二次元 FFT の並列化と画像相関処理への適用性

1E－7

水野 政治† 芦沢 賢‡ 中島 克人†

三菱電機 (株) †情報技術総合研究所, ‡通信機製作所

1 はじめに

多くの信号処理, 画像処理の分野において, 二次元 FFT(Fast Fourier Transform: 高速フーリエ変換) は頻繁に利用されている. この二次元 FFT では, いわゆるメモリアクセスにおけるコーナターンが発生するため, 並列処理により高速化を図る場合, プロセッサ間で大量のデータ転送が生じ, システムの性能低下を招くという問題が指摘されてきた[1]. 筆者らは, このコーナターンの問題を軽減する, 二次元 FFT の並列処理手法を提案している[2].

本稿では, 本手法の有効性を検証することを目的とし, 市販のマルチ DSP(Digital Signal Processor) ボード上で, 本手法を用いて画像相関処理を実現した場合の有効性と実装上の課題について報告する.

2 画像相関処理

今回の検討では, 二次元 FFT の典型的な応用例である画像相関処理を取り上げる. 画像相関処理は, 2 つの画像 (入力画像と相関フィルタ) の相関を計算するものであり, 例えば画像の識別 / 認識等に適用されている[3]. この画像相関処理は次式により定義される.

$$\text{相関係数} = \frac{(x - x_m) * (y - y_m)}{\sqrt{E_{xm} \times E_{ym}}} \qquad (1)$$

ここで, x は入力画像, x_m は入力画像のフィルタ領域内の平均値, E_{xm} は入力画像の不偏化 (平均が 0) 後のフィルタ領域内のエネルギー, y は相関フィルタ, y_m は相関フィルタの平均値, E_{ym} は相関フィルタのエネルギーである. なお, 相関フィルタは予めエネルギー正規化 ($E_{ym} = 1$) 及び不偏化 ($y_m = 0$) されているものとする. これを次の手順で計算する.

1. 相関フィルタと同一サイズで全ての要素の値が 1 である配列について, 二次元 FFT を行う.
2. 入力画像の同一要素同士の掛け算 (項別積) したものに対して, 二次元 FFT を行う.
3. 1の結果と 2の結果の項別積に対して, 二次元 IFFT を行う.
4. 入力画像について, 二次元 FFT を行う.
5. 1の結果と 4の結果の項別積に対して, 二次元 IFFT を行う.
6. 5の結果の同一要素同士の掛け算を行い, 相関フィルタの要素数で割る.
7. 3の結果から 6の結果をひき, 平方根を求める.
8. 相関フィルタについて, 二次元 FFT を行う.
9. 4の結果と 8の結果の項別積に対して, 二次元 IFFT を行う.
10. 9の結果を 7の結果で割る.

このように, 処理の大部分は二次元 FFT である.

3 システム構成

市販のマルチ DSP ボードとしては, Alex Computer Systems,Inc.[4] の SHARC6000 及び PAC80 を想定した. これは, Analog Device 社の ADSP-2106x SHARC プロセッサ (40MHz) を計 10 個搭載しており, それらの DSP 間は 40MByte/sec の通信路で接続されている (図 1). また, PAC80 は 2 枚接続することも可能である.

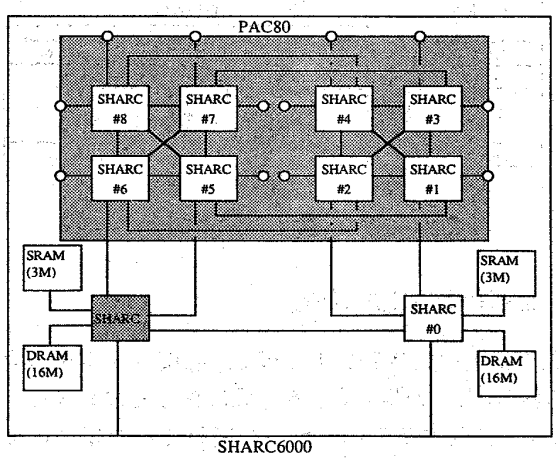

図 1: マルチ DSP ボードの概略構成

本検討においては, PAC80 上の 8 個, あるいは 16 個の DSP のみを処理に使用するものとした.

また, 各 DSP 上では, Wideband Computers,Inc. の ADSP-21K Optimized DSP Library[5] を使用することを想定した. このライブラリには, FFT をはじめ, 種々の関数が含まれいる.

Proposing a new scheme of Parallelizing 2D-FFT and its feasibility for the Image Correlation Processing
Masaji Mizuno, Satoshi Ashizawa, Katsuto Nakajima
Mitsubishi Electric Corporation

4 並列処理方式

項別積等の処理は, 各要素毎に計算が行なえる, すなわちデータ間の依存関係はなく独立に処理を行なえるので, ここでは特に二次元 FFT に着目して説明する.

