社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－ＳＬＤＭ－121（12）
2005／10／20

モバイル応用H.264コーデックＬＳＩ向け 低消費電力動き検出アルゴリズム

濱本 真生† 長井 健一† 松野 哲郎‡ 村地 勇一郎† 宮越 純一†

深山 正幸‡ 吉本 雅彦†

†神戸大学大学院自然科学研究科 〒657-8501 兵庫県神戸市灘区六甲台町 1-1
‡金沢大学大学院自然科学研究科 〒920-1192 石川県金沢市角間町

E-mail: †hamamoto@cs28.cs.kobe-u.ac.jp

あらまし H.264では16x16画素～4x4画素までの大小7つのブロックサイズでの動き補償が可能であり，これにより符号化効率が向上する反面，動き検出の演算量がブロックサイズ数に対し線形的に増加し，さらにそれを専用ハードウェアで実現するための回路構成も複雑となる．そこで，増大した演算量と回路構成の複雑度を低減させる高速ブロックサイズ決定法を開発した．高速ブロックサイズ決定法は大きなブロックサイズにおける探索法（FSLB）と小さなブロックサイズにおける探索法（FSSB）からなるブロック分割アルゴリズムである．また，探索アルゴリズムには広い探索範囲を低演算量で探索する1D-DSを導入した．高速ブロックサイズ決定法と1D-DSを組み合わせた提案法はH.264符号化シミュレーションにおいてFSに対し画質劣化を平均0.1dB程度に抑えつつ，演算量を97%削減した．

キーワード H.264，動き検出，フレーム間予測，低消費電力

A low power H.264 motion estimation algorithm for mobile video applications

Masaki HAMAMOTO† Kenichi NAGAI† Tetsuro MATSUNO‡ Yuichiro MURACHI†

Junichi MIYAKOSHI† Masayuki MIYAMA‡ Masahiko YOSHIMOTO†

† Kobe University Graduate School of Science and Technology 1-1 Rokkodai-cho, Nada-ku, Kobe-shi, Hyogo, 657-8501 Japan

‡ Kanazawa University Graduate School of Natural Science & Technology Kakuma-cho, Kanazawa-shi, Ishikawa, 920-1192 Japan

E-mail: †hamamoto@cs28.cs.kobe-u.ac.jp

Abstract H.264 employs variable block size motion compensation which supports 7 block sizes from 16x16 to 4x4. This feature enhances coding efficiency of H.264. However it increases both the computation load for motion estimation (ME) and the hardware complexity implementation. In this paper, we propose an ME algorithm using fast search method for variable block sizes and 1D-DS for H.264 video coding. Fast search method for variable block sizes consists of two search methods for the larger size block and smaller size block. The 1D-DS performs ME on large search range with low computational complexity. The experimental result shows that the proposed algorithm provides 97% reduction of workload with Image quality degradation of 0.1dB.

Keyword H.264, motion estimation, inter mode, Low power

1. はじめに

最新の動画像圧縮符号化標準であるH.264/AVC[1]（以下，H.264）は従来方式であるH.263やMPEG-4に対し最大約2倍の符号化効率を有する．しかし，その圧縮符号化のための演算量は従来方式の数十倍であり，特に最も演算負荷が大きいのは動き検出処理である．

H.264の動き補償では16x16画素（mode1）～4x4画素（mode7）までの大小7つのブロックサイズでの動き補償が可能であり，また1/4画素精度の動き補償ま

でが可能である．これにより符号化効率が向上する反面，動き検出の演算量が増大し，さらにそれを専用ハードウェアで実現するための回路構成も複雑となる．そのため，動き検出処理の演算量削減のためには複数ブロックサイズによって増大した演算量を削減するブロック分割アルゴリズムと動き検出における演算量を削減する探索アルゴリズムの両方が必要となる．

