AAS 19-531

SPACECRAFT RADIATION PRESSURE USING COMPLEX BIDIRECTIONAL-REFLECTANCE DISTRIBUTION FUNCTIONS ON GRAPHICS PROCESSING UNIT

Patrick W. Kenneally[*] and Hanspeter Schaub[†]

A faster-than-realtime approach is studied to compute the solar radiation pressure forces and torques on a complex time-varying spacecraft model. The method employs ray-tracing techniques, developed in the graphics rendering discipline, to resolve spacecraft self-shadowing, self-reflections and complex surface material optical characteristics 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, including enhancements which take advantage of the fact that the end goal is an accurate force evaluation, not a visual image. A Monte Carlo importance sampling integration method is used to evaluate the integral of complex bidirectional reflectance distribution functions (BRDF) models. The modeling approach process is implemented using C++ and OpenCL and executed on a consumer grade graphics processing unit. A model validation is presented comparing computed values to both the analytic cannonball model and ray traced LAGEOS II spacecraft model. The effects on translational motion due to various BRDF models is compared.

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), the momentum imparted to a body by impinging solar photons, becomes a dominant non-conservative force above low earth orbit (LEO).[1] 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 a 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.[2] Given the importance of SRP, knowledge of the resultant forces upon a body due to SRP are a primary consideration in the modeling and analysis of spacecraft operating above the LEO region.[3,4]

The video game industry's pursuit to create more realistic artificial worlds has resulted in highly optimized vector processing software and graphic processing unit (GPU) computer hardware capable of carrying out many thousands of floating point operations in parallel.[5] Two key themes in ray-tracing research are the pursuit of algorithmic techniques and efficient hardware utilization,

[*]Graduate Research Assistant, Aerospace Engineering, University of Colorado, Boulder.

[†]Professor and Glenn L. Murphy Endowed Chair of Engineering, Aerospace Engineering Sciences, University of Colorado at Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO 80309-0431, AAS Fellow

which increase computing efficiency, and therefore reduce the computation time required to render photo realistic models.[6] The algorithmic techniques developed in the pursuit of photo realistic model rendering have provided the tools which are leveraged in the faster than real-time high geometric fidelity SRP ray-tracing methodology presented in this paper.

The ability to model and compute faster than real-time, the SRP forces and torques on time varying spacecraft structures, presents compelling opportunities. Current SRP evaluation approaches are capable of modeling the resultant force of an articulated spacecraft where the articulation motion is known prior to evaluation. These approaches pre-compute forces and torques which are saved in a lookup table for later evaluation in an online simulation.[7] However, there are many instances in which the articulation motion and the spacecraft state are dependent on the myriad spacecraft control inputs and time varying surface optical properties.

It is evident then that a method of SRP evaluation characterized by an ability to include time varying information of the spacecraft state has potential for a wide range of applications. Effective modeling of the SRP induced perturbation of a spacecraft enables mission designers to consider SRP as a valuable actuator rather than a disturbance. Such a novel 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. Typically TCM's are performed using onboard thrusters. However, using SRP as the TCM actuator allowed the MESSENGER team to perform TCMs with more accuracy and finer control due to the smaller magnitude of the SRP induced acceleration.[2] Additionally, the MESSENGER team was able to reduce fuel and related structural accommodations in the spacecraft design to reduce overall mission cost.[8]

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. The majority of models used during flight guidance and navigation efforts are semi-analytic models. These models are comprised of both analytic and empirical components. Often an analytic model will be employed pre-flight and then post-launch the particular parameters in the model may be incorporated into a parameter estimation process. The OpenCL ray-tracing modeling method presented may be classified as an analytic model and seeks to make extensive use of the pre-launch engineering data available to spacecraft engineering and operations teams.

A commonly used basic analytic model employed, referred to as the cannonball model, is given in Eq. (1). The cannonball model is computed as the surface area upon which radiation is incident A, solar flux $\Phi_{\odot}$, the spacecraft mass M, speed of light c, heliocentric distance to the spacecraft r and the reflection, absorption and emission characteristics of the spacecraft surface which are grouped together within the coefficient of reflection C_r. It is often the case that the C_r parameter is continually estimated and updated by an orbit determination effort. This model was most notably used during the Laser Geodynamics Satellites (LAGEOS) missions and continues to prove useful for initial mission analysis.[9]

$$\boldsymbol{a}_{\odot} = -C_{\mathrm{r}} \frac{A\Phi_{\odot}}{Mc} \left(\frac{1AU}{r} \right)^2 \hat{\boldsymbol{s}} \tag{1}$$

Increased accuracy in analytic models is often achieved by defining the spacecraft as an approximation of various shapes and volumes. A common approximation is to model the spacecraft bus and solar panels as a box and panels respectively. Additionally, the individual reflection, absorption and emission material characteristics are kept distinct for each surface and set based on known spacecraft material properties.[10] However, common among shape approximation methods is that they are augmented as semi-analytic models where much of the modeling uncertainty is lumped in a parameter estimation process and the model is 'tuned' post launch to more accurately match spacecraft tracking data.

Notably Ziebart et al., developed an analytic modeling approach based on ray-tracing techniques, for the assessment of SRP force analysis of spacecraft in the GLONASS constellation.[11] Ziebart's method precomputes the body forces over all 4π steradian attitude possibilities. Ziebart's approach is also capable of modeling self-shadowing and multiple solar radiation ray reflection by ray-tracing a spacecraft model that comprises a set of volume primitives (boxes, cylinders etc.). 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.[10] 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.[12] In previous work presented by the authors the GPU computation environment OpenGL, a vector graphics GPU software interface common in video games, is used to dynamically render the spacecraft model and evaluate the force of the incident solar radiation across a spacecraft structure approximated by many thousands of facets.[13]

The method presented here leverages advances in ray-tracing and the OpenCL set of programming tools to produce a ray-tracing SRP modeling approach at faster than real-time computation speeds. OpenCL is an application programming interface (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 multicore CPUs, GPUs and other computation accelerators. OpenCL facilitates both the read and write of data on processing unit(s) and the submission of code for execution on the processing unit.

In the remainder of this paper the fundamental components and results are outlined for an OpenCL ray-tracing methodology which evaluates complex surface material BRDFs. Next the primary considerations are given to porting the serial and recursive CPU ray-tracing execution to the parallel and iterative GPU execution environment. Additionally, an overview of the ray-tracing methodology implemented is provided. The next section describes the key algorithm components are described and their importance in a GPU ray-tracing implementation discussed. The following section introduces the important consideration of the BRDF with respect to its impact on the resulting SRP force. Finally, the last section presents a validation and a comparison of the effect of different BRDF representations upon the translational motion of a simulated body.

PARALLEL RAY-TRACING

The goal of ray-tracing is to compute the color in a pixel within a view port. The light arriving at the pixel is traced backwards through the scene where its scene interactions are modeled providing the final color of the view port pixel. In a serial execution environment a single, or set of ray reflections are computed using a recursive algorithm for each individual pixel. The recursive algorithm tests for a ray intersection in the scene, and in the case it finds an intersection, the same intersection search algorithm is called again to trace the ray in the new reflected direction. A common recursion termination condition is a preset maximum number of ray reflections.

