Object-centric Reconstruction and Tracking of Dynamic Unknown Objects Using 3D Gaussian Splatting

Kuldeep R Barad[1,2], Antoine Richard[1], Jan Dentler[2],
Miguel Olivares-Mendez[1] and Carol Martinez[1]

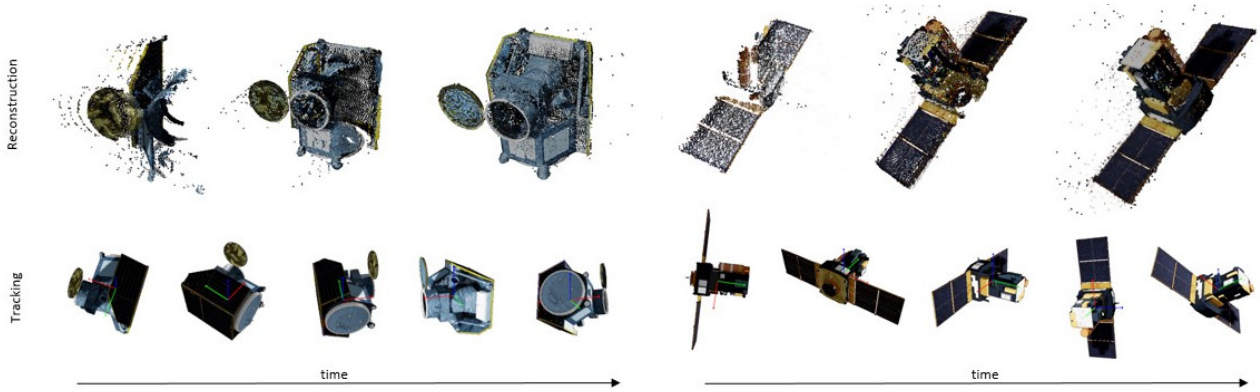


Fig. 1: Incremental reconstruction and tracking of CHEOPS (left) and SOHO (right) spacecraft from a sequence of simulated images without prior training using object-level 3D Gaussian representations and splatting-based differentiable rendering.

***Abstract*— Generalizable perception is one of the pillars of high-level autonomy in space robotics. Estimating the structure and motion of unknown objects in dynamic environments is fundamental for such autonomous systems. Traditionally, the solutions have relied on prior knowledge of target objects, multiple disparate representations, or low-fidelity outputs unsuitable for robotic operations. This work proposes a novel approach to incrementally reconstruct and track a dynamic unknown object using a unified representation- a set of 3D Gaussian blobs that describe its geometry and appearance. The differentiable 3DGS framework is adapted to a dynamic object-centric setting. The input to the pipeline is a sequential set of RGB-D images. 3D reconstruction and 6-DoF pose tracking tasks are tackled using first-order gradient-based optimization. The formulation is simple, requires no pre-training, assumes no prior knowledge of the object or its motion, and is suitable for online applications. The proposed approach is validated on a dataset of 10 unknown spacecraft of diverse geometry and texture under arbitrary relative motion. The experiments demonstrate successful 3D reconstruction and accurate 6-DoF tracking of the target object in proximity operations over a short to medium duration. The causes of tracking drift are discussed and potential solutions are outlined.**

I. Introduction

Generalizable perception systems are crucial for terrestrial and space-borne robotic systems that operate autonomously in dynamic and unstructured environments. In such use cases, the robot needs to reason about the geometric and physical properties of the objects around it for safe navigation and interaction. In challenging applications like on-orbit servicing, robotic interaction, and debris removal, prior knowledge of the target object may not be available or reliable. These object properties must be inferred online using noisy and partial observations from sensors. Due to lower size, weight, power, and cost (SWaP-C), vision sensors are a cost-effective choice for robotic systems. However, extracting object-level properties like geometric structure and motion online from image pixels is challenging. Therefore, we are interested in structure and motion recovery methods that offer: 1) efficient object-level representations and 2) generalization to an open set of objects.

Extraction of object-level structure and relative motion from visual observations concerns two fundamental problems in computer vision- 3D reconstruction and six degrees-of-freedom (6-DoF) pose tracking. The two problems are generally dealt with separately, while the solutions often use different internal representations and assumptions. For instance, the state-of-the-art methods for scene reconstruction in terrestrial applications use Neural Radiance Field (NeRF) representations [1]. NeRF-based methods usually assume a static scene and the availability of relative camera poses. On the other hand, 6-DoF pose estimation and tracking for unknown objects is addressed with large pre-trained neural networks. Many state-of-the-art approaches require a 3D model of the object as a template [2]. Furthermore, a downstream robotic task like grasping may use another neural network that takes RGB-D images as inputs to infer grasps. All these elements require separate datasets, isolated pre-training(s), and parameter tuning.

We focus on the problem of vision-based uncooperative proximity operations with an unknown target spacecraft.

