3B-2

Cubic Convolution フィルタの Cell/SPE 向け最適化実装

高橋 幸恵† 今田 敬† 高山 征大† 山下 道生† 境 隆二†
株式会社 東芝 コアテクノロジーセンター‡

1．はじめに

一般的に画像処理は計算量が多く、Cell Broadband Engine(以下 Cell)のような高性能プロセッサにとって、好適なアプリケーションの１つである。Cell は SPE と呼ぶ SIMD プロセッサを複数保持するが、その性能を最大限に引き出す為には、SPE に最適化したソフトウェアの実装が重要となる。

本稿では、画像処理の１つである Cubic Convolution フィルタについて、Cell における実装とその性能測定結果について述べる。

2．Cubic Convolution フィルタ

Cubic Convolution フィルタは、別名 Bi-Cubic とも呼ばれ、３次補間を用いた拡大・縮小フィルタであり、周辺の 16 近傍の画素を利用する[3]。例えば、拡大においては下図のような周囲 16 画素(低解像度画像)の加重平均が、高解像度画像の輝度値となる。Nearest Neighbor、Bi-Linear 法に比べ、一般に画質は良いが、処理負荷が高い。

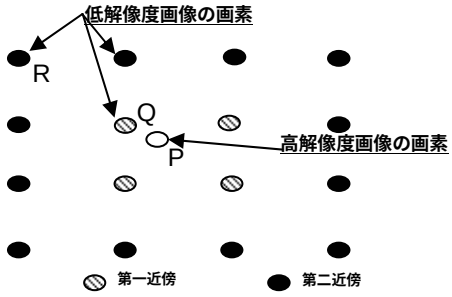

図１）Cubic Convolution フィルタで用いる画素の様子

HD1 画素を求めるために、以下の式を用いる。

□Cubic Convolution 一般式□

$$\sum_{y=0}^{3}\sum_{x=0}^{3} W(x,y)\times f(x,y)$$

f(x,y):低解像度画像の座標、W(x,y):(x,y)の重み

□行列の乗算に変形□

$$\begin{pmatrix} W_{x1} & W_{x2} & W_{x3} & W_{x4} \end{pmatrix} \begin{pmatrix} f(0,0) & f(0,1) & f(0,2) & f(0,3) \\ f(1,0) & f(1,1) & f(1,2) & f(1,3) \\ f(2,0) & f(2,1) & f(2,2) & f(2,3) \\ f(3,0) & f(3,1) & f(3,2) & f(3,3) \end{pmatrix} \begin{pmatrix} W_{y1} \\ W_{y2} \\ W_{y3} \\ W_{y4} \end{pmatrix}$$

＝1画素(高解像度画像)

□重みの計算について□

高解像度画像と低解像度画像の画素の距離を d とする。ここで d は、低解像度画像の画素間の X 方向、あるいは Y 方向の距離を 1 とする値をとる。例えば図 1 において、点 P、点 Q の X 方向の距離を d＝dx とすると、点 P、点 R の距離は d＝1＋dx となる。

・第一近傍 W＝(d－1)×(d×d－d－1)
・第二近傍 W＝－(d－1)×(d－2)×(d－2)

X,Y 座標共に同じ計算式を用い、Wx、Wy を求める。
最終的な高解像度画像に対する重みは、

W=Wx×Wy

から得られる。

3．Cell/SPE の特徴

Cell は、1 個の PPE と 8 個の SPE からなるマルチコアプロセッサである。画像処理のような計算量の多いアプリケーションは、SPE 向けに最適化実装することにより、効率的に動作させることができる。

SPE は主に、SPU と呼ぶ SIMD プロセッサ、SPU のコードと SPU が直接ロード/ストア可能なデータを保持する LS(Local Storage)、LS とメインメモリの間でデータを DMA 転送する MFC から成る。SPU が直接参照可能な LS は、256KB とサイズに限りがあるため、メインメモリと LS の間で適宜データを DMA 転送する必要がある。