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. As a result the recursive computation of ray reflections must be achieved through iteration. 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 developers ensure that each compute unit on the GPU is actively working. In the case that the algorithm is parallel by pixels, as the scene is traced the rays from certain pixels will terminate sooner than others. This leaves compute units inactive resulting in poor utilization of the GPU. Rather, by producing an algorithm which is parallel by rays cast, terminated rays may be discarded after each iteration and the reflected rays repacked for a second iteration ensure all compute units marshaled are active.

An overview of the method presented in this paper is shown in Figure 1. To initialize the process a spacecraft CAD model is input as a triangulated mesh with facet materials which define the absorption, diffusion and specular optical characteristics. As a fresh solution is computed at each time step, the spacecraft model can contain time-varying shape components, or separate meshes for spacecraft proximity operations where one craft casts a shadow on another vehicle. Such features were not feasible with prior methods that pre-computed force results. The mesh is then processed to generate the bounding volume hierarchy (BVH) data structure to accelerate the processes of intersection testing. With initialization now complete the parallel ray-tracing algorithm can be executed on the GPU. The parallel ray-tracing algorithm then iterates through ray generation, BVH traversal, intersection testing and SRP computation until all rays have reached the set termination condition. In this work the ray termination condition is either a ray exiting the scene or completion of three reflections. The aggregated force and torque values are then returned to the CPU bound process where the values can be integrated into the dynamics propagation component of a spacecraft numerical simulation.

ALGORITHM COMPONENTS

The presented approach employs a number of key techniques and algorithms to minimize the otherwise high computational load of a naive ray-tracing algorithm. These techniques include:

- generation of an acceleration data structure in particular that of a bounding volume hierarchy (BVH)
- bounding box intersection testing algorithm using clipping planes, and
- the computational fast Möller-Trumbore ray and triangle facet intersection algorithm used to test for intersections with a spacecraft facet and ray.

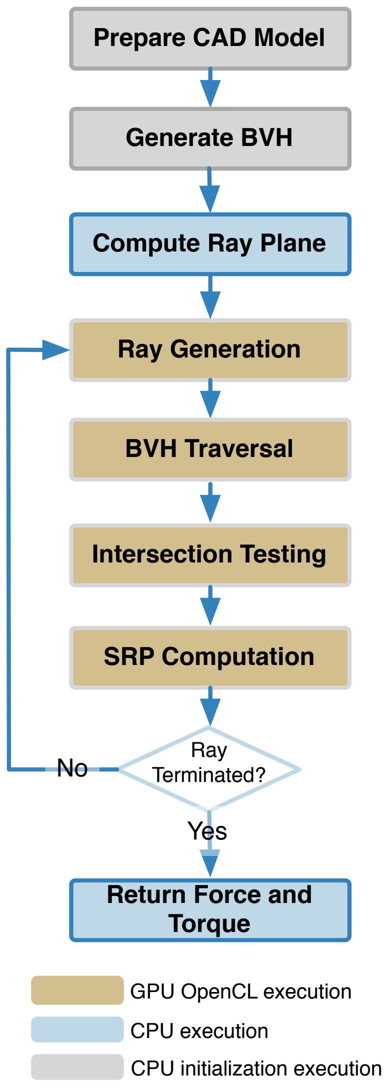


Figure 1. Ray-tracing flow of control. The gold colored stages are executed on the GPU. All otherwise colored stages are CPU bound.

All are presented in the subsequent subsections.

Acceleration Structures

Many acceleration structures are presented in ray and path tracing literature. Each of these structures offer advantages and disadvantages which are typically dependent on the model to be rendered.[14] This method employs a simple bounding volume hierarchy to efficiently reduce the ray intersection search space and therefore the required ray intersection computations performed. To build the bounding volume hierarchy a bounding volume is computed for each triangular facet in the spacecraft mesh model. In this implementation the bounding volume is computed as a bounding box aligned to the spacecraft model body frame. To begin, the list of bounding volumes is sorted along the first frame spacecraft body frame axis. The sorted list is then divided in half and a new bounding volume is computed around each half of the list. This process is carried out recursively

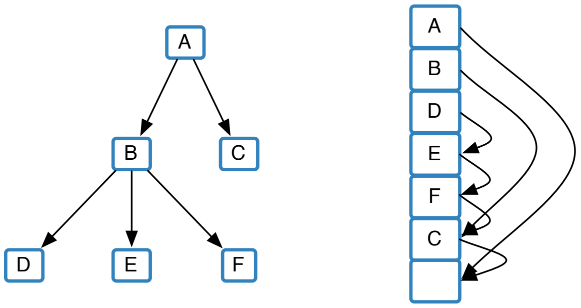


Figure 2. Two BVH traversal structures. The left structure demonstrates a simple recursive BVH traversal. The right demonstrates the same BVH as shown on the left yet organized as a depth first search array with precomputed node skip pointers.

while, at each new split, sequentially selecting the sort axis as the next axis in the body frame triad. This results in a bounding volume hierarchy that groups successive bounding volumes as containing facets spatially near to each other.

An efficient method of traversing the bounding volume hierarchy is a key aspect in the development of real-time SRP ray-tracing.[14] This implementation uses as the BVH traversal method a depth first search array as described by Smits.[14] An example BVH hierarchy comprising 6 nodes is shown in Figure 2, first as a recursive depth first search tree and second as a depth first search array with precomputed skip pointers. In the recursive tree structure, if bounding volume node A is intersected, the search recursively descends to test for an intersection against node B. If no intersection is found at node B the recursion meets a termination condition and the search moves back up the tree and proceeds down the next search branch to test node C. For the array traversal structure, if bounding volume A is intersected, the next node to try is the next node in the array which is 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 array traversal algorithm is shown as pseudo code in Algorithm 1.

The depth first array search structure 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 BVH traversal structure is that the structure results in greater memory coherency for large meshes and therefore more efficient contiguous memory accesses on the GPU given its sequential nature.[14]

Bounding Volume Intersection

Bounding volume intersection uses the algorithm originally presented by Kay and Kajiya.[15] Here the algorithm models the bounding box as 3 sets of parallel planes. The algorithm employs each set of parallel planes as clipping planes. As demonstrated in Figure 3, 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 any testing of conditional code statements referred to as code branching. Modern floating-point instruction sets are capable of computing minimum and maximum operations without branches and this parallel plane algorithm results in a ray to bounding box intersection test with no code branches or divisions operations.

```
Data: idx is the index of the current node in the BVH depth first sorted traversal array
while idx is in range do
    node = fetch next node at idx;
    if intersectBbox then
        if node is leaf then
            intersectTriangle;
        else
            idx = idx + 1 (move to first child node);
            continue;
        end
    end
    idx = skip pointer at node (follow skip pointer to next node)
end
```

Algorithm 1: Algorithm to traverse the depth first sorted BVH array using skip pointers.

Triangle Facet Intersection

The spacecraft model mesh is comprised of many thousands of triangular facets. To compute, a triangle-ray intersection the Möller-Trumbore algorithm is used. This algorithm is a fast and memory efficient triangle-ray intersection algorithm making it ideal for use in the often memory constrained GPU computation environment. The algorithm turns on the knowledge that the point of intersection of a line through a triangle in barycentric coordinates (u, v) must adhere to easily boolean testable coordinate bounds. The bounds are defined by the barycentric coordinate system which requires $u \geq 0$, $v \geq 0$ and $u + v \leq 1$.[16] As such, the triangle to be tested, as determined by traversing of the BVH, is mapped to barycentric coordinates and the ray intersection tested against the barycentric coordinate bounds.

