卒業論文

題目

スーパーハイビジョン対応
動き探索部の第一階層構成法の
研究

指導教員

近藤 利夫 教授

2014年

三重大学 工学部 情報工学科
計算機アーキテクチャ研究室

槇坂 知之 (410851)

内容梗概

4K テレビの登場，スーパーハイビジョン本放送の 2020 年開始予定など，近年，高精細化が着実に進む中で，動画像の圧縮符号化に必要な演算量が大幅に増加している．特に，スーパーハイビジョンでは，その動き検出処理の演算量が工夫無しではハイビジョンに比べ最大 128 倍にも増加し，ハードウェア規模が著しく増大する．このため，スーパーハイビジョンでの実時間処理への対応と，高精度・高効率の両立には，H.264/AVC 用動き検出器の一次探索ユニットを見直し演算量を低減させることにより，動き検出器のハードウェア規模をさらに低減させる必要がある．

そこで本研究では，探索範囲の中心は密な探索を行い，周辺部は粗く探索点数を減らして探索する粗密探索を提案し，探索精度・圧縮効率の低下を防ぎつつ，演算量を最大 64%低減できることを示した．さらに，探索点数を低減して粗密探索していくための一次探索部のハードウェア構成も示した．

Abstract

The arrival of 4K television, the main broadcast of ultra high definition (UHD) television is scheduled in 2020, and the amount of the operation in compression encoding of the video increases drastically because the high definition of video image has made progress in recent years. Especially, in UHD television, the amount of the operation of its first-stage search in a coarse-fine hierarchical motion estimator increases by 128 times more than the high-definition television. Therefore, the author has aimed at decreasing the hardware amount of first-stage search engine and the amount of the operation is reduced, by improving first-stage search engine unit of motion estimator for H.264/AVC that search performance and compression efficiency is corresponding to real time UHD encoding.

Then, its my study, the center area of the search range was performed fine search and the neighboring area was performed coarse search which reduces search points. As a result, it was shown that the amount of the operation can be decreased 64% with little deterioration of search accuracy and compression efficiency. Furthermore, it was shown that hardware composition of first-stage search engine of coarse-fine search too.

目 次

図 目 次

表 目 次

1　まえがき

1.1　背景

近年，デジタルカメラやテレビなどの電子機器で動画像の高精細化が進んでいる．また，ハイビジョン映像の符号化ユニットにおいて演算量，あるいはハードウェア規模の低減が進んでおり，現在では家庭用のデジタルカメラやスマートフォンに搭載されるほど小型化されている．

動画像の圧縮規格はH.264/AVCが主流となっているが，一方でその2倍の圧縮性能を持つ次世代の動画像における圧縮規格であるH.265/HEVCの研究も進んでいる．現在ではその圧縮規格が用いられているスーパーハイビジョン放送の研究も進んでおり，2020年に本放送の開始が予定されている．スーパーハイビジョンは，現在のハイビジョンと比べ，画素数が16倍になった超高精細映像と22.2chの3次元音響に基づく高臨場感システムである．また，スーパーハイビジョンでの演算量の大半を占める動き探索の演算量は工夫無しでは最大128倍にまで増加する．

したがって，既存の構成法で十分な探索範囲を確保しようとすると動き検出部がハードウェア規模増大の主因となる可能性が高い．

1.2 研究目的

この状況打開のため当研究室では，拡張テンプレートを複数併用して探索精度の低下を防ぐ階層探索型の動き検出アルゴリズムを開発してきた．また，このアルゴリズムを利用して演算量を大幅に低減しつつ，全探索並の精度を確保する動き検出器のハードウェア構成も検討してきた．

しかし，全体の約 7 割の演算量を占める階層探索における一次探索部のハードウェア規模は，十分低減できていない．そこで，本研究では，高精度・高効率を両立する H.264/AVC 用動き検出器の一次探索ユニットを見直すことにより，演算量を低減させて，一次探索部のハードウェア規模をさらに低減させることを目指す．

2　スーパーハイビジョン符号化の要求条件

2.1　概要

2020 年に本放送の開始を予定しているスーパーハイビジョンとは，図 2.1 よりハイビジョンの 16 倍の画素数であり，その構成画素数は縦横 4 倍の 7680x4320 で約 3300 万画素にも及ぶ「超高精細映像システム」である. フレームレートも 60fps とハイビジョンの 2 倍としているため，16x16 画素サイズのマクロブロックの数は 7680x4320/16x16=129600 にもなる．1 マクロブロック当たりの処理時間はフレームレートが 60fps なので 1/60/129600 =129ns しかない．つまり，仮に 1GHz で動作させたとして 1 マクロブロック当たり 129 サイクルしかかけられない．したがって，スーパーハイビジョン化のためにはマクロブロックの並列処理が不可欠である.

