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

2005－SLDM－122 (28)
2005／12／2

C言語設計によるリアルタイム粒子追跡システム

上甲 憲市† 大口 貴裕† 上津 寛和‡ 酒井 皓司‡
大倉 崇宜‡‡ 神戸 尚志‡

†近畿大学大学院 総合理工学研究科 〒577-8502 大阪府東大阪市小若江 3-4-1
‡近畿大学 理工学部 電気電子工学科 〒577-8502 大阪府東大阪市小若江 3-4-1
‡‡日本圧着端子製造株式会社 〒536-0014 大阪府大阪市城東区鴫野西 2-6-8
E-mail: †0533340409y@kindai.ac.jp, ‡tkambe@ele.kindai.ac.jp

あらまし 粒子追跡技術は、流れ場を空間的に連続して測定することを目的とし、様々な手法が提案されている。本論文では粒子マスク相関法(PMC 法:Particle Mask Correlation Method)と、カルマンフィルタ(Kalman-filter)とx^2検定(Chai-square Test)を組み合わせたKC法を用い、ソフトウエアとハードウエアで構成するシステムを設計し、処理の高速化によるリアルタイム動作の実現について述べる。

キーワード 粒子マスク相関法，カルマンフィルタ法，x^2検定，ハードウエア/ソフトウエア協調設計，Bach システム

C-Based design of a Real-time Particle Tracking System

Kenichi JYOKO†, Takahiro OHGUCHI†, Hirokazu UETU‡, Koji SAKAI‡,
Takanori OHKURA‡‡, and Takashi KAMBE‡

†‡Kinki University, 3-4-1 kowakae, Higashi-Osaka City, Osaka, 577-8502 Japan
E-mail: †0533340409y@kindai.ac.jp, ‡tkambe@ele.kindai.ac.jp

Abstract Various techniques are proposed to track particles in the field continuously . In this paper, a software and hardware system based on the PMC and the KC method that combines the particle mask correlation method with the kalman filter and Chai-square Test is designed to achieve real time processing. And the processing speed and the area of the system are evaluated.

Keyword Particle Mask Correlation method, Kalman-filter, Chai-square Test , software/hardware co-design, Bach system

1. はじめに

粒子追跡技術は、ある時間間隔で画像中の各トレーサ粒子の移動を自動的に追跡し、水中などの流れ場を計測する。この技術を応用し、物体や流体の微細な変形や運動を画像計測することができる。応用例として流体観測できなかった種々のミクロ・スケールの高速現象、ハードディスク等の高速回転体やそれに付随する高速流れによって生じる境界層の構造の乱れ、微細掘削加工時に生じる様々な高速現象等を立体的スローモーションで観察することが可能になる。

本研究では、水中の流れ場の様子をリアルタイムで観測することを目的とし、C 言語で記述された粒子マスク相関法(以下 PMC 法)とカルマンフィルタ(Kalman-filter)とx^2検定(Chai-square Test)を組み合わせた KC 法からなる粒子追跡プログラムを解析し、計算時間を要している部分に対してハードウエア化を行う。粒子追跡システム全体では、ハードウエア化した粒子マスク相関法と KC 法回路の高速化・回路規模削減を行い、リアルタイム処理を検討する。

2. 粒子追跡技術とハードウエア化

2.1 粒子追跡システムの概要

粒子追跡システムは粒子マスク相関法と KC 法で構成される。

PMC 法とは、円形に近い粒子画像の抽出を相関値計算を用いて行う手法である。相関値計算式を式 2.1 に示す。粒子画像は対称な二次元正規分布で近似するとし、これをマスクと呼ばれるテンプレートとして対象画像上に重ね相関係数を計算する。その値があるしきい値

