Rapid Abstraction of Spacecraft 3D Structure from Single 2D Image

Tae Ha Park* and Simone D'Amico†
Stanford University, Stanford, CA 94305

This paper presents a Convolutional Neural Network (CNN) to simultaneously abstract the 3D structure of the target space resident object and estimate its pose from a single 2D image. Specifically, the CNN predicts from a single image of the target a unit-size assembly of superquadric primitives which can individually describe a wide range of simple 3D shapes (e.g., cuboid, ellipsoid) using only a few parameters. The proposed training pipeline employs various types of supervision in both 2D and 3D spaces to fit an assembly of superquadrics to man-made satellite structures. In order to avoid numerical instability encountered when evaluating superquadrics, this work proposes a novel, numerically stable algorithm based on dual superquadrics to evaluate a point on the surface of and inside a superquadric for all shape parameters. Furthermore, in order to train the CNN, this work also introduces a novel dataset comprising 64 different satellite models and 1,000 images, binary masks and pose labels for each model. The experimental studies reveal that the proposed CNN can be trained to reconstruct accurate superquadric assemblies when tested on unseen images of known models and capture high-level structures of the unknown models most of the time despite having been trained on an extremely small dataset.

I. Introduction

Recent years have seen tremendous growth in interest in applications of Machine Learning (ML) and Neural Networks (NN) to spaceborne Guidance, Navigation and Control (GNC) problems. One such area of application is vision-only relative navigation which aims to estimate and track the position and orientation (i.e., pose) of a non-cooperative Resident Space Object (RSO) using a monocular camera on a servicer spacecraft [1]. Not only is a monocular camera an attractive choice due to low Size-Weight-Power-Cost (SWaP-C) requirements of on-board avionics, but good vision-only performance can provide a strong baseline in case of sensor fusion with differential GNSS [2], Light Detection and Ranging (LIDAR), etc., and increased redundancy through multiple cameras.

Most of the recent literature on monocular pose estimation in space assumes the knowledge of the target RSO's structure and geometry, which is a viable assumption for missions with specified client targets, such as inspection, re-fueling and general servicing. The availability of the target's 3D model has enabled: (1) generation of large-scale datasets comprising synthetic and real images with corresponding pose labels to train and validate various NN models [3–8]; and (2) design of Convolutional Neural Network (CNN) architectures which leverage not only the pose labels but also the target's surface keypoints which are pre-selected for detection [3, 9–16]. For example, the SPEED [3], SPEED+ [4] and SHIRT [5] datasets consist of synthetic images of the Tango spacecraft from the PRISMA mission [17] rendered with OpenGL and Hardware-in-the-loop (HIL) images of a mockup Tango model captured with the Testbed for Rendezvous and Optical Navigation (TRON) facility at Stanford's Space Rendezvous Laboratory (SLAB) [18]. These open-source benchmark datasets made possible international competitions [19, 20] which focus not only on ML-based pose estimation algorithms but also on tackling the domain gap problem that arises due to inherent difference between synthetic and spaceborne imageries.

Moving forward, the assumption of availability of the target 3D model must be relaxed for two reasons: it becomes tedious to generate a dataset, train a CNN, and test it in a facility such as TRON for every single mission with new targets; most importantly, it cannot be applied to missions with no a priori knowledge of the target's shape and structure, such as active debris removal. In such scenarios, it becomes desirable to autonomously and simultaneously characterize the target's shape and perform pose estimation with respect to the reconstructed model. Existing literature on monocular-based shape characterization and pose estimation of arbitrary RSO relies largely on the reconstruction of a point cloud by detecting salient features such as Harris corners [21], SIFT [22] and ORB [23] features. The 3D

*Ph. D. Candidate, Department of Aeronautics & Astronautics, 496 Lomita Mall. AIAA Student Member.
†Associate Professor, Department of Aeronautics & Astronautics, 496 Lomita Mall. AIAA Associate Fellow.

coordinates of these features in the model frame are often adaptively refined via Simultaneous Localization and Mapping (SLAM) [24] or Structure-from-Motion (SfM). For example, popular SLAM approaches would track the features as part of the state vector of the Kalman or particle filter, adaptively updating their coordinates based on the underlying dynamics of the system and measured feature locations [25].