SRP DEPENDENCE ON SURFACE MATERIAL

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 $\mathrm{d}L_r$ to the radiant incidence $\mathrm{d}E_i$.[17]

The geometry of the reflection interaction is shown in Figure 4 where the ray intersection point on a surface is $\boldsymbol{x}$, $\boldsymbol{w}_o$ is the direction of the outgoing ray, $\hat{\boldsymbol{n}}_{\boldsymbol{x}}$ is the unit normal to the surface at $\boldsymbol{x}$, and $\boldsymbol{w}_i$ is the direction of the incident radiation. The normalized vector, $\hat{\boldsymbol{h}}_{\boldsymbol{x}}$ is in the direction of the angular bisector of $\boldsymbol{w}_o$ and $\boldsymbol{w}_i$, and is defined by $\hat{\boldsymbol{h}}_{\boldsymbol{x}} = (\boldsymbol{w}_o + \boldsymbol{w}_i)/|\boldsymbol{w}_o + \boldsymbol{w}_i|$.

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, $\mathrm{d}L_o(w_o) \propto \mathrm{d}E(w_i)$. This allows for the development of the bidirectional reflectance distribution function, $f_r(w_i \to w_o)$, given in Eq. (2), where the proportionality relationship describes the observed radiance leaving a reflecting surface in the direction w_o and the projected solid angle defined as $\mathrm{d}\sigma^{\perp}(w) = |w \cdot \hat{\boldsymbol{n}}|\mathrm{d}\sigma(w)$.

$$f_r(w_i \to w_o) = \frac{\mathrm{d}L_o(w_o)}{\mathrm{d}E(w_i)} = \frac{\mathrm{d}L_o(w_o)}{L_i(w_i)\mathrm{d}\sigma^{\perp}(w_i)} \tag{2}$$

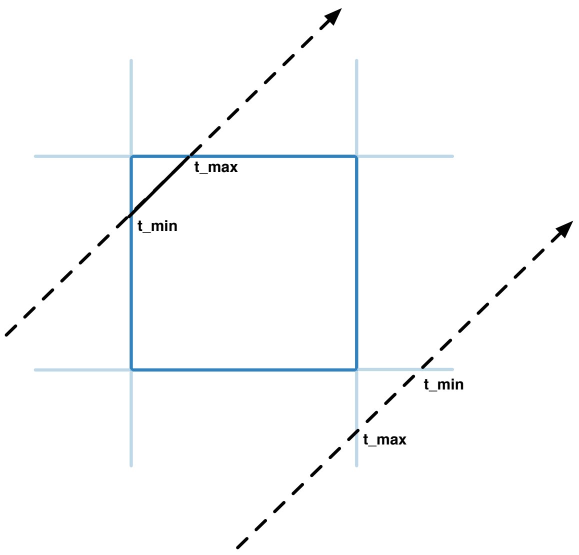


Figure 3. Example results of the parallel plane bounding box intersection algorithm. For the top left ray intersection the algorithm returns t_max as greater than or equal to t_min. For the bottom right ray miss the algorithm returns t_max as less than t_min.

The relationship between the outgoing radiance and the incoming radiance for a particular optical surface is described at Eq. (3).

$$\mathrm{d}L_o(w_o) = \mathrm{d}L(w_i) f_r(w_i \to w_o) \mathrm{d}\sigma^{\perp}(w_i) \tag{3}$$

By integration, Eq. (4) yields the total radiance, over the hemisphere, leaving a surface area element.[18]

$$L_o(w_o) = \int_{S^2} L(w_i) f_r(w_i \to w_o) \mathrm{d}\sigma^{\perp}(w_i) \tag{4}$$

This work employs physically plausible BRDFs. A physically plausible BRDF adheres to the symmetry expression given at Eq. (5) and energy conservation condition given by Eq. (6).

$$f_r(w_i \to w_o) = f_r(w_o \to w_i) \quad \text{for all} \quad w_i, w_o \tag{5}$$

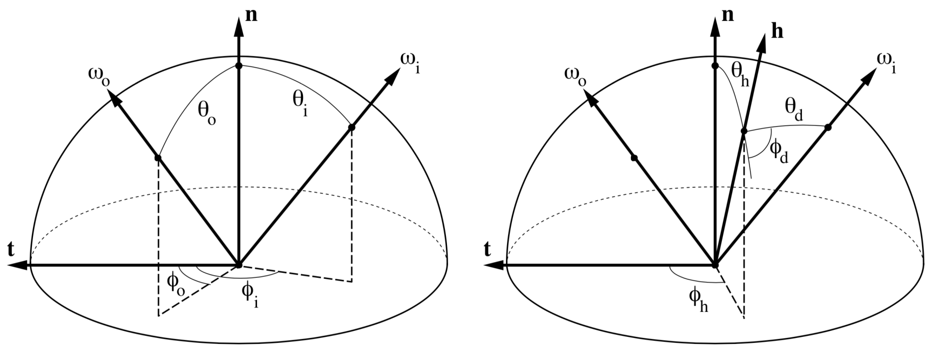


Figure 4. Two useful parameterizations of the BRDF geometry. On the left the BRDF is parameterized by the intuitive (θ_i, ϕ_i) and (θ_o, ϕ_o). On the right the BRDF is parameterized as function of the half angle (θ_h, ϕ_h) and a difference angle (θ_d, ϕ_d).

$$\int_{\mathcal{S}_o^2} f_r(w_i \to w_o) \mathrm{d}\sigma^\perp(w_o) \leq 1 \quad \text{for all} \quad w_i \in \mathcal{S}_i^2 \tag{6}$$

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

$$f_r(w_i \to w_o) = \rho R_d + s R_s \tag{7}$$

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

BRDF DESCRIPTIONS

For this work three different BRDFs are implemented and demonstrate the contrast between employing a simple BRDF and a more complex and descriptive BRDF.

Ideal BRDF

The first BRDF is the typical combination of diffuse lambertian and ideal specular mirror-like reflection. This BRDF expression is given in Eq. (8), where F_0 is the Fresnel reflection coefficient, ρ the diffuse scaling constant of the material and $\hat{\boldsymbol{r}}$ outgoing mirror reflected direction given by $\hat{\boldsymbol{r}} = 2(\hat{\boldsymbol{w}}_i \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}} - \hat{\boldsymbol{w}}_i$.

$$f_r(w_i \to w_o) = d\left(\frac{\rho}{\pi}\right) + s\left[\frac{F_0 \delta(\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{r}})}{\cos\theta_i}\right] \tag{8}$$

Mircofacet BRDF

The microfacet model is an approach to describe the specular term of a BRDF by modelling the material surface as collection of optically flat statistically distributed facets. The facets are assumed to be at a scale too small for shading considerations and significantly larger than the wavelength of the incident radiation.[19] Figure 5 presents a conceptual explanation of the microfacet geometries. Three points with the same macrosurface normal $\hat{\boldsymbol{n}}$ and microsurface normal $\hat{\boldsymbol{m}}$. The green reflection pairs are visible in both the $\hat{\boldsymbol{w}}_o$ and $\hat{\boldsymbol{w}}_i$ directions, while the red is blocked (in $\hat{\boldsymbol{w}}_i$ in this case).[20] Given these assumptions of the surface microgeometry, the distribution of the facets and the resulting specular BRDF term is given in Eq. (9).

$$R_s = \frac{DGF}{4(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{w}}_i)(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{w}}_o)} \tag{9}$$

In Eq. (9) the function D is the microgeometry normal distribution function (NDF) which describes the distribution of surface area, of facets, which are oriented with respect to the incoming radiation direction, such that the incoming radiation could reflect in a given direction on the hemisphere. The geometry function, G, governs the fraction of area of facets which is not shadowed by other facets. The function F is the Fresnel reflection factor of the active microfacet areas and controls how much of the radiation is reflected from the facets given the radiation angle of incidence.