以上になる部分に粒子が存在するとみなし、粒子の位置を推定する。

$$r = \frac{\sum_{i,j=1,1}^{n\times m}\left(I_{i,j}-\overline{I}\right)\left(M_{i,j}-\overline{M}\right)}{\sqrt{\left[\sum_{i,j=1,1}^{n\times m}\left(I_{i,j}-\overline{I}\right)^2\right]\left[\sum_{i,j=1,1}^{n\times m}\left(M_{i,j}-\overline{M}\right)^2\right]}} \quad (2.1)$$

n,m:水平,鉛直相関領域サイズ　i,j:水平,鉛直位置

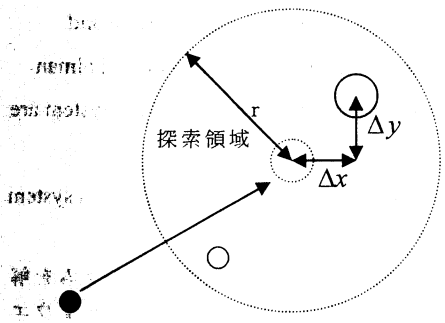
I:元画像の輝度値　$\overline{I}$:元画像の輝度値の平均

M:元画像の輝度値の平均 $\overline{M}$:マスク画像の平均値

KC法は、確立・統計学の基本に基づいたカルマンフィルタ理論を粒子追跡に適用し、x^2検定でどの予測粒子がどの実測粒子に対応しているかを求める手法である。

ある時刻tの粒子像座標、速度、加速度などの状態量がわかっていれば、次時刻t+1の予測値がカルマンフィルタによって推定される。その予測値と観測値から、同一粒子像の同定を行う。同定ができれば、観測データからさらにt+2時刻の予測値が求められ、さらに同様の操作を繰り返し行うことにより同一粒子像を時々刻々追跡していく。

図2.1を用いて同定方法を説明する。時刻tに黒点●の位置にあった粒子がt+1には点線の○の位置に移動すると推定されたとする。対応する粒子候補は点線の大円内の大小2個の○で示されている。このとき、対応する粒子候補すべてとx^2検定を行い、値が最小値となるものを同一粒子と対応付けをする。



カルマンフィルタによる予測

○:(t+1)時刻の実測粒子位置

◌:カルマンフィルタによる(t+1)時刻の実測粒子位置

●:t時刻の粒子位置

図2.1　カルマンフィルタ・x^2検定法の概念図

Fig.2.1 Concept of Kalman-filter・Chai-square Test

本システムは次のような手順で粒子追跡を行う。

① マスク相関値の計算：マスク画像を1枚の静止画像に置き、輝度分布との相関係数を求める。マスク画像の中心座標をピクセル単位で動かし、全画面上を走査しこの計算を行う。

② 2値化処理：相関係数があるしきい値以上の領域とそれ以下の領域に2値化する。二値化は粒子像を抽出するための処理である。背景の輝度値が0の場合、輝度値1をもつ画素の一つの集まりは1個の粒子像に対応する。

③ 連結領域の抽出(ラベリング):粒子画像中には粒子が多数存在する。個々の粒子を区別するためそれぞれに番号付けを行う。

④ 粒子像中心計算：番号ごとに粒子の中心を求め、その座標をその粒子位置とする。

以下の処理をすべての粒子について行う。

⑤ カルマンフィルタ：カルマンフィルタにより時刻t画面上にある粒子の(t+1)画面上における位置を推定する。

⑥ x^2 検定:(t+1)画面上において推定された粒子位置から半径rの検索領域以内にある粒子すべてについてx^2検定を行う。x^2値が最小となるものを仮の対応する粒子とし、x^2値がある棄却水準以下であれば同一粒子と確定する。棄却水準以上であれば対応する粒子がないものとする。

2.2　ハードウエア設計技術

ハードウエア設計にはBachシステムを使用した。現在のHDLによるハードウエア設計は、C言語等を用いてアルゴリズム設計・検証を行い、その後、HDLでの回路設計、論理合成、シミュレーションによる検証という手順で行われる。しかし、HDLでの記述とプログラミング言語との差は大きく、HDLによる回路設計を行った後、不具合や変更によるアルゴリズムの修正には、おおきな手間が必要となる。Bachシステムを用いた設計では、トップレベルの設計から抽象度の高いBach-Cを用い、機能設計・検証を行うことで、HDLによる記述・検証期間が削減されるたけでなく、Bach-C記述からRTLのVHDLが自動生成され、ハードウエア設計が大幅に効率化される。