[1]Space Robotics Research Group (SpaceR), Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg. kuldeep.barad@uni.lu

[2]Redwire Space Europe, Luxembourg

Supplementary materials are available at https://kuldeepbrd1.github.io/projects/oc-3dgs/

Recent works have addressed robustness challenges in pre-training Convolutional Neural Networks (CNN) models for a single known object [3]. Tracking is constrained to natural orbital motion [4] and 3D reconstruction is limited to a set of primitive shapes [5]. In this work, we tackle the 3D reconstruction and 6-DoF pose tracking of an unknown object incrementally with a unified underlying representation. Our approach assumes no prior knowledge about the object's structure or motion. We represent an object with a set of points and 3D Gaussian blobs centered at these points with color and opacity properties. Our goal is to fit this object-level representation to an incremental sequence of RGB-D observations. To accomplish this, we use a differentiable rendering framework based on 3D Gaussian Splatting (3DGS) [6] to model the image formation process. The forward map of the differentiable renderer projects the object representation and a camera pose to a rasterized image. The backward map of the renderer propagates the gradients from a rendered image to the object and camera parameters. 3D reconstruction and 6-DoF pose tracking can be accomplished by refining the object and camera pose parameters using gradient-based optimization. The gradient computation can be implemented efficiently using explicitly derived gradients making the optimization process computationally efficient for online use. Unlike neural implicit representations [1], the forward map of the rendering based on 3DGS process is significantly faster [6]. Our approach requires no data generation or pre-training. The input is a stream of RGB-D images of an unknown object in unconstrained relative motion. Through a simple rendering loss, we optimize the object representation and the camera pose alternately in 120 and 80 steps respectively. We test our approach on synthetic image sequences of 10 unique spacecraft models from ESA's science fleet [7]. We demonstrate the effectiveness of our approach on these long video sequences, highlight limitations, and provide recommendations for future development toward reliable and generalizable perception of dynamic unknown objects.

In summary, our work makes the following contributions: **(1) Development of an efficient object-centric framework for incremental reconstruction and tracking of unknown objects using 3DGS.** The original work [6] addresses offline scene reconstruction from posed images. Our work relaxes the need for a static world frame or camera poses and proposes an object-centric framework suitable for online deployment. **(2) Demonstration of high-fidelity reconstruction and 6-DoF pose tracking of dynamic unknown spacecraft**. To the best of our knowledge, we provide the first approach to tackle this problem without assuming prior knowledge about the target and its motion.

II. Related Work

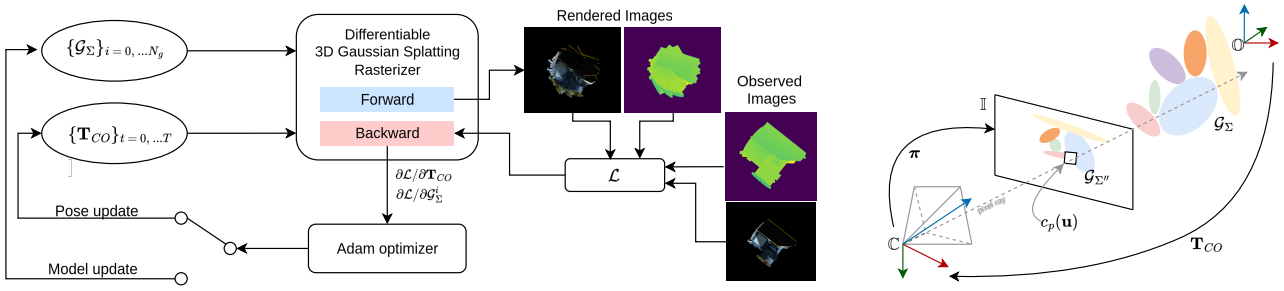
A. Pose estimation and tracking of uncooperative spacecraft

6-DoF pose estimation and tracking for cooperative and uncooperative spacecraft are important problems for proximity operations in orbit. Early works investigated a host of sensor and algorithmic choices while highlighting the need for a cost-effective and robust system [8]. Recently, the focus has shifted towards vision-based pose estimation methods due to the lower SWaP-C of vision sensors. Importantly, the progress was driven by rapidly evolving models and methods in computer vision. Early works [9], [10] utilized conventional image features that either extract parts of the structures like edges [11] and corners [12] or invariant visual descriptors like SIFT [13]. Here, the pose estimation performance relied on feature extraction and robust correspondence matching to a known 3D model. Due to the challenging visual environment, the methods lack robustness to the challenging visual environment in orbit. Subsequently, CNNs demonstrated superior 6-DoF pose estimation results on representative scenarios [14]. However, these networks require extensive pre-training which is done with synthetic data. Using synthetic data for pre-training a neural network presents the problem of performance transfer across the domain gap between the synthetic and real image formation processes. Consequently, high-quality datasets [15], image augmentations [16] and domain adaptation strategies [17], [18] have been proposed to address this issue. Simultaneously, these learned models for pose estimation were extended to state tracking assuming natural motion [4], [19]. Overall, these approaches require exhaustive and iterative cycles of dataset curation, pre-training, and ground testing for every target. Moreover, unmodeled artifacts like structural damage or material degradation remain challenging for these methods relying on the exact 3D model.

The requirement of the target model knowledge was relaxed in [5] which proposed a fast, one-shot primitive reconstruction of the target object. The shape is approximated by a fixed set of superquadrics per image. Despite being fast, the method only recovers the shape of the target and not the scale. Consequently, only a rotation is estimated per image. It also depends on a prior learned from a small dataset of objects. These attributes imply that it suffers from overfitting, viewpoint ambiguities of primitives, and a lack of fidelity. As opposed to these methods, our approach does not require pre-training and considers no apriori information about the object or its relative motion. The object is reconstructed online using a 3D Gaussian representation that simultaneously enables efficient pose tracking in the loop. Our approach applies more generally to any unknown object and can be used for other on-board applications such as robotic manipulation.

