(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2010-141922
(P2010-141922A)
(43)公開日　平成22年6月24日(2010.6.24)

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

審査請求 有　請求項の数 7　ＯＬ　外国語出願　（全 42 頁）

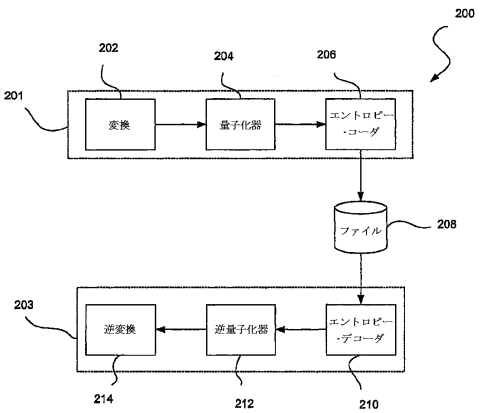
(21)出願番号　特願2010-36657 (P2010-36657)
(22)出願日　平成22年2月22日 (2010.2.22)
(62)分割の表示　特願2003-586705 (P2003-586705)の分割
原出願日　平成15年4月17日 (2003.4.17)
(31)優先権主張番号　60/374,061
(32)優先日　平成14年4月19日 (2002.4.19)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　60/373,974
(32)優先日　平成14年4月19日 (2002.4.19)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　60/373,966
(32)優先日　平成14年4月19日 (2002.4.19)
(33)優先権主張国　米国 (US)

(71)出願人　507091738
ドロップレット　テクノロジー　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　９４０２５　メンロ　パーク　アダムス　ドライヴ　１６００　スイート　２１６
(74)代理人　100147485
弁理士　杉村　憲司
(74)代理人　100143568
弁理士　英　貢
(72)発明者　クラジミール　コラロフ
アメリカ合衆国　カリフォルニア州　９４０２５　メンロ　パーク　アヴィー　アヴェニュー　２０５０

最終頁に続く

(54)【発明の名称】ウェーブレット変換システム、方法、及びコンピュータプログラム製品

(57)【要約】
【課題】データを圧縮するためのシステム、方法、及びコンピュータプログラム製品を提供する。
【解決手段】最初に、内挿補間公式を受け取る。こうした内挿補間公式を利用してデータを圧縮する。使用中には、この内挿補間公式が、入手不可能なデータ値を少なくとも１つ必要とするか否かを判定する。必要とする場合には、外挿補間演算を実行して、必要とする入手不可能なデータ値を生成する。
【選択図】図２



【特許請求の範囲】
【請求項１】
内挿補間公式を受け取るステップと；
前記内挿補間公式が、入手不可能なデータ値を少なくとも１つ必要とするか否かを判定するステップと；
外挿補間演算を実行して、前記必要とする入手不可能なデータ値を生成するステップとを具えて；
前記内挿補間公式を利用してデータを圧縮することを特徴とするデータ圧縮方法。
【請求項２】
前記内挿補間公式がウェーブレットフィルタの構成要素であることを特徴とする請求項１に記載の方法。
【請求項３】
さらに、複数のデータ値を複数のスパンにセグメント分割するステップを具えていることを特徴とする請求項１に記載の方法。
【請求項４】
さらに、前記複数のスパン中の１つのスパン内のデータ値のみを利用することによって、前記内挿補間公式に関係する演算量を低減するステップを具えていることを特徴とする請求項３に記載の方法。
【請求項５】
さらに、前記ウェーブレットフィルタを多相フィルタに置き換えるステップを具えていることを特徴とする請求項２に記載の方法。
【請求項６】
さらに、前記データ値を量子化するステップを具えていることを特徴とする請求項１に記載の方法。
【請求項７】
さらに、前記データ値の数量を低減することによって、エントロピー符号化に関連する演算量を低減するステップを具えていることを特徴とする請求項６に記載の方法。
【請求項８】
前記データ値に関係する量子化演算中に、前記データ値の数量を低減することを特徴とする請求項７に記載の方法。
【請求項９】
パイルを用いて、前記データ値の数量を低減することを特徴とする請求項７に記載の方法。
【請求項１０】
さらに、複数の前記データ値を所定のデータ範囲に再構成することに関連する演算量を低減するステップを具えていることを特徴とする請求項１に記載の方法。
【請求項１１】
単一のクリップ演算のみを実行することによって、前記演算量を低減することを特徴とする請求項１０に記載の方法。
【請求項１２】
前記ウェーブレットフィルタが、次式：
【数２１】

$$Y_{2n+1}=\left(X_{2n+1}+1/2\right)-\left\lfloor\frac{\left(X_{2n}+1/2\right)+\left(X_{2n+2}+1/2\right)}{2}\right\rfloor$$

の内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１３】
前記ウェーブレットフィルタが、次式：

$$Y_{2N+1}=(X_{2N+1}+1/2)-(X_{2N}+1/2)$$

の内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１４】
前記ウェーブレットフィルタが、次式：
【数２２】

$$(Y_{2n}+1/2)=(X_{2n}+1/2)+\left\lfloor\frac{Y_{2n-1}+Y_{2n+1}}{4}\right\rfloor$$

を含めた内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１５】
前記ウェーブレットフィルタが、次式：
【数２３】

$$(Y_0+1/2)=(X_0+1/2)+\left\lfloor\frac{Y_1}{2}\right\rfloor$$

を含めた内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１６】
前記ウェーブレットフィルタが、次式：
【数２４】

$$(X_{2n}+1/2)=(Y_{2n}+1/2)-\left\lfloor\frac{Y_{2n-1}+Y_{2n+1}}{4}\right\rfloor$$

を含めた内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１７】
前記ウェーブレットフィルタが、次式：
【数２５】

$$(X_0+1/2)=(Y_0+1/2)-\left\lfloor\frac{Y_1}{2}\right\rfloor$$

を含めた内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１８】
前記ウェーブレットフィルタが、次式：
【数２６】

$$(X_{2n+1}+1/2)=Y_{2n+1}+\left\lfloor\frac{(X_{2n}+1/2)+(X_{2n+2}+1/2)}{2}\right\rfloor$$

を含めた内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項１９】
前記ウェーブレットフィルタが、次式：

$$(X_{2N+1}+1/2)=Y_{2N+1}+(X_{2N}+1/2)$$

を含めた内挿補間公式を含むことを特徴とする請求項２に記載の方法。
【請求項２０】
内挿補間公式を受け取るためのコンピュータコードと；
前記内挿補間公式が、入手不可能なデータ値を少なくとも１つ必要とするか否かを判定するためのコンピュータコードと；
外挿補間演算を実行して、前記必要とする入手不可能なデータ値を生成するためのコン

ピュータコードとを具えて；
前記内挿補間公式を利用してデータを圧縮することを特徴とするデータ圧縮用コンピュータプログラム。
【請求項２１】
ウェーブレット方式を分析して、ウェーブレットフィルタが近似する局所的な導関数を決定する論理回路と；
ウェーブレットフィルタの特性及び利用可能なサンプル数にもとづいて、外挿補間に使用する多項式の次数を選定する論理回路と；
前記選定した多項式の次数を用いて、ウェーブレットフィルタ毎の外挿補間公式を導出する論理回路と；
前記外挿補間公式を、各場合において利用可能なサンプルと共に利用して、特定エッジのウェーブレットケースを導出する論理回路と
を具えていることを特徴とするデータ処理システム。
【請求項２２】
単一装置でデータを受け取るステップと；
前記単一装置を利用して前記データを符号化して、第１フォーマットの第１圧縮データを生成するステップと；
前記第１圧縮データを、前記単一装置を利用してコード変換して、第２フォーマットの第２圧縮データを生成するステップと
を具えていることを特徴とするデータ圧縮方法。
【請求項２３】
前記符号化をリアルタイムで行うことを特徴とする請求項２２に記載の方法。
【請求項２４】
前記コード変換をオフラインで行うことを特徴とする請求項２２に記載の方法。
【請求項２５】
前記第１圧縮データをコード変換して、前記単一装置に結合した通信ネットワークの容量に整合させるべく適応させた第２フォーマットの第２圧縮データを生成することを特徴とする請求項２２に記載の方法。
【請求項２６】
前記符号化を、第１エンコーダを利用して実行することを特徴とする請求項２２に記載の方法。
【請求項２７】
前記コード変換を、デコーダ及び第２エンコーダを利用して実行することを特徴とする請求項２６に記載の方法。
【請求項２８】
前記第１フォーマットがウェーブレット・フォーマットを含むことを特徴とする請求項２２に記載の方法。
【請求項２９】
前記第２フォーマットが、ＤＣＴベースのフォーマットを含むことを特徴とする請求項２２に記載の方法。
【請求項３０】
前記第２フォーマットが、ＭＰＥＧフォーマットを含むことを特徴とする請求項２９に記載の方法。
【請求項３１】
単一デバイス上に実現され、データを符号化して第１フォーマットの第１圧縮データを生成するエンコーダと；
前記エンコーダと同じ単一デバイス上に実現され、前記第１圧縮データをコード変換して第２フォーマットの第２圧縮データを生成するトランスコーダと
を具えていることを特徴とするデータ圧縮用単一デバイス。
【請求項３２】

前記符号化をリアルタイムで行うことを特徴とする請求項３１に記載の単一デバイス。
【請求項３３】
前記コード変換をオフラインで行うことを特徴とする請求項３１に記載の単一デバイス。
【請求項３４】
前記第１圧縮データをコード変換して、前記単一装置に結合した通信ネットワークの容量に整合させるべく適応させた第２フォーマットの第２圧縮データを生成することを特徴とする請求項３１に記載の単一デバイス。
【請求項３５】
前記符号化を、第１エンコーダを利用して実行することを特徴とする請求項３１に記載の単一デバイス。
【請求項３６】
前記コード変換を、デコーダ及び第２エンコーダを利用して実行することを特徴とする請求項３５に記載の単一デバイス。
【請求項３７】
前記第１フォーマットがウェーブレット・フォーマットを含むことを特徴とする請求項３１に記載の単一デバイス。
【請求項３８】
前記第２フォーマットが、ＤＣＴベースのフォーマットを含むことを特徴とする請求項３１に記載の単一デバイス。
【請求項３９】
前記第２フォーマットが、ＭＰＥＧフォーマットを含むことを特徴とする請求項３８に記載の単一デバイス。
【請求項４０】
単一集積回路上の複数のエンコーダを利用してデータを圧縮する方法であって、この方法が、
前記単一集積回路でデータを受け取るステップと；
前記単一集積回路に内蔵された前記複数のエンコーダを利用して、前記データを符号化するステップと
を具えていることを特徴とするデータ圧縮方法。
【請求項４１】
前記単一集積回路上の複数のチャンネルを利用して、前記データを符号化することを特徴とする請求項４０に記載の方法。
【請求項４２】
前記データを、ウェーブレットベースのフォーマットに変換することを特徴とする請求項４０に記載の方法。
【請求項４３】
単一集積回路上に実現され、第１組のデータを符号化する第１エンコーダと；
前記第１エンコーダと同じ単一集積回路上に実現され、第２組のデータを符号化する第２エンコーダと
を具えていることを特徴とする単一集積回路。
【請求項４４】
前記単一集積回路上の複数のチャンネルを利用して、前記データを符号化することを特徴とする請求項４３に記載の単一集積回路。
【請求項４５】
前記データを、ウェーブレットベースのフォーマットに符号化することを特徴とする請求項４３に記載の単一集積回路。
【請求項４６】
単一モジュールを利用して光子を受け取るステップと；
前記単一モジュールを利用して、前記光子を表現する圧縮データを出力するステップと

を具えていることを特徴とするデータ圧縮方法。
【請求項４７】
前記圧縮データを、ウェーブレットベースのフォーマットに符号化することを特徴とする請求項４６に記載の方法。
【請求項４８】
前記符号化に関連する変換操作を、アナログで実行することを特徴とする請求項４７に記載の方法。
【請求項４９】
前記単一モジュールが撮像素子を含むことを特徴とする請求項４６に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
（発明の分野）
本発明はデータ圧縮に関するものであり、特に、ウェーブレットを利用したデータ圧縮に関するものである。
【背景技術】
【０００２】
（発明の背景）
ビデオ「コーデック」（圧縮／伸長器）は、画質、プロセッサについての要求（例えば、コスト／電力消費）、及び圧縮比（即ち生成されるデータレート）を均衡させることによってデータ通信ストリームに要求されるデータレートを低減するために用いられる。現在利用可能な圧縮方法は、異なる範囲のトレードオフ（得失）をもたらし、そして、複数のコーデックのプロファイル（形）を生み出し、各プロファイルは、特定用途における必要事項を満たすべく最適化されている。
【０００３】
図１に、従来技術の、現在利用可能な種々の圧縮アルゴリズム間のトレードオフの例１００を示す。図に示すように、こうした圧縮アルゴリズムは、ウェーブレットベースのコーデック１０２、及び種々のＭＰＥＧビデオ配信プロファイルを含むＤＣＴ（Discrete Cosine Transform：離散コサイン変換）ベースのコーデック１０４を含む。
【０００４】
２Ｄ及び３Ｄのウェーブレットは、ＤＣＴベースのコーデック・アルゴリズムの現在の代替法である。ウェーブレットは、その良好な画質及び自在（フレキシブル）な圧縮比によって、大いに注目されてきて、ウェーブレットアルゴリズムをＪＰＥＧ－２０００静止画規格に採用することを、ＪＰＥＧ委員会に促してきた。不都合なことに、大部分のウェーブレットの実現は非常に複雑なアルゴリズムを用い、代替法であるＤＣＴに比べて膨大な処理パワー（力）を必要とする。これに加えて、ウェーブレットは時間圧縮にとって独特の挑戦をもたらし、３Ｄウェーブレットを特に困難にしている。
【０００５】
これらの理由により、ウェーブレットは、ＭＰＥＧのように大量に用いられる工業規格のコーデックとコストで競り勝つ利点をもたらすことが決してなく、従って、すき間的（ニッチ）な用途に採用されるに過ぎなかった。従って、３つの大きな市場部分に焦点を合わせて低電力及び低コスト用に最適化した、商業的に生き残れる３Ｄウェーブレットを実現する必要がある。
【０００６】
例えば、小型ビデオカメラがより広く用いられ、ビデオカメラの信号をディジタルで扱うことの利点は明白である。例えば、一部の国におけるセルラー（移動）電話市場の最も急速な成長は、画像及びビデオクリップの機能を有する電話機によるものである。大部分のディジタル・スチル（静止画）カメラは、ビデオクリップ機能を有する。移動無線電話機（ハンドセット）の市場では、これらの静止画及び短いビデオクリップの伝送は、装置のバッテリの能力をより一層必要とする。既存のビデオ符号化規格及びディジタル信号プ

ロセッサは、バッテリにより一層の負担をかける。

【０００７】

他の新たな用途は、視聴者が、生のＴＶ（テレビ）放送の一時停止及びタイムシフト（時間をずらす）プログラミングができるパーソナル・ビデオレコーダ（ＰＶＲ：個人用ディジタル録画編集機）である。これらの装置は、ディジタル・ハードディスク記憶装置を用いてビデオを記録し、ケーブルからのアナログビデオのビデオ圧縮を必要とする。こうした特徴を、ピクチャーイン－ピクチャ（子画面、二画面）、視聴しながらの記録として提供するために、これらの装置は複数のビデオ圧縮エンコーダ（符号化器）を必要とする。

