

SISIFOS: Specialized Illumination Simulator For Orbiting Spacecraft

Iason Georgios Velentzas^{*}, Juan-Diego Florez-Castillo[†], Noah Bruckner[‡], Mehregan Dor[§], and Panagiotis Tsiotras[¶]
Georgia Institute of Technology, Atlanta, Georgia, 30332

Despite notable advances in computer vision and deep learning, the full potential of these fields remains largely unexplored in the context of spacecraft rendezvous and proximity operations. One of the main reasons is the scarcity of annotated visual data. Although synthetic datasets have emerged as a practical alternative to the complex and costly real-world data collection process, they often lack flexibility, realism, or the capacity to model diverse scenarios in practice. To address these challenges, we introduce SISIFOS (Specialized Illumination Simulator for Orbiting Spacecraft), open-source software for realistic image generation of spacecraft in orbit. SISIFOS enables the mass generation of high-fidelity images under realistic space-environment conditions and is integrated with a comprehensive trajectory generation module, allowing users to simulate complex rendezvous and proximity operations under controlled, yet diverse configurations. The images are also accompanied by an abundance of ground-truth maps, such as depth and surface normals, emulating a complete sensor suite and enabling the use of multimodal inputs for downstream (e.g. relative navigation) tasks. The SISIFOS platform represents a powerful tool for evaluating perception and planning algorithms in orbital scenarios and contributes significantly to the development of robust autonomous systems in support of future space missions. The code and database can be found through <https://github.com/JasonVel6/SISIFOS>.

I. Introduction

IN recent years, advancements in machine learning and computer vision have revolutionized autonomous spacecraft navigation, particularly for Rendezvous and Proximity Operations (RPO). RPO capabilities are essential for on-orbit servicing, in-space assembly and manufacturing, formation flying, and space debris removal—all critical to maintaining the sustainability of the space environment [1], [2]. Vision-based navigation has emerged as a key technology for next generation space missions, particularly when encountering non-cooperative targets [3]. Unlike radar, lidar, or GPS, vision sensors offer passive operation with minimal power, weight, and volume requirements. Most significantly, they provide rich, compact information that supports precise autonomous relative navigation across distances, ranging from hundreds of kilometers to virtually zero separation [4].

Despite these recent advances, the successful deployment of robust vision algorithms in orbit remains a challenge. Visual navigation systems require large, high-quality datasets for training and evaluation, yet real-world spacecraft imagery is scarce due to the prohibitive costs and logistical challenges of space missions. Furthermore, image acquisition in space is constrained by limited on-board storage, transmission capabilities, and the precise spacecraft control necessary to obtain accurate and representative images. While synthetic datasets offer a scalable alternative, existing simulators present notable limitations—either lacking realistic visual characteristics such as complex lighting and environmental factors or lacking the diversity in environment conditions that enables networks to learn generalizable features. To the best of our knowledge, there does not exist a unified simulator framework to integrate a realistic simulation tool with actual orbital conditions.

To address these challenges, we introduce SISIFOS (Specialized Illumination Simulator For Orbiting Spacecraft) ^{*},

^{*}Ph.D. Student, Institute for Robotics and Intelligent Machines, School of Aerospace Engineering, iason.velentzas@gatech.edu

[†]Ph.D. Student, Institute for Robotics and Intelligent Machines, School of Mechanical Engineering, jdflorez@gatech.edu

[‡]Undergraduate Student, School of Electrical and Computer Engineering, nbruckner3@gatech.edu

[§]Postdoctoral Researcher, School of Aerospace Engineering, mehregan.dor@gatech.edu

[¶]David & Andrew Lewis Chair, Professor, Institute for Robotics and Intelligent Machines, School of Aerospace Engineering, tsiotras@gatech.edu

^{*}The naming SISIFOS is a nod to the myth of Sisyphus, re-imagined not as a symbol of futility but of perseverance in the pursuit of progress. As Albert Camus wrote, "The struggle itself towards the heights is enough to fill a man's heart. One must imagine Sisyphus happy." In the same spirit, SISIFOS embodies the ongoing effort to push the boundaries of space exploration, empowering the research community to climb ever closer to the hill of robust autonomous spacecraft navigation.

a simulation framework for generating realistic and flexible RPO datasets. Leveraging the render accuracy capabilities of Blender [5], SISIFOS delivers high-fidelity visuals with physically accurate lighting and materials. Unlike previous works, our simulator renders Earth and star fields as fully simulated entities, incorporating Earth’s albedo and directionally accurate solar and celestial lighting, thereby providing significantly more realistic environmental conditions than simulators using static background images. Moreover, we provide extensive configurability of target spacecraft, camera parameters, and lighting conditions to emulate realistic onboard images of inspection agents. SISIFOS includes a comprehensive trajectory generation module capable of simulating single- and multi-agent mission scenarios with onboard sensor measurements, which can then be deployed to the real-time image generation pipeline. These capabilities make SISIFOS a powerful and flexible validation platform for state-of-the-art space-bound navigation and planning algorithms.

The contributions of this work can be summarized as follows. First, we present an open-source simulation framework specifically designed to generate realistic spacecraft imagery under a wide range of orbital parameters and environmental factors that define the spacecraft’s state in orbit. Second, SISIFOS includes accurate physical modeling of illumination effects, along with ray-tracing accuracy and realistic sensor effects in orbit. Third, we provide a comprehensive trajectory generation module, which allows researchers to simulate various mission scenarios and obtain synchronized sensor measurements alongside visual imagery. Finally, we release an open-source dataset containing images of multiple spacecraft captured under controlled variations of illumination, geometry, and sensor parameters. The dataset includes both sequenced image trajectories and unordered views from different poses, enabling rigorous evaluation of perception and pose estimation algorithms.

