(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7629029号
(P7629029)

(45)発行日　令和7年2月12日(2025.2.12)

(24)登録日　令和7年2月3日(2025.2.3)

(51)国際特許分類		FI	
H04N	*19/13 (2014.01)*	H04N	19/13
H04N	*19/157 (2014.01)*	H04N	19/157
H04N	*19/176 (2014.01)*	H04N	19/176
H04N	*19/513 (2014.01)*	H04N	19/513
H04N	*19/70 (2014.01)*	H04N	19/70

請求項の数　12　（全87頁）

(21)出願番号　特願2022-566447(P2022-566447)
(86)(22)出願日　令和3年5月6日(2021.5.6)
(65)公表番号　特表2023-523839(P2023-523839A)
(43)公表日　令和5年6月7日(2023.6.7)
(86)国際出願番号　PCT/CN2021/091869
(87)国際公開番号　WO2021/219143
(87)国際公開日　令和3年11月4日(2021.11.4)
審査請求日　令和4年11月21日(2022.11.21)
(31)優先権主張番号　PCT/CN2020/088546
(32)優先日　令和2年5月1日(2020.5.1)
(33)優先権主張国・地域又は機関
中国(CN)
前置審査

(73)特許権者　520476341
北京字節跳動網絡技術有限公司
Ｂｅｉｊｉｎｇ Ｂｙｔｅｄａｎｃｅ Ｎｅｔｗｏｒｋ Ｔｅｃｈｎｏｌｏｇｙ Ｃｏ．，Ｌｔｄ．
中華人民共和国１００００４１北京市石景山区実興大街３０号院３号楼２層Ｂ－００３５房間
Ｒｏｏｍ Ｂ－００３５，２／Ｆ，Ｎｏ．３ Ｂｕｉｌｄｉｎｇ，Ｎｏ．３０，Ｓｈｉｘｉｎｇ Ｒｏａｄ，Ｓｈｉｊｉｎｇｓｈａｎ Ｄｉｓｔｒｉｃｔ Ｂｅｉｊｉｎｇ １００００４１ Ｃｈｉｎａ
(73)特許権者　520477474
バイトダンス インコーポレイテッド
最終頁に続く

(54)【発明の名称】　動き精度構文のためのエントロピーコーディング

(57)【特許請求の範囲】
【請求項１】
映像処理の方法であって、
規則に基づいて、映像の第１のブロックと、前記映像のビットストリームとの間の変換を実行すること、
を有し、
前記変換は、ＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づき、
前記規則は、ＡＭＶＲシフトに関連付けられた動きベクトル差分の解像度を規定する第１の構文要素のビンの文字列における第１のビンに対するコンテキスト増加手段（ｃｔｘＩｎｃ）の選択が、前記第１のブロックに対するコーディングモードの使用に基づいて導出されることを規定し、
前記第１のブロックに対する前記コーディングモードは、アフィンインターモード、ＩＢＣ（Ｉｎｔｒａ Ｂｌｏｃｋ Ｃｏｐｙ）モード、および、非アフィンインターモードである通常のインターモードのうちの１つであり、
第１のコンテキスト増加手段は、前記第１のブロックが前記ＩＢＣモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第２のコンテキスト増加手段は、前記第１のブロックが前記アフィンインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第３のコンテキスト増加手段は、前記第１のブロックが前記通常のインターモードを用

いてコーディングされる場合に、前記第１のビンに割り当てられ、
　前記第１のコンテキスト増加手段は、１に等しく、
　前記第２のコンテキスト増加手段は、２に等しく、
　前記第３のコンテキスト増加手段は、０に等しい、方法。
【請求項２】
　前記第１のブロックは、コーディングユニットである、請求項１に記載の方法。
【請求項３】
　３つの異なる前記コーディングモードに対応する３つの異なるコンテキスト増加手段は、前記第１のビンに適用可能であり、
　各コーディングモードは、単一のコンテキスト増加手段に対応する、請求項１または２に記載の方法。
【請求項４】
同じコンテキスト増加手段が、（１）前記第１のブロックが前記ＩＢＣモードを使用してコーディングされている場合に前記第１のブロックに対する前記第１の構文要素の前記ビンの文字列の前記第１のビン、かつ、（２）前記通常のインターモードを用いてコーディングされている第２のブロックに対する前記第１の構文要素の前記ビンの文字列の第２のビン、に対して用いられる、請求項１から３のいずれか一項に記載の方法。
【請求項５】
同じコンテキスト増加手段が、（１）前記第１のブロックが前記ＩＢＣモードを使用してコーディングされている場合に前記第１のブロックに対する前記第１の構文要素の前記ビンの文字列の前記第１のビン、かつ、（２）第３のブロックに対する第２の構文要素のビン、に対して用いられ、前記第２の構文要素は、前記動きベクトル差分の前記解像度が輝度サンプルの１／４であるか否か、または前記動きベクトル差分の前記解像度が前記第１の構文要素によって規定されるか否かを規定し、前記第３のブロックは、前記アフィンインターモードを用いてコーディングされる、請求項１から３のいずれか一項に記載の方法。
【請求項６】
同じコンテキスト増加手段が、（１）前記第１のブロックが前記通常のインターモードを使用してコーディングされている場合に前記第１のブロックに対する前記第１の構文要素の前記ビンの文字列の前記第１のビン、かつ、（２）第４のブロックに対する第２の構文要素のビン、に対して用いられ、前記第２の構文要素は、前記動きベクトル差分の前記解像度が輝度サンプルの１／４であるか否か、または前記動きベクトル差分の前記解像度が前記第１の構文要素によって規定されるか否かを規定し、前記第４のブロックは、前記アフィンインターモードを用いてコーディングされない、請求項１から３のいずれか一項に記載の方法。
【請求項７】
　初期化処理において、
　コンテキスト初期化タイプが第１のタイプである場合、前記第１の構文要素に対するコンテキストインデックスの値は、０から２の範囲にあり、
　前記コンテキスト初期化タイプが第２のタイプである場合、前記第１の構文要素に対する前記コンテキストインデックスの前記値は、３から５の範囲にあり、
　前記コンテキスト初期化タイプが第３のタイプである場合、前記第１の構文要素に対する前記コンテキストインデックスの前記値は、６から８の範囲にある、請求項１から６のいずれか一項に記載の方法。
【請求項８】
　前記変換は、前記映像を前記ビットストリームに符号化することを含む、請求項１から７のいずれか一項に記載の方法。
【請求項９】
　前記変換は、前記映像を前記ビットストリームから復号することを含む、請求項１から７のいずれか一項に記載の方法。
【請求項１０】

プロセッサと、命令を有する非一時的メモリを有する、映像データを処理するための装置であって、前記命令が前記プロセッサによって実行された際に、前記プロセッサに、
規則に基づいて、映像の第１のブロックと、前記映像のビットストリームとの間の変換を実行すること、を行わせ、
前記変換は、ＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づき、
前記規則は、ＡＭＶＲシフトに関連付けられた動きベクトル差分の解像度を規定する第１の構文要素のビンの文字列における第１のビンに対するコンテキスト増加手段（ｃｔｘＩｎｃ）の選択が、前記第１のブロックに対するコーディングモードの使用に基づいて導出されることを規定し、
前記第１のブロックに対する前記コーディングモードは、アフィンインターモード、ＩＢＣ（Ｉｎｔｒａ Ｂｌｏｃｋ Ｃｏｐｙ）モード、および、非アフィンインターモードである通常のインターモードのうちの１つであり、
第１のコンテキスト増加手段は、前記第１のブロックが前記ＩＢＣモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第２のコンテキスト増加手段は、前記第１のブロックが前記アフィンインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第３のコンテキスト増加手段は、前記第１のブロックが前記通常のインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
前記第１のコンテキスト増加手段は、１に等しく、
前記第２のコンテキスト増加手段は、２に等しく、
前記第３のコンテキスト増加手段は、０に等しい、装置。

【請求項１１】
プロセッサに、
規則に基づいて、映像の第１のブロックと、前記映像のビットストリームとの間の変換を実行すること、
を行わせ、
前記変換は、ＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づき、
前記規則は、ＡＭＶＲシフトに関連付けられた動きベクトル差分の解像度を規定する第１の構文要素のビンの文字列における第１のビンに対するコンテキスト増加手段（ｃｔｘＩｎｃ）の選択が、前記第１のブロックに対するコーディングモードの使用に基づいて導出されることを規定し、
前記第１のブロックに対する前記コーディングモードは、アフィンインターモード、ＩＢＣ（Ｉｎｔｒａ Ｂｌｏｃｋ Ｃｏｐｙ）モード、および、非アフィンインターモードである通常のインターモードのうちの１つであり、
第１のコンテキスト増加手段は、前記第１のブロックが前記ＩＢＣモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第２のコンテキスト増加手段は、前記第１のブロックが前記アフィンインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第３のコンテキスト増加手段は、前記第１のブロックが前記通常のインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
前記第１のコンテキスト増加手段は、１に等しく、
前記第２のコンテキスト増加手段は、２に等しく、
前記第３のコンテキスト増加手段は、０に等しい、命令を格納する非一時的コンピュータ可読記憶媒体。

【請求項１２】
映像のビットストリームを格納するための方法であって、
規則に基づいて、前記映像の第１のブロックのビットストリームを生成することと、
前記ビットストリームを、非一時的コンピュータ可読記録媒体に格納することと、

を有し、
前記生成することは、ＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づき、
前記規則は、ＡＭＶＲシフトに関連付けられた動きベクトル差分の解像度を規定する第１の構文要素のビンの文字列における第１のビンに対するコンテキスト増加手段（ｃｔｘＩｎｃ）の選択が、前記第１のブロックに対するコーディングモードの使用に基づいて導出されることを規定し、
前記第１のブロックに対する前記コーディングモードは、アフィンインターモード、ＩＢＣ（Ｉｎｔｒａ Ｂｌｏｃｋ Ｃｏｐｙ）モード、および、非アフィンインターモードである通常のインターモードのうちの１つであり、
第１のコンテキスト増加手段は、前記第１のブロックが前記ＩＢＣモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第２のコンテキスト増加手段は、前記第１のブロックが前記アフィンインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
第３のコンテキスト増加手段は、前記第１のブロックが前記通常のインターモードを用いてコーディングされる場合に、前記第１のビンに割り当てられ、
前記第１のコンテキスト増加手段は、１に等しく、
前記第２のコンテキスト増加手段は、２に等しく、
前記第３のコンテキスト増加手段は、０に等しい、方法。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願の相互参照
本願は、２０２０年５月１日出願の国際特許出願第ＰＣＴ／ＣＮ２０２０／０８８５４６号の優先権および利益を主張する２０２１年５月６日出願の国際特許出願第ＰＣＴ／ＣＮ２０２１／０９１８６９号に基づく。上記出願の開示全体は、参照によりここに援用される。
【０００２】
本明細書は、映像および画像コーディング技術に関する。
【背景技術】
【０００３】
デジタル映像は、インターネットおよび他のデジタル通信ネットワークにおいて最大の帯域幅の使用量を占めている。映像を受信および表示することが可能である接続されたユーザ機器の数が増加するにつれ、デジタル映像の使用に対する帯域幅需要は増大し続けることが予測される。
【発明の概要】
【０００４】
開示された技術は、コンテキストベースの符号化および復号を使用して符号化または復号を行うために、映像または画像またはエンコーダの実施形態によって使用されてもよい。
【０００５】
１つの例示的な態様において、映像を処理する方法が開示される。この方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、変換は、映像ブロックに対する動きベクトルまたは動きベクトル差分または動きベクトル予測子の表現が、適応解像度を用いてコーディングされた表現において表されるＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づいて行われ、フォーマット規則は、映像ブロックまたは映像ブロックの近傍ブロックのコーディングされた情報に依存するコンテキストモデリングによって、コーディング表現において適応解像度の使用を表現することを規定する。
【０００６】

別の例示的な態様において、別の映像を処理する方法が開示される。この方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、変換は映像ブロックに対する動きベクトルまたは動きベクトル差分または動きベクトル予測子の表現が、適応解像度を使用してコーディング表現において表されるＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づいて行われ、フォーマット規則は、ＡＭＶＲツールによって使用される精度のインデックスのための第１のビンおよび第２のビンをコーディングするために使用されるコンテキストモデリングによって、コーディングされた表現における適応解像度の使用を表現する方法を規定する。

【０００７】

別の例示的な態様において、別の映像を処理する方法が開示される。この方法は、複数の映像ブロックからなる１または複数の映像ピクチャを含む映像と、映像のコーディングされた表現との間の変換を実行することを含み、コーディング表現は、１または複数の映像ブロックのＡＭＶＲ（Ａｄａｐｔｉｖｒ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）コーディングに関する情報を信号通知するためのフォーマット規則に準拠し、フォーマット規則は、第１のコーディングモードを使用してコーディングされた第１の映像ブロックのＡＭＶＲ精度インデックスのビンと、第２のコーディングモードを使用してコーディングされた第２の映像ブロックのＡＭＶＲ精度インデックスのビンとを、同一のコンテキストを使用してコーディングすることを規定する。

【０００８】

別の例示的な態様において、別の映像を処理する方法が開示される。この方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、映像ブロックは、１または複数の垂直および／または１または複数の水平分割に分割され、コーディングされた表現は、映像ブロックの分割情報のコンテキストベースのコーディングを規定するフォーマット規則に準拠する。

【０００９】

別の例示的な態様において、別の映像を処理する方法が開示される。この方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、フォーマット規則は、変換係数の符号を示すためにコンテキストコーディングまたはバイパスコーディングのいずれを使用するかを決定するために使用されるコーディング条件を規定する。

【００１０】

別の例示的な態様において、別の映像を処理する方法が開示される。この方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、フォーマット規則は、変換スキップ残差コーディング処理の第３または残差係数走査パスにおける残りの構文要素のバイパスコーディングの開始時に、残りの許可されたコンテキストコーディングされたビンの数を規定する変数に処理が適用されることを規定する。

【００１１】

別の例示的な態様において、上述された方法は、プロセッサを含む映像エンコーダ装置によって実装されてもよい。

【００１２】

さらに別の例示的な態様において、これらの方法は、プロセッサ実行可能命令の形式で実施されてもよく、コンピュータ可読プログラム媒体に記憶されてもよい。

【００１３】

これらの、および他の態様は、本明細書でさらに説明される。

【図面の簡単な説明】

【００１４】

【図１】エンコーダブロック図の例を示す。

【図２】６７個のイントラ予測モードの例を示す。

【図３Ａ】４パラメータアフィンモデルの例を示す。
【図３Ｂ】６パラメータアフィンモデルの例を示す。
【図４】サブブロックごとのアフィンＭＶＦの例を示す。
【図５】継承されたアフィン動き予測子の一の例を示す。
【図６】制御点動きベクトル継承の例を示す。
【図７】構成されたアフィンマージモードのための候補位置の例を示す。
【図８】提案された結合された方法のための動きベクトル使用の説明図である。
【図９】サブブロックＭＶ　V_{SB}および画素Δｖ（ｉ，ｊ）の例を示す。（赤矢印）
【図１０】マルチタイプのツリー分割モードを例示する。
【図１１】ネストされたマルチタイプのツリーコーディングツリー構造を有する４分木における分割フラグの信号通知の例を示す。
【図１２】映像処理システム例のブロック図である。
【図１３】映像処理装置の例を示す。
【図１４】映像処理方法の例を示すフローチャートである。
【図１５】本開示のいくつかの実施形態による映像コーディングシステムを示すブロック図である。
【図１６】本発明のいくつかの実施形態によるエンコーダを示すブロック図である。
【図１７】本発明のいくつかの実施形態によるデコーダを示すブロック図である。
【図１８】本技術の１または複数の実施形態にしたがった映像処理方法を示すフローチャートである。
【図１９】本技術の１または複数の実施形態にしたがった別の映像処理方法を示すフローチャートである。
【図２０】本技術の１または複数の実施形態にしたがったさらに別の映像処理方法を示すフローチャートである。
【発明を実施するための形態】
【００１５】
本明細書は、展開または復号されたデジタル映像または画像の品質を向上させるために、画像または映像ビットストリームのデコーダによって使用できる様々な技術を提供する。簡潔にするために、本明細書では、用語「映像」は、一連のピクチャ（従来から映像と呼ばれる）および個々の画像の両方を含むように使用される。さらに、映像エンコーダは、さらなる符号化に使用される復号されたフレームを再構成するために、符号化の処理中にこれらの技術を実装してもよい。
【００１６】
本明細書では、理解を容易にするために章の見出しを使用しており、１つの章に開示された実施形態をその章にのみ限定するものではない。このように、ある章の実施形態は、他の章の実施形態と組み合わせることができる。
【００１７】
１．　概要
本明細書は、映像コーディング技術に関する。具体的には、画像／映像コーディングにおけるＡＭＶＲ（Ａｄａｐｔｉｖｅ　Ｍｏｔｉｏｎ　Ｖｅｃｔｏｒ　Ｒｅｓｏｌｕｔｉｏｎ）、ブロック分割などのコーディングツールに関連する。ＨＥＶＣのような既存の映像コーディング規格に適用してもよいし、規格（Ｖｅｒｓａｔｉｌｅ　Ｖｉｄｅｏ　Ｃｏｄｉｎｇ）を確定させるために適用してもよい。本発明は、将来の映像コーディング規格または映像コーデックにも適用可能である。
【００１８】
２．　初期の協議
映像コーディング規格は、主に周知のＩＴＵ－ＴおよびＩＳＯ／ＩＥＣ規格の開発によって発展してきた。ＩＴＵ－ＴはＨ．２６１とＨ．２６３を作り、ＩＳＯ／ＩＥＣはＭＰＥＧ－１とＭＰＥＧ－４　Ｖｉｓｕａｌを作り、両団体はＨ．２６２／ＭＰＥＧ－２　ＶｉｄｅｏとＨ．２６４／ＭＰＥＧ－４　ＡＶＣ（Ａｄｖａｎｃｅｄ　Ｖｉｄｅｏ　Ｃｏｄｉｎ

ｇ）とＨ．２６５／ＨＥＶＣ規格を共同で作った。Ｈ．２６２以来、映像コーディング規格は、時間予測と変換コーディングが利用されるハイブリッド映像コーディング構造に基づく。ＨＥＶＣを超えた将来の映像コーディング技術を探索するため、２０１５年には、ＶＣＥＧとＭＰＥＧが共同でＪＶＥＴ（Ｊｏｉｎｔ Ｖｉｄｅｏ Ｅｘｐｌｏｒａｔｉｏｎ Ｔｅａｍ）を設立した。それ以来、多くの新しい方法がＪＶＥＴによって採用され、ＪＥＭ（Ｊｏｉｎｔ Ｅｘｐｌｏｒａｔｉｏｎ Ｍｏｄｅ）と呼ばれる参照ソフトウェアに組み込まれてきた。２０１８年４月には、ＶＣＥＧ（Ｑ６／１６）とＩＳＯ／ＩＥＣ ＪＴＣ１ＳＣ２９／ＷＧ１１（ＭＰＥＧ）の間にＪｏｉｎｔ Ｖｉｄｅｏ Ｅｘｐｅｒｔ Ｔｅａｍ（ＪＶＥＴ）が発足し、ＨＥＶＣと比較して５０％のビットレート削減を目標にＶＶＣ規格の策定に取り組んでいる。

【００１９】

２．１． 典型的な映像コーデックのコーディングフロー

図１は、３つのインループフィルタリングブロック、すなわちＤＦ（Ｄｅｂｌｏｃｋｉｎｇ Ｆｉｌｔｅｒ）、ＳＡＯ（Ｓａｍｐｌｅ Ａｄａｐｔｉｖｅ Ｏｆｆｓｅｔ）およびＡＬＦを含むＶＶＣのエンコーダブロック図の例を示す。予め定義されたフィルタを使用するＤＦとは異なり、ＳＡＯおよびＡＬＦは、現在のピクチャのオリジナルサンプルを利用し、オフセットおよびフィルタ係数を信号通知するコーディングされた側の情報を用いて、それぞれ、オフセットを追加することにより、および、ＦＩＲ（Ｆｉｎｉｔｅ Ｉｍｐｕｌｓｅ Ｒｅｓｐｏｎｓｅ）フィルタを適用することにより、元のサンプルと再構成サンプルとの間の平均二乗誤差を低減する。ＡＬＦは、各ピクチャの最後の処理段階に位置し、前の段階で生成されたアーチファクトを捕捉し、修正しようとするツールと見なすことができる。

【００２０】

２．２． ６７個のイントラ予測モードを有するイントラモードコーディング

自然映像に表される任意のエッジ方向をキャプチャするために、指向性イントラモードの数は、ＨＥＶＣで使用されるように、３３から６５に拡張される。追加の指向性モードは、図２において赤い点線の矢印で示され、平面モードとＤＣモードは同じままである。これらのより密度の高い指向性イントラ予測モードは、すべてのブロックサイズ、および輝度および彩度イントラ予測の両方に適用される。

【００２１】

従来の角度イントラ予測方向は、図２に示すように、時計回り方向に４５度から－１３５度まで規定される。ＶＴＭにおいて、いくつかの従来の角度イントラ予測モードは、非正方形のブロックのために、広角イントラ予測モードに適応的に置き換えられる。置換されたモードは、元の方法を使用して信号通知され、構文解析後、広角モードのインデックスに再マッピングされる。イントラ予測モードの総数は変化せず、例えば、６７であり、イントラモードコーディングは変化しない。

【００２２】

ＨＥＶＣにおいて、すべてのイントラコーディングされたブロックは正方形の形状を有し、その辺の各々の長さは２の累乗である。このように、ＤＣモードを使用してイントラ予測子を生成するのに、除算演算を必要としない。ＶＶＣにおいて、ブロックは、一般的な場合、ブロックごとに除算演算を使用することが必要な長方形を有することがある。ＤＣ予測のための除算演算を回避するために、長辺のみを使用して非正方形のブロックの平均を計算する。

【００２３】

２．３． インター予測

各インター予測ＣＵに対し、動きベクトル、参照ピクチャインデックス、および参照ピクチャリスト使用インデックスで構成される動きパラメータ、並びにＶＶＣの新しいコーディング特徴に必要な追加情報が、インター予測サンプル生成に使用される。動きパラメータは、明示的または暗示的に信号通知されてもよい。ＣＵがスキップモードでコーディングされる場合、ＣＵは１つのＰＵに関連付けられ、有意な残差係数、コーディングされ

た動きベクトル差分、または参照ピクチャインデックスを有さない。マージモードが規定され、これにより、空間的および時間的候補、並びにＶＶＣに導入された追加のスケジュールを含む、現在のＣＵのための動きパラメータを、近傍のＣＵから取得する。マージモードは、スキップモードのためだけでなく、任意のインター予測されたＣＵに適用することができる。マージモードの代替案は、動きパラメータを明確に送信することであり、動きベクトル、各参照ピクチャリストおよび参照ピクチャリスト使用フラグに対応する参照ピクチャインデックス、並びに他の必要な情報が、ＣＵごとに明確に信号通知される。

【００２４】

２．４．　ＩＢＣ（Ｉｎｔｒａ　Ｂｌｏｃｋ　Ｃｏｐｙ）

ＩＢＣ（Ｉｎｔｒａ　Ｂｌｏｃｋ　Ｃｏｐｙ）は、ＳＣＣのＨＥＶＣ拡張に採用されているツールである。これにより、スクリーンコンテンツマテリアルのコーディング効率が有意に向上することが知られている。ＩＢＣモードはブロックレベルコーディングモードとして実装されるので、ＢＭ（Ｂｌｏｃｋ　Ｍａｔｃｈｉｎｇ）が、エンコーダにおいて実行され、ＣＵごとに最適なブロックベクトル（または動きベクトル）を見出す。ここで、ブロックベクトルは、現在のブロックから、現在のピクチャの内部で既に再構成された参照ブロックへの変位を示すために使用される。ＩＢＣコーディングされたＣＵの輝度ブロックベクトルは、整数精度である。彩度ブロックベクトルは、整数精度にも丸められる。ＡＭＶＲと組み合わせた場合、ＩＢＣモードは、１画素と４画素の動きベクトル精度を切り替えることができる。ＩＢＣコーディングされたＣＵは、イントラ予測モードまたはインター予測モード以外の第３の予測モードとして扱われる。ＩＢＣモードは、幅および高さの両方が６４の輝度サンプル以下のＣＵに適用可能である。

【００２５】

エンコーダ側では、ＩＢＣのためにハッシュベースの動き推定が実行される。エンコーダは、１６の輝度サンプル以下の幅または高さを有するブロックに対してＲＤチェックを行う。非マージモードの場合、まず、ハッシュベースの検索を使用してブロックベクトル検索が実行される。ハッシュ検索が有効な候補を返さない場合、ブロックマッチングベースの局所検索が実行される。

【００２６】

ハッシュベースの検索において、現在のブロックと参照ブロックとのハッシュキーマッチング（３２ビットＣＲＣ）は、全ての許容されるブロックサイズに拡大される。現在のピクチャにおけるすべての位置のためのハッシュキーの計算は、４×４のサブブロックに基づく。現在のブロックのサイズがより大きい場合、すべての４×４のサブブロックのすべてのハッシュキーが対応する参照位置のハッシュキーに合致する場合には、ハッシュキーは参照ブロックのそれに合致すると決定される。複数の参照ブロックのハッシュキーが現在のブロックのハッシュキーに合致すると分かった場合、合致した各参照ブロックのブロックベクトルコストを計算し、最小限のコストを有するものを選択する。ブロックマッチング検索において、検索範囲は前のＣＴＵおよび現在のＣＴＵの両方をカバーするように設定される。

【００２７】

ＣＵレベルにおいて、ＩＢＣモードはフラグで信号通知され、ＩＢＣ　ＡＭＶＰモードまたはＩＢＣスキップ／マージモードとして以下のように信号通知され得る。

【００２８】

－ＩＢＣスキップ／マージモード：マージ候補インデックスを使用して、近傍の候補ＩＢＣコーディングされたブロックからのリストにおいて、どのブロックベクトルを使用して現在のブロックを予測するかを示す。マージリストは、空間候補、ＨＭＶＰ候補、およびペアワイズ候補からなる。

【００２９】

－ＩＢＣ　ＡＭＶＰモード：ブロックベクトル差分を動きベクトル差分と同様にコーディングする。ブロックベクトル予測方法は、２つの候補を予測子として使用し、１つは左の近傍からのものであり、１つは上の近傍のものである（ＩＢＣコーディングされている

場合）。いずれかの近傍が利用可能でない場合、デフォルトのブロックベクトルが予測子として使用される。ブロックベクトル予測子インデックスを示すように、フラグが信号通知される。

【００３０】

２．５．　アフィン動き補償予測

ＨＥＶＣにおいて、ＭＣＰ（Ｍｏｔｉｏｎ　Ｃｏｍｐｅｎｓａｔｉｏｎ　Ｐｒｅｄｉｃｔｉｏｎ）のために並進運動モデルのみが適用される。一方、現実世界において、動きには様々な種類があり、例えば、ズームイン／ズームアウト、回転、透視運動、および他の不規則な動きがある。ＶＶＣにおいて、ブロックベースのアフィン変換動き補償予測が適用される。図３Ａから図３Ｂに示すように、ブロックのアフィン動きフィールドは、２つの制御点の動き情報（４パラメータ）または３つの制御点動きベクトル（６パラメータ）によって説明される。

【００３１】

図６は、制御点動きベクトル継承の例を示す。

【００３２】

４パラメータアフィンモーションモデルの場合、ブロック内のサンプル位置（ｘ，ｙ）の動きベクトルは以下のように導出される。

【００３３】

【数１】

$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{W}x + \frac{mv_{1y}-mv_{0y}}{W}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{W}x + \frac{mv_{1y}-mv_{0x}}{W}y + mv_{0y} \end{cases} \quad (2\text{-}1)$$

