Development of a versatile LiDAR point cloud simulation testbed for advanced RSO algorithms

Lane Fuller, Robert Karl, Jr., Bruce Anderson, Max Lee-Roller
Advanced Scientific Concepts

ABSTRACT

Due to the increasing amount of Resident Space Objects (RSOs) in the Earth's orbit, the ability to quickly and accurately extract information about them has become an important element of Space Situational Awareness (SSA). The Global Shutter Flash LiDAR (GSFL) has been proven to be an effective and highly reliable sensor to acquire multi-dimensional data about RSOs over a wide range of in-orbit distances from less than a meter to 5km, with the potential to extend this range to 50km. Various information products are available from advanced GSFL embedded processing depending on the mission needs and concept-of-operations (ConOps). This information is available on-platform in real or near real-time, or for transmission off-platform for further processing and analysis. In order to support development of the GSFL and associated system interaction, a comprehensive simulation testbed has been developed. With this tool, scientists and engineers can work through concept development and feasibility studies, mission ConOps, and system design decisions at the highest levels. The simulation testbed fits into the Model Based System Engineering (MBSE) approach in a number of ways, including functional/behavioral and performance modeling. Additionally, the simulation testbed is an important part of Machine Learning (ML) and Artificial Intelligence (AI) workflows to generate the large organized point cloud (OPC) datasets needed for Deep Learning Convolutional Neural Networks (DL-CNN), for example. As a tool for software development, the simulation testbed is used for unit and regression testing during development and sustainment, with realistic real-time dynamic scenarios and accurate GSFL organized point cloud data. The simulation testbed has been constructed with layers of abstraction and developed in an object-oriented manner for continuing capability extension and refinements as its use expands.

1. INTRODUCTION

Advanced Scientific Concepts (ASC) invented 3D Global Shutter Flash LiDAR (GSFL) imaging for space and terrestrial applications. ASC's novel array technology has allowed for the development of compact LiDAR cameras that collect a full frame 3D organized point cloud (OPC) using a single 10ns laser pulse, with no moving parts. Because a full frame of organized 3D point cloud data is captured with each laser flash, the data is immune to motion blurring and related distortion. The OPC data is in the form of an array of 3D information, consisting of x, y, z, range, and intensity (X, Y, Z, R, I), where each vector maps one-to-one to a corresponding time-of-flight pixel in the focal plane array (FPA). Since this format is inherent to the GSFL's design, it does not require computationally expensive point cloud rectification to achieve the OPC format, which is optimal for use in Deep Learning Convolutional Neural Networks (DL-CNN). ASC's GSFL camera has been demonstrated for rendezvous and berthing aboard the Dragon capsule, on-orbit testing aboard the Endeavor Space Shuttle, in deep space operations aboard the OSIRIS-Rex mission, and docking with the International Space Station (ISS) aboard the Boeing CST-100 Starliner, achieving a Technology Readiness Level 9 (TRL-9). ASC is currently using the GSFL as part of an autonomous solution for search, detection, classification, semantic segmentation, feature point extraction, and pose determination for relative navigation in maritime applications. The GSFL could be extended for RSO work to use embedded processing for AI inference using state-of-the-art Deep Learning Convolutional Neural Networks with on-orbit re-programmability for architectural change or new model libraries.

In order to support the development of future applications of this technology, ASC has developed software to simulate the GSFL camera. This GSFL point cloud simulation testbed was designed with a layered approach, with levels of abstraction ranging from orbital and line-of-sight models with frames of reference to detector physics at the lowest level. These layers are outlined as follows:

- Resident Space Object (RSO) trajectory, orbital, line-of-sight, and frames of reference modeling and visualization
- RSO Computer Aided Design (CAD)/mesh model libraries (i.e., NASA's 3D Models Library [1])
- RSO materials (bidirectional reflectance distribution function (BRDF)

- RSO model characteristics (according to a taxonomy)
- Optical Imaging System (field of view (FoV), flash laser energy, diffuser patterns, and ray tracing)
- Detector physics (focal plane array (FPA) electrical models)

This paper presents details of the GSFL point cloud simulation testbed design and the results of the work using it to simulate training datasets for DL-CNN development. The simulation testbed involves a diverse set of modeling tasks including target trajectory, CAD model transformation to OPCs within a FoV, GSFL optical system modeling, BRDF modeling using target physical properties, and GSFL detector physics. ASC makes use of the simulation testbed for Artificial Intelligence (AI) work related to RSOs, including creating large and diverse datasets for DL-CNN supervised training, test, and validation. With its extensible architecture and programming approach, the point cloud simulation testbed achieves its current modeling goals and is suitable for long-term growth to support RSO research.

2. MOTIVATION

On-orbit platforms for detection and classification of RSOs could make use of machine learning and artificial intelligence algorithms in order to minimize the on-board processing while still maintaining high performance. In order to develop these ML and AI algorithms, large training data sets would be needed. Collecting large training datasets from real-world measurements can be cost prohibitive. An alternative approach is to create high fidelity simulations of these real-world measurements to generate an extensive training dataset. In this work, ASC has developed a software testbed to simulate organized point clouds from GSFL cameras. ASC then use these simulated point clouds to develop advanced Machine Learning and Artificial Intelligence processing to achieve real-time classification of RSOs.

The GSFL simulator is a powerful tool that can be used throughout the development cycle. First, the simulator enables rapid concept-of-operation (ConOps) and feasibility studies. The accurate radiometric modeling enables decisions on the essential GSFL system parameters (laser energy and FoV) to be made quickly. The simulator also serves as a powerful visualization tool to verify assumptions made on the required spatial resolution or ground sampling distance (GSD). The simulator enables rapid and confident design decisions on these parameters and provides a useful visual spot check of the scenario. Later in the development cycle, the simulator can be used to close the loop with the application software and system interfaces. This allows for testing of control algorithms such as tracking/pointing algorithms. As the software development matures further, the simulator can be used for regression testing to ensure the core functionality remains the same as unrelated parts of the software are changed. From the ConOps development to ML and AI development and training to regression testing, the GSFL simulator could become an invaluable development tool for space situational awareness applications.

3. SIMULATOR DESCRIPTION

ASC has developed a flexible, high fidelity GSFL simulator which produces a time series of OPCs and corresponding range and intensity maps of a scenario along with ground truth data. Development began with a Navy Small Business Innovation Research (SBIR) program to aid in the design of a GSFL sensor-based system which detects, tracks, and determines relative pose and position of ocean vessel targets. The functionality has been extended to lunar rover hazard identification, unmanned aerial vehicle (UAV) obstacle avoidance, and rocket landing on unimproved sites. Example point clouds and digital elevation maps (DEM) are shown in Fig. 1. The simulator has also been extended to support RSO classification and tracking scenarios.