The primary contribution of this paper is to showcase for the first time the capability of a CNN to abstract a coarse 3D shape of an unknown target satellite from a single 2D image and simultaneously estimate its pose. The motivation is that coarse shape information will be able to kick-start the downstream SLAM or SfM algorithm, which often requires a long measurement period for a point cloud model to converge to a steady state. In fact, it is impossible for conventional algorithms to extract meaningful shape information from a single image due to the limits of perspective projection. On the other hand, shape abstraction can be achieved by a single forward pass of a NN which is a computationally cheap operation. To that end, this work proposes to represent the target's 3D shape as an assembly of superquadric primitives which can represent a wide range of geometric shapes with only a handful of parameters. Specifically, allowing for linear tapering of the primitive along one of its axes, the entire assembly of M primitives can be compactly parametrized by just $16M$ parameters. This work proposes a supervised training that utilizes both 3D mesh and 2D binary masks as labels via differentiable rendering. Moreover, a novel algorithm based on dual superquadrics is proposed to evaluate the superquadric primitives (e.g., points on the surface, inside-outside function) free of any numerical instability.

The secondary contribution of this paper is a novel dataset consisting of 3D models and imageries of 64 different satellites and spacecraft. Specifically, each 3D model is accompanied by 1,000 images and binary masks of the target rendered in a high-fidelity space scene of Unreal Engine (UE)*. The dataset is divided up into training, validation and test sets, such that validation sets are used to evaluate the model's generalization capability on unseen images of known targets (i.e., seen during training), whereas test sets help evaluate it on images of unknown targets (i.e., unseen during training). When the proposed CNN is tested on the validation data, it shows that the CNN is able to predict superquadric assemblies that match the ground-truth 3D mesh models. While it does not show such success on the unknown models in the test set, it can still reconstruct assemblies that share high-level, macroscopic structural similarities with the ground-truth models (e.g., two extended solar panels).

The rest of the paper is structured as follows. Section II provides an overview of the state-of-the-art approaches on vision-based 3D structure recovery using ML. Section III introduces the mathematics behind the shape representation using superquadrics, including a novel approach to sampling surface points and evaluating the occupancy using dual superquadrics. Then, Section IV details the proposed pipeline, including the CNN architecture and the loss functions. The following Section V introduces the dataset, followed by Section VI explaining the experiments and their results and analyses. Finally, the paper ends with the conclusion and ways forward in Section VII.

II. Related Work

ML-based 3D reconstruction from 2D images is an active area of research in computer vision. The core of existing approaches is how the 3D structure of the target can be represented so that it can be learned and predicted by NN. Early approaches discretized 3D model into N^3 voxels where N is the voxel resolution [26, 27]. As they are a natural extension of 2D image pixels, voxelized models can be readily integrated into CNN architectures. However, its memory footprint is cubic to the grid resolution, which makes it prohibitively expensive to capture fine details of the model. Some works sought to mitigate the computational burden by using octrees which selectively increase the resolution only near the region of interest [28, 29]; however, these multi-resolution methods are still computationally expensive and thus inappropriate for satellite on-board avionics.

Another representation is a point cloud which has been used extensively in traditional SLAM approaches. Point clouds have been studied as an input to NN for conventional ML tasks such as segmentation and classification [30] and as an output for exact 3D reconstruction of the target from a single view image [31, 32] and of generative models [33]. However, predicting points alone cannot ensure smooth surface reconstruction and is thus susceptible to outliers due to the lack of local connectivity information. On the other hand, another line of work reconstructs the 3D mesh model from a single image. In general, mesh provides additional connectivity information of vertices, allowing for high-quality reconstruction of the target model from any input [34, 35]. However, due to the difficulty of predicting both vertices and connectivity, many approaches deform a template uniform mesh structure (e.g., a tetrahedral mesh of uniformly spaced vertices in a unit cube) into the desired mesh model which in turn restricts its representable model genus.