【００３４】

６パラメータアフィンモーションモデルの場合、ブロック内のサンプル位置（ｘ，ｙ）の動きベクトルは以下のように導出される。

【００３５】

【数２】

$$\begin{cases} mv_x = \frac{mv_{1x}-mv_{0x}}{W}x + \frac{mv_{2x}-mv_{0x}}{H}y + mv_{0x} \\ mv_y = \frac{mv_{1y}-mv_{0y}}{W}x + \frac{mv_{2y}-mv_{0y}}{H}y + mv_{0y} \end{cases} \quad (2\text{-}2)$$

【００３６】

ここで、（mv_{0x}，mv_{0y}）は左上隅制御点の動きベクトル、（mv_{1x}，mv_{1y}）は右上隅の制御点の動きベクトル、（mv_{2x}，mv_{2y}）は左下隅の制御点の動きベクトルである。

【００３７】

動き補償予測を簡単にするために、ブロックに基づくアフィン変換予測が適用される。各４×４の輝度サブブロックの動きベクトルを導出するために、各サブブロックの中心サンプルの動きベクトルを、図４に示すように、上記方程式に従って算出し、１／１６の端数精度に丸める。そして、動き補償補間フィルタを適用し、導出された動きベクトルを用いて各サブブロックの予測を生成する。また、彩度成分のサブブロックサイズは４×４に設定される。４×４の彩度サブブロックのＭＶは、対応する４×４の輝度サブブロックのＭＶの平均値として計算される。

【００３８】

並進動きインター予測と同様に、アフィンマージモードとアフィンＡＭＶＰモードの２つのアフィン動きインター予測がある。

【００３９】

２．５．１．　アフィンマージ予測
　ＡＦ＿ＭＥＲＧＥモードを、幅および高さの両方が８以上のＣＵに適用することができる。このモードでは、空間的近傍のＣＵの動き情報に基づいて、現在のＣＵのＣＰＭＶを生成する。ＣＰＭＶＰ候補は最大５つまであり、インデックスは、現在のＣＵに使用されるべきものを示すように信号通知される。以下の３種類のＣＰＶＭ候補を使用して、アフィンマージ候補リストを形成する。
　－近傍のＣＵのＣＰＭＶから外挿した継承されたアフィンマージ候補
　－近傍のＣＵの並進ＭＶを使用して導出された構築されたアフィンマージ候補ＣＰＭＶＰ
　－ゼロＭＶ
【００４０】
　ＶＶＣにおいて、近傍のブロックのアフィン動きモデルに由来する最大２つの継承されたアフィン候補があり、１つは左の近傍のＣＵから、１つは上の近傍のＣＵからである。候補ブロックは図５に示す。左の予測子の場合、スキャン順序はＡ０－＞Ａ１であり、上の予測子の場合、スキャン順序はＢ０－＞Ｂ１－＞Ｂ２である。各側から１つ目の継承された候補のみを選択する。２つの継承された候補間でプルーニングチェックは行われない。近傍のアフィンＣＵが識別されると、その制御点動きベクトルを使用して、現在のＣＵのアフィンマージリストにおけるＣＰＭＶＰ候補を導出する。図に示すように、近傍の左下のブロックＡがアフィンモードでコーディングされる場合、ブロックＡを含むＣＵの左上隅、右上隅、左下隅の動きベクトルv_2、v_3、v_4が得られる。４パラメータアフィンモデルでコーディングする場合、v_2およびv_3により現在のユニットの２つのＣＰＭＶを算出する。ブロックＡが６パラメータアフィンモデルでコーディングされる場合、v_2、v_3およびv_4に基づいて、現在のＣＵの３つのＣＰＭＶを算出する。
【００４１】
　構築されたアフィン候補は、各制御点の近傍並進運動情報を組み合わせて候補を構築することを意味する。図７に示される特定された空間的近傍および時間的近傍から制御点の動きを導出する。$CPMV_k$（ｋ＝１，２，３，４）は、ｋ番目の制御点を表す。$CPMV_1$の場合、Ｂ２－＞Ｂ３－＞Ａ２ブロックがチェックされ、第１の使用可能なブロックのＭＶが使用される。$CPMV_2$の場合、Ｂ１→Ｂ０ブロックがチェックされ、$CPMV_3$のために、Ａ１→Ａ０ブロックがチェックされる。使用可能であれば、$CPMV_4$としてＴＭＶＰが使用される。
【００４２】
　４つの制御点のＭＶに達した後、その動き情報に基づいてアフィンマージ候補を構築する。制御点ＭＶの以下の組み合わせを使用して順番に構築する。
｛$CPMV_1$，$CPMV_2$，$CPMV_3$｝，｛$CPMV_1$，$CPMV_2$，$CPMV_4$｝，｛$CPMV_1$，$CPMV_3$，$CPMV_4$｝，｛$CPMV_2$，$CPMV_3$，$CPMV_4$｝，｛$CPMV_1$，$CPMV_2$｝，｛$CPMV_1$，$CPMV_3$｝
【００４３】
　３つのＣＰＭＶの組み合わせは、６パラメータアフィンマージ候補を構成し、２つのＣＰＭＶの組み合わせは、４パラメータアフィンマージ候補を構成する。動きスケーリングプロセスを回避するために、制御点の基準指標が異なる場合、制御点ＭＶの関連する組み合わせを廃棄する。
【００４４】
　継承されたアフィンマージ候補および構築されたアフィンマージ候補をチェックした後、リストがまだ満杯でない場合、ゼロＭＶをリストの末端に挿入する。
【００４５】
２．５．２．　アフィンＡＭＶＰ予測
　アフィンＡＭＶＰモードを、幅および高さの両方が１６以上のＣＵに適用することができる。アフィンＡＭＶＰモードが使用されるかどうかを示すために、ＣＵレベルのアフィンフラグがビットストリームにおいて信号通知され、次いで、４パラメータアフィンであるか６パラメータアフィンであるかどうかを示すために、別のフラグが信号通知される。

このモードにおいて、現在のＣＵのＣＰＭＶとその予測子ＣＰＭＶＰとの差がビットストリームにおいて信号通知される。アフィンＡＶＭＰ候補リストサイズは２であり、以下の４つのタイプのＣＰＶＭ候補を順に使用して生成される。

－近傍のＣＵのＣＰＵＭＶから外挿した継承されたアフィンＡＭＶＰ候補

－近傍のＣＵの並進ＭＶを使用して導出された構築されたアフィンＡＭＶＰ候補ＣＰＭＶＰ

－近傍のＣＵからの並進ＭＶ

－ゼロＭＶ

【００４６】

継承されたアフィンＡＭＶＰ候補のチェック順は、継承されたアフィンマージ候補のチェック順と同じである。唯一の違いは、ＡＶＭＰ候補の場合、現在のブロックと同じ参照ピクチャを有するアフィンＣＵのみを考慮することである。継承されたアフィン動き予測子を候補リストに挿入する場合、プルーニング処理は適用されない。

【００４７】

構築されたＡＭＶＰ候補は、図７に示す規定された空間的近傍から導出される。アフィンマージ候補構築で行ったものとして、同じチェック順が使用される。また、近傍のブロックの参照ピクチャインデックスもチェックする。インターコーディングされ、かつ、現在のＣＵと同じ参照ピクチャを有する、チェック順の第１のブロックが使用される。現在のＣＵが４パラメータアフィンモードでコーディングされ、かつ、mv_0およびmv_1が両方とも利用可能である場合、それらをアフィンＡＭＶＰ一覧に１つの候補として追加する。現在のＣＵが６パラメータアフィンモードでコーディングされ、かつ、３つのＣＰＭＶすべてが利用可能である場合、それらをアフィンＡＭＶＰリストにおける１つの候補として追加する。そうでない場合、構築されたＡＭＶＰ候補を利用不可能に設定する。

【００４８】

継承されたアフィンＡＭＶＰ候補および構築されたＡＭＶＰ候補をチェックした後、アフィンＡＭＶＰ一覧候補が依然として２未満である場合、利用可能であれば、mv_0、mv_1、およびmv_2の順に、現在のＣＵのすべての制御点ＭＶを予測する並進ＭＶとして追加される。最後に、まだアフィンＡＭＶＰリストがすべて満たされていない場合は、満たすためにゼロＭＶを使用する。

【００４９】

２．５．３．　アフィン動き情報記憶域

ＶＶＣにおいて、アフィンＣＵのＣＰＵＭＶは、別個のバッファに記憶される。記憶されたＣＰＭＶは、最近コーディングされたＣＵのために、アフィンマージモードおよびアフィンＡＭＶＰモードで継承されたＣＰＭＶＰを生成するためだけに用いられる。ＣＰＭＶから導出されたサブブロックＭＶは、動き補償、並進ＭＶのマージ／ＡＭＶＰリストのＭＶ導出、およびデブロッキングに用いられる。

【００５０】

追加のＣＰＵＭＶのためのピクチャラインバッファを回避するために、上のＣＴＵからのＣＵからのアフィン動きデータの継承は、通常の近傍のＣＵからの継承とは異なるように扱われる。アフィン動きデータ継承の候補ＣＵが上のＣＴＵラインにある場合、アフィンＭＶＰの導出には、ＣＰＭＶの代わりに、ラインバッファにおける左下および右下のサブブロックＭＶを用いる。このようにして、ＣＰＭＶはローカルバッファにのみ記憶される。候補ＣＵが６パラメータアフィンコーディングされている場合、アフィンモデルは４パラメータモデルに低下される。図８に示すように、ＣＴＵの上限に沿って、ＣＵの左下および右下のサブブロック動きベクトルを用いて、下限のＣＴＵにおけるＣＵをアフィン継承する。

【００５１】

２．５．４．　アフィンモードのためのオプティカルフローによる予測改善

サブブロック・ベースのアフィン動き補償は、予測精度の犠牲を払って、メモリアクセス帯域幅を節約し、ピクセル・ベースの動き補償に比べて計算の複雑さを低減することが

できる。動き補償のより微細な粒度を実現するために、ＰＲＯＦ（Ｐｒｅｄｉｃｔｉｏｎ Ｒｅｆｉｎｅｍｅｎｔ ｗｉｔｈ Ｏｐｔｉｃａｌ Ｆｌｏｗ）は、動き補償のためのメモリアクセス帯域幅を増加させることなく、サブブロックに基づくアフィン動き補償予測を改善するために用いられる。ＶＶＣにおいて、サブブロックに基づくアフィン動き補償を行った後、オプティカルフロー方程式で導出された差を加算することで、輝度予測サンプルを微調整する。ＰＲＯＦは、以下の４つのステップとして説明される。

【００５２】

ステップ１）サブブロックに基づくアフィン動き補償を行い、サブブロック予測Ｉ（ｉ，ｊ）を生成する。

ステップ２）３タップフィルタ［－１，０，１］を使用して、個々のサンプル位置において、サブブロック予測の空間的勾配g_x（ｉ，ｊ）およびg_y（ｉ，ｊ）を算出する。勾配計算は、ＢＤＯＦの勾配計算と全く同じである。

【００５３】

【数３】

$$g_x(i,j) = (I(i+1,j) \gg shift1) - (I(i-1,j) \gg shift1) \qquad (2\text{-}3)$$

$$g_y(i,j) = (I(i,j+1) \gg shift1) - (I(i,j-1) \gg shift1) \qquad (2\text{-}4)$$

【００５４】

ｓｈｉｆｔ１は勾配の精度を制御するために使用される。サブブロック（例えば、４×４）予測は、勾配計算のために各側で１つのサンプルだけ拡大される。付加的なメモリ帯域幅および付加的な補間計算を避けるために、拡大された境界線上のこれらの拡大されたサンプルは、参照ピクチャにおける最も近い整数ピクセル位置からコピーされる。

【００５５】

ステップ３）以下のオプティカルフロー方程式により輝度予測改善を算出する。

【００５６】

【数４】

$$\Delta I(i,j) = g_x(i,j) * \Delta v_x(i,j) + g_y(i,j) * \Delta v_y(i,j) \qquad (2\text{-}5)$$

【００５７】

ここで、Δｖ（ｉ，ｊ）は、図９に示すように、ｖ（ｉ，ｊ）で表される、サンプル位置（ｉ，ｊ）のため０に算出されたサンプルＭＶと、サンプル（ｉ，ｊ）が属するサブブロックのサブブロックＭＶとの差分である。このΔｖ（ｉ，ｊ）は、１／３２輝度サンプル精度の単位で量子化される。

【００５８】

サブブロック中心に対するアフィンモデルパラメータおよびサンプル位置はサブブロックごとに変化しないので、第１のサブブロックについてΔｖ（ｉ，ｊ）を計算し、同じＣＵ内の他のサブブロックに再利用することができる。ｄｘ（ｉ，ｊ）およびｄｙ（ｉ，ｊ）を、サンプル位置（ｉ，ｊ）からサブブロック（x_{SB}，y_{SB}）の中心までの水平および垂直のオフセットであるとすると、Δｖ（ｘ，ｙ）は、以下の式で導出することができる。

【００５９】

【数５】

$$\begin{cases} \mathrm{dx}(i,j) = i - x_{SB} \\ \mathrm{dy}(i,j) = j - y_{SB} \end{cases} \tag{3-6}$$

$$\begin{cases} \Delta v_x(i,j) = C * dx(i,j) + D * dy(i,j) \\ \Delta v_y(i,j) = E * dx(i,j) + F * dy(i,j) \end{cases} \tag{3-7}$$

【００６０】
精度を維持するために、サブブロック（ｘＳＢ，ｙＳＢ）の中心は、（（ＷＳＢ－１）／２，（ＨＳＢ－１）／２）として計算され、ここで、ＷＳＢおよびＨＳＢは、それぞれ、サブブロックの幅および高さである。

【００６１】
４パラメータアフィンモデルの場合、

【００６２】
【数６】

$$\begin{cases} C = F = \frac{v_{1x} - v_{0x}}{w} \\ E = -D = \frac{v_{1y} - v_{0y}}{w} \end{cases} \tag{3-8}$$

【００６３】
６パラメータアフィンモデルの場合、

【００６４】
【数７】

$$\begin{cases} C = \frac{v_{1x} - v_{0x}}{w} \\ D = \frac{v_{2x} - v_{0x}}{h} \\ E = \frac{v_{1y} - v_{0y}}{w} \\ F = \frac{v_{2y} - v_{0y}}{h} \end{cases} \tag{3-9}$$

【００６５】
ここで、（ｖ０ｘ，ｖ０ｙ）、（ｖ１ｘ，ｖ１ｙ）、（ｖ２ｘ，ｖ２ｙ）、は左上、右上、左下の制御点動きベクトルであり、ｗ、ｈはＣＵの幅および高さである。

【００６６】
ステップ４）最後に、サブブロック予測Ｉ（ｉ，ｊ）に輝度予測の改善ΔＩ（ｉ，ｊ）を加える。最終予測Ｉ’は、次の方程式のように生成される。

【００６７】
【数８】

$$I'(i,j) = I(i,j) + \Delta I(i,j) \tag{3-10}$$

【００６８】
ＰＲＯＦは、アフィンコーディングされたＣＵの場合、２つのケースで適用されない。１）すべての制御点ＭＶは同じであり、これは、ＣＵが並進運動のみを有することを示す

。２）サブブロックに基づくアフィンＭＣは、大きなメモリアクセス帯域幅要件を回避するために、ＣＵに基づくＭＣに劣化するので、アフィン運動パラメータは、規定された制限よりも大きい。

【００６９】

ＰＲＯＦを用いたアフィン動き推定の符号化の複雑度を低減するために、高速符号化方法が適用される。次の２つの状況、即ち、ａ）このＣＵがルートブロックでなく、その親ブロックがアフィンモードをそのベストモードとして選択しない場合、現在のＣＵがアフィンモードをベストモードとして選択する可能性が低いので、ＰＲＯＦは適用されず、ｂ）４つのアフィンパラメータ（Ｃ、Ｄ、Ｅ、Ｆ）の大きさがすべて予め定義された閾値よりも小さく、現在のピクチャが低遅延ピクチャでない場合、ＰＲＯＦによって導入される改善はこの場合には小さいので、ＰＲＯＦは適用されない。このようにして、ＰＲＯＦによるアフィン動き推定を高速化することができる。

【００７０】

２．６．　ブロック分割の例示的な可用性プロセス

６．４．１　許容されたクアッド分割プロセス

この処理への入力は以下の通りである。

－輝度サンプルにおけるコーディングブロックサイズｃｂＳｉｚｅ、

－マルチタイプツリーの深さｍｔｔＤｅｐｔｈ、

－単一ツリー（ＳＩＮＧＬＥ＿ＴＲＥＥ）またはデュアルツリーを使用してコーディングツリーノードを分割するかどうか、およびデュアルツリーを使用する場合、輝度（ＤＵＡＬ＿ＴＲＥＥ＿ＬＵＭＡ）または彩度成分（ＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡ）を現在処理しているかどうかを規定する変数ｔｒｅｅＴｙｐｅ。

－イントラ（ＭＯＤＥ＿ＩＮＴＲＡ）、ＩＢＣ（ＭＯＤＥ＿ＩＢＣ）、インターコーディングモードを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＡＬＬ）、またはイントラコーディングモードおよびＩＢＣコーディングモードのみを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＲＡ）、またはインターコーディングモードのみを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＥＲ）を規定する変数ｍｏｄｅＴｙｐｅ。

【００７１】

この処理の出力が変数ａｌｌｏｗＳｐｌｉｔＱｔである。

変数ａｌｌｏｗＳｐｌｉｔＱｔが、以下のように導出される。

－以下の条件の１または複数が真である場合、ａｌｌｏｗＳｐｌｉｔＱｔはＦＡＬＳＥに設定される。

－ｔｒｅｅＴｙｐｅがＳＩＮＧＬＥ＿ＴＲＥＥまたはＤＵＡＬ＿ＴＲＥＥ＿ＬＵＭＡに等しく、ｃｂＳｉｚｅがＭｉｎＱｔＳｉｚｅＹ以下である

－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、ｃｂＳｉｚｅが（ＭｉｎＱｔＳｉｚｅＣ＊ＳｕｂＨｅｉｇｈｔＣ／ＳｕｂＷｉｄｔｈＣ）以下である

－ｍｔｔＤｅｐｔｈが０に等しくない

－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、（ｃｂＳｉｚｅ／ＳｕｂＷｉｄｔｈＣ）が４　以下である

－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、ｍｏｄｅＴｙｐｅがＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＲＡに等しい

－そうでない場合、ａｌｌｏｗＳｐｌｉｔＱｔがＴＲＵＥに設定される。

【００７２】

６．４．２　許可されたバイナリ分割処理

この処理への入力は以下の通りである。

－バイナリ分割モードｂｔＳｐｌｉｔ、

－輝度サンプルにおけるコーディングブロック幅ｃｂＷｉｄｔｈ、

－輝度サンプルにおけるコーディングブロックの高さｃｂＨｅｉｇｈｔ、

－ピクチャの左上の輝度サンプルに対する、考慮されるコーディングブロックの左上の輝度サンプル位置（ｘ０，ｙ０）、

－マルチタイプツリーの深さｍｔｔＤｅｐｔｈ、
－ｍａｘＭｔｔＤｅｐｔｈがオフセットされた最大マルチタイプツリー深さ、
－最大２分木サイズｍａｘＢｔＳｉｚｅ、
－最小４分木サイズｍｉｎＱｔＳｉｚｅ、
－分割インデックスｐａｒｔＩｄｘ、
－単一ツリー（ＳＩＮＧＬＥ＿ＴＲＥＥ）またはデュアルツリーを使用してコーディングツリーノードを分割するかどうか、およびデュアルツリーを使用する場合、輝度（ＤＵＡＬ＿ＴＲＥＥ＿ＬＵＭＡ）または彩度成分（ＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡ）を現在処理しているかどうかを規定する変数ｔｒｅｅＴｙｐｅ。
－イントラ（ＭＯＤＥ＿ＩＮＴＲＡ）、ＩＢＣ（ＭＯＤＥ＿ＩＢＣ）、インターコーディングモードを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＡＬＬ）、またはイントラコーディングモードおよびＩＢＣコーディングモードのみを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＲＡ）、またはインターコーディングモードのみを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＥＲ）を規定する変数ｍｏｄｅＴｙｐｅ。

【００７３】

この処理の出力が変数ａｌｌｏｗＢｔＳｐｌｉｔである。

【００７４】

表２－１　ｂｔＳｐｌｉｔに基づくｐａｒａｌｌｅｌＴｔＳｐｌｉｔ、ｃｂＳｉｚｅの仕様

【００７５】

【表１】

	btSplit = = SPLIT_BT_VER	btSplit = = SPLIT_BT_HOR
parallelTtSplit	SPLIT_TT_VER	SPLIT_TT_HOR
cbSize	cbWidth	cbHeight

【００７６】

表２－１に示すように、変数ｐａｒａｌｌｅｌＴｔＳｐｌｉｔおよびｃｂＳｉｚｅを導出する。

変数ａｌｌｏｗＢｔＳｐｌｉｔが、以下のように導出される。

－以下の１または複数の条件が真である場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
　－ｃｂＳｉｚｅがＭｉｎＢｔＳｉｚｅＹ以下である
　－ｃｂＷｉｄｔｈがｍａｘＢｔＳｉｚｅより大きい
　－ｃｂＨｅｉｇｈｔがｍａｘＢｔＳｉｚｅより大きい
　－ｍｔｔＤｅｐｔｈがｍａｘＭｔｔＤｅｐｔｈ以上である
　－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、（ｃｂＷｉｄｔｈ／ＳｕｂＷｉｄｔｈＣ）＊（ｃｂＨｅｉｇｈｔ／ＳｕｂＨｅｉｇｈｔＣ）が１６以下である
　－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、（ｃｂＷｉｄｔｈ／ＳｕｂＷｉｄｔｈＣ）が４に等しく、ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＶＥＲに等しい
　－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、ｍｏｄｅＴｙｐｅがＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＲＡに等しい
　－ｃｂＷｉｄｔｈ＊ｃｂＨｅｉｇｈｔが３２に等しく、ｍｏｄｅＴｙｐｅがＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＥＲに等しい
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
　－ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＶＥＲに等しい
　－ｙ０＋ｃｂＨｅｉｇｈｔがｐｉｃ＿ｈｅｉｇｈｔ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅ

ｓより大きい
【００７７】
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＶＥＲに等しい
－ｃｂＨｅｉｇｈｔが６４より大きい
－ｘ０＋ｃｂＷｉｄｔｈがｐｉｃ＿ｗｉｄｔｈ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大きい
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＨＯＲに等しい
－ｃｂＷｉｄｔｈが６４より大きい
－ｙ０＋ｃｂＨｅｉｇｈｔがｐｉｃ＿ｈｅｉｇｈｔ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大きい
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｘ０＋ｃｂＷｉｄｔｈがｐｉｃ＿ｗｉｄｔｈ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大きい
－ｙ０＋ｃｂＨｅｉｇｈｔがｐｉｃ＿ｈｅｉｇｈｔ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大き　い
－ｃｂＷｉｄｔｈがｍｉｎＱｔＳｉｚｅより大きい
【００７８】
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＨＯＲに等しい
－ｘ０＋ｃｂＷｉｄｔｈがｐｉｃ＿ｗｉｄｔｈ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大きい
－ｙ０＋ｃｂＨｅｉｇｈｔがｐｉｃ＿ｈｅｉｇｈｔ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓ以下である
－そうでない場合に、以下のすべての条件が真である場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに等しく設定される。
－ｍｔｔＤｅｐｔｈが０より大きい
－ｐａｒｔＩｄｘ＝１
－ＭｔｔＳｐｌｉｔＭｏｄｅ［ｘ０］［ｙ０］［ｍｔｔＤｅｐｔｈ－１］は、ｅｑｕａｌｔｏｐａｒａｌｌｅｌＴｔＳｐｌｉｔである。
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＶＥＲに等しい
－ｃｂＷｉｄｔｈが６４以下である
－ｃｂＨｅｉｇｈｔが６４より大きい
－そうでない場合に、以下のすべての条件が満たされている場合、ａｌｌｏｗＢｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｂｔＳｐｌｉｔがＳＰＬＩＴ＿ＢＴ＿ＨＯＲに等しい
－ｃｂＷｉｄｔｈが６４より大きい
－ｃｂＨｅｉｇｈｔが６４以下である
－そうでない場合、ａｌｌｏｗＢｔＳｐｌｉｔはＴＲＵＥに設定される。
【００７９】
６．４．３　許容されたターナリ（ｔｅｒｎａｒｙ）分割処理
この処理への入力は以下の通りである。
－ターナリ分割モードｔｔＳｐｌｉｔ、

－輝度サンプルにおけるコーディングブロック幅ｃｂＷｉｄｔｈ、
－輝度サンプルにおけるコーディングブロックの高さｃｂＨｅｉｇｈｔ、
－ピクチャの左上の輝度サンプルに対する、考慮されるコーディングブロックの左上の輝度サンプル位置（ｘ０，ｙ０）、
－マルチタイプツリー深さｍｔｔＤｅｐｔｈ
－ｍａｘＭｔｔＤｅｐｔｈがオフセットされた最大マルチタイプツリー深さ、
－最大３分木サイズｍａｘＴｔＳｉｚｅ、
－単一ツリー（ＳＩＮＧＬＥ＿ＴＲＥＥ）またはデュアルツリーを使用してコーディングツリーノードを分割するかどうか、およびデュアルツリーを使用する場合、輝度（ＤＵＡＬ＿ＴＲＥＥ＿ＬＵＭＡ）または彩度成分（ＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡ）を現在処理しているかどうかを規定する変数ｔｒｅｅＴｙｐｅ、
－イントラ（ＭＯＤＥ＿ＩＮＴＲＡ）、ＩＢＣ（ＭＯＤＥ＿ＩＢＣ）、インターコーディングモードを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＡＬＬ）、またはイントラコーディングモードおよびＩＢＣコーディングモードのみを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＲＡ）、またはインターコーディングモードのみを使用できるか（ＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＥＲ）を規定する変数ｍｏｄｅＴｙｐｅ。

【００８０】

この処理の出力が変数ａｌｌｏｗＴｔＳｐｌｉｔである。

表２－２　ｔｔＳｐｌｉｔに基づくｃｂＳｉｚｅの仕様

【００８１】

【表２】

	ttSplit = = SPLIT_TT_VER	ttSplit = = SPLIT_TT_HOR
cbSize	cbWidth	cbHeight

【００８２】

表２－２に示すように、変数ｃｂＳｉｚｅを導出する。
変数ａｌｌｏｗＴｔＳｐｌｉｔが、以下のように導出される。
－以下の１または複数の条件が真である場合、ａｌｌｏｗＴｔＳｐｌｉｔはＦＡＬＳＥに設定される。
－ｃｂＳｉｚｅが２＊ＭｉｎＴｔＳｉｚｅＹ以下である
－ｃｂＷｉｄｔｈがＭｉｎ（６４，ｍａｘＴｔＳｉｚｅ）より大きい
－ｃｂＨｅｉｇｈｔがＭｉｎ（６４，ｍａｘＴｔＳｉｚｅ）より大きい
－ｍｔｔＤｅｐｔｈがｍａｘＭｔｔＤｅｐｔｈ以上である
－ｘ０＋ｃｂＷｉｄｔｈがｐｉｃ＿ｗｉｄｔｈ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大きい
－ｙ０＋ｃｂＨｅｉｇｈｔがｐｉｃ＿ｈｅｉｇｈｔ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓより大きい
－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、（ｃｂＷｉｄｔｈ／ＳｕｂＷｉｄｔｈＣ）＊（ｃｂＨｅｉｇｈｔ／ＳｕｂＨｅｉｇｈｔＣ）が３２以下である
－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、（ｃｂＷｉｄｔｈ／ＳｕｂＷｉｄｔｈＣ）が８に等しく、ｔｔＳｐｌｉｔがＳＰＬＩＴ＿ＴＴ＿ＶＥＲに等しい
－ｔｒｅｅＴｙｐｅがＤＵＡＬ＿ＴＲＥＥ＿ＣＨＲＯＭＡに等しく、ｍｏｄｅＴｙｐｅがＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＲＡに等しい
－ｃｂＷｉｄｔｈ＊ｃｂＨｅｉｇｈｔが６４に等しく、ｍｏｄｅＴｙｐｅがＭＯＤＥ＿ＴＹＰＥ＿ＩＮＴＥＲと等しい。
－そうでない場合、ａｌｌｏｗＴｔＳｐｌｉｔがＴＲＵＥに設定される。

