

Real-Time Pose and Dynamics Estimation of Non-Cooperative RSOs using Flash LiDAR for Autonomous Rendezvous and Docking

**Dr. Ricardo Delgadillo, Dr. Robert Stevenson-Karl, Dr. Roger Stettner, Lane Fuller,
Bruce Anderson, Michael Dahlin**
Advanced Scientific Concepts LLC

ABSTRACT

This study presents a Flash LiDAR-based approach for accurately determining the range and orientation of non-cooperative, free-tumbling resident space objects (RSOs). Pose determination is validated using Advanced Scientific Concepts (ASC)'s digital twin simulator and is found to be accurate at ranges up to 150 meters, with relative positioning accurate up to 500 meters. The research leverages ASC's Global Shutter Flash LiDAR (GSFL) due to its high resolution and high frame rate for precise range measurements. The computational techniques introduced are designed for robust dynamic pose determination, even for free-tumbling RSOs exhibiting rapid rotational motion. Real-time algorithms are introduced to process GSFL data, and performance is verified using a modified version of ASC's Flash LiDAR Digital Twin Simulator tailored for space rendezvous and docking applications. Results demonstrate the system's capability to accurately determine an RSO's position and orientation despite high angular rates, contributing to advancements in autonomous space operations and satellite servicing.

1. INTRODUCTION

A key challenge in real-time space situational awareness is enabling a spacecraft to rendezvous and dock with free tumbling uncooperative resident space objects (RSO). Critical to this mission is the accurate determination of the RSO's position, orientation, and velocity relative to the spacecraft. Conventional docking systems may rely on sensor suites composed of visible/IR imaging and/or scanning LiDAR which are susceptible to motion blur and are ineffective when used with free tumbling RSOs.

This work addresses the challenge of determining the dynamics of an uncooperative RSO exhibiting rapid rotational motion. The primary sensor selected is Advanced Scientific Concepts' (ASC) Global Shutter Flash LiDAR (GSFL), a Technology Readiness Level 9 (TRL-9) solid-state Flash LiDAR system that captures an entire frame of intrinsically organized 3D point cloud data with a single 10ns laser pulse. The GSFL enables an edge-computing or embedded system to compute real-time tracking of fast-moving RSOs without having to contend with motion blur, thus improving the pose estimation accuracy. GSFL output supports real-time predictive modeling and feature extraction of RSO dynamics. Furthermore, the GSFL offers several operational advantages over visible/IR imaging and/or conventional scanning LiDAR, including little or no risk of mechanical failure, and operation in the Earth's shadow and deep space where conventional cameras fail. The GSFL is radiation tolerant and can collect range data even when the Sun is in the field of view of the sensor. Furthermore, the GSFL's size, weight, and power (SWaP) efficiency surpasses that of scanning LiDAR with comparable resolution.¹

This work investigates real-time RSO dynamics determination algorithms leveraging the GSFL's high-resolution range-finding capabilities. Feasibility analysis is conducted using ASC's Flash LiDAR Digital Twin Simulator, adapted for RSO applications. The simulator provides radiometrically accurate intensity simulation via a Bidirectional Reflectance Distribution Function (BRDF), allowing for precise parameterization to closely match live data. Additionally, the simulator incorporates accurate range noise models on a per-pixel basis, and achieves photorealistic rendering results.

¹ The ASC GSFL has a very flexible design and can be special ordered from ASC with a variety of fields of view, ranges, laser wavelength, and radiation hardness.

For many docking scenarios, the geometric CAD model of the uncooperative target is assumed to be known, or can be derived from the GSFL 3D images. Pose determination is conducted during the mid-range rendezvous phase, with accurate orientation determination at distances up to 150 meters, and relative positioning accurate up to distances of to 500 meters. The algorithms in this paper apply both tracking and pose determination approaches. Results demonstrate that the GSFL, with its high-resolution and high-frame-rate data, is a suitable sensor for obtaining high-quality dynamic measurements of uncooperative RSOs, contributing to the advancement of autonomous space operations and satellite servicing.

2. SENSOR OVERVIEW: THE ASC GLOBAL SHUTTER FLASH LIDAR

A variety of depth-sensing technologies have been employed for autonomous navigation and object detection in space environments. These include structured light cameras, stereo cameras, scanning LiDAR, and Flash LiDAR systems. Each technology presents distinct trade-offs in terms of range, accuracy, power requirements, and environmental suitability.

Structured Light Cameras project IR patterns to infer depth and can operate in darkness using self-illumination. However, they suffer severely in *bright ambient conditions*—such as daylight—where interference wipes out projected signals. Their operational range is short (typically $\leq 1\text{--}2$ m), and they often require multiple exposures, making them unsuitable for dynamic or real-time scenes.

Scanning LiDAR technologies emit laser pulses and measure the time of flight to create depth maps. These systems are advantageous in that they are self-illuminating and can operate over several hundred meters using Class 1 eye-safe lasers. However, they suffer from motion blur and require mechanical components such as fast-rotating servos to reconstruct a useful 3D scene. Real-time processing is possible but computationally expensive, and obstacle detection accuracy is degraded under motion.

Flash LiDAR provides a compelling alternative. This sensor captures an intrinsically organized point cloud using a single, short (~ 10 ns) laser pulse, eliminating motion blur even for high-velocity targets. The GSFL can operate in direct sunlight or in Earth's shadow using a self-illuminating shortwave infrared (SWIR) laser, eye-safe or not, and offers high pixel resolution (16,384 pixels) for accurate range and intensity capture across its field of view.

Key advantages of GSFL for RSO pose determination include:

- Real-time AI-ready data output without the reconstruction overhead of scanning LiDARs
- Native organized point cloud (OPC) interfacing directly with AI inputs (e.g., 128×128 matrices)
- High detection fidelity, including thin structures like wires and antennas
- Low size, weight, and power (SWaP)

3. POSE ESTIMATION SYSTEM ARCHITECTURE AND THEORY

Pixel coverage for the GSFL is modeled as follows. For midrange distances between 10 and 150 meters, configuring the LiDAR system with an appropriate Field of View (FoV) ensures optimal pixel coverage of the target object. The number of pixels an object occupies in the GSFL camera's field of view (equivalently, the number of available range measurements) can be approximated by the following equation:

$$N_{pixels} \cong 2 \left[\left(\frac{Dim}{4 \|\vec{r}_{rel}\| \cdot \tan(FOV/2)} \right)^2 s^2 \left[\vec{n} \cdot \frac{\vec{r}_{rel}}{\|\vec{r}_{rel}\|} \right] \right]$$

Where:

- \vec{r}_{rel} is the vector from the target to the GSFL camera (relative viewing direction),
- \vec{n} is the surface normal of the target (at the centroid of the target),
- FoV is the camera's field of view in radians,

- $Dim = 128$ is the number of pixels per image dimension (i.e., GSFL resolution is 128×128).
- $[x] = \max(x, 0)$, ensures that only the forward-facing surface contributes to the pixel count,
- s^2 is the area of a square patch of side length s .

This approximation assumes that the object occupies a small portion of the total image. For objects of arbitrary shape, the square area (s^2) can be replaced with the object's projected area. The resulting equation resembles the light intensity distribution on an object's surface and was validated using the ASC Digital Twin Simulation Environment.

For example, for a large satellite with $\sim 0.5 \text{ m}^2$ cross-sectional area—such as MEV or DARPA's RSGS—using a GSFL camera with a fixed 3° FOV. The GSFL captures approximately:

- 3,721 pixels at 10 m
- 16 pixels at 150 m
- Roughly 1 pixel at 500 m, depending on reflectivity

Assuming independent per-pixel range errors where the standard deviation grows linearly with distance:

$\sigma(r) = 0.01 \cdot r$ (i.e. 1% error)—the standard error of the mean measurement scales as:

$$\sigma_N = \frac{\sigma(r)}{\sqrt{N_{pixels}}}$$

For a 10m distance:

$$\sigma(10) = 0.10m, \quad \sigma_N = \frac{0.10}{\sqrt{3721}} \approx 0.0016m$$

At 150m:

$$\sigma(150) = 1.5m, \quad \sigma_N = \frac{1.5}{\sqrt{16}} \approx 0.375m$$

Hence the pose accuracy at 10m is $\sim 234\times$ better than at 150 m:

$$\frac{\sigma(150)/\sqrt{16}}{\sigma(10)/\sqrt{3721}} \approx \frac{0.375}{0.0016} \approx 234$$

This disparity highlights the challenge of accurate satellite pose estimation at greater distances. However, increasing the number of effective pixels improves the accuracy at longer ranges by increasing the number of range-sampling pixels or narrowing the FOV to concentrate pixel density on the target.

