I-055

# GPU による高速 H.264 動き補償予測の性能改善
# Speed Improvement of GPU Accelerated H.264 Motion Compensated Prediction

鷹野 芙美代† 森吉 達治†
Fumiyo Takano Tatsuji Moriyoshi

## 1. はじめに

動画像符号化 H.264[1]は、その高い符号化効率から映像配信やビデオカメラ、ポータブルプレイヤ等で広く使用されている。しかし、H.264 の処理演算量は従来の MPEG-2 と比べて数倍以上と多く、さらに 1080/60p (1920x1080 画素/フル HD、progressive 60 fps)等のより高精細な映像の実時間処理のニーズに対応するため、処理の高速化が望まれている。そこで、実時間エンコーディングの実現手段の一つとして、グラフィック処理ハードウェアである GPU (Graphics Processing Unit)の利用が提案されている[2]。GPU を用いた高速処理を実現するためには、多数コアでの処理向けに並列性を高くすることや、CPU-GPU 間のデータ転送等のオーバヘッドを削減することが必要である。

我々は GPU による高速 H.264 エンコーダの研究開発を進めており、これまでにアルゴリズム並列性を高めることで 1080/30p の実時間エンコードを可能にする動き探索手法を開発した[2]。本稿では、さらに 1080/60p の実時間エンコードを実現するため、動き探索のみでなく周辺の関連処理も統合した動き補償予測全体を GPU を用いて高速処理する手法を提案する。

## 2. GPU による H.264 動き補償予測

H.264 エンコーダにおける GPU による高速化効果が大きいと予想される処理に動き補償予測処理がある。動き補償予測は、参照フレームからの動きを補正した予測画像と現フレームの差分のみを符号化することで符号量を削減する技術であり、図 1に示すように、動き探索、動き補償、小数画素補間処理から構成される。動き探索はフレーム間の動きを表す動きベクトルを探索する処理であり、動き補償は動きベクトルから予測画像を生成する処理である。H.264 でサポートされる 1/4 画素精度動き補償を行うためには、参照フレームの小数画素(1/2 画素・1/4 画素)補間が必要である[1]。

本 GPU 動き補償予測の開発には NVIDIA 社の CUDA (Compute Unified Device Architecture)[4]を用いた。CUDA による GPU 処理では、GPU 演算処理そのものの他に CPU-GPU 間データ転送やそれらを制御する API の実行が必要であり、これらによるオーバヘッドが発生する。

本稿では、まず動き補償予測全体を GPU 並列化し、各処理における処理時間を解析した結果について述べる。そして解析結果から、小数画素補間処理と GPU の制御オーバヘッドが動き探索時間に対して大きいことが判明したため、これらの改善手法を提案する。

### 2.1 動き補償予測高速化の課題

60 fps 実時間エンコードを実現する方法の一つは、エンコード時間の半分を占める動き補償予測の処理時間を 8 ms/frame 以下とすることである。そのため、今回はまず、従来の 3 種(16x16, 16x8, 8x16)のブロックサイズを探索する動き探索[2] (quality モード）をベースとして、以前の報告では未着手だった動き補償・小数画素補間を GPU 並列化した。図 2 に示すように quality モードの処理時間(表 2 の 7 画像の平均)では、処理時間の多くは動き探索が占めることがわかる。そこで、動き探索における探索ブロックサイズを 3 種から 1 種(16x16)のみに限定することで演算量の削減を試行した(fast モード)。しかし、図 2 より fast モードは、quality モードと比べて動き探索の処理時間は半分に減少したが動き補償予測全体では 30%しか短縮されず、60 fps 処理に必要な 8 ms/frame 以下とはならない。よって、高速な動き補償予測処理には動き探索の高速化だけでは不十分であり、小数画素補間や GPU 制御(画像データ以外の制御用データ転送や GPU の処理関数であるカーネル起動 API 発行等のオーバヘッド）の高速化も必要だと考えられる。そこで、以降ではこれらの高速化手法について述べる。

