doi: 10.82124/CEAS-GNC-2026-017 CEAS-GNC-2026-017

Event-Only Framework for Simultaneous Localization and Mapping in space using Gaussian Splatting

Piercarlo Fontana M.Sc. Student in Space Engineering, Politecnico di Milano, Milan, Italy.
piercarlo.fontana@mail.polimi.it

Annachiara Ippolito PhD Student in Aerospace Engineering, Politecnico di Milano, Milan, Italy.
annachiara.ippolito@polimi.it

Michele Maestrini Assistant Professor, Politecnico di Milano, Milan, Italy.
michele.maestrini@polimi.it

ABSTRACT

Event cameras are a novel class of vision sensors that operate fundamentally differently from traditional frame-based cameras. Instead of capturing images at fixed intervals, they asynchronously record changes in pixel intensity as individual events, triggered when a predefined brightness threshold is exceeded. This unique data acquisition paradigm enables extremely high temporal resolution, low latency, low power consumption, and an inherently high dynamic range, making event cameras particularly well-suited for use in visually and dynamically challenging environments. One of their primary applications is in Simultaneous Localization and Mapping (SLAM), where the goal is to estimate the trajectory of a moving camera while simultaneously reconstructing a 3D map of the environment. In this paper, Gaussian Splatting, which was originally developed for photorealistic 3D scene reconstruction, is integrated into an event-based SLAM pipeline with a particular focus on space scenarios such as autonomous docking. By leveraging temporally aggregated event representations and modeling the scene as a collection of volumetric Gaussian primitives, the approach aims to provide robust, accurate, and continuous 3D mapping from sparse visual input, while the system is designed for incremental operation so that it ensures consistent scene reconstruction and reliable pose estimation even under the extreme lighting and motion conditions typical of orbital proximity operations.

Keywords: Pose Estimation, SLAM, Event-Camera, Gaussian Splatting

1 Introduction

In recent years, the progressive shift toward autonomous space operations, including orbital servicing, debris removal and on-orbit assembly, has underscored the critical importance of developing reliable and precise techniques for determining the pose of a spacecraft, understood as its position and orientation in six degrees of freedom with respect to a known reference frame. This task, commonly referred to as pose estimation, plays a central role in enabling a wide range of proximity operations in space, in which the presence of highly variable illumination and the occurrence of relative dynamics between chaser and target spacecraft introduce significant sensing and navigation challenges.

Among the most challenging scenarios in this context are those involving non-cooperative targets, which are objects that neither transmit relative navigation information nor provide fiducial markers, and

which are often characterized by unknown mass properties, degraded structural integrity, and uncontrolled rotational motion. In these cases, which include derelict satellites, upper stages, or defunct spacecrafts, the pose must be estimated purely from exteroceptive measurements. As highlighted in the work of Opromolla et al. [1], such situations call for vision-based estimation techniques that are robust to partial occlusions, illumination changes, low-texture surfaces, and unpredictable dynamics, and that can operate within the constrained computational and power budgets typical of embedded spaceflight systems. These methods are typically classified into three broad categories: feature-based, direct, and learning-based approaches. Each of these has found successful applications in terrestrial domains and, in certain cases, in flight-tested space systems, but they still are in their infancy: the first, which remains the most widely adopted in practice, operates by extracting salient visual features from each image frame, but depends critically on the presence of texture-rich surfaces and sufficient lighting. Likewise, the other two methods exhibit similar problems, with the added tedious complexity of requiring large annotated datasets, which, currently, are scarce for orbital scenarios.

These limitations motivate the exploration of alternative sensing modalities, such as **event-based vision**, which promise to overcome many of the shortcomings of traditional approaches and to provide a more suitable foundation for future autonomous space operations. Unlike conventional SLAM systems, which operate on full-intensity images and often require highly accurate initializations or feature-rich environments, the proposed method relies solely on event streams, which encode per-pixel brightness changes at microsecond resolution. These events are processed incrementally to reconstruct the scene and optimize the continuous-time camera trajectory by minimizing a photometric loss computed between measured and synthesized brightness change maps. The resulting system is designed to be lightweight, scalable, and robust in dynamic and high-contrast conditions typical of space-based scenarios.

This work aims to investigate and demonstrate a novel event-based method for estimating the relative pose of a spacecraft during close-proximity operations, based on the work of Huang et al. [2]. The core contribution lies in the development and assessment of a fully event-driven pipeline that leverages the recently introduced 3D Gaussian Splatting framework to jointly estimate both the camera trajectory and a dense, differentiable 3D representation of the scene, using only the asynchronous output of an event camera and without relying on ground-truth poses or traditional image frames.

The structure of this work is as follows. In *section 2*, the operating principles of event cameras and the mathematical foundations of differentiable rendering using 3D Gaussians will be explained. Then, in *section 3*, the proposed SLAM pipeline will be covered. In *section 4*, the datasets used in the experiments, and their respective results, will be presented. *Section 5* contains conclusions, encountered limitations and possible further development.

2 Core Principles of Event Cameras and Gaussian Splatting

The core operational mechanism of event cameras is the detection of logarithmic changes in pixel-wise light intensity. Rather than sampling frames at fixed intervals, event cameras monitor each pixel continuously and asynchronously, generating an event when the brightness change surpasses a specified contrast threshold. In a noise-free model, an event $e = (x, y, t, p)$ is triggered at pixel location (x, y) and time t when the change in log-intensity $\ell(x, y, t)$ since the last event at that pixel reaches a threshold C: [3]

$$\Delta L(x, y, t) = L(x, y, t) - L(x, y, t_{\text{prev}}) = p \cdot C \tag{1}$$

Here, $p \in \{+1, -1\}$ denotes the *polarity* of the event, indicating whether the brightness increased (ON event) or decreased (OFF event), and t_{prev} is the timestamp of the previous event at the same pixel. The contrast threshold C is set via programmable bias currents and may differ for ON and OFF events, denoted C_{ON} and C_{OFF}, respectively. In practical devices, the contrast sensitivity C corresponds to

a brightness change of approximately 10–50%. However, experimental setups [4] have demonstrated thresholds as low as 1% under bright illumination. Reducing C increases the likelihood of detecting subtle motion but also makes the sensor more susceptible to noise, pixel mismatch, and spurious events.

Event cameras, by their very design, deliver data as sparse, asynchronous events that excel at capturing motion and illumination changes but lack a dense geometric description of the scene. This creates a gap: while the sensor provides temporally precise cues for motion estimation, it does not directly yield the structured, continuous representation needed for accurate mapping and localization. Gaussian splatting addresses exactly this shortcoming, by encoding the scene not as a continuous volumetric density function, but as a discrete and spatially sparse collection of Gaussians $\boldsymbol{\mathcal{G}} = \{G_i\}_{i=1}^{N}$ [5], each defined by a tuple $G_i = (\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}, o_i, \boldsymbol{c_i})$, where:

- $\boldsymbol{\mu_i} \in \mathbb{R}^3$ is the centroid in world coordinates;
- $\boldsymbol{\Sigma_i} \in \mathbb{R}^{3\times3}$ is the symmetric, positive semi-definite covariance matrix dictating the orientation and spread;
- $o_i \in [0, 1]$ is the scalar opacity (or alpha weight);
- $\boldsymbol{c_i} \in \mathbb{R}^3$ encodes the color, often modeled via view-dependent spherical harmonics.

Rendering a 3D scene from a given viewpoint involves projecting each Gaussian onto the image plane and computing its contribution to individual pixels, based on a 2D covariance matrix derived from the original 3D spatial distribution. This projection process consists of two main transformations:

1) **World-to-Camera Coordinate Transformation**: Each 3D Gaussian is first transformed from the global world coordinate system into the local coordinate frame of the camera using a viewing transformation matrix $W \in \mathbb{R}^{3\times3}$, which in this context corresponds to the rotation component of the camera pose. Since the covariance matrix encodes only shape and orientation, not absolute position, the translation component of the pose is not involved in this step.
2) **Projection onto the Image Plane**: Once expressed in camera coordinates, the 3D Gaussian must be projected onto the 2D image plane. Rather than employing a standard projective camera model, which introduces nonlinearity due to the division by depth, the system adopts an affine approximation. This approximation is obtained by expanding the full projective mapping using a Taylor series and retaining only the first-order terms. The Jacobian matrix $J \in \mathbb{R}^{2\times3}$, derived from this affine approximation, is then used to map the 3D covariance into the 2D image domain.

Given a 3D Gaussian with a covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, the resulting 2D covariance matrix $\Sigma' \in \mathbb{R}^{2\times2}$, which characterizes the elliptical footprint of the projected Gaussian on the image plane, is computed as:

$$\Sigma' = JW\Sigma W^\top J^\top \tag{2}$$

where J is the Jacobian matrix mapping 3D positions to 2D pixel locations.

The color at each pixel location is then computed using an ordered alpha compositing scheme, where Gaussians are sorted by depth and blended near-to-far as follows:

$$C = \sum_{i=1}^{N} \alpha_i' \cdot c_i \cdot \prod_{j=1}^{i-1} (1 - \alpha_j') \tag{3}$$

where c_i is the learned color. Then, the final opacity α_i' is given by:

$$\alpha_i' = \alpha_i \cdot \exp\left(-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu}_i')^\top \boldsymbol{\Sigma}_i'^{-1} (\boldsymbol{x}' - \boldsymbol{\mu}_i')\right) \tag{4}$$

where $\boldsymbol{x}'$ and $\boldsymbol{\mu}_i'$ are coordinates in the projected space.

The differentiable nature of the rendering function for both the Gaussian parameters and the camera poses allows for end-to-end optimization using gradient-based methods. A typical training objective involves minimizing a photometric reconstruction loss, often combined with a perceptual or structural similarity term such as SSIM.

