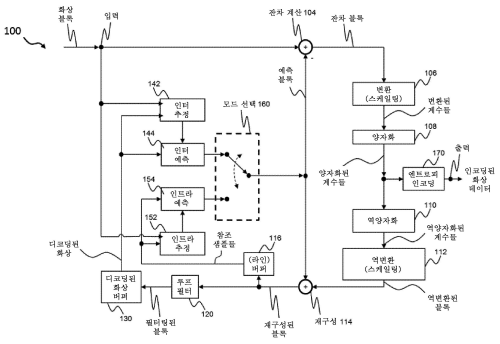
전체 청구항 수 : 총 30 항 심사관 : 김건우

(54) 발명의 명칭 **비디오 코딩에서의 필터링을 위한 장치 및 방법**

(57) 요 약

본 발명은 비디오 코딩을 위한 필터에 관한 것으로, 상기 필터는 필터링된 블록의 생성을 위해 블록을 처리하도록 구성되고, 상기 블록은 복수의 픽셀을 포함한다. 상기 필터는 하나 이상의 프로세서를 포함하고, 상기 하나 이상의 프로세서는: 상기 블록의 양자화 파라미터(QP)를 획득하고; 상기 QP에 기초하여 임계값(THR)을 획득하고; QP에 기초하여 룩업 테이블을 획득하여, 상기 임계값 및 상기 룩업 테이블에 기초하여 필터링된 블록을 생성하도록 구성된다. 비디오 코딩을 위한 효율을 개선하는 것을 허용하는 필터가 제공된다.

대 표 도



(52) CPC특허분류
H04N 19/124 (2015.01)
H04N 19/129 (2015.01)
H04N 19/18 (2015.01)
H04N 19/423 (2015.01)
H04N 19/70 (2015.01)
(72) 발명자
쿠리셰프, 드미트리
중국 518129 광둥 선전 룽강 디스트릭트 반톈 화웨이 어드미니스트레이션 빌딩
천, 젠러
미국 92130 캘리포니아주 샌 디에이고 코르테 드 티뷰론 10756
체르냑, 로만 이고리비치
중국 518129 광둥 선전 룽강 디스트릭트 반톈 화웨이 어드미니스트레이션 빌딩

(30) 우선권주장
62/725,845 2018년08월31일 미국(US)
62/731,967 2018년09월16일 미국(US)
62/731,972 2018년09월17일 미국(US)
62/735,722 2018년09월24일 미국(US)
62/757,732 2018년11월08일 미국(US)
62/793,866 2019년01월17일 미국(US)

명 세 서

청구범위

청구항 1

블록을 처리하기 위한 방법으로서,

상기 블록은 복수의 픽셀을 포함하고, 상기 방법은:

상기 블록의 양자화 파라미터(QP)를 획득하는 단계 - 상기 블록은 재구성된 블록 또는 예측된 블록임 -;

필터링 파라미터에 기초하여 임계값(THR)을 획득하는 단계 - 상기 필터링 파라미터는 상기 QP에 기초하여 도출됨 - ; 및

QP에 기초하여 룩업 테이블(LUT)을 획득하여, 상기 임계값 및 상기 룩업 테이블에 기초하여 필터링된 블록을 생성하는 단계를 포함하는, 방법.

청구항 2

제1항에 있어서,

상기 방법은:

미리 정의된 스캔 템플릿에 따라 상기 블록의 현재 픽셀 및 상기 현재 픽셀의 그것의 이웃 픽셀들을 스캐닝하는 단계;

상기 현재 픽셀 및 그것의 이웃 픽셀들에 대한 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득하는 단계;

상기 룩업 테이블 및 상기 스펙트럼 성분들을 사용하여 필터링된 스펙트럼 성분들을 획득하는 단계를 추가로 포함하는, 방법.

청구항 3

제2항에 있어서,

필터링된 스펙트럼 성분 $F(i,\sigma)$ 는:

$$F(i,\sigma)=\begin{cases} R(i)\ ,\ Abs\big(R(i)\big)\geq THR \\ LUT(R(i),\sigma),\ R(i)>0 \\ -LUT(-R(i),\sigma),\ R(i)\leq 0 \end{cases}$$

에 의해 도출되고, 여기서 (i)는 스펙트럼 성분의 인덱스이고, R(i)은 (i) 인덱스에 대응하는 스펙트럼 성분이고, σ는 상기 필터링 파라미터이고, $LUT(R(i),\sigma)$ 는 상기 룩업 테이블 내의 어레이이고, THR은 상기 QP로부터 도출되는 임계값인, 방법.

청구항 4

제3항에 있어서,

$LUT(R(i),\sigma)=\frac{R(i)^3}{R(i)^2+\mathrm{m}*\sigma^2}$ 이고, m은 스펙트럼 성분들의 수와 동등한 정규화 상수인, 방법.

청구항 5

제4항에 있어서,

상기 필터링 파라미터 σ는

$$\sigma = k * 2^{(n*(QP-s))}$$

에 의해 도출되고, 여기서 QP는 코덱 양자화 파라미터이고, k, n 및 s는 상수들인, 방법.

청구항 6

제5항에 있어서,

k=2.64, n=0.1296 및 s=11인, 방법.

청구항 7

제1항 내지 제4항 중 어느 한 항에 있어서,

상기 임계값은:

$$\frac{THR^2}{THR^2+m*\sigma^2}=C$$

에 의해 도출되고, 여기서 C는 1에 가까운 값이고, σ는 상기 필터링 파라미터이고, m은 스펙트럼 성분들의 수와 동등한 정규화 상수인, 방법.

청구항 8

제1항 내지 제4항 중 어느 한 항에 있어서,

상기 방법은:

상기 룩업 테이블을 획득하기 위해 QP들의 세트를 선택하는 단계를 추가로 포함하고, 상기 QP들의 세트는 (i) 인덱스에 대응하는 제1 QP 및 (i+1) 인덱스에 대응하는 제2 QP를 포함하고, 상기 제1 QP 및 상기 제2 QP는 1보다 큰 간격을 갖는, 방법.

청구항 9

제8항에 있어서,

상기 간격은 일정하고 8, 10 또는 16과 동등한, 방법.

청구항 10

제1항 내지 제4항 중 어느 한 항에 있어서,

상기 방법은:

상기 LUT의 테이블 값들로부터 N개의 비트를 드로핑하는 단계를 추가로 포함하고, N은 정수인, 방법.

청구항 11

제10항에 있어서,

LUT의 요소에 대한 액세스는 다음과 같고

```
tbl[(z + HTDF_TBL_RND)>> HTDF_TBL_SH]
```

여기서 `HTDF_TBL_SH`는 N의 가명이고, `HTDF_TBL_RND = 1 << (HTDF_TBL_SH - 1)`인, 방법.

청구항 12

제10항에 있어서,

LUT의 요소에 대한 액세스는 다음과 같고

$fHad[i]$의 양의 값들에 대해 $LUT[(fHad[i] + (1 \ll (tblShift - 1))) \gg tblShift]$이고,

$fHad[i]$의 음의 값들에 대해 $-LUT[(-(fHad[i] + (1 \ll (tblShift - 1))) \gg tblShift]$이고,

여기서 $tblShift$는 N의 가명이고, $fHad[i]$는 필터링될 스펙트럼 성분들인, 방법.

청구항 13

제10항에 있어서,

N은 3인, 방법.

청구항 14

제10항에 있어서,

QP들의 세트 내의 상기 제1 QP에 대해, N은 2와 동등하고, 상기 QP들의 세트는 상기 룩업 테이블(LUT)을 획득하기 위해 사용되고;

상기 QP들의 세트 내의 마지막 QP에 대해 또는 마지막 2개의 QP에 대해, N은 4와 동등하고;

상기 QP들의 세트로부터의 나머지 QP들에 대해, N은 3과 동등한, 방법.

청구항 15

제10항에 있어서,

N은 다음과 같이 정의되고;

$$tblShift = tblThrLog2[qpIdx] - 4$$

$$tblThrLog2[5] = \{ 6, 7, 7, 8, 8 \}$$

여기서 tblShift는 N의 가명이고, qpIdx는 QP에 기초하여 도출되는, 방법.

청구항 16

제15항에 있어서,

qpIdx는 다음과 같이 QP에 기초하여 도출되고;

$$if(pred_mode_flag[xCb][yCb] == 0 \ \&\& \ nCbW == nCbH \ \&\& \ min(nCbW, nCbH) >= 32)$$
$$qpIdx = Clip3(0, 4, (Qp_Y - 28 + (1 << 2)) >> 3)$$
else
$$qpIdx = Clip3(0, 4, (Qp_Y - 20 + (1 << 2)) >> 3)$$

여기서 Qp_Y는 현재 블록 QP이고, xCb, yCb는 화상 (x, y) 상의 현재 블록의 위치를 정의하고, $pred_mode_flag[xCb][yCb]$는 현재 블록의 예측 모드를 정의하고, 0과 동등하면 인터이고, 그렇지 않으면 인트라이고, $nCbW, nCbH$는 이에 대응하여 현재 블록의 폭 및 높이인, 방법.

청구항 17

제10항에 있어서,

N은 QP 값에 의존하는, 방법.

청구항 18

제1항 내지 제4항 중 어느 한 항에 있어서,

LUT 생성은 적어도 일부 양자화 파라미터(QP)에 대한 보조 함수에 기초하는, 방법.

청구항 19

제18항에 있어서,

상기 LUT 생성 $LUT(R_i, \sigma)$ 는:

$$LUT(R_i, \sigma) = \max\left(\frac{R_i^3}{R_i^2 + m * \sigma^2}, AuxiliaryFunc_{\sigma}(i)\right)$$

에 의해 도출되고, 여기서 $AuxiliaryFunc_{\sigma}(i)$ 는 상기 보조 함수를 표현하고, 상기 보조 함수는 마지막 LUT + 1의 인수에 대응하는 i에서 THR과 동등한 값을 갖는, 방법.

청구항 20

제18항에 있어서,

상기 보조 함수는 포인트 (i, THR) 및(a, 0)에 대해 나오는 직선 방정식일 수 있고, 여기서 a > 0이고 a는 필터링 파라미터 σ 또는 QP 값에 의존하는, 방법.

청구항 21

제20항에 있어서,

상기 LUT는

```
setOfLUT[ 5 ][ 16 ] =
      { 0, 0, 2,  6, 10, 14, 19, 23, 28, 32,  36,  41,  45,  49,  53,  57, },
      { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65,  73,  82,  90,  98, 107, 115, },
      { 0, 0, 1,  4,  9, 16, 24, 32, 41, 50,  59,  68,  77,  86,  94, 103, },
      { 0, 0, 3,  9, 19, 32, 47, 64, 81, 99, 117, 135, 154, 179, 205, 230, },
      { 0, 0, 0,  2,  6, 11, 18, 27, 38, 51,  64,  96, 128, 160, 192, 224, }
```

와 같이 정의되는, 방법.

청구항 22

제20항에 있어서,

상기 LUT는

```
setOfLUT[ 5 ][ 16 ] =
{ 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },
{ 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
{ 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
{ 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },
{ 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, }
```

와 같이 정의되는, 방법.

청구항 23

제20항에 있어서,

상기 LUT는

```
setOfLUT[ 5 ][ 16 ] =
{ 0, 2, 10, 19, 28, 36, 45, 53, 61, 70, 78, 86, 94, 102, 110, 118, },
{ 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
{ 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
{ 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },
{ 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, }
```

와 같이 정의되는, 방법.

청구항 24

제2항 내지 제4항 중 어느 한 항에 있어서,

상기 변환은 Hadamard 변환인, 방법.

청구항 25

제2항 내지 제4항 중 어느 한 항에 있어서,

상기 변환은 1D 변환인, 방법.

청구항 26

제1항 내지 제4항 중 어느 한 항에 따른 방법을 수행하기 위한 처리 회로를 포함하는 디코더.

청구항 27

제1항 내지 제4항 중 어느 한 항에 따른 방법을 수행하기 위한 처리 회로를 포함하는 인코더.

청구항 28

디코더로서,

명령어들을 포함하는 메모리 스토리지; 및

상기 메모리와 통신하는 하나 이상의 프로세서를 포함하고, 상기 하나 이상의 프로세서는 상기 명령어들을 실행하여 제1항 내지 제4항 중 어느 한 항에 따른 방법을 수행하는, 디코더.

청구항 29

인코더로서,

명령어들을 포함하는 메모리 스토리지; 및

상기 메모리와 통신하는 하나 이상의 프로세서를 포함하고, 상기 하나 이상의 프로세서는 상기 명령어들을 실행하여 제1항 내지 제4항 중 어느 한 항에 따른 방법을 수행하는, 인코더.

청구항 30

컴퓨터 또는 프로세서 상에서 실행될 때 제1항 내지 제4항 중 어느 한 항의 방법을 수행하기 위한 프로그램 코드를 포함하는, 컴퓨터-판독 가능 저장 매체 상에 저장된 컴퓨터 프로그램.

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

청구항 51

삭제

청구항 52

삭제

청구항 53

삭제

발명의 설명

기 술 분 야

[0001] 삭제

[0002] 삭제

[0003] 일반적으로, 본 발명은 비디오 코딩의 분야에 관한 것이다. 더 구체적으로, 본 발명은 비디오 코딩을 위한 필터, 재구성된 비디오 프레임들을 필터링하기 위한 방법, 및 비디오 블록들을 필터링하기 위한 방법뿐만 아니라, 비디오 코딩을 위한 그러한 필터를 포함하는 인코딩 장치 및 디코딩 장치에 관한 것이다.

배 경 기 술

[0004] DVD-디스크의 도입 이후 디지털 비디오가 널리 사용되고 있다. 송신 전에 비디오는 인코딩되고 송신 매체를 사용하여 송신된다. 시청자는 비디오를 수신하고 시청 디바이스를 사용하여 비디오를 디코딩하고 디스플레이한다. 수년에 걸쳐 비디오의 품질은, 예를 들어, 더 높은 해상도, 색 깊이 및 프레임 레이트 때문에 개선되었다. 이는 오늘날 인터넷 및 이동 통신 네트워크를 통해 흔히 전송되는 더 큰 데이터 스트림으로 이어졌다.

[0005] 그러나, 더 높은 해상도의 비디오들은 전형적으로 더 많은 대역폭을 필요로 하는데 그 이유는 그것들이 더 많은 정보를 가지고 있기 때문이다. 대역폭 요건을 감소시키기 위해, 비디오의 압축을 수반하는 비디오 코딩 표준들이 도입되었다. 비디오가 인코딩될 때 대역폭 요건(또는 스토리지의 경우 대응하는 메모리 요건)이 감소된다. 종종 이러한 감소는 품질의 희생을 수반한다. 따라서, 비디오 코딩 표준들은 대역폭 요건과 품질 간의 균형을 찾으려고 시도한다.

[0006] 품질을 개선하고 대역폭 요건을 감소시키기 위한 지속적인 요구가 있으므로, 감소된 대역폭 요건으로 품질을 유지하거나 대역폭 요건을 유지하면서 품질을 개선하는 솔루션들이 지속적으로 탐색된다. 더욱이, 때때로 타협이 수용될 수 있다. 예를 들어, 품질 개선이 상당하다면 대역폭 요건을 증가시키는 것이 수용될 수 있다.

[0007] HEVC(High Efficiency Video Coding)는 본 기술분야의 통상의 기술자들에게 흔히 알려져 있는 비디오 코딩 표준의 예이다. HEVC에서는, 코딩 단위(CU)를 예측 단위(PU)들 또는 변환 단위(TU)들로 분할한다. VVC(Versatile Video Coding) 차세대 표준은 JVET(Joint Video Exploration Team)로 알려진 파트너십에서 함께 작업하는, ITU-T VCEG(Video Coding Experts Group) 및 ISO/IEC MPEG(Moving Picture Experts Group) 표준화 기구들의 가장 최근의 공동 비디오 프로젝트이다. VVC는 ITU-T H.266/NGVC(Next Generation Video Coding) 표준이라고

도 지칭된다. VVC에서, 그것은 다수의 분할 유형들의 개념들을 제거하는데, 즉, 그것은 최대 변환 길이에 대해 너무 큰 크기를 갖는 CU들에 대해 필요한 경우를 제외하고는 CU, PU 및 TU 개념들의 분리를 제거하고, CU 분할 형상에 대해 더 많은 유연성을 지원한다.

[0008] 이미지의 특정 특징들을 강조하거나 필터링된 이미지의 객관적인 또는 지각 품질을 향상시키기 위해 이미지 필터링이 빈번하게 사용된다. 이미지 필터링은 다양한 잡음 소스들을 다루어야 한다. 따라서, 품질 향상을 위한 다양한 접근법들이 제안되었고 현재 사용되고 있다. 예를 들어, 적응 루프 필터(adaptive Loop filter, ALF) 방법에서, 각각의 재구성된 프레임은 작은 블록들(수퍼-픽셀들)의 세트로 분할되고 각각의 블록은 적응 루프 필터에 의해 필터링되는데, 그 이유는 필터링된 재구성된 프레임의 각각의 픽셀이 생성된 필터링된 픽셀의 위치 주위의 재구성된 프레임으로부터의 픽셀의 연결된 영역 내의 수 개의 픽셀들의 가중된 합이기 때문이다. 가중 계수들(필터 계수들이라고도 지칭됨)은 중심 대칭의 특성을 갖고 인코더로부터 디코더 측으로 송신된다. 에지들은 종종 큰 크기를 가지고 따라서 송신된 가중 계수들의 수는 효율적인 처리를 하기에 너무 많아질 수 있다. 많은 수의 가중 계수들은 송신을 위한 가중 계수들의 수를 감소시키기 위해 인코더 측에서 복잡한 레이트-왜곡 최적화(rate-distortion optimization, RDO)를 필요로 한다. 디코더 측에서 ALF는 범용 승산기들의 구현을 필요로 하고 이들 승산기는 각각의 2x2 픽셀 블록에 대해 재로딩되어야 한다.

[0009] 따라서, 낮은 복잡도로 예측 품질을 개선하고, 따라서, 비디오 코딩 효율을 증가시키는 것을 허용하는 개선된 필터 및 방법이 필요하다.

발명의 내용

[0010] 본 발명의 목적은 제한된 복잡도로 필터링 효율을 개선하고, 따라서, 비디오 코딩 효율을 증가시키는 것을 허용하는 개선된 필터 및 방법을 제공하는 것이다.

[0011] 전술한 그리고 다른 목적들은 독립 청구항들의 주제에 의해 달성된다. 추가 구현 형식들은 종속 청구항들, 설명 및 도면들로부터 명백하다.

[0012] 제1 양태에 따르면 본 발명은 비디오 코딩을 위한 필터에 관한 것으로, 상기 필터는 필터링된 블록의 생성을 위해 블록을 처리하도록 구성되고, 상기 블록은 복수의 픽셀을 포함한다. 상기 필터는 명령어들을 포함하는 메모리 스토리지; 및 상기 메모리와 통신하는 하나 이상의 프로세서를 포함한다. 상기 하나 이상의 프로세서는 상기 명령어들을 실행하여: 미리 정의된 스캔 템플릿에 따라 현재 픽셀 및 그것의 이웃 픽셀들을 선형 버퍼에 로딩하고; 상기 선형 버퍼 내의 픽셀들에 대해 1D 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득하고; 각각의 스펙트럼 성분을 이득 계수와 곱하는 것에 의해 필터링된 스펙트럼 성분들을 획득하고 - 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존함 -; 필터링된 스펙트럼 성분들에 대해 역 1D 변환을 수행하는 것에 의해 필터링된 픽셀들을 획득하고; 상기 필터링된 픽셀들에 기초하여 필터링된 블록을 생성한다.

[0013] 예로서, 상기 블록(또는 프레임)은 예측된 블록일 수 있고, 상기 필터링된 블록은 필터링된 예측된 블록이다. 다른 예로서, 상기 블록(또는 프레임)은 재구성된 블록일 수 있고, 상기 필터링된 블록은 필터링된 재구성된 블록이다.

[0014] 예로서, 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터의 함수이다. 상기 필터링 파라미터는 코덱 양자화 파라미터(QP)로부터 도출될 수 있다.

[0015] 다른 예에서, 제1 스펙트럼 성분에 대한 이득 계수 $G(i,\sigma)$가 1과 동등할 때 제1 스펙트럼 성분은 필터링 없이 바이패싱된다. 상기 제1 스펙트럼 성분은 상기 선형 버퍼 내의 샘플들의 합 또는 평균 값에 대응하고, 상기 제1 스펙트럼 성분은 DC에 대응할 수 있다.

[0016] 다른 예로서, 상기 하나 이상의 프로세서는 상기 명령어들을 실행하여: 상기 LUT의 테이블 값들로부터 N개의 비트를 드로핑하고, N은 정수이다. N은 QP 값에 의존할 수 있거나, 고정 값이다.

[0017] 예로서, 상기 미리 정의된 스캔 템플릿은 상기 재구성된 블록 내부의 현재 픽셀의 위치에 대한 공간적 또는 래스터 오프셋들의 세트로서 정의된다. 이웃 픽셀들을 지시하는 오프셋들은 상기 재구성된 블록 내부에 있다. 적어도 하나의 필터링된 픽셀은 상기 미리 정의된 스캔 템플릿에 따라 그것의 원래 위치에 배치될 수 있다. 모든 필터링된 픽셀들은 상기 미리 정의된 스캔 템플릿에 따라 누적 버퍼에 추가되고, 상기 누적 버퍼는 필터링된 스펙트럼 성분들을 획득하기 전에 0으로 초기화될 수 있다. 상기 누적 버퍼 내의 누적된 값들을 상기 누적 버퍼의 현재 위치에 추가하는 픽셀들의 수로 나눈 것으로서 최종 필터링된 픽셀들이 획득되고; 하나 이상의 프로

세서는 상기 최종 필터링된 픽셀들에 기초하여 상기 필터링된 재구성된 블록을 생성하도록 구성된다.

[0018] 옵션으로, 모든 필터링된 픽셀들과 대응하는 필터링되지 않은 픽셀들 간의 차이들이 상기 미리 정의된 스캔 템플릿에 따라 누적 버퍼에 추가되고, 상기 누적 버퍼는 필터링되지 않은 픽셀들을 상기 블록에서 추가될 픽셀 값들의 최대 수와 곱하는 것에 의해 초기화될 수 있다. 상기 최종 필터링된 픽셀들은 상기 누적 버퍼 내의 누적된 값들을 상기 블록에서 추가될 픽셀 값들의 최대 수로 나눈 것으로서 획득된다.