II. Related works

In the context of spacecraft RPO datasets, related works can be categorized into three main areas: image datasets, trajectory datasets, and simulators. Image datasets comprise collections of spacecraft imagery depicting various lighting, motion, and background scenarios. Along with trajectory datasets, which consist of time-sequenced pose data of spacecraft, they are essential for developing and validating guidance, navigation, and control algorithms across varying mission profiles, including different approach paths, target motions, and operational constraints. These can be generated in simulators, that is, software tools that create synthetic mission data under configurable physical and environmental conditions. Their primary advantage is their flexibility—researchers can systematically vary parameters such as lighting direction, sensor characteristics, and relative motion to study effects on perception and control algorithms while generating large-scale datasets for algorithm training and testing.

A. Image Datasets

Image datasets typically consist of snapshots captured from random poses around the spacecraft and are useful when temporal sequencing is unnecessary. For instance, the URSO data set [6], which was created to enhance the training of a neural network (URSONet) for the task of estimating relative poses, contains three subsets (Dragon spacecraft and two Soyuz spacecraft variants with different complexity) with 5,000 images each. URSO simulates a 90° horizontal field of view camera and incorporates the Earth, Sun, and illumination effects such as bloom and glare. However, the dataset has notable limitations: its operational range is restricted to a relative distance of 4-10 m between camera and target, and it employs image auto-exposure, which is unrealistic for actual missions. In challenging lighting conditions that produce predominantly dark or bright images, auto-exposure distorts the imagery, preventing proper evaluation of algorithm robustness under realistic operational conditions.

To address the scarcity of machine learning datasets for spacecraft applications, the European Space Agency (ESA), in collaboration with Stanford University, released the SPEED dataset [7] and its successor, SPEED+ [8]. The original SPEED dataset contains approximately 15,000 synthetic (OpenGL-based) and 300 real images of the TANGO spacecraft, while SPEED+ expands this collection to 60,000 synthetic and 10,000 real images depicting varied lighting conditions. Both datasets share the same limitation: Earth backgrounds are statically composited, with the Earth occupying 50% of the image area while neglecting complex lighting interactions such as Earth’s albedo. The SPEED dataset’s real images are rendered in the Testbed for Rendezvous and Optical Navigation (TRON) facility using light sources to emulate collimated light from the Sun and the Earth’s albedo. To create the real images for SPEED+, researchers used an improved setup featuring lightboxes with diffuser plates around the target for albedo simulation and Sun lamps for direct high-intensity homogeneous light to simulate sunlight; however, the two different light emulators were never used simultaneously, providing either direct Sun illumination, or albedo reflections.

The SPE3R [9] dataset is currently the most comprehensive open-source synthetic dataset for spacecraft imagery,

as it incorporates 64 spacecraft models with 1,000 low-resolution (256x256) images for each model. The simulation environment incorporates Earth and Sun models that accurately represent space illumination conditions. However, the dataset only provides labels for the relative poses between the camera and the target and fails to provide information about the Earth and Sun positions within the scene. Moreover, while the images’ low resolution might be suitable for rapidly abstracting 3D shapes from 2D images [10], it can be prohibitive for identifying crucial details arising from model textures and illumination effects.

B. Trajectory Datasets

Trajectory datasets provide image sequences that capture spatiotemporal information, enabling machine learning algorithms to learn from temporal data and understand motion patterns and dynamic changes over time. These datasets are vital for evaluating algorithm performance in realistic, time-evolving scenarios, allowing researchers to assess how their models perform under actual mission conditions where relative motion and changing lighting create complex visual challenges.

The SPARK dataset [11] consists of 150,000 synthetic images from ten different spacecraft models and five distinct debris models. Structured in two streams—one with randomized poses and another with trajectory sequences—each image is accompanied by the corresponding depth and segmentation mask. The dataset achieves photorealism by simulating the Earth and the Milky Way galaxy while incorporating realistic camera effects such as glare and blooming. However, the trajectory sequences contain unrealistic shadowing effects with constant illumination conditions, and the dataset is restricted to a short range of relative distances between camera and target.

The SwissCube dataset [12] contains 500 trajectories with 100 frames each, created using various velocities, distances, and angles from the target spacecraft. While the dataset accurately renders 3D models of the SwissCube, the Earth, and the Sun, it focuses exclusively on close-proximity operations ranging from one to ten times the cubesat diameter. An additional limitation of the dataset is its background implementation—images are enhanced with a backdrop of galaxies, nebulae, and star clusters—which might be seen as a background augmentation or randomization but actually substantially diminishes the photorealism of the images.

The CubesatCDT [13] dataset consists of simulated and lab-generated images of a generic 1U CubeSat model, comprising 50 synthetic trajectories in Unity, 15 synthetic trajectories in Blender, and 21 lab-generated trajectories. While the dataset focuses on close-proximity operation scenarios with varying illumination conditions, it provides no relevant annotation data besides the relative pose between camera and target. A significant limitation of this dataset stems from its primary objective: leveraging the symmetry of a 1U CubeSat for machine learning applications, which inherently limits the generalizability to more complex models.

The SPEED-UE-CUBE dataset [14] is a commercially available dataset consisting of 30,000 synthetic images captured around a 3U CubeSat and 1,200 images from a simulated trajectory. SPEED-UE-CUBE uses the same renderer as that in [9], which accurately represents scene illumination. However, despite the length of the provided trajectory, it still only represents a single trajectory with predefined distance to the target and fixed lighting conditions, limiting its utility for training across diverse operational scenarios.

The recent SHIRT dataset [15] is an extension of the SPEED+ dataset, providing two lab-generated sequences of images of a mock-up TANGO spacecraft, each accompanied by corresponding synthetic grayscale images created using the SPEED+ renderer. While the provided sequences represent different types of rendezvous scenarios in an attempt to assess the domain gap in pose tracking scenarios, it inherits the same limitations as the SPEED+ dataset.

C. Simulators

Simulators represent an essential tool for spacecraft vision applications, as they enable the generation of customized image and trajectory datasets. Their inherent flexibility allows researchers to control mission and environment variables across a wide range of scenarios. By facilitating systematic testing across various illumination conditions and mission configurations, simulators provide a comprehensive platform for testing algorithm robustness and performance, which are critical for developing adaptable and resilient vision-based navigation systems for space.

