Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research 65 (2020) 1951–1964

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Fast spacecraft solar radiation pressure modeling by ray tracing on graphics processing unit

Patrick W. Kenneally *,[1], Hanspeter Schaub [2]

University of Colorado, Boulder, United States

Received 4 March 2019; received in revised form 14 December 2019; accepted 23 December 2019
Available online 7 January 2020

Abstract

A novel method is presented to evaluate on the graphics processing unit (GPU) the force and torque on a spacecraft due to solar radiation pressure. The method employs efficient ray tracing techniques, developed in the graphics rendering discipline, to resolve spacecraft self-shadowing and reflections at faster than real-time computation speed. The primary algorithmic components of the ray tracing process which contribute to the method's computational efficiency are described. These components include two-level bounding volume hierarchy acceleration data structures, fast ray to bounding box intersection testing using the slab intersection algorithm and fast triangle intersection testing using the Möller-Trumbore algorithm. Spacecraft material optical properties are represented as a combination of Lambertian diffuse and ideal specular reflections. Both diffuse and specular ray-surface interactions are modeled. The approach is implemented using C++ and OpenCL and executed on a consumer grade GPU. Model validation is presented comparing ray traced force and torque values to the same quantities produce by a faceted analytic model. Numerical results illustrate the impact of self-shadowing on the force and torque calculation, and demonstrate the fast computational speed that is enabled with this implementation.

Keywords: Radiation pressure; Ray tracing; General purpose gpu

1. Introduction

Effective orbit determination, maneuver and mission design and mission numerical simulations require tools that enable accurate modeling of the spacecraft dynamical system. Solar radiation pressure (SRP) is the momentum imparted to a body by impinging solar photons. The SRP force and torque is often the dominant non-conservative force for missions operating at and above the Low Earth Orbit (LEO) region (Vallado, 2007). This dominance motivates the pursuit for improved knowledge of the resultant SRP forces and torques through the modeling and analysis of a spacecraft's dynamics (Fliegel and Gallini, 1996; Marshall et al., 1992). For example, to maintain a desired spacecraft attitude, the SRP-induced torque on a spacecraft is absorbed using reaction wheel devices. Under the influence of sustained torque in a constant direction, the reaction wheels will reach an operational maximum angular rate and require desaturation. The requirement to perform desaturation operations may be mitigated through a judicious choice of reaction wheel orientation or more typically by a momentum unloading process using spacecraft thrusters (O'Shaughnessy et al., January 2014).

Effective modeling of the SRP induced perturbation of a spacecraft enables mission designers to consider SRP a valuable actuator rather than a disturbance. Such a novel

* Corresponding author.
E-mail addresses: patrick.kenneally@colorado.edu (P.W. Kenneally), hanspeter.schaub@colorado.edu (H. Schaub).
[1] Graduate Research Assistant, Aerospace Engineering Sciences Department, University of Colorado, Boulder, United States.
[2] Glenn L. Murphy Chair of Engineering, Aerospace Engineering Sciences Department, Fellow of AAS and AIAA, United States.

https://doi.org/10.1016/j.asr.2019.12.028
0273-1177/

use of the SRP force in maneuver and mission design is exemplified by the MErcury Surface, Space ENvironment, GEochemistry and Ranging (MESSENGER) mission. The MESSENGER mission designers employed a solar sailing technique to perform each trajectory change maneuver (TCM) and accurately target each of the mission's six planetary flyby maneuvers. Using SRP as the TCM actuator allowed the MESSENGER team to perform TCM's with more accuracy and finer control due to the smaller magnitude of the SRP induced ΔV (O'Shaughnessy et al., 2009). Recent research is exploring how to create control formulations to exploit the SRP forces further to assist with attitude and orbital considerations (Mashtakov et al., 2018; Kenneally, 2016). In all these applications a fast tool to model the complex SRP forces on general shapes that include reflections and self-shadowing is critical to validate the dynamics and control solutions.

A survey of the current landscape of SRP research reveals a variety of approaches. The nature of the approaches can be characterized as analytic, semi-analytic or empirical. Whereas analytic models rely only on pre-launch engineering information, empirical models are constructed post-launch using flight data. Commonly, a semi-analytic model is used during a mission. These models are comprised of both analytic and empirical components with tunable parameters. Prior to flight, the tunable parameters are determined using an analytic model. Following launch, the parameters are incorporated into a parameter estimation process which tunes the model to more closely match flight data. Prominent examples of the three modeling approaches include the ROCK42 analytic model (Fliegel and Gallini, 1989), the various semi-analytic approaches which combine the Extended CODE Model (ECOM) with analytic box and wing models (Montenbruck et al., 2015) and the Jet Propulsion Lab (JPL) empirical model (Bar-Sever, 1997).

The simplest analytic model employed is referred to as the cannonball model. The cannonball model assumes the spacecraft presents a constant cross sectional area to the sun and the SRP force is strictly pointing opposite sun-spacecraft direction (Lucchesi, 2002). Increased accuracy in analytic models is often achieved by representing the spacecraft as an approximation of various volumes. A common approximation is to model the spacecraft bus and solar panels as a box and panels respectively (Rim et al., 2006). Modeling fidelity is improved by increasing the number of spacecraft surfaces with which the incident solar radiation interacts (Marshall and Luthcke, 1994). Additionally, the individual reflection, absorption and emission material characteristics are kept distinct for each surface and set based on known spacecraft material properties. However, common among shape approximation methods is that they are augmented and become semi-analytic models where much of the modeling uncertainty is delegated to a parameter estimation process and the model is 'tuned' post-launch to more accurately match spacecraft tracking data.

Early ray tracing approaches employed Monte Carlo ray tracing techniques to resolve SRP (Klinkrad et al., 1991). Notably, Ziebart developed an analytic modeling approach based on a Whitted ray tracing technique for the assessment of SRP force analysis of spacecraft in the GLONASS constellation (Ziebart, 2001). Ziebart's method precomputes the body forces over all 4π steradian attitude possibilities, outputting results to a lookup table which can then be used in analysis such as online simulation and precise orbit determination (POD) campaigns. Ziebart's approach is also capable of modeling self-shadowing and multiple ray reflections by ray tracing a spacecraft model that comprises a set of volume primitives (boxes, cylinders etc.). While Ziebart's method is able to resolve spacecraft articulations, these kinematics must be known prior to model evaluation. There is currently no approach reported in the literature which is able to capture unplanned arbitrary spacecraft articulations and time evolution of material optical properties during execution of an online spacecraft dynamical simulation. A range of further efforts to use ray tracing to produce reference force and torque for use in POD are demonstrated by Darugna et al. (2018) and Li et al. (2018).

McMahon and Scheeres extend Ziebart's approach to a semi-analytic model by aggregating the resultant SRP forces into a set of Fourier coefficients of a Fourier expansion (McMahon and Scheeres, 2010). The resulting Fourier expansion is available for both online and offline evaluation within a numerical integration process. Evaluation of the Fourier expansion in numerical simulation demonstrates successful prediction of the periodic and secular effects of SRP. Additionally, the Fourier coefficients may replace spacecraft material optical properties as parameters estimated during the orbit determination effort.

More recently, methods that make use of the parallel processing nature of GPUs have been developed. Tanygin and Beatty employ modern GPU parallel processing techniques to provide a significant reduction in time-to-solution of Ziebart's 'pixel array' method (Tanygin and Beatty, 2016). An OpenCL ray tracing implementation is demonstrated by Grey et al. (2017). Kenneally et al. (2016) demonstrate the use of the OpenGL vector graphics application programming interface (API) to dynamically evaluate the force of the incident solar radiation across a spacecraft triangular mesh model with many thousands of triangular facets.

Wetterer et al. (2014) demonstrate that accurate bidirectional reflection distribution functions (BRDF) representation is necessary in simulating the long duration propagation of spacecraft dynamics. A material's BRDF governs the amount of impinging solar radiation absorbed and reflected and the directions in which the radiation is reflected. A typical BRDF description is comprised of diffuse Lambertian and ideal, 'mirror like', specular reflection portions (Guarnera et al., May 2016). Absent in previous ray tracing approaches is the modeling of scattered ray propagation due to diffuse ray reflections.