3.1 Real-time Processing Pipeline

The algorithm presented in this paper is meant to run efficiently enough to produce real-time position and orientation of the target satellite using a suitable radiation tolerant or hardened embedded computer with compute capabilities similar to the Nvidia Jetson Orin NX. The compute capabilities include a 6-core ARM CPU running at 2.4 GHz and a GPU running at 765 MHz with 100 trillion operations per second (TOPS).

The overall data processing pipeline consists of several stages, illustrated conceptually in Fig. 1:

1. **GSFL Data Input:** The system receives range and intensity data from the GSFL.
2. **Point Cloud Coordinate Transformation:** Raw LiDAR data is transformed from the LiDAR frame into other reference frames through transformation matrices.
3. **Point Cloud Registration:** Using Iterative Closest Point (ICP) algorithms, the system aligns the measured point cloud with a reference model of the satellite, generating an accurate transformation matrix representing the satellite's pose relative to the sensor.
4. **6 Degrees of Freedom (6-DoF) Extraction:** The transformation matrix is used to extract position and orientation parameters of the satellite.

5. **Kalman Filter:** The extracted pose data is filtered using a Kalman filter to reduce noise and improve estimate stability.
6. **Quasi-Periodic Motion Estimator:** Finally, the filtered pose information is analyzed to estimate the satellite's rotational velocity and motion dynamics.

The initial stage of the pipeline involves transforming filtered range data between multiple coordinate frames.

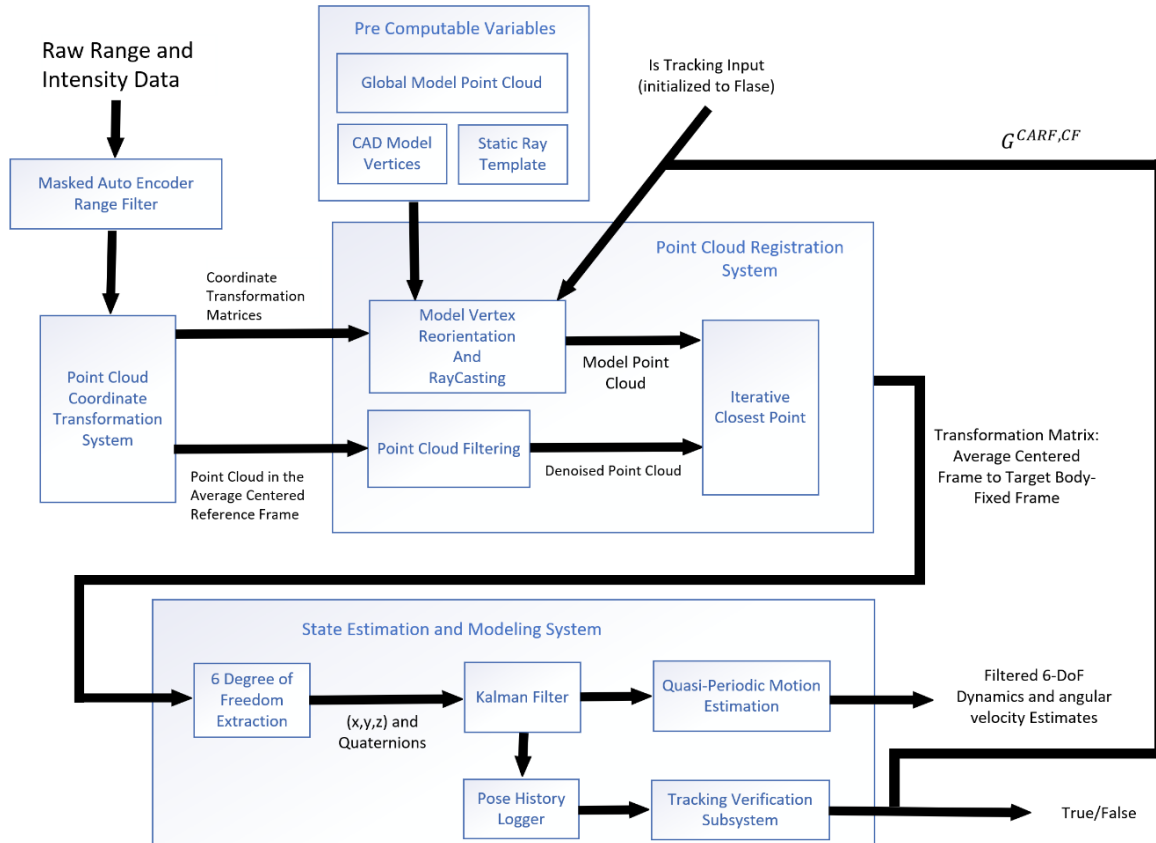


Fig. 1. Real-time GSFL-based satellite pose estimation pipeline

3.2 Reference Frames

The reference frames used throughout this paper are described in the following sections. A depiction of the coordinate reference frames and their relationships to each other is shown in Fig. 2. The red path in this figure illustrates the transformation used to position the CAD model vertices approximately near the point cloud data in the LiDAR reference frame. The black path shows how to transform point cloud data into either the Servicer Body Frame or the Target Body ISO Frame.

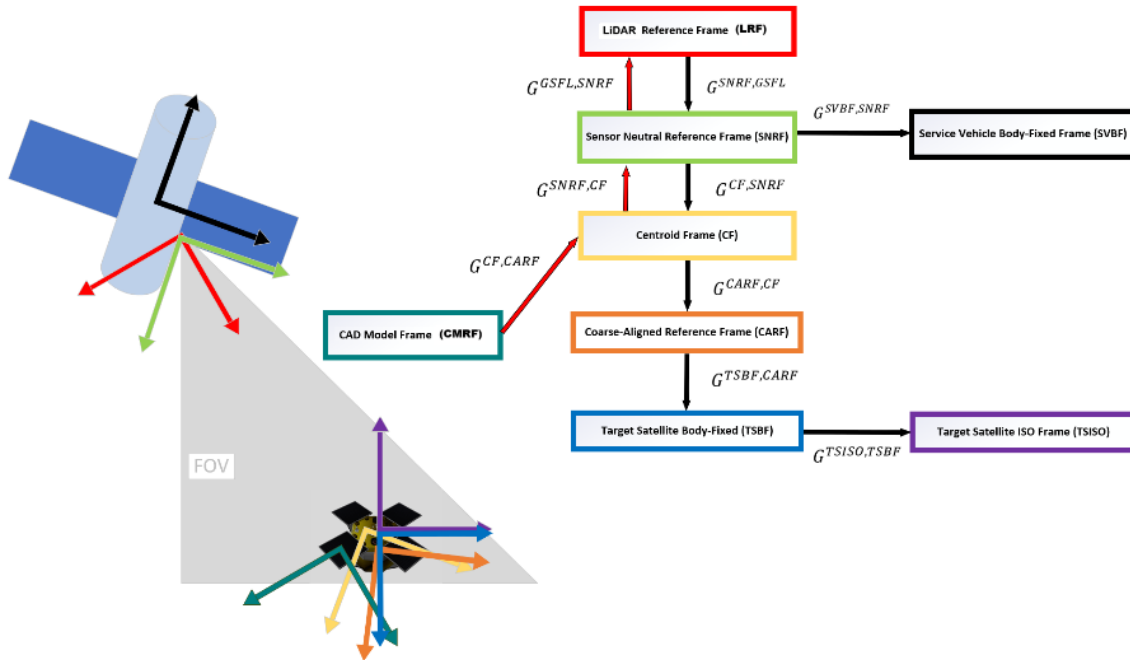


Fig. 2. Visualization of coordinate reference frames and the coordinate frame graph

The GSFL camera's body-fixed LiDAR Reference Frame (LRF) is a right-handed Cartesian coordinate system rigidly attached to the camera, with its origin at the aperture as shown in Fig. 3. In the figure, \mathcal{O} denotes the location of the aperture and \mathcal{O}' is the location of the center of the virtual image plane. The focal length is denoted as f . The Z axis is aligned along the direction of outgoing laser pulse, perpendicular to the detector plane., The X and Y axes are oriented according to the right-hand rule, defining the horizontal and vertical directions of the camera's focal plane.