[0019] 제2 양태에 따르면 본 발명은 필터링된 블록의 생성을 위해 블록을 처리하기 위한 대응하는 필터링 방법에 관한 것이고, 상기 블록은 복수의 픽셀을 포함한다. 각각의 픽셀은 픽셀 값과 연관된다. 상기 필터링 방법은: 미리 정의된 스캔 템플릿에 따라 현재 픽셀 및 그것의 이웃 픽셀들을 선형 버퍼에 로딩하는 단계; 상기 선형 버퍼 내의 픽셀들에 대해 1D 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득하는 단계; 각각의 스펙트럼 성분을 이득 계수와 곱하는 것에 의해 필터링된 스펙트럼 성분들을 획득하는 단계 - 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존함 -; 필터링된 스펙트럼 성분들에 대해 역 1D 변환을 수행하는 것에 의해 필터링된 픽셀들을 획득하는 단계; 및 상기 필터링된 픽셀들에 기초하여 필터링된 블록을 생성하는 단계를 포함한다.

[0020] 예로서, 상기 블록(또는 프레임)은 예측된 블록일 수 있고, 상기 필터링된 블록은 필터링된 예측된 블록이다. 다른 예로서, 상기 블록(또는 프레임)은 재구성된 블록일 수 있고, 상기 필터링된 블록은 필터링된 재구성된 블록이다.

[0021] 예로서, 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터의 함수이다. 상기 필터링 파라미터는 코덱 양자화 파라미터(QP)로부터 도출될 수 있다.

[0022] 다른 예에서, 제1 스펙트럼 성분에 대한 이득 계수 $G(i,\sigma)$가 1과 동등할 때 제1 스펙트럼 성분은 필터링 없이 바이패싱된다. 상기 제1 스펙트럼 성분은 DC 값에 대응할 수 있다.

[0023] 다른 예로서, 룩업 테이블(LUT)에 기초하여 상기 스펙트럼 성분들을 필터링하는 것. LUT 생성은 적어도 일부 양자화 파라미터(QP)에 대한 보조 함수에 기초할 수 있다. 보조 함수는 포인트 (i,THR) 및 (a, 0)에 대해 나오는 직선 방정식일 수 있고, 여기서 a > 0이고 a는 필터링 파라미터 σ 또는 QP 값에 의존한다. 예를 들어, QP들의 세트 내의 마지막 QP에 대해, a는 11과 동등하다; 또는 QP들의 세트 내의 두 번째 마지막 QP에 대해, a는 9와 동등하다.

[0024] 다른 예로서, 상기 방법은: 상기 LUT의 테이블 값들로부터 N개의 비트를 드로핑하는 단계를 추가로 포함하고, N은 정수이다. N은 QP 값에 의존할 수 있거나, 고정 값이다. QP들의 세트로부터의 더 높은 QP에 비해 더 낮은 QP에 대해 더 적게 N이 선택될 때, 예를 들어, QP들의 세트 내의 첫 번째 QP에 대해, N은 2와 동등하다; 또는 QP들의 세트로부터의 나머지 QP들에 대해, N은 3과 동등하다. 대안적으로, QP들의 세트로부터의 더 낮은 QP에 비해 더 높은 QP에 대해 더 많게 N이 선택될 때, 예를 들어, QP들의 세트 내의 마지막 QP에 대해 또는 마지막 2개의 QP에 대해, N은 4와 동등하다; 또는 QP들의 세트로부터의 나머지 QP들에 대해, N은 3과 동등하다. 대안적으로, QP들의 세트로부터의 나머지 QP에 비해 더 낮은 QP에 대해 더 적게 그리고 더 높은 QP에 대해 더 많게 N이 선택될 때, 예를 들어, QP들의 세트 내의 첫 번째 QP에 대해, N은 2와 동등하다; QP들의 세트 내의 마지막 QP에 대해 또는 마지막 2개의 QP에 대해, N은 4와 동등하다; 또는, QP들의 세트로부터의 나머지 QP들에 대해, N은 3과 동등하다.

[0025] 예로서, 상기 미리 정의된 스캔 템플릿은 상기 재구성된 블록 내부의 현재 픽셀의 위치에 대한 공간적 또는 래스터 오프셋들의 세트로서 정의된다. 이웃 픽셀들을 지시하는 오프셋들은 상기 재구성된 블록 내부에 있다. 적어도 하나의 필터링된 픽셀은 상기 미리 정의된 스캔 템플릿에 따라 그것의 원래 위치에 배치될 수 있다. 모든 필터링된 픽셀들은 상기 미리 정의된 스캔 템플릿에 따라 누적 버퍼에 추가되고, 상기 누적 버퍼는 필터링된 스펙트럼 성분들을 획득하기 전에 0으로 초기화될 수 있다. 상기 누적 버퍼 내의 누적된 값들을 상기 누적 버퍼의 현재 위치에 추가하는 픽셀들의 수로 나눈 것으로서 최종 필터링된 픽셀들이 획득되고; 하나 이상의 프로세서는 상기 최종 필터링된 픽셀들에 기초하여 상기 필터링된 재구성된 블록을 생성하도록 구성된다.

[0026] 옵션으로, 모든 필터링된 픽셀들과 대응하는 필터링되지 않은 픽셀들 간의 차이들이 상기 미리 정의된 스캔 템플릿에 따라 누적 버퍼에 추가되고, 상기 누적 버퍼는 필터링되지 않은 픽셀들을 상기 블록에서 추가될 픽셀 값들의 최대 수와 곱하는 것에 의해 초기화될 수 있다. 상기 최종 필터링된 픽셀들은 상기 누적 버퍼 내의 누적된 값들을 상기 블록에서 추가될 픽셀 값들의 최대 수로 나눈 것으로서 획득된다.

[0027] 제3 양태에 따르면 본 발명은 입력 비디오 스트림으로부터 현재 프레임을 인코딩하기 위한 인코딩 장치에 관한 것이고, 이 인코딩 장치는 본 발명의 제1 양태에 따른 필터를 포함한다.

[0028] 제4 양태에 따르면 본 발명은 수신된 비트스트림으로부터 현재 프레임을 디코딩하기 위한 디코딩 장치에 관한 것이고, 이 디코딩 장치는 본 발명의 제1 양태에 따른 필터를 포함한다.

[0029] 제5 양태에 따르면 본 발명은 컴퓨터 상에서 실행될 때 제2 양태에 따른 방법을 수행하기 위한 프로그램 코드를 포함하는 컴퓨터 프로그램에 관한 것이다.

[0030] 따라서, 비디오 코딩을 위한 효율을 개선하는 것을 허용하는 필터가 제공된다. 더 구체적으로, 본 발명의 실시예들에 따른 개선된 필터는 필터 파라미터 시그널링 없이 프레임 자체로부터 필터 파라미터들을 추정하고, 따라서, 종래의 필터들보다 상당히 적은 시그널링을 필요로 하고, 이는 이미지 도메인에서 필터링하기 위한 가중 계수들을 시그널링한다.

도면의 간단한 설명

[0031] 본 발명의 추가 실시예들은 다음의 도면들에 관하여 설명될 것이다.

도 1a는 실시예에 따른 필터를 포함하는 실시예에 따른 인코딩 장치를 예시하는 개략도를 도시한다.

도 1b는 실시예에 따른 필터를 포함하는 실시예에 따른 인코딩 장치를 예시하는 개략도를 도시한다.

도 2a는 실시예에 따른 필터를 포함하는 실시예에 따른 디코딩 장치를 예시하는 개략도를 도시한다.

도 2b는 실시예에 따른 필터를 포함하는 실시예에 따른 디코딩 장치를 예시하는 개략도를 도시한다.

도 3a는 실시예에 따른 필터에서 구현되는 필터링 프로세스의 양태들을 예시하는 개략도를 도시한다.

도 3b는 실시예에 따른 필터에서 구현되는 필터링 프로세스의 양태들을 예시하는 개략도를 도시한다.

도 3c는 다른 실시예에 따른 필터에서 구현되는 필터링 프로세스의 양태들을 예시하는 개략도를 도시한다.

도 4a는 정사각형 재구성된 블록 내부의 상이한 픽셀 위치에 대한 템플릿들을 예시한다.

도 4b는 하나의 픽셀에 대한 등가 필터 형상을 예시한다.

도 4c는 패딩의 예를 제공한다.

도 4d는 패딩의 다른 예를 제공한다.

도 5a는 실시예에 따른 필터링 방법의 단계들을 예시하는 흐름도를 도시한다.

도 5b는 실시예에 따른 필터링 방법의 단계들을 예시하는 흐름도를 도시한다.

도 6은 SRAM에 기초한 예시적인 하드웨어 필터 설계이다.

도 7은 플립-플롭에 기초한 2x2 그룹 필터링의 예시적인 하드웨어 설계이다.

도 8은 4개의 2x2 그룹 필터링의 결과들을 동일한 공간적 그룹 최종 필터의 결과들의 재사용과 조합하는 예를 도시한다.

도 9는 LUT를 최적화한 결과를 예로서 도시한다.

도 10은 필터 전달 함수에서의 바람직하지 않은 갭을 예로서 도시한다.

도 11은 테이블 항목들이 하나하나 플롯되어 있는 동일한 테이블을 예로서 표현한다.

도 12는 보조 함수를 사용하여 갭이 어떻게 제거될 수 있는지에 대한 방법을 예로서 예시한다.

도 13은 LUT를 생성하는 동안 2개의 값으로부터 최대값을 취하는 것에 의해 갭을 제거하는 예를 예시한다.

도 14는 위에 설명된 방법을 적용한 후의 필터 전달 함수를 예로서 예시한다.

도 15는 세트 내의 5개의 QP에 따른 필터 전달 함수의 예를 예시한다.

도 16은 대응하는 테이블들에 기초한 세트 내의 5개의 QP에 대한 필터 전달 함수의 예를 예시한다.

도 17은 대응하는 테이블들에 기초한 세트 내의 5개의 QP에 대한 필터 전달 함수의 예를 예시한다.

도 18은 실시예에 따른 장치의 예시적인 구조를 예시하는 개략도이다.

다양한 도면들에서, 동일한 참조 부호들은 동일하거나 기능적으로 동등한 특징들에 대해 사용될 것이다.

발명을 실시하기 위한 구체적인 내용

[0032] 다음의 설명에서, 본 개시의 일부를 형성하고, 본 발명이 배치될 수 있는 특정 양태들이 예시로서 도시되는 첨부 도면들이 참조된다. 본 발명의 범위를 벗어나지 않고 다른 양태들이 이용될 수 있고 구조적 또는 논리적 변화들이 이루어질 수 있다는 것이 이해된다. 따라서, 본 발명의 범위는 첨부된 청구항들에 의해 정의되므로, 다음의 상세한 설명은 제한적인 의미로 간주되어서는 안 된다.

[0033] 예를 들어, 설명된 방법과 관련한 개시가 또한 그 방법을 수행하도록 구성된 대응하는 디바이스 또는 시스템에 대해 유효할 수 있고 그 반대도 마찬가지라는 것이 이해된다. 예를 들어, 특정 방법 단계가 설명된다면, 대응하는 디바이스는 설명된 방법 단계를 수행하는 유닛을 포함할 수 있다(비록 그러한 유닛이 명시적으로 설명되거나 도면들에 예시되지 않는 경우에도). 또한, 구체적으로 달리 언급되지 않는 한, 본 명세서에 설명된 다양한 예시적인 양태들의 특징들은 서로 조합될 수 있다는 것이 이해된다.

[0034] 도 1a는 실시예에 따른 필터(120)를 포함하는 실시예에 따른 인코딩 장치(100)를 도시한다. 인코딩 장치(100)는 복수의 프레임(본 명세서에서 화상 또는 이미지라고도 지칭됨)을 포함하는 비디오 신호의 프레임의 블록을 인코딩하도록 구성되고, 여기서 각각의 프레임은 복수의 블록으로 분할가능하고, 각각의 블록은 복수의 픽셀을 포함한다. 일 실시예에서, 블록들은 매크로 블록들, 코딩 트리 단위들, 코딩 단위들, 예측 단위들 및/또는 예측 블록들일 수 있다.

[0035] 본 개시에서 용어 "블록"은 임의의 유형 블록에 대해 또는 임의의 깊이 블록에 대해 사용되고, 예를 들어, 용어 "블록"은 루트 블록, 블록, 서브-블록, 리프 노드 등에 포함되지만 이에 제한되지 않는다. 코딩될 블록들은 반드시 동일한 크기를 갖는 것은 아니다. 하나의 화상은 상이한 크기들의 블록들을 포함할 수 있고 비디오 시퀀스의 상이한 화상들의 블록 래스터도 상이할 수 있다.

[0036] 도 1a에 도시된 예시적인 실시예에서, 인코딩 장치(100)는 하이브리드 비디오 코딩 인코더의 형식으로 구현된다. 통상적으로, 비디오 신호의 제1 프레임은 인트라 예측만을 사용하여 인코딩되는 인트라 프레임이다. 이를 위해, 도 1a에 도시된 인코딩 장치(100)의 실시예는 인트라 예측을 위한 인트라 예측 유닛(154)을 포함한다. 인트라 프레임은 다른 프레임들로부터의 정보 없이 디코딩될 수 있다. 인트라 예측 유닛(154)은 인트라 추정 유닛(152)에 의해 제공되는 정보에 기초하여 블록의 인트라 예측을 수행할 수 있다.

[0037] 제1 인트라 프레임 다음의 후속 프레임들의 블록들은, 모드 선택 유닛(160)에 의해 선택되는 바와 같이, 인터 또는 인트라 예측을 사용하여 코딩될 수 있다. 이를 위해, 도 1a에 도시된 인코딩 장치(100)는 인터 예측 유닛(144)을 추가로 포함한다. 일반적으로, 인터 예측 유닛(144)은 인터 추정 유닛(142)에 의해 제공되는 모션 추정에 기초하여 블록의 모션 보상을 수행하도록 구성될 수 있다.

[0038] 더욱이, 인트라 또는 인터 예측으로부터 획득되는 도 1b에 도시된 하이브리드 인코더 실시예에서의 예측 신호는 필터(145)에 의해 더 필터링된다.

[0039] 더욱이, 도 1a 및 도 1b에 도시된 하이브리드 인코더 실시예에서, 잔차 계산 유닛(104)은 원래 블록과 그것의 예측 간의 차이, 즉, 인트라/인터 화상 예측의 예측 오류를 정의하는 잔차 블록을 결정한다. 이 잔차 블록은 변환 유닛(106)에 의해 변환되고(예를 들어, DCT를 사용하여) 변환 계수들은 양자화 유닛(108)에 의해 양자화된다. 양자화 유닛(108)의 출력뿐만 아니라, 예를 들어, 인트라 예측 유닛(154), 인터 예측 유닛(144) 및 필터(120)에 의해 제공되는 코딩 또는 부가 정보는 엔트로피 인코딩 유닛(170)에 의해 더 인코딩된다.

[0040] 하이브리드 비디오 인코더는 통상적으로 디코더 처리를 복제하여 둘 다 동일한 예측을 생성하도록 한다. 따라서, 도 1에 도시된 실시예에서, 역양자화 유닛(110) 및 역변환 유닛은 변환 유닛(106) 및 양자화 유닛(108)의 역동작들을 수행하고 잔차 블록의 디코딩된 근사치를 복제한다. 그 후 디코딩된 잔차 블록 데이터는 재구성 유닛(114)에 의해 예측의 결과들, 즉, 예측 블록에 추가된다. 그 후, 재구성 유닛(114)의 출력은 인트라 예측을 위해 사용되도록 라인 버퍼(116)에 제공될 수 있고, 필터(120)에 의해 더 처리되고, 이에 대해서는 아래에 더 상세히 기술될 것이다. 최종 화상은 디코딩된 화상 버퍼(130)에 저장되고, 후속 프레임들의 인터 예측에 이용될 수 있다.

[0041] 도 2a는 실시예에 따른 필터(220)를 포함하는 실시예에 따른 디코딩 장치(200)를 도시한다. 이 디코딩 장치(200)는 인코딩된 비디오 신호의 프레임의 블록을 디코딩하도록 구성된다. 도 2a에 도시된 실시예에서 디코딩 장치(200)는 하이브리드 디코더로서 구현된다. 엔트로피 디코딩 유닛(204)은, 특히, 인트라 예측 유닛(254) 및 인터 예측 유닛(244)뿐만 아니라, 필터(220)와 같은, 디코딩 장치(200)의 다른 컴포넌트들에 필요한, 일반적으로 예측 오류들(즉, 잔차 블록들), 모션 데이터 및 다른 부가 정보를 포함할 수 있는, 인코딩된 화상 데이터의 엔트로피 디코딩을 수행한다. 일반적으로, 도 2a에 도시된 디코딩 장치(200)의 인트라 예측 유닛(254) 및 인터 예측 유닛(244)은 모드 선택 유닛(260)에 의해 선택되고 도 1에 도시된 인코딩 장치(100)의 인트라 예측 유닛(154) 및 인터 예측 유닛(144)과 동일한 방식으로 기능하고, 따라서 인코딩 장치(100) 및 디코딩 장치(200)에 의해 동일한 예측들이 생성될 수 있다. 디코딩 장치(200)의 재구성 유닛(214)은 역양자화 유닛(210) 및 역변환 유닛(212)에 의해 제공되는 잔차 블록 및 필터링된 예측된 블록에 기초하여 블록을 재구성하도록 구성된다. 인코딩 장치(100)의 경우에서와 같이, 재구성된 블록은 인트라 예측을 위해 사용되는 라인 버퍼(216)에 제공될 수 있고 필터링된 블록/프레임은 인터 예측을 위해 필터(220)에 의해 디코딩된 화상 버퍼(230)에 제공될 수 있다.

[0042] 도 2b에 도시된 바와 같이, 인트라 또는 인터 예측으로부터 획득되는 도 2b에 도시된 하이브리드 디코더 실시예에서의 예측 신호는 필터(264)에 의해 더 필터링된다.

[0043] 이미 위에 설명된 바와 같이, 필터(120, 220)는 프레임 레벨에서 사용될 수 있는데, 예를 들어, 필터(120, 220)는 디코딩된 재구성된 비디오 스트림으로부터 재구성된 프레임을 처리하여 필터링된 재구성된 프레임을 생성하도록 구성될 수 있고, 여기서 재구성된 프레임은 복수의 블록을 포함한다. 필터(120, 220)는 또한 전체 프레임을 기다리지 않고 블록 재구성 직후에 블록 레벨에서 사용될 수 있는데, 예를 들어, 필터(120, 220)는 재구성된 블록을 처리하여 필터링된 재구성된 블록을 생성하도록 구성될 수 있고, 여기서 재구성된 블록은 복수의 픽셀을 포함한다.

[0044] 필터(120, 220, 145 또는 264)는 하나 이상의 프로세서(또는 하나 이상의 처리 유닛)를 포함한다. 아래에 더 상세히 설명되는 바와 같이, 하나 이상의 프로세서(또는 하나 이상의 처리 유닛)는: 미리 정의된 스캔 템플릿(다시 말해서, 스캔 순서, 또는 스캔 패턴)에 따라 현재 픽셀 및 그것의 이웃 픽셀들을 선형 버퍼에 로딩하고; 상기 선형 버퍼 내의 각각의 픽셀들에 대해 1D 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득하고; 각각의 스펙트럼 성분을 이득 계수와 곱하는 것에 의해 필터링된 스펙트럼 성분들을 획득하고 - 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존함 -; 필터링된 스펙트럼에 대해 역 1D 변환을 수행하는 것에 의해 필터링된 픽셀들을 획득하고; 이전 처리 단계들에서 추정된 상기 필터링된 픽셀들에 기초하여 필터링된 재구성된 블록을 생성하도록 구성된다. 예에서, 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존한다. 다른 예에서, 이득 계수는 하나 이상의 필터링 파라미터 및 하나 이상의 대응하는 스펙트럼 성분에 의존한다. 다른 예에서, 각각의 이득 계수는 하나 이상의 필터링 파라미터 및 대응하는 스펙트럼 성분뿐만 아니라 스펙트럼 성분의 좌측 및 우측의 이웃 스펙트럼 성분들에 의존할 수 있다.

[0045] 본 개시는 재구성된 프레임으로부터의 재구성된 블록의 로컬 및/또는 비-로컬 필터링을 수행하는 손실 비디오 코덱에 대한 루프 내 필터(in-loop filter)를 설명한다. 예에 따르면, 재구성된 프레임은 작은 비중첩 직사각형 블록들(CU 블록들)의 세트로 분할된다. 다음 단계에서 각각의 재구성된 블록(재구성된 CU 블록)은 다른 재구성된 블록들과 독립적으로 주파수 도메인에서 필터링된다. 필터는 또한 변환 및 재구성 후에 적용될 수 있고, 필터링된 결과는 출력뿐만 아니라 공간적 및 시간적 예측 둘 다를 위해 이용된다.

[0046] 다른 예로서, 본 개시는 재구성된 프레임의 예측 블록의 로컬 및/또는 비-로컬 필터링을 수행하는 손실 비디오 코덱에 대한 예측 필터를 설명한다.

[0047] 제1 처리 단계에서 재구성된 블록 내부의 모든 픽셀들은 서로 독립적으로 처리될 수 있다. 픽셀 r(0)의 처리를 위해, 이웃 픽셀들이 사용된다. 예를 들어, 도 3a에 예시된 바와 같이, 픽셀들 r(1) 내지 r(7)이 사용되고, 픽셀들 r(0) 내지 r(7)은 하나의 처리 그룹을 형성한다.

[0048] 도 3a 또는 도 3b는 실시예에 따른 필터에서 구현되는 필터링 프로세스의 양태들을 예시하는 개략도(300, 300')를 도시한다. 단계 302, 302'에서는, 블록으로부터의 현재 픽셀 및 그것의 이웃 픽셀들이 미리 정의된 스캔 템플릿에 따라 선형 버퍼에 로딩된다. 예로서, 블록은 예측된 블록일 수 있다. 다른 예로서, 블록은 재구성된 블록일 수 있다.

[0049] 단계 304, 304'에서는, 스펙트럼 성분들 R을 획득하기 위해 선형 버퍼 내의 픽셀 r(0) 및 그것의 이웃 픽셀들 r(1) 내지 r(7)에 대해 1D 변환이 수행된다:

[0050] $$R = 1D_Transform(r)$$

[0051] 예로서, 1D 변환은 Hadamard 변환일 수 있다.

[0052] 행의 4개의 픽셀(예를 들어, 도 3b의 예에서와 같이 픽셀들 A, B, C, D)에 대해 1D 변환을 수행할지 또는 공간적으로 위치한 픽셀들 A, B, C, D에 대해 2D 변환을 수행할지는 구현 특정과 관련된다는 것을 이해해야 한다. 2x2 블록에 위치한 4개의 픽셀 A, B, C, D에 2D 변환을 적용하는 것은 행으로서 취해지는 4개의 픽셀 A, B, C, D에 1D 변환을 적용하는 것과 동일한 결과로 이어질 수 있다. 단계 306, 306'에서는, 필터링된 스펙트럼 성분들 $F(i)$를 획득하기 위해 각각의 스펙트럼 성분 $R(i)$와 대응하는 이득 계수 $G(i,\sigma)$의 곱셈에 기초하여 주파수 도메인에서 필터링이 수행된다.