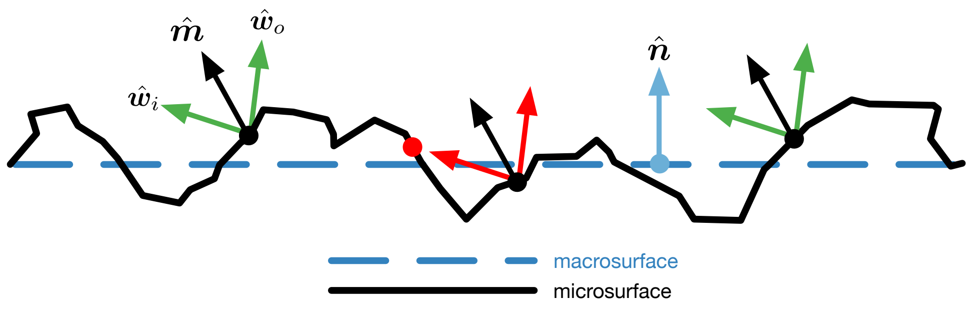


Figure 5. Conceptual illustration of microfacet with shadowing-masking geometry.

Finally, the denominator $4(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{w}}_i)(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{w}}_o)$ is a correction factor, which resolves quantities being mapped between the local microgeometry coordinate space and that of the macrosurface coordinate space.

Selection of the terms D and G determine the resulting specular distrbution of reflected radiation. The NDF function D controls the size, brightness, and overall shape of the BRDF's specular highlight.[19] Many different NDFs have been presented in the computer graphics literature. Often the NDF function is Gaussian-like and includes a parameter which describes the "roughness" or variance of microfacet normals.[21] The geometry function G is a probability that surface points with a given microgeometry normal $\hat{\boldsymbol{n}}$ will be visible from both the light direction $\hat{\boldsymbol{w}}_i$ and the view direction $\hat{\boldsymbol{w}}_o$.

The two microfacet model NDF function variants implemented in this work are the Beckmann BRDF and Trowbridge-Reitz (GGX) BRDF models. The size of the specular lobe in each distribution is controlled by the roughness parameter α and the distribution NDF for each model. Full details of their derivation can be found in their original publications by Beckmann and Spizzichino[22] and Walker et al.[20] respectively. As a demonstration of the difference between the two NDFs, Figure 6 shows the magnitude of the differential area of the distribution of lit microfacets for grazing angles $-\pi/2$ to $\pi/2$. Figure 6 has been produced for the roughness value $\alpha = 45$. The value $\alpha = 45$ is chosen to be instructive as it will be used in the following section's orbit propagation results. The α parameter controls the spread of the lobe of the specular reflection. The GGX distribution has greater magnitude at the extremities and falls off to zero more slowly, than the Beckmann, for directions far from the surface normal.

COMPUTING SOLAR RADIATION PRESSURE

At each time step during a numerical simulation a new wave of rays are generated. The current spacecraft to sun unit direction vector $\hat{\boldsymbol{s}}_B$ is computed and used as the first axis in an orthogonal Sun S frame. The direction cosine matrix $[SB]$ which defines the rotation from the body frame B to the sun frame is constructed and used to map the eight vertices, which define the extents of the spacecraft model bounding box, to the new temporary sun frame. As shown in Figure 7, the sun facing side of the rotated bounding box is used as a finite plane perpendicular to $\hat{\boldsymbol{s}}_B$ from which the origins of all ray vectors shall be defined.

The ray plane is divided into unit squares determined by the resolution units chosen by the user. For example, a 2 m x 1 m plane can be divided into 10mm sized squares giving a plane of 200 x 100 squares and a resolution of 20,000 rays. The ray intersection testing must occur in the same

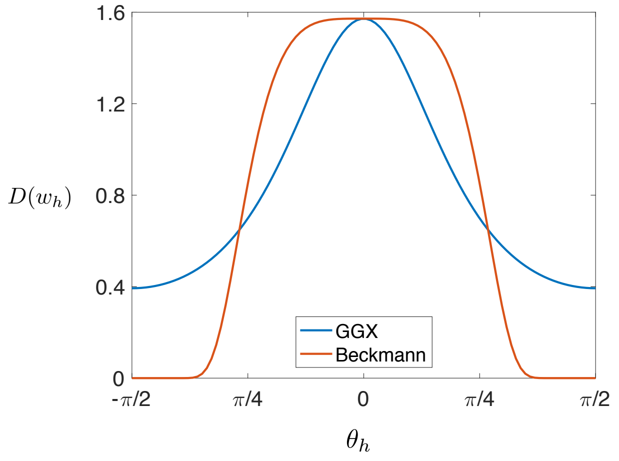


Figure 6. Isotropic Beckmann and GGX microfacet NDFs as a function of the angle between a ray direction and the the surface normal, θ_h, for roughness value $\alpha = 0.45$.

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.

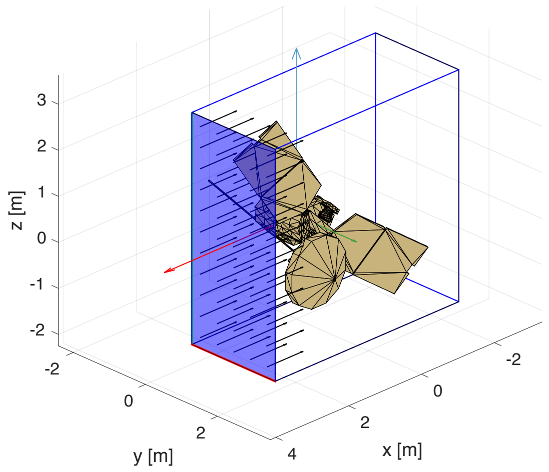
Each compute unit on the GPU launches an instance of the coded OpenCL kernel program. Each kernel instance accesses the ray wave front data in the GPU global memory space copying a specific ray and the BVH traversal array to local memory in the compute unit. If a BVH intersection is found and the intersection is a terminal node then the triangle intersection is computed and the index of the facet and the intersection location are recorded and placed in an intersection array. If no intersection is computed the miss is similarly recorded in the intersection array. To compute the direction of a reflected ray it is necessary to sample the BRDF. The choice to compute the outgoing ray is made by sampling according to the proportions controlling the diffuse (ρ) and specular (s) contributions to the BRDF as dictated in Eq. (7).