【００８３】
６．４．４　近傍のブロック利用可能性の導出処理
　この処理への入力は以下の通りである。
　－現在のピクチャの左上の輝度サンプルに対する現在のブロックの左上のサンプルの輝度位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、
　－現在のピクチャの左上の輝度サンプルに対して近傍のブロックで覆われた輝度位置（ｘＮｂＹ，ｙＮｂＹ）、
　－利用可能性が予測モードに依存するかどうかを規定する変数ｃｈｅｃｋＰｒｅｄＭｏｄｅＹ、
　－現在のブロックの色成分を規定する変数ｃＩｄｘ。
【００８４】
　この処理の出力は、位置（ｘＮｂＹ，ｙＮｂＹ）をカバーする近傍のブロックの利用可能性であり、ａｖａｉｌａｂｌｅＮと表される。
　近傍のブロックの利用可能性ａｖａｉｌａｂｌｅＮが、以下のように導出される。
　－以下の１または複数の条件が真である場合、ａｖａｉｌａｂｌｅＮはＦＡＬＳＥに設定される。
　　－ｘＮｂＹが０未満である。
　　－ｙＮｂＹが０未満である。
　　－ｘＮｂＹがｐｉｃ＿ｗｉｄｔｈ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓ以上である。
　　－ｙＮｂＹがｐｉｃ＿ｈｅｉｇｈｔ＿ｉｎ＿ｌｕｍａ＿ｓａｍｐｌｅｓ以上である。
　　－ＩｓＡｖａｉｌａｂｌｅ［ｃＩｄｘ］［ｘＮｂＹ］［ｙＮｂＹ］はＦＡＬＳＥに等しい。
　　－近傍のブロックが現在のブロックとは異なるスライスに含まれている。
　　－近傍のブロックが現在のブロックとは異なるタイルに含まれている。
　　－ｓｐ＿ｅｎｔｒｏｐｙ＿ｃｏｄｉｎｇ＿ｓｙｎｃ＿ｅｎａｂｌｅｄ＿ｆｌａｇが１に等しく、（ｘＮｂＹ＞ＣｔｂＬｏｇ２ＳｉｚｅＹ）が（ｘＣｕｒｒ＞ＣｔｂＬｏｇ２ＳｉｚｅＹ）＋１以上である。
　－そうでない場合、ａｖａｉｌａｂｌｅＮはＴＲＵＥに設定される。
　以下のすべての条件が真である場合、ａｖａｉｌａｂｌｅＮはＦＡＬＳＥに設定される。
　－ｃｈｅｃｋＰｒｅｄＭｏｄｅＹがＴＲＵＥに等しい。
　－ａｖａｉｌａｂｌｅＮはＴＲＵＥに設定される。
　－ＣｕＰｒｅｄＭｏｄｅ［０］［ｘＮｂＹ］［ｙＮｂＹ］がＣｕＰｒｅｄＭｏｄｅ［０］［ｘＣｕｒｒ］［ｙＣｕｒｒ］に等しくない。
【００８５】
２．７．　ＡＭＶＲ（Ａｄａｐｔｉｖｅ　Ｍｏｔｉｏｎ　Ｖｅｃｔｏｒ　Ｒｅｓｏｌｕｔｉｏｎ）
　ＨＥＶＣにおいて、ｕｓｅ＿ｉｎｔｅｇｅｒ＿ｍｖ＿ｆｌａｇがスライスヘッダにおいて０である場合、１／４輝度サンプルの単位でＭＶＤ（Ｍｏｔｉｏｎ　Ｖｅｃｔｏｒ　Ｄｉｆｆｅｒｅｎｃｅ）（動きベクトルとＣＵの予測動きベクトルとの差）が信号通知される。ＶＶＣにおいて、ＣＵレベルのＡＭＶＲ（Ａｄａｐｔｉｖｅ　Ｍｏｔｉｏｎ　Ｖｅｃｔｏｒ　Ｒｅｓｏｌｕｔｉｏｎ）スキームが導入される。ＡＭＶＲは、ＣＵのＭＶＤを異なる精度でコーディングすることを可能にする。現在のＣＵのモード（通常のＡＭＶＰモードまたはアフィンＡＶＭＰモードまたはＩＢＣモード）に基づいて、現在のＣＵのＭＶＤは、以下のように適応的に選択できる。
　－通常ＡＭＶＰモード：１／４輝度サンプル、１／２輝度サンプル、１輝度サンプルまたは４輝度サンプル
　－アフィンＡＭＶＰモード：１／４輝度サンプル、１輝度サンプル、または１／１６輝度サンプル
　－ＩＢＣモード：１輝度サンプルまたは１／４輝度サンプル
【００８６】

現在のＣＵが少なくとも１つの非ゼロＭＶＤ成分を有する場合、ＣＵレベルＭＶＤ解像度表示が条件付きで通知される。すべてのＭＶＤ成分（すなわち、参照リストＬ０および参照リストＬ１の水平および垂直ＭＶＤの両方）がゼロである場合、１／４輝度サンプルＭＶＤ解像度が推論される。

【００８７】

少なくとも１つの非ゼロＭＶＤ成分を有する、通常のＡＭＶＰインターモード（非ＩＢＣ、非アフィン）でコーディングされたＣＵの場合、１／４輝度サンプルＭＶＤ精度がＣＵに使用されるかどうかを示すために、第１のフラグ（例えば、ａｍｖｒ＿ｆｌａｇ）が信号通知される。第１のフラグが０である場合、さらなる信号伝達は必要とされず、現在のＣＵのために１／４輝度サンプルＭＶＤ精度が使用される。そうでない場合、第２のフラグ（例えば、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン）が、１／２輝度サンプルまたは他のＭＶＤ精度（１輝度サンプルまたは４輝度サンプル）が通常のＡＭＶＰＣＵに使用されることを示すように信号通知される。１／２輝度サンプルの場合、１／２輝度サンプル位置には、デフォルトの８タップ補間フィルタに代えて、６タップ補間フィルタが用いられる。そうでない場合、第３のフラグ（例えば、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第２のビン）は、１つの輝度サンプルまたは４つの輝度サンプルのＭＶＤ精度が通常のＡＭＶＰ＿ＣＵに使用されるかどうかを示すように信号通知される。

【００８８】

アフィンＡＭＶＰモードでコーディングされたＣＵの場合、第２のフラグは、１輝度サンプルまたは１／１６輝度サンプルのＭＶＤ精度が使用されるかどうかを示すために使用される。ＩＢＣモードでコーディングされたＣＵの場合、第１のフラグは信号通知されず、１に等しいと推測される。

【００８９】

ＡＭＶＲの現在の設計において、０に等しいａｍｖｒ＿ｆｌａｇは、動きベクトルの差の解像度が輝度サンプルの１／４であることを規定する。１に等しいａｍｖｒ＿ｆｌａｇは、動きベクトルの差の解像度がａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘによってさらに規定されることを規定する。

【００９０】

ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘは、ＡｍｖｒＳｈｉｆｔとの動きベクトルの差の分解能を表２－３に定義することを規定する。

ＡＭＶＲのための構文テーブル例

７．３．１０．５　コーディングユニット構文

【００９１】

【表３】

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	記述子
chType = treeType = = DUAL_TREE_CHROMA ? 1 : 0	
if(slice_type != I | | sps_ibc_enabled_flag) {	
if(treeType != DUAL_TREE_CHROMA && ((!(cbWidth = = 4 && cbHeight = = 4) && modeType != MODE_TYPE_INTRA) | | (sps_ibc_enabled_flag && cbWidth <= 64 && cbHeight <= 64)))	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] = = 0 && slice_type != I && !(cbWidth = = 4 && cbHeight = = 4) && modeType = = MODE_TYPE_ALL)	
pred_mode_flag	ae(v)
if(((slice_type = = I && cu_skip_flag[x0][y0] = =0) | | (slice_type != I && (CuPredMode[chType][x0][y0] != MODE_INTRA | | (((cbWidth = = 4 && cbHeight = = 4) | | modeType = = MODE_TYPE_INTRA) && cu_skip_flag[x0][y0] = = 0)))) && cbWidth <= 64 && cbHeight <= 64 && modeType != MODE_TYPE_INTER && sps_ibc_enabled_flag && treeType != DUAL_TREE_CHROMA)	
pred_mode_ibc_flag	ae(v)
}	
if(CuPredMode[chType][x0][y0] = = MODE_INTRA && sps_palette_enabled_flag && cbWidth <= 64 && cbHeight <= 64 && cu_skip_flag[x0][y0] = = 0 && modeType != MODE_TYPE_INTER && ((cbWidth * cbHeight) > (treeType != DUAL_TREE_CHROMA ? 16 : 16 * SubWidthC * SubHeightC)))	
pred_mode_plt_flag	ae(v)
if(CuPredMode[chType][x0][y0] = = MODE_INTRA && sps_act_enabled_flag && treeType = = SINGLE_TREE)	
cu_act_enabled_flag	ae(v)

【００９２】

【表４】

if(CuPredMode[chType][x0][y0] = = MODE_INTRA || CuPredMode[chType][x0][y0] = = MODE_PLT) {	
…	
} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER or MODE_IBC */	
if(cu_skip_flag[x0][y0] = = 0)	
general_merge_flag[x0][y0]	ae(v)
if(general_merge_flag[x0][y0])	
merge_data(x0, y0, cbWidth, cbHeight, chType)	
else if(CuPredMode[chType][x0][y0] = = MODE_IBC) {	
mvd_coding(x0, y0, 0, 0)	
if(MaxNumIbcMergeCand > 1)	
mvp_l0_flag[x0][y0]	ae(v)
if(sps_amvr_enabled_flag && *(MvdL0[x0][y0][0] != 0 || MvdL0[x0][y0][1] != 0))*	
amvr_precision_idx*[x0][y0]*	*ae(v)*
} else {	
if(slice_type = = B)	
inter_pred_idc[x0][y0]	ae(v)
if(sps_affine_enabled_flag && cbWidth >= 16 && cbHeight >= 16) {	
inter_affine_flag[x0][y0]	ae(v)
if(sps_affine_type_flag && inter_affine_flag[x0][y0])	
cu_affine_type_flag[x0][y0]	ae(v)
}	
if(sps_smvd_enabled_flag && !mvd_l1_zero_flag && inter_pred_idc[x0][y0] = = PRED_BI && !inter_affine_flag[x0][y0] && RefIdxSymL0 > −1 && RefIdxSymL1 > −1)	
sym_mvd_flag[x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(NumRefIdxActive[0] > 1 && !sym_mvd_flag[x0][y0])	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 0, 1)	

【００９３】

【表５】

if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 0, 2)	
mvp_l0_flag[x0][y0]	ae(v)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(NumRefIdxActive[1] > 1 && !sym_mvd_flag[x0][y0])	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] = = PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
MvdCpL1[x0][y0][0][0] = 0	
MvdCpL1[x0][y0][0][1] = 0	
MvdCpL1[x0][y0][1][0] = 0	
MvdCpL1[x0][y0][1][1] = 0	
MvdCpL1[x0][y0][2][0] = 0	
MvdCpL1[x0][y0][2][1] = 0	
} else {	
if(sym_mvd_flag[x0][y0]) {	
MvdL1[x0][y0][0] = −MvdL0[x0][y0][0]	
MvdL1[x0][y0][1] = −MvdL0[x0][y0][1]	
} else	
mvd_coding(x0, y0, 1, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 1, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 1, 2)	
}	
mvp_l1_flag[x0][y0]	ae(v)

【００９４】

【表６】

} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
}	
if((sps_amvr_enabled_flag && inter_affine_flag[x0][y0] = = 0 && *(MvdL0[x0][y0][0] != 0 | | MvdL0[x0][y0][1] != 0 | |* *MvdL1[x0][y0][0] != 0 | | MvdL1[x0][y0][1] != 0)) | |* *(sps_affine_amvr_enabled_flag && inter_affine_flag[x0][y0] = = 1 &&* *(MvdCpL0[x0][y0][0][0] != 0 | | MvdCpL0[x0][y0][0][1] != 0 | |* *MvdCpL1[x0][y0][0][0] != 0 | | MvdCpL1[x0][y0][0][1] != 0 | |* *MvdCpL0[x0][y0][1][0] != 0 | | MvdCpL0[x0][y0][1][1] != 0 | |* *MvdCpL1[x0][y0][1][0] != 0 | | MvdCpL1[x0][y0][1][1] != 0 | |* *MvdCpL0[x0][y0][2][0] != 0 | | MvdCpL0[x0][y0][2][1] != 0 | |* *MvdCpL1[x0][y0][2][0] != 0 | | MvdCpL1[x0][y0][2][1] != 0)) {*	
amvr_flag*[x0][y0]*	*ae(v)*
if(amvr_flag[x0][y0])	
amvr_precision_idx*[x0][y0]*	*ae(v)*
}	
if(sps_bcw_enabled_flag && inter_pred_idc[x0][y0] = = PRED_BI && luma_weight_l0_flag[ref_idx_l0 [x0][y0]] = = 0 && luma_weight_l1_flag[ref_idx_l1 [x0][y0]] = = 0 && chroma_weight_l0_flag[ref_idx_l0 [x0][y0]] = = 0 && chroma_weight_l1_flag[ref_idx_l1 [x0][y0]] = = 0 && cbWidth * cbHeight >= 256)	
bcw_idx[x0][y0]	ae(v)
}	
}	
…	

【００９５】

具体的には、ａｍｖｒ＿ｆｌａｇおよびａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのビンの文字列をコーディングするためのビンの文字列およびコンテキストは以下のように定義される。

【００９６】

【表７】

	amvr_flag	意味
IBC	-	
アフィン AMVR	0/1	0: 1/4 輝度サンプル 1: 1/16 輝度サンプル、または 1 輝度サンプル
通常インター (非 IBC, 非 affine)	0/1	0: 1/4 輝度サンプル 1: 1/2 輝度サンプル、1 輝度サンプル、または 4 輝度サンプル
ビンインデックス	0	

【００９７】
【表８】

IBC に対する amvr_precision_idx の値	amvr_precision_idx のビンの文字列		意味
0	0		1 輝度サンプル
1	1		4 輝度サンプル
2			
ビンインデックス	0 (コンテキスト A)		

【００９８】
【表９】

アフィン AMVR に対する amvr_precision_idx の値	amvr_precision_idx のビンの文字列		意味
0	0		1/16 輝度サンプル
1	1		1 輝度サンプル
2			
ビンインデックス	0 (コンテキスト A)		

【００９９】

【表１０】

通常インター（非 IBC、非アフィン）に対する amvr_precision_idx の値	amvr_precision_idx のビンの文字列		意味
0	0		1/2 輝度サンプル
1	1	0	1 輝度サンプル
2	1	1	4 輝度サンプル
ビンインデックス	0 (コンテキスト A)	1 (コンテキスト B)	

【０１００】

７．４．１１．５　コーディングユニット構文

ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘ［ｘ０］［ｙ０］は、ＡｍｖｒＳｈｉｆｔとの動きベクトル差の解像度を表２－３に定義することを規定する。配列インデックスｘ０，ｙ０は、ピクチャの左上の輝度サンプルに対する、考慮されるコーディングブロックの左上の輝度サンプルの位置（ｘ０，ｙ０）を規定する。ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘ［ｘ０］［ｙ０］が存在しない場合、０に等しいと推論される。

【０１０１】

表２－３　ＡｍｖｒＳｈｉｆｔの仕様

【０１０２】

【表１１】

amvr_flag	amvr_precision_idx	AmvrShift		
		inter_affine_flag = =1	CuPredMode[chType][x0][y0] = = MODE_IBC)	inter_affine_flag = =0 && CuPredMode[chType][x0][y0] != MODE_IBC
0	-	2 (1/4 輝度サンプル)	-	2 (1/4 輝度サンプル)
1	0	0 (1/16 輝度サンプル)	4 (1 輝度サンプル)	3 (1/2 輝度サンプル)
1	1	4 (1 輝度サンプル)	6 (4 輝度サンプル)	4 (1 輝度サンプル)
1	2	-	-	6 (4 輝度サンプル)

【０１０３】

９．３．３　２値化処理

【０１０４】

表１２６－構文要素および関連する２値化

【０１０５】

【表１２】

構文構造	構文要素	バイナリゼーション	
		処理	入力パラメータ
coding_tree( )	split_cu_flag	FL	cMax = 1
	split_qt_flag	FL	cMax = 1
	mtt_split_cu_vertical_flag	FL	cMax = 1
	mtt_split_cu_binary_flag	FL	cMax = 1
	mode_constraint_flag	FL	cMax = 1
coding_unit( )	cu_skip_flag[ ][ ]	FL	cMax = 1
	pred_mode_ibc_flag	FL	cMax = 1
	pred_mode_plt_flag	FL	cMax = 1
	cu_act_enabled_flag	FL	cMax = 1
	pred_mode_flag	FL	cMax = 1
	intra_bdpcm_luma_flag	FL	cMax = 1
	intra_bdpcm_luma_dir_flag	FL	cMax = 1
	intra_mip_flag[ ][ ]	FL	cMax = 1
	intra_mip_transposed_flag[ ][ ]	FL	cMax = 1
	intra_mip_mode[ ][ ]	TB	cMax = (cbWidth = = 4 && cbHeight = = 4) ? 15 : (((cbWith = = 4 | | cbHeight = = 4) | | (cbWith = = 8 && cbHeight = = 8))? 7 : 5)
	intra_luma_ref_idx[ ][ ]	TR	cMax = 2, cRiceParam = 0
	intra_subpartitions_mode_flag	FL	cMax = 1
	intra_subpartitions_split_flag	FL	cMax = 1
	intra_luma_mpm_flag[ ][ ]	FL	cMax = 1
	intra_luma_not_planar_flag[ ][ ]	FL	cMax = 1

【０１０６】

【表１３】

	intra_luma_mpm_idx[ ][ ]	TR	cMax = 4, cRiceParam = 0
	intra_luma_mpm_remainder[ ][ ]	TB	cMax = 60
	intra_bdpcm_chroma_flag	FL	cMax = 1
	intra_bdpcm_chroma_dir_flag	FL	cMax = 1
	cclm_mode_flag	FL	cMax = 1
	cclm_mode_idx	TR	cMax = 2, cRiceParam = 0
	intra_chroma_pred_mode	9.3.3.8	-
	general_merge_flag[ ][ ]	FL	cMax = 1
	inter_pred_idc[x0][y0]	9.3.3.9	cbWidth, cbHeight
	inter_affine_flag[ ][ ]	FL	cMax = 1
	cu_affine_type_flag[ ][ ]	FL	cMax = 1
	sym_mvd_flag[ ][ ]	FL	cMax = 1
	ref_idx_l0[ ][ ]	TR	cMax = NumRefIdxActive[0] − 1, cRiceParam = 0
	mvp_l0_flag[ ][ ]	FL	cMax = 1
	ref_idx_l1[ ][ ]	TR	cMax = NumRefIdxActive[1] − 1, cRiceParam = 0
	mvp_l1_flag[ ][ ]	FL	cMax = 1
	amvr_flag[ ][ ]	FL	cMax = 1
	amvr_precision_idx[ ][ ]	TR	cMax = (inter_affine_flag = = 0 && CuPredMode[0][x0][y0] != MODE_IBC) ? 2 : 1, cRiceParam = 0

【０１０７】

９．３．２．２　コンテキスト変数の初期化処理

【０１０８】

表５１－初期化処理における各ｉｎｉｔｉａｌｉｚａｔｉｏｎＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け

【０１０９】

【表１４】

構文構造	構文要素	ctxTable	initType		
			0	1	2
coding_unit( )	amvr_flag[ ][ ]	Table 88		0..1	2..3
	amvr_precision_idx[ ][ ]	Table 89	0..1	2..3	4..5

【０１１０】

表８８－ａｍｖｒ＿ｆｌａｇのｃｔｘＩｄｘのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様

【０１１１】

【表１５】

初期化変数	amvr_flag の ctxIdx			
	0	1	2	3
initValue	EP	EP	EP	EP
shiftIdx	0	0	0	0

【０１１２】
表８９－ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのｃｔｘＩｄｘのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様
【０１１３】
【表１６】

初期化変数	amvr_precision_idx の ctxIdx					
	0	1	2	3	4	5
initValue	EP	EP	EP	EP	EP	EP
shiftIdx	0	0	0	0	0	0

【０１１４】
９．３．４．２　ｃｔｘＴａｂｌｅ，ｃｔｘＩｄｘ，ｂｙｐａｓｓＦｌａｇの導出処理
９．３．４．２．１　一般
【０１１５】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て
【０１１６】
【表１７】

構文要素	binIdx					
	0	1	2	3	4	>= 5
amvr_flag[ ][ ]	inter_affine_flag[ ][ ] ? 1 : 0	na	na	na	na	na
amvr_precision_idx[ ][ ]	0	1	na	na	na	na

【０１１７】
２．８．　分割情報
ＶＶＣにおいて、バイナリおよびターナリ分割セグメンテーション構造を使用するネストされたマルチタイプツリーを有する４分木は、複数の分割ユニットタイプの概念に取って代わり、例えば、それは、最大変換長に対して大き過ぎるサイズを有するＣＵに必要な場合を除き、ＣＵ、ＰＵ、およびＴＵ概念の分離を排除し、かつＣＵ分割形状のためのより多くの柔軟性をサポートする。コーディングツリー構造において、ＣＵは正方形または長方形のいずれかを有することができる。まず、ＣＴＵ（Ｃｏｄｉｎｇ　Ｔｒｅｅ　Ｕｎｉ

ｔ）を４分木構造で分割する。そして、４分木のリーフのノードは、マルチタイプのツリー構造によってさらに分割され得る。図１０に示すように、マルチタイプツリー構造において４つ分割タイプ、垂直バイナリ分割（ＳＰＬＩＴ＿ＢＴ＿ＶＥＲ）、水平バイナリ分割（ＳＰＬＩＴ＿ＢＴ＿ＨＯＲ）、垂直ターナリ分割（ＳＰＬＩＴ＿ＴＴ＿ＶＥＲ）、水平ターナリ分割（ＳＰＬＩＴ＿ＴＴ＿ＨＯＲ）がある。マルチタイプツリーのリーフのノードは、ＣＵ（Ｃｏｄｉｎｇ　Ｕｎｉｔ）と呼ばれ、ＣＵが大き過ぎて最大変換長にならない限り、このセグメンテーションは、それ以上の分割なしに、予測および変換処理に使用される。これは、ほとんどの場合、ＣＵ、ＰＵ、およびＴＵが、ネストされたマルチタイプのツリーコーディングブロック構造を有する４分木において、同じブロックサイズを有することを意味する。サポートされる最大変換長がＣＵの色成分の幅または高さよりも小さい場合、この例外が生じる。

【０１１８】

図１１は、ネストされたマルチタイプツリーコーディングツリーを有する４分木における分割情報の信号通知メカニズムを示す。ＣＴＵ（Ｃｏｄｉｎｇ　Ｔｒｅｅ　Ｕｎｉｔ）は、４分木の根として取り扱われ、まず１つの４分木構造によって分割される。各４分木の葉ノード（十分に大きいため許容される場合）は、次に、マルチタイプツリー構造によってさらに分割される。マルチタイプツリー構造において、第１のフラグ（ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｆｌａｇ）は、ノードがさらに分割されているかどうかを示すために信号通知され、ノードがさらに分割されている場合、第２のフラグ（ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ）は分割方向を示すために信号通知され、次に第３のフラグ（ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｂｉｎａｒｙ＿ｆｌａｇ）が分割がバイナリ分割であるか、ターナリ分割であるかを示すために信号通知される。ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇおよびｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｂｉｎａｒｙ＿ｆｌａｇの値に基づいて、表２－４に示すように、ＣＵのマルチタイプツリースリットモード（ＭｔｔＳｐｌｉｔＭｏｄｅ）が導出される。

【０１１９】

表２－４　マルチタイプツリー構文要素に基づくＭｔｔＳｐｌｉｔＭｏｄｅの導出

【０１２０】

【表１８】

MttSplitMode	mtt_split_cu_vertical_flag	mtt_split_cu_binary_flag
SPLIT_TT_HOR	0	0
SPLIT_BT_HOR	0	1
SPLIT_TT_VER	1	0
SPLIT_BT_VER	1	1

【０１２１】

ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ＝０は、コーディングユニットを水平に分割することを規定する。ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ＝１は、コーディングユニットを垂直に分割することを規定する。ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇが存在しない場合、次のように推論される。

－ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒがＴＲＵＥに等しい、またはａｌｌｏｗＳｐｌｉｔＴｔＨｏｒがＴＲＵＥに等しい場合、ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇの値は０に等しいと推測される。

－そうでない場合、ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇの値は１に等しいと推測される。

【０１２２】

ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇの構文テーブルの例

９．３．２．２　コンテキスト変数の初期化処理

【０１２３】
表５１－初期化処理における各ｉｎｉｔｉａｌｉｚａｔｉｏｎＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け
【０１２４】
【表１９】

構文構造	構文要素	ctxTable	initType		
			0	1	2
coding_tree( )	mtt_split_cu_vertical_flag	Table 61	0..4	5..9	10..14

【０１２５】
表６１－ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのｃｔｘＩｎｃのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様
【０１２６】
【表２０】

初期化変数	mtt_split_cu_vertical_flag の ctxIdx														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
initValue	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP
shiftIdx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

【０１２７】
９．３．４．２　ｃｔｘＴａｂｌｅ，ｃｔｘＩｄｘ，ｂｙｐａｓｓＦｌａｇの導出処理
９．３．４．２．１　一般
【０１２８】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て
【０１２９】
【表２１】

構文要素	binIdx					
	0	1	2	3	4	>= 5
mtt_split_cu_vertical_flag	0..4 (節 9.3.4.2.3)	na	na	na	na	na

【０１３０】
９．３．４．２．３　構文要素ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのｃｔｘＩｎｃｆｏｒの導出プロセス
この処理への入力は、現在のピクチャの左上のサンプルに対する現在の輝度ブロックの左上の輝度サンプル、デュアルツリーチャネルタイプｃｈＴｙｐｅおよび輝度サンプルにおける現在のコーディングブロックの幅と高さｃｂＷｉｄｔｈ、ｃｂＨｅｉｇｈｔ、並びに節７．４．１１．４にでコーディングツリー意味論において導出された変数ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒおよびａｌｌｏｗＳｐｌｉｔである。
【０１３１】
この処理の出力はｃｔｘＩｎｃである。
位置（ｘＮｂＬ，ｙＮｂＬ）は（ｘ０－１，ｙ０）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定され

