C-008

FPGA によるパストレーシング法を用いた画像レンダリングの性能評価
Performance evaluation of image rendering using FPGA-based path tracing method

田嶋 秀成 † 増田 信之 †
Hidenari Tajima Nobuyuki Masuda

1. まえがき

近年，ゲームに用いられるようなリアルタイム映像を作成するためのラスタライズと呼ばれる手法では，光源や環境光による色の変化，反射や屈折といった現実世界に存在する光学現象の再現力に乏しい．一方で，アニメーション映画の作成に使用されるようなパストレーシング法は光学現象の表現力が高い．しかしながら，パストレーシング法は莫大な計算時間を必要としてしまうという欠点が存在する．本研究では FPGA にパストレーシング法を用いたレンダラを実装することで，CPU に対して高速化することに成功し，FPGA を用いたパストレーシングの将来性を見出した．

2. パストレーシング

パストレーシング法は，レイトレーシング法の一種である．レイトレーシング法では，以下のレンダリング方程式をコンピュータを用いて解くことによって 3DCG の色を決定する [1].

$$L_o(\vec{x_i},\vec{\omega_o}) = L_e(\vec{x_i},\vec{\omega_o}) + \int_{S^2} f_s(\vec{x},\vec{\omega_i},\vec{\omega_o}) L_i(\vec{x_i},\vec{\omega_i}) \left|\vec{\omega_i}\cdot\vec{n}\right| d\vec{\omega_i} \quad (1)$$

この方程式のそれぞれの記号の意味を以下に示す．

表 1 レンダリング方程式の各記号の意味

記号	意味
$\vec{x}$	考える物体点の位置ベクトル
$\vec{\omega_o}$	光の放射方向ベクトル
$\vec{\omega_i}$	光の入射方向ベクトル
L_o	物体点 $\vec{x}$ における放射輝度
L_e	物体点が自発光していた場合の放射輝度
L_i	物体点への入射輝度
f_s	入射光をどれだけ透過・反射するか（BDRFS）

上記のレンダリング方程式をどこまで詳細にシミュレーションするかによって 3DCG の写実性とシミュレーションにかかる時間が変わることになる．

パストレーシング法では，これらの量のうち，通常のレイトレーシング法では考慮に入らない間接光を考慮することによって，より現実に即した光学現象をシミュレーションすることが可能になる．間接光を考慮するためには，一通りの光線だけではなく，複数の光線をシミュレーションして積分を計算する必要がある．(1) 式を愚直にシミュレーションしようとすると，考慮すべき光線の量が無限大に発散してしまうため，モンテカルロ法を用いてランダムにある一定数の光線を考慮する事により，近似計算を行う．

3. FPGA を用いた専用計算機システム

3.1. 使用環境

本研究では以下の表 2 に示すハードウェアを用いて実験を行った．

表 2 評価で用いたハードウェア

用途	CPU	FPGA ボード
CPU での測定	Intel Core i5 4460	-
FPGA での測定	Intel Core i7 8700	Xilinx VC707

なお，VC707 は FPGA としては Xilinx 社の Virtex-7 を搭載し，LUT を 485,760，DSP を 2,800，BRAM を 37,080KB 搭載した評価ボードである [2].

また，それぞれで実行していたソフトウェアは以下の表 3 のとおりであった．

表 3 評価で用いたソフトウェア

用途	OS	コンパイラ
CPU での測定	Ubuntu 18.04	GCC 7.4.0
FPGA での測定	CentOS 7.7	Vivado HLS 2019.1

3.2. 制作したシステム

本研究では，パストレーシング法のすべての計算を行う専用計算機を制作するのではなく，1 ドット，1 サンプリングを行うシンプルな専用計算機を制作し，CPU での 1 ドット 1 サンプリングの計算にかかる時間との比較を行った．

専用計算機は高位合成と呼ばれる，高級言語で記述したプログラムを VHDL 等のハードウェア記述言語に動的に変換する技術を用いて開発を行った．ここで開発した専用計算機を MicroBlaze と呼ばれる FPGA 上に実装する

