「はやぶさ 2」ハードウェアシミュレータに係るレイトレーシングソフトウェアの開発と評価

三浦 昭[*1]，武井 悠人[*1]，山口 智宏[*1,*2]，高橋 忠輝[*1]，佐伯 孝尚[*1]

Development and Evaluation of the Ray-Tracing Software Used in the Hardware-in-the-loop Simulator of Hayabusa 2

Akira Miura[*1], Yuto Takei[*1], Tomohiro Yamaguchi[*1,*2], Tadateru Takahashi[*1], Takanao Saiki[*1]

概 要

2014 年 12 月に打ち上げられた小惑星探査機「はやぶさ 2」は，2018 年の夏から小惑星リュウグウ (162173, 1999 JU_3) の探査を続けている．係る探査においては，探査前に情報が得られないような状況下で，様々なクリティカル運用が予定されており，係る事前訓練や検証等が欠かせないものとなった．同様に関連する科学者も実際の観測に先立って訓練を積む必要があった．来たる実運用に向けて，信頼性を上げるための一助としてハードウェアシミュレータが開発された．その構成要素の一つとして，画像生成装置が開発され，「はやぶさ 2」搭載の様々な光学機器の模擬データを生成することとなった．本稿においては，光学機器模擬の概要を記すと共に，そこで使われるレイトレーシング機能について，性能評価の概要を述べる．

Abstract

The Hayabusa 2 asteroid explorer was launched in December 2014 and has been exploring Ryugu (162173, 1999 JU_3) since the summer of 2018. The mission includes various critical operations under previously unknown circumstances. Also, scientists need to be trained in advance for actual observations. To increase the reliability of forthcoming operations, a hardware-in-the-loop simulator has been developed. One of the components of the simulator is the asteroid image generator, which generates various simulated data of optical instruments on board Hayabusa 2. In this paper, we describe the overview of the optical instruments simulation and the performace of the ray-tracing software used therein.

Keywords: Hayabusa 2, HIL simulator, Ray tracing

1　まえがき

本稿においては，「はやぶさ 2」における各種訓練のために用意されたハードウェアシミュレータのコンポーネントである画像生成装置について，その中核をなすレイトレーシングソフトウェア開発の概要と性能評価を中心に述べる．

小惑星探査機「はやぶさ」に続いて開発された「はやぶさ 2」は，C 型小惑星のサンプルリターンミッションを担って 2014 年 12 月 3 日に打ち上げられた．「はやぶさ 2」は 2018 年 6 月に小惑星リュウグウ (162173，1999 JU_3）に到着し，2019 年末までリュウグウに滞在する予定となっている．

小惑星リュウグウについては，事前に把握できた情報が少なく，小惑星到着に備えた検討準備が不可欠であった．そのような中で，実際の探査を想定した様々な訓練が計画され，実施されてきた．その代表的なものに RIO（Real-time Integration Operation）訓練 [1] と LSS（Landing Site Selection）訓練 [2] が挙げられる．

LSS 訓練は「はやぶさ 2」のタッチダウン地点選定のための訓練であり，リュウゴイドと呼ばれる仮想小惑星 [3] を用いて本番さながらの各種観測等を模擬した訓練が 2017 年度初頭から実施された [4][5][6][7][8][9]．訓練に供された画像の例を図 1 に示す．図中に付されている名称等は，模擬観測に基づいて理学研究者が命名したものである．

RIO 訓練においては，2017 年度後半から 2018 年度初頭にかけて，実際の管制室も用いて様々な条件下でのリアルタイム訓練が繰り返し実施された．その概念図を図 2 に示す．

それらの訓練を支援するシステムのひとつが，ハードウェアシミュレータ（Hardware-in-the-Loop Simulator, HIL）である [11]．その概略を図 3 に示す．HIL シミュレータは RIO 訓練やトラブルシューティング，運用手順の検証等の用途を想定して開発されたものであり，探査

doi: 10.20637/JAXA-RR-18-008/0004
[*] 平成 30 年 12 月 19 日受付（Received December 19, 2018）
[*1] 宇宙航空研究開発機構宇宙科学研究所（Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency）
[*2] 所属：三菱電機（株）（Present affiliation is Mitsubishi Electric Corporation.）

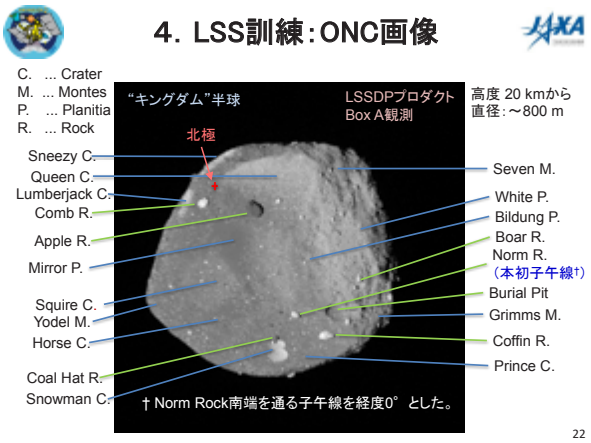


図 1: LSS 訓練における ONC 画像の例（引用元: [10]）

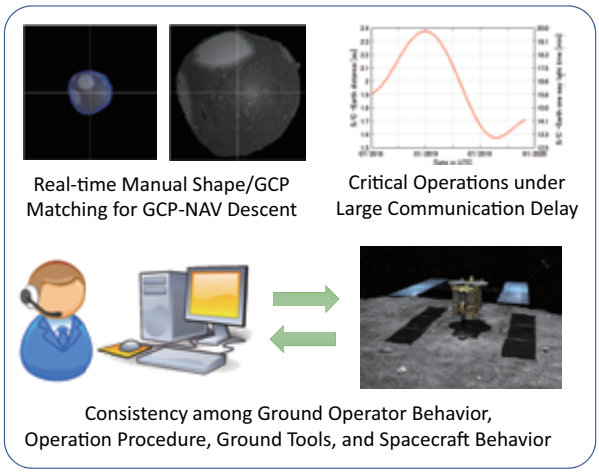


図 2: RIO 訓練概略（引用元: [11]）

機，通信環境，地上局等，様々な模擬コンポーネントで構成されている．そのひとつとして，小惑星探査に係る画像生成を主目的とした装置（画像生成装置）が開発されている．画像生成装置においては，探査機を模擬するコンポーネントから受け取ったダイナミクスデータや，模擬観測に使用される光学機器の構成等に基づいて，画像やドップラーデータ等の模擬データが生成され，関係するコンポーネントに生成結果が提供される．

画像生成装置は RIO 訓練のみならず，LSS 訓練やその他の様々な訓練・検証のための画像生成にも供されており，本稿の執筆時点では，リュウグウの形状モデルに基づいた模擬画像生成等，さらに適用範囲を広げている．その画像生成装置において中核的な機能を提供しているのが，光学機器模擬（Optical Instruments Simulator, OIS）である．

以下，次節で OIS 構築の概要を述べた後，その中でも共通要素として用いられ，かつ演算負荷の高いレイトレーシング機能に焦点を絞って，その検討・評価を行った結果について述べる．

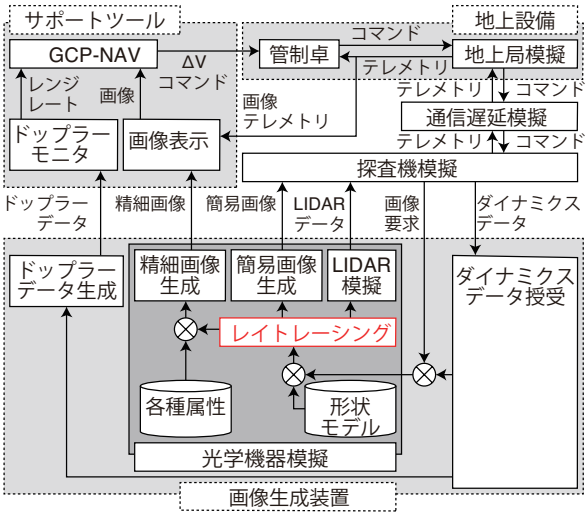


図 3: HIL シミュレータの概要と光学機器模擬（OIS）の位置付け

2 OIS

本節では，図 3 中に示される光学機器模擬（OIS）構築の概要を述べる．

2.1 OIS が満たすべき要件等

2.1.1 速度・精度

各種訓練の成果を高めるためには，OIS は要求される応答時間内に訓練遂行に十分な品質の模擬画像を生成する必要がある．

例えば RIO 訓練で許容される遅延時間は，精細画像で 10 分程度，簡易画像で 1 秒程度となる．他方，LSS 訓練においては，RIO 訓練と比べると高い精度のレンダリング結果を提供する必要がある．例えばオーバーサンプリングを考慮すると，数万 × 数万画素程度のレンダリングが必要となる場合があり，画素毎の位置精度も相応のものが必要となる．例えば理学観測に用いる望遠カメラ（ONC-T）[12] の画角は 6 度程度であるから，このような場合，1 画素あたりの分解能は 1 秒角未満となる．

