(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第3681386号

(P3681386)

(45)発行日　平成17年8月10日(2005.8.10)　(24)登録日　平成17年5月27日(2005.5.27)

(51)Int.Cl.[7]	F I		
H04N 7/24	H04N	7/13	Z
H03M 7/30	H03M	7/30	B
H04B 14/04	H04B	14/04	Z
H04N 1/41	H04N	1/41	B
H04N 11/04	H04N	11/04	Z

請求項の数 10　(全 31 頁)

(21)出願番号　特願平6-507162
(86)(22)出願日　平成5年6月14日(1993.6.14)
(65)公表番号　特表平8-502392
(43)公表日　平成8年3月12日(1996.3.12)
(86)国際出願番号　PCT/US1993/005675
(87)国際公開番号　WO1994/006098
(87)国際公開日　平成6年3月17日(1994.3.17)
審査請求日　平成12年6月14日(2000.6.14)
(31)優先権主張番号　07/938,942
(32)優先日　平成4年9月1日(1992.9.1)
(33)優先権主張国　米国(US)

(73)特許権者
アプル・コンピュータ・インコーポレーテッド
アメリカ合衆国　９５０１４　カリフォルニア州・カッパチーノ・マリアニイ　アヴェニュ・２０５２５
(74)代理人
弁理士　山川　政樹
(74)代理人
弁理士　黒川　弘朗
(74)代理人
弁理士　紺野　正幸
(74)代理人
弁理士　西山　修

最終頁に続く

(54)【発明の名称】改良されたベクトルの量子化のための前処理及び後処理

(57)【特許請求の範囲】

【請求項１】

ａ．　空間的冗長度を除去し、上記空間的冗長度から解放された第２のビット・ストリームを生成するために、画素に対して方向性フィルタリングを行ってイメージを表す第１のビット・ストリームを事前処理するステップと、

ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用するステップと、

ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームに対してインデックス・パッキングを行うステップと、を含むイメージをコード化する方法であって、

上記第４のビット・ストリームは、（１）上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと（２）上記コードブックの項目を参照するため、上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値とを含むようにして、上記各インデックスを、（１）開始点としての単一のベース・アドレスと（２）このベース・アドレスからのオフセット値を用いて表すようにしたことを特徴とするイメージをコード化する方法。

【請求項２】

ａ．　時間的冗長度を除去し、上記時間的冗長度から解放された第２のビット・ストリー

ムを生成するために、イメージの第１のベクトルと前のイメージの第２のベクトルとの同一の空間位置での差が所定の閾値より小さいか否かを判定し、小さい場合、第１のベクトルに無変化ベクトルとしてタグを付け、ベクトルの量子化が行われるべきでないことを示すように、イメージを表す第１のビット・ストリームを事前処理するステップと、

ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記タグが付されていない上記第２のビット・ストリームに対してベクトルの量子化を適用するステップと、

ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームに対してインデックス・パッキングを行うステップと、を含むイメージをコード化する方法であって、

上記第４のビット・ストリームは、（１）上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと（２）上記コードブックの項目を参照するため、上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値とを含むようにして、上記各インデックスを、（１）開始点としての単一のベース・アドレスと（２）このベース・アドレスからのオフセット値を用いて表すようにしたことを特徴とするイメージをコード化する方法。

【請求項３】

ａ．　空間的冗長度を除去するための、水平フィルタリング方法と垂直フィルタリング方法と下向き対角線フィルタリング方法と上向き対角線フィルタリングの方法を含む複数のフィルタリング方法のうち、どのフィルタリング方法が上記イメージを最も良く保持しているかを決定するステップと、

ｂ．　空間的冗長度から解放された第２のビット・ストリームを生成するために、上記決定されたフィルタリング方法を用いてイメージを表す第１のビット・ストリームを事前処理するステップと、

ｃ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用するステップと、

ｄ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームに対してインデックス・パッキングを行うステップと、を含むイメージをコード化する方法であって、

上記第４のビット・ストリームは、（１）上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと（２）上記コードブックの項目を参照するため、上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値とを含むようにして、上記各インデックスを、（１）開始点としての単一のベース・アドレスと（２）このベース・アドレスからのオフセット値を用いて表すようにしたことを特徴とするイメージをコード化する方法。

【請求項４】

ａ．　空間的冗長度を除去し、上記空間的冗長度から解放された第２のビット・ストリームを生成するために、画素に対して方向性フィルタリングを行ってイメージを表す第１のビット・ストリームを事前処理する手段と、

ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用する手段と、

ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームに対してインデックス・パッキングを行う手段と、を含むイメージをコード化する装置であって、

上記第４のビット・ストリームは、（１）上記コードブックにおける上記第３のビット・

ストリームの複数のインデックスに対する単一のベース・アドレスと（２）上記コードブックの項目を参照するため、上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値とを含むようにして、上記各インデックスを、（１）開始点としての単一のベース・アドレスと（２）このベース・アドレスからのオフセット値を用いて表すようにしたことを特徴とするイメージをコード化する装置。

【請求項５】

ａ．　時間的冗長度を除去し、上記時間的冗長度から解放された第２のビット・ストリームを生成するために、イメージの第１のベクトルと前のイメージの第２のベクトルとの同一の空間位置での差が第１の閾値より小さいか否かを判定し、小さい場合、第１のベクトルに無変化ベクトルとしてタグを付け、ベクトルの量子化が行われるべきでないことを示すように、イメージを表す第１のビット・ストリームを事前処理する手段と、

ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記タグが付されていない上記第２のビット・ストリームに対してベクトルの量子化を適用する手段と、

ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームに対してインデックス・パッキングを行う手段と、を含むイメージをコード化する装置であって、

上記第４のビット・ストリームは、（１）上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと（２）上記コードブックの項目を参照するため、上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値とを含むようにして、上記各インデックスを、（１）開始点としての単一のベース・アドレスと（２）このベース・アドレスからのオフセット値を用いて表すようにしたことを特徴とするイメージをコード化する装置。

【請求項６】

ａ．　空間的冗長度を除去するための複数の方向性フィルタのうち、どのフィルタがイメージを表す第１のビット・ストリームのベクトルと比較した最小絶対誤差を生ずるかを決定する手段と、

ｂ．　空間的冗長度から解放された第２のビット・ストリームを生成するために、上記決定されたフィルタを用いて上記イメージを表す第１のビット・ストリームを事前処理する手段と、

ｃ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用する手段と、

ｄ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームに対してインデックス・パッキングを行う手段と、を含むイメージをコード化する装置であって、

上記第４のビット・ストリームは、（１）上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと（２）上記コードブックの項目を参照するため、上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値とを含むようにして、上記各インデックスを、（１）開始点としての単一のベース・アドレスと（２）このベース・アドレスからのオフセット値を用いて表すようにしたことを特徴とするイメージをコード化する装置。

【請求項７】

ａ．　空間的冗長度を除去するために、イメージの隣接画素ブロックを含む個々のベクトルを個々のベクトルで表現された画素の輝度に基づいて選択的にサブ・サンプリングすることによって、イメージを表す第１のビット・ストリームを事前処理して、ベクトル量子化が行われる第２のビット・ストリームを生成するステップと、

ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ス

トリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用するステップと、
ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するため、上記第３のビット・ストリームのインデックスに対してインデックス・パッキングを行うステップと、を含み、
上記第４のビット・ストリームは、上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと上記コードブックの項目を参照するため上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値を含む、イメージをコード化する方法。
【請求項８】
ａ．　時間的冗長度を除去し、ベクトル量子化が行われる第２のビット・ストリームを生成するために、同一の空間位置での、イメージの第１のベクトルと前のイメージの第２のベクトル間の差が、調整された閾値より小さいか否かを判定し、小さい場合は、上記第１のベクトルに無変化ベクトルとして量子化が第１のベクトルに対して行われるべきでないことを示すタグを付けるように、イメージを表す第１のビット・ストリームを事前処理するステップと、
ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対して上記第１のベクトルにタグが付けられているか否かに従ってベクトルの量子化を適用するステップと、
ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するため、上記第３のビット・ストリームのインデックスに対してインデックス・パッキングを行うステップと、を含み、
上記第４のビット・ストリームは、上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと上記コードブックの項目を参照するため上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値を含む、イメージをコード化する方法。
【請求項９】
ａ．　ベクトル量子化が行われる第２のビット・ストリームを生成するために、イメージの局所的なイメージ特性に基づいて、複数の方向性フィルタリング方法の中のいずれのフィルタリング方法が所与のレート制御要求でイメージのディテイルを最も良く保持し、かつ、イメージの空間的冗長度を除去しているかを決定するステップと、
ｂ．　空間的冗長度から解放された第２のビット・ストリームを生成するために、上記決定されたフィルタリング方法を用いて第１のビット・ストリームを事前処理するステップと、
ｃ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用するステップと、
ｄ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するために、上記第３のビット・ストリームのインデックスに対してインデックス・パッキングを行うステップと、を含み、
上記第４のビット・ストリームは、上記コードブックにおける上記第３のビット・ストリームの複数のインデックスに対する単一のベース・アドレスと上記コードブックの項目を参照するため上記単一のベース・アドレスからの、複数のインデックスの各々に対する関連の各オフセット値を含む、イメージをコード化する方法。
【請求項１０】
ａ．　時間的冗長度を除去し、ベクトル量子化が行われる第２のビット・ストリームを生成するために、同一の空間位置での、イメージの第１のベクトルと前のイメージの第２の

ベクトル間の差が、調整された閾値より小さいか否かを判定し、小さい場合は、上記第１のベクトルに無変化ベクトルとして量子化が第１のベクトルに対して行われるべきでないことを示すタグを付けるように、イメージを表す第１のビット・ストリームを事前処理する手段と、

ｂ．　コードブックと上記コードブックを参照するインデックスを含む第３のビット・ストリームを作成するために、上記第２のビット・ストリームに対してベクトルの量子化を適用する手段と、

ｃ．　上記コードブックにおけるベース・アドレスに対する参照と上記ベース・アドレスからのオフセット値を含む第４のビット・ストリームを生成するため、上記第３のビット・ストリームのインデックスに対してインデックス・パッキングを行う手段と、

を含み、

上記第４のビット・ストリームは、上記コードブックにおける上記第３のビット・ストリームの複数のパッキングされたインデックスに対する単一のベース・アドレスと上記コードブックの項目を参照するため上記単一のベース・アドレスからの、複数のパッキングされたインデックスの各々に対する関連の各オフセット値を含む、イメージをコード化する装置。

【発明の詳細な説明】

発明の背景

１．発明の分野

本発明はビデオの圧縮、圧縮解除に関連し、特にイメージの事前処理及びベクトルの量子化（ＶＱ）を用いた改良されたビデオ圧縮／圧縮解除に関連する。

２．関連技術の背景

マルチメディアや完全動画ビデオを必要とするその他のアプリケーションのような近代的アプリケーションは、ビデオ情報の記憶、伝送及び表示に費やされる処理帯域幅を低減するビデオ圧縮の標準の開発を必要とした。これは、高解像度のフル・イメージのビデオ情報を表す伝送、記憶のための大量のデータのためである。一般に図１ａ、１ｂ及び１ｃに示すような装置は、ベクトルの量子化技術に基づく入力イメージの圧縮、圧縮解除を行うために用いられる。例えば、図１ａに示すように、イメージ１００は、冗長度を低減するため、あるいは入力イメージ１００に含まれるデータ量を低減するため、入力イメージ又は一連のイメージに対する空間的又は時間的事前処理を適用するエンコーダ１０１に入力される。エンコーダ１０１は元のイメージ１００よりも実質的に小さい圧縮されたイメージ１０２を生成する。ある種の従来技術システムでは、エンコーダ１０１は入力イメージ１００の画素パターンをマッチングさせるために用いられるコードブック１０５を用い、画素パターンが圧縮されたイメージ１０２の中で別の画素パターンにマップされるようにする。このようにして、特定の色やグラフィックの情報を送らずに、イメージ内の各領域は、インデックスでコードブックのエレメントを参照することによってアドレスされる。ある種の従来技術のアプリケーションでは、圧縮イメージでの画質は損なわれるが、イメージ１００から圧縮イメージ１０２へのイメージ・サイズの低減によって実質的節約が得られている。他の圧縮技術は「無損失（ロス・レス）」であり、追加の計算時間や大きなビット・ストリームというコストはかかるが、一般にデコードされたイメージに画質の低下はない。

逆に、圧縮解除されたイメージ１３２を生成するため、図１ｂに示すように圧縮されたイメージ１０２がデコーダ１３１に加えられる。再び、デコーダ１３１はコードブック１０５を用いて圧縮されたイメージ１０２に含まれるインデックスから、イメージ１３２に現れる画素パターンを決定する。デコーダ１３１はイメージをコード化するのに用いられた同じコードブック１０５を用いる必要がある。一般に従来技術のシステムではコードブックはコンピュータ・システムで表示するために圧縮又は圧縮解除されるイメージ又はイメージのセットに関連して固有である。

