2003－AVM－42　(4)
2003／10／3

H.264 符号化処理における動き予測の高速化に関する一検討

清水 智行† 米山 暁夫† 柳原 広昌† 中島 康之†

概要 H.264 は MPEG-4 や H.263 などの従来の動画像圧縮符号化方式に比べて最大で 2 倍の符号化効率を目標とした符号化方式である。しかし、従来方式に比べて符号化処理の複雑さが非常に大きいため、実用化のためには符号化処理を効率よく行なうための手法が必要である。本稿では、H.264 の複数ブロックサイズによる動き補償予測処理において、ブロックサイズの選択および動きベクトル探索処理を高速に行なうための手法について検討する。具体的には、最初に小さなブロックサイズでの動き探索を行ない、隣接ブロックの動きベクトルの類似性から、より大きなブロックサイズを適応的に選択し、かつ探索範囲を絞り込むことによって、H.264 参照モデル JM と比較して 0.1-0.4dB の PSNR 低下で予測誤差計算回数を 6-7%に削減することができた。

Fast Motion Estimation Algorithm for H.264 Encoder

Tomoyuki SHIMIZU* Akio YONEYAMA* Hiromasa YANAGIHARA* Yasuyuki NAKAJIMA*

Abstract H.264 is a new video coding standard, which has coding efficiency about twice as much as existing standards such as MPEG-4, H.263, etc. Because it has larger computation complexity than existing standards, faster coding algorithms are desired for practical use. We propose a fast algorithm for multiple block size motion estimation. In this algorithm, motion vectors in smaller-sized block are searched at first, and a more suitable block size is chosen and search range is limited according to similarity of the motion vectors. As a result, search steps are reduced to about 6-7% of the H.264 Reference Model (JM), while loss of PSNR is at most 0.1-0.4 dB.

1. はじめに

H.264/MPEG-4 AVC[1](以下、H.264)は、ITU-T および ISO/IEC によって策定されている動画像の圧縮符号化標準であり、従来の符号化標準である MPEG-4[2]や H.263[3]に比べて最大で約 2 倍の圧縮符号化効率を持つ。しかし、H.264 は高い圧縮符号化効率を実現するために、イントラ予測や複数ブロックサイズでの動き補償、複数参照フレームによる予測符号化、1/4 画素動き補償、デブロッキングフィルタなど、数多くの符号化ツールを採用しているため、その圧縮符号化のためには非常に複雑な処理を必要とし、従来方式の数倍～数十倍の処理時間を有する。従って、実用化の観点においては、H.264 の圧縮符号化処理を効率よく行なうための高速化手法が必要不可欠である。

本稿では、特に複数ブロックサイズにおける動き補償予測処理に着目し、動き探索の高速化について検討する。第 2 章では従来方式および H.264 における動き補償方式について説明し、第 3 章ではブロックサイズと符号化効率の関係について考察し、第 4 章で提案方式について述べ、第 5 章でシミュレーション実験の結果を示し、第 6 章で結論を述べる。

2. 動き補償

2.1. 従来方式における動き補償

従来の動画像圧縮符号化方式における動き補償予測処理では、フレーム間の画素の相関性を利用して、フレーム間の特徴量の差分のみを符号化する、フレーム間予測符号化がよく用いられている。MPEG-1 や MPEG-2、H.261 等においては、16×16 画素のマクロブロック(以下、MB)毎に参照フレーム中の画素との差分(予測誤差)を求め、予測誤差を最小化する画素の相対位置を動きベクトル(以下、MV)として求め(図 1)、予測誤差を DCT(離散コサイン変換)などによって直交変換して得られる変換係

† 株式会社 KDDI 研究所
* KDDI R&D Laboratories Inc.