【０００８】

他の成長しつつある用途領域は、監視及びセキュリティ（保安）ビデオ用のディジタル・ビデオレコーダ（ＤＶＲ）である。ここでも、記憶すべき入力ビデオのチャンネル毎に圧縮符号化を必要とする。便利で柔軟性のある（フレキシブルな）ネットワーク伝送アーキテクチャを利用するためには、カメラにおいてビデオを圧縮しなければならない。より以前の多重化レコーダ・アーキテクチャでも、複数のチャンネル圧縮エンコーダを必要とする。

【０００９】

もちろん、低電力及び低コスト用に最適化した３Ｄウェーブレットの商業的に生き残れる実現の恩恵を享受する、膨大な数の他の市場が存在する。

【００１０】

画像は、二次元正方形上の関数として考えれば、大部分の点が平滑であり一部の比較的孤立した特異点及び特異な線（縁、エッジ）を伴う多項式として良好にモデル化されることは、経験が教える所である。ビデオクリップも同様に、三次元領域でモデル化される。大部分の画像及びビデオについては、線形多項式モデルＲＭＳ（Root Mean Square：二乗平均の平方根）からの残差が5%の付近にあり、二次多項式モデルについては2%の付近にある。

【００１１】

こうした関数（画像及びビデオ）近似するために一般に用いられる方式は、次のステップを具えている：

１）　この関数を可逆的に変換して、変換した係数を「サブバンド（副帯域）」に分割可能にするステップ。

２）　「ローパス（低域通過）」サブバンドを除いたすべてのサブバンドを量子化する（即ち精度を低下させる）ステップ。

３）　量子化した係数に逆変換を適用して、これにより元の関数の近似を再構成するステップ。

【００１２】

良い方式は、関数の低次多項式の内容を、未量子化の「ローパス」サブバンド内に射影する変換を用いる。こうした方式は、理想的には、他のサブバンド内にゼロまたは非常に小さい値を生成することも行う。従って、これに続く非ローパスのサブバンドの量子化は、十分低次の多項式によって良好にモデル化された関数の変換を大幅には変更せず、元の関数を近似する再構成は非常に良好なものとなる。

【００１３】

実現の真実性は、変換された関数における値が、元の関数領域内の一部の点の小さい近傍内の値のみに依存することを、非常に望ましくする。このことは、ＪＰＥＧ及びＭＰＥＧ規格における8×8ブロックの目的の１つである。これらの仕様では、領域の近傍どうしが一致（重複）するか交わらないかのいずれかであり、画像領域を、各々が別個の境界を有する分離した一まとまりの近傍に分割する。量子化から生じる近似は、これらの境界では程度が劣りがちであり（よく知られている、離散フーリエ変換における「ギブス効果」）、再構成した近似画像内に目に付く「ブロッキング」アーティファクト（歪像）を生じさせる。

【００１４】
ウェーブレット変換は、重複（オーバラップ）する近傍を有するが、小領域の近傍特性を有する変換クラスとして大いに注目を引き付けている。一部のウェーブレット変換は、ＪＰＥＧ／ＭＰＥＧのＤＣＴに比べて、関数を主にローパス・サブバンド内に射影する作業をより良好に行う。さらに、一部のウェーブレット変換（必ずしも上記一部のものと同じものではない）は、計算密度が大幅に低い。しかし、領域の近傍の重複は、データの取り扱い、メモリー利用及びメモリー帯域幅の領域において、実現上の大きな問題を強いる。領域を「ブロック」して、領域の境界、及びこれらの境界付近の近似の問題に戻ることは、なおも有用である。

【００１５】
領域の境界における変換は、境界点の所に作られた領域の近傍が、この境界点が属する領域ブロック内に存在しない、という問題をもたらす。種々のＪＰＥＧ及びＭＰＥＧ規格において具体化された、この問題に対する従来の取り組みは、ブロック内の領域値を、境界について対称な反射像にして、要求された近傍に「仮想」値及び仮想関数を作成することである。

【００１６】
この仮想関数が一般に近傍上の定数でなければ、この仮想関数は、不連続な一次導関数から生成される先点または折り目を境界上に有する。この不連続は低次多項式によっては良好にモデル化されず、従って、前記反射像が、量子化後に大きい値のままで残る非ローパスのサブバンド係数となる。このより大きな量子化誤差は、境界における近似誤差を増加させる。

【００１７】
ＪＰＥＧ－２０００規格１）に指定された変換の１つが、次式1.1及び1.2に示す可逆５－３変換である。

【数１】

$$Y_{2n+1} = X_{2n+1} - \left\lfloor \frac{X_{2n} + X_{2n+2}}{2} \right\rfloor \qquad (1.1)$$

$$Y_{2n} = X_{2n} + \left\lfloor \frac{Y_{2n-1} + Y_{2n+1} + 2}{4} \right\rfloor \qquad (1.2)$$

【００１８】
これらの式は整数－整数の写像（マッピング）であり、Ｙについて容器に逆向きに解けるので、この変換は可逆であり、入力Ｙをビット毎に正確に逆生成する（次式を参照）。

【数２】

$$X_{2n} = Y_{2n} - \left\lfloor \frac{Y_{2n-1} + Y_{2n+1} + 2}{4} \right\rfloor \qquad (2.1)$$

$$X_{2n+1} = Y_{2n+1} + \left\lfloor \frac{X_{2n} + X_{2n+2}}{2} \right\rfloor \qquad (2.2)$$

【００１９】
これらの式より明らかに、Y_{2n+1}は(2n+1)における二次導関数の半分の負値（二次導関数の半分の値にマイナスを付けた値）の推定値であり、関数が(2n+1)において一次多項式によって良好に近似されていれば、Y_{2n+1}はおよそ０である。

【００２０】
上式の四角カッコ（[]）内で定数を加算している目的は、推定値からあらゆるＤＣバイアスを除去することにある。ウェーブレット内の無修正のバイアスは、再構成したデータに振動的な誤差を生じさせやすく、この誤差は固定パターンのノイズ（雑音）として見られる。バイアスの推定及び訂正にはいくつかの可能性があり、ＪＰＥＧ－２０００規格で

はこれらのうちの１つを選択している。

【００２１】

画像の右境界が点2N-1の所にあれば、必要な値X_{2N}が利用できないので、式1.1は計算できない。ＪＰＥＧ－２０００規格は、この場合に対して、関数を対称な正側に拡張して、$X_{2N}=X_{2N-2}$を用いることによって応えることを要求する。この代入を式1.1に対して行えば、次式のようになる。

【数３】

$$Y_{2N-1}=X_{2N-1}-\left\lfloor\frac{X_{2N-2}+X_{2N-2}}{2}\right\rfloor= \quad (1.1\text{ext})$$
$$=X_{2N-1}-\lfloor X_{2N-2}\rfloor=X_{2N-1}-X_{2N-2}$$

【００２２】

この式はY_{2N-1}を生成し、これは、内側の点である上記二次導関数の半分の負値の推定値に対する、一次導関数の推定値である。さらに、二次導関数の推定値は、２つだけでなく３つの別個の点を用いることのみによって得られることは明らかである。偶数の指標を有するＸの持上げ項に必要な２つの点を限定する必要がある、というのは、これらの２点は逆向きのステップに利用可能な唯一のものであるからである。最も近い候補の指標は2N-4である。

【００２３】

特に1.2式及び2.1式に見られるように、５－３ウェーブレットフィルタのＪＰＥＧ－２０００の公式化は、計算中に定数１または２の加算すること、及び他の制限を含む。最大の演算速度及び演算効率用に実現する際には、これらの加算及び他の制限は、全体の演算負荷を非常に細切れにすることを要求して、性能の大幅な低下を生じさせ得る。

【発明の概要】

【課題を解決するための手段】

【００２４】

（発明の開示）

本発明はデータを圧縮するシステム、方法、及びコンピュータプログラムを提供する。最初に、内挿補間公式を受け取る。こうした内挿補間公式を利用して、データを圧縮する。使用中には、前記内挿補間公式が、入手不可能なデータ値を少なくとも１つ必要とするか否かを判定する。必要とする場合には、外挿補間演算を実行して、必要とする入手不可能なデータ値を生成する。

【００２５】

１つの好適例では、前記内挿補間公式をウェーブレットフィルタの構成要素とすることができる。他の選択肢（オプション）として、前記ウェーブレットフィルタを選択的に多相フィルタに置き換えることができる。

【００２６】

他の好適例では、複数のデータ値を複数のスパン（区間）にセグメント分割（区分）することができる。これにより、これらのスパンのうちの１スパン内のみのデータ値を利用することによって、前記内挿補間公式に関係する演算量を低減することができる。

【００２７】

さらに他の好適例では、データ値を量子化することができる。こうした好適例では、データ値の数量を低減することによって、エントロピー符号化に関連する演算量を低減することができる。データ値の数量は、これらのデータ値に関係する量子化演算中に低減することができる。

【００２８】

さらに他の実施例では、データ値を所定のデータ範囲に再構成することに関連する演算量を低減することができる。こうした演算は、単一のクリップ操作のみを実行することによって低減することができる。

【００２９】
１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。
【数４】

$$Y_{2n+1}=\left(X_{2n+1}+1/2\right)-\left\lfloor\frac{\left(X_{2n}+1/2\right)+\left(X_{2n+2}+1/2\right)}{2}\right\rfloor$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。

$$Y_{2N+1}=(X_{2N+1}+1/2)-(X_{2N}+1/2)$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。
【数５】

$$\left(Y_{2n}+1/2\right)=\left(X_{2n}+1/2\right)+\left\lfloor\frac{Y_{2n-1}+Y_{2n+1}}{4}\right\rfloor$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。
【数６】

$$\left(Y_0+1/2\right)=\left(X_0+1/2\right)+\left\lfloor\frac{Y_1}{2}\right\rfloor$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。
【数７】

$$\left(X_{2n}+1/2\right)=\left(Y_{2n}+1/2\right)-\left\lfloor\frac{Y_{2n-1}+Y_{2n+1}}{4}\right\rfloor$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。
【数８】

$$\left(X_0+1/2\right)=\left(Y_0+1/2\right)-\left\lfloor\frac{Y_1}{2}\right\rfloor$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。
【数９】

$$\left(X_{2n+1}+1/2\right)=Y_{2n+1}+\left\lfloor\frac{\left(X_{2n}+1/2\right)+\left(X_{2n+2}+1/2\right)}{2}\right\rfloor$$

１つの好適例では、前記ウェーブレットフィルタが、次式を含む内挿補間公式を含む。

$$(X_{2N+1}+1/2)=Y_{2N+1}+(X_{2N}+1/2)$$

【００３０】
本発明は、データを圧縮する他のシステム及び方法を提供する。最初に、単一装置でデータを受け取る。こうしたデータを、前記単一装置を利用して符号化して、第１フォーマットの第１圧縮データを生成する。さらに、この第１圧縮データを、前記単一装置を利用してコード変換（トランスコード）して、第２フォーマットの第２圧縮データを生成する。
【００３１】
１つの好適例では、前記符号化をリアルタイム（実時間）で行うことができる。さらに、前記コード変換をオフラインで行う（後でまとめて処理する）ことができる。
【００３２】
他の好適例では、前記第１圧縮データをコード変換して、前記単一装置に結合した通信

ネットワークの容量に整合させるべく適応させた第２フォーマットの第２圧縮データを生成する。

【００３３】

選択肢として、第１エンコーダを利用して符号化を実行することができる。さらに、デコーダ（復号化器）及び第２エンコーダを利用して、前記コード変換を実行することができる。

【００３４】

さらに、前記第１フォーマットにウェーブレットベースのフォーマットを含めることができる。さらに、前記第２フォーマットにＤＣＴベースのフォーマットを含めることができる。１つの特別な好適例では、前記第２フォーマットにＭＰＥＧフォーマットを含めることができる。

【００３５】

本発明は、単一集積回路上の複数のエンコーダを利用してデータを圧縮するシステム及び方法を提供する。最初に、前記単一集積回路でデータを受け取る。次に、前記単一集積回路が内蔵する複数のエンコーダを利用してデータを符号化する。

【００３６】

１つの好適例では、前記単一集積回路上の複数のチャンネルを利用してデータを符号化することができる。さらに、これらのデータをウェーブレットベースのフォーマットに符号化することができる。

【００３７】

本発明は、データを圧縮する他の単一モジュールのシステム及び方法を提供する。使用中には、単一モジュールを利用して光子を受け取る。その後に、この単一モジュールを利用して、これらの光子を表現する圧縮データを出力する。

【００３８】

選択肢として、前記圧縮データをウェーブレットベースのフォーマットに符号化することができる。さらに、この符号化に関連する変換操作をアナログで実行することができる。前記単一モジュールはさらに、撮像素子（イメージャ）を含むことができる。

【図面の簡単な説明】

【００３９】

【図１】現在利用可能な種々の圧縮アルゴリズム間のトレードオフの例を示す図である。

【図２】本発明の一実施例によりデータを圧縮／伸長する枠組みを示す図である。

【図３】本発明の一実施例によりデータを圧縮／伸長する方法を示す図である。

【図４】図３の方法を実行する対象のデータ構造を示す図である。

【図５】本発明の一実施例によりデータを圧縮／伸長する方法を示す図である。

【図６】本発明の一実施例によりデータを圧縮するシステムを示す図である。

【図７】単一の集積回路上の複数のエンコーダを利用してデータを圧縮するシステムを示す図である。

【発明を実施するための形態】

【００４０】

（好適な実施例の説明）

図２に、本発明による、データを圧縮／伸長するための枠組み（フレームワーク）２００を示す。この枠組み２００には、コーダ（符号化器）部２０１及びデコーダ（復号化器）部２０３が含まれ、これらが一緒になって「コーデック」を形成する。コーダ部２０１は、変換モジュール２０２、量子化器２０４、及びデータをファイル２０８に記憶するために圧縮するエントロピー・エンコーダ（符号化器）２０６を含む。こうしたファイル２０８を伸長するために、デコーダ部２０３は、逆変換モジュール２１４、逆量子化器２１２、及びデータを使用する（例えば、ビデオデータの場合には視聴する）ために伸長するエントロピー・デコーダ２１０を含む。

【００４１】

使用中には、変換モジュール２０２が、逆相関（減相関、デコリレーション）を目的と

して、（ビデオデータの場合には）複数の画素の可逆の変換を実行して、この変換は線形変換であることが多い。次に、量子化器２０４が変換値の量子化を行って、その後にエントロピー・エンコーダ２０６が、量子化した変換係数をエントロピー符号化する働きをする。

【００４２】

図３に、本発明によりデータを圧縮／伸長する方法３００を示す。１つの実施例では、この方法３００を、図２の変換モジュール２０２に関連して、変換モジュール２０２が可逆の変換を実行する方法で実行することができる。しかし、方法３００は所望のものに関連して実現することができる。

【００４３】

操作３０２では、データを圧縮するための内挿補間公式を受け取る（例えば、メモリー等から識別して取得する）。本実施例の関係では、データは圧縮可能なあらゆるデータとする。さらに、前記内挿補間公式は、内挿補間（例えばウェーブレットフィルタ等）を用いたあらゆる公式を含むことができる。

【００４４】

操作３０４では、前記内挿補間公式が少なくとも１つのデータ値を必要とするか否かを判定し、ここでは必要なデータ値が入手不可能である。こうしたデータ値は、前述したデータのあらゆる部分集合を含むことができる。必要なデータ値が入手不可能であるとは、これらの必要なデータ値が不在である、範囲外である、等であり得る。

【００４５】

その後に、外挿補間演算を実行して、必要で入手不可能なデータ値を生成する。操作３０６では、外挿補間公式は外挿補間を用いたあらゆる公式を含む。この方式により、データの圧縮を拡張する。

