

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7182006号
(P7182006)

(45)発行日 令和4年12月1日(2022.12.1)

(24)登録日 令和4年11月22日(2022.11.22)

(51)国際特許分類 F I
H 0 4 N 19/70 (2014.01) H 0 4 N 19/70

請求項の数 22 (全42頁)

(21)出願番号	特願2021-534612(P2021-534612)	(73)特許権者	598036300
(86)(22)出願日	令和1年12月19日(2019.12.19)		テレフオンアクチーボラゲット エルエム
(65)公表番号	特表2022-514558(P2022-514558 A)		エリクソン (パブル)
(43)公表日	令和4年2月14日(2022.2.14)		スウェーデン国 ストックホルム エスー
(86)国際出願番号	PCT/SE2019/051323	(74)代理人	1 6 4 8 3
(87)国際公開番号	WO2020/130925		100109726
(87)国際公開日	令和2年6月25日(2020.6.25)		弁理士 園田 吉隆
審査請求日	令和3年9月3日(2021.9.3)	(74)代理人	100161470
(31)優先権主張番号	62/782,571		弁理士 富樫 義孝
(32)優先日	平成30年12月20日(2018.12.20)	(74)代理人	100194294
(33)優先権主張国・地域又は機関	米国(US)		弁理士 石岡 利康
		(74)代理人	100194320
			弁理士 藤井 亮
		(74)代理人	100150670
			弁理士 小堀 晴美

最終頁に続く

(54)【発明の名称】 ピクチャにおける均一なセグメントスプリットを使用したビデオコーディングのための方法および装置

(57)【特許請求の範囲】

【請求項1】

ビットストリームから、いくつかのユニット（8）を含むピクチャ（10）を復号するための方法（700）であって、前記ピクチャがパーティション構造（13）によっていくつかの空間セグメント（11）に区分され、空間セグメントの数は2以上であり、前記方法は、

前記ビットストリーム中の1つまたは複数のコードワードを復号すること（710）と、

前記1つまたは複数のコードワードに基づいて、前記パーティション構造が均一であることを決定すること（720）と、

前記1つまたは複数のコードワードに基づいて空間セグメントの前記数を決定すること（730）と、

セグメントユニットサイズを決定すること（740）と、

前記1つまたは複数のコードワードから、前記ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出すること（750）と

を含み、

前記ピクチャ中の空間セグメントについての前記サイズおよび／またはロケーションを導出することが、第1の次元または方向における空間セグメントの数にわたる第1のループを含み、

セグメント化されるべき前記第1の次元または方向における残っているセグメントユニットの数が前記第1のループ内で計算される、方法（700）。

10

20

【請求項 2】

空間セグメントの前記数にわたる前記第 1 のループが、前記第 1 の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第 1 のループを含む、請求項 1 に記載の方法。

【請求項 3】

前記サイズおよび／またはロケーションを導出することが、前記第 1 の次元または方向以外の第 2 の次元または方向における空間セグメントの数にわたる第 2 のループをさらに含み、セグメント化されるべき前記第 2 の次元または方向における残っているセグメントユニットの数が前記第 2 のループ内で計算される、請求項 1 または 2 に記載の方法。

【請求項 4】

前記第 1 の次元または方向におけるセグメントサイズが行中のセグメントの幅であり、前記第 2 の次元または方向におけるセグメントサイズが列中のセグメントの高さであるか、または、

前記第 1 の次元または方向におけるセグメントサイズが列中のセグメントの高さであり、前記第 2 の次元または方向におけるセグメントサイズが行中のセグメントの幅である、請求項 3 に記載の方法。

【請求項 5】

空間セグメントの前記数にわたる前記第 2 のループが、前記第 2 の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第 2 のループを含む、請求項 3 または 4 に記載の方法。

【請求項 6】

前記第 1 の次元または方向が水平の次元または方向であり、前記第 2 の次元または方向が垂直の次元または方向であるか、または

前記第 1 の次元または方向が垂直の次元または方向であり、前記第 2 の次元または方向が水平の次元または方向である、請求項 3 から 5 のいずれか一項に記載の方法。

【請求項 7】

行中の前記セグメントの幅または列中の前記セグメントの高さが、指定された順序に従う、請求項 1 から 6 のいずれか一項に記載の方法。

【請求項 8】

ピクチャの行中の前記セグメントの幅または列中の前記セグメントの高さが、前記セグメントの前記幅または高さのための指定されたデフォルト順序に従う、請求項 1 から 7 のいずれか一項に記載の方法。

【請求項 9】

前記セグメントユニットサイズがコーディングツリーユニットのサイズに等しい、請求項 1 から 8 のいずれか一項に記載の方法。

【請求項 10】

前記セグメントユニットサイズがコーディングツリーユニットのサイズよりも大きい、請求項 1 から 8 のいずれか一項に記載の方法。

【請求項 11】

セグメントがタイルである、請求項 1 から 10 のいずれか一項に記載の方法。

【請求項 12】

前記セグメント (11) が他のセグメント (11) に対して独立しており、したがって、現在セグメント (15) 中の任意のユニット (16) のための任意のイントラ予測モードの前記導出は、前記現在セグメント (15) に属するユニット (17) における前に導出されたイントラ予測モードのみに依存し、異なるセグメント (14) に属するいかなるユニット (18) におけるいかなるイントラ予測モードにも依存しない、請求項 1 から 11 のいずれか一項に記載の方法。

【請求項 13】

いくつかのユニット (8) を含むピクチャ (10) をビットストリームに符号化するための方法 (800) であって、前記ピクチャが均一なパーティション構造 (13) によっ

10

20

30

40

50

ていくつかの空間セグメント（１１）に区分され、空間セグメントの数は２以上であり、前記方法は、

１つまたは複数のコードワードを前記ビットストリームに符号化することによって、前記パーティション構造（１３）が均一であるという情報を符号化すること（８１０）と、

１つまたは複数のコードワードを前記ビットストリームに符号化することによって、空間セグメントの前記数を符号化すること（８２０）と、

セグメントユニットサイズを決定すること（８３０）と、

前記ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションを前記ビットストリームに符号化すること（８４０）と

10

を含み、

前記ピクチャ中の空間セグメントについての前記サイズおよび／またはロケーションを導出することが、第１の次元または方向における空間セグメントの数にわたる第１のループを含み、

セグメント化されるべき前記第１の次元または方向における残っているセグメントユニットの数が前記第１のループ内で計算される、方法（８００）。

【請求項１４】

空間セグメントの前記数にわたる前記第１のループが、第１の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第１のループを含む、請求項１３に記載の方法。

20

【請求項１５】

前記サイズおよび／またはロケーションを導出することが、前記第１の次元または方向以外の第２の次元または方向における空間セグメントの数にわたる第２のループを含み、セグメント化されるべき前記第２の次元または方向における残っているセグメントユニットの数が前記第２のループ内で計算される、請求項１３または１４に記載の方法。

【請求項１６】

前記第１の次元または方向におけるセグメントサイズが行中のセグメントの幅であり、前記第２の次元または方向におけるセグメントサイズが列中のセグメントの高さであるか、または、

前記第１の次元または方向におけるセグメントサイズが列中のセグメントの高さであり、前記第２の次元または方向におけるセグメントサイズが行中のセグメントの幅である、請求項１５に記載の方法。

30

【請求項１７】

空間セグメントの前記数にわたる前記第２のループが、前記第２の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第２のループを含む、請求項１５または１６に記載の方法。

【請求項１８】

前記第１の次元または方向が水平の次元または方向であり、前記第２の次元または方向が垂直の次元または方向であるか、または

前記第１の次元または方向が垂直の次元または方向であり、前記第２の次元または方向が水平の次元または方向である、請求項１５から１７のいずれか一項に記載の方法。

40

【請求項１９】

セグメントがタイルである、請求項１３から１８のいずれか一項に記載の方法。

【請求項２０】

命令（１１４４）を含むコンピュータプログラム（１１４３）であって、前記命令（１１４４）が処理回路（１１０２）によって実行されたとき、前記処理回路（１１０２）に請求項１から１９のいずれか一項に記載の方法を実行させる、コンピュータプログラム（１１４３）。

【請求項２１】

ピクチャ（１０）を復号するための復号装置（１１００）であって、前記復号装置（１

50

100)は、

コンピュータ可読記憶媒体(1142)と、

前記コンピュータ可読記憶媒体に結合された処理回路(1102)とを備え、前記処理回路は、前記復号装置(1100)に請求項1から12のいずれか一項に記載の方法を実行させるように設定された、復号装置(1100)。

【請求項22】

ピクチャ(10)を符号化するための符号化装置(1100)であって、前記符号化装置(1100)は、

コンピュータ可読記憶媒体(1142)と、

前記コンピュータ可読記憶媒体に結合された処理回路(1102)とを備え、前記処理回路は、前記符号化装置(1100)に請求項13から19のいずれか一項に記載の方法を実行させるように設定された、符号化装置(1100)。

【発明の詳細な説明】

【技術分野】

【0001】

本開示はビデオエンコーディングおよびデコーディングに関する。

【背景技術】

【0002】

HEVCおよび次世代ビデオコーディング

高効率ビデオコーディング(HEVC)、別名H.265は、時間予測と空間予測の両方を利用する、ITU-TおよびMPEGによって標準化されたブロックベースのビデオコーデックである。空間予測は、現在ピクチャ内からのイントラ(I)予測を使用して達成される。時間予測は、前に復号された参照ピクチャからのブロックレベルのインター(P)予測または双方向インター(B)予測を使用して達成される。残差(residual)と呼ばれる、元のピクセルデータと予測ピクセルデータとの間の差は、周波数領域に変換され、量子化され、次いで、予測モードおよび動きベクトルなど、同じくエントロピーコーディングされる必要な予測パラメータと一緒に送信される前に、エントロピーコーディングされる。変換された残差を量子化することによって、ビデオのビットレートと品質との間のトレードオフが制御され得る。量子化のレベルは量子化パラメータ(QP)によって決定される。デコーダは、残差を取得するために、エントロピー復号、逆量子化、および逆方向変換(inverse transformation)を実行する。デコーダは、次いで、ピクチャを再構成するためにイントラ予測またはインター予測に残差を加算する。

【0003】

MPEGおよびITU-Tは、ジョイントビデオエクスプロラトリチーム(JVET)内でHEVCの後継に関する作業を行っている。開発中のこのビデオコーデックの名称はVCCである。

【0004】

スライス

HEVCにおけるスライスの概念は、ピクチャを個々にコーディングされたスライスに分割し、ここで各スライスは、コーディングツリーユニット(CTU)のユニット中でラスタ走査順序で読み取られる。異なるコーディングタイプが同じピクチャのスライスのために使用され得、すなわち、スライスは、Iスライス、PスライスまたはBスライスのいずれかであり得る。スライスの主要な目的は、データ損失の場合に再同期を可能にすることである。

【0005】

タイル

HEVCビデオコーディング規格は、ピクチャを矩形の空間的に独立した領域に分割する、タイルと呼ばれるツールを含む。タイルを使用すると、HEVCにおけるピクチャは、タイルが行と列との交差部である、サンプルの行と列とに区分され得る。HEVCにおけるタイルはCTU境界と常に整合させられる。

10

20

30

40

50

【0006】

図1は、ピクチャについて合計20個のタイルを生じる、4つのタイル行と5つのタイル列とを使用するタイル区分の例を示す。

【0007】

タイル構造は、行の厚さと列の幅とを指定することによってピクチャパラメータセット（PPS）中でシグナリングされる。個々の行および列は異なるサイズを有することができるが、区分は、それぞれ左から右におよび上部から下部に、常にピクチャ全体にわたって行われる。同じピクチャのタイル間の復号依存性はない。このことは、イントラ予測と、エントロピーコーディングのためのコンテキスト選択と、動きベクトル予測とを含む。1つの例外は、インループ（in-loop）フィルタ処理依存性が一般にタイル間で許容されることである。

【0008】

HEVCにおけるタイル構造を指定するために使用されるPPSシンタックスは、以下の表1に記載されている。フラグ、たとえば `tiles_enabled_flag` は、タイルが使用されるか否かを示す。フラグが設定された場合、タイル列およびタイル行の数が指定される。`uniform_spacing_flag` は、列幅および行高さが明示的にシグナリングされるかどうか、またはタイル境界線を均等に離間させるための事前規定された方法が使用されるべきかどうかを指定するフラグである。明示的なシグナリングが示された場合、列幅は1つつつシグナリングされ、その後行高さが1つつつシグナリングされる。そのような列幅および行高さはCTUユニット中でシグナリングされる。`loop_filter_across_tiles_enabled_flag` フラグは、タイル境界を横断するインループフィルタが、ピクチャ中のすべてのタイル境界についてオンまたはオフにされるかどうかを指定する。

<code>pic_parameter_set_rbsp() {</code>	ディスクリプタ
...	
<code>tiles_enabled_flag</code>	u(1)
...	
<code>if(tiles_enabled_flag) {</code>	
<code> num_tile_columns_minus1</code>	ue(v)
<code> num_tile_rows_minus1</code>	ue(v)
<code> uniform_spacing_flag</code>	u(1)
<code> if(!uniform_spacing_flag) {</code>	
<code> for(i = 0; i < num_tile_columns_minus1; i++)</code>	
<code> column_width_minus1[i]</code>	ue(v)
<code> for(i = 0; i < num_tile_rows_minus1; i++)</code>	
<code> row_height_minus1[i]</code>	ue(v)
<code> }</code>	
<code> loop_filter_across_tiles_enabled_flag</code>	u(1)
<code>}</code>	
...	

