

AMD GPU と OpenCL を用いた波面記録法の実装 Computer Generated hologram by wavefront-recording method using AMD GPU and OpenCL

松戸 悠亮† 下馬場 朋禄† 西辻 崇†
Yuusuke Matsudo Tomoyoshi Shimobaba Takashi Nishitsuji

岡田 直久† 翁 剣同† 角江 崇† 増田 信之† 伊藤 智義†
Naohisa Okada Jiantong Weng Takashi Kakue Nobuyuki Masuda Ito Tomoyoshi

1. まえがき

物体の 3 次元情報を記録するホログラムを計算機上で作成したものとして CGH(Computer Generated Hologram)がある。CGH は、3 次元動画像技術を構成する重要な要素となっている[1]。しかし、CGH を作成するには膨大な計算量が必要となるという問題がある[2]。この問題を解決するための CGH 計算アルゴリズムとして波面記録法がある[3-6]。NVIDIA GPU と CUDA を用いた波面記録法の実装の事例は既にある[4,6]。近年、並列計算標準言語である OpenCL が発表された。OpenCL は、動作するデバイスの制限が少なく、AMD GPU や CPU 上でも並列計算が可能となっている。そこで、本研究では AMD GPU と OpenCL を用いて波面記録法の実装を行い、その有効性を検証した。また、波面記録法の手法の改良を行い、再生像のシミュレーション結果の改善を行った。

2. 波面記録法

2.1 直接計算を用いた CGH 計算

物体を点光源の集まりと考えた場合、CGH 面 (x_c, y_c) での複素振幅 $u_c(x_c, y_c)$ は以下の式で求められる。

$$u_c(x_c, y_c) = \sum_j^N \exp\left(\frac{2\pi}{\lambda} (\sqrt{(x_c - x_j)^2 + (y_c - y_j)^2 + z_j^2})\right) \quad (1)$$

ここで、 N は物体点の総数、 λ は光の波長、 (x_j, y_j, z_j) は各物体点の座標となっている。式(1)は直接計算と呼ばれる。

直接計算を用いる場合、計算回数は CGH 面の大きさと物体点総数 N を掛けた数となり、例えば CGH のサイズが $2,048 \times 2,048$ で、物体点総数が $N = 11,646$ 点の場合、計算回数は約 48×10^9 回必要となる。

この CGH 作成の計算時間を解消するアルゴリズムとして波面記録法がある。

2.2 波面記録法のアロリズム

波面記録法は図 1 に示すように、物体点の間に仮想的な面である波面記録面を配置することで計算量を削減するアルゴリズムである。波面記録法は、2 つのステップを踏んで CGH 面での複素振幅の計算を行う。まず、ステップ 1 では物体点からの光の波面記録面 $u_w(x_w, y_w)$ での複素振幅の計算を行う。この計算は先ほどの式(1)の計算と同様の計算であり、変化するのは、 z_j が $z_j - d$ となる部

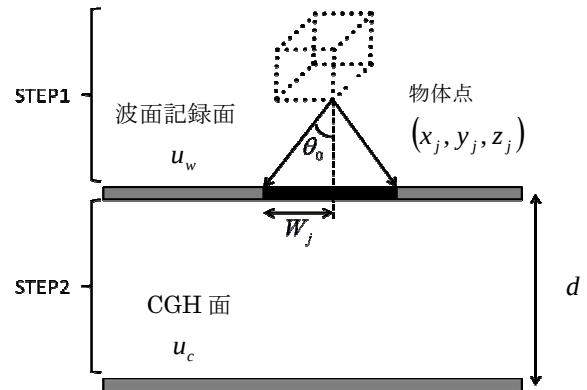


図 1 波面記録法の概略図

分だけであり、以下の計算式で計算を行う。

$$u_w(x_w, y_w) = \sum_j^N \exp\left(\frac{2\pi}{\lambda} (\sqrt{(x_w - x_j)^2 + (y_w - y_j)^2 + (z_j - d)^2})\right) \quad (2)$$

物体近傍に波面記録面を配置することにより、各物体点からの光が通過する領域を制限することができるので、直接計算の場合と異なり、ホログラムの各画素がすべての物体点について計算を行う必要がなくなり、計算時間の短縮が見込まれる。