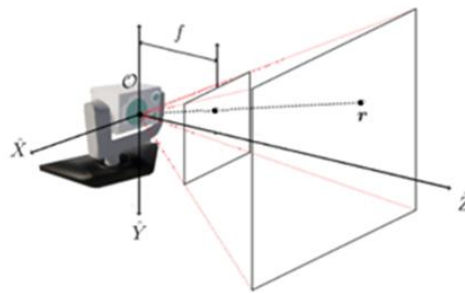


Fig. 3. GSFL camera body-fixed coordinate system

The Service Vehicle Body-Fixed Reference Frame (SVBF), shown in Fig. 4 is a right-handed Cartesian coordinate system rigidly attached to the servicer spacecraft, with its origin located at the spacecraft's center of mass. The axes are defined in accordance with the ISO 1151 convention for aerospace vehicles, where the X-axis points forward along the spacecraft's longitudinal (roll) axis. The Y-axis points towards the right side of the spacecraft, perpendicular to the X-axis in the horizontal symmetry plane. The Z-axis points downward, completing the right-hand convention, and is perpendicular to both the X and Y axes. This frame serves as the primary reference for expressing vehicle dynamics, control inputs, and component alignments.

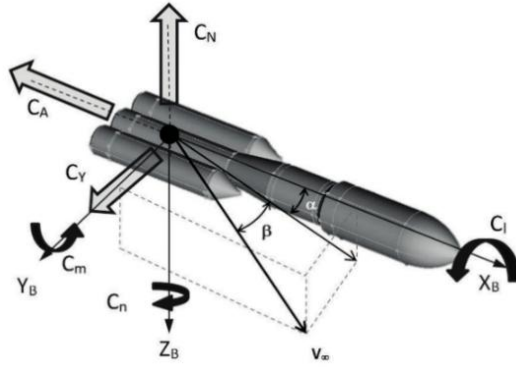


Fig. 4. Service vehicle body-fixed reference frame

The Sensor Neutral Reference Frame (SNRF) represents the orientation of the GSFL if it were mounted on a gimbal and the gimbal's yaw, pitch, and roll were all zero. In this frame, the GSFL's z-axis is aligned with the sensor's mechanical mounting direction, providing a consistent basis for interpreting gimbal angles and sensor pointing. When the GSFL is mounted on a gimbal, its orientation is altered by the gimbal's yaw, pitch, and roll rotations. As a result, the GSFL's axes, particularly the z-axis, may not align with any consistent or fixed direction in space. To define a stable and repeatable reference frame for analysis, a virtual counter-rotation is applied to the GSFL, equal in magnitude but opposite in direction to the gimbal's current yaw, pitch, and roll angles. This operation effectively removes the influence of the gimbal rotation, restoring the sensor to its nominal, zero-angle orientation.

After computing the point cloud in the SNRF, the average of the x, y, and z coordinates of the cloud is calculated to determine the centroid. By subtracting this centroid from each point, the cloud is recentered such that its mean position lies at the origin. The resulting coordinate system is called the Centroid Frame (CF). It shares the same orientation as the SNRF but differs by a translation equal to the centroid offset.

The Target Satellite Body-Fixed Reference Frame (TSBF) is a right-handed Cartesian coordinate system rigidly attached to the target satellite, with its origin located at the satellite's center of mass. The X and Y axes follow the convention that the Y-axis points backwards along the satellite's longitudinal axis, typically opposite to the nominal velocity or thrust direction, the X-axis points to the right, perpendicular to the Y axis and lies within the satellite's plane of symmetry. The Z-axis points in the direction of the cross product of the X-axis and Y-axis, completing a right-handed coordinate system. This frame is fixed relative to the satellite's structure but does not necessarily follow ISO conventions.

The Coarse-Aligned Reference Frame (CARF) is a reference frame that is approximately aligned with the TSBF, and is derived algorithmically using a pre-alignment process. This transformation brings the sensor data into rough rotational and translational alignment with the TSBF, providing a better initialization for fine alignment techniques such as the Iterative Closest Point (ICP) algorithm. Details on how this transformation is computed are provided in the following section.

The Target Satellite ISO Reference Frame (TSIRF) is a right-handed body-fixed Cartesian coordinate system rigidly attached to the target satellite, with its origin located at the satellite's center of mass. It adheres to the ISO 1151 aerospace convention, with X-axis pointing forward along the satellite's longitudinal axis, typically aligned with the nominal velocity or thrust direction, and the Y-axis pointing to the right, perpendicular to the X-axis, lying within the satellite's plane of symmetry. The Z-axis points downward, completing the right-handed coordinate system, perpendicular to both the X and Y axes. This frame is fixed relative to the satellite's structure and is used for standard-compliant alignment and interpretation.

The Computer Aided Design (CAD) Model Reference Frame (CMRF) is used to ensure that the coordinate axes align with the LiDAR Reference Frame, and that the target satellite's CAD model is in its neutral pose. This requires that the origin of the CAD Model Reference Frame coincides with the origin of the SVBF. Additionally, the X-axis

of the CAD frame points in the same direction as the y-axis of the satellite's body-fixed frame, and the Y-axis of the CAD frame points in the opposite direction of the satellite's x-axis. The Z-axis of the CAD frame is aligned with the z-axis of the satellite's body-fixed frame.

3.3 Coordinate Frame Transformations

Each of the defined coordinate systems differ from one another by a translation \mathbf{r}_{AB} followed by a rotation R_{AB} :

$$\mathbf{q}_A = \mathbf{r}_{AB} + R_{AB}\mathbf{q}_B$$

Where $q = \langle q_x, q_y, q_z \rangle$ is a vector, and A and B denote reference frames A and B respectively, as shown in Fig. 5.

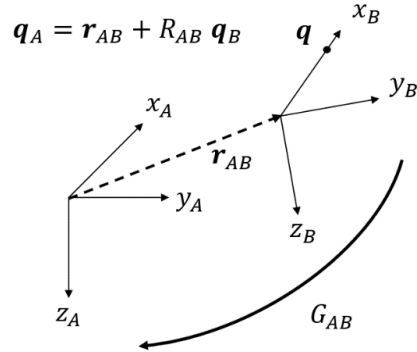


Fig. 5. Coordinate frame A and B translation and rotation differences

Translating coordinate frame origins is achieved through vector addition with \mathbf{r}_{AB} . Rotation of coordinate frames is done via a 3×3 matrix multiplication. The rotation matrix can be defined using one of 12 Euler Angle conventions or derived using a Quaternion-based rotation matrix.

3.4 Homogeneous Coordinates and Notation

Let $\langle x, y, z \rangle_i^{GSFL}$ represent the point cloud data in the LRF for $i = 1, 2, \dots, 16384$. To express these points in homogeneous coordinates, append 1 to each:

$$\vec{\mathbf{p}}_i^{LRF} = [x, y, z, 1]_i^T, \quad i = 1, 2, \dots, 16384.$$

The 4×4 homogeneous transformation matrix below is used to transform points from frame **B** to frame **A**:

$$G^{A,B} = \begin{bmatrix} \mathbf{R} & \vec{\mathbf{t}} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Where $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ is a rotation matrix and $\vec{\mathbf{t}} \in \mathbb{R}^3$ is a translation vector.

3.5 Point Cloud Notation

The shorthand notation $\vec{\mathbf{p}}_B^A$ is used to denote the entire set of points originating from point cloud **B** and expressed in reference frame **A**. The individual points in this set take the form:

$$\vec{\mathbf{p}}_i^A = [x, y, z, 1]_i^T, \quad i = 1, 2, \dots, N$$

where N , is the number of points in point cloud **B**. This may vary depending on the source and generation method.

The system processes the following categories of point cloud data:

- $\vec{\mathbf{p}}_{sensor}$: The live point cloud generated by the GSFL sensor after each laser pulse. This data captures real-time surface measurements and may include noise, occlusions, or missing points due to limited visibility.
- $\vec{\mathbf{p}}_{global}$: A synthetic point cloud created from the full CAD model of the target satellite. It consists of approximately uniform surface samples and is used for global registration and coarse alignment.