図 2.1: スーパーハイビジョンと従来の解像度との比較

2.2 符号化の演算量

演算量に関しては，画素数の増加分が 16 倍，フレーム数の増加分が 2 倍でハイビジョンの 32 倍になる．また，スーパーハイビジョンでは画像の解像度が縦横それぞれ 4 倍になるが，フレーム数がハイビジョンの 2 倍であり，フレーム間距離は 1/2 倍となる．探索範囲は縦横ともに 1/2 で良いため，探索範囲の増加分は 4 倍で済む. したがって，動き検出については，さらに探索範囲の増大分 4 倍がかかり，工夫無しでは演算量はハイビジョンの 128 倍にもなる. この演算量の増加がハードウェア規模増加の主因になる可能性が高く，実時間処理を行うためには，演算量を低減しなければならない.

この式に ITE 標準映像である「Horse Race」「Whale Show」「Soccer」の三種類のハイビジョン映像のワーストフレーム平均値 (worst fl ave) の結果を用いてスーパーハイビジョンの一秒あたりの演算量を算出した．その中では「Horce Race」の演算量が最も多く，それを実時間処理するためには動作周波数 1GHz で処理したとしても，一次探索ユニットが並列で 36 個必要になる.

この結果より，演算量を低減してユニットの並列度を少なくすることがハードウェア規模の低減につながるといえる.

3　従来の技術とその問題点

3.1　動き探索

動き探索とはフレーム間でマクロブロックごとにブロックマッチングを行い，参照画像と符号化対象画像で最も似ている位置を検出することである．この位置を検出するために参照画像と符号化対象画像において，1つずつの画素値の差を求める．そして，差分絶対値和（SAD）が最小の部分を検出し，そのブロックまでの変位量を動きベクトルとする．下記の式 (1) に符号化対象画像を X，参照画像を Y としたときの N × N ブロックごとの画素値の差分を示す．なお，(u,v) はブロックの変移量を表している．

$$SAD(X,Y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |X(i,j) - Y(i+u, j+v)| \tag{1}$$

このブロックマッチングは式 (1) からも分かるように，1画素ずつずらして探索範囲内の全てのブロックとのマッチングを行う全探索法である．この手法は，高い探索精度は誇るものの，探索点数の多さから演算量が非常に膨大となってしまう．したがって，これまで動き検出処理には様々な演算量低減手法が提案されてきた．

これまでに当研究室では，探索対象画像の画素密度を段階的に高める粗密型の「階層探索法」，さらにその階層探索の一次探索部での動き探索における精度向上が可能な「拡張テンプレート複数併用法」を提案し，全探索に比べ画質を損わず演算量を 1/100 にまで低減した一次探索ユニットの設計に成功している．

3.2 階層探索法

階層探索とは解像度を低くした縮小画像を用いて探索を行う．図 3.2 に示すように，最も解像度を低くして縦横それぞれ 1/4 に縮小した 4 画素精度画像でまず一次探索を行い，差分絶対値和の大まか位置を見つける．そして，その位置を探索中心として精度を上げた縦横それぞれ 1/2 に縮小した 2 画素精度画像と元の画像である単画素精度画像でそれぞれ二次探索と三次探索を行う．この階層探索により，演算量は 1 回のマッチングにより通常の 1/16 まで抑えることができ，さらに探索範囲も 1/16 になり全探索と比較すると 1/256 程度の演算量で済む．したがって，縮小して探索し大まかな位置を知ることで，無駄な探索をせずに済み，演算量低減につながる．しかし，元の画像を用いて行う単画素精度の探索に比べて 4 画素精度探索では，画素数が 1/16 になるため情報量が欠落するこ

とで探索精度の低下が問題となる.