[0053] 모든 스펙트럼 성분들에 대한 이득 계수들의 세트는 필터의 주파수

[0054] $$F(i) = R(i) * G(i,\sigma) \qquad (1)$$

[0055] 임펄스 응답이다.

[0056] 예에서, 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존한다. 다른 예에서, 이득 계수는 하나 이상의 필터링 파라미터 및 하나 이상의 대응하는 스펙트럼 성분에 의존한다. 다른 예에서, 각각의 이득 계수는 하나 이상의 필터링 파라미터 및 대응하는 스펙트럼 성분뿐만 아니라 스펙트럼 성분의 좌측 및 우측의 이웃 스펙트럼 성분들에 의존할 수 있다. 각각의 이득 계수가 재구성된 블록의 스펙트럼 성분 및 필터링 파라미터의 함수이거나, 또는 예측된 블록의 스펙트럼 성분 및 필터링 파라미터의 함수이면, 이득 계수 $G(i,\sigma)$는 예로서 다음의 공식에 의해 기술될 수 있다:

[0057] $$G(i,\sigma) = \frac{R(i)^2}{R(i)^2 + m*\sigma^2} \qquad (2)$$

[0058] 여기서, (i)는 스펙트럼 성분의 인덱스이고, R(i)는 (i) 인덱스에 대응하는 스펙트럼 성분이고, $G(i,\sigma)$는 R(i)에 대응하는 이득 계수이고, σ는 필터링 파라미터이고, m은 스펙트럼 성분들의 수와 동등한 정규화 상수이다. 예를 들어, m은 1, 2, 3, 4 ...이다. 상이한 스펙트럼 성분들은 동일한 이득 계수를 가질 수 있거나, 상이한 이득 계수들을 가질 수 있다.

[0059] 변환 블록의 입력 샘플들의 평균(FFT, DCT, DST 등) 또는 합(Hadamard)에 대응하는 스펙트럼 성분들(통상적으로 DC 값에 대응하는 제1 성분)을 갖는 해당 변환들에 대해, 필터링된 블록의 평균 휘도의 변화를 회피하기 위해 1과 동등한 필터링 계수를 갖는 것이 유리할 수 있다. 이는 DC 값에 대응하는 제1 스펙트럼 성분에 대한 바이패싱(필터링 없음)을 의미한다.

[0060] 필터링 파라미터로서의 σ는, 예를 들어, 다음의 공식을 사용하여, 인코더 측 및 디코더 측의 코덱 양자화 파라미터(QP)로부터 도출될 수 있다:

[0061] $$\sigma = k * 2^{(n*(QP-s))}$$

[0062] 여기서 k, n 및 s는 값들을 갖는 상수들이고 예로서: k = 2.64, n = 0.1296, s = 11이다.

[0063] 상이한 스펙트럼 성분들은 동일한 필터링 파라미터를 가질 수 있거나, 상이한 필터링 파라미터들을 가질 수 있다.

[0064] 파라미터들 k, n 및 s는 최근 비디오 코딩 표준들에서 QP 값이 6만큼 증가할 때마다 2배가 되는 양자화 스텝 크기에 시그마가 의존하게 하는 방식으로 선택될 수 있다. 파라미터들 k = 0.5, $n = \frac{1}{6}$ 및 s = 0인 예에서 σ 파라미터들은 다음과 같이 도출된다:

[0065] $$\sigma = 0.5 * 2^{\left(\frac{QP}{6}\right)}$$

[0066] 양자화 스케일링 행렬들은 비디오 압축 품질을 개선하기 위해 널리 사용된다. 이 방법에서 QP에 기초하여 도출된 양자화 스텝 크기는 비트스트림에서 송신된 스케일링 인자와 곱해진다. 그러한 방법에 대해 σ 파라미터 도출은 특정 QP에 대해 사용되는 실제 스케일링된 양자화 스텝 크기에 기초할 수 있다:

[0067] $$\sigma = k * \mathbf{Quantization_step_size(QP - s)}$$

[0068] 상수들 k, n 및 s는 σ 계산을 위한 고정 값들을 가질 수 있거나, QP, 블록 크기 및 형상, 현재 블록에 대한 예측의 유형(인터/인트라)에 따라 상이한 값들을 가질 수 있다. 예를 들어 32x32 이상의 크기를 갖는 인트라 예측된 정사각형 블록들에 대해 파라미터 s는 s = 11 + 8 = 19로서 계산될 수 있다. 등가 필터 파라미터로서 σ는 통상적으로 평탄한 영역들에 대응하는 큰 정사각형 인트라 예측된 블록에 더 적합한 더 소프트한 필터로 이어지는 더 작은 값을 갖는다. 다른 예로서, k는 픽셀들의 비트 깊이에 기초하여 수정될 수 있고 $k_{mod} = k^*(1 << (\text{bit_depth} - 8))$ 이다.

[0069] 방법(300, 300')에 따르면, 각각의 주파수에 대한 이득 계수는 재구성된 픽셀들 또는 예측된 픽셀들의 스펙트럼 성분으로부터 도출된다. 따라서, 방법(300, 300')은 필터링 파라미터들의 송신을 필요로 하지 않고 추가적인 시그널링 없이 임의의 재구성된 블록 또는 예측된 블록에 대해 적용될 수 있다.

[0070] LUT 상세 사항들이 논의된다.

[0071] 필터링은 항상 1보다 작은 스케일링 계수에 대한 스펙트럼 성분 R(i)의 곱셈을 암시한다는 점에 유의할 수 있다. 또한 높은 값들의 R(i)에서 스케일링 계수는 1에 가깝다는 것이 관찰될 수 있다. 이들 관찰에 기초하여 스펙트럼 필터링은 곱셈들 및 나눗셈들을 필터링 동작들로부터 제외하는 것을 허용하는 룩업 테이블을 사용하여 구현된다:

[0072] 스펙트럼 이득 계수는 1보다 작으므로, 필터링은 다음의 공식들에 따라 짧은 룩업 테이블(LUT) 판독에 기초하여 구현될 수 있다:

[0073] $$F(i,\sigma) = \begin{cases} R(i) \ , \ Abs\big(R(i)\big) \geq THR \\ LUT(R(i),\sigma), \ R(i) > 0 \\ -LUT(-R(i),\sigma), \ R(i) \leq 0 \end{cases}$$

[0074] 여기서 $LUT(R(i),\sigma) = \frac{R(i)^3}{R(i)^2 + \mathrm{m}*\sigma^2}$ 이다. (i)는 스펙트럼 성분의 인덱스이고, R(i)는 (i) 인덱스에 대응하는 스펙트럼 성분이고, σ는 필터링 파라미터이고, THR은 임계값이고, m은 스펙트럼 성분들의 수와 동등한 정규화 상수이다. 예를 들어, m은 1, 2, 3, 4 ...이다.

[0075] 예로서, THR은 다음의 공식으로부터 계산될 수 있고,

[0076] $$\frac{THR^2}{THR^2 + m*\sigma^2} = \mathrm{C}$$

[0077] 여기서 C는 1에 가까운 값, 예를 들어, 0.8 또는 0.9이다. LUT 크기를 감소시키기 위해 임계값 THR은 QP 값에 의존할 수 있다.

[0078] LUT 크기를 더 감소시키기 위해 작은 필터링된 값들을 0으로 대체하기 위한 제2 임계값이 도입될 수 있다. 그 경우 필터링된 스펙트럼 성분이 $F(i,\sigma)$ 이고 다음과 같이 추가로 도출된다:

$$F(i,\sigma)=\begin{cases} R(i)\ ,\ Abs\big(R(i)\big)>THR \\ 0,\quad Abs\big(R(i)\big)<THR2 \\ LUT(R(i),\sigma),\ R(i)>0 \\ -LUT(-R(i),\sigma),\ R(i)\le 0 \end{cases}$$

[0079]

[0080] 여기서 THR2는 그 아래에서 필터링된 스펙트럼 성분이 0인 것으로 간주되는 임계값을 정의한다. 제2 THR2도 QP 값에 따라 정의될 수 있다.

[0081] 주파수 도메인에서 필터링한 후에, 단계 308에서 필터링된 스펙트럼 성분에 대해 역 1D 변환을 수행하여 필터링된 픽셀들 $\boldsymbol{f}$ 를 획득한다:

[0082] $$\boldsymbol{f = 1D_Inverse_Transform(F)}$$

[0083] 단계 310, 310'에서는, 역 1D 변환의 결과가 필터링된 재구성된 픽셀들 또는 필터링된 픽셀들의 선형 버퍼에 배치된다.

[0084] 단계 312, 312'(도 3a 또는 도 3b에 도시되지 않음)에서는, 이전 처리 단계들에서 추정된 필터링된 픽셀들에 기초하여 필터링된 블록이 생성된다. 예로서, 필터링된 블록은 필터링된 예측된 블록일 수 있다. 다른 예로서, 필터링된 블록은 필터링된 재구성된 블록일 수 있다.

[0085] 실시예로서 도 3a에 도시된 바와 같이, 필터링 단계 306 후에, 필터링된 픽셀 f(0)는 미리 정의된 스캔 템플릿에 따라 그것의 원래 위치에 배치된다. 다른 필터링된 샘플들 f(1) 내지 f(7)은 사용되지 않는다. 다른 실시예에서, 필터링된 샘플들의 선형 버퍼로부터의 하나보다 많은 필터링된 픽셀들, 예를 들어, 모든 필터링된 픽셀들이 도 3a의 단계 302에서 사용된 미리 정의된 스캔 템플릿에 따라 누적 버퍼에 추가된다. 누적 버퍼는 필터링 단계 전에 0으로 초기화되어야 한다. 마지막 정규화 단계에서, 최종 필터링된 픽셀들은 누적 버퍼 내의 누적된 값들을 누적 버퍼의 현재 위치에 추가된 픽셀들의 수, 다시 말해서, 이전 처리 단계들에서 누적 버퍼의 현재 위치에 추가된 픽셀 값들의 수로 나눈 것으로서 획득된다. 그 후, 최종 필터링된 픽셀들에 기초하여 필터링된 재구성된 블록 또는 예측된 블록이 생성된다.

[0086] 다른 실시예로서, 필터는 인트라 및 인터 코딩 단위(CU) 필터링 둘 다에 대해 동일한 구현을 갖는다.

[0087] Hadamard 변환 도메인 필터는, 4x4 블록을 제외하고 슬라이스 양자화 파라미터가 17보다 크다면, 0이 아닌 변환 계수를 갖는 루마 재구성된 블록들에 항상 적용된다. 필터 파라미터들은 코딩된 정보로부터 명시적으로 도출된다. 제안된 필터는, 적용되면, 블록 재구성 직후에 디코딩된 샘플들에 대해 수행된다. 필터링된 결과는 출력뿐만 아니라 공간적 및 시간적 예측 둘 다를 위해 이용된다.

[0088] 필터링 프로세스는 도 3c에 개략적으로 제시된 바와 같이 논의된다.

[0089] 재구성된 블록으로부터의 각각의 픽셀에 대해 픽셀 처리는 다음의 단계들을 포함한다:

[0090] 스캔 패턴에 따라 현재의 것을 포함하는 처리 픽셀 주위의 4개의 이웃 픽셀에 대해 스캐닝한다.

[0091] 판독 픽셀들의 4 포인트 Hadamard 변환.

[0092] 공식(1) 및 (2)에 기초한 스펙트럼 필터링.

[0093] DC 값에 대응하는 제1 스펙트럼 성분은 필터링 없이 바이패싱된다.

[0094] 필터링된 스펙트럼의 역 4 포인트 Hadamard 변환.

[0095] 필터링 단계 후에 필터링된 픽셀들은 누적 버퍼 내로 그것의 원래 위치들에 배치된다.

[0096] 픽셀들의 필터링을 완성한 후에 누적된 값들은 각각의 픽셀 필터링에 사용되는 처리 그룹들의 수에 의해 정규화된다. 블록 주위의 하나의 샘플의 패딩의 사용으로 인해 처리 그룹들의 수는 블록 내의 각각의 픽셀에 대해 4와 동등하고, 2개의 비트에 대해 우측 시프트하는 것에 의해 정규화가 수행된다.

[0097] 최대 병렬성이 요구되는 경우에 블록 내의 모든 픽셀이 독립적으로 처리될 수 있다는 것을 알 수 있다.

[0098] 이 실시예들에서, 임계값 THR은 미리 정의된 값, 예를 들어, 128로 설정되고, 이는 간단한 구현의 경우에 각각의 QP마다 7-비트 값들의 128(1 << 7)개의 항목을 저장하는 것을 요구한다.

[0099] LUT의 크기는 온-칩 메모리 요건의 양 및 필터의 하드웨어 구현의 비용에 영향을 미친다. 온-칩 스토리지의 양을 감소시키기 위해 LUT는 8의 일정한 간격으로 QP 20으로부터 시작하는 제한된 QP들의 세트에 대해서만 계산된다. 전체 5개의 미리 정의된 LUT(5개의 QP 그룹에 대한)가 저장된다. 현재 블록의 필터링에 대해 CU의 QP는 테이블로부터 가장 가까운 것으로 라운딩된다.

[0100] LUT 크기의 추가 감소를 위해 LUT 생성 동안 N개의 최저 비트가 드로핑(또는 무시)된다. 이는 희소 테이블(sparse table) 표현을 갖는 것을 허용한다.

[0101] 예시적인 구현 A에 대해, N은 2와 동등하고, 이는 7 - 2 = 5 비트의 테이블 깊이(7-비트 값들의 32개 항목)로 이어진다;

[0102] 예시적인 구현 B에 대해, N은 3과 동등하고, 이는 7 - 3 = 4 비트의 테이블 깊이(7-비트 값들의 16개 항목)로 이어진다;

[0103] 따라서, 전체 LUT 스토리지에 필요한 총 메모리 크기:

[0104] 예시적인 구현 A에 대해: 5x32x7 비트 =1120 비트 =140 바이트;

[0105] 예시적인 구현 B에 대해: 5x16x7 비트 = 560 비트 = 70 바이트;

[0106] 예시적인 구현 B는 전체 LUT를 하나의 16 바이트 SSE 레지스터에 저장하는 능력으로 인해 소프트웨어 구현에서 병렬 액세스를 가능하게 하기 위해 16 바이트의 LUT 크기를 목표로 하고 있고 따라서 이러한 구성이 제안된다.

[0107] Hadamard 변환이 사용되고, 필터링된 픽셀이 미리 정의된 스캔 템플릿에 따라 그것의 원래 위치에 배치되면, 다음의 의사 코드는 방법(300)의 필터링 프로세스를 기술한다:

```
// reconstructed/predicted pixels scan
const int x0 = pIn[p0];
const int x1 = pIn[p1];
const int x2 = pIn[p2];
const int x3 = pIn[p3];    // p0-p3 define scan pattern

// 1D forward Hadamard transform
const int y0 = x0 + x2;
const int y1 = x1 + x3;
const int y2 = x0 - x2;
const int y3 = x1 - x3;
const int t0 = y0 + y1;
const int t1 = y0 - y1;
const int t2 = y2 + y3;
const int t3 = y2 - y3;
// frequency domain filtering
const int z0 = pTbl[t0];
const int z1 = pTbl[t1];
const int z2 = pTbl[t2];
const int z3 = pTbl[t3];

// backward Hadamard transform
const int iy0 = z0 + z2;
const int iy1 = z1 + z3;
const int iy2 = z0 - z2;
const int iy3 = z1 - z3;

    // output filtered pixel
pOut[p0_out] = iy0 + iy1;
```

[0108]

[0109] Hadamard 변환이 사용되고, 필터링된 샘플들의 선형 버퍼로부터의 하나보다 많은 필터링된 픽셀이 누적 버퍼에 추가되면, 다음의 의사 코드는 이 시나리오의 필터링 프로세스를 기술한다:

```
// reconstructed/predicted pixels scan
const int x0 = pIn[p0];
const int x1 = pIn[p1];
const int x2 = pIn[p2];
const int x3 = pIn[p3];    // p0-p3 define scan pattern

// 1D forward Hadamard transform
const int y0 = x0 + x2;
const int y1 = x1 + x3;
const int y2 = x0 - x2;
const int y3 = x1 - x3;
const int t0 = y0 + y1;
const int t1 = y0 - y1;
const int t2 = y2 + y3;
const int t3 = y2 - y3;
// frequency domain filtering
const int z0 = pTbl[t0];
const int z1 = pTbl[t1];
const int z2 = pTbl[t2];
const int z3 = pTbl[t3];

// backward Hadamard transform
const int iy0 = z0 + z2;
const int iy1 = z1 + z3;
const int iy2 = z0 - z2;
const int iy3 = z1 - z3;
```

[0110]

[0111] 대안적인 실시예로서 누적 버퍼는 필터링되지 않은 픽셀 값들을 블록에서 추가될 픽셀 값들의 최대 수와 곱하는 것에 의해 초기화되어야 한다. 블록에서 추가될 픽셀 값들의 최대 수는 스캔 템플릿에 기초하여 정의된다. 실제로 스캔 템플릿은 각각의 위치에 대해 추가된 픽셀 값들의 수를 정의한다. 그에 기초하여, 블록 내의 모든 위치들로부터의 최대 수가 선택되어 누적 버퍼 초기화 동안 사용될 수 있다. 그 후, 각각의 누적 단계 동안, 필터링되지 않은 픽셀 값이 대응하는 필터링된 값으로부터 감산되고 누적 버퍼에 추가된다:

```
// filtered pixels accumulation
pOut[p0] += iy0 + iy1    // p0-p3 define scan pattern
pOut[p1] += iy0 - iy1
pOut[p2] += iy2 + iy3
pOut[p3] += iy2 - iy3
```

[0112]

[0113] 누적 버퍼에 배치하기 전에 누적된 픽셀 값들의 비트 깊이를 감소시키기 위해 역방향 변환의 결과가 변환의 크기(m)에 대해 정규화될 수 있다:

```
pOut[p0] += ((iy0 + iy1) >> HTDF_BIT_RND4);
pOut[p1] += ((iy0 - iy1) >> HTDF_BIT_RND4);
pOut[p2] += ((iy2 + iy3) >> HTDF_BIT_RND4);
pOut[p3] += ((iy2 - iy3) >> HTDF_BIT_RND4);
```

[0114]

[0115] 여기서 HTDF_BIT_RND4는 4의 변환 크기에 대해 2와 동등하다.

[0116] 이 실시예는 시스템이 현재 위치에 추가된 픽셀들의 수를 저장하는 것을 회피할 수 있게 하고, 추가된 픽셀 값들의 최대 수가 예를 들어 2, 4, 8 등의 거듭제곱이면 대응하여 마지막 정규화 단계 및 누적 버퍼 초기화 단계에서 시프트 연산에 의해 대체 나눗셈 및 곱셈을 가능하게 한다.

[0117] 정규화 스테이지의 정밀도를 유지하기 위해, 그것은 다음의 방식으로 수행될 수 있다:

```
// normalization
pFiltered[p0] = CLIP3(0, (1 << BIT_DEPTH) - 1, (pOut[p0] + HTDF_CNT_SCALE_RND)
>> HTDF_CNT_SCALE);
```

[0118]

[0119] 여기서 HTDF_CNT_SCALE은 누적 버퍼 내에 배치된 픽셀들의 양의 Log2이고, 예를 들어, 4의 양에 대해 HTDF_CNT_SCALE은 2와 동등하고, HTDF_CNT_SCALE_RND는 (1 << (HTDF_CNT_SCALE - 1))과 동등하다. CLIP3은 필터링된 샘플이 최소 및 최대 샘플 값 간의 허용 범위 내에 있는 것을 보장하는 클리핑 함수이다.

[0120] 필터링된 블록의 평균 휘도의 변화를 회피하기 위해 위에 언급된 바와 같이 제1 스펙트럼 성분(DC 값에 대응)의 필터링을 스킵하는 것이 유리할 수 있다. 이는 추가로 필터 구현을 단순화하는 것을 허용한다. 이 경우 필터링 단계는 다음과 같다:

```
// frequency domain filtering
const int z0 = t0;
const int z1 = pTbl[t1];
const int z2 = pTbl[t2];
const int z3 = pTbl[t3];
```

[0121]

[0122] 재구성된 블록 또는 예측된 블록 내부의 각각의 픽셀에 대해, 단계 302 및 단계 310에 대해 재구성된 블록 또는 예측된 블록 내부의 필터링 픽셀의 위치에 기초하여 스캔 템플릿이 선택된다. 스캔 템플릿은 모든 픽셀들이 재구성된 CU 또는 예측된 CU 내부에 있고 처리 픽셀에 가깝게 배치되는 것을 보장하도록 선택된다. 템플릿 내부에서는 임의의 스캔 순서가 이용될 수 있다. 예를 들어, 미리 정의된 스캔 템플릿은 재구성된 블록 또는 예측된 블록 내부의 현재 픽셀의 위치에 대한 공간적 또는 래스터 오프셋들의 세트로서 정의되고, 여기서, 이웃 픽셀들을 지시하는 오프셋들은 재구성된 블록 또는 예측된 블록 내부에 있다. 아래는 스캔 템플릿의 예이다:

[0123] (0,0), (0,1), (1,0), (1,1)

[0124] 도 4a는 정사각형 블록(예를 들어, 정사각형 CU 예측된 블록, 또는 정사각형 CU 재구성된 블록) 내부의 상이한 픽셀 위치에 대한 템플릿들의 예를 예시한다. 이 도면에 따르면 경계 픽셀들은 4 포인트 변환에 기초하여 필터링될 수 있고 중앙 픽셀들은 8 포인트 변환에 기초하여 필터링될 수 있다.

[0125] 직사각형 재구성된 블록들 또는 예측된 블록들에 대해, 한 변의 크기는 다른 변의 더 큰 크기이고 스캔은 긴 변을 따라 수행되어야 한다. 예를 들어 수평 직사각형 블록에 대해 다음의 스캔 순서가 사용될 수 있고,

[0126] (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (0,3), (0,4),

[0127] 여기서 각각의 쌍 (y,x)에서 x는 수평 오프셋이고 y는 필터링된 재구성된 블록 또는 예측된 블록 내부의 필터링 픽셀의 위치에 대한 수직 오프셋이다.

[0128] 제안된 필터는 다음의 조건들에 따라 선택적으로 적용될 수 있다:

[0129] ● 0이 아닌 잔차 신호를 갖는 재구성된 블록들 또는 예측된 블록들에 대해;

[0130] ● 블록 크기에 따라, 예를 들어, 작은 재구성된 블록 또는 예측된 블록에 대해(최소 크기가 임계값보다 작음);

[0131] ● 재구성된 블록 또는 예측된 블록의 종횡비에 따라;

[0132] ● 재구성된 블록 또는 예측된 블록의 예측 모드(인트라 또는 인터)에 따라, 예를 들어, 인터 예측된 블록들에만 필터를 적용하는 것에 의해; 또는

[0133] ● 위에 설명된 조건들의 임의의 조합에 대해.