Computing solar radiation pressure requires solving the rendering equation given Eq. (4). The expression at Eq. (10a) computes the force due to SRP for a single ray and accounts for the BRDF of a particular material. The irradiance term $L(w_i)$ present in Eq. (4) has been factored out and for the first wave of rays is equal to FA_{ray}, the combination of F and A_{ray}, the solar flux scaled by the heliocentric distance $(\frac{AU}{r})^2$ and area of the cast ray respectively. With successive bounces the $L(w_i)$ intensity of each ray decreases according to the absorption and scattering of successive ray-surface interactions. The total SRP force is the sum of all rays given in Eq. (10b).

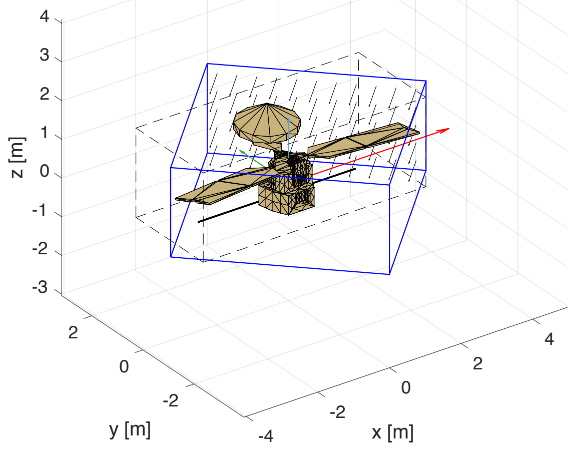
$$\boldsymbol{F}_k = \frac{L(w_i)_k}{c}\left[\boldsymbol{w}_i + \int_{S^2} f_r(w_i \to w_o)|\boldsymbol{w}_o \cdot \hat{\boldsymbol{n}}|d\sigma(w_o)\right] \tag{10a}$$

$$\boldsymbol{F} = \sum_{k=1}^{N_{rays}} \boldsymbol{F}_k \tag{10b}$$

The integral of the BRDF within the Eq. (10a) is often difficult to solve analytically. For numerical integration methods such as quadrature, the number of evaluations required to approximate the integral grows exponentially as the number of dimensions in the integral grow.[23] As such, the multidimensional integral of the BRDF are often computationally slow when solved by quadrature.[19] For this reason the Monte Carlo estimator method, shown in Eq. (11) is used to estimate this inte-



(a) S frame generated rays



(b) S frame rays mapped to the B frame

Figure 7. A Mars Reconnaissance Orbiter mesh model surrounded by a B frame oriented bounding box (dashed black) and an S frame bounding box (blue solid). The red, green and blue vectors are the B frame axes and the black vectors indicates rays originating from the blue ray-plane.

gral. As more samples X_i are taken from the probability distribution p, the variance in the estimate $\left\langle F^N \right\rangle$, of the function $f(X_i)$, decreases. This provides a method to compute the integral containing the BRDF with lower computational intensity.

$$\left\langle F^N \right\rangle = \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(X_i)}{p(X_i)}, \; X_i \sim p. \tag{11}$$

As a result, in the ray tracing instance, the integral is sampled over various directions w_o as given by Eq. (12). The probability distribution $p(w_o)$, is determined by the particular sampling method employed by the BRDF participating in a particular ray-surface interaction.

$$\left\langle F^N \right\rangle = \frac{1}{N} \sum_{i=0}^{N-1} \frac{f_r(w_i \to w_o)|\boldsymbol{w}_o \cdot \hat{\boldsymbol{n}}|}{p(w_o)}, \; X_i \sim p. \tag{12}$$

MODEL VALIDATION

Validation is performed by comparing results from an analytic cannonball model and a sphere shaped spacecraft model evaluated by the OpenCL ray-tracing method. The values for the LAGEOS II spacecraft, given in Table 1, are used in both evaluations. A spherical model spacecraft is generated to replicate the LAGEOS II spacecraft parameters. Figure 8 shows the ray-tracing rendered version of the sphere colored according to facet normal vectors. The perturbative acceleration due to SRP as given by the cannonball model and the OpenCL model are given in Table 2. The OpenCL model yields a resultant torque of 2.3×10^{-16} Nm. However, it is expected that a perfectly spherical object yields zero torque. The non-zero torque value returned by the OpenCL model is due to the faceted nature of the model not being perfectly spherical. It is observed that by increasing the number of vertices of the model to better approximate a sphere and increasing the resolution of the projected rays, the resulting torque value approaches machine precision. The agreement between the two simple evaluations provides initial confidence of the method's correctness.

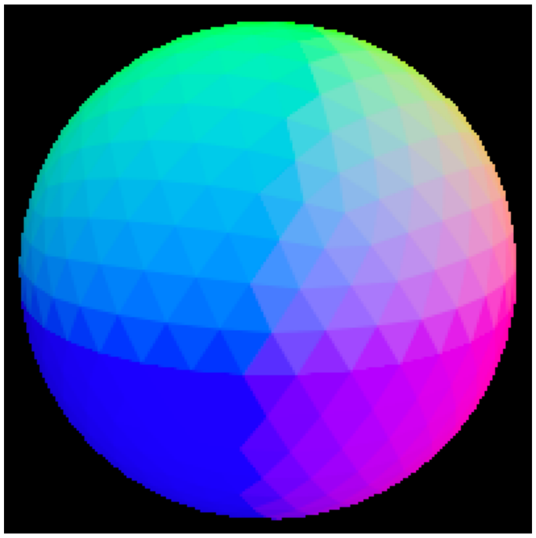


Figure 8. A ray-traced evaluation of a 1280 primitive, LAGEOS II size satellite. Facet colors are purely cosmetic to allow for visual validation of accurate model rendering.

Table 1. LAGEOS II spacecraft parameters used for computation of SRP by cannonball model and OpenCL ray-tracing model.

LAGEOS II Attribute	Value
mass	405.38 [kg]
area	0.2817 [m^2]
Φ (at 1 AU)	1.38×10^3 [W/m^2]
C_r	1.12

ORBIT PROPAGATION

The ray-tracing SRP model is integrated into the Basilisk astrodynamics simulation software as a Dynamics Effector module. The orbit of a cube of side length one meter and mass 1 kg is propagated to demonstrate the effect of employing BRDFs beyond the simple lambertian and mirror-like reflectance model. Two simulations scenarios are developed where the first simulation is a 1000 km altitude sun-synchronous LEO orbit and the second a geosynchronous equatorial orbit (GEO). The simulation orbit parameters are listed in Table 3.

For each of the LEO and GEO scenarios, three simulations are executed, where the cube object is assigned a different BRDF model. The three BRDFs are the idealized model given in Eq. (8), the Beckmann microfacet model and the GGX microfacet model. Each model is constructed using the same weights for the diffuse and specular contributions with $\rho = s = 0.5$. For the two microfacet BRDFs their roughness parameter (parameter controlling the spread of the specular lobe) is set as

Table 2. LAGEOS II spacecraft SRP induced acceleration computed by Cannonball and OpenCL models, where the OpenCL ray resolution is 10mm.

Model	SRP Acceleration $a_\odot$
Cannonball	3.58×10^{-9} [m/s^2]
Ray-Traced Cannonball	3.60×10^{-9} [m/s^2]

Table 3. Spacecraft orbit parameters for sun-synchronous LEO orbit and GEO orbit.