B. SLAM

The problem of incremental reconstruction and tracking, addressed in this work, closely relates to the broader research on SLAM. However, SLAM approaches most commonly assume a static world and aim to map and localize a moving camera within the scene [20], [21]. Sparse map recovery can achieve relatively fast localization of a moving camera. However, the sparse map is less useful for tasks beyond localization. To address this, dense SLAM approaches [22], [23] produce dense maps with point or voxel-based representation. However, mapping and localization are more difficult, if the objects in the environments move. Dynamic SLAM



(a) (b)

Fig. 3: (a) **Methodology**: We use differentiable rendering of 3D Gaussians as the core of our incremental reconstruction and tracking pipeline. We refine the camera pose or the object Gaussians using first-order gradient-based optimization by propagating the gradients backward through the rendering process. (b) **3D Gaussian Splatting**: Illustration of projecting 3D Gaussians $\mathcal{G}_\Sigma$ to 2D Gaussian splats $\mathcal{G}_{\Sigma''}$.

and object-centric SLAM methods relax the static world assumption by segmenting the object and registering a frame-to-model colored point cloud [24] using iterative closest points [25] estimation. These methods often use discrete point cloud representations that can be harder to optimize and may not capture high-fidelity details. Recent advancements in 3D reconstruction have enabled the recovery of high-fidelity models from images using differentiable rendering. In particular, neural fields [1] can be used to create a dense map of the scene [26], [27]. Neural fields have been used for object-centric reconstruction and tracking with RGB-D vision [28]. However, neural representations are bottlenecked by expensive sampling along pixel rays as they encode a volume implicitly. On the other hand, 3DGS [6] provides a new representation that enables high-speed forward rendering and the benefits of dynamic manipulation of point-like 3D Gaussian primitives. This representation has enabled state-of-the-art performance in dense SLAM [29]–[31], for large but static scenes. In this work, we investigate the applicability of Gaussian to object-centric reconstruction and tracking applied to characterizing an unknown resident space object.

III. Method

Our approach uses a differentiable rasterizer based on 3DGS [6] and a simple photometric loss $\mathcal{L}$ to incrementally optimize the object and pose parameters as shown in Fig. 3a. Intuitively, this method stores an evolving internal model of the object and camera motion to render an expected image. Then, the photometric distance or dissimilarity between this image and the observed image represents the gap between the true and the internal model. The object and camera parameters are then refined using gradient-based optimization. The input to our pipeline is a sequential set of RGB-D images. We assume the target spacecraft is sufficiently resolved in the image and its RGB appearance is view-independent. In the following, we describe each component of our method in detail.

A. Object Representation

We represent the appearance and geometry of the target object as a set of $N \in \mathbb{N}^+$ 3D Gaussian ellipsoids. Each Gaussian $\mathcal{G}^i$ is parameterized in terms of a 3D center ($\boldsymbol{\mu}^i \in \mathbb{R}^3$) and 3D covariance ($\boldsymbol{\Sigma}^i \in \mathbb{S}^3_+$), evaluated at any point $\mathbf{x}$ as:

$$\mathcal{G}_{\boldsymbol{\Sigma}}(\mathbf{x}) = \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \quad (1)$$

In addition, the appearance of each Gaussian is parameterized by its color ($\mathbf{c}^i$) and opacity (o^i), where $i = 0,\ 1,\ 2 \ldots N$. We assume that the appearance of each Gaussian is independent of the viewing direction and exclude the spherical harmonics coefficients from [6] to speed up rendering. Note that this representation is explicit and similar to point-based representations. Consequently, Gaussian ellipsoids can be added, removed, and modified dynamically during runtime. Unlike implicit representations, this enables better control over reconstruction fidelity and rendering efficiency.

B. Differentiable rendering

The forward projection of a set of 3D Gaussians result in 2D splats on the image as illustrated in Fig. 5. A 3D Gaussian in the object space ($\mathbb{O}$) can be projected to the camera space ($\mathbb{C}$) by an affine view transformation $\mathbf{T}_{CO} \in SE(3)$, with rotation component $\mathbf{R}_{CO} \in SO(3)$ and translation component $\mathbf{t}_{CO} \in \mathbb{R}^3$. The Gaussian representation is convenient as the affine transformation results in another 3D Gaussian given by Eq. 2.

$$\mathcal{G}_{\boldsymbol{\Sigma}}(\mathbf{T}_{CO}^{-1}\mathbf{x}_c) = \frac{1}{|\mathbf{R}_{CO}^{-1}|}\mathcal{G}_{\boldsymbol{\Sigma}'}(\mathbf{x}_c)\ ;\ \boldsymbol{\Sigma}' = \mathbf{R}_{CO}\,\Sigma\,\mathbf{R}_{CO}^T \quad (2)$$

For a perspective camera, a camera space Gaussian (Eq. 2) is transformed to the image space ($\mathbb{I}$) using the perspective projection ($\boldsymbol{\Pi}$), which is not affine. However, [32] introduces a locally affine assumption around a camera coordinate $\mathbf{x}_c^*$ using first-order Taylor expansion in Eq. 3. Using this approximation the image space projection is a 2D Gaussian given by Eq. 4.

