

明 細 書

発明の名称：

バイトアラインメント装置及びバイトアラインメント方法

技術分野

[0001] 本発明は、ブロック係数列を可変長符号化して得られるビットストリームに対してバイトアラインメントを行うための技術に関するものである。

背景技術

[0002] 近年、様々な環境において、高品質な映像情報の編集処理が行われるようになった。映像情報の編集処理とは、例えば、伝送路中の映像情報の認識処理、単一映像の編集処理、複数の映像の合成処理などである。このような映像情報の編集処理は、データセンターのサーバや個人のパーソナルコンピュータだけでなく、個人が所有する携帯端末や、ネットワーク家電、自動車内オーディオビジュアル装置、ネットワーク内のルータ等で行われることが想定される。ここで、映像情報の編集処理は、通常、MB群（すなわち、MBで構成されるMBライン、単独のMB、任意のMBのグループなど）を単位として行われる。したがって、映像情報の編集処理は、MB群に対応する映像ビットストリームを単位として行われる。ここでMBはマクロブロックを意味する。

[0003] 一方、映像ビットストリームは、H.264の場合、量子化されたブロック係数列を可変長符号化することにより生成される。すなわち、MB群に対応する映像ビットストリームは、MB群に属するブロック係数列の可変長符号化により、ビット単位での符号化情報として生成される。このように生成されたビットストリームは、符号化の対象とした映像信号情報（係数列情報）がバイト区切りの情報であったとしても、必ずしもバイト区切りにはならないことが知られている。

[0004] 映像ビットストリームがバイト区切りでない場合は、映像ビットストリームのコピーやシークという、データ処理作業において、ビットシフト演算処理が必要となり、機器への負担が増えるという問題がある。

[0005] 下記特許文献 1 は、H.264/AVC で用いられるスライスヘッダを用いて、フレーム単位のビットストリームをバイトアラインメントする方法について述べている。しかしながら、特許文献 1 の技術では、フレームのビットストリーム全体を対象としたスライスヘッダでのビット量調整を前提とするため、任意の MB 群におけるバイトアラインメントを行うことができず、使用上の柔軟性に欠ける。また、一つのフレームに挿入可能なスライスヘッダ数には上限があるため、任意の箇所の MB 群においてバイトアラインメントすることは難しいという問題もある。

先行技術文献

特許文献

[0006] 特許文献 1：国際公開番号：WO 2009/093647

発明の概要

発明が解決しようとする課題

[0007] 本発明は、前記の状況に鑑みてなされたものである。本発明の目的の一つは、ビットストリーム抽出処理におけるシフト演算処理を回避し、処理負荷を抑制することが可能な技術を提案することである。本発明のより具体的な目的は、MB 中のブロックの係数列を調整することにより、MB 群単位で符号化されるビットストリームのビット量を調整し、これによって、任意の MB 群から生成されるビットストリームをバイトアラインメントする技術を提供することである。

課題を解決するための手段

[0008] 前記した課題を解決する手段は、以下の項目のように記載できる。

[0009] (項目 1)

ブロック係数列を可変長符号化して得られるビットストリームに対してバイトアラインメントを行うための装置であって、

基準係数列挿入部と、所望増加ビット量算出部と、調整係数列検索部と、調整係数列挿入部とを備えており、

前記基準係数列挿入部は、量子化されたブロック係数列のうちの高周波成分である調整係数列を、既定の基準係数列に置換する構成となっており、

前記所望増加ビット量算出部は、前記調整係数列が前記基準係数列に置換された前記ブロック係数列を可変長符号化して得られるビットストリームの符号量に基づいて、バイトアラインメントのために増加させるべきビット量を算出する構成となっており、

前記調整係数列検索部は、前記増加させるべきビット量に基づいて、前記基準係数列に置換されるべき適切な調整係数列を検索する構成となっており、

前記調整係数列挿入部は、検索された前記調整係数列を前記基準係数列に対応する高周波成分に挿入する構成となっている

バイトアラインメント装置。

[0010] (項目2)

さらにガード係数挿入部を備えており、

前記ガード係数挿入部は、前記ブロック係数列における前記高周波成分係数列よりも1次数だけ低周波側に位置する係数を、ガード係数に置換する構成となっており、

前記ガード係数は、前記ガード係数よりも低周波側の係数の値による、前記ビットストリームの符号量への影響を抑制するものである

項目1に記載のバイトアラインメント装置。

[0011] (項目3)

前記調整係数列検索部は、前記増加させるべきビット量に加えて、可変長符号化後のビットストリームの符号量に影響する要因に基づいて、前記適切な調整係数列を検索する構成となっている

項目1又は2に記載のバイトアラインメント装置。

[0012] (項目4)