Currently, SPIN [16] is the only available open-source simulator for spacecraft imagery. It offers a variety of options for generating synthetic images through configurable camera parameters, environment variables, and material properties. Users provide a 3D model and the relative camera poses for the images, and the simulator outputs images along with the corresponding annotation and keypoint labels. While SPIN incorporates a lighting environment consisting of a directional light and spotlights emulating the Sun and albedo effects, it employs a simplistic approach where the Earth background is overlaid in the image—similar to the SPEED dataset—ignoring the intricacies of the illumination coming

from the Earth’s albedo. This creates significant problems simulating trajectories where continuity of Earth’s appearance in the background is important. Additionally, the currently available pre-release version lacks full control of the light sources in the scene, offering only randomized lighting directions.

III. SISIFOS Pipeline

SISIFOS simulates a spacecraft RPO scenario, where a Chaser spacecraft captures images while following at inspection orbit around a Target spacecraft. The simulator extends the basic RPO scenario by incorporating the Sun and the Earth as physical actors in the scene and the stars as an HDRI background, providing a comprehensive simulation environment. Leveraging Blender’s physically-based rendering, accurate geometry, and flexible lighting, SISIFOS generates diverse, photorealistic scenes with precise labels, which are essential for supervised learning and benchmarking. The rendering engines allow Blender to model realistic optical effects, such as soft or crisp shadows, specularities, and motion blur, which are essential for training and evaluating perception systems under challenging visual conditions. Moreover, Blender’s Python API enables scene manipulation in a programmatic way, enabling large-scale dataset generation with precise ground-truth annotations and flexibility in terms of the simulator output.

SISIFOS also features a trajectory generation module that enables user-designed relative inspection trajectories using a classical two-body propagator with Earth as the central body, but it can easily be adapted to support more sophisticated orbital motion models, if needed. By enforcing orbital energy constraints, all trajectories remain closed and periodic. Furthermore, the system generates virtual sensor data that emulates star tracker and IMU (gyroscope and accelerometer) measurements, providing realistic navigation input for algorithm validation.

A. Simulation Environment

The *Sun Actor* is modeled as a directional light source positioned at an infinite distance from the Target, mimicking the parallel light rays emitted by the Sun. The user is allowed to specify the azimuth and elevation angles relative to the Inertial frame, which fully defines the sunlight direction.

The *Earth Actor* is represented as a sphere textured using the 22K high-resolution Earth map from NASA’s Blue Marble collection [17]. To ensure stability and visually accurate rendering, both the Earth model and its relative position with respect to the other actors in the scene are uniformly scaled down while preserving realism in illumination and eclipse events. Although the actual Earth–spacecraft size and distance ratios span several orders of magnitude, reproducing these scales directly inside a rendering engine results in texture precision limitations, inconsistent shadow rays, and degraded shading quality. This rescaling does not affect the physical validity of the trajectories: the camera–target relative trajectory remains unchanged, ensuring that pose estimation algorithms are trained and tested on realistic geometric configurations. The scaled Earth thus functions as a stable albedo radiance source and occluder, while maintaining texture detail in synthetic imagery generation. This is a critical difference with pipelines that emulate the Earth as a stationary background since they do not incorporate any notion of scale and do not allow for effects of partially illuminated targets, as the Earth enters the field of view.

The Earth is also accompanied with an atmosphere and a *cloud layer* that appear as spheres slightly larger than the Earth and simulate image effects that appear on real images. The *star field* is rendered as an HDRI background textured with an 8K high-resolution image of the stars and the Milky Way galaxy.

The *Chaser Actor* is implemented as a pinhole camera actor that allows animation of subsequent frames and realistic motion blur effects. The *Target Actor* is represented as a static mesh actor. We offer a variety of preloaded spacecraft models from the ESA Science Satellite Fleet 2.0.0 project [18], and provide *Autodesk fbx* model loading functionality for custom spacecraft models. The scales and sizes of the targets are modified to match the unit sizing of the scene. In some cases, the units of the target model imposes constraints on the allowable rescaling size that can be simulated without breaking the texture patterns. Thus, using a custom target spacecraft might require a re-scaling of the overall scene, depending on the target’s units.

B. Ground Truth Annotations

For ground-truth annotation, we integrate the annotator of [19], an open-source Blender add-on that enables automated extraction of high-quality labels directly from the rendering pipeline. The tool provides pixel-accurate annotations for depth, surface normals, optical flow, segmentation masks, and instance-level object information, all computed using Blender’s internal render passes to guarantee perfect alignment with the rendered RGB images. The add-on is fully scriptable through Blender’s Python API, supporting deterministic labeling, consistent naming

conventions, and reproducible annotation settings for large-scale dataset generation. Scripting provides flexibility in the annotations, for example, each mesh of the total target can have a different segmentation ID, or we can group meshes automatically together based on their location. Thus, with little manual effort, we can distinguish higher-level objects than meshes, such as panels, antennas, and the body of the spacecraft.

C. Post-Processing Effects

Realistic environment modeling is essential for simulating the harsh and complex lighting conditions encountered in space, so our simulator provides a configurable set of post-processing effects to enhance photorealism and emulate camera-specific artifacts. For instance, *exposure time* can be adjusted through a compensation scalar that controls the exposure time of the image. Additional parameters include the *shutter speed*, which defines the duration (in seconds) during which the camera shutter remains open for image acquisition and, in combination with the animation capabilities of Blender, can emulate accurate motion blur. Users can also modify the intensity of: *blooming intensity*, which controls the simulated glow around intense light sources or highly reflective surfaces; *vignette intensity*, which emulates natural lens vignetting, darkening the corners of the image to create a tunnel-vision effect; *chromatic aberration intensity*, which simulates wavelength-dependent distortion caused by real-world lens imperfections; *chromatic aberration start offset*, which specifies the normalized radial distance from the image center at which chromatic aberration becomes perceptible; *lens flare intensity*, which simulates light scattering and reflections within the camera lens when viewing bright light sources; and *film grain intensity*, which adds stochastic texture to mimic the graininess of traditional photographic film.