$$\boldsymbol{\Pi}^*(\mathbf{x}_c) = \boldsymbol{\Pi}(\mathbf{x}_c^*) + \mathbf{J}_\Pi^*(\mathbf{x}_c - \mathbf{x}_c^*) \quad (3)$$

$$\mathcal{G}_{\boldsymbol{\Sigma}'}(\boldsymbol{\Pi}^{-1}(\mathbf{u})) = \frac{1}{|\mathbf{J}_\Pi^{*-1}|}\mathcal{G}_{\boldsymbol{\Sigma}''}(\mathbf{u}) \quad (4)$$

$$\boldsymbol{\Sigma}'' = \mathbf{J}_\Pi^*\,\boldsymbol{\Sigma}'\,\mathbf{J}_\Pi^{*T} = \mathbf{J}_\Pi^*\,\mathbf{R}_{CO}\,\boldsymbol{\Sigma}\,\mathbf{R}_{CO}^T\,\mathbf{J}_\Pi^{*T}$$

where, $\mathbf{J}^*_{\Pi}$ is the jacobian of the projective transform linearized at camera space coordinates $\mathbf{x}^*_c$ and $\mathbf{u}$ are the image space coordinates.

The contribution of each Gaussian to a pixel is dependent on the order in which they appear along a ray cast from that pixel. For each pixel, the $K \in \mathbb{N}^+$ overlapping Gaussians are sorted from front to back. Then, the intensity or color ($\mathbf{c}_p$) of each 2D pixel location ($\mathbf{u} \in \mathbb{R}^2$) in the image is obtained by alpha-blending the contributions of the K Gaussians as in Eq. 5, following the approximations to the volumetric rendering equations in [32].

$$\mathbf{c}_p(\mathbf{u}) \;=\; \sum_{k=0}^{k\leq K} \mathbf{c}^k \alpha^k \prod_{j=0}^{k-1} (1 \,-\, \alpha^j) \tag{5}$$

$$\alpha^k = o^k \mathcal{G}^k_{\mathbf{\Sigma}''}(\mathbf{u})$$

where, α^k is the contribution of the k^{th} Gaussian to the pixel intensity. This contribution is computed by decaying the assigned opacity (o^k) by the 2D Gaussian ($\mathcal{G}_{\mathbf{\Sigma}''}$).

As the 3D location of each Gaussian is known, we can similarly render a depth image using per-point z distance in the camera frame:

$$d_p(\mathbf{u}) \;=\; \sum_{k=0}^{k\leq K} z^k \alpha^k \prod_{j=0}^{k-1} (1 \,-\, \alpha^j) \tag{6}$$

This splatting and blending process is fully differentiable and does not use neural elements. The image formation model in Eq. 5 is similar to the image formation model in Neural Radiance Fields [1]. However, NeRFs implicitly represent free space inside a volume, requiring an expensive sampling process along the ray. On the other hand, the forward rendering process for Gaussians automatically ignores free-space regions in 3D along the viewing direction. As a result, rendering 3D Gaussians is considerably faster than a NeRF.

C. Reconstruction and Tracking

Assuming no constraints on the relative motion between the camera and the target object, our goal is to incrementally incorporate new information revealed in an image sequence to reconstruct and track a specific object in the scene. Consider the rendering process from the previous section, encapsulated as:

$$\mathbf{I}_r = \mathcal{R}(\boldsymbol{\theta}, \mathbf{T}_{CO}) \tag{7}$$

where, $\mathcal{R}$ is the renderer that projects a set of 3D Gaussians representing the object to a rasterized image ($\mathbf{I}_r$). $\boldsymbol{\theta}$ are the object parameters obtained by concatenating the per-point Gaussian parameters: $\boldsymbol{\theta}^i \;=\; [\boldsymbol{\mu}^i, \mathbf{\Sigma}^i, \mathbf{c}^i, o^i]$ for $i \;=\; 0, 1, 2, ..., N$. Since the renderer is fully differentiable, we can compute and propagate gradients from image pixels to the object and camera parameters. We use Adam [33] to optimize the Gaussian and pose parameters.

To reconstruct the object by optimizing the object parameters $\boldsymbol{\theta}$, we require multi-view information to avoid overfitting the representation on restricted regions of the object. On the other hand, to optimize the pose, we only require the information from the current frame to establish the frame-to-model association. Consequently, we separate optimization into two stages that are executed in alternate frames of the incoming stream. During reconstruction, we hold camera parameters constant while optimizing the object parameters. The optimization fits the object parameters to a weighted combination of photometric loss for the color image and L1 loss for the depth image:

$$L_{recon} = (1 \,-\, \lambda)(L_1)_{color} + \lambda(L_{SSIM})_{color} + \beta(L1)_{depth} \tag{8}$$

where, L_1 is the averaged pixel-wise L1 loss between the observed image and the image with a pose $\mathbf{T}^n_{CO}$ for the n^{th} image. L_{SSIM} is the structural similarity index loss based on [34]. Unlike scene-reconstruction applications of Gaussian splatting, it is computationally impractical for our application to optimize the object representation using all the images available. Therefore, we manage a fixed window of W keyframes to optimize the object representation. Like SLAM pipelines, we aim to select keyframes that are most meaningful for multi-view constraints. To accomplish this we use up to W_{max} views with farthest angular distance between them and always include the current and the previous view.