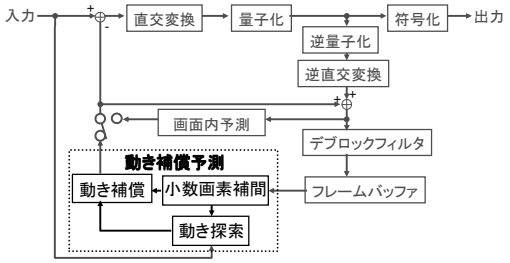


図 1 H.264 エンコーダのブロック図

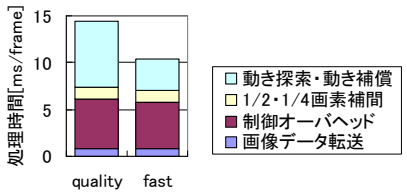


図 2 GPU 動き補償予測の処理時間内訳

表 1 整数画素あたりの小数画素利用率

| | 平均参照回数 | 事前補間処理回数 |
|---|---|---|
| 1/2 画素 | 16.1 | 3 |
| 1/4 画素 | 4.6 | 12 |

### 2.2 小数画素補間演算量の削減

小数画素は動き探索・ 動き補償で繰り返し参照される。そのため、H.264 の参照エンコーダである JM[3]ではフレーム全体の 1/2 画素・1/4 画素補間画像を事前に生成することで補間処理の低減化を図っている。しかし、この事前補間の効果は補間処理回数に対する参照回数の割合である利用率に依存するため、本提案手法における動き探索の小数画素利用率に基づいて事前補間の有効性を検討した。

† 日本電気株式会社 グリーンプラットフォーム研究所
Green Platform Research Laboratories, NEC Corporation

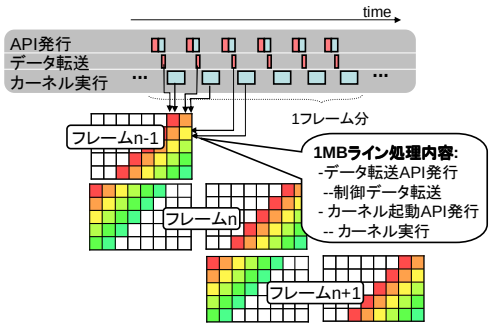


図 3　従来の処理 MB 制御

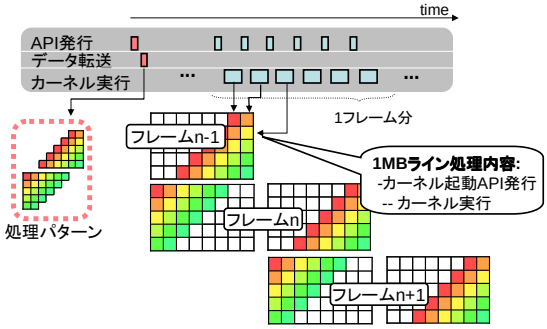


図 4　処理 MB 制御の最適化

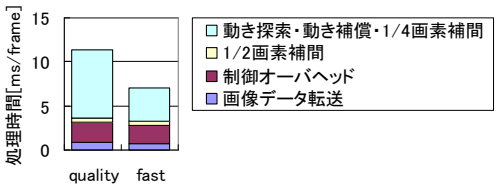


図 5　最適化後の処理時間

本動き探索では、探索アルゴリズムとして演算量対符号化効率の良い EPZS (Enhanced Predictive Zonal Search) [5]を用い、整数画素→1/2 画素→1/4 画素の順でそれぞれ直前の探索結果の近傍のみを探索する。表 1 に本動き探索における小数画素の参照回数(小数画素精度の動き探索点数から決定)と補間処理回数(整数画素あたりの小数画素数)を示す。表 1 に示すように、1/2 画素は事前補間処理回数に対して参照回数が多く、事前補間の効果が高い。一方で、1/4 画素は事前補間処理回数に対して参照回数が少なく、使用する画素のみを必要に応じて補間するのが効率的と考えられる。以上より、本動き補償予測では 1/2 画素のみを事前補間し、1/4 画素は使用時に補間することとした。