Other works seek to implicitly represent the 3D model. One example is using an assembly of simple 3D geometric primitives. Tulsiani et al. [36] pioneered the approach by using a set of 3D cuboids. The approach was extended by

*https://www.unrealengine.com/en-US/

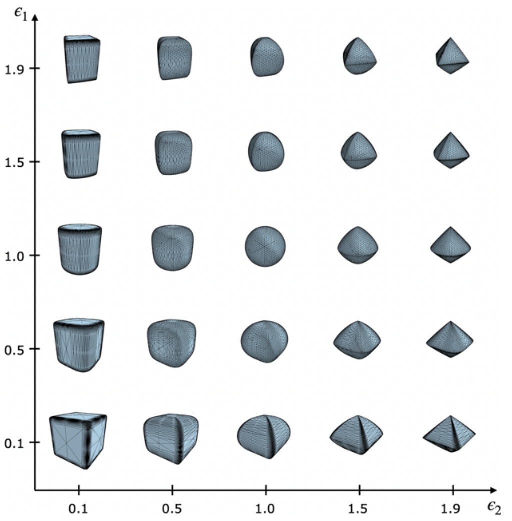


Fig. 1 Visualization of various superquadrics for different shape parameters (ϵ_1, ϵ_2).

Paschalidou et al. [37, 38] to utilize superquadric primitives which subsume various 3D geometric shapes such as cuboids, ellipsoids, spheres, cylinders and octahedra using a limited set of shape parameters. Other works employed different sets of implicit functions to represent 3D structures. For example, CvxNet [39] predicts each primitive formed by a set of hyperplanes which can be learned using signed distances. OccupancyNet [40] represents the target's 3D surface as a continuous decision boundary of a NN that is trained to predict the occupancy of an arbitrary 3D point. These implicit functions have shown to be able to accurately reconstruct the target model as a whole; therefore, they lack any parts-based information that the primitive assemblies can bring. On the other hand, primitives lack the capacity to represent complex shapes on their own due to limited degrees of freedom in their parametrization.

Finally, a new direction of research is emerging based on Neural Radiance Fields (NeRF) [41] which trains NN to render the target scene given the camera's position and viewing direction. While the aforementioned approaches are trained in advance using a large-scale dataset comprising various 3D models such as ShapeNet [42], NeRF is instead more suited for *online* reconstruction where it can collect multiple images from different viewpoints for in-situ training of the NeRF. Training the original NeRF was computationally expensive even on a state-of-the-art workstation GPU, it required known pose labels associated with each image, and its predictions were constrained to a static scene. Many works followed up to address those issues, such as InstantNGP [43] which allows instant training of NeRF on a single GPU, and D-NeRF [44] which accounts for the dynamical scene. Recently, Mergy et al. [45] and Mahendrakar et al. [46] applied various NeRF models to the problem of 3D reconstruction of unknown RSO from a set of unlabeled images. Specifically, Mahendrakar et al. [46] found that InstantNGP could be trained on an edge GPU device with flight heritage[†] in under 10 minutes. However, training NeRF required them to remove the image background and recover the pose labels via SfM based on COLMAP [47, 48], and they found that the quality of reconstructed scenes varied depending on the lighting condition and camera movement. This could be an issue when integrating NeRF into the spacecraft GNC pipeline since the prominent approaches leveraging NeRF for pose estimation [49] and SLAM [50] require making a direct comparison of the RGB pixel values between true and reconstructed scenes. Moreover, since NeRF is fundamentally a rendering function, it is not immediately clear how one can extract explicit 3D information about the target's structure for typical on-orbit servicing applications.

