H-013

# 光線サンプリング面を用いたカラーホログラム計算のマルチ GPU による高速化
# Speeding up of color-hologram generation based on a ray-sampling plane by multi-GPU

佐藤 大愛†,‡　角江 崇†　涌波 光喜‡　市橋 保之‡　大井 隆太郎‡　山本 健詞‡
下馬場 朋禄†　伊藤 智義†
Hirochika Sato　Takashi Kakue　Koki Wakunami　Yasuyuki Ichihashi　Ryutaro Oi　Kenji Yamamoto
Tomoyoshi Shimobaba　Tomoyoshi Ito

## 1. はじめに

近年，3D テレビを実現するための技術として電子ホログラフィが注目されている．電子ホログラフィは，ホログラムを空間光変調器(Spatial Light Modulator: SLM)で描画し，3 次元画像を再生する技術である[1]．この技術は，表示するホログラムを切り替えることによって 3 次元動画像の再生を可能にする．これまで，多くのホログラムの計算手法が提案されており，その中でも，光線ベースの手法を用いたものが提案されている．光線ベースの手法はオクルージョンや光沢を再現しやすく，実物体や 3D ポリゴンのような仮想物体に応用しやすいのが特徴である．一方で，光線をサンプリングした際に，サンプリング面から遠い場所にある物体がぼけてしまう問題点があった．そこで涌波らは，物体近傍に RS 面という仮想の平面を，ホログラムを計算する面とは別に用意することでこの問題を解決した[2]．しかし，この手法は多数の要素画像のレンダリングや，2 次元フーリエ変換を用いた光線から波面への変換，RS 面からホログラム面への伝搬計算が必要なため計算量が多く，リアルタイムに 3 次元動画像を再生するのは困難であった．そこで著者らは先行研究において，1 枚の GPU(Graphic Processing Unit)を用いてホログラム計算を並列化し，単色のホログラムの計算を高速化した[3]．しかし，カラーホログラムの計算は高速化できておらず，カラーの 3 次元動画像を実現するためには更なる高速化が必要である．本研究では，カラーホログラムの計算を，複数の GPU を用いて高速化し，リアルタイムの 3 次元動画像再生を可能にすることを目的とする．

## 2. 光線サンプリング面を用いたホログラム計算

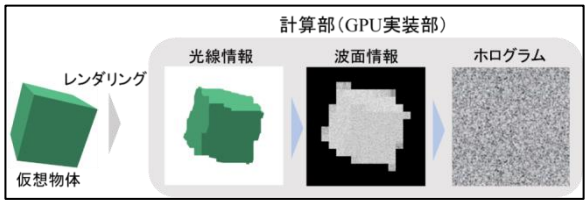


図 1　ホログラム計算の手順

### 2.1 光線情報の取得

図 1 に，RS 面を用いたホログラム計算の手順を示す．まず，図 2 のようにホログラム面とは別に，物体近傍に仮想平面である RS 面を配置し，RS 面上で光線情報のサンプリングを行う．本研究では，3DCG ソフトである Blender[4]を用い，仮想のカメラアレイを RS 面上に置き，仮想物体に対してレンダリングを行うことで光線情報を取得した．こ

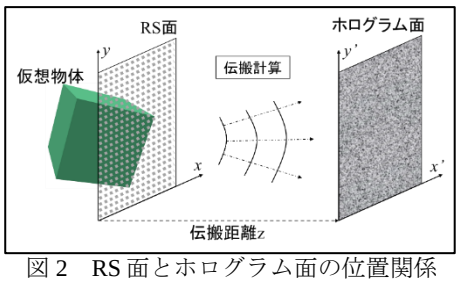


図 2　RS 面とホログラム面の位置関係

のとき，レンダリングで取得した一枚一枚の画像を要素画像と呼び，それらすべてを一つにまとめたものを RS 画像と呼ぶ．

### 2.2 波面情報の取得

ホログラム面上の波面情報を得るためには RS 面からホログラム面へ伝搬計算をする必要があるが，伝搬計算のためには RS 面上の波面情報が必要である．そこで，前節で取得した光線情報を波面情報へ変換する．ここで，$m, n$ を元の要素画像の画素の座標，$k, l$ を波面に変換後の画素の座標，$i, j$ を要素画像の座標とすると，波面情報 $P_{ij}$ は光線情報 $p_{ij}$ および FFT(Fast Fourier-Transform)を用いて，

$$P_{ij}[k,l] = \mathrm{FFT}\{p_{ij}[m,n]\exp\{j\varphi_{ij}[m,n]\}\}. \quad (2\text{-}1)$$