数を量子化したものと量子化係数(以下、QP; Quantization Parameter)、MV を可変長符号化する。

MPEG-4 Visual[2]や H.263[3]においては、1 つの MB を 8×8 画素のブロック 4 個に分割して、それぞれのブロックについて別々に動き補償を行なう 4MV モードと、MB 単位で動き補償を行うモードを MB 単

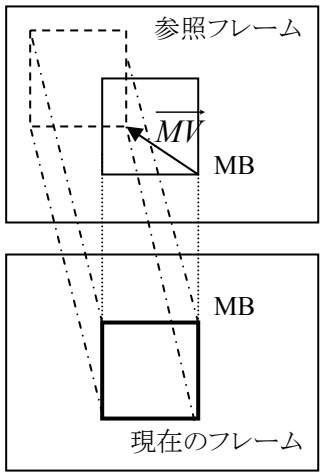

図 1: 動き補償

位で選択して利用することが可能である。

2.2. H.264 における動き補償

H.264 においては、動き補償を行なう MB の分割単位として、16×16 画素、16×8 画素、8×16 画素、8×8 画素の 4 種類のブロックサイズを選択することができる。また、8×8 画素ブロックモードを選択した場合は、さらに各 8×8 画素ブロックを 8×8 画素、8×4 画素、4×8 画素、4×4 画素の 4 種類のブロックサイズをサブブロックモードとして選択して分割することが可能である(図 2)。

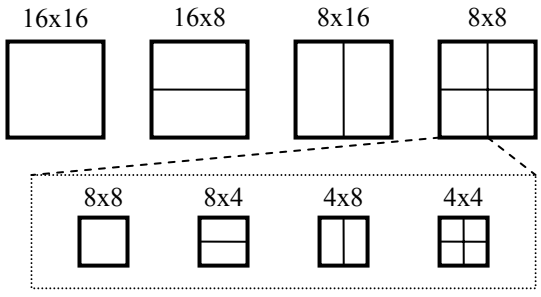

図 2: H.264 における動き補償ブロックサイズ

従って、H.264 においては、動画像内の物体の動きの大きさやばらつきに応じて適応的に動き補償ブロックサイズを選択することによって、予測誤差が小さくなるため、より効率的にフレーム間予測符号化を行なうことが可能となる。

3. ブロックサイズと符号化効率

3.1. レート歪み最適化

一つの MB を複数のブロックに分割して、それぞれのブロックに対して個別に動き補償を行なうことによって、画像によっては大幅に予測誤差を削減し、符号化効率に大きく貢献する。

一方、分割ブロックの数の増加に比例して、動きベクトルの個数も増加するため、動きベクトルの符号量も増加する。さらに、MPEG-4 Visual Version 2[4]や H.264 においては 1/4 画素精度の動き補償が採用されており、動きベクトルの符号量が従来方式に比べて多くなっている。

従って、H.264 において動き補償ブロックサイズを選択する際、予測誤差の減少と MV 符号量の増加のバランスを取ることによって、ビットレートあたりの量子化歪みを最小にするように符号化モードを決定する必要がある。

この問題を解決するために、H.264 の参照ソフトウェアとして開発されている、JM[5]の符号化処理においては、予測誤差 D と符号量 R の一次結合として求められる関数(レート歪みコスト関数)を各ブロックサイズについて求め、最小値を与えるモードを実際のモードとして選択する、レート歪み最適化を採用している[6]。

しかし、符号化処理において全てのモードに対してレート歪みコスト関数を求める場合、ブロックサイズが 1 種類しか利用できない従来方式に比べて、ブロックサイズの個数分だけ計算時間が倍増することとなる。従って、符号化処理を高速化するためには、全てのモードに対してレート歪みコスト関数を求めなくとも、最適なモードをある程度予測するための手法が必要となる。

3.2. ブロックサイズと動きベクトルの相関性

より少ない計算手間で最適な動き補償ブロックサイズを推定するために、まず小さなブロックサイズにおける MV と、大きなブロックサイズにおける MV の相関性について考察する。以下、例として、8×8 画素ブロックと 16×16 画素ブロックについて述べる。