III. 3D Shape Representation as Superquadric Assembly

The applicability of the aforementioned approaches to space missions may vary significantly depending on the requirements of the dataset diversity and computational demands. On the one hand, primitive-based methods enable a compact representation of the target 3D structure using only a handful of parameters but may require a large dataset comprising many satellite models in order to learn meaningful features and reconstruct the 3D model of an unknown RSO. The operation based on primitive representations is also computationally light on-board since the reconstruction

[†]`https://www.militaryaerospace.com/commercial-aerospace/article/14234003/usc-la-jument-3u-cubesat`

is performed with a single forward pass of an already trained NN. On the other hand, NeRF-based methods do not require offline training and therefore do not need a large dataset of various spacecraft 3D models; however, they do require enough on-board computational capability and the availability of enough images of the target from different *known* poses for in-situ training of NeRF models. This work adopts the primitive-based method due to its compactness and straightforward application to conventional SLAM approaches, but all the other methods and their applicability to spaceborne vision-based rendezvous are interesting future research directions.

To that end, the proposed CNN is designed to take in a single 2D image at an unknown pose of the target and output an assembly of superquadrics [51] which best describes the overall structure of the target [37, 38] along with its global pose with respect to the camera. A superquadric is a parametric family of various shapes such as cuboids, ellipsoids, spheres, cylinders and octahedra as shown in Fig. 1. Specifically, a superquadric is parametrized by shape parameters, $\boldsymbol{\epsilon} \in \mathbb{R}^2$, and size parameters, $\boldsymbol{\alpha} \in \mathbb{R}^3$. Then, any point on the surface of a superquadric can be retrieved via

$$\boldsymbol{r}(\eta, \omega; \boldsymbol{\alpha}, \boldsymbol{\epsilon}) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ \alpha_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ \alpha_3 \sin^{\epsilon_1} \eta \end{bmatrix}, \quad \begin{matrix} -\pi/2 \le \eta \le \pi/2 \\ -\pi \le \omega \le \pi \end{matrix}. \tag{1}$$

On top of the shape and size parameters, the CNN also predicts the pose of each superquadric with respect to the model's global reference frame. Note that these poses define the position and orientation of each superquadric within an assembly and are separate from the global pose of the assembly with respect to the camera. The pose is parametrized by the 3D translation vector ($\boldsymbol{t} \in \mathbb{R}^3$) and the 6D rotation vector ($\boldsymbol{r} \in \mathbb{R}^6$) which has shown to outperform other parametrizations such as quaternions and Euler angles in rotation regression tasks due to its continuous property [52].

Superquadrics also allow implicit representation via the following inside-outside function

$$f(\boldsymbol{x}; \boldsymbol{\alpha}, \boldsymbol{\epsilon}) = \left[\left(\frac{x}{\alpha_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{\alpha_2} \right)^{\frac{2}{\epsilon_2}} \right]^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{\alpha_3} \right)^{\frac{2}{\epsilon_1}} \tag{2}$$

which is less than 1 if the point $\boldsymbol{x} = [x, y, z]$ lies inside the superquadric primitive and greater than 1 if it lies outside. The inside-outside function can be used, for example, to classify whether a point that lies inside the ground-truth mesh is also at a location internal to a superquadric primitive.

Finally, in order to introduce more interesting, asymmetric geometry, the primitives are linearly tapered along their z-axes according to

$$x' = \left(1 - \frac{k_1}{\alpha_3} z \right) x \tag{3a}$$

$$y' = \left(1 - \frac{k_2}{\alpha_3} z \right) y \tag{3b}$$

$$z' = z \tag{3c}$$

where $\boldsymbol{k} \in \mathbb{R}^2$ are the tapering parameters.

In summary, an m-th primitive can be represented as $\boldsymbol{\lambda}_m = [\boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m, \boldsymbol{t}_m, \boldsymbol{r}_m, \boldsymbol{k}_m] \in \mathbb{R}^{16}$, so an assembly composed of M primitives are parametrized by $16M$ parameters, allowing for a compact representation of any target RSO compared to alternatives such as point clouds.

A. Uniform Sampling of Surface Points using Dual Superquadrics

Equation 1 is useful when one wishes to sample points from the primitive surface, effectively converting the assembly into a point cloud. The converted point cloud is used in one of the loss functions by minimizing its distance to another point cloud sampled from the surfaces of the ground-truth 3D mesh (see Section IV.B.1). Here, the key is to sample the points as *uniformly* as possible from the superquadric surface.