また SPE は、非対称な２本のパイプライン(even/odd pipeline)を持っている。各命令はどちらか一方のパイプラインへ発行でき、命令にレジスタ間の依存関係がなければ、2 命令同時発行が可能である。

4．フィルタの設計・最適化

4-1 処理の流れ

DMA を除いた処理の流れは、以下の通りである。

① SD 画素のロード
SD の画素をロードし、レジスタデータに格納
② Cubic Convolution 手法を用いて演算
X/Y 方向の係数と SD の画素の演算
③ Clipping 処理
生成された HD 画素の値が 0 以上 255 以下になるように調整
④ HD 画素のストア
生成された HD 画素を、用意したアドレスへストア

今回は、Cubic Convolution フィルタを用いて、SD サイズ(720×480)から HD サイズ(1920×1080)への拡大を行う（横方向に 8/3 倍、縦方向に 9/4 倍拡大する）が、演算時には第一、第二近傍の画素の参照パターンと重み係数が、HD 画素で横 8 画素、縦 9 ライン毎に繰り返される。この性質を活かし、1 回で処理する HD 画素を横方向に 8 画素、縦方向に 9 ラインとして設計を行う。

また、縦方向のライン数を少なくすると、使用レジスタ数が少なくなり、レジスタの再利用回避による性能向上が期待できる。そこで、縦方向のライン数を 2/3/5 と変化さ

An optimized implementation of cubic convolution scaling filter for Cell/SPE
†Yukie Takahashi, Kei Imada, Motohiro Takayama, Michio Yamashita, Ryuji Sakai
‡ Core Technology Center, TOSHIBA Corporation

せた場合についても評価を行い、その効果を検証する。
更に、SPE 向けに、4-2 から 4-4 に記述したような最適化を行う。

4-2 SIMD 向け実装

SPE の SIMD 機能を活かすため、例えば図 2 のように、行列の乗算を SIMD 演算を用いながら実装する。途中の画素値の計算は、全て float 型で行っているため、４つの要素を１度に演算することができる。計算部分のみならず、SD 画素のロード、HD 画素のストアをする箇所も全て SIMD を用いる。

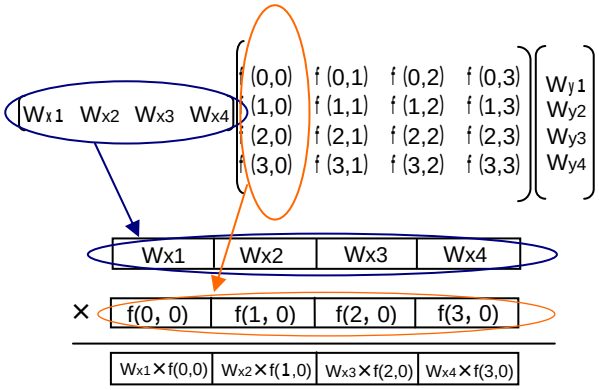

図２）SIMD 演算の様子

4-3 命令数の削減

今回は、「Y 方向への演算→X 方向への演算」の順番で Cubic Convolution の処理を行う。これにより、逆の順序で処理した場合に比べて、演算結果は同じになるが、SD 画素のロードをする際に、shuffle(*1)する回数が少なくできる。

(*1)shuffle； d = spu_shuffle(a,b,patten);
ベクタデータの要素を入れ替える演算。
ベクタ a 又は b から、ベクタ pattern の指定に従ってバイトを取り出し、それを要素として持つ新しいベクタ d を生成する[1]。

4-4 命令同時発行による高速化

4-1 で記した処理の流れにおいて、①と④の処理では odd pipeline、②と③では even pipeline で発行する命令が多い。そこで、①と③のループの繰り返しをオーバーラップさせ、レジスタ参照に依存関係のない命令を、並列に実行させることができる。すなわち、X 方向に画素を参照するループにおいて、1 ループ分ずれた①と③を並列に実行する。これにより、パイプラインを同時に利用することができ、2 命令同時発行の割合を、増加させることができる。