これまでに，複数ブロックサイズ動き補償についての演算量を削減するアルゴリズムとしてさまざまなアルゴリズムが提案されているが，その多くは評価値が予め設定した閾値よりも下回った段階で探索を打ち切るアルゴリズム[2][3]である．これらのアルゴリズムでは平均演算量を大幅に削減することができるが，入力画像によっては全てのブロックサイズで動き検出を行うことがあるため，最大演算量を削減することができない．VLSI 実装においては，最大演算量にあわせて回路構成と動作周波数を設定する必要があるため，これらの最大演算量を削減できないアルゴリズムでは回路構成の複雑度も動作周波数も低減することができない．

また，動き検出における探索アルゴリズムについては従来法として H.264 標準化の際に使用された参照ソフトウェア [4]（JM：Joint Model）に実装されている全探索法（整数画素探索法）と 8 近傍-8 近傍探索法（小数画素探索法）が挙げられるが，全探索法では演算量が膨大であり，8 近傍-8 近傍探索法は適応的処理であるために低消費電力を目的とした VLSI 実装のためのアルゴリズムとしては不向きである．そこで本研究では，最大演算量と画質に注目し，高画質を維持しながら最大演算量を削減し，かつ回路規模と動作周波数を低減することのできるブロック分割アルゴリズムと探索アルゴリズムを組み合わせた VLSI 向け H.264 動き検出アルゴリズムを提案する．

本稿では，2 章に提案ブロック分割アルゴリズムを示し，3 章で提案探索アルゴリズムを詳述する．4 章では従来法と提案法とのシミュレーション結果を比較し，5 章で本論文をまとめるものとする．

2. ブロック分割アルゴリズム

H.264 の動き補償では 7 つのブロックサイズが用意されているが（Fig. 1），全てのブロックサイズについて動き検出を行うと非常に大きな演算量が必要となる．そのため，低演算量で効率よく最適なブロックサイズを決定するブロック分割アルゴリズムが必要となる．

本章では提案するブロック分割アルゴリズムについて説明する．2.1 節では大きなブロックサイズに対するアルゴリズム FSLB（Fast Search method for Large Blocks：FSLB）を説明する．2.2 節では小さなブロックサイズに対するアルゴリズム FSSB（Fast Search method for Smaller Blocks：FSSB）を説明する．

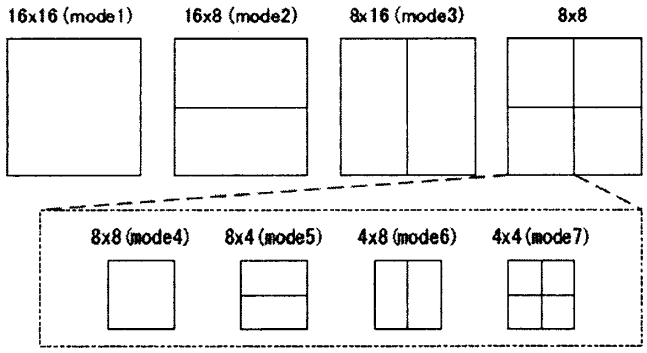

Fig. 1　H.264 動き補償ブロックサイズ

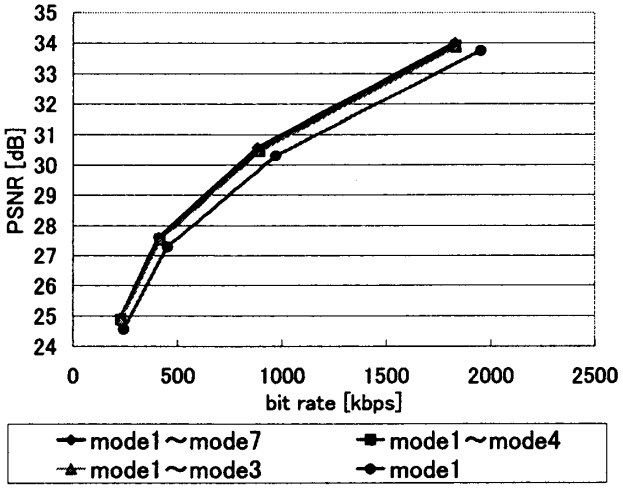

Fig. 2　複数ブロックサイズの効果（Mobile&Calendar）

2.1. 大ブロック高速探索法（Fast Search method for Larger Blocks：FSLB）

Figure 2 に複数ブロックサイズによる符号化効率向上への効果を示す．Fig. 2の曲線は R-D 曲線と呼ばれ，符号化効率を示す．横軸はビットレート，縦軸は PSNR であり，曲線が左上にあるほど高い符号化効率を有することを示す．Fig. 2は JM8.5[4]において使用可能なブロックサイズを制限した場合の符号化シミュレーション結果である．シミュレーション条件は 4 章のTable 2 と同様である．