2.1.2 インタフェース

OIS ソフトウェアと他のプロセスとのインタフェースは以下の通りである．なおコマンドやデータ格納形式等を定めたインタフェース基準書は非公開であるため，ここでは概要を述べるに留める．

- 他プロセスとのインタフェースは TCP のソケット通信を用いる．

- コマンドは当該ソケット通信により受け取り，実行にあたってのステータスはソケット通信の返信として要求元に通知する．
- コマンドのフォーマットは JSON 形式とし，係るパラメータは事前に定義された形式で受け取るものとする．もしくは JSON 形式でパラメータを格納したファイルのパスをコマンドとして受け取る．
- 実行結果となるデータ群は前述のコマンドにより指定された形式で，指定されたファイルシステムに書き出す．

2.1.3　OIS に提供されるデータ等

OIS には，以下に述べるような形状モデルや各種属性等を収めたファイル群が提供される．

形状モデル　小惑星や探査機，その他模擬に必要となる各種形状モデルが提供される．小惑星モデルは，その用途によって約 30 万ポリゴンから 4 億ポリゴン程度の形状モデルが用意される．

各種属性　以下のような属性を表すデータ群で構成されている．これらのデータは関連する理学関係者や搭載機器関係者から提供されるものであり，訓練の目的等に応じて画像生成に供される．

- 小惑星形状モデルの各ポリゴンに対して，地質等の属性を与える対応表．シミュレーションの用途によっては，他の理学データやボルダー識別等の対応表も提供される場合がある．
- 太陽やフラッシュランプ等の光源の特性.
- 各種地質属性（光学的特性等）.
- 搭載カメラ（フィルタ，CCD）等の特性.
- その他の補助的データベース.

画素値計算ライブラリ　OIS ソフトウェアが画素値を計算するにあたって必要となる関数群が収められている．

2.2　OIS ソフトウェア

OIS ソフトウェアは，前述の要件を満たすために設計・構築された，レイトレーシングを基本としたレンダリングソフトウェアであり，各種カメラや LIDAR 等の光学機器を模擬し，画像等の模擬データを生成するものである．

なお実際の画像生成装置では用途毎に複数のプロセスが並行稼働することがあるが，図 3 中では便宜的に単一の編成として表現している．例えば RIO 訓練を実施する場合，OIS としては精細画像生成を担うプロセスと簡易画像生成・LIDAR 模擬を担うプロセスの 2 つが並行稼働する．その他の関連するプロセスも同時稼働する場合がある．

2.2.1　処理の流れ

OIS ソフトウェアの処理について，その概略を以下に示す．

OIS ソフトウェアの起動　OIS ソフトウェアは，必要とされる形状モデルや各種属性情報等のデータを起動時に読み込み，以後の画像生成要求を受信するための TCP ソケットを生成する．係るポート番号はファイルに保存して他のプロセス等から参照可能な状態とする．

OIS ソフトウェアは原則として HIL シミュレータ内で各種コンポーネントと連携するものであるため，GUI 等のような対人インタフェースは持たない．その代わりに，TCP 通信をラップした，簡易のクライアント用 CLI を提供する．

画像生成要求，ダイナミクスデータの受信　OIS ソフトウェアは，画像生成の要求が発生する都度，その時点のダイナミクスデータと画像生成条件を TCP ソケット経由で受信する．LIDAR は現実には画像を生成するカメラではないが，ソフトウェア上では 1 ピクセルの深度（距離）画像として扱っている．画像生成に必要とされるダイナミクスデータは，OIS ソフトウェア外のプロセスが担っており，その生成の詳細については本稿の対象外とする．

レンダリング　OIS ソフトウェアは，受信したダイナミクスデータ等と，起動時に読み込まれた形状モデル等に基づいて画素値や距離情報を計算する．コマンド要求の内容によっては，予め提供されている各種属性や画素値計算ライブラリ等が使用される．

結果の保存　OIS ソフトウェアは，レンダリング結果の画素値や距離情報等を指定されたフォーマットでファイルに保存し，そのステータスを TCP ソケット経由で返信する．

待機　OIS ソフトウェアは，次の画像生成要求があるまで待機する．

2.2.2　設定ファイル

前述のように，OIS ソフトウェアでは JSON 形式でコマンドを得ることとなっているため，起動時に用いる各

```
{
    "proc_out_directory":              "out",
    "proc_image_name":                 "image.pgm",
    "proc_render_image_width":         2048,
    "proc_render_image_height":        2048,
    "proc_out_image_width":            512,
    "proc_out_image_height":           512,
    "proc_sensor_id":                  9,
    "proc_fast_simulation":            1,
    ...
}
```

図 4: JSON の書式例

種設定ファイルも原則として JSON で記述することとする．図 4 に JSON の書式例を示す．この例に記述されたパラメータは，起動時にも実行時のコマンドとしても使用可能なものである．

ここに例示された内容は，次のように解釈される．出力ディレクトリ（proc_out_directory）を “out” とし，ファイル名（proc_image_name）を “image.pgm” とする．レンダリング幅（proc_render_image_width）を 2048 とし，高さ（proc_render_image_height）を 2048 とする．出力幅（proc_out_image_width）を 512 とし，高さ（proc_out_image_height）を 512 とする．すなわち，この例は 4 倍のオーバーサンプリングを要求している．用いるカメラとフィルタの組み合わせ（proc_sensor_id）は 9 番とし，地質属性等に依らない高速のレンダリング（proc_fast_simulation）を要求する．

形状モデル等，読込に時間を要する一方で訓練に係る一連のレンダリングにおいて変更する必要がないデータや，変更することが望ましくないパラメーラ類は，プロセスの起動時に与えられた値のみを有効とする．そのようなパラメータの例を図 5 に示す．

```
{
    "init_pid_file":                  "run/pid.txt",
    "init_port_file":                 "run/port.txt",
    "init_ccd_db":                    "data/ccd",
    "init_model_probe":               "models/probe.stl",
    "init_model_asteroid":            "models/ast.stl",
    "init_geological_unit_id":        "models/ast.dat",
    ...
}
```

図 5: JSON の書式例（起動用設定）

ここに例示された内容は，次のように解釈される．プロセス ID（init_pid_file）を “run/pid.txt” に格納する．コマンド受信に使用する TCP ソケットのポート番号（init_port_file）は “run/port.txt” に格納する．CCD 等の特性データベース（init_ccd_db）を “data/ccd” から読み込む．探査機の形状モデル（init_model_probe）を “models/probe.stl” から読み込む．小惑星の形状モデル（init_model_asteroid）を “models/ast.stl” から読み込む．小惑星の地質情報（init_geological_unit_id）を “models/ast.dat” から読み込む．

2.3 画像生成の選択肢

OIS ソフトウェアは，大別すると 2 種類の画像生成を行うことが期待されている．ひとつは精細画像生成であり，もうひとつは簡易画像生成である．両者のおおまかな区別を表 1 に示す．ただしこれらの区別は，固定された要求仕様ではなく，訓練の用途に応じて，起動時もしくはコマンドによって与えられる各種パラメータでカスタマイズされるものである．

表 1: 画像生成に係る選択肢の比較

	精細画像生成	簡易画像生成
ポリゴン数	～4 億	～500 万
画素値計算	外部ライブラリ	簡易 Hapke モデル
地質情報	ポリゴン毎	一律
理学データ	オプション	無
付随処理	未定義	LIDAR 模擬 画心計算
オーバーサンプリング	必須	オプション

簡易画像生成にあたっては，LIDAR[13] 模擬や，生成された画像に基づいた簡易的な画心計算等の要求が併用されることがある．精細画像生成にあたっては，理学データ等，別途提供されるデータを併用したレンダリングが要求されることがある．

画素値計算は，2 種類の手段を使い分ける．

外部ライブラリを用いた画素値計算 精細画像生成が指示された場合は，係る画像生成に必要となる地質情報や角度情報（入射角，観測角，位相角）を画素毎に記録した上で，起動時に読み込まれた各種属性データ等を添えて，画素値計算ライブラリを呼び出す．

簡易 Hapke モデルを用いた画素値計算 簡易画像生成のために，簡易版の Hapke パラメータを用いた反射モデル（本稿では簡易 Hapke モデルと記す）を導入する．これは精細画像生成において外部の画素値計算ライブラリが実行している計算を，地質属性やスペクトル分布等に依らず高速計算で代用するものである．