	LEO	GEO
A km	7378	42164
e	0	0
i, deg	98	0
M_0, deg	90	90
Ω, deg	0	0
ω, deg	0	0

$\alpha = 0.45$. The cube in each scenario is given a slow initial body rate to demonstrate the variation in force due to changing incident angle as previously demonstrated by Figure 6. The initial rate for the LEO scenario is $w_{B/N} = [0.06, 0, 0]$ deg/s and for the GEO scenario $w_{B/N} = [0.006, 0, 0]$ deg/s.

The magnitude of the acceleration due to SRP for each BRDF model in the LEO and GEO scenarios is shown in Figures 9 and 10, respectively. In both orbital scenarios the cube body rate can be seen in the variation of the SRP acceleration. The simulation in which the BRDF is set as the Beckmann microfacet consistently has the greatest magnitude. This agrees with the distribution shown in Figure 6 where the Beckmann distribution models a greater area of specular microfacets 'seeing' the sun than that of the GGX distribution. Similarly, for incident light angles of greater than approximately $\pi/4$ the magnitude of Beckmann distribution drops sharply and is almost equal to the GGX distribution magnitude. This is evidenced by the sharp decrease in acceleration in the Beckmann simulation as the cube rotates to briefly orient a cube edge towards the sun bringing the incident light angle on the sides of the cube near 45 deg.

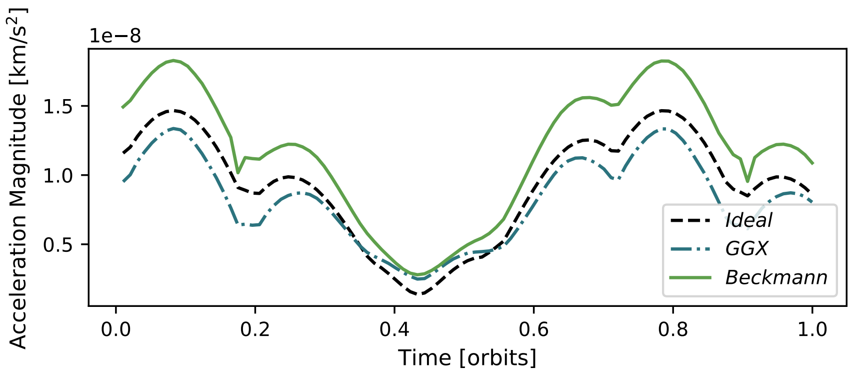


Figure 9. Magnitude of the SRP acceleration in sun-synchronous LEO over one orbital period.

The difference in position, with respect to the simulated orbit of the cube with the idealized BRDF, is plotted for the Beckmann and GGX models. The position differences are given in the radial R, intrack S and cross track W, $[RSW]$ relative position frame. The relative positions for the LEO simulation are displayed in Figure 11 and for the GEO simulation in Figure 12. The differences in acceleration magnitude of the Beckmann BRDF and GGX BRDF are easily seen in all plots due to the growth in differences in each axis. In the LEO scenario, after a single orbit, radial position differences of a meter are demonstrated. In the GEO scenario radial differences of over a kilometer are shown after less than two orbits. These differences in relative position demonstrate and reinforce the importance of modeling a spacecraft's various BRDFs with appropriate BRDF models.

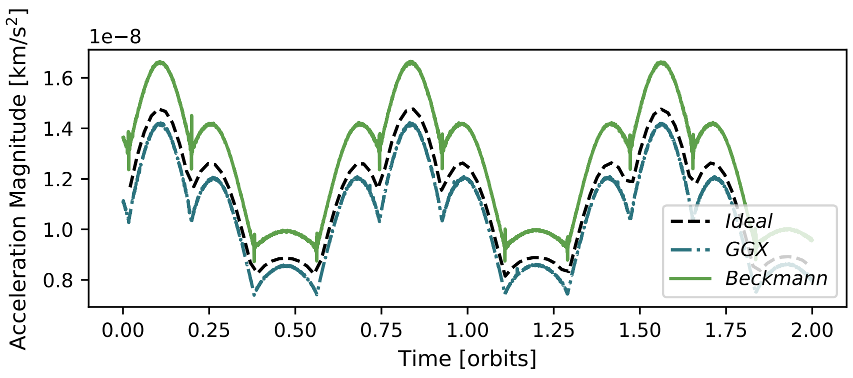
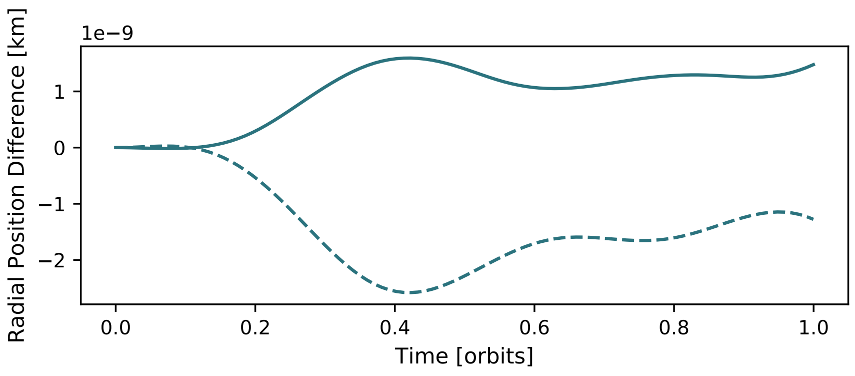


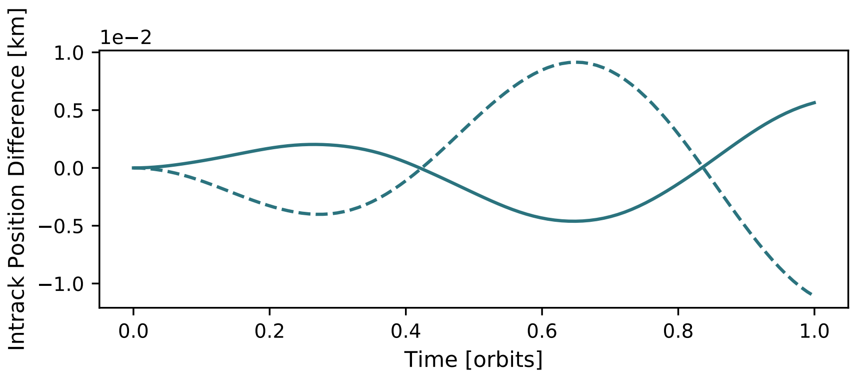
Figure 10. Magnitude of the SRP acceleration at GEO over two orbital periods.

