B-034

CUDA によるフラクタル画像符号化実装の有効性評価

目出雅之†　田中宏二郎†　若谷彰良††
†甲南大学大学院　自然科学研究科
††甲南大学　知能情報学部　知能情報学科

１．はじめに

CPU の演算性能の向上は，数年前からスローダウンしつつあるのに対して，GPU はいまだに性能が向上し続けており，すでに CPU の汎用性能をはるかに超えている．

そこで，コンピュータの処理を高速に処理させる方法に，グラフィックボードに搭載されている GPU（Graphics Processing Unit）を用いた処理が注目されている．

フラクタル画像符号化は高い圧縮性能が得られるアルゴリズムであるだけでなく，画像検索のように他のアプリケーションにも応用されるものであるが [1]，その実行時間は長く，実用化にとっては高速化が不可欠である．そこで，本論文では，高速化を目的として，CUDA（Computer Unified Device Architecture）を実装し，品質の違いによる本手法の有効性の評価をおこなう [2].

２．CUDA

今回は GPU に幅広く使用されている NVIDIA 社の CUDA を用いて実験を行う．

特徴としては SIMD (Single Instruction Multiple Data) 型演算であるという点であり，これは１つの命令で複数の膨大なデータに対して，同じ処理する演算手法である．低いクロック周波数でも性能が高い点が長所であるが，分岐が多いなど複雑な制御構造をもつ処理では，アイドルになるプロセッサが発生するという点が短所である [2].

Evaluation of effectiveness of implementation of fractal image coding by CUDA

Masayuki Mede,Koujiro Tanaka,Akiyoshi Wakatani,
†School of Natural Science ,Konan University
††Faculty of Intelligence and Informatics,Konan University

３．フラクタル画像符号化

フラクタル画像符号化とは図形の一部分が他の一部分に似ていると仮定して圧縮する． また，反復によって極限画像に収束させることによって復号する．

フラクタル画像符号化は，膨大な量の符号化計算がある．また，符号化アルゴリズムはどのレンジブロックでも同じなので SIMD 型の CUDA は適している．しかし，レンジが固定されている場合は符号化するレンジブロックが連続しているので問題にならないが，最適な圧縮を行うにはレンジサイズを適応的に変える(今回は 16×16,8×8,4×4) 必要があり，符号化するレンジブロックが連続でなくなる可能性が高い．

そこで，提案手法ではあるレンジサイズで符号化されなかったレンジブロックは新たに間接ベクトルを用意し，連続に並ぶように並び替えをし，CUDA で効率良く計算できるようにする．一般に，間接ベクトルを用いたメモリアクセスではコアレスアクセスを阻害することになるので，性能に悪影響を与えるが，今回のアプリケーションでは，そのアクセスは多重ループの最内側ループ内には存在しないので，性能に与える影響は少ないと考えられる．

４．実験

4.1　実験内容

表 1 に示す実験環境において，横 512×縦 512 の lenna 画像(人物画)と boat 画像(風景画)を用いてフラクタル符号化し，その性能を比較評価する．比較を行う項目として一つ目は，符号化を行うか否かの判断を行う対象のレンジブロックを画質の指標である PSNR の dB 値を利用し画質が一定の閾値以上であれば符号化する．同一処理内では dB の値は一定とする．そしてプログラムを走らせる度に dB の値を変え，実行時間と見積もりとを比較し,オーバーヘッ

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

ドの割合とスピードアップの違いを調べる.

表 1　実験環境

OS	Windows 7
CPU	Atom330 (1.6GHz×2)
メモリ	2.00
GPU	ION(1.1GHz)
SPの数	2×16
CUDAのバージョン	3.0
開発環境	Visual Studio 2008

4.2 実験結果と考察

図 1 と図 2 で，どの dB においても，オーバーヘッド率が 5%未満で推移している.これは符号化にかかる処理時間に比べて，間接ベクトルを用いる事によって発生するオーバーヘッドの時間の割合が非常に小さい事を示している．ここでのオーバーヘッドとは間接ベクトルの生成，間接ベクトルへの値の代入，符号対象レンジブロックの符号化判断の処理である.よって，間接ベクトルを用いて並列処理を行う事は有効である事を示している.