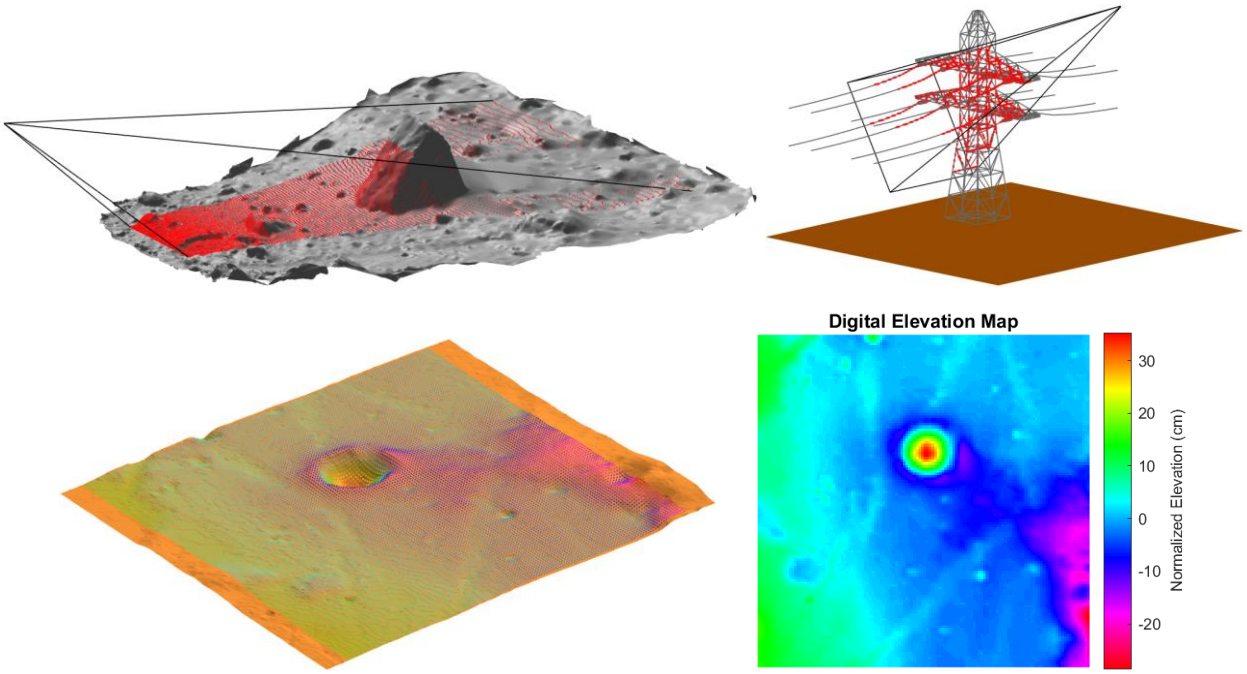

Fig. 1. (Top Left) Simulated point cloud of a lunar rover scenario. (Top Right) Simulated point cloud of a UAV obstacle avoidance scenario. (Bottom Left) Simulated point cloud of unimproved site for rocket landing. (Bottom Right) Corresponding DEM of rocket landing scenario.

The simulation workflow begins with scenario creation which includes the definition of 3D dynamics, target mesh models and BRDF, and the optical/system parameters of the GSFL. Scenarios are defined with coordinate frames and typically consist of one follower/chaser frame, one or more target frames, and an inertial frame of reference. 3D mesh models are attached to each target frame and are used to render the scene. Customizable BRDFs are applied to each mesh on either a global or per facet basis which results in radiometrically accurate renderings. For target/follow scenarios, target meshes are terrestrial vehicles such as cars or boats, or satellites for orbital space applications. For landing/obstacle avoidance scenarios, the target meshes are simulated terrain and obstacles. Easily customizable 3D dynamic models determine the movement of each coordinate frame in the scenario. This results in precise relative motion between the coordinate frames and accurate R&I video sequences.

The final step to defining a scenario is setting the GSFL system parameters. Imaging with the GSFL is similar to a standard electro-optical (EO) camera with a flash. Instead of a broadband flashbulb, the GSFL uses a pulsed 1064nm or 1570nm laser and a diffuser to illuminate the scene. The GSFL is available with several different laser power configurations to accommodate both short and long-range applications. The laser pulse is typically 6-10ns in duration which allows for accurate time of flight (ToF) measurements. Due to the global shutter, every pixel measures time of flight simultaneously, essentially freezing the scene in time and eliminating motion blur. The GSFL collects the return light with a lens and images the scene onto a focal plane array (FPA) consisting of 128x128 indium gallium arsenide (InGaAs) avalanche photodetectors (APDs) bonded to a proprietary readout. Typically, the laser divergence matches the lens FoV to illuminate the scene, but a smaller laser divergence can be used for long range applications where fewer illuminated pixels are needed. The simulator supports both fixed focal length lenses and zoom lenses along with variable diffusers.

With the simulation scenario definition complete, time-sequenced scenes are rendered using a custom ray-tracing function and a camera model. In order to simulate the range, the intersection between the ray and the closest mesh facet is recorded and the distance is calculated. In order to simulated the intensity output, the number of photons incident on each pixel is determined through radiometric calculations. The amount of return light is primarily dependent on the range to target, laser energy, BRDF, and lens specifications. Each scene is oversampled to ensure sub-pixel targets and partially illuminated pixels are rendered correctly. Finally, the number of photons on each pixel are input into the GSFL sensor model to generate accurate intensity values measured in digital counts. Noise models for both the range and intensity values have been developed and empirically verified, and are used to generate the

final result. Fig. 2 shows the signal processing chain described above. The following sections dive deeper into each of the major subcategories: scenario creation, optics and rendering, and detector response.

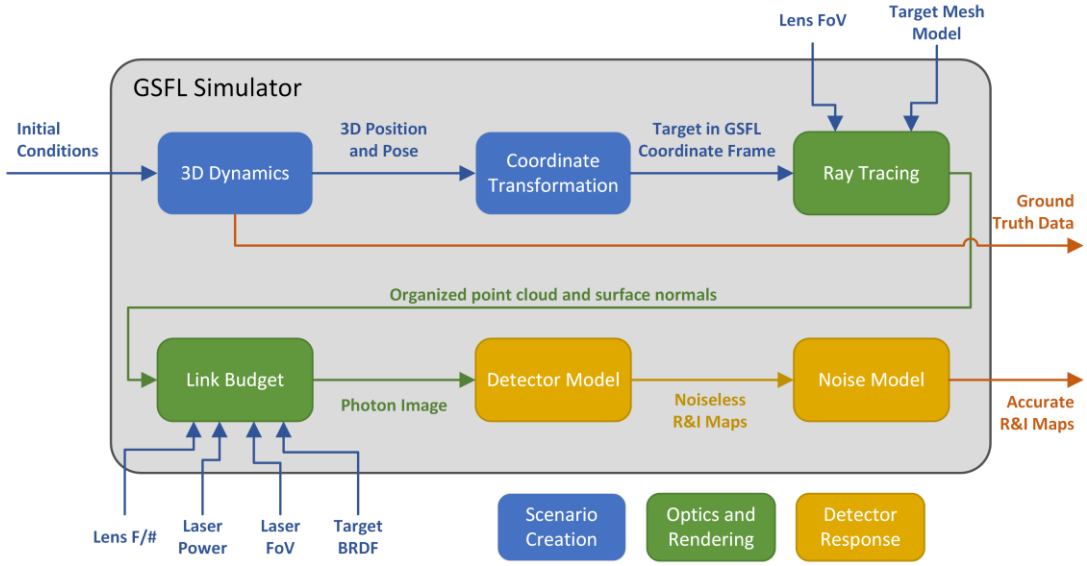

Fig. 2. GSFL Simulator signal processing flow

3.1 SCENARIO CREATION

3.1.1 COORDINATE FRAMES AND RIGID TRANSFORMATIONS

Scenario creation begins with defining all of the necessary coordinate frames to properly define the scene. For a typical target/follow scenario, the coordinate frames would be the target frame, follower frame, pan and tilt unit (PTU) frame, and the LiDAR frame. The core organizational structure of a scenario is a tree containing all the coordinate frames with the inertial coordinate frame at the root. Each frame inserted into the scenario has one parent frame and can have any number of children frames. The child frames are typically (but are not limited to) body-fixed frames, which means the coordinate frame is rigidly attached to the object. The link between parent/child frames is defined as the homogeneous or rigid transformation between the frames. This rigid transformation encodes the relative position and orientation between the frames. Therefore, points and vectors in each frame are defined as homogeneous coordinates. The homogeneous representation of a point, $\boldsymbol{q}$, and a normal vector, $\boldsymbol{n}$, in a child coordinate frame are shown in Equation **(1)**.

$$\overline{\boldsymbol{q}}_C = \begin{bmatrix} \boldsymbol{q}_C \\ 1 \end{bmatrix} = \begin{bmatrix} q_{C,x} \\ q_{C,y} \\ q_{C,z} \\ 1 \end{bmatrix}, \overline{\boldsymbol{n}}_C = \begin{bmatrix} \boldsymbol{n}_C \\ 0 \end{bmatrix} = \begin{bmatrix} n_{C,x} \\ n_{C,y} \\ n_{C,z} \\ 0 \end{bmatrix} \quad \textbf{(1)}$$

The rigid transformation of a point, $\boldsymbol{q}$, in a child frame to its parent frame is made up of a translation, $\boldsymbol{r}_{PC}$, and a rotation, R_{PC}, which is shown in Fig. 3 and Equation **(2)**. Normal vectors are only rotated by rigid transformations (this is not the case for affine transformations with scaling) due to the zero appended to the end of the vector.

