レート歪み最適化による量子化プロセスの高速化とその定量的評価

森口 元気 †　　神戸 尚志 ‡　　藤田 玄 ‡‡

† 近畿大学大学院 総合理工学研究科
‡ 近畿大学 理工学部 電気電子工学科
‡‡ 大阪電気通信大学 情報通信工学部

概要

H.264/AVC におけるレート歪み最適化による量子化（RDOQ）は複数の量子化値を候補とし，最適な量子化値をレート歪み（RD）を基に決定し, 符号化性能を向上する有効な技術である．本研究では，選択される可能性が低い量子化値の候補の除外，ビットレート推定法の変更といった複雑性を低減する改良手法を提案し，その定量的評価を行う．

An Improvement of Rate-Distortion Optimized Quantization and its quantitative evaluation

Genki Moriguchi†　　Takashi Kambe‡　　Gen Fujita‡‡

†Graduate School of Science and Engineering Research, Kinki University
‡Department of E & E, Faculty of Science and Engineering, Kinki University
‡‡Faculty of Information and Communication Engineering, Osaka Electro-Communication University

Abstract

Rate-distortion optimized quantization (RDOQ) in H.264/AVC is a important technology to improve its video coding performance. It is able to be determined the optimal value among multiple quantization candidates based on rate-distortion (RD). We propose an algorithm improvement to reduce its complexity by changing the bit rate estimation method and by excluding low scored candidates for the quantization,and their performances are evaluated.

1　はじめに

H.261 や MPEG-1 から始まる動画像圧縮技術の進歩により，H.264/AVC(以下 H.264)[1] は，高精度テレビで考えると 1/80 倍まで圧縮することが可能である．しかし，圧縮率を高めるほどアルゴリズムが複雑になり，処理に時間がかかることが問題である．本研究では，H.264 の処理の 1 つである RDOQ に対し，ビットレート推定処理の複雑性の低減手法を適用し，ハードウェア実装に有利なアルゴリズムを提案し，いくつかのデータを用いて定量的な評価を行う．

2　レート歪み最適化による量子化（RDOQ）

H.264 の参照ソフトウェア JM18.0 [2] における量子化にはレート歪み最適化手法 (以下 RDOQ)[3][4] が導入されている．RDOQ は複数の丸め候補（切り上げ丸め値 (天井関数, ceil)，切り下げ丸め値 (床関数,floor) および 0 ）のレート歪みコスト (以下 RD コスト) を比較することにより最適な量子化レベルを決定する．DCT 係数 $c_i(i = 0, .., M)$ が量子化レベル $l_i^j(j = 0, floor, ceil)$ に量子化される時，各量子化候補の RD コスト J は式 (1) によって導出され，RD コスト J が最小となった候補を最適な量子化レベルに決定する．

$$J_i^j = err_i^j + \lambda \times bits_i^j \quad (1)$$

ここで，$bits_i^j$ は量子化レベル ^j_i にエントロピー符号化を行うことで得られるビットレートを表し，err_i^j は量子化誤差 (歪) を示す．また，λ は量子化パラメータ (QP) 等によって決定される定数である．

JM18.0 における RDOQ では，CABAC による実際の符号化は行わず，ルックアップテーブルから推定ビットレートを取得する形をとっている．RDOQ は従来の量子化に比べ，より高い符号化効率を可能としているが，量子化係数毎に最適量子化値の決定，コンテキストの更新という複雑なプロセスを行うため，計算量が多く，複雑性が高い．また，ビットレート取得における係数間の依存性はハードウェア設計における並列処理の実装において不利である..

