2004－AVM－46　(6)
2004／10／1

H.264/AVCに基づくスケーラブル動画像可逆符号化

高村誠之† 八島由幸†

あらまし　可逆復号が可能な 2 段階 SNR スケーラブル符号化方式を提案する。ベースレイヤは H.264/AVC 方式互換であり、エンハンスレイヤは H.264/AVC の直交変換係数空間での残差を符号化したものである。エンハンス符号化においては、係数空間の格子点で原画素となりえないものを符号化対象から除くことで、量子化により失われる情報を効率よく符号化する。該変換特有の係数間整数値関係を用いることで効率を保ったまま処理を 4 千億倍に高速化し、さらに一部変換係数を別途伝送することでその平方根オーダに高速化する。これらにより実用的な速度での符号化を可能とした。ソフトウェア実装したところフレーム内符号化の場合 Motion JPEG2000 の可逆モードと同程度、フレーム間符号化の場合はその約 1/2～3/4 の総符号量で符号化が可能であった。

Lossless Scalable Video Coding based on H.264/AVC

Seishi Takamura† and Yoshiyuki Yashima†

Abstract:　This paper proposes two-layered SNR scalable video coding scheme having lossless enhancement layer. The base bitstream is fully compliant with H.264/AVC standard, and the enhance bitstream is derived from the residual signal of orthogonal transform coefficients of H.264/AVC. In enhance bitstream coding, grid points that cannot be the original signal are eliminated to optimize the coding efficiency. We also utilize the relationships between the coefficients to accelerate the process 400 billion times while maintaining the coding efficiency exactly the same.　We further propose a method to reduce the execution time down to square-root order of above by transmitting some of the coefficients independently.　Via computer simulation, we confirmed the intra coding efficiency of proposed method was comparable to that of Motion JPEG2000 reversible mode, while inter coding was achieved with only 1/2 - 3/4 bit amount.

1　はじめに

近年ビデオ符号化標準 H.264/AVC 方式 [1] がその圧縮率の高さから注目を集めている。本方式に代表される高能率ビデオ符号化方式は、いずれも動き補償+直交変換+量子化という枠組みを採用しているが、量子化処理の存在のため、復号される信号は原信号とは一致しない。

一旦 H.264/AVC 準拠のビデオ信号を伝送し、必要に応じ追加情報を伝送することで最終的に原信号と同じ信号を再生することができれば、素材伝送やアーカイブ、医療・学術利用等に有用である。

† 日本電信電話株式会社 NTT サイバースペース研究所
〒239-0847 横須賀市光の丘 1-1 Y-517A
NTT Cyberspace Laboratories, NTT Corporation
Phone: 046-859-電話番号, Fax: 046-859-2829
E-mail: {takamura.seishi, yashima.yoshiyuki}@lab.ntt.co.jp

Sun らは H.264/AVC 標準のフレーム内・フレーム間予測残差信号を直交変換・量子化せずに符号化することで可逆性を実現している [2]。これは段階的伝送ができない。

非可逆な符号化方式と組み合わせた段階的符号化方式としては、復号画像と原画像の差分を符号化するもの [3] がある。これはベースレイヤに MPEG-2 を用いているが、他の方式であっても適用は可能である。しかしながら、直交変換空間内での残差ではなく、原信号空間での残差を符号化対象とするため、元来原信号が存在しないはずの空間までも考慮した符号化をせざるをえず、圧縮効率に限界がある。

従来の可逆ビデオ符号化方式として Motion JPEG 2000[4] の可逆モードがあるが、静止画ベースであるためフレーム内に閉じた符号化を行うため効率に限界がある。

また MPEG-4 標準 [5] の Fine Granularity Scalable (FGS) Profile 方式のように、画像信号に離散コサイン変換を施し整数化後ビットプレーン展開し逐次伝送する方式には、