表1.HEVCにおける例示的なタイルシンタックス

【0009】

HEVCにおけるタイル構造を指定するためのセマンティクスについて、以下でさらに詳細に説明する。

【0010】

1に等しい `tiles_enabled_flag` は、PPSを参照する各ピクチャ中に2つ以上のタイルがあることを指定する。0に等しい `tiles_enabled_flag` は、PPSを参照する各ピクチャ中にただ1つのタイルがあることを指定する。

【0011】

`num_tile_columns_minus1 + 1` は、ピクチャを区分するタイル

列の数を指定する。`num__tile__columns__minus 1`は、両端値を含む、0から`PicWidthInCtbsY-1`の範囲内であるとする。存在しないとき、`num__tile__columns__minus 1`の値は0に等しいと推論される。

【0012】

`num__tile__rows__minus 1+1`は、ピクチャを区分するタイル行の数を指定する。`num__tile__rows__minus 1`は、両端値を含む、0から`PicHeightInCtbsY-1`の範囲内であるとする。存在しないとき、`num__tile__rows__minus 1`の値は0に等しいと推論される。

【0013】

`tiles__enabled__flag`が1に等しいとき、`num__tile__columns__minus 1`と`num__tile__rows__minus 1`の両方が0に等しくならないとする。

【0014】

1に等しい`uniform__spacing__flag`は、タイル列境界、および同様にタイル行境界がピクチャにわたって均一に分配されることを指定する。0に等しい`uniform__spacing__flag`は、タイル列境界、および同様にタイル行境界がピクチャにわたって均一に分配されないが、シンタックス要素`column__width__minus 1[i]`および`row__height__minus 1[i]`を使用して明示的にシグナリングされることを指定する。存在しないとき、`uniform__spacing__flag`の値は1に等しいと推論される。

【0015】

`column__width__minus 1[i]+1`はCTBのユニット中の*i*番目のタイル列の幅を指定する。

【0016】

`row__height__minus 1[i]+1`はCTBのユニット中の*i*番目のタイル行の高さを指定する。

【0017】

1に等しい`loop__filter__across__tiles__enabled__flag`は、インループフィルタ処理演算が、PPSを参照するピクチャ中のタイル境界を横断して実行され得ることを指定する。0に等しい`loop__filter__across__tiles__enabled__flag`は、インループフィルタ処理演算が、PPSを参照するピクチャ中のタイル境界を横断して実行されないことを指定する。インループフィルタ処理演算は、デブロッキングフィルタ演算と、サンプル適応オフセットフィルタ(sample adaptive offset filter)演算とを含む。存在しないとき、`loop__filter__across__tiles__enabled__flag`の値は1に等しいと推論される。

【0018】

VVCは、HEVCの場合と同様に、旧来のスライスを使用しないことが予想される。代わりに、タイルは、VRストリーミングを含む、ビデオサービスからの空間ランダムアクセスの需要の増加により、VVCにおいてより大きい役割を果たすことが予想される。

【0019】

タイルグループの概念は、最近のJVET会議において、現在のVVCドラフト中に含まれることが承認された。タイルグループは、各タイルのオーバーヘッドを低減するように複数のタイルをグループ化するために使用される。

【0020】

HEVCにおける均一なタイル区分

HEVCタイル区分では、すべてのタイル境界がCTUグリッドと整合させられる必要がある。そのことは、すべてのタイルが完全なCTUからなり、タイル中で許容される不完全なCTUのみが、ピクチャの右縁部または下縁部に位置するタイルであることを意味する。HEVCにおいて、1に等しいシンタックス要素`uniform__spacing`

10

20

30

40

50

`uniform_spacing_flag` は、タイル列境界、および同様にタイル行境界がピクチャにわたって均一に分配されることを指定する。しかしながら、この均一性はCTU細分性 (granularity) によって制限される。HEVCにおいて、両端値を含む、0から`num_tile_columns_minus1`にわたる`i`についてのリスト`colWidth[i]`は、コーディングツリーブロック (coding tree block) (CTB) のユニット中の`i`番目のタイル列の幅を指定し、以下の式 (A) のように導出される。

```
if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) -
            ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
```

10

【0021】

同様の式 (B) は、タイル行の高さ (`rowHeight[i]`) を決定するために使用される。

```
if( uniform_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) -
            ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
```

20

【0022】

フレキシブルタイルスプリッティング (flexible tile splitting)

J V E T - K 0 1 5 5 と、その後の J V E T - L 0 3 5 9 において導入されたフレキシブルタイルスプリッティングは、ピクチャを区分タイルにスプリットする機能を与え、ここで、各タイルの幅または高さは、CTUサイズよりも微細なユニットサイズの倍数である。フレキシブルタイルスプリッティングは、(ピクチャの右縁部および下縁部のみではなく) タイルごとの右縁部および下縁部における不完全なCTUの使用を許容する。図2A～図2Bは、タイルユニットサイズがCTUサイズの $1/4$ である、J V E T - L 0 3 5 9 の場合と同様にフレキシブルタイルスプリッティングを使用する 2×2 タイルセグメンテーションについての例を与える。

30

【0023】

図2A～図2Bでは、タイルは太い黒線によって示されており、CTUは薄い黒線によって示されている。図2Aは、ピクチャ中に20個のCTUがあるHEVC方法を示す。図2Bは、ピクチャ中に24個のCTUがあり、タイルユニットサイズが、破線グレー線によって示されている、CTUサイズの $1/4$ に等しい、J V E T - L 0 3 5 9 における提案された方法を示す。

【0024】

フレキシブルタイルスプリッティングは、負荷分散や、CTUサイズの倍数でないことが望ましいフェイスサイズをもつ 360° ビデオなどの適用例のために有用であり得る。

【0025】

本開示に関連がある、J V E T - L 0 3 5 9 からのシンタックスおよびセマンティクスを以下に記載し、ここで、イタリック体の部分は、L 0 3 5 9 において提案された、追加されたテキストである。

40

pic_parameter_set_rbsp() {	ディスクリプタ
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
transform_skip_enabled_flag	u(1)
tiles_enabled_flag	u(1)
if(tiles_enabled_flag) {	
tile_unit_size_idc	ue(v)
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
rbsp_trailing_bits()	
}	

10

20

【0026】

tile_unit_size_idcは、ルーマサンプル (luma sample) 中のタイルユニットブロックのサイズを指定する。変数**TileUnitSizeY**、変数**PicWidthInTileUnitsY**および変数**PicHeightInTileUnitsY**は以下のように導出される。

$TileUnitSizeY = \text{Max}(CtbSizeY \gg (tile_unit_size_idc), 8)$
(7-14)

$PicWidthInTileUnitsY = \text{Ceil}(pic_width_in_luma_samples \div TileUnitSizeY)$
(7-14)

$PicHeightInTileUnitsY = \text{Ceil}(pic_height_in_luma_samples \div TileUnitSizeY)$
(7-14)

If *tiles_enabled_flag* is equal to 1, the variables *PicWidthInCtbsY*, *PicHeightInCtbsY*, *PicSizeInCtbsY* are modified as follows:

for(*PicWidthInCtbsY* = 0, *i* = 0; *i* <= *num_tile_columns_minus1*; *i*++)

$PicWidthInCtbsY += \text{Ceil}(colWidth[i] * TileUnitSizeY \div CtbSizeY)$

(7-14)

for(*PicHeightInCtbsY* = 0, *j* = 0; *j* <= *num_tile_rows_minus1*; *j*++)

$PicHeightInCtbsY += \text{Ceil}(rowHeight[j] * TileUnitSizeY \div CtbSizeY)$

(7-14)

$PicSizeInCtbsY = PicWidthInCtbsY * PicHeightInCtbsY$

(7-14)

30

40

【0027】

50

`num_tile_columns_minus1 + 1`は、タイルユニットブロックのユニット中のピクチャを区分するタイル列の数を指定する。`num_tile_columns_minus1`は、両端値を含む、0から`PicWidthInTileUnitsY - 1`の範囲内とする。存在しないとき、`num_tile_columns_minus1`の値は0に等しいと推論される。

【0028】

`num_tile_rows_minus1 + 1`は、タイルユニットブロックのユニット中のピクチャを区分するタイル行の数を指定する。`num_tile_rows_minus1`は、両端値を含む、0から`PicHeightInTileUnitsY - 1`の範囲内とする。存在しないとき、`num_tile_rows_minus1`の値は0に等しいと推論される。

10

【0029】

変数`NumTilesInPic`は以下のように導出される。

$$NumTilesInPic = (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1)$$

【0030】

両端値を含む、0から`num_tile_columns_minus1`にわたる*i*についてのリスト`colWidth[i]`は、タイルユニットブロックのユニット中の*i*番目のタイル列の幅を指定し、以下のように導出される。

20

```

if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        posR = ((i + 1) * PicWidthInTileUnitsY) / (num_tile_columns_minus1 + 1)
    )
        posL = (i * PicWidthInTileUnitsY) / (num_tile_columns_minus1 + 1)
        colWidth[i] = min(PicWidthInTileUnitsY - posL, posR - posL)
else {
    colWidth[ num_tile_columns_minus1 ] = PicWidthInTileUnitsY
(6-3)
    for( i = 0; i < num_tile_columns_minus1; i++ ) {
        colWidth[i] = column_width_minus1[i] + 1
        colWidth[ num_tile_columns_minus1 ] -= colWidth[i]
    }
}

```

30

【0031】

両端値を含む、0から`num_tile_rows_minus1`にわたる*j*についてのリスト`rowHeight[j]`は、タイルユニットブロックのユニット中の*j*番目のタイル行の高さを指定し、以下のように導出される。

40

```

if( uniform_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        posB = ( ( j + 1 ) * PicHeightInTileUnitsY ) / ( num_tile_rows_minus1 + 1 )
        posT = ( j * PicHeightInTileUnitsY ) / ( num_tile_rows_minus1 + 1 )
        rowHeight[ j ] = min( PicHeightInTileUnitsY - posT, posB - posT )
else {
    rowHeight[ num_tile_rows_minus1 ] = PicHeightInTileUnitsY
(6-4)
    for( j = 0; j < num_tile_rows_minus1; j++ ) {
        rowHeight[ j ] = row_height_minus1[ j ] + 1
        rowHeight[ num_tile_rows_minus1 ] -= rowHeight[ j ]
    }
}

```

10

【 0 0 3 2 】

両端値を含む、0からnum_tile_columns_minus1+1にわたるiについてのリストTileColX[i]は、ルーマサンプルのユニット中のi番目のタイル列の左上ルーマサンプルのXロケーションを指定し、以下のように導出される。

20

```

for ( TileColX[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )

    TileColX[ i + 1 ] = TileColX[ i ] + colWidth[ i ] * TileUnitSizeY

```

【 0 0 3 3 】

両端値を含む、0からnum_tile_rows_minus1+1にわたるjについてのリストTileRowY[j]は、ルーマサンプルのユニット中のj番目のタイル行の左上ルーマサンプルのYロケーションを指定し、以下のように導出される。

30

```

for ( TileRowY[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )

    TileRowY[ j + 1 ] = TileRowY[ j ] + rowHeight[ j ] * TileUnitSizeY

```

【 0 0 3 4 】

セグメントグループ、セグメントおよびユニット

次に、セグメントグループ、セグメント、およびユニットについて説明する。本開示における実施形態は、異なる種類のピクチャ区分方式に適用され得、HEVCおよびVVCドラフトから知られているタイルパーティションに限られないので、セグメントという用語は、タイルよりも一般的な用語として使用される。本開示において、タイルはセグメントの一実施形態であるが、セグメントの他の実施形態もあり得る。

40

【 0 0 3 5 】

図3は、ビデオストリームのピクチャ10と、ユニット8、セグメント11およびセグメントグループ12へのピクチャの例示的な区分とを示す。図3(a)は、64個のユニット8からなるピクチャ10を示す。図3(b)は、16個のセグメント11からなる、同じピクチャ10のセグメントパーティション構造13を示す。パーティション構造13は一点鎖線によって示されている。各セグメント11はいくつかのユニットからなる。セグメントは、整数個の完全なユニット、または完全なユニットと部分ユニットとの組合せのいずれかからなることができる。いくつかのセグメントがセグメントグループを形成す

50

る。図3(c)は、8個のセグメントグループからなる、同じピクチャ10のセグメントグループ区分を示す。セグメントグループはラスタ走査順序のセグメントからなり得る。代替的に、セグメントグループは、一緒に矩形を形成する、セグメントの任意のグループからなり得る。代替的に、セグメントグループはセグメントの任意のサブセットからなり得る。

【0036】

図4は、一点鎖線が、ピクチャ10を4つのセグメントに分割するパーティション構造を示す、ピクチャ10を示す。図4はまた、3つのユニット16、17、18を示す。図に示されているように、2つのユニット16、17は1つの現在セグメント15に属し、1つのユニット18は異なる隣接セグメント14に属する。セグメントは他のセグメントに対して独立しており、そのことは、ユニットを復号するときに、セグメント境界がピクチャ境界と同様に処理されることを意味する。このことは、たとえば、イントラ予測モードの導出および量子化パラメータ値の導出など、復号中の要素の導出プロセスに影響を及ぼす。

【0037】

イントラモードは、現在の技術においてよく知られており、サンプル予測のための現在ピクチャの前に復号されたサンプルからの予測のみを使用するユニットのために使用され、シグナリングされる。現在ユニット16におけるイントラ予測モードの導出は、他の隣接ユニット17における前に導出されたイントラ予測モードに依存することが一般的である。セグメントが独立しているので、現在ユニット16におけるイントラ予測モードの導出は、現在セグメント15に属するユニット17における前に導出されたイントラ予測モードのみに依存することがあり、異なるセグメント14に属するいかなるユニット18におけるいかなるイントラ予測モードにも依存することがない。

【0038】

このことは、図4におけるパーティション構造が、異なるセグメント14中のユニット18におけるイントラ予測モードを、現在セグメント15中のユニット16のためのイントラ予測モードの導出のために利用不可能にすることを意味する。異なるセグメント中のあるユニット18と現在セグメント15中のユニット16とが同じセグメントに属していたのであれば、異なるセグメント中のあるユニット18におけるモードは、現在セグメント15中のユニット16におけるイントラ予測モードの導出のために多分に使用されていたであろうことに留意されたい。代わりに、セグメント境界は、現在セグメント15中のユニット16のためのピクチャ境界と同じ効果をイントラモード導出に対して有し得る。

【0039】

本開示の文脈では、セグメントはタイルまたはスライスであり得、セグメントグループはタイルグループであり得る。本開示において、「タイル」と「セグメント」という用語は互換的に使用され得る。いくつかの実施形態では、ユニットはCTUと等価であり得る。