3 RDOQ の高速化アルゴリズム

本章では，RDOQ のアルゴリズム高速化手法を提案する．文献 [5] では「量子化レベル最適化候補の削減」「ビットレート推定の簡略化」が提案されている．提案手法では，さらに「ビットレート推定の簡略化」を改良し，また新たに「歪計算の高速化」「最後の非ゼロ係数探索範囲の削減」「全係数 0 化判定のスキップ」を提案する．

3.1 量子化レベル最適化候補の削減

量子化レベル l_i^{float}(丸め前の値) が 1.5 以上の時，量子化レベル $l_{i,0}$ が選択される可能性は極めて低いため，量子化レベル l_i^{float} が 1.5 以上だった場合は式 (2) を用いて l_i^{floor}，l_i^{ceil} の 2 つの比較のみで最適化を行う．このように，量子化レベル l_i^{float} の値によって最適化候補を 3 つから 2 つ以下に絞ることで差分値を用いた比較が可能となり，計算回数が削減できる．ビットレートの計算は RDOQ とは別の処理で行っているため，削減する計算は歪計算および RD コスト計算となる．表 1 に JM18.0 において量子化レベル l_i^{float} が 1.5 以上の時，量子化レベル $l_{i,0}$ に最適化される確率を示す．また，表 2 に JM18.0 において本手法を幾つかのサンプル動画に適用した際の歪計算回数の削減率を示す．

$$\Delta J_i = J_i^{floor} - J_i^{ceil} = \Delta err_i + \lambda \times \Delta bits_i \quad (2)$$

$\Delta\ err_i$ および $\Delta\ bits_i$ はそれぞれ量子化歪とビットレートの差を表しており，$\Delta\ J_i < 0$ の時，l_i^{floor}，$J_i \geq 0$ の時，l_i^{ceil} を最適な量子化値と決定する．

表 1: 量子化レベル l_i^{float} が 1.5 以上の時，量子化候補 0 に最適化される確率

テストデータ	解像度	確率 [%]
news	CIF	0.0013
container	CIF	0.0022
football	CIF	0.0407
paris	CIF	0.0371
PartyScene	832x480	0.2236
In to tree	720p	0.0114

表 2: 歪計算回数の削減率

テストデータ	JM18.0	提案手法	削減率 [%]
news	12,021,409	10,404,051	13.5
container	12,653,105	10,681,845	15.6
football	14,320,665	11,417,882	20.3
paris	14,178,314	10,734,385	24.3
PartyScene	61,147,492	44,814,814	26.7
In to tree	112,233,107	81,594,873	27.3

3.2 歪計算の高速化

本節では RD 計算における歪 err_i の計算方法の改良について述べる．文献 [5] より l_i^{floor}，l_i^{ceil} の量子化歪みの差 $\Delta\ err_i$ は式 (3) によって求める．

$$\begin{aligned}\Delta err_i &= err_i^{floor} - err_i^{ceil} \\ &= (l_i^{floor} - l_i^{float})^2 - (l_i^{floor} + 1 - l_i^{float})^2 \times Qstep^2 \\ &= (2 \times \Delta l_i - 1) \times Qstep^2\end{aligned} \quad (3)$$

$\Delta\ l_i$ は l_i^{floor} と丸め前の量子化レベル l_i^{float} との差を，Qstep は量子化ステップを表す．

しかし，式 (3) において，$\Delta\ l_i$ は l_i^{floor} に丸める際に切り捨てられた値に相当し，小数点演算が必要となる．量子化処理は，正規化処理によってスケーリングされた DCT 係数 $|c_i|$ を Qstep で除算して行うが，提案手法では，量子化をビットシフトにより実現する．

除算した際に切り捨てられた値が量子化値の小数点部分になるので，切り捨てられる範囲をビット抽出すると $\Delta\ l_i$ を逆量子化した値が得られる．よって，$\Delta\ err_i$ は式 (4) のように考える．

$$\begin{aligned}\Delta err_i &= [(2 \times |c_i| \wedge (Qstep - 1) - Qstep] \times Qstep \\ &= [\{(|c_i| \wedge (Qstep - 1)) << 1\} - Qstep] << Qbits\end{aligned} \quad (4)$$