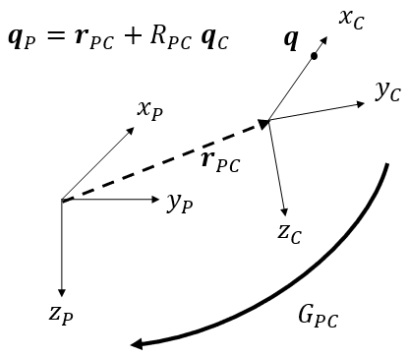

Fig. 3. Rigid transformation between two coordinate frames

$$\overline{\boldsymbol{q}}_P = \begin{bmatrix}\boldsymbol{q}_P \\ 1\end{bmatrix} = \begin{bmatrix}R_{PC} & \boldsymbol{r}_{PC} \\ \boldsymbol{0} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{q}_C \\ 1\end{bmatrix} = G_{PC}\overline{\boldsymbol{q}}_C$$
$$\overline{\boldsymbol{n}}_P = G_{PC}\overline{\boldsymbol{n}}_C \tag{2}$$

The inverse transform brings points expressed in the parent frame to the child frame and is defined in Equation **(3)**. For computational efficiency, it is worth noting that the inverse of a rotation matrix is equal to the transpose. Rigid transformations between multiple frames can be combined with matrix multiplication which is shown for three coordinate frames A, B, and C in Equation **(4)**.

$$G_{CP} = G_{PC}^{-1}$$
$$G_{CP} = \begin{bmatrix}R_{PC}^{-1} & -R_{PC}^{-1}\boldsymbol{r}_{PC} \\ \boldsymbol{0} & 1\end{bmatrix} \tag{3}$$
$$\text{where } R_{PC}^{-1} = R_{PC}^{T}$$

$$G_{AC} = G_{AB}G_{BC} \tag{4}$$

An example coordinate frame tree is shown in Fig. 4; this scenario involves a target frame, T, a follower frame, F, a PTU frame, P, and the LiDAR frame, L. The rigid transformation matrix from the target coordinate frame to the LiDAR coordinate frame is calculated on demand at each time step that the R&I data is requested. This transformation matrix is calculated by simply following the coordinate frame tree from the target frame to the LiDAR frame, which is shown in Equation **(5)**. This prepares the target mesh for the ray tracing and radiometric calculations described in Section 3.2.

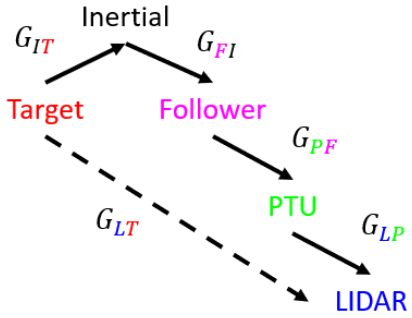

Fig. 4. Typical coordinate frame tree for a target/follow scenario

$$\overline{\boldsymbol{q}}_L = G_{LT}\overline{\boldsymbol{q}}_T \text{ where}$$
$$G_{LT} = G_{LP}G_{PF}G_{FI}G_{IT} \tag{5}$$

3.1.2 DYNAMICS AND CONTROL

The coordinate frames and rigid transformations defined in the previous section are useful for calculating the relative position/orientation between frames at any instant in time, but the dynamics of the scenario determine how each coordinate frame moves over time. Each scenario has highly application-specific dynamics so the simulator is designed to be flexible in order to support a large variety of applications. The dynamics in the simulator utilize state-space models and support non-linear, time-varying systems of the form shown in Equation **(6)**. In this model, t represents the time since the beginning of the simulation, $\boldsymbol{x}$ represents the full state of the system, and f defines the dynamics of the system and is dependent on t and the state of the system at time t, $\boldsymbol{x}(t)$.

$$\dot{\boldsymbol{x}}(t) = f(t, \boldsymbol{x}(t)) \tag{6}$$

The dynamics of each coordinate frame in the scenario have the form shown in Equation **(6)** and are defined relative to their parent frame or the inertial frame, whichever is more convenient. When defined this way, the state variable, $\boldsymbol{x}$, encodes the necessary position and pose information to define the rigid transformation matrices discussed in Section 3.1.1 at any point in time. The simulator is also flexible in the way it stores the pose information and can utilize either Euler angles or quaternions.

Since the simulator is defined in MATLAB, there is extensive support and tools to aid in modeling system dynamics. For example, the simulator can utilize the MATLAB Aerospace Toolbox for RSO target/follower scenarios. Fig. 5 shows a visualization of three satellite orbits created with the Satellite Scenario tool which enables rapid orbital dynamics modeling. This tool can be utilized to quickly create two closely matching orbits in order to accurately simulate the relative motion between target and follower RSOs.

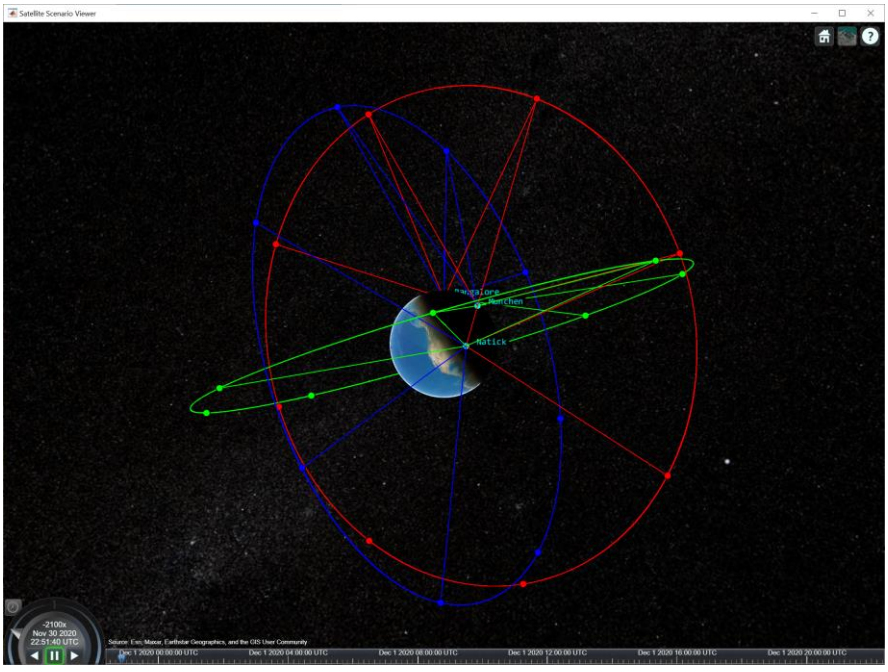

Fig. 5. MATLAB Aerospace Toolbox satellite scenario visualization

The simulator can also be used in a closed loop configuration shown in Fig. 6. In this model, $\boldsymbol{x}$ still represents the full state (and ground truth) of the system, but the outputs, $\boldsymbol{y}$, represent the measurements the system makes. These measurements include the GSFL R&I data but can also, for example, include global positioning system (GPS) or gyroscope measurements, with their associated noise models. The control scheme, K, is implemented in the application software and determines the necessary control inputs, $\boldsymbol{u}$, to feed back into the system to maintain constant tracking.