[0134] 예를 들어, 작은 블록들의 처리를 회피하기 위해, 블록 크기가 4x4 픽셀들보다 작거나 그와 동등하다면 그 필터가 바이패싱될 수 있다(적용되지 않음). 이는 통상적으로 가장 작은 블록 처리에 대응하는 최악의 경우의 복잡도를 감소시킨다.

[0135] 다른 예로서 필터는 0이 아닌 잔차 신호를 갖는 블록들에만 적용된다. 이는 양자화 또는 잔차들이 사용되었다면 유익한데, 그 이유는 필터가 양자화 오류를 개선하는 것을 목표로 하기 때문이다. 블록이 잔차를 갖지 않는다면 이는 예측이 양호하고 추가의 필터링이 요구되지 않는다는 지시일 수 있다.

[0136] 다른 예로서, 인트라 예측이 일반적으로 인터보다 나쁘기 때문에, 필터는 0이 아닌 잔차의 존재와 독립적으로 인트라 예측된 블록들에 적용될 수 있고 블록이 0이 아닌 잔차 신호를 갖는 경우에만 인터 예측된 블록들에 적용될 수 있다.

[0137] 필터 파라미터 시그마 및 스캔 패턴은 위에 열거된 조건들에 따라 달라질 수 있다.

[0138] 도 4b는 예시적인 스캔 템플릿 (0,0), (0,1), (1,0), (1,1)에 대한 현재 블록 내부의 하나의 픽셀을 고려하는 등가 필터 형상을 예시한다. 현재 픽셀의 필터링을 위해 3x3 픽셀들의 정사각형 영역이 사용된다(현재 픽셀은 3x3 정사각형의 중심에서 어두운 회색 색으로 표시되어 있다). 필터링된 픽셀은 4개의 2x2 그룹으로부터의 변환 도메인 필터링된 샘플들을 조합하는 것에 의해 획득된다. 현재 픽셀이 블록 경계(예를 들어, 상단 경계)에 위치한다면 좌측 상단 및 우측 상단 2x2 그룹들은 이용가능하지 않고 2개의 2x2 그룹(좌측 하단 및 우측 하단)만이 필터링을 위해 사용될 수 있다는 것을 이해할 수 있다. 더욱이 현재 픽셀이 블록 모서리(예를 들어, 좌측 상단 모서리)에 위치한다면 하나의 2x2 그룹(우측 하단)만이 필터링을 위해 사용될 수 있다.

[0139] 경계 및 모서리 픽셀들에 대해 더 많은 4개의 2x2 그룹을 사용하여 필터링의 품질을 증가시키기 위해, 현재 블록은 추가적인 샘플들에 의해 패딩될 수 있다. 도 4c는 좌측 및 상측의 패딩의 예를 제공한다. 패딩 샘플들은 이미 재구성된 블록들로부터 취해질 수 있다.

[0140] 블록 내의 모든 픽셀들에 대한 필터링 프로세스의 추가 통합을 위해(4개의 2x2 그룹이 현재 블록 내의 모든 픽셀들의 필터링에 사용됨), 좌측 상단 패딩에 더하여, 현재 블록이 또한 도 4d에 예시된 바와 같이 우측 하단 패딩에 의해 확장될 수 있다. 필터링의 통합은 모서리 픽셀들 및/또는 경계 픽셀들에 대한 특수한 처리 경우들을 제외하는 것에 의해 구현을 단순화하는 것으로 인해 유익하다.

[0141] 패딩 샘플들은 바람직하게는 이미 재구성된 블록들로부터의 조정된 이웃 샘플들로부터 취해진다. 최신 기술 비디오 코덱들에서, 그러한 이미 재구성된 블록들은 블록 재구성 순서에 따라 현재 블록으로부터 좌측 또는 상측에 또는 우측 또는 하측에 위치할 수 있다. 조정 샘플들로부터의 더 많은 정보를 사용하여, 이는 필터링 품질을 개선하고 블록들 간의 전이를 더 원활하게 한다.

[0142] 조정된 블록들 또는 이전에 재구성된 블록들로부터 재구성된 샘플들을 검색하는 것은 하드웨어 또는 소프트웨어 구현을 위한 추가적인 메모리 로딩을 요구할 수 있다. 추가적인 메모리를 최소화하거나 제외하기 위해, 현재 블록 경계들로 조정된 이웃 블록들로부터의 1개, 2개 또는 그 이상의 행들 및 열들로부터 흔히 취해지는 현재 블록의 인트라 예측을 위해 의도된 샘플들을 사용하는 것이 유익하다. 이들 샘플은 통상적으로 인트라 예측을 위한 용이한 액세스를 위해 고속 메모리("라인" 버퍼로도 알려짐)에 저장되고, 인트라 예측의 참조 샘플들로 불린다.

[0143] 일부 구현에서, 인트라 예측을 수행하기 전에, 참조 샘플(인트라 참조 샘플)이 예측 전에, 예를 들어, 평활화, 샤프닝(sharpening), 디링잉(de-ringing) 또는 쌍방 필터링에 의해 전처리된다는 점에 추가로 유의해야 한다. 이 경우 현재 블록의 패딩을 위해 전처리된 샘플들을 사용하는 것이 유익할 수 있다.

[0144] 조정된 블록 재구성의 순서로 인해, 패딩된 영역 내의 일부 샘플들이 이용가능하지 않다면, 도 4d에 예시된 바와 같이 필요한 샘플들이 현재 블록으로부터 패딩되어 경계 픽셀들을 패딩된 영역으로 확장할 수 있다.

[0145] 도 5a는 실시예에 따른 대응하는 루프 내 필터링 방법(500)의 단계들을 예시하는 흐름도를 도시한다. 재구성된 블록은 복수의 픽셀을 포함한다. 방법(500)은 다음의 단계들을 포함한다: 미리 정의된 스캔 템플릿에 따라 현재 픽셀 및 그것의 이웃 픽셀들을 선형 버퍼에 로딩하는 단계(502); 상기 선형 버퍼 내의 픽셀들에 대해 1D 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득하는 단계(504); 각각의 스펙트럼 성분을 이득 계수와 곱하는 것에 의해 필터링된 스펙트럼 성분들을 획득하는 단계(506) - 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존함 -; 필터링된 스펙트럼 성분들에 대해 역 1D 변환을 수행하는 것에 의해 필터링된 픽셀들을 획득하는 단계(508); 및 이전 처리 단계들에서 추정된 상기 필터링된 픽셀들에 기초하여 필터링된 재구성된 블록을 생성하는 단계(510). 방법(500)은 도 1에 도시된 바와 같은 인코딩 장치 및 도 2에 도시된 바와 같은 디코딩 장치에 의해 수행될 수 있다. 도 3a의 상세한 정보(300) 또는 도 3b의 정보(300')는 또한 도 5a에 도시된 바와 같은 방법(500)에 적용된다.

[0146] 도 5a와 유사하게, 도 5b는 다른 실시예에 따른 대응하는 루프 내 필터링 방법(500')의 단계들을 예시하는 흐름도를 도시한다. 이 예로서, 상기 블록(또는 프레임)은 예측된 블록일 수 있고, 상기 필터링된 블록은 필터링된 예측된 블록이다. 도 5b의 상세한 설명은 도 5a와 유사하다.

[0147] 하드웨어 구현이 논의된다.

[0148] Hadamard 변환 도메인 필터는 블록 재구성 직후에 배치되고 특히 인트라 예측의 참조 샘플로서 후속 블록 재구성에 수반될 수 있는 샘플들을 처리한다. 따라서, 필터에 의해 도입된 레이턴시는 전체 재구성 파이프라인이 많이 영향을 받지 않도록 보장하기 위해 최소화될 필요가 있다.

[0149] Hadamard 변환은 하드웨어 구현에 대해 비교적 간단한 것으로 간주된다. 그것의 구현을 위해 곱셈들이 아니라 가산들만이 요구된다. 아래의 의사 코드 1로부터 알 수 있는 바와 같이, 순방향 및 역방향 변환은 병렬로 또는 2개의 순차적인 가산 연산으로 중간 결과들을 재사용하는 것에 의해 행해질 수 있는 4개의 가산을 포함한다.

의사 코드 1

```
for (int r = 0; r < height - 1; ++r)
{
  Pel *pIn = &block[r*strideBlk];
  Pel *pAcc = &accBlock[r*strideAcc];

  for (int c = 0; c < width - 1; ++c, pIn++, pOut++)
  {
    const int x0 = pIn[p0];
    const int x1 = pIn[p1];
    const int x2 = pIn[p2];
    const int x3 = pIn[p3];

    // forward transform
    const int y0 = x0 + x2;
    const int y1 = x1 + x3;
    const int y2 = x0 - x2;
    const int y3 = x1 - x3;

    const int t0 = y0 + y1;
    const int t1 = y0 - y1;
    const int t2 = y2 + y3;
    const int t3 = y2 - y3;

    // filtering
    const int z0 = t0;  // skip DC
    const int z1 = RdTbl(t1, tbl, thr);
    const int z2 = RdTbl(t2, tbl, thr);
    const int z3 = RdTbl(t3, tbl, thr);

    // backward transform
    const int iy0 = z0 + z2;
    const int iy1 = z1 + z3;
    const int iy2 = z0 - z2;
    const int iy3 = z1 - z3;

    pAcc[p0_out] += ((iy0 + iy1) >> HTDF_BIT_RND4);
    pAcc[p1_out] += ((iy0 - iy1) >> HTDF_BIT_RND4);
    pAcc[p2_out] += ((iy2 + iy3) >> HTDF_BIT_RND4);
    pAcc[p3_out] += ((iy2 - iy3) >> HTDF_BIT_RND4);

    // normalization
    pIn[p0] = ClipPel((pOut[p0_out] + HTDF_CNT_SCALE_RND) >> HTDF_CNT_SCALE, clpRng);
  }
}
```

[0150]

[0151] 순방향 및 역방향 Hadamard 변환은 조합 로직을 사용하여 하드웨어로 구현될 수 있다. LUT에 대한 고속의 그리고 병렬의 액세스를 위해 더 많은 주의가 필요하다.

[0152] SRAM 기반 LUT가 논의된다.

[0153] 이 예시적인 구현에서, LUT는 온-칩 단일 포트 정적 RAM(도 6)에 저장된다.

[0154] 일단 이전 처리 단계로부터의 데이터가 클록의 상승 에지에 의해 버퍼에서 이용가능해지면, 그것은 순방향 Hadamard 변환을 구현하는 조합 로직(2개의 후속하는 가산을 포함함)에 의해 액세스된다. 조합 로직을 완성한 후에 어드레스가 각각의 LUT에 대해 이용가능하다. 인버터 및 클록의 하강 에지를 사용하여 데이터가 SRAM으로부터 액세스된다. 역방향 Hadamard 변환 및 정규화를 구현하는 제2 조합 로직이 LUT로부터의 데이터가 이용가능한 직후에 시작된다. 출력된 필터링된 샘플들은 현재 클록 사이클의 끝에서 이용가능해졌고 클록의 다음 상승 에지에서 다음 알고리즘에 의한 처리를 위해 준비된다.

[0155] 플립-플롭 기반 LUT가 논의된다.

[0156] 필터링 프로세스를 위한 하나의 테이블이 16개의 항목에 의해 제한된다는 것을 고려하면, 플립-플롭들에 기초하여 LUT를 구현하는 것이 더 효율적인 것처럼 보인다. 그러한 설계는 병렬 처리를 위한 수 개의 LUT 및 데이터 액세스를 위한 클록 에지를 필요로 하지 않는다. 하나의 2x2 그룹을 처리하기 위한 예시적인 설계를 예시하는 도 7에 묘사된 바와 같은 멀티플렉서에 의해 병렬 액세스가 제공된다. 제안된 설계에서는 필터링 프로세스 동안 병렬 액세스를 제공하기 위해 7 비트의 16개의 플립-플롭이 요구된다. QP 특정 LUT는 일단 현재 CU에 대한 QP가 이용가능해지면 플립-플롭들에 로딩될 수 있다.

[0157] 4개의 2x2 그룹 필터링의 결과들을 동일한 공간적 그룹 최종 필터 출력의 결과들의 재사용과 조합하는 것이 도

8에 묘사된 바와 같이 생성된다.

[0158] 주어진 위의 분석은 제안된 필터가 SRAM 또는 플립-플롭 기반 LUT 구현을 사용하여 하나의 클록 내에서 하드웨어로 구현될 수 있다는 결론에 이르는 것을 가능하게 한다.

[0159] 복잡도 분석이 논의된다.

[0160] 앵커(들)에 대한 비트레이트/PSNR에 대한 영향을 측정한다.

[0161] 복잡도 분석(예를 들어, 인코딩 및 디코딩 시간 측정, 아래의 표를 채움으로써 복잡도 분석).

테스트 #	필터 형상	계산 복잡도 샘플당 (곱셈/가산/시프트/체크*의 수)	곱셈의 정밀도	병렬 친화적(각각의 샘플은 다른 샘플들과 독립적으로 필터링될 수 있음)	클럭 사이클의 수	메모리 요건	필터 계수들을 도출하는 방법	필터들을 적용하기 위한 최소 CU 크기
14.3 a	3x3	라운드/5/6 에 대한 0/20+4 I-비트 가산	해당 없음	예	< 1 클럭	140 바이트 (qp 그룹당 32 개의 7-비트 값)	LUT 에서 미리 계산됨	4x8, 8x4
14.3 b	3x3	라운드 /5/6 에 대한 0/20+4 I-비트 가산	해당 없음	예	< 1 클럭	70 바이트 (qp 그룹당 16 개의 7-비트 값)	LUT 에서 미리 계산됨	4x8, 8x4

* max/min/abs 는 체크들로서 카운트됨

[0162] 표 1. CE14-3 복잡도 분석 요약

[0163] 실험 결과들이 논의된다.

[0164] 객관적인 결과들이 논의된다.

[0165] 객관적인 성능이 다음의 표들에 제시된다:

	모든 인트라 메인 10				
	VTM 구성을 갖는 BMS-2.0.1 에 걸쳐				
	Y	U	V	EncT	DecT
클래스 A1	-0.19%	0.29%	0.24%	110%	115%
클래스 A2	-0.35%	0.31%	0.32%	109%	112%
클래스 B	-0.42%	0.29%	0.32%	109%	113%
클래스 C	-0.83%	0.23%	0.42%	110%	110%
클래스 E	-0.54%	0.28%	0.19%	109%	114%
전체	-0.48%	0.28%	0.31%	109%	112%
클래스 D	-0.61%	0.47%	0.01%	109%	108%
클래스 F(옵션)	-0.82%	0.07%	0.04%	108%	107%

	랜덤 액세스 메인 10				
	VTM 구성을 갖는 BMS-2.0.1 에 걸쳐				
	Y	U	V	EncT	DecT
클래스 A1	-0.43%	-0.23%	-0.25%	105%	104%
클래스 A2	-0.92%	-0.57%	-0.43%	105%	103%
클래스 B	-0.69%	-0.02%	-0.27%	106%	104%
클래스 C	-0.75%	0.10%	0.14%	105%	104%
클래스 E					
전체	-0.70%	-0.14%	-0.19%	105%	104%
클래스 D	-0.67%	-0.14%	0.53%	104%	103%
클래스 F(옵션)	-0.77%	0.10%	0.13%	104%	102%

	저지연 B 메인 10				
	VTM 구성을 갖는 BMS-2.0.1 에 걸쳐				
	Y	U	V	EncT	DecT
클래스 A1					
클래스 A2					
클래스 B	-0.58%	0.52%	0.54%	104%	104%
클래스 C	-0.74%	0.33%	0.52%	104%	104%
클래스 E	-0.75%	0.31%	1.09%	102%	101%
전체	-0.68%	0.40%	0.67%	104%	103%
클래스 D	-0.90%	0.75%	0.28%	104%	104%
클래스 F(옵션)	-0.73%	0.20%	-0.09%	103%	102%

[0166] 표 2. 테스트 14-3a 의 코딩 성능

	모든 인트라 메인 10				
	VTM 구성을 갖는 BMS-2.0.1 에 걸쳐				
	Y	U	V	EncT	DecT
클래스 A1	-0.17%	0.28%	0.28%	110%	115%
클래스 A2	-0.33%	0.27%	0.29%	109%	112%
클래스 B	-0.40%	0.31%	0.39%	109%	113%
클래스 C	-0.82%	0.20%	0.32%	109%	110%
클래스 E	-0.54%	0.34%	0.25%	109%	113%
전체	-0.47%	0.28%	0.32%	109%	112%
클래스 D	-0.65%	0.43%	0.22%	109%	108%
클래스 F(옵션)	-0.81%	0.01%	0.12%	108%	107%

	랜덤 액세스 메인 10				
	VTM 구성을 갖는 BMS-2.0.1 에 걸쳐				
	Y	U	V	EncT	DecT
클래스 A1	-0.41%	-0.36%	-0.24%	105%	105%
클래스 A2	-0.92%	-0.49%	-0.37%	105%	103%
클래스 B	-0.69%	-0.12%	0.12%	106%	104%
클래스 C	-0.75%	0.14%	0.10%	105%	104%
클래스 E					
전체	-0.70%	-0.17%	-0.06%	105%	104%
클래스 D	-0.69%	-0.30%	0.49%	104%	103%
클래스 F(옵션)	-0.82%	0.11%	0.11%	104%	102%

	저지연 B 메인 10				
	VTM 구성을 갖는 BMS-2.0.1 에 걸쳐				
	Y	U	V	EncT	DecT
클래스 A1					
클래스 A2					
클래스 B	-0.57%	0.49%	0.69%	104%	104%
클래스 C	-0.76%	0.31%	0.42%	104%	106%
클래스 E	-0.67%	-0.66%	-0.48%	102%	102%
전체	-0.66%	0.14%	0.31%	103%	104%
클래스 D	-0.88%	0.92%	0.31%	103%	105%
클래스 F(옵션)	-0.74%	0.38%	-0.15%	103%	102%

표 3. 테스트 14-3b 의 코딩 성능

[0167]

[0168] QP당 16 바이트를 갖는 제안된 LUT 70바이트, 1 클록 HW 구현을 허용한다. 기타 등등. 다음 버전의 VTM에 Hadamard 변환 도메인 필터를 채용하는 것이 제안된다.

[0169] 다음의 참조 문헌들은 그 전체가 재현되는 것처럼 인용에 의해 본 명세서에 포함된다.

[0170] JVET(Joint Video Experts Team) 문서 JVET-K0068.

[0171] 아래는 LUT를 최적화하기 위한 예들을 예시한다.

[0172] 일 예 1로서, 양자화 파라미터(QP)들의 세트가 룩업 테이블(LUT)을 생성하도록 선택되고, 여기서 QP들의 세트는 (i) 인덱스에 대응하는 제1 QP 및 (i+1) 인덱스에 대응하는 제2 QP를 포함하고, 제1 QP 및 제2 QP는 일정한 간격을 갖는다. 예를 들어, 그 간격은 8, 10 또는 16과 동등할 수 있다.

[0173] 예를 들어, 일정한 간격이 8인 것을 예로 들어, qp = {20, 28, 36, 44, 52}에 대한 LUT를 가짐으로써 희소 테이블. 제1 gp 20과 제2 gp 28 간의 간격은 8이다. 유사하게, 제2 gp 28과 제3 gp 36 간의 간격은 8이다. 필터링 동안 가장 가까운 QP를 갖는 테이블이 선택된다.

[0174] 다른 예로서, 일정한 간격이 8인 것을 예로 들어, qp = {18, 26, 34, 42, 50}에 대한 LUT를 가짐으로써 희소 테이블. 제1 gp 18과 제2 gp 26 간의 간격은 8이다. 유사하게, 제2 gp 26과 제3 gp 34 간의 간격은 8이다. 필터링 동안 가장 가까운 QP를 갖는 테이블이 선택된다.

[0175] LUT 크기: 5x128 = 640 바이트

[0176] 아래는 어느 QP들이 룩업 테이블들(LUT)을 생성하도록 선택되는지를 반영하는 의사 코드 2이다.

의사 코드 2

```
#define HTDF_MIN_QP                                    20
#define HTDF_MAX_QP                                    (MAX_QP+1)
#define HTDF_QP_ROUND                                  8
 for (int qp_in = HTDF_MIN_QP; qp_in <  HTDF_MAX_QP; qp_in++)
  {
    int idx = ( (qp_in - HTDF_MIN_QP) + (HTDF_QP_ROUND>>1) ) / HTDF_QP_ROUND;
    int qp = idx*HTDF_QP_ROUND + HTDF_MIN_QP;

    if (idxPrev != idx)
    {
      tblGen(HTDFTable[idx], qp);
      idxPrev = idx;
    }
  }
```

[0177]

[0178] 이 의사 코드에서, HTDF_QP_ROUND는 일정한 간격을 표현한다. 2의 거듭제곱으로서 간격을 갖는 것이 인덱스 계산을 위한 나눗셈 연산을 시프트로서 구현하는 것을 가능하게 하여 유리하다. 일정한 간격의 상이한 값들, 예를 들어, 2, 4, 10, 15 또는 16 등이 선택될 수 있다는 점에 유의해야 한다. 더욱이, 대안적인 실시예로서 간격은 임의적일 수 있고 임의의 QP들의 세트에 대해 어느 LUT가 계산된다.

[0179] 필터링 프로세스 동안 주어진 QP에 대한 LUT에 대응하는 인덱스는 다음과 같이 계산된다:

[0180] int idx = ((qp - HTDF_MIN_QP) + (HTDF_QP_ROUND >> 1)) / HTDF_QP_ROUND;

[0181] 또는 대안적으로 더 적은 정밀도로:

[0182] int idx = (qp - HTDF_MIN_QP) / HTDF_QP_ROUND;

[0183] 일정한 간격이, 예를 들어, 2의 거듭제곱이면, LUT의 인덱스가 유리하게도 나눗셈 대신에 시프트 연산을 사용하여 계산될 수 있다:

[0184] int idx = (qp - HTDF_MIN_QP) >> HTDF_QP_ROUND_LOG2 = (qp - HTDF_MIN_QP) >> 3;

[0185] 다른 예 2로서, 디코더 또는 인코더는 테이블 값들로부터 N개의 비트를 드로핑하고, N은 정수이다. 이는 주어진 범위로부터의 선택된 값들만을 저장하는 희소 LUT 표현을 가능하게 한다. 예를 들어, N은 3이다. 최대 테이블 값은 127(7 비트)이고, 3 비트를 드로핑하면, 결과는 4 비트이고, 즉, 7 비트 값의 16개 항목 - 대략 16 바이트이다.

[0186] 아래는 주어진 qp 값에 따라 LUT가 어떻게 생성되는지를 기술하는 의사 코드 3이다.