さらに検索DBを備えており、

前記検索DBは、前記増加させるべきビット量、及び／又は、可変長符号化

後のビットストリームの符号量に影響する要因に基づいて、前記適切な調整係数列を検索できる構成となっている

項目 1 ～ 3 のいずれか 1 項に記載のバイトアラインメント装置。

[0013] (項目 5)

項目 1 ～ 4 のいずれか 1 項に記載のバイトアラインメント装置と、可変長符号化部とを備えており、

前記可変長符号化部は、前記バイトアラインメント装置によるバイトアラインメント処理の対象となる対象ブロックに隣接する隣接ブロックの係数列を可変長符号化する際には、

前記対象ブロックの符号化における予測モードの制限、及び／又は、符号化に用いる係数値の固定を行う構成となっている

ビットストリーム生成装置。

[0014] (項目 6)

ブロック係数列を可変長符号化して得られるビットストリームに対してバイトアラインメントを行うために用いるビットストリーム生成方法であって、

量子化されたブロック係数列のうちの高周波成分である調整係数列を、既定の基準係数列に置換するステップと、

前記調整係数列が前記基準係数列に置換された前記ブロック係数列を可変長符号化して得られるビットストリームの符号量に基づいて、バイトアラインメントのために増加させるべきビット量を算出するステップと、

前記増加させるべきビット量に基づいて、前記基準係数列に置換されるべき適切な調整係数列を検索するステップと、

検索された前記調整係数列を前記基準係数列に対応する高周波成分に挿入するステップと

を備える、ビットストリーム生成方法。

[0015] (項目 7)

項目 6 に記載の方法をコンピュータに実行させるためのコンピュータプロ

グラム。

[0016] (項目 8)

ブロック係数列によって構成されるデータ構造であって、

前記ブロック係数列における高周波成分係数は、可変長符号化後のビットストリームにおけるバイトアラインメントを行うための調整係数列とされており、

前記調整係数列よりも 1 次数だけ低周波側の係数には、ガード係数が挿入されており、

前記ガード係数は、このガード係数よりも低周波側にある低周波成分係数の値による、可変長符号化条件への影響を抑制するものとなっている

データ構造。

なお、前記したコンピュータプログラム及び／又はデータ構造については、適宜な記録媒体、例えば、電氣的、磁氣的、あるいは光学的な媒体に格納して、コンピュータにより利用することができる。また、この記録媒体は、ネットワークを介して利用可能な、例えばクラウドコンピューティング上での記録媒体であってもよい。

発明の効果

[0017] 本発明によれば、任意のMB群から生成されるビットストリームをバイトアラインメントする技術を提供することができる。

図面の簡単な説明

[0018] [図1]本発明の第 1 実施形態におけるバイトアラインメント処理部を用いた符号化装置の概略的な構成を説明するためのブロック図である。

[図2]第 1 実施形態のバイトアラインメント処理部の概略的な構成を説明するためのブロック図である。

[図3]検索テーブルの構成を説明するための説明図である。

[図4]第 1 実施形態の符号化処理の全体的手順を説明するためのフローチャートである。

[図5]第 1 実施形態のバイトアラインメント処理の手順を説明するためのフロ

ーチャートである。

[図6]ブロック係数列について説明するための説明図である。

[図7]調整係数列の挿入について説明するための説明図である。

[図8]検索テーブルの生成手順を説明するためのフローチャートである。

[図9]第1実施形態のバイトアラインメント処理手順における符号化モードの制約ないし固定を説明するためのフローチャートである。図(a)は第1実施形態での調整対象ブロックを示す。図(b)は第2実施形態での調整対象ブロックと、これに関連する符号化モードの制限ないし固定を示す。

発明を実施するための形態

[0019] 以下、添付図面を参照しながら、本発明の第1実施形態に係る符号化システムについて説明する。

[0020] (本実施形態の構成)

まず、本実施形態の符号化システムが使用される符号化装置全体の概略的構成を、図1を参照しながら説明する。

[0021] このシステムは、整数精度変換部1と、量子化部2と、バイトアラインメント処理部3と、可変長符号化部4とから構成されている。以下、各要素について説明する。

[0022] (整数精度変換部)

整数精度変換部1は、ブロック係数列（フレーム内符号化の場合）またはブロック係数列の残差信号（フレーム間符号化の場合）が入力されるものである。なお、この明細書では、ブロック係数列とその残差信号との両者を含めてブロック係数列と総称する。整数精度変換部1は、実数精度の入力値を整数精度に変換するものであり、これは、従来のH.264での実装と同様に構成できるので、これについての詳しい説明は省略する。

[0023] (量子化部)

量子化部2は、整数精度変換されたブロック係数列を量子化する機能要素である。この量子化部2も、従来のH.264での実装と同様に構成できるので、これについての詳しい説明は省略する。

[0024] (バイトアラインメント処理部)

