画像情報処理の高速化のためのソフトウェア設計と実装に関する研究

飯島　俊匡，本間　稔規，橋場　参生，髙橋　裕之

Study on Software Design and Implementation for Acceleration of Image Processing

Toshimasa IIJIMA, Toshinori HONMA, Mitsuo HASHIBA, Hiroyuki TAKAHASHI

抄　録

近年，様々な産業分野でコンピュータを用いた画像情報処理により計測や評価を行うシステムが利用されている。こうした計測及び評価システムでは計測精度の向上やリアルタイム処理などへの要求が強く，その実現のためには画像情報処理ソフトウェアの高速化が大きな課題となっている。

一方，近年のコンピュータはCPU動作速度の向上が鈍化しており，それを補うためマルチコアCPU構造や，GPU（グラフィックス描画処理装置）など汎用CPU以外のプロセッサを併用するヘテロジニアス（異種混合）構造へと転換しつつあるが，こうした構造のコンピュータの性能を引き出すためには，並列計算などを考慮したプログラムの実装が必要となる。

そこで本研究では，マルチコアCPUやヘテロジニアス構造の計算機を効果的に利用するために必要なソフトウェア設計と実装方法に取り組んだので報告する。

キーワード：画像情報処理，並列計算，GPGPU

Abstract

In recent years, measurement and evaluation systems by computer image processing has been used in various industrial fields. In order to improve the performance of these systems, speed of the image processing software is important.

On the other hand, growth of CPU speed has slowed down, and heterogeneous computer systems based on multi-core CPU and GPU is beginning to be used. However, in these systems, to implement a program using parallel computation is required.

Thus, we report the working on software design and implementation for acceleration of image processing using multi-core CPU and GPGPU.

KEY-WORDS : Image processing, Parallel computing, GPGPU

1．はじめに

近年，様々な産業分野でコンピュータを用いた画像情報処理(※)により計測や評価を行うシステムが利用されている。例えば，工業分野では工場の生産ラインにおける部品の位置決めや検品など，農水産分野では生産物の品質評価や異物検出などである。当場でも，実環境における移動体の検出・追跡技術に関する研究[1)]や，超解像処理を利用した画質改善技術に関する研究[2)]，廃棄物処分場向けの積雪寒冷地用監視カメラシステムの開発[3)]など，画像情報処理を用いた研究開発を実施してきた。

これら画像情報処理を利用した研究開発や実用化では，計測及び評価システムへの要求が益々高度化してきており，多種多数の対象物への対応や計測精度の向上，さらに処理速度

事業名：経常研究
課題名：画像情報処理の高速化のためのソフトウェア設計と実装に関する研究（平成22～23年度）

（※）画像情報処理とは，カメラやセンサなどから得られる信号を解析し，人間にとって意味を持つ情報を得る処理のことを言い，一般的にソフトウェアで実装される。

のリアルタイム性などが求められている。こうした要求に応えるためには，高解像度のカメラを用いるなど扱うデータの量を増やしたり，高度な情報解析手法を用いるなど扱う計算の量を増やすことで，より精度の高い計測や信頼度の高い評価を行うことができる。しかし，データ量や計算量が増えることで，一度に処理できる対象物の量が限定されたり，長い処理時間が必要となってしまう。したがって画像情報処理の高速化が産業応用への大きな課題となっている。

これまでの画像情報処理の高速化は，CPU（Central Processing Unit：中央演算処理装置）などのコンピュータハードウェアが年々高速化することにより，ソフトウェアで実装される画像情報処理も自動的に高速化がなされてきた。しかし，近年のコンピュータは半導体集積度を向上させることが限界に近づきつつあるため，CPUの性能向上が鈍化しており，それを補うためCPU内部に複数の演算コアを持つマルチコアCPU構造や，GPU（Graphics Processing Unit：グラフィックス描画処理装置）など汎用CPU以外のプロセッサを併用するヘテロジニアス（Heterogeneous：異種混合）構造へと転換しつつある。このような構造のコンピュータで従来の画像情報処理ソフトウェアを実行しても速度が向上することは無く，その性能を引き出すためには，並列計算などを考慮した上で適切なソフトウェアの設計及び実装が必要となる。