3. リアルタイムシステムの設計

一般にハードウエアは、高速化、低消費電力化のために用いられ、ソフトウエアは汎用性を持たせて処理を行いたい時に効果的である。ソフトウエアをハードウエア化するには、構造によって適切な部分が異なる。他の関数による影響が少なく処理が独立している関数をハードウエア化することにより、全体の処理を大きく変更することなくハードウエアとソフトウエアを組み合わせてシステムを実現することが出来る。

3.1　ハードウエア化箇所の決定

ハードウエアとソフトウエアを組み合わせた効率の

良いシステムを構築するために、元となるソフトウエアの理解、分析を行った。我々は粒子マスク相関法、KC法の各関数の役割を調べ、構造を解析し、関数の処理時間の割合を計測し、ハードウエア化に適切な箇所を決定した。

PMC 法計算部は、画像ファイルから画像データを読み込み(Imread)、粒子認識率を上げるために画像を 4 倍し(expand 関数)、相関値の計算を行い(cor 関数)、粒子の抽出と座標の出力を行う(chain 関数)。

KC 法計算部はカルマンフィルタを計算(Kalman 関数)し、χ^2検定(Chai 関数)を行い、粒子を複数の粒子と対応付けていないかチェック(Check 関数)を行い、対応付かなかった粒子への粒子情報の内挿のため、対応付いた粒子を使った平均空間を求め(Spav 関数)、結果を出力する(dataout 関数)。

プログラム構造の解析と処理時間の分析により、粒子マスク相関法は、cor が全体の 98%の処理時間であることから cor 関数のハード化を行う。KC 法は Chai と Spav が全体の処理時間に対する割合が高い。このことより Chai、Check、Spav をハード化することが有効と考えられる。また、各関数が粒子情報を用いて計算するため、独立性が低い。そのため KC 法全体に注目し、ハードウエア化を行う。

3.2 システム構成

我々は図 3.1 のようなシステムを設計した。このシステムは、主にソフトウエア部、通信部、ハードウエア部からなる。

画面データの読み込み、画像の 4 倍化はソフトウエア部で行い、PMC 法、KC 法の計算をハードウエアで行い、その結果に対し、ソフトウエア部で出力処理を行う。

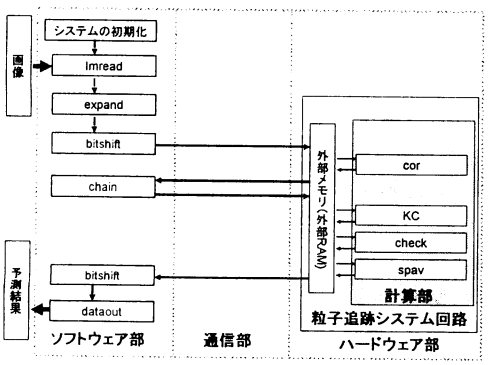


図 3.1 システムの概要

Fig.3.1 Block diagram of the target system

4. ハードウエアの最適化

高速化するために次の方法により処理の高速化を行う。

(1) 二次微分による判定

(2) 1pixel 処理

(3) キャッシュメモリによる高速化

(4) χ^2検定と spav 関数の処理範囲の限定

(5) カルマンフィルタとχ^2検定のパイプライン処理

(1) 二次微分による判定

粒子画像は主に粒子と背景に分けられるため、明らかに背景である画素については相関値計算を省くことで高速化する。注目画素が背景か粒子の一部かを判断するため、注目画素を中心に前後の画素との輝度の二次微分値を用いる。二次微分値は、粒子のような輝度値が盛り上がった場所において負の大きな値をとる。この値を検出することで、多くの背景画素における相関値計算を省ける。二次微分値を求める式を(4.1)式に示す。