For Tracking, the ground truth pose of the first frame is used to establish a canonical frame, if available. Otherwise, it is initialized with a random rotation and centered at the centroid of the partial point cloud de-projected from the first depth image. For the sake of optimization, we reduce the $SE(3)$ pose to a vector $\mathbf{x}_{CO} = [\mathbf{t}_{CO}, \mathbf{q}_{CO}]^T$, where $\mathbf{q}_{CO} = [q_w,\, q_x,\, q_y,\, q_z]$ is the unit quaternion representing relative orientation. The pose is optimized using:

$$L_{track} = (L_1)_{color} \;+\; \beta(L_1)_{depth}$$

The implementation is done using PyTorch using the 3DGS CUDA rasterizer implementation [6] to obtain the gradients of the loss with respect to the object parameters- $\partial\mathcal{L}/\partial\boldsymbol{\theta}$. The gradients of the loss with respect to the pose parameters $\mathbf{x_{CO}}$ are obtained via auto-differentiation. We project the Gaussians to the camera frame outside the CUDA rasterizer which provides the gradient flow from the analytical gradients $\partial\mathcal{L}/\partial\boldsymbol{\theta}$. At each new frame, we initialize the pose estimate by linearly propagating the pose as in Eq. 10, where t is the time index.

$$\mathbf{t}^{t+1} \;=\; \mathbf{t}^t + (\mathbf{t}^t - \mathbf{t}^{t-1}) \tag{9}$$

$$\mathbf{q}^{t+1} \;=\; \mathbf{q}^t \otimes (\mathbf{q}^t \otimes \bar{\mathbf{q}}^{t-1}) \tag{10}$$

where, For the tracking iteration, this pose is used as the initial guess and refined through optimization. On the other hand, this extrapolated pose stays fixed for the reconstruction iteration. The operations $\otimes$ and $\bar{\mathbf{q}}$ are quaternion multiplication and inverse respectively. We use 120 optimization steps for reconstruction and 80 steps for tracking.

D. Initialization, Addition, and Removal of 3D Gaussians

We use the first depth image to initialize the 3D Gaussians ($\mathcal{G}_\Sigma$) in the object space. The centers ($\boldsymbol{\mu^i}$) are centered around the point cloud obtained by de-projecting the depth image and the variances are initialized with 0.001. The color is initialized from the corresponding color image pixel value and the opacity is set to 0.5. For the camera-relative pose, if the ground-truth pose is available, it is provided only in the first frame to set the object reference frame. Otherwise, it can be arbitrarily initialized relative to the point cloud obtained from the first depth image.

As new regions of the object are revealed, we add more points to represent the previously unseen regions. Instead of having a fixed amount of points fit the new observations, this is advantageous for two reasons. First, the optimization is more convenient as a smaller number and size of steps are required to fit the new points. Second, the fidelity of the local geometry can be maintained. We use the difference between the observed depth image and the rendered depth image to add these points at every new frame. A point is added if the difference between the values in the rendered depth image and the observed depth image is larger than 10% of the difference between maximum and minimum depth. To project the points to the object frame, we need to use the latest available transformation $\mathbf{T}_{CO}$ between the object and the camera frame. As we add more points, the number of optimization parameters grows proportionally. This slows down the optimization steps. More importantly, the points are added with high density at each frame and may be redundant to the object representation. As a result, we prune points for which opacity drops below 0.6 during reconstruction. Together, this process of addition and removal of points balances the need for fidelity and ease of optimization against the computational complexity.

IV. Experiments

Our method directly optimizes the object representation and camera pose online. Consequently, no pre-training or large-scale data curation is necessary and it can be tested directly on sequential RGB-D data of any unknown object.

A. Data Generation

We test the performance of our method on a custom dataset of image sequences of 10 space objects from ESA's science fleet[1] as shown in Fig. 4. For consistency, we simulate the same relative trajectory for all the spacecraft models and the perspective camera model. The simulated relative motion is a spherical spiral as shown in Fig. 4 around a sphere of radius 16m. The trajectory allows for spacecraft to incrementally reveal new views and to have varying photometric shifts between time steps along the trajectory. We note that there are no limitations on the nature of this trajectory, as long as the spacecraft is sufficiently resolved in the image and the photometric shifts between two subsequent frames are not large and abrupt. We simplify the data generation process by scaling down the spacecraft models appropriately to fit the FOV, so we can test in under the same trajectory. Further, we simplify the spacecraft's appearance by reducing the material specularity. Since we ignore the view-based appearance model in 3DGS, our method in its current form cannot deal with flares and sharp specular reflections. We use Nvidia Omniverse Isaac Sim with the ray-tracing render engine to generate a sequence of 1000 synthetic images along the trajectory for each object. Through the replicator module, we generate an RGB image, an aligned depth image, a segmentation image, and the relative ground-truth pose for each time step of the trajectory. During data-loading, we add random noise in the color and depth images with a standard deviation of 0.2 (intensity) and 0.025m respectively. Furthermore, we add edge bleeding artefacts in the depth image to emulate a stereo depth output.