3 Methodology

The core of this work is built on IncEventGS [2], a novel pipeline, whose structure is shown in Fig. 1, that enables incremental 3D scene reconstruction and camera pose estimation using only an event camera, without relying on any ground-truth poses or conventional image frames. Its structure is analogous to a traditional SLAM system, adopting a tracking and mapping paradigm. However, it is specifically designed to work with the unique characteristics of asynchronous event data and to leverage the rendering and optimization advantages of 3D Gaussian Splatting.

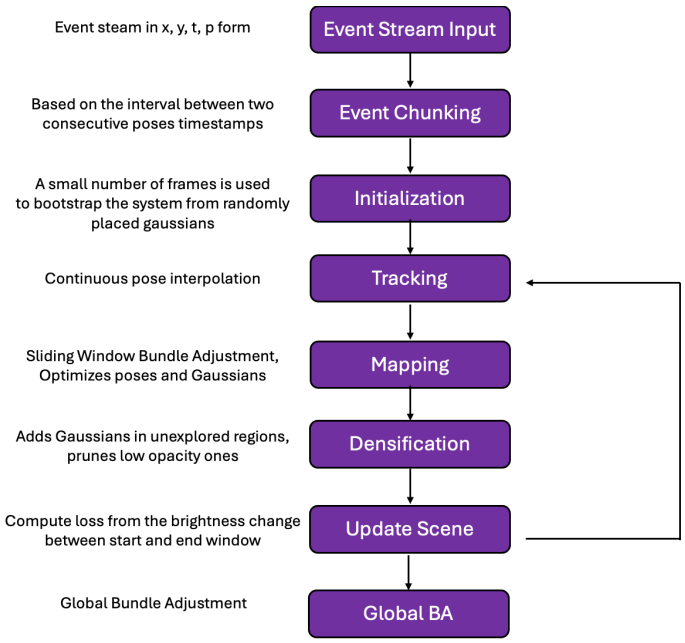

Fig. 1 Pipeline Diagram

While the underlying pipeline architecture follows the IncEventGS framework, the primary contribution of this work lies in its application to and validation within the unique constraints of space proximity operations. Unlike the standard implementation, which is optimized for high-texture in-door environments, the proposed framework introduces a predominantly black background. Therefore, the Gaussian attributes (color and opacity) are initialized toward lower values, which also benefits the initialization of the trajectory and overall scene representation.

3.1 Event Chunking

Each event is defined by a tuple (x, y, t, p) representing the pixel coordinates, timestamp, and polarity. Since event cameras emit events asynchronously and continuously, it is not feasible to process them one

by one in this pipeline. Instead, the stream is segmented into fixed-duration windows. Each chunk contains all the events between two timestamps:

$$\varsigma_i = \{e_j \| t_i < t_j < t_i + \Delta t\} \tag{5}$$

To supervise pose estimation and scene reconstruction, the pipeline does not use all events in a chunk simultaneously. Instead, it operates on randomly sampled temporal windows drawn from within each chunk. This ensures a diverse and computationally efficient training procedure.

The procedure is as follows. The pipeline creates a valid set of event indices that can be used as endpoints of a sampling window. This set begins at `max_n_winsize`, which bounds the maximum number of events in each randomly sampled temporal window when the pipeline draws training chunks from the event stream and extends to the index of the last available event, ensuring that any randomly chosen end index has enough room to subtract the maximum possible window size without underflowing the start of the chunk. If `max_n_winsize` exceeds the total number of available events, it is clamped to the maximum event index. However, if `min_n_winsize` is too large to satisfy the range constraint, the pipeline may not be able to sample valid windows and will terminate. A random index `end_idx` is selected from the range `[max_n_winsize, event_total_num]`. This acts as the ending index for the windowed sample. A window size `w` is drawn uniformly at random from the interval `[min_n_winsize, max_n_winsize]`. This determines how many events will be used in the current photometric loss computation, while the start index is computed by subtracting the sampled window size.

3.2 Initialization

A key component of the pipeline is its ability to initialize both the 3D scene and the camera motion without requiring ground-truth poses or conventional image frames. Typically, a good initial point cloud and camera poses would be obtained using Colmap [6]. However, they are difficult to obtain using only a single event camera. Therefore, the system begins by selecting the first m consecutive frames from a user-defined starting index.

3.2.1 Scene Representation

The initial scene is represented by a sparse set of N 3D Gaussian primitives, where N is the total number of initialized Gaussians, which, following the original work, was kept at 100 thousand. These are randomly sampled within a hyperparameterized 3D bounding box that approximately covers the expected size of the scene. Then, the rendering process, analyzed in-depth in section 2, outputs two maps, the depth map and the visibility (alpha) map, which are critical for bundle adjustment and scene densification. The depth at each pixel x is computed as a weighted sum of the depths d_i of all Gaussians influencing that pixel:

$$D(x) = \sum_{i=1}^{N} d_i \cdot \alpha_i(x) \cdot \prod_{j=1}^{i-1}(1 - \alpha_j(x)) \tag{6}$$

where d_i is the depth (z-coordinate in the camera frame) of the center of the i-th Gaussian. The visibility map $V(x)$ encodes the total accumulated opacity at each pixel:

$$V(x) = \sum_{i=1}^{N} \alpha_i(x) \cdot \prod_{j=1}^{i-1}(1 - \alpha_j(x)) \tag{7}$$

The central principle underpinning the optimization process is that the rendered pixel intensity $C(x)$, along with the auxiliary depth and visibility maps $D(x)$ and $V(x)$, respectively, are fully differentiable

to both the learnable parameters of the 3D Gaussian primitives, namely their position, scale, orientation, color, and opacity. This property of differentiability is essential, as it enables the use of gradient-based optimization methods to jointly refine the underlying scene geometry and the camera motion trajectories through a unified bundle adjustment framework. As a result, a typical output of the initialization process, here highlighted in Fig. 2, consists of a rendered brightness image $C(x)$, accompanied by a corresponding depth map $D(x)$ derived from the spatial configuration of the Gaussians, which together serve as the foundation for the subsequent tracking and mapping stages of the pipeline.

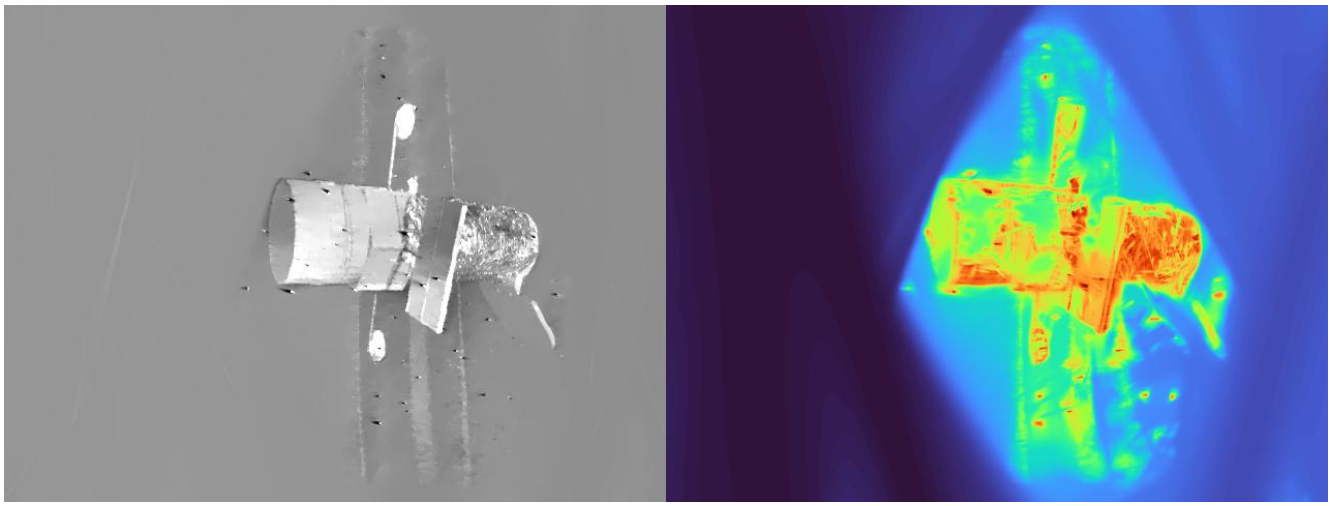

Fig. 2 Typical initialization output showing the rendered image (left) and the corresponding depth map of the Gaussians (right). Warmer colors represent closer Gaussians, while colder colors represent further away ones.

Moreover, the effectiveness of the initialization phase, which is a crucial determinant of the overall reconstruction quality, can be qualitatively assessed by inspecting intermediate outputs generated at configurable training intervals, shown in Fig. 2. These intermediate outputs, which are organized in a structured grid layout, serve as diagnostic indicators of convergence and correctness and are created automatically once a certain number of iterations is reached, as in Fig. 3.