$$(f(x+1)-f(x))-(f(x)-f(x-1)) = f(x+1)-2f(x)+f(x-1) \quad (4.1)$$

x:注目画素の画像上の x 軸位置

$f(x)$:位置 x の輝度値 d:注目画素からの距離

(4.1)式は(後画素－中心画素)－(中心画素－前画素)によって求めることができる。また、マスクの幅が 7×7 の為、幅を 7 に設定し計算式を(3 つ後の画素－中心画素)－(中心画素－3 つ前の画素)とすることで、より粒子の情報を反映した二次微分値になり、正確に正負の判別が可能となる。図 4.1 に 2 次微分の結果を示す。また、x 軸方向だけでなく y 軸方向、斜めの計 4 方向に対して 2 次微分を行う。

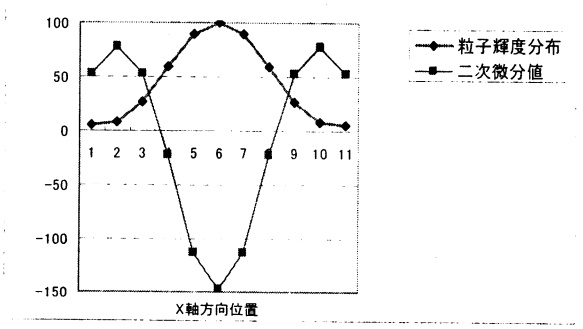


図 4.1 理想粒子輝度分布と 2 次微分値

Fig.4.1 Distribution of brightness of ideal particle and second order differential value

(2) 1pixel 処理

相関値処理の際、平均値やマスク計算、相関値計算は、各処理を全画面分について行い、結果をメモリに記憶するという方法をとっていた。1pixel 処理は、1pixel ごとに計算を行い、結果を記憶する。これによ

りメモリの容量やメモリアクセスが大幅に削減でき、処理を高速化することができる。図 4.2 に 1pixel 処理と逐次処理を示す。

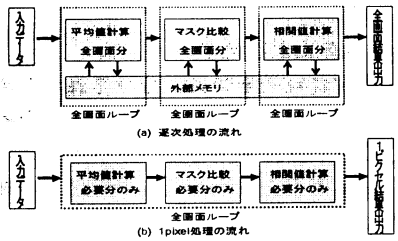


図 4.2　1pixel 処理と逐次処理の比較
Fig.4.2 Comparison of
1pixel processing and serial processing

(3)　キャッシュメモリによる高速化

1pixel 処理をさらに高速化するためにキャッシュメモリを用いる。この方法は、平均やマスクとの比較計算の際、マスクの範囲(7×7pixel)で計算を行うが、一つ一つ値を読み出しているため、メモリアクセス時間がボトルネックとなる。画像のデータをキャッシュメモリに入れることでメモリアクセスの時間を短縮させることができ、さらにこの部分を 7 段並列処理にすることにより高速化を目指す。図 4.3 に 1pixel キャッシュ処理を示す。

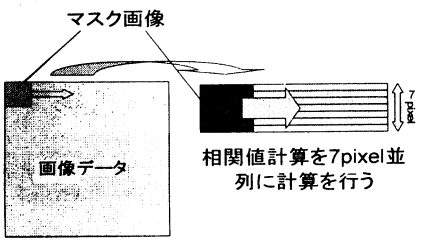


図 4.3　1pixel キャッシュ処理
Fig.4.3 1pixel cash processing

(4)　χ^2検定と spav 関数の処理範囲の限定

χ^2検定では予測された位置から、次画面上でその粒子と対応する可能性のある粒子を選択するため、次画面上のすべての粒子に対して、予測値と、実測地の差をとりχ^2検定の検定範囲内かを判定し、検定範囲内なら予測値と実測値の差の二乗和を計算する。粒子の実測値の情報はメモリに格納しているため、粒子数が増えるとメモリアクセスに膨大な時間が必要になってくる。そのため予測粒子座標との距離計算とχ^2検定処理をカルマンフィルタによる予測粒子座標の周りの粒子のみに行うことにした。画像を縦 20pixel、横 20pixel の正方形で区切り各マスに番号付けをする。画像サイズは横が 2016pixel、縦が 2036pixel なので、マスの個数は横に 101 個、縦に 102 個の全部で 10,302 個になる。

