(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년12월22일
(11) 등록번호 10-1811360
(24) 등록일자 2017년12월15일

(51) 국제특허분류(Int. Cl.)
H04N 19/11 (2014.01) *H04N 19/176* (2014.01)
H04N 19/593 (2014.01) *H04N 19/61* (2014.01)
H04N 19/82 (2014.01)
(52) CPC특허분류
H04N 19/11 (2015.01)
H04N 19/176 (2015.01)
(21) 출원번호 10-2016-7020283(분할)
(22) 출원일자(국제) 2011년07월14일
심사청구일자 2016년07월25일
(85) 번역문제출일자 2016년07월25일
(65) 공개번호 10-2016-0092055
(43) 공개일자 2016년08월03일
(62) 원출원 특허 10-2013-7000687
원출원일자(국제) 2011년07월14일
심사청구일자 2015년06월15일
(86) 국제출원번호 PCT/US2011/044014
(87) 국제공개번호 WO 2012/009540
국제공개일자 2012년01월19일
(30) 우선권주장
61/364,322 2010년07월14일 미국(US)
61/388,541 2010년09월30일 미국(US)
(56) 선행기술조사문헌
Samsung's Response to the Call for Proposals on Video Compression Technology, Joint collaborative team on viedo coding of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JCTVC-A124 (2010.04.23.)*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
가부시키가이샤 엔.티.티.도코모
일본 도쿄도 지요다쿠 나가타쵸 2쵸메 11반 1고
(72) 발명자
보센 프랭크 잰
미국 94304 캘리포니아 팔로 알토 3240 힐뷰 애비뉴 도코모 커뮤니케이션즈 라보래토리즈 미국 법인내
탄 티오 켕
싱가포르 808379 싱가포르 잘란 신도르 24
(74) 대리인
유미특허법인

전체 청구항 수 : 총 4 항 심사관 : 장석환

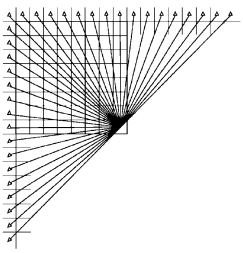
(54) 발명의 명칭 **비디오 코딩을 위한 저 복잡성 인트라 예측**

(57) 요 약

본 발명은 비디오 코딩의 효율성을 향상시키는 독특한 인트라 예측 프로세스를 제공한다. H.264/AVC는 예측할 타겟 블록의 정확히 위에 위치한 수평 경계 내 참조 화소들과, 상기 타겟 블록의 정확히 왼쪽에 위치한 수직 경계 내 참조 화소들을 이용한다. 본 발명에서, 수평 경계 화소들의 어레이나 수직 경계 화소들의 어레이 중 적어도

(뒷면에 계속)

대 표 도 - 도8



일부가 검색된다. 그런 다음 상기 검색 화소들은 다른 경계 화소들에 추가되어서 그 어레이를 연장하게 된다. 인트라 예측은 단지 경계 화소들의 상기 연장된 어레이를 기반으로만 수행된다.

(52) CPC특허분류
H04N 19/593 (2015.01)
H04N 19/61 (2015.01)
H04N 19/82 (2015.01)

명 세 서

청구범위

청구항 1

비디오 인코딩 방법으로서,

상기 비디오 인코딩 방법은, 인트라 예측 앵글(angle)을 따라 보간된 타겟 블록의 경계 화소로 타겟 블록의 예측 블록을 도출하는 인트라 예측 동작을 수행하는 비디오 인코더의 프로세서에 의해 실행되는, 컴퓨터로 실행 가능한 단계를 포함하고, 상기 경계 화소는 수평 경계 화소의 수평 어레이 및 수직 경계 화소의 수직 어레이를 포함하며, 상기 인트라 예측 동작은,

복수의 상이한 인트라 예측 앵글과 각각 관련되어 있는 인버스 앵글(inverse angle) 파라미터의 값들을 열거한 룩업 테이블로부터, 인트라 예측 앵글에 대응되는 인버스 앵글 파라미터의 값을 획득하는 단계;

상기 획득된 인버스 앵글 파라미터의 값과, 연장된 수평 어레이의 연장부(extension)에서의 위치를 나타내는 변수인 수평 위치 식별자의 값 간의 곱의 함수인 위치에서 수직 어레이 내에 위치한 수직 경계 화소 중 적어도 일부를 식별하는 단계;

상기 식별된 수직 경계 화소 중 적어도 일부를 수평 경계 화소로서 상기 연장된 수평 어레이의 연장부에 추가하는 단계; 및

상기 타겟 블록의 예측 블록을 도출하기 위해, 상기 수직 경계 화소의 사용 없이, 상기 연장된 수평 어레이 내의 수평 경계 화소만을 사용하는 단계

를 포함하는,

비디오 인코딩 방법.

청구항 2

비디오 디코딩 방법으로서,

상기 비디오 디코딩 방법은, 인트라 예측 앵글을 따라 보간된 타겟 블록의 경계 화소로 타겟 블록의 예측 블록을 도출하는 인트라 예측 동작을 수행하는 비디오 인코더의 프로세서에 의해 실행되는, 컴퓨터로 실행 가능한 단계를 포함하고, 상기 경계 화소는 수평 경계 화소의 수평 어레이 및 수직 경계 화소의 수직 어레이를 포함하며, 상기 인트라 예측 동작은,

복수의 상이한 인트라 예측 앵글과 각각 관련되어 있는 인버스 앵글 파라미터의 값들을 열거한 룩업 테이블로부터, 인트라 예측 앵글에 대응되는 인버스 앵글 파라미터의 값을 획득하는 단계;

상기 획득된 인버스 앵글 파라미터의 값과, 연장된 수평 어레이의 연장부에서의 위치를 나타내는 변수인 수평 위치 식별자의 값 간의 곱의 함수인 위치에서 수직 어레이 내에 위치한 수직 경계 화소 중 적어도 일부를 식별하는 단계;

상기 식별된 수직 경계 화소 중 적어도 일부를 수평 경계 화소로서 상기 연장된 수평 어레이의 연장부에 추가하는 단계; 및

상기 타겟 블록의 예측 블록을 도출하기 위해, 상기 수직 경계 화소의 사용 없이, 상기 연장된 수평 어레이 내의 수평 경계 화소만을 사용하는 단계

를 포함하는,

비디오 디코딩 방법.

청구항 3

비디오 인코더로서,

상기 비디오 인코더는, 컴퓨터 시스템의 프로세서와, 인트라 예측 앵글을 따라 보간된 타겟 블록의 경계 화소로 타겟 블록의 예측 블록을 도출하는 인트라 예측 동작을 수행하도록 상기 프로세서에 의해 실행 가능한 프로그램을 저장하는 메모리를 포함하고, 상기 경계 화소는 수평 경계 화소의 수평 어레이 및 수직 경계 화소의 수직 어레이를 포함하며, 상기 인트라 예측 동작은,

복수의 상이한 인트라 예측 앵글과 각각 관련되어 있는 인버스 앵글(inverse angle) 파라미터의 값들을 열거한 룩업 테이블로부터, 인트라 예측 앵글에 대응되는 인버스 앵글 파라미터의 값을 획득하고,

상기 획득된 인버스 앵글 파라미터의 값과, 연장된 수평 어레이의 연장부에서의 위치를 나타내는 변수인 수평 위치 식별자의 값 간의 곱의 함수인 위치에서 수직 어레이 내에 위치한 수직 경계 화소 중 적어도 일부를 식별하고,

상기 식별된 수직 경계 화소 중 적어도 일부를 수평 경계 화소로서 상기 연장된 수평 어레이의 연장부에 추가하고,

상기 타겟 블록의 예측 블록을 도출하기 위해, 상기 수직 경계 화소의 사용 없이, 상기 연장된 수평 어레이 내의 수평 경계 화소만을 사용하도록,

상기 프로세서에 의해 수행되는,

비디오 인코더.

청구항 4

비디오 디코더로서,

상기 비디오 디코더는, 컴퓨터 시스템의 프로세서와, 인트라 예측 앵글을 따라 보간된 타겟 블록의 경계 화소로 타겟 블록의 예측 블록을 도출하는 인트라 예측 동작을 수행하도록 상기 프로세서에 의해 실행 가능한 프로그램을 저장하는 메모리를 포함하고, 상기 경계 화소는 수평 경계 화소의 수평 어레이 및 수직 경계 화소의 수직 어레이를 포함하며, 상기 인트라 예측 동작은,

복수의 상이한 인트라 예측 앵글과 각각 관련되어 있는 인버스 앵글 파라미터의 값들을 열거한 룩업 테이블로부터, 인트라 예측 앵글에 대응되는 인버스 앵글 파라미터의 값을 획득하고,

상기 획득된 인버스 앵글 파라미터의 값과, 연장된 수평 어레이의 연장부에서의 위치를 나타내는 변수인 수평 위치 식별자의 값 간의 곱의 함수인 위치에서 수직 어레이 내에 위치한 수직 경계 화소 중 적어도 일부를 식별하고,

상기 식별된 수직 경계 화소 중 적어도 일부를 수평 경계 화소로서 상기 연장된 수평 어레이의 연장부에 추가하고,

상기 타겟 블록의 예측 블록을 도출하기 위해, 상기 수직 경계 화소의 사용 없이, 상기 연장된 수평 어레이 내의 수평 경계 화소만을 사용하도록,

상기 프로세서에 의해 수행되는,

비디오 디코더.

발명의 설명

기 술 분 야

[0001] 본 발명은 비디오 코딩에 관한 것으로, 특히 동일한 비디오 프레임에서 이전에 인코딩되고 재구축된 화소를 사용하여, 샘플 블록이 예측되는 인트라 프레임 예측에 관한 것이다.

배 경 기 술

[0002] 디지털 비디오는 미압축 방식으로 디지털 비디오 시퀀스의 각각 및 전체 프레임(예를 들면, 일련의 프레임)을 나타내기 위해 많은 양의 데이터가 필요하다. 대부분의 적용예에서 대역폭 제한 때문에 컴퓨터 네트워크를 통해 압축되지 않은 디지털 비디오를 전송하는 것이 가능한 것은 아니다. 또한, 미압축 디지털 비디오는 많은 양의 저장 공간이 필요하다. 상기 디지털 비디오는 일반적으로 몇몇 방식으로 인코딩되어서 상기 저장 요건을 줄이고

상기 대역폭 요건을 줄이게 된다.

[0003] 디지털 비디오를 인코딩하는 한 가지 기술은 인터-프레임 예측(inter- frame prediction), 또는 인터 예측(inter prediction)이다. 인터 예측은 상이한 프레임들 사이에 시간적 중복성(temporal redundancy)을 이용한다. 비디오의 시간적 인접 프레임들에는 일반적으로 실질적으로 동일하게 남아 있는 화소 블록이 포함된다. 상기 인코딩 과정에서, 모션 벡터가 한 프레임 내 화소 블록의 움직임을 다른 프레임 내 유사한 화소 블록에 연관시킨다. 따라서, 시스템은 상기 화소 블록들을 두번 인코딩할 필요가 없고, 오히려 화소 블록들을 한번 인코딩하고 다른 화소 블록들을 예측하는 모션 벡터를 제공한다.

[0004] 디지털 비디오를 인코딩하는 또 다른 기술은 인트라-프레임 예측(intra-frame prediction), 또는 인트라 예측(intra prediction)이다. 인트라 예측은 다른 프레임들의 화소를 참조하지 않고 프레임 또는 그 일부를 인코딩한다. 인트라 예측은 프레임 내의 화소 블록들 사이에 공간적 중복성을 이용한다. 공간적 인접 화소 블록들은 일반적으로 유사한 속성을 가지고 있기 때문에, 상기 코딩 과정의 효율성은 인접한 블록들 사이의 공간적 상관관계를 참조하여 향상된다. 이러한 상관 관계는 인접 블록들에 사용되는 예측 모드에 기초하여 타겟 블록의 예측에 의해 이용될 수도 있다.

발명의 내용

[0005] 본 발명은 비디오 코딩의 효율성을 향상시키는 독특한 인트라 예측 프로세스를 제공한다. H.264/AVC는 예측할 타겟 블록의 정확히 위에 위치한 수평 경계 내 참조 화소들과, 상기 타겟 블록의 정확히 왼쪽에 위치한 수직 경계 내 참조 화소들을 이용한다. 본 발명에서, 수평 경계 화소들의 어레이나 수직 경계 화소들의 어레이 중 적어도 일부가 검색된다. 그런 다음 상기 검색 화소들은 다른 경계 화소들에 추가되어서 그 어레이를 연장하게 된다. 인트라 예측은 단지 경계 화소들의 상기 연장된 어레이를 기반으로만 수행된다. 본 발명의 일 실시예에서, 상기 수직 경계 화소들의 적어도 일부가 검색되고, 상기 수평 경계 화소들에 추가되어서 그 어레이를 연장하게 된다.

[0006] 본 발명은 참조 화소들을 검색하도록 상기 수평 경계 또는 상기 수직 경계 중 하나를 선택하는 결정 과정을 없애준다. 본 발명은 또한 예측 방향으로 교차하는 상기 수직 경계의 위치를 계산하는 반복 과정을 없애주는데, 여기서 상기 반복 계산 과정에는 일반적으로 제산이 포함된다. 이러한 과정을 없애줌으로써, 상기 인트라 예측 과정이 단일 명령 다중 데이터 (SIMD) 아키텍처에서 구현될 수 있게 해서 비디오 코딩의 계산 효율성을 향상시킨다.

[0007] 본 발명에 따른 일 실시예에서, 수직 경계 화소들 중 적어도 일부는 $[\frac{size \times col}{angle}]$, 로 표현되는 수직 화소 식별자를 이용하여 검색되는데,

[0008] 여기서 *size*는 예측될 타겟 블록의 크기를 나타내고, *angle*은 예측 방향을 나타내고, *col*은 -1부터 상기 *angle*까지 1씩 감분되는 카운터이다. 상기 검색된 화소들은 상기 수직 화소 식별자 [*col*]에 의해 식별된 위치에서 상기 수평 경계 화소들에 추가된다.

[0009] 또 다른 실시 예에서, 수직 경계 화소들 중 적어도 일부를 검색하는 단계에 있어서, $\frac{N \times size}{angle}$ 로부터 *InvAngle*을 계산하는데,

[0010] 여기서 *N*은 2의 정수 멱이다. 그런 다음, [*col* x *InvAngle* >> $\log_2 N$] 로 표현된 수직 화소 식별자를 이용하여, 상기 수직 경계 화소들 중 상기 적어도 일부 화소들을 검색한다. 상기 검색된 화소들은 수평 화소 식별자 [*col*]에 의해 식별된 위치에서 상기 수평 화소들에 추가된다.

[0011] 또 다른 실시예에서, *InvAngle*는 *angle*의 값에 관한 *InvAngle*의 값들을 열거하는 룩업 테이블로부터 얻어진다.

[0012] 또 다른 실시 예에서, 화소는 수직 화소 식별자 [*row*]를 이용하여 상기 수직 경계 화소들 중에서 식별되는데, *row*는 0부터 *size*까지 1씩 증분되는 카운터이다. 상기 검색된 화소는 수평 화소 식별자 [*int* + 1]에 의해 식별된 위치에 있는 수평 경계 화소들에 추가되는데, 여기서 *int*는 예측 방향과 교차하는 화소의 위치의 정수 표현이다.