本稿で採用した簡易 Hapke モデルにおいて画素値 r を計算する式を式 1 に示す [14]．なお，この式は OIS の用途によっては変更される可能性がある．

$$r(\sigma,e,\alpha) = \frac{\omega}{4\pi}\frac{\cos\sigma}{\cos\sigma+\cos e}\{[1+B(\alpha)]\,p(\alpha)+ H(\cos\sigma)H(\cos e)-1\} \quad (1)$$

ここで σ は入射角，e は観測角，α は位相角を表す．$B(\alpha)$，$p(\alpha)$，$H(x)$ は，それぞれ式 2，式 3，式 4 で表される．

$$B(\alpha) = \frac{B_0}{1 + \frac{1}{h}\tan\frac{\alpha}{2}} \tag{2}$$

$$p(\alpha) = \frac{1-\xi^2}{(1+2\xi\cos\alpha+\xi^2)^{\frac{3}{2}}} \tag{3}$$

$$H(x) = \frac{1+2x}{1+2x\sqrt{1-\omega}} \tag{4}$$

ここで ω，B_0，h，ξ は，地質によって定まる値である．OIS においては簡易的に，前述の地質属性から代表的な値を参照する．

3　レイトレーシング

本節においては，OIS ソフトウェアの機能の要となっているレイトレーシングについて，その概略を述べる．

3.1　ソフトウェアレイトレーシングの採用

基本的な手法は，文献 [15] の評価結果に鑑みて，形状の計算，影の計算共にレイトレーシングを採用することとする．これは，特に LSS 訓練において，陰影の描写を含めて，高いレンダリング精度を要求されることに拠る．精度の観点ではまた，単精度浮動小数点演算では LSS 訓練に係る高い精度要求を満たせないため，座標計算に係る演算は倍精度浮動小数点を用いることとした．

速度の観点では GPGPU による高速化も検討課題として挙げられるが，本稿が対象とする画像生成装置においては，CPU 演算を採用することとした．最大の理由は形状モデルの規模にある．訓練に用いられる形状モデルは，4 億ポリゴン近いモデルが存在し，開発時点で入手できる GPU のメモリ容量では，そのようなサイズの形状モデルを収容することは困難であると判断された．

簡易画像の生成にあたっては軽量の形状モデルが用いられ，高い精度も要求されないため，GPGPU 対応のソフトウェアを並行して開発する選択肢や，単精度浮動小数点演算を用いる選択肢もあったが，後述のように，CPU を用いた倍精度浮動小数点演算によるレイトレーシングでも所望の時間内に処理を終えられる目処が立ったため，開発・保守コスト低減の観点から，倍精度浮動小数点による CPU 演算に一本化することとした．

3.2　Voxel 分割

レイトレーシングにおいては，voxel 分割が高速化の要点のひとつとなる．本稿においては文献 [15] に則った voxel 分割を採用し，その評価を行なった．以下に，その voxel 分割手法の概要を示す．

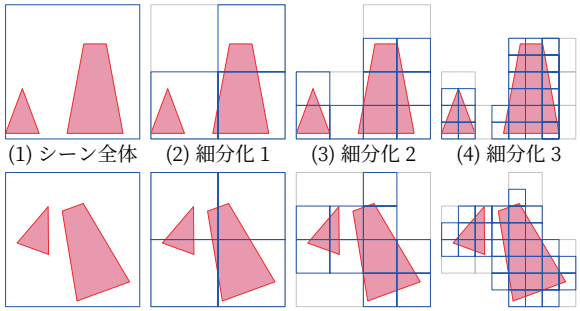


図 6: シーン毎の voxel 分割

3.2.1　Voxel 分割手法の比較

シーン毎の voxel 分割　シーン毎の voxel 分割の例を図 6 に示す．以後，本稿においては voxel 分割を簡易に表現するために 2 次元のシーン内で 4 分割を基本とした図を示すが，OIS ソフトウェアにおける voxel 分割は 3 次元空間内で，各軸方向に 2 分割，都合 8 分割することを基本として実行される．

シーン毎に voxel 分割するにあたっては，シーン内のオブジェクト配置が変化する度に voxel 分割を再実行することとなる．本稿で想定する画像生成装置においては，模擬する画像毎にオブジェクト配置が異なるため，シーン毎の voxel 分割を採用した場合，レンダリングの度に voxel 分割を待たねばならない．

オブジェクト毎の分割　これに対して，オブジェクト空間毎の voxel 分割の例を図 7 に示す．voxel の細分化は各オブジェクトの座標系内で実行されるため，シーン内のオブジェクト配置が変わっても，オブジェクト空間におけるオブジェクト形状が変化しない限りは，voxel 分割を再実行する必要はない．

OIS の用途に限れば，関連する訓練を通じて，オブジェクトの変形は考慮しなくて良い．仮に変形があったとしても，変形のない複数のオブジェクトに予め分離することで対処可能である．

また訓練に先立っての準備時間は十分に設けることができるので，訓練開始前に Voxel 分割を完了することで，訓練中は voxel 分割に要する時間を実質 0 にすることができる．

レイトレーシング時の voxel 分割の影響　レイトレーシングにおいては，レンダリング時のレイの追跡に相応の時間を要する．Voxel 分割をオブジェクト単位で実施する場合，シーン内で単一のレイ（視線，光線等）であるものを，各オブジェクト空間毎に座標変換し，異なるレイとして扱うこととなる．例えば図 7 のシーンであれば，図 8 のように 2 つのオブジェクト空間において夫々のレイを追跡することとなる．その概念図を図 9 に示す．

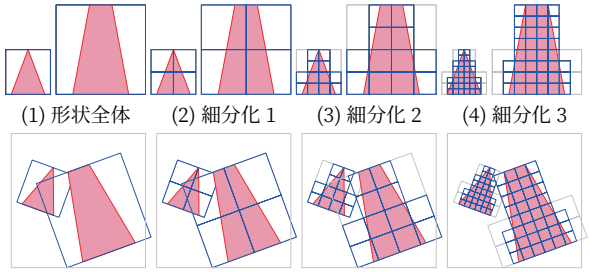


図 7: オブジェクト毎の voxel 分割

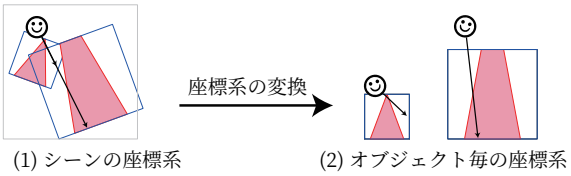


図 8: 座標系の対比

多数のオブジェクトを扱う場合は，このようなオーバーヘッドを無視できなくなる可能性がある．しかしながら OIS が扱うシーンにおいては，オブジェクト数は限られており，主要なオブジェクトは探査機と小惑星の 2 つである．訓練内容によって若干のオブジェクトが加わる程度である．

3.2.2 シーン配置の実例

実際に想定しうる小惑星と探査機の関係を図 10 に示す．同図において，左図は「はやぶさ 2」によるリュウグウの撮像例（実画像）である．右図は，この位置関係を模式的に表現したものである．リュウグウと「はやぶさ 2」のサイズは 100 倍以上の開きがあり，図中では「はやぶさ 2」は点で表現されるのみとなっている．右図の青色部分は左図を撮像したカメラ，ONC-W1[17] の視野範囲を例示している．ONC-W1 と並んで望遠側の撮像を担うカメラ，ONC-T の視野範囲も黄色で例示している．

ここで，カメラのみならず，カメラの後ろ側に位置していて決して直接は描かれない探査機のオブジェクトも同時にシーン内に配置するのは，探査機が小惑星に落とす影を描写するためである．図 10 の左図にも小さな点ではあるが「はやぶさ 2」が実際に影を落としている（同図中の矢印部分）．

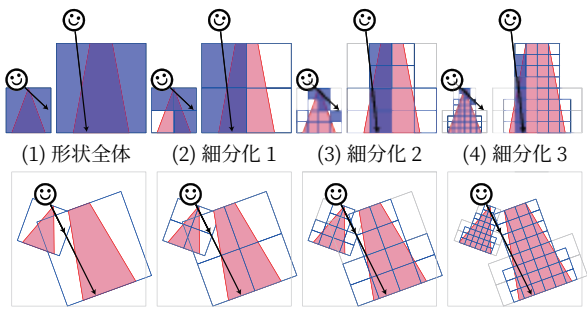


図 9: Voxel の追跡．

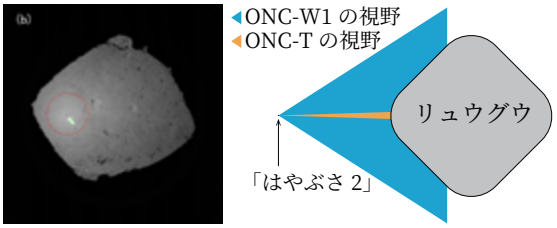


図 10: リュウグウと探査機の位置関係
左: 実画像（引用元: [16]），右: 位置関係

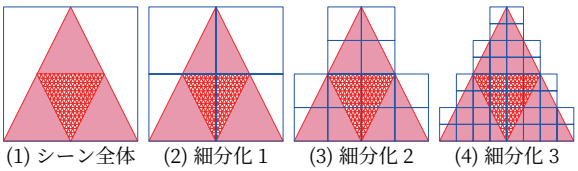


図 11: ポリゴンサイズと voxel 分割の対比

このように，各オブジェクトのサイズに比べて，探査機・小惑星を含むシーン全体のサイズは必ずしも小さくないことと，シーン内に配置されるオブジェクトのサイズに極端な相違があるのが，OIS における画像生成の実際である．