D. Trajectory Generation Module

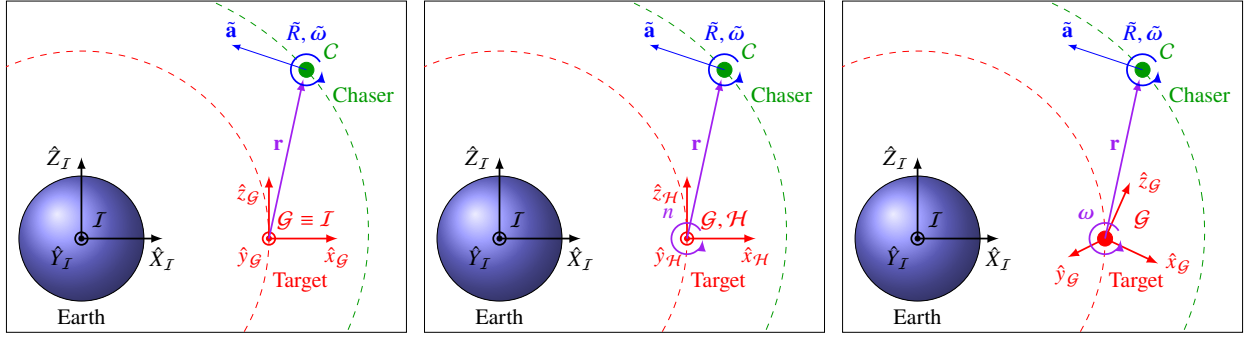


Fig. 1 Relative motion geometries for three target-attitude cases. From left to right: (a) ECI-fixed target, where the target body frame $\mathcal{G}$ is aligned with the inertial frame $\mathcal{I}$; (b) Hill-fixed target, where $\mathcal{G}$ is aligned with the Hill frame $\mathcal{H}$, which rotates with mean motion n about the orbit normal; and (c) tumbling target, where $\mathcal{G}$ rotates with angular velocity ω relative to $\mathcal{I}$. In all cases the Target (red) moves in a circular orbit around the Earth and the Chaser (green) follows a different Earth orbit, with their relative position, r , being expressed in the $\mathcal{G}$ frame. The Chaser's star-tracker and IMU (gyro and accelerometer) measurements, $\tilde{R}$, $\tilde{\omega}$, and $\tilde{a}$, are shown in blue.

The SISIFOS simulator supports custom, user-defined trajectories, for which we have defined a relative-navigation trajectory generator covering three distinct relative-motion scenarios:

- 1) *Inertially-Fixed Target*: The Target's body frame ($\mathcal{G}$) attitude is held fixed relative to the Earth-Centered Inertial (ECI) frame, $\mathcal{I}$, while both Target and Chaser (with body frame $\mathcal{C}$) follow closed Keplerian orbits around Earth. The Chaser trajectories are initialized in the Hill frame, $\mathcal{H}$, as bounded, drift-free ellipses [20]

$$\mathbf{r}_{\mathcal{C}/\mathcal{G}}^{\mathcal{H}}(t) = \begin{bmatrix} A \cos(nt + \phi) \\ -2A \sin(nt + \phi) \\ B \cos(nt + \phi_z) \end{bmatrix}, \quad (1)$$

where $n = \sqrt{\mu/a^3}$ is the mean motion (μ is the gravitational parameter and a is the semi-major axis), ϕ is the in-plane phase angle, A is the in-plane amplitude, and B is the cross-track amplitude. Because the Target satellite attitude remains constant, $R_{\mathcal{I}\mathcal{G}}(t) = R_{\mathcal{I}\mathcal{G}}(0)$, the Chaser observes the Target features from a time-varying

vantage point without any apparent rotation of the Target. This scenario isolates translational parallax from rotational effects, providing a clean baseline for algorithm testing.

- 2) *Hill-Frame-Fixed Target*: The Target's body-frame attitude co-rotates with the local Hill frame, whose axes are aligned with the radial, along-track, and cross-track directions of the Target's orbit. The Chaser is initialized with relative position and velocity offsets satisfying the drift-free constraint

$$x_0 = -\frac{\dot{y}_0}{2n}, \quad \dot{x}_0 = 0, \quad (2)$$

and propagated under the linearized Clohessy-Wiltshire equations as follows

$$\ddot{x} - 2n\dot{y} - 3n^2x = 0, \quad \ddot{y} + 2n\dot{x} = 0, \quad \ddot{z} + n^2z = 0, \quad (3)$$

where n is the mean motion of the Target's orbit. The Chaser's relative path forms a closed ellipse around the Target in the rotating frame, with the Target body frame satisfying $\omega_{\mathcal{GI}}^{\mathcal{G}} = n\hat{\mathbf{z}}_{\mathcal{H}}$. This scenario couples translational parallax with steady rotational motion, making it an ideal testbed for rendezvous, proximity operations, and formation-flying algorithms.

- 3) *Tumbling Target*: The Target undergoes torque-free rotation about its principal axes, with angular velocity $\omega_{\mathcal{GI}}^{\mathcal{G}}$ evolving according to Euler's equation:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}), \quad (4)$$

where $\mathbf{J} = \text{diag}(J_{xx}, J_{yy}, J_{zz})$ is the target's principal inertia tensor. The attitude matrix $R_{\mathcal{IG}}(t)$ is propagated using a Lie-group integrator

$$R_{\mathcal{IG}}(t + \Delta t) = R_{\mathcal{IG}}(t) \exp([\boldsymbol{\omega}^{\mathcal{G}} \Delta t]_{\times}), \quad (5)$$

where $[\cdot]_{\times}$ denotes the skew-symmetric matrix. The Chaser follows a Keplerian orbit initialized identically to the inertially-fixed case, with its position transformed into the Target's rotating body frame at each timestep.

The combination of orbital motion and continuous tumbling creates a fully non-inertial, time-varying environment, ideal for validating algorithms against spinning resident space objects.