バイトアラインメント処理部3は、本発明のバイトアラインメント装置の一実施形態に対応する機能要素である。

[0025] 本実施形態のバイトアラインメント処理部3は、図2に示すように、基準係数列挿入部31と、所望増加ビット量算出部32と、調整係数列検索部33と、調整係数列挿入部34と、ガード係数挿入部35と、検索DB36とを備えている。

[0026] 基準係数列挿入部31は、量子化されたブロック係数列のうちの高周波成分である調整用の係数列（後述のTailCoeff）を、既定の基準係数列に置換する構成となっている。

[0027] 所望増加ビット量算出部32は、調整用の係数列が基準係数列に置換された状態のブロック係数列を可変長符号化して得られるビットストリームの符号量に基づいて、バイトアラインメントのために増加させるべきビット量を算出する構成となっている。

[0028] 調整係数列検索部33は、増加させるべきビット量に基づいて、基準係数列（高周波成分係数列）に置換されるべき適切な調整係数列を検索する構成となっている。さらに、本実施形態の調整係数列検索部33は、増加させるべきビット量に加えて、可変長符号化後のビットストリームの符号量に影響する要因（後述）に基づいて、適切な調整係数列を検索する構成となっている。

[0029] 調整係数列挿入部34は、検索された調整係数列を、基準係数列に対応する高周波成分に挿入する構成となっている。

[0030] ガード係数挿入部35は、ブロック係数列における高周波成分係数列よりも1次数だけ低周波側に位置する係数を、ガード係数（以下GuardCoeffといふことがある）に置換する構成となっている。GuardCoeffには固定係数値（本例では絶対値2以上の係数）を挿入する。ここで、ガード係数とは、ガード係数よりも低周波側の係数の値による、ビットストリームの符号量への影響を抑制するものである（後述）。ガード係数の位置は、ガード係数の挿入時

には既に分かっているものとする。ガード係数の位置の探索については後述する。

[0031] 検索DB 3 6 は、増加させるべきビット量、及び／又は、可変長符号化後のビットストリームの符号量に影響する要因に基づいて、適切な調整係数列を検索できる構成となっている。本実施形態の検索DB 3 6 は、具体的にはテーブル構造となっている（図 3 参照）が、DBの構造には特に制約はない。本明細書では、検索DBを検索テーブルと称することがある。本実施形態では、ビットストリーム符号量に影響するパラメータの一部を検索テーブルの検索キーとし、残りのパラメータを固定値としている。

[0032] 本実施形態のバイトアラインメント処理部 3 の詳しい構成については、本実施形態の動作として後述する。

[0033] （可変長符号化部）

可変長符号化部 4 は、各ブロックの係数列に対して、可変長符号化を行う機能要素である。

[0034] 本実施形態の可変長符号化部 4 は、「バイトアラインメント装置によるバイトアラインメント処理の対象となる対象ブロックに隣接する隣接ブロックの係数列を可変長符号化する際には、対象ブロックの符号化における予測モードの制限、及び／又は、符号化に用いる係数値の固定を行う構成」となっている。この点を除き、本実施形態の可変長符号化部 4 は、例えばH.264における、CAVLC (Context Adaptive Variable Length Coding) を用いた可変長符号化部と同様に構成することができる。

[0035] 本実施形態の可変長符号化部 4 の詳しい構成についても、本実施形態の動作として後述する。

[0036] （本実施形態におけるビットストリーム生成手順）

以下、本実施形態におけるビットストリーム生成手順を、図 4 をさらに参照しながら説明する。なお、以下の例では、バイトアラインメントの対象となるMB群を、MBが一行に配置されて構成されたMBラインとする。また、本例において係数調整の対象とするブロックは、MBラインにおける最終MBの最終

ブロックとする（図9（b）参照）。

[0037] （図4のステップSA-1～3）

まず、符号化前の映像信号の入力を受け付け、その係数列を整数精度に変換する。さらに、整数精度に変換された係数列を量子化する。ここまでの処理は通常のH.264と同様でよいので、詳しい説明は省略する。

[0038] （図4のステップSA-4～5）

ついで、本実施形態のバイトアラインメント処理部3を用いたバイトアラインメント処理の手順を、図5をさらに参照しながら説明する。

[0039] （図5のステップSB-0）

可変長符号化部4は、「係数列の調整を行うブロック（対象ブロック）を除く各ブロック」について、可変長符号化を行う。その結果得られるビットストリームの合計ビット数により、対象ブロックを除くMBラインのビットストリームのビット量を得ることができる。このビット量と、対象ブロック係数列の符号化によるビットストリームの符号量とを加算することにより、MBライン全体のビットストリームの符号量を取得することができる。この符号量が、バイトアラインメント処理の基準となる。