【００４６】

図４に、方法３００を実行する対象のデータ構造４００を示す。図に示すように、変換中に、複数のデータ値４０２が関係する内挿補間公式４０３によって、「最良の適合（ベストフィット）」４０１を達成することができる（図３の方法３００の操作３０２を参照）。データ値４０２のうちの１つが入手不可能であることが判明していれば（４０４参照）、前記外挿補間公式を用いて、こうした入手不可能なデータ値を生成することができる。以上の技法の１つの好適な実現に関する選択肢的な詳細を、以下に図５を参照して詳細に説明する。

【００４７】

図５に、本発明によりデータを圧縮／伸長する方法５００を示す。選択肢として、この方法５００を、図２の変換モジュール２０２に関連して、変換モジュール２０２が可逆の変換を実行する方法で実行することができる。しかし、方法５００は所望のものに関連して実現することができる。

【００４８】

方法５００は、ウェーブレットフィルタ用のエッジフィルタを生成する技法を提供する。最初に、操作５０２では、ウェーブレット方式を分析して、ウェーブレットフィルタが近似する局所的な導関数を決定する。次に、操作５０４では、ウェーブレットフィルタの特性及び利用可能なサンプル数にもとづいて、外挿補間に使用する多項式の次数を選定する。次に、前記選定した多項式の次数を用いて、ウェーブレットフィルタ毎の外挿補間公式を導出する（操作５０６参照）。さらに、操作５０８では、前記外挿補間公式を、各場合において利用可能なサンプルと共に利用して、特定エッジ（縁）のウェーブレットケースを導出する。

【００４９】

ヴァンデルモンド（Vandermonde）型行列を用いて前記係数について解く選択肢的な方法は、付録Ａに記載する。さらに、好適な外挿補間公式に関する追加的で選択肢的な情報及び関連情報を、以下に詳細に説明する。

【００５０】

Y_{2N-1}を左側から近似するために、二次多項式を左側から当てはめることができる。利用可能な値を用いて、2N-1における二次導関数の半分の負値を近似することは、次式1.1Rのようになる。この外挿補間二次式の可能な決定の１つを、付録Ａに記載する。

【数１０】

$$Y_{2N-1} = -\frac{1}{3}\left(X_{2N-1} - \left\lfloor \frac{3X_{2N-2} - X_{2N-4} + 1}{2} \right\rfloor\right) \quad (1.1R)$$

【００５１】

点が最右端である際には、式1.1の代わりにしき1.1Rを用いることができる（発明の背景を参照）。上式で、３を掛けることは、（ビット）シフトと（１の）加算で達成することができる。３で割ることの方がより手間がかかる。最右端の指標が2N-1であるこの場合については、式1.2によってY_{2N-2}を計算することには全く問題がない（発明の背景を参照）。最右端の点の指標が偶数（例えば2N）である場合には、式1.1については問題ないが、式1.2には欠けている値がある。ここでの目的は、前に計算した奇数の指標Ｙだけ、この問題の場合にはY_1及びY_3を用いて、偶数のＸからＹの推定値を減算することにある。指標2Nにおいて要求されたこの推定値は、上述したように、線形外挿補間によって得ることができる。適切な公式は、次式1.2Rによって与えられる。

【数１１】

$$Y_{2N} = X_{2N} + \left\lfloor \frac{3Y_{2N-1} - Y_{2N-3} + 2}{4} \right\rfloor \quad (1.2R)$$

【００５２】

左側の境界についても、これに対応する状況が当てはまる。要求される外挿補間を左側からよりもむしろ右側（内側）から行うエッジフィルタが適用される。この場合には、適切なフィルタは次式1.1L及び1.2Lによって表わされる。

【数１２】

$$Y_0 = -\frac{1}{3}\left(X_0 - \left\lfloor \frac{3X_1 - X_3 + 1}{2} \right\rfloor\right) \quad (1.1L)$$

$$Y_0 = X_0 + \left\lfloor \frac{3Y_1 - Y_3 + 2}{4} \right\rfloor \quad (1.2L)$$

【００５３】

これらの外挿補間境界フィルタ用の逆変換フィルタは、元のフィルタと同様に、即ち逆の代入によって得ることができる。この逆変換境界フィルタは、前向き境界フィルタを用いるのと全く同じ状況で、標準的なフィルタの代わりに用いることができる。こうしたフィルタは、次式2.1Rinv、2.2Rinv、2.1Linv、及び2.2Linvによって表わされる。

【数１３】

$$X_{2N-1} = -3Y_{2N-1} + \left\lfloor \frac{3X_{2N-2} - X_{2N-4} + 1}{2} \right\rfloor \quad (2.1R\ \ inv)$$

$$X_{2N} = Y_{2N} - \left\lfloor \frac{3Y_{2N-1} - Y_{2N-3} + 2}{4} \right\rfloor \quad (2.2R\ \ inv)$$

【数１４】

$$X_0 = -3Y_0 + \left\lfloor \frac{3X_1 - X_3 + 1}{2} \right\rfloor \qquad (2.1L \quad inv)$$

$$X_0 = Y_0 - \left\lfloor \frac{3Y_1 - Y_3 + 2}{4} \right\rfloor \qquad (2.2L \quad inv)$$

【００５４】

従って、１つの実施例は、フィルタの視覚特性を保ちつつ、従来技術の追加的なステップを回避する５－３フィルタの再公式化を利用することができる（例えば、次式3.1、3.1R、3.2、3.2L参照）。

【数１５】

$$Y_{2n+1} = (X_{2n+1} + 1/2) - \left\lfloor \frac{(X_{2n} + 1/2) + (X_{2n+2} + 1/2)}{2} \right\rfloor \qquad (3.1)$$

$$Y_{2N+1} = (X_{2N+1} + 1/2) - (X_{2N} + 1/2) \qquad (3.1R)$$

$$(Y_{2n} + 1/2) = (X_{2n} + 1/2) + \left\lfloor \frac{Y_{2n-1} + Y_{2n+1}}{4} \right\rfloor \qquad (3.2)$$

$$(Y_0 + 1/2) = (X_0 + 1/2) + \left\lfloor \frac{Y_1}{2} \right\rfloor \qquad (3.2L)$$

【００５５】

こうした公式化では、上述した追加を回避するために、特定の係数を1/2のオフセットまたはバイアスを伴って計算する。なお、この公式化では1/2の加算が多いように見えるが、実際の計算では、これらの加算を行う必要がない。式3.1及び3.1Rでは、1/2の加算の影響が相殺されていることがわかり、従って、これらの加算を入力データに適用する必要はない。その代わりに、カッコ内の項（Y_0＋1/2）等は、係数として実際に計算して記憶して、ウェーブレット変換ピラミッドの次のレベルに渡す量の名前として理解することができる。

【００５６】

ちょうど前の場合のように、ＪＰＥＧ－２０００逆フィルタは、次式4.2、4.2L、4.1、4.1Rのように再公式化することができる。

【数１６】

$$(X_{2n} + 1/2) = (Y_{2n} + 1/2) - \left\lfloor \frac{Y_{2n-1} + Y_{2n+1}}{4} \right\rfloor \qquad (4.2)$$

$$(X_0 + 1/2) = (Y_0 + 1/2) - \left\lfloor \frac{Y_1}{2} \right\rfloor \qquad (4.2L)$$

$$(X_{2n+1} + 1/2) = Y_{2n+1} + \left\lfloor \frac{(X_{2n} + 1/2) + (X_{2n+2} + 1/2)}{2} \right\rfloor \qquad (4.1)$$

$$(X_{2N+1} + 1/2) = Y_{2N+1} + (X_{2N} + 1/2) \qquad (4.1R)$$

【００５７】

ここに見られるように、逆向きの計算の入力として取得した値は、式3.1～3.2Lにおける前向き計算によって生成されるのと同じ項であり、1/2による補正を明示的に計算する必要は全くない。

【００５８】

このようにして、ウェーブレット変換の計算中に実行する算術演算の総数が低減される。

【００５９】
（選択肢的な特徴）
図２～５のシステム及び方法に関連して用いることのできる追加的で選択肢的な特徴及び技法を以下に説明する。なお、こうした選択肢的な特徴は、厳密には例示目的で説明するものであり、限定的なものではない。さらに、こうした特長は、以上の図２～５のシステム及び方法とは無関係に実現することができる。
【００６０】
一般的な動作の特徴
使用中には、変換モジュール（例えば図２の変換モジュール２０２等）は、画像をサブバンドに分離するフィルタバンクとして作用するウェーブレット・ピラミッドを利用することができ、これらのサブバンドの各々が約１オクターブ（即ち係数２）をカバーする。各オクターブには、水平、垂直、及びチェッカーボード（白黒交互の碁盤模様）の形に対応する３つのサブバンドが存在し得る。１つの実施例では、前記ピラミッドを一般に３～５レベルの深さにして、同数のオクターブをカバーすることができる。元の画像が少しでも平滑であれば、ウェーブレット係数が急速に減少する。画像が2/3のホルダー（Holder）係数を有することがあり、このことは、この画像が導関数の2/3を有することをおよそ意味する。ウェーブレット係数を絶対値が減少する順に整列させれば、これらの絶対値はN^{-S}の割合で減少するように見え、ここにNは列内の位置であり、Sは画像の平滑度である。
【００６１】
ウェーブレット・ピラミッドを形成した後に、量子化器（例えば図２の量子化器２０４等）によってウェーブレット係数をスケーリング（拡大縮小、量子化）して、視聴条件及び人間の視覚コントラスト感度曲線（ＣＳＦ：Contrast Sensitivity Curve）に整合する結果を出す。人間の視覚系（ＨＶＳ：Human Visual System）の特性を考慮することによって、クロマ（色度、彩度）のサブバンドを符号化するために使用するビット数を大幅に低減することができる。
【００６２】
必要なシリコン領域を最小にして実現可能な高速アルゴリズムを提供するために、従来の算術的な符号化器（コーダ）の使用を回避することができる。例えば前述したように、乗算器は、シリコン領域内では非常に高価になるので、回避することができる。さらに、こうしたアルゴリズムは、個別の実行要素毎に非常に良好な「高速パス（径路）」を持つことができる。
【００６３】
前記コーデックは、２つのインタレース（飛越し走査）ビデオフレームの画像グループ（ＧＯＰ：Group of Pictures）、境界用のエッジフィルタ、中間的なフィールド画像圧縮、及びブロック圧縮構造を用いることができる。小型単一チップ用の実現の特定の特徴は、次の表１のようにすることができる。
（表１）
・１つの実現は短いウェーブレットベースを用いることができ、これらはＨＶＳに整合すべく量子化した自然な光景（シーン）画像に焦点を置く者に特に適している。この実現は、加算及びシフト（桁ずらし）で達成することができる。フィールド毎の、水平方向の５つのフィルタの適用及び垂直方向の３つのフィルタの適用により生成したマラー（Mallat）ピラミッドを用いることができる。このことは動的な係数を有するフィルタを生成し、これらは、ローパス（低域通過）フィルタにおける２つの係数、及びウェーブレットフィルタにおける２つ、４つ、または６つの係数（12個のウェーブレット・サブバンドを生じさせる）である。修正したエッジフィルタをブロック及び画像の境界付近で用いて、これにより実際の画像値を利用することができる。結果的なビデオ・ピラミッドは実質的に０の列を有し、実質的に非０の列も有する。従って、符号化は表検索（テーブル・ルックアップ）によって効率的に行うことができる。
・他の解決法は、ＭＰＥＧ的な方法で用いる動き補償探索の代わりに、３Ｄウェーブレッ

ト・ピラミッドによる動画像圧縮を用いることができる。時間方向の変換圧縮を、４フィールドのＧＯＰに適用することができる。２レベルの時間マラー・ピラミッドをテンソル積として空間ピラミッドと共に用いることができる。線形エッジフィルタを密レベルで、修正ハール（Haar）フィルタを粗レベルで用いて、４つの時間サブバンドを生成することができる。これらの時間サブバンドの各々が圧縮されている。
・処理を、各々が32画素の走査線8本から成るブロックの処理に落とすことができる。このことは、ＲＡＭの必要量を、ＲＡＭをＡＳＩＣそのものの内部に配置できるような値まで低減する助けとなる。このことは、チップの個数を低減して、ＲＡＭの帯域要求を満足することを簡単にする。圧縮処理は、ストライプ毎に実行することができる（ストライプ当たり２回の通過）。
・さらに他の実施例は、ウェーブレット係数の量子化を用いて、圧縮のさらなる改善を達成することができる。量子化の分母は２のべき乗であり、シフトによって実現可能である。量子化は、スケーリング係数を各サブバンドに割り当てる処理とすることができ、サブバンド内の各係数に対応するスケーリング係数を乗じて、スケーリングした係数を整数化する。

【００６４】

他の選択肢として、ウェーブレットフィルタを選択的に多相フィルタに置き換える。１つの実施例では、こうした置き換えを、データ圧縮／伸長システムの変換モジュールで行うことができる（例えば、図２の変換モジュール２０２及び／または逆変換モジュール２１４）。もちろん、こうした特長は、本明細書に記載の他の種々の特徴とは無関係に実現することができる。この選択肢的な特徴に関するより好適な情報を以下に記述する。

【００６５】

本実施例では、ビデオ圧縮コーデックの設計において、従来の[例えば、有限インパルス応答（ＦＩＲ：Finite Impulse Response）]の情報廃棄または平滑化フィルタをウェーブレット情報保存フィルタと組み合わせることができる。ＦＩＲフィルタは単一で使用されるのに対し、ウェーブレットフィルタは常に相補対をなす点で、ＦＩＲフィルタをウェーブレットフィルタと区別することができる。また、ウェーブレット変換におけるＦＩＲフィルタは必ずしも、多相フィルタバンクとしての互いにの関係を持たない。

【００６６】

ビデオ圧縮は３ステップのプロセス（処理過程）で実行することができ、時として他のステップを追加するが、３つの主な段階は前述したように、変換、量子化、及びエントリ符号化である。これらの操作は通常、一般に行われているように、量子化中に情報を廃棄するに過ぎない。実際に、この操作を省略すれば、無損失（ロスレス）圧縮法となり得る。しかし、無損失圧縮は、有損失圧縮よりもずっと小さい圧縮比に限られ、有損失圧縮は、人間の視覚系を利用して、復号化した結果においては、視覚的に差のない情報、あるいは視覚的な差を無視できる情報を廃棄する。

【００６７】

許容できる結果において失われていることのある視覚情報の１つのクラスが、微細情報である。ビデオ圧縮に用いられる大部分の変換プロセスが、量子化ステップによって微細情報を廃棄することができるが、これらの変換プロセスは、直接的なローパスフィルタの実現よりも低い効率あるいは低い視覚的忠実性で変換を行う。

【００６８】

平滑化フィルタを実現する１つの方法は、ＦＩＲ構造を用いることによるものである。平滑化フィルタを実現する代わりの方法は、無限インパルス応答（ＩＩＲ：Infinite Impulse Response）構造を用いることによるものである。

【００６９】

画像またはデータ列の大きさを変化させる際には、関連するＦＩＲフィルタから成る多相フィルタバンク（ＰＦＢ：Polyphase Filter Bank）を用いることができる。こうした方法は、一部の詳細部分を除去して、さらなる処理用の対応するより小さい画像を生成することによって、画像を処理する。