COMPUTATIONAL PERFORMANCE

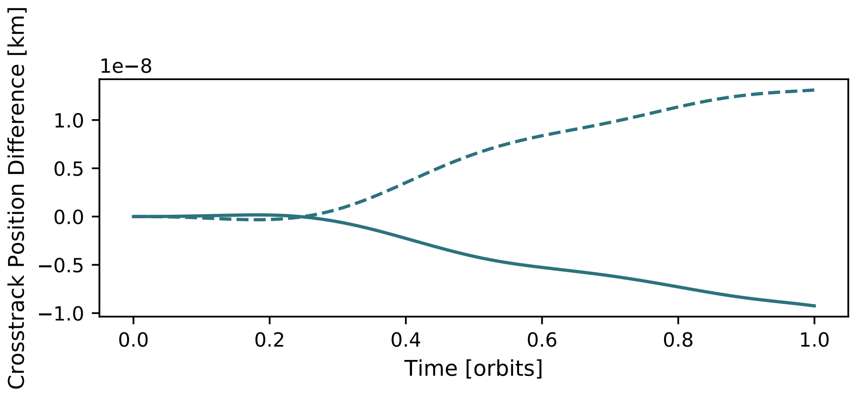
The presented method seeks to ensure both data parallelism and algorithm parallelism. Achieving these two aims will provide efficient use of the GPU execution environment. As an example of the method's performance an evaluation of the CloudSat spacecraft was executed for various ray resolutions. The debugging output of the rendered spacecraft in the sun-frame orientation is shown in Figure 13. The model coloring in the red, green and blue channels correspond to spacecraft force components in the x, y and z components of the spacecraft body-frame. The evaluations were conducted on modest GPU hardware of an embedded Intel HD Graphics 630 and AMD Radeon Pro 560. The Intel HD Graphics 630 shares an amount of dynamics random-access-memory (DRAM) with the CPU. As a result the Intel GPU can access data in the CPU memory space with a zero-copy operation and does not incur the data transfer latency experienced by separate GPU hardware.[24] However, the Intel GPU typically has a smaller memory size than the separate GPU and therefore can handles fewer rays and associated data. It is necessary to switch computation to the separate GPU, once the ray density surpasses the Intel GPU memory limit. The two colors in Figure 14 indicate the split from the Intel GPU to the AMD GPU.



(a) Radial position difference from Ideal BRDF.

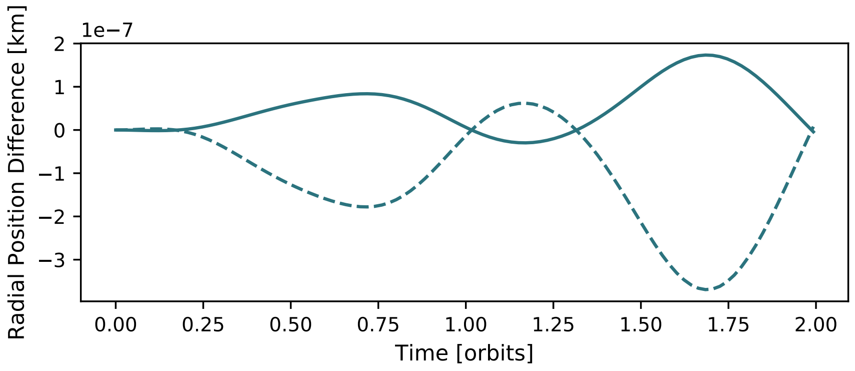


(b) Intrack position difference from Ideal BRDF.

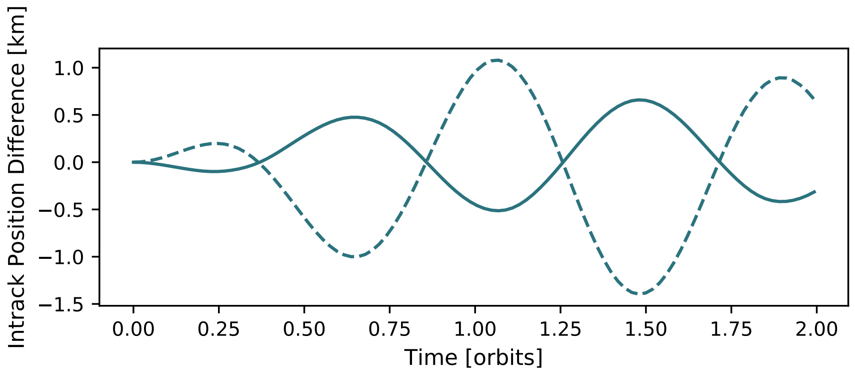


(c) Crosstrack position difference from Ideal BRDF.

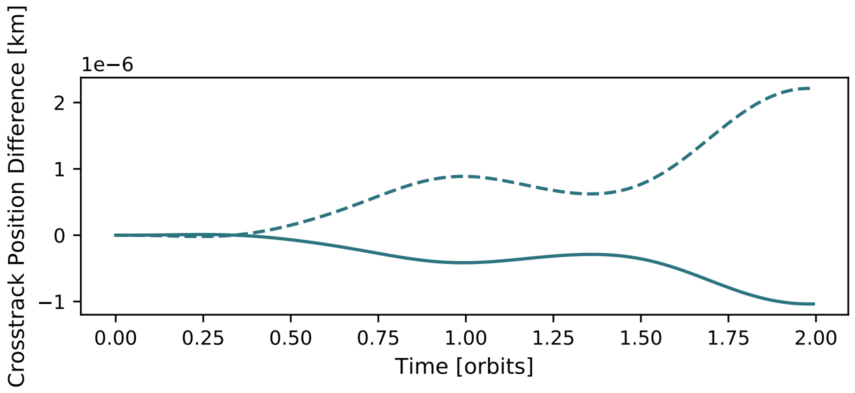
Figure 11. Radial, intrack and crosstrack differences w.r.t the position of the simulated Idealized BRDF cube at LEO. The dashed line corresponds to the Beckmann and the solid line the GGX.



(a) Radial position difference from Ideal BRDF.



(b) Intrack position difference from Ideal BRDF.



(c) Crosstrack position difference from Ideal BRDF.

Figure 12. Radial, intrack and crosstrack differences w.r.t the position of the simulated Idealized BRDF cube at GEO. The dashed line corresponds to the Beckmann and the solid line the GGX.

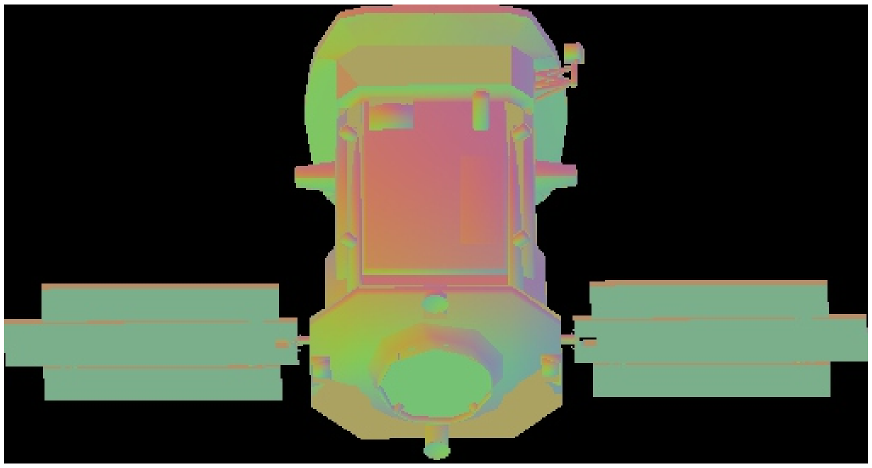


Figure 13. Resulting sun-frame rendered Cloudsat model.