Regarding the field of ray tracing techniques, the video game industry's pursuit for more vivid artificial worlds has driven the development of highly optimized vector processing software and graphics processing unit (GPU) computer hardware capable of carrying out many thousands of floating point operations in parallel (Owens et al., 2008). In the animation and movie industry, the pursuit of photo-realistic modeling has pushed the techniques employed in ray tracing algorithms to produce rendering results at near real-time computation speeds. Two key themes in ray tracing research are the pursuit of algorithmic techniques and efficient hardware utilization, which increase computing efficiency and therefore reduce the time of a photo-realistic model rendering (Wald and Slusallek, 2001). The method presented here leverages advances in ray tracing techniques and the OpenCL API to produce a ray tracing SRP modeling approach at faster than real-time computation speeds. In this work real-time computation speed is defined as the compute time required to simulate the spacecraft's dynamics due to SRP being less than the actual time being simulated. For example, in a spacecraft dynamic simulation with an integration rate of 10 Hz an algorithm must return a result in less than 0.1 s to be considered faster than real-time. Of course, the computational load is dependent on the number of rays cast and the time step of any dynamic simulation where a smaller time step may be less than the compute time. Therefore, the description of this method being faster than real-time is given on the basis of appropriate modeling fidelity for a vehicle's dynamical time spans. OpenCL is an API and C based programming language which facilitates the execution of massively-parallel computations on heterogeneous computation devices. OpenCL is a cross-platform standard for parallel programming across a range of devices including multi-core CPUs, GPUs and other computation accelerators.

This work improves on previous analytic modeling approaches in two key areas. The first improvement is the direct modeling of diffuse ray reflections. While detail on vehicle material BRDF definitions is limited, modeling the physical reflection behavior allows for varied and more accurate BRDF selection. The second, and primary, improvement is the implementation of proven ray tracing algorithms, implemented with OpenCL on the GPU, to provide a faster-than-real-time computation duration.

In the remainder of this paper the fundamental components and validation results are outlined for the OpenCL ray tracing methodology. In the following section the ray-surface interaction theory is established and the idealized combined Lambertian and specular BRDF introduced. The following section provides a summary of the often overlooked formalism for particle tracing for SRP. With the theoretical basis complete the next section details the modeling steps of model definition, intersection testing, and force and torque evaluation. The penultimate section presents modeling validation by comparing the results from the faceted evaluation methodology with those from the OpenCL ray tracing approach. Additionally, multiple ray bounces are validated and a demonstration of the impact of resolving multiple ray bounces on the final SRP force value. Finally, the paper discusses computational performance by characterizing the methods ability to produce evaluations at faster than real-time speeds.

2. Ray-surface interactions

The resultant force vector due to an impinging light ray is coupled to the nature of the ray's interaction with the spacecraft surface materials. The total spatial distribution of a reflected light ray is described by the reflecting material's BRDF. The BRDF is defined as the ratio of reflected radiance $\mathrm{d}L_\mathrm{r}$ to the incident radiance $\mathrm{d}E_\mathrm{i}$ (Nicodemus et al., 1977).

The geometry of the reflection interaction is shown in Fig. 1. The ray intersection point on a surface is denoted as $\mathbf{x}$, with ω_i the direction of the incident radiation and ω_o the direction of the outgoing radiation. An orthonormal basis is constructed using the unit normal to the surface $\hat{\boldsymbol{n}}_\mathbf{x}$, with $\hat{\boldsymbol{t}}_\mathbf{x}$ and $\hat{\boldsymbol{s}}_\mathbf{x}$ completing the basis. The normalized vector, $\hat{\boldsymbol{h}}_\mathbf{x}$ is in the direction of the angular bisector of $\boldsymbol{\omega}_\mathrm{o}$ and $\boldsymbol{\omega}_\mathrm{i}$, and is defined by $\hat{\boldsymbol{h}}_\mathbf{x} = (\boldsymbol{\omega}_\mathrm{o} + \boldsymbol{\omega}_\mathrm{i})/|\boldsymbol{\omega}_\mathrm{o} + \boldsymbol{\omega}_\mathrm{i}|$. In this paper a particular notation convention from the field of computer graphics is used for the various directional quantities denoted by ω. For instances where the quantity is solely directional the quantity is denoted as ω. Where it is meaningful to be used as a vector the quantity is written as $\boldsymbol{\omega}$ and is assumed to be a unit vector.

It is assumed that the incident light-surface interactions are occurring in the optical linear regime. Under this linear regime it has been shown experimentally that there is a proportional relationship between exitant radiance and irradiance, $\mathrm{d}L_\mathrm{o}(\omega_\mathrm{o}) \propto \mathrm{d}E(\omega_\mathrm{i})$. This allows for the development of the bidirectional reflectance distribution function, $f_\mathrm{r}(\omega_\mathrm{i} \rightarrow \omega_\mathrm{o})$, given in Eq. (1), where the proportionality relationship describes the observed radiance leaving a reflecting surface in the direction ω_o and the projected solid angle defined as $\mathrm{d}\sigma^\perp(\omega) = |\omega \cdot \hat{\boldsymbol{n}}|\mathrm{d}\sigma(\omega)$.

$$f_\mathrm{r}(\omega_\mathrm{i} \rightarrow \omega_\mathrm{o}) = \frac{\mathrm{d}L_\mathrm{o}(\omega_\mathrm{o})}{\mathrm{d}E(\omega_\mathrm{i})} = \frac{\mathrm{d}L_\mathrm{o}(\omega_\mathrm{o})}{L_\mathrm{i}(\omega_\mathrm{i})\mathrm{d}\sigma^\perp(\omega_\mathrm{i})} \tag{1}$$

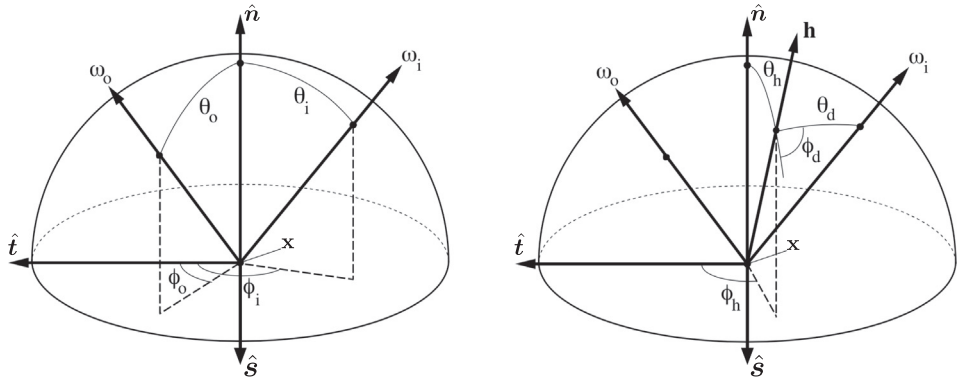
The relationship between the outgoing radiance and the incoming radiance for a particular optical surface is described at Eq. (2).

$$\mathrm{d}L_\mathrm{o}(\omega_\mathrm{o}) = \mathrm{d}L(\omega_\mathrm{i})f_\mathrm{r}(\omega_\mathrm{i} \rightarrow \omega_\mathrm{o})\mathrm{d}\sigma^\perp(\omega_\mathrm{i}) \tag{2}$$

Integrating Eq. (2) yields the total radiance, over the hemisphere, leaving a surface area element as (Veach, 1997)

$$L_\mathrm{o}(\omega_\mathrm{o}) = \int_{S^2} L(\omega_\mathrm{i})f_\mathrm{r}(\omega_\mathrm{i} \rightarrow \omega_\mathrm{o})\mathrm{d}\sigma^\perp(\omega_\mathrm{i}). \tag{3}$$

This work employs physically plausible BRDFs. A physically plausible BRDF adheres to the symmetry expression given at Eq. (4) and energy conservation condition given by Eq. (5).



(a) BRDF is parameterized by the intuitive (θ_i, ϕ_i) and (θ_o, ϕ_o)

(b) BRDF is parameterized as function of the half angle (θ_h, ϕ_h) and a difference angle (θ_d, ϕ_d)

Fig. 1. Illustrations of two common BRDF geometry descriptions.

$$f_r(\omega_i \to \omega_o) = f_r(\omega_o \to \omega_i) \quad \text{for all} \quad \omega_i,\ \omega_o \tag{4}$$

$$\int_{\mathscr{S}^2_o} f_r(\omega_i \to \omega_o) \mathrm{d}\sigma^{\perp}(\omega_o) \leqslant 1 \quad \text{for all} \quad \omega_i \in \mathscr{S}^2_i \tag{5}$$

For a large majority of materials a BRDF can be described as the combination of a specular component and diffuse component. Whereas specular reflection is due to surface reflection, diffuse reflection is due to subsurface scattering and surface microgeometry. While subsurface scattering contributes to the generation of diffuse reflections this work does not model internal material refraction nor transmission between transparent layers. As a result the contribution of subsurface scattering is represented by adding a diffuse term to the specular term giving the complete BRDF description

$$f_r(\omega_i \to \omega_o) = \rho R_d + s R_s \tag{6}$$

where ρ and s are the proportions of the surface behaving as a diffuse and specular respectively and $\rho + s = 1$.