また, PSNR の dB が上がるにつれてスピードアップが向上していることが実験で確認できた.これは dB が上がるにつれて符号化するレンジブロックに関して，よりサイズが小さいレンジブロックの数が増加しているためだと考えられる.つまり,今回の実装では shared memory サイズ上限により， 16×16 のレンジサイズでは並列効率が悪く, 16×16 のレンジサイズよりも 8×8 や 4×4 のレンジサイズ符号化処理の方が occupancy が高いため 16×16 のレンジサイズの処理時間が相対的に減っていったためだと考えられる.

２つの画像を比べると，スピードアップはどの dB でも boat 画像の方が高い事が確認できた. boat 画像の方が lenna 画像に比べてより小さなレンジサイズのレンジブロックの数が多いためだと考えられる.一般に人物画像は部分的に粗く，風景画は細かい事が反映されたものだと考えられる.

この 3 点から考えられる事は CUDA を利用する事での高速化は有効であるが，更なる高速化を望むにはレンジサイズが 16×16 のレンジブロックの数を減らすかレンジサイズが 16×16 の符号化手法を改善する必要がある.改善方法として shared memory の容量が 16KB であるという制限による occupancy の低さを改善する事が有効だと考えられる.

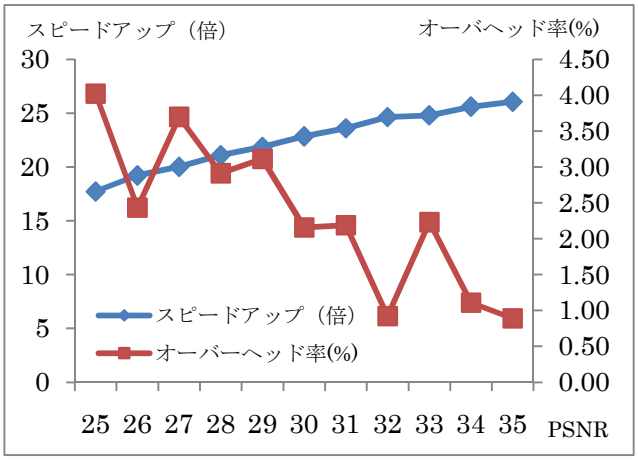

図 1 .lenna 画を用いた場合のオーバーヘッド率とスピードアップ率

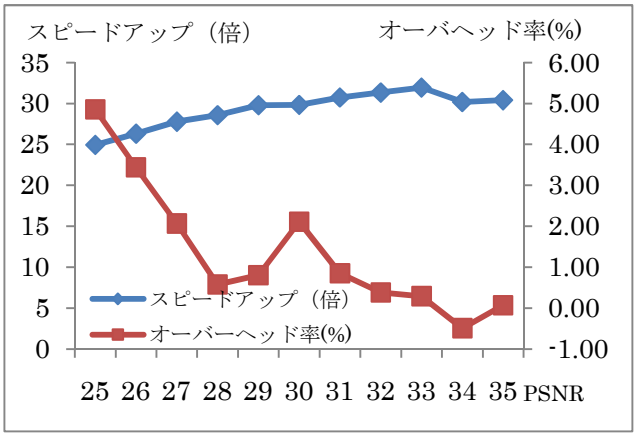

図 2 .boat 画を用いた場合のオーバーヘッド率とスピードアップ率

5. おわりに

今回の実験では間接ベクトルを用いた並列処理が有効である事が証明された．次は高速化手法を追求する．レンジブサイズが 16×16 の並列処理の有効性の指標である occupancy が 10%未満であり，高速化の妨げになっており，その点を改善していきたい.

参考文献

[1] 横山貫之，渡辺俊典，菅原研，上野芳朗, “フラクタル符号に基づく構造的な類似性抽出手法の提案,” 信学技報, PRMU2001-285, pp. 105-112, 2002.

[2] 青木尊之，額田彰, “はじめての CUDA プログラミング.” 株式会社工学社, 2009.

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.