[0013] 본 발명은 또한 수평 경계 화소들의 어레이나 수직 경계 화소들의 어레이 중 적어도 일부가 검색되는 인트라 예측 동작을 구현하는 인코더와 디코더를 제공한다. 그런 다음 상기 검색된 화소들은 다른 경계 화소들에 추가되

어서 그 어레이를 연장한다. 인트라 예측은 단지 경계 화소들의 연장된 어레이를 기반으로만 수행된다.

도면의 간단한 설명

[0014] 도 1은 본 발명을 구현할 수 있는 예시적인 하드웨어 아키텍처를 보여주는 블록도이다.

도 2는 본 발명을 적용할 수 있는 비디오 인코더의 일반적인 도시를 보여주는 블록도이다.

도 3은 본 발명을 적용할 수 있는 비디오 디코더의 일반적인 도시를 보여주는 블록도이다.

도 4는 본 발명의 일 실시예에 따른 인코더의 기능 모듈들을 보여주는 블록도이다.

도 5는 본 발명의 실시예의 인트라 예측 모듈에 의해 수행되는 인트라 예측 프로세스를 나타내는 흐름도이다.

도 6은 본 발명의 일 실시예에 따른 디코더의 기능 모듈들을 보여주는 블록도이다.

도 7은 H.264/AVC에서 지원하는 Intra_4x4 예측 모드를 설명하는 예측 방향들을 보여주는 개략도이다.

도 8은 문서 번호 JCT-VC A119에서 제안된 예측 방향들을 보여주는 개략도이다.

도 9는, 도 7에 도시한 예측 방향들 중 하나를 따라 예측 블록을 생성하는, JCT-VC A119에서 제안한 프로세스를 도시하는 흐름도이다.

도 10은, 본 발명의 일 실시예에 따라 수행된 저 복잡성 인트라 예측의 프로세스를 도시하는 흐름도이다.

도 11a은 수평 및 수직 경계 화소의 예측 블록과 어레이를 보여주는 개략도이다.

도 11b는 수직 경계 화소로 연장된 수평 경계 화소의 어레이를 보여주는 개략도이다.

도 12는 본 발명의 일 실시예에 따라 수행한 수평 경계 화소의 어레이를 연장하는 과정을 보여주는 흐름도이다.

도 13은 수평 경계 화소의 어레이를 연장하는 또 다른 실시예를 보여주는 흐름도이다.

도 14는 본 발명의 또 다른 실시예에 따라 수행한 저 복잡성 인트라 예측의 과정을 보여주는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0015] 도 1은 본 발명을 구현할 수 있는 컴퓨터(100)의 예시적인 하드웨어 아키텍처를 도시한다. 도 1에 도시한 하드웨어 아키텍처는 본 발명의 실시예들을 구현하는 비디오 인코더 및 비디오 디코더 양측에 공통일 수 있는 하드웨어임을 주목한다. 컴퓨터(100)는, 프로세서(101), 메모리(102), 저장 장치(105), 및 로컬 인터페이스(107)를 통해 통신 가능하게 연결된 하나 이상의 입력/출력(I/O) 장치(106)(또는 주변 장치)를 포함한다. 상기 로컬 인터페이스(105)는, 예를 들어, 당해 기술에 알려져 있는 바와 같은 하나 이상의 버스 또는 기타 유선이나 무선 접속부일 수 있지만, 이러한 예로 한정되지는 않는다.

[0016] 상기 프로세서(101)는, 소프트웨어를, 특히 메모리(102)에 저장된 소프트웨어를 실행하기 위한 하드웨어 장치이다. 상기 프로세서(101)는, 임의의 맞춤형 또는 상용 프로세서, 중앙 처리 유닛(CPU), 상기 컴퓨터(100)에 연관된 여러 프로세서들 중 보조 프로세서, (마이크로칩 또는 칩 세트의 형태로 된) 반도체 기판 마이크로프로세서, 또는 소프트웨어 명령을 실행하기 위한 일반적인 임의의 장치일 수 있다.

[0017] 상기 메모리(102)는, 휘발성 메모리 소자들(예를 들어, DRAM, SRAM, SDRAM 등의 랜덤 액세스 메모리(RAM)) 및 비휘발성 메모리 소자들(예를 들어, ROM, 하드 드라이브, 테이프, CDROM 등) 중 임의의 하나 또는 조합을 포함할 수 있는 컴퓨터 판독가능 매체를 포함한다. 또한, 상기 메모리(102)는, 전자 매체, 자기 매체, 광학 매체, 및/또는 다른 유형의 저장 매체를 포함할 수 있다. 컴퓨터 판독가능 매체는, 명령 실행 시스템, 기기, 또는 장치에 의해 또는 명령 실행 시스템, 기기, 또는 장치와 함께 사용하도록 이러한 프로그램을 저장, 통신, 전파, 또는 전송할 수 있는 임의의 수단일 수 있다. 상기 메모리(102)는, 다양한 부품들이 서로 원격으로 위치하지만 상기 프로세서(101)에 의해 액세스될 수 있는 분산형 아키텍처를 가질 수 있다는 점에 주목한다.

[0018] 상기 메모리(102) 내의 소프트웨어(103)는 하나 이상의 개별적인 프로그램을 포함할 수 있고, 이러한 프로그램 각각은 후술하는 바와 같이 상기 컴퓨터(100)의 논리적 기능들을 구현하기 위한 실행가능 명령들의 순서화된 리스트를 포함한다. 도 1의 예에서, 상기 메모리(102) 내의 소프트웨어(103)는 본 발명에 따라 상기 컴퓨터(100)의 비디오 인코딩 또는 비디오 디코딩 기능을 형성한다. 또한, 요구되는 것은 아니지만, 상기 메모리(102)가 운영 체제(O/S; 104)를 포함할 수 있다. 상기 운영 체제(104)는, 본질적으로 컴퓨터 프로그램의 실행을 제어하

며, 스케줄링, 입력-출력 제어, 파일 및 데이터 관리, 메모리 관리, 및 통신 제어와 관련 서비스를 제공한다.

[0019] 상기 컴퓨터(100)의 저장 장치(105)는, 고정형 저장 장치나 휴대용 저장 장치를 포함한 서로 다른 많은 유형의 저장 장치 중 하나일 수 있다. 일례로, 상기 저장 장치(105)는, 자기 테이프, 디스크, 플래시 메모리, 휘발성 메모리, 또는 다른 저장 장치일 수 있다. 또한, 상기 저장 장치(105)는 보안 디지털 메모리 카드 또는 다른 임의의 탈착가능 저장 장치(105)일 수 있다.

[0020] 상기 I/O 장치(106)는, 터치 스크린, 키보드, 마우스, 스캐너, 마이크로폰 등의 입력 장치 또는 다른 입력 장치를 포함할 수 있지만, 이러한 예들로 한정되지는 않는다. 또한, 상기 I/O 장치(106)는 또한 디스플레이 등의 출력 장치 또는 다른 출력 장치를 포함할 수 있지만, 이러한 예들로 한정되지 않는다. 상기 I/O 장치(106)는, 입력과 출력을 통해, 예를 들어, 변조기/복조기(다른 장치, 시스템, 또는 네트워크에 액세스하기 위한 모뎀), 무선 주파수(RF), 무선 또는 다른 송수신기, 전화 인터페이스, 브리지, 라우터를 통해 통신하는 장치 또는 입력과 출력 모두로서 기능하는 다른 장치를 더 포함할 수 있지만, 이러한 예들로 한정되지는 않는다.

[0021] 당업자에게는 주지된 것처럼, 비디오 압축은 비디오 시퀀스에서 중복 정보를 제거함으로써 달성된다. 서로 다른 많은 비디오 코딩 표준들이 존재하며, 예를 들어, MPEG-1, MPEG-2, MPEG-4, H.261, H.263, 및 H.264/AVC가 있다. 본 발명을 임의의 특정한 비디오 코딩 표준의 애플리케이션으로 한정하려는 것이 아님에 주목한다. 그러나, 본 발명의 이하의 설명에서는, 본 명세서에 참고로 원용되는 H.264/AVC 표준을 이용하는 예를 이용한다. H.264/AVC는 최신 비디오 코딩 표준이며, MPEG-1, MPEG-2, H.261, H.263 등의 이전의 코딩 표준들에 비해 성능을 상당히 개선한다.

[0022] H.264/AVC에서는, 비디오의 각 프레임 또는 픽처는 여러 슬라이스들로 나누어질 수 있다. 이어서, 슬라이스들은 매크로블록들이라 칭하는 16 x 16 화소들의 블록들로 나누어질 수 있으며, 이러한 블록들은 다시 8 x 16, 16 x 8, 8 x 8, 4 x 8, 8 x 4, 4 x 4 화소까지 나누어질 수 있다. H.264/AVC에 의해 지원되는 슬라이스에는 5개 종류가 존재한다. I 슬라이스에서는, 모든 매크로블록들이 인트라 예측을 이용하여 코딩된다. P 슬라이스에서는, 매크로블록들이 인트라 또는 인터 예측을 이용하여 코딩될 수 있다. P 슬라이스들에서는, 매크로블록당 하나의 모션 보상 예측(MCP) 신호만이 사용될 수 있다. B 슬라이스에서는, 매크로블록들이 인트라 또는 인터 예측을 이용하여 코딩될 수 있다. 예측당 두 개의 MCP 신호를 이용할 수 있다. SP 슬라이스에서는, P 슬라이스가 서로 다른 비디오 스트림들 간에 효율적으로 스위칭될 수 있다. SI 슬라이스는 인트라 예측만을 이용하며 랜덤 액세스 또는 오류 복구를 위한 SP 슬라이스에 대한 정확한 매치이다.

[0023] 도 2는 본 발명을 적용할 수 있는 비디오 인코더의 일반적인 도면이다. 이 도에 도시한 블록들은 상기 메모리(102) 내의 소프트웨어(103)를 실행하는 프로세서(101)에 의해 실현되는 기능 모듈들을 나타낸다. 비디오 프레임의 픽처(200)가 비디오 인코더(201)에 공급된다. 비디오 인코더는 피처(200)를 매크로블록(200A) 단위로 처리한다. 각 매크로블록은 픽처(200)의 여러 개의 화소들을 포함한다. 각 매크로블록에서는, 변환 계수로의 변환이 수행된 후, 변환 계수 레벨로의 양자화가 수행된다. 또한, 화소 데이터에 대하여 코딩 단계들을 직접 수행하는 것이 아니라 예측된 화소 값들에 대한 화소 데이터의 차에 대하여 코딩 단계들을 수행하도록, 인트라 예측 또는 인터 예측을 이용한다.

[0024] 각 슬라이스마다, 상기 인코더(201)는 각 슬라이스의 매크로블록들의 코딩된 버전을 형성하는 다수의 신택스 요소를 생성한다. 변환 계수 레벨 또는 스킵된 변환 계수 레벨을 나타내는 유의 맵(significance map) 등의 변환 계수의 코딩에 관련된 신택스 요소들의 모든 잔여 데이터 요소를 잔여 데이터 신택스 요소(residual data syntax element)라 칭한다. 이러한 잔여 데이터 신택스 요소들 외에도, 상기 인코더(201)에 의해 생성된 신택스 요소들은, 각 매크로블록이 각각 인코딩된 방식 및 디코딩되어야 하는 방식에 관한 제어 정보를 포함한 제어 정보 신택스 요소들을 포함한다. 다시 말하면, 신택스 요소들은 두 개의 카테고리로 분류될 수 있다. 제1 카테고리인 제어 정보 신택스 요소들은, 예를 들어, 슬라이스 기반 및 매크로블록 기반 제어 정보뿐만 아니라, 매크로블록 유형, 서브매크로블록 유형, 및 공간 유형과 시간 유형의 예측 모드에 관한 정보에 관련된 요소들도 포함한다. 제2 카테고리에서는, 양자화 단계들에 대응하는 레벨 단위로 표시되는 양자화 계수들의 값들 및 양자화 변환 계수들의 블록 내의 모든 유의 계수들의 위치를 나타내는 유의 맵 등의 모든 잔여 데이터 요소들은 결합되어 잔여 데이터 신택스 요소들로 된다.

[0025] 상기 인코더(201)는, 각 슬라이스마다 신택스 요소들을 인코딩하며 산술 코드워드를 생성하는 엔트로피 코더를 포함한다. 슬라이스에 대한 산술 코드를 생성하는 경우, 엔트로피 코더는 비디오 신호 비트 스트림의 신택스 요소들의 데이터 값들 중 통계 의존성을 이용한다. 상기 인코더(201)는 픽처(200)의 슬라이스에 대하여 인코딩된 신호를 도 3에 도시한 비디오 디코더(301)에 출력한다.

[0026] 도 3은 본 발명을 적용할 수 있는 비디오 디코더의 일반적인 도면이다. 마찬가지로, 이 도에 도시한 블록들은 상기 메모리(102) 내의 소프트웨어(103)를 실행하는 프로세서(101)에 의해 실현되는 기능 모듈들을 나타낸다. 상기 비디오 디코더(301)는 인코딩된 비디오 신호를 수신하고, 우선, 그 신호를 다시 신택스 요소들로 엔트로피-디코딩한다. 상기 디코더(301)는, 매크로블록마다 이어서 슬라이스마다 상기 픽처(300)의 화소들의 픽처 샘플들(300A)을 재구축하기 위해, 신택스 요소들을 사용한다.

[0027] 도 4는 비디오 인코더(201)의 기능 모듈들을 도시한다. 이러한 기능 모듈들은, 상기 메모리(102) 내의 소프트웨어(103)를 실행하는 프로세서(101)에 의해 실현된다. 입력 비디오 픽처는, 크로미넌스("크로마") 및 휘도("루마") 등의 초기 색들의 성분들(예를 들어, 색조, 포화, 및 값 등의 다른 성분들도 가능함)을 나타내는 샘플 포인트들에 의해 형성된 자연스러운 (미압축) 비디오 이미지의 필드 또는 프레임이다. 입력 비디오 픽처는, 픽처 색의 루마 성분의 16 x 16 화소로 이루어지는 정사각형 픽처 영역을 각각 나타내는 매크로블록들(400)로 나누어진다. 입력 비디오 픽처는, 또한, 픽처 색의 두 개의 크로마 성분의 8 x 8화소들을 각각 나타내는 매크로블록들로 파티션화된다. 일반적인 인코더 동작에서, 입력된 매크로블록들은 인터 또는 인트라 예측을 이용하여 시간적으로 또는 공간적으로 예측된다. 그러나, 설명을 위해, 상기 매크로블록들(400)은 모두 I 슬라이스형 매크로블록이며 인트라 예측만 적용된다고 가정한다.