- ◇ 実数変換である DCT 後、係数が整数化されるため、いくら付加情報を用いても可逆にはできない
- ◇ 後述の式 (3) のように変換後の係数が伸張される H.264/AVC 標準方式にそのまま適用すると、伸張分がそのまま符号量の無駄につながってしまう

等の問題があった。

このようにフレーム間予測や直交変換係数領域でのエンハンスを行うことでビデオ符号化効率を高め、スケーラビリティを有し、かつベースレイヤが既存の標準互換である、可逆符号化方式は提案されていなかった。

本稿は、上に述べたような問題に鑑みて、基本部分は H.264/AVC 標準と互換性を保ちながら、付加符号量をできるだけ低く抑えつつ、可逆な復号を可能とする方式を提案する。

2 H.264/AVC 方式の特徴

2.1 予測残差の直交変換

H.264/AVC 標準においては、フレーム内あるいはフレーム間で画素値を予測した後、縦横 4 画素ずつの小ブロック毎に残差の直交変換・係数の量子化を行う。ここでは原信号の小ブロックを 4x4 行列 U で、フレーム内あるいはフレーム間で該ブロックを予測した信号を同じく 4x4 行列 Y であらわす。そして予測残差信号 (4x4 行列 R) を

$$R = U - Y \tag{1}$$

とする。これらはいずれも、要素がすべて整数である。 これに次のような直交変換を施す。

$$X = TRT^T \quad \text{ただし } T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{2}$$

T^T は行列 T の転置行列を表す。ここで T は直交変換ではあるが正規直交変換ではないことに注意する。正規直交変換の行列式は常に 1 であるが、T の行列式は 40 であるので、任意の 4 次元ベクトルのノルムは T による写像後 40 倍になる。式 (2) は R の 4 行および 4 列に変換を施すため、変換後の係数 X は、変換前 (残差信号 R) に比べ

$$40^{4+4} = 6,553,600,000,000 \text{ 倍} \tag{3}$$

と、極めて疎な空間内に写像される。H.264/AVC における直交変換係数は整数となるが、係数空間の格子点は殆どが残差信号として不適である (逆変換しても整数値が得られない) ということになる。仮に係数空間の格子点をすべて対象とし、原信号に対応する格子点を符号化すると、式 (3) の 2 を底とする対数を画素数 ($4 \cdot 4 = 16$) で除した

$$\log_2(40^{4+4})/16 = 2.66 \text{ [bit]}$$

が、1 画素あたり余分に必要になる。対象としている信号が (1 カラーコンポーネントあたり)8bit であることを考えると、2.66bit の増加は致命的である。

2.2 量子化と逆変換

H.264/AVC 方式では、前述のように大きく拡大された X の各要素を、粗く量子化することにより、この拡大分を補償している。このため量子化を施された係数が復号側で逆量子化されると、元の値 X に誤差が重畳し異なる値 $X'(\neq X)$ となり、これを逆変換しても $T^{-1}X'(T^T)^{-1} \neq R$ となる。復号側では Y は符号化側と同一のものを持つことができるが、R が再現できないため原画素値 $U = R + Y$ も再現できない。

3 提案方式

前節の U を完全再現するためには、X の量子化で失われた情報を補うような付加情報を別途伝送する必要がある。

まず

$$R = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}, \; X = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix}$$

とする。例えば X の左上の要素 (A) の値は復号側では量子化前の値は正確にはわからないが、符号化側の量子化方法が復号側でも既知とすると、A がとりう

る範囲は推定できる。H.264/AVC コーデック JM[6] を例にとると

$$|\mathrm{level}_A| = (|A| * \mathrm{qc}_A + \mathrm{qpconst}) >> \mathrm{qbits}$$

のようにして量子化された値 level_A を求めている。なお level_A の正負は A のそれに一致させる。ここで qc_A は A のブロック内の位置 (この場合左上) と量子化パラメータに対応して JM が決める整数、qpconst は符号化モード、量子化パラメータに対応して JM が決める整数、qbits は量子化パラメータに対応して JM が決める整数である。