た位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＬ，ｙＮｂＬ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＬに割り当てられる。

位置（ｘＮｂＡ，ｙＮｂＡ）は（ｘ０，ｙ０－１）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＡ，ｙＮｂＡ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＡに割り当てられる。

【０１３２】

ｃｔｘＩｎｃの割り当ては、以下のように指定される。

－ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒがａｌｌｏｗＳｐｌｉｔＴＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＴＴＨｏｒより大きい場合、ｃｔｘＩｎｃは４に設定される。

－そうでない場合、ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒがａｌｌｏｗＳｐｌｉｔＴＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＴＴＨｏｒよりも小さい場合、ｃｔｘＩｎｃは４に等しく設定される。

－そうでない場合、以下が適用される：

－変数ｄＡおよびｄＬは、以下のように導出される。

ｄＡ＝ｃｂＷｉｄｔｈ／（ａｖａｉｌａｂｌｅＡ？ＣｂＷｉｄｔｈ［ｃｈＴｙｐｅ］［ｘＮｂＡ］［ｙＮｂＡ］：１）　（１５６３）

ｄＬ＝ｃｂＨｅｉｇｈｔ／（ａｖａｉｌａｂｌｅＬ？ＣｂＨｅｉｇｈｔ［ｃｈＴｙｐｅ］［ｘＮｂＬ］［ｙＮｂＬ］：１）　（１５６４）

－以下の条件のいずれかが真である場合、ｃｔｘＩｎｃは０に等しく設定される。

－ｄＡはｄＬに等しい、

－ａｖａｉｌａｂｌｅＡはＦＡＬＳＥである

－ａｖａｉｌａｂｌｅＬはＦＡＬＳＥである

－そうでない場合、ｄＡがｄＬよりも小さい場合、ｃｔｘＩｎｃは１に等しく設定される。

そうでない場合、ｃｔｘＩｎｃは０に等しく設定される。

【０１３３】

２．９．　変換スキップモードにおける係数コーディング

現在のＶＶＣ草案において、残差コーディングを変換スキップレベルの統計および信号特性に適応させるために、非ＴＳ係数コーディングに比べて、ＴＳ（ＴｒａｎｓｆｏｒｍＳｋｉｐ）モードにおける係数コーディングについていくつかの修正が提案されている。

【０１３４】

７．３．１０．１１　残差コーディング構文

【０１３５】

【表２２】

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	記述子
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx = = 0 && log2TbWidth = = 5 && log2TbHeight < 6)	
log2ZoTbWidth = 4	
else	
log2ZoTbWidth = Min(log2TbWidth, 5)	
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx = = 0 && log2TbWidth < 6 && log2TbHeight = = 5)	
log2ZoTbHeight = 4	
else	
log2ZoTbHeight = Min(log2TbHeight, 5)	
if(log2TbWidth > 0)	
last_sig_coeff_x_prefix	ae(v)
if(log2TbHeight > 0)	
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2TbWidth = log2ZoTbWidth	
log2TbHeight = log2ZoTbHeight	
remBinsPass1 = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3)	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 − log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 − log2SbH	
}	

【０１３６】

【表２３】

numSbCoeff = 1 << (log2SbW + log2SbH)	
lastScanPos = numSbCoeff	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − (log2SbW + log2SbH))) − 1	
do {	
if(lastScanPos = = 0) {	
lastScanPos = numSbCoeff	
lastSubBlock− −	
}	
lastScanPos− −	
xS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH] [lastSubBlock][0]	
yS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH] [lastSubBlock][1]	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
if(lastSubBlock = = 0 && log2TbWidth >= 2 && log2TbHeight >= 2 && !transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0)	
LfnstDcOnly = 0	
if((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2) || (lastScanPos > 7 && (log2TbWidth = = 2 || log2TbWidth = = 3) && log2TbWidth = = log2TbHeight))	
LfnstZeroOutSigCoeffFlag = 0	
if((lastSubBlock > 0 || lastScanPos > 0) && cIdx = = 0)	
MtsDcOnly = 0	
QState = 0	
for(i = lastSubBlock; i >= 0; i− −) {	
startQStateSb = QState	
xS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH] [i][0]	
yS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH] [i][1]	
inferSbDcSigCoeffFlag = 0	

【０１３７】

【表２４】

if(i < lastSubBlock && i > 0) {	
sb_coded_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
if(sb_coded_flag[xS][yS] && (xS > 3 | | yS > 3) && cIdx = = 0)	
MtsZeroOutSigCoeffFlag = 0	
firstSigScanPosSb = numSbCoeff	
lastSigScanPosSb = −1	
firstPosMode0 = (i = = lastSubBlock ? lastScanPos : numSbCoeff − 1)	
firstPosMode1 = firstPosMode0	
for(n = firstPosMode0; n >= 0 && remBinsPass1 >= 4; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(sb_coded_flag[xS][yS] && (n > 0 | | !inferSbDcSigCoeffFlag) && (xC != LastSignificantCoeffX | | yC != Last SignificantCoeffY)) {	
sig_coeff_flag[xC][yC]	ae(v)
remBinsPass1− −	
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
if(sig_coeff_flag[xC][yC]) {	
abs_level_gtx_flag[n][0]	ae(v)
remBinsPass1− −	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
remBinsPass1− −	
abs_level_gtx_flag[n][1]	ae(v)
remBinsPass1− −	
}	
if(lastSigScanPosSb = = −1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	

【０１３８】

【表２５】

AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0] + 2 * abs_level_gtx_flag[n][1]	
if(ph_dep_quant_enabled_flag)	
QState = QStateTransTable[QState][AbsLevelPass1[xC][yC] & 1]	
firstPosMode1 = n − 1	
}	
for(n = firstPosMode0; n > firstPosMode1; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(abs_level_gtx_flag[n][1])	
abs_remainder[n]	ae(v)
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] +2 * abs_remainder[n]	
}	
for(n = firstPosMode1; n >= 0; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(sb_coded_flag[xS][yS])	
dec_abs_level[n]	ae(v)
if(AbsLevel[xC][yC] > 0) {	
if(lastSigScanPosSb = = −1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	
if(ph_dep_quant_enabled_flag)	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
}	
if(ph_dep_quant_enabled_flag | | !pic_sign_data_hiding_enabled_flag)	
signHidden = 0	
else	
signHidden = (lastSigScanPosSb − firstSigScanPosSb > 3 ? 1 : 0)	

【０１３９】

【表２６】

for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if((AbsLevel[xC][yC] > 0) && (!signHidden || (n != firstSigScanPosSb)))	
coeff_sign_flag[n]	ae(v)
}	
if(ph_dep_quant_enabled_flag) {	
QState = startQStateSb	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(AbsLevel[xC][yC] > 0)	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (2 * AbsLevel[xC][yC] − (QState > 1 ? 1 : 0)) * (1 − 2 * coeff_sign_flag[n])	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
} else {	
sumAbsLevel = 0	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(AbsLevel[xC][yC] > 0) {	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = AbsLevel[xC][yC] * (1 − 2 * coeff_sign_flag[n])	
if(signHidden) {	
sumAbsLevel += AbsLevel[xC][yC]	
if((n = = firstSigScanPosSb) && (sumAbsLevel % 2) = = 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
}	
}	
}	
}	
}	

【０１４０】

【表２７】

residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	記述子
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3)	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 − log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 − log2SbH	
}	
numSbCoeff = 1 << (log2SbW + log2SbH)	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − (log2SbW + log2SbH))) − 1	
inferSbCbf = 1	
RemCcbs = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
for(i =0; i <= lastSubBlock; i++) {	
xS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][0]	
yS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][1]	
if(i != lastSubBlock || !inferSbCbf)	
sb_coded_flag[xS][yS]	ae(v)
if(sb_coded_flag[xS][yS] && i < lastSubBlock)	
inferSbCbf = 0	
/* First scan pass */	
inferSbSigCoeffFlag = 1	
lastScanPosPass1 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(sb_coded_flag[xS][yS] && (n != numSbCoeff − 1 || !inferSbSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
RemCcbs− −	
if(sig_coeff_flag[xC][yC])	

【０１４１】

【表２８】

inferSbSigCoeffFlag = 0	
}	
CoeffSignLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC] {	
coeff_sign_flag[n]	ae(v)
RemCcbs− −	
CoeffSignLevel[xC][yC] = (coeff_sign_flag[n] > 0 ? −1 : 1)	
abs_level_gtx_flag[n][0]	ae(v)
RemCcbs− −	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
RemCcbs− −	
}	
}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
lastScanPosPass1 = n	
}	
/* Greater than X scan pass (numGtXFlags=5) */	
lastScanPosPass2 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
AbsLevelPass2[xC][yC] = AbsLevelPass1[xC][yC]	
for(j = 1; j < 5; j++) {	
if(abs_level_gtx_flag[n][j − 1]) {	
abs_level_gtx_flag[n][j]	ae(v)
RemCcbs− −	
}	
AbsLevelPass2[xC][yC] + = 2 * abs_level_gtx_flag[n][j]	
}	
lastScanPosPass2 = n	
}	

【０１４２】

【表２９】

/* remainder scan pass */	
for(n = 0; n <= numSbCoeff − 1; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if((n <= lastScanPosPass2 && AbsLevelPass2[xC][yC] >= 10) || (n > lastScanPosPass2 && n <= lastScanPosPass1 && AbsLevelPass1[xC][yC] >= 2) || (n > lastScanPosPass1 && sb_coded_flag[xS][yS]))	
abs_remainder[n]	ae(v)
if(n <= lastScanPosPass2)	
AbsLevel[xC][yC] = AbsLevelPass2[xC][yC] + 2 * abs_remainder[n]	
else if(n <= lastScanPosPass1)	
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
else { /* bypass */	
AbsLevel[xC][yC] = abs_remainder[n]	
if(abs_remainder[n])	
coeff_sign_flag[n]	ae(v)
}	
if(BdpcmFlag[x0][y0][cIdx] = = 0 && n <= lastScanPosPass1) {	
absLeftCoeff = xC > 0 ? AbsLevel[xC − 1][yC]) : 0	
absAboveCoeff = yC > 0 ? AbsLevel[xC][yC − 1]) : 0	
predCoeff = Max(absLeftCoeff, absAboveCoeff)	
if(AbsLevel[xC][yC] = = 1 && predCoeff > 0)	
AbsLevel[xC][yC] = predCoeff	
else if(AbsLevel[xC][yC] > 0 && AbsLevel[xC][yC] <= predCoeff)	
AbsLevel[xC][yC]− −	
}	
}	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (1 − 2 * coeff_sign_flag[n]) * AbsLevel[xC][yC]	
}	
}	

【０１４３】

２．９．１．　符号フラグｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇのコンテキストモデリングおよびコンテキストインデックスオフセット導出

【０１４４】

表５１－初期化処理における各ｉｎｉｔｉａｌｉｚａｔｉｏｎＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け

【０１４５】

【表３０】

構文構造	構文要素	ctxTable	initType		
			0	1	2
residual_coding( )	last_sig_coeff_x_prefix	Table 119	0..22	23..45	46..68
	last_sig_coeff_y_prefix	Table 120	0..22	23..45	46..68
	sb_coded_flag[ ][ ]	Table 121	0..7	8..15	16..23
	sig_coeff_flag[ ][ ]	Table 122	0..62	63..125	126..188
	par_level_flag[ ]	Table 123	0..32	33..65	66..98
	abs_level_gtx_flag[ ][ ]	Table 124	0..73	74..147	148..220
	coeff_sign_flag[ ]	Table 125	0..5	6..11	12..17

【０１４６】
表１２５－ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇのｃｔｘＩｎｃのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様
【０１４７】
【表３１】

初期化変数	coeff_sign_flag の ctxIdx																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
initValue	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP
shiftIdx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

【０１４８】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て
【０１４９】
【表３２】

構文要素	binIdx					
	0	1	2	3	4	>= 5
coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] == 0 || RemCcbs == 0 || slice_ts_residual_coding_disabled_flag	バイパス	na	na	na	na	na
coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] == 1 && RemCcbs >= 0 && !slice_ts_residual_coding_disabled_flag	0..5 (節 9.3.4.2.10)	na	na	na	na	na

【０１５０】
９．３．４．２．１０　変換スキップモードのための構文要素ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇのｃｔｘＩｎｃの導出プロセス

この処理への入力は、色成分インデックスｃＩｄｘ、現在のピクチャの左上のサンプルに対して現在の変換ブロックの左上のサンプルを規定する輝度位置（ｘ０，ｙ０）、現在の係数スキャン位置（ｘＣ，ｙＣ）である。
このプロセスの出力は変数ｃｔｘＩｎｃである。
変数ｌｅｆｔＳｉｇｎおよびａｂｏｖｅＳｉｇｎは、以下のように導出される。
ｌｅｆｔＳｉｇｎ＝（ｘＣ＝＝０）？０：ＣｏｅｆｆＳｉｇｎＬｅｖｅｌ［ｘＣ－１］［ｙＣ］　（１５９５）
ａｂｏｖｅＳｉｇｎ＝（ｙＣ＝＝０）？０：ＣｏｅｆｆＳｉｇｎＬｅｖｅｌ［ｘＣ］［ｙＣ－１］　（１５９６）
変数ｃｔｘＩｎｃは、以下のように導出される。
－ｌｅｆｔＳｉｇｎが０に等しく、ａｂｏｖｅＳｉｇｎが０に等しい場合、またはｌｅｆｔＳｉｇｎが－ａｂｏｖｅＳｉｇｎに等しい場合、以下が適用される。
ｃｔｘＩｎｃ＝（ＢｄｐｃｍＦｌａｇ［ｘ０］［ｙ０］［ｃＩｄｘ］＝＝０？０：３）　（１５９７）
－そうでない場合、ｌｅｆｔＳｉｇｎが０以上かつａｂｏｖｅＳｉｇｎが０以上である場合、以下が適用される。
ｃｔｘＩｎｃ＝（ＢｄｐｃｍＦｌａｇ［ｘ０］［ｙ０］［ｃＩｄｘ］？１：４）　（１５９８）
－そうでない場合、以下が適用される：
ｃｔｘＩｎｃ＝（ＢｄｐｃｍＦｌａｇ［ｘ０］［ｙ０］［ｃＩｄｘ］？２：５）　（１５９９）

【０１５１】

３．　開示される技術的解決策および実施形態によって解決される技術的課題

ＡＭＶＲ精密インデックスおよび分割ＣＵ垂直フラグのためのコンテキスト導出プロセスの現在の設計は、以下の問題を有する。
１．ブロックを水平または垂直に分割することを規定する構文要素（例えば、ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ）のコンテキストモデリングは、「ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ」と「ａｌｌｏｗＳｐｌｉｔＴＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＴＨｏｒ」との間の関係に依存する。しかし、ＢＴ／ＴＴ水平を許可するよりも、分割情報とＢＴ／ＴＴ垂直を許可する方が相関が大きいことに留意されたい。
２．現在のＶＶＣにおいて、ＡＭＶＲ精度インデックス（例えば、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘ）の第１のビンは、ブロックがＩＢＣモードでコーディングされるか、アフィンモードでコーディングされるか、または通常のインターモード（非ＩＢＣ、非アフィン）でコーディングされるかを考慮せずに、１つのコンテキストでコンテキストコーディングされる。ＡＭＶＲ精度インデックスをコーディングすることは、あまり効率的でない場合がある。また、通常のインターモードを有するブロックに対してコーディングされるａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第２のビンは、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンに使用されるコンテキストとは別個のコンテキストを使用する。構文要素のコーディングに使用される複数のコンテキストは、構文要素のコーディング頻度が低い場合、最適でない場合がある。
３．係数コーディングは、画面コンテンツのコーディングにおいてコーディングの利点を実現することができるが、係数コーディングおよびＴＳモードは、依然としていくつかの欠点を有する可能性がある。
ａ．符号フラグにバイパスコーディングを使用するか、コンテキストコーディングを使用するかは、このケースでは不明である。
ｉ．残りの許可されたコンテキストコーディングされたビンの数（ＲｅｍＣｃｂｓで表される）は、０に等しい。
ｉｉ．現在のブロックはＴＳモードでコーディングされる。
ｉｉｉ．ｓｌｉｃｅ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｄｉｓａｂｌｅｄ

＿ｆｌａｇは偽である。
【０１５２】
４．　技術的解決策および実施形態の一覧
以下の項目は、一般的な概念を説明するための例であると考えられるべきである。これら項目は狭い意味で解釈されるべきではない。さらに、これらの項目は、任意の方法で組み合わせることができる。
【０１５３】
本開示では、用語ＡＭＶＲは、ＭＶ（Ｍｏｔｉｏｎ　Ｖｅｃｔｏｒ）／ＭＶＤ（ＭＶ　Ｄｉｆｆｅｒｅｎｃｅ）コーディングまたはＭＶＰ（ＭＶ　Ｐｒｅｄｉｃｔｏｒ）のために適応動きベクトル差分解像度を使用するコーディングツールを表す。本発明は、ＶＶＣに記載されているＡＭＶＲおよびブロック分割技術に限定されるものではない。
【０１５４】
ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘは、許容される動きベクトルの差分解像度のインデックス（または指標）を規定する構文要素を表す。一例において、それはＶＶＣテキストにおいて定義されるａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘであってもよい。なお、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘは、１または複数のビンを含むことができるビンの文字列に２値化されてもよい。
【０１５５】
ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇは、コーディングブロックを垂直方向に分割するか否かを規定する構文要素を表す。一例において、それはＶＶＣテキストに定義されたｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇであることができる。
【０１５６】
ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのコンテキストの派生
１．ＡＭＶＲの使用を示すＳＥ（Ｓｙｎｔａｘ　Ｅｌｅｍｅｎｔ）の第１のビン（ビンインデックスが０に等しい）および／またはビンの文字列の他のビン（例えば、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘおよび／またはａｍｖｒ＿ｆｌａｇ）のためのコンテキストモデリング（例えば、コンテキストをどのように選択するか）は、現在のブロックおよび／または近傍のブロックのコーディングされた情報（例えば、コーディングされたモード）に依存してもよい。
ａ．一例において、コーディングされた情報は、ＩＢＣ、アフィンＡＭＶＲ、および通常のインター（例えば、非ＩＢＣ、非アフィン）モード、双予測および／または単予測、現在のブロックのブロック寸法および／または近傍のブロックのブロック寸法のうちの少なくとも１つを備えてもよく、コーディングされた情報に基づいてビン（例えば、第１のビン）をコーディングするために異なるコンテキストを利用してもよい。
ｉ．あるいは、ＩＢＣコーディングされたブロックの１つのビン（例えば、第１のビン）は、ＣｔｘＭで示される単一のコンテキストでコーディングされる。
ｉｉ．あるいは、アフィンコーディングされたブロックのためのＳＥの１つのビン（例えば、第１のビン）は、ＣｔｘＮによって表される単一のコンテキストでコーディングされる。
ｉｉｉ．あるいは、さらに、通常のインター（例えば、非アフィンおよび非ＩＢＣ）コーディングされたブロックのためのＳＥの１つのビン（例えば、第１のビン）は、ＣｔｘＰによって表される単一のコンテキストでコーディングされる。
ｉｖ．あるいは、３つのコンテキストＣｔｘＭ、ＣｔｘＮ、ＣｔｘＰのうち少なくとも１つのコンテキストが他の２つのコンテキストと異なる。
ｖ．あるいは、３つのコンテキストＣｔｘＭ、ＣｔｘＮ、ＣｔｘＰは、それぞれ異なる。
ｖｉ．例えば、ＳＥの１つのビン（例えば、第１のビン）は、双予測ブロックの場合、ＣｔｘＢｉで示される単一のコンテキストでコーディングされ、単予測コーディングブロックの場合、ＣｔｘＵｎｉで示される単一のコンテキストでコーディングされる。

１）一例において、ＣｔｘＢｉは、ＣｔｘＵｎｉとは異なる。

【０１５７】

ｂ．一例において、２つ以上のコンテキストが、ＳＥのビンストリングの第１のビンおよび／または他のビンをコーディングするために利用されてもよい。

ｉ．一例において、第１のビンのためにＸ個のコンテキストを利用することができ、ここで、Ｘ＞１である。

１）一例において、Ｘ＝３である。

ａ）あるいは、さらに、コンテキストの選択は、コーディングされた情報（例えば、上述したモード）に依存する。

２）一例において、Ｘ＝２である。

ａ）あるいは、さらに、コンテキストの選択は、コーディングされた情報（例えば、上述したモード）およびＩＢＣコーディングブロックのための１つのコンテキストに依存し、他のブロック（例えば、アフィンまたは通常のインターコーディング）のための他のコンテキストに依存する。

ｂ）あるいは、コンテキストの選択は、コーディングされた情報（例えば、上述したモード）、ＩＢＣおよびアフィンＡＭＶＲコーディングされたブロックの１つのコンテキスト、および、他のブロック（例えば、通常のインターコーディング）の１つのコンテキストに依存する。

ｃ．一例において、コーディングされた情報（例えば、コーディングモード）に基づいて、ＳＥのビンの文字列の第１のビン（ビンインデックスが０に等しい）および／または他のビンの異なるモデルを、異なる初期化値で初期化してもよい。

ｄ．一例において、コーディングされた情報（例えば、コーディングモード）に基づいて、ＳＥのビンの文字列の第１のビン（ビンインデックスが０に等しい）および／または他のビンの異なるモデルを、同じ初期化値で初期化してもよい。

【０１５８】

２．ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのビンの文字列の第１および第２のビンのために様々なコンテキストを使用する代わりに、第２のビンをコーディングするために使用されるコンテキストは、ビンの文字列の第１のビンをコーディングするために使用されるコンテキストの１または複数と同じであってもよいことが提案される。

ａ．あるいは、ビンの文字列の第２のビンは、通常のインター（例えば、非アフィンおよび非ＩＢＣ）コーディングブロックに対してのみコーディングされる。

ｂ．あるいは、ビンの文字列の第２のビンは、ＣｔｘＱで表される単一のコンテキストでコーディングされる。

ｃ．あるいは、同じコンテキストが、ＩＢＣコーディングブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために用いられ、通常のインターコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第２のビンをコーディングするために使用されてもよい。

ｄ．あるいは、同じコンテキストが、アフィンコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために使用され、通常のインターコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第２のビンをコーディングするために使用されてもよい。

ｅ．あるいは、同じコンテキストが、通常のインターコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために使用され、通常のインターコーディングブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第２のビンをコーディングするために使用されてもよい。

【０１５９】

３．ＩＢＣコーディングされたブロック（ＣｔｘＭで示す）に対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンと通常のインターコーディングされたブロック（ＣｔｘＱで示す）のためのａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第２のビンをコーディングするために、ＣｔｘＭ＝ＣｔｘＱのように、同じコンテキストが使用されてもよい。

ａ．一例において、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘをコーディングするために、Ｘ１（例えば、Ｘ１＝３）コンテキストを利用してもよい。

ｂ．あるいは、非ＩＢＣコーディングされたブロックに対し、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのビンの文字列の第１のビンをコーディングするための様々なコンテキストを利用してもよい。

【０１６０】

４．ＩＢＣコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、アフィンコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、および通常のインターコーディングに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために、ＣｔｘＭ＝ＣｔｘＮ＝ＣｔｘＰのように、同じコンテキストが使用されてもよい。

ａ．一例において、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘをコーディングするために、Ｘ２（例えば、Ｘ２＝２）コンテキストを利用してもよい。

ｂ．あるいは、さらに、非ＩＢＣおよび非アフィンコーディングされたブロックに対し、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのビンの文字列の第１のビンをコーディングするための異なるコンテキストが利用されてもよい。

【０１６１】

５．ＩＢＣコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンおよび通常のインターコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために、ＣｔｘＭ＝ＣｔｘＰのように、同じコンテキストが使用されてもよい。

ａ．一例において、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘをコーディングするために、Ｘ３（例えば、Ｘ３＝３）コンテキストを利用してもよい。

ｂ．あるいは、非ＩＢＣおよび非通常のインターコーディングされたブロック（例えば、アフィンＡＭＶＲでコーディングされた）に対し、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのビンの文字列の第１のビンをコーディングするための異なるコンテキストが利用されてもよい。

ｃ．あるいは、さらに、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのビンの文字列の第２のビンをコーディングするための異なるコンテキストが利用されてもよい。

【０１６２】

６．ＩＢＣコーディングされたブロックのためのａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、およびアフィンコーディングされたブロックのためのａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために、ＣｔｘＭ＝ＣｔｘＮのように、同じコンテキストが使用されてもよい。

７．ＩＢＣコーディングされたブロック、アフィンコーディングされたブロック、および通常のコーディングされたブロックに対し、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのすべてのビンをコーディングするために、同じコンテキストを使用してもよい。

ａ．一例において、単一のコンテキストは、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘをコーディングするために利用してもよい。

【０１６３】

８．複数のコンテキストが、ＩＢＣコーディングされたブロック、アフィンコーディングされたブロック、および通常のインターコーディングされたブロックに対し、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンをコーディングするために用いられてもよく、単一のコンテキストが、第１のビンをコーディングするために使用されるものとは異なる第２のビンをコーディングするために用いられてもよい。

ａ．一例において、ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘをコーディングするために、Ｘ４（例えば、Ｘ４＝４）コンテキストを利用してもよい。

ｂ．例えば、ＣｔｘＭ！＝ＣｔｘＱ！＝ＣｔｘＮ！＝ＣｔｘＰである。

【０１６４】

９．ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘをコーディングするために使用される少な

くとも１つのコンテキストは、ａｍｖｒ＿ｆｌａｇをコーディングするために使用されるコンテキストと同じであってもよいことが提案される。

ａ．アフィンコーディングブロックのＡＭＶＲフラグ（例えば、ａｍｖｒ＿ｆｌａｇ）をコーディングするためのコンテキストと同じコンテキストが、ＩＢＣコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、または／およびアフィンコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、または／および通常のインターコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンまたは／および第２のビンのために用いられてもよい。

ｂ．非アフィンコーディングされたブロックのＡＭＶＲフラグ（例えば、ａｍｖｒ＿ｆｌａｇ）をコーディングするためのコンテキストと同じコンテキストが、ＩＢＣコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、または／およびアフィンコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビン、または／および通常のインターコーディングされたブロックに対するａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンまたは／および第２のビンのために用いられてもよい。

ｃ．ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘの第１のビンのコンテキストモデリングは、ブロックに対してアフィンモードが適用されるかどうかに依存する。

ｉ．あるいは、さらに、１つのコンテキストが、アフィンコーディングされたブロックの第１のビンをコーディングするために使用され、他のコンテキストが、非アフィンコーディングされたブロック（例えば、通常のインターコーディングされたブロックおよびＩＢＣコーディングされたブロックを含む）のために使用される。

ｉｉ．あるいは、さらに、第１のコンテキストが、アフィンコーディングされたブロックの第１のビンをコーディングするために使用され、第２のコンテキストが、非アフィンコーディングされたブロック（例えば、通常のインターコーディングされたブロックおよびＩＢＣコーディングされたブロックを含む）のために使用される。第１のコンテキストは、アフィンコーディングされたブロックのａｍｖｒ＿ｆｌａｇのコーディングのために使用されるコンテキストと同じであり、第２のコンテキストは、非アフィンコーディングされたブロックのａｍｖｒ＿ｆｌａｇのコーディングに使用されるコンテキストと同じである。

