2020 年 2 月 6 日 電気通信大学 情報理工学域 コンピュータサイエンスプログラム 卒研発表会資料

レイトレーシング法に特化した GPU の性能モデル作成

発表者： 1610031 荒生 太一
指導教員： 成見 哲

1 はじめに

1.1 背景

近年，従来の汎用プロセッサに加えて特定の問題に特化させた専用プロセッサを搭載した Graphics Processing Unit(GPU) が普及し始めている．2018 年に発売された NVIDIA 社製の GPU，NVIDIA RTX ではレイトレーシング法の計算を高速化するために従来の汎用プロセッサに加えて 2 種類の専用プロセッサを搭載している．

GPU を利用してプログラムの高速化を試みる際，労力に見合った高速化が達成されるかどうか事前に判断することができれば，実際に高速化を行うかどうかの検討を行う際に有用である．GPU 上での計算時間を事前に予測するための方法として，性能モデルを作成する研究が存在している．伊藤らは一般的な GPU において，テクスチャの入出力にかかる時間から性能モデルを作成している [1]．この性能モデルは最悪で 20%程度の誤差があるとされている．また，島田らは OpenCL による数種類の計算命令の呼び出し回数を回帰分析することで異種アーキテクチャに対応した性能モデルを作成している [2]．これらの性能モデルは複数種類のプロセッサが搭載された RTX のような GPU を想定していないため，そのまま適用することができない．

1.2 研究目的

以上の背景を踏まえ，本研究では RTX や従来 GPU でレイトレーシング法を計算する際の性能モデルの作成を目指す．作成する性能モデルは，いくつかのパラメータ (ポリゴン数，GPU のコア数等) を入力するとレイトレーシング法の計算にかかる時間を推定するようなものである．この性能モデルには，計算時間への影響が大きいパラメータを絞り込み，ユーザーに最適化の指標を提示したり，他の GPU で計算したときの処理時間を推定することで GPU を変更した際の費用対効果を予測する等といった活用法があると考える．

2 既存技術

2.1 レイトレーシング法

レイトレーシング法は，光の経路をシミュレーションすることで 3D 画像を生成する手法の一つである．図 1 にレイトレーシング法の概要図を示す．光源から出た光は，物体の表面で反射や屈折を繰り返し最終的に人の目に入る．レイトレーシング法ではこの目に入る光の経路を逆に辿ることで光の経路をシミュレーションする．レイトレーシング法で行われる主な処理として，光線を投射する処理，物体と光線の交差判定，物体表面で起きる物理現象の計算の 3 つがある．2 つ目の交差判定処理が最も計算時間がかかる部分で，3D 空間上にある物体の数が多くなるほど交差判定を行う回数が増加する．光線と物体の交差判定を高速化する工夫として，物体を木構造に格納する手法がある．交差判定を行う時は木構造を探索すれば良いため，線形的に探索するよりも効率が良くなる．

2.2 NVIDIA RTX

従来の NVIDIA 社製 GPU と RTX の違いを図 2 に示す．RTX には汎用プロセッサである CUDA コアに加え，レイトレーシング法の計算に特化した RT コアと機械学習の計算に特化した TENSOR コアの 2 種類が追加されている．

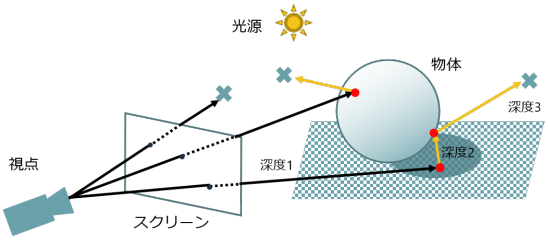

図 1: レイトレーシング法の概念図

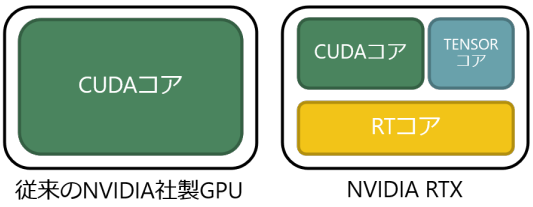

図 2: 従来 GPU と RTX の構造の違い

2.3 DirectX Raytracing(DXR)

DirectX Raytracing(DXR) は，DirectX と呼ばれる Microsoft 社製グラフィックス API で RTX を制御するための命令セットである．一般に 3D オブジェクトはポリゴンと呼ばれる三角形で構成されているが，DXR ではポリゴンをメモリに格納する形式として次の 2 つのデータ構造を使用している．