[0028] 인트라 예측은 인트라 예측 모듈(401)에서 달성되며, 그 동작은 이하에서 상세히 설명한다. 상기 인트라 예측 모듈(401)은, 프레임 메모리(403)에서 이전에 인코딩되고, 재구축되고, 저장된 이웃하는 블록들의 수평 및 수직 경계 화소들로부터 예측 블록(402)을 생성한다. 타겟 블록(400)과 상기 예측 블록(402) 사이의 차인 예측 블록(402)의 잔차(404)는, 비디오 코딩 분야의 당업자에게 알려져 있는 방법 및 기술을 이용하여 변환/양자화 모듈(405)에서 변환되고, 스케일링되고, 양자화된다. 이어서, 양자화된 변환 계수(406)는 엔트로피 코딩 모듈(407)에서 엔트로피-코딩되고 인코딩된 비디오 신호(408)로서 (인트라 예측에 관한 다른 정보와 함께) 전송된다.

[0029] 상기 비디오 인코더(201)는 타겟 블록에 대한 인트라 예측을 수행하기 위한 디코딩 기능을 포함한다. 디코딩 기능은, 상기 예측 블록(402)에 더해지는 디코딩된 예측 잔차(410)를 생성하도록 양자화된 변환 계수(406)에 대하여 역 양자화 및 역 변환을 수행하는 역 양자화/변환 모듈(409)을 포함한다. 상기 디코딩된 예측 잔차(410)와 예측 블록(402)의 합이 재구축된 블록(411)이며, 이 합은 상기 프레임 메모리(403)에 저장되며, 이 메모리로부터 판독되며, 인트라 예측 모듈(401)에 의해 사용되어 다음 타겟 블록(400)의 디코딩을 위한 예측 블록(402)이 생성된다.

[0030] 도 5는 상기 인트라 예측 모듈(401)에 의해 수행되는 프로세스를 나타내는 흐름도이다. H.264/AVC 표준에 따르면, 인트라 예측은, 이전에 인코딩되고 재구축된 이웃하는 블록들의 경계 화소들("참조 화소")의 보간을 이용하여 복수의 예측 모드에서의 타겟 블록(400)의 각 화소를 예측하는 것을 포함한다. 예측 모드들은 양의 정수 0, 1, 2,…에 의해 식별되며, 각각은 타겟 블록(400) 내의 특정 화소를 예측하기 위한 다른 명령 또는 알고리즘에 연관된다. 상기 인트라 예측 모듈(401)은 각 예측 모드에서 인트라 예측을 수행하고, 서로 다른 예측 블록을 생성한다. 전체 검색(full search; FS) 알고리즘 하에서는, 생성된 예측 블록들의 각각을 타겟 블록(400)과 비교하여, 예측 모드들 중에서 예측 잔차(404)를 최소화하거나 더욱 작은 예측 잔차(404)를 생성하는 최적의 예측 모드를 찾는다. 최적의 예측 모드의 식별은 압축되어 제어 정보 신택스 요소들과 함께 상기 디코더(301)에 전송된다.

[0031] 말로 설명한 바와 같이 예측의 일반적인 설명에 의해 각 예측 모드를 설명할 수 있다(즉, 수평으로 위, 수직 및 대각선 아래 왼쪽). 예측 방향은, 도 7에 도시한 바와 같이 화살표로 도면을 통해 표현되는 각도 방향에 의해 그래픽으로 설명될 수 있다. 이러한 유형의 도면에서, 각 화살표는 예측 방향 또는 예측 모드를 나타내는 것으로 고려될 수 있다. 예측 모드에 대응하는 각도는, 타겟 화소를 예측하는 데 사용되는 참조 화소들의 가중된 평균 위치로부터 타겟 화소 위치로의 방향에 대하여 일반적인 관계를 갖는다. 예측 모드들은, 어떠한 예측 방향과도 연관되지 않으며 이에 따라 다른 예측 모드들과는 달리 도면에서 그래픽으로 설명될 수 없는 DC 예측 모드를 포함한다는 점에 주목한다. DC 예측 모드에서, 상기 예측 블록(402)은, 예측 블록(402)의 각 화소가 참조 화소들의 평균 값으로 균일하게 설정되도록, 생성된다.

[0032] 다시 도 5를 참조해 보면, 상기 예측 모드는 단계 501에서 개시된다. 단계 502에서, 상기 예측 모드가 DC 예측을 가리키는지 여부를 결정한다. DC 예측을 가리킨다면, 흐름은 단계 503으로 진행하여, 단계 503에서 참조 화소들의 평균 값을 갖는 DC 예측 블록(402)이 생성된다. 예측 모드가 다른 것을 가리킨다면, 이하에서 후술될 단계 504에서 예측 모드에 연관된 알고리즘 또는 명령에 따라 예측 블록(402)이 생성된다. 단계 503 또는 단계 504 후에, 흐름은 단계 505로 진행하여, 모든 예측 모드들에 대하여 예측 블록들이 생성되는지 여부를

결정한다. 인트라 예측이 모든 예측 모드들 하에서 실행되면, 흐름은 단계 506으로 진행한다. 인트라 예측이 모든 예측 모드들 하에서 실행되지 않는다면, 예측 모드는 단계 507에서 증분되고, 흐름은 단계 502로 되돌아간다. 단계 506에서는, 생성된 예측 블록들의 각각을 타겟 블록(400)과 비교하여, 예측 잔차(404)를 최소화하는 최적 예측 모드를 결정한다.

[0033] 도 6은 상기 비디오 디코더(301)의 기능 모듈들을 도시한다. 이러한 기능 모듈들은 상기 메모리(102) 내의 소프트웨어(103)를 실행하는 프로세서(101)에 의해 실현된다. 인코더(201)로부터의 인코딩된 비디오 신호는 우선 엔트로피 디코더(600)에 의해 수신되고 양자화된 변환 계수(601)로 다시 엔트로피-디코딩된다. 양자화된 변환 계수(601)는 역 양자화/변환 모듈(602)에 의해 역 양자화 및 변환되어 예측 잔차(603)를 생성하게 된다. 인트라 예측 모듈(604)은 인코더(201)에 의해 선택된 예측 모드를 통지받는다. 선택된 예측 모드에 따라, 인트라 예측 모듈(604)은 도 5의 단계 502, 503, 504에서 수행된 인트라 예측 프로세스와 유사한 인트라 예측 프로세스를 수행하여, 이전에 재구축되어 프레임 메모리(606)에 저장된 이웃하는 블록들의 경계 화소들을 이용하여 예측 블록(605)을 생성한다. 예측 블록(605)은 예측 잔차(603)에 가산되어 디코딩된 비디오 신호의 블록(607)을 재구축한다. 재구축된 블록(607)은 다음 블록의 예측에 사용되도록 프레임 메모리(606) 내에 저장된다.

[0034] 이하에서는, DC예측 모드를 제외하고, 예측 모드들 중 하나에서 예측 블록을 생성하도록 인트라 예측 모듈들(401, 604)에 의해 수행되는 단계 504의 프로세스를 상세히 설명한다. H.264/AVC는 인트라_4x4 예측, 인트라_8x8 예측, 및 인트라_16x16 예측을 지원한다. 인트라_4x4 예측은 픽처에 상당한 상세가 있는 경우 흔히 사용된다. 인트라_4x4 예측은 하나의 매크로블록 내의 16개의 4x4 루마 블록을 개별적으로 예측한다. 인트라_4x4 예측은 하나의 DC 예측 모드를 포함한 9개의 예측 모드에서 수행된다. 인트라_4x4 예측이 수행되는 공간 예측 방향은 도 7에 도시되어 있다. 인트라_8x8 예측은 하나의 DC 예측 모드를 포함한 9개의 예측 모드에서 수행된다. 인트라_16x16 예측은 하나의 DC 예측 모드를 포함한 4개의 예측 모드에서 수행된다.

[0035] 최근의 연구에서는, 예측 방향의 개수 또는 예측 모드의 개수를 증가시키면, 일반적으로 비디오 코딩에서의 압축 효율이 개선되는 것으로 알려져 있다. 예를 들어, 본 명세서에 참고로 원용되는, Joint Collaborative Team on Video Coding (JCT-VC)에 제출된 문헌 Nos. JCT-VC A119 ("Angular Intra Prediction") 및 JCT-VC A124 ("Arbitrary Direction Intra")를 참조한다. 예측 방향의 개수를 증가시킴으로써, 이용가능한 예측 방향들의 각도 간격의 개수가 증가하고, 이에 따라 예측 블록 후보들의 개수가 증가한다. 예측 블록 후보들의 개수가 증가함으로써, 인코딩된 타겟 블록과 거의 동일한 예측 블록을 가질 가능성이 간단히 증가한다. 도 8은 문헌 JCT-VC A119에서 제안된 예측 방향을 도시한다. 도 8에서, 참조 화소들은 17개의 수평 화소 및 17개의 수직 화소로 이루어지며, 여기서 상좌(upper left) 화소는 수평 및 수직 경계들 모두에 대하여 공통된다. 따라서, 33개의 서로 다른 예측 방향을 이용하여 8 x 8 블록의 예측 화소들을 생성할 수 있다. JCT-VC A124는, 예측될 블록의 크기에 따라 예측 방향들의 개수가 조절되는 임의의 방향 인트라 예측을 제안한다.

[0036] 도 9는, 도 8에 도시한 예측 방향들 중 하나를 따라 예측 블록을 생성하는 JCT-VC A119에서 제안한 프로세스를 도시하는 흐름도이다. 프로세스의 이하의 설명에서는, 설명의 편의상 일부 알고리즘을 간략화한다. 또한, 설명한 프로세스는 대체로 수직인 예측 방향을 따른 인트라 예측으로 한정된다. 주로 수평인 예측 방향을 따른 인트라 예측은, JCT-VC A119에 의해 제공되는 소프트웨어에서 실증되듯이 도 9에 도시한 프로세스에 대하여 대칭적으로 구현될 수 있다. 도 8은 예측될 8 x 8 블록을 도시하고 있지만, 도 9에 도시한 프로세스는 서로 다른 구성의 화소들의 다양한 개수들에 적용되도록 확장가능하다. 예를 들어, 예측될 블록은 화소들의 4 x 4 어레이를 포함할 수 있다. 예측 블록은, 또한, 화소들의 8 x 8 어레이, 화소들의 16 x 16 어레이, 또는 더 큰 화소들의 어레이를 포함할 수 있다. 정사각형 어레이와 직사각형 어레이를 포함하는 다른 화소 구성도 예측 블록을 형성할 수 있다.

[0037] 도 9의 단계 900에서, 타겟 블록의 바로 위와 왼쪽에 있는 수평 경계와 수직 경계의 참조 화소들은, 이전에 인코딩되고 재구축되고 도 4에 도시한 프레임(403) 등의 프레임 메모리에 저장된 각각 이웃하는 블록들로부터 판독된다. 수평 경계로부터의 화소들은 "*refH*"라 칭하는 메모리 영역에 저장된다. 수직 경계로부터의 화소들은 "*refV*"라 칭하는 다른 메모리 영역에 저장된다. 도 8로 돌아가서, 참조 화소들은, 8 x 8 블록의 상좌 화소 위치를 원점으로서 갖는 좌표계의 해당 좌표들에 의해 식별된다. 따라서, 수평 경계 화소들은 p[x, y]에 의해 표현되는 좌표를 갖고, 여기서 x=0, 1,…16 및 y=0이다. 수직 경계 화소들은 p[x, y]에 의해 표현되는 좌표를 갖고, 여기서 x=0, y=0, -1, -2,…-16이다.

[0038] 메모리 영역 *refH*에 저장된 수평 경계 화소들은 x=0, 1,…16인 논리 어드레스(x)에 의해 식별되고 수직 메모리 영역 *refV*에 저장된 경계 화소들도 마찬가지로 y=0, -1 , -2...-16인 논리 어드레스(y)에 의해 식별되고 각 화

소가 판독되는 좌표의 수를 어드레스에 저장된다고 가정한다. 따라서, 수평 화소 및 수직 화소가 도 8에 그래픽으로 표현되어 있듯이, 메모리 영역들 *refH* 및 *refV*는, 선형으로 서로 직교 연장되며 각각의 영역은 2 x *size* + 1의 길이를 갖고 여기서 "*size* 는 타겟 블록의 크기를 나타내는 파라미터라고 고려될 수 있다. *size* 는 4, 8, 16… 등의 2의 정수 멱과 같은 값을 갖는다고 가정한다. 선택 사항으로, H.264/AVC 의 섹션 8.3.2.2.1에서 설명한 바와 같은 로우 패스 필터가 *refH* 및 *refV*의 화소들에 적용될 수 있다.

[0039] 단계 901에서, "*row*"라 칭하는 카운터가 제로("0")로 설정된다. 카운터 *row* 는 0 내지 *size*의 값을 취하며, 예측 블록의 예측 화소의 행 위치를 가리킨다. 단계 902에서, "*pos*"라 칭하는 파라미터는 *angle* x (*row* +1)에 의해 계산된다. *angle* 은 고정 소수점 방식으로 된 분수를 갖는 파라미터이다. 이처럼, *angle* 은 정수부 및 분수부와 함께 형성된다. 분수부는 고정된 개수의 이진 디지트로 이루어지며, 도 8에 도시한 예측 방향들 중 하나를 나타낸다. 예를 들어, "*angle* = -*size*"는 도 8의 좌표 [x=0, y=0]를 통과하는 예측 방향을 식별한다. 양의 값을 갖는 *angle* 은 수평 경계만을 교체하는 예측 방향을 식별하는 반면, 음의 값을 갖는 *angle* 은 수평 및 수직 경계 모두를 교차하는 예측 방향을 식별하며, *angle* 은 사용될 예측 방향들의 개수에 의해 결정되는 범위 내에서 가변된다. JCT-VC A 124에서 제안되어 있듯이, 사용될 예측 방향들의 개수는 예측될 블록의 크기에 따라 결정될 수 있다. 이하의 설명에서는, 각도가 "-*size*" 내지 "*size*" 범위 내에서 가변되는 분수를 취한다고 가정한다. 각도의 범위 한계는 다른 값들과 함께 정의될 수 있다는 점에 주목한다.

[0040] *angle* 처럼, 파라미터 *pos*는 정수부와 분수부로 이루어지고, 이 분수부는 각도 범위 한계의 2를 밑으로 하는 로그와 동일한 고정된 개수의 이진 디지트로 이루어지며, 이는 각도의 범위 한계가 제로로 설정된다는 위 가정에 따라 *log2_ size* 로 표현될 수 있으며, *pos*는 *angle* 에 의해 표현되는 예측 방향과 수평 경계 간의 교차점을 식별한다. 단계 902로 되돌아가서, "*pos*>> *log2_ size* " 연산은 pos의 정수부를 식별하고, 이는 파라미터 "*int*"에 저장되며, "*pos*& (*size* - 1)" 연산은 *pos*의 분수부를 식별하는, 이는 파라미터 "*frac*"에 저장된다. 연산자 ">>"는 이진 디지트의 산술 우측 시프트를 필요로 한다. 연산자 "&" 는 비트 단위 "and" 연산을 필요로 한다.

[0041] 단계 903에서는, *angle* 이 제로("0") 이상의 값을 갖는지 여부를 결정한다. *angle* 이 제로 이상의 값을 가지면, 흐름은 단계 904로 진행한다. 그렇지 않다면, 흐름은 단계 913으로 진행하고, 제로 이상의 *angle* 은 예측 블록의 예측 화소들을 도출하도록 수평 경계에 위치하거나 *refH*에 저장된 참조 화소들만을 의존할 수 있음을 나타낸다. 반면에, 제로 미만의 *angle* 은, 예측 블록의 예측 화소들을 도출하는 데 수직 경계에 위치하거나 *refV*에 저장된 참조 화소들이 필요함을 나타낸다.