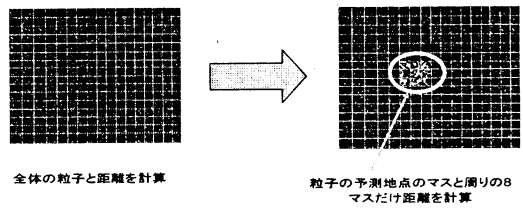


図 4.4　マス分けした粒子画像
Fig.4.4 Partitioning of particle image

まず、粒子マスク相関法で画像から抽出した粒子がどのマスに入っているかを判定しておく。次にカルマンフィルタで予測した粒子位置がどのマスに入っているのかを検索し、そのマスの周りの 8 個のマスを選択する。オリジナルのχ^2検定における検定範囲は 15pixel なので、χ^2検定は粒子予測地点と周りの 8 個との計 9 マスだけを検索すれば十分である。

また、spav 関数についても対応付かなかった粒子と画像内のすべての粒子の距離をとり、処理範囲かどうかを判定する。そのためここでも同じ方法で範囲限定を行う。ここでは処理範囲が 16pixel なためマスの大きさを縦 17pixel、横 17pixel とする。

(5)　カルマンフィルタとχ^2検定のパイプライン処理

カルマンフィルタとχ^2検定は処理が独立しているためのパイプライン処理を行う。

パイプライン処理は以下の 3 つの関数で構成する。

(a) パイプライン処理関数
(b) カルマンフィルタ
(c) χ^2検定

パイプライン処理を行うためにカルマンフィルタ関数とχ^2検定関数を並列に実行する必要がある。またカルマンフィルタ関数とχ^2検定関数の同期をとる必要がある。カルマンフィルタとχ^2検定はそれぞれ 1 個の粒子ずつ処理を行う。1 個の粒子ごとにカルマンフィルタ計算部が終わるとχ^2検定計算部にχ^2検定起動信号を送信し、χ^2検定計算を開始する。

5. 実験結果

実験を行う上で設計した回路は二次微分による判定、1pixel キャッシュ処理、χ^2検定、spav 関数の処理範囲の限定、カルマンフィルタとχ^2検定のパイプライン処理の 5 種類である。以下の実験では Bach での回路面積の見積もり、RTL でのシミュレーション処理時間を測定する。

(a) PMC 計算部の高速化

PMC 計算部は、二次微分による判定、1pixel キャッシュ処理を用いて高速化した。元の回路、二次微分、1pixel の各回路について回路面積、処理時間を測定した。結果を表 5.1 に示す。二次微分を用いた回路は逐次処理と比較して、約 23 倍の高速化を実現できた。また 1pixel キャッシュについては約 28 倍の高速化を行うことができた。

	処理時間(ms)	回路規模(ゲート)
改良前	818	258,022
二次微分	36	261,639
1pixel キャッシュ	29	325,112

表 5.1 二次微分の実行時間と回路面積

Table.5.1 Experimental results (speed and area) of second order differential with cash memory

(b)KC 計算部の高速化

KC 計算部は χ^2 検定と spav 関数の処理範囲を限定することと、カルマンフィルタと χ^2 検定をパイプライン処理することで高速化した。χ^2 検定の処理範囲を限定した回路と χ^2 検定の処理範囲を限定していない回路について、回路面積と処理時間を測定した。その結果を表 5.2 に示す。

	χ^2 検定実行時間(μs)	回路面積（ゲート）
処理範囲の限定なし	11,146,523	652,030
χ^2 検定の範囲限定	24,505	682,705

表 5.2 エリア限定の実行時間と回路面積(χ^2 検定)