- \vec{p}_{local} : A view-dependent point cloud generated by ray casting into the CAD model, based on the current estimated sensor pose and the relative pose between the Servicer Satellite and the Target Satellite. This point cloud represents a noiseless approximation of \vec{p}_{sensor} and is used during the tracking phase for fine alignment (see subsequent subsections).

3.6 Ray Casting Rays

The GSFL simultaneously captures 128×128 range values arranged in an array format with 128 rows and 128 columns with each value denoted by $r_{i,j}$. Range is simulated by using 16,384 unique rays originating from the aperture of the GSFL camera. The direction of each ray is determined by the GSFL's focal length, pixel pitch ($pp = 0.1mm$), and specific row and column position on the image plane. The unnormalized direction vectors for each ray is defined as:

$$\vec{u}_{i,j}^{Unnormalized} = f\vec{z} + pp\left(-\frac{N-1}{2} + j\right)\vec{x} + pp\left(-\frac{N-1}{2} + i\right)\vec{y}, \quad i, j = 0, 1, 2, \dots, N-1$$

This is then normalized to obtain the unit direction vector:

$$\vec{u}_{i,j} := \frac{\vec{u}_{i,j}^{Unnormalized}}{\|\vec{u}_{i,j}^{Unnormalized}\|}$$

These unit vectors $\vec{u}_{i,j}$ define the Static Ray Template, a fixed set of rays expressed in the LRF. This ray set is reused throughout the system for:

- Generating simulated range data in the Digital Twin Simulator
- Performing ray casting into the CAD model to generate the local model point cloud (defined in a later section).

3.7 GSFL Range Data to GSFL Reference Frame

Starting from the GSFL range output, the point cloud coordinates with respect to the LRF are calculated by:

$$x = \frac{px \cdot R}{\sqrt{f^2 + p_x^2 + p_y^2}}, \quad y = \frac{py \cdot R}{\sqrt{f^2 + p_x^2 + p_y^2}}, \quad z = \frac{f \cdot R}{\sqrt{f^2 + p_x^2 + p_y^2}}$$

Where:

- f is the focal length,
- p_x and p_y are the pixel coordinates in millimeters, calculated as:

$$p_x = pp(i - 64.5), \quad p_y = pp(j - 64.5), \quad i, j = 1, 2, \dots, 128$$

3.8 GSFL Frame to Sensor Neutral Reference Frame

To express this point cloud in the SNRF, apply a rotation matrix derived from the gimbal angles ψ_{gimbal} , θ_{gimbal} , and ϕ_{gimbal} , (yaw, pitch, and roll, respectively). The corresponding quaternion components are:

$$\vec{q}_0 = \cos\left(\frac{\psi_{gimbal}}{2}\right) \cos\left(\frac{\theta_{gimbal}}{2}\right) \cos\left(\frac{\phi_{gimbal}}{2}\right) + \sin\left(\frac{\psi_{gimbal}}{2}\right) \sin\left(\frac{\theta_{gimbal}}{2}\right) \sin\left(\frac{\phi_{gimbal}}{2}\right)$$

$$\vec{q}_1 = \cos\left(\frac{\psi_{gimbal}}{2}\right) \cos\left(\frac{\theta_{gimbal}}{2}\right) \sin\left(\frac{\phi_{gimbal}}{2}\right) - \sin\left(\frac{\psi_{gimbal}}{2}\right) \sin\left(\frac{\theta_{gimbal}}{2}\right) \cos\left(\frac{\phi_{gimbal}}{2}\right)$$

$$\vec{q}_2 = \cos\left(\frac{\psi_{gimbal}}{2}\right) \sin\left(\frac{\theta_{gimbal}}{2}\right) \cos\left(\frac{\phi_{gimbal}}{2}\right) + \sin\left(\frac{\psi_{gimbal}}{2}\right) \cos\left(\frac{\theta_{gimbal}}{2}\right) \sin\left(\frac{\phi_{gimbal}}{2}\right)$$

$$\vec{q}_3 = \sin\left(\frac{\psi_{gimbal}}{2}\right) \cos\left(\frac{\theta_{gimbal}}{2}\right) \cos\left(\frac{\phi_{gimbal}}{2}\right) - \sin\left(\frac{\psi_{gimbal}}{2}\right) \sin\left(\frac{\theta_{gimbal}}{2}\right) \sin\left(\frac{\phi_{gimbal}}{2}\right)$$

The corresponding rotation matrix R is:

$$R = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_3q_0) & 2(q_1q_3 - q_2q_0) \\ 2(q_1q_2 - q_3q_0) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - q_1q_0) \\ 2(q_1q_3 - q_2q_0) & 2(q_2q_3 - q_1q_0) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

The homogeneous transformation matrix from the LRF to the SNRF is:

$$G^{SNRF, LRF} = \begin{bmatrix} R^{SNRF, LRF} & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$$

Applying this transformation to the point cloud data:

$$\vec{p}^{SNRF} = G^{SNRF, LRF} \vec{p}^{LRF}.$$

3.9 Sensor Neutral Reference Frame to the Service Vehicle Body Fixed Frame

To align with the ISO convention, a rotation matrix is required to reorient the basis vectors. Let $\vec{t}^{SVBF, SNRF}$ be the translation from the sensor (SNRF) to the SVBF. The transformation matrix is:

$$G^{SVBF, SNRF} = \begin{bmatrix} 1 & 0 & 0 & \vec{t}_x^{SVBF, SNRF} \\ 0 & 0 & 1 & \vec{t}_y^{SVBF, SNRF} \\ 0 & -1 & 0 & \vec{t}_z^{SVBF, SNRF} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

3.10 Sensor Neutral Reference Frame to the Centroid Frame

To express the point cloud in the Centroid Reference Frame, compute the mean of the coordinates in the SNRF:

$$\vec{t}^{CF, SNRF} = [-\langle \vec{p}_x^{SNRF} \rangle \quad -\langle \vec{p}_y^{SNRF} \rangle \quad -\langle \vec{p}_z^{SNRF} \rangle]^T$$

The transformation matrix from the SNRF to the CF is:

$$G^{CF, SNRF} = \begin{bmatrix} I_{3 \times 3} & \vec{t}^{CF, SNRF} \\ \vec{0}^T & 1 \end{bmatrix}.$$

Applying this transformation gives the point cloud in the CF:

$$\vec{p}^{CF} = G^{CF, SNRF} \vec{p}^{SNRF}.$$

3.11 Centroid Frame to Coarse aligned Reference Frame

This transformation defines the relationship between the CF and the CARF. There is no single, fixed transformation, as it depends on the current operational state and the pose of the target satellite.

- In non-tracking mode (e.g., at mission start or after losing track of the target), the transformation is initialized using a pre-alignment algorithm that provides an approximate alignment between CF and the TSBF.
- In tracking mode, the transformation is set to the most recently estimated transformation from CF to TSBF, preserving coarse alignment during ongoing pose estimation.

This transformation enables initialization or recovery of the CARF by aligning sensor data with a rough estimate of the target's orientation, supporting robust transition to fine alignment procedures.

3.12 CARF to Target Satellite Body-Fixed Reference Frame

This transformation maps the CARF to the TSBF. Like the CF-to-CARF transformation, it is not uniquely defined, as it depends on the current relative pose between the sensor and the target satellite. To compute the transformation $G^{TSBF,CF}$, two components are needed:

- A translation vector that maps the origin of the CARF to the origin of the TSBF frame origin.
- A rotation matrix that aligns the basis vectors/axis of the CARF and target TSBF.

These translation and rotation components are obtained using the Iterative Closest Point (ICP) algorithm, which is detailed in the next subsection.

3.13 Target Satellite Body Frame to Target Satellite ISO Frame

The TSBF and the TSISO Frame differ by a fixed rotation. This transformation accounts for a change in orientation between the two frames, while their origins remain coincident.

The transformation matrix $G^{CF,SNRF}$ is defined as:

$$G^{CF,SNRF} = \begin{bmatrix} \mathbf{R} & \vec{\mathbf{0}} \\ \vec{\mathbf{0}}^T & 1 \end{bmatrix}$$

where the rotation matrix \mathbf{R} is:

$$\mathbf{R} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This rotation realigns the axis according to the ISO standards orientation for the satellite, enabling consistent interpretation of body-fixed data in standardized reference frames.

