(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2012-135017
(P2012-135017A)
(43)公開日　平成24年7月12日(2012.7.12)

(51)Int.Cl.			F I			テーマコード（参考）
H04N	7/30	(2006.01)	H04N	7/133	Z	5C159
H03M	7/30	(2006.01)	H03M	7/30	B	5C178
H03M	7/40	(2006.01)	H03M	7/40		5J064
H04N	1/41	(2006.01)	H04N	1/41	Z	

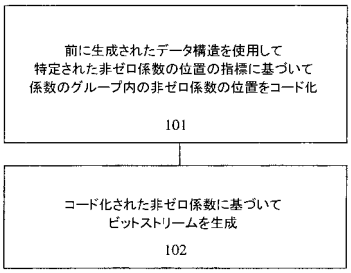
審査請求 有　請求項の数 2　ＯＬ　外国語出願　（全 30 頁）

(21)出願番号　特願2012-32861 (P2012-32861)
(22)出願日　平成24年2月17日 (2012.2.17)
(62)分割の表示　特願2010-161644 (P2010-161644)の分割
原出願日　平成17年12月15日 (2005.12.15)
(31)優先権主張番号　60/639,430
(32)優先日　平成16年12月22日 (2004.12.22)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　11/172,052
(32)優先日　平成17年6月29日 (2005.6.29)
(33)優先権主張国　米国 (US)

(71)出願人　392026693
株式会社エヌ・ティ・ティ・ドコモ
東京都千代田区永田町二丁目１１番１号
(74)代理人　100088155
弁理士　長谷川　芳樹
(74)代理人　100113435
弁理士　黒木　義樹
(74)代理人　100121980
弁理士　沖山　隆
(74)代理人　100128107
弁理士　深石　賢治
(72)発明者　ボッセン，　フランク
アメリカ合衆国，　カリフォルニア州，
サン　ノゼ，　ノースファースト　ストリート　１７００　ナンバー２４２
最終頁に続く

(54)【発明の名称】係数の位置をコード化する方法及び装置

(57)【要約】　（修正有）
【課題】係数のようなデータの位置をコード化し、又、デコードする方法及び装置を提供する。
【解決手段】ツリーデータ構造を使用して特定された、データがゼロであるか又は非ゼロであるかの指標に基づいて、データのベクトル中のデータをコード化するデータコード化ステップと、コード化されたデータに基づいてビットストリームを生成するビットストリーム生成ステップとを備える符号化を行い、そのようにして生成されたビットストリームを受信し復号を行う。
【選択図】図１Ａ



【特許請求の範囲】
【請求項１】
データのベクトル中のデータを、ツリーデータ構造を使用して特定された前記データがゼロであるか又は非ゼロであるかの指標に基づいてコード化するデータコード化ステップと、
コード化されたデータに基づいてビットストリームを作成するビットストリーム作成ステップと、
を備える方法。
【請求項２】
データのベクトルを受信する入力と、
前記入力に接続され、前記データのベクトル中のデータを、ツリーデータ構造を使用して特定された前記データがゼロであるか又は非ゼロであるかの指標に基づいてコード化する、エントロピーエンコーダと、
を備える、係数を符号化する装置。
【請求項３】
システムによって実行されたとき、前記システムに、
データのベクトル中のデータを、ツリーデータ構造を使用して特定された前記データがゼロであるか又は非ゼロであるかの指標に基づいてコード化するデータコード化ステップと、
コード化されたデータに基づいてビットストリームを作成するビットストリーム作成ステップと、
を備える方法を実行させる命令を記憶する１つ以上の記録可能な媒体を有する製品。
【請求項４】
ビットストリームを受信するビットストリーム受信ステップと、
復号化されたデータのベクトルを、ツリーデータ構造によって特定されるような前記データがゼロであるか又は非ゼロであるかの指標に基づいて作成するため、前記ビットストリームを復号化するビットストリーム復号化ステップと、
を備える方法。
【請求項５】
ビットストリームを受信する入力と、
前記入力に接続され、復号化されたデータのベクトルを、ツリーデータ構造によって特定されるような前記データがゼロであるか又は非ゼロであるかの指標に基づいて作成する、エントロピーデコーダと、
を備える、量子化された係数を復号化する装置。
【請求項６】
システムによって実行されたとき、前記システムに、
復号化されたデータのベクトルを、ツリーデータ構造によって特定されるような前記データがゼロであるか又は非ゼロであるかの指標に基づいて作成するため、ビットストリームを復号化するビットストリーム復号化ステップを備える方法を実行させる命令を記憶する１つ以上の記録可能な媒体を有する製品。
【発明の詳細な説明】
【技術分野】
【０００１】
[0001]本特許文献の開示の一部は著作権保護の対象となる素材を含んでいる。著作権所有者は、特許商標局における特許ファイル又は記録物において明白であるので、何人による特許文献又は特許情報開示の複製にも異議を唱えないが、それ以外は、いかなるものであれ全ての著作権を留保する。
［優先権］
[0002]本特許出願は、発明の名称が“Ｍｅｔｈｏｄ ａｎｄ Ａｐｐａｒａｔｕｓ ｆｏｒ Ｃｏｄｉｎｇ Ｐｏｓｉｔｉｏｎｓ ｏｆ Ｎｏｎ－ｚｅｒｏ Ｃｏｅｆｆｉｃｉ

ｅｎｔｓ　ｉｎ　ａ　Ｂｌｏｃｋ”である、２００４年１２月２２日に出願された、対応する仮特許出願第６０／６３９，４３０号に基づく優先権を主張する。

【０００２】

[0003]本発明は、コード化の分野に関し、より詳しくは、本発明は、グループ（例えば、係数のブロック）内のある種の係数（例えば、非ゼロ係数）のコード化に関係する。

【背景技術】

【０００３】

[0004]データ圧縮は、大量のデータを蓄積及び伝送するための非常に有用なツールである。例えば、文書のネットワーク伝送のような画像を伝送するために要する時間は、画像を再現するために必要とされるビット数を減少させるため圧縮が使用されたとき、大幅に短縮される。

【０００４】

[0005]多種多様なデータ圧縮技術が従来技術に存在している。圧縮技術は、損失のあるコード化と損失のないコード化の２つに大別される。損失のあるコード化は、元データの完全な再構成の保証がないほどに、情報の損失を生じるコード化を含む。損失のある圧縮の目標は、元データへの変更が、異議の余地なしに、又は、検出できないように行われる。損失のない圧縮では、全ての情報が保持され、完全な再構成を可能にするようにデータは圧縮される。

【０００５】

[0006]画像符号化器及びビデオ符号化器において、画像は典型的にブロックの組に区分化される。各ブロックは係数の集合に変換され量子化される。係数は、変換（例えば、離散コサイン変換（ＤＣＴ）、ウェーブレット変換）をデータに適用することにより作成できる。例えば、ＪＰＥＧでは、ＤＣＴ変換は、係数を作成するため画像データに適用される。その後、これらの係数は量子化されてもよい。ブロック毎に、どの係数が非ゼロ値を有するかを記述する情報が伝送されなければならない。

【０００６】

[0007]量子化された係数は、典型的に値の１次元配列に配置される。配列中の係数の順序は、走査順序によって、例えば、ジグザグ走査順序によって決定される。非ゼロ係数の位置は、その後に、ランレングス符号化法を使用してコード化される。

【０００７】

[0008]走査方法は、多くの場合に効率が制限されている。例えば、ジグザグ走査方法に関して、ブロックが水平周波数だけ又は垂直周波数だけを収容するとき、多数のゼロ係数がジグザグパターンに沿って訪問され、効率が悪いという結果になる。

【発明の概要】

【０００８】

係数のようなデータの位置をコード化する方法及び装置が記載されている。一実施形態では、方法は、ツリーデータ構造を使用して特定された、データがゼロであるか又は非ゼロであるかの指標に基づいて、データのベクトル中のデータをコード化するデータコード化ステップと、コード化されたデータに基づいてビットストリームを作成するビットストリーム作成ステップとを備える。

【０００９】

[0009]本発明は、後述される詳細な説明と、本発明を特定の実施形態に制限するように解釈されるべきでなく、説明と理解のためだけに用いられる本発明の種々の実施形態の添付図面とから、より十分に理解される。