[0042] 단계 904에서는, *frac* 이 제로가 아닌지 여부를 결정한다. *frac* 이 제로가 아니면, 흐름은 단계 905로 진행한다. *frac* 이 제로이면, 흐름은 단계 906으로 진행한다. 제로와 같은 *frac* 은, 예측 블록의 예측 화소가 수평 경계의 참조 화소로부터 직접 복사될 수 있음을 나타낸다. 넌제로 분수는, 예측 방향이 비정수 위치에서 수평 경계와 교차하고, 예측 블록의 예측 화소를 도출하는 데 하나의 참조 화소보다 많은 참조 화소들의 보간이 필요하다는 점을 나타낸다.

[0043] 단계 905에서는, "*col*"이라 칭하는 카운터가 제로("0")로 설정된다. 카운터 col을 이용하여 *refH*의 참조 화소를 어드레싱한다. 단계 907에서는, "*int* + *col* + 1"와 "*int* + *col* + 2"인 두 개의 참조 화소를 *refH*로부터 검색한다. 이러한 두 개의 참조 화소는 *frac* 으로 가중 평균화되거나 보간되어 예측 화소 *v*를 도출한다. 구체적으로, " *int* + *col* +1" 에 의해 식별되는 *refH*내의 참조 화소는 "*size* - *frac*"으로 승산되어 파라미터 *a*에 저장된다. " *int* + *col* + 2"에 의해 식별되는 *refH*내의 참조 화소는 "*frac*"으로 승산되어 파라미터 *b*에 저장된다. 이어서, 파라미터들 *a*와 *b*는 더해지고 *size* 에 의해 나누어진다, 즉, (*size* - *frac*) + *frac* 이다. *size* 에 의한 제산은 *log2_ size* 에 의해 우측 시프트로 교체될 수 있다. 도출된 예측 화소 *v*는, 특정한 예측 방향으로 타겟 블록을 위한 예측 블록을 나타내는 "*pred*"라 칭하는 메모리 영역들의 어레이에 저장된다. *pred* 내의 각 메모리 영역은 파라미터 *row*와 *col*에 의해 식별된다. 이어서, *col* 은 단계 908에서 1씩 증분되고 단계 909에서 *size* 와 비교된다. *col* 이 *size* 보다 작은 한, 단계 907과 908이 반복된다. *col* 이 *size* 보다 커지면, 흐름은 단계 920으로 진행한다.

[0044] 단계 904에서 *frac* 이 제로로 결정되면, 카운터 *col* 은 단계 906에서 제로로 설정된다. 단계 910에서, 예측 화소 *v*는 *refH* (*int* + *col* + 1)로부터 직접 복사된 후, *pred* 내의 대응하는 메모리 영역에 저장된다. 이어서, *col*은 단계 911에서 1씩 증분되고 단계 912에서 *size*와 비교된다. *col* 이 *size* 보다 작기만 하면, 단계 910과 911이 반복된다. *col* 이 *size* 와 동일해지면, 흐름은 단계 920으로 진행한다.

[0045] 단계 903으로 돌아가서, 제로보다 작은 *angle* 은 *refV*로부터의 참조 화소들이 예측 블록의 예측 화소들을 도출

하는 것을 필요로 한다. 카운터 *col* 은 단계 913에서 제로로 설정된다. 이어서, 단계 914에서는, "*int* + *col* + 1"이 제로 미만인지를 결정하고, 제로 이상의 "*int* + *col* + 1"은, 예측 블록의 예측 화소들을 도출하도록 *refH*에 저장된 참조 화소들만이 여전히 의존할 수 있음을 나타내며, 흐름은 단계 915로 진행한다. 단계 915에서 수행되는 프로세스는 단계 907의 프로세스와 유사하며, 이에 따라 여기서는 그 설명을 반복하지 않으며, *col* 을 단계 916에서 1씩 증분하여 단계 917의 *size* 와 비교한다. *col* 이 *size* 보다 작은 한, 단계 914, 915, 916이 반복된다. *col*이 *size* 와 같아지면, 흐름은 단계 920으로 진행한다.

[0046] 단계 914에서 "*int* + *col* + 1"이 제로보다 작다고 결정되면, *refV*에 저장된 참조 화소들은 예측 블록의 예측 화소들을 도출하는 데 필요하다. 단계 918에서는, 예측 방향과 수직 경계 간의 교차점을 우선 결정한다. 단계 918에서, 위치는 *pos2*로 표현된다. 단계 902에서, *pos*, 즉, 수평 경계와 예측 방향 간의 교차점의 위치가 "*angle* x {*row* + 1)"에 의해 결정된다는 점에 주목한다. *angle* 이 수평 및 수직 차들의 비율을 나타내는 경우에, "*angle* (*row* + 1) 대신에 " $angle^{-1}$ x (*col* + 1)"을 계산하여, 수직 경계와 예측 방향 간의 교차점의 위치를 결정한다. 위에서 가정한 바와 같이, *angle* 은 - *size* 내지 *size* 의 범위(-*size* ≤ *angle* ≤ *size*) 내에 있다. 따라서, *angle* 과 *size* 간의 비율은 아래와 같이 정의된다.

[0047] $$\alpha = \frac{angle}{size} \ (-1 \le \alpha \le 1)$$

[0048] 이어서, $angle^{-1}$ 은 아래와 같이 정의된다.

[0049] $$angle^{-1} = \frac{size}{\alpha} \text{ or } \frac{size^2}{angle}$$

[0050] 이처럼, *pos2*는, 단계 918에서 *col* +1에 의해 승산된 후 *size* 의 제곱으로 결정된 후 아래와 같이 *angle* 의 절대값에 의해 제산된다.

[0051] $$pos2 = \frac{size^2 \times (col+1)}{|angle|}$$

[0052] *pos*처럼, *pos2*는 정수부와 분수부로 형성되는 고정 소수점 방식으로 된 분수부를 갖는다. 정수부는 *log2_size* 에 의해 결정되는 이진 디지트의 수로 이루어진다. *pos2*의 정수부는 파라미터 *int2*에 저장되고, *pos2*의 분수부는 파라미터 *frac2*에 저장된다. 단계 919에서는, "*int2* + *row* + 1"과 " *int2* + *row* + 2"에 의해 식별되는 두 개의 참조 화소가 *refV*로부터 검색된다. 이러한 두 개의 참조 화소는 *frac2*로 가중 평균화되거나 보간되어 예측 화소 *v*를 도출하게 된다. 구체적으로, *refV*(*int2* + *row* + 1)로부터의 참조 화소는 "*size* - *frac2*"로 승산되고 파라미터 *a*에 저장된다. *refV*(*int2* + *row* + 2)로부터의 참조 화소는 "frac2"로 승산되고 파라미터 *b*에 저장된다. 이어서, 파라미터 *a*와 *b*를 더하고 *size* 에 의해 나누거나 *log2_size*에 의해 우측 시프트한다. 도출된 예측 화소 *v*는 대응하는 메모리 영역 *pred*에 저장된다. 단계 914, 918, 919, 916은 단계 917에서 *col*이 *size* 와 같아질 때까지 반복된다.

[0053] 단계 920에서는, *row* 를 1씩 증분한다. 이어서, 단계 921에서는 *row* 가 *size* 보다 작은지를 결정한다. *row* 가 *size* 보다 작은 한, 단계 902부터의 단계들이 반복되어 예측 블록의 예측 화소를 도출한다. 흐름은 단계 921에서 *row* 가 *size* 와 같아지면 종료된다.

[0054] 전술한 바와 같이, 예측 블록 후보들의 개수가 증가하면 코딩 효율이 개선되는 반면, 예측 블록 후보들의 개수가 증가함으로써 계산 작업부하가 증가하게 된다. 따라서, 예측 블록 후보들의 개수를 증가시켜 코딩 효율을 개선하기 위해서는, 예측 블록 후보들을 생성하는 프로세스를 고찰하여 프로세스의 효율성을 더 달성할 필요가 있다. 도 9에 도시한 프로세스의 고찰시, 두 개의 계산 병목 현상을 식별할 수 있다. 제1 계산 병목 현상은 단계 914의 비교 및 브랜칭 연산이며, 이는 루프 내에서 반복된다. 제2 계산 병목 현상은 단계 918의 제산이며, 이것도 루프 내에서 반복된다.

[0055] 오늘날에는, 효율적은 계산을 위해 단일 명령 다중 데이터(SIMD)가 이용가능하다. SIMD는 다중 처리 요소들을 갖춘 컴퓨터가 다중 데이터에 대하여 동일한 연산을 동시에 수행할 수 있게 한다. 그러나, 통상적인 SIMD 아키텍처는 루프에서의 계산/브랜칭 및 제산의 구현을 지원하지 않으며, 이에 따라, 도 9에 도시한 프로세스를 구현하는 데 사용될 수 없으며, 그 이유는 단계 907과 910에서 시작하는 루프가 SIMD로 구현될 정도로 강건함에도

불구하고 루프가 단계 914와 918을 포함하기 때문이다. 따라서, 본 발명의 목적은, 도 9에 도시한 프로세스로부터 계산 병목 현상을 제거하고 저 복잡성 인트라 예측을 제공하는 것이며, 이는 도 8에 도시한 모든 예측 방향을 따라 통상적인 SIMD 아키텍처가 병렬 처리를 구현할 수 있게 한다.

[0056] 도 10은, 본 발명의 일 실시예에 따라 도 5의 단계 504에서의 프로세스의 구현시 도 9의 프로세스를 대체하도록 설계된 저 복잡성 인트라 예측의 프로세스를 도시하는 흐름도이다. 도 10에서는, 도 9에서 수행되는 단계와 동일한 단계가, 단계 900, 901, 902, 904, 905, 906, 907, 908, 909, 910, 911, 912, 920, 및 921 등의 도 9에서 사용되는 단계 번호와 동일한 단계 번호에 의해 식별된다. 이러한 공통 단계들의 설명은 여기서 반복하지 않는다. 단계 1000 및 1001은 도 10의 프로세스의 고유한 단계들이다. 도 9에 도시한 프로세스와의 비교로 명백하듯이, 도 10의 프로세스는 단계 903의 비교 단계 및 단계 903부터 왼쪽으로 브랜칭된 모든 단계들을 제거하며, 이들은 *angle*이 zero보다 작은 경우 수행되며, 이에 따라 단계 914와 918의 계산 병목 현상을 제거한다.

[0057] 추가된 단계 1000과 1001에서는, *angle*이 -1 이상인지 여부를 결정한다. *angle*이 -1 이상이면, 수평 경계에 위치하는 참조 화소들은 예측 블록의 예측 화소를 생성하는 데 충분하고, 수직 경계에 이는 참조 화소들은 필요하지 않다. 반면에, *angle*이 -1보다 작으면, 수직 경계에 있는 참조 화소들은 예측 블록의 예측 화소들을 생성하는 데 필요하다. 단계 1001에서, *refH*에 저장된 참조 화소들은, *refV*에 저장된 화소들 중 적어도 일부를 이용하여 음의 방향으로 연장된다. 도 11a와 도 11b는 단계 1001에서 수행되는 *refH*의 연장을 도시하는 개략적인 표현이다. 도 11a에서, *refH*에 저장된 참조 화소들(1102)은 타겟 블록(1101) 위에 위치하는 수평 경계로부터 온 것이다. *refV*에 저장된 참조 화소들(1103)은 타겟 블록(1102)의 좌측에 위치하는 수직 경계로부터 온 것이다. 도 11b에 도시한 바와 같이, 도 10의 단계 1001 후에, *refV*에 있는 참조 화소들 중 일부는 *refH*에 복사되고, *refH*는 음의 방향으로 연장되는 연장부(1104)를 갖는다.

[0058] 도 12는 단계 1001에서 수행되는 프로세스의 상세를 도시하는 흐름도이다. 단계 1201에서, 카운터 *col*은 -1로 설정된다. *col*은 *refH*의 연장부의 어드레스를 식별하는 데 사용된다. 단계 1202에서, *refH*의 연장부로 복사될 *refV*에 있는 참조 화소는 아래의 식에 의해 식별된다.

[0059] $$\frac{size \times col}{angle}$$

[0060] 위 식에서의 제산은 정수 제산이며, 이 식에 따라 정수를 얻게 된다. 이 식은 도 9에 도시한 단계 918의 프로세스와 마찬가지로 기능한다. 단계 918에서, *pos2*의 정수 값은 아래의 식에 의해 계산된다.

[0061] $$\frac{(size^2 \times (col+1))}{angle} >> \log 2\_size$$

[0062] *log2_size* 에 의한 우측 시프트는 *size*에 의한 제산과 등가임에 주목한다.

[0063] 단계 1203에서는, *col*을 1씩 감분한다. 이어서, 단계 1204에서 *col*이 *angle*과 같은지 여부를 결정한다. *col*이 *angle* 과 같지 않으면, 흐름은 단계 1202로 되돌아간다. 단계 1202와 1203은, *col*이 *angle*과 같아질 때까지 반복된다. 따라서, 참조 화소들은, 올림 차순으로 *refV*로부터 즉 수직 경계의 상부로부터 하부로 판독되고, 또한 내림 차순으로 즉 수평 경계의 우측으로부터 좌측으로 *refH* 내로 복사된다. 또한, *refV* 내의 모든 참조 화소들이 *refH* 내로 복사되지는 않는다. 상부로부터 예측 방향의 교차부로 향하는 범위 내에 위치하는 참조 화소들만이 *refV* 로부터 *refH* 로 복사된다.

[0064] 도 10으로 되돌아가서, 단계 902부터 시작되는 프로세스 단계들은 도 9로부터 복사되며, 도 9의 단계 903의 비교 단계로부터 우측으로 브랜칭된 예측 화소들을 생성하기 위한 단계들을 포함한다. 그러나, 예측 화소들을 생성하기 위한 도 10의 단계들은 연장된 *refH*(도 11b의 1102 + 1104 부분들의 합)를 사용하는 반면, 도 9의 대응 단계들은 초기 *refH*(도 10A의 1102 부분)를 사용한다는 점에 주목한다. *refH* 는 음의 방향으로 연장되기 때문에, 도 9의 단계 903으로부터 좌측으로 브랜칭된 *refV* 등에 저장된 참조 화소들을 사용하도록 구체적으로 설계된 별도의 인트라 예측 연산은 *angle*의 부호에 상관 없이 필요하지 않다.

[0065] 도 13은 *refV* 의 참조 화소들을 이용하는 연장 *refH* 를 위한 프로세스의 다른 실시예를 도시하는 흐름도이다. 도 11과 도 12에 도시한 프로세스는, 도 9에 도시한 단계 914와 918의 병목 현상 단계들을 제거하기 때문에, 인트라 예측 프로세스의 효율을 개선할 것으로 예상된다. 도 13에 도시한 프로세스는, 참조 화소들을 *refV* 로부터 *refH*로 복사하기 위한 루프로부터 도 12의 단계 1202에서 수행되는 제산을 제거한다. 루프로부터 제산을 제

거함으로써, 도 13에 도시한 프로세스는 인트라 예측 프로세스의 효율을 더 개선할 것으로 예상된다.