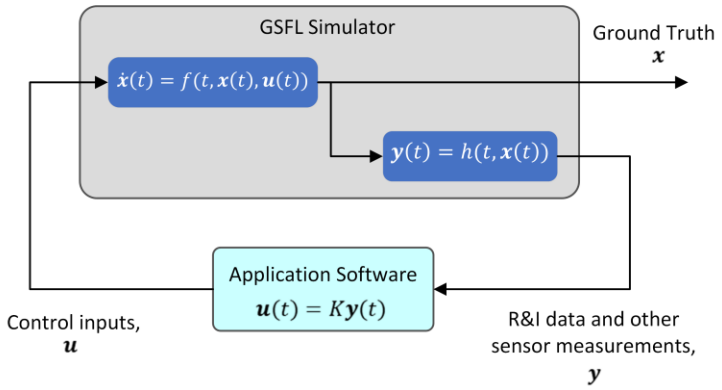

Fig. 6. Closed loop simulator configuration for RSO tracking

The GSFL simulator is equipped with generic, first and second order models for common actuators like gimbals and PTUs. For a high fidelity RSO tracking scenario, if the dynamics for spacecraft attitude control or gimbal are well understood, these models can be imported into the simulator to enhance the realism of the simulation. When the simulator is utilized in the closed loop or software-in-the-loop (SIL) configuration, the user can rapidly test algorithm development and control schemes. With the high fidelity GSFL simulator and accurate actuator models, the user has a high degree of confidence that their tracking/control algorithms will work the same way in hardware as they do in simulation.

3.1.3 MESH MODELS

Like many other computer graphics applications, the GSFL simulator relies on 3D models to render a scene. Highly accurate 3D parametric models can be created with a variety of computer aided design (CAD) software such as Solidworks and Blender. Parametric CAD software also support exporting the parametric models as mesh models so they can be utilized by the simulator. An example mesh model of the Juno space probe (taken from the NASA CAD library) is shown in Fig. 7.

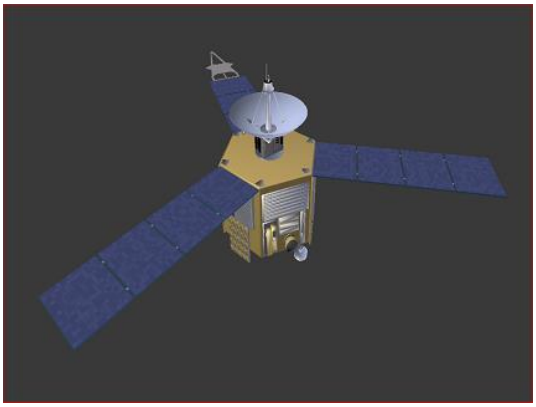

Fig. 7. Mesh model of Juno spacecraft (NASA CAD library)

Mesh models come in a variety of file formats (STL, OBJ, IGES, STEP, etc.), but all of them represent a 3D object in a similar manner: they all describe the surface geometry with a series of polygons. In addition, mesh models store the individual polygon surface normal vectors which is important for BRDF and rendering. Currently, the simulator only supports the STL format, which represents the 3D object as a series of triangles, but conversion between formats is possible. Mesh models also store texture, color, and BRDF functions for each polygon. This is shown in the rendering in Fig. 7 where the three solar panel arrays reflect a mottled blue color and the main body reflects gold light. Generally, mesh models contain BRDFs for the visible spectrum for EO camera renderings. For accurate GSFL renderings, BRDFs must be defined for 1064nm or 1570nm light (see Section 3.2 for more discussion on BRDF and rendering).

3.1.4 GSFL SYSTEM PARAMETERS

The final step to defining a scenario is setting the GSFL system parameters. The core technology and operating principle of the GSFL is as follows: the scene is illuminated with a pulsed laser (typically 6-10ns in duration) then a lens images the scene onto the FPA that measures the ToF and the intensity of the returned pulse. The FPA remains the same for each camera, but the transmit and receive optics are configurable. The FPA has a fixed 100um pixel pitch so the focal length of the lens determines the field of view (FoV) of the GSFL. The other relevant lens parameter is the aperture diameter (or equivalently, the f-number). In general, a larger aperture will result in more signal on the sensor and a longer focal length will result in a narrower field of view. In the simulator, the FoV of the lens, the orientation of the GSFL, and the range to the target define what portion of the scene is rendered and the spatial resolution of the point cloud. The range to target, lens aperture, and laser power determine how much light is collected by the lens.

The GSFL is available with several different laser power configurations to accommodate both short and long-range applications. The GSFL is also available with two different laser wavelength configurations: 1064nm and 1570nm. The 1064nm option has a simpler construction and is more efficient, but poses an eye safety hazard. The 1570nm option is much more eye safe and is utilized for scenarios where this is a priority. In the simulator, the wavelength primarily impacts the BRDF of the target objects as these functions are dependent on wavelength. Typically, the laser divergence is designed to illuminate the entire lens FoV. However, a smaller laser divergence can be used in order to increase the amount of light per unit area on the target. This is particularly useful for long range applications where maximum range performance is generally more important than a large field of view. For computational efficiency the simulator only renders the portion of the scene that is illuminated by the laser. The simulator software supports both fixed lens and laser settings in addition to variable settings, as is the case with a variable zoom lens or a set of actuated diffusers.

In the GSFL simulator, all of the system parameters described above are easily configurable. This enables rapid ConOps development, feasibility studies, and quick, well-informed design decisions.

3.2 OPTICS AND RENDERING

The GSFL simulator uses a ray tracing or ray casting algorithm to create a 3D point cloud and render a scene. Rendering a scene with a GSFL is straightforward due to two inherent advantages. First, the laser is the only illumination source and is located next to the FPA with a fixed offset. Second, because of ASC's global shutter technology and extremely short exposure time, the scene is essentially frozen in time and motion within scene has no impact on the rendered output – unlike scanning LiDAR and traditional cameras. A simple example of ray tracing is demonstrated in Fig. 8. Primary rays (rays from the focus to the center of each pixel) are cast out and the intersection between each ray and each mesh in the scene is tested. If the ray intersects multiple objects, the closest object is returned.

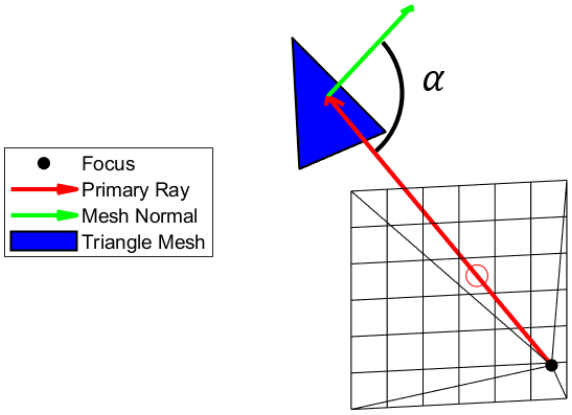

Fig. 8. Ray tracing a triangular mesh

The scene is oversampled (multiple rays per pixel) in order to render sub-pixel or partially illuminated pixels correctly. The ray tracing algorithm outputs the x, y, and z coordinates of the intersection point and the incident angle, α, between the primary ray and the normal vector of the mesh. Once the intersection points are calculated, the length of

each ray is calculated to define the range map which is output by the GSFL. An example 3D point cloud and oversampled range map of a Tracking and Data Relay Satellite (TDRS) model is shown in Fig. 9.