一般に１０５のようなコードブックはコードブック発生器１５２に加えられるイメージ又はイメージのトレーニング・セット１５１から生成される。コードブックは圧縮される１

つ又は多くのイメージに対して特に生成され、そのコードブックは生成されたイメージをデコードするのに用いられる。コードブックは更に、将来コード化される一連のイメージの妥当な統計的表現である長いトレーニング・シーケンスに対して最適化することによって生成できる。このトレーニング・コードブックは大きな範囲のイメージ特性を表すと考えられる。このトレーニング・コードブックはしばしばエンコーダ及びデコーダで固定化されるが、コードブックの部分部分は順応して改良され得る。ある種の従来技術の体系では、コードブック発生器１５２及びエンコーダ１０１は一体となっている。コード化はコードブックの生成と同時に行われ、コードブックはトレーニング・イメージからでなくコード化されたイメージから導かれる。

図２はイメージのコード化及びデコードに対するベクトルとして知られる別々の領域にイメージ２００がどのように区画化されるかを示す。１つの従来技術のアプローチでは、２００のようなイメージは、「ベクトル」として知られる２０１及び２０２のような一連の２×２画素ブロックに分割される。２０１のような各ベクトルは４つの画素２０１a、２０１b、２０１c及び２０１dから成る。イメージがこのようなベクトルに分解されると、ビット・ストリーム内の各ベクトルを使って、（a）コードブックの生成を含むイメージのコード化を行い、（b）イメージのデコードを行う。イメージ２００における２０１、２０２のような各ベクトルは、イメージ２００を表すのに用いられる。イメージに含まれるベクトルの近似であるコードブックのエレメントを参照することによって１つのイメージが表される。従って、２０１aから２０１dのような４つの別々の画素を用いてイメージを表す代わりに、ベクトル２０１に含まれる情報を近似するコードブック・インデックスを参照してイメージが表される。コードブック内の項目数によって、イメージ・ベクトルを参照するコードブック・インデックスを使用すると、実際の画素値２０１a－２０１dを用いてイメージを表すのではないため、ベクトルを表す記憶域を実質的に低減できる。

このような従来技術の装置は、図１aから１cを参照して論じたようにＣＯＤＥＣ（コード化／デコード）として知られる装置に実施されており、これは対応するコードブックからの一連のイメージに対する圧縮されたビット・ストリームを生成し、コードブックを用いて後でイメージの圧縮解除を行う。例えば、このようなＣＯＤＥＣは図３に装置３００として示されている。ＣＯＤＥＣ３００は２つの部分から成る。すなわちエンコーダ３０１とデコーダ３５１である。エンコーダ３０１は入力データ３１０としてビデオ、音声その他圧縮したいデータを受け取る。しかし、ビデオのコード化／デコードを論ずるこの出願の残余のために、同様の体系が他のタイプのデータにも適用できることを当業者は理解されたい。このような入力データはプリ・プロセッサ３２０に与えられ、コード化／デコードを簡単なタスクにするために特定のパラメータが調整されてデータを事前処理する。プリ・プロセッサ３２０は、イメージをある方法でコード化するためにベクトルの量子化を用いるベクトル量子化装置３３０に供給し、冗長度を等しく低減する。次にベクトル量子化装置３３０はパック／コード化処理３４０に出力し、更にビット・ストリームを圧縮する。レート制御メカニズム３４５は圧縮されたビット・ストリーム３５０のサイズに関する情報を受け取り、所望のデータ・レートを達成するため、様々なパラメータがプリ・プロセッサ３２０内で調整される。更に、プリ・プロセッサ３２０はコード化されたビット・ストリームをサンプリングし、画質のセッティングを調整する。

ＣＯＤＥＣ３００は更に、コードブック再生器３６０を用いて圧縮されたビット・ストリーム３５０を受け取り、デコードするデコーダ３５１を含む。エンコーダ内のデコーダは、イメージをデコードするためにパッキング３４０又はアンパッキング３７０の処理を行う必要はない。デコーダでは、コードブック再生器３６０はアンパッキング処理３７０に供給され、完全なビット・ストリームに戻す。この処理の結果はポスト・フィルタ３７５に送られ、ディザリング３８０がイメージに対して行われ、最終的にイメージが表示される（３９０）。

従来技術の量子化処理の例は、次の文献に見られる：Gray,Ｒ．Ｍ．「Vector Quantization」（グレイ，Ｒ．Ｍ．による「ベクトルの量子化」－１ＩＥＥＥＡＳＳＰ Magazine ４－２９（１９８４年４月）（「グレイ」）及びNasrabadi,Ｎ．Ｍ．「Image Codi

ng Using Vector Quantization」（ナスラバディ，Ｎ．Ｍ．による「ベクトル量子化を使ったイメージのコード化」）－Ａ Ｒeview「ＣＯＭＭ－３６ ＩＥＥＥ Ｔransaction on Ｃommunication, ９５７－９７１（１９８８年８月）（「ナスラバディ」）、このようなベクトルの量子化はツリー・サーチ（tree searched）のベクトル量子化装置の作成を含み、グレイの記事１６－１７頁及びナスラバディの記事７５頁に記載されている。
コードブックの生成は反復的であり、計算機的に高価である。従って、フレーム毎にコードブックを必要とするいくつかの従来技術の方法では、コード化は低速となる。更に、トレーニング・シーケンスを用いる従来技術のシステムの欠点は画質であり、トレーニング・シーケンスのイメージと同様でない多くのシーケンスは受容できないであろう。全体的性能もまた気がかりである。いくつかの従来技術のテクニックは法外な処理を要し、リアル・タイムの圧縮を行うことができない上に許容可能な圧縮も達成されない。高速のデコード能力に対する需要は切迫しており、さもなくば、リアル・タイムの再生は不可能である。多くの従来技術システムは計算機的に高価なデコーダを有する。

発明の概要及び目的

本発明の１つの目的は、ベクトル量子化によってコードブックを効率的に生成し、イメージの空間的、時間的冗長性を低減する装置と方法及び圧縮システムの帯域幅を節約するためのイメージの関連処理を提供することである。
本発明の他の目的は、一般的従来技術のベクトル量子化技術に関連したエラーを低減するため、イメージを効率的に区画化し、処理する手段を提供することである。
本発明の他の目的は、一般的従来技術のベクトル量子化技術に関連した計算を更に低減する手段を提供することである。
本発明の他の目的は、限定された帯域幅のチャネルでの円滑な再生に適合するため、圧縮されたシーケンスの結果的データ・レートを効率的及び効果的に制御する手段を提供することである。
本発明の他の目的は、圧縮されたデータのリアル・タイムのデコーディングを可能にする単純なデコード構造を提供することである。
本発明のこれら及び他の目的は、事前処理されたブロックをベクトル量子化装置へ送る、データをコード化する方法と装置によって提供される。ベクトル量子化装置は、イメージ・ブロックをインデックスによって代表的ベクトルのテーブルに表現し（コードブックと呼ばれる）、忠実度の基準を最小にしている。１実施例では、イメージ・ベクトルの輝度とクロミナンス（ＹＵＶ）を用いて、コードブックとイメージ又はイメージのセットと関連するコードブックのインデックスを決定する。別の実施例では、どの程度の変化が起こるかに適合する時間的フィルタリングが行われ、動きの産物を低減する。別の実施例では、もし前のコード化されたフレームと現在のフレームの差が順応的閾値より小さければ、イメージ・ベクトルは送られない。実施例では、異なったコードブックを参照する異なったイメージ領域及び可変サイズの領域を提供する。実施例ではまた、異なったコードブックを参照する異なったブロック・タイプを提供する。実施例が、ベクトル量子化に関して共用され、可変サイズのコードブックに関して提供される。実施例が更にイメージの端を避けるプリ・フィルタリングに関して提供される。実施例が更にレート制御体系に関して提供される。実施例が更に順応的空間的サブ・サンプリングによる、空間的冗長度の低減及び時間的サブ・サンプリングによる時間的冗長度の低減に関して提供される。

【図面の簡単な説明】

本発明は例示によって示され、付随する数値に限定されるものではなく、類似の参照は類似のエレメントを示す。
図１ａ－１ｃは、ビデオ・イメージを圧縮／圧縮解除するために用いられる従来技術のコード化／復号化装置を示す。
図２はイメージを２×２画素ブロックから成るベクトルに分割する従来技術の体系を示す。
図３は従来技術のＣＯＤＥＣ（コード化／デコード）の機能ブロック図を示す。
図４は、無変化のブロックを識別するプリ・プロセシング技術を示す。

図５a及び５bは、望ましい実施例で用いられるサブ・サンプリングの例を示す。
図６は望ましい実施例によって提供される改良したベクトルの量子化を用いて作成されるベクトル量子化ツリーを示す。
図７及び図８は、図６に示すツリーを作成するために用いられる改良したベクトルの量子化処理を示す。
図９a及び図９bは、「ゼロ」セルを除去し、残りのノードで反復することによって、ノードがベクトル・ツリー内でどのように更新されるかを示す。
図１０は、望ましい実施例で用いられるビット・ストリームを示す。
図１１－１６は、図１０を参照して論じられるビット・ストリームに含まれるデータの詳細図を示す。

詳細な説明

本発明はベクトル量子化の改良した方法に関する。以下の説明において、説明の目的で、本発明の完全な理解を与えるために、特定タイプのデータ、アプリケーション、データ構造、ポインタ、インデックス及びフォーマットが述べられている。しかし当業者には、本発明がこれらの詳細事項なしに実行できることは明白であろう。また、本発明を不必要にあいまいにしないため、既知の構造やデータはブロック図の形で示されている。
本発明の望ましい実施例は、図３に３００として示される従来技術のＣＯＤＥＣと類似の態様で構成される。これらはディスプレイ、プロセッサ及び様々な静的及び動的記憶装置を含む汎用のプログラムされたコンピュータ・システムで実施できる。これはまた、特別目的のアプリケーション用に設計された特別目的のビデオ用コード化又はデコード装置を含むことができる。もちろん、当業者には、望ましい実施例の方法と装置は、アプリケーションの要件に適合するように独立した論理装置、ファームウエア、アプリケーション固有の集積回路（ＡＳＩＣ）又はプログラム論理アレイで実施できることが分かるであろう。
望ましい実施例は「Ｃ」プログラム言語のようなハイレベル・プログラム言語で実施され、汎用コンピュータ・システムで動作する。望ましい実施例を実施するために書かれたルーチンは実行可能なオブジェクト・コードにコンパイル及びアセンブルされ、ランタイムにシステムのプロセッサにロードされ、実行される。
本発明の議論は、ビデオ情報を参照して具体的に記載されているが、ここで論じた技術と装置は、ベクトル量子化を利用したオーディオの分野等他の分野にも同様の適応性があり、この出願のビデオ情報の議論が本発明を限定しているとみるべきでない。

プリ・プロセシング

ＣＯＤＥＣから出力されるデータ・レートは、プリ・プロセッサ３２０を通してベクトル量子化処理に至る情報量を制御する。これは全体的及び局所的の２つのレベルで行われる。空間的解像度に対する全体的変化は、イメージの帯域幅を変える低域入力フィルタを入力イメージに対して適用することによって行われる。このフィルタの通過帯域幅は必要なデータ・レートにおける誤差で変わる。誤差が少なくなるにつれて、入力フィルタの帯域幅は増加し、ＣＯＤＥＣにより多くの情報が到着できるようにする。逆に所望のデータ・レートでの誤差が増加すると、入力フィルタの帯域幅が減少し、ＣＯＤＥＣに到達する情報を制限する。時間的解像度に対する全体的変化は、現在と前のフレームの間の差を判定することによって行われる。もし変化が閾値以下なら、現在のフレームはスキップされる。閾値はデータ・レート誤差から決定される。時間的帯域幅が減少する別の全体的メカニズムは２つのフレームの間の誤差の定義を拡張することによって、誤差計算の前にフレームの変形を可能にすることである。このような変形はパン（pan）やズームの補償を含むが、それらに限定されるものではない。
ベクトル量子化処理に到達することを許される情報量の局所的制御は、空間的サブ・サンプリングと時間的ブロック（又はより一般的には動きが補償されたブロックの局所的決定）を含む。望ましい実施例のシステムは、図３の３３０で示す改良したベクトル量子化装置を実施し、コード化されるイメージのような非常に大きなベクトルのセットから代表的イメージ・ベクトルの小さなセット、コードブックという、を生成するのに非常に効率的

である。このようなベクトル量子化装置によって生成されたコードブックからデコーダ３５１によって再構築されたイメージは、ある基準に関して元のものに近い。全体的圧縮／圧縮解除体系の性能は望ましい実施例において、ベクトル量子化装置の前にプリ・プロセッサ３２０によってビット・ストリームの内容を制御することにより更に向上する。このプリ・プロセシングはベクトル量子化装置３３０に対して透明である。プリ・プロセッサ３２０は、画質の損失を最小にしてイメージをコード化するのに用いられる情報量を実質的に低減する。望ましい実施例ではタグを用いて丁度その時に変わらないベクトルをコード化する代わりに明示する。ある閾値に従って変わらないため、「無変化」のブロックとして知られている。望ましい実施例では、ブロックは更に空間的サブ・サンプリングを用いて処理され、より良好な圧縮を達成する。更にプリ・プロセッサ３２０は、スピードを増大し又は画質を向上させるため、赤、緑、青（ＲＧＢ）表すコード化から輝度とクロミナンス（ＹＵＶ）を用いて表すコード化への変換等を行って、イメージ空間の特性を変えることができる。