【発明の概要】

【0040】

HEVC式(A)およびHEVC式(B)によると、タイル列幅(および行高さ)は

$$\lfloor (i+1) \cdot k \rfloor - \lfloor i \cdot k \rfloor$$

の形態で2つの項を互いから減算することによって計算され、ここで、 i は非負整数であり、 k は、 PicWidthInCtbsY に等しい分子と、 $\text{num_tile_columns_minus1} + 1$ に等しい分母とをもつ有理数である。 k が整数でないときの、そのような計算の出力は、 k および i の値に応じて

$$\lfloor k \rfloor \text{または} \lfloor k \rfloor$$

に等しくなり得る。この固有の特徴は、タイル列幅サイズおよびタイル行高さサイズに1個のCTUほど大きい変動を引き起こす。本開示の文脈では、そのような変動はタイルサ

イズリップル (tile size ripple) と呼ばれる (いくつかの例について以下の表 2 参照)。
このリップルのパターンは一定でなく、 k の値を与える、CTU におけるピクチャの幅
ならびにタイル列およびタイル行の数と、 i によって決定されるタイルグリッド上のタイ
ルの配置とに依存する。1 に等しい `uniform_spacing_flag` をもつ H
EVC タイル区分を使用するタイル列幅におけるリップルのいくつかの例が表 2 に示され
ている。同じ例はタイル行高さにも適用され得る。

PicWidthInCtbsY	num_tile_columns_minus1	k	colWidth[i], i=0, 1, ..., num_tile_columns_minus1
8	2	8/3	2, 3, 3 (より小さいタイルでグリ ッドを開始する) 2 3 3
10	3	10/4	2, 3, 2, 3 (混合されたタイルサイ ズ) 2 3 2 3
100	30	100/31	3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 3, 3, 3, 4, 3, 3, 3 , 4, 3, 3, 3, 4 (タイルサイズの不規則な混合)

表2

【0041】

表 2 は、1 に等しい `uniform_spacing_flag` と、`PicWidthInCtbsY` および `num_tile_columns_minus1` についての所与
の値とをもつ H EVC タイル区分を使用して計算されたタイル列幅を示す。`colWidth[ i ]` の値におけるリップルは可視である。

【0042】

最後のタイルサイズにおけるこの不一致は、最後のタイルサイズ値を予測するために入
力値およびコードの詳細を調査する必要があるので、望ましくない。リップル問題は水平
方向と垂直方向の両方において起こり得る。このことは、詳細を検査することなしに、タ
イルグリッド中の特定のタイルの正しい水平サイズおよび垂直サイズを決定することをよ
り困難にする。

【0043】

別の問題は、タイルの均一な離間のための H EVC の現在の実装において、ピクチャ中
のタイルのいくつかの行または列が削除された場合、ピクチャの残っている一部の内側の
タイル境界が新しいリップルパターンに応じて移動し得ることである。このことは、タイ
ル抽出プロセスにおけるタイルサイズおよびアドレスを再計算することを必要とする。表
500 として図 5 に示された例は、`uniform_spacing_flag` が 1 に等
しいとき、いくつかのタイルが元のピクチャから抽出された場合、H EVC 均一タイル間
隔 (uniform tile spacing) においてタイル列境界、したがってタイルサイズがどのよう
に変化し得るかを示す。元のピクチャ中のタイル境界と比較して、いくつかのタイル列を
削除した場合に変化するタイル境界は太線で示されている。

【0044】

図 5 における表 500 は、H EVC 均一タイル間隔を使用するタイル境界を示す。内部

タイル境界は、元のピクチャと、元のピクチャからいくつかのタイルを削除した後との間で変化する。元のピクチャについてのパラメータは以下のように設定されている。uniform_spacing_flag=1、PicWidthInTileUnitsY=10、num_tile_columns_minus1=3、num_tile_rows_minus1=1。

【0045】

J V E T − L 0 3 5 9 は、（ピクチャの右縁部および下縁部のみではなく）タイルごとの右縁部および下縁部において不完全なCTUを使用する可能性を与えることによって、より微細なタイルユニットサイズ細分性を可能にする、フレキシブルタイルスプリッティングを提案する。図2は、タイルユニットサイズがCTUサイズの1/4であり、uniform_spacing_flagが1に等しい、J V E T − L 0 3 5 9 の場合と同様にフレキシブルタイルスプリッティングを使用する、2×2タイルセグメンテーションについての例を与える。

【0046】

J V E T − L 0 3 5 9 において提案されているように、uniform_spacing_flagが1に等しい場合、タイル列の幅は以下の式を使用して決定される。

```

if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        posR = ( ( i + 1 ) * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )
    )
        posL = ( i * PicWidthInTileUnitsY ) / ( num_tile_columns_minus1 + 1 )
        colWidth[ i ] = min( PicWidthInTileUnitsY - posL, posR - posL )
where:
PicWidthInTileUnitsY = Ceil( pic_width_in_luma_samples ÷ TileUnitSizeY )

```

【0047】

同様の式が、タイル行の高さ (rowHeight [i]) を決定するために使用される。

【0048】

上記の実装形態は、J V E T − L 0 3 5 9 提案についてもリップル問題が存在することを示す。CTU細分性よりも微細な細分性をもつタイルサイズを規定する可能性がもたらされたことにより、ここで説明するように、一貫性のないタイルサイズのまた別のアーティファクトがもたらされる。L 0 3 5 9 において提案されたアルゴリズムを使用すると、タイルユニットサイズが、たとえば1/2または1/4に変化したとき、タイル区分は、いくつかのピクチャ幅およびタイルユニットサイズ値について一貫したままではない。理由は、L 0 3 5 9 における均一タイルスプリッティングについての提案された式は、異なるタイルユニットサイズを許容するが、タイルグリッド中のわずかにより大きいまたはより小さいタイルの構成を正則化しないことである。以下の表3は、ピクチャサイズが固定され、タイルユニットサイズが変更される、たとえば、L 0 3 5 9 において提案されたフレキシブルタイルスプリッティングを使用して1/2または1/4に分割されるケースについてのいくつかの例を示す。最後のタイル列幅は、より大きいタイル幅がピクチャの左側にあるときもあり、ピクチャの右側にあるときもあるようなタイルユニットサイズの変更の結果として、反転する。この予測不可能性は望ましくない。

10

20

30

40

50

(ルー マサン プルに おける)ピク チャ幅	タイル ユニッ トサイ ズ	PicWidthInTileUnitsY	num_tile _column s_minus1	k	colWidth[i], i=0, 1, ..., num_til e_columns_minus1 (タイルユニット サイズにおける)	colWidth[i], i=0, 1, ..., num_til e_columns_minus1 (ルーマサンプル における)
2160	32	Ceil(67.5) = 68	1	68/2	34, 34	1072, 1088 より小さいタイル でグリッドを開始 する
	16	135	1	135/2	67,68	1088, 1072 より大きいタイル でグリッドを開始 する
1080	16	Ceil(67.5) = 68	1	68/2	34, 34	544, 536 より大きいタイル でグリッドを開始 する
	8	135	1	135/2	67,68	536, 544 より小さいタイル でグリッドを開始 する
480	128	Ceil(3.75) = 4	1	4/2	2, 2	256, 224 より大きいタイル でグリッドを開始 する
	32	15	1	15/2	7, 8	224, 256 より小さいタイル でグリッドを開始 する

表3

【0049】

表3は、uniform_spacing_flagが1に設定されたときに、L0359の場合と同様にフレキシブルタイルスプリットングを使用して計算されたタイル列幅を示す。

【0050】

タイルサイズに対する丸め効果の不一致を編成するためのルールにより、コードおよび入力値の詳細な調査の必要なしに、最後のタイルサイズを決定することが容易になると結論することができる。タイルサイズリップルを正則化するためのシステムティックな手法はまた、元のピクチャと、元のピクチャからのタイルのサブセットとについての一貫したタイル区分結果を与える。

【0051】

分割の制限された細分性に応じて、セグメントサイズが厳密になることができないとき、好ましいオーダーをもつセグメントのサイズでリップルを正則化する実施形態が本明細書で提案されている。提案された実施形態では、丸め効果に応じて、わずかに小さいサイズまたはわずかに大きいサイズをもつタイルが好ましい構成を有する。一実施形態では、左から右の走査方向におけるタイル幅は同じにとどまるか、または減少するのみである。本明細書で開示する、提案された実施形態は、タイルグリッド中のタイル列幅および／またはタイル行高さ、または個々のタイルの幅および高さに適用され得る。一実施形態では、（所与の細分性および丸め効果により）わずかに大きいサイズをもつタイルはピクチャ

10

20

30

40

50

の左上部に位置し、わずかに小さいサイズをもつタイルはピクチャの右下部に位置する。

【0052】

一態様では、ビットストリームからピクチャを復号するための方法が提供される。ピクチャはいくつかのユニットを含み、ピクチャはパーティション構造によっていくつかの空間セグメントに区分され、空間セグメントの数は2以上である。一実施形態では、本方法は、ビットストリーム中の1つまたは複数のコードワードを復号することを含む。本方法はまた、1つまたは複数のコードワードに基づいて、パーティション構造が均一であることを決定することを含む。本方法はまた、1つまたは複数のコードワードに基づいて空間セグメントの数を決定することを含む。本方法はまた、セグメントユニットサイズを決定することを含む。本方法はまた、1つまたは複数のコードワードから、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することを含む。ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出するステップは、第1の次元または方向における空間セグメントの数にわたる第1のループを含む。さらに、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数は第1のループ内で計算される。

10

【0053】

一態様では、いくつかのユニットを含むピクチャをビットストリームに符号化するための方法が提供される。ピクチャは、均一なパーティション構造によっていくつかの空間セグメントに区分され、空間セグメントの数は2以上である。本方法は、1つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造が均一であるという情報を符号化することを含む。本方法はまた、1つまたは複数のコードワードをビットストリームに符号化することによって、空間セグメントの数を符号化することを含む。本方法はまた、セグメントユニットサイズを決定することを含む。本方法はまた、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化することを含む。ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出するステップは、第1の次元または方向における空間セグメントの数にわたる第1のループを含む。さらに、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数は第1のループ内で計算される。

20

【0054】

一態様では、ピクチャを符号化するための方法が提供され、本方法は、ピクチャをいくつかの均一な空間セグメントに分割するパーティション構造を規定することであって、各空間セグメントが少なくとも1つのユニットを含み、セグメントサイズの導出が、第1の次元または方向における空間セグメントの数にわたる第1のループ、および第2の次元または方向における空間セグメントの数にわたる第2のループ中で実行され、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数が第1のループ内で計算され、セグメント化されるべき第2の次元または方向における残っているセグメントユニットの数が第2のループ内で計算される、パーティション構造を規定することを含む。本方法はまた、複数のコーディングされた空間セグメントを生成するためのパーティション構造に従って、複数の空間セグメントを符号化することであって、各々のコーディングされた空間セグメントがパーティション構造の空間セグメントのうちの1つに対応し、各々のコーディングされた空間セグメントは独立しており、したがって、第1の空間セグメントの第1のユニットのための任意のイントラ予測モードの導出は、第1の空間セグメントの第2のユニットのための導出されたイントラ予測モードに依存し、パーティション構造の他の空間セグメントのユニットのためのいかなるイントラ予測モードとも無関係である、複数の空間セグメントを符号化することを含む。本方法は、複数のコーディングされた空間セグメントと、ピクチャを複数の空間セグメントに分割するために使用される均一なパーティション構造を示す情報とを含む、ビットストリームを生成することをさらに含む。

30

40

【0055】

50

一態様では、処理回路によって実行されたとき、処理回路に本明細書で開示する方法のうちのいずれか1つを実行させる命令を含む、コンピュータプログラムが提供される。一態様では、コンピュータプログラムを含んでいるキャリアも提供される。キャリアは、電子信号、光信号、無線信号、およびコンピュータ可読記憶媒体のうちの1つである。

【0056】

別の態様では、ビットストリームからピクチャを復号するための復号装置が提供される。ピクチャはいくつかのユニットを含み、ピクチャはパーティション構造によっていくつかの空間セグメントに区分され、空間セグメントの数は2以上である。本復号装置は、ビットストリーム中の1つまたは複数のコードワードを復号するように設定される。本復号装置はまた、1つまたは複数のコードワードに基づいて、パーティション構造が均一であることを決定するように設定される。本復号装置はまた、1つまたは複数のコードワードに基づいて空間セグメントの数を決定するように設定される。本復号装置はまた、セグメントユニットサイズを決定するように設定される。本復号装置はまた、1つまたは複数のコードワードから、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出するように設定される。ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することは、第1の次元または方向における空間セグメントの数にわたる第1のループを含む。さらに、本復号装置は、第1のループ内で、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数を計算するように設定される。いくつかの実施形態では、本復号装置は、コンピュータ可読記憶媒体と、コンピュータ可読記憶媒体に結合された処理回路とを含み、処理回路は、本復号装置に本明細書で開示するいずれかの復号方法を実行させるように設定される。

【0057】

別の態様では、ピクチャをビットストリームに符号化するための符号化装置が提供される。ピクチャはいくつかのユニットを含み、ピクチャは均一なパーティション構造によっていくつかの空間セグメントに区分され、空間セグメントの数は2以上である。本符号化装置は、1つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造が均一であるという情報を符号化するように設定される。本符号化装置はまた、1つまたは複数のコードワードをビットストリームに符号化することによって空間セグメントの数を符号化するように設定される。本符号化装置はまた、セグメントユニットサイズを決定するように設定される。本符号化装置は、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化するようにさらに設定される。ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することは、第1の次元または方向における空間セグメントの数にわたる第1のループを含む。さらに、本符号化装置は、第1のループ内で、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数を計算するようにさらに設定される。

【0058】

別の態様では、ピクチャを符号化するための符号化装置が提供され、本符号化装置は、ピクチャをいくつかの均一な空間セグメントに分割するパーティション構造を規定するように設定され、各空間セグメントは少なくとも1つのユニットを含み、セグメントサイズの導出は、第1の次元または方向における空間セグメントの数にわたる第1のループおよび第2の次元または方向における空間セグメントの数にわたる第2のループ中で実行され、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数は第1のループ内で計算され、セグメント化されるべき第2の次元または方向における残っているセグメントユニットの数は第2のループ内で計算される。本符号化装置はまた、複数のコーディングされた空間セグメントを生成するためのパーティション構造に従って、複数の空間セグメントを符号化するように設定され、各々のコーディングされた空間セグメントはパーティション構造の空間セグメントのうちの1つに対応し、各々のコーディングされた空間セグメントは独立しており、したがって、第1の空間セグメントの第1のユニットのための任意のイントラ予測モードの導出は、第1の空間セグメントの第2