3.14 Inverse Transformation Matrices

To fully traverse the coordinate frame graph, the inverse of each transformation matrix is required. These can be efficiently computed as:

$$\begin{aligned} G_{BA} &= G_{AB}^{-1} \\ &= \begin{bmatrix} R_{AB}^{-1} & -R_{AB}^{-1} \mathbf{r}_{AB} \\ \mathbf{0}^T & 1 \end{bmatrix} \end{aligned}$$

Where R_{AB}^T is the transpose (inverse) of the rotation matrix.

3.15 Point Cloud Registration System

It is assumed that the target satellite's geometric model is known and available from a CAD library representation, or can be uploaded. This model is crucial for applying the Iterative Closest Point (ICP) algorithm, which aligns live point cloud data with the reference geometry to estimate the transformation matrix $G^{TSBF,CARF}$.

The registration process operates within the framework of the Point Cloud Coordinate Transformation System, illustrated in Fig. 1. This system provides the point cloud data expressed in the CF as well as all relevant transformation matrices except $G^{CARF,CF}$ and $G^{TSBF,CARF}$, which are specifically computed during this step.

Additionally, the CAD model vertices, defined in the CAD Model Frame, are available for use in the alignment process. These inputs enable accurate registration of the sensor-derived point cloud to the known satellite geometry, forming the basis for pose estimation in the TSBF.

3.16 Preconditioning System

The Preconditioning Subsystem takes as input a Boolean flag, Tracking State, initialized to false at the start of the mission, and the transformation matrix $G^{TSBF,CF}$, which is initialized to the identity matrix at mission start and subsequently updated from the previous frame of GSFL sensor data.

Using these inputs, the subsystem computes the transformation $G^{TSBF,CF}$ based on the current tracking mode.

If Tracking State is true then:

$$G^{CARF,CF} = G^{TSBF,CF}$$

If Tracking State is false then:

$$G^{CARF,CF} = \text{PreconditioningFunction}()$$

The Preconditioning Function selects a transformation that minimizes the alignment error between the point cloud in the CF and the CAD model in the TSBF frame:

$$G^{CARF,CF} = \min \|\vec{p}^{TSBF} - G \vec{p}_{Global}^{CF}\|,$$

The Preconditioning Function is based on the Fast Point Feature Histograms algorithm (FPFH) to capture local geometry statistically (Rusu, Blodow, & Beetz, 2009).

3.17 CAD Model Repositioning and Local Point Cloud Model

After obtaining $G^{CARF,CF}$ from the preconditioning step, the following steps are applied:

1. Apply $G^{CF,CARF}$ followed by $G^{SNRF,CF}$ and then $G^{LRF,SNRF}$ to the vertices of the CAD model.
2. Apply ray casting using the rays defined above and the model repositioned in this way. This provides a model point cloud in the LRF, \vec{p}_{local}^{LRF} , this point cloud is roughly in the same location and orientation as \vec{p}_{sensor}^{LRF} .
3. Translate the model point cloud \vec{p}_{local}^{LRF} to the CF i.e. apply $G^{SNRF,GSFL}$ followed by $G^{CF,SNRF}$ to obtain \vec{p}_{local}^{CF} .

After step 3, the ICP algorithm is applied to register both \vec{p}_{local}^{CF} and \vec{p}_{sensor}^{CF} to obtain $G^{TSBF,CARF}$.

3.18 ASC Shallow Neural Network Performance and Results

Typically, before this is done, it is common to filter the point cloud data. The algorithm used to obtain a filtered point cloud is discussed in the next section along with a discussion of the significant differences it makes for pose determination of RSOs in the section on computational results. The goal is to generate a clean, high-quality range image, and thus a refined point cloud, \vec{p}_{sensor}^{LRF} . The processed range image is the first step of in the pipeline and is essential for high accuracy pose determination.

ASC uses a shallow neural network (ASC-SNN) to effectively denoise the target data and produce a smoothed image. The ASC-SNN is similar to a masked autoencoder, and can effectively learn a denoising mapping for the target satellite. The ASC-SNN filter is trained using two channel pixel data. The first channel is a noisy range measurement and the second channel is a binary mask, where a one indicates a valid range and a zero indicates an invalid range. An augmented dataset, produced from the target satellite CAD model, is generated from noiseless range maps with multiple poses and ranges. Multiple noise profiles for each of the range maps add additional dataset images. During NN training the objective is to minimize the mean squared error (MSE), defined as the average of the squared differences between the network's predictions and the true range values. The MATLAB Deep Learning Toolbox was used to design, train, and test the ASC-SNN.

The ASC-SNN performance was compared to the MATLAB `pcdenoise` and `bilateral` filter functions using simulated target satellite data at a 150-meter range. The shallow network runs about 4 times slower than the MATLAB `pcdenoise` function, which removes outlier points based on nearest-neighbor distance thresholding. However, the model is sufficiently compact and lightweight for embedded deployment. Furthermore, NN optimizations are possible to improve the timing, using tools within the MATLAB environment.

At larger ranges, larger absolute range jitter becomes more difficult for classical algorithms to filter, and causes artifacts such as over smoothing valid edges or inadequately dealing with noise under standard parameter settings. By contrast, the ASC-SNN, trained with synthetic range and mask data from satellite CAD models learns the distance-dependent noise profile and preserves geometric detail without manual tuning, yielding significantly lower

MSE and cleaner and more spatially consistent point clouds with only a modest runtime cost. In the Computational Results section, it is demonstrated that ASC-SNN makes a significant difference in achieving high accuracy pose estimation. ASC is working on next-generation versions that add reflectance (intensity) input and increased network capacity to support more complex or multi-part geometries.