The BRDF which will be computed directly is the typical combination of diffuse Lambertian and ideal specular mirror-like reflection. This BRDF expression is given in Eq. (5)

$$f_r(\omega_i \to \omega_o) = d\left(\frac{\rho}{\pi}\right) + s\left[\frac{F_0 \delta(\hat{\boldsymbol{\omega}}_i - \hat{\boldsymbol{r}})}{\cos\theta_i}\right], \tag{7}$$

where F_0 is the Fresnel reflection coefficient, ρ the diffuse scaling constant of the material. The outgoing mirror reflected direction $\hat{\boldsymbol{r}}$ is given by

$$\hat{\boldsymbol{r}} = 2(\hat{\boldsymbol{\omega}}_i \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}} - \hat{\boldsymbol{\omega}}_i. \tag{8}$$

3. Radiation pressure particle tracing formulation

In this section a formalism is introduced which describes the process of generating and evaluating a set of weighted sample rays. This formalism initially introduced by Veach (1997), rigorously describes the ray tracing algorithm employed in this work. Veach presents a general description of how an estimate of some quantity can be computed with respect to some measure, by generating a set of weighted sample rays. Here, the quantity will be the solar radiation pressure force due to radiance incident on the spacecraft. Two key assumptions will greatly simplify the resulting description. The first assumption is that ray samples are taken uniformly from a plane wave, discretized into smaller square areas, of collimated radiation. The side length of a discretized unit area is referred to as the ray's resolution (e.g. 1 mm × 1 mm unit area, is a ray resolution of 1 mm). The second assumption is that the mechanism controlling ray continuation is a predetermined maximum number of ray-surface interactions rather than a probabilistic measure such as Russian Roulette (Veach, 1997).

This path tracing algorithm produces an unbiased estimate of the integral of force over the spacecraft mesh surface. In the case of radiation pressure the sample weight α_i is the radiation throughput of each sample ray. The unbiased estimate is constructed by generating a set of weighted sample rays

$$(\alpha_i, \boldsymbol{r}_i), \tag{9}$$

where $\boldsymbol{r}_i$ is a ray and α_i the corresponding weight/throughput. The goal is to produce samples that give an unbiased representation of the equilibrium force F over the spacecraft that holds for any importance function W_e. This estimator is given in Eq. 10 where N is the number of rays. Again the first assumption dictates that all samples are given equal importance, therefore W_e is trivially set as 1.

$$E\left[\frac{1}{N}\sum_{n=i}^{N} \alpha_i W_e(\boldsymbol{r}_i)\right] = \langle W_e, F \rangle, \tag{10}$$

To begin the particle tracing process, an initial ray is defined as $\boldsymbol{r}_0 = (\mathbf{x}_0, \omega_0)$ where $\mathbf{x}_0$ is an origin vertex and ω_0 a direction. Each sample ray has a corresponding weight

α and each ray has an initial state of $(\alpha_0, \boldsymbol{r}_0)$. For radiation transport the initial weighting is the incident radiance throughput

$$\alpha_0 = \frac{L(\boldsymbol{r}_0)}{p_0(\boldsymbol{r}_0)} \tag{11}$$

where $p_0(\boldsymbol{r}_0)$ is the probability density from which $\boldsymbol{r}_0$ is sampled. Here, the first assumption, of uniformly distributed rays, simplifies the throughput to simply

$$\alpha_0 = L(\boldsymbol{r}_0). \tag{12}$$

Given the current state of a ray $(\alpha_i, \boldsymbol{r}_i)$, if $i < k$, with k being the maximum number of ray-surface interactions, the ray is continued. If an intersection occurs, $\mathbf{x}_{i+1}$ is the intersection point of the ray $\boldsymbol{r}_i = (\mathbf{x}_i, \omega_i)$. A random scattering direction ω_{i+1} is chosen from the BRDF, $f_r(\mathbf{x}_{i+1}, \omega_{i+1} \rightarrow -\omega_i)$, according to a probability density function approximating the BRDF $p_{i+1}(w_{i+1})$. Employing importance sampling, the next ray throughput (weight) is computed as

$$\alpha_{i+1} = \alpha_i \frac{f_r(\mathbf{x}_{i+1}, \omega_{i+1} \rightarrow -\omega_i)|\hat{\boldsymbol{\omega}}_{i+1} \cdot \hat{\boldsymbol{n}}(\mathbf{x}_{i+1})|}{p_{i+1}(\omega_{i+1})}. \tag{13}$$

From the recursive relationship in (13), the ray continuation step is repeated until the maximum number of ray-surface interactions is reached and results in a set of sample rays with weights given as

$$\alpha_i = L(\mathbf{x}_0, \omega_0) \prod_{j=0}^{i-1} \times \frac{f_r(\mathbf{x}_{j+1}, \omega_{j+1} \rightarrow -\omega_j)|\hat{\boldsymbol{\omega}}_{j+1} \cdot \hat{\boldsymbol{n}}(\mathbf{x}_{j+1})|}{p_{j+1}(\omega_{j+1})}. \tag{14}$$

The process computes a set of sample rays $\boldsymbol{r}_0, \ldots \boldsymbol{r}_k$, each with weight α_i.

4. Modeling steps

In the following sub-sections the key modeling steps are introduced in the order shown in Fig. 2. The grey colored CPU stages indicate operations which are computed only once for an entire simulation, while the blue CPU stages are computed at each simulation time step. The gold colored stages are carried out on the GPU. The parallel GPU computing environment requires two primary changes to the serial ray tracing algorithm. The first is required because recursive function execution is not available in current GPU execution environments. The second change is that rather than making the algorithm parallel by pixel as is suggested by the serial implementation, the algorithm should be parallel by ray. The Single Instruction Multiple Device (SIMD) GPU execution environment is most efficient when each compute unit on the GPU is actively working. In the case that the algorithm is parallel by pixels, rays from certain pixels will terminate sooner than others. This leaves compute units inactive resulting in poor utilization of the GPU's computing resources. Rather, an algorithm which is parallel by rays cast may discard terminated ray paths at each iteration. Continued ray paths are then repacked for a second iteration ensuring marshaled compute units are maximally active. Fig. 3, exemplifies these two algorithmic changes and shows the notional state of two OpenCL GPU work groups stepping through two ray bounces and three iterations of computing ray-surface interactions. Depicted in both Fig. 3(a) and (b) are two GPU work groups each containing four compute units. In Fig. 3(a) the relationship between data and a single compute unit is configured as one ray's data allocated to one compute unit, whereas in Fig. 3(b) the relationship is one pixel's data per compute unit. In both configurations the two work groups are initially fully occupied with either eight active rays or eight pixels. Imagining that of the eight rays traced, if four intersections are found, then four ray continuations are computed in a second iteration. A GPU is most efficient when the compute units within a work group are maximally utilised. Mapping pixel data to compute units produces inactive compute units and underutilized work groups. Mapping rays to compute units and compacting ray data after each iteration yields maximally utilized work groups.

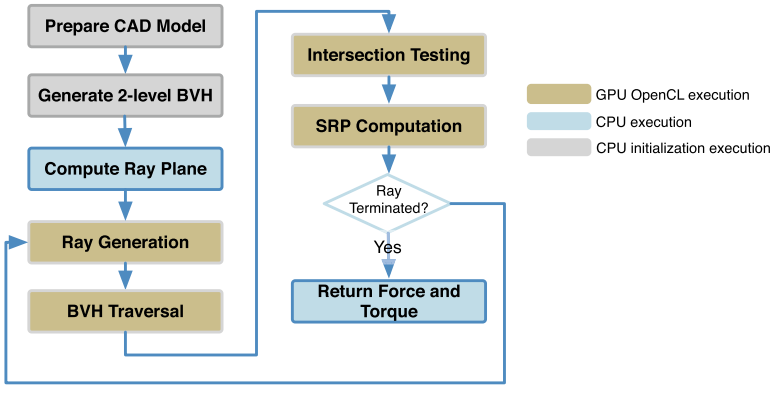


Fig. 2. Parallel ray tracing algorithm steps.