【図面の簡単な説明】

【００１０】

【図１Ａ】データを符号化するプロセスの一実施形態のフローチャートである。

【図１Ｂ】データを復号化するプロセスの一実施形態のフローチャートである。

【図２】エンコーダ及びデコーダのシステムの一実施形態のブロック図である。

【図３】変換エンコーダの一実施形態のブロック図である。

【図４】ツリー構造データの例である。
【図５】係数エンコーダの一実施形態のブロック図である。
【図６】係数エンコーダによって実行される符号化プロセスの一実施形態のフローチャートである。
【図７】値をツリーのノードに割り当てるプロセスの一実施形態のフローチャートである。
【図８】ツリー中のノードを符号化するプロセスの一実施形態のフローチャートである。
【図９】子ノードの値を符号化するプロセスの一実施形態のフローチャートである。
【図１０】子ノードの値を符号化するプロセスの別の実施形態のフローチャートである。
【図１１】変換デコーダの一実施形態のブロック図である。
【図１２】係数デコーダの一実施形態のブロック図である。
【図１３】量子化された係数を復号化するプロセスの一実施形態のフローチャートである。
【図１４】ノードを復号化するプロセスの一実施形態のフローチャートである。
【図１５】子ノードの値を復号化するプロセスの一実施形態のフローチャートである。
【図１６】子ノードの値を復号化するプロセスの代替的な実施形態のフローチャートである。
【図１７】図４のツリー構造の表現の例を説明する図である。
【図１８】量子化された係数を復号化するプロセスの一実施形態のフローチャートである。
【図１９】係数デコーダの一実施形態のブロック図である。
【図２０】コンピュータシステムの一実施形態のブロック図である。
【発明を実施するための形態】
【００１１】
[0031]データのグループ（例えば、データのブロック）内の係数（例えば、非ゼロ係数）の位置をコード化する方法及び装置について記載されている。一実施形態では、係数は二分木に配置される。ツリーは節（ノード）を系統的に一つ残らず一回ずつなぞられ（以下、トラバースされ）、非ゼロ係数の位置をコード化する。本発明の実施形態は、伝統的なジグザグ走査を使用する係数のコード化への代替案を提供し、ツリー内での係数の体系化に依存し、それによって、単一の走査より高い柔軟性及び良い効率を提供する。
【００１２】
[0032]後に続く実施形態は係数を用いて動作するものとして記載されているが、本明細書に記載された技術が任意のデータのベクトルに適用されることは当業者に明白であろう。
【００１３】
[0033]以下の記載では、本発明のより徹底的な説明を行うために様々な詳細が示されている。しかし、本発明が、これらの特定の詳細を用いることなく実施されてもよいことは当業者にとって明白であろう。他の例では、よく知られている構成及び機器は、本発明を不明瞭化することを避けるために、詳細にではなく、ブロック図形式で示されている。
【００１４】
[0034]後述する詳細な説明の一部は、コンピュータメモリ内のデータビットに関する演算のアルゴリズム及び記号的表現という点で提示されている。これらのアルゴリズム的な記述及び表現は、データ処理技術の当業者によって、自分の業績の内容をその技術における他の当業者へ最も効率的に伝達するために使用される手段である。アルゴリズムは、ここでは、一般的に、望ましい結果をもたらすステップの首尾一貫したシーケンスであると考えられる。ステップは物理量の物理的操作を必要とするステップである。通常、不可欠ではないが、これらの量は、記憶、転送、合成、比較及びそれ以外の操作がなされ得る、電気的又は磁気的信号の形式をとる。主として慣用上の理由で、これらの信号を、ビット、値、要素、記号、文字、項、数などと呼ぶことが、時として好都合であることがわかった。

【００１５】
[0035]しかし、上記の用語及び類似した用語の全ては、適切な物理量と関連付けられるべきであり、これらの量に当てはめられた便宜的なラベルに過ぎないことに留意されたい。以下の説明から明白であるように、特に断らない限り、説明の全体を通じて、「処理」又は「コンピューティング」又は「計算」又は「決定」又は「表示」などのような用語を利用する説明は、コンピュータシステムのレジスタ及びメモリ内で物理（電子）量として表現されたデータを、コンピュータシステムメモリ若しくはレジスタ内、又は、その他のこのような情報記憶装置、伝送機器若しくは表示機器内で同じように物理量として表現されたその他のデータへ操作、変換する、コンピュータシステム又は類似した電子コンピューティング機器の作用及びプロセスを指すことが認められる。

【００１６】
[0036]本発明は、本明細書に記載された演算を実行する装置にも関係する。この装置は、要求された目的のため特に構成されるか、又は、コンピュータに格納されたコンピュータプログラムによって選択的に作動若しくは再構成される汎用コンピュータを備えてもよい。このようなコンピュータプログラムは、限定されることはないが、例えば、フロッピー（登録商標）ディスク、光ディスク、ＣＤ－ＲＯＭ、及び、磁気光ディスクを含む任意のタイプのディスク、リードオンリーメモリ（ＲＯＭ）、ランダムアクセスメモリ（ＲＡＭ）、ＥＰＲＯＭ、ＥＥＰＲＯＭ、磁気若しくは光カード、又は、電子命令を格納するのに適当であり、それぞれコンピュータシステムバスに接続されている任意のタイプの媒体のような、コンピュータ読み取り可能な記憶媒体に格納されてもよい。

【００１７】
[0037]本明細書に提示されているアルゴリズム及びディスプレイは、本質的に特定のコンピュータ又はその他の装置に関連していない。種々の汎用システムが本明細書における教示に従うプログラムと共に使用されてもよく、又は、要求された方法ステップを実行するためにより特化された装置を構築すると好都合であることがわかる。多種多様のこれらのシステムに要求される構成は以下の記載から明らかであろう。さらに、本発明は特定のプログラミング言語に関して記載されていない。多種多様のプログラミング言語が、本明細書に記載されているような発明の教示を実施するために使用されてもよいことが認められるであろう。

【００１８】
[0038]機械読み取り可能な媒体は、機械（例えば、コンピュータ）によって読み取り可能な形式で情報を格納又は伝送する任意の仕組みを含む。例えば、機械読み取り可能な媒体は、リードオンリーメモリ（ＲＯＭ）、ランダムアクセスメモリ（ＲＡＭ）、磁気ディスク記憶媒体、光記憶媒体、フラッシュメモリデバイス、電気的、光学的、音響的又はその他の形式の伝搬信号（例えば、搬送波、赤外線信号、デジタル信号など）などを含む。

【００１９】
（概要）
[0039]図１Ａはデータを符号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングはファームウェアによって実行されてもよい。

【００２０】
[0040]図１Ａを参照すると、プロセスは、プロセッシングロジックが、前に生成されたデータ構造を使用して特定された非ゼロ係数の位置の指標に基づいて、係数のグループ内の非ゼロ係数の位置をコード化することによって開始する（処理ブロック１０１）。一実施形態では、データ構造はツリーを表現する。ツリーは二分木でもよい。一実施形態では、係数のグループは係数のブロックを含む。