５．評価と考察

前項の設計最適化により、当初 1 フレームあたり 318[ms]かかっていた処理を、大幅に短縮することができた。例えば、1 回で処理する画素数を 8×9=72 とした場合には、1 フレームあたりの処理時間が、3.8[ms]に減少した。
また 4-1 で述べたように、1 回で処理する縦方向のライン数を変化させた場合に、性能への影響があることから、ライン数を 2/3/5/9 と変化させた場合について、評価を行った。表 1 にその結果を示す。

1 回に処理する画素数 [pixel] (縦×横)	実行時間 [ms/Frame]	命令数	サイクル数	1 画素あたりのサイクル数
2×8＝16	5.9	248	147	9.1
3×8＝24	4.5	308	171	7.1
5×8＝40	4.5	496	275	6.9
9×8＝72	3.8	772	420	5.8

表１）１フレーム作成するのにかかる実行時間、命令数、サイクル数と 1 画素あたりのサイクル数（Y 成分）

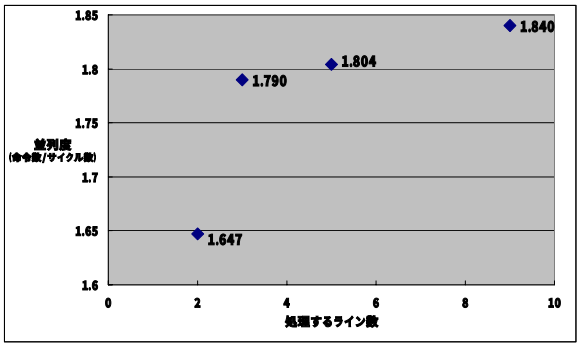

図３）処理するライン数と並列度の関係

表１の結果より、1 回に処理するライン数が 9 の場合に、最も実行時間が小さい。1 回に処理するライン数が 9 の場合に、レジスタの再利用による性能低下が懸念されたが、その影響は小さいといえる。また処理ライン数が 3 と 5 の時に、実行時間に違いが生じていない。5 ラインずつ処理した場合には、9 ラインの繰り返しに対して、2 回の処理で 1 ラインずつ余分に処理するため、処理のオーバーヘッドが大きくなる。同様な現象は、2 ラインずつ処理する時にも起きていることから、1 回に処理するライン数を 9 の約数にする、ということも非常に重要な条件となることが分かる。
また、ライン数ごとの並列度について求めた値を、図３にまとめた。これにより、処理ライン数の増加に伴い、並列度が向上していることが分かる。1 回に処理するライン数を増やすことにより、ループ内の命令数が増え、コンパイラによる最適化が、より効果的に働いたと思われる。
以上の評価結果により、SPE 向けに Cubic Convolution を実装する場合、1 回に処理する画素数を縦方向に 9 ラインずつにすると、効率良く処理できることがわかった。

６．おわりに

本稿では、Cubic Convolution フィルタを、Cell/SPE 向けに実装し、性能評価を行った。その結果、パイプラインのストールを最小限に抑えた実装最適化を実現できた。

参考文献

[1]http://cell.scei.co.jp/j_download.html
・CBE_Architecture_v10_j.pdf
・CBE_Public_Registers_v11.pdf
・SPU_assembly_language_v13_j.pdf
・SPU_language_extensions_v21_j.pdf
[2]http://www128.ibm.com/developerworks/power/cell/docs_documentation.html
Cell Broadband Engine programming handbook
[3]http://mikilab.doshisha.ac.jp/dia/monthly/monthly01/20011222/personal_kawasaki.pdf
[4]http://www.cc.tohoku.ac.jp/refer/pdf_data/v34-3p19-26.pdf