Table 5.2 Experimental results (speed and area) of area limitation for Chai-square Test

χ^2 検定の処理範囲を限定した回路は元の回路より回路面積が 5%増加しただけだが χ^2 検定の実行時間は約 455 倍の高速化ができた。

χ^2 検定の処理範囲を限定した回路に spav 関数の範囲限定をした回路を加えた。spav 関数の処理範囲を限定していない回路との比較を表 5.3 に示す。処理範囲を限定した回路は元の回路より 6%の回路面積が増加したが処理時間では約 23 倍の高速化ができた。

	spav 実行時間(μs)	回路面積（ゲート）
処理範囲の限定なし	1,122,307	682,705
spav の範囲限定	49,808	721,601

表 5.3 範囲限定の実行時間と回路面積(spav)

Table 5.3 Experimental results (speed and area) of area limitation for spav function

カルマンフィルタと χ^2 検定のパイプライン処理の回路と逐次処理の処理時間を測定した。結果を表 5.4 に示す。χ^2 検定よりカルマンフィルタ処理の方が処理時間が大きな割合を占めているためパイプライン後の処理時間は逐次より約 20ms の高速化に留まった。

	実行時間(μs)	回路面積(ゲート数)
逐次処理	104,232	682,705
パイプライン処理	80,795	799,821

表 5.4 パイプライン化に対する実行時間と回路面積

Table 5.4 Experimental results (speed and area) of pipeline processing

6. システム全体のパイプライン化

本システムをより高速にするためにシステム全体でパイプライン処理を行う。表 6.1 に CPU の処理時間と各ハードウエアの各処理時間を示し、処理の流れを図 6.1 に示す。

	処理内容	処理時間(ms)
CPU	Imread, expand, ビットシフト	30
	chain	250
	ビットシフト, dataout	35
ハードウエア	cor	100
	KC	80

表 6.1 ソフト部とハード部の各処理時間

Table 6.1 Processing time of each part

（ソフトウエア部は Pentium4 3.4GHz を使用）

ソフトウエアで Imread, expand, chain, ビットシフト, dataout の各処理を行い、cor, KC についてはハードウエアで処理を行う(図 6.1)。このようにパイプライン処理を行うことによって 1 枚の画像の処理には 1575ms の時間が必要になるが、連続して処理を行う場合には 315ms おきに処理を行うことが可能となる。

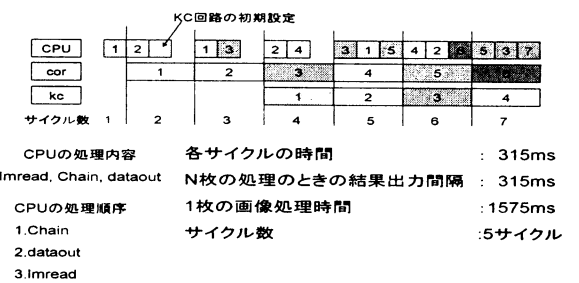


図 6.1 パイプラインシステムの流れ

Fig.6.1 Flow of pipeline processing

7.システムの改良

設計したシステムでは、ソフトウエアによる処理がボトルネックになっているためソフトウエアの処理の一部をさらにハードウエアで行うことによって高速化を考える。

①Chainのハードウエア化を行う。

② ソフトウエア部で expand した後にデータをハードウエアに送る場合はデータ量が 4 倍になる。さらに画像を 4 倍する際に周りの画素の平均を取るため、データ型が float になり 32bit のバスを使用する場合は 1 回の通信で 1 画素のデータしか送ることができない。ファイルから入力したデータを直接ハードウエアに送信し、ハードウエアで expand の処理を行うことによって、データ通信量を 1/4 に減らす。また、画像からの入力データは 0 から 255 までの整数値であるため、ビットシフトを行う必要もなく bit 数は 8bit あればよい。これにより 1 回の通信で 4 画素分のデータを送信することができ、通信回数をさらに 1/4 に削減する。