【００２１】
[0041]次に、プロセッシングロジックはコード化された非ゼロ係数に基づいてビットス

トリームを生成する（処理ブロック１０２）。
【００２２】
[0042]図１Ｂはデータを復号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングはファームウェアによって実行されてもよい。
【００２３】
[0043]図１Ｂを参照すると、プロセスは、プロセッシングロジックがビットストリームを受信することによって開始する（処理ブロック１１１）。次に、プロセッシングロジックは、前に生成されたデータ構造によって特定された量子化された係数の位置の指標に基づいて、係数のグループ内の量子化された係数の位置を得るために前に生成されたデータ構造を使用することを含めて、量子化された係数の集合を生成するためにビットストリームを復号化する（処理ブロック１１２）。
【００２４】
[0044]一実施形態では、データ構造表現はツリーである。ツリーは二分木でもよい。一実施形態では、係数のグループは係数のブロックを含む。
【００２５】
[0045]図２はエンコーダ及びデコーダシステムの一実施形態のブロック図である。一部のシステムはエンコーダ又はデコーダだけを含んでもよい。
【００２６】
[0046]図２を参照すると、エンコーダ２０２はソース画像２０１を、ソース画像２０１の圧縮表現であるビットストリーム２０３に変換する。デコーダ２０４は、ビットストリーム２０３を、再構成画像２０５に変換する。再構成画像２０５は、ソース画像２０１の近似（損失のある圧縮コンフィギュレーションにおける）又は（損失のない圧縮コンフィギュレーションにおける）正確な複製である。ビットストリーム２０３は、通信チャネル（例えば、インターネットなど）を介して、又は、物理的な媒体（例えば、ＣＤ－ＲＯＭ）を介して、エンコーダ２０２からデコーダ２０３へ送信されてもよい。
【００２７】
[0047]図３は、変換エンコーダの一実施形態のブロック図である。図３の変換エンコーダは、図２のエンコーダ２０２のコンポーネントでもよい。エンコーダは、ソース画像を１個以上の画像データのブロック、例えば、４×４又は８×８のサイズのブロックに区分化できる。その他の画像データのグループ分けが使用されてもよい。
【００２８】
[0048]図３を参照すると、変換ユニット３０２は、画像データのブロック３０１を係数３０３の集合に変換する。一実施形態では、変換ユニット３０２は、離散コサイン変換（ＤＣＴ）を実施できる。量子化器３０４は、係数３０３を量子化された係数３０５の集合に変換する。一実施形態では、係数は実数値をとり、量子化された係数３０５のうちの１個の値は整数に制限される。一実施形態では、量子化器３０４は、所定の数によって各係数を除算し、結果を最も近い整数に丸め、量子化された係数を生成する。係数エンコーダ３０６は、量子化された係数３０５をビットストリーム３０７に変換する。一実施形態では、ビットストリーム３０７は、量子化された係数３０５の表現を含むビットの順序付き集合である。一実施形態では、係数エンコーダ３０６はハフマン符号化を使用する。代替的に、係数エンコーダ３０６は、算術符号化を、単独で又はハフマン符号化と併せて使用する。その他のコード化スキームもまた同様に使用できる。
【００２９】
[0049]本明細書における目的のため、ツリーを記述する用語は以下の通り使用される。図４は二分木の例である。図４を参照すると、ツリーは、５個のノードＡ、Ｂ、Ｃ、Ｄ及びＥを収容している。ノードＡはルートノードである。ノードＤ及びＥは、本明細書では、Ｃの子ノードと呼ばれる。ノードＤはリーフノードの例である。リーフノードは子ノー

ドを所有しない。ノードＢ及びＥもまたリーフノードである。

【００３０】

[0050]以降の記載は二分木を考慮するが、二分木以外のツリー（例えば、三分木）も同様に使用される。

【００３１】

[0051]図１７は、ツリー構造データの例である。図１７を参照すると、ツリー構造データは、ノードタイプデータ１７０１及びリーフインデックスデータ１７０２を収容してもよい。ノードタイプデータ１７０１は、図４の二分木のようなツリー中の各ノードのノードタイプを収容する。例えば、値０はリーフノードではないノードに対し使用されてもよく、値１はリーフノードであるノードに対し使用される。一実施形態では、ノードタイプの順序付けは、ツリーの深さ優先トラバースによって決定される。代替的に、幅優先トラバースが使用される。リーフインデックスデータ１７０２はツリー中のリーフ毎に係数インデックスを収容する。ツリーのリーフにおけるインデックスは、量子化された係数の配列へのインデックスを表現する。一実施形態では、リーフインデックスの順序付けは、ツリーの深さ優先トラバースによって決定できる。代替的に、幅優先トラバースが考慮される。図１７の例では、図４に描かれたツリーが記述され、ノードＢはインデックス０を、ノードＤはインデックス２を、ノードＥはインデックス１を有することを仮定している。

【００３２】

[0052]図５は、係数エンコーダの一実施形態のブロック図である。図５を参照すると、ツリー生成器５０２は、量子化された係数５０１を受信し、メモリ５１０からツリー構造データ５０６を取り出す。ツリー構造データ５０６は、二分木を記述する。一実施形態では、ツリー構造データ５０６は、ツリーのトポロジーに関係する情報と、ならびに、ツリー中の各リーフのインデックスとを含む。ツリー生成器５０２は、値をツリーの各ノードに割り当てる。最初に、ツリー生成器５０２は、ツリーの各リーフに、対応する量子化された係数がゼロと一致するならば、ゼロ値を割り当てる。そうでなければ、ツリー生成器５０２は非ゼロ値をリーフに割り当てる。次に、値が、ツリー中のリーフではない各ノードに再帰的に割り当てられる。このようなノードはそれぞれが２個の子を保有する。両方の子が値ゼロを有するならば、ゼロ値がノードに割り当てられる。そうでなければ、ツリー生成器５０２は非ゼロ値をノードに割り当てる。ノードに値が割り当てられているツリー５０３は、エントロピーエンコーダ５０４へ送信される。

【００３３】

[0053]エントロピーエンコーダ５０４は、ツリー生成器５０２からのツリー５０３と量子化された係数５０１とを受信する。エントロピーエンコーダ５０４は、メモリ５１０からツリー構造データ５０６をさらに受信する。エントロピーエンコーダ５０４は、さらに後述されるように、ツリー５０３のノードに割り当てられた値を符号化する。その後、非ゼロである量子化された係数毎に、エントロピーエンコーダ５０４は量子化された係数の値を符号化する。一実施形態では、ツリー及び係数の符号化は、ハフマン符号化を使用して実行される。別の実施形態では、算術符号化が使用されるが、どのようなコード化スキームが使用されてもよい。エントロピーエンコーダ５０４の出力は、コード化された情報を収容するビットストリーム５０５である。

【００３４】

[0054]選択的に、エントロピーエンコーダ５０４は、メモリ５１０から取り出されたビットストリーム５０５にツリー構造データを挿入してもよい。一実施形態では、ツリー構造データは、符号化された形式のビットストリームと共に挿入される。ノードタイプ情報は、１ノード当たり１ビットずつで符号化されてもよく、ノードがリーフであるか否かを示す。リーフインデックスに関する情報は、ツリーのサイズに依存する多数のビットにより符号化されてもよい。例えば、３個のリーフインデックスを伴うツリーは、これらのインデックスを２ビットで符号化可能であり、１６個のリーフインデックスを備えるツリーはこれらのインデックスを４ビットで符号化可能である。一般に、必要なビット数は、$\log_2$（リーフインデックスの個数）におおよそ等しい。ツリー構造データは様々な時点

で送信されてもよい。例えば、代替的な実施形態では、ツリー構造データは、１ピクチャー毎に１回ずつ、ピクチャーのグループ毎に１回ずつ、又は、ビデオシーケンス全体について１回ずつ送信されてもよい。

【００３５】

[0055]図６は、係数エンコーダによって実行される符号化プロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、係数エンコーダのプロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。

【００３６】

[0056]図６を参照すると、プロセスは、プロセッシングロジックが、例えば、量子化器から量子化された係数を受信することによって開始する（処理ブロック６０１）。プロセッシングロジックは、その後、係数の全部がゼロであるか否かをテストする（処理ブロック６０２）。全係数がゼロであるならば、フラグは全係数がゼロであることを示し、プロセスは終了する。そうでなければ、プロセッシングロジックは、ブロック中の少なくとも１個の係数がゼロに等しくないということを特定するフラグを符号化し（処理ブロック６０４）、メモリからツリー構造データを取り出し（処理ブロック６０５）、さらに後述されるように、量子化された係数に基づいてツリー内の各ノードに値を割り当てる（処理６０６）。換言すると、プロセッシングロジックはツリーノード値を生成する。

【００３７】

[0057]ツリーノード値を生成した後、プロセッシングロジックは、さらに後述されるように、ツリー内の各ノードに割り当てられた値を符号化する（処理ブロック６０７）。ツリーを符号化した後、プロセッシングロジックは、非ゼロの量子化された係数の値を符号化する（処理ブロック６０８）。一実施形態では、ハフマン符号化が使用される。代替的に、算術符号化又は別のコード化スキームが使用されてもよい。

【００３８】

[0058]図７は、ツリーのノードの値を割り当てるプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。

【００３９】

[0059]プロセスはルートノードに設定された現ノード（以下、カレントノード）で始まる。図７を参照すると、プロセッシングロジックは、カレントノードがリーフノードであるかどうかを判定する（処理ブロック７０１）。カレントノードがリーフノードであるならば、プロセッシングロジックは、カレントノードに関連付けられた量子化された係数がゼロ係数であるかどうかを判定する（処理ブロック７０２）。係数がゼロであるならば、プロセッシングロジックは０という値をカレントノードに割り当て（処理ブロック７０３）、プロセスは終了し、そうでなければ、プロセッシングロジックは１という値をカレントノードに割り当て（処理ブロック７０４）、プロセスは終了する。