のユニットのための導出されたイントラ予測モードに依存し、パーティション構造の他の空間セグメントのユニットのためのいかなるイントラ予測モードとも無関係である。本符号化装置は、複数のコーディングされた空間セグメントと、ピクチャを複数の空間セグメントに分割するために使用される均一なパーティション構造を示す情報とを含む、ビットストリームを生成するようにさらに設定される。

【0059】

別の態様では、ピクチャを符号化するための符号化装置が提供され、本符号化装置は、コンピュータ可読記憶媒体と、コンピュータ可読記憶媒体に結合された処理回路とを備える。処理回路は、本符号化装置に本明細書で開示する符号化方法のうちのいずれか1つを実行させるように設定される。

【0060】

本開示の実施形態は、現在タイルの幅を前のタイルとは無関係に保つが、残っている数のタイルに区分されることになる（列または行における）残っている数のユニット上でのみである。結果として、タイルの均一な離間のための実施形態は、タイルが抽出された場合に、抽出されたおおよび残っているタイルについてタイル区分をそのままに保つ。これにより、ピクチャスプリットまたはタイル抽出の場合にタイルサイズを再計算する必要がなくなる。タイルの均一な離間のためのHEVCの現在の実装では、ピクチャ中のタイルのいくつかの行または列が削除された場合、リップルパターンが変化することがあり、ピクチャの残っている一部の内側のタイル境界が新しいリップルパターンに応じて移動することがある。

【0061】

一貫性は、実施形態の別の利点である。タイルの均一な離間のための実施形態は、入力パラメータまたはコードの詳細な調査なしに、タイルサイズ（たとえばタイル列幅およびタイル行高さ）を編成し、タイルサイズの最終構成についてのルールを与える。たとえば、一実施形態では、タイルユニットにおけるピクチャ幅／高さがタイル列／行の数によって割り切れないとき、より大きいサイズに丸められたすべてのタイルはピクチャの左／上部に見つけられる。実施形態はまた、タイルユニットサイズの変更の場合、均一タイル間隔におけるタイルサイズのリップルを調整する。

【0062】

提案された方法はまた、HEVC、およびJVET-L0359において提案されたフレキシブルタイルスプリットと比較して、複雑さを低減する。提案された方法は、乗算を有さず、タイル列の幅とタイル行の高さとを決定するための繰り返し当たり1回のみの除算を有するが、HEVC方法は、繰り返し当たり2回の乗算と2回の除算とを使用する。

【0063】

本明細書に組み込まれ、本明細書の一部を形成する添付の図面は、様々な実施形態を示す。

【図面の簡単な説明】

【0064】

【図1】一実施形態による、タイル区分の例を示す図である。

【図2A】HEVCタイル区分を使用したタイルセグメンテーションを示す図である。

【図2B】フレキシブルタイルスプリッティングを使用したタイルセグメンテーションを示す図である。

【図3】ビデオストリームのピクチャと、例示的な区分とを示す図である。

【図4】いくつかの実施形態による、ピクチャを示す図である。

【図5】いくつかの実施形態による、テーブルを示す図である。

【図6】いくつかの実施形態による、テーブルを示す図である。

【図7】一実施形態による、プロセスを示すフローチャートを示す図である。

【図8】一実施形態による、プロセスを示すフローチャートを示す図である。

【図9】一実施形態による、デコーダの機能ユニットを示すダイアグラムである。

【図10】一実施形態による、エンコーダの機能ユニットを示すダイアグラムである。

10

20

30

40

50

【図 1 1】いくつかの実施形態による、装置のブロック図である。

【図 1 2】一実施形態による、プロセスを示すフローチャートを示す図である。

【発明を実施するための形態】

【0065】

実施形態について説明するために、以下の用語を使用した。

【0066】

算術演算子「 $\div$ 」は、0 に向かう結果の切り捨てがある、整数除算のために使用される。たとえば、 $7 \div 4$ および $-7 \div -4$ は切り捨てられて 1 になり、 $-7 \div 4$ および $7 \div -4$ は切り捨てられて -1 になる。

【0067】

算術演算子「 $\div$ 」は、切り捨てまたは丸めが意図されない、数式における除算のために使用される。

【0068】

$\text{Ceil}(x)$ は、 x よりも大きいかまたはそれに等しい、最も小さい整数を与える。

【0069】

$\text{Floor}(x)$ は、 x よりも小さいかまたはそれに等しい、最も大きい整数を与える。

【0070】

「タイル列幅」という用語と「タイル幅」という用語は互換的に使用され、そのことは、実施形態が、タイル列幅がタイルグリッド中で計算されているとき、または（たとえばタイルグリッドがない場合）タイル幅が個々に計算されているときに適用され得ることを意味する。

【0071】

「タイル行高さ」という用語と「タイル高さ」という用語は互換的に使用され、そのことは、実施形態が、タイル行高さがタイルグリッド中で計算されているとき、または（たとえばタイルグリッドがない場合）タイル高さが個々に計算されているときに適用され得ることを意味する。

【0072】

`uniform_spacing_flag = 1` についてのタイルの幅（および同等に高さ）を規定するための実施形態は以下の 4 つの要素を有する。

【0073】

(1) 各タイルの幅を規定するためのループにおいて、タイルの幅は、タイルにまだ割り振られていない行におけるユニットサイズの数意味する、利用可能な残っているユニットサイズを使用して規定される。

【0074】

(2) 残っているユニットサイズの数、次いで、行における残っているタイルの数によって除算される。

【0075】

(3) 得られたタイルサイズは、タイルの幅を整えるための所与のルールに従って、より大きいまたはより小さい整数 ($\text{Ceil}()$ 関数または $\text{Floor}()$ 関数) に向かって丸められる。

【0076】

(4) 各タイルの幅を規定するためのループの繰り返しごとに、残っているユニットサイズ（たとえば CTU）の数を再計算するための随意の要素。

【0077】

同じ要素は、タイルの高さを規定することに適用される。

【0078】

本明細書で開示する、実施形態は、ビットストリームからのピクチャ 10 を復号するためのデコーダ方法について説明しており、本方法は、ビットストリームからのピクチャ中のセグメント（たとえば、ピクチャ中のすべてのセグメント）についてのサイズおよび／またはロケーションを導出することを含み、ピクチャ 10 はいくつかのユニット 8 からな

10

20

30

40

50

り、パーティション構造 13 は、ピクチャを少なくとも 2 つのセグメント 11 に区分し、デコーダは、ビットストリーム中の 1 つまたは複数のコードワードを復号することによって、空間セグメンテーションが均一であることを決定し、デコーダは、ビットストリーム中の 1 つまたは複数のコードワードを復号することによって、空間セグメントの数を決定し、デコーダはタイルユニットサイズを決定し、セグメントを均一な幅または高さに区分することは、空間セグメントの数とは無関係な固定のリップルパターンに従い、セグメントサイズの導出は、セグメントの数にわたるループ中で行われ、ループ内部で、セグメント化されるべき残っているタイルユニットの数が計算される。残っているセグメントの数もループ内で計算され得る。

【0079】

一例では、実施形態は、以下の H E V C のライン

```
if( uniform_spacing_flag )
  for( i = 0; i <= num_tile_columns_minus1; i++ )
    colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) -
      ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
```

を以下のラインと置き換える。

```
if( uniform_spacing_flag ){
    A = PicWidthInCtbsY
    B = num_tile_columns_minus1 + 1
    for( i = 0; i <= num_tile_columns_minus1; i++ ){
        colWidth[ i ] = Ceil( A ÷ B )
        A -= colWidth[ i ]
        B -= 1
    }
}
```

【0080】

提案された方法はまた、H E V C、および J V E T - L 0 3 5 9 において提案されたフレキシブルタイルスプリットと比較して、複雑さを低減する。提案された方法は、乗算を有さず、タイル列の幅とタイル行の高さを決定するための繰り返し当たり 1 回のみの除算を有するが、H E V C 方法は、繰り返し当たり 2 回の乗算と 2 回の除算とを使用する。

【0081】

実施形態 1. 単調リップル

一実施形態では、行中のセグメントの幅または列中のセグメントの高さは、事前規定された走査方向に従う昇順でもなく、降順でもない。セグメントはピクチャ中のタイルであり得、したがって、タイルサイズは走査方向において単調である（上昇もせず、下降もしない）。たとえば、タイル幅が左から右の走査方向において上昇しないことは、タイルの幅が、第 1 のタイルの左側に空間的に位置する同じ行上の別のタイルの幅よりも大きくな

【0082】

第 1 の実施形態では、H E V C の上の上昇しないタイル列幅（左から右への走査方向）およびタイル行高さ（上から下への走査方向）について、以下の H E V C 関数

10

20

30

40

50

```

if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) -
            ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
and
if( uniform_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) -
            ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )

```

10

は、以下の関数と置き換えられる。

```

if( uniform_spacing_flag ) {
    A = PicWidthInCtbsY
    B = num_tile_columns_minus1 + 1
    for( i = 0; i <= num_tile_columns_minus1; i++ ) {
        colWidth[ i ] = Ceil( A ÷ B)
        A -= colWidth[ i ]
        B -= 1
    }
}
and
if( uniform_spacing_flag ) {
    C = PicHeightInCtbsY
    D = num_tile_rows_minus1 + 1
    for( j = 0; j <= num_tile_rows_minus1; j++ ) {
        rowHeight[ j ] = Ceil( C ÷ D)
        C -= rowHeight[ j ]
        D -= 1
    }
}

```

20

30

【0083】

colWidthリストおよびrowHeightリスト中の得られた値はルーマコーディングツリーブロックの単位である。たとえば、CTUサイズが 128×128 に等しい場合、値は128個のルーマサンプルの単位であり、したがって、2の値は256個のルーマサンプルを意味する。

【0084】

colWidthリストおよびrowHeightリストは、次いで、ピクチャ中のブロックの走査順序を決定するためにデコーダによって使用される。ブロックデータが復号されるとき、ブロックの空間位置はcolWidthリストおよびrowHeightリスト中の値に基づく。デコーダは、colWidthおよびrowHeightの値を使用して、タイル走査アドレスからラスタ走査アドレスへの変換リスト、およびその逆の変換リストを構築し得る。デコーダは、次いで、ブロックの空間位置を決定するために復号中に変換リストを使用し得る。HEVCでは、変換リストが使用され、CtbAddrRsToTsおよびCtbAddrTsToRsと呼ばれる。

40

【0085】

図6は、HEVCのためのタイルサイズ割振りと第1の実施形態の上記の例示的な実装形態とを比較する表600を示す。表600に示されているように、実施形態では、タイ

50

ル列がピクチャの左側もしくは右側または両側から削除されたとき、タイル境界は元のピクチャと同じに保たれる。丸め効果は、タイル列幅を決定するために、タイルにわたるループの繰り返しごとに、 $\text{Ceil}()$ 関数を使用し、残っているタイルユニットの数を再計算して、すべての場合において、ピクチャの左側にわずかに広いタイルを置くように定められる。

【0086】

表600は、内部タイル境界がそのままに保たれる間、HEVC均一タイル間隔を使用して元のピクチャからいくつかのタイル列を削除した後に内部タイル境界がどのように変化するかを示す。元のピクチャについてのパラメータは以下のように設定されている。 $\text{uniform_spacing_flag}=1$ 、 $\text{PicWidthInCtbsY}=10$ 、 $\text{num_tile_columns_minus1}=3$ 、 $\text{num_tile_rows_minus1}=1$ 。

【0087】

以下は、CTUサイズよりも小さいタイルサイズ細分性をサポートする、JVET-L0359において提案された式の上に構築された、第1の実施形態の別の例である。以下の式変更はJVET-L0359の上に提案されており、

```

if( uniform_spacing_flag ) {
    A = PicWidthInTileUnitsY
    B = num_tile_columns_minus1 + 1
    for( i = 0; i <= num_tile_columns_minus1; i++ ) {
        posR = ((i + 1) * PicWidthInTileUnitsY) / (num_tile_columns_minus1 + 1)
        posL = (i * PicWidthInTileUnitsY) / (num_tile_columns_minus1 + 1)
        colWidth[i] = min(PicWidthInTileUnitsY - posL, posR - posL)
        colWidth[i] = Ceil( A ÷ B)
        A -= colWidth[i]
        B -= 1
    }
}

```

where:

$\text{PicWidthInTileUnitsY} = \text{Ceil}(\text{pic_width_in_luma_samples} \div \text{TileUnitSizeY})$

タイル行高さについて同等に、以下の変更が提案されている。

10

20

30

40

50

```

if( uniform_spacing_flag ) {
    C = PicHeightInTileUnitsY
    D = num_tile_rows_minus1 + 1
    for( j = 0; j <= num_tile_rows_minus1; j++ ) {
        posB = ((i + 1) * PicHeightInTileUnitsY) / (num_tile_rows_minus1 + 1)
        posT = (i * PicHeightInTileUnitsY) / (num_tile_rows_minus1 + 1)
        rowHeight[ i ] = min( PicHeightInTileUnitsY - posT, posB - posT )
        rowHeight[ j ] = Ceil( C ÷ D )
        C -= rowHeight[ j ]
        D -= 1
    }
}

```

where:

$\text{PicHeightInTileUnitsY} = \text{Ceil}(\text{pic_height_in_luma_samples} \div \text{TileUnitSizeY})$

【0088】

colWidthリストおよび**rowHeight**リスト中の得られた値はルーマタイルユニットの単位である。たとえば、タイルユニットサイズが 32×32 に等しい場合、値は32個のルーマサンプルの単位であり、したがって、2の値は64個のルーマサンプルを意味する。

【0089】

表4は、異なるピクチャ幅値について、J V E T - L 0 3 5 9と一実施形態との間でタイルの幅についての結果を比較している。J V E T - L 0 3 5 9についての結果は、タイルユニットサイズが変化したときに、タイル列の幅(**colWidth[i]**)の不一致を示すが、その実施形態は、わずかに大きいタイルにピクチャの左側に置かれるように一貫して優先度を付けるので、その実施形態は、タイルユニットサイズが変化したときに、タイル列の一貫した幅(**colWidth[i]**)を与える。