### 2.3 GPU 制御オーバヘッドの削減

本 GPU 動き補償予測では、符号化効率を維持しつつ並列性を向上するマクロブロック（MB）並列処理手法[2]を用いており、そのために GPU 制御のオーバヘッドが大きくなる。この手法では、図 3 の同色 MB のような複数のフレームへまたがる斜めに隣接した MB 群を 1 カーネルで並列処理し、フレーム全体の処理が完了するまで繰り返しカーネルを呼び出す。図 3 は 2 フレームを並列に処理する例で、フレーム n-1 の後半とフレーム n の前半の同色の MB が並列に処理される。このような処理では、カーネル毎に不規則な位置のフレーム・MB を処理するため、一般的な行列計算の GPU 処理のようにスレッド ID のみから処理データ位置を決定するのは難しい。そのため CPU からの制御が必要となり、制御オーバヘッドが大きくなる。この制御では、GPU スレッドの処理 MB 位置・フレーム番号等が書かれた制御データテーブルを CPU で作成してカーネル呼び出し毎に GPU へ転送し、GPU スレッドは制御テーブルから処理データ位置を決定する。この処理は図 3 のように並列に処理される MB 群である MB ラインごとにカーネル処理とデータ転送、それらに伴う GPU API 発行が必要であるため、多数の MB ラインを実行することによりオーバヘッドが大きくなる。

そこで、処理に必要な全パターンの制御データをまとめて GPU へ転送するように制御方法を変更することで、制御データ転送量・API 発行回数を削減した。図 4 のように、フレーム n の処理が開始されてからフレーム n+1 の処理が開始されるまでの処理パターンは、他のフレームでもフレーム番号が異なるだけで MB 位置は同じである。この性質を利用して、処理パターンを最初に転送しておき、処理フレーム番号はフレームの先頭でのみ更新することとした。これにより各カーネル呼び出し時には処理パターンの何ライン目かという情報のみをカーネル引数で指定すれば処理 MB 情報を決定可能になり、制御データ転送時間とその API 発行のオーバヘッドを削減できる。

表 2　評価環境

| | |
|---|---|
| GPU | NVIDIA GeForce GTX 580: 512 core, 1.54 GHz |
| CPU | Intel Core i7: 4 core, 2.67GHz |
| エンコードパラメータ | Baseline Profile, fixedQP 28, N=30<br>1 reference frame, RD Optimization OFF |
| 評価画像 | 1080p (Blue_sky, Pedestrian_area, Riverbed, Rush_hour, Station2, Sunflower, Tractor) |

## 3. 評価

表 2 の評価環境での処理時間を図 5 に示す。図 5 と図 2 の fast モードを比較すると、小数画素補間と動き探索の合計時間は 0.5ms/frame、制御オーバヘッドは 3.0 ms/frame 削減し、全体の処理時間は 28%削減した。fast モードの処理時間は、従来手法である図 2 の quality モードと比較すると 2.0 倍の高速化である 7.0 ms/frame となり、60 fps 実時間エンコードが十分実現可能と考えられる。

## 4. おわりに

本稿では、動き補償予測の GPU 高速化手法を提案した。本手法では、動き探索の演算量削減に加え、小数画素の参照回数を基に補間演算量を削減し、さらに GPU 制御を最適化することで制御オーバヘッドであるデータ転送と API 発行を削減した。その結果、従来手法[2]より 2.0 倍高速化して処理時間は 7.0 ms/frame となり、1080/60p の実時間エンコードが可能となった。

### 参考文献

[1]ITU-T Recommendation H.264 “Advanced video coding for generic audiovisual services,” 2003.
[2]鷹野, 森吉, “マクロブロック間依存制約緩和による GPU H.264 動き推定の高速化”, FIT2011, 2011.
[3]A. M Tourapis, K. Sühring, G. Sullivan, “H.264/MPEG-4 AVC Reference Software Manual” , ITU-T SG16 Q.6, 2007.
[4]NVIDIA , “CUDA Programing Guide” , 2012.
[5]A. M. Tourapis, “Enhanced predictive zonal search for single and multiple frame motion estimation,” Proc.VCIP, 2002, pp. 1069–1079