【００４０】

[0060]カレントノードがリーフノードでないならば、プロセスは処理ブロック７１０へ移り、プロセッシングロジックが、カレントノードの第１の子をカレントノードとして用いて図７のプロセスを再帰的に実行し（処理ブロック７１０）、その後、カレントノードの第２の子をカレントノードとして用いて再び図７のプロセスを再帰的に実行する（処理ブロック７１１）。

【００４１】

[0061]その後、プロセッシングロジックは、両方の子ノードが０という値を割り当てられているかどうかを判定する（処理ブロック７１２）。両方の子ノードが０という値を保有するならば、プロセッシングロジックは０という値をカレントノードに割り当て（処理ブロック７０３）、プロセスは終了し、そうでなければ、プロセッシングロジックは１と

いう値をカレントノードに割り当て（処理ブロック７０４）、プロセスは終了する。

【００４２】

[0062]カレントノードがルートノードに設定されるプロセスが終了した後、ツリー中の全ノードは０又は１という値が割り当てられている。ルートノードは、１個以上の係数がゼロに等しくないならば、値１を保有する。

【００４３】

[0063]図８は、ツリー中のノードを符号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。

【００４４】

[0064]図８を参照すると、プロセスは最初にルートノードを引数として用いて呼び出される。プロセスは、プロセッシングロジックがノードはリーフであるかどうかを判定することによって開始する（処理ブロック８０１）。ノードがリーフであるならば、プロセッシングロジックは、そのノードの対応する係数の値を符号化し（処理ブロック８０２）、プロセスは終了する。種々の符号化スキーム（例えば、ハフマン符号化、算術符号化など）が使用されてもよい。

【００４５】

[0065]ノードがリーフでないならば、プロセッシングロジックはノードの２個の子の値を符号化する（処理ブロック８０３）。符号化はさらに後述されている。その後、プロセッシングロジックは第１の子が１という値を保有するかどうかをテストする（処理ブロック８０４）。第１の子が値１を保有するならば、処理ブロックは、第１の子を引数として用いてノード符号化プロセスを再帰的に実行する（処理ブロック８０６）。プロセッシングロジックは、その後、第２の子の値が１であるかどうかをテストする（処理ブロック８０７）。第１の子が値０を保有するか、又は、第２の子が値１を有するならば、プロセッシングロジックは、第２の子を引数として用いてノード符号化プロセスを再帰的に実行する（処理ブロック８０５）。その後、プロセスは終了する。

【００４６】

[0066]図９は、子ノードの値を符号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。

【００４７】

[0067]図９を参照すると、プロセスは、プロセッシングロジックが第１の子の値を符号化することによって開始する（処理ブロック９０１）。一実施形態では、値はビットストリームに付加される単一ビットとして符号化される。代替的に、値は算術符号化器を使用してさらに圧縮される。

【００４８】

[0068]プロセッシングロジックは、その後、第１の子の値が１であるかどうかを決定する（処理ブロック９０２）。第１の子の値が０に等しいならば、プロセスは終了する。そうでなければ、プロセッシングロジックは第２の子の値を符号化する（処理ブロック９０３）。一実施形態では、値はビットストリームに付加される単一ビットとして符号化される。代替的に、値は算術符号化器を使用してさらに圧縮される。第２の子の値を符号化した後、プロセスは終了する。

【００４９】

[0069]図１０は、子ノードの値を符号化するプロセスの別の実施形態のフローチャートである。これは図９に記載されたプロセスへの代替的な実施形態であることに留意されたい。代替的に、方法の選択はノードに基づいて行われてもよい。選択情報は、メモリに格

納され、ビットストリームに符号化されてもよい。このような符号化は選択された方法を示すためにビットを使用して行われてもよい。代替的に、エンコーダ及びデコーダの両方が、ノード毎にどの方法を使用すべきであるかに関する所定の引数を保有してもよい。

【００５０】

[0070]図１０のプロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよいプロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。

【００５１】

[0071]図１０を参照すると、プロセスは、プロセッシングロジックが２個の子ノードの結合された値を符号化することにより開始する（処理ブロック１００１）。一実施形態では、結合された値は、両方の子の値の間の論理ＡＮＤ演算の結果によって決定される。プロセッシングロジックは、次に、両方の子が１という値を保有するかどうかを判定する（処理ブロック１００３）。コード化された値が１であるならば、プロセスは終了する。そうでなければ、プロセッシングロジックは第１の子の値を符号化し（処理ブロック１００４）、プロセスは終了する。一実施形態では、値を符号化するとき、値はビットストリームにそのままで付加される。代替的に、値は算術符号化器を使用してさらに圧縮されてもよい。

【００５２】

[0072]代替的に、三つ組のアルファベットが子ノードの値を表現するために使用されてもよい。このような場合、単一記号がビットストリームに付加される。

【００５３】

[0073]図１１は、変換デコーダの一実施形態のブロック図である。図１１を参照すると、ビットストリーム１１２０が係数デコーダ１１１０に入力されている。係数デコーダ１１１０は、ビットストリーム１１２０からのビットを量子化された係数１１１１の集合に変換する。変換プロセスはさらに後述されている。逆量子化器１１０１は、量子化された係数１１１１を係数１１０２の集合にスケール変換する。逆変換１１０３は、係数１１０２を再構成画像データ１１０４に変換する。

【００５４】

[0074]図１２は、係数デコーダの一実施形態のブロック図である。図１２を参照すると、エントロピーデコーダ１２０２は、さらに後述されているように、ビットストリーム１２０１からのビットを量子化された係数１２０３の集合に変換する。このため、エントロピーデコーダ１２０２は、メモリ１２１０に格納されているツリー構造データに依存する。エントロピーデコーダ１２０２は、ビットストリーム１２０１からツリー構造データを読み、メモリ１２１０に格納してもよい。

【００５５】

[0075]図１３は、量子化された係数を復号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。

【００５６】

[0076]図１３を参照すると、プロセスは、プロセッシングロジックが集合中の全係数をゼロに設定することによって開始する（処理ブロック１３０１）。係数をゼロに設定した後、プロセッシングロジックはフラグを復号化する（処理ブロック１３０２）。プロセッシングロジックは、その後、全係数がゼロであるかどうかを判定する（処理ブロック１３０３）。全係数がゼロであることを示す第１の値をフラグがとるならば、プロセスは終了する。そうでなければ、プロセッシングロジックは、さらに後述されるように、ツリー中のノード値を復号化し（処理ブロック１３０４）、同様に係数値を復号化する（処理ブロック１３０５）。その後、プロセスは終了する。

【００５７】
[0077]図１４は、ノードを復号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。
【００５８】
[0078]図１４を参照すると、プロセスはルートノードを引数として用いて開始する。プロセッシングロジックはノードがリーフであるかどうかを判定する（処理ブロック１４０１）。ノードがリーフノードであるならば、プロセッシングロジックは係数の値をさらに復号化し（処理ブロック１４０２）、その後にプロセスが終了する。復号化は種々の復号化技術（例えば、ハフマン符号化、算術符号化など）を使用して実行されてもよい。
【００５９】
[0079]ノードがリーフでないならば、プロセッシングロジックはそのノードの２個の子の値を復号化する（処理ブロック１４０３）。復号化はさらに後述されている。その後、プロセッシングロジックは、第１の子が１という値を保有するかどうかをテストする（処理ブロック１４０４）。第１の子が値１を保有するならば、処理ブロックは、第１の子を引数として用いてノード復号化プロセスを再帰的に実行する（処理ブロック１４０６）。プロセッシングロジックは、その後、第２の子が値１を保有するかどうかをテストする（処理ブロック１４０７）。第１の子が値０を保有するか、又は、第２の子が値１を保有するならば、プロセッシングロジックは、第２の子を引数として用いてノード復号化プロセスを再帰的に実行する（処理ブロック１４０５）。その後、プロセスは終了する。
【００６０】
[0080]図１５は、子ノードの値を復号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。
【００６１】
[0081]図１５を参照すると、プロセスは第１の子の値を復号化することによって開始する（処理ブロック１５０１）。復号化プロセスは、ビットストリームから単一ビットを取り出すこと、又は、算術デコーダを用いてビットストリームからバイナリ記号を復号化することを伴ってもよい。プロセッシングロジックは、第１の子の値が１であるかどうかを判定する（処理ブロック１５０２）。復号化された値が０であるならば、プロセスは終了し、そうではなく、値が１であるならば、プロセッシングロジックは、第１の値と同様に、第２の子の値を復号化し（処理ブロック１５０３）、その後、プロセスは終了する。
【００６２】
[0082]図１６は、子ノードの値を復号化するプロセスの代替的な実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。
【００６３】
[0083]図１６を参照すると、プロセスは両方の子の結合された値を復号化して開始する（処理ブロック１６０１）。結合された値は、両方の子が値１を保有するかどうかを示すことができる。プロセッシングロジックは、両方の子が１という値を保有するかどうかを判定する（処理ブロック１６０２）。両方の子が１という値を保有するならば、プロセスは終了し、そうではなく、値が１であるならば、プロセッシングロジックは第１の子の値を復号化し、第２の子の値を反対の値に設定し（処理ブロック１６０３）、その後、プロセスは終了する。

