処理量可変アルゴリズムによるリアルタイムソフトエンコーダの検討

大迫 史典　正満 峰夫　石橋 聡　小林 直樹

NTT ヒューマンインタフェース研究所
〒239-0847 神奈川県 横須賀市 光の丘 1－1
TEL: 0468-59-2831 FAX: 0468-59-2829
E-mail: osako@nttvdt.hil.ntt.co.jp

近年のコンピュータの CPU 能力の伸びに対応して，ソフトウェアによって画像の符号化および復号化を実行する機会が増えている．特にリアルタイムソフトウェア符号化では，使用可能な CPU パワーが時々刻々と変化する状況下で動き補償等莫大な演算を行う必要がある．筆者らは先に，ソフトウェア CODEC に適した符号化・復号化アルゴリズムの概念として，処理時間拘束のもとで画質の高い復号画像を提供できる演算量スケーラブルアルゴリズムを提案した．今回，典型的な符号化アルゴリズムとして，動き補償と直交変換を用いた場合の符号化処理を例にとって，演算量可変方法について検討を行い，演算量スケーラブルアルゴリズムにより，H．263符号化をリアルタイムで実行できる見通しを得たので報告する．

A Dynamic Computation Resource Scalable Algorithm for Real-time Software Video CODEC

Fuminori Osako　Mineo SHIYOUMAN　Satoshi ISHIBASHI　Naoki KOBAYASHI

NTT Human Interface Laboratories
1-1 Hikari-no-Oka, Yokosuka City, Kanagwa Pref., 239-0847 JAPAN
Phone: +81 468 59 2831 FAX: +81 59 2829
E-mail: osako@nttvdt.hil.ntt.co.jp

These days video coding is often performed using software alone. One of the problems in developing software codecs, however, is how to treat computation resource variation. This paper proposes a new concept in software video coding called a dynamic computation resource-scalable algorithm. It is a coding-time constrained resource-scalable algorithm for controlling the computation complexity of modules in coding software. Scalable methods are considered for motion compensation (MC) and discrete cosine transform (DCT) modules in the H.263 encoder side. We applied this algorithm to real-time software codec. Simulation results show that decoded images with high quality can be obtained by the proposed method even when the amount of computing resources varies.

1. はじめに

動画像の圧縮技術に関しては，ITU-H.261[1] や MPEG1[2]符号化方式をはじめとして，汎用的な高品質動画像圧縮標準アルゴリズム MPEG2[3]符号化方式等，マルチメディア時代を支える基盤技術の1つと考えられ，多くの研究・開発や製品化がなされている[4]. 従来，動画像処理は符号化処理ボード等専用ハードウェアで行っていたが，MMX 等の CPU 処理能力の向上により，WS や PC 上でソフトウェアのみ[5]で映像の圧縮/伸長を行うことが可能になってきた.

ソフトウェアのみによって，定められた時間内(例えばリアルタイム処理)に符号化処理を終了するという課題が課された場合の問題点として，WS や PC の能力によって処理時間が異なる，同じ WS や PC でもその時の負荷状態によって使用可能な CPU パワーが時々刻々と変化する，等を挙げることができる. このような CPU パワーの変動に対してどのように対処していくのかが，ソフトウェア CODEC 実現のキー技術である. この問題を解決するためには，使用可能な CPU パワーに応じて，符号化・復号化アルゴリズムの複雑さ，言い替えれば演算量を変化させれば実現できる.

筆者らは先に，符号化対象を可変にするのではなく符号化に要する処理量を可変にすることにより，使用可能な演算量を制御する演算量スケーラブル符号化を提案した[6]-[9]. 本稿では，この演算量スケーラブルアルゴリズムをリアルタイム符号化へ応用する検討を行ったので報告する.