と表される．$\varphi_{ij}[m,n]$は RS 面における波面の位相であり，本研究では光を拡散させるためにランダムな値を使用した．

### 2.3 伝搬計算

前節で取得した RS 面上の波面情報を用いてホログラム面への光の伝搬計算を行い，ホログラム面上の波面情報を取得する．伝搬計算には式(2-1)に示す畳み込み式のフレネル回折計算式を用いた．

$$u(X,Y) = \mathcal{F}^{-1}\{\mathcal{F}\{g(x,y)\}\cdot\mathcal{F}\{h(x,y)\}\},$$
$$h(x,y) = \frac{1}{j\lambda z}\exp\{-\frac{jk}{2z}((X-x)^2+(Y-y)^2+2z^2)\}. \quad (2\text{-}2)$$

ここで，$x, y$ および $X, Y$ は各平面上の座標であり，$z$ は伝搬距離である．この式から，ホログラム面上の波面情報 $W_H(k_H, l_H)$は，RS 面上の波面情報$W_{RS}(k_{RS}, l_{RS})$ を 用いて，

$$W_H(k_H,l_H) = \mathcal{F}^{-1}\{\mathcal{F}\{W_{RS}(k_{RS},l_{RS})\}\cdot \mathcal{F}\{h(x,y)\}\}, \quad (2\text{-}3)$$

と表される．

### 2.4 キノフォーム型ホログラムの計算

本研究では，ホログラムの再生には位相変調型 SLM を用いたため，キノフォーム型のホログラム計算式を用いた．

† 千葉大学, Chiba University
‡ 情報通信研究機構, NICT

このとき，ホログラムの画素$H[k_H, l_H]$は，ホログラム面の波面情報$W_H(k_H, l_H)$を用いて，次式で表される．

$$H(k_H, l_H) = \arg(W_H) = \tan^{-1}\frac{\mathrm{Im}(W_H)}{\mathrm{Re}(W_H)}. \quad (2\text{-}4)$$

## 3. GPU での実装

先行研究では 1 台の GPU を用いて，単色のホログラムの計算を高速化した[3]が，本研究では 3 台の GPU を用いて，カラーホログラムの計算を高速化した．各 GPU で青色，緑色，赤色のホログラムを計算し，リアルタイムに 3 次元動画像を再生することを目的とした．GPU を用いたプログラミングの開発環境として CUDA を使用した[5].

### 3.1 波面情報の取得

波面情報の取得には，2 次元フーリエ変換を，要素画像の数と同じ回数行う必要がある．そこで，本研究では CUDA 内のライブラリである cuFFT ライブラリの中の cufftPlanMany 関数を用いて，多数の 2 次元フーリエ変換を並列化して高速化を図った．また，ランダム位相に関しても，CUDA のライブラリである cuRAND ライブラリを用いた．

### 3.2 伝搬計算

本研究で使用した畳み込み式のフレネル回折計算式は，2 次元フーリエ変換のみで表される．このうち$h(x, y)$に関しては波長及び伝搬距離が不変であれば定数とみなして良いので，各 GPU に対応した色の$\mathcal{F}\{h(x, y)\}$の値を事前に計算しておき，各 GPU のメモリに保存しておくことで，伝搬計算の時間を短縮した．それ以外の計算に関しては，前節と同じく cuFFT ライブラリを用いて高速化した． また，FFT をかける際には，FFT の計算効率を上げるためと，高次の像が重ならないようにするため，ゼロパディングを行い，画像サイズを拡大している．