【００６４】
[0084]代替的に、両方の子の値は三つ組のアルファベットを用いて表現される。このような場合、単一記号が子ノードの値を取得するために復号化されてもよい。
【００６５】
[0085]図１８は量子化された係数を復号化するプロセスの一実施形態のフローチャートである。プロセスは、ハードウェア（例えば、回路、専用ロジックなど）、（汎用コンピュータシステム又は専用機械上で動かされるような）ソフトウェア、又は、両方の組み合わせを備えてもよい、プロセッシングロジックによって実行される。プロセッシングロジックはファームウェアを備えてもよい。
【００６６】
[0086]図１８を参照すると、プロセスは、プロセッシングロジックが状態を１に設定し、スタックを空状態に初期化することにより開始する（処理ブロック１８０１）。その後、プロセッシングロジックはメモリからノードタイプを取り出す（処理ブロック１８０２）。プロセッシングロジックは、その後、ノードがリーフノードであるかどうかを判定する（処理ブロック１８０３）。ノードタイプがリーフノードであることを示すならば、処理は処理ブロック１８２０へ移り、そうでなければ、処理は処理ブロック１８１０へ移る。
【００６７】
[0087]処理ブロック１８２０において、プロセッシングロジックはメモリからリーフインデックスＩを取り出し、状態が値０を有するかどうかをテストする（処理ブロック１８２１）。状態が値０を有するならば、処理は処理ブロック１８２３へ移り、プロセッシングロジックはインデックスＩをもつ係数に値０を割り当て、その後、処理は処理ブロック１８２４へ移る。状態が値１を有するならば、処理は処理ブロック１８２２へ移り、プロセッシングロジックはビットストリームから係数値を復号化し、インデックスＩをもつ係数に復号化された係数値を割り当て、その後、処理は処理ブロック１８２４へ移る。
【００６８】
[0088]処理ブロック１８２４において、プロセッシングロジックは、スタックが空であるかどうかを判定する。スタックが空であるならば、プロセスは終了する。スタックが空でなければ、処理は処理ブロック１８２５へ移り、プロセッシングロジックはスタックから値をポップし、状態をその値に設定する。その後、処理は処理ブロック１８０２へ移る。
【００６９】
[0089]ノードがリーフノードでないならば（処理ブロック１８０３）、プロセッシングロジックは状態が１に等しいかどうかを判定する（処理ブロック１８１０）。そうでなければ（すなわち、値が０であるならば）、プロセッシングロジックは２個のノードの値をゼロに設定し、処理ブロック１８１４へ移る。状態が１に等しければ、プロセッシングロジックはビットストリームから２個のノード値を復号化する（処理ブロック１８１２）。ノード値を復号化する方法の例は、図１５及び１６に記載されている。２個のノード値を復号化した後、プロセッシングロジックは状態を第１のノード値に設定し（処理ブロック１８１３）、処理ブロック１８１４へ移る。処理ブロック１８１４において、プロセッシングロジックは第２のノード値をスタックへプッシュする。その後、処理は処理ブロック１８０２へ移る。
【００７０】
[0090]図１９は、係数デコーダの一実施形態のブロック図である。図１９を参照すると、係数デコーダは状態レジスタ１９２０を含む。一実施形態では、状態レジスタ１９２０は最初に値１に設定される。エントロピーデコーダ１９０２は、ノードタイプメモリ１９２３からノードタイプを取り出す。ノードタイプメモリ１９２３は、図４に記載されているように、ツリーの構造に関するデータを収容する。ノードタイプがリーフノードを示し、状態レジスタの値が１であるならば、エントロピーデコーダ１９０２はビットストリームから量子化された係数を復号化し、それを選択器１９０３へ送信する。選択器１９０３

は、量子化された係数を選択し、それを係数メモリ１９０４へ送信する。係数メモリ１９０４へのインデックスは、リーフインデックスメモリ１９２１から取り出されたリーフインデックスによって決定される。そうではなく、状態レジスタ１９２０が０であるならば、選択器１９０３は値０を選択し、それを係数メモリ１９０４へ送信する。量子化された係数が量子化された係数メモリ１９０４に格納された後、値がスタック１９２２からポップされ、状態レジスタ１９２０に割り当てられる。そうでなく、ノードタイプがリーフノードを示さないならば、係数デコーダは以下の通り動作する。状態レジスタ１９２０が１であるならば、エントロピーデコーダ１９０２は、例えば、図１５及び１６に記載されているように、ビットストリーム１９０１から２個のノード値を復号化する。第１のノード値はエントロピーデコーダ１９０２によって状態レジスタ１９２０に割り当てられ、第２のノード値はエントロピーデコーダ１９０２によってスタック１９２２へプッシュされる。そうではなく、状態レジスタ１９２０が０であるならば、状態レジスタ１９２０の内容がスタックへプッシュされる。

【００７１】

[0091]一実施形態では、デコーダは、量子化された係数メモリ内の全ての量子化された係数に値が割り当てられるまで、この方式で動作する。量子化された係数メモリの内容は、その後、量子化された係数の順序付き集合として出力される。

【００７２】

[0092]上述のように、本明細書に記載された技術は任意のデータのベクトルに適用されてもよい。例えば、これらの技術は、前方向動きベクトルに対応する水平移動及び垂直移動と後方向動きベクトルに対応する水平移動及び垂直移動とを有するデータのブロックを含む次元４のベクトルをコード化するためにｂフレームをビデオコード化するときに使用されてもよい。この場合、本明細書に記載された技術は、ゼロでない方を示すために使用されてもよい。

【００７３】

（コードの例）

[0093]以下のサンプルＣコードは、本発明に記載されているような係数を復号化するために使用できる。ツリーの深さ優先トラバース及び幅優先トラバースの例が提供されている。

【数１】

```
// sample decoder using depth –first traversal
#define N 16 // example with 16 coefficients
int nodes_types; // contains values of node types
int leaf_indices [N]; // contains indices of leaves

void decode_coefficients(int coefficients[N] {
   int node, state, stack, index, val , first, second ;

   index = 0;
   state =  entropy_decod_root_value ( ) ;
   stack = 0; // any value may be used in this example
   for (node=0; node <2*N+1; node++) {
      if ((node_types>>nodes) & 1) {
         if (state == 0)
            val = 0;
         else
            val = entropy_decode_coefficient ( ) ;
         coefficients[leaf_indices[index++]] = val;
         state = stack & 1;
         stack = stack >> 1;
      }
      else {
         stack = stack << 1;
         if (state == 1) {
            entropy_decode_node_values(&first, &second);
            state = first;
            stack = stack | second;
         }
      }
   }
}
```

【数２】

```
// sample decoder using breadth –first traversal
// node_types and leaf_indices need to be changed accordingly
#define N 16 // example with 16 coefficients
int node_types; // contains values of node types
int leaf_indices[N]; // contains indices of leaves

void decode coefficients(int coefficients[N] ) {
   int node, queue, index, val, first, second, tail;

   index = 0;
   tail = 0;
   state = entropy_decode_root_value( ) ;
   queue = state << tail++;
   for (node=0; node <2*N+1; node++) {
      if ( (node_types >>node) & 1) {
            if ((queue>>node) & 1)
               val = entropy_decode_coefficient ( );
            else
               val = 0;
            coefficients[leaf_indices[index++]] = val;
      }
      else {
         if ((queue>>node) & 1) {
            entropy_decode_node_values(&first, &second);
            queue = queue | first << tail++;
            queue = queue | second << tail++;
         }
      }
   }
}
```