[0040] （図5のステップSB-1）

まず、バイトアラインメント処理部3は、量子化部2から、対象ブロックについての量子化処理が行われたブロック係数列を取得する。ここで、ブロック係数列の例を、図6を参照しながら補足して説明する。H.264では、1枚の画像を構成するフレーム100が、複数のMBライン101～10nで構成されており、各MBラインは、複数のMBで構成されている（図6（a）参照）。そして、各MBは、複数のブロックにより構成されている（図6（b）参照）。H.264では、各ブロックは、4×4画素形成されている。図6では、各ブロックに1～16のインデックスを付している。そして、量子化により、各インデックスに対応する量子化された係数値を特定することができる。このような各インデックスの係数値を、ジグザグスキャン（図6（c）参照）することによって、ブロックにおける係数列を生成することができる（図6（

d) 参照)。図 6 (d) に示す数字は、係数のインデックスを示しており、係数値を示すものではない。図 6 (d) において、11 番の係数インデックスが、後述するガード係数となっており、それよりも高周波側（図 6 において右側）の係数列が、調整のために置換される係数列（以下TailCoeffとすることがある）であり、それよりも低周波側（図 6 において左側）の係数列が、信号の内容を示す係数列（以下LeftPartCoeffとすることがある）である。なお、本例において係数列を調整する対象となるブロックのインデックス番号は16である（図 6 (b) 参照）。

[0041] (図 5 のステップ S B - 2)

ついで、基準係数列挿入部 31 は、前記したTailCoeffを、予め準備しておいた基準係数列（以下BaseCoeffとすることがある）に置換する。本実施形態では、基準係数列として、(0, 0, 0, 0, 0) という値が用いられている。これは、CAVLCにおいては、(0, 0, 0, 0, 0) という係数列を用いたときに、符号化後のビット量を最小とすることができるためである。

[0042] (図 5 のステップ S B - 3)

ついで、当該ブロック係数列の符号化が、初期のTailCoeffを置換した後の最初のものであるかどうかを確認する。すなわち、当該ブロック係数列の可変長符号化は、バイトアラインメントのために2度行われる。1度目は、TailCoeffを基準係数列で置き換えた係数列を可変長符号化し、2度目は、TailCoeffを調整係数列で置き換えた係数列を可変長符号化する。ここでの判定がYesであればステップ S B - 4 に進む。

[0043] (図 5 のステップ S B - 4 及び S B - 5)

ついで、TailCoeffをBaseCoeffに置換し、置換後の係数列を用いて、可変長符号化部 4 により可変長符号化を行う。

[0044] (図 5 のステップ S B - 6 及び S B - 7)

つぎに、所望増加ビット量算出部 32 は、可変長符号化により得られた対象ブロックの符号量と、ステップ S B - 0 で取得していた符号量とを合計して得られた符号量（つまりMBライン全体の符号量）をXとし、8を法とする

Xの剰余量、つまり $X \bmod 8$ を計算する。そして、この剰余量と8との差を算出する。その結果が、ビットストリームのビット数についての所望増加量となる。

[0045] (図5のステップS B-8)

ついで、得られた所望増加量と、他の係数特性とを用いて、検索テーブル36から、TailCoeffに挿入されるべき調整係数列を検索して取得する。すなわち、検索テーブル36においては、増加ビット量に影響する各要因（本例では所望増加ビット量、LeftPartTotalCoeff、NeighborTotalCoeff）に基づいて、適切な調整係数列を検索できるようになっている。この検索についてはさらに後述する。

[0046] なお、ガード係数の位置（つまりGuardIndex）は、図8に示す検索テーブルの生成時に取得されるが、これについても後述する。

[0047] ここで、係数列の説明に用いられる用語について、下記表1に示す。

[0048] [表1]

表1： バイトアラインメント対象ブロックにおける係数列の構造

情報	値	説明
絶対値 A	3	TailCoeff の数値定義域 (絶対値が3の場合: -3,-2,-1,0,1,2,3 の値をとる)
GuardCoeff	2	固定係数値
GuardIndex	11	GuardCoeff のインデックス (位置)
GuardTotalCoeff	1	GuardCoeff の非ゼロ係数個数
LeftPartCoeff	係数列	GuardCoeff より低周波成分に位置する係数列
LeftPartTotalCoeff	6	LeftPart の非ゼロ係数個数
TailCoeff	係数列	調整係数列 (事前生成した調整係数列と置き換わる係数列で、LeftPartTotalCoeff, 隣接ブロックの TotalCoeff, 所望増加ビット数を基に、事前生成した調整係数列テーブルから対応する調整係数列が取得され置き換わる)
TailTotalCoeff	2	TailCoeff の非ゼロ係数個数

[0049] また、可変長符号化における符号化テーブルの選択要因となるパラメータ

と、本実施形態における措置とを下記表2に示す。

[0050] [表2]