【００７０】
多相フィルタバンクは、同じ帯域あるいは周波数選択特性を共用するが、元のサンプル上あるいはサンプル間の異なる位置を内挿補間した画素を生成する一組のＦＩＲフィルタを含むことができる。
【００７１】
例えば、多相フィルタバンクを用いて、画像（即ちビデオのフレーム）を元の幅の2/3に縮小することができる。多相フィルタバンクは、元の各画素の中間に内挿補間画素を算出して、元の位置に平滑化した画素を算出し、そして結果的な画素流（画素ストリーム）の３画素毎に１画素のみを保持することによって、このことを行う。
【００７２】
この方法により、保持されない画素の計算を省略することができ、画像の大きさを低減するより効率的な方法がもたらされる。このプロセスは、他の合理的な、部分的な大きさの変更に容易に広げられる。このようにして、多相フィルタバンクが小量の微細部分を円滑に除去して、１未満の係数で画像をスケーリングすることができる。この係数は1/2より大きくすることができる。
【００７３】
本発明は、多相フィルタをウェーブレットベースの画像圧縮プロセスの第１段として用いることによって、円滑な細部除去の利点を、ウェーブレット変換符号化の画質と組み合わせる。この組合せを用いることによって、多相バンクフィルタを用いることによる、円滑で、高品質で、アーティファクト（歪像）のない微細部分、及びこれらの微細部分を表現するために必要なビットを除去する利点を、ウェーブレット変換を画像及びビデオ圧縮の基本として用いることによる高速で効率的な演算及び高画質という周知の利点に加えることができる。
【００７４】
本発明の方法の第１の実施例では、まず多相フィルタバンクを画像の一方向、通常は水平方向に適用して、次に、従来の方法における量子化及び符号化の前に、ウェーブレット変換を画像に適用することができる。
【００７５】
本発明の方法の第２の実施例では、最初の特定方向のウェーブレット演算の前に、この方向に多相フィルタを適用することができるが、他の方向のウェーブレット演算後に行うこともあり得る。
【００７６】
さらに他の実施例では、いくつかの方向の各々について、この方向の最初のウェーブレット演算の前に、この方向に多相フィルタを適用することができるが、他の方向のウェーブレット演算後に行うこともあり得る。
【００７７】
少なくとも一部のウェーブレットまたはＤＣＴ変換の段階の前に無損失のフィルタリング（フィルタ処理）ステップを適用する本発明の方法には、いくつかの利点がある。例えば、ウェーブレット的な関数に限定されず、ＦＩＲ設計または多相設計のようなフィルタを、より高品位及び少ないアーティファクトのために設計することができる。ウェーブレットフィルタは、情報を廃棄することなしに２つの部分に分ける対の形に設計することができる。
【００７８】
変換操作の後よりも前に変換操作を適用することは、変換演算をより少ないデータに対して実行し、従って、演算時間をより少なくして、演算中の中間的な記憶容量をより少なくすることができることを意味する。変換は一般に圧縮プロセスの高価な部分であるので、この低減は、圧縮プロセス全体にわたって速度及び効率の大幅な改善をもたらす。
【００７９】
パイルを用いた平方ウェーブレット変換
さらに他の操作として、データ量を低減することによって、エントロピー符号化に関連

する演算量を低減する。１つの実施例では、こうした低減を、データ圧縮／伸長システムの量子化器において行う（図２の量子化器２０４参照）。もちろん、こうした特徴は、本明細書に記載した他の種々の特徴とは無関係に実現することができる。この選択肢的な特徴に関するより好適な情報を以下に述べる。

【００８０】

本実施例では、パイルを、復号化演算における演算として用い、従ってパイルは、これに続くステップの演算に直ちに使用できる。パイルに関するさらなる情報は、付録Ｂに記載する。

【００８１】

行列（マトリクス）データの希薄表現と称されるものを提供することは、特定の演算分野ではよく知られている。通常の行列は、行列要素である数の完結したアレイとして表現され、「稠密な」表現と称される。一部のプログラム・パッケージは、「希薄行列」に対する記憶、変換、及び操作を行い、希薄行列では０のエントリは１つずつ明示的に表現せず、暗示的に表現する。こうした「希薄な」表現の１つはゼロ－ラン（ゼロ列長）符号化であり、この符号化では、まとまって発生する０の個数によってゼロを表現する。この個数そのものは、０にも（２つの非ゼロ値が隣接している際）、１にも（単独のゼロ値）、より大きい値にもなり得る。

【００８２】

しかし、ビデオデータが行列でない場合には、通常はこのビデオデータに対して行列演算（即ち、乗算、逆行列計算、固有値分解、等）は適用しない。希薄行列演算の基礎的な原理を取り出して、ビデオ変換に移すことができる。

【００８３】

簡単に言えば、パイルは対のアレイから成り、各対が、非ゼロのアイテム（項目）の通常データのアドレス（またはオフセット）を、当該アイテムの値と共に与える。これらのアドレスまたはオフセットは並べ替え（ソート）した順序であり、このため、パイルを調べて、非ゼロ要素に対して、これらの要素のデータセット（データ集合）全体中の箇所を考慮に入れて操作を行うことによって、データ全体を隅から隅まで調べることができる。

【００８４】

パイルは、いくつかのデータ・アイテムに対して一度に行う同一操作を用いてデータを並列的に処理するコンピュータ（即ち：ＳＩＭＤプロセッサ（Single Instruction stream-Multiple Data stream Processor：同一命令で複数データを並列処理するプロセッサ））、及び制御の条件転移を行う比較的高価なコンピュータ上で効率的に実現可能なように特別に設計する。これらのプロセッサは、一般的な使用では、ビデオ及びオーディオを取り扱うために用いられ、時として「メディア・プロセッサ」と称される。

【００８５】

２つのデータセットに対して何らかの操作を実行する必要があり、両方のデータセットが希薄である際には、データが稠密に表現される際にはしなかった考慮が生じる。即ち、「データ・アイテムが互いに一致するのはいつか」ということである。

【００８６】

パイルとして表現される２つのデータセットに対する操作において、一致しているデータ・アイテムを識別するための基本的な操作は「マッチ・アンド・マージ（整合と結合）」と称される。２つのパイルを調べる際には、開始後の操作毎に、各パイルからのアドレス、及び出力値を生成した直後の、この出力値を割り当てたアドレスを得ることができる。値を生成して割り当てることができる次のアドレスを見出すために、２つの入力パイルが表現する２つのアドレスの小さい方を見出すことができる。両方のパイルがこのアドレスに合意すれば、各パイルからの利用可能なデータ・アイテムが存在し、これら２つの値に対して操作を行って所望の結果を生成することができる。そして、両方のパイル上の次のアイテムに進むことができる。

【００８７】

２つのパイル中の次のアドレスが異なる場合には、一方のパイル（データセット）中に

は非ゼロ値が存在するが、他方のデータセット（パイルによって暗示的に表現される）中にはゼロ値が存在し、１つの値及び０に対して演算を行って、ある値を生成することができる。あるいはまた、入力が０である際に、実行中の演算が０を生成すれば、何の値も生成されない。いずれの場合にも、小さいほうのアドレスを有するパイルのみについて、次のアイテムに進むことができる。

【００８８】

結果の値はある箇所に配置し、この箇所は、（アドレスを２つ以上進める際に常に明示的に０を書き込むことによる）稠密なアレイか、出力パイル中かのいずれかとする。

【００８９】

前述したように、ウェーブレット変換は、ウェーブレットフィルタ対を一組のデータに反復的に適用することであり、このデータは一次元でも二次元以上でもよい。ビデオ圧縮用には、２Ｄウェーブレット変換（水平及び垂直）または３Ｄウェーブレット変換（水平、垂直、及び時間）を用いることができる。

【００９０】

ビデオ圧縮器内の変換段の意図は、原画像のエネルギーまたは情報を集めて、画像または画像シーケンス（列）中の局所的な類似性及びパターンを利用することによって、できる限り小さい形にすることにある。あり得るすべての入力をできる限り圧縮することのできる圧縮器はないが、「一般的な」入力に対して良好に作用するように圧縮器を設計して、これらの圧縮器が「ランダム」あるいは「病的」な入力を圧縮し損なうことを無視することはできる。

【００９１】

変換が良好に作用して、画像情報が良好に集められて少数の変換係数にされると、残りの係数の多くは０になる。

【００９２】

前述したように、結果を量子化することも、ビデオ圧縮器の一段階である。この段階では、０に近い計算値は０で表現する。最終的な変換結果を量子化するか、あるいは、最終的な変換結果の量子化に加えて算出した係数を量子化するよりも、あるいはよりも、ウェーブレット変換の演算中に、算出した係数を量子化する方が望ましいことがある。

【００９３】

従って、一部のウェーブレット係数データ中に多くの０を得ることがあり、このことは、データに対する演算をもっと行う必要がある間に起り得る。

【００９４】

これに加えて、圧縮した画像またはビデオを表示するために復号化している際には、エントロピー符号化した重要な係数から、完全に満たされた（値を入れられた）表示用画像に向けての作業を行うことができる。最初の復号化ステップ、即ちエントロピー符号の復号化の一般的な出力は、デフォルトで０であると考えることのできる非重要な係数を多数伴う重要な係数の集合である。

【００９５】

このことが生じた際には、多くの０を伴う稠密なデータを希薄な表現に変換することは価値があり、このことは、前述したようにデータをパイル化することによって行うことができる。パイル表現は前記ゼロ－ラン表現に似ているが、通常は、ランレングス（ラン長：アドレスの差）ではなく、アドレスまたはオフセットを記憶する。このことは、パイルを作成するため、及びこのパイルを後に稠密な表現に拡張するための高速の処理を共に可能にする。

【００９６】

復号化の場合には、データが稠密な形式ではなく、エントロピー・デコーダ内で直接パイルを構成する方がより自然である。

【００９７】

ウェーブレット変換の処理は、パイル化の処理を受けるいくつかの場合をもたらし、これらを次の表２に示す。

（表２）
・伸長、両帯域をパイル化
・伸長、一方の帯域をパイル化
・伸長、入力がパイル化で出力が稠密
・圧縮、入力が稠密で出力がパイル化

【００９８】
１つの例を考える：圧縮されたビデオのフレームの復号化であり、符号化プロセスが、０に量子化される非常に多くの係数を生成している。伸長の最初の段階は、非ゼロ係数のエントロピー符号化またはビット符号化を元に戻し、フレーム内の各値の値及びその位置を与える。このことは単にパイルで表現される情報であり、間にあるすべてのゼロ値に明示的な値を入れることによってこの情報を直ちに稠密表現に拡張するよりも、パイルを用いてこの情報を記憶する方が非常に好都合である。

【００９９】
この段階では、逆ウェーブレット変換によって操作できる係数がある。逆変換の最終結果は、伸長されて直ちに表示可能な画像であり、この画像は一部が粗くなっているに過ぎない。

【０１００】
逆ウェーブレット変換の第１段階（各段階も同様）は、係数データの２つの領域または「帯域」からデータを取得して、これらのデータを組み合わせて中間的な帯域にするフィルタ演算であり、この中間的な帯域は同じプロセスのさらなる段階で使用する。この第１段階では、両帯域についてのデータが希薄であり、パイルで表現される。この段階の出力もパイルで生成することができ、ゼロに値を入れる必要はない。以下の表３の演算は、「帯域」パイルP_1及びP_2に対して行い、その結果は新たなパイルＲの形で生成され、前記２つの帯域からの係数対に対してフィルタ演算ステップW(p,q)を実行する。

（表３）
```
while not both EOF(P1), EOF(P2) {
I1=0; I2=0;
guard(P1.index ≦ P2.index, Pile_Read(P1, I1));
guard(P1.index ≦ P2.index, Pile_Read(P2, I2));
Conditional_Append(R, true, W(I1, I2)); };
Destroy_Pile(P1); Destroy_Pile(P2);
```

【０１０１】
なお、以上の演算は、付録Ｂに示すように、並列演算用に展開することができる。

【０１０２】
ウェーブレット変換を計算するために要する時間は、希薄表現、パイルを、多くのゼロ値を有する中間結果用に用いることによって低減することができる。こうした方法は、ウェーブレットベースの画像圧縮及びビデオ圧縮製品の性能及び演算効率を改善する。

【０１０３】
変換範囲の制限
さらに他の選択肢として、データ値を所定のデータ範囲に再構成することに関連する演算量を低減することができる。こうした演算は、単一のクリップ操作のみを実行することによって低減することができる。１つの実施例では、こうした低減を、データ圧縮／伸長システムの逆量子化モジュール（図２の逆量子化器２１２参照）内で行う。もちろん、こうした特徴は、本明細書に記載した他の種々の特徴とは無関係に実現することができる。この選択肢的な特徴に関するより好適な情報を以下に記述する。

【０１０４】
ディジタル画像圧縮及びディジタルビデオ圧縮法では、画像（またはフレーム）を数値のアレイとして表現して、各数値が、領域の明るさ、あるいはこの領域内の特定色（例えば赤色）の量を表現する。これらの領域は画素と称され、上記数値はサンプル値または成分値と称される。

【０１０５】
画像圧縮またはビデオ圧縮は、広範囲にわたる異なる方法で行われる。前述したように、これらの方法の多くは、変換の演算をステップとして含み、一連の算術演算を通して、画像を表現するサンプルのアレイを、係数と称する数値から成る異なるアレイに変換して、これらの数値は画像情報を含むが、個々の数値は小領域の明るさまたは色に対応しない。変換は同じ画像情報を含むが、この情報は、これらの数値にわたって、圧縮法のさらなる演算にとって有利なように分布する。
【０１０６】
こうした方法によって圧縮した画像またはフレームを再生する際には、圧縮したデータを伸長しなければならない。このことは通常、係数のアレイを取得してサンプルのアレイを生成する逆変換を計算することをステップとして含む。
【０１０７】
画像またはフレームのサンプルは一般に、小さいサイズ（桁数）、通常は８バイナリ（二進）ビットの整数によって表現される。こうした８ビットの数は256個の異なる値しか表現できず、これらの応用では、これらの値は一般に、0から255までの範囲の整数[0，255]であると考えられている。
【０１０８】
多くの規格及び動作条件が、この範囲より制約された範囲を強いる。例えば、CCIR-601（ITU-R BT．601-4）ディジタルビデオにおける画素成分（Y，U，V）のサンプル値は、[0，255]よりも小さい範囲内に存在する。特に、スクリーンの光のある部分における輝度Ｙ成分の有効範囲は、[16，235]内に存在すべく指定され、クロマ（色度）Ｕ、Ｖの範囲は[16，240]内に存在すべく指定されている。これらの範囲外の値は、明るさ以外の意味を持ち、例えばシンク・イベント（同期事象）を表わす。
【０１０９】
画像及びビデオ圧縮法は２つのカテゴリに分けることができ、即ち無損失（ロスレス）及び有損失である。無損失圧縮法は、伸長によって、圧縮用に提供されたのと全く同じ値を生成する方法で動作する。これらの方法については、範囲の問題は存在しない、というのは、出力が入力と同じ数値の範囲を占めるからである。
【０１１０】
しかし、有損失圧縮は、元の入力を近似することを想定した伸長出力を生成するに過ぎず、ビット単位で整合しない。この、画像を少し変更するという自由度を利用して、有損失法はずっと大きい圧縮比を得ることができる。
【０１１１】
有損失圧縮法の伸長部分では、算出したサンプルが対応する元のサンプルと同一であることが保証されておらず、従って、同じ値の範囲を占めることも保障されていない。従って、画像規格の範囲条件を満足するために、計算値を指定範囲に限定またはクリップ（頭打ち）するステップを含めなければならない。
【０１１２】
このクリップするステップを実行する簡単な方法は次の通りである：算出したサンプルｓ毎に、ｓ＞max（最大値）であるか否かをテスト（判定）して、そうであればｓをｓ＝maxに設定して、ｓ＜min（最小値）であるか否かをテストして、そうであれば、ｓ＝minに設定する。
【０１１３】
このステップを実行する他の方法は、ある演算プラットフォームで見出したMAX及びMIN演算子を使用し、ここでも、各サンプルに２つの操作を適用することができる。以上示した両方の方法、及び他の多くの方法は、加算及び減算のような単純な算術演算よりも、計算が高価になる。
【０１１４】
このプロセスは、画像またはフレーム内のすべてのサンプル値（すべての画素）について別個に実行することができるので、伸長法における演算の重要部分である。なお、通常