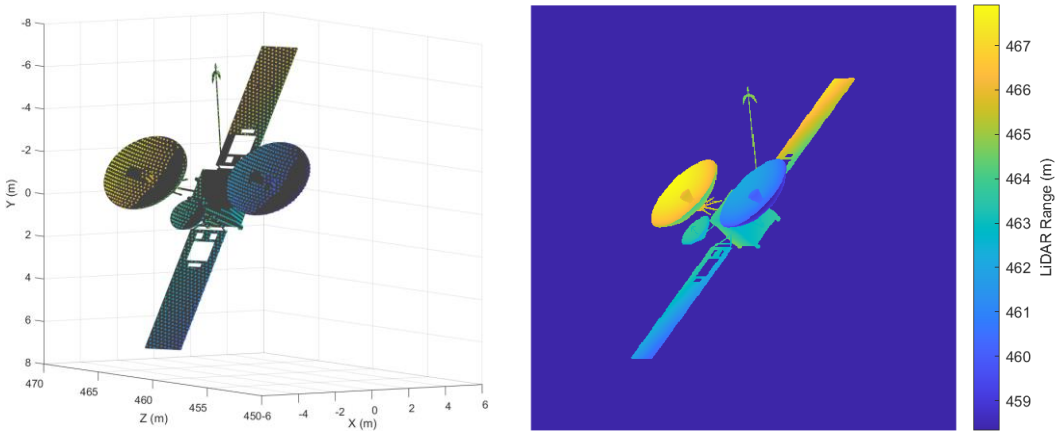

Fig. 9. (Left) Simulated OPC of a TDRS model. (Right) Oversampled TDRS range map.

In order to calculate the amount of light reflected onto each pixel, radiometry is applied to the range map with the assumption that all targets are extended targets. The radiometric equation is shown in **(7)** where E_p is the energy reflected on a pixel in photons, E_{tx} is the total transmitted laser energy in photons, R is the range to the target (for that pixel) in meters, θ_l is the full angle divergence of the laser in radians which is created with a square diffuser to match the focal plane, ρ is the BRDF function, D is the diameter of the aperture in meters, $IFOV$ is the instantaneous field of view of the lens in radians, η_{opt} is the transmission of the optical system, and α is the incident angle in radians.

$$E_p = \frac{E_{tx}}{R^2 \tan^2(\theta_l/2)} \cdot \rho \cdot \frac{\pi D^2}{4} \cdot IFOV^2 \cdot \eta_{opt} \cdot \cos(\alpha) \tag{7}$$

The BRDF (ρ) of a material (or mesh object) is defined in **(8)** where x is the position on the object surface, ω_i is the direction of the incident light, ω_o is the direction of the reflected light, $dL_o(x, \omega_o)$ is the differential amount of radiance that is reflected at point x in the direction ω_o, $L_i(x, \omega_i)$ is the amount of radiance coming in at a point x along direction ω_i through $d\omega_i$, $\cos(\alpha)$ is the cosine of the angle between ω_i and the surface normal at x, and $d\omega_i$ is a differential angle around ω_i.

$$\rho(x, \omega_i, \omega_o) = \frac{dL_o(x, \omega_o)}{L_i(x, \omega_i) \cos(\alpha)\, d\omega_i} \tag{8}$$

The simulator currently supports two BRDF models: the Lambertian and the modified Phong model. The Lambertian model, shown in Equation **(9)**, defines the reflectance of perfectly Lambertian or diffuse material. These materials reflect light equally in all directions so the amount of reflected light depends only on ρ_0, the Lambertian reflectance of the material.

$$\rho_{Lamb} = \frac{\rho_0}{\pi} \tag{9}$$

Almost all materials have BRDFs that are somewhere in between perfectly diffuse and perfectly specular. The modified Phong model [2], defined in Equation **(10)** and demonstrated in Fig. 10, attempts to model the complicated mix of diffuse and specular reflections in a computationally efficient way with tunable parameters. The Phong BRDF is dependent on k_d, the diffuse reflectivity, k_s, the specular reflectivity, n, the specular exponent, and φ, the angle between the perfectly specular reflective direction and the outgoing direction ($\varphi \approx 2\alpha$). The factors k_d, k_s, and n are tuned to match a material's reflectance using empirical measurements.

$$\rho_{Phong} = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \varphi \tag{10}$$

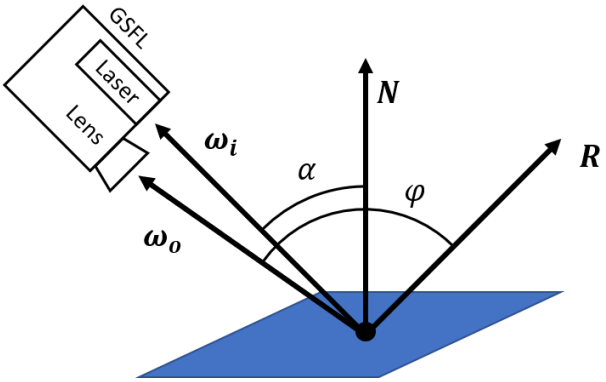

Fig. 10. Modified Phong BRDF diagram

It is important to note that BRDFs are wavelength dependent and all BRDFs defined for the simulator are modeled at either 1064nm or 1570nm, depending on the laser configuration.

The left side of Fig. 11 shows the photon image that is calculated from the oversampled point cloud in Fig. 9 with a Lambertian reflectance model with $\rho_0 = 0.30$ applied to the mesh. This image is then converted to the resolution of the GSFL (128x128) which is shown on the right side of Fig. 11. Note that as the pixel size increases, the number of photons per pixel increases as expected.

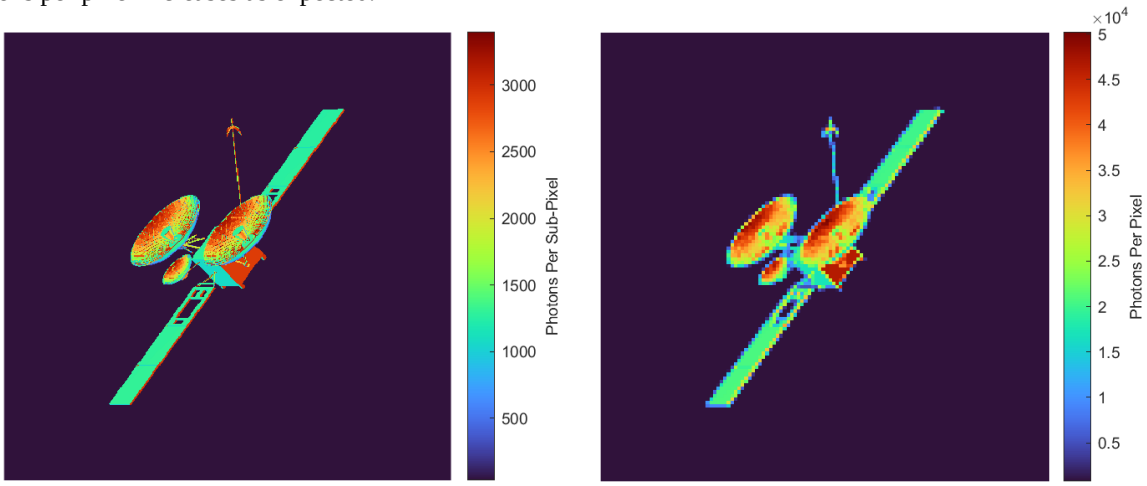

Fig. 11. (Left) Oversampled photon image of a TDRS model. (Right) Photon image converted to the GSFL resolution.

3.3 DETECTOR RESPONSE

The GSFL FPA consists of 128x128 InGaAs APDs which have a gain factor of ~10. These APDs are bonded to a proprietary readout for acquisition. The global shutter, which is part of the readout design, enables each pixel to simultaneously measure the time of flight and the intensity waveform of the return pulse. In order to collect and digitally sample the return pulse, the readout has an identical circuit in each pixel called the unit cell (UC). The first stage of the UC consists of a transimpedance amplifier (TIA) that converts the current output by the APD into a voltage. This voltage is then buffered, sampled 20 times when the return pulse arrives, and converted to digital values with an analog to digital converter (ADC). This buffer, along with the ToF data, is sent to a field-programmable gate array (FPGA) which applies a range algorithm to convert this raw waveform data into a range measurement. The range algorithm applies digital filtering techniques to the 20-cell buffer data to calculate the range to a precision that is better than the system clock speed. The result of this signal chain is a digital value for the range to the target and the intensity of light for every pixel in the array. Each stage of this signal chain is modeled in the simulator. The result of applying the detector model to the photon image is shown in Fig. 12.