この時のソフト処理時間測定の結果を表 7.1 に示す。

	処理内容	処理時間 (ms)
CPU	Imread	10 以下
	ビットシフト, dataout	35
ハードウエア	expand	設計中
	cor	100
	chain	設計中
	kc	80

表 7.1 改良後の処理時間

Table 7.1 Processing time after improvement

この結果からソフトウエア全体の処理時間は cor, kc の各処理時間より大幅に小さくなり、システム全体の処理時間はハードウエアに依存することになる。ハードウエアは目標である 1 秒間に 15 枚の処理を行うために、expand, cor, chain, kc の各処理を 66ms で行えるように更なる高速化を行えば、最終的には図 7.1 で示すシステムが実現できる。

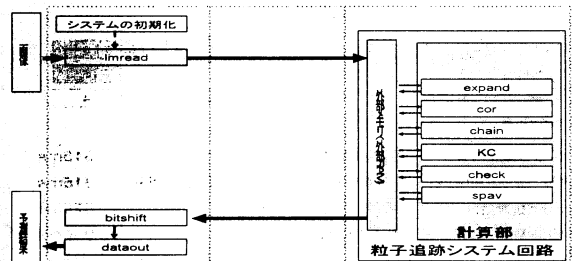


図 7.1 改良後のシステム構成

Fig.7.1 Target system after improvement

このシステムは各処理を並列に行うため 1 枚の画像の処理には 528ms の時間がかかるが、n 枚の処理を行う場合には 66ms 間隔で処理を行うことが可能となり、目標の 1 秒間に 15 枚の処理を行うことが可能となる。

8. まとめと今後の課題

本研究では、粒子追跡システムで計算時間を多く要している部分をハードウエア化することによりシステム全体の効率化を図る方法を示した。

今回設計を行ったシステムはソフトウエアの処理が大きく高速化したハードウエアの性能を使いきることができなかった。そのためシステムの改良を行いソフトウエアの処理の一部をハードウエアで行うことによってソフトウエア,ハードウエアともに性能を使いきることができる効率の良いシステムの設計を行う。

今後の課題として目標の 1 秒に 15 枚の処理を可能にするためにハードウエア部の各処理を 66ms 以下にする必要がある。その方法として、キャッシュメモリの利用、必要ビット精度の論理的推定による使用ビット数の削減、ハードウエアに適するようにアルゴリズムの見直しなどが上げられる。また、現在 Bach システムを使用して回路すべての設計を行っているが、相関値計算部などの多くの処理時間が必要になる部分を HDL 設計による専用演算器を用いることで、処理時間の高速化を行う。

謝辞

粒子追跡技術を使用した研究を行なうにあたり、粒子追跡法のソースとして粒子マスク相関法を提供し、ソフトウエアの解析にあたって直接ご指導いただいた近畿大学理工学部社会環境工学科江藤剛治教授、竹原幸生助教授に深くお礼申し上げます。

Bach を用いたハードウエア設計を実現するに当たり、多大なるご指導を頂いたシャープ株式会社山田晃久様をはじめ、BACH 開発グループの皆様に厚く御礼申し上げます。

文　献

[1] 可視化情報学会編：“PIV ハンドブック”、森北出版株式会社、東京、1～4（2002）

[2] 江藤剛治、竹原幸生、道奥康治、久野悟志：“PTV のための粒子画像抽出法に関する検討－粒子マスク相関法について－”、水工学論文集、**40**、1051～1057（1996）

[3] 江藤剛治、竹原幸生、岡本孝司：“標準画像を用いた粒子マスク相関法と KC 法の性能評価”、日本機械学会論文集、**65**、184～191（1999）

[4] 吉田たけお、尾知博：“ＶＨＤＬで学ぶディジタル回路設計”、ＣＱ出版、東京、(2002)

[5] “Bach システムマニュアル”、シャープ株式会社提供(2003).