To enable comprehensive validation, the simulator supports Monte Carlo trajectory generation. The simulator is configured with fixed inputs: the nominal orbit radius, number of chasers (N), motion scenario, and sensor parameters. For each of M trials, the following parameters are independently sampled from uniform distributions $\mathcal{U}(a, b)$ over the interval $[a, b]$:

- Orbital elements: inclination $i \sim \mathcal{U}(0, \pi)$, eccentricity $e \sim \mathcal{U}(0.005, 0.05)$
- Initial attitude: Euler angles $(\psi, \theta, \phi) \sim \mathcal{U}(0, 2\pi) \times \mathcal{U}(0, \pi) \times \mathcal{U}(0, 2\pi)$
- Chaser trajectory phase: $\phi_0 \sim \mathcal{U}(0, 2\pi)$
- Amplitude perturbations: $\epsilon \sim \mathcal{U}(-0.1, 0.1)$

Each trial produces N complete datasets, each with a ground-truth relative trajectory and synchronized sensor measurements (star tracker and IMU). Sensor measurements are corrupted by zero-mean Gaussian noise and slowly-varying biases modeled as first-order Gauss-Markov processes:

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b}_g + \boldsymbol{\eta}_g, \quad \dot{\mathbf{b}}_g = -\frac{1}{\tau_g}\mathbf{b}_g + \mathbf{w}_g, \quad (6)$$

where $\boldsymbol{\eta}_g$ is zero-mean Gaussian white noise, $\mathbf{b}_g$ is the bias, τ_g is the correlation time, and $\mathbf{w}_g$ is the driving noise. This $M \times N$ structure provides a statistically diverse collection of trajectories and sensor logs for validating GNC and SLAM algorithms.

E. System Architecture

Figure 2 illustrates the simulator's architecture and its core functionalities. SISIFOS is designed to operate asynchronously and in parallel, continuously monitoring a user-specified input folder for image metadata files. Each metadata file contains the general simulation parameters and the individual scene configuration for each image. Upon receiving a metadata file, the simulator initiates image generation and stores the resulting labeled image dataset and its associated metadata in a user-specified output directory. There are three available workflows for generating image metadata, depending on the intended use case:

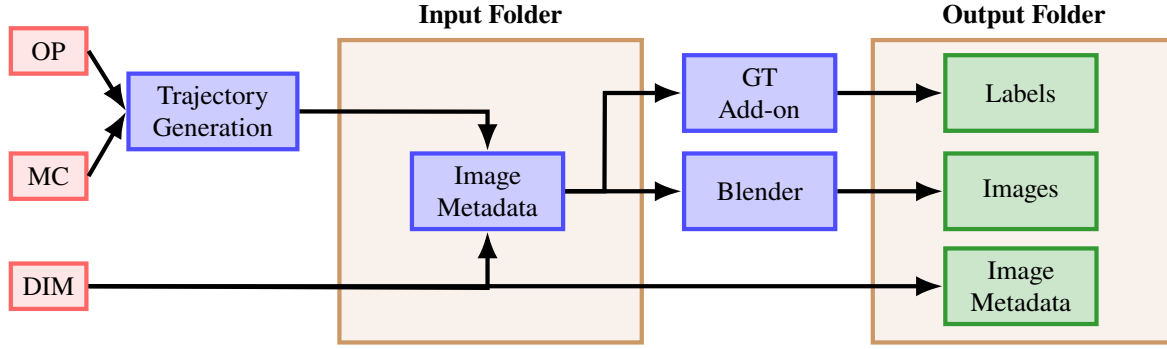


Fig. 2 System architecture showing the connected modules and interfaces. In red are the different options for the user to create the image metadata file.

- 1) Direct Image Metadata (DIM): In this mode, the user manually specifies all relevant parameters for each configuration of the images. This approach is ideal for image datasets without temporal progression. For example, the users can select random relative poses from the camera to the target within a specified operational range. In this case, the star tracker and IMU data are set to zero.
- 2) Trajectory Generation using Orbital Parameters (OP): Here, the user provides the orbital parameters and initial conditions of the desired relative trajectory, and the simulator automatically generates the corresponding image metadata based on this trajectory.
- 3) Monte Carlo Trajectory Generation (MC): This method involves stochastic sampling of orbital parameters and initial conditions. The user specifies only the general simulation parameters and the trajectory type. The trajectory generation module handles the randomized trajectory generation internally.

Additionally, the simulator allows for post-processing of metadata files. The users may modify specific metadata parameters, such as the Sun direction, after initial metadata generation using a provided read-and-write metadata script, which enables modification of all parameters. A modified metadata file can then be placed in the main simulator input folder, where it will be used by the Blender rendering module for image dataset generation.

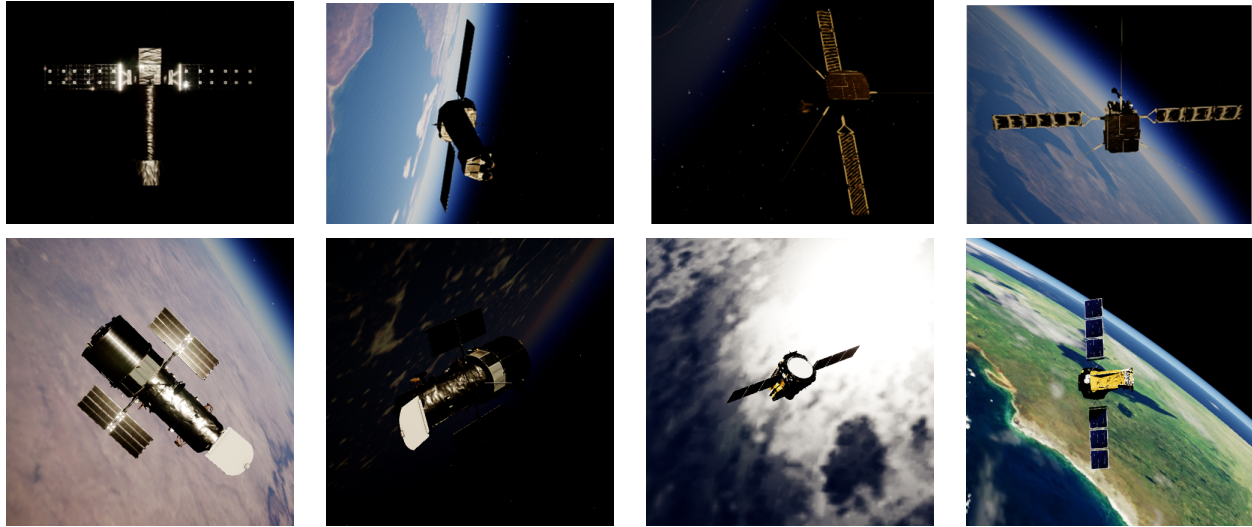


Fig. 3 Gallery of selected images of different spacecraft models under various parameter settings of the SISIFOS simulator