表2: 符号量に影響するパラメータの種類

No	項目	定義値	使用方法	備考
1	TailTotalCoeff	調整係数列で置き換わったTailCoeffのTailTotalCoeffは、FixedTailTotalCoeffと同値	固定値	TailCoeffの非ゼロ個数(TotalCoeffの要素) ※TotalCoeff = LeftPartTotalCoeff + GuardTotalCoeff + TailTotalCoeff
2	TailCoeffの数値定義域(絶対値A)	絶対値3以下	固定値	理由: TailCoeffの数値定義域の値によっては、LeftPartCoeffのLevelに使用される符号化テーブルが変化するため、本設定値によりこれを抑制する。
3	GuardCoeff	絶対値2以上の係数	固定値	理由: 絶対値0および1の場合、TrailingOnes、TrailingOnesFlag、RunBeforeの判定の際に、LeftPartCoeffの係数も対象とする可能性があり、符号量に影響を及ぼすため、GuardCoeffに絶対値2以上の係数を置くことでこれを抑制する。
4	GuardTotalCoeff	1 (GuardCoeffの係数値により必ず1になる)	固定値	GuardCoeffの非ゼロ係数個数(TotalCoeffの要素)
5	GuardIndex	11 (運用方法によって適切な値を設定)	固定値	GuardCoeffのインデックス位置で、検索テーブル作成時に運用方法によって適切な位置を決定する。
6	FixedTailTotalCoeff	2 (運用方法によって適切な値を設定)	固定値	調整係数列の非ゼロ係数個数で、検索テーブル作成時に運用方法によって適切な値を決定する。
7	LeftPartTotalCoeff	---	検索テーブルの索引	LeftPartCoeffの非ゼロ係数個数(TotalCoeffの要素)
8	IncreaseCodeSize	---	検索テーブルの索引	所望増加ビット量
9	NeighborTotalCoeff	---	検索テーブルの索引 符号化テーブルの索引	隣接ブロックの非ゼロ係数個数の平均値
10	BaseCoeff	全て0の係数列(0,0,0,0,0)	固定値	基準係数列
11	TrailingOnes	GuardCoeffによりTailCoeff固有の値となる		
12	Runbefore	GuardCoeffによりTailCoeff固有の値となる		
13	TrailingOnesFlag	GuardCoeffによりTailCoeff固有の値となる		

[0051] (図5のステップSB-3, SB-9, SB-5)

ついで、手順は再びステップSB-3に戻り、その後ステップSB-9に進む。さらに、調整係数列挿入部34は、検索テーブルから得られた調整係数列を、TailCoeffに挿入する。調整係数列が挿入されたブロック係数列に対して可変長符号化を行い、得られたビットストリームを出力する。このようにして得られたビットストリームを用いて、MBラインのビットストリームを構成することにより、バイトアラインメントを実行することができる。

[0052] 本実施形態では、前記のようにして、バイトアラインメントを、簡便な手段で、実現することができる。さらに、任意のMBに属するブロックを対象として、前記したバイトアラインメントの処理を実行することができる。

[0053] また、本実施形態では、ブロック係数列にガード係数を挿入したので、TailCoeffを調整係数列に置換しても、符号化条件への影響を抑制できるという利点もある。ただし、CAVLCを前提にすると、ガード係数の値は2又は3となる。この点は後述する。

[0054] ここで、ガード係数挿入の利点について補足説明する。

[0055] 実際の可変長符号化の際には、TotalCoeffとTrailingOnesの組合せであるCoeffTokenパラメータも符号化されるので、その符号量が全体ビットストリームの符号量に影響する。ガード係数挿入により、TailCoeffの変更によるCoeffTokenパラメータの符号量への影響を抑制することができる。但し、CoeffTokenの符号化テーブルは、隣接ブロックの非ゼロ係数個数の平均値(NeighborTotalCoeff)により決定するため、係数列全体が同一でも、NeighborTotalCoeffに応じてCoeffTokenの符号量は異なる。よって、あるLeftPartCoeffに対してあるビット量増加を実現するTailCoeffが存在する場合でも、NeighborTotalCoeffが異なると、同一のTailCoeffであっても、所望のビット量増加を実現することは出来ない。以上から、NeighborTotalCoeffが同じ(CoeffTokenを符号化するテーブルが同じ)であるならば、上記パラメータ設定により、LeftPartCoeffとTailCoeffを独立に考えることが可能となり、TailCoeffの係数列調整によって、それ以外の係数列(LeftPartCoeff, GuardCoeff)の符号

化に影響を与えることなく1から7ビットの増加量を制御することが可能となる。

[0056] (図4のステップS A-6)