[1] https://scifleet.esa.int/

TABLE I: Chamfer distance evaluation

Spacecraft	CD@0	CD@250	CD@500	CD@1000
CHEOPS	1.18±0.02	0.61±0.02	0.54±0.02	**0.54±0.02**
SOHO	1.96±0.00	1.53±0.03	**1.04±0.14**	1.27±0.16
Giotto	0.62±0.01	0.05±0.01	0.02±0.00	**0.01±0.00**
Herschel	0.99±0.01	0.34±0.01	0.26±0.01	**0.22±0.00**
LISA Pathfinder	1.04±0.01	0.57±0.01	0.50±0.01	**0.49±0.01**
Euclid	0.61±0.01	0.19±0.01	0.17±0.00	**0.16±0.01**
Proba 2	0.78±0.01	0.45±0.01	**0.38±0.01**	0.39±0.01
Proba 3 CSC	1.43±0.01	1.11±0.02	0.94±0.12	**0.83±0.08**
Ulysses	**9.94±0.01**	31.97±1.82	34.60±1.31	33.80±1.57
Integral	2.35±0.01	1.96±0.01	1.53±0.16	**1.23±0.27**

B. Evaluation

We evaluate the performance of our approach by running optimization for reconstruction and tracking in alternate frames. The reconstruction performance is measured by the bi-directional chamfer distance between the ground-truth surface point clouds and the point cloud obtained from the online reconstruction. The chamfer distance metric quantifies the similarity between two point clouds by averaging the pair-wise distances of the closest points. This is given by:

$$\mathrm{CD}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2n_1}\sum_{i=1}^{n_1} |x_1^i - x_2^{i*}| + \frac{1}{2n_2}\sum_{j=1}^{n_2} |x_2^j - x_1^{j*}| \quad (11)$$

where, x^{i*} and x^{j*} are the nearest neighbour points in the other point cloud. We compute chamfer distance over 20000 points by uniformly sampling from each point cloud.

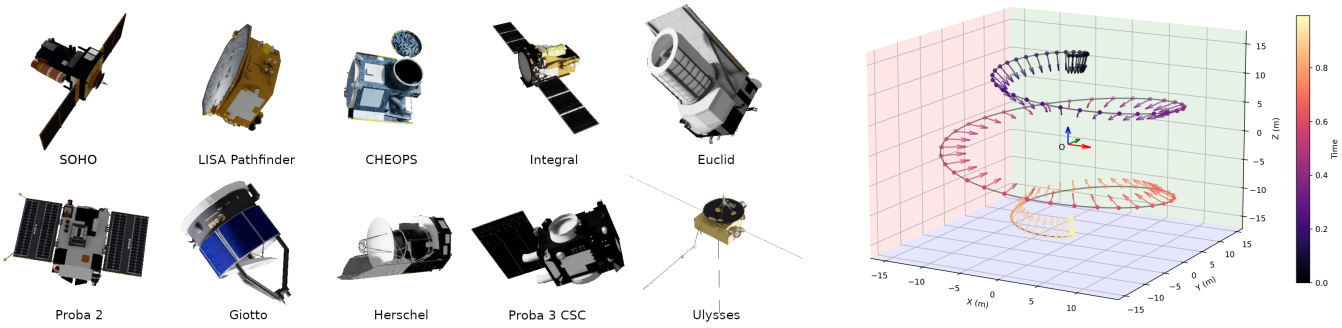
The tracking error is simply computed as the translation error and rotation error to the annotated ground truth pose:

$$\begin{aligned} t_{err} &= ||\mathbf{t}_{CO} - \mathbf{t}^*_{CO}||_2 \\ \theta_{err} &= 2\ \arccos(\mathbf{q_{CO}} \otimes \bar{\mathbf{q}}^*_{\mathbf{CO}})_w \end{aligned} \quad (12)$$

where, $\mathbf{t}^*_{CO}$ and $\mathbf{t}^*_{CO}$ are ground truth position and orientation.

V. Results and Discussion

While the reconstruction and tracking performance are coupled in our method, it is helpful to analyze their evolution



(a) (b)

Fig. 4: (a) **Dataset objects**: ESA science fleet spacecraft models used to generate the dataset. (b) **Test Trajectory**: Visualization of the camera trajectory relative to the target spacecraft, whose body frame is denoted by the axes at O. The arrows show the boresight direction at each point.

over the trajectory separately at first. Table I shows the variation of the bi-directional chamfer distance between the reconstructed point cloud and the reference point cloud for each object. As newer views of the target object are revealed, the proposed method generally leads to consistent improvement in the accuracy and completeness of the reconstructed 3D model. The convergence of the reconstruction error is sensitive to the pose tracking error. When new 3D Gaussians are added, the last estimated relative pose is used to project them from the camera frame to the object frame. Consequently, larger pose tracking errors can lead to diverging reconstruction errors. This can be seen in the case of SOHO, Proba 2, and Ulysses models where the chamfer distance at the last step is not the lowest. In the case of Ulysses, the reconstruction errors present a clear sign of early tracking drift leading to sharp divergence. Another notable observation is the much lower improvement in error between the 500^{th} and the 1000^{th} step. Depending on the spacecraft geometry, most regions of the spacecraft can be revealed by step 500. This effect is dominated by stray points in the reconstruction far away from the local surfaces of the object, resulting from the noise in the depth images. This effect can be observed in the initial point cloud shown in Fig. 5 as well as the final reconstructions shown in Fig. 1.

The tracking results for each of the spacecraft in terms of translation and rotation error are shown in Fig. 5. For clarity, we separate the tracking error for the first 500 steps from that of the entire 1000-step sequence. We outline three direct observations. First, our method allows accurate tracking of diverse target objects over short durations. Second, over 500 steps, the tracking error for 6/10 spacecraft models is maintained under the tight bounds of 0.5 m in translation and 10 degrees in rotation. Finally, over the entire 1000 steps, the tracking error for 6/10 objects diverges uncontrollably, while that of 4/10 objects remains within a reasonable error bound. Together with the reconstruction results, these observations provide the preliminary validation of our incremental approach using 3D Gaussian representations to track unknown objects. On a system with 12th Gen Intel Core i7-12800H CPU and Nvidia RTX 3080 Ti Laptop GPU, it takes approximately 1.4s and 1.1s to optimize the reconstruction (120 steps) and pose (80 steps) respectively.