The most prevalent sampling strategy is via parametric sampling of the elevation and azimuth angles (η, ω) [53]. However, existing sampling algorithms fail when $\epsilon \longrightarrow 0$. As shown in Fig. 1, $\epsilon \longrightarrow 0$ is the regime where the primitives begin to have straight edges, a property extremely common in structures such as solar panels and spacecraft buses. In fact, one can observe from Fig. 1 that as ϵ becomes very small, the sampled points tend to aggregate along the edges when (η, ω) are linearly sampled. Therefore, a novel strategy is desired for efficient, uniform and numerically stable sampling of the surface points on superquadrics, particularly those with small ϵ.

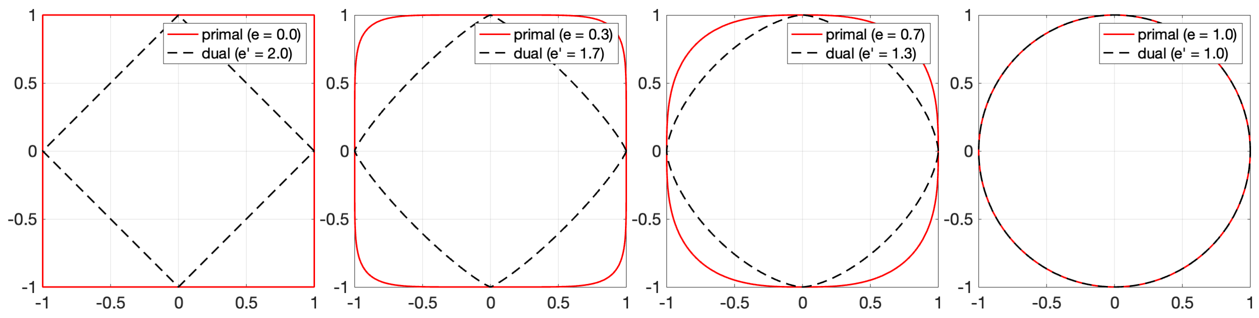


Fig. 2 Primal and dual superellipses for various ϵ values.

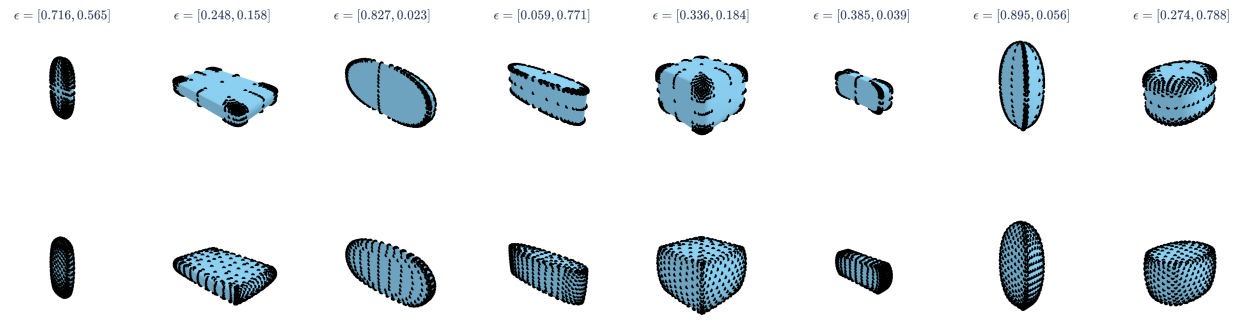

$\epsilon = [0.716, 0.565]$
$\epsilon = [0.248, 0.158]$
$\epsilon = [0.827, 0.023]$
$\epsilon = [0.059, 0.771]$
$\epsilon = [0.336, 0.184]$
$\epsilon = [0.385, 0.039]$
$\epsilon = [0.895, 0.056]$
$\epsilon = [0.274, 0.788]$

Fig. 3 Sampled points from various superquadrics using primal (*top*) and dual (*bottom*) approaches.