Fig. 2より H.264 の複数ブロックサイズ動き補償においては大きなブロックサイズほど圧縮性能向上への寄与が大きいことが分かる．また，VLSI 実装においては小さなブロックにおける動き検出ほど回路構成が複雑となる．そのため，探索を行うのは大きなブロックサイズ（mode1～mode3）のみの方が，回路構成，圧縮性能の観点からみて効率がよい．そこで本研究では，mode1～mode3 の整数画素探索を効率よく行う手法と

して FSLB を提案する．

FSLB は mode1～mode3 までのブロックサイズに対するブロック分割アルゴリズムである．mode1 の最適 MV は，Fig. 3に示される mode2 の上下のブロック A，B，mode3 の左右のブロック C，D の動き検出によって得られた 4 つの MV の間に存在すると考えられる．そこで FSLB は，まず mode2，mode3 による整数画素探索を行う．次に，mode1 の整数画素探索を予測ベクトルと mode2，mode3 での整数画素探索結果をから得られる候補ベクトルより mode1 の MV を決定する．ここで，下記に mode1 探索用の候補ベクトル(CMV)を示す．候補ベクトルは全 8 本であり，予測ベクトル(PMV)と mode2,3 の結果の 7 本のベクトルで構成される．

FSLB により mode1 の整数画素探索演算量を 8 点のブロックマッチングのみに抑えることができる．

$$\begin{cases} CMV_1 = PMV & (1) \\ CMV_2 = MV_A & (2) \\ CMV_3 = MV_B & (3) \\ CMV_4 = MV_C & (4) \\ CMV_5 = MV_D & (5) \\ CMV_6 = (MV_A + MV_B)/2 & (6) \\ CMV_7 = (MV_C + MV_D)/2 & (7) \\ CMV_8 = (MV_A + MV_B + MV_C + MV_D)/4 & (8) \end{cases}$$

*MV : Motion Vector
*PMV : Predicted Motion Vector
*CMV : Candidate Motion Vector

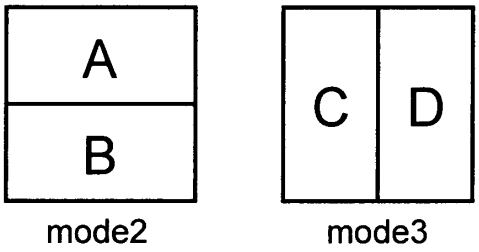

Fig. 3　mode1 の探索候補点算出に用いるブロック

2.2. 小ブロック高速探索法(Fast Search method for Smaller Blocks:FSSB)

前節に述べたように，小さなブロックサイズにおける動き補償は，VLSI 実装における回路構成を複雑にする．そこで，本研究では単純な回路構成，および低演算量で小さなブロックサイズの探索を，高画質を維持したまま実行できる FSSB アルゴリズムを提案する．FSSB は，小さなブロックサイズである mode4,5,6,7 の動き検出を，大きなブロックサイズである mode1,2,3 の動き検出の過程より算出するアルゴリズムである．

mode1～mode3 の小数画素探索の過程では，それら mode に含まれる 4x4 画素ブロックの評価値 SATD (Sum of Absolute Transformed Difference) を得ることができる．つまり，それらの 4x4SATD の加算の仕方を工夫することで，すべての mode の SATD 評価値を算出することが可能である．これから，FSSB は mode1～mode3 の探索によって得られる SATD を利用して mode4～mode7 の探索を実行するアルゴリズムであるといえる．これは mode4～7 までの探索を行わずに SATD 結果を得られることを示している． FSSB の探索手順を以下に示す．手順中の記号はFig. 4に示す．各 8x8 画素ブロックにおける mode4～mode7 の動き検出を以下の手順で実行し mode4～mode7 の MV を決定する．