復号側では量子化パラメータや符号化モードおよび量子化方法を知ることができるので、符号化側と共通の level_A, qpconst, qbits を持つことができる。まず level_A の値から整数値

$$\begin{aligned} x &= \mathrm{level}_A << \mathrm{qbits} \\ y &= x + (1 << \mathrm{qbits}) - 1 \\ \mathrm{mmin} &= (x - \mathrm{qpconst} + \mathrm{qc}_A - 1)/\mathrm{qc}_A \\ \mathrm{mmax} &= (y - \mathrm{qpconst})/\mathrm{qc}_A \end{aligned}$$

を求める。"/" は C 言語の整数除算である。ついで上下限 $A_{\min}$, $A_{\max}$ が以下のように求められる。

$$(A_{\min}, A_{\max}) = \begin{cases} (-\mathrm{mmax}, -\mathrm{mmin}) & (\mathrm{level}_A < 0) \\ (-\mathrm{mmax}, \mathrm{mmax}) & (\mathrm{level}_A = 0) \\ (\mathrm{mmin}, \mathrm{mmax}) & (\mathrm{level}_A > 0) \end{cases}$$

このようにして、量子化前の係数がとりうる上下限値

$$(A_{\min}, A_{\max}),\ (B_{\min}, B_{\max}),\ ...,\ (P_{\min}, P_{\max})$$

を得ることができる。

3.1 基本方式

この上下限を元に、以下のような 16 重ループを実行することで直交変換後の空間において妥当なものを漏れ・無駄なく列挙することができる。

1) index ←cases ←0
2) A の範囲を 1 間隔にループ
3) B の範囲を 1 間隔にループ
   …
4) P の範囲を 1 間隔にループ
5) $R \leftarrow T^{-1}X(T^T)^{-1}$
6) R の要素に一つでも整数でないものがあれば 11 へ
7) $U \leftarrow R + Y$
8) U の要素に一つでも [0〜255] におさまっていないものがあれば 11 へ
9) X が原変換係数に一致していれば index ←cases
10) cases ←cases + 1
11) 16 重ループ継続
12) cases 分の情報量を用いて index を符号化

3.2 変換係数間の整数的値関係の利用

第 3.1 節の方法により原理的に符号化は可能となるが、16 重のループをそれぞれ 1 間隔で回すため、総ループ回数は非常に多くなる。しかしながら、直交変換係数間に存在する整数的関係を用いることで、符号化効率は同じままで、ループ回数を大幅に削減することができる。