This work proposes a novel uniform sampling approach which draws inspiration from Liu et al. [54] who leverage the geometric similarity of *dual* superquadrics to avoid local optima while fitting superquadrics to 3D point cloud inputs. First, note that the surface evaluation function of Eq. 1 can be decomposed into

$$\boldsymbol{r}(\eta,\omega) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ \alpha_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ \alpha_3 \sin^{\epsilon_1}\eta \end{bmatrix} = \begin{bmatrix} \cos^{\epsilon_1}\eta \\ \alpha_3 \sin^{\epsilon_1}\eta \end{bmatrix} \otimes \begin{bmatrix} \alpha_1 \cos^{\epsilon_2}\omega \\ \alpha_2 \sin^{\epsilon_2}\omega \end{bmatrix} \tag{4}$$

where $\otimes$ denotes spherical product. Eq. 4 indicates that a point on a 3D superquadric is equivalent to a spherical product of two points on two 2D superellipses respectively parametrized by $(1, \alpha_3, \epsilon_1)$ and $(\alpha_1, \alpha_2, \epsilon_2)$.

Next, note that a 2D superellipse and a 3D superquadric defined by the shape parameters $\epsilon_i \in [0, 2]$ has a dual defined by $\epsilon_i' = 2 - \epsilon_i$. More precisely, a point on the surface of a dual superquadric at (η, ω) is equivalent to the scaled surface normal vector of the primal superquadric at the same coordinate and vice versa [55]. As shown in Fig. 2, the primal and dual superellipses are 45° apart, identical in shape when $\epsilon = 0$ or 1, and marginally different for all other values. Therefore, one can instead sample points from two dual superellipses, properly rotate and scale them, and then perform a spherical product to recover the approximate 3D surface points on the original primal superquadric. Specifically, the dual sampling approach for a 2D superellipse parametrized by (a_1, a_2, ϵ) consists of the following steps:

1) uniformly sample points from a unit-size dual superellipse defined by $\epsilon' = 2 - \epsilon$
2) rotate the points by 45°
3) scale the points with

$$\left\| \begin{bmatrix} \cos^{\epsilon_i'} 45^\circ \\ \sin^{\epsilon_i'} 45^\circ \end{bmatrix} \right\|_2^{-1} = 2^{(\epsilon_i'-1)/2} \tag{5}$$

4) apply size parameters (a_1, a_2) along each axis.