의사 코드 3

```
#define HTDF_TBL_SH      3
#define HTDF_TBL_RND     ((1<<HTDF_TBL_SH)>>1)


void HTDFilter::tblGen(HTDF_TBL_TYPE *tbl, int qp)
{
  float sigma = (2.64f)*powf(2.0f, (0.1269f)*(qp - 11));
  int sigma_l = (int)(sigma * HTDF_SIGMA_SCALE );

  sigma_l *= sigma_l;
  sigma_l *= 4;

  for (int k_in = 0; k_in < HTDF_SHORT_TBL_THR; k_in++)
  {
    const int Idx = (k_in + HTDF_TBL_RND)>>HTDF_TBL_SH;
    const int k = Idx << HTDF_TBL_SH;

    unsigned int num = (k*k*k) << 10;
    int den = ((k*k) << 10) + sigma_l;
    int value_long = num / den;
    HTDF_TBL_TYPE value_int = (HTDF_TBL_TYPE)value_long;

    tbl[Idx] = value_int;
  }
}
```

[0187]

[0188] 주어진 예에서 HTDF_TBL_SH는 1, 2, 3, 4 등일 수 있는 드로핑할 비트의 수를 정의한다.

[0189] 아래는 필터링 동안 희소 LUT에 대한 액세스를 예시하는 의사 코드이다.

[0190]
```
tbl[(z + HTDF_TBL_RND)>> HTDF_TBL_SH]
```

[0191] 상기 예 1 및 예 2를 조합할 때, 도 9는 LUT를 최적화한 결과를 예로서 도시한다. 도 9에 도시된 바와 같이, 예 1 및 예 2를 조합하면: 5개 테이블 x 16개 항목 x 7 비트 = 560 비트= 70 바이트이다.

[0192] LUT 항목들의 양은 HTDF_SHORT_TBL_THR([0076] 단락으로부터의 임계값) 및 HTDF_TBL_SH 드로핑된 비트의 수에 의해 정의된다는 점에 유의해야 한다. 128(즉, 1 << 7)과 동등한 임계값 및 드로핑된 비트 3을 고려하면 테이블 항목들의 수가 1 << (7 - 3) = 1 << 4 = 16과 동등하다. 위에 설명된 바와 같이 LUT 임계값은 바람직하게는 [0077] 단락으로부터의 방정식의 결과를 1에 가깝게 만들도록 선택되고 거기에 또한 설명된 바와 같이 임계값은 QP에 따라 상이할 수 있다. 따라서 더 높은 QP 값들에 대한 LUT 생성을 위해 임계값을 128(1 << 7)로부터 예를 들어 256(1 << 8)로 증가시키는 것이 유익할 수 있다. 그 경우 LUT 항목들(예를 들어, 3 비트가 드로핑된)의 동일한 정밀도를 유지하는 것은 32개의 항목(32 = 1 << (8 - 3) = 1 << 5)을 필요로 할 것이다. 대안적으로 더 낮은 QP들에 대해서와 같이 더 높은 QP들에 대해 동일한 LUT 크기를 유지하기 위해 정밀도는 테이블의 16개 항목 16 = 1 << (8 - 4) = 1 << 4를 유지하기 위해 4로 더 감소될 수 있다.

[0193] 일부 구현에서 LUT 크기를 제한되게 유지하는 것과 [0077} 단락으로부터의 방정식을 유지하기 위해 THR을 갖는 것은 모순될 수 있다. 실제로 높은 QP 값들(이는 높은 σ 값으로 이어짐)에서, 예를 들어, LUT 크기를 16개의 항목에 의해 제한되게 유지하는 것은 도 10에 묘사된 바와 같이 값 120 주위에 필터 전달 함수(LUT에 의해 표현됨)에서 바람직하지 않은 갭으로 이어질 수 있다(3개의 최하위 비트를 드로핑하는 것에 의해 LUT 서브샘플링의 방법을 또한 포함함).

[0194] 도 11은 인덱스 15에서의 마지막 LUT 항목과 인덱스 16에서의 필터 전달 함수에서의 다음 포인트 간의 갭을 예시하는 LUT 서브샘플링 효과를 고려하지 않고 테이블 항목들이 하나하나 플롯되어 있는 동일한 테이블을 표현한다.

[0195] 도 12는 마지막 LUT 항목 + 1에 대응하는 인수에서 THR 값을 통해 들어오는 보조 함수를 사용하여, 예를 들어, 1만큼 증가된 마지막 LUT 항목의 인수(15임)에 대응하는 인수 16 또는 다시 말해서 LUT 크기(16임)에서 포인트 THR =128을 통해 들어오는 그리고 어떤 포인트에서(예를 들어, 주어진 예에서와 같이 11의 값에서) x-축과 교차하는, 직선 방정식(녹색으로 지시됨)을 사용하여 갭이 제거될 수 있는 방법을 예시한다. 동일한 원리, 예를 들어, 지수, 로그, 포물선 등 또는 이들의 조합에 기초하여 다른 유형들의 보조 함수들이 사용될 수 있다는 점에 유의해야 한다.

[0196] 도 13은 LUT를 생성하는 동안 2개의 값으로부터 최대값을 취하는 것에 의해 갭을 제거하는 예를 예시하고, 여기서 제1 값은 위에 설명된 바와 같은 LUT 항목 $LUT(R_i,\sigma)=\frac{R_i^3}{R_i^2+m*\sigma^2}$ 이고; 제2 값은 동일한 인수 i에서의 AuxiliaryFunction(이 예에서는 직선)의 값:

[0197] $$LUT(R_i,\sigma)=\max\left(\frac{R_i^3}{R_i^2+m*\sigma^2},AuxiliaryFunc_\sigma(i)\right)$$ 이고;

[0198] 여기서, $AuxiliaryFunc_\sigma(i)$ 는 보조 함수를 표현하고, 보조 함수는 마지막 LUT + 1의 인수에 대응하는 i에서 THR과 동등한 값을 갖는다.

[0199] 도 14는 3개의 최하위 비트를 드로핑하는 것에 의해 보조 직선 방정식 및 LUT 서브샘플링을 사용하는 위에 설명된 방법을 적용한 후의 필터 전달 함수를 예시한다.

[0200] [00144] 단락에서 기술된 바와 같이, QP들의 그룹에 대해 하나의 LUT가 사용될 수 있다. 가능한 QP 범위를 커버하기 위해 미리 정의된 QP 세트가 사용되고 세트로부터의 각각의 QP에 대해 하나의 LUT가 생성된다. 도 15는 세트 내의 5개의 QP 및 대응하는 테이블들(0 내지 4)에 따른 필터 전달 함수의 예를 예시한다. 이 예에서, 테이블 4는 11의 값에서 x-축과 교차하는 직선 보조 함수로 [00156]-[00157] 단락들에 기술된 방법을 사용하여 생성되고; 표 3은 9의 값에서 x-축과 교차하는 직선 보조 함수로 동일한 방법을 사용하여 생성된다. 이 예에서 사용되는 테이블들은 다음에 따른 값들을 갖는다:

table0 = { 0, 2, 10, 19, 28, 36, 45, 53, 61, 70, 78, 86, 94, 102, 110, 118, },
table1 = { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
table2 = { 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
table3 = { 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },

[0201] table4 = { 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, },

[0202] [00150]-[00153] 단락들에서 위에 설명된 바와 같이 테이블 서브샘플링의 방법은 테이블 크기를 감소시키기 위해 테이블 값들로부터 N개의 비트를 드로핑할 수 있다. 또한 이들 단락에서 언급된 바와 같이 N은 특정 테이블 생성을 위해 사용되는 QP 및 이 테이블에 대한 선택된 THR 값에 따라 상이할 수 있다. 예를 들어, 더 낮은 QP 값들에 대해 필터링 파라미터 σ는 더 높은 QP들에 대한 것보다 상대적으로 더 낮다. 따라서, 성능을 희생시키지 않고 THR의 절대값이 감소될 수 있다. 더욱이 세트 내의 모든 QP들에 대해 테이블 크기를 동일하게 유지하고(이는 구현을 단순화하기 위해 유익함) 더 낮은 QP들에 대한 서브샘플링 레이트를 감소시키기 위해(이는 더 적은 압축 레벨 및 더 양호한 재구성된 화상 품질에 대응함), 예를 들어, 더 낮은 QP에 대해 N을 2와 동등하게 설정하고 THR을 64로 설정하는 것에 의해, 다른 QP 테이블들에 비해 드로핑된 비트의 양 N을 감소시키는 것이 유익할 수 있다. 도 16은 대응하는 테이블들(0 내지 4)에 기초한 세트 내의 5개의 QP에 대한 필터 전달 함수의 예를 예시하고, 여기서 N은 첫 번째 테이블(더 낮은 QP 범위에 대응함)에 대해 2로 설정되고 N은 다른 테이블들에 대해 3으로 설정된다. 이 예는 또한 [00159]-[00160] 단락들에 기술된 바와 같이 테이블 3 및 테이블 4 생성을 위해 *AuxiliaryFunction*을 사용하는 방법을 포함한다. 테이블 4에 대해 직선 보조 함수는 11의 값에서 x-축과 교차한다. 테이블 4에 대해 직선 보조 함수는 9의 값에서 x-축과 교차한다. 이 예에서 사용되는 테이블

들은 다음에 따른 값들을 갖는다:

table0 = { 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },
table1 = { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
table2 = { 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
table3 = { 0, 0, 0, 1, 3, 5, 9, 14, 19, 25, 32, 40, 55, 73, 91, 110, },
table4 = { 0, 0, 0, 0, 0, 1, 2, 4, 6, 8, 11, 14, 26, 51, 77, 102, },

[0203]

[0204] [00150]-[00153] 단락들에서 위에 설명된 바와 같이 테이블 서브샘플링의 방법은 테이블 크기를 감소시키기 위해 테이블 값들로부터 N개의 비트를 드로핑할 수 있다. 또한 이들 단락에서 언급된 바와 같이 N은 특정 테이블 생성을 위해 사용되는 QP 및 이 테이블에 대한 선택된 THR 값에 따라 상이할 수 있다. 예를 들어 더 높은 QP 값들에 대해 필터링 파라미터 σ는 [0077] 단락으로부터의 방정식을 1에 더 가깝게 유지하기 위해 THR 값의 증가를 필요로 할 수 있는 더 낮은 QP들에 대한 것보다 상대적으로 더 높다. 동시에 세트 내의 모든 QP들에 대해 LUT 크기를 동일하게 유지하기 위해(이는 구현의 단순화로 인해 유익함) 그리고 또한 더 높은 QP 값들에 대해 재구성된 화상이 더 많은 왜곡들을 갖고 LUT의 서브샘플링을 증가시키는 것이 강한 압축 아티팩트들의 존재 시에 LUT 서브샘플링의 주관적으로 눈에 띄지 않는 효과로 인해 수용된다는 것을 고려하여 드로핑된 최하위 비트들의 값 N은 예를 들어 세트 내의 마지막 및 두 번째 마지막 테이블에 대해 4로 증가될 수 있다. 도 17은 대응하는 테이블들(0 내지 4)에 기초한 세트 내의 5개의 QP에 대한 필터 전달 함수의 예를 예시하고, 여기서 N은 첫 번째 테이블(더 낮은 QP 범위에 대응하는 테이블 0)에 대해 2로 설정되고 N은 마지막 및 두 번째 마지막 테이블(테이블 3 및 테이블 4)에 대해 4로 설정되고 N은 다른 테이블들에 대해 3으로 설정된다. 이 예에서 TRH는 첫 번째 테이블 생성을 위해 64로 설정되고, 마지막 및 두 번째 마지막 테이블에 대해 256으로 설정되고, 나머지 테이블들에 대해 128로 설정된다. 이 예는 또한 [00159]-[00160] 단락들에 기술된 바와 같이 테이블 3 및 테이블 4 생성을 위해 *AuxiliaryFunction*을 사용하는 방법을 포함한다. 테이블 4에 대해 직선 보조 함수는 6의 값에서 x-축과 교차한다. 테이블 4에 대해 직선 보조 함수는 8의 값에서 x-축과 교차한다. 이 예에서 사용되는 테이블들은 다음에 따른 값들을 갖는다:

table0 = { 0, 0, 2, 6, 10, 14, 19, 23, 28, 32, 36, 41, 45, 49, 53, 57, },
table1 = { 0, 0, 5, 12, 20, 29, 38, 47, 56, 65, 73, 82, 90, 98, 107, 115, },
table2 = { 0, 0, 1, 4, 9, 16, 24, 32, 41, 50, 59, 68, 77, 86, 94, 103, },
table3 = { 0, 0, 3, 9, 19, 32, 47, 64, 81, 99, 117, 135, 154, 179, 205, 230, },
table4 = { 0, 0, 0, 2, 6, 11, 18, 27, 38, 51, 64, 96, 128, 160, 192, 224, },

[0205]

[0206] 도 18은 다양한 실시예들을 구현하기 위해 사용될 수 있는 장치(600)의 블록도이다. 장치(600)는 도 1에 도시된 바와 같은 인코딩 장치 및 도 2에 도시된 바와 같은 디코딩 장치일 수 있다. 추가적으로, 장치(600)는 설명된 요소들 중 하나 이상을 호스팅할 수 있다. 일부 실시예들에서, 장치(600)는 스피커, 마이크로폰, 마우스, 터치스크린, 키패드, 키보드, 프린터, 디스플레이 등과 같은 하나 이상의 입력/출력 디바이스를 구비한다. 장치(600)는 버스에 연결된 하나 이상의 중앙 처리 유닛(CPU)(610), 메모리(620), 대용량 스토리지(630), 비디오 어댑터(640), 및 I/O 인터페이스(660)를 포함할 수 있다. 버스는 메모리 버스 또는 메모리 제어기, 주변 버스, 비디오 버스 등을 포함하는 임의의 유형의 수 개의 버스 아키텍처들 중 하나 이상이다.

[0207] CPU(610)는 임의의 유형의 전자 데이터 프로세서를 가질 수 있다. 메모리(620)는 SRAM(static random access memory), DRAM(dynamic random access memory), SDRAM(synchronous DRAM), ROM(read-only memory), 이들의 조합 등과 같은 임의의 유형의 시스템 메모리일 수 있거나, 이를 가질 수 있다. 일 실시예에서, 메모리(620)는 부팅 시에 사용하기 위한 ROM, 및 프로그램들을 실행하는 동안 사용하기 위한 프로그램 및 데이터 스토리지를 위한 DRAM을 포함할 수 있다. 실시예들에서, 메모리(620)는 비-일시적이다. 대용량 스토리지(630)는 데이터, 프로그램들, 및 다른 정보를 저장하고 데이터, 프로그램들, 및 다른 정보를 버스를 통해 액세스 가능하게 만드는 임의의 유형의 스토리지 디바이스를 포함한다. 대용량 스토리지(630)는, 예를 들어, 솔리드 스테이트 드라이브, 하드 디스크 드라이브, 자기 디스크 드라이브, 광 디스크 드라이브 등 중 하나 이상을 포함한다.

[0208] 비디오 어댑터(640) 및 I/O 인터페이스(660)는 외부 입력 및 출력 디바이스들을 장치(600)에 결합하기 위한 인터페이스들을 제공한다. 예를 들어, 장치(600)는 클라이언트들과의 SQL 커맨드 인터페이스를 제공할 수 있다. 예시된 바와 같이, 입력 및 출력 디바이스들의 예들은 비디오 어댑터(640)에 결합된 디스플레이(690) 및 I/O 인

터페이스(660)에 결합된 마우스/키보드/프린터(670)의 임의의 조합을 포함한다. 다른 디바이스들이 장치(600)에 결합될 수 있고, 추가적인 또는 더 적은 수의 인터페이스 카드들이 이용될 수 있다. 예를 들어, 프린터에 대한 직렬 인터페이스를 제공하기 위해 직렬 인터페이스 카드(도시되지 않음)가 사용될 수 있다.

[0209] 장치(600)는 또한 이더넷 케이블 등과 같은 유선 링크들, 및/또는 액세스 노드들 또는 하나 이상의 네트워크(680)에 대한 무선 링크들을 포함하는 하나 이상의 네트워크 인터페이스(650)를 포함한다. 네트워크 인터페이스(650)는 장치(600)가 네트워크들(680)을 통해 원격 유닛들과 통신할 수 있게 한다. 예를 들어, 네트워크 인터페이스(650)는 데이터베이스와의 통신을 제공할 수 있다. 일 실시예에서, 장치(600)는 다른 처리 유닛들, 인터넷, 원격 스토리지 설비들 등과 같은 원격 디바이스들과의 통신 및 데이터 처리를 위해 로컬-영역 네트워크 또는 광역 네트워크에 결합된다.

[0210] 루프 내 필터 또는 예측 필터의 제안된 설계는 ALF와 같은 종래의 적응 필터링 방법에 대해 다음의 이점들을 갖는다:

[0211] • 제안된 주파수 도메인 필터는 디코더 측에서 재구성된 프레임 또는 예측된 블록으로부터 필터링 파라미터들(주파수 도메인 이득 계수들)을 도출하고 따라서 필터링 파라미터들이 인코더로부터 디코더 측으로 전송될 필요가 없다.

[0212] • ALF는 송신을 위한 가중 계수들의 수를 감소시키기 위해 인코더 측에서 복잡한 레이트 왜곡 최적화(rate distortion optimization, RDO)를 필요로 한다. 제안된 방법은 인코더 측에서 복잡한 RDO를 필요로 하지 않고 (파라미터 전송이 없음) 미리 정의된 조건들을 충족시키는 모든 블록들에 적용된다.

[0213] • ALF는 픽셀 도메인에서 선형 필터이다. 제안된 필터는 비-선형인데 그 이유는 각각의 1D 스펙트럼 성분에 대한 이득 계수가 이 스펙트럼 성분 값에 의존하기 때문이다. 이는 비-선형 처리로부터 추가적인 코딩 이득을 달성할 수 있게 한다.

[0214] • ALF는 디코더 측에서 범용 승산기들을 필요로 한다. 제안된 방법에서 필터링은 룩업 테이블로서 구현될 수 있는데, 그 이유는 각각의 스펙트럼 계수에 대한 이득이 1보다 작기 때문이다. 따라서 제안된 방법은 어떠한 곱셈도 없이 구현될 수 있다.

[0215] 따라서, 낮은 복잡도로 비디오 코딩을 위한 효율을 개선하는 것을 허용하는 필터가 제공된다.

[0216] 본 개시의 특정 특징 또는 양태가 수 개의 구현들 또는 실시예들 중 하나에 대해서만 개시되었을 수 있지만, 그러한 특징 또는 양태는 임의의 주어진 또는 특정 응용에 대해 원하고 유리할 수 있는 다른 구현들 또는 실시예들의 하나 이상의 다른 특징 또는 양태와 조합될 수 있다. 더욱이, 용어들 "포함하다(include)", "갖다(have)", "갖는(with)", 또는 이들의 다른 변형들이 상세한 설명 또는 청구항들에서 사용되는 한, 그러한 용어들은 용어 "포함하다(comprise)"와 유사한 방식으로 포괄적인 것으로 의도된다. 또한, "예시적인(exemplary)", "예를 들어(for example)" 및 "예를 들어(e.g.)"라는 용어들은, 최상 또는 최적이라기보다는, 단지 예로서 의도된다. "결합된(coupled)" 및 "연결된(connected)"이라는 용어들이 파생어들과 함께 사용될 수 있다. 이들 용어는 2개의 요소가 서로 협력하거나 상호작용하는 것을 지시하기 위해 사용되었을 수 있다는 것을 이해해야 한다 - 그것들이 직접 물리적 또는 전기적 접촉하고 있는지, 또는 그것들이 서로 직접 접촉하고 있지 않은지에 관계없이 -.

[0217] 특정 양태들이 본 명세서에 예시되고 설명되었지만, 본 기술분야의 통상의 기술자들은 본 개시의 범위를 벗어나지 않고 다양한 대안 및/또는 등가 구현들이 도시되고 설명된 특정 양태들을 대체할 수 있다는 것을 이해할 것이다. 본 출원은 본 명세서에서 논의된 특정 양태들의 임의의 적응들 또는 변형들을 포함하도록 의도된다.

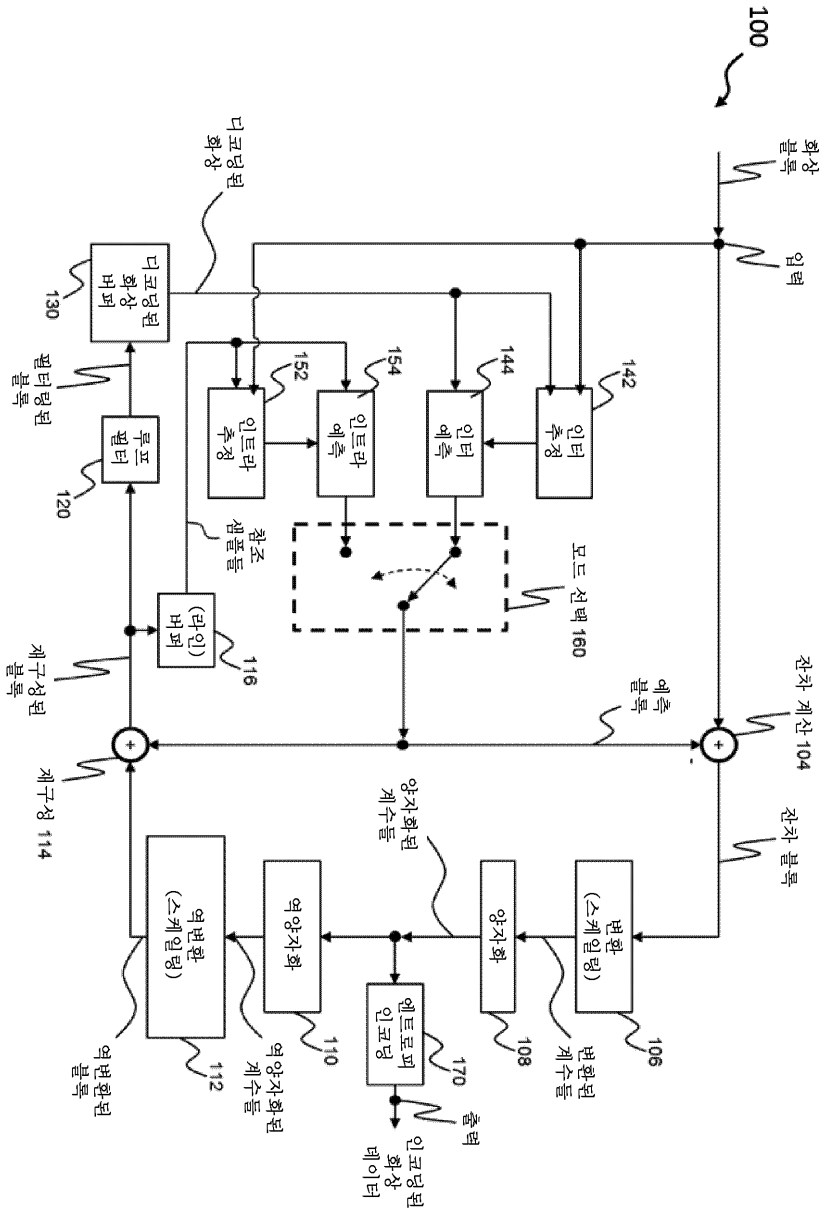
[0218] 다음의 청구항들에서의 요소들은 대응하는 라벨링과 함께 특정 시퀀스로 열거되지만, 그 청구항 열거들이 해당 요소들의 일부 또는 전부를 구현하기 위한 특정 시퀀스를 달리 암시하지 않는 한, 해당 요소들은 반드시 그 특정 시퀀스로 구현되는 것으로 제한되도록 의도되는 것은 아니다.