Qbits は Qstep での除算をビットシフトで表したものである．これにより，小数点乗算を省き，シフト演算のみで $\Delta\ err_i$ の演算を実現し高速化を図る．

表 3: 丸め前の値によるレベル分け

noLevels	条件 1	条件 2
3	1 以上かつ小数部が 0.5 以上	1.5 以上
2	1 以上かつ小数部が 0.5 未満， または 1 未満かつ小数部が 0.5 以上	0.5 以上 1.5 未満
1	0.5 未満	0.5 未満

表 4: JM18.0 および提案手法の差分歪計算適用比率

テストデータ	JM18.0[%]	提案手法 1[%]	提案手法 2[%]
news	1.9	11.9	13.6
container	2.4	14.9	17.0
football	3.7	21.9	24.9
paris	3.3	25.4	28.7
PartyScene	4.3	31.9	35.8
In to tree	1.6	6.7	7.8

3.3 最後の非ゼロ係数の最適化の探索範囲の削減

RDOQ では 1 係数毎の最適化を行う前に,「最後の非ゼロ係数の最適化」を行う．最後の非ゼロ係数の最適化とは，RD コストにより高周波成分の切り捨てを符号化性能に影響がない範囲で行う処理である．量子化の際に量子化レベル $l_{i,float}$ に応じてレベル (noLevels) 分けを行う (表 3 中の条件 1)．最後の非ゼロ係数最適化はジグザグスキャンの順序で最後に出てくる $noLevels = 3$ からそれ以降の $noLevels = 2$ の間を探索範囲として探索する．ここで，0 と l_i^{floor} のそれぞれの量子化歪み $err(c_i, 0)$, $err(c_i, floor)$ が必要となる．しかし，最後に出てくる $noLevels = 3$ より係数番号が小さい係数は探索範囲外であるため，歪み $err(c_i, 0)$，$err(c_i, floor)$ の両方を個別に計算しておく必要はなく，式 (4) を適用することができる．

提案手法 1 として，ジグザグスキャンを逆順にし，最初は各量子化レベルの候補の歪をそれぞれ求め，$noLevels = 3$ の量子化レベルが出現したら，それ以降の歪計算に式 (4) を適用することで符号化性能を維持しつつ高速化を図る．

次に探索範囲の削減について述べる．JM18.0 では表 3 中の条件 1 における $noLevels = 2$ の量子化レベルを切り捨てるか否かの探索を行っている．しかし，量子化レベル l_i^{float} が 1.5 以上の時，量子化候補 0 に最適化される確率は低い (表 1)．

提案手法 2 として noLevels のレベル分けを表 3 中の条件 2 に変更し，探索範囲を削減する．表 4 に JM18.0，提案手法 1 および提案手法 2 における式 (4) の適用比率を示す．

係数番号i	15	14	13	12	11	10	9	8	7	···
noLevels	1	2	2	2	1	3	2	2	1	···

探索範囲

図 1: 最後の非ゼロ係数の最適化探索範囲

3.4 ビットレート推定手法

本節では，CABAC(コンテキスト適応型 2 値算術符号化) の発生ビットレートを高速かつ効率的に推定する手法を提案する．

3.4.1 推定方法

量子化レベルのビットレートは，主に 4 つの要素，すなわち，significant_coeff_flag, last_significant_coeff_flag，coeff_greater_one_flag，coeff_abs_level_minus2 に関連している．ここで，coeff_greater_one_flag，coeff_abs_level_minus2 とは，coeff_abs_level_minus1 を 2 つに分割して考えた要素である．coeff_greater_one_flag は，量子化レベル l_i の絶対値 $|l_i|$ が 1 か否かを示すフラグである．また，coeff_abs_level_minus2 は，量子化レベル $|l_i|$ から 2 を引いた値であり，符号化はその値の数だけ symbol=1 を符号化し，最後に symbol=0 を 1 つ符号化する．