3.2.3 ポリゴンサイズと voxel 分割

ここでポリゴンサイズと voxel 分割の関係について検討する．

図 11 に，不均一なポリゴンサイズが混在するオブジェクトの例を示す．これはオブジェクトの一部にポリゴンサイズの極端に異なる領域が存在する例である．局所的な現象やオペレーション等を詳細にレンダリングする必要が生じた場合に，このような不均一なモデルが使われる可能性がある．

これをオブジェクト単位で均等に voxel 分割した場合，細分化が進むにつれて，サイズの大きなポリゴンに対しては過剰な voxel 分割が進行し，一方でサイズの小さなポリゴンに対しては voxel サイズが十分に細分化されないという不均衡な状態となる．このような不均衡は，大きなポリゴンで構成されるオブジェクトと小さなポリゴンで構成される局所的なオブジェクトとに分離することで対処できる可能性がある．

表 2: 開発環境の主要諸元

CPU	Intel Core i9-7900X
クロック	3.30GHz
CPU 数	1
コア数	10
PassMark 値	21997[18]
主記憶	128GB
OS	CentOS 7.5
コンパイラ	gcc 4.4.7

表 3: HIL 画像生成装置の主要諸元

CPU	Intel Xeon CPU E5-2640 v4
クロック	2.40GHz
CPU 数	2
コア数	20
PassMark 値	20765[19]
主記憶	256GB
OS	CentOS 6.8
コンパイラ	gcc 4.4.7

3.2.4　Voxel 分割の方針

以上に鑑み，本稿においては，voxel 分割を以下の方針で行う．

Voxel 分割はオブジェクト単位として事前に実施することにより，訓練中に voxel 分割のコストが発生するのを抑止する．

同一オブジェクトの中にポリゴンサイズの極端な不均衡が存在する場合は，その大小や分布状況に応じて複数のオブジェクトに分離して取り扱う．

Voxel 分割は訓練の開始前に十分な準備時間をもって完了できると想定されるため，voxel 分割そのものに要する時間は本稿の検討対象外とする．

4　評価

本節においては，前節の方針に基づいた voxel 分割と，レイトレーシングの所要時間について評価を行う．

4.1　評価環境

ほとんどの性能評価には，開発環境を用いる．開発環境の主要諸元を表 2 に示す．また HIL における画像生成装置の主要諸元を表 3 に示す．PassMark 社のベンチマークで示される性能値（Average CPU Mark）[18][19] としては，ほぼ同程度の CPU 性能となっており，用いるコンパイラのバージョンも同一とした．

なお画像生成装置の主記憶が開発環境の 2 倍となっているが，画像生成装置においては複数のタスクが並行して稼働するため，1 つの OIS プロセスが開発環境と比べて 2 倍の主記憶を占有できる訳ではない．

4.2　取り扱うデータとレンダリング条件

評価に用いた形状モデルを表 4 に示す．これらは実際の訓練に供されたリュウゴイドの形状モデル及びそれらと同一の手法で生成された形状モデルである．均一分割モデルは，いずれも同一の仮想小惑星（リュウゴイド）に由来しており，その相違はポリゴン分割の程度にある．すなわち，表の下位に位置するモデルは直近上位のモデルに対して，ポリゴン（三角形）を 4 分割した上で，一段階詳細な形状生成と更なるボルダー配置を施したモデルとなっている．

不均一分割モデルも，概形はリュウゴイドに準じている．名称末尾「M」のモデルは，規模の異なるポリゴンサイズが混在した形状である．この内，局所的な詳細ポリゴンを抽出したものが，末尾「S」のモデルであり，残りの大型ポリゴンで全体形状を構成するのが，末尾「L」のモデルである．また N31ML と N31MS は常に 1 組で用いられており，この組を「N31ML+S」と表現する．同様に N151ML と N151MS の組も「N151ML+S」と表現する．

各シーンにおいて入力として与えられる，主要なダイナミクス情報を表 5 に示す．ここで位置の単位は [m]，姿勢はクォータニオン（虚部，実部の順）で表現されている．

なお「F-Theta」のシーンは，F-θ 撮像系を模擬するために参考のため加えたものである．本稿で評価したカメラの中では，DCAM3 が最もこれに近い．映像出力としては，全天周映像（ドームマスター [21]）のフォーマットで使用される形式でもある．

生成する画像は，オーバーサンプリング無しの，縦 1024 画素，横 1024 画素の画像とし，画素値計算には簡易 Hapke モデルを用いる．

係るデータに基づいてレンダリングしたシーンの例を示す．均一分割モデルのレンダリング結果を図 12 に示す．同図は 369M モデルを用いた例であるが，均一分割の各モデルにおいては，ポリゴンサイズの大小を除いて，同図と同様のレンダリング結果となる．

不均一分割モデルのレンダリング結果を図 13 に示す．同図は N31ML+S モデルを用いた例であるが，不均一分割の各モデルにおいては，ポリゴンサイズの大小を除

表 4: 形状モデルの概略

名称	ポリゴン		備考
	個数 [千]	サイズ	
仮想小惑星，均一分割モデル			
331K	331	4m	直径 約 900m
1.3M	1,344	2m	以下同様
5.5M	5,450	1m	
22M	22,417	50cm	
92M	92,238	25cm	
369M	368,955	13cm	
仮想小惑星，不均一分割モデル			
N31M	30,598		混在
N31ML	22,728	50cm	全景
N31MS	7,870	8mm	詳細化部分
N151M	150,779		混在
N151ML	22,728	50cm	全景
N151MS	128,051	2mm	詳細化部分
探査機形状モデル			
hayabusa2	213	混在	約 4m×6m×3m

表 5: 各シーンの主要ダイナミクス情報等

ONC-W1 a, 不均一分割モデル	
カメラ	ONC-W1[17]
探査機位置	0, 0, 458
探査機姿勢	0.36017, 0.47590, 0.78029, 0.18692
ONC-W1 b, ONC-W2	
カメラ	「ONC-W1 b」: ONC-W1 「ONC-W2」: ONC-W2[17]
探査機位置	-194.69,-6.35,500.27
探査機姿勢	0.79954, 0.08406, 0.27253, 0.52858
ONC-T	
カメラ	ONC-T[12]
探査機位置	-194.69, -6.35, 9000.27
探査機姿勢	0.79954, 0.08406, 0.27253, 0.52858
DCAM3，F-Theta	
カメラ	「DCAM3」: DCAM3[20] 「F-Theta」: F-θ 撮像系模擬
探査機位置	-971, 0, 528
探査機姿勢	0.57485, 0.35713, 0.65536, 0.33542

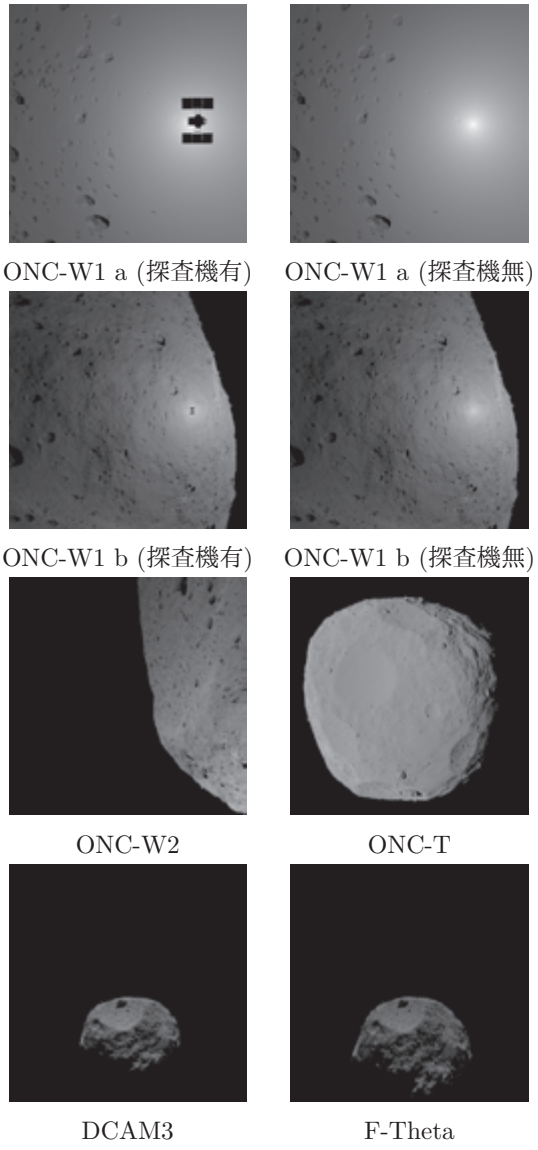


図 12: 評価に用いたシーン（均一分割モデル）

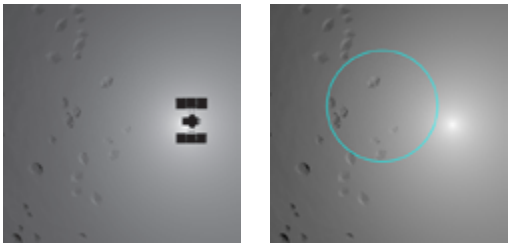
いて，同図と同様のレンダリング結果となる．同図中の青丸は詳細化されたポリゴン（N31MS）の範囲を示す．