[0219] 상기 교시에 비추어 다수의 대안들, 수정들 및 변형들이 본 기술분야의 통상의 기술자들에게 명백할 것이다. 물론, 본 기술분야의 통상의 기술자들은 본 명세서에 설명된 것들 이외에 본 발명의 다수의 응용이 있다는 것을 쉽게 인식한다. 본 발명이 하나 이상의 특정 실시예를 참조하여 설명되었지만, 본 기술분야의 통상의 기술자들은 본 발명의 범위를 벗어나지 않고 그에 대해 다수의 변화들이 이루어질 수 있다는 것을 인식한다. 따라서, 첨부된 청구항들 및 그것들의 균등물들의 범위 내에서, 본 발명은 본 명세서에서 구체적으로 설명된 것과 달리

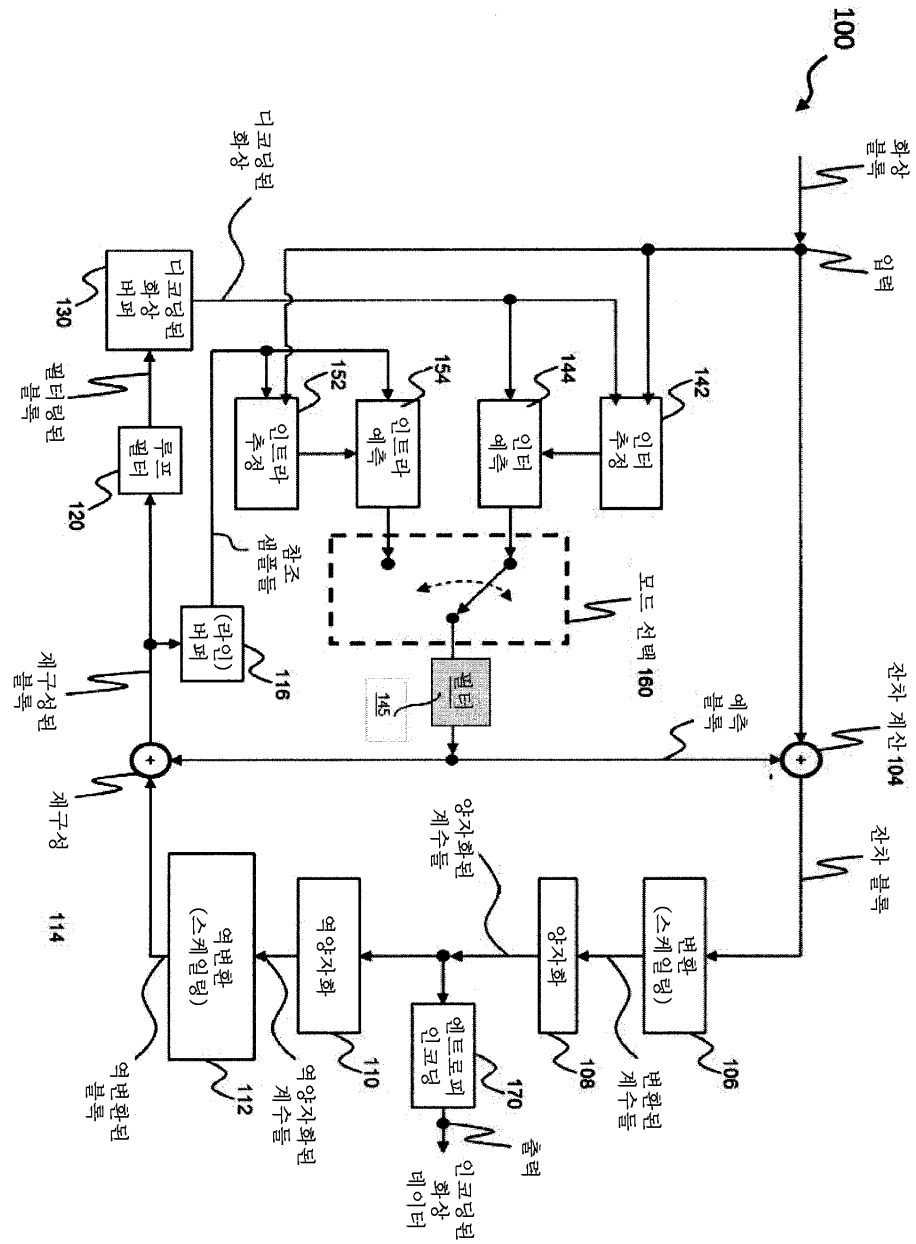
실시될 수 있다는 것을 이해해야 한다.

도면

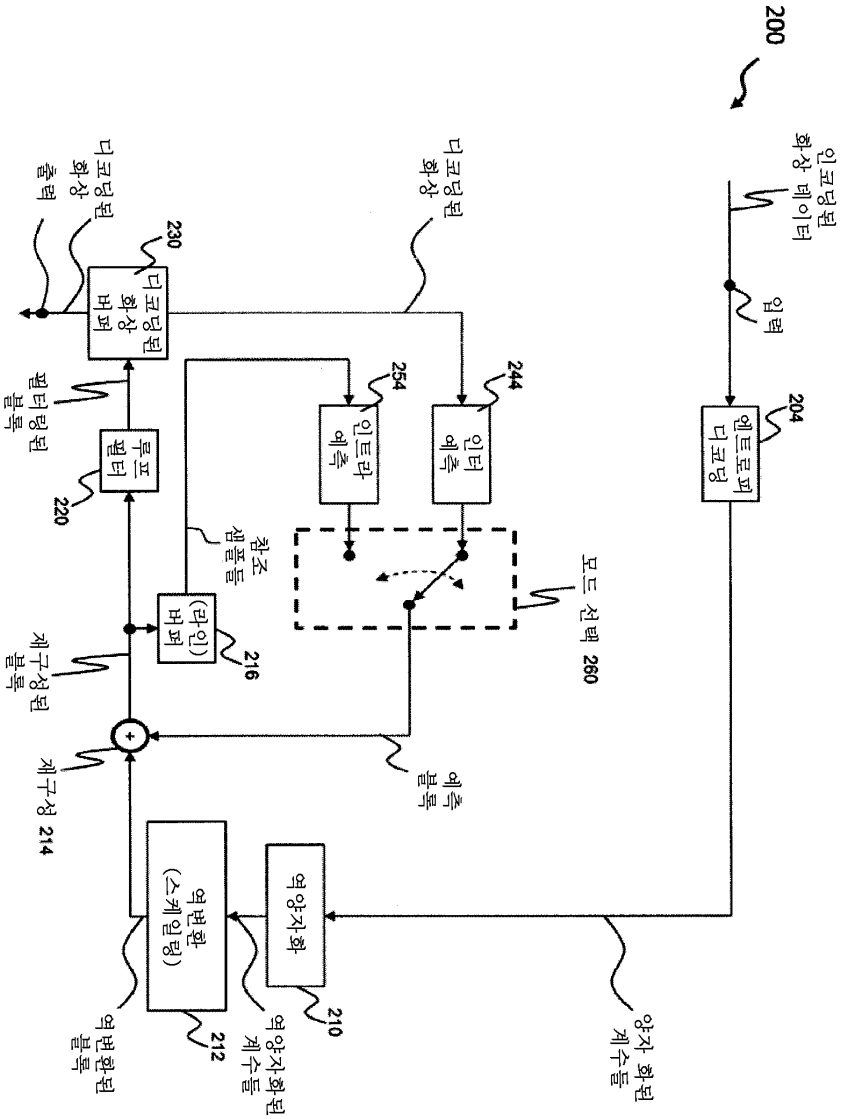
도면1a



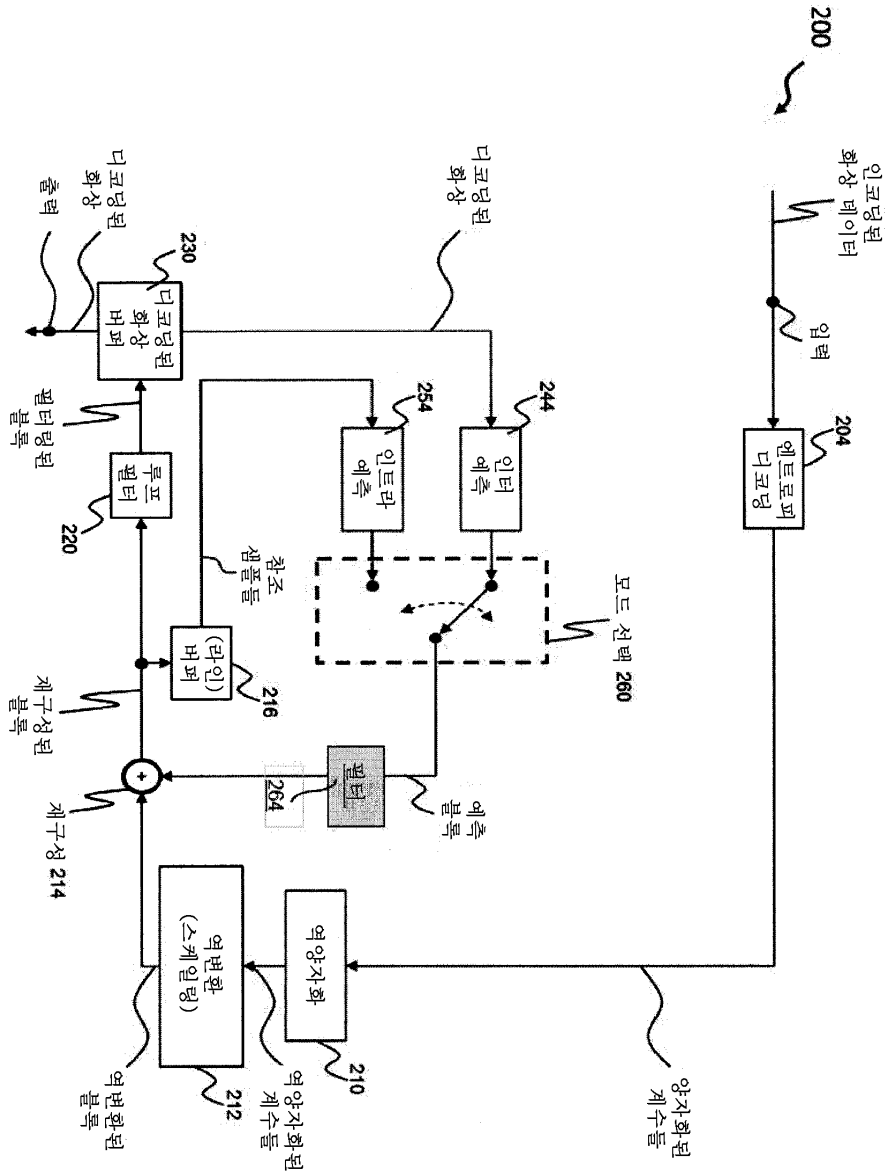
도면1b


100
화상 블록
입력
디코딩된 화상
디코딩된 화상 버퍼
130
필터링된 블록
루프 필터
120
152
인트라 추정
154
인트라 예측
144
인터 예측
142
인터 추정
참조 샘플들
모드 선택 160
(라인) 버퍼
116
필터
145
예측 블록
잔차 계산 104
재구성된 블록
재구성
114
양자화된 계수들
잔차 블록
역변환 (스케일링)
112
역양자화
110
양자화
108
변환 (스케일링)
106
엔트로피 인코딩
170
역변환된 블록
역양자화된 계수들
변환된 계수들
출력
인코딩된 화상 데이터

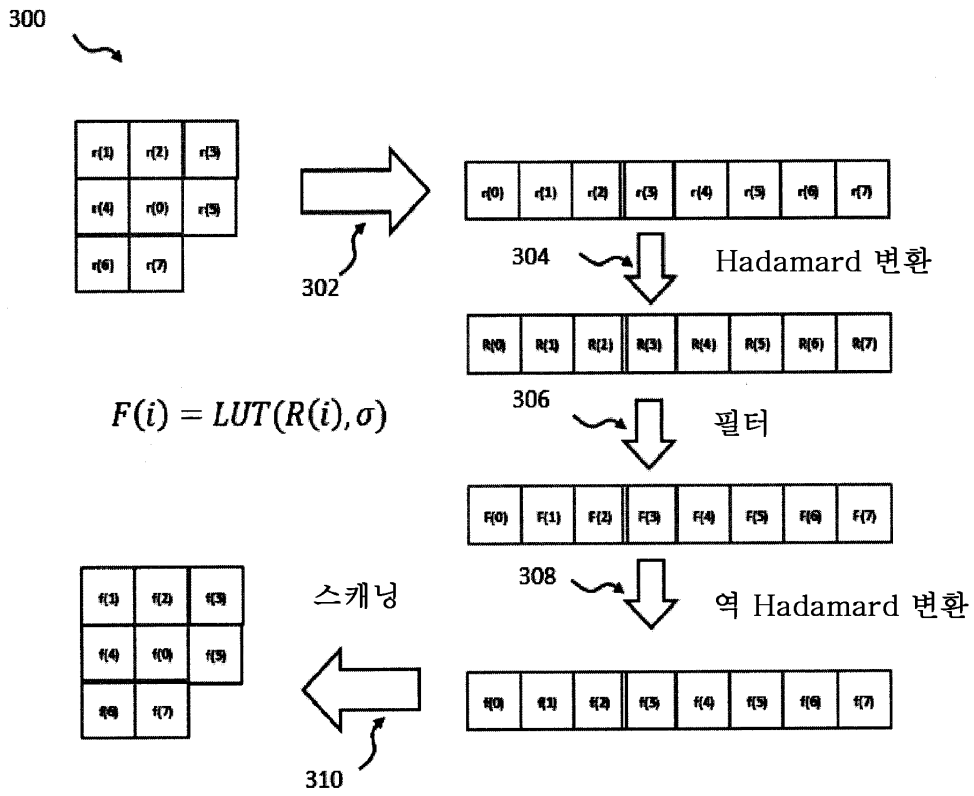
도면2a


200
인코딩된 확장 데이터
입력
204
엔트로피 디코딩
양자 화된 계수들
역양자화
210
역양자화된 계수들
역변환 (스케일링)
212
역변환된 블록
디코딩된 확장 출력
디코딩된 화상 버퍼
230
디코딩된 확장
필터링된 블록
루프 필터
220
인트라 예측
254
인터 예측
244
참조 샘플들
(라인) 버퍼
216
모드 선택 260
예측 블록
제구성된 블록
제구성 214

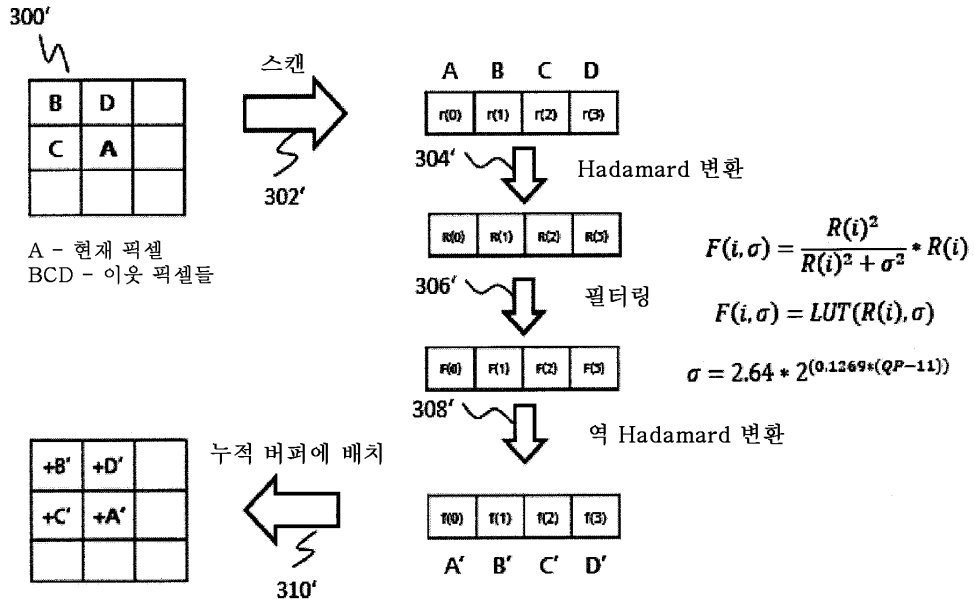
도면2b


200
인코딩된 화상 데이터
입력
204
엔트로피 디코딩
양자화된 계수들
디코딩된 화상
244
인터 예측
254
인트라 예측
모드 선택 260
참조 샘플들
(라인) 버퍼
216
루프 필터
220
필터링된 블록
디코딩된 화상 출력
230
디코딩된 화상 버퍼
예측 블록
필터
264
재구성 214
재구성된 블록
역양자화
210
역양자화된 계수들
역변환 (스케일링)
212
역변환된 블록

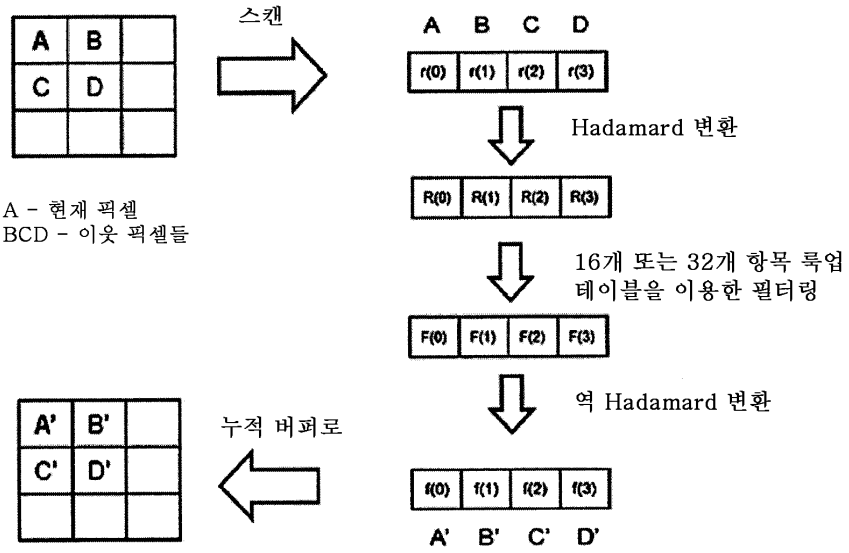
도면3a


300
302
304
Hadamard 변환
306
필터
$F(i) = LUT(R(i), \sigma)$
308
역 Hadamard 변환
스캐닝
310

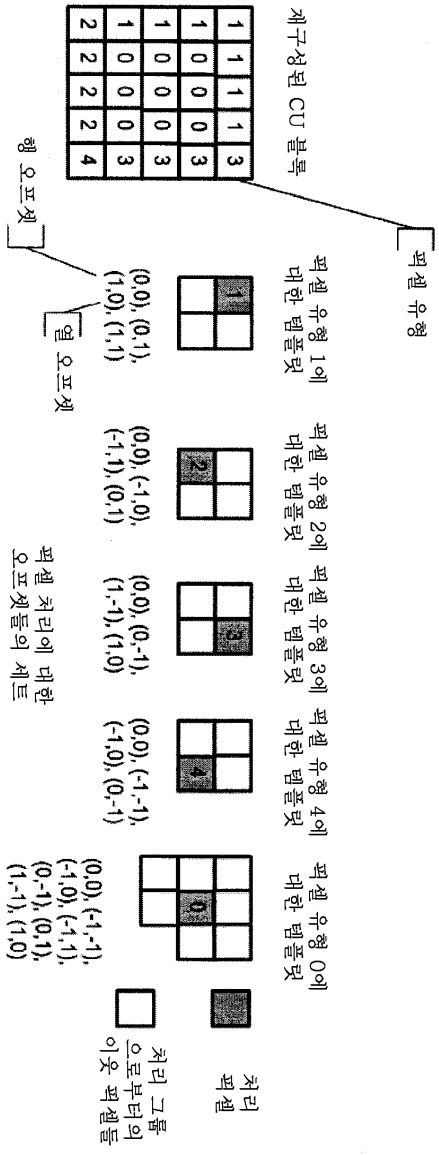
도면3b


300'
스캔
302'
A - 현재 픽셀
BCD - 이웃 픽셀들
304'
Hadamard 변환
306'
필터링
$F(i,\sigma) = \frac{R(i)^2}{R(i)^2 + \sigma^2} * R(i)$
$F(i,\sigma) = LUT(R(i),\sigma)$
$\sigma = 2.64 * 2^{(0.1269*(QP-11))}$
308'
역 Hadamard 변환
누적 버퍼에 배치
310'

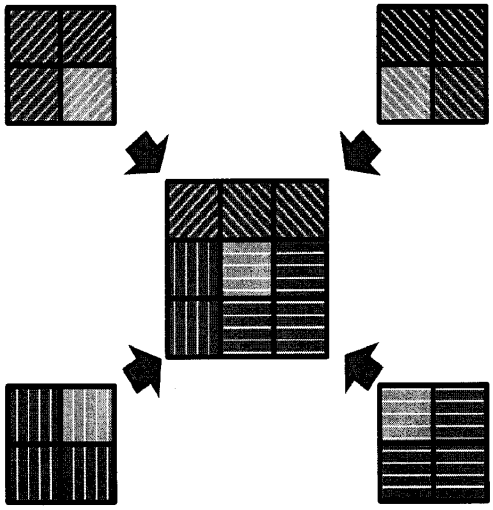
도면3c


A
B
C
D
스캔
A B C D
r(0) r(1) r(2) r(3)
Hadamard 변환
R(0) R(1) R(2) R(3)
16개 또는 32개 항목 룩업
테이블을 이용한 필터링
F(0) F(1) F(2) F(3)
역 Hadamard 변환
f(0) f(1) f(2) f(3)
A' B' C' D'
누적 버퍼로
A'
B'
C'
D'
A - 현재 픽셀
BCD - 이웃 픽셀들