二次元 FFT の並列処理は, 一般的な手法 (方式 1) と筆者らの提案する手法 (方式 2) との二方式について評価した.

方式 1 行方向 FFT を行なった後, 各 DSP へのデータ分配を変更し, 列方向 FFT を行う. この場合, 列方向 FFT 後に DSP 間のデータ交換は行わず, 以降の項別積, 二次元 IFFT を経て, 元のデータ分配に戻す (図 2).

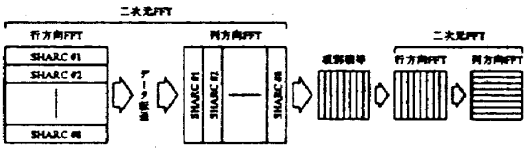

図 2: 並列処理方式 1

方式 2 二次元 FFT は行方向 FFT と列方向 FFT に分解できる. さらに, n 点の FFT は, $n=n_1 \times n_2$ と分解されるとき, n_1 点 FFT した結果に回転子を乗じ, さらに n_2 点 FFT を行うことによって実現できるという重要な性質がある [6]. これを列方向 FFT に適用し, 並列化するものである (図 3)[2].

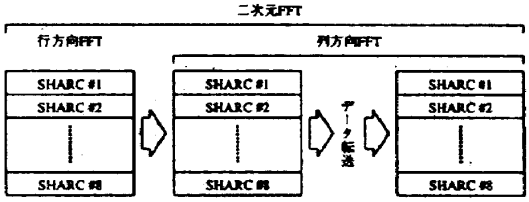

図 3: 並列処理方式 2

5 両方式の比較評価

入力画像の大きさを 512x512, 相関フィルタの大きさを 64x64 とした場合の, 各方式におけるおおよその実行時間を, 使用するライブラリの実行時間を参考に算出した (表 1). なお, 実際には, 入力画像, 相関フィルタ及び処理結果は, SHARC#0(図 1参照) に付加されたメモリ (DRAM) に格納するが, ここでは各 DSP に分配されているものとし, SHARC#0 とのデータ転送は含めていない. また, DSP 間のデータ転送は 20MByte/sec で行えるものと仮定し, 処理と平行して行うものとする.

その結果, 方式 2 は, DSP 数 =8 の場合には方式 1 に劣るものとなった. これは, 方式 2 では列方向 FFT を分割して行うため, FFT を計算する関数の呼び出し回数が増大し, オーバーヘッドが大きくなるためである.

表 1: 各方式の実行時間見積り

		処理時間 [ms]	(高速化率)
	DSP 数 = 1	2,190.0	
方式 1	DSP 数 = 8	341.9	(6.41)
	DSP 数 = 16	259.3	(8.44)
方式 2	DSP 数 = 8	481.8	(4.55)
	DSP 数 = 16	257.8	(8.50)

しかし, DSP 数 =16 の場合には, 方式 2 はこのような処理のオーバーヘッドがあるにも関わらず, 僅かながら方式 1 よりも効果があることを確認した.

これは, DSP 数の増加にともない, 各 DSP の処理が小さくなる反面, データ転送は大きくなるため, 処理とデータ転送をよりオーバーラップできる方式 2 の効果が表れるためである.

大規模な画像においては, データ転送が相対的により大きくなるため, 方式 2 はさらに有効になると考える.

6 実装上の課題

今回は机上計算にて両方式を比較したが, 実際に本画像相関処理を今回想定したマルチ DSP ボード上に適用する場合, DSP(SHARC プロセッサ) 内部の SRAM(512KByte) にデータが格納しきれないため, いずれの方式もこのままでは実現することができないという問題がある. 入力画像を 256x256 に 4 分割して処理を行う等の工夫が必要である.

7 おわりに

我々の提案した二次元 FFT の並列処理手法について, 典型的な応用例である画像相関処理を取り上げ, 市販のマルチ DSP ボード上での有効性と実装上の課題について示した.

参 考 文 献

1) Angelopoulos, G. and Pitas, I. : "Two-dimensional FFT algorithms on hypercube and mesh machines", In *Signal Processing*, Vol.30, pp.355–371 (1993).

2) 水野政治, 宮田裕行 : 「二次元 FFT の並列処理手法の検討」, 情報処理学会研究報告, Vol.98, No.18(98-ARC-128/98-HPC-70), pp.13–18 (1998).

3) Casasent, D. and Ashizawa, S. : "Synthetic aperture radar detection, recognition, and clutter rejection with new minimum noise and correlation energy filters", In *Optical Engineering*, Vol.36, No.10, pp.2729–2736 (1997).

4) http://www.alexusa.com/

5) http://www.wideband.com/ADSP.html

6) Bailey, D. H. : "FFTs in External or Hierarchical Memory", In *The J. Supercomputing*, Vol.4, pp.23–35 (1990).