10

20

30

40

50

(ルー マサン ブルに おける)ピク チャ幅	タイル ユニッ トサイ ズ	PicWidthInT ileUnitsY	JVET-L0359		第1の実施形態	
			colWidth[i], i=0, 1 (タイルユニッ トサイズにお ける)	colWidth[i], i=0, 1 (ルーマサンプ ルにおける)	colWidth[i], i=0, 1 (タイルユニッ トサイズにお ける)	colWidth[i], i=0, 1 (ルーマサンプ ルにおける)
2160	32	Ceil(67.5) = 68	34, 34	1072, 1088 より小さいタ イルでグリッド を開始する	34, 34	1088, 1072 より大きいタ イルでグリッド を開始する
	16	135	67,68	1088, 1072 より大きいタ イルでグリッド を開始する	68,67	1088, 1072 より大きいタ イルでグリッド を開始する
1080	16	Ceil(67.5) = 68	34, 34	544, 536 より大きいタ イルでグリッド を開始する	34, 34	544, 536 より大きいタ イルでグリッド を開始する
	8	135	67,68	536, 544 より小さいタ イルでグリッド を開始する	68,67	544, 536 より大きいタ イルでグリッド を開始する
480	128	Ceil(3.75) = 4	2, 2	256, 224 より大きいタ イルでグリッド を開始する	2, 2	256, 224 より大きいタ イルでグリッド を開始する
	32	15	7, 8	224, 256 より小さいタ イルでグリッド を開始する	8, 7	256, 224 より大きいタ イルでグリッド を開始する

表4

【0090】

表4は、第1の実施形態と比較してJVET-L0359の場合と同様にフレキシブル
タイルスプリッティングを使用して、uniform_spacing_flag=1、
num_tile_columns_minus1=1のときに計算されたタイル列幅を
示す。

【0091】

第1の実施形態では、デコーダは以下のステップのすべてまたはサブセットを実行し得
る。

【0092】

1. ビットストリーム中の1つまたは複数のシンタックス要素から、1つまたは複数の
ピクチャが2つ以上のセグメントに区分されるという情報を復号する。シンタックスは、
好ましくは、ピクチャパラメータセット中に位置する。シンタックスは、セグメント行の
数を示す数Rと、セグメント列の数を示す数Cとを指定し得る。

【0093】

2. ビットストリーム中の1つまたは複数のシンタックス要素から、空間セグメンテー
ションが均一であるという情報を復号する。シンタックスは、好ましくは、ピクチャパラ
メータセット中に位置する。シンタックスは、空間セグメンテーションが均一であるか否
かを指定する1ビットフラグからなり得る。

【0094】

3. 1つまたは複数のシンタックス要素から、または事前規定されたセグメントユニッ

10

20

30

40

50

トサイズを使用することによって、セグメントユニットサイズ S を決定する。事前規定されたユニットサイズが使用される場合、事前規定されたユニットサイズは、 CTU のサイズ、または 1 つの次元における CTU のサイズに等しくなり得る。たとえば、 CTU サイズが 128×128 に等しい場合、セグメントユニットサイズ S は 128 （または 128×128 ）に等しくなり得る。

【0095】

4. セグメントユニットの数におけるピクチャのサイズを計算する。計算は、水平サイズ (HS) が、セグメントユニットサイズ S によって除算された、ルーマサンプル中のピクチャ幅に等しく設定されるように、ピクチャの高さと幅とについて別個に行われ得る。垂直サイズ (VS) は、セグメントユニットサイズによって除算された、ルーマサンプル中のピクチャ高さに等しく設定され得る。

10

【0096】

5. ピクチャ中のセグメントの数およびセグメントユニットサイズ S からセグメント（たとえば、すべてのセグメント）の幅および高さを導出する。導出は、まだセグメント化されていないセグメントの数と、セグメントユニットの数におけるまだセグメント化されていないピクチャのサイズとがループの各繰り返しにおいて更新される、セグメントの数にわたるループ中で行われる。

【0097】

導出は 2 つの別個のループ中で行われ得る。

a. 以下のサブステップによって、タイルユニットの数におけるピクチャ幅およびセグメント列の数 C からセグメント列幅を導出する。

20

i. まだセグメント化されていないピクチャ幅 (A) を値 HS に等しく設定する。

ii. まだセグメント化されていないセグメント列の数 (B) を値 C に等しく設定する。

iii. 繰り返し当たり 1 つの幅値が導出され、まだセグメント化されていないセグメント列の数 (B) とまだセグメント化されていないピクチャ幅 (A) の両方が各繰り返し中に更新される、ループ中で列幅を導出する。繰り返しは C 回実行され得る。

1. 導出された列幅 W は $Ce\ i\ l\ (A \div B)$ に等しく設定され得る。

2. 変数 A は、次いで、 $A - W$ に更新され得、変数 B は $B - 1$ に更新され得る。

b. 以下のサブステップによって、タイルユニットの数におけるピクチャ高さおよびセグメント列の数 R からセグメント行高さを導出する。

30

i. まだセグメント化されていないピクチャ高さ (A) を値 VS に等しく設定する。

ii. まだセグメント化されていないセグメント行の数 (B) を値 R に等しく設定する。

iii. 繰り返し当たり 1 つの高さ値が導出され、まだセグメント化されていないセグメント行の数 (B) とまだセグメント化されていないピクチャ高さ (A) の両方が各繰り返し中に更新される、ループ中で行高さを導出する。繰り返しは R 回実行され得る。

1. 導出された行高さ H は $Ce\ i\ l\ (A \div B)$ に等しく設定され得る。

2. 変数 A は、次いで、 $A - H$ に更新され得、変数 B は $B - 1$ に更新され得る。

【0098】

6. 導出されたセグメント幅および導出されたセグメント高さを使用して、現在ブロックについての空間ロケーションを導出する。

40

【0099】

7. 導出された空間ロケーションを使用して現在ブロックを復号する。現在ブロックのための復号されたサンプル値を、導出された空間ロケーションに対応するメモリ位置におけるメモリに記憶する。

【0100】

この実施形態の変形態では、関数 $Ce\ i\ l\ ()$ は、水平方向のみもしくは垂直方向のみ、または両方の方向において関数 $F\ l\ o\ o\ r\ ()$ によって置き換えられ得る。

【0101】

この実施形態の変形態では、フラグは 2 つの関数 $Ce\ i\ l\ ()$ と $F\ l\ o\ o\ r\ ()$ との間で選定し得る。いくつかの実施形態では、水平方向および垂直方向のための 2 つの独立し

50

たフラグがあり得る。

【0102】

実施形態2．指定された順序

第2の実施形態では、行中のセグメントの幅または列中のセグメントの高さは、指定された順序に従う。この指定された順序は、ビットのシーケンスの形態のテンプレートパターンを使用してビットストリーム中でシグナリングされ得る。セグメントはピクチャ中のタイルであり得、したがって、この実施形態では、タイルサイズに関する好ましいリップルの形状が指定される。たとえば、すべてのわずかに広いタイルはピクチャの左側にある。このことは、タイル幅（または高さ）を指定するためにタイルにわたるループの繰り返しごとに $Ceil()$ 関数または $Floor()$ 関数を指定する、ビットのシーケンスの形態で表現される、テンプレートパターンによって行われ得る。テンプレートビットシーケンス中で、関数 $Ceil()$ は1によって表され得、関数 $Floor()$ は0によって表され得る。例として、テンプレート110は、タイルサイズを指定するためのループの、最初の2つのタイル中で $Ceil()$ 関数を指定し、第3の繰り返し中で関数 $Floor()$ を指定するであろう。タイルの数がテンプレートビットシーケンスの長さよりも大きい場合、パターンは周期的に繰り返され得る。

【0103】

実施形態3．バイナリタイルスプリット (binary tile split)

第3の実施形態では、セグメント幅または高さの明示的な順序付けをもつバイナリセグメントスプリットがピクチャの幅または高さに適用される。セグメントはピクチャ中のタイルであり得、したがって、ピクチャを均一なタイルサイズをもつ 2^n 個のタイルに区分するために、バイナリ区分が使用される。（第2の実施形態の場合と同様の）事前規定されたリップルパターンが、この実施形態と組み合わせて使用され得る。バイナリ区分は、アルゴリズムが、指定された数のセグメントに到達するまで、または規定された数のステップまでのみ、階層的に継続し得、セグメント分割の残りは他の方法によって実行される。例として、合計サイズが第1のステップにおいて2によって除算され、そこで、より小さいまたはより大きいセグメントを左側または右側に割り当てるように指定される。後続のステップの各々において、左側セグメントと右側セグメントの各々は、次のバイナリ区分レベル、およびより小さいまたはより大きい可能性があるセグメントの位置についての指定されたルールを使用して、2つの部分に分割される。

【0104】

実施形態4．リップルのデフォルト順序付け

第4の実施形態では、ピクチャの行中のセグメントの幅または列中のセグメントの高さは、セグメントの幅または高さのための指定されたデフォルト順序に従う。セグメントはピクチャ中のタイルであり得、したがって、セグメントサイズにおけるリップルを管理するためのデフォルトの好ましい順序付けが規定される。デフォルトリップルパターンはビットストリーム中でシグナリングされ得る。この実施形態の変形態では、デフォルトパターンは上書きされ得る。この実施形態の別の変形態では、フラグは、デフォルト順序付けが使用されているか、または別の指定された順序付けが使用されているかを指定することができる。

【0105】

図7は、一実施形態による、プロセス700を示すフローチャートである。プロセス700は、ビットストリームから、いくつかのユニット8を含むピクチャ10を復号するための方法であり、ピクチャはパーティション構造13によって少なくとも2つの空間セグメント11に区分される。プロセス700はステップ710において開始し得る。ステップ710は、ビットストリーム中の1つまたは複数のコードワードを復号することを含む。ステップ720は、1つまたは複数のコードワードに基づいて、パーティション構造が均一であることを決定することを含む。ステップ730は、1つまたは複数のコードワードに基づいて空間セグメントの数を決定することを含む。ステップ740は、セグメントユニットサイズを決定することを含む。ステップ750は、1つまたは複数のコードワ

ドから、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することを含む。一実施形態では、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することは、第1の次元または方向における空間セグメントの数にわたる第1のループを含み、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数は第1のループ内で計算される。いくつかの実施形態では、第1の次元における残っているセグメントの数は第1のループ内で計算される。

【0106】

いくつかの実施形態では、セグメントサイズおよび／またはロケーションの導出は、第1の次元または方向以外の第2の次元または方向における空間セグメントの数にわたる第2のループを含み、セグメント化されるべき第2の次元または方向における残っているセグメントユニットの数は第2のループ内で計算される。いくつかの実施形態では、第2の次元または方向における残っているセグメントの数は、第2のループ内で計算される。

10

【0107】

いくつかの実施形態では、第1の次元または方向は水平の次元または方向であり、第2の次元または方向は垂直の次元または方向である。

【0108】

いくつかの実施形態では、行中のセグメントの幅または列中のセグメントの高さは、指定された順序に従う。

【0109】

いくつかの実施形態では、行中のセグメントの幅または列中のセグメントの高さは昇順でもなく、降順でもない。

20

【0110】

いくつかの実施形態では、セグメント幅または高さの明示的な順序付けをもつバイナリセグメントスプリットがピクチャの幅または高さに適用される。

【0111】

いくつかの実施形態では、ピクチャの行中のセグメントの幅または列中のセグメントの高さは、セグメントの幅または高さのための指定されたデフォルト順序に従う。

【0112】

いくつかの実施形態では、セグメントユニットサイズはコーディングツリーユニット (CTU) のサイズに等しい。

30

【0113】

いくつかの実施形態では、セグメントユニットサイズはコーディングツリーユニット (CTU) のサイズよりも小さい。いくつかの実施形態では、セグメントユニットサイズはCTUのサイズよりも大きい。いくつかの実施形態では、セグメントがタイルである。

【0114】

いくつかの実施形態では、サイズを導出することは、

`A = PicWidthInSegmentUnits`

`B = NumberOfSegmentColumns`

`for (i = 0; i < NumberOfSegmentColumns; i++) {`

`Sizes[i] = Round(A ÷ B)`

`A = A - Sizes[i]`

`B = B - 1`

`}`

としてリスト `Sizes[]` を導出することを含む、

40

ここで、`NumberOfSegmentColumns` はセグメント列の数であり、`PicWidthInSegmentUnits` はセグメントユニットにおけるピクチャの幅であり、`Round()` は `Floor()` 関数または `Ceil()` 関数のいずれかであり、`÷` は切り捨てまたは丸めなしの除算である。

【0115】

いくつかの実施形態では、サイズを導出することは、

50

```

A = PicHeightInSegmentUnits
B = NumberOfSegmentRows
for (i = 0; i < NumberOfSegmentRows; i++) {
    Sizes[i] = Round(A ÷ B)
    A = A - Sizes[i]
    B = B - 1
}

```

としてリスト `Sizes []` を導出することを含み、

ここで、`NumberOfSegmentRows` はセグメント行の数であり、`PicHeightInSegmentUnits` はセグメントユニットにおけるピクチャの高さであり、`Round()` は `Floor()` 関数または `Ceil()` 関数のいずれかであり、 $\div$ は切り捨てまたは丸めなしの除算である。

10

【0116】

いくつかの実施形態では、セグメント 11 は他のセグメント 11 に対して独立しており、したがって、現在セグメント 15 中の任意のユニット 16 のための任意のイントラ予測モードの導出は、現在セグメント 15 に属するユニット 17 における前に導出されたイントラ予測モードのみに依存し、異なるセグメント 14 に属するいかなるユニット 18 におけるいかなるイントラ予測モードにも依存しない。

【0117】

いくつかの実施形態では、本方法は、空間セグメントの数とは無関係な固定のリップルパターンに従う、セグメントを均一な幅または高さに区分するさらなるステップを含む。

20

【0118】

図 8 は、一実施形態による、プロセス 800 を示すフローチャートである。プロセス 800 は、いくつかのユニット 8 を含むピクチャ 10 をビットストリームに符号化するための方法であり、ピクチャは、均一なパーティション構造 13 によって少なくとも 2 つの空間セグメント 11 に区分される。本方法は、1 つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造 13 が均一であるという情報を符号化すること（ステップ 810）と、1 つまたは複数のコードワードをビットストリームに符号化することによって空間セグメントの数を符号化すること（ステップ 820）と、セグメントユニットサイズを決定すること（ステップ 830）と、ピクチャ中のすべての空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化すること（ステップ 840）とを含み、セグメントサイズの導出は、第 1 の次元または方向における空間セグメントの数にわたる第 1 のループを含み、セグメント化されるべき第 1 の次元または方向における残っているセグメントユニットの数は第 1 のループ内で計算される。いくつかの実施形態では、第 1 の次元または方向における残っているセグメントの数も第 1 のループ内で計算される。