そこで本研究では，マルチコアCPUやヘテロジニアス構造の計算機を効果的に利用するために必要なソフトウェア設計と実装方法に取り組んだので報告する。

2．コンピュータ構造とソフトウェア開発の転換

2.1 マルチコア化

コンピュータの演算はCPUで行われており，その性能は基本的に動作周波数×IPC（Instruction Per Cycle：1サイクルで実行できる命令の数）で決まる。CPUを製造する半導体メーカーは，2004年頃までは1つの演算コアを持つシングルコア構造のCPUに対して，動作周波数を上げることによりその性能向上を実現してきた。しかし，動作周波数に比例して増大する消費電力とCPUから発生する熱を処理しきれなくなり，更なる周波数向上が困難となったためCPUの性能向上が鈍化した。

そこで半導体メーカーは，動作周波数の向上によるCPU性能の強化から，1つのCPU内部に複数の演算コアを持つマルチコア構造のCPUへと転換することによりIPCを増やし，その性能向上を実現してきた。

例えば，汎用コンピュータ向けで大きなシェアを持つインテル社[4)]のCPUでは，Pentium4以前はシングルコアであり，PentiumDやCoreシリーズ以降のCPUは全てがマルチコアとなっている。また，アドバンスト・マイクロ・デバイセズ（AMD）社[5)]のCPUでは，Athlon64以前はシングルコアCPUであり，Athlon64 X2以降は全てマルチコアとなっているなど，現在市販されている汎用コンピュータのほとんどはマルチコアCPUを搭載している。

図1　マルチコアCPU
（左：インテル Core i7，右：AMD FX-Series）

2.2 ヘテロジニアス化

汎用コンピュータには映像を出力するハードウェアが搭載されており，近年は3次元グラフィックス描画処理を効率よく行うため，高速な浮動小数点演算回路を持ち，大規模な並列計算が可能なプロセッサ構造を持つGPUが搭載されている。代表的なものではエヌビディア（NVIDIA）社[6)]のGeForceシリーズ，AMD社のRADEONシリーズなどがある。

図2　GPU
（左：NVIDIA GeForce GTX580，右：AMD RADEON HD7950）

これらのGPUはCPUに比べて単機能の演算コアしか持たないが，コアの数がCPUは4～8個であるのに対し，GPUは数百個持つものもあり，大規模な並列計算が可能となっているため大幅な高速処理が可能である。図3は，2002年以降のインテル社のCPUとNVIDIA社のGPUの性能差をGFLOP/s（1秒あたりの浮動小数点演算性能）で表したグラフである。

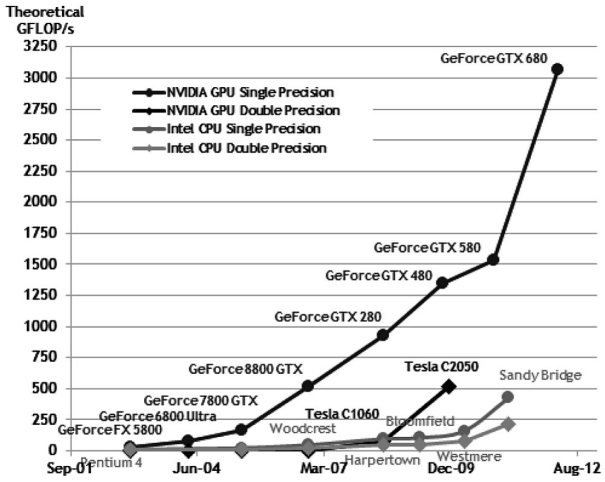

図３　CPUとGPUの理論性能差の変遷
（“CUDA C Programming Guide”[7]より引用）

このようにGPUはCPUに比べて大幅に高速な処理が可能であることが分かる。ただし，この図が示す値は全ての演算コアを無駄なく利用した場合の理論値である。

現在はこうした高性能なGPUをグラフィックス描画処理だけではなく，一般的な数値演算に利用する試みがなされてきており，そのような使い方を総称してGPGPU（General Purpose Graphics Processing Unit）と言う。

前節で述べたように，速度向上が鈍化したCPUに加えてGPGPUを利用したヘテロジニアスコンピューティングが今後のコンピュータ利用環境の主流を占めていくと考えられ，それに対応したソフトウェア開発が重要となっている。