2. 動的演算量可変アルゴリズム

2. 1 演算量可変方式

本稿では，典型的な符号化アルゴリズムであり，かつ MPEG 方式とも共通性の高い ITU-T H.263[10]を対象として検討を進める. H.263 は動き補償予測と離散コサイン変換を基本とした圧縮アルゴリズムであり，演算量を可変にできるモジュールとしては，動きベクトル検出部および，DCT/逆 DCT 部等を挙げることができ，さらに動きベクトル検出部の中でも，動きベクトル評価部や探索範囲などはそれぞれ独立した可変モジュールとして扱うことが可能である. また，動き検出部及び DCT/逆 DCT 部の演算量が占める割合が多いので，図1に示すように，これらのモジュールの演算量を変化させることにより，符号化全体の演算量の増減レンジを大きくすることが可能であると思われる. よって本検討では演算量可変モジュールとして，動き検出部と DCT 変換部に焦点を絞って議論を進める.

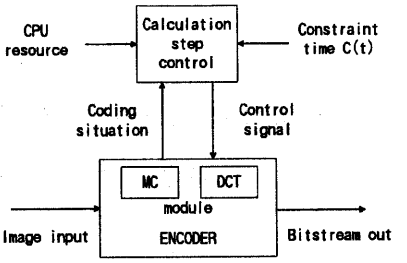


図1. 演算量スケーラブルアルゴリズムの概念

2. 2 演算量可変モジュール

2. 2. 1 動きベクトル探索時評価関数

動きベクトル探索では，探索範囲内の各動きベクトルに対して，動き補償の単位であるマクロブロックをベクトル分だけずらしてマッチングを行ない，対応する画素間の差分絶対値和が最小となるものを，最終的な動きベクトルとして選択する. ここでは，差分絶対値和演算対象の画素数に着目し，演算量可変制御に利用する. すなわち，多くの演算が行なえる状態ではマクロブロック内すべての画素によって評価し，演算量を削減する状態ではその削減量に応じて差分絶対値和を計算する画素数を決定する. 評価に用いる画素数が減っても，評価結果に大きな違いがないように，図2に示すように，水平方向に I 間隔，垂直方向に J 間隔ごとの画素 $16/I \times 16/J$ 個のみを用いて差分絶対値和を計算する.

原画像 $f(i,j)$，1フレーム前の画像 $f(i,j)$に対して，ベクトル(u,v)に対する評価関数 $E(u,v)$として式

$$E(u,v)=\sum_{i=0,I}^{15}\sum_{j=0,J}^{15}\left|f(i,j)-f'(i+u,j+u)\right| \qquad (1)$$

により指定された画素のみの差分絶対値和を用いる. このとき演算量圧縮比 Q_1は，

$$Q_1=\frac{16/I\times 16/J}{256}=\frac{1}{(I\times J)} \qquad (2)$$

となる.

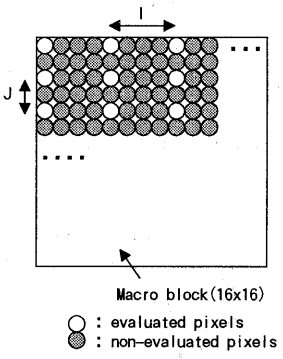


図2. 動きベクトル評価画素数

2. 2. 2 動きベクトル探索範囲

ベクトル探索範囲を水平・垂直方向ともに $\pm R$ とし, R を変化させることで演算量を可変にする. この場合, 常に不動位置に相当する(0,0)ベクトルを中心として探索範囲を $\pm R$ とする手法(ゼロベクトル参照手法)も考えられるが, 動きがある程度大きい場合には全く予測が当たらないことになる. そこで, 図3に示すように, 近傍ブロックにて予測されたベクトルを中心として探索範囲を $\pm R$ とするような, 近傍ベクル参照手法を用いる. 近傍ベクトル参照手法の場合には, 参照ブロックが大きく離れてしまうとベクトル予測がはずれてしまうという欠点がある. これを防ぐため, 水平 B_H, 垂直 B_V ブロックごと(これを基本ブロックと呼ぶ)に常に(0,0)ベクトルを中心として $\pm R_0 \times \pm R_0$ の探索範囲をサーチすることにより, リフレッシュする. 探索にあたっては, まず, 基本ブロックを探索し, その他のブロック(従属ブロック)は, 最も近傍にある基本ブロックで求めたベクトルを中心として $R \leq R_0$ の範囲で探索を行なう. ここで, R_0 は, 演算量を全く削減しない場合の探索範囲である.