まず 4x4 行列 R を、上の行から下の行へ順に並べなおした 16 次元の列ベクトル $\vec{x}$ により等価表現する。

$$\vec{x} = [a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p]^T$$

式 (2) を $\vec{x}$ を用いて書き換えると、

$$A = \vec{t}_A\vec{x},\ B = \vec{t}_B\vec{x},\ \cdots,\ P = \vec{t}_P\vec{x}$$

のようになる。ここで

$\vec{t}_A$=[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
$\vec{t}_B$=[2, 1, −1, −2, 2, 1, −1, −2, 2, 1, −1, −2, 2, 1, −1, −2]
$\vec{t}_C$=[1, −1, −1, 1, 1, −1, −1, 1, 1, −1, −1, 1, 1, −1, −1, 1]
$\vec{t}_D$=[1, −2, 2, −1, 1, −2, 2, −1, 1, −2, 2, −1, 1, −2, 2, −1]
$\vec{t}_E$=[2, 2, 2, 2, 1, 1, 1, 1, −1, −1, −1, −1, −2, −2, −2, −2]
$\vec{t}_F$=[4, 2, −2, −4, 2, 1, −1, −2, −2, −1, 1, 2, −4, −2, 2, 4]
$\vec{t}_G$=[2, −2, −2, 2, 1, −1, −1, 1, −1, 1, 1, −1, −2, 2, 2, −2]
$\vec{t}_H$=[2, −4, 4, −2, 1, −2, 2, −1, −1, 2, −2, 1, −2, 4, −4, 2]
$\vec{t}_I$=[1, 1, 1, 1, −1, −1, −1, −1, −1, −1, −1, −1, 1, 1, 1, 1]
$\vec{t}_J$=[2, 1, −1, −2, −2, −1, 1, 2, −2, −1, 1, 2, 2, 1, −1, −2]
$\vec{t}_K$=[1, −1, −1, 1, −1, 1, 1, −1, −1, 1, 1, −1, 1, −1, −1, 1]
$\vec{t}_L$=[1, −2, 2, −1, −1, 2, −2, 1, −1, 2, −2, 1, 1, −2, 2, −1]
$\vec{t}_M$=[1, 1, 1, 1, −2, −2, −2, −2, 2, 2, 2, 2, −1, −1, −1, −1]
$\vec{t}_N$=[2, 1, −1, −2, −4, −2, 2, 4, 4, 2, −2, −4, −2, −1, 1, 2]
$\vec{t}_O$=[1, −1, −1, 1, −2, 2, 2, −2, 2, −2, −2, 2, −1, 1, 1, −1]
$\vec{t}_P$=[1, −2, 2, −1, −2, 4, −4, 2, 2, −4, 4, −2, −1, 2, −2, 1]

である。

ここで $\vec{t}_A + \vec{t}_C$ を計算すると

$$\begin{aligned} \vec{t}_A + \vec{t}_C &= [2, 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, 2] \\ &= 2[1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1] \end{aligned}$$

したがって

$$A + C = (\vec{t}_A + \vec{t}_C)\vec{x} = 2[1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1]\vec{x}$$

となるので、任意の整数要素ベクトル $\vec{x}$ に対し $A+C$ は必ず偶数となる。

つまり A の値がわかれば、C の下位第 1 ビットは A のそれと同じとわかるので、C は、新しい下限

$$C'_{\min} = C_{\min} + ((C_{\min} + A)\ \&\ 1) \qquad (4)$$

から 2 おきに、$C_{\max}$ を超えない範囲に存在していることになる。同じ位置関係である、E と G, I と K, M と O なども同様である。

同様に縦方向においても、$A + I$ も偶数であることから、A の値がわかっている場合 I が間隔 2 で存在する範囲がわかる。同じ位置関係である、B と J, C と K, D と L も同様である。

次に、任意の整数要素ベクトル $\vec{x}$ に対し、$B+(C >> 1)+(A >> 1)$ は常に偶数になる。これは、$\vec{x}$ の各要素を 0,1,2,3 と変化させたすべての場合について確認することで証明される。つまり A と C の値がわかっている場合、B の下位第 1 ビットは $(C >> 1)+(A >> 1)$ のそれと同じとわかるので、上記 C の場合と同様に B の存在範囲がわかり、そこに間隔 2 で存在することになる。同じ位置関係、および縦に同じ位置関係である、

$$\begin{aligned} F &\quad と \quad (E >> 1) + (G >> 1) \\ E &\quad と \quad (A >> 1) + (I >> 1) \\ G &\quad と \quad (C >> 1) + (K >> 1) \end{aligned}$$

なども同様である。

次に

$$2.5(\vec{t}_A + \vec{t}_C) + 2\vec{t}_B + \vec{t}_D = 10[1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0]$$

となるので、A, B, C の値がわかっている場合、D の存在範囲がわかり、その中に間隔 10 で存在することになる。同じ位置関係や、縦に同じ位置関係、たとえば M と A, E, I の間にも同様の関係がある。

