2K-6

# 計算機ホログラム設計のGPUを用いた高速化

宮田 裕章 † 馬場 敬信 † 大津 金光 † 大川 猛 † 横田 隆史 †
† 宇都宮大学工学部情報工学科

## 1 はじめに

計算機ホログラム（CGH : Computer Generated Hologram）は, 情報処理や医療など様々な分野での実用化が期待されている. しかし, CGHの実用化にはいくつかの大きな課題があり, その一つがCGH設計にかかる時間が非常に長いことである. そのためCGH設計には高速な計算が必要不可欠になっている.

GPU（Graphics Processing Unit）は, 数百から千個単位の演算コアを有しており, データ並列性の高い処理においてはCPUの性能を遥かに上回る. 本研究では, CGH設計に関する計算の多くがデータ並列性を持っていることに注目し, CGH設計のGPUを用いた高速化手法を検討する.

## 2 計算機ホログラムの計算

通常のホログラムは, 記録物体からの光 (物体光) と参照光を記録媒体 (ホログラム) 上で干渉させることによって生成するが, 計算機ホログラムではその際の光の伝搬, 干渉, 回折といった計算を計算機上で行うことによって生成する.

本稿では, CGH設計に最適回転角（ORA : Optimal Rotation Angle）法を用いる [1, 2]. ORA法によるCGH設計は, 比較的良質なCGHを生成できるが, 回折点の数に比例して膨大な計算が必要になる. 回折点とは, 記録する物体を点の集合と考えた場合の1つ1つの点のことである. ORA法では最適化を行うために, 生成したCGHに最適回転角を適用しながら, より良質なCGHを生成する作業を繰り返し行うため, 最適化処理を行う回数も計算時間に大きな影響を与える.

$i$ 回目の最適化処理においてホログラム座標 $h$ におけるCGHの振幅 $a_h$, 位相 $\phi_h$ と, 回折点の座標 $r$ の複素振幅 $U_r^{(i)}$ の関係は式 (1) で表される.

$$U_r^{(i)} = w_r^{(i)} \sum_h a_h exp[i(\phi_{hr} + \phi_h^{(i)})] \qquad (1)$$

$w_r^{(i)}$ は回折強度を制御する重み, $\phi_{hr}$ はホログラム座標 $h$ から回折点 $r$ に寄与する位相であり, 式 (1) における振幅 $|U_r^{(i)}|$ が最大になるように式 (2) で表される最適回転角 $\Delta\phi_h^{(i)}$ を適用する.

$$\Delta\phi_h^{(i)} = \begin{cases} arctan(C_1/C_2) & (C_1 > 0) \\ arctan(C_1/C_2) + \pi & (C_1 < 0) \\ (\pi/2)sign(C_2) & (C_1 = 0) \end{cases} \qquad (2)$$

$C_1$, $C_2$ は次の式 (3) で表される.

$$C_1 = \sum_r a_h w_r^{(i)} cos\theta_{hr}$$
$$C_2 = \sum_r a_h w_r^{(i)} sin\theta_{hr} \qquad (3)$$

$\theta_{hr}$ は $\phi_{hr}$ と回折点の位相 $\phi_r$ から求められる.

## 3 CGH設計プログラムと並列化

ORA法による最適化は, あるCGHパターンとその再生像に依存して行う. そのためCGH設計プログラムでは初期データとして, ランダムな値を用いて計算に必要なCGHパターンとその再生像を生成しておく. その後, 最適化回数に依存した繰り返し処理を行う. 繰り返し処理中は1つ前のループで得られたCGHパターンと再生像の値から, 式 (1), (2), (3) により最適回転角の計算およびその適用を行い, 新たにCGHを生成する. 生成したCGHから回折計算により再生像を生成し, 再生像の点1つ1つの光強度を求める. 点全体の光強度の均一性を求め, 1つ前のCGHパターンのものと比較し, 均一性が高い方を保存し, 次の最適化処理で使用する. この処理を繰り返すことで最適化を行い, より良質なCGHを生成する.

逐次実行のプログラムを分析した結果, 式 (1), (2), (3) の計算部分で実行時間の94%〜99%を占めていることがわかった. この部分は繰り返し処理で計算されるが, この計算は以前の計算結果に依存しない. したがって, すべての計算を並列に処理できる. 先述したGPUは並列演算に適したアーキテクチャを有していることに注目し, 本研究ではGPUでこの計算部分の高速化を図る.

図1に逐次プログラムの並列化対象部と並列化後のプログラムのカーネル関数の概要を示す. CGHの画素数をN × N, 回折点数を $n_r$ とする. グリッドサイズをN × N, ブロックサイズを $n_r$ としてカーネル関数を起動し, 各スレッドで式 ( 3 ) の各項の計算を行う. その後, 各ブロック内ですべての項の総和計算を行なって $C_1$, $C_2$ を求める. これによってCGHの各画素に対する $C_1$ と $C_2$ が求められ, スレッドIDが0のスレッドで式 (2) による最適回転角の計算を行い, これを用いて式 (1) に対する最適回転角の適用を行う.