30

【0119】

いくつかの実施形態では、セグメントサイズおよび／またはロケーションの導出は、第 1 の次元または方向以外の第 2 の次元または方向における空間セグメントの数にわたる第 2 のループを含み、セグメント化されるべき第 2 の次元または方向における残っているセグメントユニットの数は第 2 のループ内で計算される。いくつかの実施形態では、第 2 の次元または方向における残っているセグメントの数も第 2 のループ内で計算される。

40

【0120】

いくつかの実施形態では、第 1 の次元または方向は水平の次元または方向であり、第 2 の次元または方向は垂直の次元または方向である。

【0121】

図 9 は、いくつかの実施形態による、デコーダ 902 の機能ユニットを示すダイヤグラムである。図 9 に示されているように、デコーダ 902 は、ビットストリーム中の 1 つまたは複数のコードワードを復号するための復号ユニット 904 と、1 つまたは複数のコー

50

ドワードに基づいて、パーティション構造が均一であることを決定するための第1の決定ユニット906と、1つまたは複数のコードワードに基づいて空間セグメントの数を決定するための第2の決定ユニット908と、セグメントユニットサイズを決定するための第3の決定ユニット910と、1つまたは複数のコードワードから、ピクチャ中のすべての空間セグメントについてのサイズおよび／またはロケーションを導出するための導出ユニット912とを含み、セグメントサイズおよび／またはロケーションの導出は、第1の次元における空間セグメントの数にわたる第1のループと、第2の次元における空間セグメントの数にわたる第2のループとを含み、セグメント化されるべき第1の次元における残っているセグメントユニットの数は第1のループ内で計算され、セグメント化されるべき第2の次元における残っているセグメントユニットの数、および第2の次元における残っているセグメントの数は第2のループ内で計算される。いくつかの実施形態では、第1の次元における残っているセグメントの数も第1のループ内で計算され、第2の次元における残っているセグメントの数も第2のループ内で計算される。

【0122】

図10は、いくつかの実施形態による、エンコーダ1002の機能ユニットを示すダイヤグラムである。図10に示されているように、エンコーダ1002は、1つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造13が均一であるという情報を符号化するための第1の符号化ユニット1004と、1つまたは複数のコードワードをビットストリームに符号化することによって空間セグメントの数を符号化するための第2の符号化ユニット1006と、セグメントユニットサイズを決定するための決定ユニット1008と、ピクチャ中のすべての空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化するための導出ユニット1010とを含み、セグメントサイズの導出は、第1の次元における空間セグメントの数にわたる第1のループと、第2の次元における空間セグメントの数にわたる第2のループとを含み、セグメント化されるべき、第1の次元における残っているセグメントユニットの数は、第1のループ内で計算され、セグメント化されるべき、第2の次元における残っているセグメントユニットの数は、第2のループ内で計算される。

【0123】

いくつかの実施形態では、エンコーダは、ピクチャをいくつかの均一な空間セグメントに分割することであって、各空間セグメントは少なくとも1つのユニットを含み、セグメントサイズの導出は、第1の次元における空間セグメントの数にわたる第1のループ、および第2の次元における空間セグメントの数にわたる第2のループ中で実行され、セグメント化されるべき、第1の次元における残っているセグメントユニットの数は、第1のループ内で計算され、セグメント化されるべき、第2の次元における残っているセグメントユニットの数は、第2のループ内で計算される、ピクチャをいくつかの均一な空間セグメントに分割することと、複数のコーディングされた空間セグメントを生成するためのパーティション構造に従って複数の空間セグメントを符号化することであって、各々のコーディングされた空間セグメントはパーティション構造の空間セグメントのうちの1つに対応し、各々のコーディングされた空間セグメントは独立しており、したがって、第1の空間セグメントの第1のユニットのための任意のイントラ予測モードの導出は、第1の空間セグメントの第2のユニットのための導出されたイントラ予測モードに依存し、パーティション構造の他の空間セグメントのユニットのためのいかなるイントラ予測モードとも無関係である、複数の空間セグメントを符号化することと、複数のコーディングされた空間セグメントと、ピクチャを複数の空間セグメントに分割するために使用される均一なパーティション構造を示す情報とを含むビットストリームを生成することとを行う、パーティション構造を規定するように設定される。

【0124】

図11は、いくつかの実施形態による、デコーダ902および／またはエンコーダ1002を実装するための装置1100のブロック図である。装置1100がデコーダ902

を実装するとき、装置 1100 は「復号装置 1100」と呼ばれることがあり、装置 1100 がエンコーダ 1002 を実装するとき、装置 1100 は「符号化装置 1100」と呼ばれることがある。図 11 に示されているように、装置 1100（別名「ノード」）は、1 つまたは複数のプロセッサ（P）1155（たとえば、汎用マイクロプロセッサ、および／または特定用途向け集積回路（ASIC）、フィールドプログラマブルゲートアレイ（FPGA）など、1 つまたは複数の他のプロセッサ）を含み得る、処理回路（PC）1102 と、装置 1100 が、ネットワークインターフェース 1148 が接続されたネットワーク 1110（たとえばインターネットプロトコル（IP）ネットワーク）に接続された他のノードにデータを送信し、それらのノードからデータを受信することを可能にするための送信機（Tx）1145 と受信機（Rx）1147 とを含む、ネットワークインターフェース 1148 と、1 つまたは複数の不揮発性記憶デバイスおよび／または 1 つまたは複数の揮発性記憶デバイスを含み得る、ローカルストレージユニット（別名、「データ記憶システム」）1108 とを備え得る。PC 1102 がプログラマブルプロセッサを含む実施形態では、コンピュータプログラム製品（CPP）1141 が提供され得る。CPP 1141 は、コンピュータ可読命令（CRI）1144 を含むコンピュータプログラム（CP）1143 を記憶するコンピュータ可読媒体（CRM）1142 を含む。CRM 1142 は、磁気媒体（たとえばハードディスク）、光媒体、メモリデバイス（たとえば、ランダムアクセスメモリ、フラッシュメモリ）など、非一時的コンピュータ可読媒体であり得る。いくつかの実施形態では、コンピュータプログラム 1143 の CRI 1144 は、PC 1102 によって実行されたとき、CRI が装置 1100 に本明細書で説明されたステップ（たとえば、フローチャートを参照して本明細書で説明されたステップ）を実行させるように設定される。他の実施形態では、装置 1100 は、コードの必要なしに、本明細書で説明されるステップを実行するように設定され得る。すなわち、たとえば、PC 1102 は 1 つまたは複数の ASIC のみからなり得る。したがって、本明細書で説明された実施形態の特徴はハードウェアおよび／またはソフトウェアにおいて実装され得る。

【0125】

様々な実施形態の概要

A1a. ビットストリームから、いくつかのユニット 8 を含むピクチャ 10 を復号するための方法 700 であって、ピクチャはパーティション構造 13 によっていくつかの空間セグメント 11 に区分され、空間セグメントの数は 2 以上であり、本方法は、ビットストリーム中の 1 つまたは複数のコードワードを復号すること 710 と、1 つまたは複数のコードワードに基づいて、パーティション構造が均一であることを決定すること 720 と、1 つまたは複数のコードワードに基づいて空間セグメントの数を決定すること 730 と、セグメントユニットサイズを決定すること 740 と、1 つまたは複数のコードワードから、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出すること 750 とを含み、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することが、第 1 の次元または方向における空間セグメントの数にわたる第 1 のループを含み、セグメント化されるべき第 1 の次元または方向における残っているセグメントユニットの数が第 1 のループ内で計算される、方法 700。

【0126】

A1b. 空間セグメントの数にわたる第 1 のループが、第 1 の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第 1 のループを含む、実施形態 A1a に記載の方法。

【0127】

A2a. サイズおよび／またはロケーションを導出することが、第 1 の次元または方向以外の第 2 の次元または方向における空間セグメントの数にわたる第 2 のループをさらに含み、セグメント化されるべき第 2 の次元または方向における残っているセグメントユニットの数が第 2 のループ内で計算される、実施形態 A1a または A1b に記載の方法。

【0128】

A2b. 第 1 の次元または方向におけるセグメントサイズが行中のセグメントの幅であ

り、第2の次元または方向におけるセグメントサイズが列中のセグメントの高さであるか、または、第1の次元または方向におけるセグメントサイズが列中のセグメントの高さであり、第2の次元または方向におけるセグメントサイズが行中のセグメントの幅である、実施形態A2aに記載の方法。

【0129】

A2c. 空間セグメントの数にわたる第2のループが、第2の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第2のループを含む、実施形態A2aまたはA2bに記載の方法。

【0130】

A3. 第1の次元または方向が水平の次元または方向であり、第2の次元または方向が垂直の次元または方向であるか、または第1の次元または方向が垂直の次元または方向であり、第2の次元または方向が水平の次元または方向である、実施形態A2a、A2bまたはA2cに記載の方法。

10

【0131】

A4. 行中のセグメントの幅または列中のセグメントの高さが、指定された順序に従う、実施形態A1aからA3のいずれか1つに記載の方法。

【0132】

A5. 行中のセグメントの幅または列中のセグメントの高さが昇順でもなく、降順でもない、実施形態A1aからA4のいずれか1つに記載の方法。

【0133】

A6. セグメント幅または高さの明示的な順序付けをもつバイナリセグメントスプリットがピクチャの幅または高さに適用される、実施形態A1aからA5のいずれか1つに記載の方法。

20

【0134】

A7. ピクチャの行中のセグメントの幅または列中のセグメントの高さが、セグメントの幅または高さのための指定されたデフォルト順序に従う、実施形態A1aからA6のいずれか1つに記載の方法。

【0135】

A8. セグメントユニットサイズがコーディングツリーユニット(CTU)のサイズに等しい、実施形態A1aからA7のいずれか1つに記載の方法。

30

【0136】

A9. セグメントユニットサイズがコーディングツリーユニット(CTU)のサイズよりも小さい、実施形態A1aからA7のいずれか1つに記載の方法。

【0137】

A10. セグメントユニットサイズがCTUのサイズよりも大きい、実施形態A1aからA7のいずれか1つに記載の方法。

【0138】

A11. セグメントがタイルである、実施形態A1aからA10のいずれか1つに記載の方法。

【0139】

A12. サイズを導出することは、
`A=PicWidthInSegmentUnits`
`B=NumberOfSegmentColumns`
`for (i=0; i<NumberOfSegmentColumns; i++) {`
`Sizes[i]=Round(A÷B)`
`A=A-Sizes[i]`
`B=B-1`
`}`

40

としてリストSizes[]を導出することを含み、

ここで、NumberOfSegmentColumnsはセグメント列の数であり、

50

`PicWidthInSegmentUnits`はセグメントユニットにおけるピクチャの幅であり、`Round()`は`Floor()`関数または`Ceil()`関数のいずれかであり、 $\div$ は切り捨てまたは丸めなしの除算である、実施形態A1aからA11のいずれか1つに記載の方法。

【0140】

A13. サイズを導出することは、

`A = PicHeightInSegmentUnits`

`B = NumberOfSegmentRows`

`for (i = 0; i < NumberOfSegmentRows; i++) {`

`Sizes[i] = Round(A ÷ B)`

`A = A - Sizes[i]`

`B = B - 1`

`}`

としてリスト`Sizes[]`を導出することを含み、

ここで、`NumberOfSegmentRows`はセグメント行の数であり、`PicHeightInSegmentUnits`はセグメントユニットにおけるピクチャの高さであり、`Round()`は`Floor()`関数または`Ceil()`関数のいずれかであり、 $\div$ は切り捨てまたは丸めなしの除算である、実施形態A1aからA12のいずれか1つに記載の方法。

【0141】

A14. セグメント11が他のセグメント11に対して独立しており、したがって、現在セグメント15中の任意のユニット16のための任意のイントラ予測モードの導出は、現在セグメント15に属するユニット17における前に導出されたイントラ予測モードのみに依存し、異なるセグメント14に属するいかなるユニット18におけるいかなるイントラ予測モードにも依存しない、実施形態A1aからA13のいずれか1つに記載の方法。

【0142】

A15. 本方法は、空間セグメントの数とは無関係な固定のリップルパターンに従う、セグメントを均一な幅または高さに区分することをさらに含む、実施形態A1aからA14のいずれか1つに記載の方法。

【0143】

A16. 第2の次元または方向における残っているセグメントの数が第2のループ内で計算される、実施形態A2aからA15のいずれか1つに記載の方法。

【0144】

A17. 第1の次元または方向における残っているセグメントの数が第1のループ内で計算される、実施形態A1aからA16のいずれか1つに記載の方法。

【0145】

B1a. いくつかのユニット8を含むピクチャ10をビットストリームに符号化するための方法800であって、ピクチャは均一なパーティション構造13によっていくつかの空間セグメント11に区分され、空間セグメントの数は2以上であり、本方法は、1つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造13が均一であるという情報を符号化すること810と、1つまたは複数のコードワードをビットストリームに符号化することによって、空間セグメントの数を符号化すること820と、セグメントユニットサイズを決定すること830と、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化すること840とを含み、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することは、第1の次元または方向における空間セグメントの数にわたる第1のループを含み、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数が第1のループ内で計算される、方法800。

【0146】

B 1 b . 空間セグメントの数にわたる第 1 のループが、第 1 の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第 1 のループを含む、実施形態 B 1 a に記載の方法。

【 0 1 4 7 】

B 2 a . セグメントサイズおよび／またはロケーションの導出が、第 1 の次元または方向以外の第 2 の次元または方向における空間セグメントの数にわたる第 2 のループを含み、セグメント化されるべき第 2 の次元または方向における残っているセグメントユニットの数が第 2 のループ内で計算される、実施形態 B 1 a または B 1 b に記載の方法。