[0066] 도 13에 도시한 프로세스는 도 12의 단계 1202를 단계 1301과 1302로 대체한다. 단계 1302는, 참조 화소들을 *refV*로부터 *refH*로 복사하기 위한 루프 내에 해당되는 반면, 단계 1301은 그 루프 외부에 존재한다. 단계 1301은 "*InvAngle*"이라 칭하는 새로운 파라미터를 도입한다. *InvAngle*은 아래의 식에 의해 정의된다.

[0067] $$256 \times \frac{size}{angle}$$

[0068] 256을 승산하는 것은, 8만큼의 좌측 시프트와 등가이며, "*size*/*angle*"의 연산으로부터 발생하는 모든 비트가 *refV*의 참조 화소를 식별하는 계산을 고려함을 보장한다. 단계 1302에서, *refH*의 연장부 내로 복사될 *refV*의 참조 화소의 어드레스는 아래의 식에 의해 식별된다.

[0069] $$col \times InvAngle >> 8$$

[0070] "*col* x *InvAngle*"의 결과는, 8만큼 우측 시프트되어 단계 1301에서 수행되는 좌측 시프트 연산을 무효로 한다(undo). 단계 1302에서의 우측 시프트 연산은 "*col* x *InvAngle*"의 결과를 내림(round down)하도록 기능한다는 점에 주목한다. 가장 가까운 정수로의 내림을 위해, 우측 시프트 연산이 수행되기 전에 128의 반올림 오프셋이 "*col* x *InvAngle*"의 결과에 더해질 수 있다. 숫자 "256"은 단지 일례이며, 그 숫자가 "*size*/*angle*"의 연산으로부터 발생하는 모든 비트들을 유지할 정도로 크다면 단계 1301이 다른 오프셋 숫자를 채용할 수 있으며, 바람직하게는, 2의 정수 멱을 채용할 수 있다는 점에 주목한다. 예를 들어, 그 숫자는 단계 1301에서 256 대신에 64일 수 있으며, 단계 1302에서 우측 시프트의 숫자는 8 대신에 6이다. 64를 채용하면, 반올림 오프셋이 32로 된다.

[0071] 단계 1301에서 수행되는 계산은 룩업 연산으로 대체되어 계산 작업 부하를 더 저감할 수 있다. 다시 말하면, *InvAngle* 의 값을 *angle*의 값에 관련지어 저장한 룩업 테이블을 준비한다. 이하의 표 1은 단계 1301에서의 룩업을 위한 예시적인 표이다.

표 1

[0072]

angle	1	2	3	4	5	6	7	8
InvAngle	2048	1024	683	512	410	341	293	256

[0073] 위 표에서, *size*는 8이고, *angle*은 1 내지 8의 정수 값을 취한다고 가정한다. 그러나, *size*는 8로 한정되지 않으며 4와 16 등의 다른 값을 취할 수 있다는 점에 주목한다. 또한, *angle*은 전술한 바와 같이 고정 소수점 방식으로 분수일 수 있다.

[0074] 도 12의 단계 1202 또는 도 13의 단계 1302에서 참조 화소가 *refV*로부터 *refH*로 복사되면, 참조 화소가 로우 패스 필터를 거쳐 예측 블록에서의 가능한 애일리어싱을 저감시킬 수 있다. 로우 패스 필터의 강도는 *angle*의 값에 따라 가변될 수 있다. 예를 들어, *angle*이 -*size*와 동일하면 약한 로우 패스 필터링을 적용할 수 있고, *angle*이 -2이면 강한 로우 패스 필터링을 적용할 수 있다.

[0075] 전술한 바와 같이, *refV*내의 모든 참조 화소들이 *refH* 내로 복사되지는 않는다. *refV* 내의 모든 참조 화소들이 복사되지는 않으므로, 화소들의 복사시 일부 정보가 손실된다. 정보 손실을 완화하도록, *refH*와 *refV*의 참조 화소들의 해상도를 두 배로 하여, *refH*와 *refV*가 이전에 인코딩되고 재구축된 블록들로부터의 화소들을 포함할 뿐만 아니라 두 개의 인접하는 재구축된 화소들 사이에 있으며 두 개의 인접 화소들을 보간함으로써 생성되는 하나의 화소도 포함할 수 있다. 두 개의 인접하는 화소들은 간단히 평균화되어 보간 화소를 생성할 수 있다. 보간 프로세스는 참조 화소들이 도 9의 단계 900에서 판독되는 경우 수행될 수 있다. 화소들의 해상도가 *refH*와 *refV*에서 두 배로 되면, 도 9의 단계 907, 910, 915, 919 및 도 10의 단계 1001에서 수행되는 바와 같은, *refH*와 *refV*에 저장된 참조 화소들의 어드레스의 식별을, 스케일링할 필요가 있다. 예를 들어, 단계 907, 910, 915에서 수행되는 "*int* + *col* + 1"은 "*int* + 2 x *col* + 2"로 변경될 필요가 있다. 단계 907, 910, 915에서 수행되는 "*int* + *col* + 2"는 "*int* + 2 x *col* + 3"으로 변경될 필요가 있다. 단계 919에서 수행되는 "*int2* + *row* + 1"과 "*int2* + *row* + 2"는, "*int2* + 2 x *row* + 2"와 "*int2* + 2 x *row* + 3"으로 각각 변경될 필요가 있다.

[0076] 다른 일 실시예에서, 도 12의 단계 1202의 프로세스를 "*refH*[*col*]←*refV*[-*col*]"로 간단히 변경하여 복사 프로세

스를 더욱 간단하게 할 수 있다. 예측 정확성이 떨어지지만, 본 실시예는 인트라 예측 연산에 최저 복잡성을 제공한다.

[0077] 도 11b는 *refH*에 더해진 연장부(1104)를 도시한다. 연장부(1104)는 *refV*로부터의 참조 화소들로 형성될 필요가 없다. 연장부(1104)는, 연장부(1104)의 위치에 공간적으로 대응하는 이전에 재구축된 블록의 영역으로부터의 화소들로 형성될 수 있다. 도 11b에서는, 음의 방향으로 연장되어 있으므로, 연장된 *refH*(1102와 1104 부분)의 범위는 -*size*+1 내지 2 x *size*이다. 연장된 *refH*의 범위는, 연장된 *refH*의 참조 화소들을 어드레싱하는 경우 적절한 오프셋을 더함으로써 0 내지 3 x *size*- 1로 리스케일링될 수 있다. 이는 *refV*의 범위를 리스케일링하는 경우에도 동일하게 적용가능하다.

[0078] 또 다른 일 실시예에서, *angle*의 범위 한계는 자유롭게 선택될 수 있다. 위 실시예들에서는, *angle*이 -*size* 내지 *size*의 값(-*size* ≤ *angle* ≤ *size*)을 취한다고 가정하고 있다. 다시 말하면, 위 실시예들에서, *angle*의 범위 한계는 타겟 블록의 크기와 함께 형성된다. *angle*의 범위 한계를 2의 정수 멱으로 정의하여 *log2_rangelimit*이 양의 정수이고 "*rangelimit* = 1 << log2_*rangelimit*" 식이 참으로 되게 하는 것이 여전히 바람직하지만, *angle*의 범위 한계는 타겟 블록의 크기와는 독립적으로 정의될 수 있다는 점에 주목한다. *rangelimit*에 대한 적절한 큰 수를 선택함으로써, 많은 예측 방향들을 설정할 수 있고, 충분히 넓은 각도 간격으로 *angle*의 값들에 의해 표현할 수 있다.

[0079] *angle*의 범위 한계가 타겟 블록의 크기와는 독립적으로 정의되면, 단계 909, 912, 917, 921을 제외하고, 도 9와 도 10에 보이는 size는 *rangelimit*으로 대체될 필요가 있고, *log2_size*는 *log2_rangelimit*으로 대체될 필요가 있다. 도 10의 단계 1000에서 수행되는 "*angle* ≥ -1"도, "*angle* x *size*/*rangelimit* ≥ -1" 또는 "*angle* x *size* ≥ -*rangelimit*"으로 대체될 필요가 있다. 또한, 도 12와 도 13의 단계 1202와 13011203에서 보이는 *size*는 *rangelimit*으로 대체될 필요가 있고, 단계 1204에서 수행되는 "*col*=*angle*?"의 비교를 "*col* = *angle* x *size*/*rangelimit*?"으로 대체할 필요가 있다.

[0080] *rangelimit* 이 *angle*의 범위 한계로서 도입되면, (위에서 제공된) 표 1는 다음과 같이 변경될 수 있다.

표 2

[0081]

*Angle**	2	5	9	13	17	21	26	32
InvAngle	4096	1638	910	630	482	390	315	256

[0082] 표 2에서, *rangelimit*은 32로 설정되어 있다. *Angle**은 "*rangelimit* x tan (π x *angle*/8)"의 정수 근사화에 같고, 여기서 *angle*은 1, 2, 3, 4, 5, 6, 7, 8이다. *InvAngle*은 256 x *rangelimit*/*angle**과 같다. 표 2의 값들 모두는 올림에 의해 도출되는 정수이다. 올림 대신에, 이 값들을 내림할 수 있다. 이하의 표 3에서는, *InvAngle*이 32 x *rangelimit*/*angle**과 같다. "256"대신에 "32"가 사용되므로, 예측 정확성은 표 2의 예측 정확성보다 필수적으로 낮다.

표 3

[0083]

*Angle**	2	5	9	13	17	21	26	32
InvAngle	512	204	113	78	60	48	39	32

[0084] 도 14는 도 10에 도시한 프로세스를 더욱 간략화하는 또 다른 일 실시예를 도시하는 흐름도이다. 참조 화소들을 *refV*로부터 *refH*로 복사하는 도 10에 도시한 프로세스는 흐름이 주요 예측 루프로 진입하기 전에 수행되는 반면, 도 14에 도시한 복사 프로세스는 주요 예측 루프 내에서 수행된다. 또한, 도 14에 도시한 프로세스는 변수 *InvAngle*을 제거한다. 도 14에 도시한 단계 900, 902, 921은 도 10의 대응하는 단계에 해당되는 것이다.

[0085] 단계 1401에서, 카운터 *lastInt*가 -1로 초기화된다. *lastInt*는 *refH*에 더해진 최종 화소의 인덱스를 나타낸다. 단계 902에서, *pos*는 *angle* x (*row* + 1)에 의해 계산된다. 전술한 바와 같이, *pos*는 *angle* 에 의해 표현되는 예측 방향과 경계들 간의 교차의 위치를 식별한다. 도 9의 문맥에서, 단계 902는 *pos*를 출력하고, 이는 *angle* 에 의해 표현되는 예측 방향과 수평 경계 간의 교차의 위치를 식별한다. 또한, 단계 902에서, *pos*의 정수부는 *int*에 저장되고, *pos*의 분수부는 "*frac*"에 저장된다. 단계 1402에서는, *int*가 *lastInt*보다 작은지 여부를 결정

한다. *int*가 *lastInt*보다 작으면, *row*에 의해 식별된 *refV*의 참조 화소가 "*int*+1"에 의해 식별된 어드레스에서 *refH*에 복사된다. 단계 1404는, 도 9와 도 10에 도시한 단계들 904, 905, 906, 907, 908, 909, 910, 911, 912로 이루어지지만, 그 설명은 여기서 반복하지 않는다. 단계 1405, *int*는 *lastInt*로 복사된다. *int*를 *lastInt*로 복사하는 동작은 단계 1405 대신에 단계 1403에서 수행된다.

[0086] 단계 1403에서의 복사 동작에 따라, 단계 1202와 1302에서 복사된 화소와 동일한 화소를 복사하게 되며, 이러한 단계들에서는 내림을 이용한다. 단계 902에서 계산된 분수 위치 *frac*이 *rangelimit* + (*angle* >> 1)에 의해 정의된 *offset* 보다 크면, 단계 1403를 수정하여 단계 1403에서의 "*row*" 대신에 "*row* +1"을 조건적으로 이용함으로써 가장 가까운 정수로 반올림을 행할 수 있다. *angle* 이 -ve이고, *frac*이 +ve라는 점에 주목한다. "*row*+1"을 이용함으로써, 올림이 발생한다. *row* 을 1씩 조건적으로 증분하려면, 단계 1403에서 수행되는 프로세스를 *refH*[*int* + 1] ← *refV*[*row* - ((*offset* - *frac*) >> 31)로 변경하며, 32비트 산술에서, *frac*이 *offset*보다 큰 경우 "*offset* - *frac*"의 우측 시프트에 따라 -1이 발생하고 그 외에의 경우에는 0이 발생한다고 가정한다. 따라서, *frac*이 *offset*보다 큰 경우 어드레스 식별자 " *row* - ((*offset* - *frac*) >> 31)"은 "*row* + 1"로 되고, 그 외의 경우에는 "*row*"로 된다. *offset* 이 *rangelimit*으로 설정되면, "*offset-frac*"은 항상 양의 값이며 따라서 반올림이 발생하지 않는다.

[0087] 도 14에 도시한 프로세스를 구현하는 C++ 프로그래밍 언어에서 개발된 소스 코드가 이하에 열거되어 있다. 소스 코드는, 에서 이용가능한 JCT-VC에 의해 개발된 TMuC 0.7 소프트웨어의 일부인 TComPrediction.cpp 파일에서 발견되는 TComPrediction::xPredIntraAng 함수로부터 수정된다.