2.3 ソフトウェア開発への要求

一方，ソフトウェア開発においては，2004年頃まではシングルコアCPUのコンピュータ環境しかなかったこともあり，ソフトウェアの機能向上に主眼が置かれ，高速化はCPUの性能向上に依存してきた。つまり，ソフトウェア側は何もしなくても，CPU性能が上がるのを待っていれば，同じ設計のソフトウェアが速く動くようになっていた。

ところが，2005年以降コンピュータ環境がマルチコアCPUになると，これまでのソフトウェアをそのまま実行しても，１つの演算コアでしか処理が行われずCPUの能力を一部しか使えない。そのため，CPUの能力を最大限に活かすためには，CPUの持つコア数に応じて処理を分割し，各コアの計算量が同程度になるようにするなど，ソフトウェアを改めて設計してプログラムを実装する必要がある。

また，GPGPUを利用する場合も同様であり，GPUの持つ大規模な並列回路を効果的に利用するようソフトウェアの設計を変える必要があり，GPUを用いて並列計算を実行するようにプログラムを実装する必要がある（図４）。

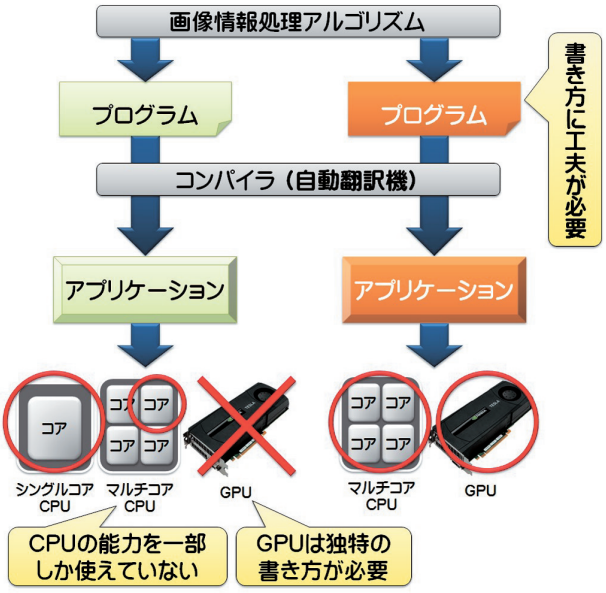

図４　実行環境によるソフトウェア開発の違い

3．ソフトウェアの設計とプログラムへの実装

3.1 アルゴリズムの並列計算化

一般的に画像情報処理ソフトウェアは，そのアルゴリズム（計算手法）に基づき，入力されたデータの端から一つずつ計算を行う逐次処理でプログラムに実装されているが，マルチコアCPUやGPGPUの性能を引き出すために並列計算を用いて実装する必要がある。

あるアルゴリズムの一部または全部が並列計算可能かどうかはその処理内容によるが，ソフトウェア設計の基本的な戦略は以下のようになる。

まず，逐次処理は複数の処理から成り立っているので，それぞれの処理で分割することを検討する。また，扱うデータが複数ある場合には，それぞれのデータ毎に処理を分割することを検討する。