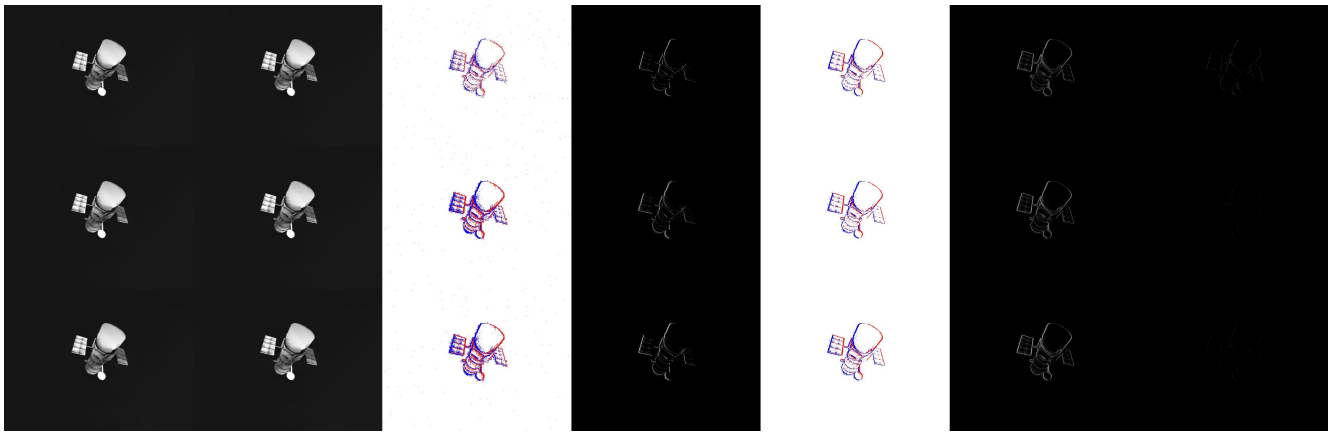

Fig. 3 Intermediate outputs during the initialization stage. Each row corresponds to a sample from the training batch. From left to right: (i) rendering at the segment's starting timestamp, (ii) rendering at the segment's ending timestamp, (iii) ground-truth accumulated events (color), (iv) ground-truth accumulated events (grayscale), (v) predicted accumulated events from the current Gaussian model (color), (vi) predicted accumulated events (grayscale), and (vii) per-pixel squared error map between predicted and ground truth.

Since this phase is critical for the overall success of the pipeline, it is necessary to identify and tune the 1.22 that exert the strongest influence on both the visual quality of the reconstruction and the accuracy of the estimated trajectory. Among these, the most relevant are the `pose_lr`, which controls the learning rate for the Adam optimizer, the `bounding_size`, which determines the initial spatial extent of the scene representation, and the `num_opti_steps_for_init`, which fixes the number of optimization iterations performed during initialization.

3.2.2 Event Loss & SSIM

Control-knot timestamps correspond to the start and end of the selected event frames. For each timestamp, a pose is created by sampling a small random SE(3) perturbation, so all initial camera poses lie near the origin with slight random deviations. These poses act as control points for the continuous trajectory model, where both the poses and the Gaussian parameters are jointly optimized to minimize the photometric error between the observed event stream and the synthetic brightness changes rendered from the current 3D scene estimate. The loss function is formulated as a weighted combination of two complementary components: the first term, $\mathcal{L}_{\text{event}}$, enforces photometric consistency by directly comparing the brightness variations observed in the event stream with those predicted by the current Gaussian-splatting model. The second term, $\mathcal{L}_{\text{SSIM}}$, introduces a structural similarity constraint that preserves local contrast and perceptual quality in the rendered images, acting as a regularizer to mitigate the sensitivity of purely pixel-wise losses. The complete objective is thus expressed as:

$$\mathcal{L} = (1 - \lambda) \cdot \mathcal{L}_{\text{event}} + \lambda \cdot \mathcal{L}_{\text{SSIM}} \tag{8}$$

where $\lambda \in [0, 1]$ is a hyperparameter balancing parameter that weights the contribution of each term, in this work kept between 0.05 and 0.15. The predicted brightness change is defined as:

$$\hat{E} = \log \hat{I}_{t+\Delta t} - \log \hat{I}_t \tag{9}$$

where $\hat{I}_k$ and $\hat{I}_{k+\Delta t}$ are synthetic brightness images rendered from the 3D Gaussians at the estimated start and end poses of the event chunk. The event loss is then computed as the L1 norm over all valid pixels:

$$\mathcal{L}_{\text{event}} = \frac{1}{N} \sum_{x \in \Omega} \left| E - \hat{E} \right| \tag{10}$$

This formulation aligns the predicted contrast change with the temporal log-intensity variation captured by the event stream, providing direct supervision for both camera motion and scene geometry. To improve robustness to local noise and enhance perceptual quality, an auxiliary term based on the Structural Similarity Index (SSIM) is added. This term compares the structure and contrast of the two rendered images:

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(\hat{I}_t, \hat{I}_{t+\Delta t}) \tag{11}$$

The SSIM index is computed over local patches and captures differences in luminance, contrast, and structural alignment between the two views. Including this term encourages the model to preserve consistent structural features across viewpoints, even when brightness variations exist. Once the initialization phase has produced an initial 3D scene and estimated camera poses for a small set of chunks, the pipeline enters the tracking phase.

3.3 Tracking

The tracking module is responsible for estimating the camera trajectory associated with each new event chunk, using the current 3D Gaussian scene as a fixed reference. The key idea is to fit a continuous camera motion model that aligns the rendered brightness change with the one derived from the event stream. For each batch of event data, the method begins by gathering the corresponding event timestamps and streams. Every new frame (or chunk) introduces a new control knot in the camera trajectory. Initial poses at each knot are predicted from previously estimated poses to maintain temporal continuity. In the default two-knot mode, only the first and last control knots in the batch are optimized. The intermediate poses are inferred via interpolation. The timestamps of the control knots are adjusted such that they bracket the entire window of sampled events.

For any timestamp $t \in [t_k, t_{k+\Delta t}]$, the camera pose $T(t) \in \mathrm{SE}(3)$ is computed as:

$$T(t) = T_k \cdot \exp\left(\frac{t - t_k}{t_{k+\Delta t} - t_k} \cdot \log\left(T_k^{-1} T_{k+\Delta t}\right)\right) \tag{12}$$

This continuous interpolation allows the system to compute differentiable camera poses at arbitrary timestamps within the chunk, which is essential for rendering brightness images aligned with the asynchronous event data. Then, to estimate the optimal start and end poses, the system compares the measured brightness change computed from a subset of events within the chunk and the rendered brightness change obtained from the current 3D Gaussian model using interpolated camera poses. After each new chunk is processed in the tracking stage to estimate its start and end poses, the pipeline enters the mapping phase, where both the camera trajectory and the 3D scene are jointly refined.

3.4 Mapping

While tracking provides a local estimate of motion for each event chunk, mapping ensures global consistency and improved accuracy by performing optimization over a sliding temporal window. This process is conceptually related to bundle adjustment in traditional SLAM systems. Optimization is restricted to a hyperparameterized sliding window of recent event chunks, which typically contains from 20 to 30 frames, depending on the sequence size. Rather than optimizing the entire trajectory at once, the system focuses on refining only the poses and Gaussians visible within the current window.

A key process in this phase is the **densification strategy**, which refers to how already existing Gaussians' properties are modified: the so-called default strategy aims at enhancing the scene's quality by acting on occluded, crowded, or sparse areas. This is accomplished through several hyperparameters, among which the most essential are: `prune_opa`, which sets the minimum opacity below which a Gaussian is pruned; `prune_scale3d`, which defines the minimum allowed 3D scale for a Gaussian to remain in the model; `grow_grad2d`, which measures the image-space gradient magnitude to prioritize feature-rich areas; `grow_scale3d`, which controls the minimum 3D scale of newly added Gaussians; `event_threshold`, which specifies the minimum magnitude an event must have to be considered valid and filters out pixels whose accumulated event value does not surpass the threshold when selecting pixels for scene initialization; and finally `num_opti_steps_for_BA`, which fixes the number of optimization steps performed during the bundle adjustment stage, and was set to 3001 for all datasets.

3.5 Global Bundle Adjustment

During Global Bundle Adjustment, the algorithm first compiles a list of candidate event windows spanning the entire sequence and, at each optimization iteration, selects a random subset of these by generating a collection of randomized start–end index pairs, from which it samples a shuffled group constrained by the training batch size parameter, ensuring that each sampled window may originate from any point in the temporal sequence. For each selected window, the corresponding start and end

timestamps are used to interpolate camera poses, which are then employed to render the scene at both endpoints, thereby producing a pair of images that capture the object from differing viewpoints and orientations.

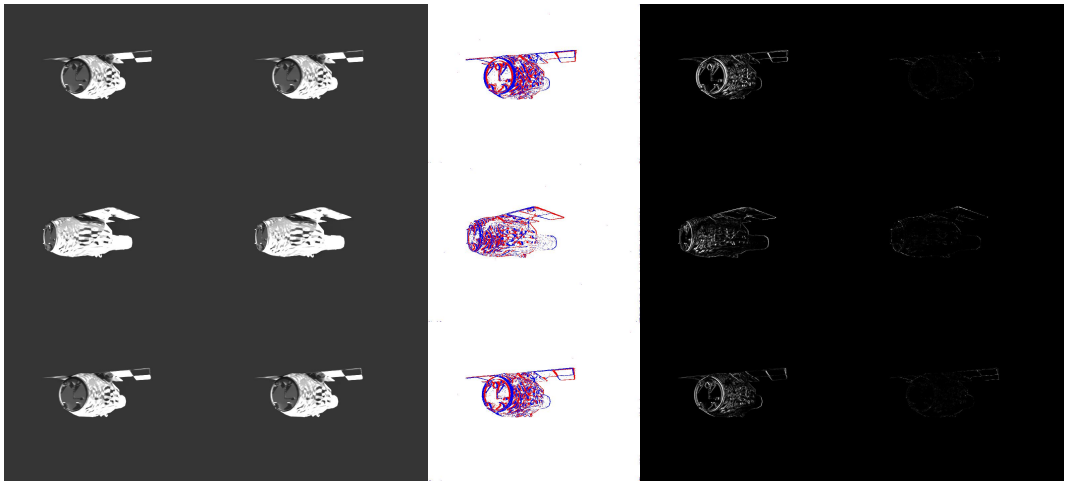

Fig. 4 Global BA Output on the Orbit's Blender sequence: random segments throughout the run are selected depending on the batch and then optimized. The maximum number of iterations for this phase is limited to 6001.