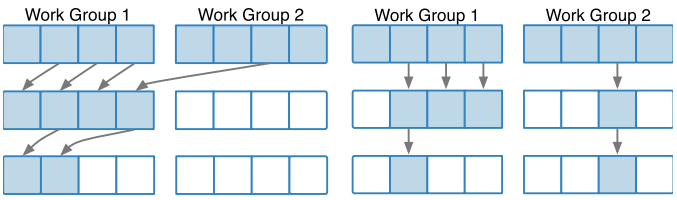


Fig. 3. Increased GPU Work Group occupancy when tracing by ray rather than by pixel.

4.1. Spacecraft mesh model definition

This presented approach uses a triangulated mesh model to approximate the spacecraft shape with high geometric accuracy. A triangulated mesh model provides a commonly available and consistent model definition input to the method and removes the need for code which handles a multitude of other primitive types. The model data format used in this work is the Wavefront Object (OBJ),[3] however, any of the many triangulated mesh file formats are easily employed. The OBJ format stores vertex positions and facet normal vectors in lists. Vertices are defined as x, y, z and w where w is an optional scaling component and defaults to 1.0. Primitive normal vectors may be provided as x, y, z coordinates. If normal vectors are not defined in the file the import code will generate consistent facet normal vectors using the counter clockwise ordered list of vertices defining the facet.

The OBJ format is accompanied by one or more Material Template Library (MTL) files. In the computer graphics context, the MTL file defines common material properties associated with model shading or rendering. For this work a number of the MTL file variables have their meaning overloaded. Two key examples of this are the `Kd` and `Ks` parameters which indicate the RGB color mixture of the diffuse and specular optical phenomena for a material. Here, these variables are used as the diffuse and specular reflection coefficients associated with Eq. (7). Overloading these variables allows for rapid manipulation of the spacecraft mesh model's material properties. Finally, model validation operations such as dimensions, being 'water-tight', and without non-manifold geometry are also easily carried out via existing Python scripts in the Blender 3D animation tool[4].

4.2. Ray plane generation

At each time update a dedicated OpenCL ray creation kernel generates a new wave of ray vectors. The ray plane is divided into unit areas determined by the resolution chosen by the user. For example, a 2 m x 1 m plane can be divided into areas of 1 mm x 1 mm giving a plane of 2000×1000 squares, producing 2,000,000 rays. Ray intersection testing must occur in the same coordinate frame in which the spacecraft vertices are defined. As a result, the ray vectors are mapped from sun-frame S to the body-frame B using the $[BS]$ rotation matrix (Schaub and Junkins, 2014).

4.3. Bounding volume hierarchy

A bounding volume hierarchy (BVH) reduces the ray intersection search space and therefore the required number of ray intersection tests. The use of BVH testing has a long heritage for ray tracing within the computer graphics discipline. In the context of SRP, ray tracing BVH structures have appeared only recently in Li et al. (2018). In the case of a naive BVH implementation, the BVH is a static data structure computed once at the beginning of an evaluation. Were a portion of the spacecraft mesh model to undergo some transformation, then the BVH structure would need to be rebuilt entirely. This is a computationally costly process. The key difference presented in this work is the application of a two-level BVH structure (Smits, 1999). As shown for the Aqua spacecraft in Fig. 4, a mesh model is divided into a number of sub-meshes each with their own bounding box. A BVH hierarchy structure is generated for each sub-mesh of the spacecraft model. These sub-BVH data structures are then combined into a coarse top level BVH which contains all the model's sub-meshes as demonstrated in Fig. 5. Associated with each sub-mesh is a homogeneous transformation matrix. If a sub-mesh is to be transformed then that sub-mesh's homogeneous transformation matrix is inversely applied to each ray being tested for an intersection on that sub-mesh's volume. Mapping rays in this way is a relatively computationally cheap operation carried out on the GPU and therefore avoids the computational cost of rebuilding the BVH.

An efficient method of traversing the BVH is a key aspect of the development of real-time SRP ray tracing. This implementation uses a depth-first search array BVH traversal method as described by Smits (1999) and demonstrated in Fig. 6. For the depth-first search array, if bounding volume A is intersected, then the next node to be tested is the next node sequentially in the array, node B. If the bounding volume at node B is not intersected, the next node is found by following the precomputed skip pointer to the next sibling in the array, which for node B is node C. The depth-first search array avoids the function call overhead inherent in a recursive search-tree traversal and takes advantage of the fact that the next node in the search-tree can be precomputed and stored with the left-most sibling as a skip-pointer to the next node. An additional benefit to the array type BVH traversal, particularly for large meshes, is the greater memory coherency which yields

[3] http://paulbourke.net/dataformats/obj/.
[4] https://www.blender.org

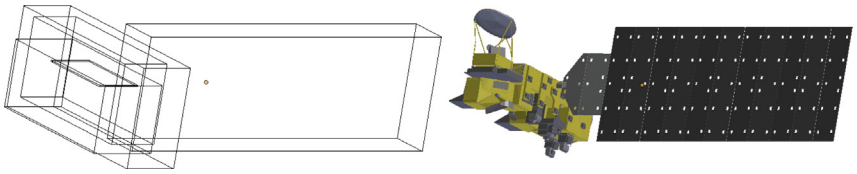
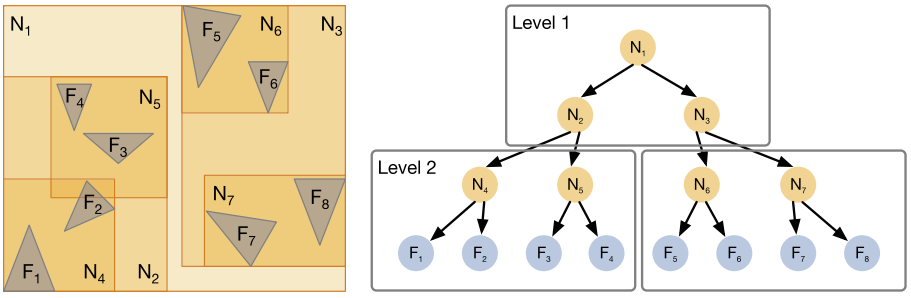
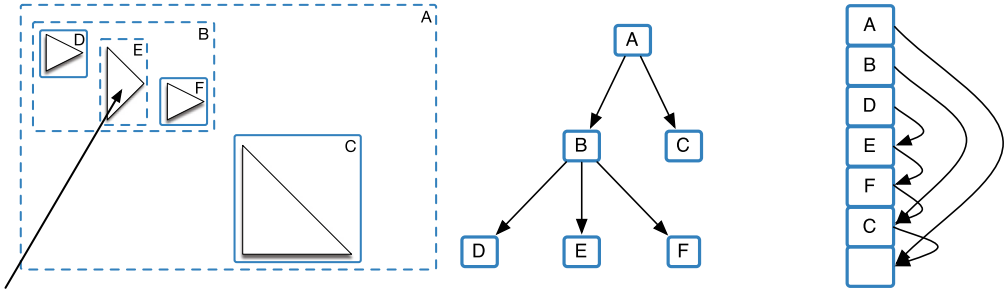


Fig. 4. Illustration of spacecraft bounding volumes.



(a) Bounding volume grouped facets. (b) BVH split into two-levels.

Fig. 5. Two-level BVH allowing for articulation of each of the volumes at the second level.



(a) Notional BVH intersection. (b) Two BVH traversal structures

Fig. 6. A notional BVH illustrated on the left. On the right, the notional BVH shown as a recursive depth-first search-tree and depth-first search array with precomputed node skip pointers.

more efficient contiguous memory accesses on the GPU (Smits, 1999).

4.4. Bounding volume intersection

Bounding volume intersection uses the algorithm originally presented by Kay and Kajiya (Kay and Kajiya, 1986). The algorithm models the bounding box as three sets of parallel planes. The algorithm employs each set of parallel planes as clipping planes. As demonstrated in Fig. 7, once the ray is clipped by each set of planes, any remaining portion of ray inside the bounding volume indicates an intersection. The algorithm is particularly suited to implementation in the GPU environment because it does not require code branching (the execution of divergent code paths based on conditional code statements). This parallel plane algorithm employs the non-branching `min()` and `max()` functions and results in an intersection test with no code branching or division operations.

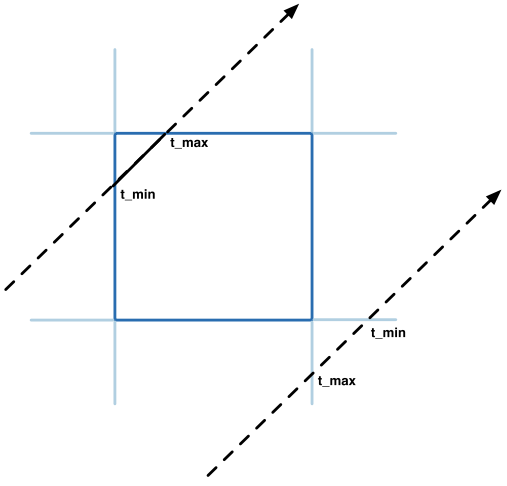


Fig. 7. Example result of the parallel plane bounding box intersection algorithm. For the top left ray intersection, the algorithm returns t_max as greater than or equal to t_min. For the bottom right ray miss, the algorithm returns t_max as less than t_min.