次に，分割を検討した処理やデータに依存関係が無いか調べ，処理の実行に順序関係がある場合は正しい実行順序になるようグループ化して処理を行う。

最後に，分割した複数の処理から読み書きされるデータが，正しく書き込まれるような排他制御やデータの同期を行う。

例えば，画像のフィルタ処理の並列化は，分割した画像同士に依存関係が無いため，データ並列性を利用した並列計算が可能である（図５）。

また，逐次処理の実行順序に依存関係が無い収束計算処理などの並列化は，分割した処理同士に依存関係が無いため，タスク並列性を利用した並列計算が可能である（図６）。

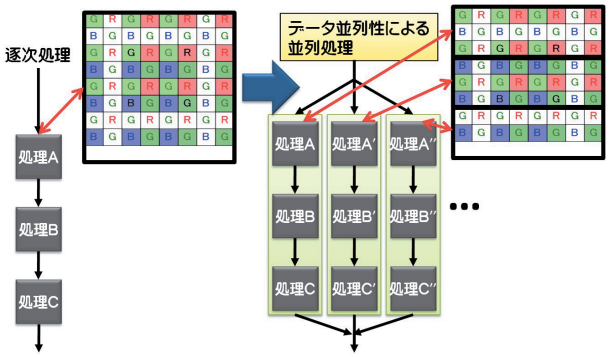

図５　データ並列性による並列計算

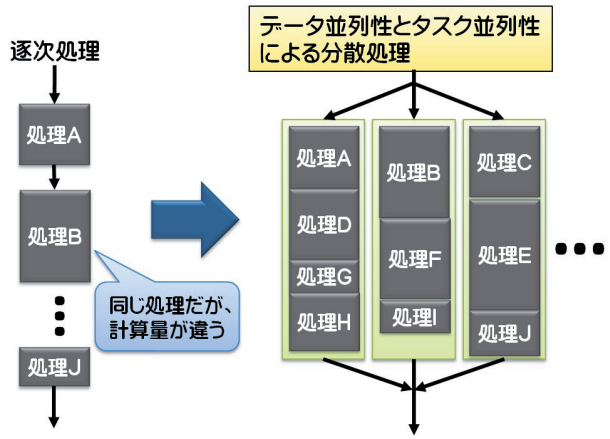

図６　タスク並列性による並列計算

3.2 プログラムへの実装

前節の戦略に基づきソフトウェア設計した画像情報処理アルゴリズムをプログラムに実装する。

3.2.1 SIMD

近年のCPUはSIMD（Single Instruction Multiple Data）演算機能を持ち，１回の命令で複数のデータを同時に演算することができる。例えば，赤青緑の各色を256段階（8bit）で表現した大きさが640×480画素のカラー画像を従来の手法で計算する場合，307,200回の演算が必要である。しかし，128Bitの演算が可能なSIMDを用いれば５画素を一度に演算することができ，単純計算で５倍の速度で処理が完了する。ただし，境界問題やメモリアクセス効率などの問題により，実際の処理速度は５倍にはならない。また，SIMDはメモリアクセスの境界合わせ（アライメント）を行わないと，処理速度が遅くなってしまう。

表１は，Intel Core i7-950（動作周波数3.07GHz）のCPUでカラー画像をグレースケールに変換する処理について，実装方法の違いによる処理時間の違いを表したものである。通常の実装方法に比べ，SIMDで実装した方が2.79～3.04倍速いことが分かる。

表１　グレースケール変換処理時間の比較

カラー画像の大きさ		通常の実装（固定小数）	SIMD（SSE2）※	SIMD（SSE2）
1920x1080	処理時間[ms]	32.44	23.99	10.68
	通常実装との比	1.00	1.35	3.04
6400x4800	処理時間[ms]	490.02	341.47	175.41
	通常実装との比	1.00	1.44	2.79

※アライメント無視

3.2.2 マルチスレッド

マルチコアCPUを効率的に利用するためには，プログラムをマルチスレッドで実装することが有効である。スレッドとは，逐次的に動作する一連のプログラムの流れを言い，マルチスレッドはスレッドを複数同時に実行することである。

図５の左側のように１つの逐次処理で動作するプログラムをシングルスレッドプログラムと言い，図５の右側のようにスレッドを複数同時に実行するものをマルチスレッドプログラムと言う。これまでの画像情報処理はほとんどがシングルスレッドプログラムで実装されていた。

また，現在使われているWindows7などのOSは，マルチスレッドプログラムを実行すると，各スレッドを自動的に各CPUコアに振り分けるようになっている。そのため，マルチコアCPUを効率的に利用するためには，コア数に応じたスレッド数となるようにマルチスレッドプログラムを実装する必要がある。

3.2.3 GPGPU

GPUを制御するためのプログラムを作成する環境がGPUメーカーから提供されており，NVIDIA社によるCUDA（Compute Unified Device Architecture）[8]や，AMD社によるAPP SDK（Accelerated Parallel Processing）[9]を利用すれば容易にGPGPUを実現可能となっている。ここではCUDAを用いて実装を試みた。

NVIDIA社のGPUは図７のような構成になっており，演算コアはGPU内部のデバイスメモリにしかアクセスできない。そのため入出力するデータは必ずデバイスメモリを介する必要があり，CPUとGPU間のメモリ転送処理にかかる待ち時間（オーバーヘッド）が生じる。したがって小さなデータを小刻みに転送するよりも，大きなデータを一度に転送するよう実装すると，高速化の効果が得られる。