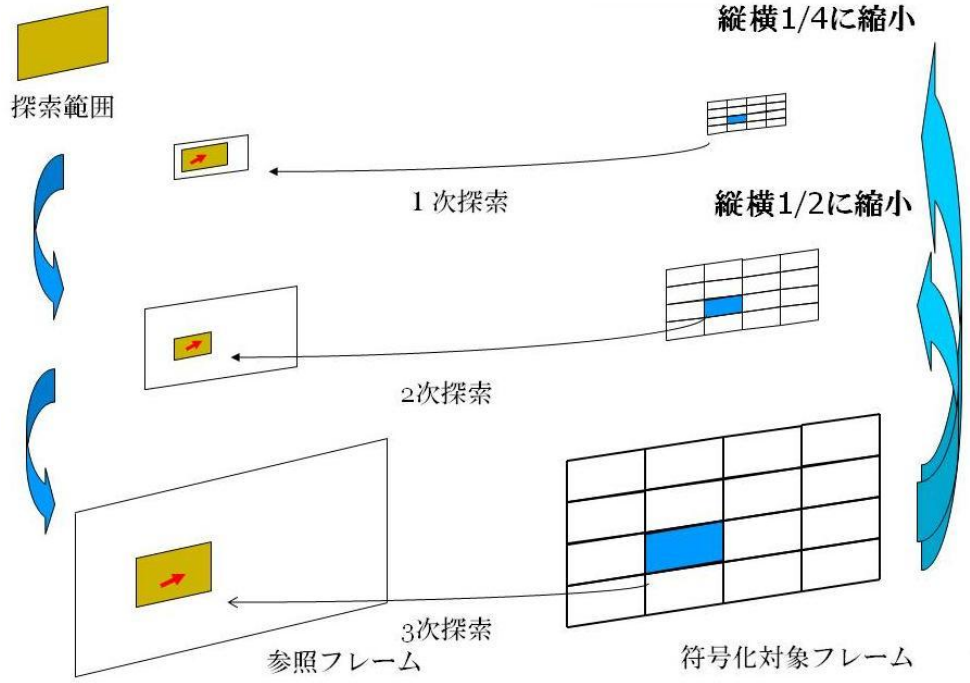

図 3.2: 階層探索

3.3　拡張テンプレート複数併用法

階層探索を用いると，画像の細かい変化を識別できずに探索精度が低下してしまう問題があった．そこで，当研究室では階層探索で解像度の最も低い一次探索に拡張テンプレート複数併用法を用いている．これにより図3.3に示すように，符号化対象ブロックの隣接ブロックを利用して拡張テンプレート単位でマッチングを行うことで，いずれかの方向の隣接ブロックが同じ動きをしていれば，その動きのそろう方向の拡張テンプレートにより正しい動きが検出できる．したがって，構成画素数が増えるものの，誤検出に至る可能性を減らすことができ，探索精度が向上される．当研究室では図3.3のように最大2 × 2の複数の組み合わせ方で拡張して，それらを用いて一次探索の精度を向上させる2 × 2サイズの拡張テンプレート複数併用法を提案している．

しかし，通常のブロックに加えて図3.3の8つのブロックも探索する必要がある．また，全ての拡張テンプレートの計算を行うと通常の9倍のSAD演算が必要となり演算量が増加してしまう．

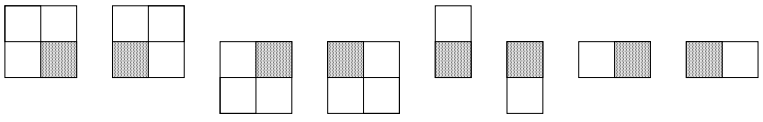

図 3.3: 拡張テンプレートの形状

3.3.1 スーパーテンプレート単位での一括探索

拡張テンプレート複数併用法を用いると演算量が増大する問題点があった．そこで，互いに共有するテンプレート構成ブロックのSAD値を再利用することで演算量の増加を抑止する．具体的には，拡張テンプレートでの演算量の増大を複数のマクロブロックを合わせたスーパーテンプレート単位で一括して探索することにより演算量を低減させる．今回のハードウェア設計では，図3.4に示すように，縦横5×6 (30) 個の4×4サイズのマクロブロック（スーパーマクロブロック）の範囲で同時に探索を行うことで，並列に演算することが可能となる．