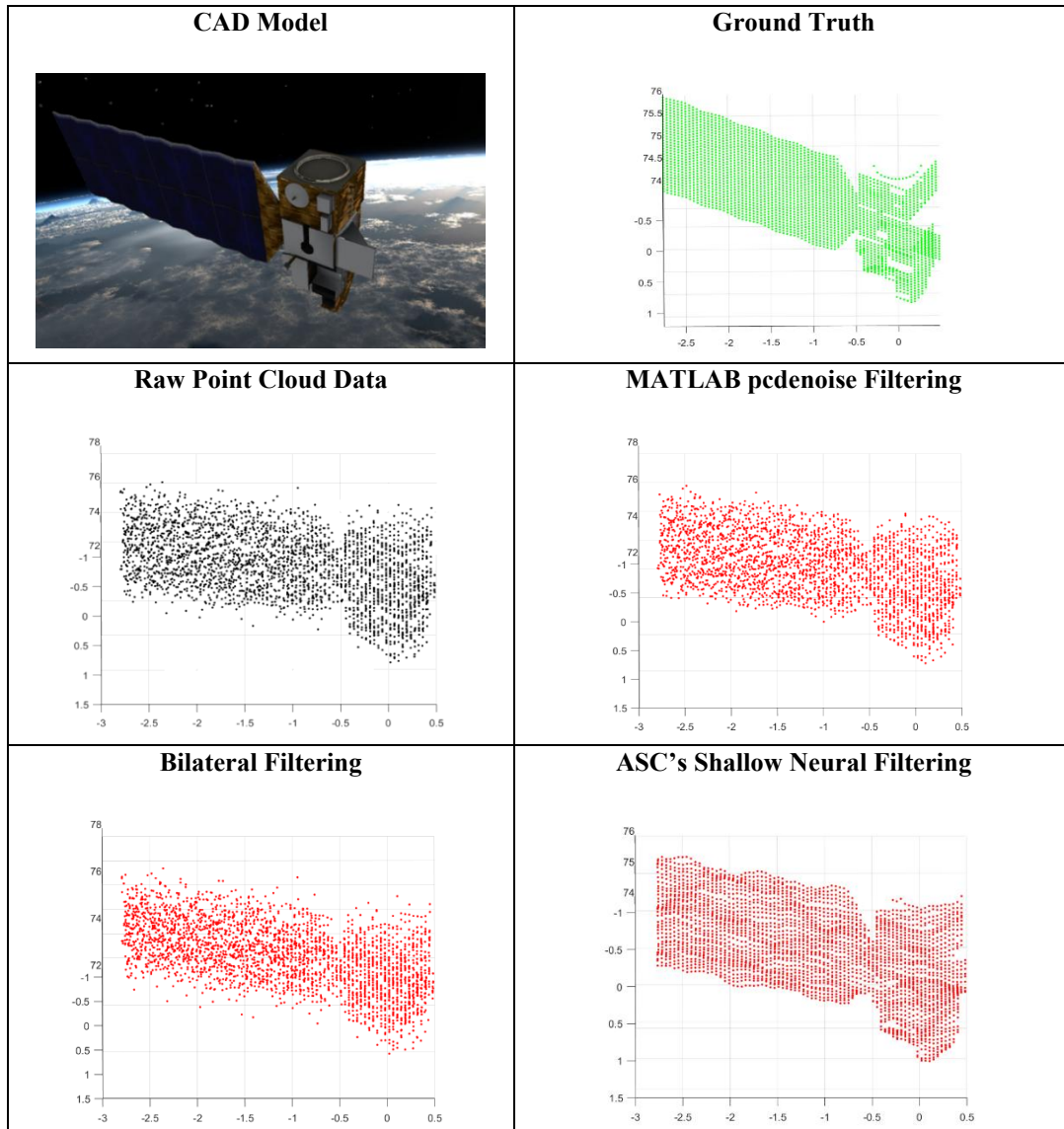


Fig. 6. Point-cloud comparisons for CAD model, ground truth, simulated raw data, and denoising results.

Among the three filtering methods shown, the ASC-SNN delivers significantly improved denoising performance with the lowest mean squared error metric as shown in Fig. 6.

3.19 Iterative Closest Point

The Iterative Closest Point (ICP) algorithm is applied after the Model Vertex Repositioning and Point Cloud Filtering steps as shown in Fig. 1. The ICP is applied to the filtered point clouds \vec{p}_{sensor}^{CF} and \vec{p}_{local}^{CF} , both expressed in the Centroid Frame. The output of this process is the transformation matrix $G^{TSBF,CARF}$, which aligns the model to the sensor data. Although ICP can be applied to any two point clouds defined in the same reference frame, the Centroid Frame was chosen to improve numerical stability during the optimization process.

The CAD model of the target satellite is defined as a set of (x, y, z) vertices in a local body-fixed reference frame, as originally defined in modeling software such as Blender or Maya. This frame is arbitrary with respect to the LiDAR's coordinate frame, and must be aligned through transformation. The live point cloud data must first be transformed into the average-centered reference frame. Concurrently, the CAD model, which is defined in its local body-fixed frame, must be transformed into the LiDAR frame, positioned in close proximity to the live data. Ray casting is then performed on this transformed CAD model to generate a model point cloud which exists in the LiDAR frame. Next, transformation matrices are applied to convert the model point cloud into the average-centered frame. When both point clouds share the same reference frame the Iterative Closest Point (ICP) algorithm can accurately compute the translation and rotation needed to align the model point cloud with the live point cloud data.

The transformation matrices required to navigate throughout the entire coordinate frame hierarchy are obtained after the ICP step. This includes the transformation matrix from the servicer's body frame to the satellite's body frame. Let $G^{TSBF,SVBF}$ denote the homogeneous transformation matrix mapping coordinates from the service vehicle body-fixed frame to the TSBF. This matrix contains both rotation and translation components. The rotation part (the top-left 3×3 submatrix) can be converted to Euler angles (yaw, pitch, roll) using standard rotation-to-Euler conversion methods. The translation vector $\vec{t}^{TSBF,SVBF}$ can be extracted from the last column of the matrix (elements of rows 1-3, column 4).

Further accuracy improvement is possible by using a Kalman Filter. However, one must be careful in applying a Kalman Filter to improve quaternion-based orientation data. Because quaternions \mathbf{q} and $-\mathbf{q}$ represent the same rotation, to maintain continuity and avoid sign flips between successive quaternions, it is important to enforce the conditions

$$\mathbf{q}_k \cdot \mathbf{q}_{k-1} \geq 0 \quad (k \text{ denotes the current timestep})$$

ensuring quaternion alignment. For a robust orientation filter, it is most effective to use either the Multiplicative Extended Kalman Filter (MEKF) [8] or the Indirect Kalman Filter [9]. These variants operate naturally on the quaternion manifold and preserve the unit-norm constraint while avoiding the singularities and covariance issues encountered in naive quaternion filtering approaches. The mathematical details are not presented here since quaternion Kalman filtering is a well-established solution in attitude estimation literature. The pipeline is fully compatible with MEKF or similar predictive state estimation methods.

4. DIGITAL TWIN SIMULATION ENVIRONMENT

The ASC Digital Twin Simulation Environment replicates the optical, geometric, and computational components of a GSFL-based satellite pose estimation system, enabling both algorithm validation and system design. It includes scenario elements such as satellite CAD models, defined coordinate frames, and dynamic behaviors assigned to CAD model actors. This section outlines the simulation of camera physics and the acquisition of simulated data through mesh models.

4.1 Range Data Simulation

The GSFL simultaneously captures 128×128 range measurements arranged in an array format with 128 rows and 128 columns and denoted by $r_{i,j}$. Range is simulated by using 16,384 unique rays originating from the aperture of the GSFL camera, described in more detail in the Ray-Casting section.

In a 3D environment, surfaces are represented as a mesh composed of triangles, each defined by three vertices: $\vec{\mathbf{p}}_1$, $\vec{\mathbf{p}}_2$, and $\vec{\mathbf{p}}_3$. Although these vertices are typically expressed in the world coordinate frame of the 3D environment, they are often transformed into the LRF frame for consistency. With the 3D environment defined, ray-casting determines the distance along each ray-template direction to the first intersected mesh triangle in the scene. It also records the mesh's normal vectors and mesh labels, which are the integers representing the material properties of the intersected triangle.

Before adding simulated range noise, the simulator passes the resulting data into a link budget model to first model the intensity, or detected photons. Once this is determined, the magnitude of the intensity determines how noisy the resulting signal will be.

4.2 Intensity Data Simulation

Radiometric analysis is applied to the range map in order to calculate the amount of light reflected onto each pixel. The radiometric equation used is:

$$E_p^{i,j} = \frac{E_{tx}}{r_{i,j}^2 \tan^2(FOV/2)} \cdot \rho(\omega_i, \omega_o) \cdot \frac{D^2}{4} \cdot IFOV^2 \cdot \eta_{opt} \eta_{atm} \cdot \cos(\alpha)$$

Where:

- $E_p^{i,j}$ is the energy reflected onto the pixel at row i and column j ,
- E_{tx} is the total transmitted laser energy (in photons),
- $r_{i,j}$ is the range to the target (in meters),
- FOV is the full field of view (in radians),
- D is the aperture diameter (in meters),
- $IFOV$ is the instantaneous field of view of the lens (in radians),
- η_{opt} is the optical system's transmission efficiency,
- η_{atm} is the atmospheric transmission efficiency,
- α is the incident angle (in radians),
- $\rho(\omega_i, \omega_o)$ is the bidirectional reflectance distribution function (BRDF),
- ω_i and ω_o are the incident and reflected light directions, respectively.

The BRDF $\rho(\omega_i, \omega_o)$ describes how light is reflected off an opaque surface and is defined as:

$$\rho(\omega_i, \omega_o) = \frac{dL_0(x, \omega_o)}{L_i(x, \omega_i) \cos(\alpha) d\omega_i}$$

Where:

- $dL_0(x, \omega_o)$ is the differential amount of radiance reflected at point x in the direction ω_o ,
- $L_i(x, \omega_i)$ is the incoming radiance at point x from direction ω_i ,
- $d\omega_i$ is the differential solid angle around ω_i ,
- $\cos(\alpha)$ is the cosine of the angle between the incident direction ω_i and the surface normal at point x .

Because accurately modelling $\rho(\omega_i, \omega_o)$ is often complex, a simplified version is typically used. The simulated GSFL system adopts a modified Phong reflectance model (Phong, 1975), expressed as:

$$\rho_{phong} = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n(\varphi)$$

Where:

- k_d and k_s , are the diffuse and specular reflectance coefficients, respectively.
- n is the shininess parameter,
- $\varphi = 2\alpha$ is the angle between the reflected and incident light directions as shown in Fig. 7.