4.3 Voxel 分割の評価

本節では均一分割モデルを用いて，voxel 分割の評価を行う．

4.3.1 Voxel 分割に伴う voxel 数の増加

Voxel 分割の回数と分割後の voxel 数の関係を図 14 に示す．横軸が voxel 分割の回数，縦軸が分割後の voxel 数を表している．1 段階前の voxel 数との比を図 15 に示す．横軸が voxel 分割の回数，縦軸は voxel 分割後の voxel 数と，一段階前の voxel 分割時の voxel 数との比を表している．これらより画像生成装置に用意される小惑



探査機有　　探査機無

図 13: 評価に用いたシーン（不均一分割モデル）

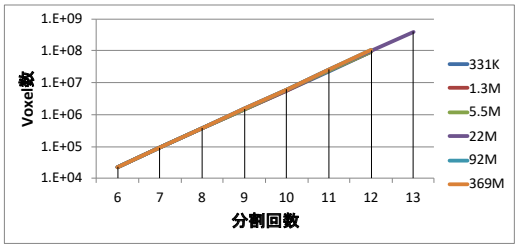


図 14: Voxel 分割の回数と分割後の voxel 数

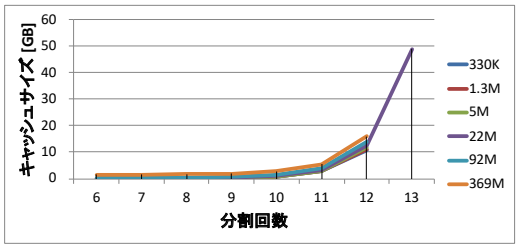


図 16: Voxel 分割の回数とキャッシュサイズ

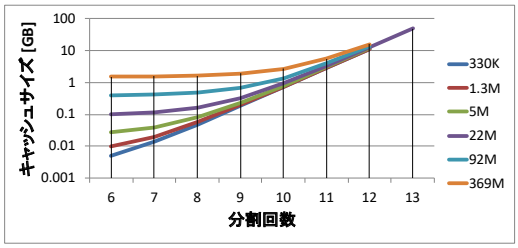


図 17: Voxel 分割の回数とキャッシュサイズ（対数スケール）

星形状モデルについては，voxel 分割の都度，4 倍程度 voxel 数が増加していることが見て取れる．これは，それぞれの voxel 分割にあたって，所属するポリゴンを持たずに消失した voxel が半数程度存在したと考えられる．

4.3.2　キャッシュサイズ

OIS において，voxel 分割は関連する一連のレンダリングに先立って実施され，その結果がメモリ上に常駐した状態でレンダリングの指示を待つ．これらのデータは，同一形状モデルである限りにおいては再利用可能であり，不使用時はファイルシステム等にキャッシュしておくことが可能である．係るキャッシュサイズを見積もった結果を図 16，図 17 に示す．キャッシュの内容は，各 voxel の情報と，それに属する一連のポリゴンの番号である．

横軸が voxel 分割の回数，縦軸が分割後のキャッシュサイズを表している．この図によると，主記憶容量と比較して，voxel 分割回数が 11 回を超える辺りからキャッシュサイズが急激に増加している．メモリ占有量を考えると，画像生成装置における voxel 分割の回数は 12 回程度が現実的な上限になると考えられる．

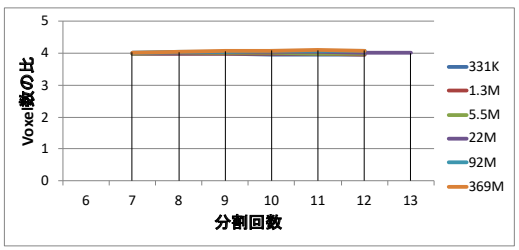


図 15: Voxel 分割後の voxel 数の比
（1 段階前の voxel 分割と比較）

4.3.3　計算量の見積

ここで，複数の voxel に重複して所属するポリゴンは比較的少ないと仮定してレイトレーシングに係る計算量のオーダを見積もる．

ポリゴン数を M とすると，voxel を n 回分割した際に 1 つの voxel 内に所属するポリゴン数の平均は，前節の結果より，およそ $M/(4^n)$ となると考えられる．voxel 階層 1 回の追跡に係るコストを a とすると n 回分割した voxel の階層（$n+1$ 階層）を追跡するコストは $a(n+1)$ のオーダとなる．Voxel 内のポリゴン追跡に係るコストを b とすると，計算量のオーダ o は式 5 で表される．現実にはこれらの値は a や b のような単純な定数で表現できるものではないが，ここではおおまかな目安を得るために，簡略化した表現を用いる．

$$o = a(n+1) + \frac{bM}{4^n} \tag{5}$$

Voxel 分割回数 n を変化させた時に，係る計算量が最小となるのは，微分が 0 となる付近，すなわち式 6 が成立する付近と考える．

$$a + \frac{bM}{4^n} \log \frac{1}{4} = 0 \tag{6}$$

表 6: Voxel 分割回数の目安

形状モデル	分割回数の目安
331K	8
1.3M	9
5.5M	10
22M	11
92M	12
369M	13
N31M	11
N31ML	11
N31MS	10
N151M	12
N151ML	11
N151MS	12

これを変形すると，式 10 が導かれる．

$$a = \frac{bM}{4^n} \log 4 \tag{7}$$

$$4^n = \frac{bM}{a} \log 4 \tag{8}$$

$$n \log 4 = \log M + \log \left(\frac{b}{a} \log 4 \right) \tag{9}$$

$$n = \frac{\log M}{\log 4} + \frac{\log \left(\frac{b}{a} \log 4 \right)}{\log 4} \tag{10}$$

すなわち，$(\log M)/(\log 4)$ に何らかの値を加減したものが，目安になると考えられる．本稿においては簡単のため，$(\log M)/(\log 4)$ を voxel 分割回数の目安とみなすこととする．表 6 に，本稿で用いた形状モデルに対する voxel 分割回数の目安を示す．

以後，voxel 分割回数を “相対” 値で記す場合は，各形状モデルのポリゴン数から計算した voxel 分割回数の目安に対する差を用いることとする．

ここで，実際に voxel 分割した際にポリゴンが複数の voxel にどの程度重複して含まれる状況であるかを図 18 に示す．横軸は voxel 分割回数の相対値，縦軸は各ポリゴンが所属する voxel 数の平均値を表している．1 つのポリゴンが 1 つの voxel のみに所属する場合に縦軸の値が 1 となる．現実には voxel 分割回数の相対値が-2 を超える辺りから縦軸の値が急増し，以後は指数関数的に増加している．前述の計算量のオーダの仮定は voxel 分割回数が少ない時にのみ成立するものであることが，この図から読み取れる．一方で同仮定から導き出した voxel 分割回数の目安を基準に横軸を整理することで，異なるポリゴン数のモデルに対して，グラフの傾向を比較することが容易になることがわかった．

ただしこれは，広く一般的な形状モデルに対して，この目安が適切であることを主張するものではない．

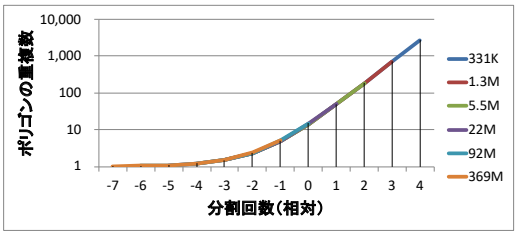


図 18: ポリゴンの重複数

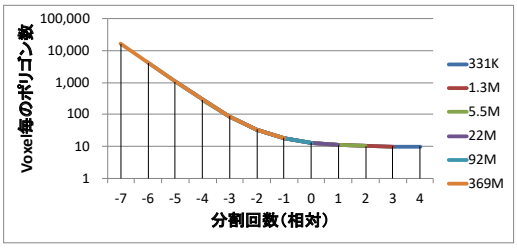


図 19: Voxel 毎のポリゴン数（平均値）

4.3.4 Voxel に含まれるポリゴン数

図 19 には，各 voxel に含まれるポリゴン数の平均値を示す．横軸は voxel 分割回数の相対値，縦軸は voxel 毎のポリゴン数（平均値）を表している．Voxel 分割回数の相対値が-2 を超えた辺りから voxel 毎のポリゴン数の減少が鈍化し，相対値が 0 以上になると，voxel 毎のポリゴン数はほぼ一定となる．これはすなわち，voxel 分割回数の相対値が 0 付近を超えると，voxel 追跡のコストのみが増加し，voxel 内のポリゴン追跡に係るコストは減少しなくなる傾向にあると考えられる．