ついで、バイトアラインメント処理が行われたブロック係数列の可変長符号化により生成されたビットストリームを出力する。これらの処理自体は従来と同様でよいので、詳しい説明は省略する。

[0057] (図4のステップS A-7: 検索テーブル生成手順)

次に、図8を参照して、調整係数列を検索するための検索テーブルの生成手順について説明する。なお、検索テーブルの生成は、前記した調整係数列の検索よりも前に行われる。

[0058] (図8のS C-1)

まず、前提として、GuardCoeffの値は2又は3に設定し、以下で使用する絶対値Aは3以下とする。また、ガード係数の位置(GuardIndex)は、この段階では任意の値であるが、この例では高周波側の値(本例では16)から順次検証していき、1~7ビットのビット量増減値に対応できるGuardIndex(本実施形態では11)をみつける。以下、これらの理由を補足する。

[0059] ・TailCoeffの数値定義域の絶対値Aの値を3以下とすること:

(理由) TailCoeffの絶対値の大きさによって、LeftPartCoeffのLevelに使用される符号化テーブルが変化するため、本設定値によりこれを抑制する。Levelの符号化テーブルはレベル値の絶対値の大きさが0, 3, 6, 12, 24, 48を閾値として変化するため、LeftPartCoeffのLevelに使用される符号化テーブルをTailCoeffの調整と無関係にするためには、TailCoeffの係数値の絶対値の大きさが全て、前記閾値のうち3以下である必要がある。

[0060] ・GuardCoeffの値は、絶対値2以上の係数とすること

(理由) 本設定により、TrailingOnes, TrailingOnesFlag, RunBeforeパラメータの符号量をLeftPartCoeffとTailCoeffで独立することが出来る。絶対値0および1の場合、TrailingOnes, TrailingOnesFlag, RunBeforeの判定の際に、LeftPartCoeffの係数も対象とする可能性がある。そこで、GuardCoeff

に絶対値 2 以上の係数を置くことでこれを抑制することができる。

[0061] ・GuardIndexの値は11（最低 周波の係数インデックスを 1 とした場合）であること。

（理由）この設定値は、以下の説明する検索テーブルの生成に伴って得られるものである。詳しくは図 8 に示す通りであるが、以下において、概要を説明する。すなわち、上記二つの条件である、GuardCoeffが2、絶対値Aが3以下という設定の下、ブロック係数列末尾（つまり右端）の位置からGuardIndexの探索を始める。この場合、いかなる入力 of の組み合わせに対しても、1から7ビット全ての調整が可能なTailCoeffを生成できるGuardIndexが、GuardIndex=11のとき初めて見つかる。後述する処理負荷軽減および高速化を目的とした、事前生成によるTailCoeff検索テーブルの使用時においては、いかなる入力の組み合わせに対しても、1から7ビット全ての調整が可能なTailCoeffの生成が必要となるため、GuardIndexの値として11は好ましいと言える。但し、例えばある入力係数列に対しては、GuardIndex が4のときに、1, 2, 5, 6ビットを調整することができ、GuardIndex が5のときに、2, 3, 5, 7ビットを調整することができるということも想定される。つまり、必要な調整ビット量に応じて、GuardIndexを切り替えて使用することも可能である。このため、GuardIndexの設定値としては、運用（実装）方法によって適切な値を設定することができる。

[0062] また、本例のFixedTailTotalCoeffは2に設定される。

[0063] なお、以下ではまず、LeftPartTotalCoeff（図 7 参照）を全て0とする。ここでは、LeftPartTotalCoeffは、検索テーブル作成のために任意に設定できることが前提である。

[0064] （図 8 の SC-2 及び SC-3）

LeftPartTotalCoeffの値が「0からLeftPartTotalCoeff がとりうる最大値（GuardIndex位置に依存する）まで」の全てのケースにおいて、調整係数列の生成を行うため、LeftPartTotalCoeffがGuardIndexよりも小さい場合は、非ゼロ係数個数がLeftPartTotalCoeffとなるようなLeftPartCoeffを適宜に設

定し、CAVLCの既定のテーブルインデックスを0とする。なお、このインデックスの最大値は、CAVLCにおいて3である。

[0065] (図8のSC-4及びSC-5)

CAVLCのテーブルインデックスが4より小さいときは、TailCoeffにBaseCoeffを設定して、可変長符号化を行う。そして、この符号化後のビット量をBaseCodeSizeとする。

[0066] (図8のSC-6)

ついで、絶対値Aとなる整数値をTailCoeffの各要素とした場合の、すべての組み合わせについて、ブロック係数列を可変長符号化する。つまり、(-3, -3, -3, -3, -3) から、(3, 3, 3, 3, 3) までの全ての組み合わせについて可変長符号化を行い、そのときの符号量をArrangeCodeSizeとする。そして、 $(ArrangeCodeSize - BaseCodeSize) \bmod 8$ の値を、IncreaseCodeSizeとする。