Figure 3 demonstrates the diverse simulation scenarios achievable through this flexible simulation pipeline. It depicts a gallery of simulated images depicting different spacecraft models under various parameter configurations.

IV. Experiments

A. Lighting consistency

In the first experiment, we assess the simulator’s capability to illuminate the scene consistently. We present four images in which the Earth, the Chaser, and the Target actors remain fixed, while only the elevation angle of the Sun actor varies smoothly. The azimuth angle of the Sun is kept constant; therefore, we expect that the resulting shadow cast by the third solar panel will move along a straight path. As shown in Figure 4, the highlighted rectangle confirms our expectation, where the shadow consistently shifts along a straight line across the body surface of the spacecraft.

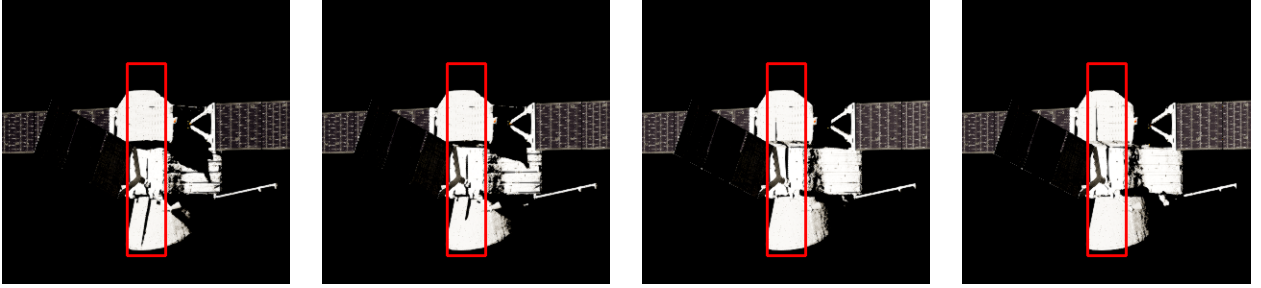


Fig. 4 Evolution of the shadow cast by the third panel, as the Sun actor’s elevation angle is varied. The red rectangles highlight the area around the line segment on which the shadow is being cast.

Moreover, the appearance of the bottom side of the shadow-casting panel also supports the simulator’s capability for consistent scene illumination. In the first two images, this thin region—almost a line—appears completely dark, whereas in the third and fourth images it is slightly illuminated. This smooth change is consistent with the evolution of the elevation angle across the images and demonstrates the simulator’s ability to model realistic lighting dynamics.

B. Earth albedo effect

To showcase the significant impact that Earth’s albedo can have on simulated images, we generated images of the same scene with and without the Earth albedo effect enabled. We designed a specific scene configuration in which the camera and the Sun are positioned on opposite sides of the Target spacecraft, resulting in a severely shadowed view of the spacecraft due to lack of direct sunlight. The Earth actor is positioned on the Sun-facing side of the spacecraft, outside the camera’s field of view. Under these conditions, we expect the spacecraft to appear almost entirely dark when the Earth albedo is disabled, with only faint specular glints remaining on the Target. In contrast, enabling Earth albedo produces a conic illumination pattern in which sunlight reflected by the Earth forms a broad, soft cone of secondary weaker light that illuminates the spacecraft. This condition does not heavily influence the image to the human eye, but can create feature contrast, which image processing algorithms can exploit to keep track of the Target.

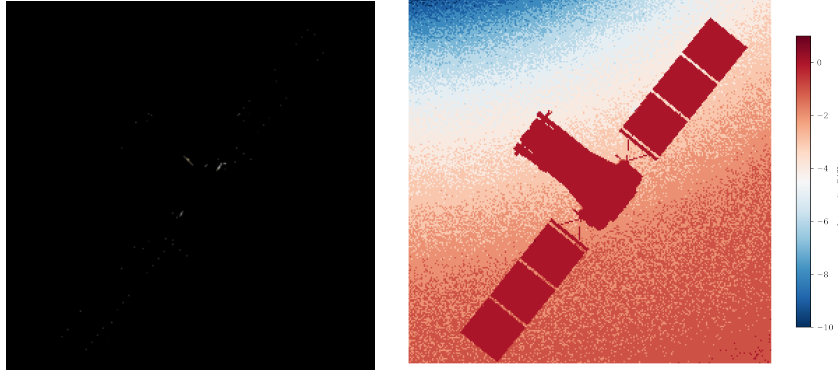


Fig. 5 Left: The original grayscale image. Right: Pixel-wise difference between the original image and the same scene configuration with albedo effect enabled.