Step1：最小評価値の更新

①mode1 の小数画素探索

mode1 の探索に含まれる E，F，G，H の 8x8 画素ブロックに対して mode4～mode7 の各ブロックの SATD を算出し，最小評価値を取得する

②mode2 の上ブロックの小数画素探索

mode2 の上探索に含まれる E，G の 8x8 画素ブロックに対して mode4～mode7 の各ブロックの SATD を算出し最小評価値を更新する

③mode2 の下ブロックの小数画素探索

②と同様に，F，H の 8x8 画素ブロックに対して SATD を算出し最小評価値を更新する．

④mode3 の左ブロックの小数画素探索

mode3 左探索に含まれる E，F の 8x8 画素ブロックに対して mode4～mode7 の各ブロックの SATD を算出し最小評価値を更新する．

⑤mode3 の右ブロックの小数画素探索

④と同様に G，H の 8x8 画素ブロックに対して SATD を算出し最小評価値を更新する．

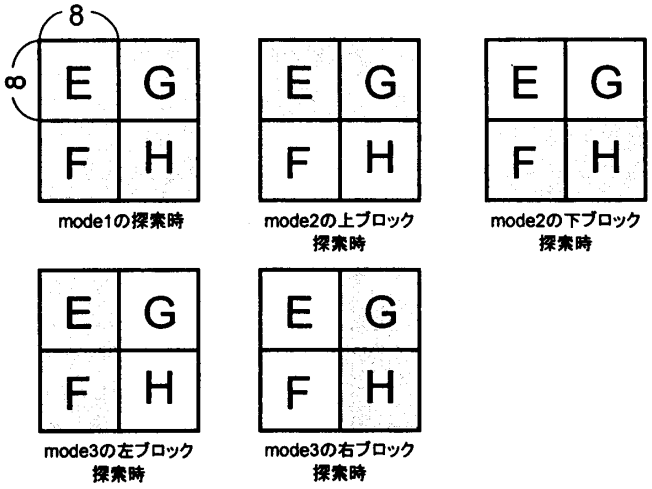

Fig. 4　評価値を更新する 8x8 画素ブロックの位置

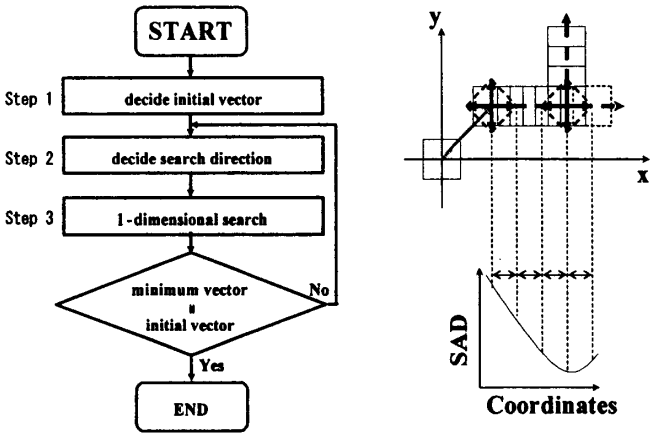

Fig. 5　1D-DS アルゴリズム

Step2：最適 MV 決定

全ての 8x8 画素ブロックにおける mode4～mode7 に対し，Step1 で得られた最小評価値を与える点をそれぞれの mode の各ブロックにおける最適 MV とする．

3. 探索アルゴリズム

本章では，各 mode における動きベクトルの検出法(探索アルゴリズム)の最適化を行う．3.1 では整数画素精度の探索アルゴリズムとして低演算量を実現する 1 Dimensional – Diamond Search(1D-DS)[5]の詳細とその導入について説明する．3.2 では小数画素精度の探索アルゴリズムとして VLSI 実装に適した 35 点 FS（Full Search：FS）の詳細とその導入について説明する．