[0088] // 단순화된 각도 인트라 예측을 유도하기 위한 함수

```
Void TComPrediction: :xPredIntraAng( Int* pSrc, Int iSrcStride, Pel*& rpDst,
Int iDstStride, UInt iWidth, UInt iHeight, UInt uiDirMode, Bool bAbove, Bool
bLeft ){

  Int k,l;

  Int deltaInt, deltaFract, refMainIndex;

  Int intraPredAngle       = 0;

  Int absAng        = 0;

  Int signAng              = 0;

  Int blkSize              = iWidth;

  Bool modeDC              = false;

  Bool modeVer             = false;

  Bool modeHor             = false;

  Pel* pDst         = rpDst;

// 모드 인덱스를 주 예측 방향 및 각도로 맵핑

  if (uiDirMode == 0)

    modeDC = true;

  else if (uiDirMode < 18)

    modeVer = true;

  else

    modeHor = true;


  intraPredAngle = modeVer ? uiDirMode - 9 : modeHor ? uiDirMode - 25 : 0;

  absAng              = abs(intraPredAngle);

  signAng      = intraPredAngle < 0 ? -1 : 1;


  // 비트이동을 세팅하고 각도파라미터를 사이즈2로 스케일링

  Int iAngTable[9]  = {  0,  2,  5,  9,  13,  17,  21,  26,  32};

  absAng = iAngTable[absAng];

  intraPredAngle = signAng * absAng;


  // DC 예측

  if (modeDC){

    Pel dcval = predIntraGetPredValDC(pSrc, iSrcStride, iWidth, iHeight,
bAbove, bLeft);
```

[0089]

```
  for (k=0;k<blkSize;k++){
    for (l=0;l<blkSize;l++){
      pDst(k*iDstStride+l] = dcval;
    }
  }
}
// 각도 예측
else {
  Pel tmp;
  Int *pSrcTL = pSrc - iSrcStride - 1;
  Int iStepMain = (modeVer) ? 1 : iSrcStride;
  if (intraPredAngle == 0){
    for (k=0;k<blkSize;k++){
      for (l=0;l<blkSize;l++){
        pDst [k*iDstStride+l] = pSrcTL[(l+1) * iStepMain];
      }
    }
  }
  else {
    Int iStepSide = (modeVer) ? iSrcStride 1;
    int lastDeltaInt = -1;
    Int iOffset = 32 + (intraPredAngle >> 1);        // 가장 근접한 사이드 참조로 라
                                                     운딩 가능하게 함
// Int iOffset = 32;                                // 라운딩 무
    Pel ref [2*MAX_CU_SIZE];
    Pel* refMain =  ref + ((intraPredAngle < 0) ? blkSize : 0);
    if (intraPredAngle > 0){
      for (k = 0; k  <  2*blkSize; k++)
        refMain[k] = pSrcTL[(k+1)  *  iStepMain];
    }
    else {
      for (k = -1; k < blkSize; k++)    // 필요시, 나머지는 이후1403단계에서 복
                                        사됨
```

[0090]

```
        refMain[k] = pSrcTL[(k+1) * iStepMain];
      }

      for (k = 0; k < blkSize; k++){
      Int deltaPos = (k+1) * intraPredAngle;
      deltaInt       = deltaPos >> 5;
      deltaFract = deltaPos & (32 - 1);
      if (deltaInt < lastDeltaInt) { //1402 단계
        lastDeltaInt = deltaInt;
        refMain[deltaInt] = pSrcTL[(k-((iOffset-deltaFract)>>31))*iStepSide];
// step 1403
      }

      // 1404 단계
      if (deltaFract){
        // 선형 필터링
        for (l=0;l<blkSize;l++){
          refMainIndex = l+deltaInt;
          pDst[k*iDstStride+l] = (Pel) ( ((32-deltaFract) *
refMain[refMainIndex] + deltaFract * refMain[refMainIndex+1] + 16) >> 5 );
        }
      }
      else {
        // 정수 샘플 복사
        for (l=0;l<blkSize;l++) {
          pDst[k*iDstStride+l] = refMain[l+deltaInt];
        }
      }
    }
  }
  // 수평 모드인 경우 블록 플립
  if (modeHor){
    for (k=0;k<blkSize-1;k++){
      for (l=k+1;l<blkSize;l++){
```

[0091]

```
          tmp = pDst[k*iDstStride+l];
          pDst[k*iDstStride+l] = pDst[l*iDstStride+k];
          pDst[l*iDstStride+k] = tmp;
        }
      }
    }
  }
```

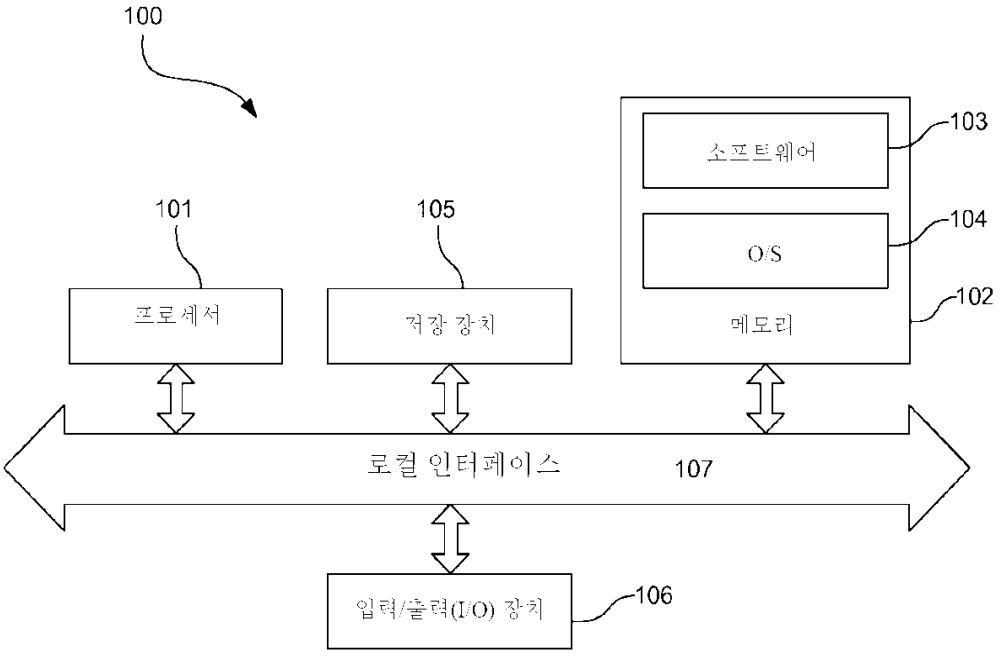
[0092]

[0093] 전술한 상세한 설명을 읽고 나서, 본 발명에 대한 많은 변경과 변형이 의심의 여지없이 당업자들에게 명백할 것이며, 도시하는 방식으로 보여주고 설명한 임의의 구체적인 실시예가 제한을 고려한 의도가 없다는 사실을 이해해야 한다. 따라서, 다양한 실시예들에 대한 설명을 참조하는 것은 본 특허청구범위들의 범위를 제한하려는 의

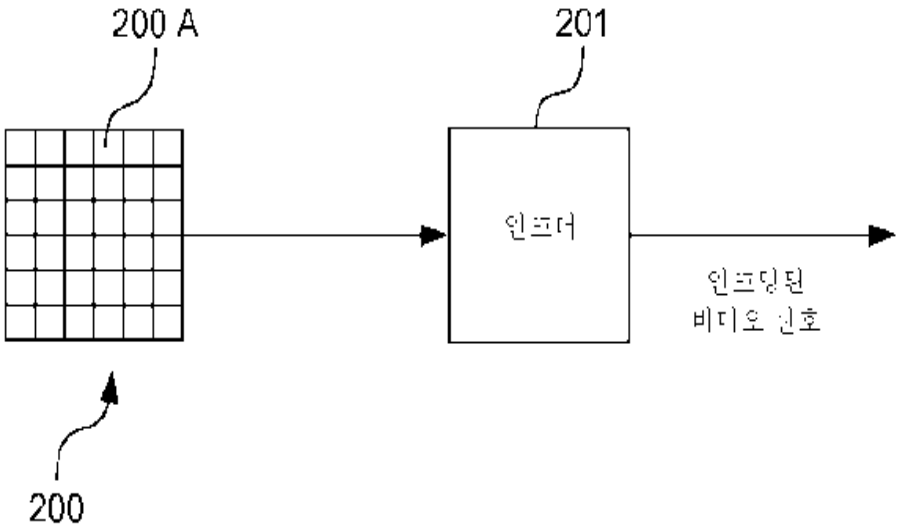
도가 없으며, 그 자체로 본 발명의 사상으로 간주되는 특징들을 원용할 뿐이다.

도면

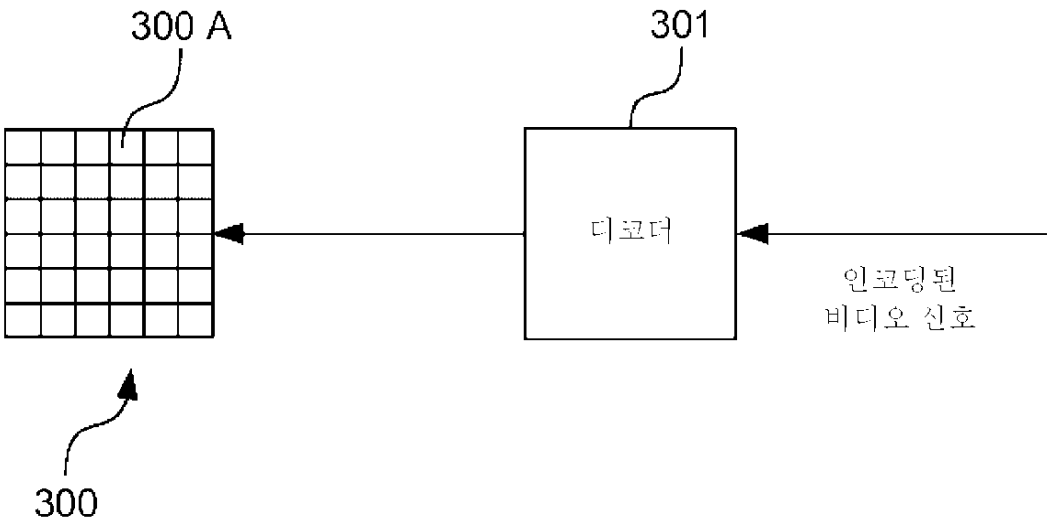
도면1



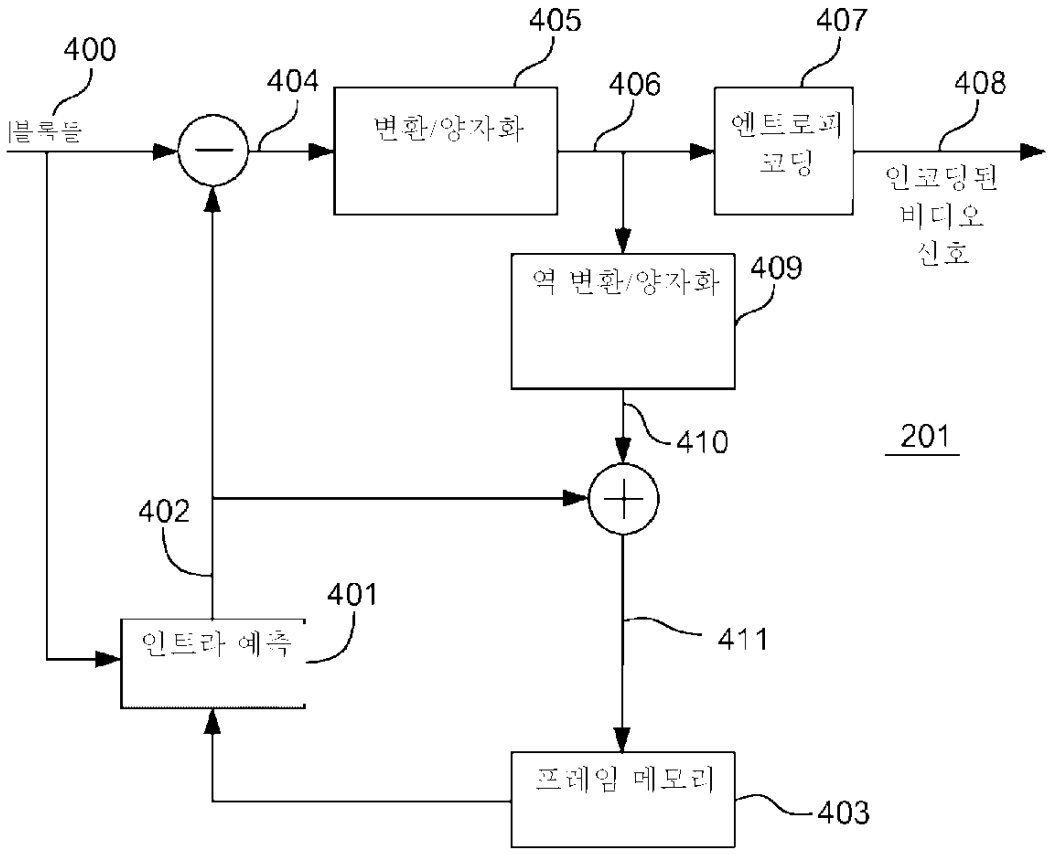
도면2



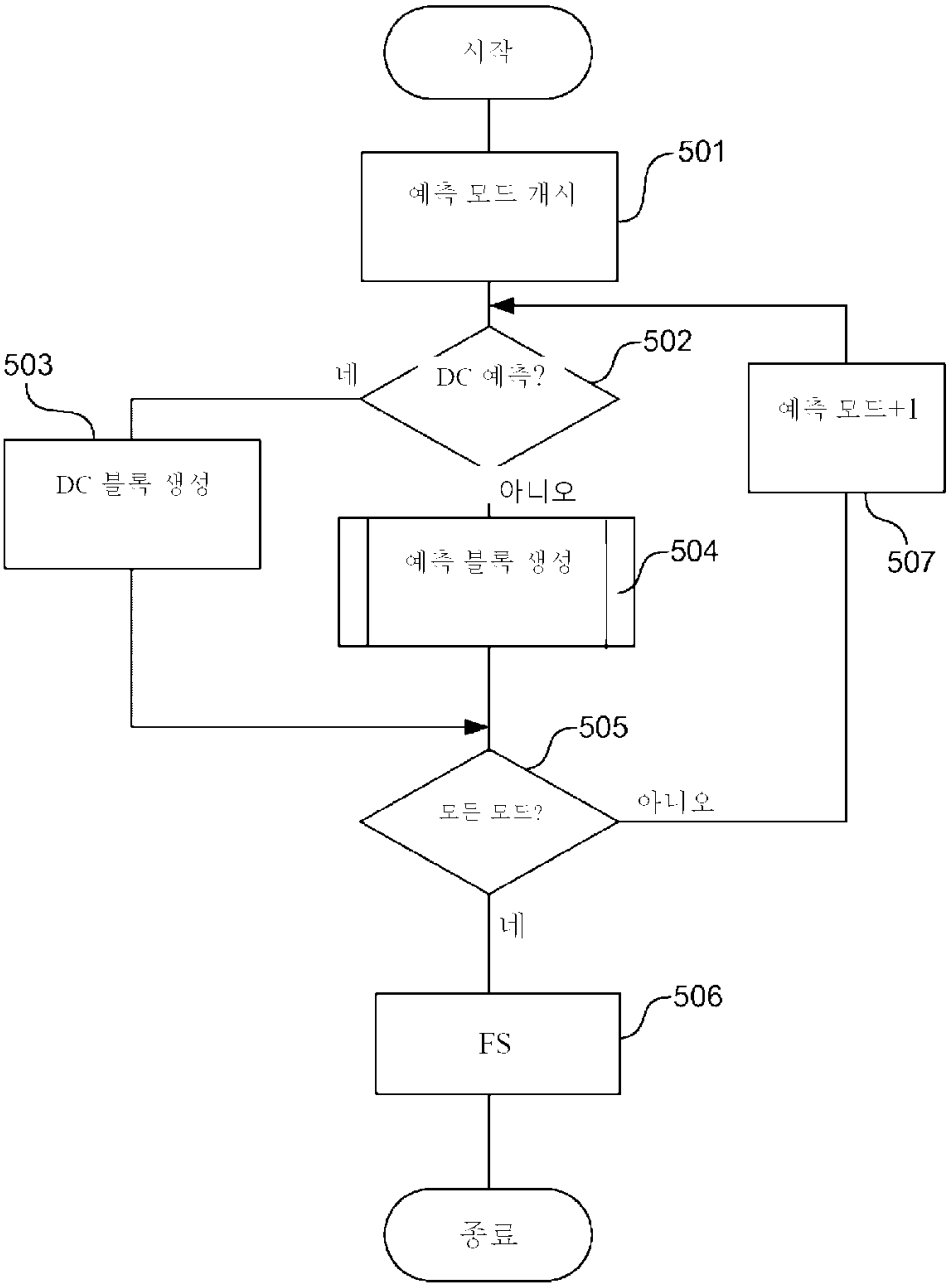
도면3


300 A
301
디코더
인코딩된
비디오 신호
300

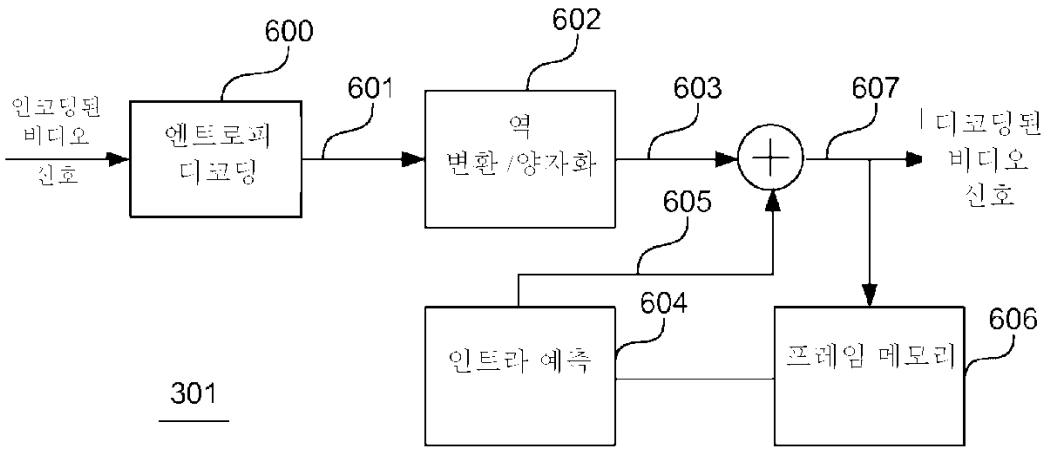
도면4


400
블록들
404
405
변환/양자화
406
407
엔트로피
코딩
408
인코딩된
비디오
신호
역 변환/양자화
409
410
201
402
401
인트라 예측
411
프레임 메모리
403