は十分、要求された範囲内に存在する算出したほとんどすべてのサンプルについて、上記両方のテストがなされておらず、従って両方のテストを演算しなければならない。

【０１１５】

上述した変換演算は一般に、次の特性を有する：結果的な係数のうちの１つが、フレーム全体かあるいはフレームの主要部分（ＭＰＥＧ技術ではブロック）全体の明るさのレベルを表わす。この係数はＤＣ係数と称される。変換を計算する方法に起因して、ＤＣ係数を変更すれば、当該フレームまたはブロック内の全サンプルの値が同様に、即ち行った変更に比例して変更される。従って、例えば、逆変換を計算する直前に、当該ブロック用に適切に選定した定数をＤＣ係数に加算することによって、ブロック内のあらゆるサンプルの値を同量だけ増加させることができる。

【０１１６】

圧縮法を実行する計算（コンピュータ）エンジンは一般に、飽和特性のある算術命令を有し、結果が計算されると、この結果がコンテナの表現範囲（８ビット量については[0, 255]）を超えていれば、結果をクリップしてこの範囲内に入れる。例えば、飽和減算命令に4及び9の値を与えれば、結果（4-9＝）-5がクリップされて、代わりに結果０が戻される。同様に、飽和加算命令は、250+10に対して結果255を戻す。

【０１１７】

多くの圧縮法における、画素成分値をクリップする低コストの方法を以下に説明し、この方法は、適切な限界への復号化に由来する。本実施例は、部分値にバイアスをもってきて、MAX/MIN演算子の一方のみを残すことによって、飽和算術計算を伴う２つのクリップの一方を実行する。要求される範囲が[llim(下限), ulim(上限)]＝[16, 240]である際の、より詳細な例を、次の表４に示す。

（表４）

１．各ブロック内のＤＣ係数にバイアスを加えて、これにより、すべての変換フィルタ後に、各部分が負の値-16（一般化した表現は-llim）だけオフセットされる。
コスト：画像またはブロック当たり１回の算術演算。
２．必ず、逆変換の最終的な算術ステップが０に飽和（クリップ）するようにする。
コスト：大部分の計算エンジンにおいてコストがかからない。
３．224（一般化した表現はulim-llim）による（分割）MAX演算（224に最大化する演算）を適用する。
コスト：サンプル当たり１回のMAX演算。
４．ADD 16（一般化した表現はllim）（16を加算する演算）を用いて、前記バイアスを除去する。直前のMAX演算により、これによるオーバーフローはあり得ないので、このバイアス除去は飽和算術演算を以って行う必要はない。
コスト：サンプル当たり１回のADD（加算）演算。

【０１１８】

ここで明らかなように、必要な範囲限定の演算コストは、サンプル当たり２回のMAX/MIN（最大化／最小化）演算から、ブロック当たり１回のADD（加算）演算、１回のMAX（最大化）演算、及び１回の単純なADD（加算）演算に低減される。

【０１１９】

一部の計算エンジン、例えばEQUATOR MAP-CAプロセッサ上では、本方法の使用による節減は、以上の説明より直ちに明らかである以上に、ずっと大幅なものとなり得る。これらのエンジン上では、いくつかのサンプルを組み合わせてワードにして、同時に演算することができる。しかし、これらの分割演算は、プロセッサの特定部分に限定され、圧縮用途では、性能を限定する元となり得る。こうしたエンジン上では、上記ステップ４におけるADD演算がオーバーフローし得ないということが非常に重要である。ステップ４は、空間分割したADD演算を用いる必要はないが、通常のADD演算を用いて、いくつかのサンプルに対して、これらがあたかも分割されているが如く一度に演算を行うことができる。この通常の演算は、プロセッサの、さほど高負荷がかかっておらず、他の必要な分割演算との重複、あるいは同時実行が可能な部分を用いて行うことができ、逆変換の計算時間の大幅な

節約ができる。

【０１２０】

図６に、本発明の一実施例によりデータを圧縮するシステム６００を示す。選択肢として、システム６００を、以上説明したことに関係して実現することができる。しかし、もちろん、システム６００はあらゆる所望のことに関係して実現することができる。

【０１２１】

システム６００は、単一デバイス６０４上に具現したエンコーダ６０２を具えて、エンコーダ６０２は、データを符号化して第１フォーマットの第１圧縮データを生成する。さらに、トランスコーダ６０６を、エンコーダ６０２と同じ単一デバイス６０４上に具現して、トランスコーダ６０６は第１圧縮データをコード変換（トランスコード）して第２フォーマットの第２圧縮データを生成する。

【０１２２】

使用中には、データは単一デバイス６０４で受信される。こうしたデータは単一デバイス６０４を利用して符号化されて、第１フォーマットの第１圧縮データが生成される。さらに、この第１圧縮データは、単一デバイス６０４を利用してコード変換されて、第２フォーマットの第２圧縮データが生成される。

【０１２３】

１つの実施例では、前記符号化をリアルタイムで行うことができる。さらに、前記コード変換をオフラインで行うことができる。他の実施例では、第１圧縮データをコード変換して、単一デバイス６０４に結合した通信ネットワークの容量に整合すべく適応させた第２フォーマットの第２圧縮データを生成する。

【０１２４】

選択肢として、第１デコーダを利用して符号化を実行することができる。さらに、図６に示すように、デコーダ及び第２エンコーダを利用してコード変換を実行することができる。

【０１２５】

さらに、前記第１フォーマットはウェーブレットベースのフォーマットを含むことができる。さらに、前記第２フォーマットはＤＣＴベースのフォーマットを含むことができる。１つの特別な実施例では、前記第２フォーマットがＭＰＥＧフォーマットを含むことができる。追加的で選択肢的な特徴に関するより好適な情報を以下に記述する。

【０１２６】

前述したように、画像及びビデオ・シーケンスを用いた通信モードがいくつか存在する。直接的なリアルタイムの視聴に加えて、画像またはビデオ・シーケンスを捕捉して、後の時間に伝送することができ、この後の時間は、捕捉直後でも、より先の時間まで遅延させてもよい。

【０１２７】

これに加えて、ビデオ・シーケンスの受信は、テレビを見るようにビデオを見るが記憶しないリアルタイム・モードでも、後の視聴用にシーケンスを記憶する他のモードでも行うことができる。

【０１２８】

これらの種々の選択肢は、他の組み合わせに加えて、３通りの使用のシナリオに組み入れられる。これら３通りのシナリオは次の通りである。

１．送信機と受信機が共にリアルタイムで動作する、上述したビデオフォンまたはピクチャフォン（テレビ電話）。この動作は、圧縮、符号化、及び伸長のすべてを、ビデオを捕捉する速度でリアルタイムで実行する必要があり、そして伝送チャンネルは、圧縮したビデオのフルレート（最大速度）を搬送する必要がある。

２．ソースまたはネットワークにおいてビデオを捕捉し記憶して、受信機においてリアルタイムで視聴するストリーム動作。この動作は、リアルタイムの復号化を必要とするが、伝送の前にシーケンスを処理することを可能にする。このモードは、圧縮したビデオのフルレートを搬送するための、少なくともネットワークから受信機までの伝送チャンネルを

必要とする。これに加えて、大部分の伝送チャンネルについては、受信機がいくらかの量のシーケンスを一時蓄積（バッファ）して、伝送レート（速度）に変動が存在しても円滑な再生を維持しなければならない。
３．ソースにおいてビデオを捕捉して記憶して、非リアルタイムで受信機に伝送して、受信機において後の再生用に記憶するメッセージまたはファイル転送モード。このモードは、リアルタイム・ビデオのフルレートが搬送不可能な伝送チャンネル上での動作を可能にし、そして受信者が繰り返し再生、一時停止することを可能にするか、さもなければ視聴体験を制御することを可能にする。
【０１２９】
捕捉して１つのフォーマットに圧縮した画像またはビデオは、他の圧縮フォーマットに変換することができる。この動作はコード変換（トランスコーディング）と称される。この動作は、最悪の場合には、入力フォーマットを完全な画像またはビデオに伸長した上で、所望の出力フォーマットに圧縮することによって行う。多くのフォーマット対については、この最悪の場合の方法よりも廉価な、利用可能な方法が存在し得る。
【０１３０】
セル電話ネットワークのような多くのネットワークでは、異なるユーザが、画像またはビデオ用の異なるフォーマットを好むか、あるいは必要とし得る。このことは、たとえすべてのユーザが例えばＭＰＥＧ－４規格に固まっても起り得る、というのは、こうした規格は外形（プロファイル）、サイズ（大きさ）、及び他のパラメータについて多くの選択肢を提供するからである。この理由及び他の理由により、送信装置と受信装置とが、特定伝送において使用すべきフォーマットについて交渉することが望ましいことがある。最も簡単な場合には、各装置が、自分が取り扱い可能なフォーマットのリストを提供して、両者が、両者のリストの共通部分から、互いに受け入れ可能な１つを選定する。こうした交渉にはより複雑な形態が存在するが、概略の効果は同じであり、送信者は、接続開始後に伝送すべきフォーマットのみを知る。
【０１３１】
接続の一部としてコード変換が必要な際には、コード変換は、伝送元の装置でも中間的な位置でも実行することができる。一部のネットワークは、自前の能力が全く異なる装置間の相互通信を提供するために、ネットワークの動作の一部としてコード変換サービスを提供することができる。このことは、移動装置の複雑性、及び従ってコストを低く保つことの手助けとなる。
【０１３２】
上述した、ビデオデータのレート（速度）と伝送チャンネルのレートが異なるため、次の新たなモードで動作させることが有利であり得る。装置がビデオを捕捉して、以下に説明する複雑度の低い圧縮法を用いてこのビデオをリアルタイムで圧縮して、圧縮したビデオ・シーケンスを記憶する。そして後に、装置はこのビデオ・シーケンスをコード変換して、受信者またはネットワークにとって受け入れ可能なフォーマットにすることができる。このことは、ネットワークのフォーマット規格との完全な互換性と共に、低電力動作、長いバッテリ寿命、及びより簡単な装置内の回路を可能にする。
【０１３３】
この動作スタイルの選択肢的な利点は柔軟性（フレキシビリティ）であり、リアルタイム圧縮の選定は、装置が直接通信可能な受信機の範囲を限定しない。上述したように、伝送フォーマットは、転送呼びの時点で交渉することができる。このようにして、装置は、より広いフォーマットの範囲をサポート（支援）することができる、というのは、装置は、広く最適化した各自のリアルタイム実現を持つ必要がないからである。
【０１３４】
上述した動作スタイルの他の選択肢的な利点は、前記コード変換はビデオ捕捉の速度で動作させる必要はないが、この速度よりずっと低いことが多い伝送ネットワークの速度に整合させることができる、ということである。より低速度のコード変換は、より小さく、かつ標準的なリアルタイム・プロセッサが消費するよりも少ない電力を消費する回路で行

うことができる。従って、装置全体の電力消費、装置のバッテリ寿命、複雑性、及びコストが低減される。

【０１３５】

この動作のスタイルのさらに他の選択肢的な利点は、画像及びビデオの伝送を、日中の電話料金のようにコストが高い時間帯から、夜間料金のようにコストがより低い時間帯（あるいは、現在のセル電話の課金方式では、無料の時間帯さえもある）まで延期できることにある。

【０１３６】

前記伝送は、他の時間には、時間帯以外の要因により、より低コストとなり得る。例えば、セル電話は、ホーム領域（自社のサービスエリア）に戻った際には、「ローミング（他社のサービスエリアでの通話）」時よりも低料金を課せられる。

【０１３７】

上述した延期伝送は、何らかの延期動作を行うための装置の使用を必ずしも必要としない。伝送は、伝送レート及び伝送スケジュールについて装置が有する情報にもとづいて、装置によって自動的にスケジュールすることができる。従って、ユーザの利便性は保たれる。

【０１３８】

もちろん、一部のメッセージは他のものより認知されるべき緊急性が高く、ユーザは、伝送を延期すべきか否か、及び延期させる時間を容易に指定することができる。

【０１３９】

画像及びビデオを非リアルタイムで転送する際には、転送の進行中に、装置のユーザが発呼を行いたいこと、あるいは発呼を着信すること、あるいは他の何らかの理由で接続が切断されることがあり得る。情報の既に良好に転送された部分を再送しなければならないことなしに、中断された転送の再開を可能にする情報を提供することは、コンピュータ・ネットワークの分野においてよく知られている。

【０１４０】

こうした中断可能な転送は、発呼を入れるような意図した中断、及び接続が失われるような意図しない中断を共に可能にする。

【０１４１】

受信装置がビデオ・シーケンス全体を記憶する容量を持つ必要はない。コード変換のソース（送信元）装置は、送信装置よりもずっと簡単でずっと能力の低い受信機を含むストリーミングモード受信機への送信を行うことができる。このことは、進んだコード変換装置を、既存の装置のネットワーク内に取り入れることを可能にする。

【０１４２】

標準的な画像及びビデオフォーマットは、エラー（誤り）検出法、エラー訂正法、及びバーストエラー（まとまった単発的なエラー）制御法を提供する。これらの標準的なフォーマットにコード変換することによって、装置は、複雑度が低く低電力の捕捉圧縮法を用いつつ、標準的なエラー回復機能を十分に利用することができる。

【０１４３】

低い複雑度のリアルタイム処理を用いて対象の信号を捕捉して、後に伝送、記憶、及びさらなる処理により適したフォーマットにコード変換する思想は、画像及びビデオ以外の信号、無線伝送以外の使用、及び移動個人端末以外の装置にも適用することができる。例えば、軍事諜報センシング、赤外線リモートセンシング、ソナー、分光望遠鏡、電波望遠鏡の信号、ＳＥＴＩ（Searching for Interstellar Communications：電波天文学）チャンネル、生化学的測定、地震信号、及び他の多くのものが、この基本方式を利用することができる。

【０１４４】

図７に、単一集積回路７０４（例えばＡＳＩＣ）上の多数のエンコーダ７０２を利用してデータを圧縮するシステム７００を示す。選択肢として、システム７００は、以上に説明した概念に関係して実現することができる。しかし、もちろん、システム７００はあら

ゆる所望のものに関係して実現することができる。

【０１４５】

図に示すように、第１組のデータを符号化する第１エンコーダを、単一集積回路７０４上に具現する。さらに、第２組のデータを符号化する第２エンコーダを、第１エンコーダと同じ単一集積回路７０４上に具現する。もちろん、同様の目的で、単一集積回路７０４上により多数のエンコーダを具現することができる。

【０１４６】

使用中には、データは単一集積回路７０４で受信される。そしてこのデータは、単一集積回路７０４が内蔵する複数のエンコーダ７０２を利用して符号化される。

【０１４７】

１つの実施例では、単一集積回路７０４上の複数のチャンネルを利用して、データを符号化することができる。さらに、データをウェーブレット・フォーマットに符号化することができる。

【０１４８】

多くのビデオ圧縮の応用（アプリケーション）が、ＡＳＩＣを含む複数の符号化または復号化段によって、より良好に行われる。その例は、ＴｉＶｏ（登録商標）及びリプレイ（繰り返し再生）ＴＶの製品のような、パーソナル（個人用）ビデオレコーダ（ＰＶＲ）あるいはディジタル・ビデオレコーダ（ＤＶＲ）のカテゴリ（範疇）であり、ここでは圧縮及び伸長のプロセスを同時に実行しなければならない。他の例はビデオ・サーベイランス（映像監視）レコーダであり、ここではカメラからの多数のビデオ信号をまとめて、多重化、圧縮、及び記録しなければならない。