## 4 カーネル関数の最適化

### 4.1 総和計算の最適化

並列化後のプログラムにおいて, $C_1, C_2$ の総和計算を行なっている部分のリダクション計算は, 偶数番号スレッドが隣のスレッドの値を自分に足し込むという処理を繰り返すことで総和を求めている [3]. このとき

Speed-up of Computer Generated Hologram Design by Using GPU
†Hiroaki Miyata, Takanobu Baba, Kanemitsu Ootsu, Takeshi Ohkawa and Takashi Yokota
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

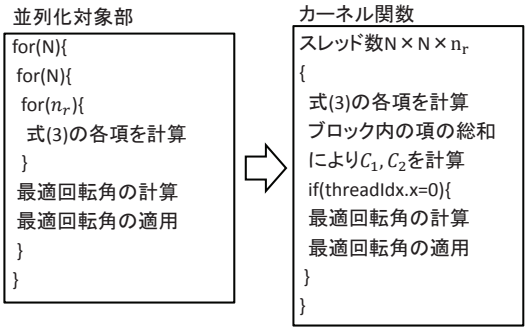


図 1: 並列化部のプログラム概要

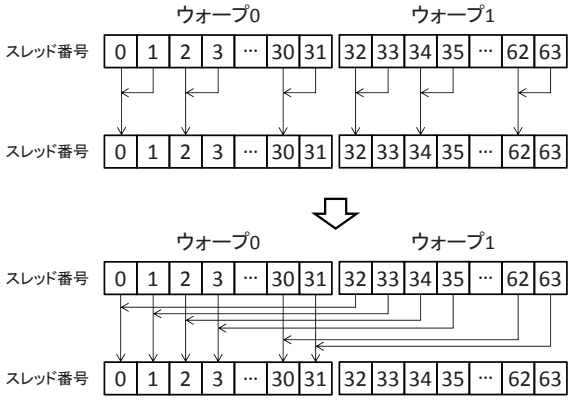


図 2: 総和計算の最適化

計算を行うのは偶数番号のスレッドで, 奇数番号のスレッドは何もしていない. したがって, ウォープ内で違う処理を行なっていることになるため, ウォープダイバージェントが発生し, 性能低下の原因になる. そこで, ブロック内のスレッドを前半と後半に分割し, 前半のスレッドのみが計算を行うようにすることで, ウォープ内の処理が統一され, 分岐を減らすことができる. 総和計算の最適化の概略を図 2 に示す.

また, この方法を用いても, 計算を行うスレッド数が 32 以下になると, 1 つのウォープ内で処理の分割をしなければならないのでウォープダイバージェントが発生してしまう. ウォープ内のスレッドは全て同期処理されるので, ループアンローリングを行い, 全てのスレッドに同じ処理をさせることで分岐が発生しないようにした.

### 4.2 ブロックサイズの最適化

ブロックサイズを変更すると実行時間に影響が生じる. そこで, 今まで 1 つのブロックに回折点数分 ($n_r$ 個) のスレッドを確保していたが, スレッドを複数ブロックに分けて確保することでブロックサイズを可変にした.

ブロックサイズは使用している GPU の SM(Streaming Multiprocessor) 内の SP(Streaming Processor) 数の倍数にするのが望ましい. そうすることで, 何も処理しない SP の数が減って処理効率が良くなるためである. 今回使用している GPU の SM あたりの SP 数が 32 であり, このプログラムではスレッドの数を回折点数に依存するように設計したので, 回折点数も 32 の倍数に設定することで処理効率を上げた. そのため, 1 ブロックあたりのスレッド数を 32, 64, 128, 256, 512 として性能評価を行った. その結果, スレッド数が 256 のときに最も良い性能となったため, 実行結果はスレッド数を 256 に固定して評価した.

表 1: 最適化後のプログラムの実効性能

| 回折点数 | 全体 | カーネル関数部 |
|---|---|---|
| 128 | 6.05s(15.8 倍) | 2.5s(37.5 倍) |
| 512 | 12.6s(29.7 倍) | 9.06s(41.0 倍) |
| 1024 | 20.9s(35.6 倍) | 17.5s(42.6 倍) |

## 5 実行結果

実行環境について, CPU は Intel Core i3 3220(動作周波数 3.30GHz), GPU は Tesla C2075(コア数 448, 動作周波数 1.15GHz) を用いた. 表 1 に最適化後のプログラムの実行性能を示した. CGH の画素数は 512 × 512, 最適化回数は 20 回で固定している. 回折点はランダムに配置し, 各回折点数における全体の実行時間 (秒) と, カーネル関数部分にかかった時間 (秒) を示した. 括弧内の数字は CPU のみで計算した場合の実行時間との速度向上比である.

回折点が増えると速度向上も大きくなり, 回折点数が 1024 のとき 35.6 倍の高速化が実現できた. 並列化部分だけを見ると 42.6 倍の速度向上が得られた.

## 6 おわりに

本稿では ORA 法による CGH 設計の GPU を用いた高速化を検討した. GPU アーキテクチャに合わせたプログラミングをすることで最適化を行い, 逐次処理プログラムと比較して, 最大で 35.6 倍の高速化を達成した.

**謝辞**

本研究について貴重なご意見を頂いた本学オプティクス教育研究センター 早崎芳夫教授, 大学院生 鈴木大地氏に感謝する.

本研究は, 一部日本学術振興会科学研究費補助金 (基盤研究 (C)24500055, 同 (C)24500054) の援助による.

## 参考文献

[1] Jorgen Bengtsson: “Kinoform design with an optimal-rotation-angle method”, APPLIED OPTICS, Vol.33, No.29, pp.6879-6884, 1994.

[2] Daichi Suzuki: “Optimization of a computer-generated hologram with GPU for holographic femtodecond laser processing”, International Workshop Holography and Related Technologies 2011 (IWH 2011), P30, pp.81-82, November 16-17, 2011.

[3] 青木 尊之, 額田 彰: “はじめての CUDA プログラミング”, 工学社, p.247, 2009.