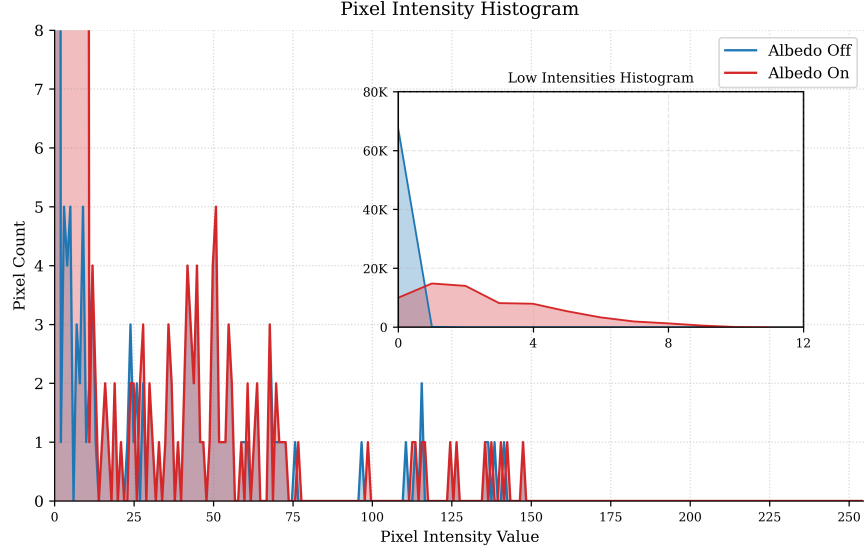


Fig. 6 Histogram of pixel intensity values of the two images shown in Figure 5. The inset plot zooms in the low intensity pixel value portion of the plot

While the pixel-wise difference might be too subtle for the human eye to perceive, the difference map in Figure 5 clearly demonstrates the variation in pixel intensity with and without albedo. Furthermore, the histogram intensities in Figure 6 confirm that, without albedo, the vast majority of image pixels are completely black. In contrast, when light from albedo is present in the scene, the enhanced pixel intensity values can help feature detection algorithms.

C. Post-processing Effects

In this experiment, we demonstrate the outcome of specific post-processing effects on rendered images to validate their intended visual effects. We first evaluate *bloom*, which simulates the glow around highly reflective surfaces. We then jointly evaluate *chromatic aberration*, which mimics color fringing due to lens dispersion, and *film grain noise*, which introduces a textured visual noise pattern. The paired effects distort the baseline image’s pixel intensities, but in distinct ways.

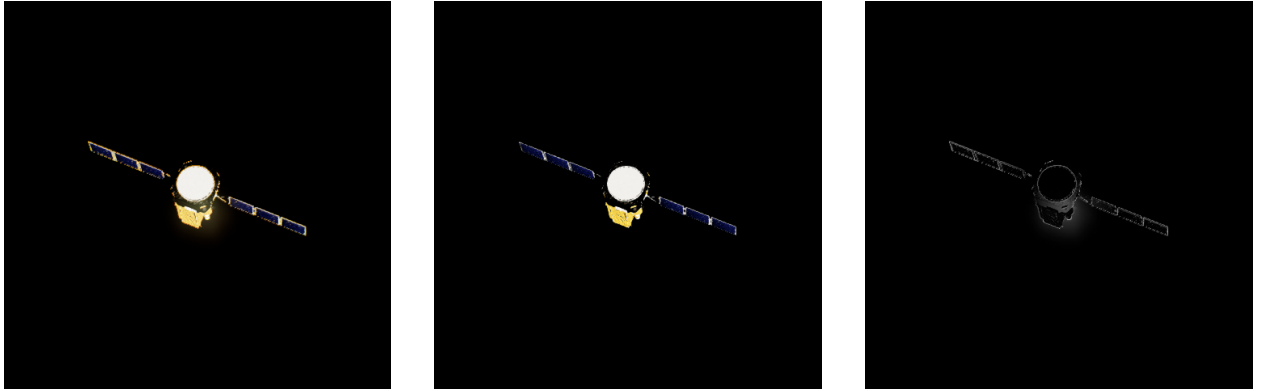


Fig. 7 Left: baseline image. Middle: same image with enhanced bloom effect. Right: pixel-wise difference between the baseline and the bloom-enhanced image.

As shown in Figures 7 and 8, the bloom effect is localized around the bright reflective surfaces of the spacecraft, producing a visually realistic glow. In contrast, Figures 9 and 10 highlight the effects of enhanced film grain noise and chromatic aberration intensity, which are distributed in all regions of the image with sufficient pixel intensity. The third column of each figure presents the pixel-wise difference between the baseline and the enhanced images.



Fig. 8 Left: baseline image with the Earth partially in the field of view. Middle: same image with enhanced bloom effect. Right: pixel-wise difference between the baseline and the bloom-enhanced image.

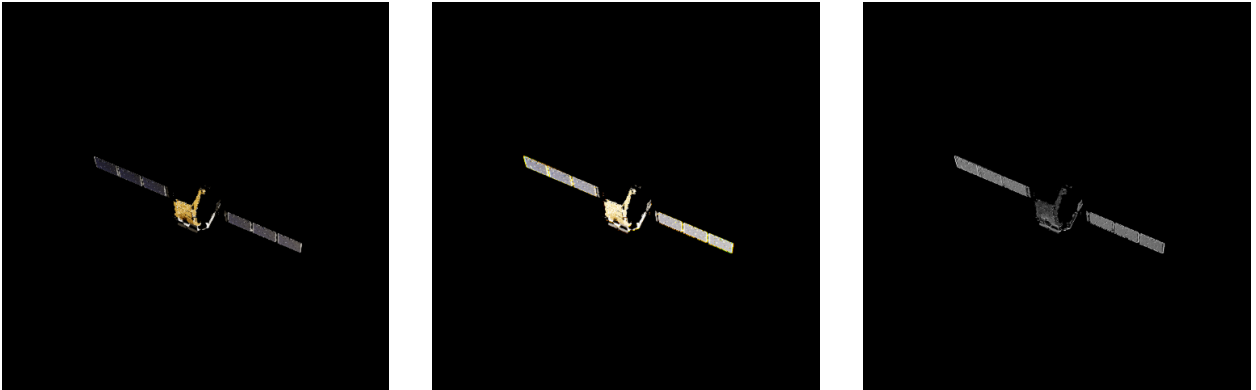


Fig. 9 Left: baseline image. Middle: same image with enhanced chromatic aberration intensity and film grain effects. Right: pixel-wise difference between the baseline and the enhanced image.



Fig. 10 Left: baseline image with the Earth partially in the field of view. Middle: same image with enhanced chromatic aberration intensity and film grain effects. Right: pixel-wise difference between the baseline and the enhanced image.