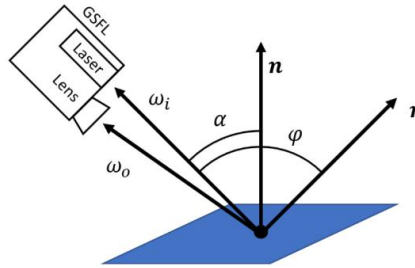


Fig. 7. Modified Phong BRDF diagram

By inserting known GSFL system parameters and reflectance values, the reflected energy at each pixel can be computed and compared to the minimum threshold required to trigger the GSFL detectors. The resulting intensity values are scaled between 0 to 4095, with 4095 corresponding to saturation of the GSFL sensor. Refer to Fig. 8 for an illustration of the modified Phong based intensity model and to reference [3] for a more comprehensive discussion of ASC’s digital-twin simulator at the photon level.

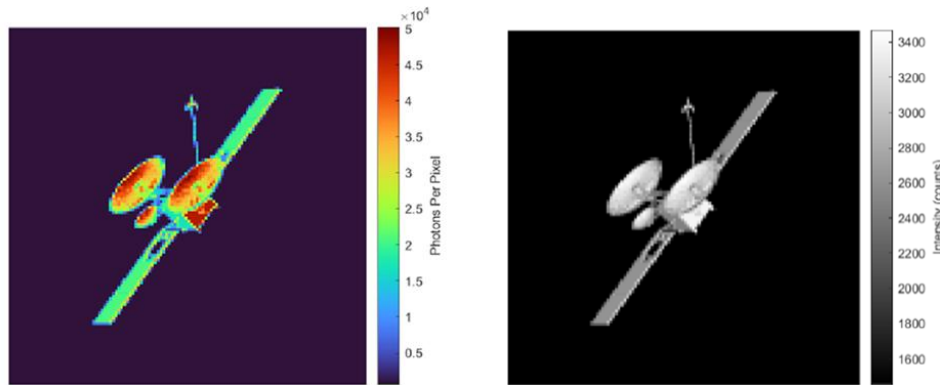


Fig. 8. Intensity simulation of a Tracking and Data Relay Satellite (TDRS).
Left: photon image of the TDRS model. Right: Intensity image of the TDRS model.

4.3 3D Scene Simulation and Data Processing

The simulation environment is built around CAD models, which represent the primary actors which are a Servicer Satellite and a Target Satellite. In this study, it is assumed that a CAD model of the target satellite is available. For demonstration purposes a NASA Aura satellite was used as the target satellite as shown in Fig. 9. For embedded system applications, it is recommended that this CAD model includes only the outer (visible) surfaces, omitting fine details such as screws or bolts to improve processing efficiency.



Fig. 9. CAD model of the NASA Aura satellite [5] rendered in the Godot 4.4 game engine [12]

Key features of the Digital Twin simulator include the capability to test pose estimation algorithms in a simulated environment, support for testing with live input data, modular unit testing of subsystems, and 3D visualization of system behavior during simulation. The Digital Twin simulator is implemented in MATLAB and includes a MATLAB app for data visualization. The Godot game engine was used to render the 3D scene. The Godot project was configured with both of the CAD models, a controllable servicer-mounted gimbal, and time-dependent trajectories for the target CAD model. Ray casting was implemented using the predefined ray template. During simulation, ray casting results, including pixel hit locations, ranges, surface normal vectors, and mesh labels, were recorded in JSON files. The MATLAB app processes these JSON files to generate simulated range and intensity measurements, apply a realistic noise model, and display both the data and relevant performance metrics.

5. COMPUTATIONAL RESULTS

All experiments were conducted by configuring the target satellite CAD model with a randomly oriented angular momentum vector with angular velocity of $\omega = 0.628 \text{ rad/s}$ ($2\pi/10 \text{ s}$). The GSFL camera was simulated to produce range and intensity data at a frame rate of 10Hz. The objective of these experiments is to verify that pose estimation is feasible for uncooperative objects at a distance of up to 150 meters (potentially farther, depending on the object's size). For error analysis, 100 frames of GSFL data were collected and processed through the data pipeline to generate 100 quaternion time-series measurements. The noise level was specified based on ASC GSFL camera's range noise estimates at distances of 100 to 150 meters. During time-series pose estimation using quaternions, the predicted and ground truth values may occasionally differ in sign (i.e., \mathbf{q} vs $-\mathbf{q}$). Since both represent the same rotation, this sign flip does not affect the actual orientation. For consistency in comparison, the quaternion time series was preprocessed to ensure that the predicted and ground truth quaternions have matching signs.

5.1 Experiment 1 – 100 meter range

In Experiment 1 a 3° FoV was used with the target placed 100 meters from the GSFL camera. The objective was to determine the extent of accuracy enhancement provided by the ASC-SNN versus raw unprocessed GSFL sensor data. The metric used to evaluate the results was the MSE of the 100 step time series quaternion data. The ASC-SNN produced superior results with an MSE of 8.5×10^{-5} , compared to raw unfiltered data with an MSE of 9.15×10^{-4} . This underscores the benefit of using advanced algorithms to maintain accurate pose predictions at extended ranges. Unfiltered results are plotted in Fig. 10, and the ASC-SNN based filtering is plotted in Fig. 11.

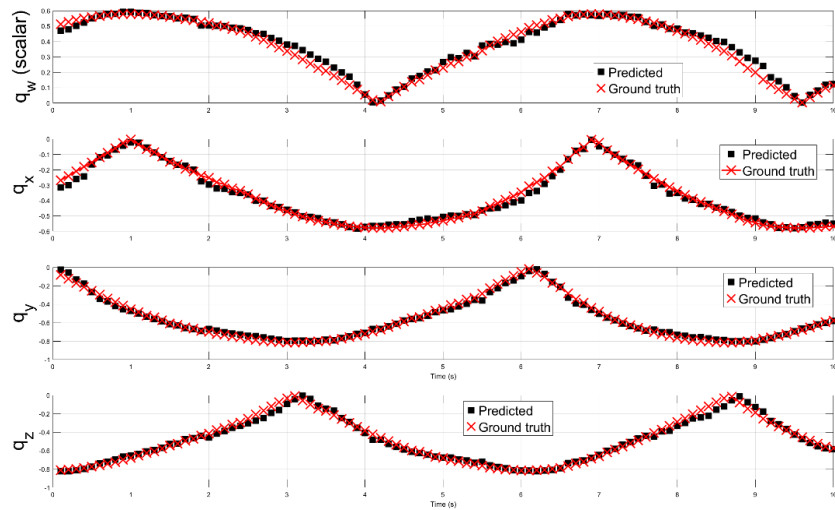


Fig. 10. Time-series of quaternion components \mathbf{q}_w , \mathbf{q}_x , \mathbf{q}_y , \mathbf{q}_z , for an object located at a 100 m distance, based on unfiltered (raw) sensor data. Black: predicted orientation; Red: ground truth orientation

The divergence across all components implies that raw sensor data alone is not enough for accurate pose estimation. Filtered results are shown in Fig. 10, with the predicted data using ASC-SNN. The near-perfect overlap across all components confirms accurate long-range attitude estimation by the model.

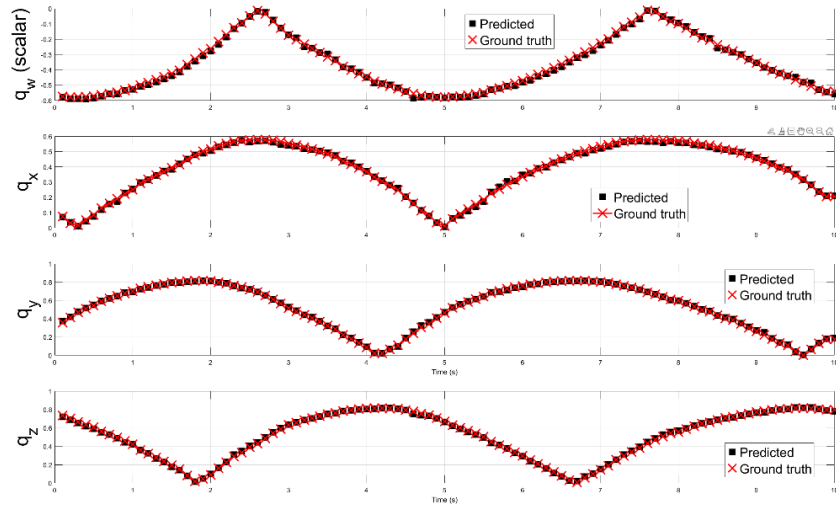


Fig. 11. Time-series of quaternion components q_w , q_x , q_y , q_z , for an object located at a 100 m distance. Black: predicted orientation; Red: ground truth orientation

5.2 Experiment 2 – 150 meter range

In Experiment 2 the target was placed at 150 meters from the camera. Because of the greater distance and the resulting $\mathcal{O}(1/r^2)$ decrease in the number of pixels, the predicted results show increased noise compared to the simulation at 100m. As expected, by applying ASC-SNN to filter the data, there is a significant improvement in pose estimation accuracy. The ASC-SNN produced superior results with an MSE of 1.06×10^{-4} , compared to raw unfiltered data with an MSE of 1.80×10^{-3} . Filtered results are plotted in Fig. 12 and ASC-SNN based filtering is plotted in Fig. 13.