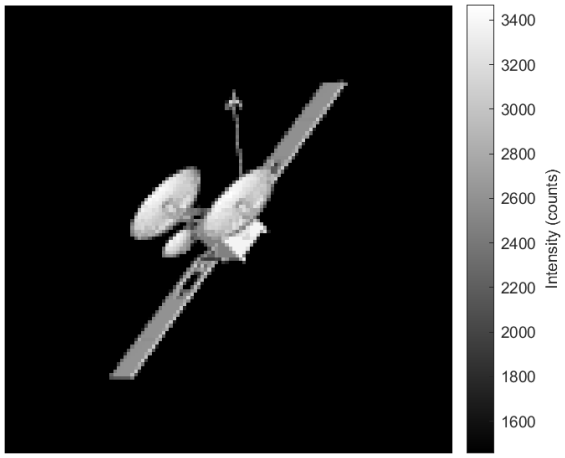

Fig. 12. Simulated intensity image of a TDRS model

The fidelity of the simulated detector response has been empirically verified at ASC. Fig. 13 shows the results of this verification testing for both high and low photon flux.

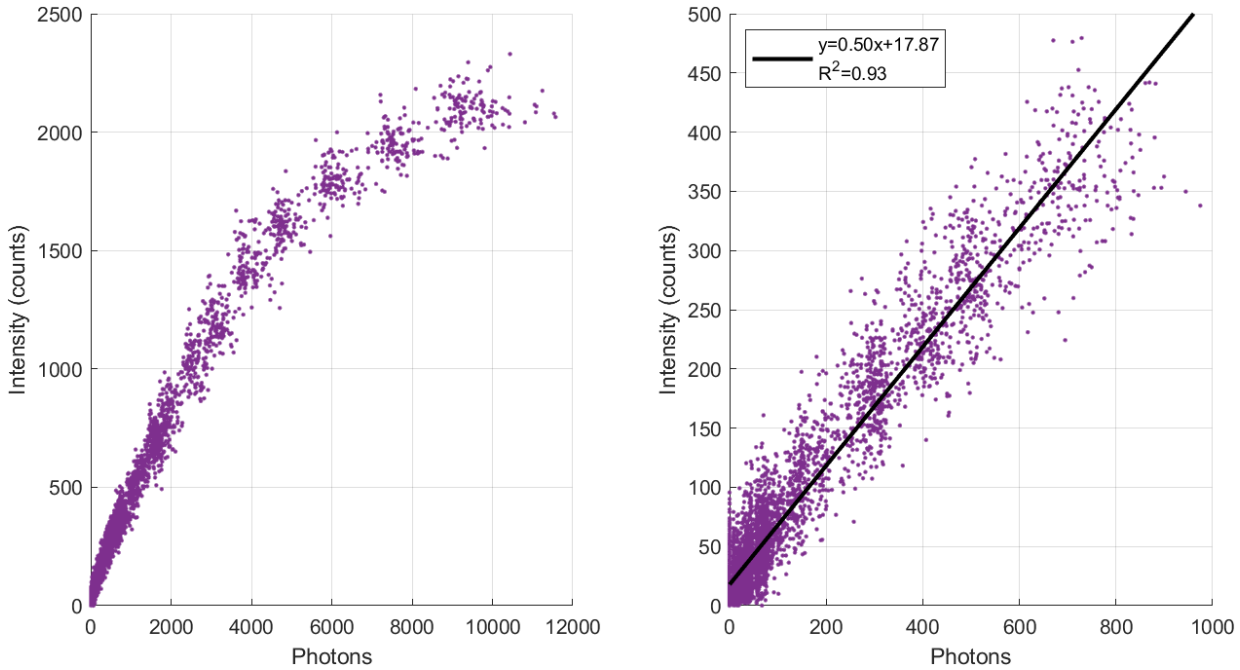

Fig. 13. (Left) Relationship between photons incident on a pixel and GSFL intensity. (Right) Same data set zoomed in to the linear regime.

In order to trigger the sampling and ADC circuits, each UC has a Schmidt Trigger which latches when enough return light has arrived. This Schmidt Trigger has an associated probability of detection (PD) that is dependent on the photon flux. With the current readout design, the PD is 95% for a flux of 1000 photons and decreases for lower photon flux. In order to improve sensitivity and maximum range performance, ASC is investigating range gating and other techniques which enable reliable detection down to 30 photons [3]. The range gating mode is not currently supported in the simulator but is in development.

ASC has also developed noise models for both the intensity and range outputs that are modeled after empirical data collected at ASC. The intensity noise is mostly Gaussian but the standard deviation changes as the detector approaches saturation. The range noise also follows a Gaussian distribution that is dependent on the intensity of the return pulse. This intensity dependence is due to the use of sampled intensity data to calculate the range value: the higher the signal-to-noise ratio (SNR), the more precisely the digital filter can calculate the range. The GSFL has a nominal range

precision (1 sigma) of 5cm with sufficient SNR. As the SNR decreases, the range precision degrades. With these empirically verified detector and noise models, the simulator produces highly accurate R&I maps that are suitable for algorithm development and AI workflows.

4. DEEP LEARNING WORKFLOW FOR RSOS

ASC uses the simulation testbed for DL-CNN work related to RSOs, including creating large and diverse datasets for supervised network training, test, and validation. To produce these datasets ASC uses a Deep Learning workflow based on MATLAB and its associated Toolboxes, which integrates well into our MATLAB simulation and rapid prototyping software development approach. The MATLAB Deep Learning workflow that ASC uses is summarized at a high level in Fig. 14.

4.1 RSO DATASET GENERATION

The RSO Dataset Generation phase of the workflow uses NASA satellite CAD models as a starting point for the RSO Deep Learning workflow. The end result of the Dataset Generation phase is a large set of realistic OPCs representing these models. This is accomplished by using the GSFL simulator in a scripting or batch workflow to generate a large number of realistic OPCs with the same form and fidelity as those generated by the ASC GSFL cameras. The GSFL frame data is an OPC in the form of an array of vectors of information consisting of 3D (x, y, z) data, range, and intensity, where each vector conceptually maps one-to-one to its corresponding 2D time-of-flight pixel indexed by a row and column in the simulated FPA. The OPC format for the GSFL simulation also provides an efficient matrix-based interface to the pre-processing steps in the workflow and to the Deep Learning networks.