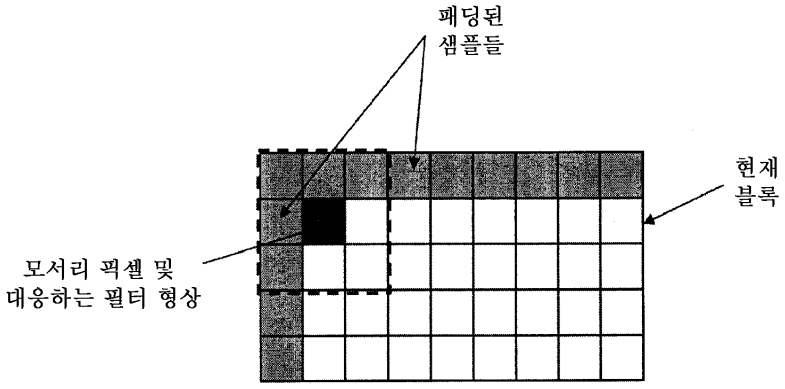
도면4a


재구성된 CU 블록
픽셀 유형
행 오프셋
열 오프셋
픽셀 유형 1에 대한 템플릿
픽셀 유형 2에 대한 템플릿
픽셀 유형 3에 대한 템플릿
픽셀 유형 4에 대한 템플릿
픽셀 유형 0에 대한 템플릿
(0,0), (0,1), (1,0), (1,1)
(0,0), (-1,0), (-1,1), (0,1)
(0,0), (0,-1), (1,-1), (1,0)
(0,0), (-1,-1), (-1,0), (0,-1)
(0,0), (-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0)
픽셀 처리에 대한 오프셋들의 세트
처리 픽셀
처리 그룹으로부터의 이웃 픽셀들

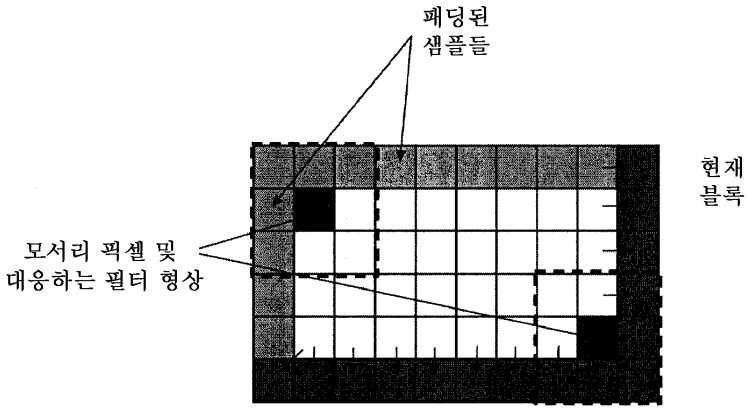
도면4b



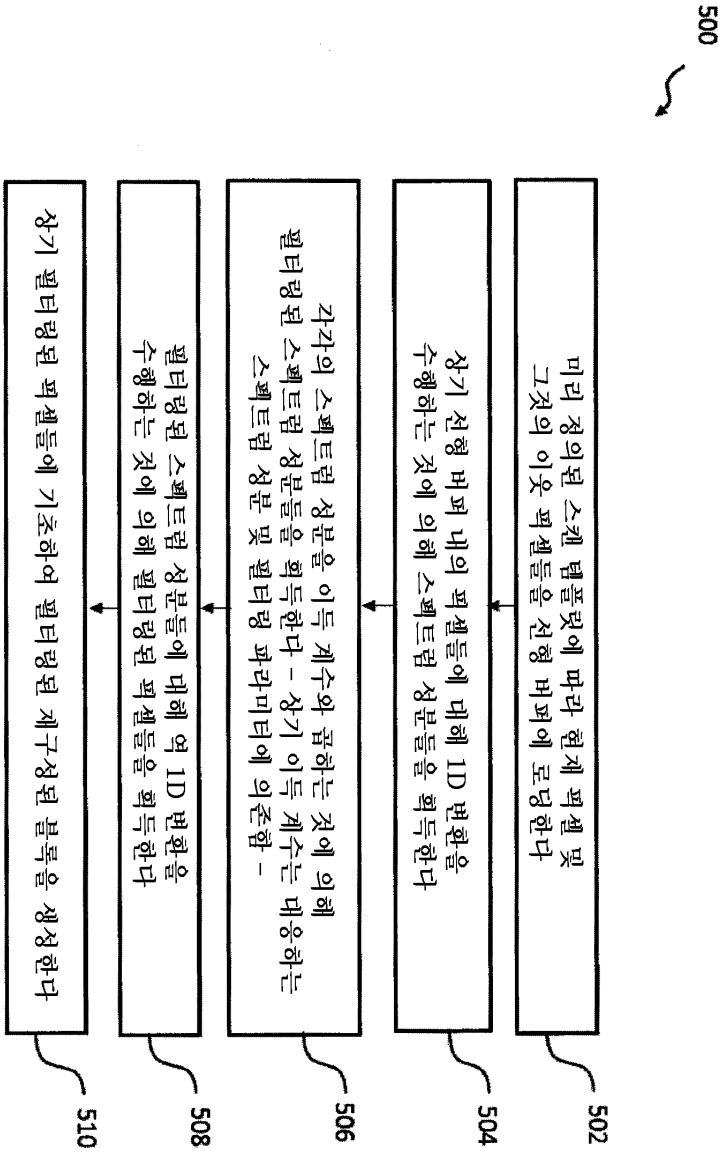
도면4c



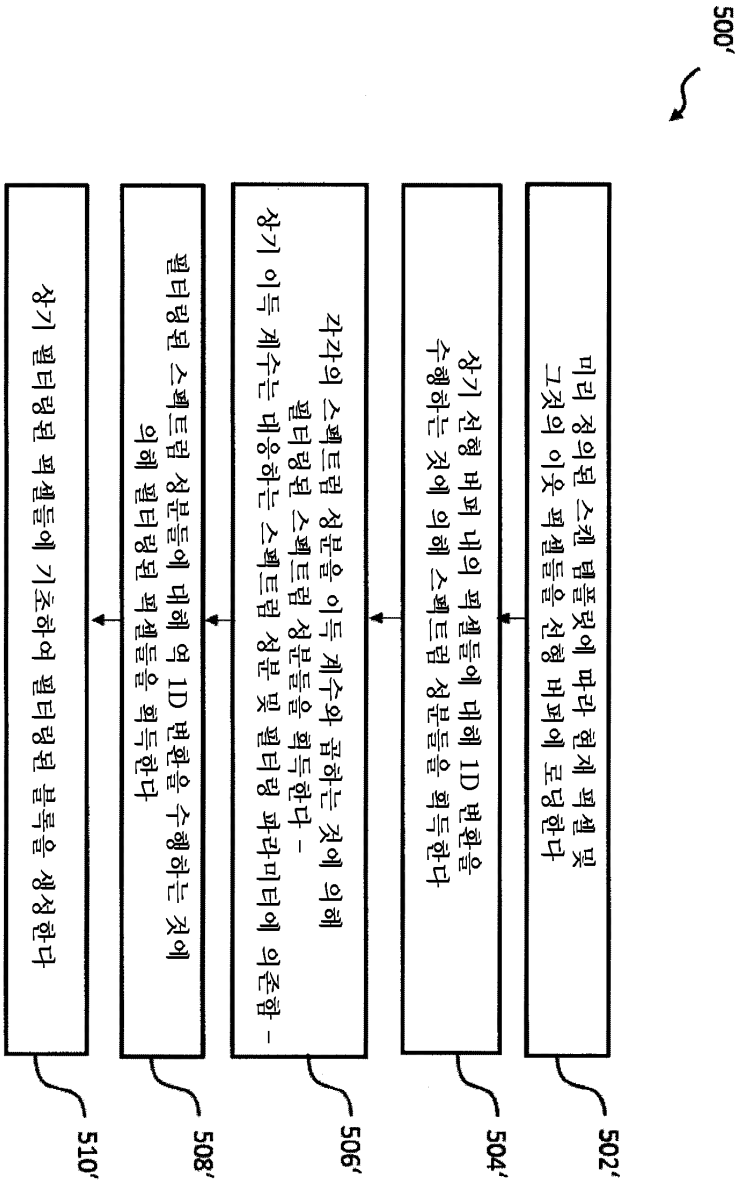
도면4d



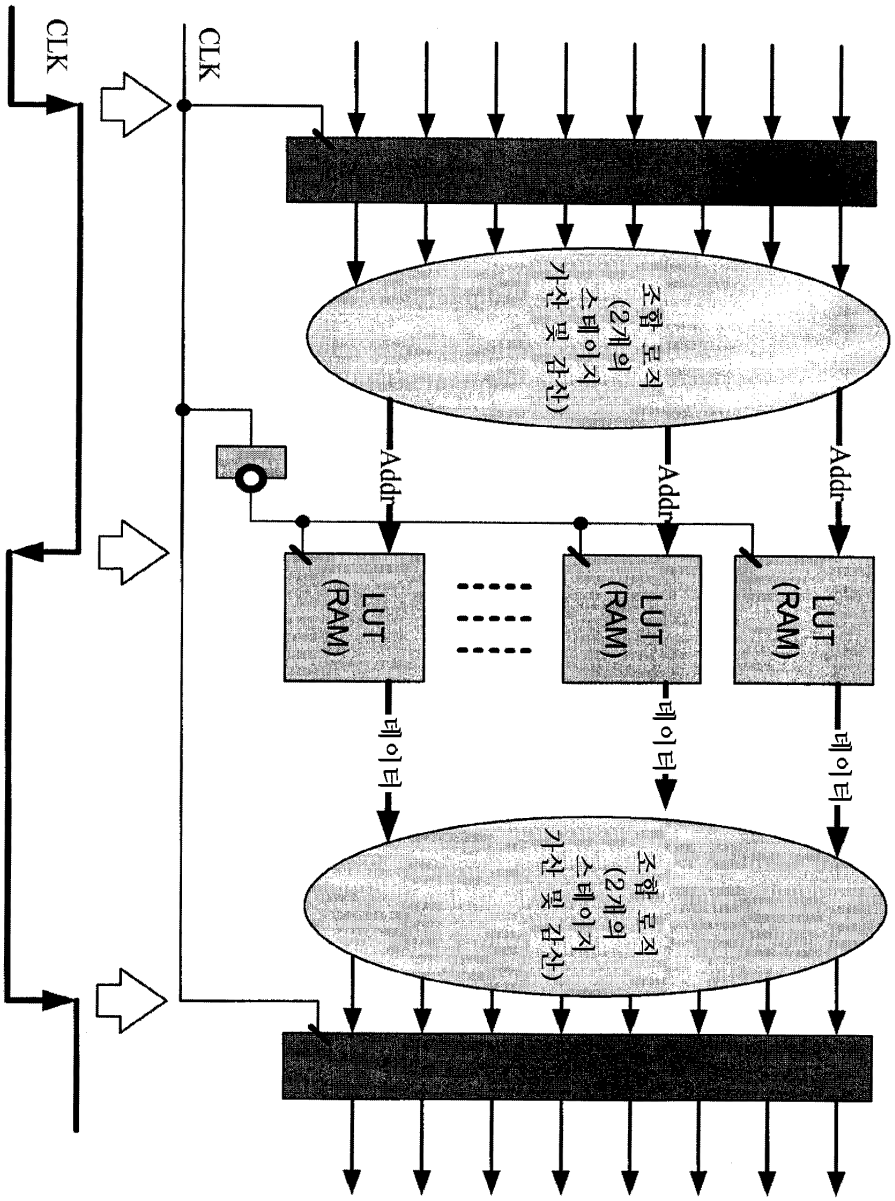
도면5a


500
미리 정의된 스캔 템플릿에 따라 현재 픽셀 및 그것의 이웃 픽셀들을 선형 버퍼에 로딩한다
502
상기 선형 버퍼 내의 픽셀들에 대해 1D 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득한다
504
각각의 스펙트럼 성분을 이득 계수와 곱하는 것에 의해 필터링된 스펙트럼 성분들을 획득한다 - 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존함 -
506
필터링된 스펙트럼 성분들에 대해 역 1D 변환을 수행하는 것에 의해 필터링된 픽셀들을 획득한다
508
상기 필터링된 픽셀들에 기초하여 필터링된 재구성된 블록을 생성한다
510

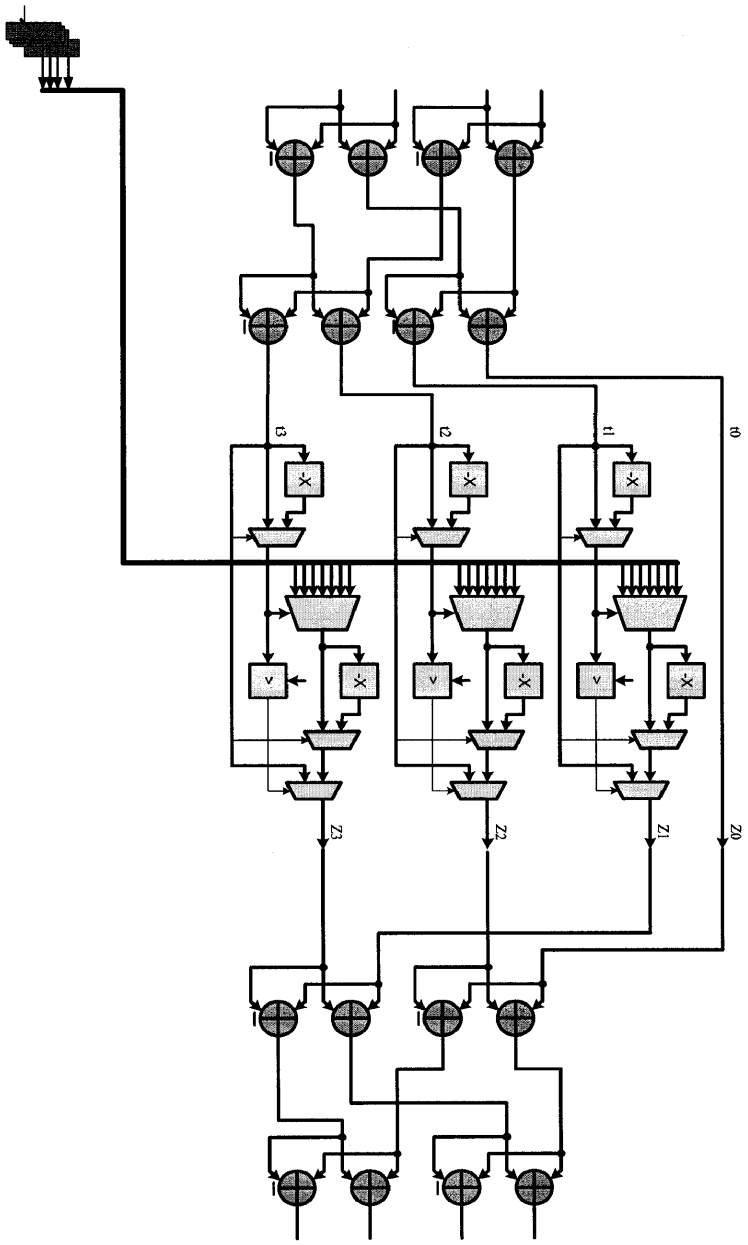
도면5b


500'
미리 정의된 스캔 탭플릿에 따라 현재 픽셀 및 그것의 이웃 픽셀들을 선형 버퍼에 로딩한다
502'
상기 선형 버퍼 내의 픽셀들에 대해 1D 변환을 수행하는 것에 의해 스펙트럼 성분들을 획득한다
504'
각각의 스펙트럼 성분을 이득 계수와 곱하는 것에 의해 필터링된 스펙트럼 성분들을 획득한다 - 상기 이득 계수는 대응하는 스펙트럼 성분 및 필터링 파라미터에 의존함 -
506'
필터링된 스펙트럼 성분들에 대해 역 1D 변환을 수행하는 것에 의해 필터링된 픽셀들을 획득한다
508'
상기 필터링된 픽셀들에 기초하여 필터링된 블록을 생성한다
510'

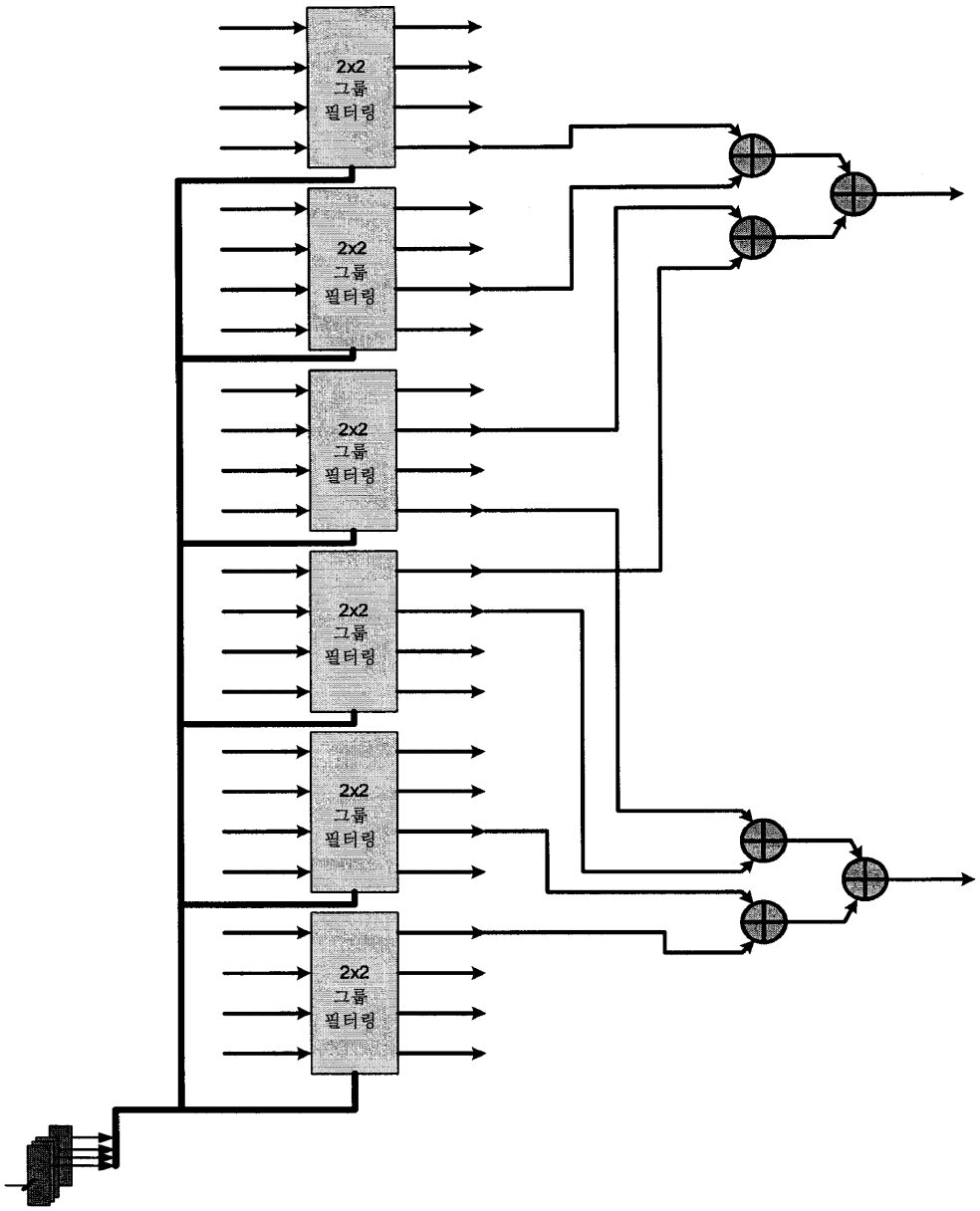
도면6


CLK
CLK
조합 로직
(2개의
스테이지
가산 및 감산)
Addr
Addr
Addr
LUT
(RAM)
LUT
(RAM)
LUT
(RAM)
데이터
데이터
데이터
조합 로직
(2개의
스테이지
가산 및 감산)