The experiments demonstrate that the proposed approach and its application to unknown spacecraft tracking are effective. However, the current implementation is limited in robustness and practicality by several factors that can be addressed in future works. The foremost limitation of the approach is the tracking drift over a longer duration. The key factor behind this is the lack of a global pose optimization scheme. Since the pose optimization only considers a single frame, any tracking error cascades negatively. Remember that we use the latest pose available in the previous iteration for optimizing the object parameters and adding newer points. While the repeated reconstruction compensates the tracking error to a certain extent, it diverges beyond a certain margin of error. We observe two main sources that cause this error: (1) Visual ambiguity in regions with low-intensity values and (2) view-dependent changes in appearance. The first source is evident from the tracking results of Integral, Proba-2, and Proba-3-CSC spacecraft which have large regions of surfaces that are dark and do not provide enough contribution to the loss. The tracking drift starts rising noticeably when large parts of the spacecraft observable in the image have close to zero intensity. The second factor is observed in the case of Ulysses where the simulated images contain sharp view-dependent artefacts on the long booms of the spacecraft from data generation.

The improvement of the loss function to deal with low-intensity regions can alleviate abrupt tracking errors seen in the experiments. On the other hand, better keyframe management and pose graph optimization can solve the long-term tracking drift resulting from accumulating errors. Further, naive extrapolation of the relative pose used directly to reconstruct the object and add more Gaussians limits the magnitude of pose shift that can be handled between two consecutive images. From a design perspective, the requirement of depth images can be relaxed by initializing points with a high variance around the existing points based on the color image only. The requirement of a segmentation mask at each frame may also be relaxed by only taking the

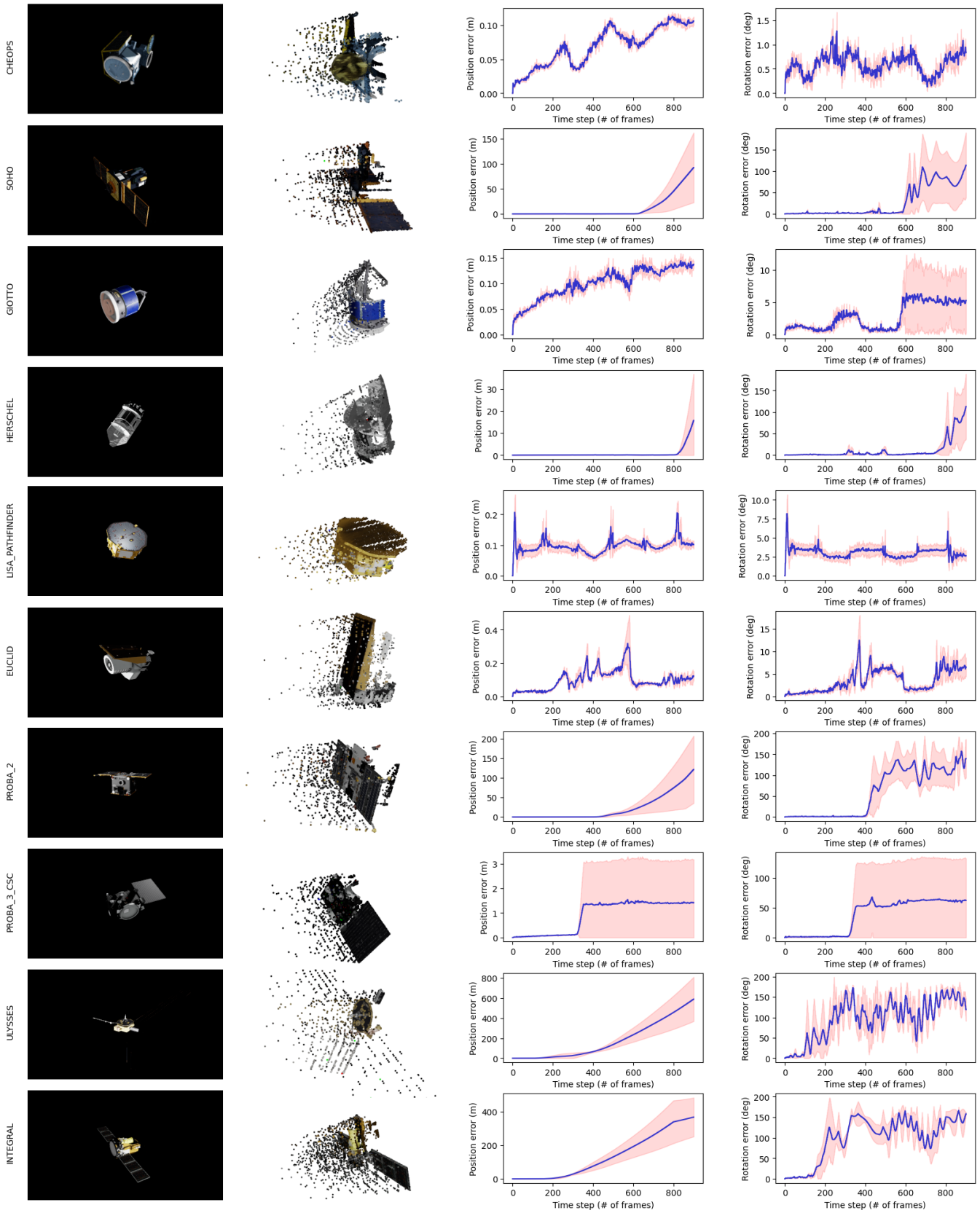


Fig. 5: **Results**: Pose tracking errors over 1000 frames for 10 synthetic spacecraft models of diverse geometry and textures. A sample input image and the point cloud of the first RGB-D frame for each spacecraft are shown on the left.