【 0 1 4 8 】

B 2 b . 第 1 の次元または方向におけるセグメントサイズが行中のセグメントの幅であり、第 2 の次元または方向におけるセグメントサイズが列中のセグメントの高さであるか、または、第 1 の次元または方向におけるセグメントサイズが列中のセグメントの高さであり、第 2 の次元または方向におけるセグメントサイズが行中のセグメントの幅である、実施形態 B 2 a に記載の方法。

【 0 1 4 9 】

B 2 c . 空間セグメントの数にわたる第 2 のループが、第 2 の次元または方向におけるセグメントサイズに等しいサイズをもつ空間セグメントの数にわたる第 2 のループを含む、実施形態 B 2 a または B 2 b に記載の方法。

【 0 1 5 0 】

B 3 . 第 1 の次元または方向が水平の次元または方向であり、第 2 の次元または方向が垂直の次元または方向であるか、または第 1 の次元または方向が垂直の次元または方向であり、第 2 の次元または方向が水平の次元または方向である、実施形態 B 2 a 、 B 2 b または B 2 c に記載の方法。

【 0 1 5 1 】

B 4 . 第 2 の次元または方向における残っているセグメントの数が第 2 のループ内で計算される、実施形態 B 2 a 、 B 2 b 、 B 2 c または B 3 に記載の方法。

【 0 1 5 2 】

B 5 . 第 1 の次元または方向における残っているセグメントの数が第 1 のループ内で計算される、実施形態 B 1 a から B 4 のいずれか 1 つに記載の方法。

【 0 1 5 3 】

C 1 . ピクチャ 1 0 を符号化するための方法 1 2 0 0 であって、本方法は、ピクチャ 1 0 をいくつかの均一な空間セグメントに分割するパーティション構造を規定すること 1 2 1 0 であって、各空間セグメントが少なくとも 1 つのユニットを含み、セグメントサイズの導出が、第 1 の次元または方向における空間セグメントの数にわたる第 1 のループ、および第 2 の次元または方向における空間セグメントの数にわたる第 2 のループ中で実行され、セグメント化されるべき第 1 の次元または方向における残っているセグメントユニットの数が第 1 のループ内で計算され、セグメント化されるべき第 2 の次元または方向における残っているセグメントユニットの数が第 2 のループ内で計算される、パーティション構造を規定すること 1 2 1 0 と、複数のコーディングされた空間セグメントを生成するためのパーティション構造に従って、複数の空間セグメントを符号化すること 1 2 2 0 であって、各々のコーディングされた空間セグメントがパーティション構造の空間セグメントのうちの 1 つに対応し、各々のコーディングされた空間セグメントは独立しており、したがって、第 1 の空間セグメントの第 1 のユニットのための任意のイントラ予測モードの導出は、第 1 の空間セグメントの第 2 のユニットのための導出されたイントラ予測モードに依存し、パーティション構造の他の空間セグメントのユニットのためのいかなるイントラ予測モードとも無関係である、複数の空間セグメントを符号化すること 1 2 2 0 と、複数のコーディングされた空間セグメントと、ピクチャを複数の空間セグメントに分割するために使用される均一なパーティション構造を示す情報とを含む、ビットストリームを生成すること 1 2 3 0 とを含む、方法 1 2 0 0 。

【 0 1 5 4 】

D 1. 処理回路 1 1 0 2 によって実行されたとき、処理回路 1 1 0 2 に実施形態 A 1 a から C 1 のいずれか 1 つに記載の方法を実行させる命令 1 1 4 4 を含む、コンピュータプログラム 1 1 4 3。

【0 1 5 5】

D 2. 実施形態 D 1 に記載のコンピュータプログラムを含んでいるキャリアであって、本キャリアは、電子信号、光信号、無線信号、およびコンピュータ可読記憶媒体 1 1 4 2 のうちの 1 つである、キャリア。

【0 1 5 6】

E 1. ビットストリームから、いくつかのユニット 8 を含むピクチャ 1 0 を復号するための復号装置 1 1 0 0 であって、ピクチャはパーティション構造 1 3 によっていくつかの空間セグメント 1 1 に区分され、空間セグメントの数は 2 以上であり、本復号装置 1 1 0 0 は、ビットストリーム中の 1 つまたは複数のコードワードを復号することと、1 つまたは複数のコードワードに基づいて、パーティション構造が均一であることを決定することと、1 つまたは複数のコードワードに基づいて空間セグメントの数を決定することと、セグメントユニットサイズを決定することと、1 つまたは複数のコードワードから、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することとを行うように設定され、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することは、第 1 の次元または方向における空間セグメントの数にわたる第 1 のループを含み、本復号装置は、第 1 のループ内で、セグメント化されるべき第 1 の次元または方向における残っているセグメントユニットの数を計算するように設定された、復号装置 1 1 0 0。

【0 1 5 7】

E 2. 本復号装置 1 1 0 0 は、実施形態 A 2 から A 1 7 のいずれか 1 つに記載の方法を実行するようにさらに設定された、実施形態 E 1 に記載の復号装置 1 1 0 0。

【0 1 5 8】

F 1. ピクチャ 1 0 を復号するための復号装置 1 1 0 0 であって、本復号装置 1 1 0 0 は、コンピュータ可読記憶媒体 1 1 4 2 と、コンピュータ可読記憶媒体に結合された処理回路 1 1 0 2 とを備え、処理回路は、本復号装置 1 1 0 0 に実施形態 A 1 a から A 1 7 のいずれか 1 つに記載の方法を実行させるように設定された、復号装置 1 1 0 0。

【0 1 5 9】

G 1. いくつかのユニット 8 を含むピクチャ 1 0 をビットストリームに符号化するための符号化装置 1 1 0 0 であって、ピクチャは均一なパーティション構造 1 3 によっていくつかの空間セグメント 1 1 に区分され、空間セグメントの数は 2 以上であり、本符号化装置 1 1 0 0 は、1 つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造 1 3 が均一であるという情報を符号化することと、1 つまたは複数のコードワードをビットストリームに符号化することによって空間セグメントの数を符号化することと、セグメントユニットサイズを決定することと、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化することとを行うように設定され、ピクチャ中の空間セグメントについてのサイズおよび／またはロケーションを導出することは、第 1 の次元または方向における空間セグメントの数にわたる第 1 のループを含み、本符号化装置は、第 1 のループ内で、セグメント化されるべき第 1 の次元または方向における残っているセグメントユニットの数を計算するようにさらに設定された、符号化装置 1 1 0 0。

【0 1 6 0】

G 2. 本符号化装置 1 1 0 0 は、実施形態 B 2 から B 5 のいずれか 1 つに記載の方法を実行するようにさらに設定された、実施形態 G 1 に記載の符号化装置 1 1 0 0。

【0 1 6 1】

H 1. ピクチャ 1 0 を符号化するための符号化装置 1 1 0 0 であって、本符号化装置 1 1 0 0 は、ピクチャをいくつかの均一な空間セグメントに分割するパーティション構造を

規定することであって、各空間セグメントが少なくとも1つのユニットを含み、セグメントサイズの導出が、第1の次元または方向における空間セグメントの数にわたる第1のループ、および第2の次元または方向における空間セグメントの数にわたる第2のループ中で実行され、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数が第1のループ内で計算され、セグメント化されるべき第2の次元または方向における残っているセグメントユニットの数が第2のループ内で計算される、パーティション構造を規定することと、複数のコーディングされた空間セグメントを生成するためのパーティション構造に従って、複数の空間セグメントを符号化することであって、各々のコーディングされた空間セグメントがパーティション構造の空間セグメントのうちの1つに対応し、各々のコーディングされた空間セグメントは独立しており、したがって、第1の空間セグメントの第1のユニットのための任意のイントラ予測モードの導出は、第1の空間セグメントの第2のユニットのための導出されたイントラ予測モードに依存し、パーティション構造の他の空間セグメントのユニットのためのいかなるイントラ予測モードとも無関係である、複数の空間セグメントを符号化することと、複数のコーディングされた空間セグメントと、ピクチャを複数の空間セグメントに分割するために使用される均一なパーティション構造を示す情報とを含む、ビットストリームを生成することを行うように設定された、符号化装置1100。

10

【0162】

I1. ピクチャ10を符号化するための符号化装置1100であって、本符号化装置1100は、コンピュータ可読記憶媒体1142と、コンピュータ可読記憶媒体に結合された処理回路1102とを備え、処理回路は、本符号化装置1100に実施形態B1からB5のいずれか1つまたは実施形態C1に記載の方法を実行させるように設定された、符号化装置1100。

20

【0163】

PA1. ビットストリームから、いくつかのユニット8を含むピクチャ10を復号するための方法であって、ピクチャはパーティション構造13によって少なくとも2つの空間セグメント11に区分され、本方法は、ビットストリーム中の1つまたは複数のコードワードを復号することと、1つまたは複数のコードワードに基づいて、パーティション構造が均一であることを決定することと、1つまたは複数のコードワードに基づいて空間セグメントの数を決定することと、セグメントユニットサイズを決定することと、1つまたは複数のコードワードから、ピクチャ中のすべての空間セグメントについてのサイズおよび／またはロケーションを導出することとを含み、セグメントサイズおよび／またはロケーションの導出は、第1の次元または方向における空間セグメントの数にわたる第1のループを含み、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数、および第1の次元または方向における残っているセグメントの数が第1のループ内で計算される、方法。

30

【0164】

PA2. セグメントサイズおよび／またはロケーションの導出が、第1の次元または方向以外の第2の次元または方向における空間セグメントの数にわたる第2のループを含み、セグメント化されるべき第2の次元または方向における残っているセグメントユニットの数、および第2の次元または方向における残っているセグメントの数が第2のループ内で計算される、実施形態PA1に記載の方法。

40

【0165】

PA3. 第1の次元または方向が水平の次元または方向であり、第2の次元または方向が垂直の次元または方向である、実施形態PA2に記載の方法。

【0166】

PA4. 行中のセグメントの幅または列中のセグメントの高さが、指定された順序に従う、実施形態PA1からPA3のいずれか1つに記載の方法。

【0167】

PA5. 行中のセグメントの幅または列中のセグメントの高さが昇順でもなく、降順で

50

もない、実施形態 P A 1 から P A 3 のいずれか 1 つに記載の方法。

【0168】

P A 6 . セグメント幅または高さの明示的な順序付けをもつバイナリセグメントスプリットがピクチャの幅または高さに適用される、実施形態 P A 1 から P A 3 のいずれか 1 つに記載の方法。

【0169】

P A 7 . ピクチャの行中のセグメントの幅または列中のセグメントの高さが、セグメントの幅または高さのための指定されたデフォルト順序に従う、実施形態 P A 1 から P A 3 のいずれか 1 つに記載の方法。

【0170】

P A 8 . セグメントユニットサイズがコーディングツリーユニット (C T U) のサイズに等しい、実施形態 P A 1 から P A 7 のいずれか 1 つに記載の方法。

【0171】

P A 9 . セグメントユニットサイズがコーディングツリーユニット (C T U) のサイズよりも小さい、実施形態 P A 1 から P A 8 のいずれか 1 つに記載の方法。

【0172】

P A 10 . セグメントユニットサイズが C T U のサイズよりも大きい、実施形態 P A 1 から P A 8 のいずれか 1 つに記載の方法。

【0173】

P A 11 . 任意のセグメントがタイルである、実施形態 P A 1 から P A 10 のいずれか 1 つに記載の方法。

【0174】

P A 12 . サイズを導出することは、
`A=PicWidthInTileUnits`
`B=NumberOfSegmentColumns`
`for (i=0; i<NumberOfSegmentColumns; i++) {`
`Sizes[i]=Round(A÷B)`
`A=A-Sizes[i]`
`B=B-1`
`}`

としてリスト `Sizes []` を導出することを含み、

ここで、`NumberOfSegmentColumns` はセグメント列の数であり、`PicWidthInTileUnits` はタイルユニットにおけるピクチャの幅であり、`Round ()` は `Floor ()` 関数または `Ceil ()` 関数のいずれかであり、`÷` は切り捨てまたは丸めなしの除算である、実施形態 P A 1 から P A 11 のいずれか 1 つに記載の方法。

【0175】

P A 13 . サイズを導出することは、
`A=PicHeightInTileUnits`
`B=NumberOfSegmentRows`
`for (i=0; i<NumberOfSegmentRows; i++) {`
`Sizes[i]=Round(A÷B)`
`A=A-Sizes[i]`
`B=B-1`
`}`

としてリスト `Sizes []` を導出することを含み、

ここで、`NumberOfSegmentRows` はセグメント行の数であり、`PicHeightInTileUnits` はタイルユニットにおけるピクチャの高さであり、`Round ()` は `Floor ()` 関数または `Ceil ()` 関数のいずれかであり、`÷` は切り捨てまたは丸めなしの除算である、実施形態 P A 1 から P A 12 のいずれか 1 つに記載

10

20

30

40

50

の方法。

【0176】

PA14. セグメント11は他のセグメント11に対して独立しており、したがって、現在セグメント15中の任意のユニット16のための任意のイントラ予測モードの導出は、現在セグメント15に属するユニット17における前に導出されたイントラ予測モードのみに依存し、異なるセグメント14に属するいかなるユニット18におけるいかなるイントラ予測モードにも依存しない、実施形態PA1からPA13のいずれか1つに記載の方法。

【0177】

PA15. 本方法は、空間セグメントの数とは無関係な固定のリップルパターンに従う、セグメントを均一な幅または高さに区分することをさらに含む、実施形態PA1からPA14のいずれか1つに記載の方法。

【0178】

PB1. いくつかのユニット8を含むピクチャ10をビットストリームに符号化するための方法であって、ピクチャは均一なパーティション構造13によって少なくとも2つの空間セグメント11に区分され、本方法は、1つまたは複数のコードワードをビットストリームに符号化することによって、パーティション構造13が均一であるという情報を符号化することと、1つまたは複数のコードワードをビットストリームに符号化することによって空間セグメントの数を符号化することと、セグメントユニットサイズを決定することと、ピクチャ中のすべての空間セグメントについてのサイズおよび／またはロケーションを導出し、それらのサイズおよび／またはロケーションをビットストリームに符号化することとを含み、セグメントサイズの導出は、第1の次元または方向における空間セグメントの数にわたる第1のループを含み、セグメント化されるべき第1の次元または方向における残っているセグメントユニットの数、および第1の次元または方向における残っているセグメントの数が第1のループ内で計算される、方法。