8×8 画素ブロックを選択した方が符号化効率を

低くする場合、以下の条件のうち少なくとも一つが成立しているものと考えられる。

- 1 つの MB 中に含まれる 4 個の 8×8 画素ブロックの MV がいずれも近い向きと大きさを示している(図 3)。
- 1 つの MB 中に含まれる 4 個の 8×8 画素ブロックの MV について、ばらつきが非常に大きく、予測誤差の減少以上に MVD の符号量が大きくなっている(図 4)。

特に、4 個の MV が極めて類似している場合、その平均ベクトルで 16×16 画素ブロックにおける動きベクトルとして近似できるものと予想される。このとき、ベクトル間の距離を探索範囲として平均ベクトルの周辺を動き探索することで、より予測誤差の小さい点を効率よく探索することが可能であると考えられる。

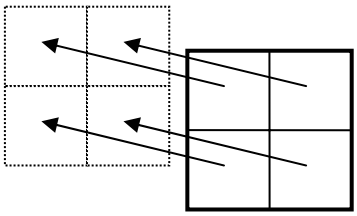

図 3: 動きベクトルが非常に近い場合

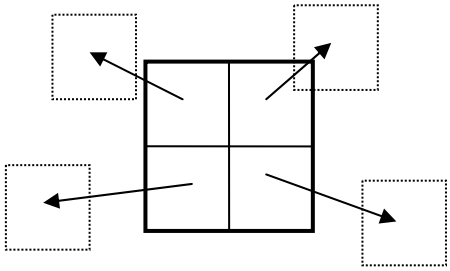

図 4: 動きベクトルにばらつきがある場合

4. 提案手法: 動き補償予測の高速化

3.2 節での考察により、小さなブロックサイズでの動き予測の結果から、大きなブロックサイズでの動き予測の方がレート歪み特性の観点から適切かどうかをある程度推定することが可能であると考えられる。

なお、予備実験によって、4×4、4×8、8×4 の 3 種類のブロックサイズを併用しても、PSNR の増加が対同一ビットレートで 0.1-0.2dB 程度であることが確認されている。そこで本実験では、16×16、16×8、8×16、8×8 の 4 種類のブロックサイズにおける動き予測について、動き予測を効率よく行なうための手法について考察する。

本提案手法の概要を図 5 に示す。

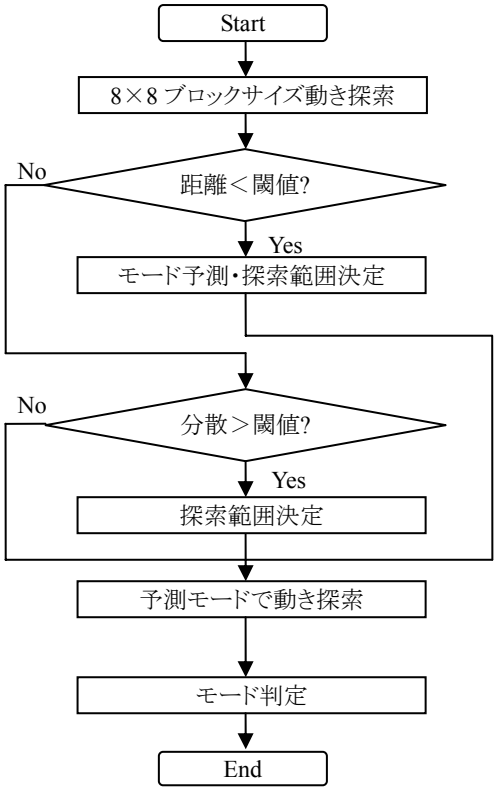

図 5: 提案手法概要

4.1. 手順 1: 8×8 ブロックでの動き予測

最初に、8×8 画素ブロック単位で探索範囲 r_{init} の動き探索を行なう。

探索アルゴリズムとしては、従来の高速サーチ手法、例えば 4SS[7]や Hexagon Search[8]等を用いることが可能であるが、ここでは、筆者らが提案した高速動き探索アルゴリズムである Ripple Search[9]を用いることとする。Ripple Search は MPEG-2 においてフルサーチと比較して 0.1-0.2 dB の PSNR の損失に抑えつつ約 50 倍の処理時間を実現している。