また，GPUの演算コアで実行されるCUDAのプログラムはカーネル関数と呼ばれ，１つのコアで１つのスレッドが実行される。CUDAでは，同じカーネル関数を実行する複数のスレッドを集めたスレッドブロックという単位で並列計算を行い，さらにそのスレッドブロックを複数集めて一連の処理を行う（図８）。

このとき，あるスレッドブロックの演算が他のスレッドブロックの演算対象データに影響が無いよう実装しなければな

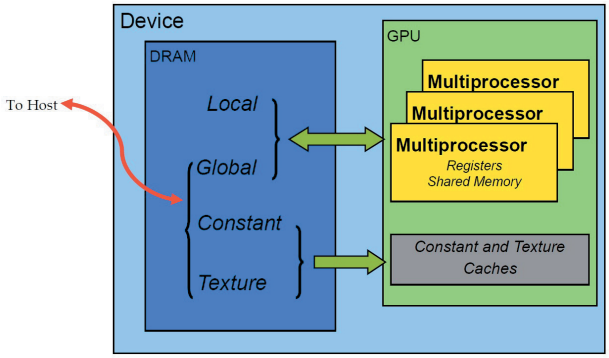

図7　NVIDIA GPUのメモリ空間
（“CUDA C Best Practices Guide”[7]より引用）

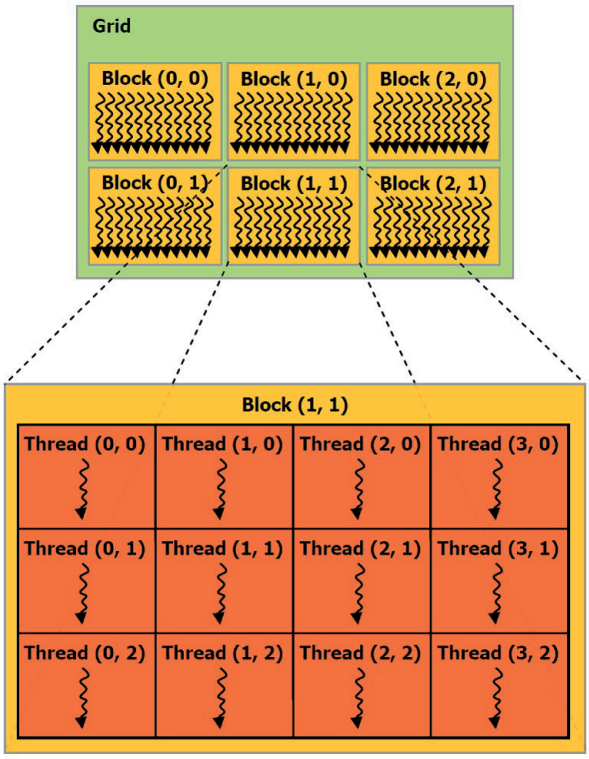

図8　スレッドとスレッドブロック
（“CUDA C Programming Guide”[7]より引用）

らないため，3.1節のデータ並列性に基づいてソフトウェア設計することが重要である。

4．アプリケーションへの適用

既存の画像情報処理アプリケーションを対象に，データ並列性やタスク並列性に基づいてソフトウェアを設計し直し，マルチコアCPUやGPGPUを用いて高速化したアプリケーションを試作し，その効果を検証した。

4.1　移動物体検出アプリケーション

以前の研究[1]で作成した移動物体検出アプリケーション（図9）は，入力された画像の背景画像をLMedS（Least Median of Squares）推定を用いて背景モデルを作成し，背景差分処理により移動物体を検出するアルゴリズムである。

図9　移動物体検出アプリケーション

その中で用いている画像データのグレースケール変換処理，画像データの差分処理は，計算対象となる画素が空間軸方向に連続しており，かつ計算が他の画素に依存しないため，SIMDを用いた局所並列計算を行うように実装した。

また，背景画像のLMedS推定処理は，計算対象となる画素が空間軸方向には連続していないが，計算が他の画素に依存しないため並列計算が可能である。そこで，LMedS推定処理は画面を上下2分割し，それぞれの処理をマルチスレッドを用いて実装した。

以前の移動物体検出アプリケーションと，新たに実装したアプリケーションをそれぞれIntel Core i7-950（動作周波数3.07GHz）のCPUで実行した結果を表2に示す。

表2　移動物体検出処理の速度比較

移動物体検出アプリケーション	これまでの実装	高速化した実装	向上率
背景更新処理時間 [s]	11.15	5.08	2.19
動作フレームレート [FPS]	40.78	45.44	1.11