【０１４９】

いくつかの圧縮回路を単一ＡＳＩＣ上に置くか、あるいは圧縮回路と伸長回路の組合せを単一ＡＳＩＣ上に置くことは、直接的及び間接的な利点を共にもたらす。直接的な利点は、パッケージ数の低減、ピン数の低減、電力消費の低減、及び回路ボード面積の低減である。これらのすべてが、製品コストの低減に寄与する。

【０１５０】

間接的な利点は、ビデオ選択回路と多重化回路を同一チップに内蔵可能であることを含み、ピン数及びボード（基板）面積をさらに低減する。

【０１５１】

ビデオ圧縮法が存在し、これは例えば、Droplet Technology, Inc.（登録商標）によって開発された、図２～５を参照して説明したアルゴリズムであり、これらのアルゴリズムは、実現に必要な回路が、従来の標準的な圧縮法よりもずっと少ない。これらの進んだ圧縮法の複数の例は、その優れた設計により、単一ＡＳＩＣ上あるいは他の集積回路上に集積することができる。

【０１５２】

データを圧縮するための、他の単一モジュールシステム及び方法が提供される。使用中には、単一モジュールを利用して光子を受け取る。その後に、これらの光子を表現する圧縮データを、この単一モジュールを利用して出力する。

【０１５３】

選択肢として、圧縮データをウェーブレット・フォーマットに符号化する。さらに、符号化に関連する変換操作をアナログで実行する。前記単一モジュールはさらに、撮像素子（イメージャ）を含むことができる。

【０１５４】

本実施例を実現して、撮像アレイ－ＣＭＯＳまたはＣＣＤカメラあるいは他の装置を構成して、ビデオを捕捉して圧縮したディジタルビデオを伝送するプロセス全体を促進することができる。

【０１５５】

直接ディジタル化した画像及びビデオは多数のビットを占め、一般に、記憶、伝送、及び他の使用のために画像及びビデオを圧縮する。いくつかの基本的な圧縮の方法、及び非

常に多数のこれらの変形法が知られている。一般的な方法は、３段階のプロセス、即ち変換、量子化、及びエントロピー符号化によって特徴付けられる。

【０１５６】

ビデオ圧縮器内の変換段の意図は、原画像のエネルギーまたは情報を集めて、画像または画像シーケンス中の局所的な類似性及びパターンを利用することによって、できる限り小さい形にすることにある。本実施例は、「一般的な」入力に対して良好に作用して、「ランダム」あるいは「病的」な入力の圧縮し損ないは無視する。

【０１５７】

ＪＰＥＧ[１]、ＭＰＥＧ－２[２]、及びＭＰＥＧ－４[４]のような多くの画像圧縮及びビデオ圧縮法は、変換段として離散コサイン変換（ＤＣＴ）を用いる。

【０１５８】

ＪＰＥＧ－２０００[３]及びＭＰＥＧ－４テキスチャ[４]のような一部のより新しい画像圧縮及びビデオ圧縮法は、変換段として種々のウェーブレット変換を用いる。

【０１５９】

ウェーブレット変換は、一組のデータにウェーブレットフィルタ対を反復的に適用することから成り、一次元でも二次元以上でもよい。画像圧縮用には、２－Ｄウェーブレット変換（水平及び垂直）を用いることができ、ビデオ圧縮用には、３－Ｄウェーブレット変換（水平、垂直、及び時間）を用いることができる。

【０１６０】

ウェーブレットフィルタ対は、画像（または画像の一部分）を処理して２つの画像を生成して、これらの画像の各々が入力画像の半分の大きさであり、一方が「ローパス（低域通過）」または「平均」または「ぼかし」と考えられ、他方が「ハイパス（高域通過）」または「詳細」または「エッジ（縁）」と考えられる。入力画像の完全な情報が保たれ、（多くの場合には）変換した画像対から原画像を正確に再構成することができる。ウェーブレットフィルタ対は一般に、１つの次元の画像を処理して、この次元は水平、垂直、及び時間（フレームの時系列にわたる）のいずれかである。完全なウェーブレット変換は、いくつかの次元に順次適用する一連のステップから成る。一般に、前のステップの結果のすべてが後のステップに引き継がれるわけではなく、ハイパス画像はさらなるフィルタリング（フィルタ処理）なしに保たれることがある。

【０１６１】

カメラは、その心臓部に撮像デバイスを有し、撮像デバイスは、変化する光の輝度及び色に応答して、後の表示及び他の使用のためにこれを記録するものである。今日のディジタル・スチルカメラ及びビデオカメラ用の一般的な撮像デバイスは、ＣＣＤ及びＣＭＯＳアレイである。これらの両者が、画素毎に光に応答して電荷を蓄積して、この電荷の量を転送して読み出す方法が、両者で異なる。

【０１６２】

ＣＭＯＳ（Complementary Metal-Oxide Semiconductor：相補性金属酸化膜半導体）撮像デバイスは、より新しい技術であり、ＣＣＤよりも廉価に作製することができる。ＣＭＯＳ撮像デバイスのキーとなる利点は、撮像チップの処理がディジタル論理チップの処理にかなり近いことにある。このことは、制御及び他の機能を同じチップ上に含めることをより容易にする。しかし、両種類のチップ共、目に見える光量を表現するアナログ電荷または電圧または電流を測定するために、必然的に最低レベルのアナログ回路で構成することになる。

【０１６３】

ＣＭＯＳ撮像デバイスは、ＤＲＡＭ（Dynamic Random-Access Memory：記録保持動作が必要な随時書込み読出しメモリー）と構造が非常に類似しており、画素で見える光を表現する電荷を、アレイを横断する金属トレース（線）の格子に沿ってアレイの端に転送する。この読出し法はメモリーチップにとって標準的な慣用法であり、産業において十分に発達している。

【０１６４】

ＣＣＤ撮像デバイスは、より古い技術であるが、十分に発達し、より低いノイズ及びより良好な感度を提供する。ＣＣＤ（Charge-Coupled Devices：電荷結合デバイス）は、画素に見える光を表現する電荷を、バケツリレーのようにセルからセルへ渡すことによって、アレイの端に転送する。

【０１６５】

ＣＭＯＳ撮像デバイスまたはＣＣＤ撮像デバイスは、アレイの端に転送される電荷が「０」または「１」ビット値を表わすだけでなく、明るさの値の範囲を表わす点で、ディジタル・メモリーデバイスとは異なる。従って、アナログ－ディジタル変換器が必要になる。この変換を進めるに当たり、信号は増幅され、そして、エラー、及びチップの製造及び動作上のばらつきを打ち消すための他の処理を受けることが多い。一般的な処理ステップは「相関二重サンプリング」であり、ここでは、この回路部分についての漏洩電流の尺度としての暗サンプルを取得して記憶し、そしてこの暗サンプルを画像サンプルから減算して、ノイズパターンを低減する。

【０１６６】

アナログ処理は差動増幅器内で行われ、差動増幅器は、主に、いずれかの入力の絶対値ではなく入力間の差に応答する回路である。

【０１６７】

光の捕捉と記憶しているディジタル画像との間の処理連鎖（チェーン）中のある点で、信号をアナログ（電荷、電圧、または電流）表現からディジタル表現に変換しなければならない。

【０１６８】

アナログ－ディジタル変換を、連鎖中の先の方で行うか後の方で行うかは選択可能なので、処理全体中の一部の段階をアナログ形式で行うかディジタル形式で行うかの選択肢が存在する。

【０１６９】

ウェーブレットの一段階であるウェーブレットフィルタ対は、一部の実現では、隣接する画素値及び近傍の画素値どうしの加算及び減算の非常に単純な組から成る。例えば、”Harr Wevelet（ハー・ウェーブレット）”と称される有用なフィルタ対は、次式1.1H及び1.2Hの合計及び差だけである。

$$L_n = X_{2n} + X_{2n+1} \qquad \text{式1.1H}$$

$$L_n = X_{2n} - X_{2n+1} \qquad \text{式1.2H}$$

【０１７０】

上式は、入力画像「Ｘ」の同じ２つのサンプルから、「ハイ（High）」変換画像の１サンプル及び「ロー（Low）」変換画像の１サンプルを生成する。

【０１７１】

他のウェーブレットフィルタも可能であり使用され、一部のものは非常に複雑であるが、一部のものは少数のHarrステップを実行する程度に簡単であり、これらのHarrステップを総計して、一定量でスケーリング（拡大縮小）する。

【０１７２】

例えば、ＪＰＥＧ－２０００規格[１]に指定されている変換の１つが、式1.1及び1.2で前述した可逆５－３変換である。

【０１７３】

式に見られるように、ウェーブレットフィルタ対全体は、５回の加算／減算演算及び２回のスケーリング演算を行い、連続アナログ領域ではフロア演算が消滅している。

【０１７４】

アナログ値を総計することは容易であり、差動増幅器（加算用でも減算用でも）によって当然達成されること、及び一定量によるスケーリングは、アナログ信号についてのすべての演算中で最も簡単な演算であり、１個または２個のレジスタしか必要としないことが判明している。

【０１７５】

これとは対照的に、ディジタル領域で値を総計することは、ビット毎の加算論理回路及びキャリー（繰り上がり）の連鎖を必要とし、ある特定の一定量によるスケーリングは容易であるが、一般的なスケーリングはディジタル論理回路では安価ではない。

【０１７６】

ＣＭＯＳ及びＣＣＤ撮像デバイスは現在、増幅、及びチップ上の画素サンプルからのノイズの減算を行うために差動増幅器を用いているので、アナログ－ディジタル変換の前に、一部の簡単な処理ステップをチップ上で実行することはかなり容易である。これらのステップの実行は、チップにある程度のアナログ回路を追加することになるが、少量の回路とすることができる。

【０１７７】

好適なものを含めたウェーブレット変換の一部の実現では、演算の最初のステップが最も高価であることが判明している。このことは、最初の各ステップが、後段で処理すべき画像の量を低減し、各フィルタ段による「ハイパス」画像出力のさらなる処理は必ずしも行わないからである。従って、アナログ－ディジタル変換を行う前に、最初のステップあるいは最初のいくつかのステップを実現することにより、ディジタル処理を大幅に低減することができる、というのは、ディジタルで処理しなければならないのは「ローパス」画像のみだからである。この利点は、ディジタル回路の量を低減することによって、このディジタル回路が占めるチップ面積を低減するか、あるいは、ディジタル回路をより低速で動作させて、その電力消費及び熱発生を低減することのいずれかに役立てることができる。

【０１７８】

画像またはビデオ圧縮の変換段はＤＣＴを用いて実行することができ、この処理は画像をスペクトルに変換し、このスペクトルの逐次的なサンプルは、画像内の空間的周波数の範囲の内容を表現する。ＤＣＴの一部の実現はHaarステップを使用し、これらのステップは、アナログで行うことの恩恵も受けることができる。

【０１７９】

ウェーブレット変換では通常、水平のフィルタ対を最初のステップとして計算することができる。このことは、アナログ・フィルタリングにとっても好都合と考えられる。最初の垂直フィルタ・ステップを実行する前に２回の水平ステップを実行することができ、このことはアナログにおいても好都合である。

【０１８０】

垂直フィルタ・ステップは、垂直に隣接する画素が同時に存在することを必要とする。従来のラスター順序の画像走査（左上から右下まで水平ラインを順次走査する）では、こうした画素どうしが大きな時間（ライン時間）を隔てて出現する。しかし、ＣＭＯＳ撮像デバイスのようなチップ撮像素子では、何本かのラインがまとまって出現するように走査順序の再編成を考えることが合理的であり、そうすれば、垂直フィルタ・ステップもアナログで実行することは、最初の水平フィルタ・ステップの前でも後でも実現可能である。

【０１８１】

カラー画像を捕捉する撮像チップは一般に、各画素の前面にカラーフィルタを配置し、この画素を赤色、緑色、または青色の応答うちの１つに限定している。これらのフィルタは、画像内の至る所でこれらの３色のすべてが隣接してサンプリング（標本化）されるようなパターンに配置する。

【０１８２】

しかし、ディジタルビデオ規格は、ＲＧＢ以外の成分配置の方が好ましい。最も広範に用いられているものは、ＹＵＶまたはYC_bC_rであり、ここではＹ成分が白黒の明るさまたは「輝度」を表現し、Ｕ及びＶ成分がそれぞれ、青色または赤色と輝度との色差を表現する。この表現の理由は、人間の視覚応答はＣ成分における分解能がより低く、従って、より小さい画像のディジタル表現を可能にするからである。ＹＵＶ表現は、圧縮にも好都合である。カラー撮像チップは、ＲＧＢ画素値をＹＵＶ値に変換する動作を行う回路を、アナログ（変換前）かディジタル（変換後）かのいずれかで提供するものもある。

【０１８３】
カラー変換とウェーブレットフィルタ・ステップとは、いくつかの方法のいずれかで組み合わせることができる。例えば、アナログ・カラー変換を最初のアナログ・ウェーブレットフィルタ・ステップに先行させることができ、この場合には、ウェーブレットフィルタがＹ成分の全帯域、及びＵ及びＶ成分の半分の帯域に作用する。あるいはまた、ウェーブレットフィルタを、撮像アレイからのＲ、Ｇ、及びＢ成分に最初に適用し、これに続いてＹＵＶへのカラー変換を行い、この場合には、フィルタは３つの成分信号の全帯域に作用する。
【０１８４】
他の構成では、従来のカラー変換ステップをすべてまとめて省略して、ＲＧＢ成分をウェーブレット変換に供給する。ウェーブレット変換には、ＹＵＶへの変換をその動作の一部として達成するバージョン（版）が存在する。この構成では、カラー変換を行うアナログ回路を、最初のウェーブレット変換を行うアナログ回路に置き換えて、アナログ回路の正味の増加なしにディジタル回路を低減して、ディジタル・ウェーブレット圧縮処理とのインタフェースを非常に明確にする。
【０１８５】
このように、最初のウェーブレットフィルタ・ステップのアナログ演算を含めることにより、圧縮したディジタルビデオを捕捉するサブシステムをより効率的にする方法が示された。このことは、モノクロ画像に対しても行うことができ、そしていくつかの方法で、カラーディジタル撮像素子のカラー変換段と組み合わせることができる。この方法は、ウェーブレットベースの画像圧縮及びビデオ圧縮製品の性能及び演算効率を改善する。
【０１８６】
以上では種々の実施例を説明してきたが、これらは例として提供するものに過ぎず、限定的なものではないことは明らかである。従って、本発明の好適例の範囲は、上述した好適な実施例のいずれによっても限定されるべきものではなく、特許請求の範囲及びこれと等価なものによってのみ限定される。
【０１８７】
（付録Ａ）
３つの値［X_{2N-1} X_{2N-2} X_{2N-4}］を持つことができ、そして次の二次方程式用の３つの係数を必要とする。
【数１７】

$$\begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix} \begin{bmatrix} x^0 \\ x^1 \\ x^2 \end{bmatrix} = a_0 + a_1 x + a_2 x^2$$

二次導関数の半分の負値は$-(1/2)2a_2$となり、従って重要なのはa_2のみである。この場合には、二次式は次式のようにより簡単に見出される。
【数１８】

$$\begin{bmatrix} \tilde{a}_0 & \tilde{a}_1 & \tilde{a}_2 \end{bmatrix} \begin{bmatrix} (x-2N)^0 \\ (x-2N)^1 \\ (x-2N)^2 \end{bmatrix} = \tilde{a}_0 + \tilde{a}_1 (x-2N) + \tilde{a}_2 (x-2N)^2 .$$

