B-036

GPU を用いた波面記録法による計算機合成ホログラムの作成

Making Computer Generated hologram by wavefront-recording method using GPU

岡田 直久 † 西辻 崇 † 老川 稔 † 杉山 充 †

NAOHISA OKADA TAKASI NISITUJI MINORU OIKAWA MITURU SUGIYAMA

下馬場 朋禄 † 増田 信之 † 伊藤 智義 †

TOMOYOSI SIMOBABA NOBUYUKI MASUDA TOMOYOSI ITO

1. はじめに

ホログラフィ(Holography)は，物体の放つ光を忠実に再現する究極の三次元動画像表示技術として知られている．電子ホログラフィでは，コンピュータによって作られた計算機合成ホログラム(CGH ：Computer Generated Hologram)を利用して三次元像を表示することができる．しかし現時点では，ホログラムの作成に膨大な時間を要することや，視野角を広げるためにもっと高精細なディスプレイが必要であるといった問題がある．

一方，GPU(Graphics Processing Unit)は近年演算能力が著しく向上しており，さらに内部処理がプログラミング可能になったことから，汎用的な演算処理に応用されるようになってきた．また，2007 年に NVIDIA 社から提供された総合開発環境 CUDA(Compute Unified Device Architecture)によって，グラフィックス処理に関する知識を必要とせずに GPU プログラミングをすることができるようになった．

本研究では CGH 作成における計算量削減を行うためのアルゴリズムとして波面記録法に着目し，GPU における波面記録法の実装を行い，CGH 作成の有効性の検証を行った．

2. 波面記録法

2.1 従来手法による CGH 計算

CGH の干渉縞は，物体の座標から，以下の式で求めることができる．

$$I(x_\alpha, y_\alpha) = \sum_j^N \frac{A_j}{R} \cos(\frac{2\pi}{\lambda} R) \quad (1)$$

I はホログラム上の点(x_α, y_α)における光強度を，R は物体のある 1 点(x_j、y_j、z_j)とホログラムの距離を表している．また，λは参照光波長である．物体点が N 個ある場合には，式(1)を j について N 回足し合わせればよい．

2.2 波面記録法による CGH 計算

波面記録法では 2 ステップの計算を行なうことで計算量の削減をした CGH の作成を可能としている．

波面記録法の第 1 ステップ(以下 STEP1)として物体点と波面記録面の間でレイトレーシングを行い波面記録面に物体点の複素振幅の記録を行う．振幅の記録は以下の方程式を用い，三次元物体におけるすべての物体点での複素振幅 u_w の記録を行う．

$$u_w(x_w, y_w) = \sum_j^N \frac{A_j}{R_{wj}} \exp(i\frac{2\pi}{\lambda} R_{wj}) \quad (2)$$

このとき，$R_{wj} = \sqrt{(x_w - x_j)^2 + (y_w - y_j)^2 + d_j^2}$ は物体点と波面記録面上の座標(x_w,y_w)との距離である．また，(x_j,y_j,z_j)および A_j は CGH 上に原点を置いた場合の j 番目の物体点の座標および輝度，λは参照光の波長であり，N は物体点の総数，$d_j = z_j - z$ は波面記録面と物体点との垂直距離，そして $i = \sqrt{-1}$ である．波面記録面を物体の近くに置いたとき，物体光が波面記録面を通過する領域は微小となる(このときの領域の半径を W_j とする)．このことから，式(2)における計算量が削減されることとなる．

第 2 ステップ(以下 STEP2)では波面記録面と CGH 間で回折計算を行うことで CGH 上の複素振幅 $u(x, y)$ を計算する．波面記録面には物体光の振幅情報と位相情報が記録されているため，回折計算は物体から CGH 上の複素振幅を計算する場合と等しくなる．ここではフレネル回折計算を用いる

$$u(x,y) = \frac{\exp(i\frac{2\pi}{\lambda} z)}{i\lambda z} \iint u_w(x_w, y_w) \exp(i\frac{\pi}{\lambda z}((x - x_w)^2 + (y - y_w)^2)) dx_w dy_w$$

$$= \frac{\exp(i\frac{2\pi}{\lambda} z)}{i\lambda z} F^{-1}[F[u_w(x,y)] \cdot F[h(x,y)]] \quad (3)$$

ここで，$F[\cdot]$ と $F^{-1}[\cdot]$ はそれぞれフーリエ変換と逆フーリエ変換の演算子，z は波面記録面と CGH との垂直距離，$h(x,y) = \exp(i\frac{\pi}{\lambda z}(x^2 + y^2))$ はインパルス応答である．

最後に，光伝播計算の結果と参照光を干渉させ CGH の計算を行う．

3. 波面記録法の GPU への実装

本研究の実装環境を表 3.1 に示す．

表 3.1：実装環境

OS	Windows XP
CPU	Xeon 2.83GHz
GPU	GeForce GTX 480
プロセッサ数	480
シェーダクロック	1.401GHz
CUDA SDK	version 3.2

波面記録法を実装するに当たって 2 通りの並列化方法を考案し実装を行った．

3.1 画素による並列化

波面記録面の 1 画素の計算を GPU のスレッド 1 つに割り当てて計算を行う方法である．スレッドの画素への対応イメージを図 3.1 に示す．

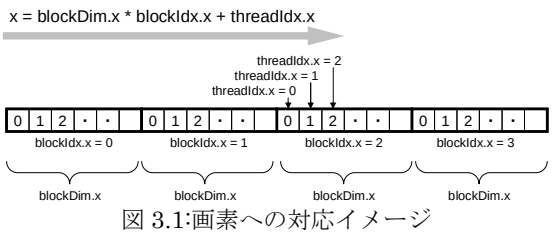

図 3.1:画素への対応イメージ

† 千葉大学

しかし，子の方法では全物体点に対して各画素で物体光が通過するかしないかの判定を行わねばならず．計算量の削減にはつながり難い．

そこで複数画素を 1 つの領域として設定しその中の代表点において物体光が通過の判定を行うこととし．その領域内で計算を行う物体点を事前に判定することで計算時間の削減を図った．