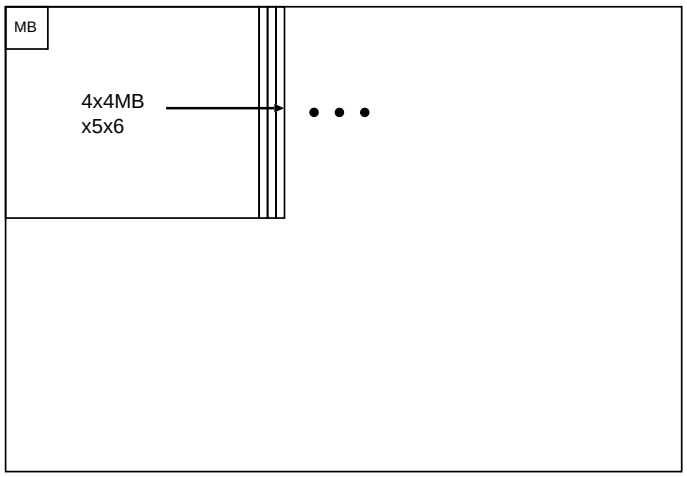

図 3.4: スーパーマクロブロックの形状

3.4 問題点

当研究室では，上記の「階層探索法」「拡張テンプレート複数併用法」といった手法により,「全探索法」に比べ画質の低下を抑えて，演算量を1/100 まで低減することに成功した．しかし，スーパーハイビジョンの演算量を想定した場合，実時間処理を行うためには，さらなる性能の向上が必要である．スーパーハイビジョン化に向けて，演算量をさらに低減させて，ハードウェア規模を低減させる必要がある．

4　提案手法

4.1　ハードウェア上でのPEアレイによる探索

差分絶対値を計算するブロックマッチング用の演算器配列（PEアレイ）を用いて探索器を構成する．一次探索部では，拡張テンプレートでの演算量の増大に対して，複数のマクロブロックを合わせて，互いに共有するテンプレート構成ブロックのSAD値を再利用するスーパーテンプレートによる探索を行う. 現在の設計では，図4.5に示すような1ブロック当たり4×4画素のPEアレイを6×5（30）個並べており，30マクロブロック（480画素）を同時に演算することのできる構成となっている．PEアレイを並列に設けることで差分絶対値和を並列に求めることが可能になる．なお，上下左右へのシフトは2×5（10）個並べられた4×4画素のシフトレジスタ，上下の探索の折り返しの際には2×5（10）個並べられた4×4画素のレジスタを用いている．したがって，これらPEアレイ部で探索を行うためには下方向の画素である横1列40画素をバッファからPEアレイ部に送る必要がある．

そして，そのPEアレイによる探索では，図4.6に示すように，探索領域において左上から探索を開始する．そして，図4.7に示すように，横方向への往き8画素分，上下方向に1画素ずらしての帰り8画素分の探索

を上端から下端まで繰り返した後，右方向に 8 画素折り返して繰り返すジグザグ探索走査を繰り返すことで，探索領域全域の探索を可能とする方式を採る．ハードウェア上ではジグザグ走査により探索を行うことで，局所性が高まりメモリアクセスの効率化につながる．したがって，これらにより高並列演算と転送のオーバーヘッド低減の両立が可能となる．