【0179】

PB2. セグメントサイズおよび／またはロケーションの導出が、第1の次元または方向以外の第2の次元または方向における空間セグメントの数にわたる第2のループを含み、セグメント化されるべき第2の次元または方向における残っているセグメントユニットの数、および第2の次元または方向における残っているセグメントの数が第2のループ内で計算される、実施形態PB1に記載の方法。

【0180】

PB3. 第1の次元または方向が水平の次元または方向であり、第2の次元または方向が垂直の次元または方向である、実施形態PB2に記載の方法。

【0181】

PC1. ピクチャをいくつかの均一な空間セグメントに分割するパーティション構造を規定することであって、各空間セグメントが少なくとも1つのユニットを含み、セグメントサイズの導出は、第1の次元における空間セグメントの数にわたる第1のループ、および第2の次元における空間セグメントの数にわたる第2のループ中で実行され、セグメント化されるべき第1の次元における残っているセグメントユニットの数、および第1の次元における残っているセグメントの数が第1のループ内で計算され、セグメント化されるべき第2の次元における残っているセグメントユニットの数、および第2の次元における残っているセグメントの数が第2のループ内で計算される、パーティション構造を規定することと、複数のコーディングされた空間セグメントを生成するためのパーティション構造に従って、複数の空間セグメントを符号化することであって、各々のコーディングされた空間セグメントがパーティション構造の空間セグメントのうちの1つに対応し、各々のコーディングされた空間セグメントは独立しており、したがって、第1の空間セグメントの第1のユニットのための任意のイントラ予測モードの導出は、第1の空間セグメントの第2のユニットのための導出されたイントラ予測モードに依存し、パーティション構造の他の空間セグメントのユニットのためのいかなるイントラ予測モードとも無関係である、

10

20

30

40

50

複数の空間セグメントを符号化することと、複数のコーディングされた空間セグメントと、ピクチャを複数の空間セグメントに分割するために使用される均一なパーティション構造を示す情報とを含む、ビットストリームを生成することとを行うように設定された、エンコーダ回路。

【0182】

様々な実施形態について本明細書（もしあれば、付属書類を含む）で説明したが、それらの実施形態は限定ではなく、例としてのみ提示されていることを理解されたい。したがって、本開示の広さおよび範囲は、上記で説明した例示的な実施形態のいずれかによって限定されるべきでない。その上、上記で説明した要素のすべての可能な変形形態におけるそれらの要素の任意の組合せは、本明細書で別段に示されていない限り、またはさもなければ、文脈によって明らかに否定されない限り、本開示によって包含される。

10

【0183】

さらに、上記で説明され、図面に示されたプロセスはステップのシーケンスとして示されているが、このことは例示のために行われたに過ぎない。したがって、いくつかのステップは追加され得、いくつかのステップは省略され得、ステップの順序は並べ替えられ得、いくつかのステップは並行して実行され得ることが企図される。

20

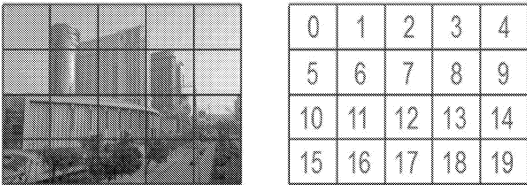
30

40

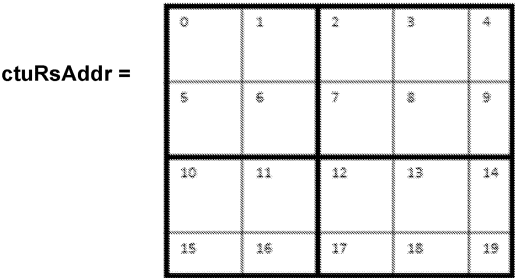
50

【図面】

【図 1】

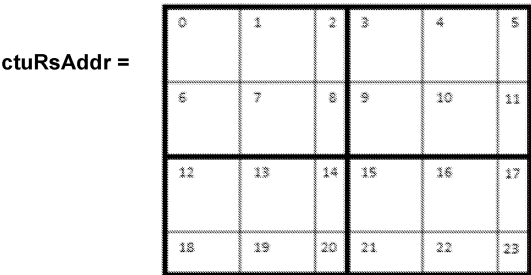


【図 2 A】

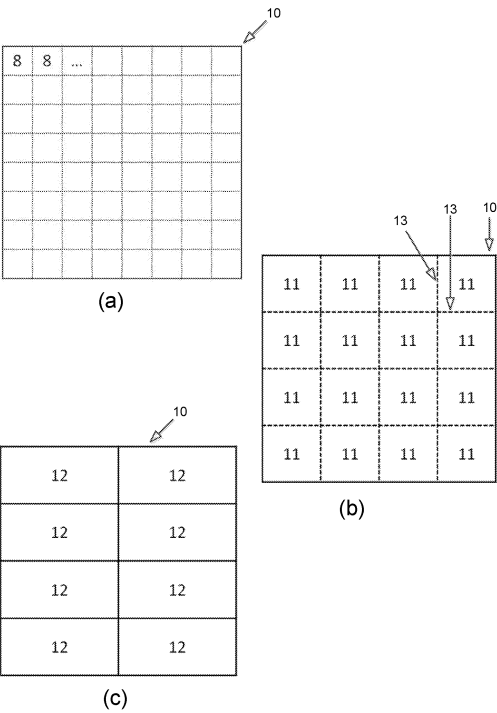


10

【図 2 B】



【図 3】



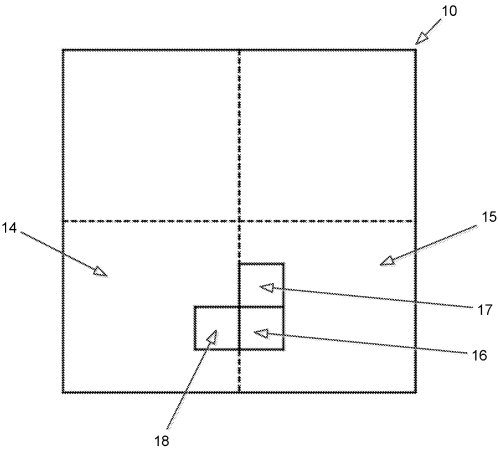
20

30

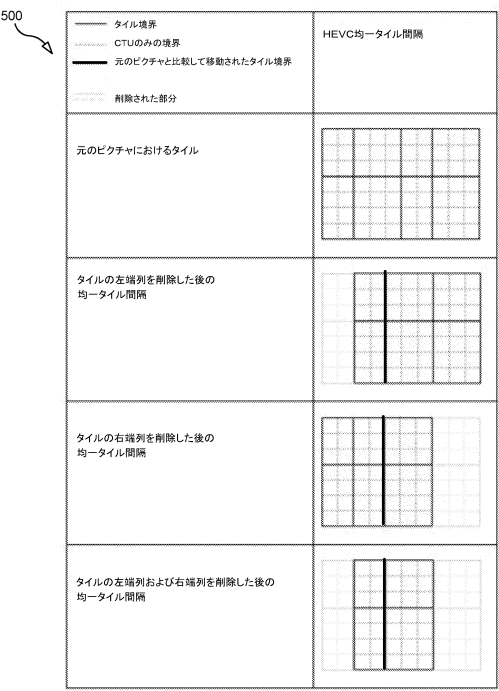
40

50

【図 4】



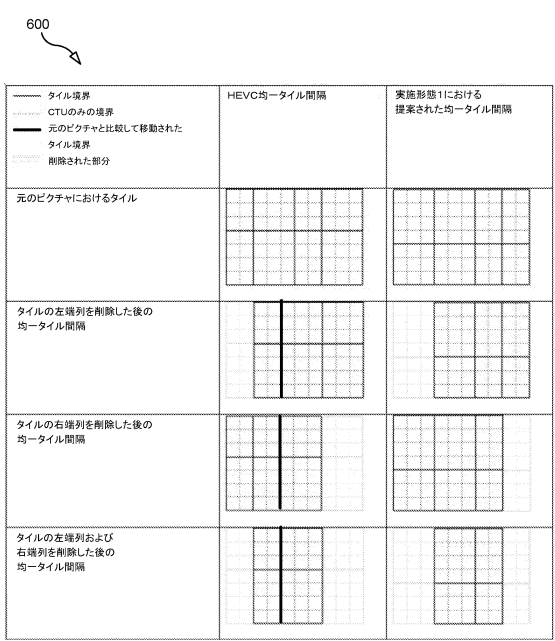
【図 5】



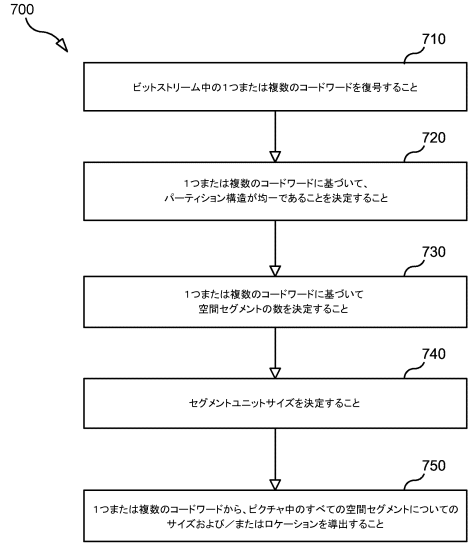
10

20

【図 6】



【図 7】

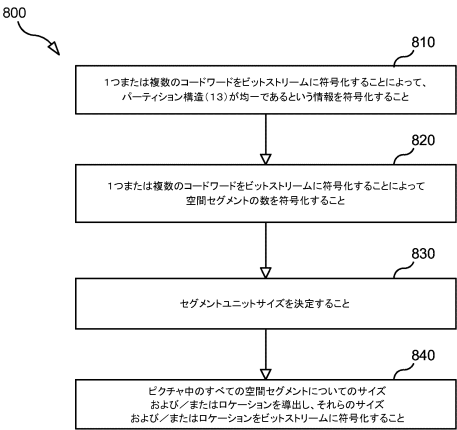


30

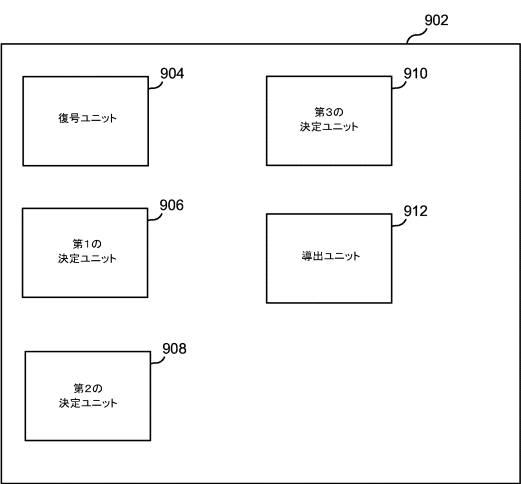
40

50

【図 8】

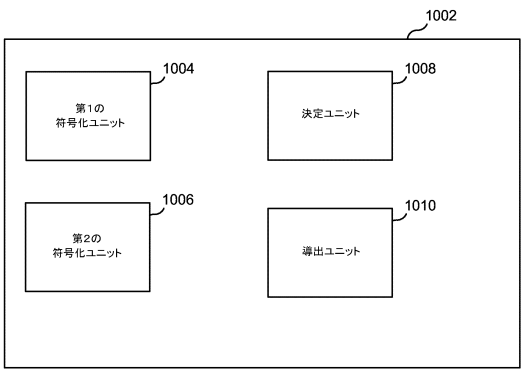


【図 9】

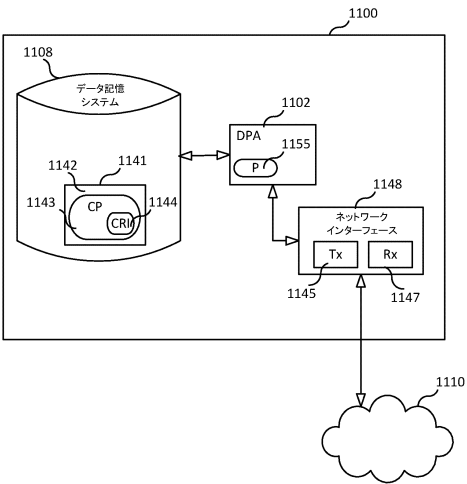


10

【図 10】



【図 11】



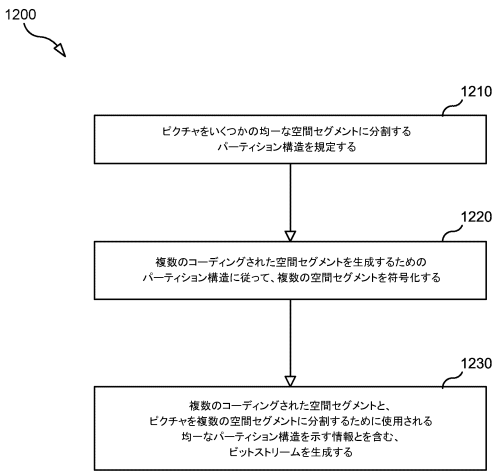
20

30

40

50

【図 12】



10

20

30

40

50

フロントページの続き

- (72)発明者 ダムガニアン, ミトラ
 スウェーデン国 1 9 7 9 2 ウップランズ ブロー, マルコールヴェーゲン 3
- (72)発明者 ショバーリ, リキヤルド
 スウェーデン国 1 1 3 6 2 ストックホルム, サンクト イェリクスガータン 8 4
- (72)発明者 ペッテション, マルティン
 スウェーデン国 エスエー 1 8 6 3 4 ヴァレンチューナ, ステンヴァッラヴェーゲン 1 0
- 審査官 富樫 明
- (56)参考文献 国際公開第 2 0 1 5 / 0 5 3 2 8 7 (WO, A 1)
 Yukinobu Yasugi et al., AHG12: Flexible Tile Partitioning, JVET-K0155-v1, 2018年07月
- (58)調査した分野 (Int.Cl., D B 名)
 H 0 4 N 1 9 / 0 0 - 1 9 / 9 8