無変化ブロック

望ましい実施例において、イメージ・ベクトルをコード化するか「無変化」ブロックのタグを送るかを決めるために一連の決定が行われる。「無変化」ブロックの場合、そのイメージ・ブロックに関してインデックスは送られる必要がないので、圧縮は殆ど常に向上する。コードブックを作成し、そのインデックスを見つけるイメージ・ベクトルが少ないので、コード化速度は向上する。前のフレームからのデコードされたブロックの上にスクリーン上で新しいブロックを配置しなくて良いため、デコード時間も向上する。従って、コードブックのエレメントを参照するインデックスを送る代わりに、プリ・プロセッサ３２０によって無変化タグが送られ、同一位置で、前のフレームのブロックからそのブロックが実質的に変わっていないことを示してベクトル量子化装置３３０を素通りする。これは図４を参照して示され、論じられる。処理４００はステップ４０１で始まり、ステップ４０２でフレームＮの次のブロックを取り出す。ステップ４０３でフレームＮのこのイメージ・ブロックは次にプリ・プロセッサ３２０によって、デコードされたフレームＮ－１からの同一位置のイメージ・ブロックと比較される（デコードされたフレームＮ－１はエンコーダのビット・ストリームの出力から抽出されデコードされる。）。もしステップ４０４で検出されるように、２つのブロック間の誤差が閾値μより大きいと、ステップ４０６でそのブロックは変更されずにベクトル量子化装置３３０にコード化のために渡される。それ以外の場合、そのブロックはＶＱ３３０に対して「無変化」のタグが付けられ、ステップ４０５に示すようにベクトル量子化は行われない。別の実施例では、無変化ブロックは前のフレームのブロックのうち、どれがサーチ領域内で良好な一致がとれたかを示す画素オフセットを持つことができることに留意されたい。

所望のデータ・レート及び画質が非常に高い場合、無変化のブロックとしてμをパスするイメージ・ブロックは、無変化ブロックとしてタグを付ける前に、より厳しいテストをされる。ブロックが無変化のブロック、ブロックの「エージ（age）」という、であるフレームの数は、チェックされ、最大許容エージを超えていないことが確かめられる。もし最大許容エージを超えていなければ、そのブロックは「無変化」ブロックとしてとどまる。もし最大許容エージを超えていれば、そのブロックと前のデコードされたフレームの同位置のブロックとの間の誤差がよりきつい閾値、例えばμ／２と比較される。これは無変化ブロックが所定の位置に長時間残っており、見る人から気付かれるのを防止するために行われる。ブロック・エージを用いる副作用は、多くのブロックがエージ化し、一緒に最大エージとなるときに起きる。これによって突然データ・レートが最大し、イメージの内容に無関係なデータ・レートの変動を起こすきっかけとなる。これを防止するため、望ましい実施例では各ブロックは様々な開始エージに初期化され周期的にリセットするようにしている。これはランダムに行うことができるが、もし連続したイメージの区画で行われた場合、エージ化はブロック・ヘッダでビット・ストリームをくずすことはあまりない。無変化ブロックのエージ化する主な欠点は、データ・レートが高いことである。従って、所望のデータ・レートが非常に高い圧縮を要求せず、高い画質を要求するとき使用するのに

適している。処理４００は、ステップ４０７で決まるように、フレームが完全に処理されたときステップ４０８で終わる。ブロックに「無変化」としてタグを付ける決定は、一旦空間的サブ・サンプリングが行われると覆ることがある（例えばブロック・データが送られる）。もしデコーダに対して後続のブロックは「無変化」であることを知らせるためのブロック・ヘッダのオーバーヘッドによって、「無変化」のブロックを有する圧縮の正味の利益がなくなれば、「無変化」のブロックは先行するか後続するブロック・タイプに変更される。現在の実施例でこれが起こる例は、サブ・サンプルされたブロックの流れの中間で単一の４×４ＮＣ（４－２×２無変化）ブロックがあるときである。単一の４×４ＮＣブロックは先行する１つのヘッダと後続する１つのヘッダを必要とし、サブ・サンプルされたブロックの流れから分離し、ブロック・ヘッダ毎に１バイトとして１６ビットを生ずる。もし単一の４×４ＮＣブロックがサブ・サンプルされたブロックに変えられたとすれば、単に８ビットのインデックス（２５６のエントリ項目のコードブックに対して）を要し、伝送ビット数について云えば４×４ＮＣブロックにしておくよりも少ない犠牲で済む。処理４００において無変化ブロック選択の決定に対して有用な様々な誤差と閾値の計算がある。望ましい実施例においてブロック比較に用いる誤差の基準は２乗誤差計算である。ＳＮＲ（信号電力対ノイズ電力比）も別の実施例で用いることができ、高い輝度の領域に対する大きな誤差が許されるので有用である。これは人間の目が高輝度の領域における輝度の変化に鈍感であるという事実と関連する（ウェーバーの法則）。閾値μは望ましい実施例ではユーザの画質設定によって初期に決定されるが、レート制御要求と前の一連のフレームの平均２乗誤差（frame_mse）に順応して初期値から変わり得る。望ましい実施例に用いられるアプローチは、無変化の閾値及びμを次のように計算することである。

$$ncthreshfactor_n = ncthreshfactor_{(n-1)} + \beta * long_term_error_{(n-1)} \quad (\beta = 0.001)$$

$$\mu_n = ncthreshfactor_n * frame_mse_n$$

$$min\ mse\ bound < \mu < max\ mse\ bound$$

改善されたレート制御メカニズム３４５の議論で以下に詳しく論ぜられるlong_term_error（長時間誤差）は、期間中の必要なデータ・レートを達成するベンチマークを提供する。もしlong_term_errorが、データ・レートは高すぎることを示すと、無変化のブロックはより頻繁にフラグが付けられる。逆に、もしlong_term_errorが、生成されたデータ・レートは所望値より低いということを示すと、無変化のブロックはそれほど頻繁にフラグを付けられない。瞬間的に反応する代わりにμはβによってバッファされ、データ・レートを変える反応時間の時定数（又は遅れ）を効果的に制御する。これは振動的データ・レートを防止し、完全にデータ・レートで駆動されるのでなく、多くのビットを生成する多くのバリエーションを有する複雑なイメージと、少ないビットを生成する少ないバリエーションを有するあまり複雑でないイメージに対する公差を許す。所与のシーケンスにおける達成可能な画質幅があるため、無変化の閾値μはframe_mseを考慮してシーケンスの直前にコード化された部分の画質を維持する。frame_mseはレート制御３４５によっても用いられ、これについてはレート制御の章で詳細に論ずる。

空間的サブ・サンプリング

望ましい実施例においてプリ・プロセッサ３２０によって行われる別の技術は空間的サブ・サンプリングである。サブ・サンプリングはベクトル量子化装置３３０によってコード化される情報量を低減するのに用いられる。これによっていくらかの空間的画質の犠牲で高速のコード化と高度の圧縮が得られる。主なチャレンジは高画質、高圧縮を維持することである。望ましい実施例で取り得る２つのアプローチがあり、それぞれ異なった利点がある。第１のアプローチでは、イメージは「スムーズ」と「ディテイル」の領域に分けられ、「スムーズ」のブロックはデータ・レートの要求に従ってサブ・サンプリングされる。例えば、「スムーズ」領域は元のブロックと対応するサブ・サンプリング及びアップ・サンプリング（upsampling）されたブロックの間で平均２乗誤差を比較することによって決まる。これはサブ・サンプリングされた「スムーズ」領域は、通常最小の産物すなわち

誤差を生成するので有利である。このアプローチに対する別の利点は、２つの別々のコードブックがサブ・サンプリングされた２×２Ｃ（「変化」）ブロックに関して生成され、各コードブックが数フレームに亘って共有されるときに発生する。「円滑性（smoothness）」に完全に依存するサブ・サンプリングによって、２つのコードブックは「スムーズ」及び「ディテイル」の領域を多くのフレームに亘って表すことができる。これは「スムーズ」領域のイメージ・ベクトルは通常多くのフレーム間で類似しており、また同じことが「ディテイル」領域についても云えるためである。ゾーン（zone）が用いられる第２のアプローチでは、イメージ内のブロックの位置もサブ・サンプリングの決定に影響を与える。第２のアプローチの利点はイメージのどの領域がポスト・フィルタに行くのかをデコーダに対して効率的に通信する（ビットに関して）能力を有することであり、サブ・サンプリングのブロックを集めることによって、より効率的ブロック・ヘッダのコード化の能力を有することである。サブ・サンプリングの処理は図５ａを参照して論ぜられる。サブ・サンプリングに関して、イメージは図５ａに示すように４×４のブロックに分割される。各４×４ブロックは、サブ・サンプリングとして選択されると５１０のような２×２ブロックに縮小される。望ましい実施例で行われるフィルタ・サブ・サンプリング動作は、４つの４×４画素ブロックのそれぞれの加重平均を用いて（例えば画素１－３、５－７、９－１１及び１７－２３から成るブロック５１８）サブ・サンプリングされたブロック５１６（ブロック５１８の場合画素１、２、５及び６のブロック）を表す。別の実施例では、図示のように単一の画素（例えば１、３、９及び１１）がサンプリングされ、より簡単なサブ・サンプリングの体系でサブ・サンプリングされたブロック５１０の代わりに用いられる。もしイメージ全体がこれらの技術の何れかを用いてサブ・サンプリングされると、改良されたベクトル量子化装置３３０に入るベクトルの数は４のファクタで低減し、よって、最終ビット・ストリームのコードブック・インデックスの数もまた４のファクタで低減する。別の実施例では、サブ・サンプリングは水平方向にのみ又は垂直方向にのみ或いは、２以上のファクタで４×４画素以上のブロックを２×２画素ブロックにサンプリングすることによって各方向に行うこともできる。デコード中、改良されたデコーダ３５１は、インデックスに先行するヘッダにおいて、５１０のようなブロックの含まれるインデックスはサブ・サンプリングされたブロックを指示していることを検出し、５２０のような完全な４×４ブロックを再生するために各画素を水平、垂直の両方向に１つずつ模写する（例えば、４画素から成るブロック５２１を見ると、それぞれは単純サブ・サンプリングの場合における画素１と同じである）。ブロック５２１は４つの１の代わりに４つのγで表され、γはブロック５１８の加重平均であることに留意されたい。別の実施例では、既存の画素の間の画素は良好な結果を得るため、相隣れる画素から補完することができる。しかしこれはデコーダの速度に有害な効果を与え得る。

「円滑性（smoothness）」が判定される方法は、もし１つのブロックがサブ・サンプリングされる場合、どの程度２乗誤差が生ずるかに依存している。サブ・サンプリング動作は次の誤差計算に示されるようにフィルタリングも含む。２乗誤差εは図５ｂに示す５６０（画素a_0-a_3からなる）のような２×２ブロックのそれぞれとそれを囲む４×４ブロック５５５（画素a_0-a_3及びb_0-b_{11}からなる）の平均γとの間で計算される。

$$\gamma = \frac{1}{16}\left(\sum_{i=0}^{11} b_i + \sum_{i=0}^{3} a_i\right)$$

ブロック５１８から計算されたγは２×２ブロック５２１の画素１の値の代わりに用いられる。もし５６０のような２×２ブロックがサブ・サンプリングされると、それを囲む４×４γ（ブロック５５５）の平均が４つの個々の画素値a_0-a_3の代わりに送られる。平均γはブロッキネス（blockiness）を低減するのに役立つ。従って図５を参照して示されるように、値γは、ブロック５３０の元の４つの画素値a_0-a_3の代わりに送られる。次に２乗誤差εは加重計数κによってスケーリングされ、人間のシステム輝度に対する感度に近似される（ＭＳＥの代わりにＳＮＲをおおよその近似として用いることもできる）。従って、サブ・サンプリング誤差が同じであると仮定して高輝度の領域は容易にサブサン

プリングすることができる。４つのスケーリングされた誤差は加算されて５６０のような各２×２ブロックに関連する誤差を生成する。

$$\varepsilon = \sum_{i=0}^{3} k[Y_i] * (a_i - \gamma)^2$$

Y_j：画素a_jの量子化された輝度値

サブ・サンプリングのための候補として４×４ブロック５００を順位づけるため、４×４５００の角に配置された４つの２×２ブロックからのサブ・サンプリング誤差εのそれぞれが加算される。レート制御が、所望のフレーム・サイズに合うように十分なブロックがサブ・サンプリングされたことを判定するまで、ブロックは最小の誤差ブロックから最大の誤差ブロックに向けてサブ・サンプリングのために選択される。別の実施例では、イメージの端がサブ・サンプリングされるのを防止するため、当業者には既知の端検出方法によってイメージの端が抽出される。サブ・サンプリングの決定をサブ・サンプリング誤差に依存することは、サブ・サンプリング及び端を越えるアップ・サンプリングが最大誤差を生ずる傾向があるため、多くの端を保護する傾向がある。しかし、端検出で見つかった端をはっきり保護することは、ある場合には有用である。