【０１６５】

ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのコンテキストの導出

変数ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒが、現在のコーディングツリーノードに対して垂直ＢＴ分割、水平ＢＴ分割、垂直ＴＴ分割、水平ＴＴ分割が許可されたかどうかをそれぞれ示すとする。ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＴＨｏｒの値は０または１に等しくてもよく、これらは章２．６で導出される。現在のブロックの幅、現在のブロックの高さ、左の近傍のブロックの幅、左の近傍のブロックの高さ、上の近傍のブロックの幅、および上の近傍のブロックの高さを、それぞれ、ｃｕｒＷ、ｃｕｒＨ、ｌｅｆｔＷ、ｌｅｆｔＨ、ａｂｏｖｅＷ、およびａｂｏｖｅＨで表す。「ｎｕｍＶ」をａｌｌｏｗＳｐｌｉｔＢｔＶｅｒとａｌｌｏｗＳｐｌｉｔＴｔＶｅｒの和に等しい値とし、「ｎｕｍＨ」をａｌｌｏｗＳｐｌｉｔＢｔＨｏｒとａｌｌｏｗＳｐｌｉｔＴＨｏｒの和に等しい値とする。

【０１６６】

１０．ブロック分割情報を示すＳＥ（例えば、ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ）のコンテキストモデリングのためのコンテキストモデリングは、垂直分割を許可する数（例えば、ＢＴおよびＴＴ）および水平分割を許可する数（例えば、ＢＴおよびＴＴ）に依存してもよい。

ａ．一例において、水平分割と比較して垂直分割が許可された場合が多い場合（例え

ば、ｎｕｍＶ＞ｎｕｍＨ）、第１のコンテキストのセットが利用される。
ｂ．一例において、水平分割に比べて垂直分割が許可された場合が少ない（例えば、ｎｕｍＶ＜ｎｕｍＨ）場合、第２のコンテキストのセットが利用される。
ｃ．一例において、水平分割と比較して垂直分割が許可された場合が同じである（例えば、ｎｕｍＶ＝ｎｕｍＨ）場合、第３のコンテキストのセットが利用される。
ｄ．あるいは、さらに、第１／第２／第３のセットにおけるコンテキストはいずれも同じではない。
ｅ．あるいは、さらに、第１／第２／第３のセットにおけるコンテキストのうち少なくとも１つは、別のセットに含まれるコンテキストと同じである。
ｆ．あるいは、さらに、３つのセットそれぞれのコンテキストの数は、セットインデックスに依存してもよい。
ｉ．一例において、１つのコンテキストのみが第１および／または第２のセットに含まれる。
ｉｉ．一例において、複数のコンテキストが第３のセットに含まれる。
１）あるいは、第３のセットからのコンテキストの選択は、上および左の近傍のブロックの利用可能性、および／または現在のブロックのブロック寸法および近傍のブロックのブロック寸法にさらに依存してもよい。
ｇ．１つの例が、章５．４の実施形態＃４に示されている。
ｈ．１つの例が、章５．５の実施形態＃５に示されている。

【０１６７】

１１．１つのブロックを垂直に分割するかどうかを示すＳＥ（例えば、ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ）は、ＢＴ／ＴＴ分割が許可されたかどうか、または／および現在のブロックの幅／高さ、または／および近傍のブロックの幅／高さに依存してよい、Ｎ個のコンテキストでコーディングされる。
ｉ．一例において、ＳＥをコーディングするためにどのコンテキストが使用されるかは、ｎｕｍＶおよびｎｕｍＨに依存してよい。
ｉ．例えば、ｎｕｍＶがｎｕｍＨよりも大きいかどうかに依存する。
ｉｉ．例えば、ｎｕｍＶがｎｕｍＨよりも小さいかどうかに依存する。
ｉｉｉ． 例えば、ｎｕｍＶがｎｕｍＨに等しいかどうかは、平均に依存する。

【０１６８】

ｊ．一例において、ＳＥのビンストリングは、ＢＴ／ＴＴ分割が許可されたかどうかに基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。
ｉ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨよりも大きい場合、ＣｔｘＡによって表されるコンテキストでコーディングされる。
ｉｉ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨ未満である場合、ＣｔｘＢによって表されるコンテキストでコーディングされる。
ｉｉｉ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨに等しい場合、ＣｔｘＣによって表されるコンテキストでコーディングされる。
ｉｖ．一例において、ＣｔｘＡはＣｔｘＢに等しく、ＣｔｘＢはＣｔｘＣに等しく（例えば、ＣｔｘＡ＝ＣｔｘＢ＝ＣｔｘＣ）、例えば、ＣｔｘＡ＝ＣｔｘＢ＝ＣｔｘＣ＝０である。
ｖ．一例において、ＣｔｘＡ！＝ＣｔｘＢ！＝ＣｔｘＣであり、例えば、ＣｔｘＡ＝０，ＣｔｘＢ＝１，ＣｔｘＣ＝２である。

【０１６９】

ｋ．一例において、ＳＥのビンの文字列は、現在のブロックの幅／高さ、および／または近傍のブロックの幅／高さに基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。
ｉ．一例において、近傍のブロックは、上の近傍のブロック、または／および左の近傍のブロックを参照してよい。
ｉｉ．一例において、ＳＥは、現在のブロックの幅または高さ、および／または近

傍のブロックの幅または高さの関数に依存してよい、Ｎ個のコンテキストでコーディングされる。ｄＡ＝ｃｕｒＷ／ａｂｏｖｅＷおよびｄＬ＝ｃｕｒＨ／ｌｅｆｔＨを表す。

１）一例において、ＳＥは、左の近傍のブロックまたは上の近傍のブロックのいずれかが利用可能でない場合、またはｄＡがｄＬに等しい場合、ＣｔｘＤによって表されるコンテキストでコーディングされる。

２）一例において、ＳＥは、ｄＡがｄＬ未満である場合、ＣｔｘＥによって表されるコンテキストでコーディングされる。

３）一例において、ＳＥは、ｄＡがｄＬよりも大きい場合、ＣｔｘＦによって表されるコンテキストでコーディングされる。

【０１７０】

ｌ．一例において、ＳＥのビンの文字列は、ＢＴ／ＴＴ分割が許可されたかどうか、および／または現在のブロックの幅／高さ、および／または近傍のブロックの幅／高さに基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。

ｉ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨよりも大きい場合、ＣｔｘＡによって表されるコンテキストでコーディングされる。

ｉｉ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨ未満である場合、ＣｔｘＢによって表されるコンテキストでコーディングされる。

ｉｉｉ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨに等しい（左の近傍のブロックまたは上の近傍のブロックのいずれかが利用可能でない、またはｄＡがｄＬに等しい）場合、ＣｔｘＣによって示されるコンテキストでコーディングされる。

ｉｖ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨに等しく、ｄＡがｄＬ未満である場合、ＣｔｘＥによって表されるコンテキストでコーディングされる。

ｖ．一例において、ＳＥは、ｎｕｍＶがｎｕｍＨに等しく、ｄＡがｄＬよりも大きい場合、ＣｔｘＦによって表されるコンテキストでコーディングされる。

一例において、Ｎ＝５，ＣｔｘＡ！＝ＣｔｘＢ！＝ＣｔｘＣ！＝ＣｔｘＥ！＝ＣｔｘＦである。

【０１７１】

ｍ．一例において、ＳＥのビンの文字列は、現在のブロックがピクチャ境界にあるかに基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。

ｎ．一例において、ＳＥのビンの文字列は、デュアルツリーおよび／またはローカルデュアルツリーのいずれが適用されるかに基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。

ｏ．一例において、ＳＥのビンの文字列は、分割されるサンプルの色成分に基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。

ｐ．一例において、ＳＥのビンの文字列は、現在のブロックの幅／高さに基づいて、Ｎ個のコンテキストでコンテキストコーディングされてもよい。

ｉ．一例において、コンテキスト増加手段は、ブロックの幅または高さの関数に設定されてもよい。

【０１７２】

１２．分割ＣＵ垂直フラグ（例えば、ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇ）は、単一のコンテキストでコーディングされてもよい。

【０１７３】

<u>係数符号フラグのためにバイパスコーディングまたはコンテキストコーディングを使用する方法</u>

【０１７４】

１３．変換係数レベルの符号（例えば、構文要素ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ）のためにコンテキストコーディングを使用するか、バイパスコーディングを使用するかは、残りの許可されたコンテキストコーディングされたビンの数（例えば、ＲｅｍＣｃｂｓ）および／または現在のブロックに使用される変換の種類（例えば、ＤＣＴ２、ＤＳＴ７、または変換スキップ）に依存する。

ａ．一例において、変換スキップ残差コーディングの処理において、ＲｅｍＣｃｂｓがＴ１より大きい（例えば、Ｔ１＝０）場合、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇにコンテキストコーディングを使用してもよい。

ｉ．また、変換スキップ残差コーディングの手順において、ＲｅｍＣｃｂｓがＴ１に等しい（例えば、Ｔ１＝０）場合、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇにバイパスコーディングを使用してもよい。

ｂ．一例において、変換スキップ残差コーディングの処理において、ＲｅｍＣｃｂｓがＴ２以上（例えば、Ｔ２＝３）である場合、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇにコンテキストコーディングを用いてもよい。

ｉ．また、変換スキップ残差コーディングの処理において、ＲｅｍＣｃｂｓがＴ２より小さい場合（例えば、Ｔ２＝３）、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇに対してバイパスコーディングを使用してもよい。

【０１７５】

１４．変換スキップ残差コーディング処理の第３／剰余係数走査パスにおける残りの構文要素（例えば、構文要素ａｂｓ＿ｒｅｍａｉｎｄｅｒおよびｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ）に対するバイパスコーディングの開始時に、残りの許可されたコンテキストコーディングされたビンの数（例えば、ＲｅｍＣｃｂｓ）を規定する変数に１つの演算を適用してもよい。

ｃ．一例において、この動作は、ＲｅｍＣｃｂｓをある値（例えば、０）に等しくなるように設定してよい。

ｄ．いくつかの実施形態において、動作は、ＲｅｍＣｃｂｓを除く少なくとも１つの変数または構文要素に基づいて、ＲｅｍＣｃｂｓを値に等しく設定することであってもよい。

ｉ．一例において、この動作は、ＲｅｍＣｃｂｓをＲｅｍＣｃｂｓから１を減算したものに等しくなるように設定することができる。

【０１７６】

１５．１つの例が、章５．７の実施形態＃７に示されている。

１６．　１つの例が、章５．８の実施形態＃８に示されている。

【０１７７】

１７．変換係数レベル（例えば、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ）がバイパスモードでコーディングされるか、またはコンテキストコーディングモードでコーディングされるかは、残りの許可されたコンテキストコーディングされたビンの数（例えば、ＲｅｍＣｃｂｓ）に依存してもよい。

ｅ．残りの許可されたコンテキストコーディングされたビンの数（例えば、ＲｅｍＣｃｂｓ）がＮよりも小さい場合、変換係数レベルの符号（例えば、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ）をバイパスモードでコーディングすることが提案される。

ｆ．一例において、符号フラグは、ＲｅｍＣｃｂｓ＜＝Ｎである場合、バイパスモードでコーディングされる。

ｉ．あるいは、一例において、ＲｅｍＣｃｂｓ＞Ｎである場合、符号フラグはコンテキストモードでコーディングされる。

ｇ．一例において、ＲｅｍＣｃｂｓがＮに等しい場合、符号フラグはバイパスモードでコーディングされる。

ｉ．　あるいは、一例において、ＲｅｍＣｃｂｓ＞Ｎである場合、符号フラグはバイパスモードでコーディングされる。

ｉｉ．一例において、Ｎは４に等しく設定されてもよい。

１）あるいは、一例において、Ｎは０に等しく設定されてもよい。

ｉｉｉ．一例において、ＲｅｍＣｃｂｓは、変換係数レベルの残りの絶対値を復号する前に、Ｘに修正されてもよく、ここで、ＸはＮに等しい。

【０１７８】

ｈ．一例において、ＲｅｍＣｃｂｓがＮ未満の場合、符号フラグはバイパスモードで

コーディングされる。
ｉ．あるいは、一例において、ＲｅｍＣｃｂｓ＞＝Ｎである場合、符号フラグはコンテキストモードでコーディングされる。
ｉｉ．一例において、Ｎは３に等しく設定されてもよい。
ｉｉｉ．一例において、ＲｅｍＣｃｂｓは、変換係数レベルの残りの絶対値を復号する前に、Ｘに修正されてもよく、ここで、ＸはＮよりも小さい。
ｉ．一例において、Ｎは整数であり、以下に基づいてもよい。
ｉ．ＳＰＳ／ＶＰＳ／ＰＰＳ／ピクチャヘッダ／スライスヘッダ／タイルグループヘッダ／ＬＣＵ行／ＬＣＵのグループ／ＬＣＵ／ＣＵにおいて信号通知された指示
ｉｉ．現在のブロックおよび／またはその近傍のブロックのブロック寸法
ｉｉｉ．現在のブロックおよび／またはその近傍のブロックのブロック形状
ｉｖ．カラーフォーマットの表示（例えば、４：２：０、４：４：４）
ｖ．別個またはデュアルのコーディングツリー構造が使用されているかどうか
ｖｉ．スライスのタイプおよび／またはピクチャのタイプ
ｖｉｉ．色成分の数

【０１７９】
ｊ．変換係数レベルをコーディングするために使用されるコーディングコンテキスト（例えば、ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ）は、残りの許可されたコンテキストコーディングされたビンの数（例えば、ＲｅｍＣｃｂｓ）の数に依存してよい。
ｋ．上記の例は、ＢＤＰＣＭコーディングされたブロックを含む、または含まない変換ブロックおよび／または変換スキップブロックに適用されてもよい。

【０１８０】
一般

【０１８１】
１８．上記開示された方法を適用するかどうかおよび／またはどのように適用するかは、例えば、シーケンスヘッダ／ピクチャヘッダ／ＳＰＳ／ＶＰＳ／ＤＰＳ／ＤＣＩ／ＰＰＳ／ＡＰＳ／スライスヘッダ／タイルグループヘッダにおいて、シーケンスレベル／ピクチャレベル／スライスレベル／タイルグループレベルで信号通知されてもよい。
１９．上述した開示された方法を適用するかどうか、および／またはどのように適用するかは、カラーフォーマット、シングル／デュアルツリー分割等のコーディングされた情報に依存してもよい。

【０１８２】
５．　実施形態
以下は、上記第４章に要約されたいくつかの発明の態様のためのいくつかの例示的な実施形態であり、ＶＶＣ仕様に適用できる。太字のイタリック体において、既に追加または修正された最も関連する部分には下線を付し、削除された部分のうちのいくつかは、［［］］を使用して示す。

【０１８３】
５．１．　実施形態１
９．３．２．２　コンテキスト変数の初期化処理

【０１８４】
表５１－初期化処理における各ｉｎｉｔｉａｌｉｚａｔｉｏｎＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け

【０１８５】
【表３３】

構文構造	構文要素	ctxTable	initType		
			0	1	2
coding_unit( )	amvr_precision_idx[ ][ ]	Table 89	0..[[1]]*2*	[[2]]*3*..[[3]]*5*	[[4]]*6*..[[5]]*8*

【０１８６】
表８９－ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのｃｔｘＩｄｘのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様
【０１８７】
【表３４】

初期化変数	amvr_precision_idx の ctxIdx								
	0	1	2	3	4	5	6	7	8
initValue	EP	EP	EP	EP	EP	EP	EP	EP	EP
shiftIdx	0	0	0	0	0	0	0	0	0

【０１８８】
９．３．４．２　ｃｔｘＴａｂｌｅ，ｃｔｘＩｄｘ，ｂｙｐａｓｓＦｌａｇの導出処理
９．３．４．２．１　一般
【０１８９】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て
【０１９０】
【表３５】

構文要素	binIdx					
	0	1	2	3	4	>= 5
amvr_precision_idx[ ][ ]	[[0]]*(CuPredMode[0][x0][y0] == MODE_IBC) ? X : (inter_affine_flag == 0 ? Y : Z)*	[[1]]*X*	na	na	na	na

【０１９１】
【化１】
Ｘ、Ｙ、Ｚのうちのいずれか２つが互いに等しくない（例えば、Ｘ！＝Ｙ！＝Ｚ）場合、例えば、Ｘ＝１、Ｙ＝０、Ｚ＝２である。
代替１：Ｘ！＝Ｙ！＝Ｚ，Ｘ＝０，Ｙ＝２，Ｚ＝１．
代替２：Ｘ！＝Ｙ！＝Ｚ，Ｘ＝０，Ｙ＝１，Ｚ＝２．
代替３：Ｘ！＝Ｙ！＝Ｚ，Ｘ＝１，Ｙ＝２，Ｚ＝０．
代替４：Ｘ！＝Ｙ！＝Ｚ，Ｘ＝２，Ｙ＝０，Ｚ＝１．
代替５：Ｘ！＝Ｙ！＝Ｚ，Ｘ＝２，Ｙ＝１，Ｚ＝０．
代替的には、以下を適用する。

【０１９２】

【表３６】

構文要素	binIdx					
	0	1	2	3	4	>= 5
amvr_precision_idx[ ][ ]	[[0]]*(CuPredMode[0][x 0][y0] == MODE_IBC) ? X : (inter_affine_flag == 0 ? Y : Z)*	[[1]]*W*	na	na	na	na

【０１９３】
上記の例において、Ｘ！＝Ｙ，Ｘ！＝Ｚ，Ｙ！＝Ｚである。
代替的には、さらに以下を適用する：
１）一例において、ＷはＸに等しい。
２）代替的に、ＷはＹに等しい。
３）代替的に、ＷはＺに等しい。
【０１９４】
５．２．　実施形態２
９．３．２．２　コンテキスト変数の初期化処理
【０１９５】
表５１－初期化処理における各ｉｎｉｔｉａｌｉｚａｔｉｏｎＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け
【０１９６】
【表３７】

構文構造	構文要素	ctxTable	initType		
			0	1	2
coding_unit( )	amvr_precision_idx[ ][ ]	Table 89	0..[[1]]*3*	[[2]]*4*..[[3]]*7*	[[4]]*8*..[[5]]*11*

【０１９７】
表８９－ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのｃｔｘＩｄｘのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様
【０１９８】
【表３８】

初期化変数	amvr_precision_idx の ctxIdx											
	0	1	2	3	4	5	*6*	*7*	*8*	*9*	*10*	*11*
initValue	EP	EP	EP	EP	EP	EP	*EP*	*EP*	*EP*	*EP*	*EP*	*EP*
shiftIdx	0	0	0	0	0	0	*0*	*0*	*0*	*0*	*0*	*0*

【０１９９】
９．３．４．２　ｃｔｘＴａｂｌｅ，ｃｔｘＩｄｘ，ｂｙｐａｓｓＦｌａｇの導出処理
９．３．４．２．１　一般
【０２００】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て
【０２０１】

【表３９】

構文要素	binIdx					
	0	1	2	3	4	>= 5
amvr_precision_idx[ ][ ]	[[0]]*(CuPredMode[0][x 0][y0] == MODE_IBC) ? X : (inter_affine_flag == 0 ? Y : Z)*	[[1]]*W*	na	na	na	na

【０２０２】
【化２】
X、Y、Z、Wのうちいずれか２つが互いに等しくない（例えば、X！＝Y！＝Z！＝W）場合、例えば、X＝０、Y＝１、Z＝２、W＝３である。

【０２０３】
５．３．　実施形態３
９．３．２．２　コンテキスト変数の初期化処理
【０２０４】
表５１－初期化処理における各ｉｎｉｔｉａｌｉｚａｔｉｏｎＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け
【０２０５】
【表４０】

構文構造	構文要素	ctxTable	initType		
			0	1	2
coding_unit( )	amvr_precision_idx[ ][ ]	Table 89	0[[..1]]	[[2..3]]*1*	[[4..5]]*2*

【０２０６】
表８９－ａｍｖｒ＿ｐｒｅｃｉｓｉｏｎ＿ｉｄｘのｃｔｘＩｄｘのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様
【０２０７】
【表４１】

初期化変数	amvr_precision_idx の ctxIdx					
	0	1	2	[[3	4	5]]
initValue	EP	EP	EP	[[EP	EP	EP]]
shiftIdx	0	0	0	[[0	0	0]]

【０２０８】
９．３．４．２　ｃｔｘＴａｂｌｅ，ｃｔｘＩｄｘ，ｂｙｐａｓｓＦｌａｇの導出処理
９．３．４．２．１　一般
【０２０９】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て

【０２１０】
【表４２】

構文要素	binIdx					
	0	1	2	3	4	>= 5
amvr_precision_idx[ ][ ]	0	[[1]]*0*	na	na	na	na

【０２１１】
５．４．　実施形態４
作業草案は、以下のように変更することができる。
９．３．４．２．３　構文要素ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのｃｔｘＩｎｃｆｏｒの導出プロセス
この処理への入力は、現在のピクチャの左上のサンプルに対する現在の輝度ブロックの左上の輝度サンプル、デュアルツリーチャネルタイプｃｈＴｙｐｅおよび輝度サンプルにおける現在のコーディングブロックの幅と高さｃｂＷｉｄｔｈ、ｃｂＨｅｉｇｈｔ、並びに節７．４．１１．４にでコーディングツリー意味論において導出された変数ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒおよびａｌｌｏｗＳｐｌｉｔである。
【０２１２】
この処理の出力はｃｔｘＩｎｃである。
位置（ｘＮｂＬ，ｙＮｂＬ）は（ｘ０－１，ｙ０）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＬ，ｙＮｂＬ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＬに割り当てられる。
位置（ｘＮｂＡ，ｙＮｂＡ）は（ｘ０，ｙ０－１）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＡ，ｙＮｂＡ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＡに割り当てられる。
【０２１３】

【化３】

ｃｔｘＩｎｃの割り当ては、以下のように指定される。
－ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋［［ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ］］ ***ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ*** が［［［ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ］］ ***ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ*** ＋ａｌｌｏｗＳｐｌｉｔＴｔＨｏｒよりも大きい場合、ｃｔｘＩｎｃは４に等しく設定される。
－ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋［［ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ］］ ***ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ*** が［［［ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ］］ ***ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ*** ＋ａｌｌｏｗＳｐｌｉｔＴｔＨｏｒより小さい場合、ｃｔｘＩｎｃは［［４］］ ***３*** に等しく設定される。
－そうでない場合、以下が適用される。
－変数ｄＡおよびｄＬは、以下のように導出される。
ｄＡ＝ｃｂＷｉｄｔｈ／（ａｖａｉｌａｂｌｅＡ？ＣｂＷｉｄｔｈ［ｃｈＴｙｐｅ］［ｘＮｂＡ］［ｙＮｂＡ］：１）　（１５６３）
ｄＬ＝ｃｂＨｅｉｇｈｔ／（ａｖａｉｌａｂｌｅＬ？ＣｂＨｅｉｇｈｔ［ｃｈＴｙｐｅ］［ｘＮｂＬ］［ｙＮｂＬ］：１）　（１５６４）
－以下の条件のいずれかが真である場合、ｃｔｘＩｎｃは０に等しく設定される。
－ｄＡはｄＬに等しい、
－ａｖａｉｌａｂｌｅＡはＦＡＬＳＥである
－ａｖａｉｌａｂｌｅＬはＦＡＬＳＥである
－そうでない場合、ｄＡがｄＬよりも小さい場合、ｃｔｘＩｎｃは１に等しく設定される。
－そうでない場合、ｃｔｘＩｎｃは［［０］］ ***２*** に等しく設定される。

【０２１４】

５．５．　実施形態５

作業草案は、以下のように変更することができる。

９．３．４．２．３　構文要素ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのｃｔｘＩｎｃｆｏｒの導出プロセス

この処理への入力は、現在のピクチャの左上のサンプルに対する現在の輝度ブロックの左上の輝度サンプル、デュアルツリーチャネルタイプｃｈＴｙｐｅおよび輝度サンプルにおける現在のコーディングブロックの幅と高さｃｂＷｉｄｔｈ、ｃｂＨｅｉｇｈｔ、並びに節７．４．１１．４にでコーディングツリー意味論において導出された変数ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒおよびａｌｌｏｗＳｐｌｉｔである。

この処理の出力はｃｔｘＩｎｃである。

位置（ｘＮｂＬ，ｙＮｂＬ）は（ｘ０－１，ｙ０）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＬ，ｙＮｂＬ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＬに割り当てられる。

位置（ｘＮｂＡ，ｙＮｂＡ）は（ｘ０，ｙ０－１）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＡ，ｙＮｂＡ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＡに割り当てられる。

【０２１５】

【化４】

ｃｔｘＩｎｃの割り当ては、以下のように指定される。

－ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋［［ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ］］ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒが［［［ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ］］ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ＋ａｌｌｏｗＳｐｌｉｔＴｔＨｏｒよりも大きい場合、ｃｔｘＩｎｃは、［［［４］］Ｘに等しく設定される。

－ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋［［ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ］］ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒが［［［ａｌｌｏｗＳｐｌｉｔＴｔＶｅｒ］］ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ＋ａｌｌｏｗＳｐｌｉｔＴｔＨｏｒ未満である場合、ｃｔｘＩｎｃは［［４］］Ｙに等しく設定される。

－そうでない場合、以下が適用される。

－変数ｄＡおよびｄＬは、以下のように導出される。

ｄＡ＝ｃｂＷｉｄｔｈ／（ａｖａｉｌａｂｌｅＡ？ＣｂＷｉｄｔｈ［ｃｈＴｙｐｅ］［ｘＮｂＡ］［ｙＮｂＡ］：１）　（１５６３）

ｄＬ＝ｃｂＨｅｉｇｈｔ／（ａｖａｉｌａｂｌｅＬ？ＣｂＨｅｉｇｈｔ［ｃｈＴｙｐｅ］［ｘＮｂＬ］［ｙＮｂＬ］：１）　（１５６４）

－以下の条件のいずれかが真である場合、ｃｔｘＩｎｃは［［０］］Ｚに等しく設定される。

－ｄＡはｄＬに等しい、

－ａｖａｉｌａｂｌｅＡはＦＡＬＳＥである

－ａｖａｉｌａｂｌｅＬはＦＡＬＳＥである

－そうでない場合、ｄＡがｄＬよりも小さい場合、ｃｔｘＩｎｃを［［［１］］Ｗに等しく設定する。

－そうでない場合、ｃｔｘＩｎｃは［［０］］Ｖに等しく設定される。

ここで、Ｘ！＝Ｙ！＝Ｚ！＝Ｗ！＝Ｖ、例えば、Ｘ＝４、Ｙ＝３、Ｚ＝０、Ｗ＝１、Ｖ＝２

代替１：Ｘ＝３，Ｙ＝４，Ｚ＝０，Ｗ＝１，Ｖ＝２．

代替２：Ｘ＝４，Ｙ＝３，Ｚ＝２，Ｗ＝１，Ｖ＝０．

代替３：Ｘ＝０，Ｙ＝１，Ｚ＝２，Ｗ＝３，Ｖ＝４．

【０２１６】

５．６．　実施形態６

作業草案は、以下のように変更することができる。

９．３．２．２　コンテキスト変数の初期化処理

【０２１７】

表５１－　初期化プロセスにおける各ｉｎｉｔｉａｌｉｚａｔｉｏＴｙｐｅのｃｔｘＩｄｘと構文要素の関連付け

【０２１８】

【表４３】

構文構造	構文要素	ctxTable	initType		
			0	1	2
coding_tree( )	mtt_split_cu_vertical_flag	Table 61	0[[..4]]	[[5..9]]1	[[10..14]]2

【０２１９】

表６１－ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのｃｔｘＩｎｃのｉｎｉｔＶａｌｕｅおよびｓｈｉｆｔＩｄｘの仕様

【０２２０】

【表４４】

初期化変数	mtt_split_cu_vertical_flag の ctxIdx														
	0	1	2	[[3	4	5	6	7	8	9	10	11	12	13	14]]
initValue	EP	EP	EP	[[EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP	EP]]
shiftIdx	0	0	0	[[0	0	0	0	0	0	0	0	0	0	0	0]]