4.3.5 Voxel 分割に関するまとめ

前述の結果から，メモリ使用効率の観点では voxel 分割の回数が 12 を超えない範囲が妥当であり，また所要時間の観点では voxel 分割の相対値が 0 を超えない範囲が妥当であると考えられる．

4.4 レンダリングの所要時間に係る評価

続いて，4.2 節のデータに基づいて，各シーンのレンダリング所要時間の評価を行う．

ここで測定対象とする装置は Linux システムであり，測定以外のプロセスを完全に排除することは困難である．そのため，異常な測定値が結果に影響しないように，レンダリングするシーン毎に 30 回繰り返して所要時間を計測し，その中で所要時間の短い 25 サンプルを計測結

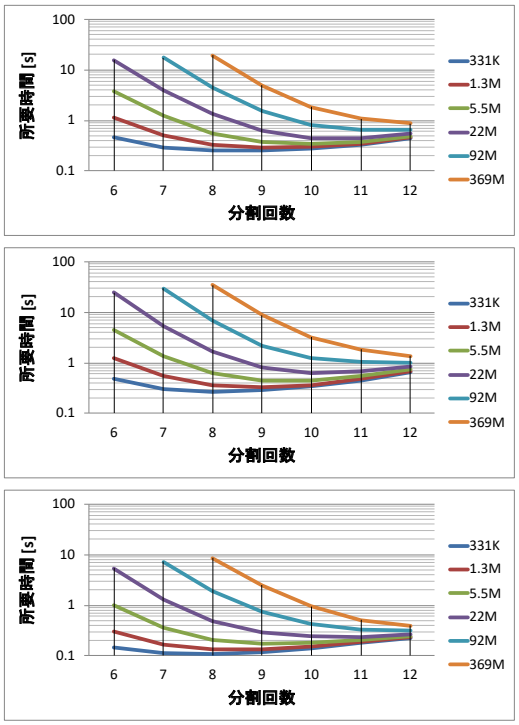


図 20: 各シーンのレンダリングに係る所要時間
（小惑星形状のみ）
上: ONC-W1 a，中: ONC-T，下: DCAM3

果として採用することとする．この回数は経験則によるものである．

4.4.1　均一分割モデルの個別評価（探査機無）

小惑星形状モデルのみを用いて，均一分割モデルのレンダリングに係る所要時間を計測した結果を図 20 に示す．横軸は voxel 分割回数，縦軸は所要時間を表している．以後の結果も含めて，例として 3 種類のシーンの結果を示しているが，他のシーンについてもおおまかな傾向は同様であった．

また開発環境と画像生成装置（HIL）との実行結果比較を図 21 に示す．比較に用いたのは “ONC-W1 a” シーンの “5.5M” モデルである．画像生成装置は開発環境と比べて他プロセスの影響が大きいため，あくまでも参考値となるが，概ね開発環境に若干勝る実行速度となっている．傾向として，画像生成装置と開発環境に決定的な相違はないと考えられるので，以後は開発環境を用いて評価を行うこととする．

図 20 の結果に基づいて，voxel 分割回数の相対値を横軸にとった比較を図 22 に示す．縦軸は，計測対象とした voxel 分割回数の中で所要時間が最短となったものを 1 とおいて相対値を記したものである．全体として，voxel 分割回数の相対値が-1 もしくは 0 の時に所要時間が最短となり，そこから離れるに従い所要時間が増加す

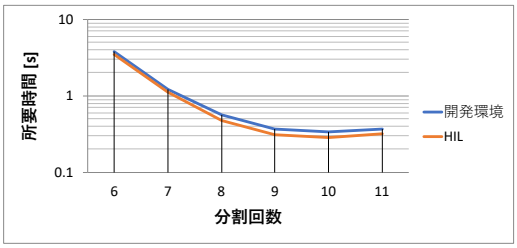


図 21: レンダリングに係る所要時間比較
（ONC-W1 a, 5.5M, 小惑星形状のみ）

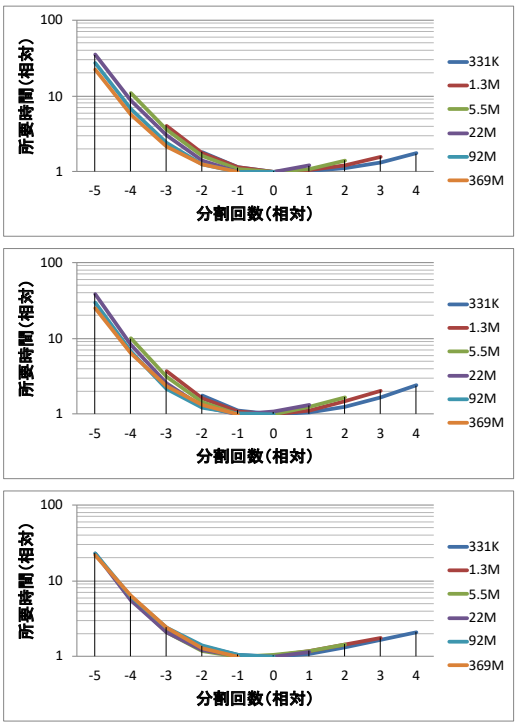


図 22: 各シーンのレンダリングに係る所要時間
（相対値，小惑星形状のみ）
上: ONC-W1 a，中: ONC-T，下: DCAM3

る傾向にある．これは 4.3.5 節と矛盾しない結果である．

“369M” モデルは voxel 分割回数の相対値の上限が-1（絶対値としては 4.3.5 節で述べた，妥当と思われる voxel 分割回数の上限である 12）であったため，voxel 分割回数の相対値が 0 以上となるまで計測している他のケースとは状況が異なるが，他のケースからの類推として，これ以上 voxel 分割回数を増やしてメモリを消費したとしても，所要時間に目立った改善は見られないと推測される．