純粋に誤差に基づいたサブ・サンプリングは、多くの場合うまく動作するがサブ・サンプリングされたブロックが必ずしも互いに隣接して発生しないイメージがある。従って、サブ・サンプリングされないブロックの隣のサブ・サンプリングされたブロックの出現は、可視的に見ている人を悩ます閃光効果を起こし得る。いくつかのブロックはサブ・サンプリングされ、その他ブロックはサブ・サンプリングされないのでブロックが動いているように見える。第２に、もしサブ・サンプリングされたブロックと標準的にコード化されたブロックが、空間的に混在すると、プリ・プロセッサ３２０によって識別されるブロック・ヘッダによってブロック・タイプの変化を表さねばならないので、かなりの帯域幅（ビットにおける）が消費される（ブロック・ヘッダについては以下にビット・ストリームのシンタックスを参照して詳しく論ずる）。このようなイメージにおいては、別の実施例のコード化体系において、ゾーンを用いて誤差のみに基づくサブ・サンプリングする前述の２つの欠点を低減できる。イメージはプリ・プロセッサ３２０によって３２の矩形ゾーン（水平８、垂直４）に分割され、それぞれは自らに関連した重みを有する。明らかに、ゾーンの数とそのサイズは全く多様である。１実施例では、中心のゾーンをサブ・サンプリングされにくいように境界ゾーンのイメージの重み付けが行われる。このことは、カメラは大旨対象物の中心に向けられるため、見る人は端にあまり注意を払っていないと云うことを前提としている。別の実施例では速い動きを使って、サブ・サンプリングの産物を隠している。もし当業者に既知の運動予測アルゴリズムによって、動きが速くないと判定されると、動きの領域をサブ・サンプリングすることを難しくすることは有用である。このことは、見ている人は運動物体を追跡し、動きが速くなければサブ・サンプリングの産物に気付くということを前提としている。

望ましい実施例の第２のアプローチにおいて、ゾーンはゾーンの誤差、

平均２乗誤差$\hat{\varepsilon}$

に従って分類される。

$$\varepsilon_j = \sum_{\text{zone } j} \varepsilon$$

$$\hat{\varepsilon}_j = \frac{\varepsilon_j}{\#\text{ of subsampled pixels in zone } j}$$

各ゾーンはその位置に従って重み付けされ、ゾーン誤差ＺＥを生成する。

$$ZE_j = \hat{\varepsilon}_j * zone_weight[j], \quad \text{zone } j$$

サブ・サンプリングのためにタグを付けられたブロックはゾーン誤差に関して最良ゾーンから最悪ゾーンの順にサブ・サンプリングされる。これはレート制御３４５によって要求

されたサブ・サンプリングの数に達するまで行われる。改良されたデコーダ３５１は、入力ビット・ストリーム３５０から特定の基準で（画質設定等）どのゾーンがサブ・サンプリングされたかを判定でき、ブロッキネス（blockiness）を柔らげるためにこれらのゾーンをポストフィルタ（処理３７５）にかけるかどうか決める。サブ・サンプリングは帯状であるので、デコーダ３５１はイメージ全体をポストフィルタにかけないで、どこに努力を集中すればよいかを知っている。この情報をデコーダに送るのにオーバーヘッドは最小であり、３２の矩形ゾーンの場合わずか３２ビットである。ゾーン全体がサブ・サンプリングされるのを防止するため、edge_mseより小さい誤差を有するブロックだけがゾーン内でサブ・サンプリングされる。edge_mseの値はレート制御によって制御されるので、所望の圧縮フレーム・サイズが大きければ多くのブロックがサブ・サンプリングから保護される。

$edge_mse_n = edge_mse_{(n-1)} + x * long_term_error$

別の実施例ではedge_mseは重み付けることができ、当業者に既知の端検出方法で抽出されたイメージの端は、サブ・サンプリングから保護される。

方向性フィルタリング

空間的冗長度も、別の実施例における「方向性」フィルタリングを行うことによって端とディテール（detail）の最少の損傷で低減することができる。この処理は、画素を囲む領域に対して水平、垂直、上向き対角線、下向き対角線のフィルタを実行し、最小誤差を生成するフィルタを選択する。もしフィルタの長さが３タップ（tap）、（フィルタ係数）であれば、図５ａの画素６のフィルタをかけた値を計算することは、画素６に関するフィルタをかけた値を生成するため、画素５、６及び７に対して水平フィルタを適用し、画素２、６及び１０に対して垂直フィルタを適用し、画素１、６及び１１に対して下向き対角線フィルタを適用し、画素９、６及び３に対して上向き対角線フィルタを適用することを意味する。例えば、「水平フィルタ」を実行するために値はf_hであらわされ、f_hは次のように計算される。

$f_h = a_1 \cdot 画素5 + a_2 \cdot 画素6 + a_3 \cdot 画素7$

ここで、a_1、a_2及びa_3は重み係数であり、それぞれ０．２５、０．５及び０．２５であり、３×３ブロックの中央画素に多くの重みが与えられ、結果のf_hは計算的に安価なシフト演算で計算される。これらのフィルタは３次元空間に適用でき、追加の次元は、別の実施例では、時間であることに留意されたい。

方向性フィルタの結果を比較することによって、イメージの端の方向付けも得られる。端の方向付けは、直交方向の対に関連する誤差の比を比較することによって抽出される。第１のステップは最小誤差min_directional_errerを生成した方向を選択することであり、この誤差を他の３方向のフィルタに関する誤差と比較することである。最小誤差のフィルタの方向に方向性の端があることを示す特徴は次の事項を含む。

・最小誤差を生成した最小誤差フィルタの方向と直交する方向

・最大誤差のフィルタは特にそれ自身と直交する方向と比較したとき、他の３方向よりも格別に大きな誤差を有する。フィルタをかけた領域が互いに他と非常に近い方向性誤差を有する場合、その領域は「無方向」である。「無方向」のブロックの領域は、再びその領域に対してフィルタをかけることができる。最小誤差のフィルタは、画素の周囲の特徴に従ってどの画素に対してもその特性を変えるので、非常に順応性がある。

ＹＵＶ変換

望ましい実施例はまた、コードブックの生成とベクトル量子化装置３３０に関してベクトルの輝度とクロミナンス値（ＹＵＶ）を用いて速度や画質を向上させる。ＹＵＶ値は、ベクトルにおける画素の赤、緑、青（ＲＧＢ）値からその再構築が計算的に安価な単純な変換を通して計算できる。例えば乗算の代わりにビット・シフトで実現できる次の変換である。

$$Y = \frac{R}{4} + \frac{B}{4} + \frac{G}{2}$$

$$U = \frac{R - Y}{2}$$

$$V = \frac{B - Y}{2}$$

ベクトル量子化装置３３０においてＹＵＶを用いてコードブックの生成を行うことは、成分間のきつい動的範囲と相対的非相関性によってクラスタリングを向上させる。従って画質の向上は顕著である。コード化速度が重要である状況では、クロミナンス（Ｕ．Ｖ）値は２又は４でサブ・サンプリングされ、ベクトル量子化ステップ３３０において重み付け（例えばシフトで）られる。

望ましい実施例において、輝度及びクロミナンスは、入力イメージのベクトルのサブ・サンプリング又はフィルタリングのようなＲＧＢの事前処理の後、プリ・プロセッサ３２０によってベクトル量子化装置３３０に送られる。別の実施例では、ＹＵＶ変換は最初に行われ、サブ・サンプリングのような事前処理はＹＵＶ変換の後に行われる。如何なるレートにおいても、結果の事前処理データは改良ＶＱ３３０にＹＵＶのフォーマットで送られる。

改良されたベクトル量子化装置

ベクトル量子化（ＶＱ）はブロック又はデータのベクトルを表すのに効率的方法である。一連のデータ、画素、オーディオ・サンプル又はセンサー・データはしばしば各データを独立して取り扱うことによって量子化される。これをスカラー量子化という。一方、ＶＱはデータのブロック又はベクトルを量子化する。主なＶＱの問題は、データ・セットの許容し得る近似であり、コードブックと呼ばれる代表的ベクトルのセットを見つける必要があることである。許容性は通常元のデータ・セットと再生されたデータ・セットの間の平均２乗誤差を用いて測定される。コードブック生成の一般的技術はＬinde,Ｙ、ＢuS0, Ａ及びＧray,Ｒによる「An Algorithm for Vector Quantizer Design（ベクトル量子化装置設計のアルゴリズム）」ＣＯＭ－２８ＩＥＥＥ Transactions on Communications1（１９８０年１月）（ＬＧＢアルゴリズムとして知られている）に記載されている。

コードブックを生成するためにＬＧＢアルゴリズムを採用した技術は、コードブックの初期予測を生成するためイメージからの入力ベクトルをサンプリングすることで始まる。次に各入力ベクトルはコードブックの項目と比較され、最も近いコードブックの項目と関連させられる。コードブックの項目は各コードブックの項目と関連する平均ベクトルを計算し、現在の項目を平均ベクトルで置き換えることによって繰り返し更新される。次にコードブックが前回より大幅に向上したかどうかの判定が行われ、もし向上していなければ、入力ベクトルをコードブックの項目と比較し、再関連させる等によって処理をくり返す。このコードブックの生成は大きなイメージ・シーケンスすなわちトレーニング・セットに対して行われるか、又はコードブックは各フレームに対して再生される。更に、この技術は効率向上のため、特定の従来技術のベクトル量子化システムに用いられるバイナリ・ツリーに適用できる。

改良ベクトル量子化装置３３０はツリー構造に編成されている。特定の従来技術の体系で用いられているバイナリ・ツリーではなく図６に示すようにツリーのルートでＮ個の子供ノード６１０が初期に生成される。これはいろいろなテクニックを用いて行うことができる。例えば、１実施例では、セグメンター（segmenter）を用いてイメージから代表的セントロイド（centroid）を抽出し、中心値を有するＮ個の初期ノードを生成する。別の実施例では、初期のセントロイドは、そのイメージ自身からＮ個のベクトルを抽出すること

によって、１つのイメージから決定される。従来技術のバイナリ・ツリーは単に２つの初期ノードの設定に依存している。バイナリ・ツリーには、２つの初期ノードにおける誤差がツリー内の残りのノードに伝搬するという欠点の悩みがある。望ましい実施例では、Ｎ個のノードが用いられ、値Ｎはイメージの特性によって変わる。これの利点は、多くの初期ノードがルート・レベルでの間違ったビンニング（binning）のチャンスを低減するという事実に関連している。良好な画質と速い収束は、ツリー作成においてＮ個の初期ノードを用いることから達成され、Ｎはイメージに適合し、通常２よりも大きい。

イメージに対して行われる改善されたベクトル量子化処理７００は図６、７及び８を参照して示され、論ぜられる。Ｎ個の初期ノードの作成は図７のステップ７０２で行われる。ツリーの最上層６１０は、ステップ７０３で初期ノードの値をくり返し調整し、ベクトルをそれらに関連させることによりＮ個の初期ノードから改善される。この繰り返し処理は、繰り返しのノード・ビンニング／再計算処理を示す図８を参照して以下に記載されている。ステップ７０４で、最悪の歪を有するノードが決定され、その歪はノードのセントロイド値とその関連ベクトルの間の比較から計算される。望ましい実施例において、ノードに関連するベクトルとノードのセントロイド値の間の平均２乗誤差は歪の尺度として用いられる。どのノードが最も歪んでいるかの判定は、別の実施例で多くの尺度を用いて行われることに留意されたい。これらの尺度は、個体数、ノードに関する合計歪、ノードに関する平均歪あるいはノードに関するピーク歪が含まれる。どのレートにおいても、ステップ７０４で一度最大歪ノードが決定されると、ステップ７０５でこのノードは２つの子供ノードに分割される。もちろん、望ましい実施例では２つの子供ノードが記述され、用いられるが、別の実施例では２つ以上の子供ノードが作成されても良い。最良の代表ベクトルを得るため、ステップ７０６で子供ノードに対する繰り返し処理が行われる。この処理は、図８を参照してより詳しく説明されている。最も歪んだノードから作られた子供ノードに適用されたステップ７０３又はステップ７０６のような繰り返し処理は図８に示されている。この処理はステップ８０１で始まる。ステップ８０２で、親ノードと関連するベクトルのグループから図６の６７０に示す子供ノードに、代表的セントロイドを割り当てる。ルート・ノードの場合は、イメージの全ベクトルを用いて代表的セントロイドを作成する。次に各ベクトルは最も近いセントロイドを有するノードと関連づけられる（ビンニング）。次にステップ８０４で、各セントロイドと関連するベクトルとセントロイド自身との間の誤差が決定される。誤差計算は色々なテクニックを用いて行われるが、望ましい実施例においては平均２乗計算が用いられる。ステップ８０５で１度誤差計算が決まると、誤差の変化が特定の閾値以下になったかどうか判定される。ステップ８０６で、ステップ８０３からのノードと関連するベクトルから新しいセントロイドが計算され、これはステップ８０３からの全てのノードに対して行われる。７０６に示す処理の最初の繰り返しでは、誤差の変化は非常に大きく、大きなプリセットの値から、計算された誤差値に進んでいく。しかしステップ８０３から８０６のループの次の繰り返しでは、誤差の変化は小さくなり、最終的に閾値より小さくなる。もし現在分割されているノードに関する誤差の合計が、ステップ８０５で決まるように、閾値より小さくなければ、ステップ８０６で新しいセントロイドが再計算され、処理７０３（７０６）が続いて行われ、ステップ８０３から８０６が必要に応じて再びくり返される。これは、ステップ８０５で検出されるように、誤差の変化が所定の閾値より小さくなるまで行われる。ステップ８０５で検出されるように、誤差の変化が閾値より小さくなると、処理７０３（７０６）はステップ８０７で終了し、図７の処理７００に戻る。