ここに、

$$a_2 = \tilde{a}_2$$

ヴァンデルモンド（Vandermonde）型係数行列を有する３つの線形方程式を、次式のように解くことができる。

【数１９】

$$[\tilde{a}_0 \quad \tilde{a}_1 \quad \tilde{a}_2]\begin{bmatrix}(-1)^0 & (-2)^0 & (-4)^0\\(-1)^1 & (-2)^1 & (-4)^1\\(-1)^2 & (-2)^2 & (-4)^2\end{bmatrix}=[X_{2N-1} \quad X_{2N-2} \quad X_{2N-4}]$$

$$[\tilde{a}_0 \quad \tilde{a}_1 \quad \tilde{a}_2]=[X_{2N-1} \quad X_{2N-2} \quad X_{2N-4}]\frac{1}{6}\begin{bmatrix}16 & 12 & 2\\-12 & -15 & -3\\2 & 3 & 1\end{bmatrix}$$

ここに、二次導関数の半分の負値は次式のようになる。

【数２０】

$$-\frac{1}{2}2a_2=-\frac{1}{2}2\tilde{a}_2=-\frac{1}{6}[X_{2N-1} \quad X_{2N-2} \quad X_{2N-4}]\begin{bmatrix}2\\-3\\1\end{bmatrix}=-\frac{2}{6}X_{2N-1}+\frac{3}{6}X_{2N-2}-\frac{1}{6}X_{2N-4}$$

【０１８８】

（付録Ｂ）

パイルへの導入部

並列プロセッサは、要求されるアルゴリズムが狭いデータ幅、直列的なデータ依存性、あるいは頻繁な制御文（例えば”if”、”for”、”while”文）を有する際には、高い処理速度（スループット）向けにプログラムすることが困難である。この具体例は、これら３つの問題を単独で、あるいは組み合わせて克服する。エントロピー符号化のアプリケーションは、これら３種類の問題をすべて有するアプリケーションの重要なクラスである。

【０１８９】

並列処理

プロセッサにおいて有利に使用可能な、次の３種類の並列化が存在する。

１）　第１の種類のものは、複数の機能ユニットによってサポート（支援）され、各機能ユニット内で処理を同時に進行させる。スーパースカラ・プロセッサアーキテクチャ及びＶＬＩＷ（Very Long Instruction Word：128ビット以上の命令長の並列処理）プロセッサアーキテクチャは、同一サイクル上で、いくつかの機能ユニットの各々に命令を発行することを可能にする。一般に、レイテンシ（待ち時間）あるいは完了時間は、一種類の機能ユニットと他の機能ユニットとで変化する。最も簡単な機能（例えばビット単位のAND）は通常１サイクルで完了するが、浮動小数点（フローティング）の加算機能は３サイクルまたはそれ以上を要する。

・　第２の種類の並列処理は、個々の機能のパイプライン化によってサポートされる。例えば、浮動小数点加算は完了に３サイクルを要し、３つの連続する副機能で実現することができ、各副機能は１サイクルを要する。副機能間のパイプライン・レジスタを設けることによって、１番目の浮動小数点加算が第２副機能を開始したサイクルと同じサイクル上で、２番目の浮動小数点加算の第１副機能を開始することができる。この手段によって、個々の浮動小数点加算が完了に３サイクルを要しても、すべてのサイクルで浮動小数点加算を開始及び終了することができる。

３）　利用可能な第３の種類の並列処理は、異なるワードのフィールド分割を、同じ計算の異なる瞬時に割り当てることである。例えば、32ビットのプロセッサ上の32ビットのワードを、各々が８ビットの４つのフィールド区分に分割する。データ・アイテムが８ビットに収まるほど十分に小さければ、これら４つの値すべてを同じ単一命令で処理することができる。

各単一サイクル中には、フィールド区分の数×機能ユニットの開始数の積に等しい数のデータ・アイテムを処理することができる。

【０１９０】
ループ・アンローリング
複数かつ／またはパイプライン化した機能ユニットをプログラムする従来の一般的な方法が存在し、同じ計算の多くの例を見出して、各例からの対応する演算をまとめて実行する。これらの例は、ループ・アンローリングの技法によって、あるいは同じ計算の他のソースによって生成することができる。
ループ・アンローリングは一般的に適用可能な技法であるが、特定例がその利点を学ぶ助けとなる。例えば、次のプログラムＡ）を考える。

for i＝0:1:255, {S(i)};

ここに、体S(i)は、ｉに依存する演算の列（シーケンス）{S_1(i); S_2(i); S_3(i); S_4(i); S_5(i);}であり、ｊ≠ｉであれば演算S(i)は演算S(j)とは完全に独立である。演算S_1(i); S_2(i); S_3(i); S_4(i); S_5(i);が互いに独立であると仮定してはならず、逆に、１つの演算から次の演算への依存性が（演算の）並べ替えを禁止すると仮定することはできる。
また、これらの同じ依存性が、前の演算が完了するまでは次の演算を開始しないことを要求する、と仮定することもできる。（パイプライン化した）演算の各々が完了に２サイクルを必要とするとすれば（パイプライン化した実行ユニットが各サイクルで新たな結果を生成しても）、上記５つの演算の列は完了に10サイクルを必要とする。これに加えて、ループ分岐は一般に、プログラミング・ツールがS_4(i)及びS_5(i)を分岐遅延と重複させることができなければ、ループ当たり３サイクルを必要とする。分岐遅延の重複ができれば、プログラムＡ）は完了に256/4×10＝640サイクルを必要とし、分岐遅延の重複ができなければ、完了に256/4×13＝832サイクルを必要とする。
次のプログラムＢ）

for n＝0:4:255, {S(n); S(n+1); S(n+2); S(n+3);};

はプログラムＡ）と完全に等価である。ループは４回「アンロール（展開）」されている。このことは、高価な制御フロー変化を４分の１に低減する。より重要なこととして、このプログラムは、４つの構成演算S(i)の各々を並べ替える機会を提供する。従って、プログラムＡ）及びＢ）は次のプログラムＣ）と等価である。

for n＝0:4:255, {S_1(n); S_2(n); S_3(n); S_4(n); S_5(n);
S_1(n+1); S_2(n+1); S_3(n+1); S_4(n+1); S_5(n+1);
S_1(n+2); S_2(n+2); S_3(n+2); S_4(n+2); S_5(n+2);
S_1(n+3); S_2(n+3); S_3(n+3); S_4(n+3); S_5(n+3);
};

上述した依存性及び独立（非依存）性についての仮定を以ってすれば、次の等価なプログラムＤ）を作成することができる。

for n＝0:4:255, {S_1(n); S_1(n+1); S_1(n+2); S_1(n+3);
S_2(n); S_2(n+1); S_2(n+2); S_2(n+3);
S_3(n); S_3(n+1); S_3(n+2); S_3(n+3);
S_4(n); S_4(n+1); S_4(n+3); S_4(n+3);
S_5(n); S_5(n+1); S_5(n+3); S_5(n+3);
};

１番目のサイクルには、S_1(n); S_1(n+1);を発行することができ、２番目のサイクルには、S_1(n+2); S_1(n+3);を発行することができる。３番目のサイクルの開始時には、S_1(n); S_1(n+1);を完了し（２サイクルが経過している）、従ってS_2(n); S_2(n+1);を発行することができる。従ってプログラムＤ）は次のように進む：これに続く各サイクルにおいて、次の２つの演算を発行することができ、プログラム全体は同じ10サイクルで実行することができる。プログラムＤ）は、プログラムＡ）の４分の１未満の時間で動く。
最も並列的なプロセッサは必然的に条件分岐命令を有し、この条件分岐命令は、命令そのものと分岐が実際に行われる点との間に数サイクルの遅延を必要とする。この遅延期間中に、他の命令を実行することができる。分岐条件が十分事前に既知であり、そしてコン

パイラまたは他のプログラミング・ツールが前記遅延期間中の命令の実行をサポートする限りは、この分岐のコストは、１つの命令を発行する機会と同じくらい少ない。この技法はプログラムＡ）にも適用することができる、というのは、ループの最上部において分岐条件（ｉ＝255）が既知だからである。

過剰なアンローリングは生産性に反する。第１には、一旦、（プログラムＤにおけるように）すべての発行の機会を利用すると、追加的なアンローリングによるさらなる速度向上がなくなる。第２には、アンローリングしたループのターン（周回）の各々が、一般に、特定のターンについての状態を保持するための追加的なレジスタを必要とする。必要なレジスタ数は、アンローリングしたターンの数に正比例する。必要なレジスタの総数が利用可能な数を超えれば、一部のレジスタ（の内容）をキャッシュに「流出」させて、次のループのターン時に復帰させなければならない。ループのアンローリングが結局速度向上にならなければ、この流出及び再ロード（復帰）をサポートするために発行する必要のある命令が、プログラムの時間を長くする。こうしたループをアンロール（展開）する回数には最適値が存在する。

【０１９１】

例外処理を含むループのアンローリング

ここで、次のプログラムＡ’）を考える。

For Ｉ＝0:1:255, {Ｓ(i); if Ｃ(i) then Ｔ(Ｉ(i)) };

ここに、Ｃ(i)は、真であることの少ない（例えば1/64）例外条件であり、Ｓ(i)のみに依存し、Ｔ(Ｉ(i))は、例えば1024演算（オペレーション、命令）の長い例外処理である。Ｉ(i)はＳ(i)によって計算する情報であり、例外処理に必要である。例えば、Ｔ(Ｉ(i))が、プログラムＡ）における各ループ・ターンに、平均的に16演算を加えるものとし、この量は、ループの本体の4演算を超える。こうした、まれであるが長い例外処理は、プログラムに共通の問題である。アンローリングの利点を損なうことなくこの問題を取り扱う方法について以下に説明する。

【０１９２】

ガード命令

１つの方法はガード（保護）命令の使用によるものであり、これらの命令は多くのプロセッサ上で利用可能な装備である。ガード命令は、追加的なオペランドとしてブール代数値を指定し、この命令は想定される機能ユニットを常に占有するが、ガードが失われれば結果の保持が停止されるという意味を伴う。

If－then－else構文を実現するに当たり、ガードがif条件であると解釈する。Then節（クローズ）の命令がif条件によって保護されて、else節の命令がif条件の否定によって保護される。いずれの場合にも両方の節を実行する。ガードが真となる場合のみにthen節の結果によって更新される。ガードが偽となる場合のみにelse節の結果によって更新される。すべての場合に両方の節の命令を実行して、制御フローにおける条件変化によって要求されるパイプライン遅延の不利益（ペナルティ）よりは、こうした（両方の節を実行する）不利益を受忍する。

このガードの方法は、プログラムＡ’）のように、ガードが真であることが圧倒的に多く、かつelse節が大きければ、大きな不利益をこうむる。この場合には、大きなelse節は少数の場合のみに関係するにもかかわらず、すべての場合に大きなelse節を実行する不利益をこうむる。条件Ｃによってガード（保護）すべき演算Ｓがある場合には、このことを次のようにプログラムすることができる。

Guard(Ｃ, Ｓ);

【０１９３】

最初のアンローリング

プログラムＡ’）は、次のプログラムＤ’）にアンロールすることができる。

for n=0:4:255, {S_1(n); S_1(n+1); S_1(n+2); S_1(n+3);
S_2(n); S_2(n+1); S_2(n+2); S_2(n+3);
S_3(n); S_3(n+1); S_3(n+2); S_3(n+3);

```
S4(n); S4(n+1); S4(n+3); S4(n+3);
S5(n); S5(n+1); S5(n+3); S5(n+3);
if C(n) then T(I(n));
if C(n+1) then T(I(n+1));
if C(n+2) then T(I(n+2));
if C(n+3) then T(I(n+3));
};
```

上記の例のパラメータにおいて、ループ・ターンの77%ではT(I(n))が実行されず、ループ・ターンの21%ではT(I(n))が１回実行され、T(I(n))が２回以上実行されるのは、ループ・ターンの2%に過ぎない。演算T(I(n))、T(I(n+1))、T(I(n+2))、及びT(I(n+3))を入れ替えることによって得られるものはわずかであることは明らかである。

【０１９４】

パイル処理

新たな代替法はパイル処理である。パイルとは、一般にＲＡＭに記憶される連続的な記憶対象（シーケンシャル・メモリー・オブジェクト）である。パイルは、連続的に書き込まれ、先頭から連続的に読み出されることを意図している。パイル・オブジェクトについて多くの方法が規定されている。

並列処理環境において実用的なパイル及びパイルを扱う方法については、パイルの実現はインライン・コード（サブルーチンへの戻り分岐のない）の少数の命令であることが要求される。このインライン・コードが分岐命令を含まないことも要求される。こうした方法の実現は以下に説明する。こうした実現の可能性が、パイルを新規で価値あるものにする。

１）　パイルは、方法Create_Pile(Ｐ)によって作成する。この方法は、記憶装置を割り当てて、内部状態変数を初期化する。

２）　パイルを書き込むための主要な方法はConditional_Append(pile, condition, record)である。この方法は、condition（という条件）が真である場合のみに、record（というパラメータ値）をpileというパイルに追加する。

３）　パイルを完全に書き込むと、方法Rewind_Pile(Ｐ)によって、読出し準備完了となる。このことは、書き込んだ最初のレコード（記録）から読出しが始まるように、内部変数を調整する。

４）　方法EOF(Ｐ)は、パイルのすべてのレコードを読み出したか否かを示すブール代数値を生成する。

５）　方法Pile_Read(Ｐ, record)は、次のシーケンシャル・レコード（順次記録）をパイルＰから読み出す。

６）　方法Destroy_Pile(Ｐ)は、パイルＰのすべての状態変数を、（記憶装置の）割り当て解除することによって、パイルＰを破壊する。

【０１９５】

パイルを用いて条件処理を分割する

パイルＰによって、プログラムＤ’）をプログラムＥ’）に変換することができる。

```
Create_Pile(P);
for n=0:4:255, {S1(n); S1(n+1); S1(n+2); S1(n+3);
S2(n); S2(n+1); S2(n+2); S2(n+3);
S3(n); S3(n+1); S3(n+2); S3(n+3);
S4(n); S4(n+1); S4(n+3); S4(n+3);
S5(n); S5(n+1); S5(n+3); S5(n+3);
Conditional_Append(P, C(n), I(n));
Conditional_Append(P, C(n+1), I(n+1));
Conditional_Append(P, C(n+2), I(n+2));
Conditional_Append(P, C(n+3), I(n+3));
};
```

```
Rewind(Ｐ);
While not EOP(Ｐ) {
Pile_Read(Ｐ, Ｉ);
Ｔ(Ｉ);
};
Destroy_Pile(Ｐ);
```

プログラムＥ’）は、パイルＰ上での例外演算Ｔに必要な情報Ｉを保存することによって動作する。例外条件Ｃ(n)に対応するＩのレコードだけを書き込み、このため、Ｐ内のＩのレコード数（例えば16）は、元のプログラムＡ）中のループ・ターン数（例えば256）よりもずっと少ない。その後に、独立した”while”ループがパイルＰを読み通して、すべての例外計算Ｔを実行する。Ｃ(n)が真であった場合についてのみ、Ｐがレコード Ｉを含むので、これらの場合のみが処理される。

２番目のループは１番目のループよりも少し扱いにくい、というのは、２番目のループのターン数が、この例では平均16であるが、中途半端だからである。従って、”for”ループよりもむしろ”while”ループが必要であり、方法EOFが、すべてのレコードをパイルから読み出したことを示すと、終了する。

以上及び以下に記述するように、方法Conditional_Appendの起動は、インラインかつ分岐なしで実現することができる。このことは、１番目のループが、非生産的な少数の発行の機会を有して、効果的な方法でまだアンロールされていることを意味する。