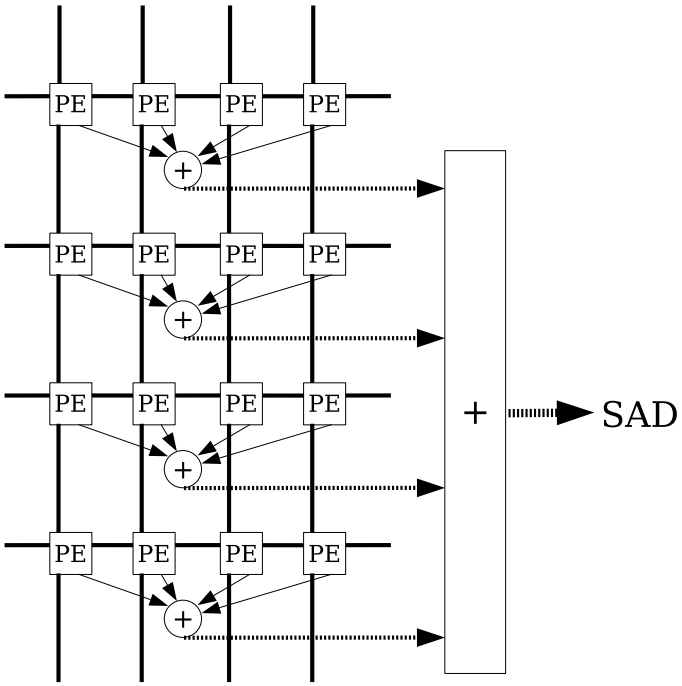

図 4.5: 4 × 4PE アレイ

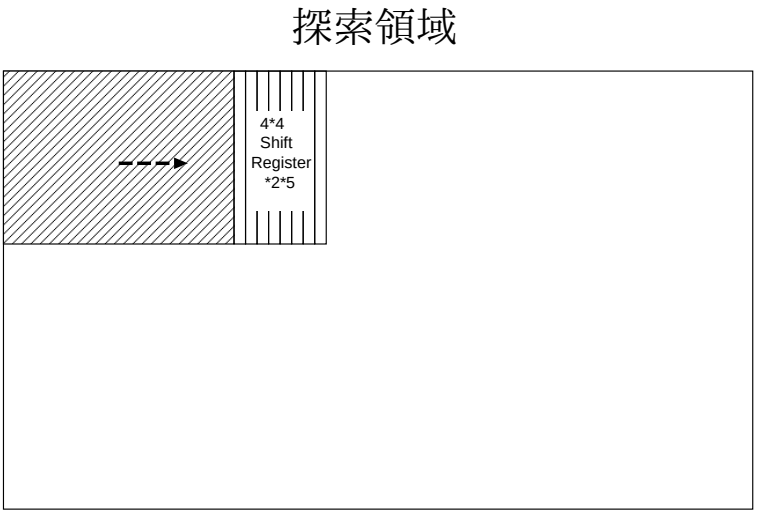

図 4.6: ジグザグ走査の開始地点

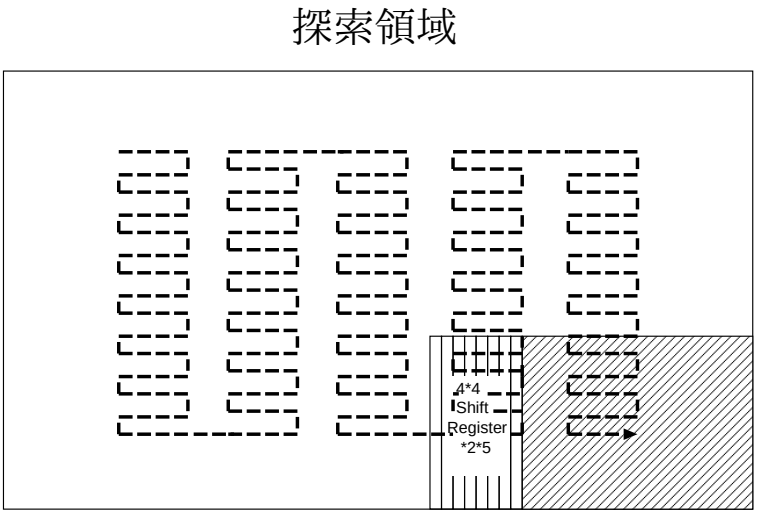

図 4.7: ジグザグ走査とその折り返し

4.2 演算量低減のための粗密探索

しかし，従来の探索方式では演算量がいまだに多く，スーパーハイビジョンに向けてさらなる演算量の低減が必要である．そこで，演算量の低減のために画素を飛ばしてシフトを行い，探索点数を減らして探索することを考える．通常 PE アレイは上下左右に 1 画素ずつシフトするが，シフトレジスタにより水平方向と垂直方向の画素のシフト量を変更する．

今回は上記の 1 画素ずつのシフトとは異なり，探索範囲全てにおいて水平方向の探索を 2 画素ずつ，垂直方向への探索も 2 画素ずつシフトを行うことで探索点数を低減する．表 4.1 では，最大の探索範囲± 128 (4 画素精度では± 32) でのステップ数を求めた．探索のステップ数とは別に