【０２２１】

９．３．４．２　ｃｔｘＴａｂｌｅ，ｃｔｘＩｄｘ，ｂｙｐａｓｓＦｌａｇの導出処理
９．３．４．２．１　一般
【０２２２】
表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て
【０２２３】
【表４５】

構文要素	binIdx					
	0	1	2	3	4	>= 5
mtt_split_cu_vertical_flag	0[[..4 (節 9.3.4.2.3)]]	na	na	na	na	na

【０２２４】
［［９．３．４．２．３　構文要素ｍｔｔ＿ｓｐｌｉｔ＿ｃｕ＿ｖｅｒｔｉｃａｌ＿ｆｌａｇのｃｔｘＩｎｃｆｏｒの導出プロセス
この処理への入力は、現在のピクチャの左上のサンプルに対する現在の輝度ブロックの左上の輝度サンプル、デュアルツリーチャネルタイプｃｈＴｙｐｅおよび輝度サンプルにおける現在のコーディングブロックの幅と高さｃｂＷｉｄｔｈ、ｃｂＨｅｉｇｈｔ、並びに節７．４．１１．４にでコーディングツリー意味論において導出された変数ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ、ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＶｅｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒ、ａｌｌｏｗＳｐｌｉｔＴＨｏｒおよびａｌｌｏｗＳｐｌｉｔである。
この処理の出力はｃｔｘＩｎｃである。
位置（ｘＮｂＬ，ｙＮｂＬ）は（ｘ０－１，ｙ０）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＬ，ｙＮｂＬ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＬに割り当てられる。
位置（ｘＮｂＡ，ｙＮｂＡ）は（ｘ０，ｙ０－１）に等しく設定され、節６．４．４に規定される近傍のブロックの利用可能性の導出処理は、（ｘ０，ｙ０）に等しく設定された位置（ｘＣｕｒｒ，ｙＣｕｒｒ）、（ｘＮｂＡ，ｙＮｂＡ）に等しく設定された近傍位置（ｘＮｂＹ，ｙＮｂＹ）、ＦＡＬＳＥに設定されたｃｈｅｃｋＰｒｅｄＭｏｄｅＹおよびｃＩｄｘを入力として実行され、出力がａｖａｉｌａｂｌｅＡに割り当てられる。
【０２２５】
ｃｔｘＩｎｃの割り当ては、以下のように指定される。
－ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒがａｌｌｏｗＳｐｌｉｔＴＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＴＴＨｏｒより大きい場合、ｃｔｘＩｎｃは４に設定される。
－そうでない場合、ａｌｌｏｗＳｐｌｉｔＢｔＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＢｔＨｏｒがａｌｌｏｗＳｐｌｉｔＴＶｅｒ＋ａｌｌｏｗＳｐｌｉｔＴＴＨｏｒよりも小さい場合、ｃｔｘＩｎｃは４に等しく設定される。
－そうでない場合、以下が適用される：
－変数ｄＡおよびｄＬは、以下のように導出される。
ｄＡ＝ｃｂＷｉｄｔｈ／（ａｖａｉｌａｂｌｅＡ？ＣｂＷｉｄｔｈ［ｃｈＴｙｐｅ］［ｘＮｂＡ］［ｙＮｂＡ］：１）　（１５６３）
ｄＬ＝ｃｂＨｅｉｇｈｔ／（ａｖａｉｌａｂｌｅＬ？ＣｂＨｅｉｇｈｔ［ｃｈＴｙｐｅ］［ｘＮｂＬ］［ｙＮｂＬ］：１）　（１５６４）
－以下の条件のいずれかが真である場合、ｃｔｘＩｎｃは０に等しく設定される。

－ｄＡはｄＬに等しい、
－ａｖａｉｌａｂｌｅＡはＦＡＬＳＥである、
－ａｖａｉｌａｂｌｅＬはＦＡＬＳＥである。
－そうでない場合、ｄＡがｄＬよりも小さい場合、ｃｔｘＩｎｃは１に等しく設定される。
そうでない場合、ｃｔｘＩｎｃは０に等しく設定される。］］

【０２２６】

５．７．　実施形態７

作業草案は、以下のように変更することができる。

７．３．１０．１１　残差コーディング構文

【０２２７】

【表４６】

residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	記述子
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3)	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 − log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 − log2SbH	
}	
numSbCoeff = 1 << (log2SbW + log2SbH)	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − (log2SbW + log2SbH))) − 1	
inferSbCbf = 1	
RemCcbs = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
for(i =0; i <= lastSubBlock; i++) {	
xS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][0]	
yS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][1]	
if(i != lastSubBlock | | !inferSbCbf)	
sb_coded_flag[xS][yS]	ae(v)
if(sb_coded_flag[xS][yS] && i < lastSubBlock)	
inferSbCbf = 0	
/* First scan pass */	
inferSbSigCoeffFlag = 1	
lastScanPosPass1 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(sb_coded_flag[xS][yS] && (n != numSbCoeff − 1 | | !inferSbSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
RemCcbs− −	

【０２２８】

【表４７】

if(sig_coeff_flag[xC][yC])	
inferSbSigCoeffFlag = 0	
}	
CoeffSignLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC] {	
coeff_sign_flag[n]	ae(v)
RemCcbs− −	
CoeffSignLevel[xC][yC] = (coeff_sign_flag[n] > 0 ? −1 : 1)	
abs_level_gtx_flag[n][0]	ae(v)
RemCcbs− −	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
RemCcbs− −	
}	
}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
lastScanPosPass1 = n	
}	
/* Greater than X scan pass (numGtXFlags=5) */	
lastScanPosPass2 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
AbsLevelPass2[xC][yC] = AbsLevelPass1[xC][yC]	
for(j = 1; j < 5; j++) {	
if(abs_level_gtx_flag[n][j − 1]) {	
abs_level_gtx_flag[n][j]	ae(v)
RemCcbs− −	
}	
AbsLevelPass2[xC][yC] + = 2 * abs_level_gtx_flag[n][j]	
}	
lastScanPosPass2 = n	
}	

【０２２９】

【表４８】

/* remainder scan pass */	
RemCcbs−−	
for(n = 0; n <= numSbCoeff − 1; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if((n <= lastScanPosPass2 && AbsLevelPass2[xC][yC] >= 10) || (n > lastScanPosPass2 && n <= lastScanPosPass1 && AbsLevelPass1[xC][yC] >= 2) || (n > lastScanPosPass1 && sb_coded_flag[xS][yS]))	
abs_remainder[n]	ae(v)
if(n <= lastScanPosPass2)	
AbsLevel[xC][yC] = AbsLevelPass2[xC][yC] + 2 * abs_remainder[n]	
else if(n <= lastScanPosPass1)	
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
else { /* bypass */	
AbsLevel[xC][yC] = abs_remainder[n]	
if(abs_remainder[n])	
coeff_sign_flag[n]	ae(v)
}	
if(BdpcmFlag[x0][y0][cIdx] = = 0 && n <= lastScanPosPass1) {	
absLeftCoeff = xC > 0 ? AbsLevel[xC − 1][yC]) : 0	
absAboveCoeff = yC > 0 ? AbsLevel[xC][yC − 1]) : 0	
predCoeff = Max(absLeftCoeff, absAboveCoeff)	
if(AbsLevel[xC][yC] = = 1 && predCoeff > 0)	
AbsLevel[xC][yC] = predCoeff	
else if(AbsLevel[xC][yC] > 0 && AbsLevel[xC][yC] <= predCoeff)	
AbsLevel[xC][yC]−−	
}	
}	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (1 − 2 * coeff_sign_flag[n]) * AbsLevel[xC][yC]	
}	
}	

【０２３０】

表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て

【０２３１】

【表４９】

構文要素	binIdx					
	0	1	2	3	4	>= 5
coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] = = 0 || RemCcbs ***<u>< 3</u>*** [[= = 0]] || slice_ts_residual_coding_disabled_flag	バイパス	na	na	na	na	na
coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] = = 1 && RemCcbs >= [[0]]***<u>3</u>*** && !slice_ts_residual_coding_disabled_flag	0..5 (節 9.3.4.2.10)	na	na	na	na	na

【０２３２】

５．８．　実施形態８

　作業草案は、以下のように変更することができる。

７．３．１０．１１　残差コーディング構文

【０２３３】

【表５０】

residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	記述子
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3)	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 − log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 − log2SbH	
}	
numSbCoeff = 1 << (log2SbW + log2SbH)	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − (log2SbW + log2SbH))) − 1	
inferSbCbf = 1	
RemCcbs = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
for(i =0; i <= lastSubBlock; i++) {	
xS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][0]	
yS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][1]	
if(i != lastSubBlock | | !inferSbCbf)	
sb_coded_flag[xS][yS]	ae(v)
if(sb_coded_flag[xS][yS] && i < lastSubBlock)	
inferSbCbf = 0	
/* First scan pass */	
inferSbSigCoeffFlag = 1	
lastScanPosPass1 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(sb_coded_flag[xS][yS] && (n != numSbCoeff − 1 | | !inferSbSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
RemCcbs− −	

【０２３４】

【表５１】

if(sig_coeff_flag[xC][yC])	
inferSbSigCoeffFlag = 0	
}	
CoeffSignLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC] {	
coeff_sign_flag[n]	ae(v)
RemCcbs− −	
CoeffSignLevel[xC][yC] = (coeff_sign_flag[n] > 0 ? −1 : 1)	
abs_level_gtx_flag[n][0]	ae(v)
RemCcbs− −	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
RemCcbs− −	
}	
}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
lastScanPosPass1 = n	
}	
/* Greater than X scan pass (numGtXFlags=5) */	
lastScanPosPass2 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
AbsLevelPass2[xC][yC] = AbsLevelPass1[xC][yC]	
for(j = 1; j < 5; j++) {	
if(abs_level_gtx_flag[n][j − 1]) {	
abs_level_gtx_flag[n][j]	ae(v)
RemCcbs− −	
}	
AbsLevelPass2[xC][yC] + = 2 * abs_level_gtx_flag[n][j]	
}	
lastScanPosPass2 = n	
}	

【０２３５】

【表５２】

/* remainder scan pass */	
RemCcbs = 0	
for(n = 0; n <= numSbCoeff − 1; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if((n <= lastScanPosPass2 && AbsLevelPass2[xC][yC] >= 10) || (n > lastScanPosPass2 && n <= lastScanPosPass1 && AbsLevelPass1[xC][yC] >= 2) || (n > lastScanPosPass1 && sb_coded_flag[xS][yS]))	
abs_remainder[n]	ae(v)
if(n <= lastScanPosPass2)	
AbsLevel[xC][yC] = AbsLevelPass2[xC][yC] + 2 * abs_remainder[n]	
else if(n <= lastScanPosPass1)	
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
else { /* bypass */	
AbsLevel[xC][yC] = abs_remainder[n]	
if(abs_remainder[n])	
coeff_sign_flag[n]	ae(v)
}	
if(BdpcmFlag[x0][y0][cIdx] = = 0 && n <= lastScanPosPass1) {	
absLeftCoeff = xC > 0 ? AbsLevel[xC − 1][yC]) : 0	
absAboveCoeff = yC > 0 ? AbsLevel[xC][yC − 1]) : 0	
predCoeff = Max(absLeftCoeff, absAboveCoeff)	
if(AbsLevel[xC][yC] = = 1 && predCoeff > 0)	
AbsLevel[xC][yC] = predCoeff	
else if(AbsLevel[xC][yC] > 0 && AbsLevel[xC][yC] <= predCoeff)	
AbsLevel[xC][yC]− −	
}	
}	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (1 − 2 * coeff_sign_flag[n]) * AbsLevel[xC][yC]	
}	
}	

【０２３６】

表１３１－コンテキストコーディングされたビンを有する構文要素へのｃｔｘＩｎｃの割り当て

【０２３７】

【表５３】

構文要素	binIdx					
	0	1	2	3	4	>= 5
coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] = = 0 | | RemCcbs = = 0 | | slice_ts_residual_coding_disabled_flag	バイパス	na	na	na	na	na
coeff_sign_flag[ ] transform_skip_flag[x0][y0][cIdx] = = 1 && RemCcbs >[[=]] 0 && !slice_ts_residual_coding_disabled_flag	0..5 (節 ***9.3.4.2.10***)	na	na	na	na	na

【０２３８】

図１２は、本明細書で開示される様々な技術が実装され得る例示的な映像処理システム１２００を示すブロック図である。様々な実装形態は、システム１２００のコンポーネントの一部または全部を含んでもよい。システム１２００は、映像コンテンツを受信するための入力１２０２を含んでもよい。映像コンテンツは、未加工または非圧縮フォーマット、例えば、８または１０ビットのマルチコンポーネント画素値で受信されてもよく、または圧縮または符号化されたフォーマットで受信されてもよい。入力１２０２は、ネットワークインタフェース、周辺バスインタフェース、または記憶インタフェースを表してもよい。ネットワークインタフェースの例は、イーサネット（登録商標）、ＰＯＮ（Ｐａｓｓｉｖｅ　Ｏｐｔｉｃａｌ　Ｎｅｔｗｏｒｋ）等の有線インタフェース、およびＷｉ－Ｆｉ（登録商標）またはセルラーインタフェース等の無線インタフェースを含む。

【０２３９】

システム１２００は、本明細書に記載される様々なコーディングまたは符号化方法を実装することができるコーディングコンポーネント１２０４を含んでもよい。コーディングコンポーネント１２０４は、入力１２０２からの映像の平均ビットレートをコーディングコンポーネント１２０４の出力に低減し、映像のコーディングされた表現を生成してもよい。従って、このコーディング技術は、映像圧縮または映像コード変換技術と呼ばれることがある。コーディングコンポーネント１２０４の出力は、コンポーネント１２０６によって表されるように、記憶されてもよいし、接続された通信を介して送信されてもよい。入力１２０２において受信された、記憶されたまたは通信された映像のビットストリーム（またはコーディングされた）表現は、コンポーネント１２０８によって使用されて、表示インタフェース１２１０に送信される画素値または表示可能な映像を生成してもよい。ビットストリーム表現からユーザが見ることができる映像を生成する処理は、映像伸張（映像展開）と呼ばれることがある。さらに、特定の映像処理動作を「コーディング」動作またはツールと呼ぶが、コーディングツールまたは動作は、エンコーダおよびそれに対応する、コーディングの結果を逆にする復号ツールまたは動作が、デコーダによって行われることが理解されよう。

【０２４０】

周辺バスインタフェースまたは表示インタフェースの例は、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　ＢＵｓ）またはＨＤＭＩ（Ｈｉｇｈ　Ｄｅｆｉｎｉｔｉｏｎ　Ｍｕｌｔｉｍｅｄｉａ　Ｉｎｔｅｒｆａｃｅ；登録商標）またはディスプレイポート等を含んでも

よい。ストレージインタフェースの例は、ＳＡＴＡ（Ｓｅｒｉａｌ　Ａｄｖａｎｃｅｄ　Ｔｅｃｈｎｏｌｏｇｙ　Ａｔｔａｃｈｍｅｎｔ）、ＰＣＩ、ＩＤＥインタフェース等を含む。本明細書に記載される技術は、携帯電話、ノートパソコン、スマートフォン、またはデジタルデータ処理および／または映像表示を実施可能な他のデバイス等の様々な電子デバイスに実施されてもよい。

【０２４１】

図１３は、映像処理装置３６００のブロック図である。装置３６００は、本明細書に記載の方法の１または複数を実装するために使用されてもよい。装置３６００は、スマートフォン、タブレット、コンピュータ、ＩｏＴ（Ｉｎｔｅｒｎｅｔ　ｏｆ　Ｔｈｉｎｇｓ）受信機等に実施されてもよい。装置３６００は、１または複数のプロセッサ３６０２と、１または複数のメモリ３６０４と、映像処理ハードウェア３６０６と、を含んでもよい。１つまたは複数のプロセッサ３６０２は、本明細書に記載される１または複数の方法を実装するように構成されてもよい。１または複数のメモリ３６０４は、本明細書で説明される方法および技術を実装するために使用されるデータおよびコードを記憶するために使用してもよい。映像処理ハードウェア３６０６は、本明細書に記載される技術をハードウェア回路にて実装するために使用してもよい。

【０２４２】

図１５は、本開示の技法を利用し得る例示的な映像コーディングシステム１００を示すブロック図である。

【０２４３】

図１５に示すように、映像コーディングシステム１００は、送信元デバイス１１０と、送信先デバイス１２０と、を備えてもよい。送信元デバイス１１０は、コーディング映像データを生成するものであり、映像コーディング機器とも称され得る。送信先デバイス１２０は、送信元装置１１０によって生成された、符号化された映像データを復号してよく、映像復号デバイスと呼ばれ得る。

【０２４４】

送信元デバイス１１０は、映像ソース１１２と、映像エンコーダ１１４と、入出力（Ｉ／Ｏ）インタフェース１１６と、を備えてもよい。

【０２４５】

映像ソース１１２は、映像キャプチャデバイスなどのソース、映像コンテンツプロバイダからの映像データを受信するためのインタフェース、および／または映像データを生成するためのコンピュータグラフィックスシステム、またはこれらのソースの組み合わせを含んでもよい。映像データは、１または複数のピクチャを含んでもよい。映像エンコーダ１１４は、映像ソース１１２からの映像データを符号化し、ビットストリームを生成する。ビットストリームは、映像データのコーディングされた表現を形成するビットのシーケンスを含んでもよい。ビットストリームは、コーディングされたピクチャおよび関連付けられたデータを含んでもよい。コーディングされたピクチャは、ピクチャのコーディング表現である。関連付けられたデータは、シーケンスパラメータセット、ピクチャパラメータセット、および他の構文構造を含んでもよい。Ｉ／Ｏインタフェース１１６は、変復調器（モデム）および／または送信機を含んでもよい。符号化された映像データは、ネットワーク１３０ａを介して、Ｉ／Ｏインタフェース１１６を介して送信先デバイス１２０に直接送信されてよい。符号化された映像データは、送信先デバイス１２０がアクセスするために、記録媒体／サーバ１３０ｂに記憶してもよい。

【０２４６】

送信先デバイス１２０は、Ｉ／Ｏインタフェース１２６、映像デコーダ１２４、および表示装置１２２を含んでもよい。

【０２４７】

Ｉ／Ｏインタフェース１２６は、受信機および／またはモデムを含んでもよい。Ｉ／Ｏインタフェース１２６は、送信元デバイス１１０または記憶媒体／サーバ１３０ｂから符号化された映像データを取得してもよい。映像デコーダ１２４は、符号化された映像デー

タを復号してもよい。表示装置１２２は、復号された映像データをユーザに表示してもよい。表示装置１２２は、送信先デバイス１２０と一体化されてもよく、または外部表示装置とインタフェースで接続するように構成される送信先デバイス１２０の外部にあってもよい。

【０２４８】

映像エンコーダ１１４および映像デコーダ１２４は、ＨＥＶＣ（Ｈｉｇｈ Ｅｆｆｉｃｉｅｎｃｙ Ｖｉｄｅｏ Ｃｏｄｉｎｇ）規格、ＶＶＭ（Ｖｅｒｓａｔｉｌｅ Ｖｉｄｅｏ Ｃｏｄｉｎｇ）規格、および他の現在のおよび／または更なる規格等の映像圧縮規格に従って動作してもよい。

【０２４９】

図１６は、映像エンコーダ２００の一例を示すブロック図であり、映像エンコーダ２００は、図１５に示されるシステム１００における映像エンコーダ１１４であってもよい。

【０２５０】

映像エンコーダ２００は、本開示の技術のいずれかまたは全部を行うように構成されてもよい。図１６の例において、映像エンコーダ２００は、複数の機能コンポーネントを備える。本開示で説明される技法は、映像エンコーダ２００の様々なコンポーネント間で共有され得る。いくつかの例では、プロセッサは、本開示で説明される技術のいずれかまたはすべてを行うように構成してもよい。

【０２５１】

映像エンコーダ２００の機能コンポーネントは、分割部２０１、予測部２０２、残差生成部２０７、変換部２０８、量子化部２０９、逆量子化部２１０、逆変換部２１１、再構成部２１２、バッファ２１３、およびエントロピー符号化部２１４を含んでもよく、予測部２０２は、モード選択部２０３、動き推定部２０４、動き補償部２０５、およびイントラ予測部２０６を含む。

【０２５２】

他の例において、映像エンコーダ２００は、さらに多くの、さらに少ない、または異なる機能コンポーネントを含んでもよい。一例において、予測部２０２は、ＩＢＣ（Ｉｎｔｒａ Ｂｌｏｃｋ Ｃｏｐｙ）部を含んでもよい。ＩＢＣ部は、少なくとも１つの参照ピクチャが、現在の映像ブロックが位置するピクチャであるＩＢＣモードにおいて予測を実行してもよい。

【０２５３】

さらに、動き推定部２０４および動き補正部２０５などのいくつかのコンポーネントは、高度に統合されてもよいが、説明のために、図１６の例においては別々に表されている。

【０２５４】

分割部２０１は、ピクチャを１または複数の映像ブロックに分割してもよい。映像エンコーダ２００および映像デコーダ３００は、様々な映像ブロックサイズをサポートしてもよい。

【０２５５】

モード選択部２０３は、例えば、誤りの結果に基づいて、イントラまたはインターのコーディングモードのうちの１つを選択し、得られたイントラまたはインターコーディングされたブロックを、残差ブロックデータを生成するために残差生成部２０７に供給し、符号化されたブロックを参照ピクチャとして使用するために再構成するために再構成部２１２に供給してもよい。いくつかの例において、モード選択部２０３は、インター予測信号およびイントラ予測信号に基づいて予測を行うＣＩＩＰ（Ｃｏｍｂｉｎａｔｉｏｎ ｏｆ Ｉｎｔｒａ ａｎｄ Ｉｎｔｅｒ Ｐｒｅｄｉｃｔｉｏｎ）モードを選択してもよい。モード選択部２０３は、インター予測の場合、ブロックのために動きベクトルの解像度（例えば、サブピクセルまたは整数ピクセル精度）を選択してもよい。

【０２５６】

現在の映像ブロックに対してインター予測を実行するために、動き推定部２０４は、バッファ２１３からの１または複数の参照フレームと現在の映像ブロックとを比較すること

により、現在の映像ブロックのために動き情報を生成してもよい。動き補償部２０５は、現在の映像ブロックに関連付けられたピクチャ以外のバッファ２１３からのピクチャの動き情報および復号されたサンプルに基づいて、現在の映像ブロックのための予測映像ブロックを判定してもよい。

【０２５７】

動き推定部２０４および動き補償部２０５は、現在の映像ブロックがＩスライスであるか、Ｐスライスであるか、またはＢスライスであるかに基づいて、例えば、現在の映像ブロックに対して異なる動作を行ってもよい。

【０２５８】

いくつかの例において、動き推定部２０４は、現在の映像ブロックに対して単一方向予測を行い、動き推定部２０４は、現在の映像ブロックに対して、参照映像ブロック用のリスト０またはリスト１の参照ピクチャを検索してもよい。そして、動き推定部２０４は、参照映像ブロックと、現在の映像ブロックと参照映像ブロックとの間の空間的変位を示す動きベクトルとを含む、リスト０またはリスト１における参照ピクチャを示す参照インデックスを生成してもよい。動き推定部２０４は、参照インデックス、予測方向インジケータ、および動きベクトルを、現在の映像ブロックの動き情報として出力してもよい。動き補償部２０５は、現在の映像ブロックの動き情報が示す参照映像ブロックに基づいて、現在のブロックの予測映像ブロックを生成してもよい。

【０２５９】

他の例において、動き推定部２０４は、現在の映像ブロックを双方向予測してもよく、動き推定部２０４は、現在の映像ブロックに対する参照映像ブロックについて、リスト０から参照ピクチャを検索してもよく、また、現在の映像ブロックに対する別の参照映像ブロックについて、リスト１における参照ピクチャも検索してもよい。そして、動き推定部２０４は、参照映像ブロックを含むリスト０およびリスト１における参照ピクチャを示す参照インデックスと、参照映像ブロックと現在の映像ブロックとの間の空間的変位を示す動きベクトルとを生成してもよい。動き推定部２０４は、現在の映像ブロックの参照インデックスおよび動きベクトルを、現在の映像ブロックの動き情報として出力してもよい。動き補償部２０５は、現在の映像ブロックの動き情報が示す参照映像ブロックに基づいて、現在の映像ブロックの予測映像ブロックを生成してもよい。

【０２６０】

いくつかの例において、動き推定部２０４は、デコーダの復号処理のために、動き情報のフルセットを出力してもよい。

【０２６１】

いくつかの例では、動き推定部２０４は、現在の映像のための動き情報のフルセットを出力しなくてもよい。むしろ、動き推定部２０４は、別の映像ブロックの動き情報を参照して、現在の映像ブロックの動き情報を信号通知してもよい。例えば、動き推定部２０４は、現在の映像ブロックの動き情報が近傍の映像ブロックの動き情報に十分に類似していることを判定してもよい。

【０２６２】

一例において、動き推定部２０４は、現在の映像ブロックに関連付けられた構文構造において、現在の映像ブロックが別の映像ブロックと同一の動き情報を有することを映像デコーダ３００に示す値を示してもよい。

【０２６３】

他の例において、動き推定部２０４は、現在の映像ブロックに関連付けられた構文構造において、別の映像ブロックと、ＭＶＤ（Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｄｉｆｆｅｒｅｎｃｅ）とを識別してもよい。動きベクトル差分は、現在の映像ブロックの動きベクトルと、示された映像ブロックの動きベクトルとの差分を示す。映像デコーダ３００は、示された映像ブロックの動きベクトルおよび動きベクトル差を使用して、現在の映像ブロックの動きベクトルを決定してもよい。

【０２６４】

上述したように、映像エンコーダ２００は、動きベクトルを予測的に信号通知してもよい。映像エンコーダ２００によって実装されてよい予測信号通知技術の２つの例は、ＡＭＶＰ（Ａｄｖａｎｃｅｄ　Ｍｏｔｉｏｎ　Ｖｅｃｔｏｒ　Ｐｒｅｄｉｃｔｉｏｎ）およびマージモード信号通知を含む。

【０２６５】

イントラ予測部２０６は、現在の映像ブロックに対してイントラ予測を行ってもよい。イントラ予測部２０６が現在の映像ブロックにてイントラ予測を実行する場合、イントラ予測部２０６は、同じピクチャにおける他の映像ブロックの復号されたサンプルに基づいて、現在の映像ブロックのための予測データを生成してもよい。現在の映像ブロックのための予測データは、予測された映像ブロックおよび様々な構文要素を含んでもよい。

【０２６６】

残差生成部２０７は、現在の映像ブロックから現在の映像ブロックの予測された映像ブロックを減算することによって（例えば、マイナス符号によって示されている）、現在の映像ブロックに対する残差データを生成してもよい。現在の映像ブロックの残差データは、現在の映像ブロックにおけるサンプルの異なるサンプル成分に対応する残差映像ブロックを含んでもよい。

【０２６７】

他の例において、例えば、スキップモードにおいて、現在の映像ブロックに対する残差データがなくてもよく、残差生成部２０７は、減算動作を行わなくてもよい。

【０２６８】

変換処理部２０８は、現在の映像ブロックに関連付けられた残差映像ブロックに１または複数の変換を適用することによって、現在の映像ブロックのための１または複数の変換係数映像ブロックを生成してもよい。