これらの所要時間について，ばらつきを求めた結果を図 23 に示す．縦軸の標準偏差は，所要時間の平均値に対する百分率で示したものである．

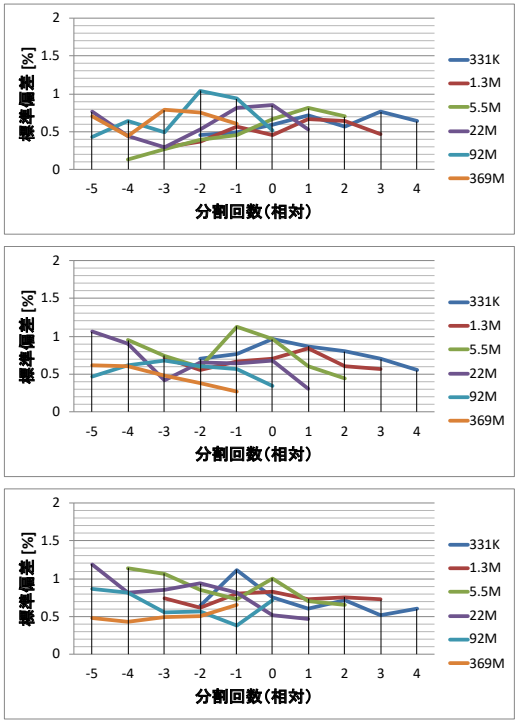


図 23: レンダリングに係る所要時間の標準偏差
（相対値，小惑星形状のみ）
上: ONC-W1 a，中: ONC-T，下: DCAM3

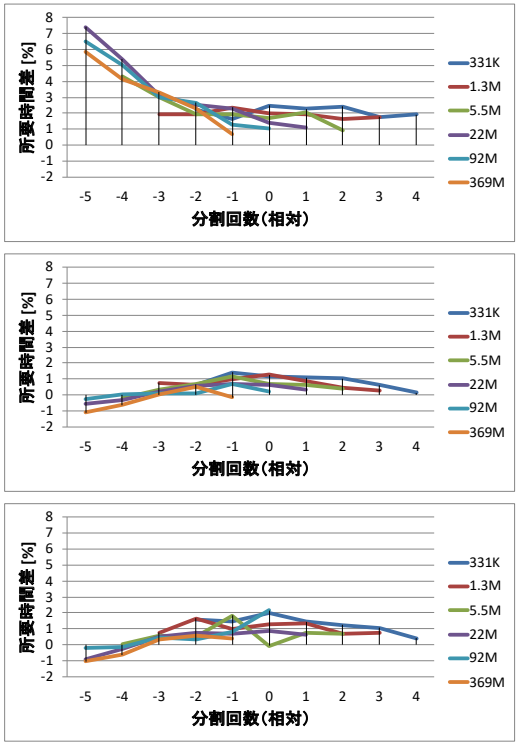


図 24: レンダリングに係る所要時間の差
（相対値，探査機形状の有無の差）
上: ONC-W1 a，中: ONC-T，下: DCAM3

4.4.2 均一分割モデルの個別評価（探査機有）

探査機の形状モデルを併用した場合と小惑星形状モデル単独の場合とで，所要時間（平均値）の差を求めた結果を図 24 に示す．図 25 には，探査機の形状モデルを併用した場合の所要時間のばらつきを示す．図 24 における縦軸の値は，所要時間（平均値）の差を小惑星形状モデル単独での所要時間（平均値）に対する百分率で表したものである．所要時間が最短となることが期待される，voxel 分割回数（相対）が-1 から 0 の付近では，その差は高々2%程度であった．これは所要時間計測の際のばらつきに対して十分大きな値ではなく，探査機の形状モデルを併用することの所要時間増は，実用上，大勢に影響ないと考えられる．

“ONC-W1 a” のケースは，voxel 分割回数が少ない領域において，所要時間が目立って増加する傾向が見られたが，所要時間が最も短くなると期待される，voxel 分割回数の相対値が-1～0 の付近では，さほどの増加傾向が見られず，これも実用上の支障は無いと判断される．

4.4.3 均一分割モデルの評価（まとめ）

以上をまとめて，各シーンのレンダリングに係る所要時間の内，最短となるケースを図 26 に示す．図中に参考値として記載した線は，ポリゴン数の 1/4 乗の傾きを持っている．これらの比較から，レンダリング時間の増分は，概ねポリゴン数の 1/4 乗の増加に抑えられていると考えられ，大規模形状モデルのレイトレーシングにあたっても，高々1 秒余りの時間でレンダリングを終了する結果となった．

簡易画像は概ね “1.3M” から “5.5M” 付近の規模の形状モデルを用いることが想定されているが，この付近のモデルは 1 秒に対して十分早い時間でレンダリングを実行できることを示している．この傾向は，画像生成装置を用いた測定結果図 21 からも読み取れるものであり，簡易画像生成に対する要求を満たすものであると考えられる．

また本稿で導入した voxel 分割回数の目安は，ポリゴン数の異なるモデルに対して適切な voxel 分割回数を導く目安として，実際の画像生成装置にも実装され，voxel 分割のパラメター指定の簡略化に寄与している．すなわち，形状モデル毎に voxel 分割回数を指定することなく，相対的な voxel 分割回数（-1, 0 等）を起動時のパラメータに設定するだけで，様々な規模の小惑星形状モデルに対して，概ね良好な実行結果が得られることとなった．

精細画像の生成にはオーバーサンプリングを伴うが，レイトレーシング機能単体で評価する限りにおいては，例えば 4 倍 ×4 倍のオーバーサンプリングを想定しても，RIO 訓練で要求される応答時間（10 分）に対しては十

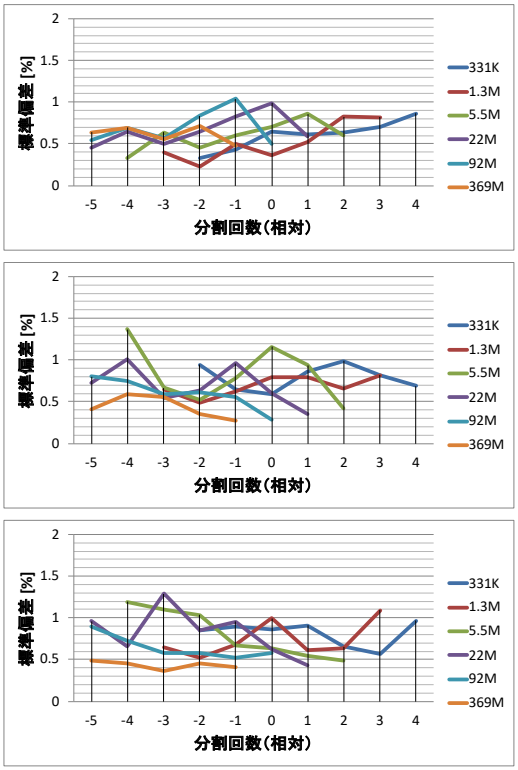
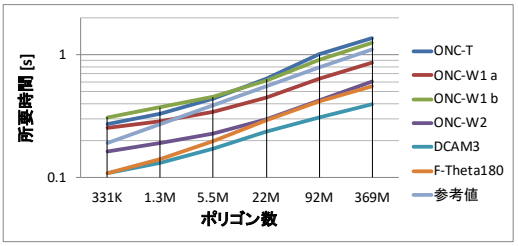
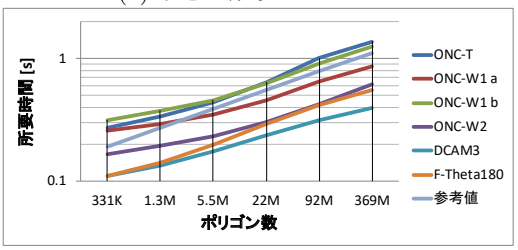


図 25: レンダリングに係る所要時間の標準偏差
（相対値，小惑星形状＋探査機形状）
上: ONC-W1 a，中: ONC-T，下: DCAM3



(a) 小惑星形状モデルのみ



(b) 小惑星形状モデルと探査機形状モデル

図 26: 各シーンのレンダリングに係る所要時間

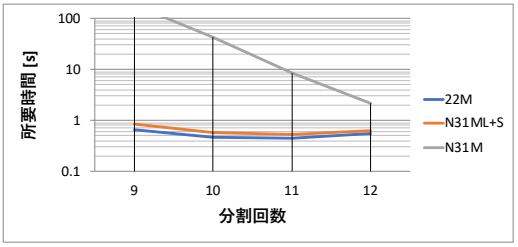


図 27: 不均一形状モデルを用いた場合の所要時間

分高速にレンダリングを終了できる見込みとなった．だが実際には，外部の画素値計算ライブラリ等を含めた実行時間も鑑みつつ何倍のオーバーサンプリングまで許容されるか，HIL を稼働させる中で適切な値を推定する必要がある．

4.4.4　不均一分割モデルの評価

不均一分割モデルは，その用途や生成条件等によって様々な形態が想定されるが，ここでは一例を示すにとどめる．

図 27 に，N31ML+S の “ONC-W1 a” シーンにおける測定結果を示す．この図は小惑星と探査機をシーンに収めた場合の比較結果であるが，小惑星のみの場合も同様の比較結果となった．

所要時間の測定対象としては，N31M（不均一分割モデル），N31ML+S（N31M をポリゴンサイズが均一となるような 2 種類の形状モデルに分離したもの），および N31ML を構成するポリゴンとサイズが最も近い，22M を選定した．横軸が voxel 分割回数，縦軸が所要時間を示している．ここで N31ML+S については，N31ML の voxel 分割回数を代表として示している．N31MS の voxel 分割回数は，それぞれ N31ML に対して 1 回少ない値となっている．N31ML+S にあっては，voxel 分割回数が 11，すなわち表 6 より，voxel 分割回数の相対値が 0 の場合に最も所要時間が短くなっている．N31M にあっては，本稿で想定していた voxel 分割回数の上限である 12 回でも十分に所要時間が短くなっているとは考えられず，所要時間の短縮にためには，さらなる voxel 分割が必要と考えられる．しかしながら，これ以上の voxel 分割はメモリ利用効率の観点で，現実的ではないと考えられる．

この比較結果より，不均一分割モデルをポリゴンサイズに応じた複数のモデルに分離することでポリゴンサイズの局所的な不均一を解消すれば，比較的少ないリソースでレンダリング時間の短縮が図れることが示唆された．

ただし，どのような不均一分割モデルに対してどのような効果があるかについては，更なる比較実験や考察が必要であり，均一分割モデルほどの画一的な voxel 分割の目安が見出されたものではない．

参考までに，同一シーンで N151M と N151ML+S を用いた場合の所要時間比較について，一例を表 7 に示す．この例の Voxel 分割回数（相対）は 0 である．

表 7: 不均一形状モデルを用いた場合の所要時間

モデル	所要時間 [ms]
N151M	56,850
N151ML+S	617

5 あとがき

本稿においては，HIL シミュレータの画像生成装置におけるレイトレーシングソフトウェアの概要とその評価について述べた．

用いられる形状モデルに対して，voxel 分割回数の目安となる簡易の式を導き出すことができたため，画像生成装置におけるコンフィグレーションの構築にあたって，高速化のための試行錯誤を低減できるようになった．