最初に画素の読み込みにかかるアイドル時間のステップ数は今回は考慮していない．表 4.1 より画素を飛ばして探索点数を低減して探索した場合は，ステップ数は約 72%低減される．

H.264 用のソフトウェアエンコーダである JM11.0 で HD 標準画像である Horse と Bronze を用いて条件はフレームレートを 30fps，フレーム数を 50, 最大の探索範囲を ± 128 にそれぞれ設定し，探索範囲全てでの探索点数低減を行った結果を表 4.2 に示す．表 4.2 より演算回数は最大約 72%低減されているが，最大で PSNR は 0.005db 低下し，BitRate は 1.94%上昇している．

したがって，探索範囲全てで探索点数を減らしてしまうと精度が低下してしまい，圧縮効率も悪くなってしまう．そこで，探索範囲の中心付近の候補ベクトル近傍は密に探索を行い，探索領域の周辺部は探索点数を減らして粗く探索を行う．そうすることで探索精度と圧縮効率の低下を防ぎつつ，演算量の低減をはかる．

	ステップ数
全探索	4608
1 画素飛ばし	1280

表 4.1: ステップ数の比較

	探索方式	演算回数	PSNR(db)	BitRate(kbps)
Horse	全探索	69760	33.537	11954.04
	縦横 1 画素飛ばし	22208	33.535	12185.54
Bronze	全探索	23680	33.068	26280.66
	縦横 1 画素飛ばし	6592	33.063	26400.53

表 4.2: 探索点数低減した場合との比較

4.3 実装上の課題と解決法

PE アレイによるジグザグ走査では，水平方向の往きあるいは帰りの 8 画素の探索に 8 サイクル必要である．探索点数低減のために水平方向に 2 画素ずつのシフトを行うと水平方向 8 画素の探索が 4 サイクルで完了する．しかし，現在の設計では，フレームメモリから探索領域バッファへは 8 画素ずつ読み込まれており，水平方向 1 列の探索に必要な 40 画素をバッファに格納するには 5 サイクル必要となり，水平方向の探索完了時間に間に合わず，アイドルが発生する．スーパーハイビジョンでの実時間処理を行うためにはアイドルなしでの探索を行う必要がある．そこで，フレームメモリからバッファへの読み込みを 10 画素ずつに変更し，水平方向 40 画素の読み込みが 4 サイクルで完了し，アイドル無しで探索を行えるようにする．

また，同様にさらに垂直方向にも 2 画素ずつシフトを行って探索点数

を低減することも検討する．垂直方向に 2 画素シフトする時には水平方向 2 ライン分である 80 画素を 4 サイクルでバッファに格納しておく必要がある．したがって，1 サイクルで 20 画素ずつのバッファへの読み込みが必要となる．そこで，メモリからバッファへの読み込みを 20 画素ずつに変更し，水平方向 80 画素の読み込みが 4 サイクルで完了し，アイドル無しの探索が可能になる．

5　設計

今回の提案手法である探索点数を減らして探索する場合，縦横 2 画素ずつシフトしていく時はフレームメモリから探索領域バッファに最低でも 20 画素ずつの読み込みをしなければ間に合わない．画素のシフト量を大きくすると，ビット線のアクセス幅の増加などによりハードウェア規模が増大すると考えられるので，最小限の 2 画素ずつのシフトを採用する．

外付けのフレームメモリのアクセス幅は，2 のべき乗である．このため 20 画素幅のままメモリのアドレスを管理しようとすると，最小限のアクセス幅が 16 画素になることにより，4 画素分の容量が無駄になる．そこで，フレームメモリのアクセス幅を 32 画素とする．そして，メモリから読み出す 32 画素幅の画素列から 20 画素幅で切り出して探索領域バッファへ読み出していく．

従来の探索領域バッファは 1 ライン分に必要な 40 画素だけで済んでいたが，今回の提案手法により，2 ライン分である 80 画素に拡張する．しかし，探索領域バッファへは PE アレイ部に必要な 80 画素しか格納でき