演算量圧縮比は, $\pm R_0 \times \pm R_0$ 内の全探索をベースに考えた場合, ゼロベクトル参照手法の場合,

$$Q_{2Z} = \begin{cases} \dfrac{(2R+1)\times(2R+1)}{(2R_0+1)\times(2R_0+1)}; & (R \geq 1) \\ 0; & (R=0) \end{cases} \qquad (3)$$

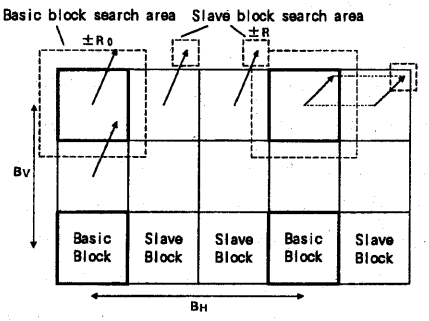


図3. 近傍ベクトル参照形動き探索方法

近傍ベクトル参照手法の場合,

$$Q_{2Z} = \begin{cases} \dfrac{1}{B_H B_V} + (1 - \dfrac{1}{B_H B_V}) \\ \quad \times \dfrac{(2R+1)\times(2R+1)}{(2R_0+1)\times(2R_0+1)}; & (R \geq 1) \\ \dfrac{1}{B_H B_V}; & (R=0) \end{cases} \qquad (4)$$

となる.

演算量削減率が小さくて良い場合には (B_H, B_V) のリフレッシュ制御を小さく, R を大きく設定し, 削減率を大きくする場合には (B_H, B_V) を大きく, R を小さく設定すれば効率がよい. また, 基本ブロックの間隔が小さい場合には従属ブロックの探索範囲は比較的小さくても良い.

2. 2. 3 直交変換処理

直交変換・逆直交変換は, エンコーダでは動きベクトル探索についで演算量が多く, デコーダでは逆直交変換のみであるが, 最も演算量を多く消費するモジュールである. DCT 部における演算量を変化させるには, DCT 係数算出個数を制御すればよい. この場合, 非算出係数については強制的に0とする. 図4(a)に示すように, どのブロックについてもジグザグスキャンの順序で常に低周波から定められた個数の係数のみを計算するようにすれば可変制御は可能である. すなわち視覚的な影響が比較的小さい高次係数を演算せずに0に置き換えれば良い. しかし, このような一律高周波数カット形の方法では, 常に高次の係数が0になるため, エッジの部分やテクスチャ領域に大きな歪み

が発生することは避けられない.

そこで, 図4(b)に示すように, ブロックごとに演算する DCT 係数を適応的に変化させる手法を考える. 適応化は動き補償予測誤差を参照する手法を導入する. すなわち, 動き補償予測誤差が大きいブロックは高次係数まで有効係数が存在する可能性が大きい, と仮定し予測誤差の小さいブロックに対してはその値に応じて計算する DCT 係数の個数を減らし, 予測誤差の大きいブロックに対しては予測誤差の小さいブロックに比べてより高次の DCT 係数まで計算するように制御する. この仮定の妥当性を確かめるため, 予測誤差(*E*)と有効係数(量子化後のゼロでない係数)のうちジグザグスキャンの順序で最高次の位置(*C*)の関係を調べる.