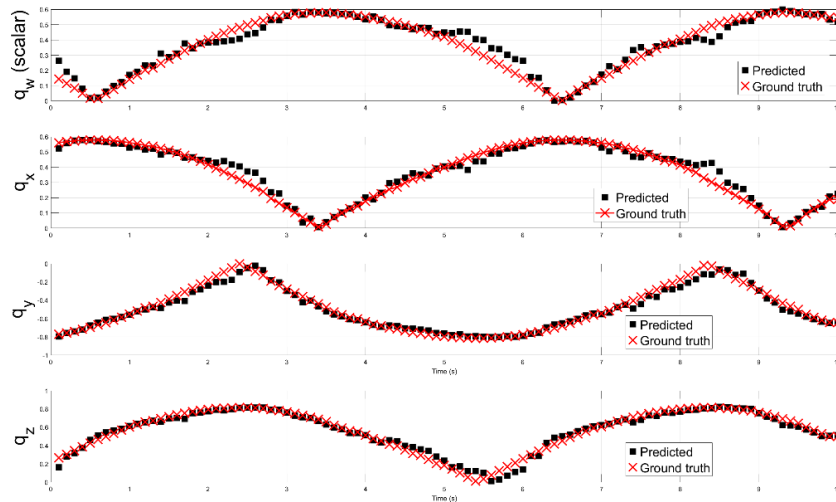


Fig. 12. Time-series of quaternion components q_w , q_x , q_y , q_z , for an object located at a 150 m distance, based on unfiltered (raw) sensor data. Black: predicted orientation; Red: ground truth orientation.

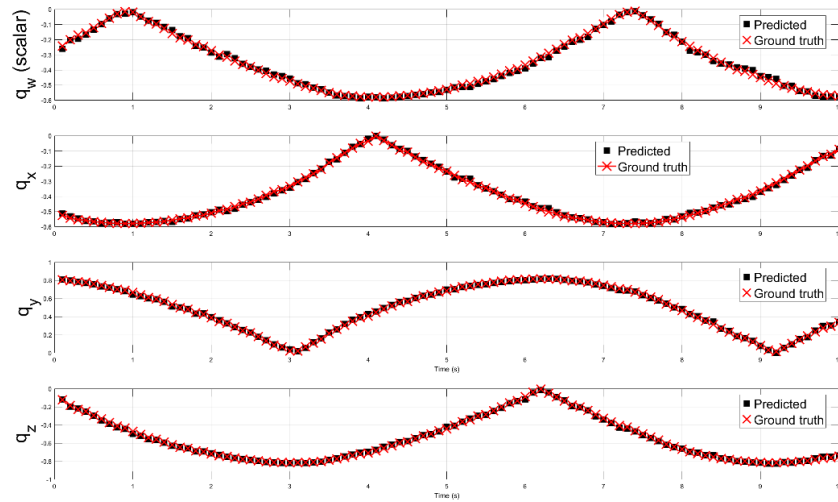


Fig. 13. Time-series of quaternion components q_w , q_x , q_y , q_z , for an object located at a 150 m distance. The predicted data utilized ASC-SNN. Black: predicted orientation; Red: ground truth orientation.

6. CONCLUSION AND FUTURE WORK

We have developed a real-time pipeline using Flash LiDAR that leverages dense range data to determine the pose of Resident Space Objects (RSOs) at mid-range distances (approximately 10 to 150 m). As shown in Equation (1), pose estimation accuracy depends on factors like the object's size, geometry, distance, and the number of range measurements. Since both high measurement density and high frame rate are critical for accurate and timely pose estimation, Flash LiDAR is the optimal choice. It captures the entire scene in a single laser pulse, producing dense point clouds while avoiding motion-induced distortion making it uniquely suited for dynamic tracking scenarios.

Our neural network denoises the raw point cloud at high frame rate, significantly reducing range noise. As a result, pose estimation error measured in the mean square sense improves by approximately one order of magnitude compared to using unfiltered data. Coordinate transformations are highly efficient, requiring around 10^{-4} seconds on a Jetson Orin NX 8GB. To further expedite pose alignment, we employ a ray casting-based strategy that generates a model point cloud positioned near the live sensor data for rapid registration. This method benefits from simplified CAD models composed of fewer triangles, boosting speed and convergence of the ICP algorithm. Although effective, highly symmetric RSOs still pose challenges, as they can lead to pose ambiguity. Looking forward, we plan to explore the integration of GSFL intensity data and enhanced neural network techniques, such as RSO-type recognition to select the appropriate CAD model and a robust denoising neural network capable of handling multiple geometries to improve pose estimation accuracy and resilience.

7. REFERENCES

- [1] Wang, J., Zhang, G., and You, Z. Design rules for dense and rapid Lissajous scanning, *Microsystems & Nanoengineering* (2020) 6:101.
- [2] Fuller, L., Carl, A., Spagnolia, J., Short, B., and Dahlin, M. "Photon counting linear mode global shutter flash LIDAR for improved range performance." *Proceedings of SPIE*, 12110, Laser Radar Technology and Applications XXVII, 1211005, 3 June 2022. <https://doi.org/10.1117/12.2619047>
- [3] Fuller, L., Karl, R. Jr., Anderson, B., and Lee-Roller, M. "Development of a versatile LiDAR point cloud simulation testbed for advanced RSO algorithms." *Proceedings of The Advanced Maui Optical and Space Surveillance Technologies (AMOS) Conference*, September 2022. Available at: <https://amostech.com/2022-technical-papers/>

- [4] Möller, T. T., and Trumbore, B. "Fast, Minimum Storage Ray-Triangle Intersection." *Journal of Graphics Tools*, 2(1):21–28, 1997. <https://doi.org/10.1080/10867651.1997.10487468>
- [5] NASA. "3D Models." *NASA 3D Resources*, 15 August 2023. Available at: <https://nasa3d.arc.nasa.gov/models>
- [6] Cortinhal, T., Tzelepis, G., and Aksoy, E. E. "SalsaNext: Fast, Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving." *Advances in Visual Computing*, 207–222, 2020. https://doi.org/10.1007/978-3-030-64559-5_16
- [7] ISO. *Flight dynamics — Concepts, quantities and symbols — Part 5: Quantities used in measurements*, ISO 1151-5:1987, 1987.
- [8] Lefferts, E. J., Markley, F. L., and Shuster, M. D. "Kalman Filtering for Spacecraft Attitude Estimation." *Journal of Guidance, Control, and Dynamics*, 5(5):417–429, 1982. <https://doi.org/10.2514/3.56190>
- [9] Trawny, N., and Roumeliotis, S. I. "Indirect Kalman Filter for 3D Attitude Estimation." *Technical Report 2005-002*, Department of Computer Science & Engineering, University of Minnesota, March 2005. Available at: <https://mars.cs.umn.edu/tr/reports/Trawny05b.pdf>
- [10] Rusu, R. B., Blodow, N., and Beetz, M. "Fast Point Feature Histograms (FPFH) for 3D Registration." *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12–17, 2009, pp. 3212–3217. IEEE. DOI: 10.1109/ROBOT.2009.5152473
- [11] The MathWorks, Inc., MATLAB, R2024b Update 2 (v24.2.0.2773142), Natick, MA, 2024.
- [12] *Godot Engine*, version 4.4, The Godot Engine contributors, 2025. Available at: <https://godotengine.org/>