4.5. Triangle facet intersection

The spacecraft mesh model is comprised of many thousands of triangular facets. To compute a ray-triangle intersection, the Möller-Trumbore algorithm is used. The basis of the algorithm is the knowledge that the point of intersection of a line through a triangle in barycentric coordinates (u, v) must lie within coordinate frame's bounds which are easily testable as boolean values. The bounds defined by the barycentric coordinate system require $u \geqslant 0, v \geqslant 0$ and $u + v \leqslant 1$ (Moller and Trumbore, 1997). It is assumed that the spacecraft mesh model does not contain any degenerate triangles. Mesh editing tools such as Blender are used during a model preprocessing step to find and resolve degenerate geometry (non-manifold geometry, degenerate triangles, non-triangle primitives). As a result, this algorithm is a fast and memory-efficient ray-triangle intersection algorithm making it particularly suited for use in the memory constrained GPU computation environment.

4.6. Evaluating ray-surface interaction

Monte Carlo importance sampling is used to evaluate the integral in Eq. (3). In the context of graphics rendering, Monte Carlo ray tracing casts many rays from a single pixel to estimate the radiance received at that pixel. The number of rays cast for a single pixel is given by the quantity N in Eq. (10). With a sufficiently large value for N a good estimate of the scattering equation can be attained and in turn its effect on the resultant SRP force direction and magnitude. However, this typically requires casting many rays per pixel in multiple ray waves. Each ray wave incurs the communication and data overhead of launching the various OpenCL ray generation and tracing kernels. This communication overhead is a significant source of latency in general purpose GPU (GPGPU) programming. To overcome this latency and thus maintain faster than real-time evaluations, it is assumed that features of the spacecraft mesh model surface area are much larger (10's of cm^2) than the ray cross sectional area (mm^2) and that each feature on the spacecraft mesh model possess a common BRDF. As a result, rather than casting $N = 100$ waves of rays at a particular ray resolution e.g. 1 cm^2, a single densely packed wave of rays at ray resolution 1 mm^2 is cast. This densely packed wave of 1 mm^2 rays is equal to the 100 waves of 1 cm^2 rays.

At each ray-surface interaction the ray throughput α_i, its directional force $\boldsymbol{F}_i$ and reflected direction ω_o, must be computed. Due to the importance sampling approach, the throughput of the ray continuation is a function of the BRDF and probability density function from which the outgoing ray direction ω_o is sampled. This importance sampling is accommodated in Eq. (14).

Computing ω_{j+1} and $p_{j+1}(\omega_{j+1})$ for a specular reflection is the reflected direction computed by Eq. (8) and $p_{j+1}(\omega_{j+1}) = 1$. The reflected ray direction for a diffuse interaction is computed by uniformly sampling the hemisphere above the intersection point. The components of ω_{j+1} for a diffuse ray are given by Eq. (15) where ϵ_1 and ϵ_2 are random samples chosen from a uniform distribution (Pharr and Humphreys, 2017).

$$x = \cos(2\pi\epsilon_2)\sqrt{1-\epsilon_1^2} \tag{15a}$$

$$y = \sin(2\pi\epsilon_2)\sqrt{1-\epsilon_1^2} \tag{15b}$$

$$z = \epsilon_1 \tag{15c}$$

The probability of selecting this direction is given by Eq. (16), where θ_o corresponds to the angle marked in Fig. 1.

$$p_{j+1}(\omega_{o_{j+1}}) = \frac{\cos(\theta_o)}{\pi} \tag{16}$$

4.7. Force and torque evaluation

Where an intersection is found, the force on the spacecraft due to the incident throughput of the ray is evaluated in the spacecraft body frame as

$$\boldsymbol{F}_i = \alpha_i \boldsymbol{\omega}_i + \alpha_i \frac{f_r(\mathbf{x}_{i+1}, \omega_{i+1} \rightarrow -\omega_i)|\boldsymbol{\omega}_{i+1} \cdot \hat{\boldsymbol{n}}(\mathbf{x}_{i+1})|}{p_{i+1}(\omega_{i+1})}. \tag{17}$$

The flux of all ray-surface interactions of a ray originating from a pixel is given as

$$\boldsymbol{F}_k = \sum_{i=1}^{K} A\boldsymbol{F}_i \tag{18}$$

where K is the maximum number of permitted ray-surface interactions and A is the cross sectional area of the ray.

The total force is computed by summing the flux components for each wave plane unit area (pixel) and finally multiplying the flux by the solar irradiance and scaling by the spacecraft distance to the sun as shown in Eq. (19).

$$\boldsymbol{F} = \frac{\Phi A U^2}{cr^2} \sum_{k=1}^{N} \boldsymbol{F}_k \tag{19}$$

Here N denotes the total number of pixels, Φ is the radiation flux (solar radiation flux at 1 AU for SRP), AU is one astronomical unit, c the speed of light and r the sun-spacecraft distance.

Following the force computation, the torque $\boldsymbol{L}_k$ contribution of a single intersection is given as

$$\boldsymbol{L}_i = \mathbf{x}_{i+1} \times \boldsymbol{F}_i. \tag{20}$$

5. Model validation

Model validation is performed by computing the percentage error of the force vector for a surface evaluated with the ray tracing approach relative to a surface evaluated with the analytic faceted approach. The magnitude percentage relative error is computed as

$$\frac{|F_{\text{ray}} - F_{\text{facet}}|}{|F_{\text{facet}}|}, \tag{21}$$

and similarly for each force vector component. Additionally, to demonstrate the ray tracing method's ability to capture the diffusely reflected rays, the relative error is computed for a completely specular surface, completely diffuse surface and a mixed specular and diffuse surface. For the completely specular and completely diffuse surface the cube mesh shown in Fig. 8 is evaluated at the sun heading $\hat{\boldsymbol{s}}_{\mathcal{B}} = (1.0, 0.0, 0.0)$. To demonstrate the dependence of force direction and magnitude on ray resolution, the mixed case is evaluated at $\hat{\boldsymbol{s}}_{\mathcal{B}} = (0.7071, 0.7071, 0.0)$. The spacecraft model is a simple cube of side length one meter shown in Fig. 8. Material characteristics are controlled by the coefficients for absorption ρ_a, diffuse ρ_d and specular ρ_s surface interactions as shown in Table 1. The error in the $\hat{\boldsymbol{x}}_{\mathcal{B}}$ component is consistently 2.6×10^{-6} %. This small relative error is attributed to single-precision floating point errors in the representation of the model's facet normal vectors. These small errors manifest the same reflection geometry for each evaluation at different ray resolutions and therefore a small consistent force direction error. The percentage force error $\hat{\boldsymbol{y}}_{\mathcal{B}}$ and $\hat{\boldsymbol{z}}_{\mathcal{B}}$ directions are of order 10^{-14} and less.

For a Lambertian diffuse cube mesh evaluation the error of ray traced force components relative to the faceted force norm are shown in Fig. 9. In the following discussion an error percentage of 1% is employed as a means to compare error percentage across the specular, diffuse and mixed material evaluations. It is evident that as the ray resolution decreases the relative error to the faceted model also decreases to less than half a percent. As the ray resolution decreases, the number of rays being cast to approximate the integral of the diffuse BRDF increases. For ray resolutions less than approximately 5 mm the error remains below one percent. The increased number of rays produces an improved estimate to the integral. This simple test demonstrates that this ray tracing method is accurately capturing the diffuse ray reflections and their impact on the resultant force.