各物体点が通過する領域の半径 W_j は、

$$W_j = (z_j - d) \tan(\theta_0) \quad (3)$$

で与えられる。ここで、 d は波面記録面と CGH 面の距離、 θ_0 は波面記録面に対する垂線と物体光のなす角度で、その最大角は、

$$\theta_0 = \sin^{-1}(\lambda / 2p) \quad (4)$$

で与えられる。ここで、 p は波面記録面と CGH 面の画素ピッチである。

次に、ステップ 2 では、波面記録面での複素振幅情報を用いて、フレネル回折計算を行い CGH 面での複素振幅 $u_c(x_c, y_c)$ を求める。

† 千葉大学大学院工学研究科

$$\begin{aligned}
 u_c(x,y) &= \frac{\exp(i\frac{2\pi}{\lambda}z)}{i\lambda z} \iint u_w(x_w,y_w) \exp(i\frac{\pi}{\lambda z}((x_c-x_w)^2+(y_c-y_w)^2)) dx_w dy_w \\
 &= \frac{\exp(i\frac{2\pi}{\lambda}z)}{i\lambda z} F^{-1}[F[u_w(x_w,y_w)] \cdot F[h(x_w,y_w)]]
 \end{aligned}
 \quad (5)$$

フレネル回折計算は、式(5)から分かるようにフーリエ変換の形で表すことができる。よって、高速フーリエ変換(FFT)を用いることで、物体の点数に依存せずに、CGHの大きさのみに依存した計算時間となる。

3. 通過領域判定の手法の改善

3.1 従来の通過領域判定方法

物体からの光が、波面記録面のどの領域を通過するか判定する方法としては、波面記録面の全画素について判定を行う方法が考えられるが、判定に大きな時間がかかってしまう。

そこで、従来の通過領域の判定は、波面記録面の全ての画素について通過の判定を行うのではなく、図2に示すように、波面記録面の複数の画素を集めてブロックを作成し、ブロックの代表点で判定を行うことで条件判定の負荷を減らしていた。判定では、領域の半径 W_j と、物体点の x - y 座標 (x_j, y_j) と各ブロックの代表点の座標の距離 R を比較して、 $W_j \geq R$ であれば通過すると判定する。

しかし、この判定方法では図3のように実際には光が通過しない領域も計算してしまう問題がある。

ステップ2のフレネル回折計算はFFTを用いて計算を行うが、波面記録面の情報を入力としてFFTで信号処理を行っていると見ることができる。信号処理として見た場合、図3のような判定方法では、実際には通過しない領域の値は雑音となっていると考えられる。

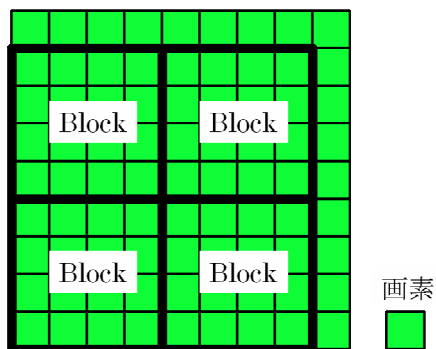


図2 波面記録面の画素のブロック化

3.2 今回の通過領域判定方法

本研究で用いる新しい通過領域判定の方法を以下に示す。まず、従来と同様に画素を集めてブロックを作成する。次に、通過領域に内接する正方形を考え、この正方形を新たな通過領域とする。図4では円に内接している正方形が対応している。

この判定方法を用いることで、実際には通過しない領域の計算を減らすことができる。また、領域が長方形となっているため、通過領域判定において条件判断をなくすることができる。

今回用いた方法では、図3(従来の手法)と図4(今回用いた手法)を比較すると円の外部の領域を計算している量が大幅に減少していることが分かる。

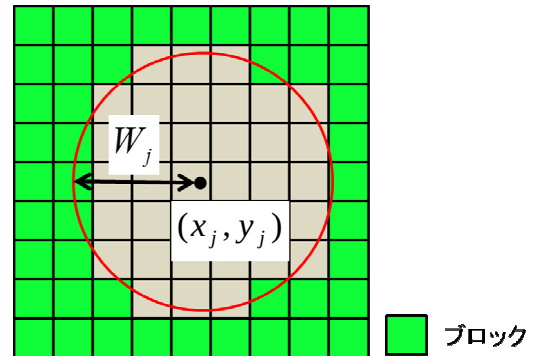


図3 通過領域判定(従来)

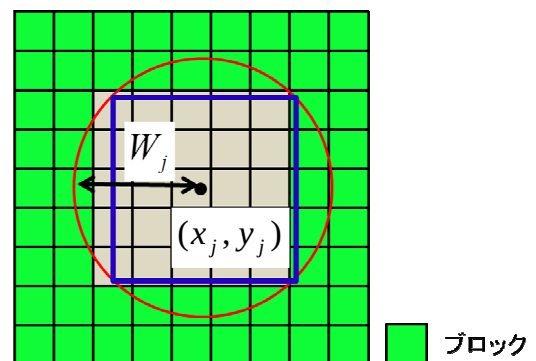


図4 通過領域判定(今回)