The resulting visualization image, shown in Fig. 4, for this specific batch size of three, is constructed by concatenating five distinct panes: the rendered image at the start timestamp, the image at the end timestamp, the ground-truth event accumulation, the predicted event accumulation produced by the Gaussians, and their per-pixel squared difference. Each row displays the object under a unique pose configuration, varying from row to row due to the inherent randomness in the temporal window selection.

4 Metrics

To assess the accuracy of the estimated trajectories, a two-step registration process is implemented to align the estimated poses with the ground-truth data. Since the event-only SLAM system operates without an absolute scale, the **Umeyama method** [7] is first employed, providing a closed-form solution for a similarity transformation, effectively compensating for differences in translation and scale. The method effectively comprises three steps:

1) **Correlation Analysis:** A cross-covariance matrix is constructed between the centered trajectories to determine the spatial relationship between the two paths.
2) **Rotation Estimation:** Using Singular Value Decomposition on the covariance matrix, the optimal orthogonal rotation is extracted. A correction step is applied to ensure the result is a valid rotation.
3) **Scale and Translation Recovery:** The isotropic scale factor is determined by comparing the spread of the two point sets. Finally, the translation vector is recovered by mapping the centered estimate back to the ground-truth coordinate system.

This alignment is applied exclusively for evaluation purposes, ensuring the internal operations of the SLAM pipeline remain unaltered, and is particularly valuable for event-only or monocular SLAM systems because it accounts for the absolute scale, which is not directly observable by the sensor.

Moreover, in the absence of a shared global reference frame, the algorithm may not fully resolve rotational offsets. Consequently, a **first-pose alignment** strategy is adopted to evaluate the orientation error accurately. Let $\mathbf{R}_{gt,1}$ and $\mathbf{R}_{est,1}$ denote the rotation matrices of the first pose in the ground-truth and estimated trajectories, respectively. The rotation alignment transformation is defined as:

$$\mathbf{T}_{\text{align}} = \begin{bmatrix} \mathbf{R}_{\text{gt},1}\mathbf{R}_{\text{est},1}^{-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{13}$$

This transformation is applied to each estimated pose $\mathbf{T}_{\text{est},i}$ to obtain the aligned pose $\mathbf{T}_{\text{est-aligned},i}$:

$$\mathbf{T}_{\text{est-aligned},i} = \mathbf{T}_{\text{align}}\mathbf{T}_{\text{est},i} \tag{14}$$

Following global bundle adjustment, the translational accuracy is quantified using the Absolute Trajectory Error (ATE). The performance is summarized by the Root Mean Squared Error (RMSE) over N poses:

$$\text{ATE}_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{p}_i^{\text{gt}} - \mathbf{p}_i^{\text{aligned}}\right\|^2} \tag{15}$$

To provide a unified measure of orientation error, the relative rotation between the aligned estimate and the ground truth is calculated for each frame i:

$$\mathbf{R}_{\text{err},i} = \mathbf{R}_{\text{gt},i}^{\top}\mathbf{R}_{\text{est-aligned},i} \tag{16}$$

The corresponding angular deviation θ_i is derived as:

$$\theta_i = \cos^{-1}\left(\frac{\text{tr}(\mathbf{R}_{\text{err},i}) - 1}{2}\right) \tag{17}$$

Scene reconstruction fidelity is first evaluated with the Peak Signal-to-Noise Ratio (PSNR), a traditional metric that relates the maximum pixel intensity of an image to the average squared difference between reconstructed and reference frames, expressed in decibels so that higher values correspond to greater similarity. While simple and widely adopted, PSNR is very sensitive to pixel-level errors and often misaligns with human perception, as shown in cases where darker Gaussians artificially boost scores by matching background intensity without improving structure. To overcome this, the Learned Perceptual Image Patch Similarity (LPIPS) metric was introduced, which compares images in the feature space of deep convolutional networks such as AlexNet [8], aggregating activations across layers with learned weights to better reflect perceptual quality. The resulting score ranges from zero to one, with lower values indicating stronger similarity, and is more robust to shifts, deformations, and structural differences, making it particularly suitable for evaluating neural renderings and reconstructions. Both metrics were computed with the Image Quality Assessment (IQA) toolbox [9].

5 Mission Scenarios & Results

5.1 SEENIC

A critical step toward validating event-based visual odometry systems in practical space applications is evaluating their performance on data that reflects real-world operational conditions. To this end, the **SEENIC dataset (Spacecraft pose Estimation with NEuromorphIC vision)** [10], provides one of the few publicly available benchmarks explicitly designed to support spacecraft relative navigation using event cameras.

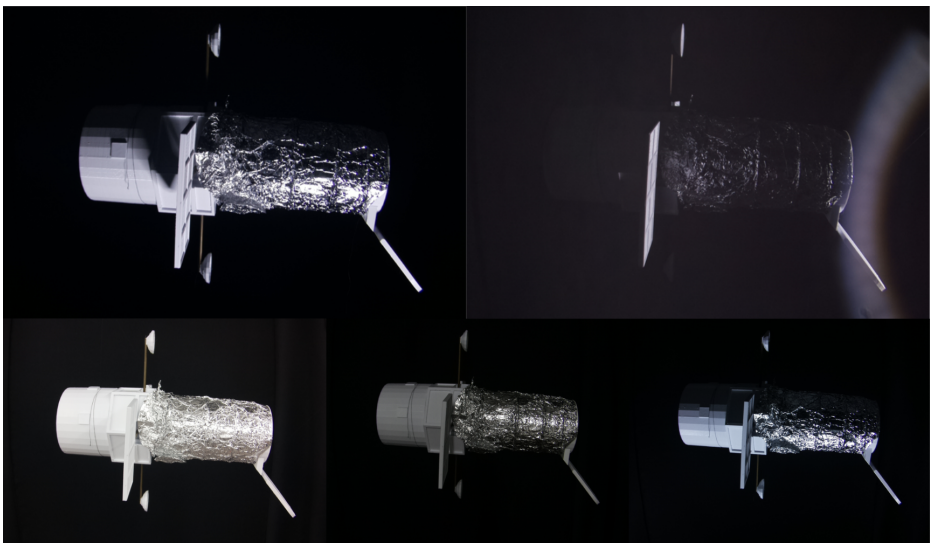

Fig. 5 SEENIC's different light configurations, from top to bottom, left to right: ringbelow, ringside, ambient, centre, lightbox

The dataset was constructed to replicate typical rendezvous and inspection scenarios that might occur during proximity operations around a spacecraft. To do so, the authors used a 3D-printed model of the Hubble Space Telescope (HST), which was wrapped in foil to mimic the highly reflective and feature-sparse surfaces commonly found on real spacecraft. The event data were collected using an Inivation DVXplorer event camera, having a resolution of 640 by 480 pixels, which was mounted on a 6-DoF UR5e robotic manipulator. Each recorded sequence follows a predefined path around the target, ensuring comparability across experimental conditions, with two distinct motion profiles used to simulate different operational phases: a linear approach trajectory resembling a docking maneuver and a circular orbit trajectory consistent with close-range inspection. For both motion types, two velocity profiles, slow and fast, were implemented, resulting in four combinations of motion and speed. To further increase realism and test perceptual robustness under varying orbital conditions, the dataset includes five distinct lighting configurations, which range from ambient illumination, where the scene (Fig. 5) is evenly lit, to highly directional lighting schemes that introduce strong contrast, shadows, and backlighting effects.

While the pipeline demonstrates robustness against speckle noise, primarily because isolated and spatially compact Gaussians are naturally eliminated through the pruning mechanism embedded in the densification strategy, noise lines pose a more persistent difficulty, as they tend to trigger the aggregation of small, spatially near Gaussians, which the system erroneously interprets as coherent structural features and therefore preserves or even consolidates.

Table 1, which reports the pose estimation and reconstruction metrics obtained on the SEENIC dataset, shows that, despite the presence of noise, the translational accuracy remains consistently within sub-millimetric levels (0.11–0.20 cm), whereas the drift in orientation varies more noticeably across scenarios.

Table 1 Pose Estimation & Reconstruction Metrics @ SEENIC | In-depth results in Section 6.

Sequence	Frames	RMSE	Drift
Orbit - Ambient	375-425	0.11 cm	10.1°
Approach - Ambient	90-140	0.11 cm	5.0°
Orbit - Centre	20-120	0.20 cm	7.4°

5.2 E-M-S

The second scenario analyzed is based on the **Event-Monocular Spacecraft (E-M-S)** dataset, introduced in [11] as the first publicly available synthetic dataset for spacecraft pose estimation combining RGB images and neuromorphic event data. The dataset was generated with Unreal Engine 4, using a resolution of 640 by 480 pixels, with precise lighting models to simulate both standard and low-light illumination scenarios. The synthetic images were post-processed to reflect real-world degradations, such as Gaussian blur and speckle noise, and subsequently converted into asynchronous event streams using the *vid2e* simulator [12]. Each sequence in the dataset comprises 100 continuous frames, each paired with a ground-truth 6D pose and relative event stream, as shown in Fig. 6 & Fig. 7. The relative motion trajectories between the spacecraft and a weakly-textured target (the Hubble Space Telescope model) were designed to emulate realistic docking maneuvers, covering varied velocities, distances, and viewpoints.