4.2. 手順 2: ブロックサイズの選択および探索範囲の決定

4.1 節で求めた 4 個の MV をそれぞれ $\overrightarrow{MV_{8\times8,1}}$ (左上)、$\overrightarrow{MV_{8\times8,2}}$ (右上)、$\overrightarrow{MV_{8\times8,3}}$ (左下)、$\overrightarrow{MV_{8\times8,4}}$ (右下)とする。また、$\overrightarrow{MV_{8\times8,n}} = (x_{8\times8,n}, y_{8\times8,n})$ とする(図 6)。

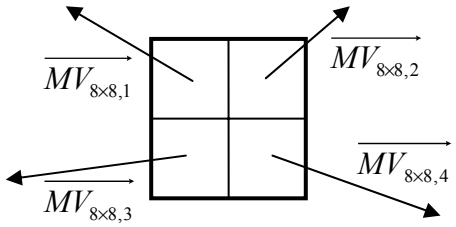

図 6: 8×8 ブロックにおける MV

まず、4 つの MV の平均ベクトル $\overrightarrow{MV_{8\times8,avg}}$ 分散 $\sigma_{8\times8}^2$、2 個のベクトル間の距離 D_{12}、D_{13}、D_{24}、D_{34} を次式(1), (2), (3)により求める。

$$\begin{cases} \overrightarrow{MV_{8\times8,avg}} = \left(x_{8\times8,avg}, y_{8\times8,avg}\right) \\ x_{8\times8,avg} = \dfrac{1}{4}\displaystyle\sum_{n=1}^{4} x_{8\times8,n} \\ y_{8\times8,avg} = \dfrac{1}{4}\displaystyle\sum_{n=1}^{4} y_{8\times8,n} \end{cases} \quad (1)$$

$$\sigma_{8\times8}^2 = \sum_{n=1}^{4} \frac{\left(x_{8\times8,n} - x_{8\times8,avg}\right)^2 + \left(y_{8\times8,n} - y_{8\times8,avg}\right)^2}{4} \quad (2)$$

$$D_{ij} = \sqrt{\left(x_{8\times8,i} - x_{8\times8,j}\right)^2 + \left(y_{8\times8,i} - y_{8\times8,j}\right)^2} \quad (3)$$

このようにして求められた平均ベクトル $\overrightarrow{MV_{8\times8,avg}}$ 分散 $\sigma_{8\times8}^2$、2 個のベクトル間の距離 D_{12}、D_{13}、D_{24}、D_{34} を用いて、適切な動き補償ブロックサイズの推定、および選択されたブロックサイズにおける探索原点 $\vec{C}$ と探索範囲 r を決定する。

4.2.1. 16×16 ブロックの選択

0 節で求めた 4 個の MV が全て互いに類似している場合は 16×16 画素ブロックがより適切である可能性が高い。従って、次式(4)が成立する場合は、16×16 画素ブロックを候補として動き探索を行なう。

$$\begin{cases} D_{12} < Thr1 \\ D_{13} < Thr1 \\ D_{24} < Thr1 \\ D_{34} < Thr1 \end{cases} \quad \left(Thr1\text{は定数}\right) \quad (4)$$

このとき、探索原点 $\vec{C}$ と探索範囲 r はそれぞれ次式(5), (6)によって決定する。

$$\vec{C} = \overrightarrow{MV_{8\times8,avg}} \quad (5)$$

$$r = \min\left(\min\left(D_{12}, D_{13}\right), \min\left(D_{24}, D_{34}\right)\right) \quad (6)$$