【００７４】

[0094]適切なツリー構造データは、係数ブロックの集合から導出できる。以下のサンプルＣコードは、係数の各組み合わせが非ゼロであることを頻度を用いて示す統計量の集合からツリー構造データを構築する。

【数３】

```
#include <stdio.h>
#include <math.h>

int param[47]; / / contains tree structure data
int p0, p1;
double *min_entropy;
int *best_submask;
int *best_hist[3];

/ / compute entropy of a histogram
double entrop (int *hist, int n) {
   int i, sum;
   double ent;

   ent = 0.0;
   sum = 0;

   for (i=0; i<n; i++)
      if (hist [i] != 0) {
         sum += hist[i];
         ent -= hist[i] * log(hist[i]);
      }

   if (sum != 0)
      ent += sum*log(sum);
```

【数４】

```
    ent /= log(2.0);

    return ent;
}

void split(int mask, int *stats) {
    int hist[3];
    int i,j,m,ncoeff;
    double e;

// count number of coefficients in set defined by mask
ncoeff = 0;
for (i=0; i<16; i++)
    ncoeff += (mask >> i) & 1;

if (ncoeff <= 1) { // if only one, cannot further split
    min_entropy[mask] = 0.0;
    best_submask[mask] = mask;
}
else {
    min_entropy[mask] = le38;

    for (m=1; m <= (mask>>1); m++) {
        if ((m & mask) != m)
            continue;

        e = min_entropy[m] = min_entropy[m^mask];

        if (e >= min_entropy[mask])
            continue;
```

【数５】

```
        for (i=0; i<3; i++)
            hist[i] = 0;

        for (i=0; i<65536; i++)
            if ((i & mask) != 0) {
                j = (i & m) != 0;
                j += (i & (m ^ mask) ) == 0;

                / / value of j:
                / / 0: first child = 0, second child = 1
                / / 1: first child = 1, second child = 1
                / / 2: first child = 1, second child = 0

                hist[j] += stats[i];
            }
            e += entrop(hist, 3);

            if (e < min_entropy[mask]) {
                min_entropy[mask] = e;
                best_submask[mask] = m;
                for (j=0; j<3; j++)
                    best_hist[j] [mask] = hist[j];
            }
        }
    }
    printf ("%4x %f      \ r", mask min_entropy);
    fflush(stdout);
}

void printtree (int mask) {
    int i;
```

【数６】

```
        int c0, c1;

        if (best_submask[mask] != mask) {
            printf("X ");
            param[p0++] = 1;

            printtree (best_submask[mask]);
            printtree (best_submask[mask] ^ mask);
        }
        else {
            i = 0;
            while (mask != (1 <<i) )
                i++;

            param[p0++] = 0;
            param[p1++] = i;
            printf ("%d ", i);
        }
    }

    main ( ) {
        FILE *f1;
        FILE *f4;
        int i, j;
        int stats [65536];
        char filename [256];
        int cnt;
        int step, type;
        int z2;
        int intra;
```

【数７】

```
min_entropy = new double [65536];
best_submask = new int[65536];
for (i=0; i<3; i++)
    best_hist[i] = new int[65536];

f1 = fopen ("freq.txt", "r");

for (i=0; i<65536; i++) {
    fscanf (f1, "%d", &cnt );
    stats[i] += cnt;
}

fclose (f1);
stats[0] = 0;

printf {"%f\n", entrop(stats, 65536) );

step = 1;

for (i=step p; i<0x10000; i+=step)
    split (i, stats);

printf ("Minimum entropy: %f \n", min_entropy[0x10000 -step]);

p0 = 0;
p1 = 31;

printtree (0x10000 -step);
printf ("\n");

f4 = fopen ("tree.txt", "w");
```

【数８】

```
        for (i=0; i<47; i++)
            fprintf (f4, "%d ", param[i]);
        frprintf (f4, "\n");
        fclose (f4);
    }
```

【００７５】

（典型的なコンピュータシステム）

[0095]図２０は、本明細書に記載された１個以上の演算を実行できる典型的なコンピュータシステムのブロック図である。コンピュータシステム２０００は、典型的なクライアント又はサーバーコンピュータシステムを備えてもよい。コンピュータシステムに関して記述されたコンポーネントは、ハンドヘルド機器又はモバイル機器（例えば、携帯電話機）の一部であってもよい。

【００７６】

[0096]図２０を参照すると、コンピュータシステム２０００は、情報を通信する通信メカニズム又はバス２０１１と、バス２０１１と接続され情報を処理するプロセッサ２０１２とを備える。プロセッサ２０１２は、マイクロプロセッサを含むが、例えば、Ｐｅｎｔｉｕｍ（登録商標）プロセッサなどのようなマイクロプロセッサに限定されない。

【００７７】

[0097]システム２０００は、バス２０１１に接続され、情報とプロセッサ２０１２によって実行されるべき命令とを格納する、ランダムアクセスメモリ（ＲＡＭ）又はその他のダイナミック記憶装置２００４（メインメモリとして参照される）を、さらに備える。メインメモリ２００４はまた、プロセッサ２０１２による命令の実行中に、一時変数又はその他の中間情報を格納するためにも使用できる。

【００７８】

[0098]コンピュータシステム２０００は、バス２０１１に接続され、スタティック情報及びプロセッサ２０１２のための命令を格納するリードオンリーメモリ（ＲＯＭ）及び／又はその他のスタティック記憶装置２００６と、磁気ディスク又は光ディスク、及び、それに対応するディスクドライブのようなデータ記憶装置２００７とを、さらに備える。データ記憶装置２００７は、バス２０１１に接続され、情報及び命令を格納する。

【００７９】

[0099]コンピュータシステム２０００は、バス２０１１に接続され、情報をコンピュータユーザへ表示する陰極線管（ＣＲＴ）又は液晶ディスプレイ（ＬＣＤ）のようなディスプレイ機器２０２１に、さらに接続されてもよい。英数字キー及びその他のキーを含む英数字入力機器２０２２は、同様にバス２０１１に接続され、情報及びコマンド選択をプロセッサ２０１２へ通信できる。付加的なユーザ入力機器は、カーソル制御機器２０２３である。このカーソル制御機器２０２３は、例えばマウス、トラックボール、トラックパッド、スタイラス、又は、カーソル方向キーである。カーソル制御機器２０２３は、方向情報及びコマンド選択をプロセッサ２０１２へ通信し、ディスプレイ２０２１上のカーソル移動を制御するために、バス２０１１に接続されている。

【００８０】

[00100]バス２０１１に接続できる別の機器は、ハードコピー機器２０２４である。ハードコピー機器２０２４は、紙、フィルム、又は、類似したタイプの媒体のような媒体上に情報を残すために使用できる。バス２０１１に接続できる別の機器は、電話機又はハンドヘルドパーム機器へ通信するための有線／無線通信機能部２０２５である。

【００８１】

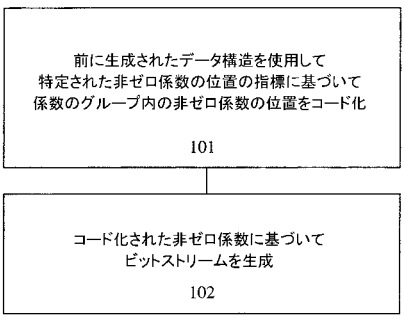
[00101]システム２０００のコンポーネント、及び、関連したハードウェアの何れか又

は全部が本発明において使用できることに留意されたい。しかし、コンピュータシステムのその他の構成は、機器の一部又は全部を含んでもよいことが認められるであろう。

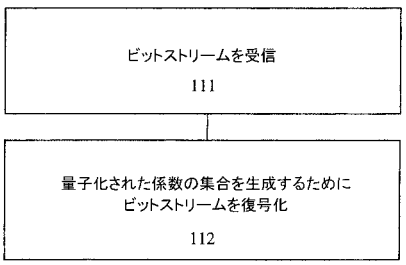
【００８２】

[00102]以上の説明を読んだ後に、本発明の多数の代替及び変形が当業者に明白になることは確実であるが、説明のために図示され記載された特定の実施形態が制限的であるとみなされることは決して意図されていないことが理解されるべきである。したがって、種々の実施形態の詳細への言及は、本発明に不可欠であると考えられる構成要件だけをそれ自体に列挙する請求項の範囲を制限することを意図していない。