一般に，CABAC による符号化で発生するビットレートは，コンテキストモデルの確率分布のエントロピーが出力される．したがって，あるビットレート B は，確率分布 P より式 (5) を用いて推定できる．

$$B = -log_2 P \tag{5}$$

significant_coeff_flag，last_significant_coeff_flag，coeff_greater_one_flag，coeff_abs_level_minus2 の確率分布をそれぞれ P_{sc}, P_{ls}, P_{cg}, P_{ca} と表し，これらの確率から得られるビットレートを B_{sc}, B_{ls}, B_{cg}, B_{ca} とする．このとき，量子化レベル l_i のビットレートの合計 Bits は式 (6) で求める．

$$\begin{aligned}\Delta Bits_i = {} & B_{sc} + B_{ls} + B_{cg} \\ & + (l_i - 2) \times B_{ca}^{symbol=1} + B_{ca}^{symbol=0}\end{aligned} \tag{6}$$

文献 [5] の手法では，式 (6) を $B_{ca}^{symbol=0}$, $B_{ca}^{symbol=1}$ のように B_{ca} を分けておらず，coeff_abs_level_minus2 の推定ビットレートを $(l_i - 2) \times B_{ca}$ としている．しかし，coeff_abs_level_minus2 が symbol $= 1$ となる確率のみ求めているので，量子化レベル $l_i = 2$ のとき

表 5: シンボル一覧

N_l	最後の非ゼロ係数の前のゼロの累積数
N_o	絶対レベル 1 の累積数
N_g	絶対レベル 2 以上の累積数
N_b	1 以上の係数があったブロックの累積数
S_g	1 よりも大きい絶対レベル値の総和

coeff_abs_level_minus2 の推定ビットレートが 0 になってしまう問題がある．

提案手法では，推定ビットレート B_{ca} を symbol=1 の推定ビットレートを $B_{ca}^{symbol=1}$，量子化レベル l_i が 2 以上の時，1 ビットだけ符号化する symbol = 0 を推定ビットレート $B_{ca}^{symbol=0}$ とする．量子化レベル l_i が 2 の場合でも，$B_{ca}^{symbol=0}$ があるため，coeff_abs_level_minus2 の推定ビットレートが 0 になる問題を回避する．各シンタックス要素の確率分布 P_{sc}, P_{ls}, P_{cg}, P_{ca} の計算式を式 (7)〜(10) に示す．これらの確率は，現在のスライスで既に符号化されたブロックの量子化レベル $l_i(i = 0, ..., M)$ の出現パターンの累積数に従って得られる．式 (7)〜(10) に使用するシンボルのリストを表 5 に示す．

$$P_{sc} = \begin{cases} \dfrac{N_l}{N_l + N_o + N_g} & significant\_coeff\_flag = 0 \\ 1 - \dfrac{N_l}{N_l + N_o + N_g} & significant\_coeff\_flag = 1 \end{cases} \quad (7)$$

$$P_{ls} = \begin{cases} 1 - \dfrac{N_b}{N_o + N_g} & last\_significant\_coeff\_flag = 0 \\ \dfrac{N_b}{N_o + N_g} & last\_significant\_coeff\_flag = 1 \end{cases} \quad (8)$$

$$P_{cg} = \begin{cases} \dfrac{N_g}{N_o + N_g} & coeff\_greater\_one\_flag = 0 \\ 1 - \dfrac{N_g}{N_o + N_g} & coeff\_greater\_one\_flag = 1 \end{cases} \quad (9)$$

$$P_{ca} = \begin{cases} \dfrac{N_g + 1}{S_g + 1} & coeff\_abs\_level\_minus2 = 0 \\ 1 - \dfrac{N_g + 1}{S_g + 1} & coeff\_abs\_level\_minus2 = 1 \end{cases} \quad (10)$$