開発環境における評価結果は要求要件を満足するものであり，それと同等以上の環境が用意された画像生成装置においても十分な性能を発揮できると期待されるものである．しかしながら画像生成装置において，複数プロセスが並行稼働する環境下にあっては，実際にコンフィグレーションを調整しながら，システム全体として所望の機能・性能を発揮できるようにするための努力が欠かせない．

一方でポリゴンサイズが不均一な形状モデルに対しては，事前にオブジェクトを分離する等の手作業が発生している．このようなケースでもオブジェクトを分離することなく，現実的なメモリ使用量でレンダリング速度を向上させることも，改善の余地として残されていると考えられる．

本稿で述べた光学機器模擬の手法は,「はやぶさ 2」ミッションに閉じず，広く関連する課題等への応用も期待されるものであり，今後も様々な要求に応じた開発が続くと考えられる．

参考文献

[1] 佐伯孝尚, リュウグウ近傍での運用をシミュレートする— RIO 訓練, ISAS ニュース 2018 年 4 月号, 2018.4

[2] H. Yabuta, N. Hirata, R. Honda, Y. Ishihara, K. Kitazato, M. Komatsu, A. Miura, K. Matsumoto, T. Morota, T. Nakamura, A. Nakato, T. Noguchi, T. Okada, N. Sakatani, S. Sugita, S. Tachibana, S. Tanaka, E. Tatsumi, S. Watanabe, T. Yamaguchi, Y. Yamamoto, LSS AA Team (Hayabusa2 Project), “Hayabusa2 landing site selection (LSS) training: Summary report of scientific evaluation”, 49th Lunar and Planetary Science Conference, The Woodlands, Texas, March 2018.

[3] 三浦 昭, 山口 智宏, 本田 理恵, 横田 康弘, 千秋 博紀, 北里 宏平, 山本 幸生, 中村 智樹, 野口 高明, 本田 親寿, 山田 学, 和田 浩二, 佐伯 孝尚, 田中 智, “はやぶさ 2 着陸地点選定運用訓練のための仮想 Ryugu データ作成”, 日本惑星科学会 2017 年秋季講演会予稿集, H15, 2017.9.27-29

[4] 渡邊誠一郎，石原吉明，田中智，山口智宏，三浦昭，山本幸生，平田成， 諸田智克，坂谷尚哉，北里宏平，松本晃治，藪田ひかる，“はやぶさ 2 LSS データ解析検討チーム, はやぶさ 2 着陸点選定訓練: データ解析・検討”, 日本惑星科学会 2017 年秋季講演会予稿集, H16, 2017.9.27-29

[5] 平田成, 平田直之, 杉山貴亮, 金丸仁明, 千秋博紀, 北里宏平, 田中小百合, 西川直輝, 渡邊誠一郎, 石原吉明, 田中智, 山口智宏, 三浦昭, 山本幸生, はやぶさ 2 プロジェクト LSS データ解析検討チーム, “はやぶさ 2 着陸点選定訓練における小惑星形状復元”, 日本惑星科学会 2017 年秋季講演会予稿集, P53, 2017.9.27-29

[6] 杉山貴亮, 平田成, 渡邊誠一郎, 石原吉明, 田中智, 山口智宏, 三浦昭, 山本幸生, LSS データ解析検討チーム, “形状復元ソフト Photoscan のはやぶさ 2 着陸点選定訓練における活用の成果と課題”, 日本惑星科学会 2017 年秋季講演会予稿集, P54, 2017.9.27-29

[7] 坂谷 尚哉, 千秋 博紀, 荒井 武彦, 滝田 隼, 岡田 達明, 田中 智, 平田 成, 三浦 昭, 山口 智宏, はやぶさ 2 TIR チーム, 坂谷尚哉, “はやぶさ 2 着陸地点選定訓練データを用いた熱慣性決定手法の評価”, 日本惑星科学会 2017 年秋季講演会予稿集, P56, 2017.9.27-29

[8] 諸田智克, 千秋博紀, 横田康弘, 坂谷尚哉, 巽瑛理, 杉田精司, 本田理恵, 本田親寿, 山田学, 平田成, 平田直之, 三浦昭, 山口智宏, 田中智, はやぶさ 2 光学航法カメラチーム, “はやぶさ 2ONC データによる Ryugu 表面ラフネスの推定　着陸点選定訓練データを用いた検討”, 日本惑星科学会 2017 年秋季講演会予稿集, P57, 2017.9.27-29,

[9] 杉田精司,*諸田智克, 巽瑛理, 本田理恵, 本田親寿, 山田学, 神山徹, 横田康弘, 平田成, 三浦昭, 山口智宏, 田中智, 山本幸生, 石原吉明, 千秋博紀, “はやぶさ 2 光学航法カメラチーム, はやぶさ 2ONC データの処理システムとプロダクト　着陸点選定訓練データの例”, 日本惑星科学会 2017 年秋季講演会予稿集, P58, 2017.9.27-29

[10] JAXA はやぶさ 2 プロジェクト, “2018 年の小惑星リュウグウ到着にむけて 小惑星探査機「はやぶさ 2」の近況”, 小惑星探査機「はやぶさ 2」の記者説明会 (17/12/14)，2017.12.14

[11] Y. Takei, T. Takahashi, T. Yamaguchi, T. Saiki, A. Miura, H. Takeuchi, Y. Tsuda, “A Hardware-in-the-loop simulator for deep space touchdown operation training of Hayabusa2”, 68th International Astronautical Congress (IAC), Adelaide, Australia, September 2017.

[12] S. Kameda, H. Suzuki, T. Takamatsu, Y. Cho, T. Yasuda, M. Yamada, H. Sawada, R. Honda, T. Morota, C. Honda, M. Sato, Y. Okumura, K. Shibasaki, S. Ikezawa, S. Sugita, “Preflight Calibration Test Results for Optical Navigation Camera Telescope (ONC-T) Onboard the Hayabusa2 Spacecraft”, Space Science Review, Volume 208, Issue 1-4, pp.17–31, July 2017.

[13] T. Mizuno, T. Kase, T. Shiina, M. Mita, N. Namiki, H. Senshu, R. Yamada, H. Noda, H. Kunimori, N. Hirata, F. Terui, Y. Mimasu, “Development of the laser altimeter (LIDAR) for Hayabusa2”, Space Science Reviews, Volume 208, Issue 1-4, pp.33–47, July 2017.

[14] 五味広美，佐々修一，山口功，二宮哲次郎，濱田吉郎，“Shape from shading による月平坦地の実時間検出”，航空宇宙技術研究所報告，TR-1447，2002.8

[15] 三浦昭，山本幸生，吉川真，“「はやぶさ 2」運用補助のための可視化手法 — 影の描写の検討 —”，宇宙科学情報解析論文誌 第五号，pp.133-148，JAXA-RR-15-006，2016 年 3 月.

[16] はやぶさ 2 プロジェクト，“探査機の影と衝効果”，http://www.hayabusa2.jaxa.jp/topics/20180915/，2018 年 11 月 20 日現在.

[17] H. Suzuki, M. Yamada, T. Kouyama, E. Tatsumi, S. Kameda, R. Honda, H. Sawada, N. Ogawa, T. Morota, C. Honda, N. Sakatani, M. Hayakawa, Y. Yokota, Y. Yamamoto, S. Sugita, “Initial inflight calibration for Hayabusa2 optical navigation camera (ONC) for science observations of asteroid Ryugu”, Icarus 300, pp.341–359, 2018

[18] PassMark Software, “PassMark - Intel Core i9-7900X @ 3.30GHz - Price performance comparison”, https://www.cpubenchmark.net/cpu.php?id=3035, 2018 年 11 月 20 日現在.

[19] PassMark Software, “PassMark - [Dual CPU] Intel Xeon E5-2640 v4 @ 2.40GHz - Price performance comparison”, https://www.cpubenchmark.net/cpu.php?id=2752&cpuCount=2, 2018 年 11 月 20 日現在.

[20] K. Ishibashi, K. Shirai, K. Ogawa, K. Wada, R. Honda, M. Arakawa, N. Sakatani, Y . Ikeda, “Performance of Hayabusa2 DCAM3-D camera for short-range imaging of SCI and ejecta curtain generated from the artificial impact crater formed on asteroid 162137 Ryugu (1999 JU_3)”, Space Science Reviews, Vol. 208, Issue 1-4, pp.213–238, July 2017.

[21] IMERSA / AFDI Dome Standards Group, “FULLDOME FILE SPECIFICATIONS”, https://www.imersa.org/fulldome-file-specifications, 2018 年 11 月 20 日現在.