- Bottom-Level Acceleration Structure(BLAS)
- Top-Level Acceleration Structure(TLAS)

BLAS は 3D モデルを構築しているポリゴンを木構造に格納しており，TLAS は複数の BLAS を木構造で格納している．ポリゴンを木構造で管理することで，レイトレーシング法において光線と衝突するポリゴンを高速に検索することができる．また，光線の生成やポリゴンと交差したときの挙動をシェーダで自由にプログラムすることができる．

3 性能モデル作成のための準備

性能モデルを作成するための前段階として，実際に DXR でレイトレーシング法を計算するプログラムを作成しその実行時間を測定した．使用した GPU と 3D シーンの詳細を表 1 および表 2 に示す．3D シーンは Computer Graphics Archive[3] から入手した．これらについて，深度を 1 ずつ増やしながら 100 になるまで実行時間を測定した．

表 1: 使用した GPU の詳細

GPU 名	RTX2080	RTX2060	GTX1660
CUDA コア数	2944	1920	1408
RT コア数	46	30	0
ブーストクロック [MHz]	1710	1680	1785

表 2: 使用した 3D シーンの詳細

モデル名	ポリゴン数
CornellSphere	2188
Sponzacry	262267
living_room	580629
San_Miguel	5617450
conference	331179
sibenik	75284

表 3: 性能モデルに当てはめるパラメータ

パラメータ名	説明
T_{ray}	プログラムの計算時間 (ms)
M_{ray}	TraceRay() の呼び出し回数
C_{search}	シーンのポリゴン数
C_{surf}	1
$Perf_{\mathrm{cuda}}$	GPU の FLOPS (CUDA コア数*ブーストクロック*2)
$Perf_{\mathrm{ray}}$	$\frac{\text{深度 1 の時のレイの本数}}{\text{深度 1 の時の実行時間}-\text{深度 0 の時の実行時間}}$
$isRTX$	1=RTX，0=従来 GPU

4 性能モデルの作成および評価

4.1 パラメータ選定

レイトレーシング法は，レイの探索を行った後表面処理を行うため，それぞれの計算にかかるコストおよびそれぞれの計算を処理する GPU の性能は性能モデルの項に含める必要がある．また，RTX では RT コアでレイの探索を行い CUDA コアで表面処理をしているが，従来 GPU では CUDA コアのみで両者の計算を行うため，計算負荷がかかった際に従来 GPU のほうが RTX よりも急速に性能が低下すると考えられる．そのため，性能モデルに RTX か否かを 2 値で表す補正項 $isRTX$ を導入する必要があると考える．以上より，作成された性能モデルを式 (1) に，各変数の説明を表 3 に示す．なお，各項の係数 $\beta_0 \sim \beta_3$ は回帰分析によって計算した．

$$T_{\mathrm{ray}} = \beta_0 + \beta_1 \frac{M_{\mathrm{ray}} \times C_{\mathrm{search}}}{Perf_{\mathrm{ray}}} + \beta_2 \frac{M_{\mathrm{ray}}}{Perf_{\mathrm{cuda}}} + \beta_3 isRTX \quad (1)$$

4.2 評価

図 3 に性能モデルを各シーンに適用したときの相対誤差を示す．どのシーンでも深度が低い場合精度が悪いが，深度が高い場合はほとんどのシーンで平均 80%の精度で実行時間を推測できていた．深度が高い場合でも精度が悪かったのは RTX2080，RTX2060 における CornellSphere，livng_room，sibenik であった．これらの理由として，まず CornellSphere と sibenik はポリゴン数が少なく，深度が高くなってもあまり計算負荷が高くないことが上げられる．今回の分析に用いたデータは計算時間が大きいものが多いため，計算時間が低い=計算負荷が低い場合のデータを上手く近似できていないと考えられる．