segmentation mask in the first image and then using the rendered image to mask the future frames. An important factor to include in future work is the efficient representation of specularity, which may be done using spherical harmonics coefficients for the Gaussians. The computation time can be improved by using a second-order optimizer. Finally, the analysis can be extended to a comprehensive range of relative trajectories including active maneuvers to assess the generalizability of our approach.

VI. Conclusions

This work proposes a novel approach for incremental reconstruction and pose tracking of unknown dynamic objects using 3D Gaussian representation. It assumes no prior knowledge of the object or its motion and requires no pre-training or neural elements. Relying on a differentiable rendering framework that is fast, it is more suitable for online applications than other contemporary methods. The effectiveness of the method is validated on the task of tracking an unknown dynamic spacecraft. On a custom dataset of ten spacecraft with diverse geometry and appearance, the tracking accuracy over shorter duration is less than 0.5m in translation and 10 deg in rotation. The aspects of longer duration tracking drift are discussed along with recommendations for improvements, paving the way forward for generalizable object-centric perception in space robotics.

Acknowledgments

This work is supported by the Fonds National de la Recherche (FNR) Industrial Fellowship grant (15799985) and Redwire Space Europe.

References

[1] B. Mildenhall *et al.*, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] Y. Labbé *et al.*, "Megapose: 6d pose estimation of novel objects via render & compare," *arXiv preprint arXiv:2212.06870*, 2022.

[3] L. Pauly *et al.*, "A survey on deep learning-based monocular spacecraft pose estimation: Current state, limitations and prospects," *Acta Astronautica*, 2023.

[4] L. P. Cassinis *et al.*, "Evaluation of tightly-and loosely-coupled approaches in cnn-based pose estimation systems for uncooperative spacecraft," *Acta Astronautica*, vol. 182, pp. 189–202, 2021.

[5] T. H. Park and S. D'Amico, "Rapid abstraction of spacecraft 3d structure from single 2d image," in *AIAA SCITECH 2024 Forum*, 2024, p. 2768.

[6] B. Kerbl *et al.*, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023.

[7] "Esa science satellite fleet." [Online]. Available: https://scifleet.esa.int/

[8] R. Opromolla *et al.*, "A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations," *Progress in Aerospace Sciences*, vol. 93, pp. 53–72, 2017.

[9] B. J. Naasz *et al.*, "The hst sm4 relative navigation sensor system: overview and preliminary testing results from the flight robotics lab," *The Journal of the Astronautical Sciences*, vol. 57, no. 1-2, pp. 457–483, 2009.

[10] S. D'Amico, M. Benn, and J. L. Jørgensen, "Pose estimation of an uncooperative spacecraft from actual space imagery," *International Journal of Space Science and Engineering 5*, vol. 2, no. 2, pp. 171–189, 2014.

[11] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[12] C. Harris, M. Stephens *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[14] S. Sharma and S. D'Amico, "Neural network-based pose estimation for noncooperative spacecraft rendezvous," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 6, pp. 4638–4658, 2020.

[15] T. H. Park *et al.*, "Speed+: Next-generation dataset for spacecraft pose estimation across domain gap," in *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 1–15.

[16] K. Black *et al.*, "Real-time, flight-ready, non-cooperative spacecraft pose estimation using monocular imagery," *arXiv preprint arXiv:2101.09553*, 2021.

[17] J. I. B. Pérez-Villar, Á. García-Martín, and J. Bescós, "Spacecraft pose estimation based on unsupervised domain adaptation and on a 3d-guided loss combination," in *European Conference on Computer Vision*. Springer, 2022, pp. 37–52.

[18] T. H. Park and S. D'Amico, "Online supervised training of spaceborne vision during proximity operations using adaptive kalman filtering," *arXiv preprint arXiv:2309.11645*, 2023.

[19] T. H. Park and S. D'Amico, "Adaptive neural-network-based unscented kalman filter for robust pose tracking of noncooperative spacecraft," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 9, pp. 1671–1688, 2023.

[20] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[21] C. Forster *et al.*, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[22] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[23] S. Izadi *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.

[24] L. Ma *et al.*, "Simultaneous localization, mapping, and manipulation for unsupervised object discovery," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1344–1351.

[25] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[26] E. Sucar *et al.*, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[27] M. M. Johari, C. Carta, and F. Fleuret, "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 408–17 419.

[28] B. Wen *et al.*, "Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 606–617.

[29] H. Huang *et al.*, "Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras," *arXiv preprint arXiv:2311.16728*, 2023.

[30] H. Matsuki *et al.*, "Gaussian splatting slam," *arXiv preprint arXiv:2312.06741*, 2023.

[31] N. Keetha *et al.*, "Splatam: Splat, track & map 3d gaussians for dense rgb-d slam," *arXiv preprint arXiv:2312.02126*, 2023.

[32] M. Zwicker *et al.*, "Ewa splatting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223–238, 2002.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] Z. Wang *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.