In particular, we observe that the Earth background in Figures 8, 10 remains unaffected by the bloom effect, consistent with the diffuse characteristics of the Earth texture. Furthermore, notice that in Figures 9 and 10 the intensity of the film grain introduces perceptible distortions to the appearance of the spacecraft panels, particularly through the presence of yellow speckles. While chromatic aberration introduces color shifts in the entire image, the exaggerated grain effect contributes to a textured, noisy appearance of the final image.

V. Conclusion

We have introduced SISIFOS, an open-source spacecraft simulation framework addressing critical limitations in existing datasets with its emphasis on photorealism and reconfigurability. By carefully modeling the illumination environment and integrating a flexible trajectory generation module, SISIFOS enables the creation of highly customized image sequences for training and evaluation of vision-based navigation systems. Possible next steps involve the careful curation of an open-source dataset using existing spacecraft models, which will highlight the effects of varying the simulation parameters in different scenarios and enable research on robustness against various space-intrinsic characteristics and modalities.

Funding Sources

This work was supported by the NSF awards FRR 2101250 and FRR 2530577 and by the AFOSR SURI award FA9550-23-1-0723. Iason Georgios Velentzas also acknowledges financial support from the Bodossaki Foundation.

References

- [1] Jongjittanon, N., “Contribution of In-Orbit Servicing on Space Sustainability,” M.Sc. thesis, TBS Education, Toulouse, France, 2022.
- [2] Morin, J., and Richard, G., “Tipping Points of Space Debris in Low Earth Orbit,” *The Commons Journal*, Vol. 18, No. 1, 2024, pp. 17–31.
- [3] Delpech, M., Berges, J., Djalal, S., and Christy, J., “Vision based rendezvous experiment performed during the prisma extended mission,” *23rd International Symposium on Space Flight Dynamics (ISSFD)*, Pasadena, CA, 2012.
- [4] Beierle, C., Sullivan, J., and D’Amico, S., “Design and Utilization of the Stanford Vision-Based Navigation Testbed for Spacecraft Rendezvous,” *9th International Workshop on Satellite Constellation and Formation Flying (IWSCFF)*, Boulder, CO, 2017, pp. 17–26.
- [5] Community, B. O., “Blender - a 3D modelling and rendering package,” <https://www.blender.org>, 2024. Version 4.5.3.
- [6] Proença, P. F., and Gao, Y., “Deep learning for spacecraft pose estimation from photorealistic rendering,” *IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 6007–6013.
- [7] Sharma, S., Park, T. H., and D’Amico, S., “Spacecraft Pose Estimation Dataset (SPEED),” Stanford Digital Repository, 2019. Available at: <https://purl.stanford.edu/dz692fn7184>.
- [8] Park, T. H., Märtens, M., Lecuyer, G., Izzo, D., and D’Amico, S., “Next Generation Spacecraft Pose Estimation Dataset (SPEED+),” Stanford Digital Repository, 2021. Available at: <https://purl.stanford.edu/wv398fc4383>.
- [9] Park, T. H., and D’Amico, S., “SPE3R: Synthetic dataset for satellite pose estimation and 3D reconstruction,” Stanford Digital Repository, 2024. Available at: <https://purl.stanford.edu/pk719hm4806>.
- [10] Park, T. H., and D’Amico, S., “Rapid abstraction of spacecraft 3D structure from single 2D image,” *AIAA Scitech Forum*, Orlando, FL, 2024.
- [11] Musallam, M. A., Gaudillière, V., Ghorbel, E., Ismaeil, K. A., Perez, M. D., Poucet, M., and Aouada, D., “Spacecraft recognition leveraging knowledge of space environment: simulator, dataset, competition design and analysis,” *IEEE International Conference on Image Processing Challenges (ICIPC)*, Anchorage, Alaska, 2021, pp. 11–15.
- [12] Hu, Y., Speierer, S., Jakob, W., Fua, P., and Salzmann, M., “Wide-depth-range 6D object pose estimation in space,” *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, online, 2021, pp. 15865–15874.

- [13] Musallam, M. A., Rathinam, A., Gaudillière, V., del Castillo, M. O., and Aouada, D., “CubeSat-CDT: A cross-domain dataset for 6-DoF trajectory estimation of a symmetric spacecraft,” *ECCV Workshop*, Tel Aviv, Israel, 2022.
- [14] Park, T. H., Ahmed, Z., Bhattacharjee, A., Fazel-Rezai, R., Graves, R., Saarela, O., Teramoto, R., Vemulapalli, K., and D’Amico, S., “Spacecraft pose estimation dataset of a 3U CubeSat using Unreal Engine (SPEED-UE-Cube),” Stanford Digital Repository, 2024.
- [15] Park, T. H., and D’Amico, S., “SHIRT: Satellite Hardware-In-the-loop Rendezvous Trajectories Dataset,” Stanford Digital Repository, 2022. Available at: <https://purl.stanford.edu/zq716br5462>.
- [16] Montalvo, J., Pérez-Villar, J. I. B., García-Martín, Á., Carballeira, P., and Bescós, J., “SPIN: Spacecraft Imagery for Navigation,” *arXiv preprint arXiv:2406.07500*, 2024.
- [17] National Aeronautics and Space Administration, “NASA Visible Earth Collection: Blue Marble,” <https://visibleearth.nasa.gov/collection/1484/blue-marble>, 2000. Accessed: 2025-05-22.
- [18] European Space Agency, “ESA Science Satellite Fleet,” <https://scifleet.esa.int/>, 2018. Version: 2.0.0, Accessed: 2025-05-22.
- [19] Cartucho, J., Tukra, S., Li, Y., S. Elson, D., and Giannarou, S., “VisionBlender: a tool to efficiently generate computer vision datasets for robotic surgery,” *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 2020, pp. 1–8.
- [20] Clohessy, W., and Wiltshire, R., “Terminal guidance system for satellite rendezvous,” *Journal of the aerospace sciences*, Vol. 27, No. 9, 1960, pp. 653–658.