significant_coeff_flag の確率分布 (式 (7))[5] は，は最後の非ゼロ係数までの全係数の累計数の内，ゼロの割合を求めている．よって，significant_coeff_flag = 0 が存在する確率となる．last_significant_coeff_flag と coeff_greater_one_flag は significant_coeff_flag = 1 の場合のみ算術符号化を行う．式 (8) と式 (9) はそれぞれのシンタックス要素の累積数を $N_o + N_g$ で除算することでそれぞれの確率を求める．

last_significant_coeff_flag は 1 ブロックの非ゼロ係数の内 1 つの係数のみ flag=1 となるため，確率計算を複雑にしてまで推定精度を維持する必要はない．したがって，last_significant_coeff_flag の確率分布式 (8) は文献 [5] の式より，単純な確率計算を採用する．

coeff_abs_level_minus2 は，前述したように量子化レベルから 2 を引いた値と同じ数だけ symbol=1 を符号化し，最後に 1 ビットだけ symbol=0 を符号化するため，量子化レベル−2 の大きさ分 symbol=1 が出現し，coeff_greater_one_flag=1 の累積数分 symbol=0 が出現する．よって coeff_abs_level_minus2 の確率分布は式 (10) となる．coeff_abs_level_minus2 は JM18.0 では，量子化レベル l_i が 27 未満の場合はルックアップテーブルから推定ビットレートを取得し，27 以上の場合は指数ゴロム符号を基に推定する．指数ゴロム符号は，量子化レベル l_i の値が大きければ大きいほど推定ビットレートの計算が長くなってしまう．本研究では量子化レベルの大きさに関わらず，1 シンボル分の推定ビットレートを 1 回取得するのみであるため，JM18.0 に比べビット推定処理の複雑性を低減できる．

JM18.0 の RDOQ では，significant_coeff_flag と last_significant_coeff_flag では係数毎に位置情報に応じて異なった推定ビットレートを取得している．また，coeff_greater_one_flag および coeff_abs_level_minus2 では現在の係数の前に coeff_greater_one_flag が 0 または 1 が連続しているか，という情報を基に異なる推定ビットレートを取得しているため，合計ビット計算の処理で係数間に依存関係が存在している．しかし，提案手法ではブロック内の全係数で式 (5)，式 (7)〜式 (10) によって求まった推定ビットレートを用いる．これにより位置情報毎に推定ビットレートを取得する必要がなく，処理の初めに 1 回だけ取得しておくだけでよいため，推定ビットレートを格納しているレジスタへのアクセス回数が 1 ブロックにつき 1 回で良い．さらに，保持するデータを大きく削減することでハードウェアにおいて回路面積の削減になる．また，係数間の依存関係も無いため，ハードウェアにおいてブロック内の全係数の並列処理を可能にする．表 6 に significant_coeff_flag と last_significant_coeff_flag のデータアクセス削減回数を示す．

推定するビットレートの要素は上記 4 種のシンタックス要素の他に CBP（Coded Block Pattern）がある．CBP はブロック内の非ゼロ係数の有無を示すフラグであるが，推定するために隣接ブロックの情報が必要となり，複数のデータアクセスが必要となる．しかし，CBP は発生ビットレートが小さく，最適化における影響力は少ない．したがって，提案手法では CBP のビットレート推定は行わないことで複雑性を低減する．

表 6: 提案手法による推定ビットレートのデータアクセス削減

significant_coeff_flag			
テストデータ	JM18.0	提案手法	削減率 [%]
news	4,208,705	276,031	93.4
container	5,825,528	322,797	94.5
football	8,348,793	398,406	95.2
paris	7,226,950	364,320	95.0
PartyScene	38,608,598	1,580,599	95.9
In to tree	62,593,826	3,147,906	95.0
last_significant_coeff_flag			
テストデータ	JM18.0	提案手法	削減率 [%]
news	3,502,458	276,031	92.1
container	4,628,884	322,797	93.0
football	6,915,035	398,406	94.2
paris	6,189,006	364,320	94.1
PartyScene	32,242,538	1,580,599	95.1
In to tree	41,993,888	3,147,906	92.5