4.2.2. 16×8 ブロックの選択

$\overrightarrow{MV_{8\times8,1}}$ と $\overrightarrow{MV_{8\times8,2}}$、もしくは $\overrightarrow{MV_{8\times8,3}}$ と $\overrightarrow{MV_{8\times8,4}}$ のうちいずれかの組が互いに類似している場合、16×8 画素ブロックがより適切である可能性が高い。従って、次式(7)が成立する場合は、16×8 画素ブロックを候補として動き探索を行なう。

$$D_{12} < Thr2 \quad or \quad D_{34} < Thr2 \quad \left(Thr2\text{は定数}\right) \quad (7)$$

このとき、上下 2 個の 16×8 画素ブロックそれぞれについて、探索原点 $\overrightarrow{C_1}, \overrightarrow{C_2}$ と探索範囲 r_1, r_2 を次式(8), (9)のように決定する。

$$\begin{cases} \overrightarrow{C_1} = \left(\dfrac{x_{8\times8,1} + x_{8\times8,2}}{2}, \dfrac{y_{8\times8,1} + y_{8\times8,2}}{2} \right) \\ \overrightarrow{C_2} = \left(\dfrac{x_{8\times8,3} + x_{8\times8,4}}{2}, \dfrac{y_{8\times8,3} + y_{8\times8,4}}{2} \right) \end{cases} \quad (8)$$

$$\begin{cases} r_1 = D_{12} \\ r_2 = D_{34} \end{cases} \quad (9)$$

4.2.3. 8×16 ブロックの選択

$\overrightarrow{MV_{8\times8,1}}$ と $\overrightarrow{MV_{8\times8,3}}$、もしくは $\overrightarrow{MV_{8\times8,2}}$ と $\overrightarrow{MV_{8\times8,4}}$ のうちいずれかの組が互いに類似している場合、8×16 画素ブロックがより適切である可能性が高い。従って、次式(10)が成立する場合は、8×16 画素ブロックを候補として動き探索を行なう。

$$D_{13} < Thr2 \quad or \quad D_{24} < Thr2 \quad (10)$$

このとき、上下 2 個の 8×16 画素ブロックそれぞれについて、探索原点 $\overrightarrow{C_1}, \overrightarrow{C_2}$ と探索範囲を次式(11), (12)のように決定する。

$$\begin{cases} \overrightarrow{C_1} = \left(\dfrac{x_{8\times8,1} + x_{8\times8,3}}{2}, \dfrac{y_{8\times8,1} + y_{8\times8,3}}{2} \right) \\ \overrightarrow{C_2} = \left(\dfrac{x_{8\times8,2} + x_{8\times8,4}}{2}, \dfrac{y_{8\times8,2} + y_{8\times8,4}}{2} \right) \end{cases} \quad (11)$$

$$\begin{cases} r_1 = D_{13} \\ r_2 = D_{24} \end{cases} \quad (12)$$

4.2.4. 動きベクトルのばらつきが大きい場合

4.1 節で求めた 4 個の MV のばらつきが非常に大きい場合は 16×16、16×8、8×16 画素ブロックそれぞれについて動き探索を行なう。

次式(13)が成立する場合は、16×16 画素ブロックを候補として動き探索を行なう。

$$\sigma_{8\times8}^2 > Thr3 \qquad (Thr3\text{は定数}) \quad (13)$$

このとき、探索原点$\vec{C}$または$\overrightarrow{C_1},\overrightarrow{C_2}$は 4.2.1 節、4.2.2 節、4.2.3 節と同様にして求める。探索範囲rまたはr_1,r_2は次式(14)により決定する。

$$r = r_1 = r_2 = r_{init} \quad (14)$$

4.3. 手順 3: 選択されたブロックサイズでの動き探索

4.2 節の式(4), (7), (10), (13)のうちいずれかが成立する場合は、選択されたブロックサイズにおいて動き探索を行なう。ここで、探索原点は$\vec{C}$もしくは$\overrightarrow{C_1},\overrightarrow{C_2}$、探索範囲は$r$もしくは$r_1,r_2$である。

ここでは探索アルゴリズムとして 4SS[7]を用いることとする。ただし、探索範囲の絞込みのため、、ステップ数を$\frac{r}{2}$もしくは$\frac{r_1}{2},\frac{r_2}{2}$とする。