さらに以下のような関係がある:

$$\vec{t}_A + \vec{t}_C + \vec{t}_I + \vec{t}_K = 4[1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1]$$

$$\vec{t}_B - \vec{t}_J + \vec{t}_E - \vec{t}_G = 4[0,1,1,0,1,1,0,-1,1,0,-1,-1,0,-1,-1,0]$$

$$2.5(\vec{t}_A + \vec{t}_C + \vec{t}_I + \vec{t}_K) + 2(\vec{t}_E + \vec{t}_G) + \vec{t}_M + \vec{t}_O = 20[1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]$$

$$2.5(\vec{t}_A + \vec{t}_C + \vec{t}_I + \vec{t}_K) + 2(\vec{t}_B + \vec{t}_J) + \vec{t}_D + \vec{t}_L = 20[1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]$$

$$6.25(\vec{t}_A + \vec{t}_C + \vec{t}_I + \vec{t}_K) + 5(\vec{t}_B + \vec{t}_E + \vec{t}_G + \vec{t}_J) + 2.5(\vec{t}_D + \vec{t}_L + \vec{t}_M + \vec{t}_O) + 4\vec{t}_F + 2(\vec{t}_H + \vec{t}_N) + \vec{t}_P = 100[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$$

これらを用い、第 3.1 節での多重ループ部分を以下のように等価に書き直せる:

1) A の範囲を 1 間隔にループ
2) C の範囲を 2 間隔にループ (A を利用)
3) B の範囲を 2 間隔にループ (A, C を利用)
4) D の範囲を 10 間隔にループ (A, B, C を利用)
5) I の範囲を 2 間隔にループ (A を利用)
6) E の範囲を 2 間隔にループ (A, I を利用)
7) M の範囲を 10 間隔にループ (A, I, E を利用)
8) K の範囲を 4 間隔にループ (A, C, I を利用)
9) G の範囲を 2 間隔にループ (E を利用)
10) F の範囲を 2 間隔にループ (E, G を利用)
11) H の範囲を 10 間隔にループ (E, F, G を利用)
12) J の範囲を 4 間隔にループ B, E, G を利用)
13) L の範囲を 20 間隔にループ (A, B, C, D, I, J, K を利用)
14) N の範囲を 10 間隔にループ (B, F, J を利用)
15) O の範囲を 20 間隔にループ (A, C, E, G, I, K, M を利用)
16) P の範囲を 100 間隔にループ (A〜O を利用)

こうして第 3.1 節の方法に比べ、総ループ時間を

$$2 \cdot 2 \cdot 10 \cdot 2 \cdot 2 \cdot 10 \cdot 4 \cdot 2 \cdot 2 \cdot 10 \cdot 4 \cdot 20 \cdot 10 \cdot 20 \cdot 100 = 409,600,000,000$$ 倍

に高速化できる。

3.3 ループ多重度削減

第 3.2 節節の方法をさらに高速化するため、本方式では以下のようにしてループの多重度を減らしている。

まず係数 A の伝送において、

$$Z_A = A - A_{\min}$$

を伝送すれば、復号側で $A = Z_A + A_{\min}$ とすることで A を復元できる。なお、このとき A のとりうる場合の数が $A_{\max} - A_{\min} + 1$ であるので、Z_A を符号化するのに必要な情報量は、$\log_2 (A_{\max} - A_{\min} + 1)$ [bit] となる。これは復号側でも共有できるので、Z_A を復号することは可能である。

B〜P についてもこの A のように伝送することも可能であるが、残差信号は直交変換後の空間内で非常に疎に分布していることから、符号量が無駄になる。そこで例えば C については、A が既知の場合 C に関するループを 2 間隔にできたことから式 (4) を用いて

$$Z_C = (C - C'_{\min})/2$$

を伝送すれば、復号側で $C = 2Z_C + C'_{\min}$ により C を復元できる。Z_C を符号化するのに必要な情報量は、$\log_2\left((C_{\max} - C'_{\min})/2 + 1\right)$ [bit] となる。B〜K についても同様である。したがって、

$$Z_A, Z_C, Z_B, Z_D, Z_I, Z_E, Z_M, Z_K$$

をこの順で符号化することで、対応する 8 係数が無駄なく伝送される。残る 8 係数

$$F, G, H, J, L, N, O, P$$

については、第 3.2 節と同様、まとめて 1 つの数 index で表現し伝送する。なお index を記述するのに必要な情報量は $\log_2$ cases [bit] である。これは復号側が付加情報なしに持つことができ、かつ index の復号に必要な情報である。