この反復処理が図７のステップ７０７で完了すると、ツリー内で所望の数のターミナル・ノードが作られたかどうか判定される。ノードが分割される度に、２つ又はそれ以上の子供ノードがＶＱツリー６００に生成される。従って望ましい実施例では、要求されるターミナル・ノードの合計数は、ＶＱツリー６００のノードが何回分割されるかを決定する。処理７００は、必要なターミナル・ノードの数がツリー内に作られるまでステップ７０４から７０７を続ける。一度必要な数のターミナル・ノードが作られると処理７００はステップ７０８で完了し、コードブックは出力ビット・ストリーム上に伝送され、図３に示す

パック／コード化装置３４０に送られる。
望ましい実施例でノードに対して用いられるタイプ構成は次のような「Ｃ」プログラム言語で定義される。

```
typedef struct model{
  unsigned long   *centroid:          //  このノードに関するセントロイド
                                      //  に対するポインタ

  unsigned long   *vert_index_list:   //  このノードに関するベクトル・インデックスのリストに対するポインタ
  unsigned long   *num_vect:          //  このノードに関するベクトル数
  unsigned long   distortion:         //  このノードに関する合計歪
  unsigned long   avg_dist:           //  このノードに関する平均歪
  unsigned long   peak_dist:          //  このノードに関するピーク歪
  unsigned long   percent_dist:       //  このノードに関する歪のパーセンテージ
  unsigned long   num_children:       //  子供の数
  unsigned long   ic_method:          //  このノードの初期化方法
  struct mode     **children:         //  このノードの子供ノードに関する構造のリストに対するポインタ
  struct mode     *parent:            //  このノードの親に対するポインタ
  unsigned char   terminal:           //  これがターミナル・ノードかどうかを示すフラグ
  unsigned long   *childrencptrs:     //  次に対するポインタのアレイに対するポインタ
                                      //  子供のセントロイド（歪計算を単純
                                      //  化し高速化するために用いられる）
}
```

６００のようなツリーＶＱを構成するノードは、それぞれが上で定義したようなデータを有し、色々な歪尺度、ビンニングされたベクトル数、子供の数などのノードに関する特定の情報を維持する。この情報は上で論じたツリーの生成に役立つ。
６００のようなＶＱツリーの生成に対する望ましい実施例のベクトル量子化処理７００は多くの新しい技術を用いて実行される。
第１に、順応収束閾値（８０５で使用）は繰り返しの数を制御し、コードブック・ツリー

を生成するのに用いられる。これは２つの方法のうちの１つで動作する。
１．もし完全なツリーが更新されることになれば、ゆるい収束基準が初期のＮ個のノードに適用される。完全なツリーは場面の変更が起こったか又はイメージが前のイメージから著しく変わった場合に更新される必要がある。
２．もし前のツリーからのルート・ノードが現在のツリー構築に用いられるとすれば、そのルート・ノードには繰り返しは行われない。ルート・ノードは類似のイメージ・シーケンスがコード化され、場面変化が検出されない場合再使用できる。従って６１０のようなＮ個の初期ノードは、前のフレームのＶＱから再使用できる。
第２に、望ましい実施例では、再生イメージの画質向上のため、修正距離尺度が用いられる。通常イメージ・ベクトルとコードブック項目の間の平均２乗誤差（mse）が用いられて所与のベクトルに対して一番近く一致するコードブック項目を決定する（例えば、図８のステップ８０３）。ツリー生成の初期の段階では、望ましい実施例はこの計算を修正して大きな誤差には２乗誤差より重い重み付けを行う。このようにして、大きな誤差には小さな誤差よりも多く重み付けがなされる。
第３に、複数の基準を使ってどのノードが分割されるべきかを決定する。採用される尺度には次のものが含まれるが、これらに限定されない。
１．特定のノードに関する合計歪。
２．特定のノードに関する平均歪。
３．特定のノードに関する個体数。
４．特定のノードに関する歪のパーセンテージ。
５．特定のノードに関する最大歪。
６．特定のノードに関する最大歪と最小歪の比。
ノードに関する合計歪は望ましい実施例で用いられる。しかし、別の実施例のツリー生成の最終段階では、もし個体数を尺度として用いれば良好な画質の結果が達成できる。もし平均２乗誤差が歪尺度として用いられれば、歪の合計は平均２乗誤差の合計である。他の歪尺度又はそれらの組み合わせの使用は、他の実施例で用いることができ、それぞれは、イメージの内容や所望の画質に応じて特定の利点を有する。
第４に、ノードを分割するのに複数の試行が行われる。たまに特定のノードを分割する試行が失敗する。この場合、多くの他の初期条件が生成され分割がうまく行くように支援する。例えば、これが行われる１つの方法は、初期の分割に対してノイズを加えることである。平坦な、又は非常に滑らかに変化するカラー又は輝度の領域で特徴づけられる特定のイメージについては、ノードの分割は難しい。少量のノイズが分割に先立ってイメージ・ベクトルに加えられる。ノイズは疑似ランダムであり、入力イメージ・データのゼロと２つの最下位ビットの間の範囲を有する。ノイズ生成の１つの態様は、疑似ランダム・ノイズ発生器を用いることである。この値がコード化される各ベクトルの各画素のＲＧＢ成分のそれぞれに加えられる。各画素のＲＧＢ成分のそれぞれに加えられたランダム・ノイズは分割を成功させるのに十分それらを区別させる。一般にどのノードを分割するかの決定が行われたと仮定して、アルゴリズムは次のことを行う。
１．そのノードに関するベクトルのリストをサブ・サンプリングすることによって初期ノードのＫ個の候補を生成する。
２．これらの初期ノードを用いてベクトル・リストをクラスタリングする。
３．もしクラスタリングが失敗すれば（すなわち、全てのベクトルが１つのノードに集中する）、このノードをこの方法でクラスタリングするのに失敗したとして識別する。
４．このノードを分割する次の試行が行われるとき、ノード・セントロイドに対する異なった初期予測を用いる。この予測を生成するテクニックには次のものが含まれるが、これに限定されない。
ａ．親ノードのセントロイドを乱れさす。
ｂ．ノード・ベクトル・リストの中の最も歪んだベクトルを初期セントロイドとして捕らえる。
５．これらの初期ノードを用いてクラスタリングの試行が更に行われる。もし全ての方法

がベクトル・リスト中で分割を生成するのに失敗すると、そのノードはターミナル・ノードとしてタグが付けられ、更に分割の試行が行われることはない。
第５に、コードブックの第１の層を複数フレームの間で再使用する。多くのイメージ・シーケンスにおいて、主なイメージの特徴は時間と共にゆっくり変わる（例えば、背景イメージは変わるかゆっくり動きがちである）。Ｎ個の初期ノードから成るコードブックのツリー６１０の最上層はこれらの特徴を捕らえる。計算速度と高画質に関する向上した性能は、１つのフレームから次のフレームに対してツリーの最上層を再使用することによって得られる。この再使用はＣＯＤＥＣで高いレベルからオーバーライドされる。例えば、場面変更の場合、これはＣＯＤＥＣで検出されるのであるが、ルート・ノードが再使用されるよりもむしろ再生された方が高画質が達成できる。
第６に、コードブックの利用可能な項目を最高に使用するため、コード化の前にベクトルの平均値を除去するのが一般的である。これは良好な再生イメージの画質をもたらす一方、デコーダで追加の複雑さが発生する。望ましい実施例はデコーダの複雑さなしに平均剰余ＶＱの多くの利点を与える技術を利用する。この技術は次のように動作する。平均値が大きなイメージ又は「ゾーン」に対して計算され、この平均値は大きなゾーンの全てのベクトルから減算される。残りのベクトルは通常のやり方でコード化される。デコーダでは大きなゾーンのそれぞれに対するコードブックは再構築される。これは、大きなゾーンの平均値を残りのコードブックに加えることによって行われる。この結果は、エンコーダでの大きなゾーンの数と同じ数のコードブックの生成となる。

可変サイズ、共用化、イメージに対する複数のコードブック

固定コードブックと適合性コードブックの組み合わせも別の実施例では可能であるが、望ましい実施例では各イメージはそのイメージの特性に適合したコードブックに関連しており、訓練された普遍的コードブックではない。別の実施例では、各イメージは正確に１個のコードブックまたはある固定サイズのコードブックを有するように限定される必要はない。別の実施例は可変サイズであり、複数のフレーム又は一連のフレームの間で共用でき、イメージをコード化する複数のコードブックを使用することを含む。これら全ての別の実施例において、利点は画質の損失を最小にして圧縮度を増大できることである。

可変サイズのコードブック

可変サイズのコードブックに関して、ツリー内のノードはある基準に達するまで分割され、これは指定した数のターミナル・ノードが存在する以前に起こる。１実施例ではコードブック・ベクトルの数は前のフレームから変化するブロックの数と共に増加する。言い換えれば、無変化ブロックの数が多ければ多いほどコードブックは小さい。この実施例では、コードブックのサイズは明らかに絵のサイズに関係する。望ましい実施例で用いられる確固とした基準はフレームの平均２乗誤差（無変化ブロックは含まない）の維持に依存する。もし２５６個でなく１２８個の２×２コードブック・ベクトルが用いられれば、正味節約はそのフレームで７６８バイトである。この節約は、各２×２ブロックが輝度情報に関して画素毎に１バイトを含み、ＵとＶのクロミナンス情報（ＹＵＶ４：１：１の場合）に対しては２×２ブロック毎に１バイトであるので達成される。コードブック・ベクトルの数を２５６から１２８に減らすことによって１２８・６＝７６８バイトの合計節約を生ずる。１２８のコードブック・ベクトルがＭＳＥについて適切な画質を与えるイメージに関して、節約された７６８バイトはサブ・サンプリングされるブロックの数を低減するのに用いられ、従って見る人に対して画質を向上させる。

共用コードブック

望ましい実施例によって与えられる別の特徴は、共用コードブックの使用である。１つのコードブックを１つ又は多くのフレームに共用させることは、コードブックのオーバーヘッドを低減するために類似の内容を有するフレームを利用できる。共用コードブックの使用は、無変化のブロックを用いて効率的にコード化できない時間的相関関係を利用する。このような場合の例はパン（pan）されたシーケンスである。もし２つのフレームが１つの２５６エレメントのコードブックを共用するとすれば、節約は各フレームに１２８エレメントのコードブックを別々に使用させるのと等しい。しかし、もしフレームが完全に異

なっていなければ画質は向上する。明らかに、別々に１２８エレメントのコードブックを用いる場合は、８ビットでなく７ビットのインデックスを用いることができるが、バイトの不整合によってビット・ストリームのパック／アンパックが扱いにくくなる。コードブックのオーバーヘッドの低減だけが共用コードブックの利点ではない。例えば、同一のコードブックを用いてイメージ間の時間的相関を増加することによって、時間的フリッカを低減できる。全く新しいコードブックがビット・ストリームからアンパックされなくても良く、また各フレームに関してＲＧＢに逆変換されなくても良いので、デコード速度の利益もある。

前のフレームから構築された共用コードブックがコード化しようとするフレームを良く表していることを確かめるため、共用コードブックは１つずつ新しいコードブックで置き換えることもできるし又は更新することもできる。第１に、フレームは共用コードブックを用いてコード化され、frame_mse（元のフレームとコード化されたフレーム間の平均２乗誤差）が計算される。もしframe_mseが前のフレームからのframe_mse又は前のフレームからの平均frame_mseよりもあるパーセンテージで大きければ、共用コードブックは新しいコードブックで置き換えられる。frame_mseがこのテストをパスしても、もしフレーム全体に関する平均ＭＳＥに対して、ある割合以上のＭＳＥを有するブロックの数が、ある数以上（最悪のブロック）であれば、共用コードブックは置き換えることができる。この場合、エンコーダはコードブックに対する更新だけで最悪誤差のブロックを修復するのは難しいと仮定し、コードブック全体を再生する。別のやり方では、エンコーダは先ずコードブック更新リストの生成を選択し、最悪誤差のブロックがいくつあるかチェックし、もしある閾値以上の悪いブロックがあれば全く新しいコードブロックを生成する。

ベクトル量子化の章で述べたように望ましい実施例は、共用コードブックの生成に用いたツリー構造を用いて共用コードブックを更新する。新しいフレームからの各イメージ・ベクトルはツリーのターミナル・ノードの１つと関連している（コードブックのベクトルと共に）。これは、ツリーのルートで始まり、どの子供が２乗誤差について近いか選択し、どの子供の子供が良く一致するかを選択する等によって達成される。イメージ・ベクトルは、このようにしてツリーのルート・ノードからターミナル・ノードへたどって行く。網羅的サーチを行えるけれども、網羅的サーチでなくツリー構造を用いてイメージ・ベクトルとコードブック・ベクトルとのマッチングをさせることによってコード化時間が向上する。更にツリー構造は、共用コードブックを更新するために、新しいノードを生成するのに役立つ。