【０２６９】

変換処理部２０８が現在の映像ブロックに関連付けられた変換係数映像ブロックを生成した後、量子化部２０９は、現在の映像ブロックに関連付けられた１または複数の量子化パラメータ（ＱＰ：Ｑｕａｎｔｉｚａｔｉｏｎ　Ｐａｒａｍｅｔｅｒ）値に基づいて、現在の映像ブロックに関連付けられた変換係数映像ブロックを量子化してもよい。

【０２７０】

逆量子化部２１０および逆変換部２１１は、変換係数映像ブロックに逆量子化および逆変換をそれぞれ適用し、変換係数映像ブロックから残差映像ブロックを再構成してもよい。再構成部２１２は、予測部２０２によって生成された１または複数の予測映像ブロックに対応するサンプルに再構成された残差映像ブロックを追加して、バッファ２１３に格納するための現在のブロックに関連付けられた再構成された映像ブロックを生成してもよい。

【０２７１】

再構成部２１２が映像ブロックを再構成した後、映像ブロックにおける映像ブロッキングアーチファクトを縮小するために、ループフィルタリング動作が行われてもよい。

【０２７２】

エントロピー符号化部２１４は、映像エンコーダ２００の他の機能コンポーネントからデータを受信してもよい。エントロピー符号化部２１４がデータを受信した場合、エントロピー符号化部２１４は、１または複数のエントロピー符号化動作を行い、エントロピー符号化されたデータを生成し、エントロピー符号化されたデータを含むビットストリームを出力してもよい。

【０２７３】

図１７は、映像デコーダ３００の一例を示すブロック図であり、この映像デコーダ３００は、図１５に示すシステム１００における映像デコーダ１１４であってもよい。

【０２７４】

映像デコーダ３００は、本開示の技術のいずれかまたは全てを行うように構成されてもよい。図１７の例において、映像デコーダ３００は、複数の機能コンポーネントを備える。本開示で説明される技法は、映像デコーダ３００の様々なコンポーネント間で共有され

てもよい。いくつかの例では、プロセッサは、本開示で説明される技術のいずれかまたはすべてを行うように構成してもよい。

【０２７５】

図１７の例において、映像デコーダ３００は、エントロピー復号部３０１、動き補正部３０２、イントラ予測部３０３、逆量子化部３０４、逆変換部３０５、および再構成部３０６、並びにバッファ３０７を備える。映像デコーダ３００は、いくつかの例では、映像エンコーダ２００（図１６）に関して説明した符号化パスとほぼ逆の復号パスを行ってもよい。

【０２７６】

エントロピー復号部３０１は、符号化されたビットストリームを取り出す。符号化されたビットストリームは、エントロピーコーディングされた映像データ（例えば、映像データの符号化されたブロック）を含んでもよい。エントロピー復号部３０１は、エントロピーコーディングされた映像データを復号し、エントロピー復号された映像データから、動き補償部３０２は、動きベクトル、動きベクトル精度、参照ピクチャリストインデックス、および他の動き情報を含む動き情報を決定してもよい。動き補償部３０２は、例えば、ＡＭＶＰおよびマージモードを行うことで、このような情報を判定してもよい。

【０２７７】

動き補償部３０２は、動き補償されたブロックを生成してもよく、場合によっては、補間フィルタに基づいて補間を行う。構文要素には、サブピクセルの精度で使用される補間フィルタのための識別子が含まれてもよい。

【０２７８】

動き補償部３０２は、映像ブロックの符号化中に映像エンコーダ２００によって使用されるような補間フィルタを使用して、参照ブロックのサブ整数画素のための補間値を計算してもよい。動き補償部３０２は、受信した構文情報に基づいて、映像エンコーダ２００により使用される補間フィルタを決定し、予測ブロックを生成に補間フィルタを使用してしてもよい。

【０２７９】

動き補償部３０２は、エンコードされた映像シーケンスのフレームおよび／またはスライスをエンコードするために使用されるブロックのサイズを判定するための構文情報、エンコードされた映像シーケンスのピクチャの各マクロブロックがどのように分割されるかを記述する分割情報、各分割がどのようにエンコードされるかを示すモード、各インターエンコードされたブロックに対する１または複数の参照フレーム（および参照フレームリスト）、およびエンコードされた映像シーケンスをデコードするための他の情報のいくつかを使用してもよい。

【０２８０】

イントラ予測部３０３は、例えば、ビットストリームにおいて受信したイントラ予測モードを使用して、空間的に近傍のブロックから予測ブロックを形成してもよい。逆量子化部３０３は、ビットストリームに提供され、エントロピー復号部３０１によって復号された量子化された映像ブロック係数を逆量子化（例えば、逆量子化）する。逆変換部３０３は、逆変換を適用する。

【０２８１】

再構成部３０６は、残差ブロックと、動き補償部２０２またはイントラ予測部３０３によって生成された対応する予測ブロックとを合計し、復号されたブロックを形成してもよい。所望であれば、ブロックアーチファクトを除去するために、復号されたブロックをフィルタリングするためにデブロッキングフィルタを適用してもよい。デコードされた映像ブロックは、バッファ３０７に記憶され、バッファ３０７は、後続の動き補償／イントラ予測のために参照ブロックを提供し、また表示デバイスに表示するためにデコードされた映像を生成する。

【０２８２】

次に、いくつかの実施形態において好適な解決策を列挙する。

【０２８３】
以下の解決策は、前章（例えば、項目１）で論じた技術の例示的な実施形態を示す。
【０２８４】
１．映像処理方法（例えば、図１４に示される方法１４００）は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、変換は、映像ブロックに対する動きベクトルまたは動きベクトル差分または動きベクトル予測子の表現が、適応解像度を用いてコーディングされた表現において表されるＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づいて行われ、フォーマット規則は、映像ブロックまたは映像ブロックの近傍のブロックのコーディングされた情報に依存するコンテキストモデリングによって、コーディングされた表現において適応解像度の使用を表現することを規定する。
【０２８５】
２．コーディングされた情報は、イントラブロックコピーモードを使用することを含む、解決策１に記載の方法。
【０２８６】
３．コーディングされた情報は、アフィンＡＭＶＲモードまたは非アフィンおよび非イントラブロックコピーモード、双予測または単一予測モードの使用を含む、解決策１に記載の方法。
【０２８７】
４．コーディングされた情報は、映像ブロックの寸法を含む、解決策１から３のいずれかに記載の方法。
【０２８８】
以下の解決策は、前章（例えば、項目２）で論じた技術の例示的な実施形態を示す。
【０２８９】
５．映像処理の方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、変換は映像ブロックに対する動きベクトルまたは動きベクトル差分または動きベクトル予測子の表現が、適応解像度を使用してコーディング表現において表されるＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づいて行われ、フォーマット規則は、ＡＭＶＲツールによって使用される精度のインデックスのための第１のビンおよび第２のビンをコーディングするために使用されるコンテキストモデリングによって、コーディングされた表現における適応解像度の使用を表現する方法を規定する。
【０２９０】
６．フォーマット規則は、第１のビンを使用することを規定し、第２のビンは同じコンテキストを使用してコーディングされる、解決策５に記載の方法。
【０２９１】
７．フォーマット規則は、映像ブロックをコーディングされた表現で表すために非アフィンおよび非イントラブロックコピーモードが使用される場合、かつその場合にのみ、第２のビンをコーディングされた表現でコーディングすることを規定する、解決策５に記載の方法。
【０２９２】
以下の解決策は、前章（例えば、項目３から項目８）で論じた技術の例示的な実施形態を示す。
【０２９３】
８．映像処理の方法は、複数の映像ブロックを含む１または複数の映像ピクチャを含む映像と、映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現は、１または複数の映像ブロックのＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）コーディングに関す

る情報を信号通知するためのフォーマット規則に準拠し、フォーマット規則は、第１のコーディングモードを使用してコーディングされた第１の映像ブロックのＡＭＶＲ精度インデックスのビンと、第２のコーディングモードを使用してコーディングされた第２の映像ブロックのＡＭＶＲ精度インデックスのビンとを、同一のコンテキストを使用してコーディングすることを規定する。

【０２９４】

９．第１のコーディングモードは、イントラブロックコピーモードに対応し、第２のコーディングモードは、インターコーディングに対応し、第１の映像ブロックのビンは、ＡＭＶＲ精度インデックスの第１のビンであり、第２の映像ブロックのビンは、対応するＡＭＶＲ精度インデックスの第２のビンである、解決策８に記載の方法。

【０２９５】

１０．第１のコーディングモードは、イントラブロックコピーモードに対応し、第２のコーディングモードは、インターコーディングに対応し、第１の映像ブロックのビンは、ＡＭＶＲ精度インデックスの第１のビンであり、第２の映像ブロックのビンは、対応するＡＭＶＲ精度インデックスの第１のビンである、解決策８に記載の方法。

【０２９６】

１１．第１のコーディングモードは、イントラブロックコピーモードに対応し、第２のコーディングモードは、インターコーディングに対応し、第１の映像ブロックのビンは、ＡＭＶＲ精度インデックスの第１のビンであり、第２の映像ブロックのビンは、対応するＡＭＶＲ精度インデックスの第１のビンである、解決策８に記載の方法。

【０２９７】

１２．第１のコーディングモードは、イントラブロックコピーモードに対応し、第２のコーディングモードは、アフィンコーディングに対応し、第１の映像ブロックのビンは、ＡＭＶＲ精度インデックスの第１のビンであり、第２の映像ブロックのビンは、対応するＡＭＶＲ精度インデックスの第１のビンである、解決策８に記載の方法。

【０２９８】

１３．フォーマット規則は、イントラブロックコピーモード、アフィンモードおよびインターコーディングモードを有する、第１の映像ブロック、第２の映像ブロック、および第３の映像ブロックのすべてのビンをコーディングするために、同じコンテキストを使用することをさらに規定する、解決策８に記載の方法。

【０２９９】

１４．フォーマット規則は、イントラブロックコピーモード、アフィンモード、およびインターコーディングモードを有する、第１の映像ブロック、第２の映像ブロック、および第３の映像ブロックの第１のビンをコーディングするための異なるコンテキスト、および第１の映像ブロック、第２の映像ブロック、および第３の映像ブロックの第２のビンをコーディングするための同じコンテキストを用いることさらに規定する、解決策８に記載の方法。

【０３００】

以下の解決策は、前章（例えば、項目９）で論じた技術の例示的な実施形態を示す。

【０３０１】

１５．フォーマット規則は、精度値をコーディングするために使用される少なくとも１つのコンテキストが、ＡＭＶＲツールの適用可能性を示すフラグをコーディングするために使用されるコンテキストと同じであることをさらに規定する、解決策１から１４のいずれかに記載の方法。

【０３０２】

以下の解決策は、前章（例えば、項目１０、１１）で論じた技術の例示的な実施形態を示す。

【０３０３】

１６．映像処理の方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、映像ブロックは、１または複数の垂直および／または１ま

たは複数の水平分割に分割され、コーディングされた表現は、映像ブロックの分割情報のコンテキストベースのコーディングを規定するフォーマット規則に準拠する。

【０３０４】

１７．フォーマット規則は、分割情報を示す構文要素のコンテキストモデリングが、映像ブロックに対して許可された垂直分割の数および／または前記映像ブロックに対して許可された水平分割の数に依存することを規定する、解決策１６に記載の方法。

【０３０５】

１８．フォーマット規則は、映像ブロックに対して許可された垂直分割の数が映像ブロックに対して許可された水平分割の数よりも大きいかどうかに依存する、解決策１７に記載の方法。

【０３０６】

１９．フォーマット規則は、構文要素をコーディングするためにＮ個のコンテキストを使用することを規定し、Ｎは映像ブロックの寸法または近傍の映像ブロックの寸法に基づく、解決策１７から１８のいずれかに記載の方法。

【０３０７】

以下の解決策は、前章（例えば、項目１２）で論じた技術の例示的な実施形態を示す。

【０３０８】

２０．フォーマット規則は、映像ブロックへの垂直分割の適用可能性を示すフラグをコーディングするために、単一のコンテキストを使用することを規定する、解決策１６から１９のいずれかに記載の方法。

【０３０９】

以下の解決策は、前章（例えば、項目１３、１７）で論じた技術の例示的な実施形態を示す。

【０３１０】

２１．映像処理の方法は、映像の映像ブロックと映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、フォーマット規則は、変換係数の符号を表現するためにコンテキストコーディングまたはバイパスコーディングのいずれを使用するかを決定するために使用されるコーディング条件を規定する。

【０３１１】

２２．コーディング条件は、残りの許可されたコンテキストコーディングされたビンの数に対応する、解決策２１に記載の方法。

【０３１２】

２３．コーディング条件は、映像ブロックとコーディングされた表現との間の変換に使用される変換の種類に対応する、解決策２１に記載の方法。

【０３１３】

以下の解決策は、前章（例えば、項目１４）で論じた技術の例示的な実施形態を示す。

【０３１４】

２４．映像処理の方法は、映像の映像ブロックと、映像のコーディングされた表現との間の変換を実行することを含み、コーディングされた表現はフォーマット規則に準拠し、フォーマット規則は、変換スキップ残差コーディング処理の第３または残差係数走査パスにおける残りの構文要素のバイパスコーディングの開始時に、残りの許可されたコンテキストコーディングされたビンの数を規定する変数に演算が適用されることを規定する。

【０３１５】

２５．変換は、映像をコーディングされた表現に符号化することを含む、解決策１から２４のいずれか１つに記載の方法。

【０３１６】

２６．変換は、映像の画素値を生成するためにコーディングされた表現を復号することを含む、解決策１から２４のいずれか１つに記載の方法。

【０３１７】

２７．解決策１から２６の１または複数に記載の方法を実装するように構成されたプロセッサを備える、映像復号装置。

【０３１８】

２８．解決策１から２６の１つまたは複数に記載の方法を実装するように構成されたプロセッサを備える、映像符号化装置。

【０３１９】

２９．コンピュータコードが記憶されたコンピュータプログラムプロダクトであって、コードは、プロセッサにより実行された際に、プロセッサに、解決策１から２６のいずれか１つに記載の方法を実装させるコンピュータプログラムプロダクト。

【０３２０】

３０．本明細書に記載の方法、装置またはシステム。

【０３２１】

図１８は、本技術の１つまたは複数の実施形態にしたがった映像処理方法１８００を示すフローチャートである。方法１８００は、動作１８１０において、規則に従って映像のブロックと映像のビットストリームとの間の変換を実行することを含む。変換は、ＡＭＶＲ（Ａｄａｐｔｉｖｅ Ｍｏｔｉｏｎ Ｖｅｃｔｏｒ Ｄｉｆｆｅｒｅｎｃｅ Ｒｅｓｏｌｕｔｉｏｎ）ツールに基づいて行われ、規則は、ブロックのコーディングモードの使用に基づいて、ＡＭＶＲシフトに関連する動きベクトル差分の解像度を規定する第１の構文要素のビンの文字列内の第１のビンのためのコンテキストの選択を導出することを規定する。

【０３２２】

いくつかの実施形態において、ブロックはコーディングユニットである。いくつかの実施形態において、ブロックのコーディングモードは、アフィンインターモード、イントラブロックコピーモードまたはノーマルインターモードのうち１つである。いくつかの実施形態において、異なるコーディングモードに対応する複数のコンテキストが第１のビンに適用されてもよい。いくつかの実施形態において、複数のコンテキストは３つのコンテキストを含む。いくつかの実施形態において、各コーディングモードは単一のコンテキストに対応する。

【０３２３】

いくつかの実施形態において、ＩＢＣモードを利用してブロックをコーディングする場合、第１のビンに対する第１のコンテキストが第１の値に割り当てられ、ＩＢＣモードを利用してブロックをコーディングしない場合、第１のコンテキストとは異なる少なくとも１つのコンテキストが、少なくとも１つのインターコーディングモードのための第１のビンに適用可能である。いくつかの実施形態において、第１のビンに対する第２のコンテキストは、ブロックがアフィンインターモードを使用してコーディングされている場合、第２の値に割り当てられ、ブロックが、非アフィンインターモードである通常インターモードを使用してコーディングされている場合、第１のビンに対する第３のコンテキストは第３の値に割り当てられる。第２の値および第３の値は、それぞれ異なる値である。

【０３２４】

いくつかの実施形態において、ビンの文字列の第２のビンのコンテキストは第１のビンに用いられる１または複数のコンテキストと同じである。いくつかの実施形態において、ビンの文字列の第２のビンは単一のコンテキスト値でコーディングされる。いくつかの実施形態において、ＩＢＣモードを利用してコーディングされる第１のブロックに対するビンの文字列の第１のビンと、非アフィンインターモードである通常のインターモードを利用してコーディングされる第２ブロックに対するビンの文字列の第２ビンには同じコンテキストが選択される。

【０３２５】

いくつかの実施形態において、ＩＢＣモードまたはアフィンインターモードを使用してブロックをコーディングする場合、ビンの文字列は第１のビンから構成される。非アフィンインターモードである通常のインターモードを使用してブロックをコーディングする場合、ビンの文字列は、第２のビンをさらに備える。いくつかの実施形態において、第１の

ビンに適用される複数のコンテキストの少なくとも１つは、動きベクトル差分の解像度が輝度サンプルの１／４であるか、または第１の構文要素によって規定されるかどうかを規定する第２の構文要素のために選択された少なくとも１つのコンテキストと同じである。いくつかの実施形態において、ＩＢＣモードを使用してブロックをコーディングしている場合、動きベクトル差分の解像度を規定する第１の構文要素のためのコンテキストは、動きベクトル差分の解像度が輝度サンプルの１／４であるか、または第１の構文要素によって規定されるかどうかを規定する第２の構文要素のために選択されたコンテキストと同じである。いくつかの実施形態において、ＩＢＣモードまたはアフィンモードを使用してブロックをコーディングしていない場合、動きベクトル差分の解像度を規定する第１の構文要素のためのコンテキストは、動きベクトル差分の解像度が輝度サンプルの１／４であるか、または第１の構文要素によって規定されるかどうかを規定する第２の構文要素のために選択されたコンテキストと同じである。いくつかの実施形態において、ビンの文字列内の第１のビンのためのコンテキストにＣｔｘＭの値が割り当てられ、ビンの文字列を有する第２のビンのコンテキストにはＣｔｘＱの値が割り当てられ、ここで、ＣｔｘＭ＝ＣｔｘＱである。いくつかの実施形態において、第１のビンと比較し、第２のビンに異なるコンテキストが選択される。

【０３２６】

いくつかの実施形態において、ブロックがＩＢＣモードにてコーディングされる場合の第１のビンに対する第１のコンテキストと、ブロックがアフィンモードを使用してコーディングされる場合の第１のビンに対する第２のコンテキストと、ブロックがＩＢＣモードもアフィンモードも使用せずにコーディングされる場合の第１のビンに対する第３のコンテキストは同じである。いくつかの実施形態において、ブロックがＩＢＣモードにてコーディングされる場合の第１のビンに対する第１のコンテキストと、ブロックがＩＢＣモードもアフィンモードも使用せずにコーディングされる場合の第１のビンに対する第２のコンテキストは同じである。いくつかの実施形態において、アフィンモードを使用してブロックをコーディングする場合の第１のビンに対する第３コンテキストは、第１のコンテキストおよび第２のコンテキストと異なる。いくつかの実施形態において、ブロックがＩＢＣモードにてコーディングされる場合の第１のビンに対する第１のコンテキストと、ブロックがアフィンモードでコーディングされる場合の第１のビンに対する第２のコンテキストは同じである。いくつかの実施形態において、ブロックがＩＢＣモードにてコーディングされる場合のビンの文字列内のすべてのビンに対するコンテキストと、ブロックがアフィンモードを使用してコーディングされる場合のビンの文字列内のすべてのビンに対するコンテキストと、ブロックがＩＢＣモードもアフィンモードも使用せずにコーディングする場合のビンの文字列内のすべてのビンのためのコンテキストは同じである。

【０３２７】

本発明の実施形態において、ＡＭＶＲツールは、動きベクトルの差分の解像度をブロック単位で適応的に調整するコーディングツールである。

【０３２８】

図１９は、本発明の１または複数の実施形態による映像処理方法１９００を示す流れ図である。方法１９００は、動作１９１０において、規則に従って映像の現在のブロックと映像のビットストリームとの間の変換を実行することを含む。規則は、ブロックを水平方向に分割するか垂直方向に分割するかを規定する構文要素をコーディングするコンテキストを、許可された垂直方向の分割数および許可された水平方向の分割数に基づいて選択することを規定する。許可された垂直分割の数は、許可されたバイナリ（ｂｉｎａｒｙ）垂直分割の数と許可されたターナリ（ｔｅｒｎａｒｙ）垂直分割の数とを含み、許可された水平分割の数は、許可されたバイナリ水平分割の数と許可されたターナリ水平分割の数とを含む。

【０３２９】

いくつかの実施形態において、ブロックはコーディングユニットである。いくつかの実施形態において、コンテンツは許可された垂直分割の数と許可された水平分割の数とを比

較することにより選択される。いくつかの実施形態において、コンテキストは、許可された垂直分割の数が許可された水平分割の数より大きい場合、第１のコンテキストのセットから選択される。いくつかの実施形態において、コンテキストは、許可された垂直分割の数が許可された水平分割の数より少ない場合、第２のコンテキストのセットから選択される。いくつかの実施形態において、第１のコンテキストのセットおよび第２のコンテキストのセットはそれぞれ単一のコンテキストを含む。いくつかの実施形態において、第１のコンテキストのセットにおける単一コンテキストの値は４である。いくつかの実施形態において、第２のコンテキストのセットにおける単一コンテキストの値は３である。

【０３３０】

いくつかの実施形態において、コンテキストは、許可された垂直分割の数が許可された水平分割の数と等しい場合、第３のコンテキストのセットから選択される。いくつかの実施形態において、第３のコンテキストのセットは複数のコンテキストを含む。いくつかの実施形態において、第３のコンテキストのセットは、０の値を有する第３のコンテキスト、１の値を有する第４コンテキスト、および２の値を有する第５のコンテキストとを含む。

【０３３１】

いくつかの実施形態において、第３のコンテキストのセットからのコンテキストの選択は、（１）現在のブロックの上に位置する第１の近傍のブロックと現在のブロックの左に位置する第２の近傍のブロックの利用可能性、（２）現在のブロックの寸法、および／または（３）近傍のブロックの寸法に、さらに基づく。いくつかの実施形態において、コンテキストは、（１）現在のブロックの上に位置する第１の近傍のブロックまたは現在のブロックの左に位置する第２の近傍のブロックのいずれかが利用可能でない場合、または（２）ｄＡがｄＬに等しい場合、ＣｔｘＤの値に割り当てられ、ｄＡは現在のブロックの上に位置する第１の近傍のブロックの幅で除した現在ブロックの幅を表し、かつｄＬは現在のブロックの左に位置する第２の近傍のブロックの高さで除した現在のブロックの高さを表す。いくつかの実施形態において、コンテキストは、ｄＡがｄＬより小さい場合、ＣｔｘＥの値に割り当てられ、ここで、ｄＡは現在のブロックの上に位置する第１の近傍のブロックの幅で除した現在ブロックの幅を表し、かつｄＬは現在のブロックの左に位置する第２の近傍のブロックの高さで除した現在のブロックの高さを表す。いくつかの実施形態において、コンテキストは、ｄＡがｄＬより大きい場合、ＣｔｘＦの値に割り当てられ、ここで、ｄＡは現在のブロックの上に位置する第１の近傍のブロックの幅で除した現在ブロックの幅を表し、かつｄＬは現在のブロックの左に位置する第２の近傍のブロックの高さで除した現在のブロックの高さを表す。

【０３３２】

いくつかの実施形態において、第１のコンテキストのセット、第２のコンテキストのセット、および第３のコンテキストのセットにおけるコンテキストは互いに異なる。

【０３３３】

図２０は、本発明の１または複数の実施形態による映像処理方法２０００を示す流れ図である。方法２０００は、動作２０１０において、規則に従って映像の現在のブロックと映像のビットストリームとの間の変換を実行することを含む。規則は、変換係数レベルの符号を規定する構文要素に対してコンテキストコーディングを使用するかまたはバイパスコーディングを使用するかが、現在のブロックに対して使用される残りの許可されたコンテキストコーディングされたビンの数または変換のタイプに基づくことを規定する。

【０３３４】

いくつかの実施形態において、残りの許可されたコンテキストコーディングされたビンの数が閾値以上である場合、現在のブロックのための変換スキップ残差コーディング処理において、構文要素に対してコンテキストコーディングが用いられる。いくつかの実施形態において、残りの許可されたコンテキストコーディングされたビンの数が閾値より少ない場合、現在のブロックのための変換スキップ残差コーディング処理において、構文要素に対してバイパスコーディングする。いくつかの実施形態において、閾値は、０または３である。

【０３３５】
いくつかの実施形態において、残りの許可されたコンテキストコーディングされたビンの数がＮ以下である場合、構文要素にバイパスコーディングが用いられる。いくつかの実施形態において、残りの許可されたコンテキストコーディングされたビンの数がＮ以上である場合、構文要素にコンテキストコーディングが用いられる。いくつかの実施形態において、残りの許可されたコンテキストコーディングされたビンの数は、変換における変換係数レベルの残りの絶対値を処理する前に、Ｎ以下に修正される。いくつかの実施形態において、Ｎは０、３、または４である。いくつかの実施形態において、Ｎは現在のブロックの特徴に基づく整数である。いくつかの実施形態において、現在のブロックの特徴は、シーケンスパラメータセット、映像パラメータセット、ピクチャパラメータセット、ピクチャヘッダ、スライスヘッダ、タイルグループヘッダ、ラージコーディングユニットの行、ラージコーディングユニットのグループ、ラージコーディングユニットまたはコーディングユニットにおける指示を含む。いくつかの実施形態において、現在のブロックの特徴は、現在のブロックまたは現在のブロックの近傍のブロックの寸法または形状を含む。いくつかの実施形態において、現在のブロックの特徴は、映像のカラーフォーマットの指示を含む。いくつかの実施形態において、現在のブロックの特徴は、変換に別個の、またはデュアルコーディングツリー構造が用いられるかを示す指示を含む。いくつかの実施形態において、現在のブロックの特徴は、スライスタイプまたはピクチャタイプを含む。いくつかの実施形態において、現在のブロックの特徴は、映像の色成分の数を含む。

【０３３６】
いくつかの実施形態において、構文要素のコンテキストコーディングは、残りの許可されたコンテキストコーディングされたビンの数に基づく。いくつかの実施形態において、残りの許可されたコンテキストコーディングされたビンの数を規定する変数は、変換スキップ残差コーディング処理の第３のまたは残りの係数スキャンパスにおける残りの構文要素のバイパスコーディングの開始時に修正される。いくつかの実施形態において、変数は、０の固定値に設定される。いくつかの実施形態において、変数は、１にてデクリメントされる。いくつかの実施形態において、現在のブロックは、ブロックベースの差分パルス符号変調コーディングされたブロックを含むか或いは含まない変換ブロックまたは変換スキップブロックを含む。

【０３３７】
いくつかの実施形態において、本方法を応用するかどうかは、シーケンスレベル、ピクチャレベル、スライスレベルまたはタイルグループレベルで示される。いくつかの実施形態において、指示は、シーケンスヘッダ、ピクチャヘッダ、シーケンスパラメータセット、映像パラメータセット、デコーダパラメータセット、復号能力情報、ピクチャパラメータセット、適応パラメータセット、スライスヘッダまたはタイルグループヘッダに含まれる。いくつかの実施形態において、この方法を適用するかどうか、またはどのように適用するかは、映像のコーディングされた情報に基づく。

【０３３８】
いくつかの実施形態において、変換は、映像をビットストリームに符号化することを含む。いくつかの実施形態において、変換は、ビットストリームから映像を復号することを含む。