ない．そこで，図 5.8 に示すように，32 画素ずつフレームメモリから読み出し，フレームメモリと探索領域バッファの間に 32 画素から 20 画素の変換するためのバッファを挿入し，20 画素ずつの読み出しを可能にする．したがって，探索領域バッファへの格納が 4 サイクルで完了し，提案手法を実現することが可能となる．なお，図 5.8 の設計では 1 画素は 8bit としている．

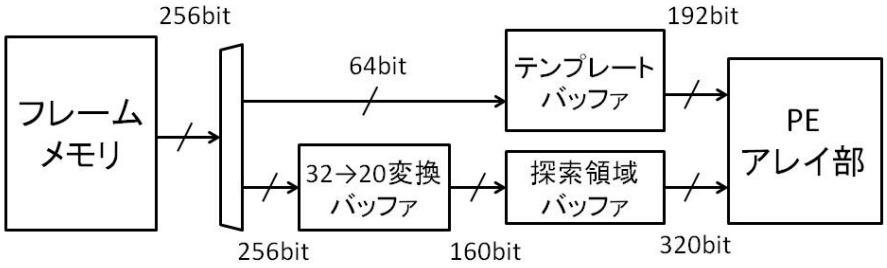

図 5.8: 一次探索部の構成

6　評価

6.1　評価方法

H.264 用のソフトウェアエンコーダである JM11.0 に今回の提案手法を実装した. また，評価画像には HD 標準画像の Horse と Bronze を用いて，フレームレートは 30fps，フレーム数は 50, 最大の探索範囲は± 128 にそれぞれ設定した.

評価には，マクロブロック当たりの一次探索部の差分絶対値演算回数，PSNR(db)，BitRate(kbps) を用いた．さらに，ハードウェアに合わせて縦横 1 画素飛ばしで探索点数を減らす範囲を水平方向だけ変更していき，表 6.3 の条件のもと評価を取った.

case	探索点数低減	探索点数低減を適用する範囲
A	なし	なし
B	あり	水平方向± 128〜64
C	あり	水平方向± 128〜32
D	あり	水平方向± 128〜 8

表 6.3: 探索条件

6.2 評価結果

上記の表 6.3 の条件のもと，マクロブロック当たりの一次探索部の差分絶対値演算回数，PSNR(db)，BitRate(kbps) で評価を行った結果を下記の表 6.4 に示す.

	case	演算回数	PSNR(db)	BitRate(kbps)
Horse	A	69760	33.537	11954.04
	B	52032	33.534	12043.11
	C	38272	33.533	12038.45
	D	25280	33.533	12070.52
Bronze	A	23680	33.068	26280.66
	B	23168	33.068	26249.84
	C	18176	33.067	26291.30
	D	9728	33.065	26387.53

表 6.4: JM による評価結果

6.3 考察

今回の提案手法では，縦横 1 画素飛ばしで探索点数を減らす周辺部の範囲を変えていき評価を行った．ハードウェア上では水平方向は 8 刻みのジグザグ走査なので，水平方向は 8 (単画素精度では 32) 区切りで範囲の変更を行う．B の± 64 (4 画素精度では± 16) までは PSNR，BitRate ともにほとんど変化がない．しかし，演算回数は最大でも 25%しか低減できていない．C の± 32 (4 画素精度では± 8) まで粗く探索した場合は最大で，演算回数の低減は 45%まで増え，PSNR の低下を 0.004db，BitRate の上昇は 0.71%となった．

そこで，さらに探索点数低減の範囲を広くし演算回数の低減を試みる．ハードウェアは 8 区切りではあるが，1 画素飛ばしに対応できる最小の± 2 (単画素精度では± 8) の範囲まで変更を行う．適用した D では，最大で PSNR の低下を 0.004db,BitRate の上昇を 0.97%で 1%以内に抑えつつ，演算回数を最大 64%低減させることができた．しかし，その場合のハードウェア構成は，水平方向の探索の途中で粗い探索から密な探索へ切り替えなければならないために，制御が複雑になる可能性があるため，設計する際には工夫が必要である．

7　あとがき

本研究では，探索範囲内の周辺部の探索密度を粗くすることにより，探索精度と圧縮率の低下を抑えながら演算量を低減できることと，この探索法に対応する一次探索部の構成と動作法を示した．この演算量低減手法により，一次探索ユニットの探索時間が半分以下に低減されることから，同一の探索能力を得るのに必要なハードウェア規模を半分以下に低減することが可能になる．