4. 波面記録法の AMD GPU への実装

今回は、CGH面と波面記録面のサイズを2,048×2,048とし、物体点数は11,646点、44,647点、95,949点の場合の計算を行った。実装に用いるAMD GPUは、RadeonHD6870(AMD)を用いた。また、CUDAとNVIDIA GPUとの比較を行うために、GeForceGTX590(NVIDIA)を用いて波面記録法を計算した。GTX590は2チップ搭載しているが、今回は1チップのみで動作させた。

従来手法ではステップ1の計算を行うためにはまず通過領域の判定が必要となる。GPUは条件判断に不向きなため、今回は、CPU上で判定を行った。領域判定の計算時間の結果を表1に示す。CPUにはAMD Phenom9750を用いた。

表1 通過領域の判定時間(CPU 1スレッド)

点数	11,646	44,647	95,949
従来手法 [ms]	27	100	213
提案手法 [ms]	3	11	26

表1から提案手法を用いることで従来手法より、高速に通過領域判定を行うことができた。

これは、先ほど述べたように今回用いた判定方法では、各ブロックに対して条件判断を行う必要がなく、物体点の座標と半径 W_j を用いるだけで通過するブロックを決定することができるためである。

次に、通過領域判定後に OpenCL を用いてステップ1（式(2)）の計算を行う場合について説明する。OpenCL では、計算を行う単位としてワークアイテムがある。また、ワークアイテムの集まりであるワークグループがある。ワークグループ内のワークアイテムは、高速にアクセス可能なメモリ領域を共有できる。

今回の実装では、波面記録面の1つの画素と1つのワークアイテムを対応させ、ブロック(図2)と1つのワークグループを対応させる。

GPU に波面記録法のステップ1（式(2)）を計算させた結果を以下に示す。

表2 ステップ1の計算時間(RadeonHD6870)

点数	11,646	44,647	95,949
従来[ms]	22	54	87
今回[ms]	19	40	66

表3 ステップ1の計算時間(GeForceGTX590)

点数	11,646	44,647	95,949
従来[ms]	19	62	128
今回[ms]	14	42	86

RadeonHD6870(AMD)、 GeForceGTX590(NVIDIA)ともに、従来の方法と比べてステップ1（式(2)）の計算時間が高速化されている。

計算時間が高速化された理由としては、図3と図4を比較すると、今回用いた判定方法では、従来よりも判定する通過領域が小さくなるため、波面記録面の各ブロックを通過する物体点数の合計が減少し、その結果、計算量が減り計算時間が短くなったと考えられる。

表2と表3を比較すると物体点数が増加するにつれてAMD GPUの方が高速に計算を行えていることが分かる。

ステップ1の計算に対しては、AMD GPUとOpenCLを用いることで、NVIDIA GPUとCUDAを用いた場合と、同程度以上の演算性能が得られることが確認できる。

次に、ステップ2(フレネル回折)の計算時間を比較する。ステップ2は波面記録面のサイズにのみ依存するので、ステップ1の手法を変えても計算時間は変化しない。

波面記録面のサイズが $2,048 \times 2,048$ の場合、ステップ2の計算時間は、RadeonHD6870(AMD)では、240[ms]であり、GeForceGTX590(NVIDIA)では、20[ms]であった。

AMDとCUDAで計算時間に大きな差が生じているが、理由としては、CUDAには高速にFFTが計算可能なCUFFTというライブラリが提供されており、今回はこれを用いて計算を行ったため、計算時間に大きな差が出たと考えられる。

最後に、従来の判定方法と今回の判定方法を用いて波面記録法で作成したCGHを数値再生した結果を比較し、今回の手法を用いた波面記録法と直接計算の像の比較を行う。

図6と図7は、ティラノサウルス(11,646点)の数値再生の結果の頭部を拡大した図であるが、従来(図3)ではティラノの顔の部分のぼけがあったが、今回の方法(図4)を用いることでぼけが解消されていることが分かる。

画像のぼけが解消された理由としては、通過領域判定の手法を変えた結果、実際には光が通過しない領域の計算を削減したためだと考えられる。

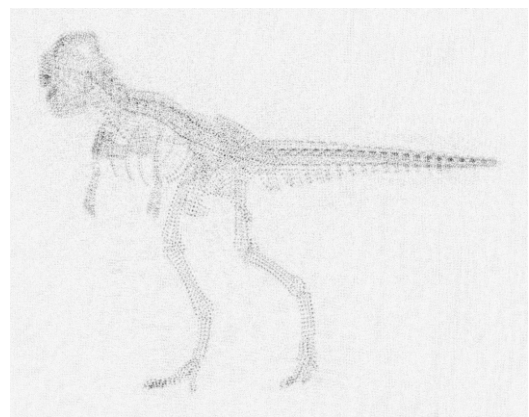


図5 ティラノサウルス(11,646点)