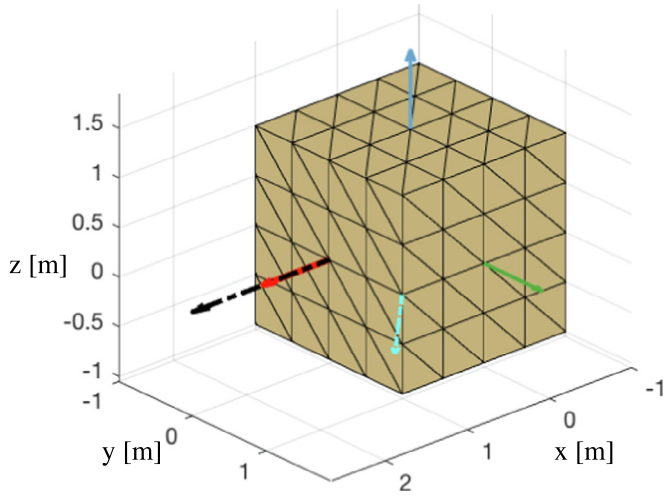


Fig. 8. Test cube spacecraft model. Black and cyan vectors indicate body-frame sun headings evaluated. Red, green and blue vectors denote first, second and third body-frame axes respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
SRP force for faceted evaluations.

Method (ρ_a, ρ_d, ρ_s)	Force [N] $\times 10^{-5}$
Diffuse (0.2, 0.8, 0.0)	−2.783188, 0, 0
Specular (0.2, 0.0, 0.8)	−3.267220, 0, 0
Mix (0.2, 0.4, 0.4)	−2.157385, −2.157385, 0

The error of the ray traced force components relative to the faceted force norm for a mixed (diffuse and specular) material evaluation are shown in Fig. 10. The error relative to the faceted evaluation decreases with increased ray density and remains below one percent for ray resolutions approximately less than 3 mm. It is evident that for the mixed material case a smaller ray size is required to achieve a less than one percent error. For the mixed material the ray resolution required is 3 mm, whereas in the solely diffuse case a ray resolution of 5 mm is sufficient. This is due to the number of rays being probabilistically selected as either diffuse or specular reflection. For example, given a 1 mm ray size, 100 rays will intersect an area of 1 cm^2. If the contributions of diffuse and specular phenomena are equal then it is likely that 50 rays will reflect as specular and 50 as diffuse. This reduces by half the number of diffuse rays approximating the diffuse scattering function. A smaller ray size therefore increases the number of rays intersecting the 1 cm^2 area and provides an improved estimate of the scattering integral of Eq. (3).

6. Multiple ray reflections

To demonstrate that multiple reflections are being effectively captured, the model and sun-spacecraft heading, shown in Fig. 11 is evaluated. A manual computation of the faceted SRP approach is carried out to compute the force direction for the two bounces which will occur for the incoming sun-spacecraft direction. The model material is completely specular with $\rho_s = 0.8$, $\rho_d = 0.0$ and $\rho_a = 0.2$. The resulting faceted force is $\boldsymbol{F}_{\text{facet}} = (0.0, -2.977676, -2.977676) \times 10^{-5}$ [N]. Computing the ray traced evaluation with multiple bounces yields the percent error in force relative to the faceted method of the order 10^{-6} %.

To investigate the importance of resolving multiple ray continuations an evaluation of a high-fidelity OSIRIS-REx spacecraft model is carried out for a uniform distribution of spacecraft sun-headings ${}^{\mathcal{B}}\hat{\boldsymbol{s}}$ in the 4π str attitude space. The ray resolution is set at 2.5 mm. The spacecraft mesh model is shown in Fig. 12 and contains ~14,500 vertices, which define ~26,000 triangular facets. To convey an intuitive sense for the magnitude of the percentage difference, the percentage difference is computed with respect

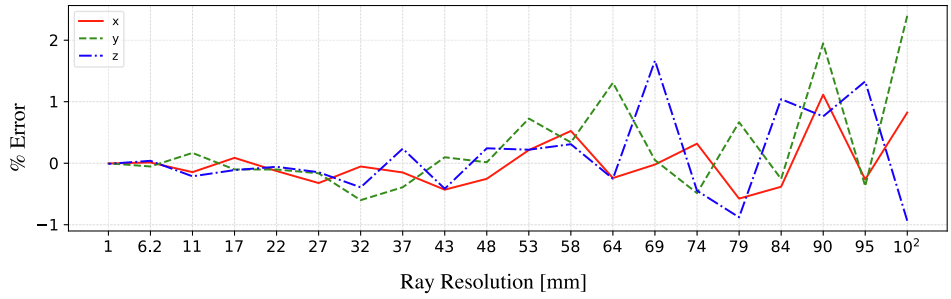


Fig. 9. Error of the ray traced force components relative to the faceted force norm for a diffuse material evaluation.

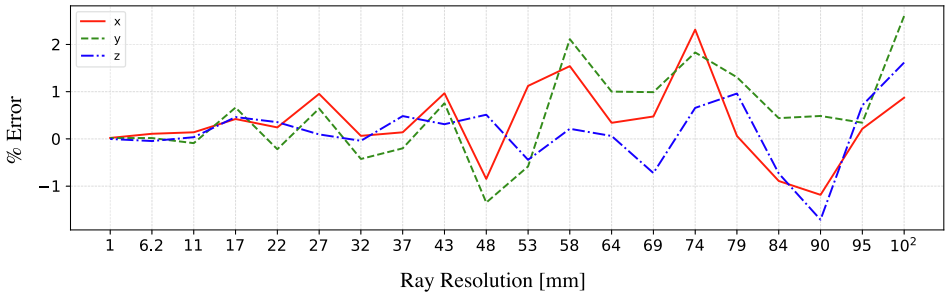


Fig. 10. Error of the ray traced force components relative to the faceted force norm for a mixed (diffuse and specular) material evaluation.

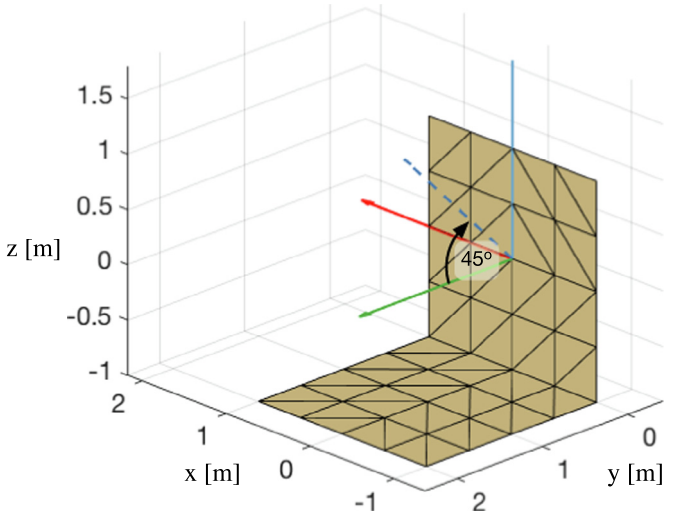


Fig. 11. Right angled test model. The dotted blue vector indicates body-frame sun heading evaluated. Red, green and blue vectors denote first, second and third body-frame axes respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

to a baseline value as shown in Eq. (22). The baseline value is computed using the force or torque computed from a single ray bounce. In both plots the percentage difference is computed according to Eq. (23), where i indicates the number of bounces used to compute the force $\boldsymbol{F}$.

$$F_{\text{base}} = \frac{1}{N}\sum_{n=1}^{N}|\boldsymbol{F}_{\text{n}}| \tag{22}$$

$$F_{\text{i+1,i}} = \frac{|\boldsymbol{F}_{\text{i+1}}| - |\boldsymbol{F}_{\text{i}}|}{F_{\text{base}}} \times 100 \tag{23}$$

The force magnitude percentage difference between the first and second ray bounce is shown in Fig. 13(a), and the difference between second and third shown in Fig. 13(b). It is evident that the majority of the force difference from scattered radiation is captured in tracing rays beyond the first intersection. Fig. 13(a) demonstrates that if one is concerned only with a sun point attitude, which is equivalent to latitude and longitude of approximately $(0°, 0°)$, then the resultant error for computing only the first surface interaction is a small over prediction of approximately 2%. However, for most other attitudes computing only the first intersection produces an under prediction of the force of at least 3% up to almost 8%. The torque magnitude difference between the first and second bounce is shown in Fig. 14(a), and the difference between second and third shown in Fig. 14(b). The same relative difference measure as used in Eq. (23) is used here, yet for torque values rather than force. Fig. 14(a) shows that computing torque for only one bounce results in torque magnitude under prediction of approximately 10% for almost all sun-headings. As with the force magnitude results computing at least two bounces results in a significant reduction in torque magnitude error.