コードブックの更新処理は数ステップを要する。先ず、９０１（イメージ・ベクトルと関連しないコードブック・ベクトル）のようなゼロ・セルが見つけられ、ツリー９００から除去される。このブランチは図９ａに示す。ゼロ・セルに関連するターミナル・ノード番号（コードブック・インデックス）が記録され、コードブックの更新は、ゼロ・セルであったコードブックの項目を置き換える。ツリー・ポインタは９０２が子供９１２及び９１３を指すように変えられる。これは図９ａに変形されたツリーとして示されている。ツリーは次に、改良ベクトル量子化装置３３０に関して上で述べたように、ある基準で選択されたノード（図９ｂ）（最悪の全体歪を有するｎ個のノード）を分割する。これは図９ｂに９２０のツリーを９３０のツリーに変形して示されている。ゼロ・セルのために捨てられ（９０１）、又は分割によって親になったターミナル・ノードは、新しい更新されたベクトルで上書きされるようにタグが付けられる。最後に、ノード分割からの新しい子供は、上書きのためタグを付けたこれらのコードブックを上書きする。実際の上書きはデコーダで発生し、上書き情報はビット・ストリームを通して与えられる（以下を参照）。もしゼロ・セルがなければ、各ノード分割は２コードブック・ベクトル・スロットを要し、そのうちの１つは分割前のそのノードの親のものである。残りの子供は捨てたコードブック・ベクトルに対する単なる置き換えでなく追加のコードブック・ベクトルとして伝送される。

コードブックの共用によって、１つのフレーム又はフレームのセットから全体的に生成されたコードブックは、最大のコードブック・サイズ（例えば２５６）より小さいサイズ（

例えば５０％）にセットされ、追加のコードブック・ベクトルを共用コードブックを用いて追加できる。
別の分割及び置き換え方法は、前にターミナル・ノードであった親を置き換える必要がない。その代わり、２つの子供のうち１つが親と等しいと制約することによって、その親は置き換えられなくて良い。他の子供はゼロ・セルを置き換えるか又は追加のコードブック・ベクトルとして送られる。
複数コードブック
別の実施例では、別々のコードブックを各ブロック・タイプに対して生成することによって、又はそのイメージの異なった領域に対して別々のコードブックを生成することによって、複数のコードブックを１つのイメージと関連づけることができる。前者は、圧縮の損失を最小（コードブックが共用の場合損失なし）にして画質を向上するのに効果的であり、後者は画質の損失を最小にして圧縮比を増大させるのに非常に効果的である。
別々のコードブックを用いてサブ・サンプリングされ及びサブ・サンプリングされていないイメージ・ベクトルをコード化することによって、従来技術のテクニックに比べていくつかの利点が得られる。独立したツリーが２つの異なったタイプのブロックの特性に対して特別に調整される。そのブロックはサブ・サンプリングされた領域には「スムーズ」となり、サブ・サンプリングされないブロックに対しては「ディテイル」となる。ブロック・タイプは空間的サブ・サンプリングの章で説明した誤差計算によって分けられる。「スムーズ」と「ディテイル」の領域の分離は、所望の圧縮がサブ・サンプリングを必要としないときに発生する。これは、「スムーズ」及び「ディテイル」のブロックが別々にコード化されているとき、別々のコードブックは非常によく動作するためである。各インデックスはブロック・タイプを通してコードブックと関連づけられており、コードブック・ベクトルの数はインデックス毎のビットを変えずに、又はＶＱのクラスタリング時間を増加せずに２倍にすることができることに留意されたい。これは、画質の顕著な向上をもたらす。更に、サブ・サンプリングされたブロックのコードブックと２×２Ｃブロックのコードブックは同じタイプの前のフレームのコードブックと共用できる。この場合、「スムーズ」領域と「ディテイル」領域を別々に維持し、数フレームに亘って各コードブックで一貫性があることが重要である。スムーズ及びディテイル領域への分離は、イメージのカテゴリに対して別々のツリーを定義するという一般的アイデアの特別の場合であることに留意されたい。カテゴリは、類似属性を有するイメージ内の領域を識別する分類子によって決められる。上で述べた簡単な場合には、２つのカテゴリ、スムーズ及びディテイルが用いられている。その他のカテゴリとしては端領域、テクスチャ及び平均値や偏差値など類似の統計を有する領域がある。
簡単に述べたように、複数のツリーがイメージ内の異なった領域と関連づけられる。これはコード化時間の短縮や圧縮比の増大に効果的である。例えば、粗いグリッド（等しいサイズの８個の矩形）は８個の１６エレメント・ツリーでコード化される。最悪誤差の矩形領域は、次に再び分割され、各矩形領域の各半分が１６エレメント・ツリーを用いる。これは１６個の矩形、従って合計２５６のコードブック・ベクトルになるまで続けられる。各インデックスは８ビットでなく４ビットを用いてコード化され、追加の２：１圧縮を与える。もしイメージが１６個の固定した初期領域に分割され、それ以上領域の分割がなければ、コード化計算時間は大幅に短縮される。このテクニックは特に低画質、高圧縮、高速コード化モードに適応する。イメージの小片に対して、多くの小さなコードブックを用いることとイメージ全体に対して１つの２５６項目のコードブックを用いることとの間の妥協は、画質がそれ以上悪くならなければ画質を維持し、もっと多くの圧縮を得るのに効果的である。このような妥協では、非常に均一でわずかなコードブック・ベクトルしか必要としないイメージの部分にのみ小さなコードブックが用いられ、正規の２５６項目のコードブックは残りのイメージに用いられる。もし小さなコードブックに関連するイメージの部分が矩形であると考えられれば、小さなコードブックにスイッチするとき、デコーダに知らせるのに殆どオーバーヘッドを要せず、小さなインデックス（１６項目のコードブックには４ビット又は６４項目のコードブックには６ビット）となる。各コードブックに

関連する領域が矩形であると考えられなければ、類似の画素を１つの領域にグループ化する当業者に知られた区画化テクニックで画質を向上できる。

レート制御

レート制御３４５は圧縮された素材が限定された帯域幅チャネルでデコードされると考えられるとき、改良ビデオ圧縮システムの重要なエレメントである。同期体系又はネットワーク又は電話線においてＮフレーム／秒を維持するため、デコーダ３５１は１／Ｎ秒間に限定された帯域幅チャネルから１フレームのデータを読み、情報をデコードし、イメージをスクリーン上に表示しなければならない。レート制御３４５は最大フレーム・サイズを、ある数（アプリケーションによって異なる）以下に維持し、限定された帯域幅チャネルからデータを読むのに要する時間を短縮しようとする。これは２つのステップで行われる。すなわち、（１）データ・レートの観点から望ましいデータ・レートは何かを決定する。（２）画質の要件（ユーザによって定義されるか又はその他の方法で定義される）と共にこの望ましいフレーム・サイズを用いてコード化処理のパラメータを制御する。

レート制御体系は、望ましいフレーム・サイズが何であるかを過去のパフォーマンスと望ましいデータ・レートに基づいて決定する。target_frame_length（目標フレーム長）は次のように計算される。

$$\text{target_frame_length} = \frac{\text{desired data rate}}{\text{desired frame rate}}$$

現在のフレームＮに対するdesired_frame_length（望ましいフレーム長）は、目標フレーム長から誤差項frame_errorを引いたものに等しく、あるフレーム数、例えばビデオ・データの１秒分の、平均である。

desired_frame_length=target_frame_length+frame_error

許容されるオーバーシュート又はアンダーシュートであるframe_errorはＩＩＲ（無限インパルス応答）フィルタとして帰納的に平均されることに留意されたい。これはまた、ＦＩＲ（有限インパルス応答）フィルタとして別の実施例で実施できる。αの値は、どれ位速く現在のフレーム誤差（target_frame_length_avg_frame_length）が長期フレーム誤差に応答させるかに影響する。現在の誤差はtarget_frame_lengthといくつかのフレーム（例えば１秒分）の平均フレーム長（avg frame length）として定義される。このレート制御体系は、望ましいデータ・レートを超えない過去１秒の平均データ・レートを維持する。フレーム・サイズのばらつきはフレーム単位で起こるが、これらのばらつきは平均効果で低下する。これらの関係は次のように決まる。

$$(frame_error)_n = (1-\alpha)(frame_error)_{n-1} + \alpha(target_frame_length - (avg_frame_length)_n)$$

$$(avg_frame_length)_n = \sum_{i=n-k}^{n} a_i * frame_size_i$$

$$\text{where} \quad \sum_{i=n-k}^{n} a_i = 1$$

desired_frame_lengthがＮに対して決まった後、それはコード化パラメータ（ncthreshfactor及びedge_mse）に影響を与えるために用いられ、時間的フィルタリング及び空間的サブ・サンプリングが用いられる実施例において、そのパラメータはどれ位時間的処理と空間的サブ・サンプリングを適用するかを制御する。これらのコード化パラメータはユーザによって決められる空間的、時間的画質の好みによってセットされるが、それらは、システムがどの程度データ・レートの要求を維持しているかに従って画質設定に関する変動が許される。短時間にこれらのパラメータを大きく変動させることを許すよりも、それらは次のように計算される長期誤差を追跡する。

$(long_term_error)_n = (1-\beta)(long_term_error)_{n-1} + \beta(target_frame_length) - (avg_frame_length)_n$

従って、long_term_error（長期誤差）に対する計算とフレーム誤差に対する計算の差異はαとβの差である。効果的であると決められた値は、α＝０．２０でβ＝０．０２で、これらは望ましい実施例で用いられる。当業者は他のα、βの加重値が使えることが分かるであろう。

もしlong_term_errorが空間的サブ・サンプリング及び無変化ブロックに対するコード化パラメータ値を制御するのに用いられなければ、望ましいフレーム長が、どの程度データ・レートが維持されているかを追跡するのに用いられる。この場合無変化及びサブ・サンプリングの閾値はユーザの画質設定によってのみ決まることを前提としている。しかし、これはサブ・サンプリングと無変化のブロックがフレーム・サイズをdesired_frame_size（望ましいフレーム長）に短縮することを保証するものではない。この場合long_term_errorの値は、サブ・サンプリング及び無変化ブロックのパラメータ（ncthreshfactor及びedge_mse）を変えることによって画質を低下させるのに用いられ、従ってデータ・レートを下げる。

コードブック・インデックスの伝送

イメージが改良された処理３３０によって、ベクトル量子化を通してコードブックに対するインデックスと関連づけられた後、ビット・ストリームは従来技術より効果的にパックされ、将来の変化と両立する柔軟性を可能にし、余分なコード化オーバーヘッドを生成することなしに、イメージをデコードするのに必要な情報を通信する。インデックスは、それぞれコードブックに対するインデックス又はコードブックのベース・インデックスからのオフセットとして伝送される。前者の場合、２５６項目のコードブックのどのベクトルが最も良くマッチするかを示すのに、イメージ・ベクトル当たり８ビットを必要とする。後者の場合、インデックス間の差は一般に２５６よりも大幅に小さいので、インデックス間に多くの相関があればより少ないビットでよい。イメージのある部分は互いに大きく離れたインデックスを有し、他の部分は強く相関するインデックスを有するので、通常２つの組み合わせが必要である。

図１０を参照して示すように、ビット・ストリーム・シンタックスはシーケンス・ヘッダ１００１、チャンク・ヘッダ１０１１、フレーム・ヘッダ１０２１及びコードブック・ヘッダ１０１２、１０１４を含む。コードブック・インデックスがこれらに続く。コードブック・インデックスは続くインデックスがどのブロック・タイプを参照するかを示すブロック・タイプ・ヘッダによって表される。２×２変化（２×２Ｃ）、２×２無変化（２×２ＮＣ）、４×４無変化（４×４ＮＣ）、４×４変化（４×４Ｃ）、サブ・サンプリングされた（４×４ＳＳ）、混合ブロックの異なった組み合わせ及び行画素ブロックは有用なブロック・タイプの例である。デコーダ３５１は、各イメージ・ブロックに対してどのコードブロック・ヘッダを用いるか及びアップ・サンプリングを行うかどうかを知って、イメージを再構築することができる。ビット・ストリーム・シンタックスについて以下に論ずる。

シーケンス・ヘッダ１００１はシーケンス全体に対する情報を伝達する。これらにはフレームの合計数、シーケンスがコードされたコーダのバージョン及びイメージ・サイズが含まれる。シーケンスは例えば動画全体を含む。単一のシーケンス・ヘッダ１００１は一連のイメージに先行し、シーケンスについての情報を指定する。シーケンス・ヘッダ１００１は殆どどんな長さでも良く、その長さを１つのフィールドに持つ。シーケンス・ヘッダに現在定義されているいくつかのフィールドを図１１に示す。シーケンス・ヘッダ１００１はシーケンス・ヘッダＩＤ１１０１を含み、それによってデコーダがシーケンス・ヘッダであることを識別することができる。これはユーザに対してランダム・アクセスの再生を可能にするアプリケーションには有用である。更にシーケンス・ヘッダ１００１は、シーケンス・ヘッダ１００１の長さを指定する長さフィールド１１０２を含む。次のフィールドはシーケンス内のフレーム数を指定するフレーム数フィールド１１０３である。これは整数値で符号なしの長語として格納され、シーケンスの長さが2^{32}個のフレームまで可能にしている。シーケンス・ヘッダの次のフィールド１１０４は、現在予備として確保されており、次の２つのフィールド、１１０５及び１１０６はシーケンス内のイメージの幅