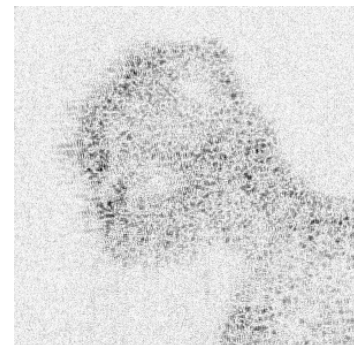


図6 ティラノ頭部(従来)

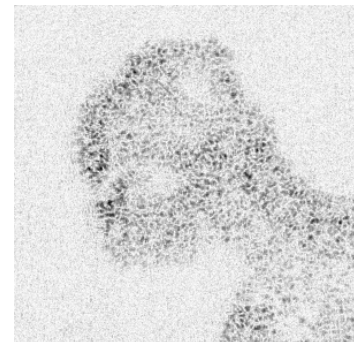


図7 ティラノ頭部(今回)

図 8 と図 9 は、波面記録法(今回の手法)と直接計算のティラノサウルスの尾の拡大図である。

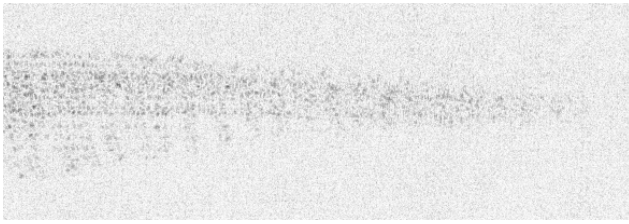


図 8 ティラノ尾(波面記録法 今回の手法)

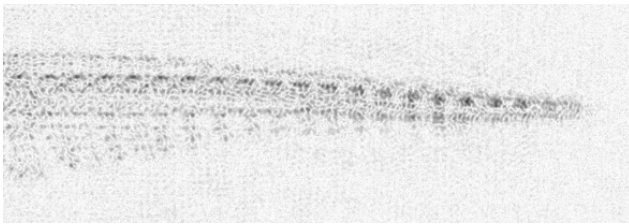


図 9 ティラノ尾(直接計算)

図 8 と図 9 を比較すると波面記録法では全体的にぼやけており、図 9 にみられるような点の集合を確認することができない。

今回の手法では、直接計算と比較すると物体点が集中している部分の再生像が良く得られないことがわかる。今後の課題として、点が集中している部分での再生像をきれいに得るために更なる判定方法の改善が必要となる。

5. まとめ

今回用いた通過領域判定手法を用いることで、CPU 上での領域判定時間の削減を行え、また数値再生の結果の改善が得られた。

また、波面記録法の計算に AMD GPU を用いることで、波面記録法の計算の一部を NVIDIA GPU と CUDA を用いた場合よりも高速に計算が行えることが確認できた。しかし、全体の計算時間では、NVIDIA GPU の方が高速に計算が行えている。これは、CUDA が提供している豊富なライブラリの存在が理由であり、AMD GPU と OpenCL を用いるよりも、容易に実装が行える。今後、OpenCL に対応した、高速に FFT を実行可能なライブラリ類が登場すれば、全体の計算時間の短縮が見込まれ、実装も容易になると考えられる。

謝辞

本研究は、総務省・戦略的情報通信研究開発推進制度(SCOPE)(課題番号 09150542)、文部科学省科学研究費補助金若手研究(B)(課題番号 23700103)、及び、中島記念国際交流財団による補助のもとで行われました。深く感謝の意を表します。

参考文献

[1] C. Slinger, C. Cameron, and M. Stanley, "Computer-Generated Holography as a Generic Display Technology", *Computer* 38, 46-53 (2005).

[2] M. Lucente, "Interactive Computation of holograms using a Look-up Table", *J. Electron. Imaging* 2, 28-34 (1993).
 [3] T. Shimobaba, N. Masuda, and T. Ito, "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane", *Opt. Lett.* 34, 3133-3135 (2009).
 [4] T. Shimobaba, H. Nakayama, N. Masuda and T. Ito, "Rapid calculation of Fresnel computer-generated-hologram using look-up table and wavefront-recording plane methods for three-dimensional display", *Opt. Express* 18, 19504-19509 (2010).
 [5] P. Tsang, W.-K. Cheung, T.-C. Poon, and C. Zhou, "Holographic video at 40 frames per second for 4-million object points", *Opt. Express* 19, 15205-15211 (2011).
 [6] J. Weng, T. Shimobaba, N. Okada, H. Nakayama, M. Oikawa, N. Masuda and T. Ito, "Generation of real-time large computer generated hologram using wavefront recording method", *Opt. Express* 20, 4018-4023 (2012).