3.1. 整数画素精度探索アルゴリズム 1D-DS

本節では，整数画素精度の探索法として低演算量を実現する 1D-DS を提案する．1D-DS は最急降下法に基づいた探索アルゴリズム（勾配法）の一種である．1D-DS の詳細をFig. 5に示す．ステップ 1 は最初の探索開始点を複数の候補から選ぶ．ステップ 2 では初期ベクトル周辺の上下左右 4 点の差分絶対値和 (SAD) を求め，最も SAD の小さい方向に探索方向を決定する．ステップ 3 ではステップ 2 において決定された探索方向に向かって一次元探索を行い，SAD 値が最も小さい点を求める．その後，ステップ 2.3 を複数回繰り返すことにより，最適点を探索する．ただし，一次元探索において，探索開始点が最適点であった場合はその後の探索は打ち切られる．1D-DS は演算量を大幅に削減することが可能である．

3.2. 小数画素精度探索アルゴリズム

小数画素精度探索アルゴリズムの最も基本的な従来法として JM に実装されている 8 近傍-8 近傍探索が挙げられる．8 近傍-8 近傍探索は±0.75x±0.75 画素（1/4 画素精度で計 49 点）の探索範囲で行われる．8 近傍-8 近傍探索は以下の二つの step により構成される．

Step1　1/2 画素精度において探索開始点の周囲 8 点を探索

Step2　1/4 画素精度において step1 で最小評価値を与えた点の周囲 8 点を探索

以上の結果から，探索開始点と step.1 と step.2 で探索した 16 点，合わせて 17 点のうち最も小さい評価値を与える点を最終的な動きベクトルとする．

本節では 8 近傍-8 近傍探索の VLSI 実装における問題点を挙げ，その解決策として 35 点 FS による小数画素精度探索を提案する．

3.2.1. VLSI 向け最適化

8 近傍-8 近傍探索は step1 の結果を基に step2 の探索点を決定する 2step の適応的探索である．このため VLSI 実装時は step1 と step2 とで同じ画素に対し 2 度のメモリアクセスを行うか，step1 の探索時に生成した画素を step2 実行時まで保存しておくためのキャッシュを持つ必要がある．また，8 近傍-8 近傍探索では step1 が終了しないと step2 の探索が始められないため，パイプライン・ストールが起こる．加えて，H.264 の小数画素精度探索では 6 タップ・フィルタによる 1/2 画素生成など複雑な演算処理に伴ってパイプライン段数も大きくなると考えられるため，パイプライン・ストールによる無効サイクルは無視できない．

そこで本稿では小数画素精度探索として FS を提案する．FS は予め探索点が決まっており，さらに画素の再利用性が極めて高いことから VLSI 実装においてパイプライン・ストールを起こすことがなく，メモリア

クセス数も抑えることができるという特徴を持っている．そのため，小数画素探索のように狭い範囲の探索には有効な探索法である．本稿では小数画素精度探索アルゴリズムとして FS を導入し，自然画像に横方向の動きが大きいことから探索範囲を±0.75x±0.5 画素（1/4 画素精度で計 35 点）とした．

3.2.2. 平均演算量削減

小数画素探索アルゴリズムが FS の場合，mode 間で探索開始点が一致したブロック，つまり mode 間で整数画素の探索結果が一致したブロックは全く同一の探索を行うことになる．そのため，mode 間で整数画素探索の結果が一致した場合は SATD 算出を省略することができる．これにより小数画素探索の平均演算量を削減することができる．本稿で提案するブロック分割アルゴリズムにおいては mode1 の小数画素探索で算出した SATD を保持しておき，mode2，または mode3 の各ブロックの整数画素探索で mode1 の整数画素探索結果と一致したブロックの小数画素探索における SATD 算出を省略する．

4. シミュレーション結果および考察

本章では，2 章で提案したブロック分割アルゴリズム，3 章で提案した探索アルゴリズムを JM8.5[4]に実装し，符号化シミュレーション実験によって提案法の性能を評価する．ここでは従来法として FS で探索を行う元の JM（従来法 1），1D-DS を実装した JM（従来法 2）を挙げ，提案法との性能比較を行った．従来法と提案法におけるブロック分割，整数画素探索，及び小数画素探索アルゴリズムを Table 1に示す．1D-DS のパラメータは探索開始点の候補点を Hexagon Search[6]で採用されている初期探索点と同一の点とし，一括探索点数を 5 点，最大探索回数を 2 回とした．シミュレーション条件を Table 2に示す．