【０３３９】
本明細書では、「映像処理」という用語は、映像符号化、映像復号、映像圧縮、または映像展開を指してよい。例えば、映像圧縮アルゴリズムは、映像の画素表現から対応するビットストリーム表現への変換、またはその逆の変換中に適用されてもよい。現在の映像ブロックのビットストリーム表現は、例えば、構文によって規定されるように、ビットストリーム内の同じ場所または異なる場所に拡散されるビットに対応していてもよい。例えば、１つのマクロブロックは、変換およびコーディングされた誤り残差値の観点から、かつビットストリームにおけるヘッダおよび他のフィールドにおけるビットを使用して符号化されてもよい。さらに、変換中、デコーダは、上記解決策で説明されているように、判

定に基づいて、いくつかのフィールドが存在しても存在しなくてもよいという知識を持って、ビットストリームを構文解析してもよい。同様に、エンコーダは、特定の構文フィールドが含まれるべきであるか、または含まれないべきであるかを判定し、構文フィールドをコーディングされた表現に含めるか、またはコーディングされた表現から除外することによって、それに応じてコーディングされた表現を生成してもよい。

【０３４０】

本明細書に記載された開示された、およびその他の解決策、例、実施形態、モジュール、および機能動作の実装形態は、本明細書に開示された構造およびその構造的均等物を含め、デジタル電子回路、またはコンピュータソフトウェア、ファームウェア、もしくはハードウェアで実施されてもよく、またはそれらの１または複数の組み合わせで実施してもよい。開示された、およびその他の実施形態は、１または複数のコンピュータプログラムプロダクト、たとえば、データ処理装置によって実装されるため、またはデータ処理装置の動作を制御するために、コンピュータ可読媒体上に符号化されたコンピュータプログラム命令の１または複数のモジュールとして実施することができる。このコンピュータ可読媒体は、機械可読記憶デバイス、機械可読記憶基板、メモリデバイス、機械可読伝播信号をもたらす物質の組成物、またはこれらの１または複数の組み合わせであってもよい。「データ処理装置」という用語は、例えば、プログラマブルプロセッサ、コンピュータ、または複数のプロセッサ、若しくはコンピュータを含む、データを処理するためのすべての装置、デバイス、および機械を含む。この装置は、ハードウェアの他に、当該コンピュータプログラムの実行環境を作るコード、例えば、プロセッサファームウェア、プロトコルスタック、データベース管理システム、オペレーティングシステム、またはこれらの１または複数の組み合わせを構成するコードを含むことができる。伝播信号は、人工的に生成された信号、例えば、機械で生成した電気、光、または電磁信号であり、適切な受信装置に送信するための情報を符号化するために生成される。

【０３４１】

コンピュータプログラム（プログラム、ソフトウェア、ソフトウェアアプリケーション、スクリプト、またはコードとも呼ばれる）は、コンパイルされた言語または解釈された言語を含む任意の形式のプログラミング言語で記述することができ、また、それは、スタンドアロンプログラムとして、またはコンピューティング環境で使用するのに適したモジュール、コンポーネント、サブルーチン、または他のユニットとして含む任意の形式で展開することができる。コンピュータプログラムは、必ずしもファイルシステムにおけるファイルに対応するとは限らない。プログラムは、他のプログラムまたはデータを保持するファイルの一部（例えば、マークアップ言語文書に格納された１または複数のスクリプト）に記録されていてもよいし、当該プログラム専用の単一のファイルに記憶されていてもよいし、複数の調整ファイル（例えば、１または複数のモジュール、サブプログラム、またはコードの一部を格納するファイル）に記憶されていてもよい。コンピュータプログラムを、１つのコンピュータで実行するように展開することができ、あるいは、１つのサイトに位置する、または複数のサイトにわたって分散され通信ネットワークによって相互接続される複数のコンピュータで実行するように展開することができる。

【０３４２】

本明細書に記載された処理およびロジックフローは、入力データに対して動作し、出力を生成することによって機能を行うための１または複数のコンピュータプログラムを実行する１または複数のプログラマブルプロセッサによって行うことができる。処理およびロジックフローはまた、特定用途のロジック回路、例えば、ＦＰＧＡ（Ｆｉｅｌｄ Ｐｒｏｇｒａｍｍａｂｌｅ Ｇａｔｅ Ａｒｒａｙ）またはＡＳＩＣ（Ａｐｐｌｉｃａｔｉｏｎ Ｓｐｅｃｉｆｉｃ Ｉｎｔｅｇｒａｔｅｄ Ｃｉｒｃｕｉｔ）によって行うことができ、装置はまた、特別目的のロジック回路として実装することができる。

【０３４３】

コンピュータプログラムの実行に適したプロセッサは、例えば、汎用および専用マイクロプロセッサの両方、並びに任意の種類のデジタルコンピュータの任意の１または複数の

プロセッサを含む。一般的に、プロセッサは、リードオンリーメモリまたはランダムアクセスメモリまたはその両方から命令およびデータを受信する。コンピュータの本質的な要素は、命令を実行するためのプロセッサと、命令およびデータを記憶するための１つ以上の記憶装置とである。一般的に、コンピュータは、データを記憶するための１または複数の大容量記憶デバイス、例えば、磁気、光磁気ディスク、または光ディスクを含んでもよく、またはこれらの大容量記憶デバイスからデータを受信するか、またはこれらにデータを転送するように動作可能に結合されてもよい。しかしながら、コンピュータは、このようなデバイスを有する必要はない。コンピュータプログラム命令およびデータを記憶するのに適したコンピュータ可読媒体は、あらゆる形式の不揮発性メモリ、媒体、およびメモリデバイスを含み、例えば、ＥＰＲＯＭ、ＥＥＰＲＯＭ、フラッシュ記憶装置、磁気ディスク、例えば内部ハードディスクまたはリムーバブルディスク、光磁気ディスク、およびＣＤ－ＲＯＭおよびＤＶＤ－ＲＯＭディスク等の半導体記憶装置を含む。プロセッサおよびメモリは、特定用途のロジック回路によって補完されてもよく、または特定用途のロジック回路に組み込まれてもよい。

【０３４４】

本特許明細書は多くの特徴を含むが、これらは、任意の主題の範囲または特許請求の範囲を限定するものと解釈されるべきではなく、むしろ、特定の技術の特定の実施形態に特有であり得る特徴の説明と解釈されるべきである。本特許文献において別個の実施形態のコンテキストで説明されている特定の特徴は、１つの例において組み合わせて実装してもよい。逆に、１つの例のコンテキストで説明された様々な特徴は、複数の実施形態において別個にまたは任意の適切なサブコンビネーションで実装してもよい。さらに、特徴は、特定の組み合わせで作用するものとして上記に記載され、最初にそのように主張されていてもよいが、主張された組み合わせからの１または複数の特徴は、場合によっては、組み合わせから抜粋されることができ、主張された組み合わせは、サブコンビネーションまたはサブコンビネーションのバリエーションに向けられてもよい。

【０３４５】

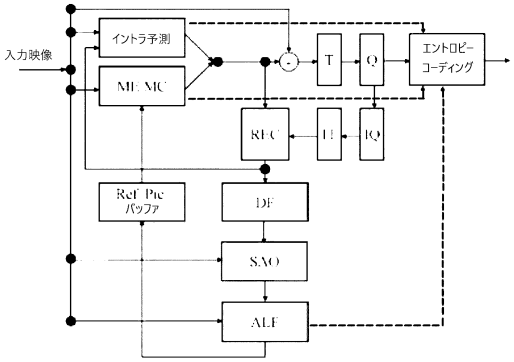
同様に、動作は図面において特定の順番で示されているが、これは、所望の結果を達成するために、このような動作が示された特定の順番でまたは連続した順番で行われること、または示された全ての動作が行われることを必要とするものと理解されるべきではない。また、本特許明細書に記載されている実施形態における様々なシステムの構成要素の分離は、全ての実施形態においてこのような分離を必要とするものと理解されるべきではない。

【０３４６】

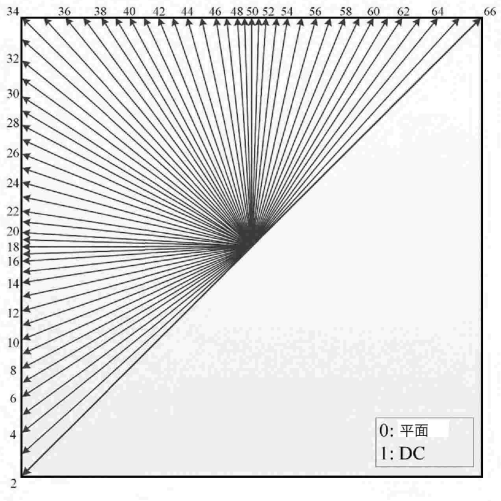
いくつかの実装形態および実施例のみが記載されており、この特許文献に記載され図示されているコンテンツに基づいて、他の実施形態、拡張および変形が可能である。

【図面】

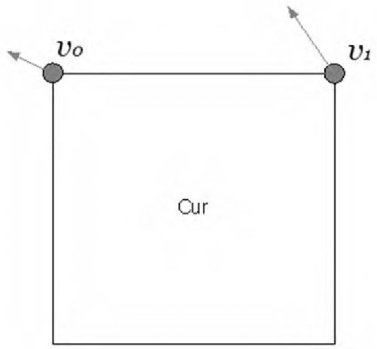
【図１】



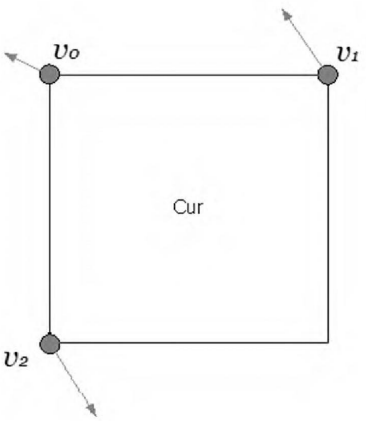
【図２】



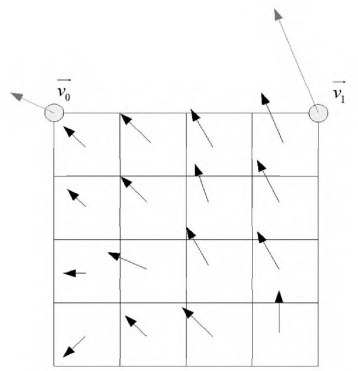
【図３Ａ】



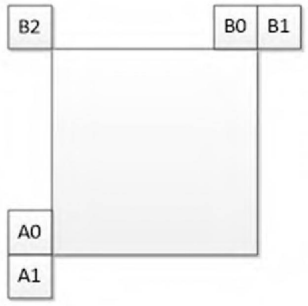
【図３Ｂ】



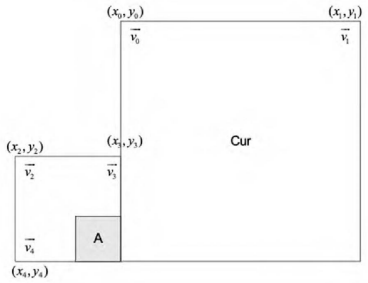
【図４】


$\vec{v}_0$
$\vec{v}_1$

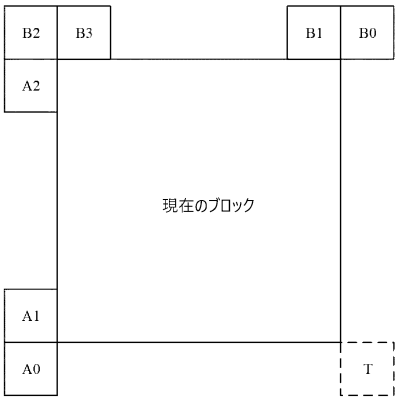
【図５】


B2
B0
B1
A0
A1

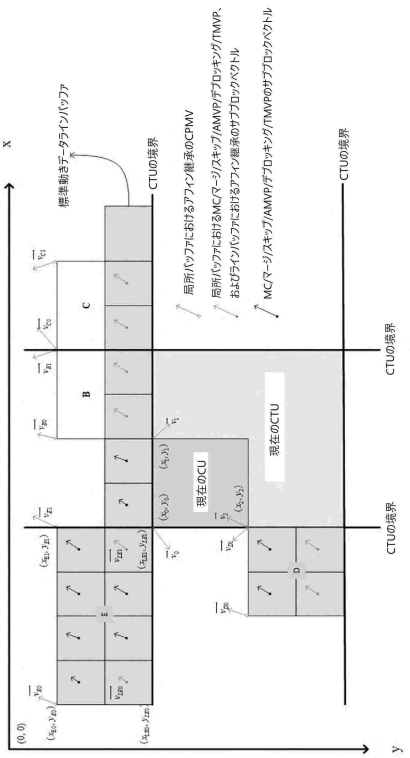
【図６】


(x_0, y_0)
(x_1, y_1)
$\vec{v}_0$
$\vec{v}_1$
(x_2, y_2)
(x_3, y_3)
Cur
$\vec{v}_2$
$\vec{v}_3$
A
$\vec{v}_4$
(x_4, y_4)

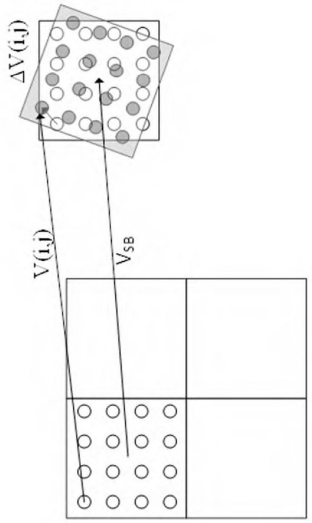
【図７】


B2
B3
B1
B0
A2
現在のブロック
A1
A0
T

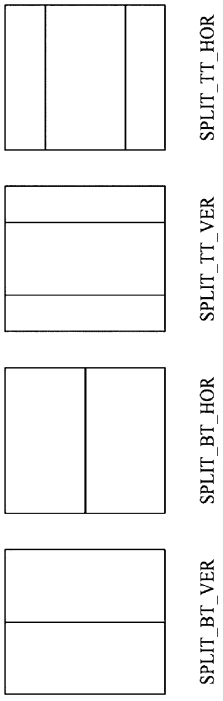
【図８】


x
y
(0, 0)
標準動きデータラインバッファ
CTUの境界
局所バッファにおけるアフィン継承のCPMV
局所バッファにおけるMC/マージ/スキップ/AMVP/デブロッキング/TMVP、
およびラインバッファにおけるアフィン継承のサブブロックベクトル
MC/マージ/スキップ/AMVP/デブロッキング/TMVPのサブブロックベクトル
現在のCU
現在のCTU
A
B
C
D
E

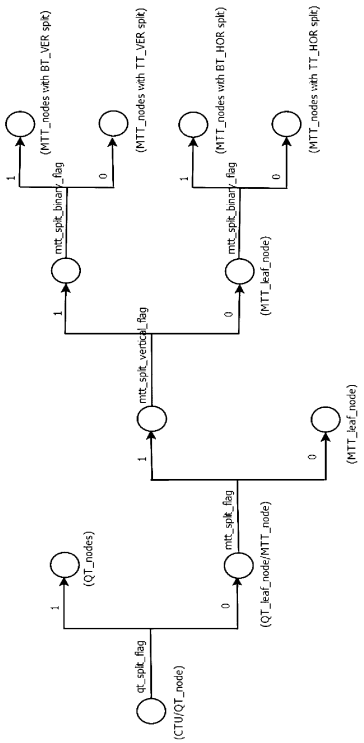
【図９】


ΔV(i,j)
V(i,j)
V_SB

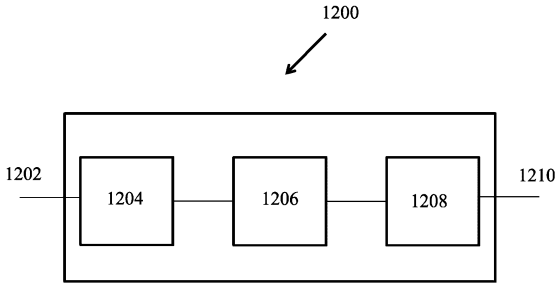
【図１０】


SPLIT_BT_VER
SPLIT_BT_HOR
SPLIT_TT_VER
SPLIT_TT_HOR

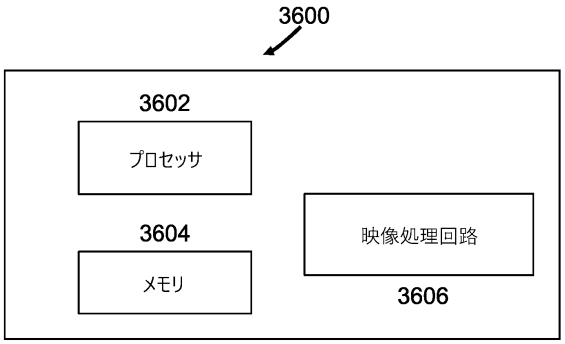
【図１１】


(CTU/QT_node)
qt_split_flag
(QT_nodes)
(QT_leaf_node/MTT_node)
mtt_split_flag
(MTT_leaf_node)
mtt_split_vertical_flag
mtt_split_binary_flag
(MTT_nodes with BT_VER split)
(MTT_nodes with TT_VER split)
(MTT_nodes with BT_HOR split)
(MTT_nodes with TT_HOR split)

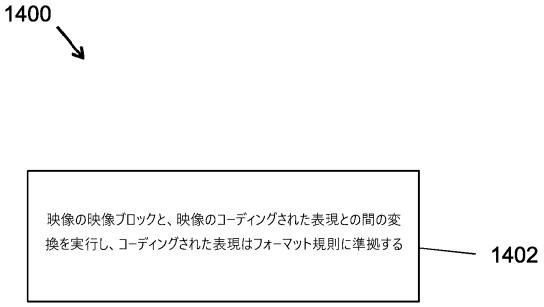
【図１２】



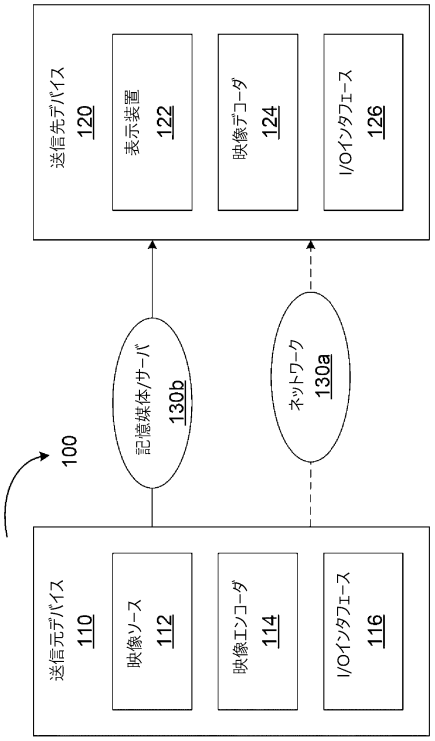
【図１３】



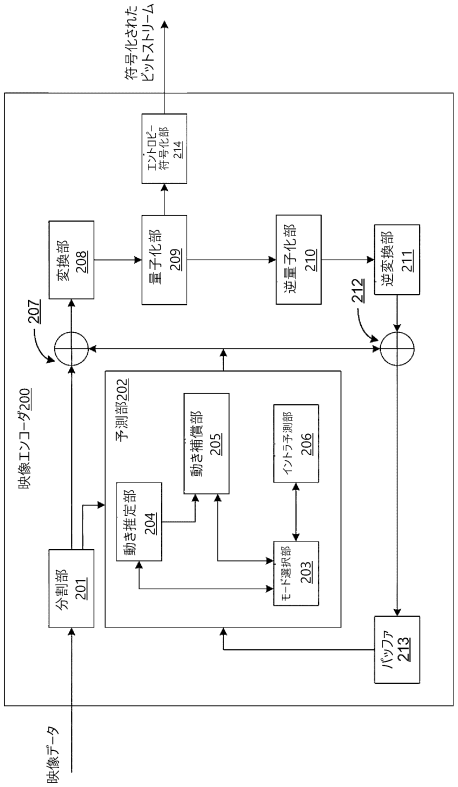
【図１４】



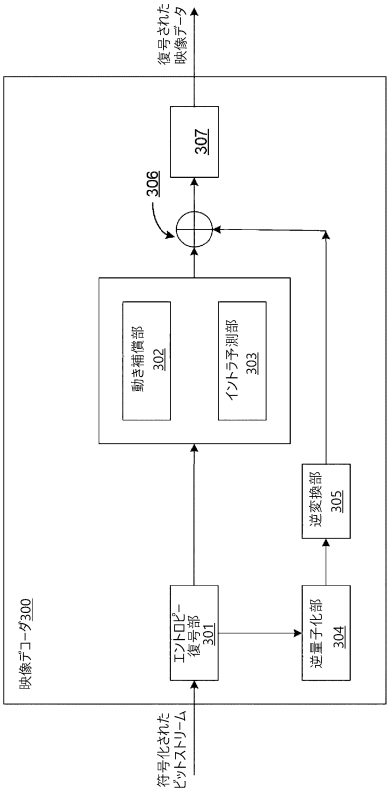
【図１５】



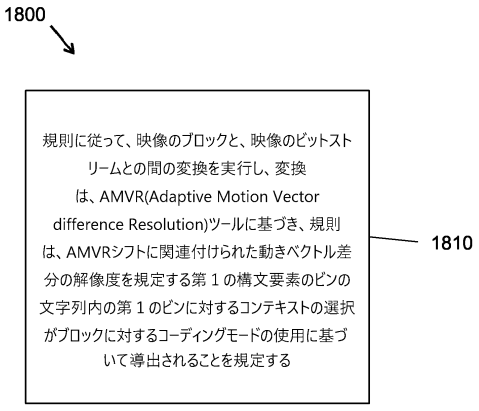
【図１６】



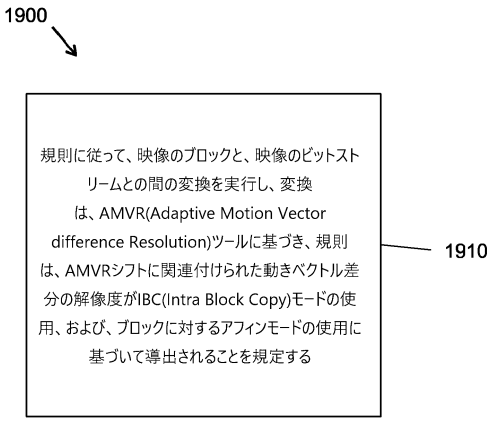
【図１７】



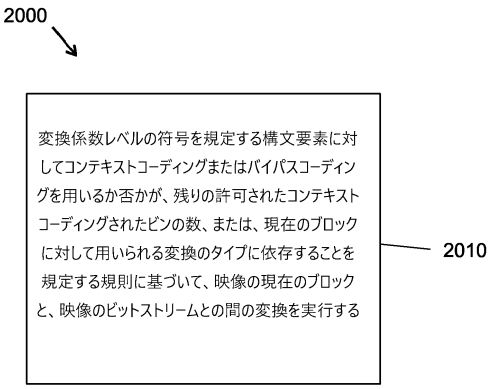
【図１８】



【図１９】



【図２０】


2000
変換係数レベルの符号を規定する構文要素に対してコンテキストコーディングまたはバイパスコーディングを用いるか否かが、残りの許可されたコンテキストコーディングされたビンの数、または、現在のブロックに対して用いられる変換のタイプに依存することを規定する規則に基づいて、映像の現在のブロックと、映像のビットストリームとの間の変換を実行する
2010

フロントページの続き

ＢＹＴＥＤＡＮＣＥ ＩＮＣ．
アメリカ合衆国 カリフォルニア州 ９００６６，ロサンジェルス，ウェスト ジェファーソン ブールヴァード １２６５５，シックスス フロアー，スイート ナンバー・１３７
１２６５５ West Ｊｅｆｆｅｒｓｏｎ Ｂｏｕｌｅｖａｒｄ，Ｓｉｘｔｈ Ｆｌｏｏｒ，Ｓｕｉｔｅ Ｎｏ．１３７ Ｌｏｓ Ａｎｇｅｌｅｓ，Ｃａｌｉｆｏｒｎｉａ ９００６６ Ｕｎｉｔｅｄ Ｓｔａｔｅｓ ｏｆ Ａｍｅｒｉｃａ
(74)代理人 110002000
弁理士法人栄光事務所
(72)発明者 ワン ヤン
中華人民共和国 １０００８０ ベイジン， ハイディアン ディストリクト， ヂーチュン ロード，チャイナ サテライト コミュニケーションズ タワー， ナンバー６３， ジンリトウティアオ ポストオフィス
(72)発明者 ジャン リー
アメリカ合衆国 ９００６６ カリフォルニア州， ロサンゼルス， ウェスト ジェファーソン ブールバード １２６５５， シックスス フロア， スイート ナンバー１３７
(72)発明者 ドン ジピン
中華人民共和国 １０００８０ ベイジン， ハイディアン ディストリクト， ヂーチュン ロード，チャイナ サテライト コミュニケーションズ タワー， ナンバー６３， ジンリトウティアオ ポストオフィス
(72)発明者 ジャン カイ
アメリカ合衆国 ９００６６ カリフォルニア州， ロサンゼルス， ウェスト ジェファーソン ブールバード １２６５５， シックスス フロア， スイート ナンバー１３７
(72)発明者 リウ ホンビン
中華人民共和国 １０００８０ ベイジン， ハイディアン ディストリクト， ヂーチュン ロード，チャイナ サテライト コミュニケーションズ タワー， ナンバー６３， ジンリトウティアオ ポストオフィス

審査官 岩井 健二

(56)参考文献

国際公開第２０２０／２５３８２９（WO，Ａ１）
LIU, Hongbin et al.，CE2: Adaptive Motion Vector Resolution for Affine Inter Mode (Test 2.1.2)，Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting: Marrakech, MA, 9-18 Jan. 2019, [JVET-M0246_r1]，JVET-M0246 (version 2)，ITU-T，2019年01月09日，＜URL:https://jvet-experts.org/doc_end_user/documents/13_Marrakech/wg11/JVET-M0246-v2.zip＞: JVET-M0246_r1.docx: pp.1-10
YANG, Yu-Ciao and LIN, Po-Han，Non-CE4: Unified context model of AMVR and Affine AMVR，Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 14th Meeting: Geneva, CH, 19-27 March 2019, [JVET-N0434]，JVET-N0434 (version 2)，ITU-T，2019年03月21日，＜URL:https://jvet-experts.org/doc_end_user/documents/14_Geneva/wg11/JVET-N0434-v2.zip＞: JVET-N0434.docx: pp.1-3
LI, Guichun, et al.，Non-CE4: Unified AMVR signaling，Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 16th Meeting: Geneva, CH, 1-11 October 2019, [JVET-P0442]，JVET-P0442 (version 3)，ITU-T，2019年09月29日，＜URL:https://jvet-experts.org/doc_end_user/documents/16_Geneva/wg11/JVET-P0442-v3.zip＞: JVET-P0442-v1.docx: pp.1-5
BROSS, Benjamin, et al.，Versatile Video Coding (Draft 8)，Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 17th Meeting: Brussels, BE, 7-17 January 2020, [JVET-Q2001-vE]，JVET-Q2001 (version 15)，ITU-T，2020年03月12日，＜URL:https://jvet-experts.org/doc_end_user/documents/17_Brussels/wg11/JVET-Q2001-v15.zip＞: JVET-Q2001-vE.docx: pp.380-400

(58)調査した分野 (Int.Cl.，ＤＢ名)

H04N 19/00 - 19/98