3.2 物体点による並列化

GPU に波面記録面の計算を実装する 2 つ目の方法として物体点を各スレッドに割り当てることで実装することができる．この方法では各スレッド内で N 番目の物体点の物体光が通過する波面記録面上の画素の領域 W_j を計算により導出し，その領域内の複素振幅 u_w を計算し最後に他の物体点で導出された複素振幅 u_w と統合することで波面記録面を計算するという方法である．

物体点で並列化をした際の問題の一つに物体点数が物体によって変化し，分割が出来ない場合があるということがある．今回は物体点数 N を $N \approx 2^{n-1}$ (ただし $2^{n-1} \geq N$ となる n である)として分割が容易に可能となるようにした．このとき N 番目以上のスレッドでは計算を行わない．また，この実装方法ではアルゴリズムやメモリ量の関係から CUDA での一般的な高速化手法である shared memory の使用やループアンローリング等が使用できない．

4. 計算時間の比較

本研究の実験条件を表 4.1 に示す．

表 4.1：実験条件

波面記録面と CGH 間の垂直距離	300mm
参照光波長	532nm
画素間隔	8 μ m
ホログラムサイズ	1,024×1,024

使用した物体点は各点の座標が$(x_j,y_j,1)$となる平面の物体を用いた．

4.1 計算時間比較(STEP1)

物体から波面記録面までの距離を一定($2.0\times10^{-3}m$)としたとき，物体点数を変化させた場合の計算時間を表 4.2 に示す．

表 4.2：物体から波面記録面までの距離を一定としたときの計算時間

物体点数	512	1,024	2,048	4,096	8,192
CPU (ms)	10	18	38	74	156
GPU (ms) 画素による並列化	5	7	12	21	39
GPU (ms) 物体点による並列化	2	2	2	2	3

物体点数 8,192 点では画素による並列化では CPU に比べ約 4 倍の高速化，物体点による並列化では CPU に比べ約 50 倍の高速化がなされている．画素による並列化では物体点の増加に伴い計算時間が増加している．この理由としては物体点に対して判定を行っているため物体点数が処理時間に影響を与えていると考えられる．一方，物体点による並列化では物体点の増加に対して処理時間の増加が比較的緩やかである．この理由としては物体点の座標データを GPU に転送する等の処理は物体点数に影響を受けているが波面記録面計算の処理時間には物体点数の影響が小さいためと考えられる．

次に，物体点数を 8,192 点としたとき，物体から波面記録面までの距離を変化させた場合の計算時間を表 4.3 に示す．

表 4.3：物体点数を一定としたときの計算時間

物体から波面記録面までの距離(m)	2.0×10^{-3}	1.0×10^{-2}	2.0×10^{-2}
CPU (ms)	156	423	1,780
GPU (ms) 画素による並列化	39	44	70
GPU (ms) 物体点による並列化	3	64	256

波面記録法では物体近傍に波面記録面を配置することで物体から出た光が当たる領域を微小とすることで計算時間を削減している．そのため，距離が離れると領域が大きくなり計算時間の削減効果が薄くなる．表 4.3 からもそのことが見て取れる．物体点による並列化で計算時間の増加が顕著なのはスレッド内で領域内全ての計算を行っているため計算領域の増加に伴い 1 スレッドでの計算時間が増えるためと考えられる．一方，画素による並列化で比較的変化が緩やかである理由としては，光が当たる領域が増加したとしても画素によって並列化しているため影響を受けにくいためであると考えられる．

4.2 計算時間比較(STEP2)

波面記録面から CGH への回折計算および CGH の計算における処理時間は表 4.4 のようになった．GPU に実装する際には GWO(GPU-based Wave Optics)ライブラリを用いた．この GWO ライブラリは GPU を用いた回折計算を行うための数値計算ライブラリである．

表 4.4：STEP2 における処理時間

物体点数	512	1,024	2,048	4,096	8,192
CPU (ms)	455	452	456	456	458
GPU (ms)	84	83	83	83	84

表 4.4 より GPU では CPU に比べて物体点数 8192 点のとき約 5 倍の高速化が行われていることが見て取れる．また，物体点数が増加しても処理時間に大きな変化が見られない．これは波面記録面から CGH への回折計算および CGH の計算の処理時間がホログラムサイズに依存しているためと考えられる．

5. まとめと今後の展望

本研究では CGH 作成における計算量削減を行うためのアルゴリズムとして波面記録法に着目し，GPU における波面記録法の実装を行い，CGH 作成の有効性の検証を行った．

波面記録面計算では，2 通りの並列化方法によって GPU に実装を行った．物体点による並列化では CPU に比べて約 50 倍の，画素による並列化では約 4 倍の高速化を行うことに成功した．また，物体点による並列化を行った場合では物体点の増加に対して処理時間の増加を抑えられることが，画素による並列化では物体から波面記録面までの距離が増加した時の処理時間の増加が物体点による並列化に対して少ないことが分かった．波面記録面から CGH への回折計算および CGH の計算では GPU は CPU に対して約 5 倍の高速化を行なうことに成功した．これらの研究結果から GPU を用いた波面記録法による CGH 計算は有効であるといえる．

今後の展望としては，新しい GPU およびアーキテクチャに沿ったプログラムを行なうことによる高速化，GPU クラスタに実装することによる高速化などが期待できる．