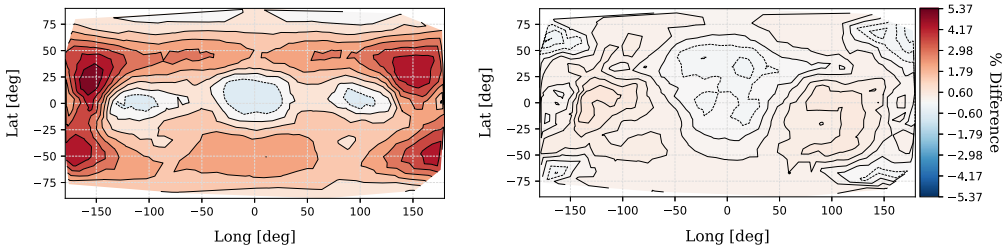
In the interest of paper length, further validation efforts using flight data are described in the recent publication by Geeraert et al. (2019).

7. Computational performance

To recapitulate the metric of faster than real-time computation; for a spacecraft dynamic simulation with an integration rate of 10 Hz the algorithm must return a result in

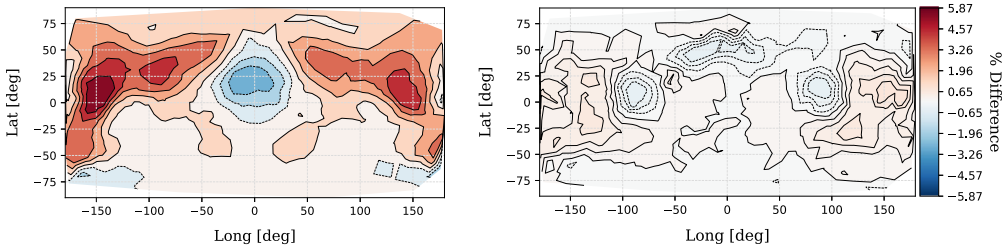
(a) Front view (b) Side view

Fig. 12. OSIRIS-REx model.



(a) Force intersection 2 - 1 (b) Force intersection 3 - 2

Fig. 13. Difference in force magnitude for resolving multiple bounces on high-fidelity OSIRIS-REx.



(a) Torque intersection 2 - 1 (b) Torque intersection 3 - 2

Fig. 14. Difference in torque magnitude for resolving multiple bounces on high-fidelity OSIRIS-REx.

less than 0.1 s to be considered suitable for faster than real-time application. This integration rate is common for spacecraft simulations containing both attitude and orbital motions as the attitude control system requires sub-second control and sensor timings. The computation time, for the same OSIRIS-REx model, is computed for a range of ray resolutions and number of bounces. As shown in Fig. 15, for the modest laptop GPU hardware employed (Radeon Pro 560X), real-time computation speeds can be achieved for ray resolutions of 6 mm or greater while resolving up to ten bounces. Resolving each additional bounce beyond the first is achieved in approximately constant time increase.

The number of rays that a GPU can accommodate is also dependent on the GPU's maximum memory capacity. Given this hardware limitation the significantly slower execution time of the 2 mm case is a result of the algorithm no longer processing all rays in one pass. Rather, the C++ code which supports the interface between the spacecraft dynamics simulation and OpenCL on the GPU, partitions the ray plane into tiles of size sufficient to optimally fill the GPU. Each tile is submitted to the GPU for evaluation and recombined with all other tiles to provide the final ray traced force evaluation. While this tiling allows for higher resolution computation, the data communication latency incurred by submitting multiple tiled ray waves to the GPU, significantly slows the time to solution on the less capable laptop GPU used here. When paired with the force and torque resolutions shown in Figs. 13 and 14 respectively, this result demonstrates that the ray resolution and number of bounces are tunable parameters to this method. These two parameters can be adjusted based on the task to which the modeling method is being applied. One may trade resolution to further increase computational speed.

The computational cost of model evaluation is directly driven by the number of rays traced and the number of

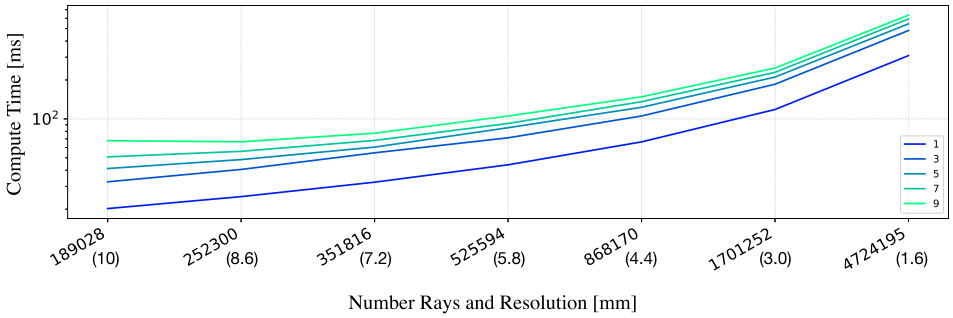


Fig. 15. High-fidelity OSIRIS-REx computation time for a range of resolutions and number of bounces from one to nine.

bounces resolved. An evaluation of the OSIRIS-REx spacecraft model is carried out for a range of ray resolutions and bounces. Fig. 16 demonstrates the attrition rate of rays for each bounce at a given resolution. This evaluation shows that at ten bounces at least 99.7% of all rays have terminated.

An analysis is performed to characterize the execution time of the ray tracing approach on a variety of GPU hardware. The computation times for ray resolutions from 1.5 mm to 10 mm, with a maximum of three ray continuations recorded for an evaluation of the OSIRIS-REx model. The three modest consumer grade GPUs employed are an AMD Radeon Pro 560 4096 Mb, an integrated Intel HD Graphics 630 1536 Mb and a NVIDIA GTX 1070 8 Gb. The computation times for each GPU are shown in Fig. 17. The most computationally capable GPU (NVIDIA) exhibits computation times below 10 ms for ray resolutions of 5 mm or greater. At the lowest resolution of 1.6 mm the computation time is 30 ms. It is interesting to note that a comparison of both the integrated Intel GPU and discrete AMD GPU trace the same performance curve for ray resolutions of less than 6 mm. The integrated GPU benefits from low latency shared memory access with the CPU. However, the discrete AMD GPU has greater compute units than the integrated GPU. Therefore it is clear that the reduced communication overhead of the integrated GPU is matched by the increased computational power of the

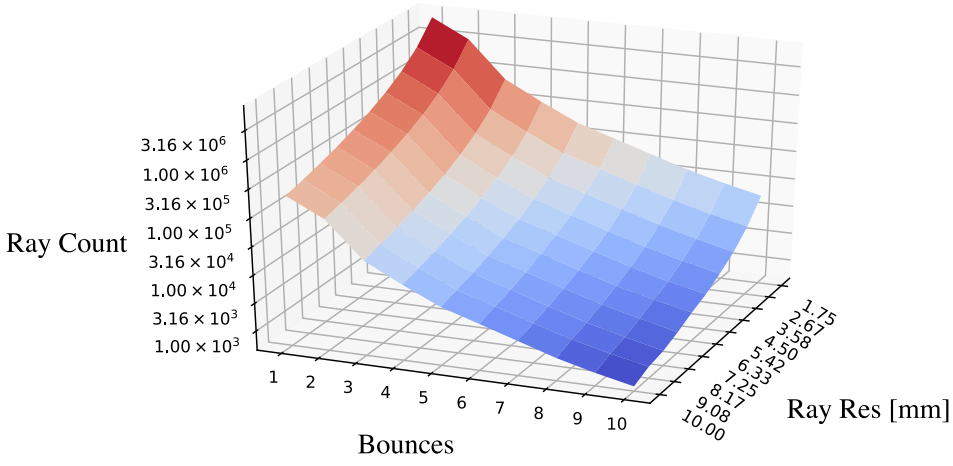


Fig. 16. Number of active rays at each successive bounce for various ray resolutions.

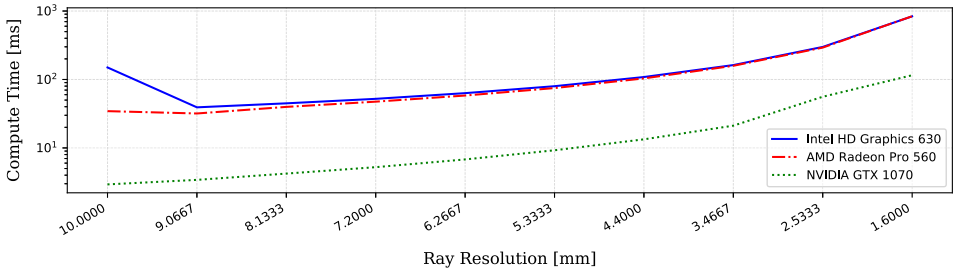


Fig. 17. Execution times for ray resolutions from 0.01 mm to 0.0016 mm, for a maximum of three bounces.

discrete AMD GPU. This demonstrates that the GPU ray tracing approach is appropriately computationally bound rather than communication bound.