Table 1　従来法と提案法のアルゴリズム

	ブロック分割	整数画素探索	小数画素探索
従来法 1	全mode探索	FS	8近傍-8近傍探索
従来法 2	全mode探索	1D-DS	8近傍-8近傍探索
提案法	FSLB，FSSB	1D-DS	35点FS

Table 2　シミュレーション条件

プロファイル	Baseline Profile
画像サイズ	CIF（352×288ピクセル）
フレームレート	30 fps
QP	28，32，36，40（全フレーム固定）
動き探索範囲	±16
参照ピクチャ数	3枚
レート歪最適化	オフ
アダマール変換	オン

符号化効率の評価は画質劣化（BDPSNR）とビットレート増分（BDBR）で行った（Table 3）[7]．これらの値は，4 つの等間隔の QP 値におけるビットレートおよび PSNR の値から 3 次の近似曲線を描き，この RD 曲線を比較することによって求められる．

提案法における符号化効率は Foreman を除き画質劣化が 0.1dB 程度と良好な結果となった．Foreman において 0.26dB の画質劣化が生じた原因は，FSSB によって mode4～mode7 についての整数画素探索を省略していることが主な原因であると考えられる．

Table 3　符号化効率の比較

Sequence	従来法2		提案法	
	BDBR[%]	BDPSNR[dB]	BDBR[%]	BDPSNR[dB]
Foreman	2.75	-0.14	5.40	-0.26
Mobile&Calendar	0.05	0.00	1.07	-0.05
Bus	1.34	-0.06	3.00	-0.13
Susie	1.32	-0.06	1.95	-0.09
Akiyo	0.63	-0.03	1.50	-0.07
FlowerGarden	0.13	-0.01	1.21	-0.06
average	1.04	-0.05	2.35	-0.11

演算量の評価は従来法 1 の演算量を基準（100%）として平均演算量と最大演算量で行った．平均演算量は 4 つの QP 値における平均演算量の平均値をとった．演算量はブロックマッチング演算量と小数画素生成の演算量の和として表し，加算一回相当の演算を 1 operation として見積もった．

Fig. 6は各アルゴリズムでの最大演算量の比較を示している．最大演算量の比較において，提案法は従来法 1 に対し 4.5%，従来法 2 に対して 58%まで演算量を削減した．最大演算量の内訳として小数画素精度の探索による演算量が支配的となったが，これは小数画素探索に 35 点 FS を導入しているためだと考えられる．

Table 4は従来法 2 における平均演算量を示しており，Table 5は提案法における平均演算量を示している．平均演算量の比較において，提案法は従来法 1 に対し 2.3%～3.0%，従来法 2 に対し 50%程度まで演算量を削減した．また，提案法における小数画素探索について平均演算量と最大演算量を比較すると，3.2.2 節で述べた 35 点 FS の平均演算量削減法の効果により 50%以上の平均演算量が削減されていることが分かる．

Fig. 7は各アルゴリズムにおける平均画質劣化（BDPSNR）と平均演算量（全体演算量）をまとめたグラフであり，各アルゴリズムの性能を示す．このグラフより，提案法は従来法 1 に対し，画質劣化を 0.1dB 程度に抑えつつ，演算量を 97%以上削減するアルゴリズムであることが分かる．さらに，提案法は 35 点 FS によるパイプラインストール回避と最大演算量削減の効果により，従来法 2 に対して必要サイクル数を半分以下に削減することができ，低消費電力化向きのアルゴリズムである．