と高さを指定する。シーケンス・ヘッダ１００１の最後のフィールドはバージョン・フィールドで、使用されるエンコード／デコード装置の現在のバージョンを指定する整数フィールドである。これは特定の特性を持っていたり、持っていなかったりする新しいシーケンスと古いシーケンスを区別するためのものである。これはシーケンスとエンコード／デコード体系の上位及び下位の互換性を可能にする。シーケンス・ヘッダはまた、イメージのシーケンスを指定するＡＳＣＩＩ又は文字列を含む（図示せず）。

図１０に戻って、チャンク・ヘッダ１０１１は、共用コードブックが用いられているかどうかの、フレームの次のチャンク（chunk）についての情報を伝達するチャンク・タイプを持つ。チャンク・ヘッダはフレームのチャンクに対していくつのコードブックが用いられているかを指定することができる。チャンク・ヘッダ１０１１はフレームの「チャンク」に先行する。望ましい実施例において、チャンクは１つ又は多くのフレームで場面変化検出アルゴリズムのような装置によって他の「チャンク」から区別できるものである。別の実施例ではフレームのグループは、レート制御メカニズムのような別の技術を用いて関連づけられる。

２つのコードブック・ヘッダが図１０のシーケンス例１０００に示されており、これによって１フレーム当たり２つのコードブックの使用が可能となる。２つのコードブックを使用する例は、固定コードブック（フレームの「チャンク」に対して静的）と順応性コードブック（フレーム毎に変わる）の使用である。コードブックのタイプとサイズは図１３ａに示すようにコードブック・ヘッダ１０１２と１０１４に含まれる。図１０の１０１２又は１０１４のような各コードブック・ヘッダは、コードブック・タイプ・フィールド１３０１を有し、コードブック・タイプ、例えば固定か順応か、を指定する。コードブック・タイプはＹＵＶ（サブ・サンプリングされたＵＶ又はサブ・サンプリングされないＵＶ）、ＲＧＢ及びＹＵＶ更新コードブックを含む。その他のタイプは、本発明の精神と範囲の中で考慮されている。コードブックの「更新」については、コードブックに対する更新はコードブック・ヘッダに続いて送られる。コードブックのサイズはフィールド１３０２にバイト数で指定され、デコーダは、いつ次のフィールドが始まるかを検出できる。もしコードブック・タイプが「更新」コードブック（すなわち共用コードブック）なら、図１３ｂに示す情報１０１３（又は１０１５）がコードブック・ヘッダ１０１２（又は１０１４）に続いて期待できる。この更新コードブックは、更新が必要なコードブック項目を識別するビットマップ１３７０を有する。このフィールドの後に、更新される各ベクトルに対するベクトル更新１３７１－１３７３が続く。このようにコードブック全体が再生成されるのではなく、選択された部分だけが更新され、更にデータ・レートの低減が行われる。もしサブ・サンプリングされたU及びＶと共にＹＵＶが用いられると、更新ベクトル１３７１－１３７３のそれぞれは６バイトから成り、４バイトはブロック内の各画素の輝度のためであり、１バイトずつがそれぞれＵとＶのためである。コードブックの更新について図９ａ及び９ｂを参照して説明した。

更にコードブックのオーバーヘッドを低減するために、１０１３及び１０１５のようなコードブックはＹＵＶ（輝度とクロミナンス）フォーマットに変形され、ＵとＶは水平及び垂直方向（ＹＵＶ４：１：１）に、ファクタ２でサブ・サンプリングされる。従って、コードブックはサブ・サンプリングされた情報を伝送することによりサイズが更に小さくなり、コードブックのサイズを２のファクタで低減する。

図１２を参照して示すように、フレーム・ヘッダ１０２１はイメージ・サイズを幅フィールド１２０１と高さフィールド１２０２とに有し、フレーム・サイズの変動をいつでも可能にする。フレーム・ヘッダ１０２１はフレーム・タイプ・フィールド１２０３を有し、そのビット・パターンは、スキップ・フレームに対するヌル（null）フレームであるか、全体がサブ・サンプリングされたフレームであるか、キー・フレームであるか、又はフレームが他のフレームとコードブックを共用しているかを示す。他のタイプのフレームは、本発明の精神の中で考慮されている。サブ・サンプリングされたゾーン・フィールド１２０４は３２ビットのビット・マップ・パターンであり、どのゾーンが（もしあれば）サブ・サンプリングされたかを示し、望ましい実施例では最大2^{32}ゾーンまで可能にしている

。
図１４の部分１０２２に示すブロック・ヘッダは、デコーダ３５１に対してどのタイプのブロックがインデックスのセットと関連しているか及びいくつのインデックスがそのセットにあるかを知らせる。これは図１４を参照して示される。ヘッダ１４０１の最初の３ビットは、続くインデックスのセットが２×２Ｃブロック（変化ブロック）であるか、４×４ＮＣブロック（無変化ブロック）であるか、４×４ＳＳブロック（サブ・サンプリングされたブロック）であるか、混合ブロックであるか又は行画素値であるかを示す。もし最初の３ビットが、ブロック・タイプは混合でないことを指定すると、ヘッダ１４０１の最後の５ビットはいくつのインデックス１４０２がブロック・ヘッダ１４０１に続くかを示す整数である。これは「ランレングス（runlength）」ブロック・ヘッダと呼ばれる。ブロック・ヘッダは２×２Ｃと２×２ＮＣブロックの混合のような混合ブロックを指定する。この場合、長さのために確保されたヘッダの５ビットは、２×２Ｃと２×２ＮＣブロックの混合のうち、いくつの４×４がコード化されたかを指定する。別のやり方では、５ビットのうち１ビットは、もっと多くの混合の可能性を許すために用いられる。ビット・マップが続き、一番近いバイトにパッドされる。２×２Ｃ－２×２ＮＣ混合の例において、ビット・マップは「１」でブロック・タイプが２×２Ｃであることを、「０」でブロック・タイプが２×２ＮＣであることを示す。ブロックは４×４の単位でも混合できる。もしビット・マップ・ヘッダがビット数をランレングス・ヘッダよりも少なくすれば、計算が簡単になる。「１００１０１１０１０１」のような一連の交互のブロック・タイプは、ビット・マップ・ブロック・ヘッダでうまくコード化され、１つのヘッダ・タイプが長く続くようなもの（例えば１１１１１１１１１０００００００００）はランレングス・ヘッダ・タイプでうまくコード化される。ブロックをより効率的にコード化するブロック・ヘッダが選択される。ビット・マップ・ヘッダは、頻繁に発生する短く続くブロックの効率的コーディンタを可能にする。
「変化」ブロックの流れの中間において、「無変化」ブロックとタグを付けられたブロックの前後におかれるブロック・タイプ・ヘッダ１４０１の２バイトのオーバーヘッドのために、もし１行に少なくとも４個の２×２無変化ブロックがあれば、望ましい実施例のランレングス・ブロック・ヘッダは、ヘッダを有するインデックスの構造を乱すだけである。１４１０のようなヘッダを有するビット・ストリームにおいて区別するために、望ましい実施例のランレングス・ヘッダは４個の２×２ＮＣ（無変化）ブロックが一緒に発生し、１つの４×４ＮＣ（無変化）を作ることを要する。同じ位置の、前のフレームのブロックが代わりに用いられるので、次に続くＮ個のブロックが４×４ＮＣ（無変化）タイプであることを示す１４１０のようなブロック・ヘッダは、インデックスについてバイトを浪費する必要ない。デコーダ３５１は新しいイメージに対して、いくつのブロックをスキップするかを知る必要があるだけである。実際の画素値又は特異な２×２ブロックが用いられるので、１４０２のような２×２Ｃブロックのインデックスは、４個のセットで発生する必要がない。もしある実施例で、実際の画素値又は特異な２×２Ｃ及び２×２ＮＣブロックがサポートされていなければ、２×２Ｃブロックが４個のセットで発生することを仮定することは、１４０１のような２×２Ｃブロック・ヘッダと関連するブロック数を増加し、従ってブロック・ヘッダによる効果的オーバヘッドを減少させる。例えば、もし２×２Ｃブロックがサポートされなければ、１つのブロックは８個の２×２Ｃ（変化）ブロックを識別し、それを８グループの４－２×２Ｃブロックの意味として解釈する。（２－２×２Ｃブロックが２セットの４－２×２Ｃブロックとして解釈される図１５、１６の例を参照）。
更に、２×２Ｃブロックを参照する図１４のインデックス１４０２は、４×４ＳＳブロックを参照するインデックス１４２１と同じコードブックからである必要はない。このビット・ストリームの柔軟性は、２５６以上のコードブックを有し、バイト整合されないインデックス・サイズ（５１２のコードブック・ベクトルに対する扱いにくい９ビットのような）にジャンプすることなしに、非常に少ない圧縮の低下で高い画質のサポートを可能にする。

インデックス・パッキング

もしイメージ・ブロックがコードブック内で近接しており、類似のＲＧＢカラー空間にあれば、ビット・ストリームに単にリストするのでなく、インデックスをコード化するとき、ベース・アドレスを用いるのが有利である。コードブック・ベクトルは「最悪」誤差のノードを分割して生成されるので、類似のイメージ・ベクトルがコードブック内に近く集まる傾向がある。似たイメージ・ブロックがイメージ空間で発生する傾向があるので（すなわちブロック間で空間的相関がある）、互いに近いインデックス値が一緒に発生する傾向がある。コードブック・インデックスの割り当てもまた、空間でのインデックスの差が最小となる方法で行うことができる。どのようにこれを用いてロスなしにビット数を減らすかの例が図１５及び１６に示され、説明されている。このパッキング処理は、図３のエンコーダ３０１の３４０で行われ、アンパックはデコーダ３５１の処理３７０で行われる。

図１５で、もしコードブックが２５６の項目を持っていれば、ビット・ストリーム１５００のコードブック・インデックスは、それぞれ８ビットを必要とする。言い換えれば、各インデックスはコードブックのエレメントに対して完全な参照を有する。上で論じたように、空間的相関のため、これらのインデックス値はベース・アドレスからのオフセットを用いることによって、もっと低減できる。これは図１６に示されている。図１６において、コードブック・インデックスのそれぞれは、もしインデックスが伝送されたベース・アドレスから、－８から＋７であるオフセットとして表されれば、わずか４ビットを要する。これはビット・ストリーム１６００の１６０１に示される。ベース・アドレス１６０１は開始点として用いられ、１６０４のような現在のブロックのオフセット値は、現在ブロック１６０３に先行するインデックスにおける変化を参照することができる。ベース・アドレス・ヘッダ１６０１はベース・アドレスを定義して送られることを要し、差のコードが用いられる。大きく、可変であるコードブック・インデックス（コードブックの一端から他端）のセットを有する領域は、図１５に示す完全なインデックスの伝送を用いて効率的にコード化され、ブロック・レベルで類似の領域は、図１６に示す１６００のようなビット・ストリームを用いてより効率的にコード化される。図１６に示すように、ベース・アドレスからのオフセットを用いることは、元のインデックス値がベース・アドレスにオフセット値を加えることによって計算できるので、図１５に示すテクニックと同様無損失である。

ビデオ・データを圧縮、圧縮解除する発明について記載した。この明細書において、本発明は図１から図１６の特定の実施例を参照して説明された。しかし、添付請求の範囲に述べたように、本発明の広い精神と範囲から離れることなく様々な修正や変更がなされ得ることは明らかであろう。従って、明細書及び図面は、例示と考えるべきであり、限定と考えるべきでない。