しかし，まだハードウェアのことを考えて設計を行うには，粗い探索と密な探索を探索範囲内で切り替える際にレジスタやビット線の増加によるハードウェアの複雑化を防ぎ，簡潔な構成を考案する必要がある．また，ハードウェアの実装において，条件分岐命令を多く使用するとハードウェア規模増加に繋がってしまうために，数式を用いて探索範囲の切り替えを行うなど動作制御においてもさらなる工夫が必要である．

今後の課題として，スーパーハイビジョンに対応するためにはさらなる演算量の低減が望まれる．そのために，探索点数低減を適用する最適な範囲の調査を行い，さらなる演算量の低減と探索精度との両立を目指す．また，実際に論理設計を行い，ハードウェア規模を厳密に調べる必要もある．

謝辞

本研究を進めるにあたり，様々な御指導，御助言を頂きました近藤利夫教授，並びに多くの助言を頂きました佐々木敬泰助教授に深謝いたします．また，本研究の様々な局面にてお世話になり，日常の議論を通じて多くの知識や示唆を頂いた計算機アーキテクチャ研究室の皆様にも心より感謝いたします．

参考文献

[1] 近藤　利夫，渡部　謹二，佐々木　敬泰，大野　和彦：方向別拡張テンプレートを複数併用する粗密探索構成の動き検出法，電子情報通信学会論文誌，Vol.J95-D，No.1，pp.85-96，2012.

[2] 片野　智健，上念　三郎，佐々木　敬泰，大野　和彦，近藤　利夫：異形状拡張テンプレート併用型 4 画素ビット切り詰め探索ユニットの設計，電子情報通信学会技術研究報告．IDC，集積回路，Vol.107，No.382，pp.71-76，20071206.

A　実験手順

今回の粗密探索のシミュレーションは H.264 拡張 MEv2.8 のソフトウェアエンコーダ JM11.0 において行った．

mv.search.c の中にある関数 ExpTemp_search のコードの一部を条件に合わせて変更し，評価を行った．今回の探索点数低減の縦横 1 画素飛ばしのシミュレーションは配列 spiral_search_x[pos]（水平方向）と spiral_search_y[pos]（垂直方向）が共に偶数点に存在する（2 で除算した剰余が 0 の場合）条件の場合のみ探索を行うようにした．

また，探索点数低減の範囲の制限は，密に探索を行う範囲（探索中心）を spiral_search_x[pos] でもれなく探索できるよう数値で不等号を用いて示す．この関数においては探索範囲の数値は一次探索部のため 4 画素精度になっているので，単画素精度で考えると 4 倍しなければならない．さらに，探索範囲の数値には正負があるために注意が必要である．そのようにして，探索範囲を制限する数値を基本的には 4 画素精度で ± 8 ずつ変更して評価を行った．

したがって，下記のように条件文を作成してシミュレーションを行う．下記の例では ± 64（4 画素精度では ± 16）の範囲では密な探索を行い，それ以外の周辺部では縦横 1 画素飛ばしの探索を行っている．この条件文

にマッチした時のみ差分絶対値の計算を行う.

（例）

```
if(!(spiral_search_x[pos]%2)&&!(spiral_search_y[pos]%2)
||(spiral_search_x[pos]<=16&&spiral_search_x[pos]>=-16))
```

全探索の演算回数等の調査の際には，上記の例の条件文をコメントアウトして，下記のようなコードを追加した.

```
if(ex_search_range>1)
```

最大の探索範囲の設定は encoder.cfg ファイルの変数 SearchRange の値により変更できる．常に探索範囲は 1 より大きく設定していたので，これにより虱潰しに探索する全探索の場合の演算回数を求めることができる.

さらに，全ての探索範囲において探索点数低減の縦横 1 画素飛ばしのシミュレーションは，上記の例の 2 行目の範囲を制限するコードをコメントアウトして 1 行目のみで行った.

また，/home/share/makisaka/2014/grad_program にあるプログラムの grad_program には注釈を，readme には使い方等の説明を記述しているので参考にするとよい．