8. Conclusions

This paper demonstrates how SRP forces and torques can be resolved for complex spacecraft structures accurately at faster than real-time computation speeds using an OpenCL GPU ray tracing methodology. Previous analytic SRP modeling techniques resolved the dynamics due to diffuse ray-surface interactions but not diffuse reflection. This paper advances analytic SRP modeling approaches by directly generating ray continuation based on the spacecraft surface material BRDFs. Capturing the effect of both diffuse and specularly reflected rays is validated by comparison to the same evaluation computed with the faceted SRP evaluation technique. Additionally, it is shown that with increasing ray density the resultant force vector converges towards a 'truth' evaluation for ideal specular and Lambertian BRDFs. Multiple ray continuation is validated for both a simple mesh model and a complex spacecraft mesh model. The importance of capturing spacecraft radiation self-reflection is revalidated where a spacecraft mesh model evaluation shows that three or four ray continuations is frequently sufficient to reduce the relative force error to less than one percent. Further work will implement complex spacecraft surface material BRDF representations and augment the implementation to accommodate thermal radiation effects.

References

Bar-Sever, Y.E., 1997. New and improved solar radiation models for GPS satellites based on flight data final report. Tech. rep., Jet Propulsion Laboratory.

Darugna, F., Steigenberger, P., Montenbruck, O., Casotto, S., 2018. Ray-tracing solar radiation pressure modeling for QZS-1. Advances in Space Research 62 (4), 935–943. https://doi.org/10.1016/j.asr.2018.05.036.

Fliegel, H.F., Gallini, T.E., 1989. Radiation pressure models for Block II GPS satellites. In: Proceedings of the Fifth International Geodetic Symposium on Satellite Positioning. NOAA, National Geodetic Survey, Rockville, MD, pp. 789–798.

Fliegel, H.F., Gallini, T.E., 1996. Solar force modeling of block IIR Global Positioning System satellites. Journal of Spacecraft and Rockets 33 (6), 863–866.

Geeraert, J.L., Leonard, J.M., Kenneally, P.W., May, C.W., Antreasian, P.G., Moreau, M.C., Aug 11–15 2019. OSIRIS-REx navigation small force models. In: AAS/AIAA Astrodynamics Specialist Conference. Portland, Main.

Grey, S., Marchand, R., Ziebart, M., Omar, R., 2017. Sunlight illumination models for spacecraft surface charging. IEEE Transactions on Plasma Science 45 (1), 1898–1905.

Guarnera, D., Guarnera, G., Ghosh, A., Denk, C., Glencross, M., May 2016. BRDF representation and acquisition. Computer Graphics Forum 35 (2), 625–650, URL http://doi.wiley.com/10.1111/cgf.12867.

Kay, T.L., Kajiya, J.T., 1986. Ray tracing complex scenes. Comput. Graph. (SIGGRAPH '86 Proc.), vol. 20, 4, pp. 169–278.

Kenneally, P.W., 2016. High geometric fidelity solar radiation pressure modeling via graphics processing unit. Master's thesis. University of Colorado, Boulder.

Kenneally, P.W., Schaub, H., Feb. 14–18 2016. High geometric fidelity modeling of solar radiation pressure using graphics processing unit. In: AAS/AIAA Spaceflight Mechanics Meeting. Napa Valley, California, Paper No. AAS-16-500.

Klinkrad, H., Koeck, C., Renard, P., 1991. Key features of a satellite skin force modelling technique by means of Monte-Carlo ray tracing. Adv. Space Res. 11 (6), 147–150.

Li, Z., Ziebart, M., Bhattarai, S., Harrison, D., Grey, S., 2018. Fast solar radiation pressure modelling with ray tracing and multiple reflections. Adv. Space Res. 61 (9), 2352–2365. https://doi.org/10.1016/j.asr.2018.02.019.

Lucchesi, D.M., 2002. Reassessment of the error modelling of non-gravitational perturbations on LAGEOS II and their impact in the Lense-Thirring derivation–part II. Planet. Space Sci. 50 (10–11), 1067–1100, URL http://linkinghub.elsevier.com/retrieve/pii/S0032063302000521.

Marshall, J.A., Luthcke, S.B., 1994. Modeling radiation forces acting on TOPEX/Poseidon for precision orbit determination. J. Spacecr. Rock. 31 (1), 99–105. https://doi.org/10.2514/3.26408.

Marshall, J.A., Luthcke, S.B., Antreasian, P.G., Rosborough, G.W., 1992. Modeling Radiation Forces Acting on TOPEX/Poseidon for precision orbit determination. Tech. rep.

Mashtakov, Y., Tkachev, S., Ovchinnikov, M., 2018. Use of external torques for desaturation of reaction wheels. J. Guid., Control, Dynam. 41 (8), 1663–1674.

McMahon, J.W., Scheeres, D.J., 2010. New solar radiation pressure force model for navigation. J. Guid., Control, Dynam.

Moller, T., Trumbore, B., 1997. Fast, minimum storage ray/triangle intersection. J. Graph. Tools 2 (1), 21–28.

Montenbruck, O., Steigenberger, P., Hugentobler, U., 2015. Enhanced solar radiation pressure modeling for Galileo satellites. J. Geodesy 89 (3), 283–297.

Nicodemus, F.E., Richmond, J.C., Hsia, J.J., Ginsberg, I.W., Limperis, T., 1977. Geometrical considerations and nomenclature for reflectance. Tech. rep., Final Report National Bureau of Standards, Washington, DC. Inst. for Basic Standards.

O'Shaughnessy, D.J., McAdams, J.V., Bedini, P.D., Calloway, A.B., Williams, K.E., Page, B.R., January 2014. MESSENGER's use of solar sailing for cost and risk reduction. Acta Astronaut. 93, 483–489, URL http://linkinghub.elsevier.com/retrieve/pii/S0094576512003876.

O'Shaughnessy, D.J., McAdams, J.V., Williams, K.E., Page, B.R., 2009. Fire Sail: MESSENGER'S use of solar radiation pressure for accurate mercury flybys. In: Guidance and control 2009: proceedings of the 32nd Annual AAS Rocky Mountain Guidance and Control Conference held January 30 to February 4, 2009, Breckenridge, Colorado. pp. 1–16. URL: <http://messenger.jhuapl.edu/the_mission/publications/AAS09-014.pdf>.

Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C., 2008. GPU Computing. Proc. IEEE, vol. 96, 5. URL: <http://ieeexplore.ieee.org.proxy.library.nd.edu/xpl/articleDetails.jsp?arnumber=4490127&navigation=1>.

Pharr, M., Humphreys, G., 2017. Physically Based Rendering: From Theory to Implementation, 3rd Edition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rim, H.-J., Webb, C., Yoon, S., Schutz, B., 2006. Radiation pressure modeling for ICESat precision orbit determination. In: AIAA/AAS Astrodynamics Specialist Conference and Exhibit (August), 1–7. URL: https://doi.org/10.2514/6.2006-6666.

Schaub, H., Junkins, J.L., 2014. Analytical Mechanics of Space Systems, third ed. AIAA Education Series, Reston, VA.

Smits, B., 1999. Efficiency issues for ray tracing. J. Graph. Tools 3 (2), 1–14.

Tanygin, S., Beatty, G.M., 2016. GPU-accelerated computation of drag and Srp forces and torques with graphical encoding of surface normals. In: Conference: 26th AAS/AIAA Space Flight Mechanics MeetingAt: Napa, CA. No. February. pp. 1–20.

Vallado, D., 2007. Fundamentals of Astrodynamics and Applications. Springer, New York.

Veach, E., 1997. Robust monte carlo methods for light transport simulation. Dissertation at the Department of Computer Science of Stanford University 134 (December), 759–764. URL: <http://graphics.stanford.edu/papers/veach_thesis/>.

Wald, I., Slusallek, P., 2001. State of the art in interactive ray tracing. State of the Art Reports, EUROGRAPHICS, 21–42. URL: <http://www.flipcode.net/archives/State-of-the-Artininteractiveraytracing.pdf>.

Wetterer, C.J., Linares, R., Crassidis, J.L., Kelecy, T.M., Ziebart, M.K., Jah, M.K., Cefola, P.J., 2014. Refining space object radiation pressure modeling with bidirectional reflectance distribution functions. J. Guid., Control, Dynam. 37 (1), 185–196, URL http://arc.aiaa.org/doi/abs/10.2514/1.60577.

Ziebart, M., 2001. High precision analytical solar radiation pressure modelling for gnss spacecraft. Ph.D. thesis. University of East London.