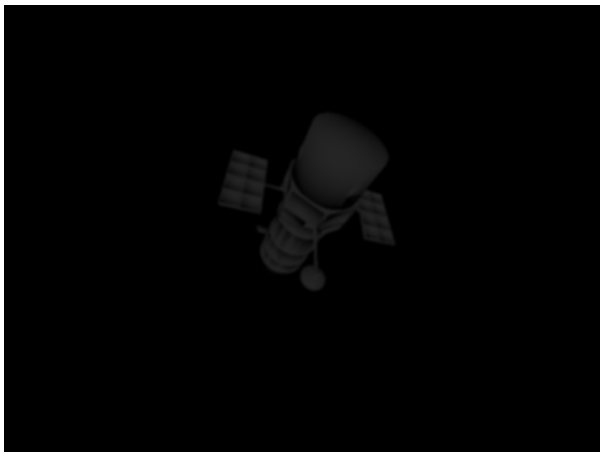

Fig. 6 A regular frame from EMS 100

Fig. 7 The event frame from EMS 100

This dataset yielded some of the strongest performances, highlighted in Table 2, since in most cases the entire sequence was successfully exploited for both translation and orientation estimation, although the latter consistently exhibited a slightly inferior evolution in accuracy. Nevertheless, the experiments also revealed a recurrent limitation, particularly evident in long sequences, namely the gradual increase in brightness affecting the reconstructed three-dimensional scene.

Table 2 Pose Estimation & Reconstruction Metrics @ EMS. In-depth results in Section 6.

Sequence	Frames	RMSE	Drift	PSNR	LPIPS
EMS 0	0-60	0.24 cm	10.5°	11.71 dB	0.37
EMS 69	0-65	0.30 cm	10.1°	13.19 dB	0.35
EMS 100	0-75	0.31 cm	8.9°	13.05 dB	0.34
EMS 420	0-65	1.01 cm	23.8°	13.24 dB	0.26
EMS 479	0-100	0.91 cm	23.3°	10.85 dB	0.37

5.3 Blender

Inspired by SEENIC and E-M-S, a set of custom synthetic sequences was created using Blender, choosing as a model the publicly available Suzaku X-ray satellite [13], reported in Fig. 8, due to its geometry and diverse panel textures.

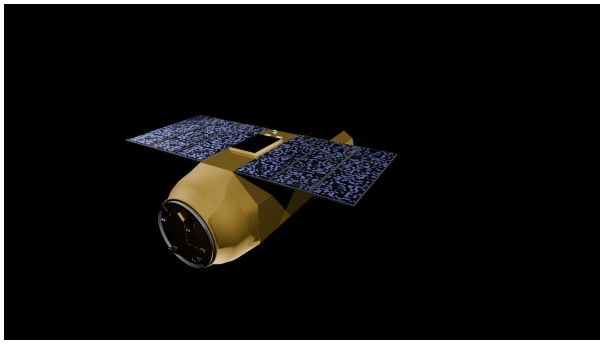

Fig. 8 Suzaku Vanilla model

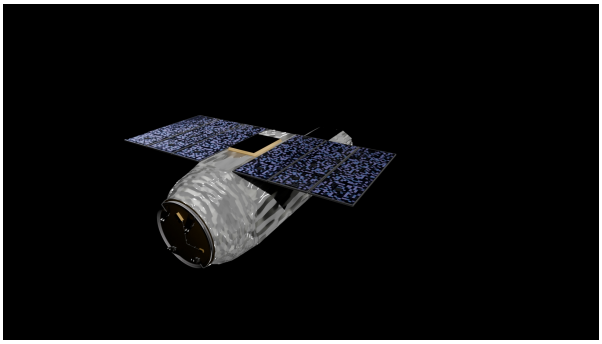

Fig. 9 Suzaku model after applying textures

To better reproduce the photometric dynamics of space environments, a tin-foil-like texture mimicking Multi-Layer Insulation (MLI, Fig. 9) was applied to the spacecraft model, generated through procedural noise with varying roughness levels to test how reflectance affects event generation compared to a textureless surface. The scene was illuminated by four adjustable light sources, three arranged in a triangular configuration above the target and one placed below, set to either 100 W for dim conditions or 10,000 W for bright scenarios. Three camera trajectories were designed: a straight approach, a sinusoidal bent path completing half a cycle, and a circular orbit under perfect lighting; in practice, only subsets of frames were processed due to demanding hardware resources. RGB frames were converted into event streams using the *vid2e* tool [12], with the `--hdr` option enabled in low-light sequences to increase event density, improving 3D reconstruction.

Analyzing the outcomes, the better-lit Blender sequences exhibited greater translation accuracy, both in the approach and swirl modality, while in terms of orientation error, both scenarios behaved similarly. Additionally, despite being far from the regular Gaussian Splatting results, they still provide good scene reconstruction, with LPIPS values being better for Blender scenes, which were not tampered with noise, as opposed to the E-M-S sequences.

Table 3 Pose Estimation & Reconstruction Metrics @ Blender. In-depth results in Section 6.

Sequence	Frames	RMSE	Drift	PSNR	LPIPS
Docking Low-Light	0-75	2.06 cm	14.8°	/	/
Docking Good-Light	0-75	1.86 cm	18.6°	13.82 dB	0.24
Swirl Low-Light	0-75	6.11 cm	8.9°	/	/
Swirl Good-Light	0-75	1.54 cm	10.5°	12.86 dB	0.18
Orbit	31-129	6.94 cm	8.9°	12.18 dB	0.19

5.4 ELOPE

The ELOPE challenge (Event-based Lunar Optical-flow Pose Estimation) [14] focuses on egomotion estimation, specifically, the descent trajectory of a lunar lander using only asynchronous event streams, with no reliance on conventional RGB imaging or prior scene knowledge.

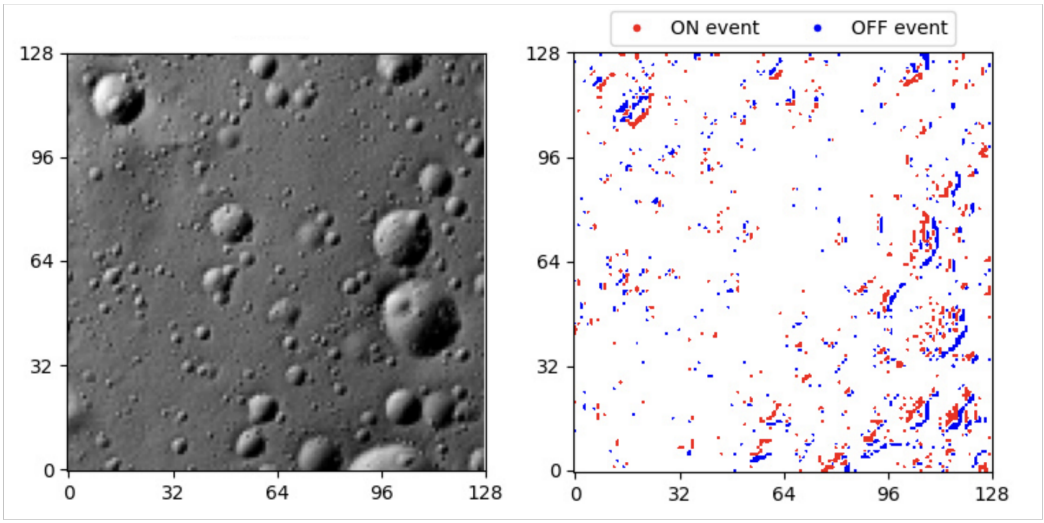

Fig. 10 The Elope dataset consists of event streams synthesized from simulated descents.

The dataset underlying the challenge was generated using ESA's PANGU simulator [15], a physics-based planetary scene generator that can simulate surface geometry and realistic lighting conditions. Event data were synthesized using the *vid2e* [12] tool, applied to renderings of the lunar surface. Each sequence in the dataset represents the descent phase of a hypothetical lunar lander approaching the South Pole and simulates altitude changes from a few kilometers down to approximately 150 meters above the surface. A total of 93 descent sequences are provided, split between training and testing sets. Each includes a time-stamped event stream and ground truth 6-DoF poses, and simulated laser rangefinders. The test set poses are withheld to enable blind evaluation but were not used in this work due to the lack of ground truth data, which is a necessity in order to perform Umeyama alignment.

As the system provides only position estimates and no direct velocity output, linear velocity was derived numerically from the translational trajectory using first-order differencing across consecutive frames, after which the estimated values were compared with the ground truth.

$$\mathbf{v}_i^{\text{est}} = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\Delta t_i} \tag{18}$$

The challenge itself was evaluated based on the scalar score E, defined as the mean normalized velocity error across all frames and sequences, where the difference between estimated and true velocities was scaled by the ground truth altitude at each step Z_i^{gt}; the lower its value, the better the overall performance. Although the pipeline was tested only on a small subset of sequences, the average score is 0.0228, which would have placed it just above the baseline threshold of 0.0179, in fourth place out of 21 groups. A detailed breakdown of the results across the evaluated sequences is provided in Table 4, which reports both the trajectory RMSE and the associated score.

$$E = \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{\left\| \mathbf{v}_i^{\text{est}} - \mathbf{v}_i^{\text{gt}} \right\|}{\left| Z_i^{\text{gt}} \right|} \tag{19}$$

Table 4 Pose Estimation & Reconstruction Metrics @ ELOPE. In-depth results in Section 6.

Sequence	Frames	RMSE	Score
ELOPE 0000	50-120	27.6 m	0.0219
ELOPE 0007	50-120	31.3 m	0.0262
ELOPE 0013	50-120	21.3 m	0.0221
ELOPE 0015	50-120	22.9 m	0.0203
ELOPE 0016	50-120	23.1 m	0.0239

5.5 Comparison with Colmap

To contextualize the performance of the proposed method and provide a benchmark against a widely adopted standard in computer vision, a comparative analysis with Colmap [6], a state-of-the-art structure-from-motion pipeline that has become the de facto tool for high-fidelity 3D reconstruction from regular imagery, was carried out on the synthetic Blender sequences, for which full ground truth RGB images are available. The trajectories retrieved by Colmap were then aligned with both the event-based estimates and the ground-truth, and for quantitative evaluation, the root mean square error was computed (Table 5), after discarding trajectory points deviating more than 100 cm from ground-truth, to avoid interference caused by isolated outliers. Colmap successfully recovered trajectories in the well-lit docking and swirl sequences; however, in the corresponding low-light settings, it failed to converge, underscoring the limitations of classical methods under challenging illumination conditions. The pipeline, on the other hand, achieved translational results an order of magnitude better than Colmap, while still having comparable orientation evolution, and did not fail in low-light conditions.