背景更新処理時間は平均して11.15秒から5.08秒となり，動作速度の向上率が約2.2倍となった。また，移動物体検出処理の動作フレームレートは，毎秒40.78フレームから45.44フレームに向上した。動作フレームレートの向上率が約1.1倍にとどまるのは，SIMDを利用して並列計算を行った部分が，処理全体に対して2割程度であったためと考えられる。

4.2　光散乱シミュレーションソフトウェア

農水産物の内部状態や異物混入の有無を検査するために，近赤外光などの光を用いる方法が知られている。光を用いた計測技術は，光検出器で得られた情報と入力した光を基に物

質内部での光学特性分布を推定して計測する。推定した光学特性分布が正しいかどうかは，光伝搬解析を計算した結果と測定値を比較して評価する。この光伝搬解析をコンピュータでシミュレーションする手法としてモンテカルロ（Monte Carlo）法がある。

モンテカルロ法は，図10のように光を多数の光子と考え，この光子の吸収散乱体内部での伝搬現象をシミュレーションするもので，精度を上げるためには光子数を増やして計算する必要があるが，計算時間がかかるという欠点がある。

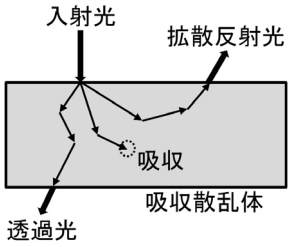

図10　モンテカルロ法

Lihong Wangによる光散乱シミュレーションソフトウェア[10]を用いて，拡散反射光を捕捉する検出器で光子を1,000個検出したときの吸収散乱体での光子の吸収を表したものが図11である。光子はx＝y＝z＝0の点（図の矢印）から入射し，検出はx＝0.2，y＝z＝0にある0.04×0.04の大きさの検出器で捕捉する。上部はy＝0のx-z平面，下部はz＝0のx-y平面を表し，白い部分ほど吸収が多いことを示す。

このモンテカルロ法のアルゴリズムでは，光子は吸収散乱体に１個ずつ投入して計算され，その処理結果が他の光子に影響を与えることは無いため，タスク並列性を利用した並列計算が可能である。また，シミュレーションでは光子を大量に投入するため，マルチコアCPUによる実装よりもGPGPUを用いた実装の方が適している。そこで，CUDAを用いて光伝搬処理をカーネル関数で記述し，光子１個の動きが１つのスレッドで処理されるようプログラムを実装した。

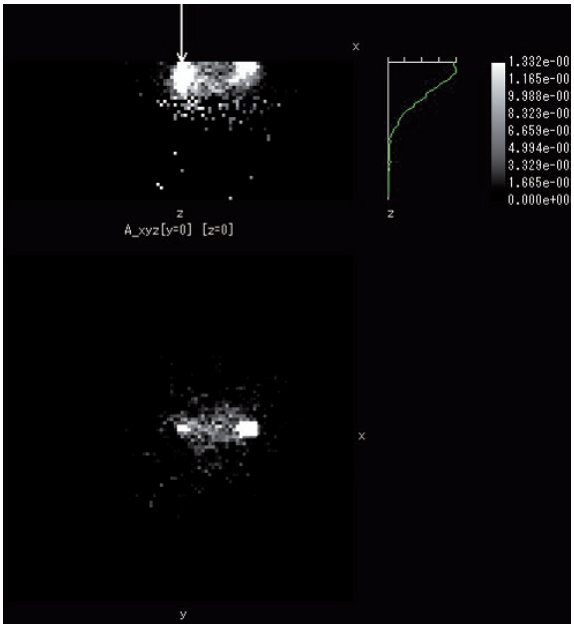

図11　光散乱シミュレーション結果

光子を一定数投入する場合と，検出器で光子を一定数検出するまで光子を投入し続ける場合について，今回実装したプログラムでシミュレーションした処理時間の結果を図12，表３，表４に示す。