4.4. 手順 4: 最終モード選択

手順 1-3 によって探索を行なった全てのモード(ブロックサイズ)のうち、レート歪みコスト関数が最小となるモードを最終的に選択するモードとする。

5. 実験および考察

4 章で示した提案方式を JM 7.3[5]に実装し、符号化シミュレーション実験を行なった。

表 1: 符号化条件

評価動画像	Foreman, Mobile & Calendar
画像サイズ	CIF (352×288 ピクセル)
フレームレート	30fps
ピクチャ構造	(M, N) = (3, 15)
動き補償 ブロックサイズ	16×16, 16×8, 8×16, 8×8 または 8×8 のみ
レート制御	なし (QP 固定)
エントロピー符号化	CABAC
レート歪み最適化	オン
その他	P, B フレームにおいてイントラ MB 使用不可

本実験においては、動き探索のみの性能評価を行なうため、P, B フレームにおいてイントラ予測 MB を使わないように符号化処理を行なうこととする。また、処理時間の比較のため、8×8 ブロックサイズのみを使用して Ripple Search を行なった結果も示す。

まず、符号化ビットレートと PSNR の関係を図 7 および図 8 に示す。元の JM (JM Original)と提案方式を実装した JM (Proposed)の差は、Foreman において高ビットレートで 0.2dB、低ビットレートでは 0.4dB、Mobile & Calendar においては高低ビットレートを通じて 0.1dB 程度である。

なお、元の JM においては、整数画素精度での探索はフルサーチによって行なわれている。

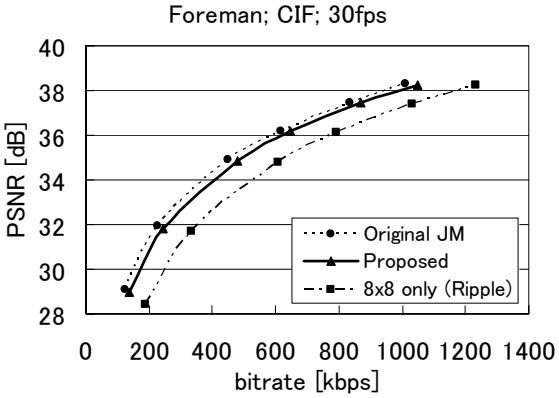

図 7: レート歪み特性(Foreman)

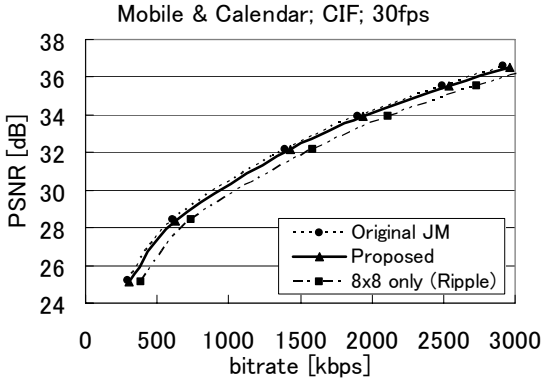

図 8: レート歪み特性(Mobile & Calendar)

表 2: ブロックサイズ選択比率[%] (P フレーム)

Block Size	QP = 28		QP = 35	
	Original	Proposed	Original	Proposed
Skip or Direct	23.36	24.98	41.95	43.07
16×16	31.46	28.40	30.50	27.78
16×8	15.61	16.24	11.50	12.28
8×16	17.28	16.47	12.38	11.82
8×8	12.29	13.90	3.67	5.05

表 3: ブロックサイズ選択比率[%] (B フレーム)

Block Size	QP = 28		QP = 35	
	Original	Proposed	Original	Proposed
Skip or Direct	5.55	6.46	14.38	16.88
16×16	68.04	62.78	76.29	71.34
16×8	10.58	10.37	4.73	5.39
8×16	11.03	10.34	3.59	4.03
8×8	4.80	10.04	1.00	2.36