Table 5 Translation RMSE comparison against Colmap.

Sequence	Pipeline	Colmap
Docking Low-Light	2.06 cm	/
Docking Good-Light	1.86 cm	39.64 cm
Swirl Low-Light	6.11 cm	/
Swirl Good-Light	1.54 cm	44.87 cm

Given the extensive range of hyperparameters already mentioned in Sections 3.2.1 and 3.4, a comprehensive list for every sequence is omitted for brevity. Instead, to demonstrate the sensitivity of the pipeline to the most impactful parameter, primarily the **training batch size**, which controls how many event chunks are used to compute one optimization step, the sequence 420 of the EMS dataset was tested varying the hyperparameter, as shown in Table 6. Each test was conducted on the same hardware, being an NVIDIA RTX 4070 paired with a Ryzen 5 7600 CPU.

Increasing the batch size would theoretically lead to a better result in terms of trajectory accuracy, however, during testing with batch sizes greater than 2, a common issue encountered was that reconstruction itself exhibited visual artifacts over long sequences, where the scene would gradually become unnaturally bright or dark (Fig. 11) or where gaussians would erroneously be instantiated directly in front of the camera (Fig. 12), which occluded the field of view and disrupted all subsequent optimization steps, ultimately reducing the reliability of the system in extended runs. This issue lies deep in the densification settings of standard Gaussian Splatting and requires further investigation beyond the scope of this work.

Table 6 Batch Size Impact on Run Time @ Sequence 420 EMS

Batch Size	Run Time	Time Per Frame	RMSE
1	54 min	74.20 s	0.34 cm
3	96 min	130.44 s	0.62 cm
5	122 min	183.23 s	0.82 cm
7	188 min	257.15 s	1.18 cm

Fig. 11 After many frames, the scene gets very bright due to wrong densification settings.

Fig. 12 A stray, high opacity Gaussian placed between object and camera.

6 Conclusions

This work introduced an event-only approach to spacecraft relative navigation, integrating event camera data with 3D Gaussian splatting to perform joint pose estimation and scene reconstruction. The motivation was the need for navigation techniques capable of operating in the non-cooperative and extreme illumination conditions that characterize future orbital servicing and docking missions.

The developed pipeline demonstrated that event-based sensing, when coupled with an explicit Gaussian representation, can provide accurate trajectory estimation and consistent 3D mapping without relying on conventional image frames, fiducial markers, or external ground-truth poses. Across multiple datasets, the system achieved trajectory accuracy on par with or exceeding classical frame-based baselines in difficult conditions, while also producing geometrically faithful reconstructions of the target. In low-light and texture-poor scenarios, the method succeeded where established pipelines such as Colmap consistently failed. At the same time, the work highlighted several limitations.

Chief among these was the scale ambiguity inherent in monocular event-based systems, requiring post-hoc alignment to ground truth. Additional challenges included the accumulation of drift on long sequences, sensitivity to hyperparameter tuning, and the computational burden of operating on limited hardware, which constrained the exploration of parameter space. Reconstruction artifacts, such as overexposure or spurious Gaussians, further underscored the need for refined densification and initialization strategies. Looking forward, three roads appear particularly promising. The first is the integration of learned depth priors, ideally trained on spaceborne data, to resolve scale ambiguity and stabilize initialization. The second is the systematic use of more powerful hardware, enabling larger batch sizes and longer sequence training for improved robustness. The third is the refinement of the Gaussian representation

itself, either through adaptive densification strategies or hybrid models that combine event-based input with auxiliary sensors.

A decisive step forward, with the ability to directly address the most critical limitation of scale ambiguity, lies in the integration of depth information, either through external sensors, at the cost of sacrificing the purely monocular formulation, or, more promisingly, through the use of neural depth estimation networks trained specifically on spaceborne sequences, which would allow scale to be recovered in a self-contained fashion. Such an approach would not only resolve the scale issue but also substantially improve scene reconstruction, since the corrected positioning of Gaussians would reduce or eliminate the failure cases where stray primitives are instantiated directly in front of the camera, while at the same time eliminating part of the randomness inherent to the initialization process by re-placing Gaussians in physically consistent locations, thereby preserving the stability of the pipeline over long trajectories.

Appendix

Following is the detailed analysis of a single test sequence from each dataset. Additional material is available on the GitHub Page and on the repository's site.

ELOPE: 0000

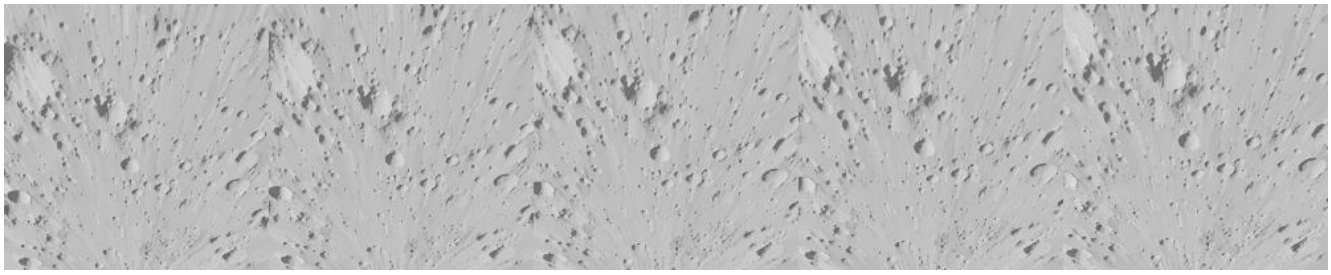

Fig. 13 Elope 0000: Rendered Images @ End of Initialization

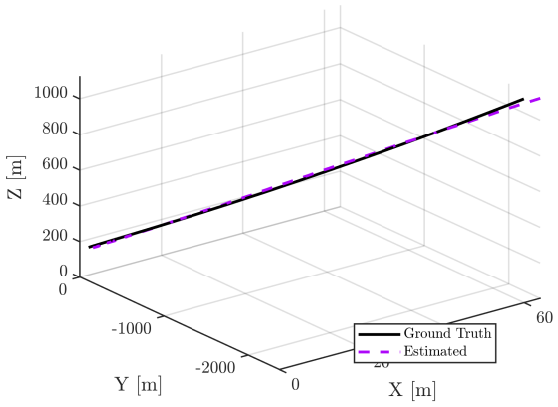

Fig. 14 Elope 0000: Translation Evolution

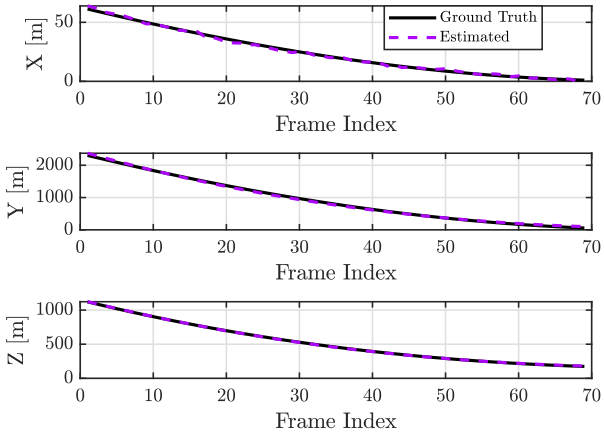

Fig. 15 Elope 0000: Translation Evolution

Table 7 Elope 0000: Translation Metrics

Axis	RMSE
X	1.040 m
Y	27.487 m
Z	2.606 m
Total	27.630 m

Table 8 Elope 0000: Orientation Metrics

Axis	RMSE
Yaw	0.687°
Pitch	0.513°
Roll	1.124°

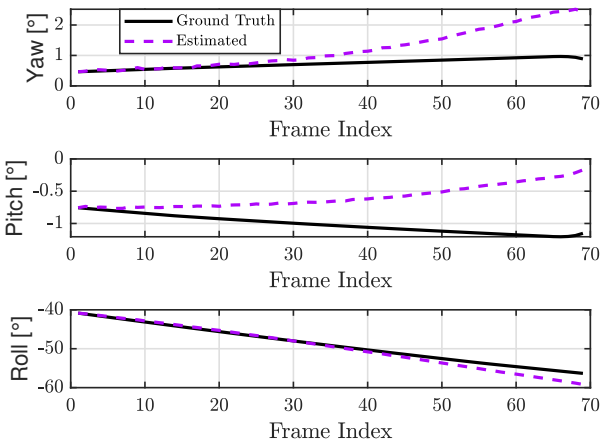

Fig. 16 Elope 0000: Roll, Pitch, Yaw Evolution

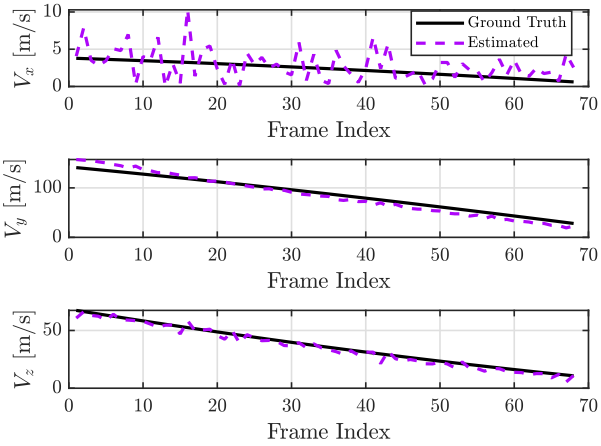

Fig. 17 Elope 0000: Velocity - Score: 0.0219

Blender: Orbit

Fig. 18 Orbit: 3D Scene Reconstruction (Left) vs Ground Truth (Right)