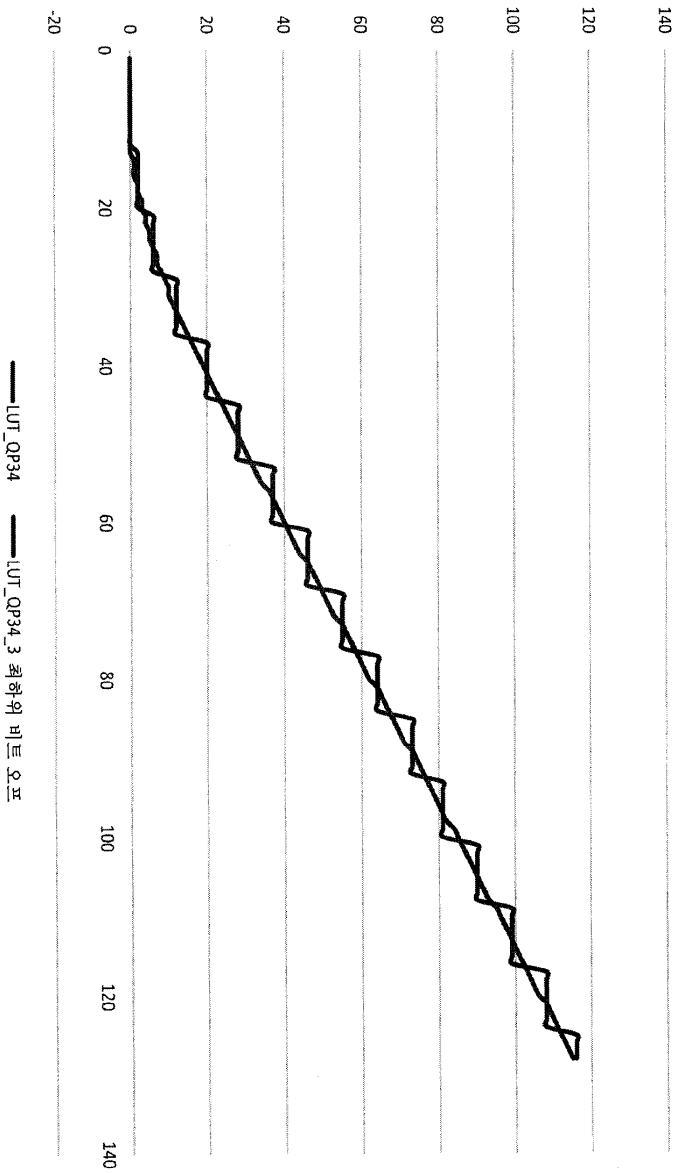
도면7



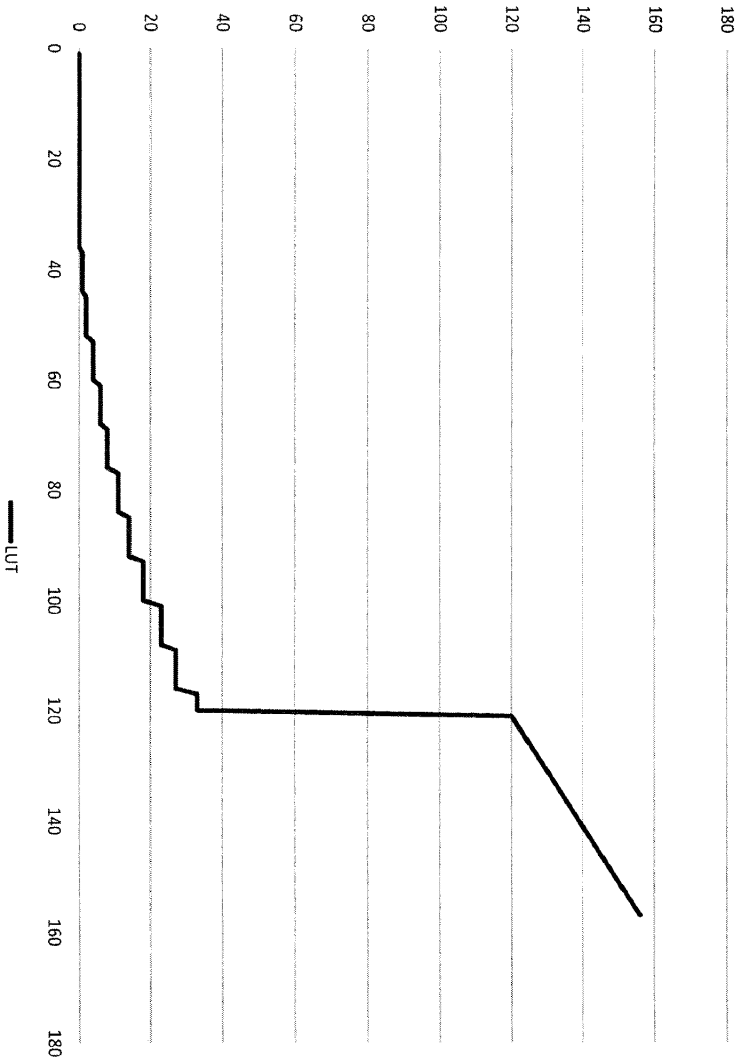
도면8



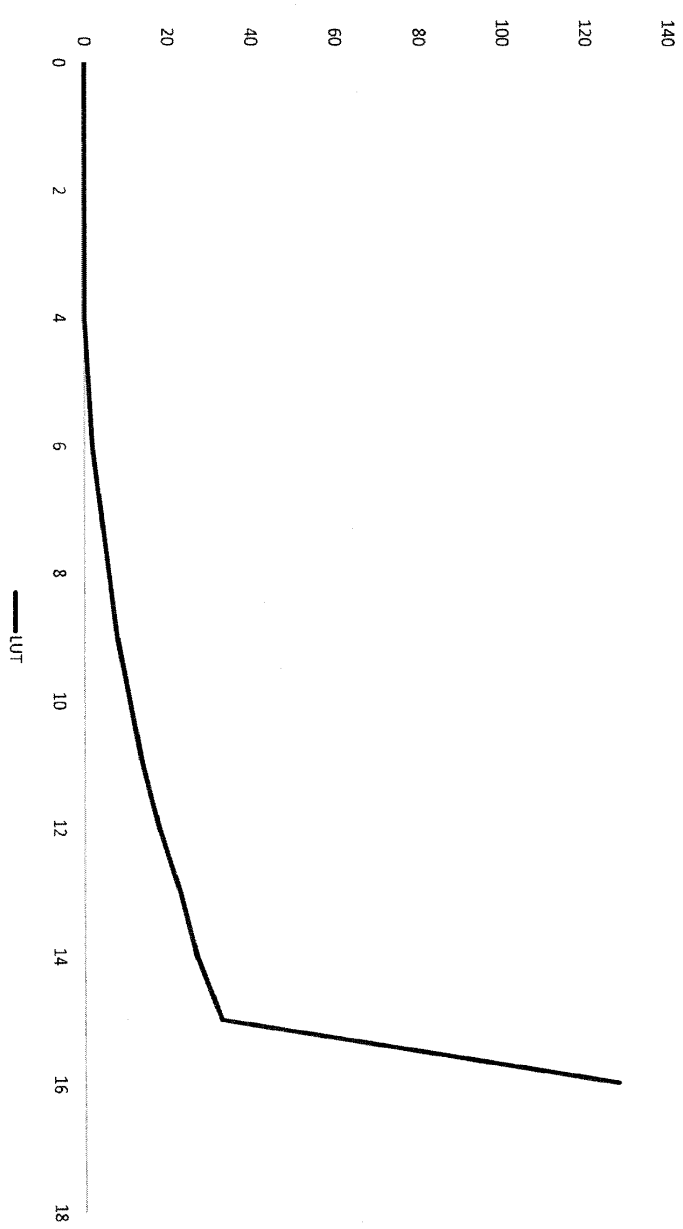
도면9


-20
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140
LUT_QP34
LUT_QP34_3 최하위 비트 오프

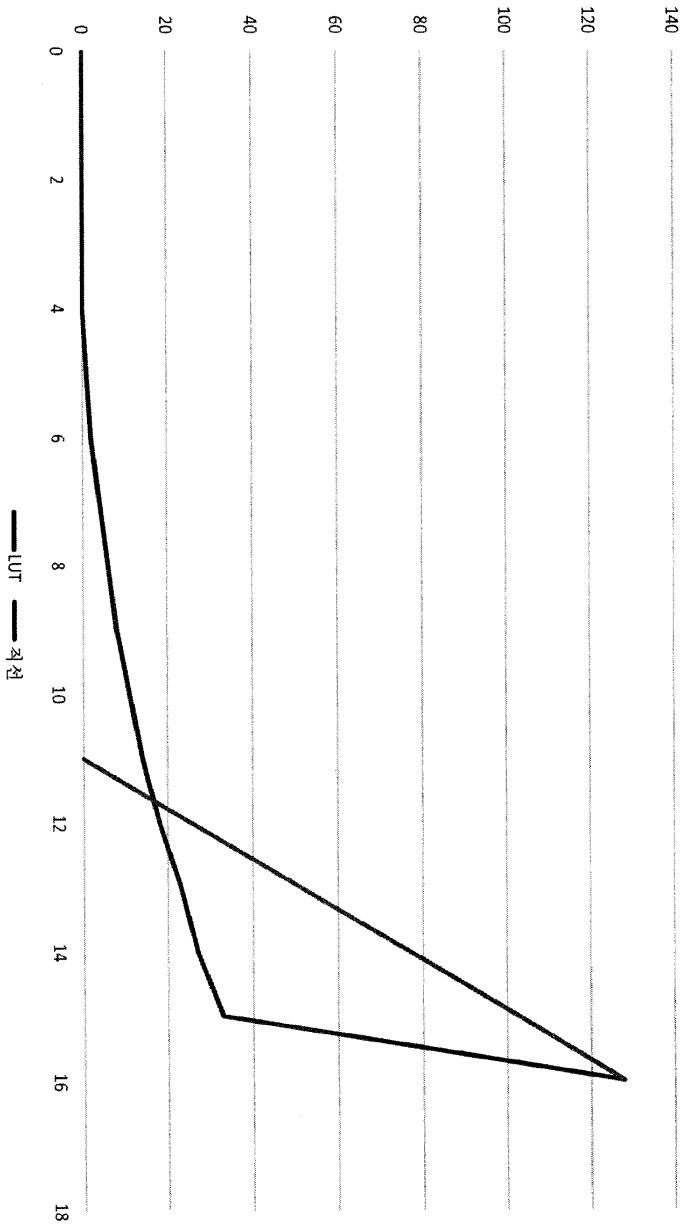
도면10



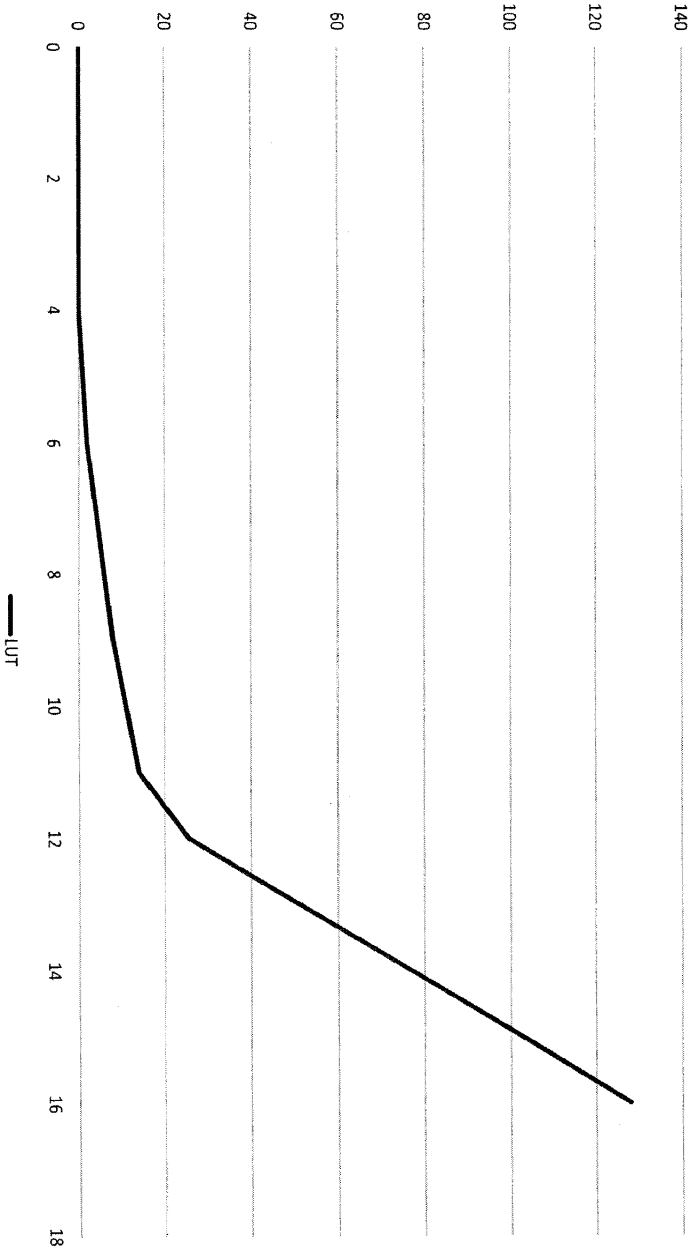
도면11



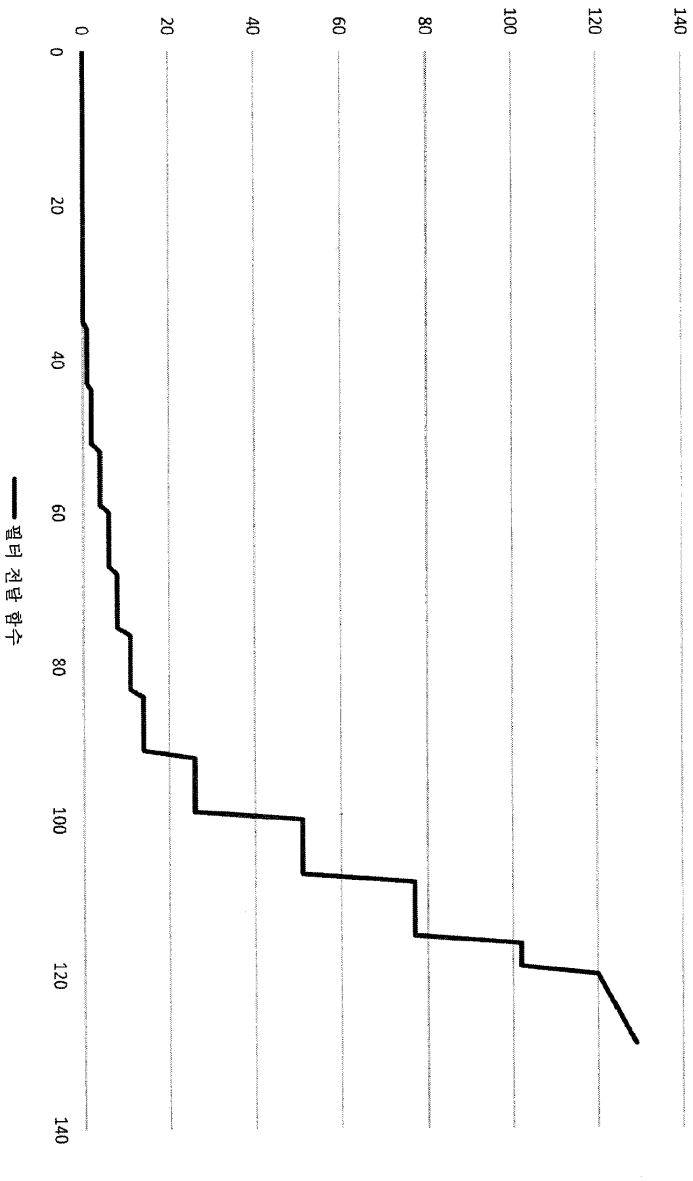
도면12



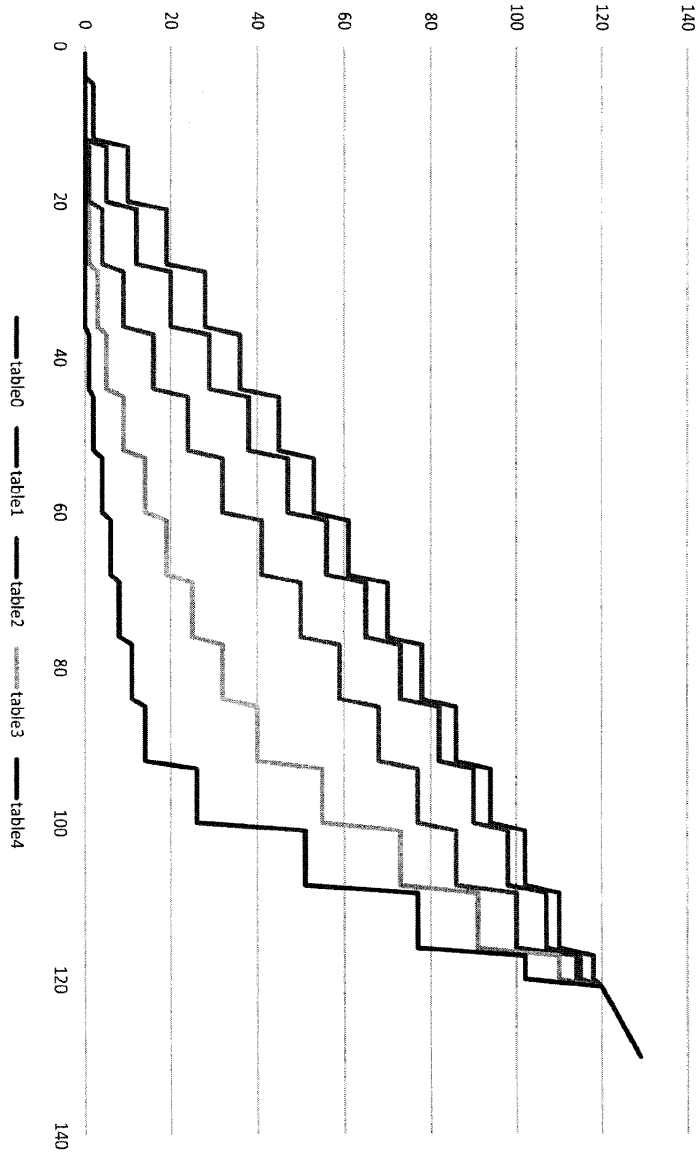
도면13



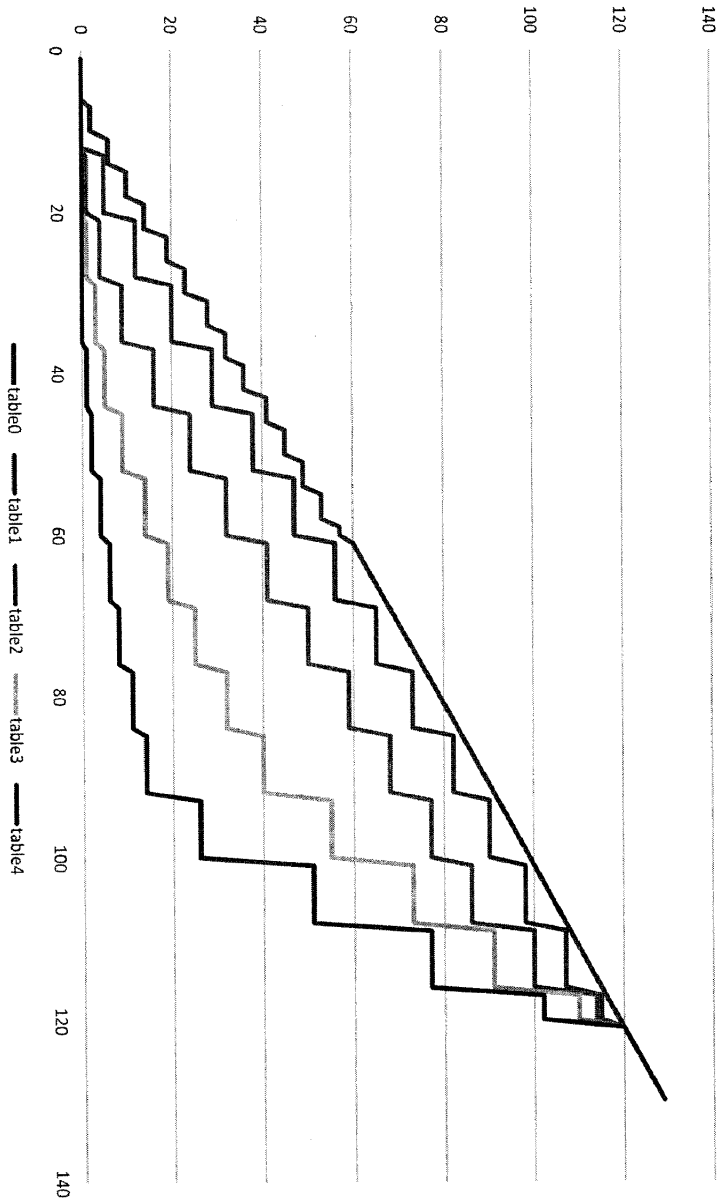
도면14



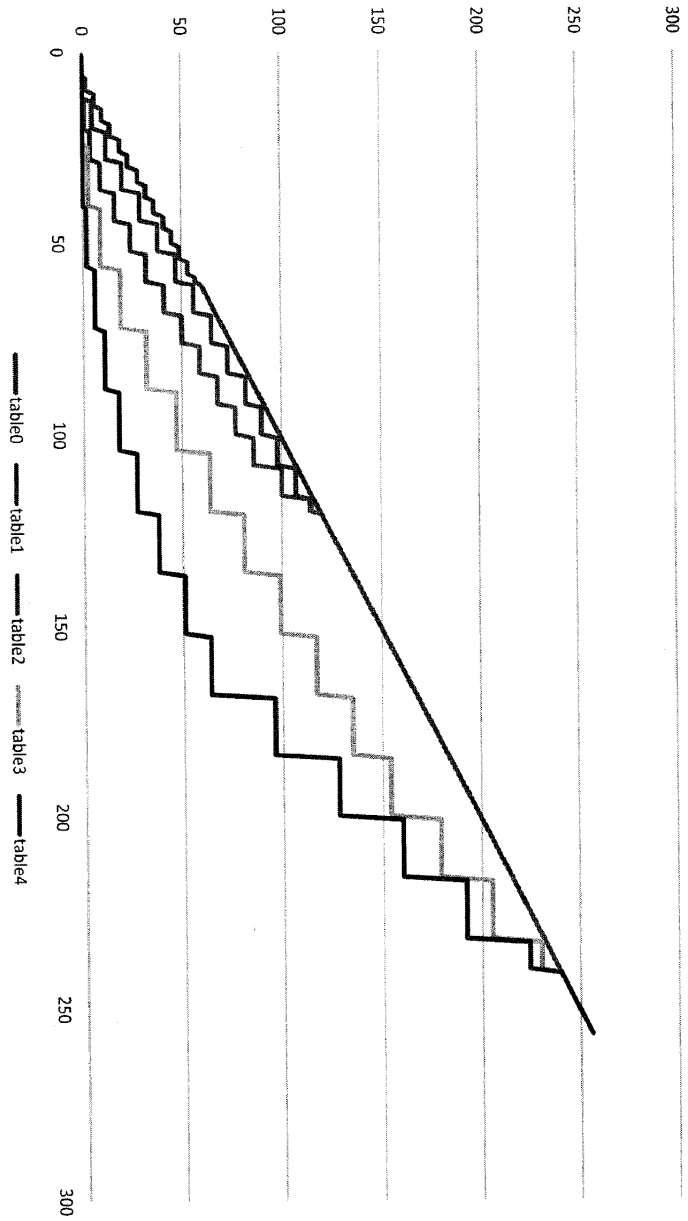
도면15



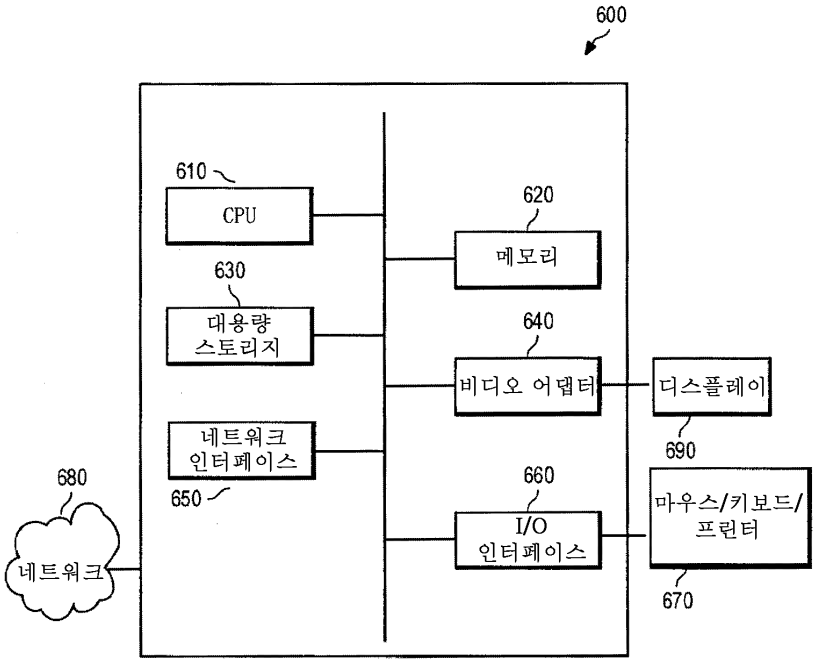
도면16



도면17



도면18


600
610
CPU
620
메모리
630
대용량
스토리지
640
비디오 어댑터
디스플레이
690
네트워크
인터페이스
650
680
네트워크
660
I/O
인터페이스
마우스/키보드/
프린터
670