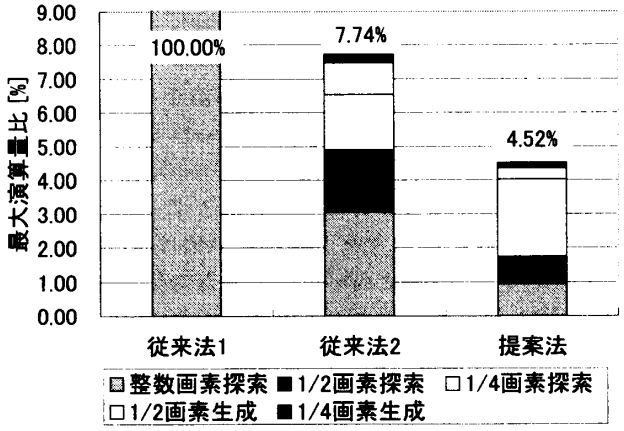

Fig. 6　最大演算量の比較

Table 4　平均演算量の比較（従来法 2）

Seqence	従来法2		
	整数画素探索演算量[%]	小数画素探索演算量[%]	全体演算量[%]
Foreman	1.49	4.70	6.19
Mobile&Calendar	1.45	4.70	6.15
Bus	1.70	4.70	6.41
Susie	1.39	4.70	6.10
Akiyo	1.56	4.70	6.26
FlowerGarden	1.17	4.70	5.87
average	1.46	4.70	6.16

Table 5　平均演算量の比較（提案法）

Seqence	提案法		
	整数画素探索演算量[%]	小数画素探索演算量[%]	全体演算量[%]
Foreman	0.67	2.28	2.95
Mobile&Calendar	0.67	1.97	2.64
Bus	0.72	2.29	3.01
Susie	0.64	2.18	2.83
Akiyo	0.70	2.04	2.74
FlowerGarden	0.60	1.68	2.28
average	0.67	2.07	2.74

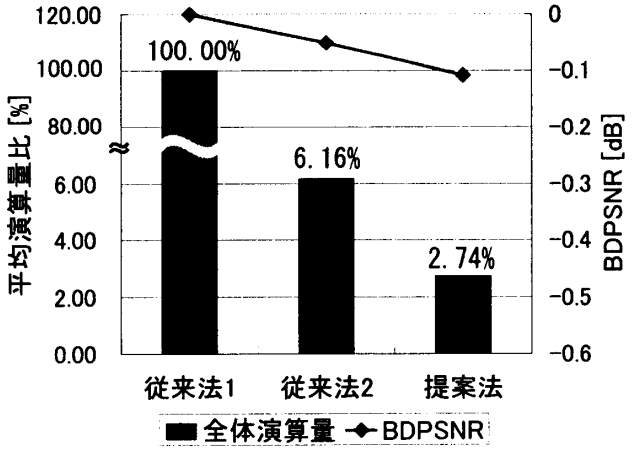

Fig. 7　各アルゴリズムの性能比較

5. まとめ

本稿ではブロック分割アルゴリズムと探索アルゴリズムを組み合わせた VLSI 向け H.264 動き検出アルゴリズムを提案した．提案法は元の JM（従来法 1）に対し，画質劣化を 0.05dB～0.26dB に抑えつつ，平均演算量 3%以下を達成した．また，本提案法は VLSI 実装を意識しており，VLSI 実装において低消費電力を実現する動き検出アルゴリズムである．

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] Woong Il Choi, Byeungwoo Jeon and Jechang Jeong, "FAST MOTION ESTIMATION WITH MODIFIED DIAMOND SEARCH FOR VARIABLE MOTION BLOCK SIZE," The IEEE International Conference on Image Processing(ICIP), Vol.3, pp.371-374, Sept. 2003.

[3] Libo Yang, Keman Yu, Jian Li and Shipeng Li, "An Effective Variable Block-Size Early Termination Algorithm for H.264 video Coding," IEEE Transactions on Circuits and Systems for Video Technology, Vol.15, No.6, pp.784–788, June 2005.

[4] JM 8.5, http://iphome.hhi.de/suehring/tml/.

[5] Yuichiro MURACHI, et al., "A 95 mW MPEG2 MP@HL Motion Estimation Processor Core or Portable High-Resolution Video Application", IEICE Trans. Fundamentals, (in press).

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

[7] G. Bjontegard, "Calculation of Average PSNR Differences between RD-curves," ITU-T VCEG-M33, 2001.