また，living_room の誤差が大きい理由については，ポリゴンの偏りが考えられる．表 2 より，living_room のポリゴン数は他のシーンよりも多いが，実際は半分近くのポリゴンが狭い空間に偏っている．このような狭い空間にポリゴンが多数存在するシーンでは，その部分に衝突しないレイが多いため実際に探索されるポリゴン数は少ないと考えられる．従って，living_room では実際にはあまり多くのポリゴンとの交差判定が行われておらず，負荷があまり高くないため計算精度が低くなっていると考える．

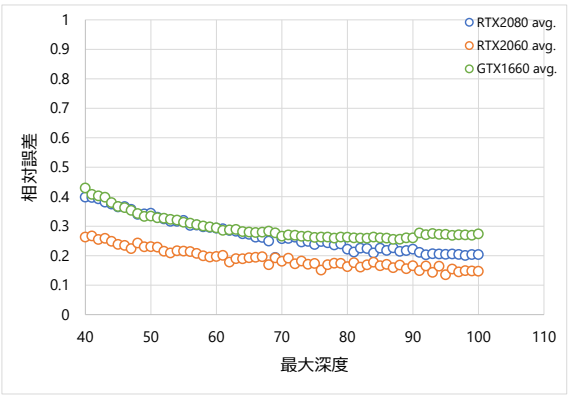

図 3: 性能モデルの精度

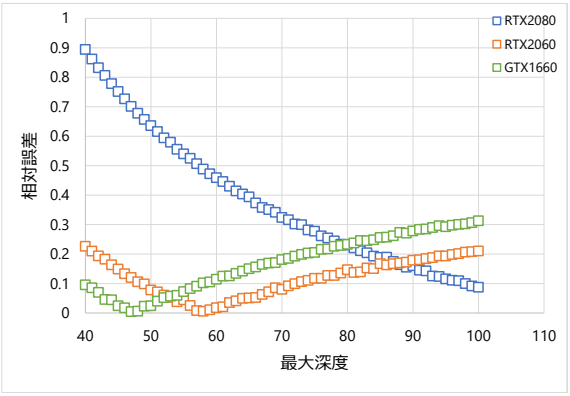

図 4: 性能モデルを未知シーンに適用したときの精度

4.3 未知シーンへの適用

式 (1) を未知シーンに適用した．結果を図 4 に示す．計算負荷が高い場合，今回の性能モデルは平均で 80%程度の精度で実行時間を推測できることが確認できた．

5 まとめと今後の課題

5.1 まとめ

本研究では，レイトレーシング法に特化した GPU である RTX と従来 GPU 両方に対応した性能モデルを作成した．作成した性能モデルは，RTX や従来 GPU でレイトレーシング法を計算したときの実行時間を推定するものである．精度を評価したところ，より負荷が高い場合良い精度で予測できていることが確認できた．また，未知シーンに対して適用したところ，平均で 80%の精度で計算時間を推定できていることが確認できた．

5.2 今後の課題

課題として，計算負荷が低い場合での精度向上が上げられる．また，シーンのポリゴンの偏りを表すような値を用いることで精度向上が期待できると考えられる．今後様々な GPU やシーンで計算していくことで精度の向上を図ることができると考える．

参考文献

[1] 伊藤信悟, 伊野文彦, 萩原兼一. GPGPU アプリケーションの開発を支援するための性能モデル. 情報処理学会論文誌コンピューティングシステム（ACS）, Vol. 48, No. 13, pp. 235–246, Aug, 2007.

[2] 島田大地, 遠藤敏夫, 丸山直也, 松岡聡. OpenCL を用いた異種 GPU における性能特性に応じた最適化 (ハイパフォーマンスコンピューティング (HPC) Vol.2010-HPC-128). 情報処理学会研究報告, Vol. 2010, No. 5, pp. 1–7, Feb, 2011.

[3] Morgan McGuire. Computer Graphics Archive, July, 2017. `https://casual-effects.com/data`(アクセス日 2020-01-31).