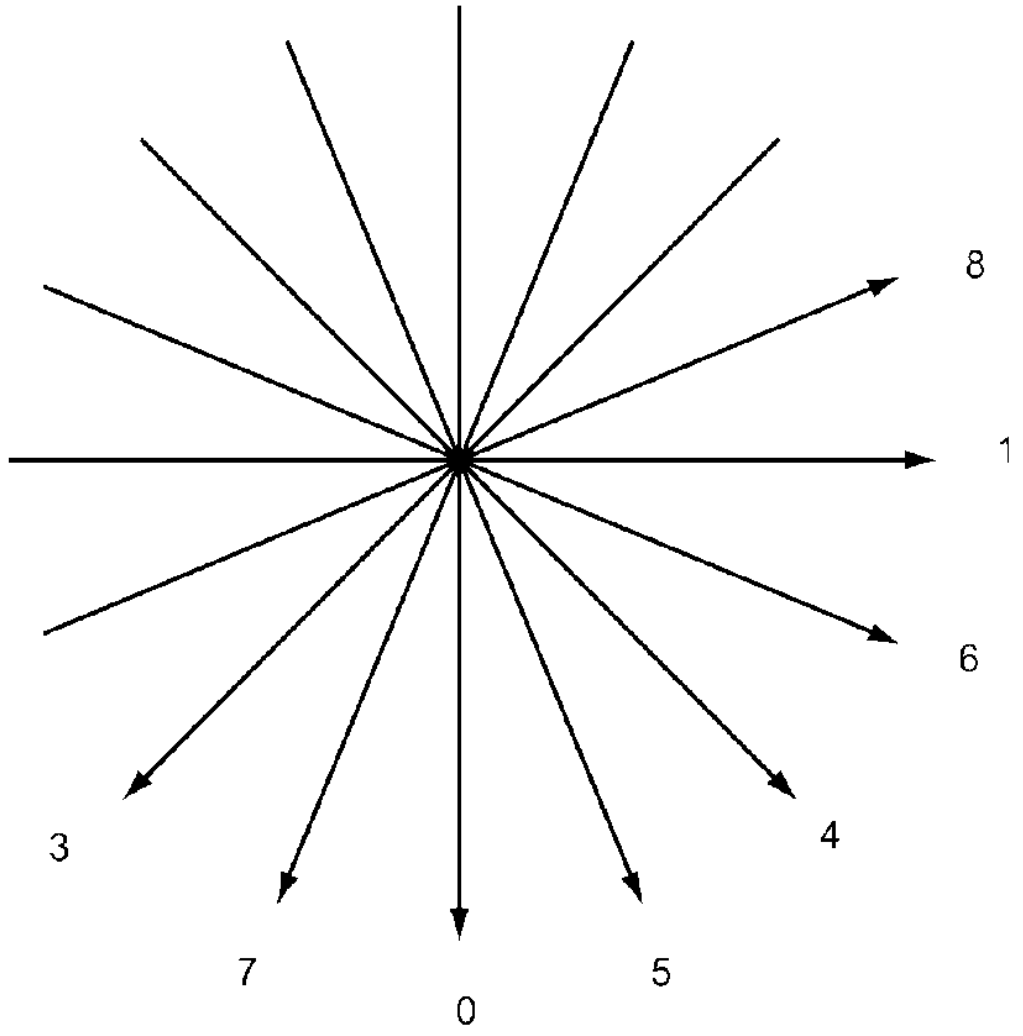
도면5


시작
예측 모드 개시
501
DC 예측?
502
네
503
DC 블록 생성
아니오
예측 블록 생성
504
예측 모드+1
507
모든 모드?
505
아니오
네
FS
506
종료

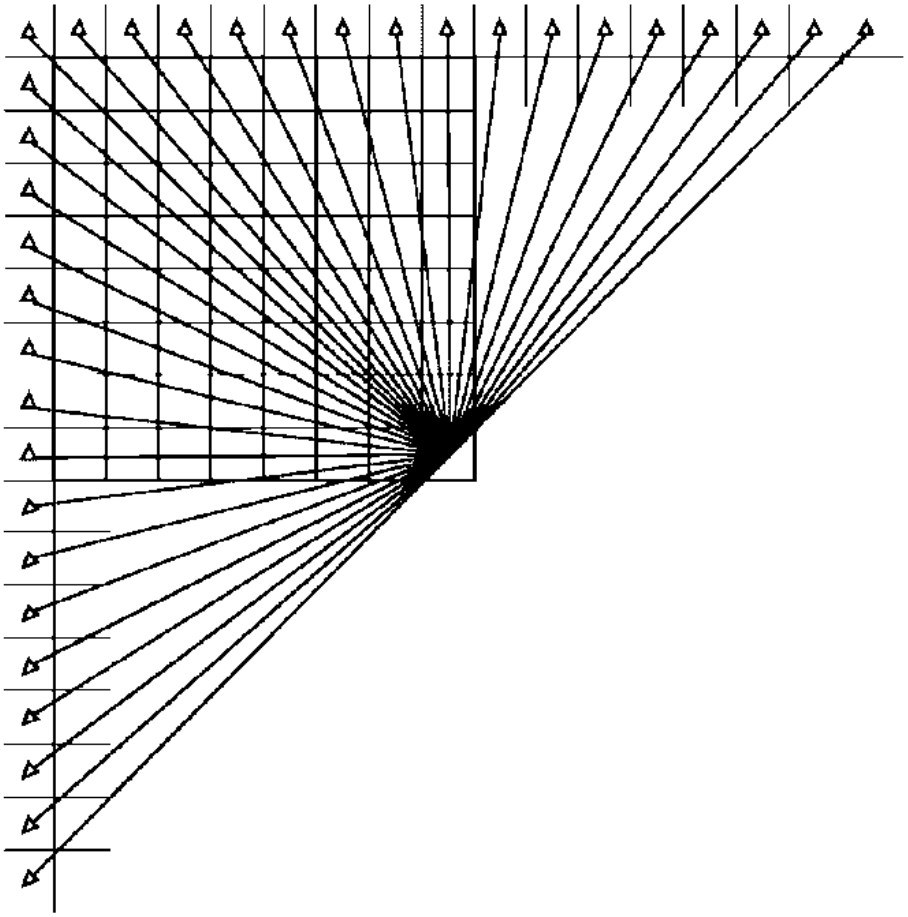
도면6


600
602
601
603
607
인코딩된
비디오
신호
엔트로피
디코딩
역
변환 /양자화
디코딩된
비디오
신호
605
604
606
인트라 예측
프레임 메모리
301