こうして、16 重だったループが 8 重に削減され、処理がさらに高速 (元の平方根オーダ) になる。

4 議論、計算機シミュレーション

4.1 符号量の性質予想

ここでベース・エンハンス双方の符号量を定性的に予想する。H.264/AVC 方式では QP と量子化ステップ Q の間には

$$Q \propto 2^{QP/6}$$

の関係がある。またベース符号量 R_b は MPEG-2 と同様の特性を持つと仮定し

$$R_b \propto 1/Q \propto 2^{-QP}$$

と見積もる [7]。エンハンス符号量 R_e は、$4 \cdot 4 = 16$ 要素の量子化誤差が作る存在空間内の格子点の個数 (≃ 体積) の情報量 (bit) に比例するので、

$$R_e \propto \log_2(Q^{16}) \propto \log Q \propto QP$$

となる。したがって総符号量 R_{tot} は

$$R_{tot} = R_B + R_e = k_b 2^{-QP} + k_e QP$$

となる。ここで k_b, k_e は適当な比例定数である。$R_b(QP)$ は単調減少で下に凸、$R_e(QP)$ は単調増加で直線であるので、R_{tot} は下に凸となる。k_b, k_e の値にも依存するが、ある QP で R_{tot} は最小値をとることが予想される。

4.2 符号化実験

H.264/AVC コーデック JM Ver 8.3[6] に本アルゴリズムを実装した。ベース情報は H.264/AVC 完全互換であるので、参照画像は常に lossy である。

符号化方式は III.., IPP..., IBP…BP, IBBP…BBP の 4 通りで、それぞれ III, IPP, IBP, IBBP と表記する。符号化枚数は上記すべての符号化に共通な 295 枚とした。4 種の 4:2:0 QCIF 画像を符号化対象とした。その他符号化条件は以下のようである:

- ◇ エントロピ符号化 = CABAC
- ◇ 参照フレーム数 = 5
- ◇ 探索範囲 = 16
- ◇ ME 時 Hadamard 変換 使用

比較に Motion JPEG 2000 コーデック Kakadu [8] の kdu_v_compress を用いた。符号化オプションは Creversible=yes および (QCIF で最も圧縮率が高かった) Clevels=2、Motion JPEG2000 ファイルフォーマットである。

結果を表 1 に示す。本方式の符号量は H.264/AVC 方式互換のベース符号量と、エンハンス情報のエントロピの和である。III は常に Motion JPEG2000 よりわずかに劣る結果となった。またフレーム内符号化 (III) とフレーム間符号化 (それ以外) を比較すると、Stefan 以外は約 50% →約 25%と、ほぼ 2 倍に圧縮率が向上した。Stefan は動きが激しいためフレーム間予測が奏功せず約 60%→約 45%の改善にとどまった。IBP, IBBP では IPP に比べ圧縮率が落ちているケースがあるが、探索範囲や量子化パラメータが最適化されてはいないことも一因と考えられる。

図 1 に、Silent Voice (III) におけるベース、エンハンス、総符号量のグラフを示す。第 4.1 節の予想が成り立っていることがわかる。

また本実装では総符号量最小化を規準としたため、ベースレイヤのみで考えると RD 最適化されたものよりも性能が劣っている可能性がある。III におけるこの比較を図 2 に示す。同一 PSNR で約 4%の符号量増加が観察される。ほかの画像やフレーム間符号化

表 1: Motion JPEG2000, 本方式の可逆圧縮率比較 (原画に対する圧縮後サイズ%) 括弧内は I(,P(,B)) の *QP* 値

	JPEG2000	III	IPP	IBP	IBBP
Container Ship	47.3	47.4(9)	26.7(11,6)	26.2(11,6,7)	26.7(11,6,7)
Silent Voice	50.7	51.8(10)	26.8(11,6)	26.8(11,6,7)	27.1(11,6,7)
News	46.9	47.0(9)	24.1(11,5)	24.6(11,6,6)	25.1(11,6,6)
Stefan	59.8	60.9(12)	44.6(11,9)	45.2(11,6,9)	45.8(11,6,9)