### 3.3 ホログラムの計算

ホログラムの計算は画素ごとに計算が独立しているので，画素の数と同じ数のスレッドを GPU 内に立てて，並列に計算を行った．

## 4. 結果

表 1 に本研究で用いた実装環境を，表 2 に計算および再生に用いたパラメータを示す．表 3 に 1 フレームのホログラム計算にかかった時間を示す．表 3 における転送時間 HtoD はホスト PC からデバイス(GPU)への転送時間，DtoH はデバイスからホストへの転送時間である．

表 1　実装環境

| | |
|---|---|
| OS | Windows 10 |
| CPU | Intel(R) Core(TM) i7-6800K CPU |
| メモリ | 32.00GB |
| 使用コア | 12 |
| コンパイラ | Microsoft Visual C++ 2013 |
| GPU | NVIDIA Geforce GTX 1080×3 |
| GPU クロック | 1,823MHz |
| メモリクロック | 5,005MHz |
| メモリ | 8,192MB |
| コア数 | 2,560 |
| CUDA | Version 8.0 |

表 2　計算パラメータ

| | |
|---|---|
| RS 画像のサイズ[pixel] | 3,072×3,072 |
| FFT 実行時のサイズ[pixel] | 4,096×4,096 |
| ホログラムのサイズ [pixel] | 1,920×1,080 |
| 画素ピッチ [μm] | 8.0 |
| 波長（青） [nm] | 450 |
| 波長（緑） [nm] | 532 |
| 波長（赤） [nm] | 650 |
| 伝搬距離 [m] | 0.5 |

表 3　計算時間

| プロセッサ | CPU | GPU |
|---|---|---|
| 転送時間 HtoD[s] | | 0.003 |
| 波面情報の取得[s] | 2.04 | 0.004 |
| 伝搬計算[s] | 7.32 | 0.009 |
| ホログラムの画素計算[s] | 0.36 | 0.002 |
| 転送時間 DtoH[s] | | 0.001 |
| 描画時間[s] | 0.012 | |
| 合計時間[s] | 9.84 | 0.031 |

表 3 を見てわかる通り，本研究では GPU による並列計算を行うことで，CPU に比べてカラーホログラムの計算を約 300 倍高速化することに成功した．また，実際に光学系を用いて 3 次元動画像を再生し，約 30[fps]で 3 次元動画像が再生できることを確認した．

## 5. 結論

本研究では，RS 面を用いたホログラム計算手法を GPU で実装し，カラーホログラムの計算を CPU のみで計算した場合よりも約 300 倍の高速化に成功した．また，実際に光学系を用いて，約 30[fps]でカラー3 次元動画像を再生できたことを確認した．

今後の課題として，今回は描画のためにホログラムのデータを一度ホストに転送しているが，ホログラムが大きくなった場合に転送時間 DtoH が増加してボトルネックになる可能性があるため，これを防ぐために GPU から直接 SLM にホログラムを描画することなどが挙げられる．

### 参考文献

[1] P. S. Hilaire, S. A. Benton, M. Lucente, M. L. Jepsen, J. Kollin, H. Yoshikawa, and J. Underkoffler: “Electronic display system for computational holography,” Proc. SPIE, 1212-20, pp. 174-182 (2008).

[2] K. Wakunami, and M. Yamaguchi: “Calculation for computer generated hologram using ray-sampling plane.” Optics Express, Vol.19, No.10, pp. 9086-9101 (2011).

[3] H. Sato, T. Kakue, K. Wakunami, Y. Ichihashi, R. Oi, K. Yamamoto, T. Shimobaba, T. Ito: “Acceleration of hologram generation using ray-sampling plane by GPU,” The 8th International Conference on 3D systems and Applications (3DSA 2016), 3DSA 1-6, (7-9, Dec. 2016).

[4] “Blender,” https://www.blender.org/

[5] “CUDA Toolkit Documentation - NVIDIA Documentation,” http://docs.nvidia.com/cuda/