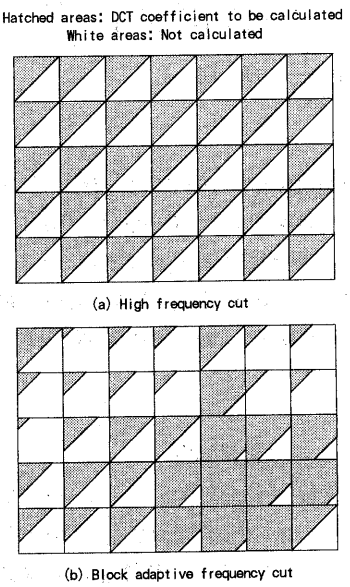


図4. DCT係数演算個数

E としては, フレーム間モードのブロックについては動き補償予測誤差の標準偏差, フレーム内モードのブロックについてはブロック内画素標準偏差を用いた. 図5は画像MIT-sequenceについて, 固定量子化ステップで量子化を行なった場合の, 上記 *E* と *C* を各ブロックごとにプロットしたものである. 図5より, *E* が小さい場合には, 最高次位置が小さいことがわかる. この最高次位置よりも高次の係数はすべて0であることから, 演算する必要がない.

次に, 量子化ステップ依存性を評価する. 図6は, 4種類の動画像シーケンスについて, 予測誤差(*E*)の値ごとに, 量子化 DCT 係数のうちジグザグスキャンの順序で最も高次に出現したゼロでない係数の位置の平均値 $\overline{C}(mq,E)$ を示したものである.

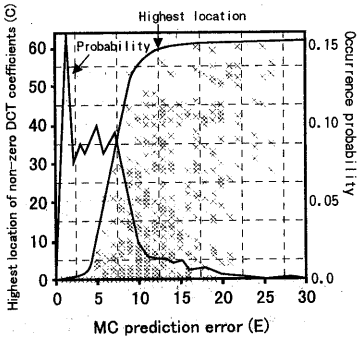


図5. MC差分標準偏差と非ゼロ係数最高次位置との関係

図6は輝度信号に対するものであり, 図中の *mq* は DCT 係数量子化パラメータであり, 実際の量子化ステップは *mq*×2 である. 図6から, 量子化パラメータに依存せず, 予測誤差(*E*)に比例して,有効係数の最高次数位置 $\overline{C}$(mq,E)が高くなることがわかる. ただし, 比例の程度は, 量子化パラメータ *mq* によって異なり, 細かい量子化が行なわれる場合には予測誤差が小さくても多数の係数が有効になるのに対して, 粗い量子化が行なわれる場合には, 予測誤差が比較的大きくても有効係数最高次位置は小さくなる. したがって, 予測誤差参照手法に量子化パラメータによる調整項を加え, ブロックごとの動き補償予測誤差のブロック内 RMSE 値(*E*), および量子化パラメータ *mq* に基づいてブロックごとに計算すべき係数の個数 *C(mq,E)*を求め, 各ブロックごとに定められた *C(mq,E)*個のみの DCT 係数を計算するようにすれば良い. 具体的には, まず, 図6の測定結果に基づいて, 動き補償予測誤差のブロック内 RMSE 値 *E*, 量子化パラメータ *mq* と演算すべき DCT 係数最高次位置 *C(mq,E)*の関係を定式化する. ここで, 実測された, $\overline{C}(mq,E)$のばらつき