† 東京理科大学 先進工学研究科

CPU を用いて制御することにより，結果の抽出を行った．

本研究で使用したシステムのブロック図を以下に示す．

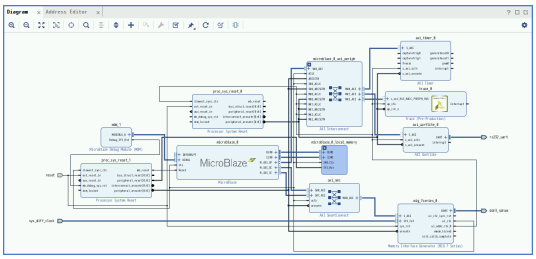

図 1　本研究で使用したシステムのブロック図

図中の mig_7series_0 がクロックの生成と VC707 に搭載の DDR3 メモリの制御の両方を担っており，ui_clk から 200MHz，ui_addn_clk_0 から 100MHz のクロックを生成させた，また，proc_sys_reset_0 に 100MHz，proc_sys_reset_1 に 200MHz に同期したリセット信号を生成させた．更に，すべての IP 間でのデータのやり取りは AXI4 という規格を用いて行い，100MHz と 200MHz の 2 種類のクロックに同期した信号を吸収して同じように扱うことができる，microblaze_0_axi_periph と名前のついている AXI Interconnect と呼ばれる製品を採用した．

4.　評価

4.1.　リソース使用量

まずは Vivado HLS にて生成された IP コアと MicroBlaze をつなぎこんだ図 1 に示した回路を FPGA 実装した際のリソース使用量を示す．

表 4　リソース使用量

リソース	使用量	使用可能量	使用率/%
LUT	52,997	303,600	17.46
FF	41,793	607,200	6.88
BRAM	9.50	1,030	0.92
DSP	313	2,800	11.18

この結果から，実際に画像を作成する際には本研究で作成した 1 ドット 1 サンプリングを行うための構成をそのまま 5 個乗せることができることがわかる．

4.2.　計算時間

2 種類のランダムシードを用いてそれぞれ CPU と FPGA にて計算時間の測定を行った．なお，計算時間はそれぞれ 10 回測定し，その平均値を掲載する．また，CPU の実行結果は gcc の -O3 オプションを付けた値である．

この結果を見ると，2 種類のランダムシードでそれぞれ FPGA は CPU に対して 1.3 倍の高速化を達成していることがわかる．

表 5　二種類のランダムシードでの CPU と FPGA それぞれの計算時間

使用機材	ランダムシード初期値	計算時間/us
CPU	10	32.40
CPU	2055728506	33.10
FPGA	10	24.64
FPGA	2055728506	24.83

4.3.　計算精度

次に，前節で掲載した 2 種類のランダムシードそれぞれでの出力の精度を示す．

表 6　二種類のランダムシードでも CPU と FPGA それぞれの計算結果

使用機材	ランダムシード初期値	R	G	B
CPU	10	0.00	0.00	0.00
CPU	2055728506	6.75	2.25	2.25
FPGA	10	0.00	0.00	0.00
FPGA	2055728506	6.75	2.25	2.25

ただし，色の出力，R，G，B はそれぞれ Red，Green，Blue を示し，最大 12 で正規化された値である．

5.　まとめと今後の展望

本研究により，並列化を全くしない，また，クロック数も 10 分の 1 という非常に厳しい条件ながら光線の追跡を CPU に対して 1.3 倍高速に行うことができることを示すことができた．これは，より効率的な実装を行い，並列化して FPGA に実装を行った場合，CPU に対して高いパフォーマンスを得ることができる可能性を示している．

今後は高位合成ではなく，直接 Verilog を用いた実装を行うとともに，PCIe 接続を行い，パソコンの CPU から専用計算機の制御を行うことで，高速化を目的とした専用計算機の開発を行っていきたい．

参考文献

[1] shocker，“レンダリング方程式 (Rendering Equation) 1”，https://rayspace.xyz/CG/contents/LTE1/，2021 年 6 月 17 日参照

[2] Xilinx，“Xilinx Virtex-7 FPGA VC707 Evaluation Kit”，https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html#overview，2021 年 6 月 17 日参照