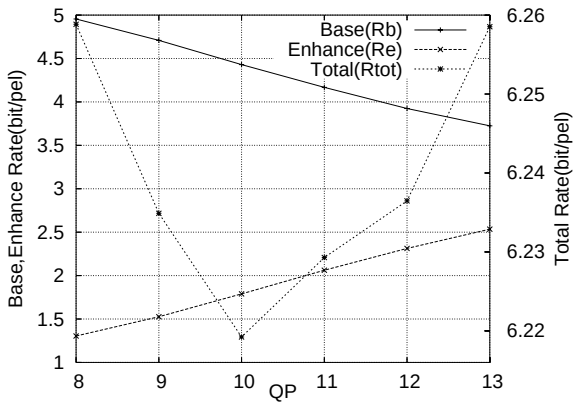

図 1: Silent Voice (III) におけるベース、エンハンス、総符号量 (総符号量のみ右側のスケール)

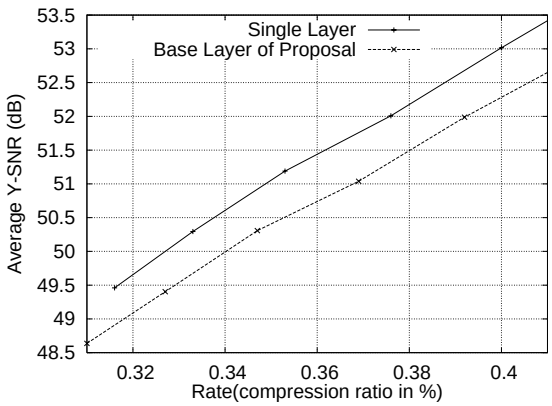

図 2: Silent Voice (III) における単一レイヤ符号化と本方式ベースレイヤそれぞれの符号量-歪み特性

においてもほぼ同様の結果であった。

5 おわりに

H.264/AVC 互換のベースレイヤを持つ 2 段階ロスレス SNR スケーラビリティ符号化方式を提案した。

符号化実験により, フレーム内符号化の効率は Motion JPEG2000 の可逆モードと同程度、フレーム間符号化の効率はその約 1/2〜3/4 の総符号量であることがわかった。

今後は、効率改善に加え、ベースレイヤの RD 性能の劣化を抑えつつ総符号量の増加も抑える方式、より粒度の細かい方式などを検討していく。

謝辞 アルゴリズムの実装に多大なご協力を賜りました NTT ソフトウェアの米原紀子氏に感謝します。

参考文献

[1] ISO/IEC 14496-10:2003 Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding

[2] S. Sun: “Lossless Coding and QP Range Selection,” JVT of ISO/IEC MPEG & ITU-T VCEG, JVT-C023, May, 2002.

[3] 中嶋, 八島, 小林: “MPEG-2 符号化パラメータに基づく階層的ロスレス符号化の検討”, 信学総大 D-11-49, Mar. 2000.

[4] ISO/IEC 15444-3:2002 Information technology – JPEG 2000 image coding system – Part 3: Motion JPEG 2000

[5] ISO/IEC 14496-2:2003 Information technology – Coding of audio-visual objects – Part 2: Visual

[6] http://bs.hhi.de/˜suehring/tml/, “JM Reference Software version 8.4”, Jul 2004.

[7] S. Takamura, H. Watanabe and N. Kobayashi: “One-pass VBR algorithm for an MPEG-2 encoder chip SuperENC,” Picture Coding Symposium Japan, pp. 39–40, Sep. 1999.

[8] http://www.kakadusoftware.com, Linux Executable Version 4.2, Feb. 2004.