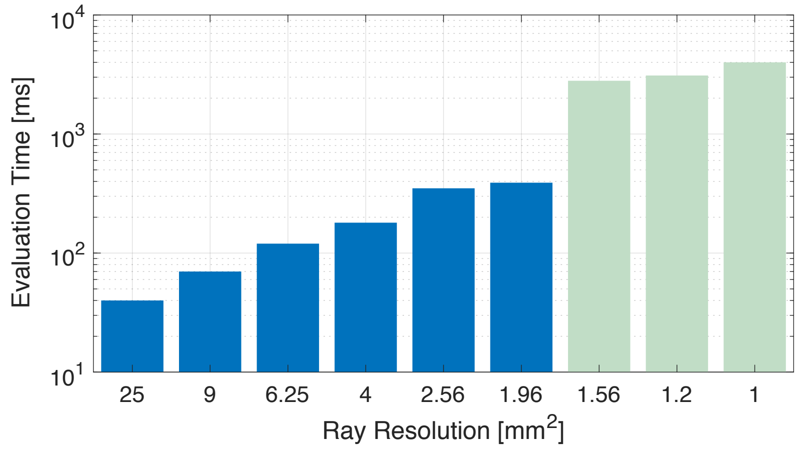


Figure 14. Cloudsat model evaluation time for rays cast. Evaluations in blue are done on Intel HD Graphics 630 and green on AMD Radeon Pro 560.

CONCLUSIONS

This paper demonstrates how SRP forces and torques can be resolved for complex spacecraft structures more accurately and at high speed using an OpenCL GPU focused ray-tracing methodology. Spacecraft self-shadowing, self-reflection and arbitrary spacecraft articulation are implicitly captured by a ray-tracing method resulting in a faster than real time modeling evaluation.

The method accommodates complex and varied BRDFs which are computed with the computationally efficient Monte Carlo integration. It is shown that the resultant SRP acceleration of the spacecraft is sensitive to the choice and parameterization of the BRDF. Compared to the idealized lambertian and mirror reflectance BRDF, it is shown that more physically plausible BRDFs such as the microfacet Beckmann and GGX result in differences in force magnitude and result in significant positional differences over just a few orbital periods. These differences in relative position demonstrate and reinforce the importance of modeling a spacecraft's various BRDFs with appropriate physically plausible BRDFs. This method presents mission analysts with a tool that requires minimal set up and has the potential to make use of the wealth of pre-launch spacecraft engineering data.

REFERENCES

[1] D. Vallado, *Fundamentals of astrodynamics and applications*. New York: Springer, 2007.

[2] D. J. O'Shaughnessy, J. V. McAdams, P. D. Bedini, A. B. Calloway, K. E. Williams, and B. R. Page, "Messenger's use of solar sailing for cost and risk reduction," *Acta Astronautica*, Vol. 93, January 2014, pp. 483–489, 10.1016/j.actaastro.2012.10.009.

[3] H. F. Fliegel and T. E. Gallini, "Solar force modeling of block IIR Global Positioning System satellites," *Journal of Spacecraft and Rockets*, Vol. 33, No. 6, 1996, pp. 863–866, 10.2514/3.26851.

[4] J. A. Marshall, S. B. Luthcke, P. G. Antreasian, and G. W. Rosborough, "Modeling Radiation Forces Acting on TOPEX/Poseidon for Precision Orbit Determination," tech. rep., 1992.

[5] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing," *Proceedings of the IEEE*, Vol. 96, No. 5, 2008, 10.1109/JPROC.2008.917757.

[6] I. Wald and P. Slusallek, "State of the art in interactive ray tracing," *State of the Art Reports, EUROGRAPHICS*, 2001, pp. 21–42, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.6266.

[7] M. Ziebart, S. Adhya, a. Sibthorpe, S. Edwards, and P. Cross, "Combined radiation pressure and thermal modelling of complex satellites: Algorithms and on-orbit tests," *Advances in Space Research*, Vol. 36, No. 3, 2005, pp. 424–430, 10.1016/j.asr.2005.01.014.

[8] D. J. O'Shaughnessy, J. V. McAdams, K. E. Williams, and B. R. Page, "FIRE SAIL: MESSENGER'S USE OF SOLAR RADIATION PRESSURE FOR ACCURATE MERCURY FLYBYS," 2011, pp. 1–16.

[9] D. M. Lucchesi, "Reassessment of the error modelling of non-gravitational perturbations on LAGEOS II and their impact in the Lense–Thirring derivation—Part II," *Planetary and Space Science*, Vol. 50, No. 10-11, 2002, pp. 1067–1100, 10.1016/S0032-0633(02)00052-1.

[10] J. W. McMahon and D. J. Scheeres, "New Solar Radiation Pressure Force Model for Navigation," *Journal of Guidance, Control, and Dynamics*, 2010, 10.2514/1.48434.

[11] M. Ziebart, *High Precision Analytical Solar Radiation Pressure Modelling for GNSS Spacecraft*. PhD thesis, University of East London, 2001.

[12] S. Tanygin and G. M. Beatty, "GPU-Accelerated Computation of SRP Forces with Graphical Encoding of Surface Normals," *AAS/AIAA Astrodynamics Specialist Conference*, Vail, CO, Aug. 9-13 2015.

[13] P. W. Kenneally and H. Schaub, "High Geometric Fidelity Modeling of Solar Radiation Pressure Using Graphics Processing Unit," *AAS/AIAA Spaceflight Mechanics Meeting*, Napa Valley, California, Feb. 14–18 2016. Paper No. AAS-16-500.

[14] B. Smits, "Efficiency Issues for Ray Tracing," *Journal of Graphics Tools*, Vol. 3, No. 2, 1999, pp. 1–14, 10.1080/10867651.1998.10487488.

[15] T. L. Kay and J. T. Kajiya, "Ray Tracing Complex Scenes," *Computer Graphics (SIGGRAPH '86 Proceedings)*, Vol. 20, No. 4, 1986, pp. 169–278.

[16] T. Moller and B. Trumbore, “Fast , Minimum Storage Ray / Triangle Intersection,” *Journal of Graphics Tools*, Vol. 2, No. 1, 1997, pp. 21–28, 10.1145/1198555.1198746.

[17] C. J. Wetterer, R. Linares, J. L. Crassidis, T. M. Kelecy, M. K. Ziebart, M. K. Jah, and P. J. Cefola, “Refining Space Object Radiation Pressure Modeling with Bidirectional Reflectance Distribution Functions,” *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 1, 2014, pp. 185–196.

[18] E. Veach, “Robust Monte Carlo Methods for Light Transport Simulation,” *Dissertation at the Department of Computer Science of Stanford University*, Vol. 134, No. December, 1997, pp. 759–764, 10.1016/S1350-9462(98)00033-0.

[19] M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[20] B. Walter, S. Marschner, H. Li, and K. Torrance, “Microfacet models for refraction through rough surfaces,” *Eurographics*, 2007, pp. 195–206, 10.2312/EGWR/EGSR07/195-206.

[21] R. L. Cook and K. E. Torrance, “A reflectance model for computer graphics,” *ACM SIGGRAPH Computer Graphics*, Vol. 15, No. 3, 1981, pp. 307–316, 10.1145/965161.806819.

[22] P. Beckmann and A. Spizzichino, *The scattering of electromagnetic waves from rough surfaces*. 1987.

[23] W. C. Harvey Gould, Jan Tobochnik, *An introduction to computer simulation methods : applications to physical systems*. Pearson Addison Wesley, 3rd ed., 2007.

[24] A. Lake, “Getting the Most from OpenCLTM 1.2: How to Increase Performance by Minimizing Buffer Copies on Intel® Processor Graphics,” tech. rep., Intel Corporation, 2014.