3.4.2 推定ビットレートのスケーリング λ

RDO では，原画像と復号画像との 2 乗誤差和である符号化歪と発生ビットレートを用いて RD コスト計算を行うが，RDOQ では，量子化歪と推定ビットレートを用いて RD コスト計算を行う．推定ビットレートは発生ビットレートを推定した値なのでスケールに大きな差はない．しかし，RDOQ の量子化歪は符号化歪に比べ，スケールが大きい．このため，量子化歪のスケールに合わせるために推定ビットレートに λ を乗算してスケーリングを行う．しかし，提案する推定手法では JM18.0 のものより簡略化し高速化しており，推定精度は下がる危険がある．

そこで，提案手法では，推定精度を考慮してスケーリング値 λ を設定することで，符号化性能の維持を図る．

3.4.3 全係数 0 化判定のスキップ

RDOQ では，処理ブロック内の全量子化係数を RD コストにより最適化した後，ブロック内係数の RD コスト合計と全て 0 とした場合の RD コスト合計を比較し，全係数を 0 に最適化するか否かを判定する．判定式を式 (11) に示す．但し，式 (11) 中の $bits_{CBP}$ は提案手法では推定しないため，0 として計算する．式 (11) に示す通り，全係数 0 化判定には全係数の歪 (0) の合計を使用するため，差分歪計算に加えて 0 の歪計算も行う必要がある．

提案手法では，ブロック内の係数に量子化レベル l_i^{float} が 1.5 以上が存在する場合，判定処理のスキップを行っても符号化性能に影響がほとんどないことを利用し，差分歪計算の適用を可能とする．表 7 に全係数 0 化判定をスキップした場合の符号化性能への影響を示す．

表 7: 全係数 0 化判定スキップによる符号化性能への影響

テストデータ	BD-BR[%]
news	-0.063
container	0.073
football	-0.109
paris	0.072
PartyScene	-0.013
In to tree	-0.090
平均	-0.022

$$\sum_{i=0}^{M} err_{i,0} + \lambda \times bits_{CBP} < \sum_{i=0}^{M} J_{i,best} \qquad (11)$$

4 高速化アルゴリズムの性能評価

本節では，提案手法である高速化アルゴリズムの性能評価とその考察を示す．

4.1 推定ビットレートの精度評価

本項では推定ビットレートの精度について述べる．

提案手法では，CABAC が同じシンタックス要素でもストリームの性質に応じて別々に確率情報を持つところを 1 つにまとめている．これにより係数の依存関係をなくし高速化している．推定精度はやや下がるが，提案手法では推定ビットレートより量子化歪の比重を大きくし，RD コストを計算することで精度の低下をカバーしている．

4.2 符号化性能評価

符号化性能の測定環境を表 8 に示す．提案手法と JM18 の RDOQ を，RD 曲線を比較することによって，BD-Bitrate [6] について考察する．表 9 に JM18.0 の RDOQ と比べた提案手法の性能を示す．また，表 9 より JM18.0 の RDOQ に対する提案手法の BD-Bitrate は平均 0.02 %，BD-PSNR は平均-0.00239dB と JM18.0 の RDOQ の符号化性能を維持している．

4.3 処理時間評価

処理時間の測定環境を表 10 に示す．また，表 11 に JM18.0 の RDOQ に対する提案手法の処理時間

表 8: 符号化性能測定環境

コンパイル	Visual Studio 2010(最適化無し)
CPU	Core i7-3770(動作周波数 3.40GHz)
フレーム数	50 フレーム
QP	22,27,32,37
QP の最適化	5 パターン