[0067] (図8のSC-7)

ついで、IncreaseCodeSizeとTailCoeffで構成される組を出力する。

[0068] (図8のSC-8)

TailTotalCoeffがFixedTailTotalCoeffと同じで且つ、IncreaseCodeSizeの1〜7ビット全てを満たすTailCoeffが存在するか判定する。判定がYesであればステップSC-9に進む。なお、本例では、FixedTailTotalCoeffの値は2に固定される。つまり、非ゼロ係数個数が2以外の調整係数列は破棄されることになる。ここで、TailTotalCoeffをFixedTailTotalCoeffに固定するのは、以下の理由による。すなわち、可変長符号化のためのテーブル(VLCテーブル)は、TailCoeffによって変わる(他の要因も存在する)。TotalCoeffは、LeftPartTotalCoeffとGuardTotalCoeffとTailTotalCoeffとの和であり、GuardTotalCoeffは本例では1になる。よって、TailTotalCoeffを固定することにより、TotalCoeffに関してはLeftPartTotalCoeffの影響だけを考慮すれば良い。LeftPartTotalCoeffは検索テーブルの索引となる。

[0069] (図8のSC-9)

ついで、IncreaseCodeSizeとTailCoeffで構成される組(調整係数列候補)

が、IncreaseCodeSizeの1～7ビット各々において、複数ある場合には、画質劣化への影響を考慮し以下の基準のもと、一つの調整係数列を選択する。これらの基準を採用したのは、視覚的な影響をなるべく低減させるためである。

- [0070] ・係数列レベル値の加算合計が低い。
- ・係数がより高周波成分である。

[0071] （図8のSC-10）

 ついで、LeftPartTotalCoeff、VLCTableIndex、IncreaseCodeSize、TailCoeffからなる組を、検索テーブルに登録する。

[0072] （図8のSC-11）

 ついで、VLCのTableIndexに1を加えて、ステップSC-4に戻る。

[0073] （図8のSC-12）

 ステップSC-8での判断がNoであれば、いかなる入力 of の組み合わせに対しても、1から7ビット全ての調整が可能なTailCoeffの生成を必要とする条件を満たしていないため、検索テーブルを破棄し、GuardIndex及びFixedTailTotalCoeffのいずれかを再設定する。GuardIndexの再設定を行う場合は、本実施形態では、GuardIndexの値を一つ少なくする。この繰り返しにより、GuardIndexの適切な値を探索することができる。

[0074] （図8のSC-13）

 ステップSC-4での判断がNoであれば、LeftPartTotalCoeffに1を加えて、ステップSC-2に戻る。これを繰り返すことにより、全てのLeftPartTotalCoeffについて、対応した検索テーブルを生成することができる。

[0075] 以上の処理により、図3に示す検索テーブルを生成することができる。

[0076] （第2実施形態）

 次に、第2実施形態に係るバイトアラインメント手順を、図9をさらに参照して説明する。

[0077] 前記した第1実施形態では、バイトアラインメントを行うブロック（つまり対象ブロック）を、MB中の最終ブロックとしていた（図9（a）参照）。

この場合は、対象ブロックを参照するブロックがないので、前記した係数列調整を行なっても、参照不一致による画質劣化の恐れはない。

[0078] しなしながら、仮に、最終ブロック以外のブロックをバイトアラインメントの対象とした場合には、このブロックを参照するブロックがあるために、参照情報の制限ないし固定が必要となる。ただし、これらの制限及び固定は、これらのブロックの符号化モードがintra 4x4である場合にのみ必要となる。

[0079] 以降の説明では、図9（b）に示すb 1 1ブロックが係数列調整の対象であると仮定する。すると、これに隣接するb 1 2（右隣接）、b 1 5（下隣接）、b 1 6（斜め下隣接）のブロックについては、図9（b）に示すような、符号化モードの固定ないし制限が必要となる。さらに、b 1 2とb 1 5のブロックについては、b 1 1のTotalCoeffが固定値であると仮定して符号化を行い、符号化条件の不整合を防止している。つまり、図9（b）の例では、b 1 1をビット量調整の対象ブロックとしたとき、

b 1 2の予測モード：intra_4x4_Vertical

b 1 5の予測モード：intra_4x4_Horizontal

b 1 6の予測モード：下記以外の予測モード

- ・ intra_4x4_Diagonal_Down_Right

- ・ intra_4x4_Vertical_Right

- ・ intra_4x4_Horizontal_Down

とする。さらに、b 1 2とb 1 5については、b 1 1のTotalCoeffを参照して符号化を行うために、符号化時に、b 1 1のTotalCoeffを
LeftPartTotalCoeff+1+TailTotalCoeff
として符号化する。