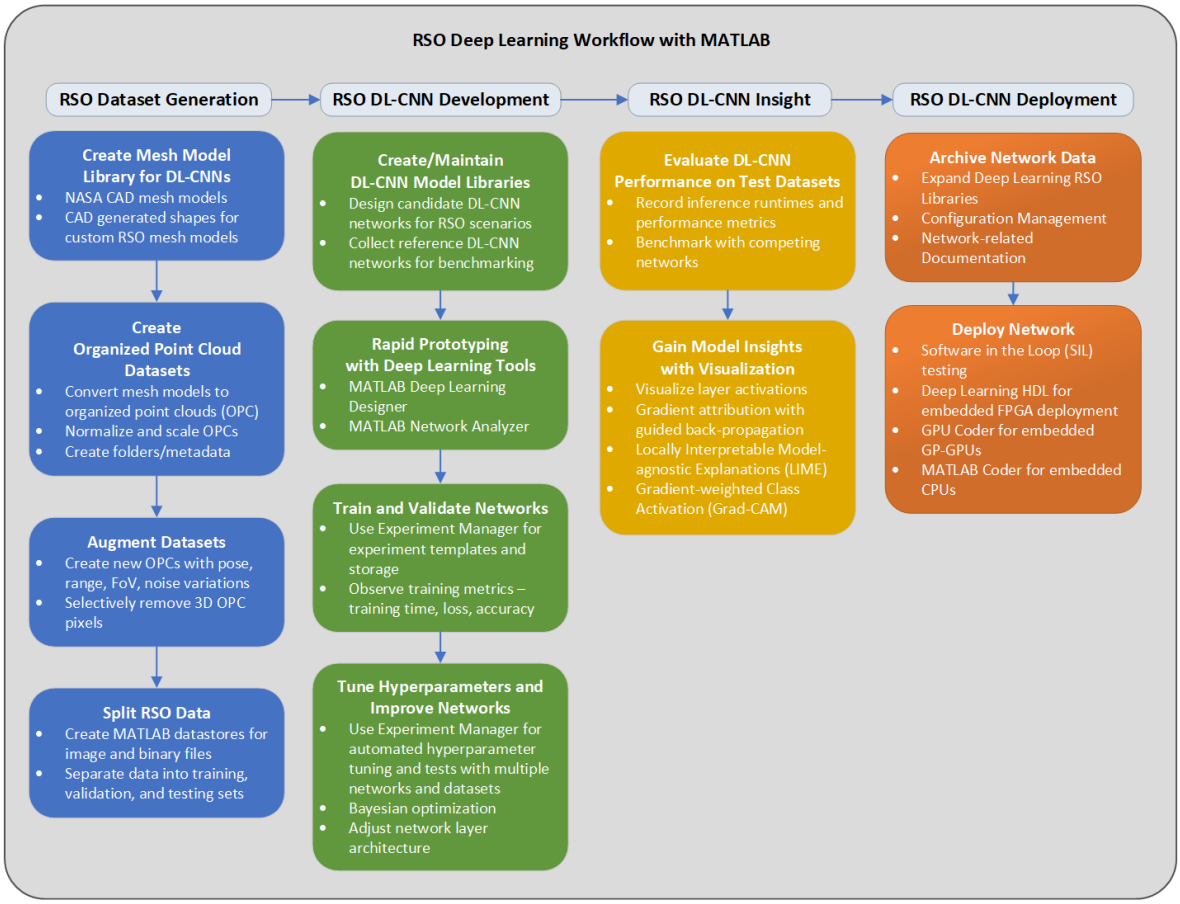

Fig. 14. MATLAB Deep Learning Workflow used at ASC

The satellite model library uses NASA CAD models as an initial source used for scaled mesh models and 2D visualization. Additional mesh models are generated with custom CAD models. The mesh models are used by the

simulator with scenarios and the GSFL camera model to generate OPCs as described in Section 3. To create the large amount of data needed to effectively train and validate the DL-CNNs, these OPC datasets are augmented by rotating the mesh models in 3 dimensions and re-simulating to create numerous poses. The range from GSFL camera to the satellite is also varied and re-simulated. This results in the number of GSFL pixels impinging on the satellite mesh models varying in proportion to the range. Changes in the GSFL configuration can also be simulated independently to increase the number of OPCs generated. Finally, randomization at low levels of the simulator, such as GSFL range and intensity noise and probability of detection (randomized missing pixels), add additional dimensions to the dataset. All of these simulation variations are controlled through scripting for a batch processing or single scenario approach.

To complete the RSO Dataset Generation phase, the OPC data generated is organized into MATLAB compatible formats designated as datastores or imagestores. This allows for efficient input to MATLAB functions called by Deep Learning scripts or object-oriented programs. These datastores are further organized into sets for DL-CNN training, validation, and test. The selection of OPCs assigned to these sets may be programmatically shuffled as well.

4.2 RSO DL-CNN DEVELOPMENT

The RSO DL-CNN Development phase uses the simulation data generated in the RSO Dataset Generation phase to assist in creating new networks which are effective for RSO inference tasks including classification, semantic segmentation, pose determination, and super-resolution. The efficiency and successful outcome of this task is largely dependent on techniques using features in the MATLAB Deep Learning Toolbox and the quantity and quality of the simulated organized point cloud data. The layered architecture of the DL-CNN networks is graphically displayed and edited in the Deep Learning Designer. This tool also performs error checking to resolve layer interface issues, for example. Similarly, the Network Analyzer graphically displays additional layer information and is used to resolve design issues. After the network layer and interconnection design is complete and error-free, the DL-CNN is ready for training and validation. ASC uses NVIDIA general purpose GPU cards such as the RTX 3090 running at 1.56 GHz with 24 GB of RAM and 10752 cores to improve training times and accommodate complex DL-CNN designs. Tuning training and validating hyper-parameters is simplified using the Experiment Manager, and multiple DL-CNNs may also be accommodated.

4.3 RSO DL-CNN INSIGHT

The RSO DL-CNN Insight phase incorporates numerous performance evaluations of the DL-CNN candidates, and at a deeper level, observations with visualization methods to gain a more complete understanding of behavior at the network layers. ASC uses MATLAB scripting to set up OPC-based test sets to evaluate candidate DL-CNNs with industry and academic metrics including accuracy, precision and recall, as described in [4]. The results of these tests are compared with various DL-CNNs for RSO inference tasks to benchmark candidate DL-CNNs. The goal of this phase is to evaluate any improvements in inference speed and performance. Additionally, ASC is exploring the use of visualization functions such as Gradient-weighted Class Activation (Grad-CAM) [5], Locally Interpretable Model-agnostic Explanations (LIME) [6], and gradient attribution techniques [7] to gain a better understanding of network layer design and the effectiveness of various deep learning approaches.

5. SIMULATION AND DL-CNN WORKFLOW DEMONSTRATION

In order to demonstrate the capabilities and functionality of the GSFL simulator, ASC developed a classification DL-CNN workflow utilizing the NASA CAD library. ASC selected seven models from the library that span a wide range of shapes and sizes (4m to 30m in the largest dimension). The seven models selected are: the Dawn space probe, the Juno space probe, the Lunar Reconnaissance Orbiter (LRO), the Suomi National Polar-orbiting Partnership (Suomi NPP), the Rosetta space probe, the Stardust space probe, and the Tracking and Data Relay Satellite (TDRS). Intensity images generated with the simulator of each of these spacecrafts are shown in Fig. 15.