を考慮して，図6に示した平均値に，各(mq,E)ごとの標準偏差$\sigma(mq,E)$を加えたものを基準とする．例として輝度信号に対して，

$\overline{C}(mq,E) + \sigma(mq,E)$を測定したものを図6に示す．これに基づいて，演算すべきDCT係数最高次位置を，

$$C_y(mq,E) = \frac{72 \times E}{mq} - 20 \qquad (5)$$

で定式化する．

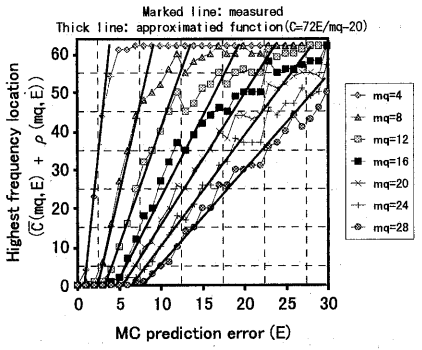


図6. 非ゼロ係数最高次位置(平均値＋標準偏差)と定式化関数

定式化された関数を図6中に太線で示す．ここで，$C_y(mq,E)$は，$0 \leqq C_y(mq,E) \leqq N$ にクリッピングする．クリッピング係数Nは$0 \leqq N \leqq 63$を満たす整数であり，通常63に設定される．予測誤差Eは，ほとんどの画像に対して$20/256$以下の範囲に8割以上が分布し，この範囲において上記のモデル式は，有効係数最高次位置を極めてよく近似している．同様にして色差信号に対しては，

$$C_c(mq,E) = \frac{36 \times E}{mq} - 10 \qquad (6)$$

で定式化する．式(5)，式(6)を用いて，そのブロックの演算すべきDCT係数最高次位置を輝度信号・色差信号それぞれ$C_y(mq,E)$, $C_c(mq,E)$とすることで演算量制御を行う．

本アルゴリズムは，本来ゼロである確率が非常に高い高次係数を演算しないのであるから，量子化ステップ決定にほとんど影響を与えずSNRにはほとんど影響を与えない．また，符号化ビットレートが低くなるにつれて演算量を大きく削減することができると言える．これはビットレートが低くなると結果的に量子化後に0となってしまう係数が増えるため，強制的に0にしても影響が少なくなるためである．

3. シミュレーション実験と考察

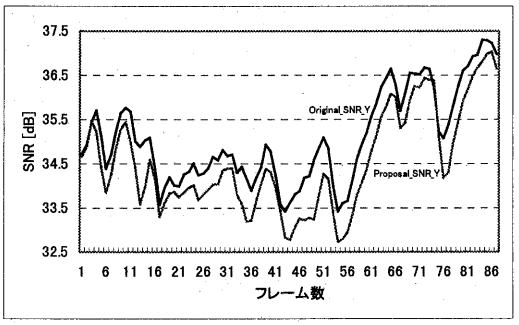
3.1 シミュレーション条件

H.263 において，フレームレート 30 frame/sec，画面サイズが CIF で動きベクトル探索範囲(15x15)の演算量は約 4800MOPS，QCIF サイズで探索範囲(15x15)の演算量は約 1300MOPS である．現状の高速な CPU で約 500MIPSであるので，動き検出も含めてそのままでリアルタイムで符号化するのはまだ困難である．そこで，提案手法を H.263 ベースとしたアルゴリズムへの組み込みを行った．また，この場合演算量スケーラブルアルゴリズムの拘束時間を 1 フレームあたり 1/30 秒に時間を設定することがリアルタイム処理に相当する．

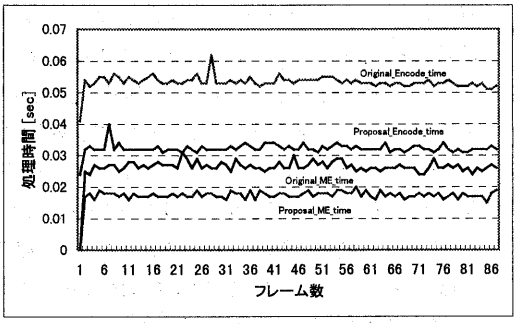
評価用画像としては，Mother&daughterを用いた．画像のサイズは QCIF であり，ビットレートは64K bits/secに設定した．H.263 画像符号化評価用ソフトウェアと演算量制御を組み込んだものとの比較を行い，マシン環境は CPU: Pentium II 300MHz, OS: Windows95上で実験を行った．