[0080] なお、本発明は、前記した実施の形態に限定されるものではなく、本発明の要旨を逸脱しない範囲内において種々の変更を加え得るものである。

[0081] 例えば、前記した各構成要素は、機能ブロックとして存在していればよく、独立したハードウェアとして存在しなくても良い。また、実装方法として

は、ハードウェアを用いてもコンピュータソフトウェアを用いても良い。さらに、本発明における一つの機能要素が複数の機能要素の集合によって実現されても良く、本発明における複数の機能要素が一つの機能要素により実現されても良い。

[0082] また、本発明を構成する各機能要素は、離散して存在しても良い。離散して存在する場合には、例えばネットワークを介して必要なデータを受け渡すことができる。各部の内部における各機能も、同様に、離散して存在することが可能である。例えば、グリッドコンピューティングやクラウドコンピューティングを用いて、本実施形態における各機能要素あるいはその一部分を実現することも可能である。

請求の範囲

- [請求項1] ブロック係数列を可変長符号化して得られるビットストリームに対してバイトアラインメントを行うための装置であって、
- 基準係数列挿入部と、所望増加ビット量算出部と、調整係数列検索部と、調整係数列挿入部とを備えており、
- 前記基準係数列挿入部は、量子化されたブロック係数列のうちの高周波成分である調整係数列を、既定の基準係数列に置換する構成となっており、
- 前記所望増加ビット量算出部は、前記調整係数列が前記基準係数列に置換された前記ブロック係数列を可変長符号化して得られるビットストリームの符号量に基づいて、バイトアラインメントのために増加させるべきビット量を算出する構成となっており、
- 前記調整係数列検索部は、前記増加させるべきビット量に基づいて、前記基準係数列に置換されるべき適切な調整係数列を検索する構成となっており、
- 前記調整係数列挿入部は、検索された前記調整係数列を前記基準係数列に対応する高周波成分に挿入する構成となっている
- バイトアラインメント装置。
- [請求項2] さらにガード係数挿入部を備えており、
- 前記ガード係数挿入部は、前記ブロック係数列における前記高周波成分係数列よりも1次数だけ低周波側に位置する係数を、ガード係数に置換する構成となっており、
- 前記ガード係数は、前記ガード係数よりも低周波側の係数の値による、前記ビットストリームの符号量への影響を抑制するものである
- 請求項1に記載のバイトアラインメント装置。
- [請求項3] 前記調整係数列検索部は、前記増加させるべきビット量に加えて、可変長符号化後のビットストリームの符号量に影響する要因に基づいて、前記適切な調整係数列を検索する構成となっている

請求項 1 又は 2 に記載のバイトアラインメント装置。

[請求項4]

さらに検索DBを備えており、

前記検索DBは、前記増加させるべきビット量、及び／又は、可変長符号化後のビットストリームの符号量に影響する要因に基づいて、前記適切な調整係数列を検索できる構成となっている

請求項 1 ～ 3 のいずれか 1 項に記載のバイトアラインメント装置。

[請求項5]

請求項 1 ～ 4 のいずれか 1 項に記載のバイトアラインメント装置と、可変長符号化部とを備えており、

前記可変長符号化部は、前記バイトアラインメント装置によるバイトアラインメント処理の対象となる対象ブロックに隣接する隣接ブロックの係数列を可変長符号化する際には、

前記対象ブロックの符号化における予測モードの制限、及び／又は、符号化に用いる係数値の固定を行う構成となっている

ビットストリーム生成装置。

[請求項6]

ブロック係数列を可変長符号化して得られるビットストリームに対してバイトアラインメントを行うために用いるビットストリーム生成方法であって、

量子化されたブロック係数列のうちの高周波成分である調整係数列を、既定の基準係数列に置換するステップと、

前記調整係数列が前記基準係数列に置換された前記ブロック係数列を可変長符号化して得られるビットストリームの符号量に基づいて、バイトアラインメントのために増加させるべきビット量を算出するステップと、

前記増加させるべきビット量に基づいて、前記基準係数列に置換されるべき適切な調整係数列を検索するステップと、

検索された前記調整係数列を前記基準係数列に対応する高周波成分に挿入するステップと

を備える、ビットストリーム生成方法。

[請求項7] 請求項6に記載の方法をコンピュータに実行させるためのコンピュータプログラム。

[請求項8] ブロック係数列によって構成されるデータ構造であって、
前記ブロック係数列における高周波成分係数は、可変長符号化後のビットストリームにおけるバイトアラインメントを行うための調整係数列とされており、

前記調整係数列よりも1次数だけ低周波側の係数には、ガード係数が挿入されており、

前記ガード係数は、このガード係数よりも低周波側にある低周波成分係数の値による、可変長符号化条件への影響を抑制するものとなっている

データ構造。