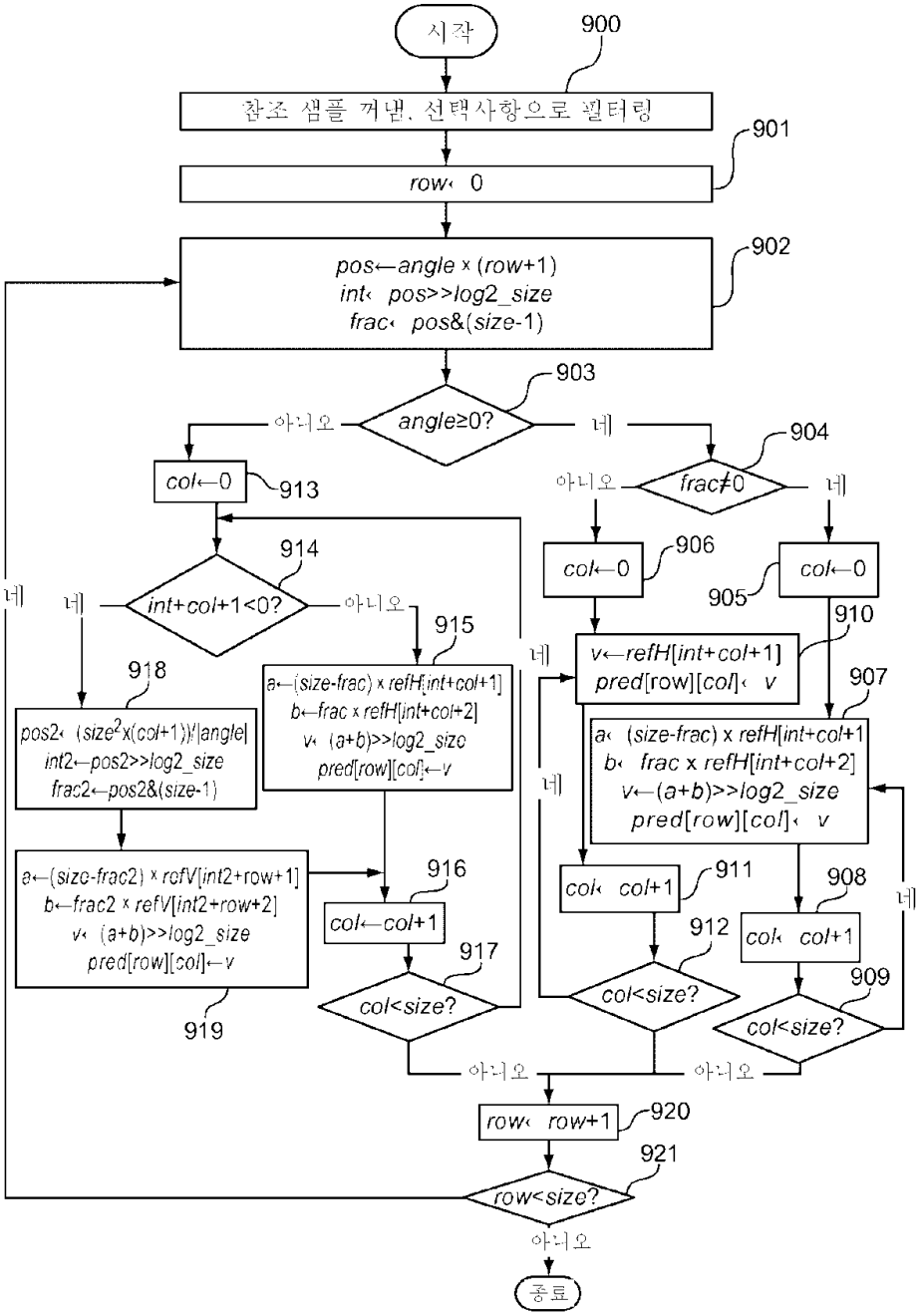
도면7


8
1
6
3
4
7
5
0

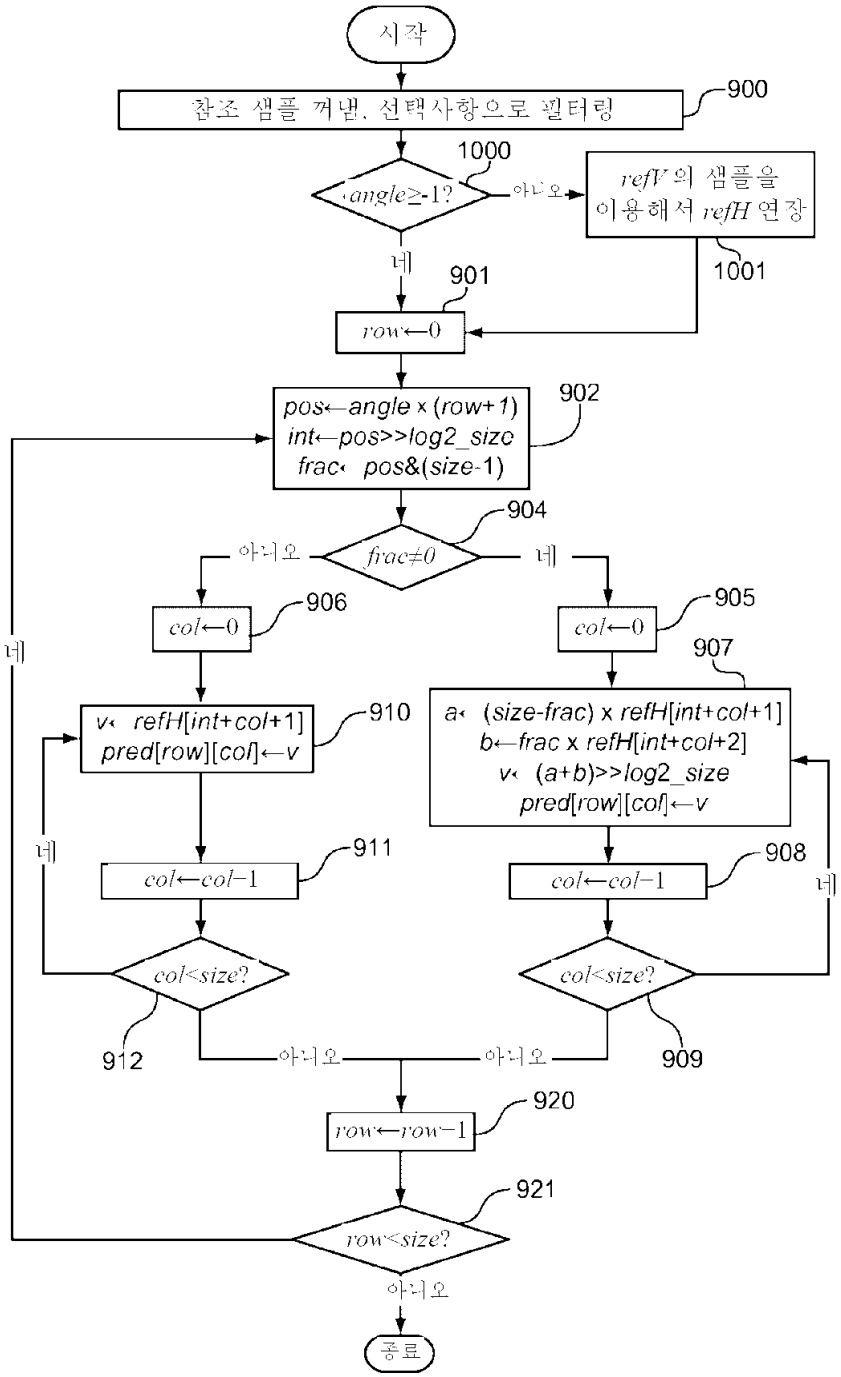
도면8



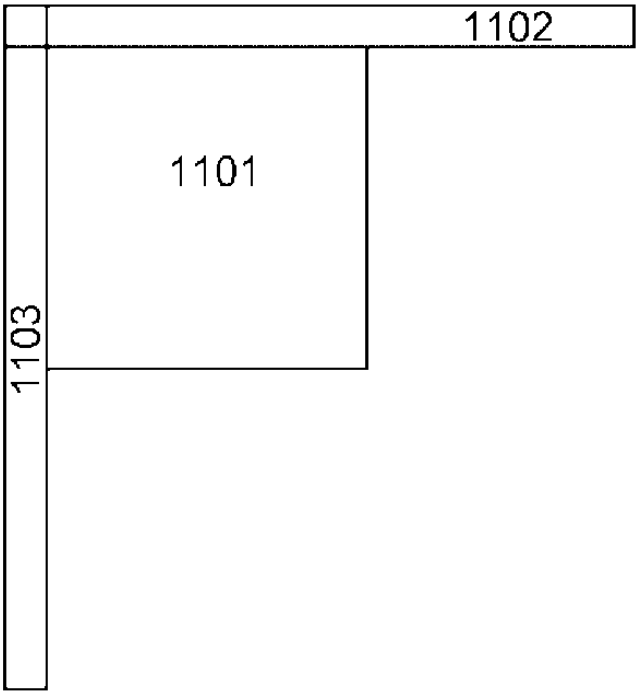
도면9


시작
900
참조 샘플 꺼냄, 선택사항으로 필터링
901
row← 0
902
pos←angle x (row+1)
int← pos>>log2_size
frac← pos&(size-1)
903
angle≥0?
아니오
네
904
frac≠0
아니오
네
col←0
913
914
int+col+1<0?
네
네
아니오
915
a←(size-frac) x refH[int+col+1]
b←frac x refH[int+col+2]
v← (a+b)>>log2_size
pred[row][col]←v
918
pos2← (size²x(col+1))/|angle|
int2←pos2>>log2_size
frac2←pos2&(size-1)
a←(size-frac2) x refV[int2+row+1]
b←frac2 x refV[int2+row+2]
v← (a+b)>>log2_size
pred[row][col]←v
919
916
col←col+1
917
col<size?
아니오
col←0
906
905
col←0
910
네
v←refH[int+col+1]
pred[row][col]← v
907
네
a← (size-frac) x refH[int+col+1]
b← frac x refH[int+col+2]
v←(a+b)>>log2_size
pred[row][col]← v
911
col← col+1
912
col<size?
아니오
908
col← col+1
909
col<size?
아니오
네
920
row← row+1
921
row<size?
아니오
종료

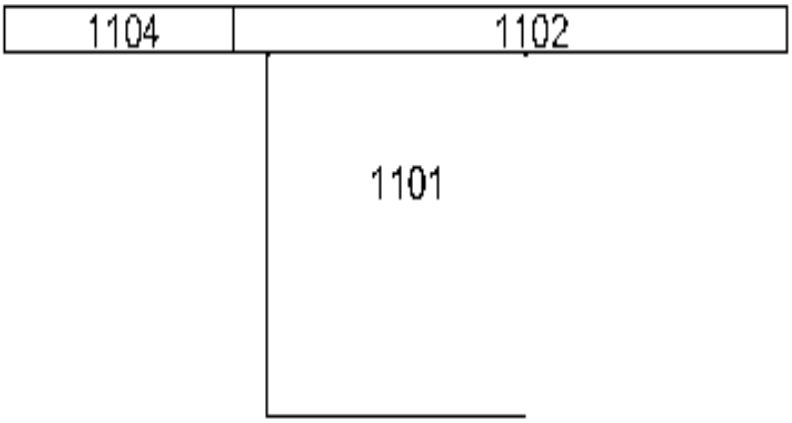
도면10


시작
참조 샘플 꺼냄, 선택사항으로 필터링
900
1000
angle≥-1?
아니오
refV 의 샘플을 이용해서 refH 연장
1001
네
901
row←0
902
pos←angle x (row+1)
int←pos>>log2_size
frac← pos&(size-1)
904
frac≠0
아니오
네
906
col←0
905
col←0
907
910
v← refH[int+col+1]
pred[row][col]←v
a← (size-frac) x refH[int+col+1]
b←frac x refH[int+col+2]
v← (a+b)>>log2_size
pred[row][col]←v
911
col←col+1
908
col←col+1
네
네
네
col<size?
col<size?
912
909
아니오
아니오
920
row←row+1
921
row<size?
아니오
종료

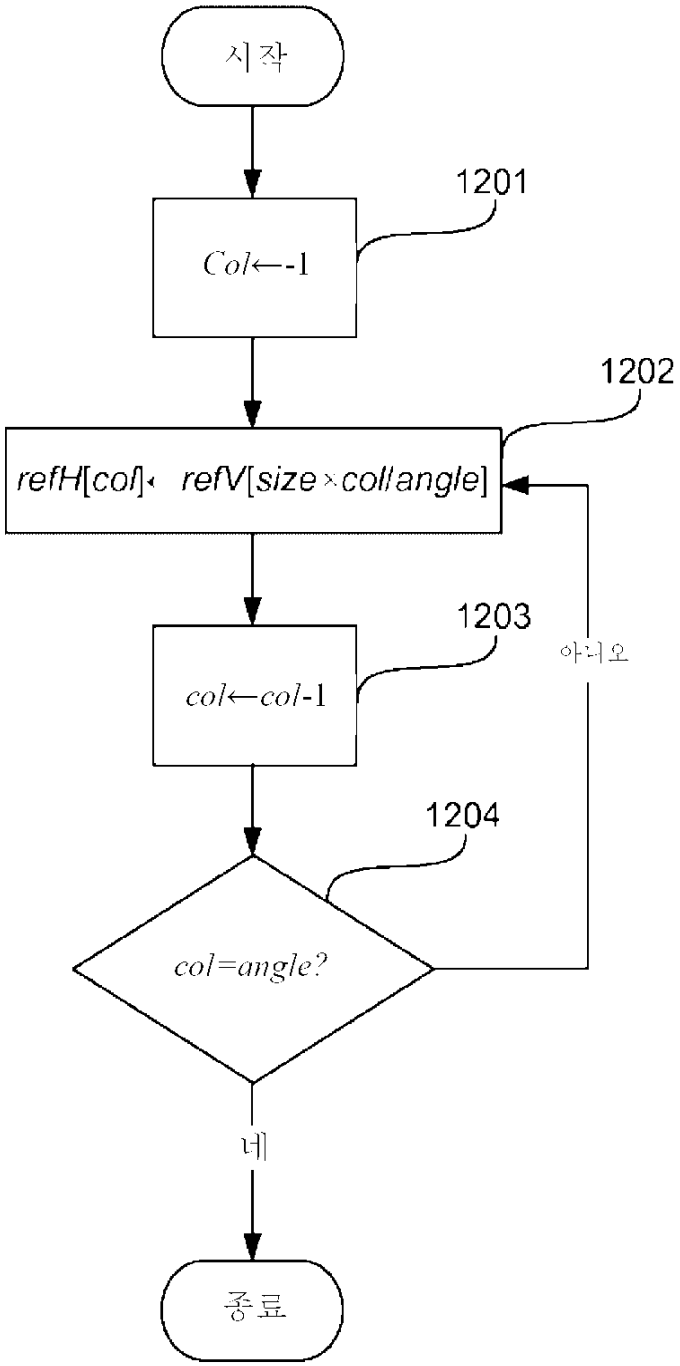
도면11a



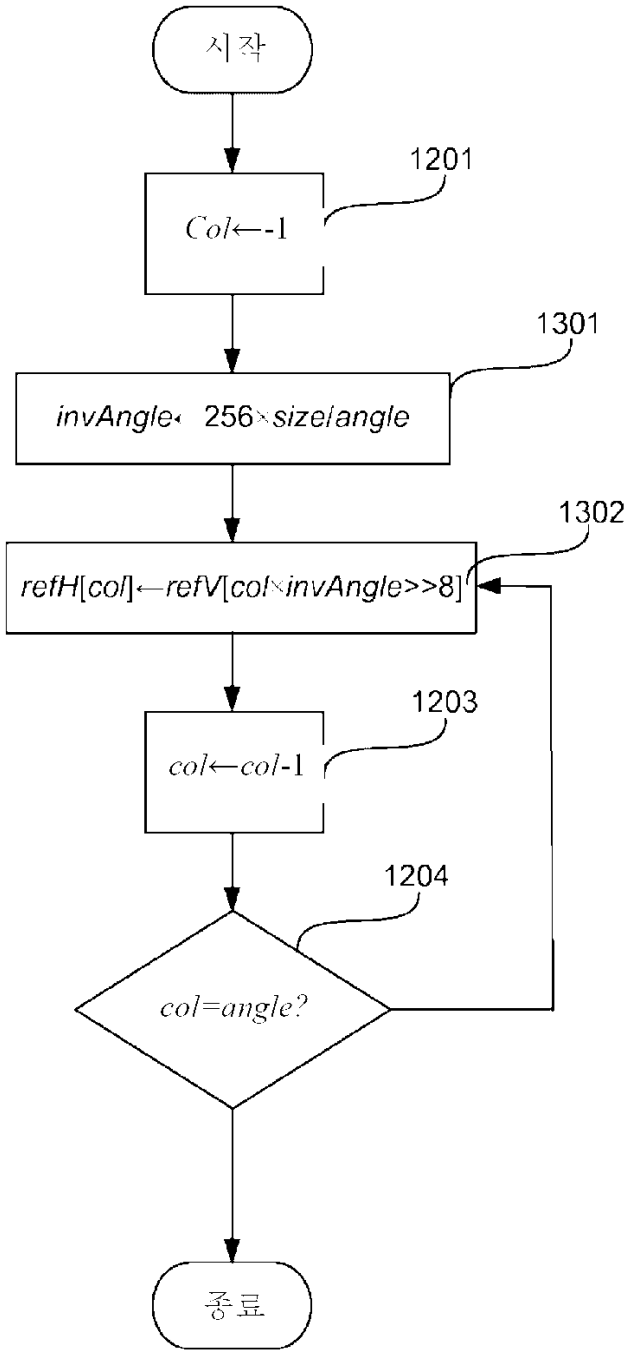
도면11b



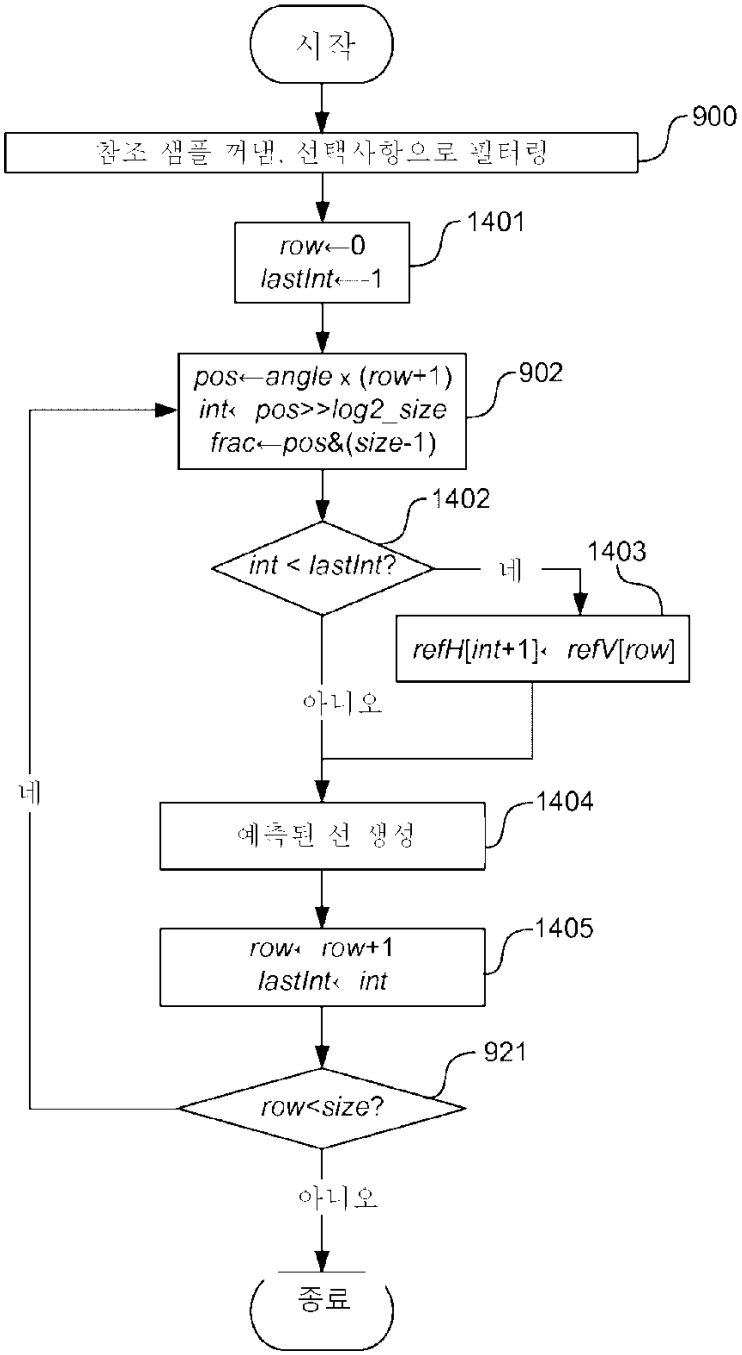
도면12


시작
1201
Col←-1
1202
refH[col]← refV[size×col/angle]
1203
col←col-1
아니오
1204
col=angle?
네
종료

도면13


시작
1201
Col←-1
1301
invAngle← 256×size/angle
1302
refH[col]←refV[col×invAngle>>8]
1203
col←col-1
1204
col=angle?
종료

도면14


시작
참조 샘플 꺼냄, 선택사항으로 필터링
900
row←0
lastInt←-1
1401
pos←angle x (row+1)
int← pos>>log2_size
frac←pos&(size-1)
902
1402
int < lastInt?
네
1403
refH[int+1]← refV[row]
아니오
네
예측된 선 생성
1404
row← row+1
lastInt← int
1405
921
row<size?
아니오
종료