【０１９６】

２番目のループのアンローリング

プログラムＥ’）中の２番目のループはアンロールされておらず、まだ非効率である。しかし、プログラムＥ’）は、パイル$Ｐ_1$、$Ｐ_2$、$Ｐ_3$、$Ｐ_4$によって次のプログラムＦ’）に変換することができる。その結果は、Ｆ’）が、効率の改善を伴って両方のループをアンロールする、ということである。

```
Create_Pile(Ｐ1); Create_Pile(Ｐ2); Create_Pile(Ｐ3); Create_Pile(Ｐ4)
;
for n=0:4:255, {Ｓ1(n); Ｓ1(n+1); Ｓ1(n+2); Ｓ1(n+3);
Ｓ2(n); Ｓ2(n+1); Ｓ2(n+2); Ｓ2(n+3);
Ｓ3(n); Ｓ3(n+1); Ｓ3(n+2); Ｓ3(n+3);
Ｓ4(n); Ｓ4(n+1); Ｓ4(n+2); Ｓ4(n+3);
Ｓ5(n); Ｓ5(n+1); Ｓ5(n+2); Ｓ5(n+3);
Conditional_Append(Ｐ1, Ｃ(n), Ｉ(n));
Conditional_Append(Ｐ2, Ｃ(n+1), Ｉ(n+1));
Conditional_Append(Ｐ3, Ｃ(n+2), Ｉ(n+2));
Conditional_Append(Ｐ4, Ｃ(n+3), Ｉ(n+3));
};
Rewind(Ｐ1); Rewind(Ｐ2); Rewind(Ｐ3); Rewind(Ｐ4);
While not all EOF(Pi) {
Pile_Read(P1,I1); Pile_Read(P2,I2);
Pile_Read(P3,I3); Pile_Read(P4,I4);
Guard(not EOF(P1), S); T(I1);
Guard(not EOF(P2), S); T(I2);
Guard(not EOF(P3), S); T(I3);
Guard(not EOF(P4), S); T(I4);
};
Destroy_Pile(P1); Destroy_Pile(P2); Destroy_Pile(P3); Destroy_Pile(P4)
;
```

プログラムＦ’）は２番目のループをアンロールしたプログラムＥ’）である。このアンローリングは、プログラムＥ’）の単一のパイルを、各々が互いに無関係に処理可能な４

つのパイルに分割することによって達成される。プログラムF’）中の２番目のループの各ターンは、これら４つのパイルの各々からの１レコードを処理する。各レコードを独立して処理するので、各Tの演算は、他の３つのTの演算と並べ替えることができる。

すべてのパイルを処理するまでは、”while”ループの制御を”to”ループに修正しなければならない。そして、一般に、すべてのパイルが同じループ・ターンで完了するわけではないので、”while”ループ中の演算Tをガードしなければならない。２つのパイル中のレコード数が互いに大幅に異なる際に常にある程度の非効率が存在するが、確率論（大数の法則）によれば、これらのパイルは似たようなレコード数を含む。

もちろん、このパイル化技法は反復的に適用することができる。Tそのものが長い条件節T’を含む場合には、いくつかの追加的なパイルを以って２番目のループからT’を分割して、３番目のループをアンロール（展開）することができる。実際のアプリケーションの多くは、このようなネスト（入れ子）にされた例外節（クローズ）をいくつか有する。

【０１９７】

パイル処理の実現

パイル・オブジェクト及びその方法の実現は、上述した実現基準を満足するために、簡単さを保たなければならない。

ａ）　方法の実現は、Create_Pile及びDestroy_Pileを除いて、インライン・コードの少数の命令のみにしなければならない。

ｂ）　この実現は、分岐命令を含まないべきである。

パイルの心臓部は、ＲＡＭ内に割り当てたリニア・アレイ、及びポインタ”index”から成り、ポインタの現在値は、次に読出しまたは書込みを行うべき記録の位置である。このアレイの書込みサイズ”sz”はポインタであり、その値は、パイルの書込み中の”index”の最大値である。方法EOFは、インライン条件文（sz＜index）として実現することができる。ポインタ”base”は、パイルに書き込む最初の位置を示す値を有する。この値は、方法Create_Pileによって設定される。

方法Conditional_Appendは、値”index”から始まるパイルのアレイにレコードをコピーする。そして、計算した量だけ”index”を増加させて、この計算した量は０かレコードのサイズ（sz_record）のいずれかである。パラメータ”condition”が、真に対して１の値、偽に対して０の値を有するので、”index”は分岐なしで、次式のように計算することができる。

index＝index＋condition×sz_record;

もちろん、この計算には多くの変形が存在し、それらの多くは、変数の特別な値を与えられる乗算を含まない。この計算は、次のガードを用いて計算することもできる。

guard(condition, index＝index＋sz_record);

なお、レコードは”condition”とは無関係にパイルにコピーされる。”condition”が偽であれば、このレコードはすぐ次のレコードによって上書きされ、”condition”が真であれば、すぐ次のレコードは現在のレコードに続けて書き込まれる。この次のレコード自体は、その後のレコードによって上書きされることもそうでないこともあり得る。結果として、たとえ、レコードを読み出して処理する際にいくらかの（冗長な）データを再計算することになっても、パイルにできる限り少なく書き込むことが一般に最適である。

方法Rewindは、sz＝index; 及び index＝base; によって簡単に実現することができる。この演算は、方法EOF用に書き込んだデータ量を記録し、そして”index”を先頭の値にリセットする。

方法Pile_Readは、次式のように、（長さsz_recordの）パイルの次の部分をＩにコピーして、”index”を増加させる。

index＝index＋sz_record;

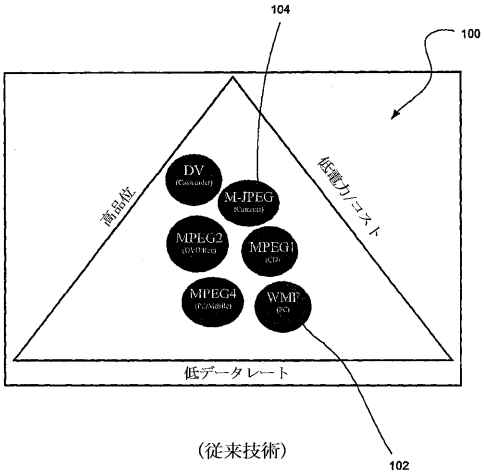
Destroy_Pileは、パイルに割り当てた記憶装置を解放する。

（Create_Pile及びDestroy_Pileを除いた）これらの方法のすべてが、少数のインライン命令で、かつ分岐なしで実現することができる。

こうして、パイル処理は、ループのアンローリングを可能にし、その結果、分岐が存在

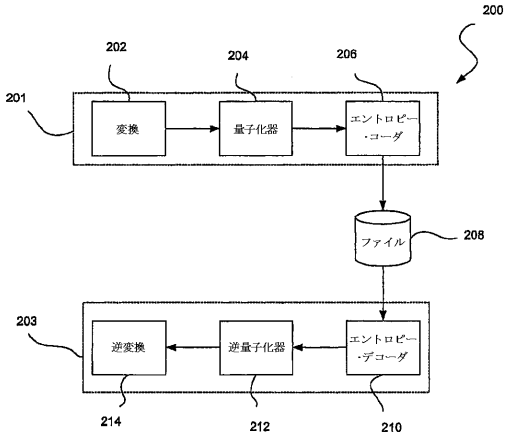
する際の性能改善を可能にする。この技法は特に、長い例外節（クローズ）の並列実行を可能にする。このためのコストは、少量のデータをＲＡＭに書き込んで再び読み出す要求程度である。

【図１】

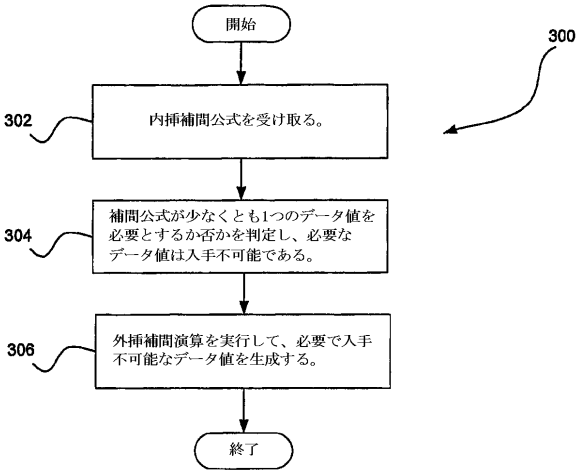


（従来技術）

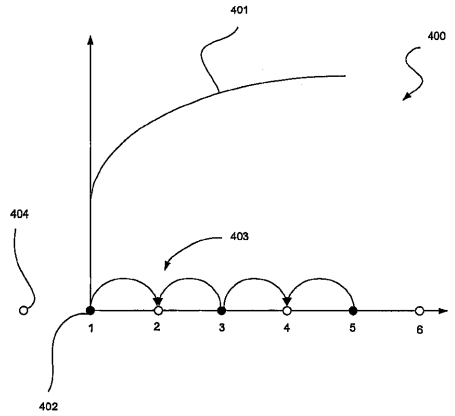
【図２】



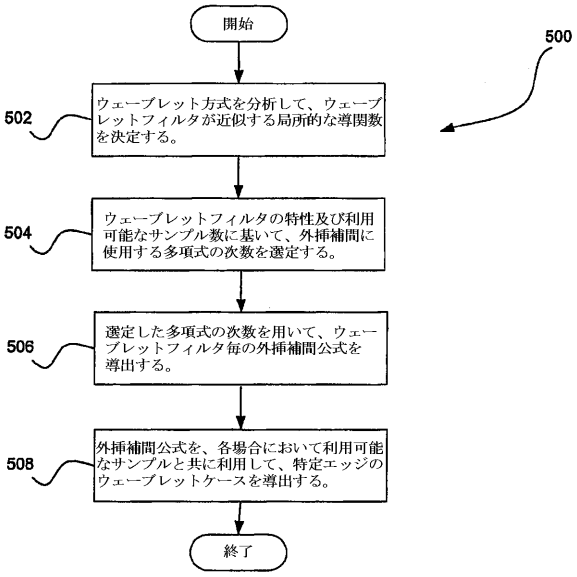
【図３】


開始
300
302
内挿補間公式を受け取る。
304
補間公式が少なくとも1つのデータ値を必要とするか否かを判定し、必要なデータ値は入手不可能である。
306
外挿補間演算を実行して、必要で入手不可能なデータ値を生成する。
終了

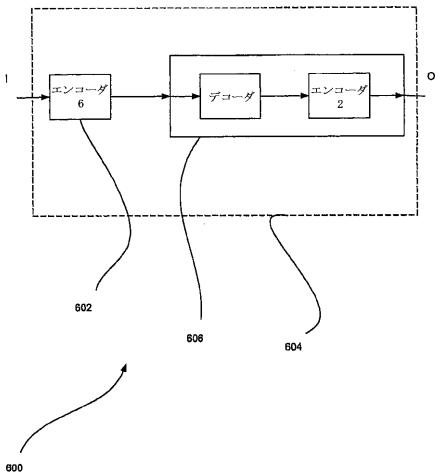
【図４】


401
400
404
403
1
2
3
4
5
6
402

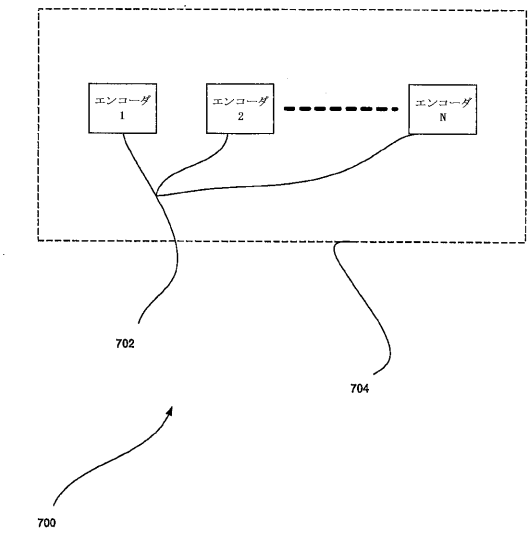
【図５】


開始
500
502
ウェーブレット方式を分析して、ウェーブレットフィルタが近似する局所的な導関数を決定する。
504
ウェーブレットフィルタの特性及び利用可能なサンプル数に基いて、外挿補間に使用する多項式の次数を選定する。
506
選定した多項式の次数を用いて、ウェーブレットフィルタ毎の外挿補間公式を導出する。
508
外挿補間公式を、各場合において利用可能なサンプルと共に利用して、特定エッジのウェーブレットケースを導出する。
終了

【図６】


I
エンコーダ 6
デコーダ
エンコーダ 2
O
602
606
604
600

【図７】



【手続補正書】
【提出日】平成22年3月24日(2010.3.24)
【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】
単一モジュールを利用してデータを圧縮する方法において、
回路を含む前記単一モジュールを利用して光子を受け取るステップと；
前記単一モジュールを利用して、前記光子を表現する電子的に圧縮したデータを出力するステップと
を具え、
前記データを、少なくとも１つのエンコーダを利用して圧縮することを特徴とするデータ圧縮方法。
【請求項２】
前記圧縮したデータを、ウェーブレットベースのフォーマットに符号化することを特徴とする請求項１に記載の方法。
【請求項３】
前記符号化に関連する少なくとも１つの変換操作を、アナログで実行することを特徴とする請求項２に記載の方法。
【請求項４】
前記単一モジュールが撮像素子を含むことを特徴とする請求項１に記載の方法。
【請求項５】

単一モジュールを利用してデータを圧縮する方法において、
前記単一モジュールを利用して光子を受け取るステップと；
前記単一モジュールを利用して、前記光子を表現するデータを出力するステップと
を具え、
前記圧縮に関連する少なくとも１つの変換操作を、アナログで実行することを特徴とするデータ圧縮方法。
【請求項６】
前記圧縮したデータを、ウェーブレットベースのフォーマットに符号化することを特徴とする請求項５に記載の方法。
【請求項７】
前記単一モジュールが撮像素子を含むことを特徴とする請求項５に記載の方法。

フロントページの続き

(31)優先権主張番号 60/374,069
(32)優先日 平成14年4月19日(2002.4.19)
(33)優先権主張国 米国(US)
(31)優先権主張番号 60/385,254
(32)優先日 平成14年5月28日(2002.5.28)
(33)優先権主張国 米国(US)
(31)優先権主張番号 60/390,383
(32)優先日 平成14年6月21日(2002.6.21)
(33)優先権主張国 米国(US)
(31)優先権主張番号 60/390,380
(32)優先日 平成14年6月21日(2002.6.21)
(33)優先権主張国 米国(US)

(72)発明者 ウィリアム シー リンチ
アメリカ合衆国 カリフォルニア州 ９４３０３ パロ アルト トーマス ドライヴ ３３３１
(72)発明者 スティーヴン イー ソーンダース
アメリカ合衆国 カリフォルニア州 ９５０１４ カペルティーノ シャディーグローヴ ドライヴ ６０６９
(72)発明者 トーマス エイ ダーボン
アメリカ合衆国 カリフォルニア州 ９５０６５ サンタ クルズ グラニテ クリーク ロード １８４７

Ｆターム(参考) 5C159 KK03 KK52 KK53 KK61 LB05 LB11 MA00 MA23 MA41 MA42
MC11 MC38 ME01 PP15 PP16 SS14 UA02 UA05 UA15
5C178 AC07 BC52 BC62 BC93 CC67 DC03
5J064 AA01 AA02 BA09 BA16 BB04 BC01 BC16 BD02 BD03

【外国語明細書】
2010141922000001.pdf