3.2 演算量制御シミュレーションと考察

本稿で検討した演算量制御手法は3つの演算量可変モジュール(ベクトル評価画素数，近傍ベクトル探索範囲，適応DCT係数演算個数設定)を組み合わせることにより，H. 263オリジナルの場合の約25%の演算量で処理を行った．ここで，H.263オリジナルの設定は，15x15 の範囲のスパイラル探索，ハーフペル予測精度，FAST-DCT，オプションなし，の条件で行い，符号化処理全体としての演算量に対するSNRの変化をシミュレートし，符号化品質を比較した．



(a) フレーム毎のSNRの変化



(b) 1フレーム当たりの処理時間の変化

図7. 演算量制御手法とオリジナルとのシミュレーション結果

図7に提案手法とオリジナルのフレーム毎のSNR(輝度信号)と符号化処理時間(動き補償と全処理時間)の実験結果を示す. これより演算量制御を用いた場合, H. 263オリジナルと比較してSNRを平均で約0. 55dBの劣化に抑えつつ, 1フレーム当たりの平均処理時間を約0. 02sec 縮め, 全体として約40%符号化処理時間を高速化できることがわかった. したがって提案手法を用いれば, CPU の時変動に対して処理量を可変にすることでSN劣化を抑えつつ, リアルタイム処理を行う可能性を示すことができた.

4. むすび

本稿では, PC や WS 上で走行するソフトウェアCODEC に適した符号化・復号化アルゴリズムの概念として, 「リアルタイム」という符号化時間拘束のもとで画質の高い復号画像を提供できる動的演算量スケーラブルアルゴリズムを提案した. また, 符号化に要する処理量を可変にすることにより使用可能な演算量を制御する演算量スケーラブルアルゴリズムのH. 263への組み込みを行い, リアルタイム符号化への検討を行った.

今後は, 画質優先/コマ数優先のクオリティトレードオフ, そして CPU 負荷測定機能の組み込みや復号側での演算量スケーラブルアルゴリズムの追加検討を進め, 最終的には CPU 負荷に追従し, 処理量を可変にすることでリアルタイム処理を実現する.

[参考文献]

[1] ITU-T: Recommendation H.261, "Video codec for audio visual services at p×64 kbit/s", 1993.

[2] ISO/IEC 11172-2, "Information Technology - Coding of moving pictures and assosiated audio for digital storage media at up to about 1.5Mbit/s - Part2:Video", 1993.

[3] ISO/IEC 13818-2, "Information Technology - Generic coding of moving pictures and associated audio information - Part2:Video", 1994.

[4] Y. Tashiro, T. Izuoka, K. Yanaka, Y. Ito, N. Ono, Y. Yashima, H. Yamauchi, and H. Kotera, "PC/AT MPEG2 Video and Audio CODEC Board Set", GLOBECOM'95 Conference record, Vol.1, pp. 483-487, Nov. 1995.

[5] C. Fogg, P. Au, S. Eckart, T. Hanamura, K. Oosa, B. Quandt, H. Watanabe, "ISO/IEC Software Implementation of MPEG-1 Video", SPIE Vol. 2187, pp. 249-257, 1994.

[6] 大迫他 "動的演算量スケーラブルアルゴリズムによるソフトウェア符号化", 電子情報通信学会論文誌, Vol. J80-DII, pp. 444 - 458, 1997.

[7] 大迫他 "ソフトウェア符号化のための演算量スケーラブルアルゴリズム", 電子情報通信学会総合大会, D-11-93, 1997.

[8] F.Osako, S.Ishibashi, Y.Yashima, and H.Kotera, "A Dynamic Computation Resource Scalable Algorithm for Software Video CODEC," PCS'97, pp.515-518, Sept. 1997

[9] 大迫他 "ソフトウェア符号化のための演算量可変型アルゴリズム", 電子情報通信学会画像工学研究会, IE-9715, 1997.

[10] ITU-T: Recommendation H.263, "Video coding for low bit rate communication", 1996.