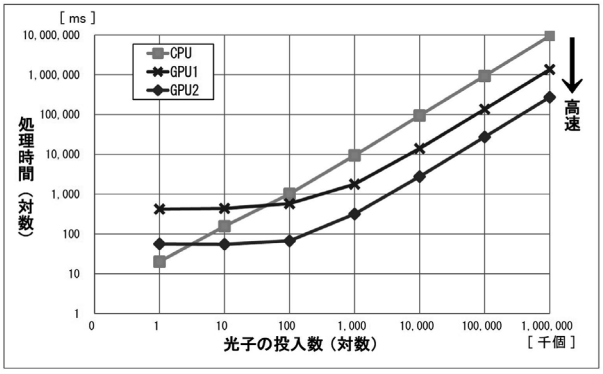

図12　光散乱シミュレーションの速度比較グラフ

表３　光散乱シミュレーションの速度比較

光子の投入数	CPU	GPU1		GPU2	
[個]	[ms]	[ms]	CPU比	[ms]	CPU比
1,000	20	421	0.05	56	0.36
10,000	156	437	0.36	55	2.84
100,000	1,014	578	1.75	67	15.13
1,000,000	9,391	1,778	5.28	317	29.62
10,000,000	94,616	14,009	6.75	2,770	34.16
100,000,000	938,279	136,048	6.90	27,349	34.31
1,000,000,000	9,422,591	1,357,405	6.94	273,103	34.50

表４　光散乱シミュレーションの速度比較
（検出器で一定数検出するまで光子を投入）

光子の検出数	CPU	GPU1		GPU2	
[個]	[s]	[s]	CPU比	[s]	CPU比
1,000	7.46	1.62	4.60	0.43	17.35
10,000	79.85	11.70	6.82	2.08	38.39
100,000	736.97	112.29	6.56	20.67	35.65
1,000,000	7,386.71	1,110.52	6.65	206.00	35.86

表中のCPUはIntel Core i7-950（動作周波数3.07GHz），GPU１はNVIDIA NVS3100M（演算コア16個），GPU２はNVIDIA Tesla C2050（演算コア448個）である。

投入する光子数が少ない場合，つまり計算量に比べてデータ数が少ない場合はGPUの方がCPUより処理時間がかかっている。これはGPUで計算する際に必要なメモリ転送のオーバーヘッドのためである。しかし，光子数が多くなるとCPUでの処理速度に比べGPUの方が速くなり，GPU１では約７倍，GPU２では約35倍の実行速度が得られ大幅な高速化がなされている。

5．おわりに

マルチコアCPUやヘテロジニアス構造の計算機における画像情報処理ソフトウェアを高速化するためのソフトウェア設計と実装について取り組み，移動体検出アプリケーションと光散乱シミュレーションソフトウェアの試作を通じてその効果を確認した。

これまでデータ量や反復計算が多くリアルタイムに処理できなかった画像処理やパターン認識，信号処理などに本研究成果を適用することで，より高度な計測システムの実現が可能となる。また，マルチコアCPUやヘテロジニアス構造の計算機に対する適切なソフトウェア設計手法に関する知見は，様々な産業分野に展開が見込まれ，今後は一次産品の加工工程や工場の生産ラインにおける計測システムなどの実用機に適用を図る予定である。

引用文献

1）堀武司・波通隆・飯島俊匡：“実環境における移動体の検出・追跡技術に関する研究”，北海道立工業試験場報告No.305，pp.9-15（2006）

2）飯島俊匡・髙橋裕之・橋場参生：“超解像処理を利用した画質改善技術に関する研究”，北海道立総合研究機構工業試験場報告No.309，pp.17-22（2010）

3）飯島俊匡・髙橋裕之：“積雪寒冷地用監視カメラシステムの構築”，北海道立総合研究機構工業試験場成果発表会要旨，pp.28（2011）

4）インテル，http://www.intel.com

5）アドバンスト・マイクロ・デバイセズ（AMD），http://www.amd.com

6）エヌビディア（NVIDIA），http://www.nvidia.com

7）“CUDA C Programming Guide”，“CUDA C Best Practices Guide”，http://developer.nvidia.com/nvidia-GPU-computing-documentation

8）“CUDA Zone”，http://developer.nvidia.com/category/zone/cuda-zone

9）“AMD Accelerated Parallel Processing（APP）SDK”，http://developer.amd.com/sdks/AMDAPPSDK

10）Lihong Wang，Steven L. Jacques，Liqiong Zheng：“MCML - Monte Carlo modeling of light transport in multi-layered tissues”，Computer Methods and Programs in Biomedicine 47，pp.131-146（1995）