【図１ａ】

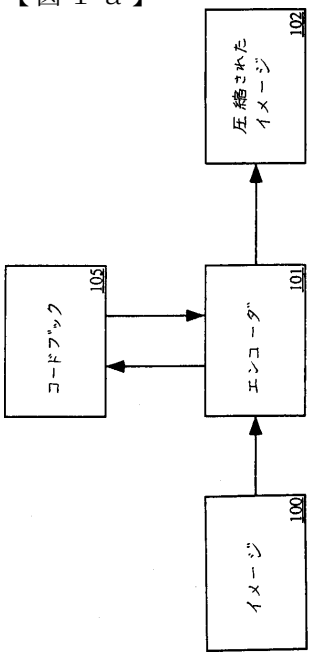


FIG. Ia（従来技術）

【図１ｂ】

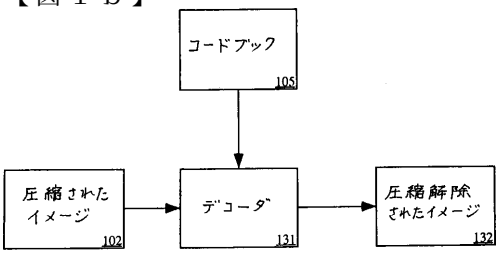


FIG. Ib（従来技術）

【図１ｃ】

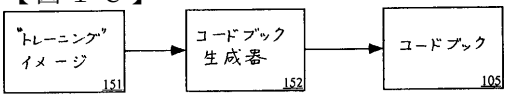


FIG. Ic（従来技術）

【図２】

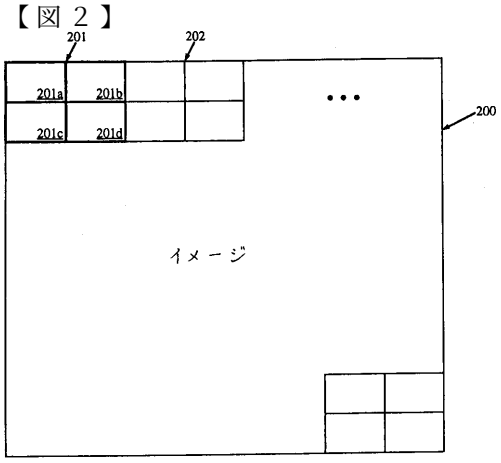


FIG. 2（従来技術）

【図３】

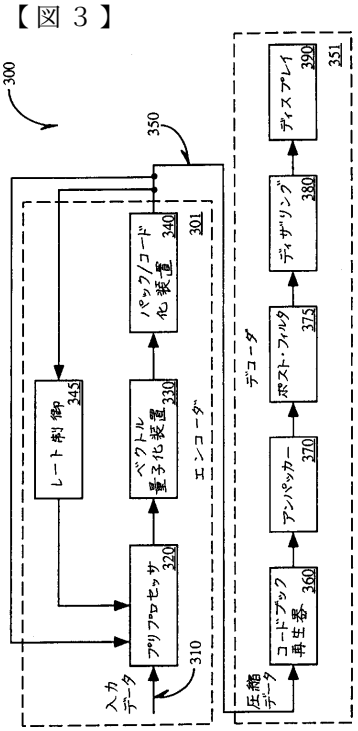


FIG. 3（従来技術）

【図４】

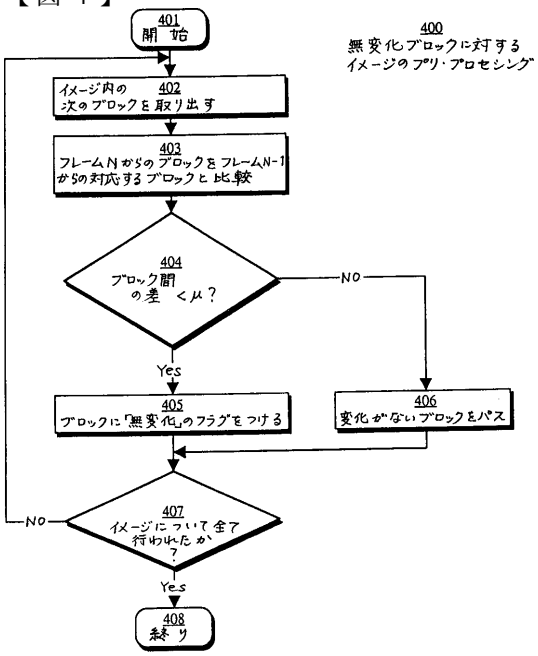


FIG. 4

【図５ａ】

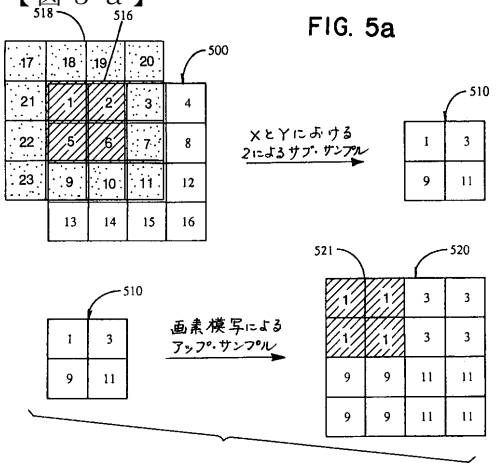
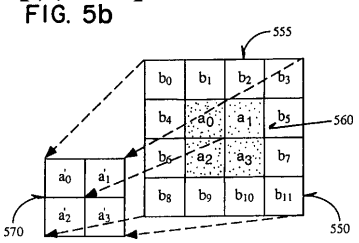


FIG. 5a

【図５ｂ】

FIG. 5b



【図６】

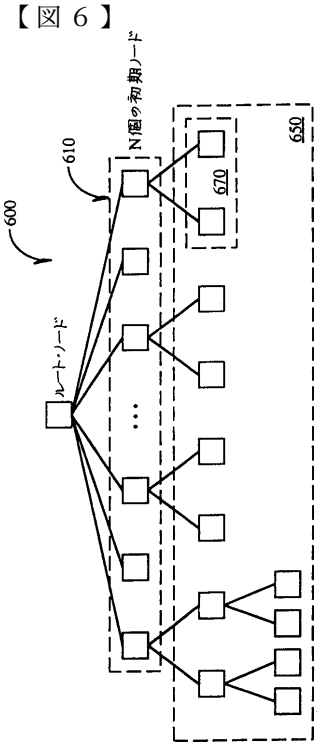


FIG. 6

【図７】

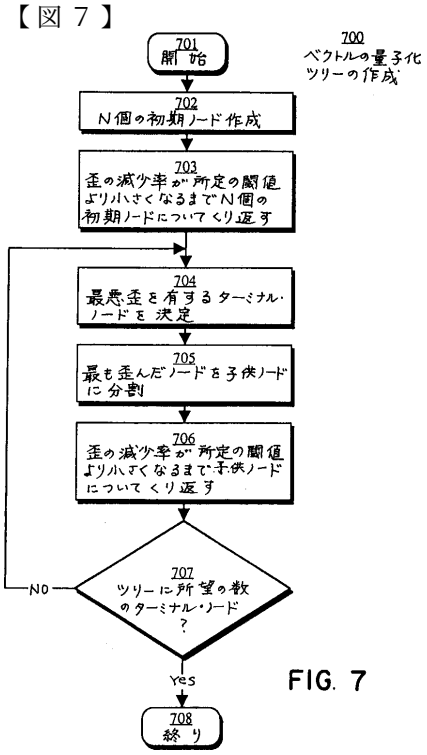


FIG. 7

【図８】

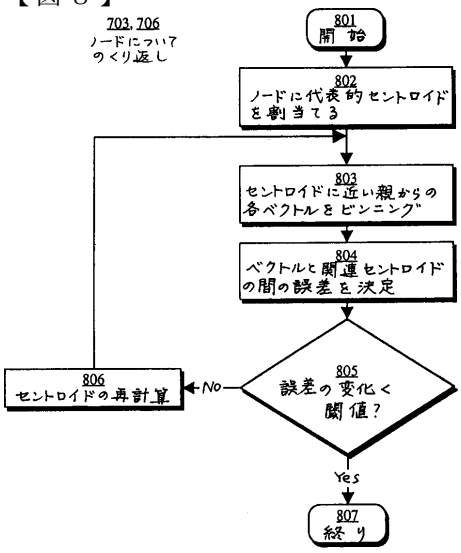


FIG. 8

【図９ａ】

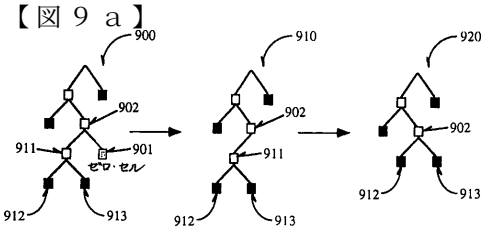


FIG. 9a

【図９ｂ】

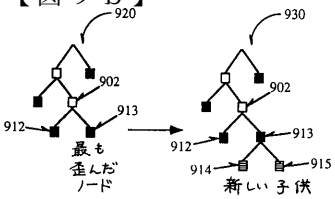


FIG. 9b

【図１０】

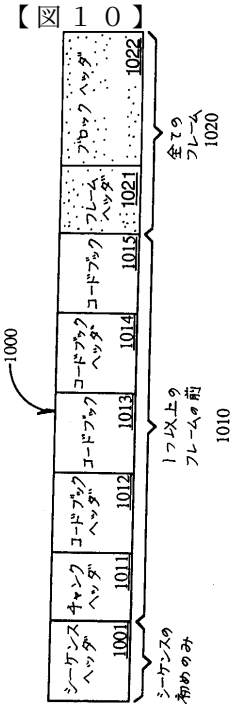


FIG. 10

【図１１】

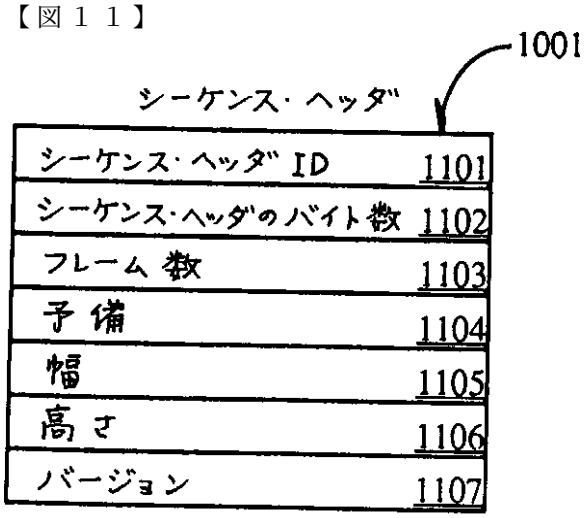


FIG. 11

【図１２】

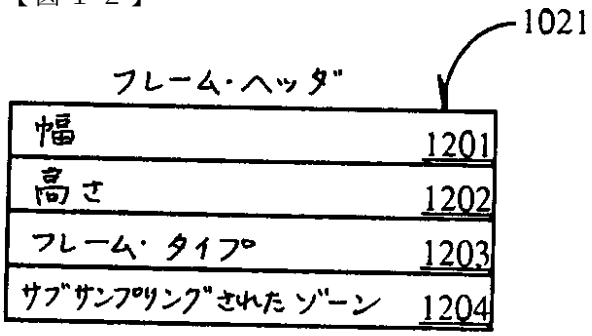


FIG. 12

【図１３ａ】



FIG. 13a

【図１３ｂ】

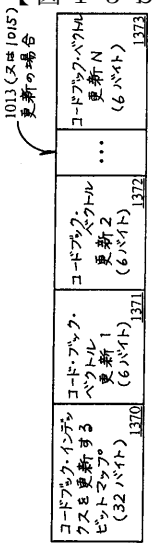


FIG. 13b

【図１４】

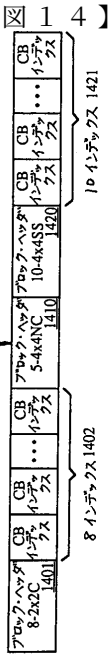


FIG. 14

【図１５】

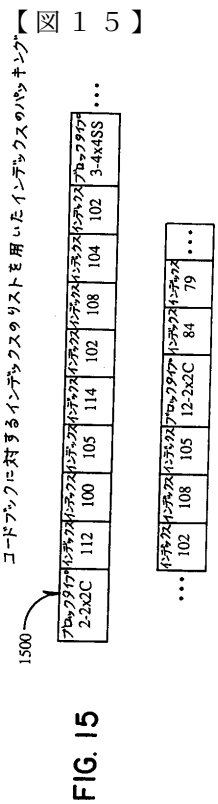


FIG. 15

【図１６】

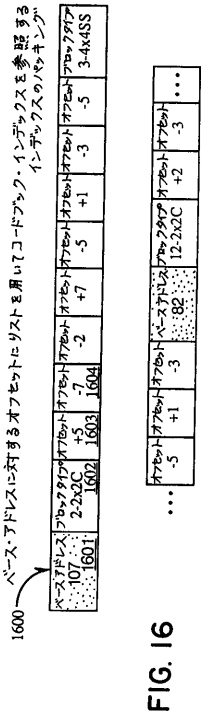


FIG. 16

フロントページの続き

(74)代理人
弁理士 鈴木 二郎
(74)代理人
弁理士 山川 茂樹
(72)発明者 ノーマイル, ジェイムズ・オリバー
アメリカ合衆国 ９４０８７ カリフォルニア州・サニー ヴェイル・テナカ プレイス・ナンバーブイ１・１５７５
(72)発明者 ワング, キャサリン・シューウェイ
アメリカ合衆国 ９５１２９ カリフォルニア州・サン ホゼ・ウィルミントン アヴェニュ・１０７２
(72)発明者 ウー, シージュン
アメリカ合衆国 ９５０１４ カリフォルニア州・カッパチーノ・ローズウッド ロード・１０６５エイ

審査官 菅原 道晴

(56)参考文献 米国特許第４７３３２９８（ＵＳ，Ａ）
米国特許第４９４１１９４（ＵＳ，Ａ）
米国特許第５１２１１９１（ＵＳ，Ａ）
米国特許第５１２４７９１（ＵＳ，Ａ）
特開昭６２－０３２７８７（ＪＰ，Ａ）
特開昭６０－１４６３６３（ＪＰ，Ａ）
NASSER M.NASRABADI, ROBERT A.KING, ”Image Coding Using Vector Quantization: A Review”, IEEE Transactions on Communication, 米国, IEEE, １９８８年 ８月, Vol.36, No.8, pp.72-86

(58)調査した分野(Int.Cl.[7], ＤＢ名)
H04N 7/24 - 7/68
H04N 1/41
H03M 7/30