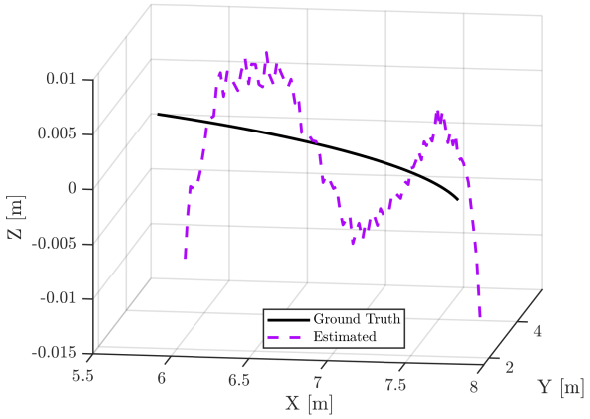

Fig. 19 Orbit: Trajectory

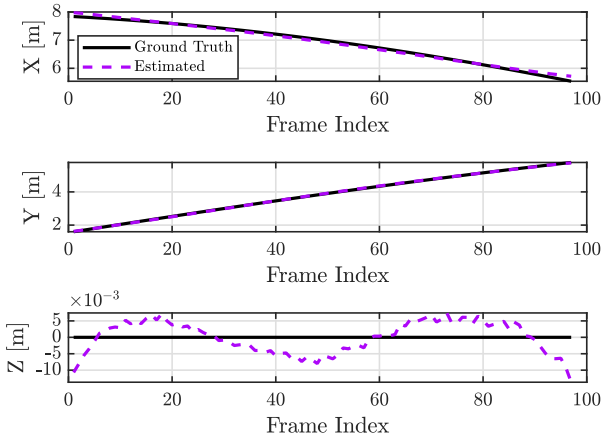

Fig. 20 Orbit: Translation Evolution

Table 9 Orbit: Translation Metrics

Axis	RMSE
X	6.771 cm
Y	1.454 cm
Z	0.483 cm
Total	6.942 cm

Table 10 Orbit: Orientation Metrics

Axis	RMSE
Yaw	2.203°
Pitch	4.512°
Roll	3.708°

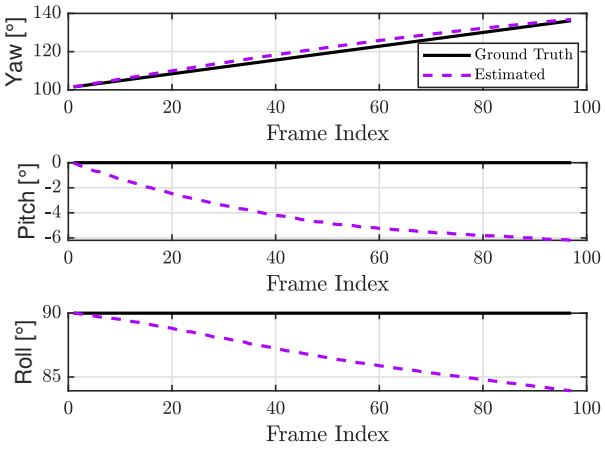

Fig. 21 Orbit: Roll, Pitch, Yaw Evolution

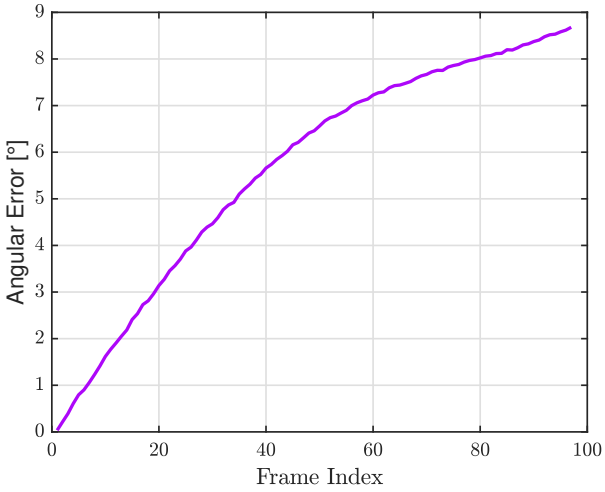

Fig. 22 Orbit: Orientation Error Evolution - Drift @ End Frame: 8.860°

E-M-S: 479

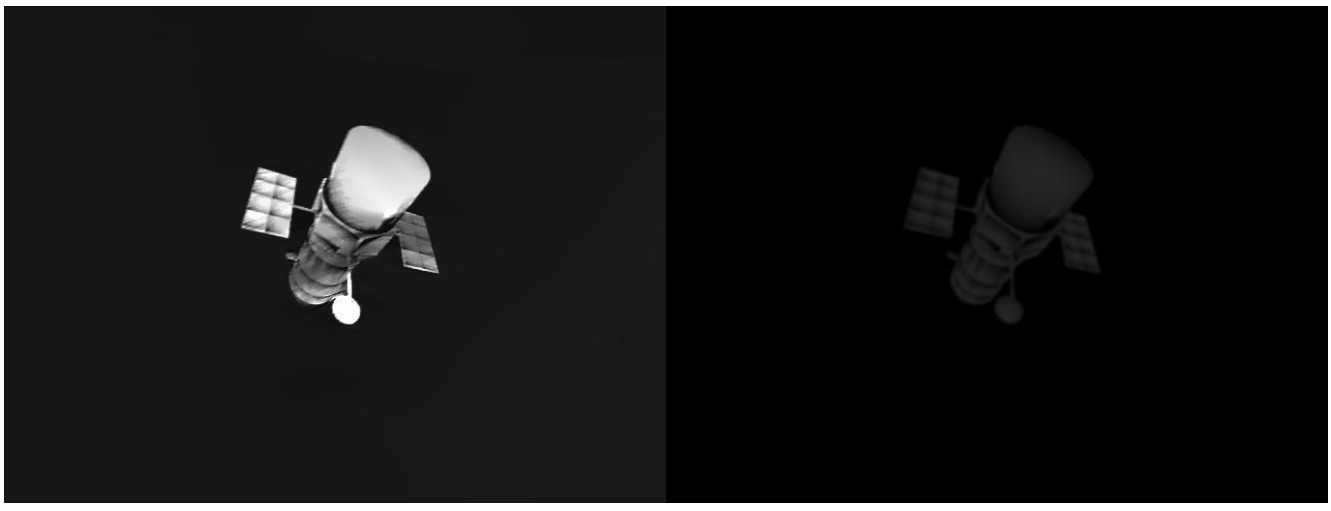

Fig. 23 EMS 479: 3D Scene Reconstruction (Left) vs Ground Truth (Right)