次に、Foreman について、P, B ピクチャにおけるブロックサイズモードの選択比率を表 2 および表 3 に示す(単位: マクロブロック)。

表 2 および表 3 において、P ピクチャ、B ピクチャのいずれについても、提案手法において 16×16 ブロックモードが選択された MB の数が元の JM より多くなっており、ビットレートを増加させている原因と考えられる。

処理速度を比較するため、Foreman において、提案手法および 8×8 モードのみの探索を行なった場合について、1MB 当りの 4×4 ブロック換算による予測誤差計算回数を表 4 に、1 フレーム当りの符号化処理に要した平均時間を表 5 に示す。

表 4: 計算回数 (QP=28)

	予測誤差計算回数(4×4ブロック換算)	
	P-frame	B-frame
Original JM	14080.00	34848.00
8×8 only (Ripple)	568.80[4.04%]	1395.60[4.01%]
Proposed	961.09[6.83%]	2185.93[6.27%]

表 5: 符号化処理時間 (QP=28)

	符号化処理時間[msec/frame]	
	P-frame	B-frame
Original JM	1133.213	2222.626
8×8 only (Ripple)	235.550[20.79%]	446.707[20.10%]
Proposed	554.300[48.91%]	1129.338[50.81%]

表 4 によると、提案手法においては、予測誤差の計算回数が元の JM の約 6-7%になっている。8×8 モードのみの場合に比べると、モード数が 4 倍となっているのに対して、4×4 予測誤差計算回数は約 1.6 倍となっている。

一方、表 4 および表 5 によると、提案手法における P および B フレームの処理時間は、元の JM に比べて約 50%程度に削減されている。8×8 ブロックサイズのみの Ripple Search の場合に比べると、モード数が 4 倍となっているのに対して、処理時間は約 2.5 倍となっている。

なお、JM においては、ブロックサイズ決定より前に、各ブロックサイズの動き探索に対して 1/4 画素動きベクトルの処理を行うように実装されている。従って、整数画素精度において最適ブロックサイズを決定してから 1/4 画素精度の処理を行なうことによって、さらに高速化されるものと考えられる。

6. おわりに

本稿では H.264 における複数ブロックサイズ動き補償予測の高速化手法を提案した。本手法においては、元の JM と比較して、PSNR の劣化を 0.1-0.4 dB 程度に抑えつつ、予測誤差の計算回数を約 50%に、動き探索の処理時間を約 50%にまで削減することができた。

本手法においては、8×8 ブロックサイズにおける動きベクトルの類似度の比較によって、レート歪み特性の観点においてより適切なブロックサイズを選択する。しかし、単純にベクトル間の距離および分散によって判定を行なうため、QP やビットレートなどの影響によるブロックサイズの選択の変化を考慮するなど、更なる考察が必要であるものと思われる。

参考文献

[1] ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification,” 2003.

[2] ISO/IEC 14496-2, “Information technology – Coding of audio-visual objects Part 2: Visual,” 2001.

[3] ITU-T Rec. H.263, “Video coding for low bitrate communication,” 1996.

[4] ISO/IEC 14496-2: 2001/Amd.2: 2002 Part 2: Visual, "AMENDMENT 2: Streaming video profile," 2002.

[5] JM 7.3, http://bs.hhi.de/~suehring/tml/.

[6] ISO/IEC | ITU-T VCEG, JVT-B118r8, 2002.

[7] L-M. Po and W-C. Ma, “A Novel Four-Step Search Algorithm for Fast Block Motion Estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, No. 3, pp.313-317, 1996.

[8] ISO/IEC | ITU-T VCEG, “Fast Integer Pel and Fractional Pel Motion Estimation for JVT,” JVT-F017, 2002.

[9] Y. Nakajima, A. Yoneyama, et al, “A Fast Motion Estimation Algorithm for MPEG2 Video Using Ripple Shaped Search,” *IEEE Proc. ISCAS*, vol. 4, pp.207-210, 1999