表 9: JM18.0 の RDOQ に対する提案手法の符号化性能

シーケンス	解像度	BD-BR[%]	BD-PSNR[dB]
news	CIF	-0.02	0.00099
container	CIF	-0.02	0.00072
football	CIF	0.47	-0.02644
paris	CIF	0.15	-0.00793
Desert	720x400	0.71	-0.02857
PartyScene	832x480	0.28	-0.01194
BBCPan13	720x576	-1.12	0.03516
old town cross	720p	0.28	-0.01194
In to tree	720p	-0.31	0.00692
rush hour	1080p	0.42	-0.01051
crowd run	1080p	0.15	-0.00561
平均		0.02	-0.00239

削減率を示す．推定ビットレートの計算は JM18.0 では様々なコンテキスト情報に対してルックアップテーブルにより求める．提案手法では依存性のあるコンテキスト情報は省いているので JM18.0 に比べ求める推定ビットレートが少なくて済んでいる．これにより平均で約 77 %の削減となる．歪計算部は JM18.0 より平均で約 2 %の削減となる．あまり高速化しなかった理由として，歪計算の式変換はハードウェアを考慮しているためと考えられる．次に最後の非ゼロ係数最適化および RD コスト計算部に関しては，探索範囲の削減や推定ビットレートの配列へのアクセス回数の大幅削減により，平均で約 43 %の削減となる．また，提案手法では CBP のビットレート推定を行わないため，JM18.0 で CBP 推定にかかる処理時間だけ高速化している．全体では平均で約 30 %の削減となった．

表 10: 処理時間測定環境

コンパイル	Visual Studio 2010(最適化無し)
CPU	Core i7-3770(動作周波数 3.40GHz)
フレーム数	3 フレーム
QP	27
QP の最適化	5 パターン

表 11: JM18.0 の RDOQ に対する提案手法の処理時間

シーケンス	解像度	削減率 [%]
news	CIF	31.01
container	CIF	31.57
football	CIF	32.43
paris	CIF	33.18
Desert	720x400	28.93
PartyScene	832x480	33.48
BBCPan13	720x576	31.46
old town cross	720p	29.84
In to tree	720p	29.45
rush hour	1080p	27.13
crowd run	1080p	31.65
平均		30.92

5 まとめと今後の課題

本文では RDOQ のハードウェア実装に有利な高速化アルゴリズムの提案，定量的な評価を行った．

提案する RDOQ の高速化アルゴリズムは H.264/AVC の参照ソフトウェアと比較し，符号化性能の維持しつつ約 30 %の高速化を実現し，また，係数間の依存関係をなくすことでハードウェア実装による並列処理を可能とした．

今後の課題として，次世代ビデオコーディング HEVC [7][8] に対して，同様の提案手法の適用を検討する．

参考文献

[1] 大久保榮, 角野眞也, 菊池義浩, 鈴木輝彦, “改定三版 H.264/AVC 教科書,”インプレス R&D，東京，2009.

[2] JVT reference software version 18.0 http://iphome.hhi.de/suehring/tml/download.

[3] Marta Karczewicz，Yan Ye，Peisong Chen:“Rate Distortion Optimized Quantization”，JVT-AA026.

[4] Limin Liu, and Alexis Tourapis: ”Rate distortion optimized quantization in the JM reference software,”JVT-AA027, April 24-29, 2008.

[5] Jing He, Fuzheng Yang: “High-speed implementation of rate-distortion optimized quantization for H.264/AVC,”©Springer-Verlag London 2013.

[6] G. Bjontegaard: “Calculation of average PSNR difference between RD-curves,” VCEG-M33, Austin, Texas, USA, April 2001.

[7] 村上篤道，浅井光太郎，関口俊一: ”高効率映像符号化技術 HEVC/H.265 とその応用，”オーム社，2013.

[8] 大久保榮, 鈴木輝彦, 高村誠之, 中條健“H.265/HEVC 教科書,”株式会社インプレスジャパン，東京，2013.

[9] 森口元気, et al. ”レート歪み最適化による量子化プロセスの高速化手法とその評価”, 電子情報通信学会信学技報 113(454): 67-72, 2014.