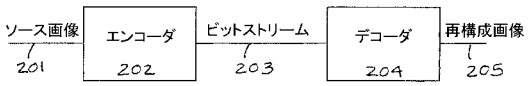
【図１Ａ】



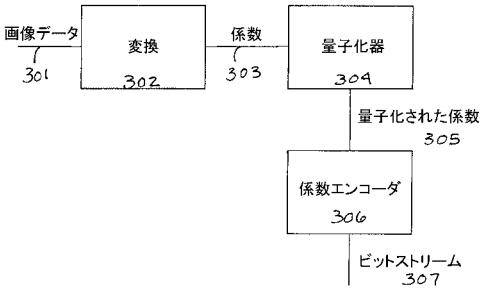
【図１Ｂ】



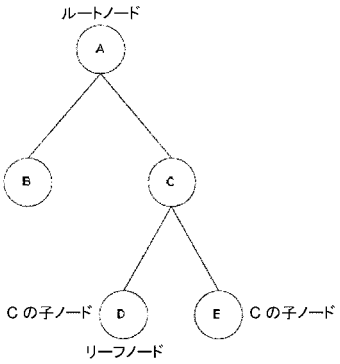
【図２】


ソース画像
201
エンコーダ
202
ビットストリーム
203
デコーダ
204
再構成画像
205

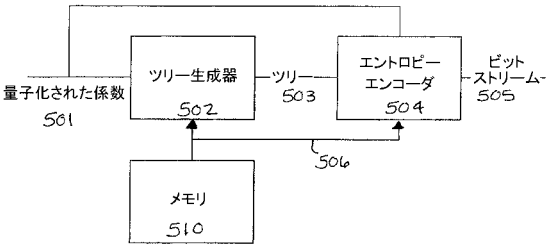
【図３】


画像データ
301
変換
302
係数
303
量子化器
304
量子化された係数
305
係数エンコーダ
306
ビットストリーム
307

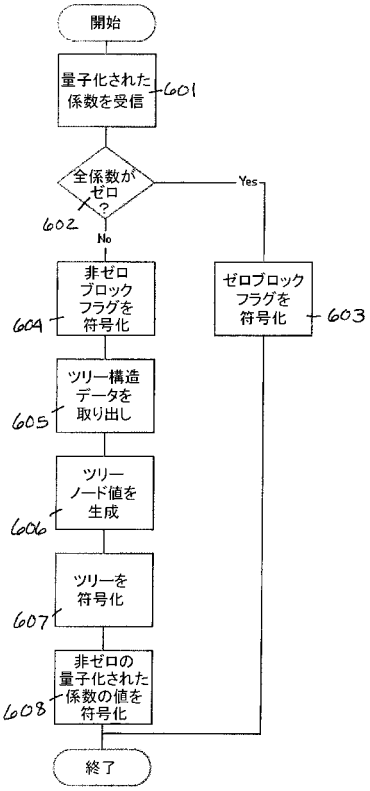
【図４】


ルートノード
A
B
C
Cの子ノード
D
E
Cの子ノード
リーフノード

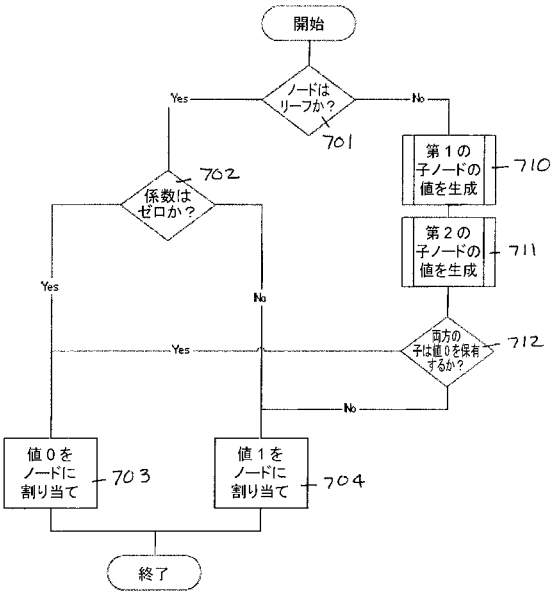
【図５】


量子化された係数
501
ツリー生成器
502
ツリー
503
エントロピー
エンコーダ
504
ビット
ストリーム
505
506
メモリ
510

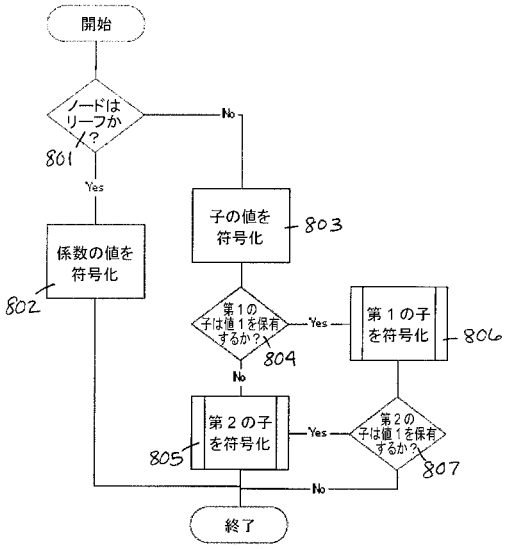
【図６】


開始
量子化された係数を受信
601
全係数がゼロ？
602
Yes
No
非ゼロブロックフラグを符号化
604
ゼロブロックフラグを符号化
603
ツリー構造データを取り出し
605
ツリーノード値を生成
606
ツリーを符号化
607
非ゼロの量子化された係数の値を符号化
608
終了

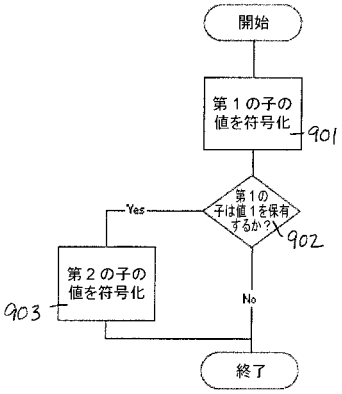
【図７】


開始
ノードはリーフか？
701
Yes
No
係数はゼロか？
702
第１の子ノードの値を生成
710
第２の子ノードの値を生成
711
両方の子は値０を保有するか？
712
値０をノードに割り当て
703
値１をノードに割り当て
704
終了

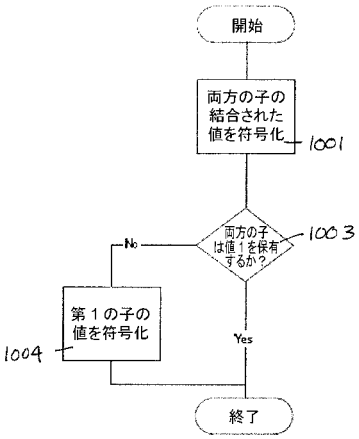
【図８】


開始
ノードはリーフか？
801
Yes
No
係数の値を符号化
802
子の値を符号化
803
第１の子は値１を保有するか？
804
Yes
No
第１の子を符号化
806
第２の子を符号化
805
第２の子は値１を保有するか？
807
Yes
No
終了