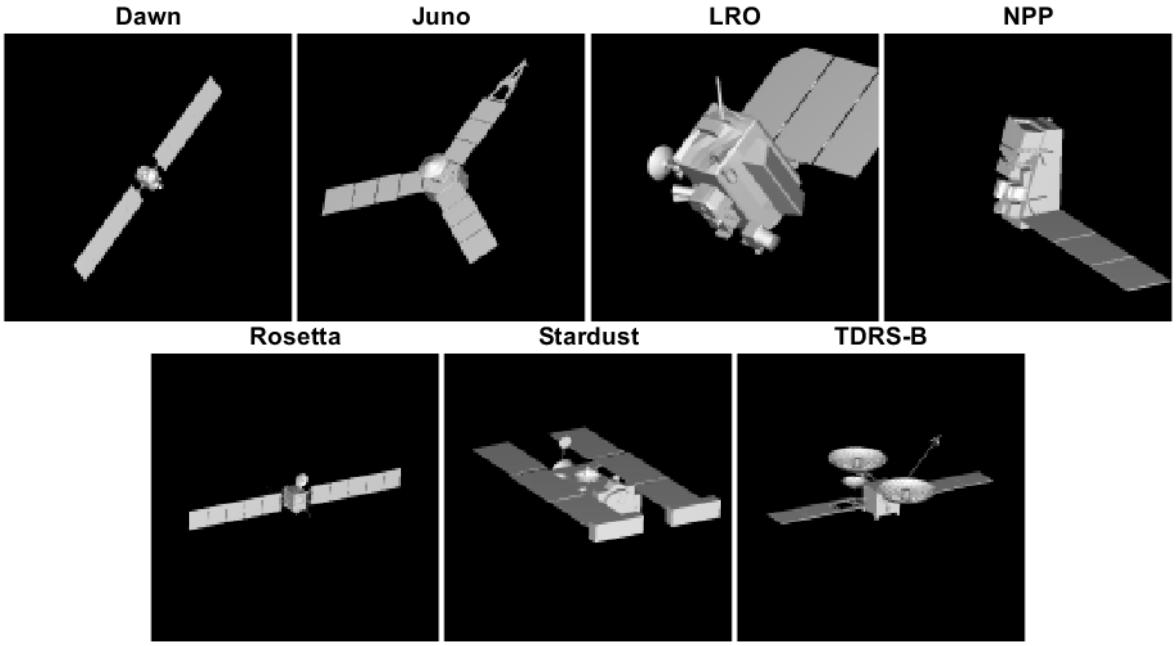

Fig. 15. Simulated GSFL intensity images of seven RSO spacecraft

Using the techniques described in Section 4.1, ASC created a datastore of 2,800 (400 per satellite) images and OPCs. The GSFL system parameters were optimized for long range imaging while keeping the size, weight, and power (SWAP) low: the lens FoV is 2.5°, the lens f-number is 2.8, the laser wavelength is 1064nm, the laser energy is 50mJ, the laser divergence is 2.5°, and each mesh model has a Lambertian BRDF with $\rho_0 = 0.30$. This GSFL configuration has a maximum range of about 3km, and using the simulator, each satellite was placed at 400 ranges between 3km and the minimum range where the satellite fills the full FoV (dependent on the satellite size). The satellites were also given random orientations at each range. The datastore was then divided into training and test data with 60% going to training and the remaining 40% saved in a test datastore. A DL-CNN classifier was developed and trained according to the MATLAB workflow described in Sections 4.2 and 4.3. The classifier was tested on the test datastore and the results are shown in Fig. 16, which shows the precision-recall curves for each satellite.

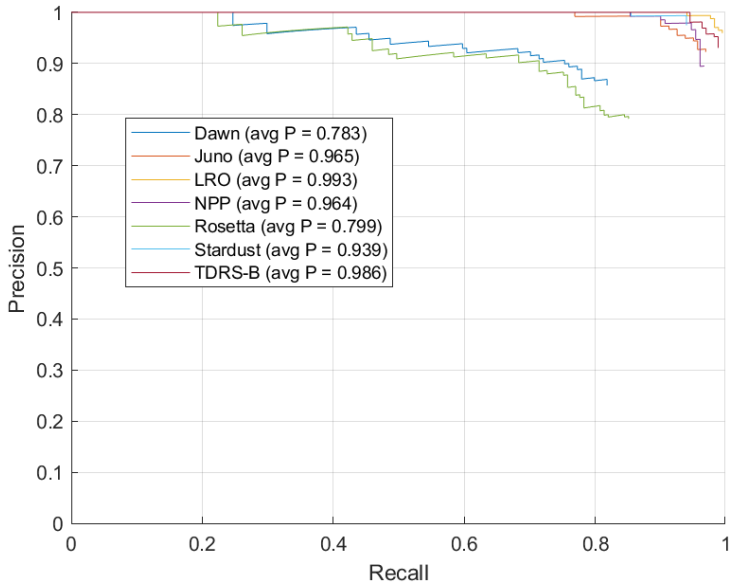

Fig. 16. Precision-Recall curves for each satellite

As seen above, the DL-CNN struggles to differentiate between the Dawn and Rosetta spacecrafts. This is not surprising as these two satellites have very similar shapes and size. Further analysis shows the classifier's dependency on range; since the satellites have varying sizes and random orientations in each frame of OPC data, the number of illuminated pixels is a more apt metric to judge the performance. This is dependency shown in Fig. 17, which shows that approximately 100 (10x10 square) pixels are needed for reliable classification. With an advantageous relative orientation, this occurs at about 1km for the smaller satellites and >3km for the larger ones. In order to extend this range, the GSFL would need a larger lens with a narrower field of view. The laser divergence could also be adjusted to only illuminate the necessary pixels 10x10 or 20x20 square to extend the range even farther. These performance and SWAP tradeoffs depend on the ConOps and requirements of the application.

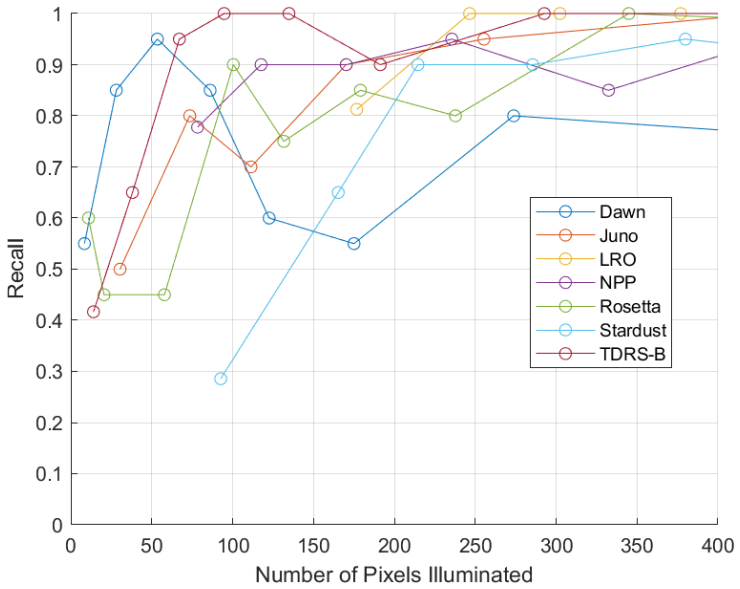

Fig. 17. Classification recall as a function of number of pixels illuminated

Larger datasets and further algorithm development could improve the results, but that is not the primary goal of this exercise. This example instead shows how the GSFL simulator can easily build a large training data set and how it fits seamlessly into the DL-CNN workflow. The end result is a powerful DL-CNN developed at little to no cost.

6. CONCLUSION AND FUTURE WORK

ASC is actively working on increasing the fidelity and adding features to the GSFL simulator. To increase fidelity, ASC is developing improved lens models that support realistic point spread functions (PSF) and distortion. ASC performs lens design in Zemax which models these parameters. The simulator will be modified to accept these parameters to improve the lens modeling. The next step in fidelity improvements is to implement the range gating mode discussed in Section 3.3 and [3]. This will enable the development of the necessary software and firmware algorithms to support effective deployment of this mode for RSO and altimetry applications. In terms of additional features, software in the loop simulation is an active development area and continues to grow in importance. Emphasis will be placed on real-time or near real-time simulations with system level interfaces. In addition, a graphical user interface (GUI) will be developed to allow for intuitive and efficient scenario creation.

7. REFERENCES

[1] NASA 3D Models. (n.d.). Nasa 3D Resources. Retrieved August 1, 2022, from https://nasa3d.arc.nasa.gov/models
[2] Eric Lafortune, et al., "Using the Modified Phong Reflectance Model for Physically Base Rendering", Nov 1994.
[3] Lane Fuller, Amy Carl, Joe Spagnolia, Brad Short, Michael Dahlin, "Photon counting linear mode global shutter flash LIDAR for improved range performance," Proc. SPIE 12110, Laser Radar Technology and Applications XXVII, 1211005 (3 June 2022); https://doi.org/10.1117/12.2619047

[4] Gad, A. F. “Accuracy, Precision, and Recall in Deep Learning.” Paperspace Blog (April 9, 2021). https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/
[5] Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.” 2017 (October 2017): 618–626, https://doi.org/10.1109/ICCV.2017.74.
[6] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier.” In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–44. San Francisco California USA: ACM, 2016. https://doi.org/10.1145/2939672.2939778
[7] Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” *ArXiv:1312.6034 [Cs]*, April 19, 2014. http://arxiv.org/abs/1312.6034