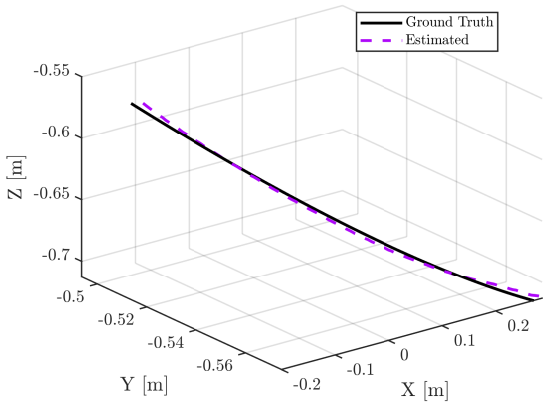

Fig. 24 EMS 479: Translation Evolution

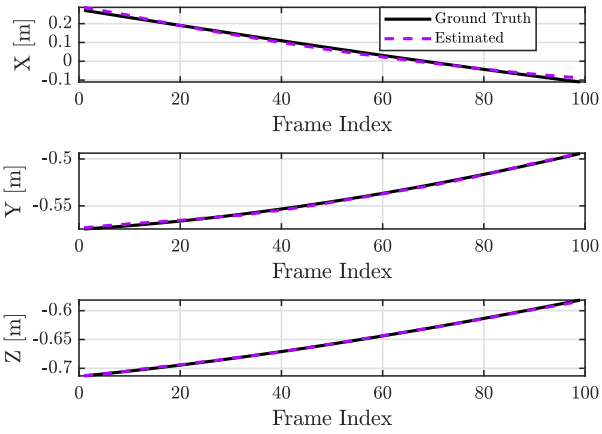

Fig. 25 EMS 479: Translation Evolution

Table 11 EMS 479: Translation Metrics

Axis	RMSE
X	0.907 cm
Y	0.096 cm
Z	0.070 cm
Total	0.915 cm

Table 12 EMS 479: Orientation Metrics

Axis	RMSE
Yaw	9.924°
Pitch	8.858°
Roll	5.134°

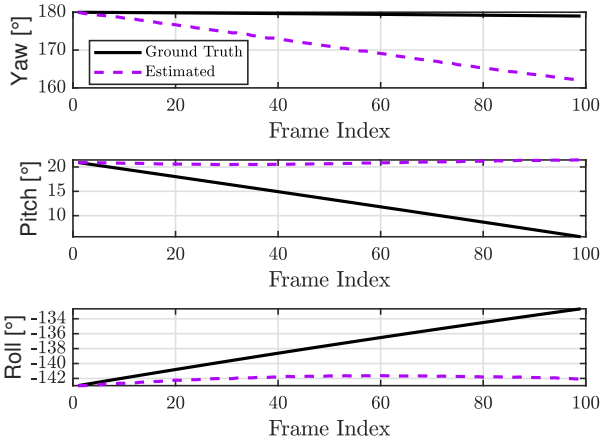

Fig. 26 EMS 479: Roll, Pitch, Yaw Evolution

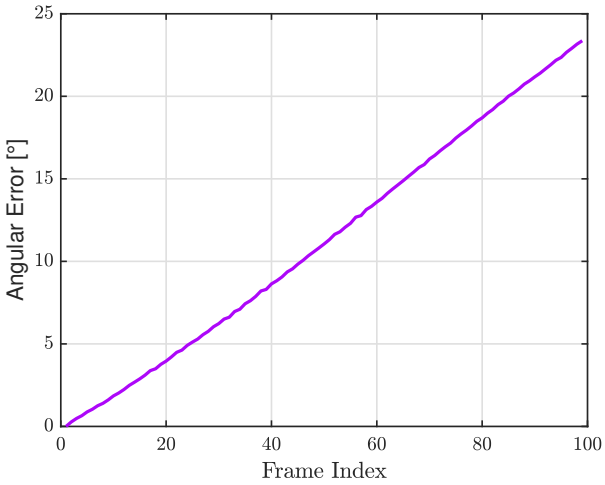

Fig. 27 EMS 479: Orientation Error Evolution - Drift @ End Frame: 23.365°

SEENIC: Orbit - Ambient

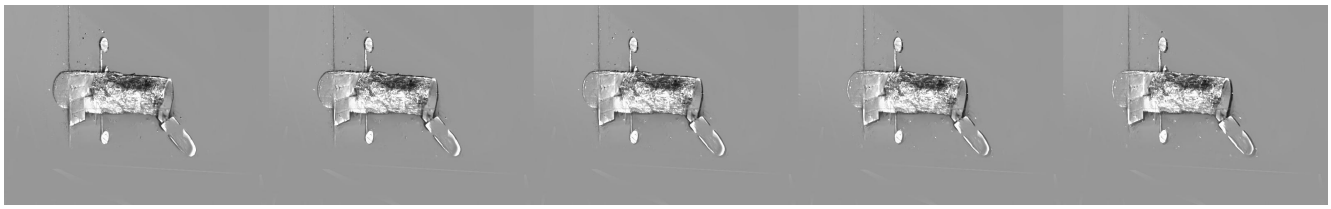

Fig. 28 Orbit Ambient: Rendered Images @ End of Initialization

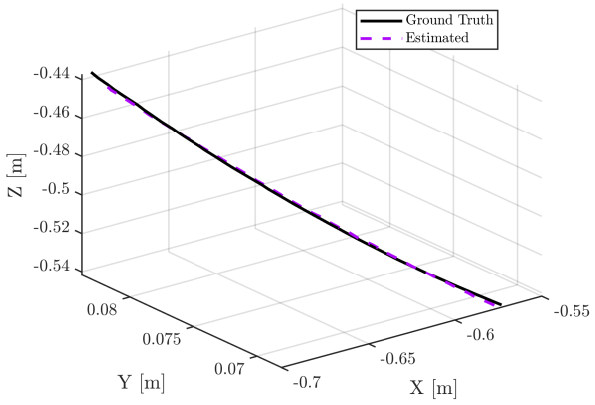

Fig. 29 Orbit Ambient: Trajectory

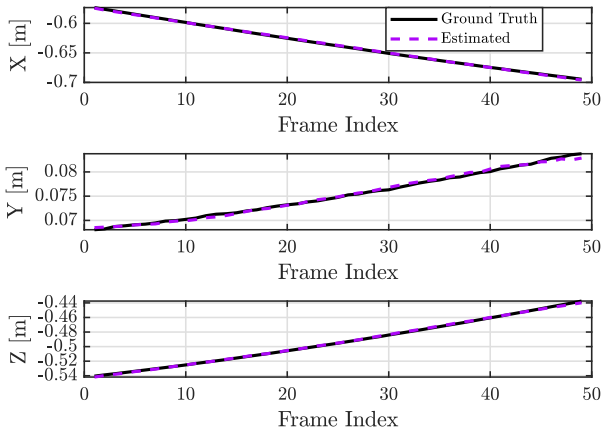

Fig. 30 Orbit Ambient: Translation Evolution

Table 13 Orbit Ambient: Translation Metrics

Axis	RMSE
X	0.071 cm
Y	0.035 cm
Z	0.073 cm
Total	0.108 cm

Table 14 Orbit Ambient: Orientation Metrics

Axis	RMSE
Yaw	3.847°
Pitch	4.884°
Roll	0.662°

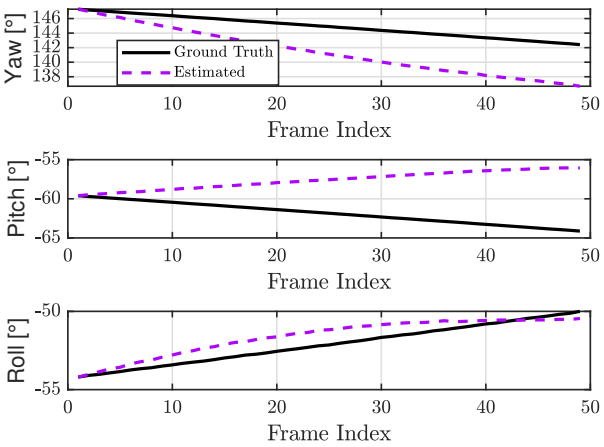

Fig. 31 Orbit Ambient: Roll, Pitch, Yaw Evolution

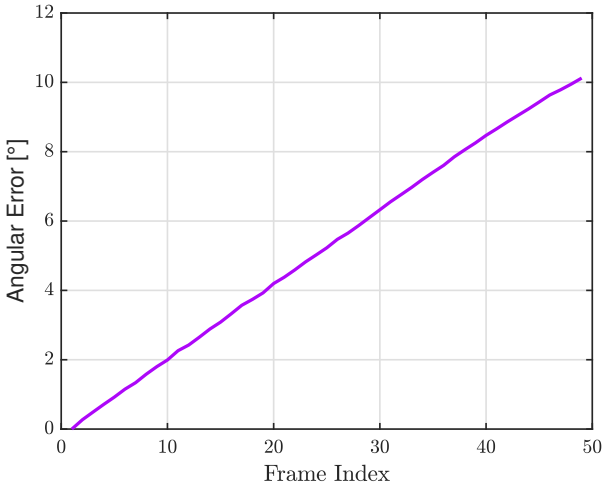

Fig. 32 Orbit Ambient: Orientation Error Evolution - Drift @ End Frame: 10.123°

Acknowledgments

Funded by the European Union through the European Innovation Council (EIC) within the AstrAware project (Grant agreement ID: 101221404). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Innovation Council (EIC). Neither the European Union nor the granting authority can be held responsible for them. Usa il codice con cautela.

Declaration of Use of Artificial Intelligence

Artificial intelligence was used during programming, in order to convert some of the functions present in the EVO repository, from Python to MatLab; specifically, the functions that performed Umeyama alignment and output plots. Furthermore, AI was used in aid of grammar checks; specifically, the Chrome extension Grammarly was used in order to facilitate spotting errors.

References

[1] Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, and Michele Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93:53–72, Aug. 2017. ISSN: 0376-0421. doi: 10.1016/j.paerosci.2017.07.001.

[2] Jian Huang, Chengrui Dong, Xuanhua Chen, and Peidong Liu. IncEventGS: Pose-Free Gaussian Splatting from a Single Event Camera, Mar. 2025. arXiv:2410.08107. doi: 10.48550/arXiv.2410.08107, `http://arxiv.org/abs/2410.08107`.

[3] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-Based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, Jan. 2022. ISSN: 1939-3539. doi: 10.1109/TPAMI.2020.3008413.

[4] Christy Aerne. DVS Videos. `https://inivation.com/developer/videos/`.

[5] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3D Gaussian Splatting as a New Era: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 31(8):4429–4449, Aug. 2025. ISSN: 1941-0506. doi: 10.1109/TVCG.2024.3397828.

[6] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, Las Vegas, NV, USA, June 2016. IEEE. ISBN: 9781467388511. doi: 10.1109/CVPR.2016.445, http://ieeexplore.ieee.org/document/7780814/.

[7] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, Sept. 1976. ISSN: 0567-7394. doi: 10.1107/S0567739476001873.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

[9] Chaofeng Chen and Jiadi Mo. IQA-PyTorch: PyTorch Toolbox for Image Quality Assessment, 2022. Published: [Online]. Available: https://github.com/chaofengc/IQA-PyTorch.

[10] Mohsi Jawaid, Ethan Elms, Yasir Latif, and Tat-Jun Chin. Towards Bridging the Space Domain Gap for Satellite Pose Estimation using Event Sensing, Sept. 2022. arXiv:2209.11945. doi: 10.48550/arXiv.2209.11945, http://arxiv.org/abs/2209.11945.

[11] Yishi Wang, Maestrini Michele, Massarib Mauro, Di Lizia Pierluigi, and Zexu Zhang. E-M-S: Event-Monocular-Spacecraft dataset for 6D pose estimation of non-cooperative targets., Feb. 2025. doi: 10.5281/zenodo.14774652, https://zenodo.org/records/14774652.

[12] SensorsINI/v2e, Aug. 2025. original-date: 2019-06-05T05:42:04Z. https://github.com/SensorsINI/v2e.

[13] Jul 2025. https://science.nasa.gov/3d-resources/.

[14] Ondřej Dvořák, Marcus Märtens, and Dario Izzo. Event-based Lunar OPtical flow Egomotion estimation (ELOPE) dataset, May 2025. doi: 10.5281/zenodo.15421707, https://zenodo.org/records/15421707.

[15] S.M. Parkes, I. Martin, M. Dunstan, and D. Matthews. Planet Surface Simulation with PANGU. In *Space OPS 2004 Conference*, Montreal, Quebec, Canada, May 2004. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2004-592-389, https://arc.aiaa.org/doi/10.2514/6.2004-592-389.