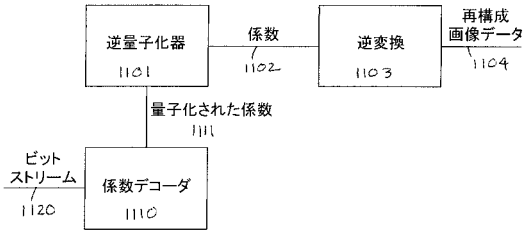
【図９】


開始
第１の子の値を符号化
901
第１の子は値１を保有するか？
902
Yes
No
第２の子の値を符号化
903
終了

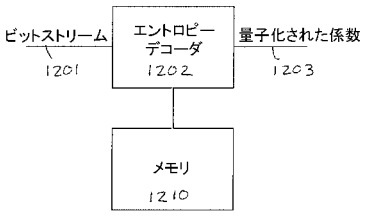
【図１０】



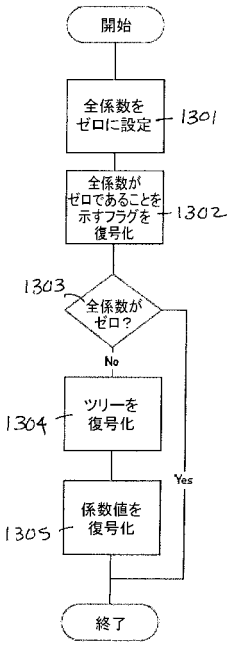
【図１１】



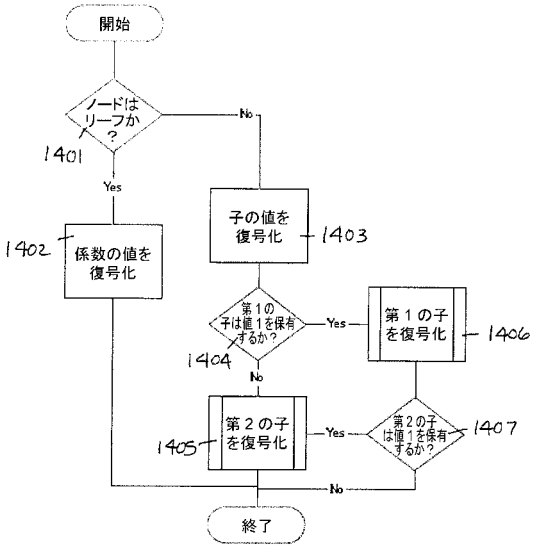
【図１２】



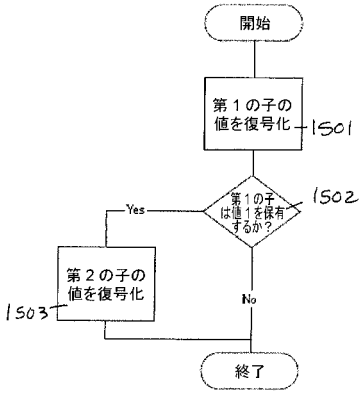
【図１３】



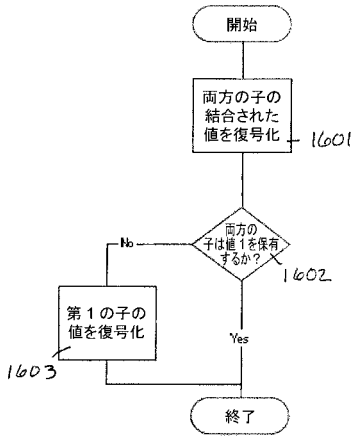
【図１４】



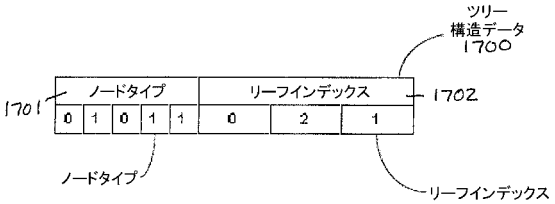
【図１５】



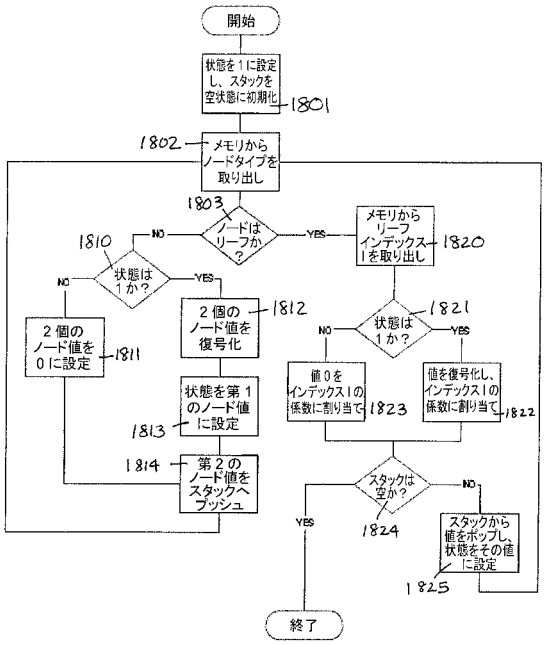
【図１６】



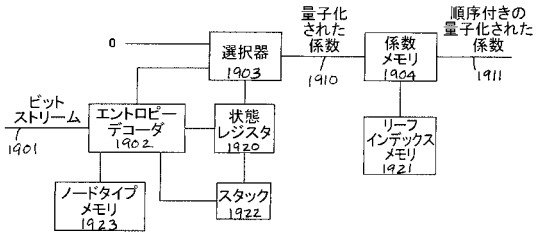
【図１７】



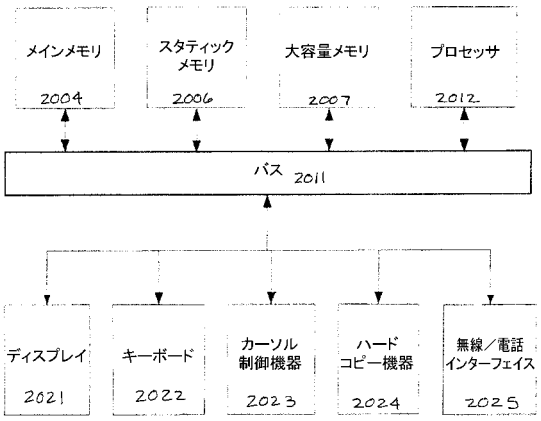
【図１８】


開始
状態を１に設定し、スタックを空状態に初期化
1801
1802
メモリからノードタイプを取り出し
1803
ノードはリーフか？
YES
NO
メモリからリーフインデックスIを取り出し
1820
1810
状態は１か？
YES
NO
2個のノード値を復号化
1812
2個のノード値を０に設定
1811
状態は１か？
1821
NO
YES
値０をインデックスIの係数に割り当て
1823
値を復号化し、インデックスIの係数に割り当て
1822
状態を第１のノード値に設定
1813
1814
第２のノード値をスタックへプッシュ
スタックは空か？
1824
NO
YES
スタックから値をポップし、状態をその値に設定
1825
終了

【図１９】


0
選択器
1903
量子化された係数
1910
係数メモリ
1904
順序付きの量子化された係数
1911
ビットストリーム
1901
エントロピーデコーダ
1902
状態レジスタ
1920
リーフインデックスメモリ
1921
ノードタイプメモリ
1923
スタック
1922

【図２０】


メインメモリ
2004
スタティックメモリ
2006
大容量メモリ
2007
プロセッサ
2012
バス
2011
ディスプレイ
2021
キーボード
2022
カーソル制御機器
2023
ハードコピー機器
2024
無線／電話インターフェイス
2025

【手続補正書】
【提出日】平成24年2月17日(2012.2.17)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
画像から生成された複数のデータ要素からなるベクトルを受信するステップと、
ツリーの複数のノードの値を符号化するステップであって、前記ツリーは前記ベクトル中のどのデータ要素が非ゼロであるかを識別する、ステップと、
前記ベクトルの複数の非ゼロデータ要素の値を符号化するステップであって、前記非ゼロデータ要素が前記ツリーによって識別される、ステップと、
前記ツリーのノードの符号化された値と前記非ゼロデータ要素の符号化された値とを含むビットストリームを出力するステップと、
を備える方法。
【請求項２】
画像の圧縮表現であるビットストリームを受信するステップと、
前記ビットストリームを復号化し、復号化された複数のデータ要素からなるベクトルを、ツリーデータ構造によって識別される、前記復号化されたデータ要素のどれが非ゼロであるかの指標に基づいて作成するステップであって、前記ツリーデータ構造はツリーを表現し、前記復号された複数のデータ要素からなるベクトルは前記画像の再構成画像であり、前記ツリーは複数のリーフノードを有する、ステップと、
を備える方法。

フロントページの続き

Ｆターム(参考) 5C159 MA23 MC11 MC31 ME02 ME11 TA59 TC04 TC06 TC41 UA02
UA05
5C178 BC05 BC56 BC74 BC91 CC68 CC69 FC02
5J064 AA01 AA02 BA09 BA13 BD02 BD03

【外国語明細書】
2012135017000001.pdf