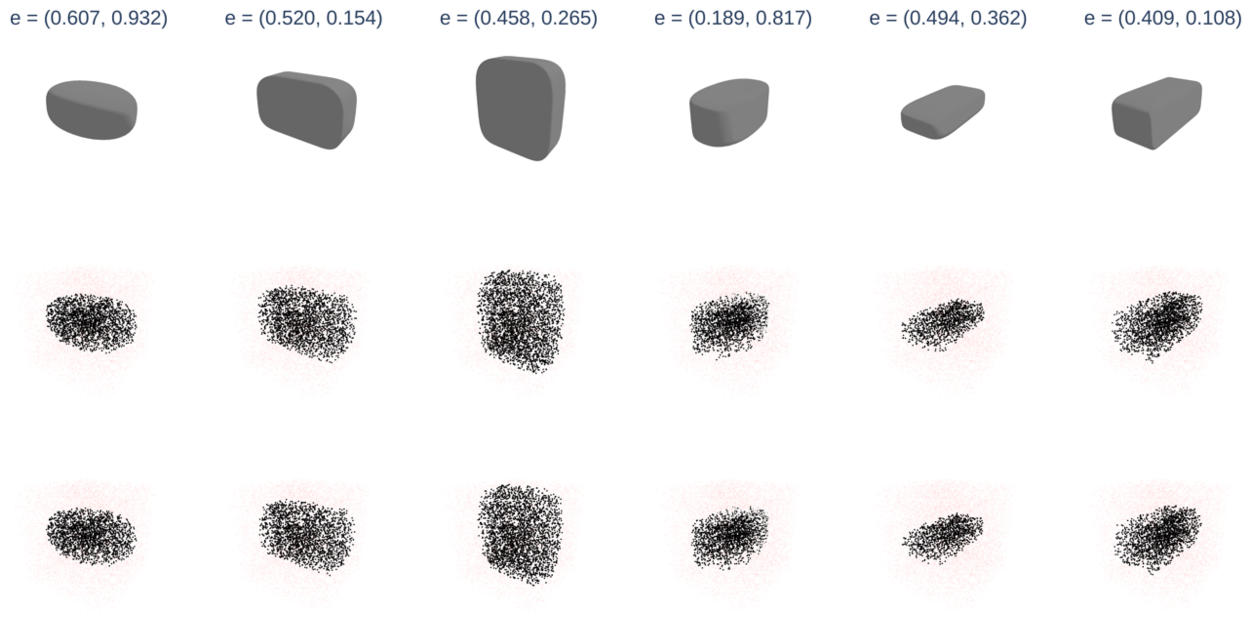


Fig. 4 Visualization of the ground-truth superquadric (*top*) and 10,000 random points labeled either inside (*black*) or outside (*red*) of the corresponding superquadric. The points are labeled using either the primal approach in Eq. 2 (*middle*) or the dual approach in Eq. 8 (*bottom*).

The above process for a single 2D superellipse can be consolidated into the following equation:

$$\boldsymbol{r}_{\text{2D}}(\theta; a_1, a_2, \epsilon) = \gamma \begin{bmatrix} a_1(\cos^{\epsilon'}\theta' + \sin^{\epsilon'}\theta') \\ a_2(-\cos^{\epsilon'}\theta' + \sin^{\epsilon'}\theta') \end{bmatrix}, \quad \text{where} \quad \theta' = \theta - 45^\circ, \ \gamma = 2^{\epsilon'/2-1} \tag{6}$$

so that a point on a 3D superquadric can be recovered via a spherical product as

$$\boldsymbol{r}_{\text{dual}}(\eta, \omega; \boldsymbol{\alpha}, \boldsymbol{\epsilon}) = \boldsymbol{r}_{\text{2D}}(\eta; 1, \alpha_3, \epsilon_1) \otimes \boldsymbol{r}_{\text{2D}}(\omega; \alpha_1, \alpha_2, \epsilon_2) \tag{7}$$

Equations 6, 7 are fully differentiable, and when $\epsilon \in [0, 1]$ and $\epsilon' = 2 - \epsilon \in [1, 2]$, the computation is free of numerical instability at $\epsilon \to 0$ suffered in the original formulation of Eq. 1. Figure 3 compares sampled points from various superquadrics using primal and dual methods, where the elevation and azimuth angles are obtained from the vertices of an icosphere. Observe that for superquadrics with one of the shape parameters near 0, primal sampling fails to yield uniformly distributed points, whereas dual sampling succeeds despite small ϵ. One limitation of the proposed dual sampling approach is that it does not account for sample density based on different size or taper parameters, which results in a higher density of samples along the shorter axis.

B. Numerically Stable Inside-Outside Function using Dual Superquadrics

The inside-outside function of Eq. 2 also suffers from numerical instability when $\epsilon \to 0$. However, Eq. 2 can also be converted into a numerically stable formulation by leveraging the dual superquadrics as well. Therefore, this work also presents a novel dual inside-outside function which is derived from Eqs. 6, 7 with full derivation available in Appendix. The dual inside-outside function (f_{dual}) is defined as

$$f_{\text{dual}}(\boldsymbol{x}; \boldsymbol{\alpha}, \boldsymbol{\epsilon}) = \phi(\phi(\bar{x}, \bar{y}, \epsilon_2), \bar{z}, \epsilon_1) = 1 \tag{8}$$

where $\bar{x} = x/(\alpha_1\gamma_1\gamma_2)$, $\bar{y} = y/(\alpha_2\gamma_1\gamma_2)$, $\bar{z} = z/(\alpha_3\gamma_1)$, γ_i is from Eq. 6, and

$$\phi(x, y, \epsilon) = \left[\left(\frac{x+y}{2}\right)^{\frac{2}{\epsilon'}} + \left(\frac{x-y}{2}\right)^{\frac{2}{\epsilon'}}\right]^{\frac{\epsilon'}{2}} \tag{9}$$

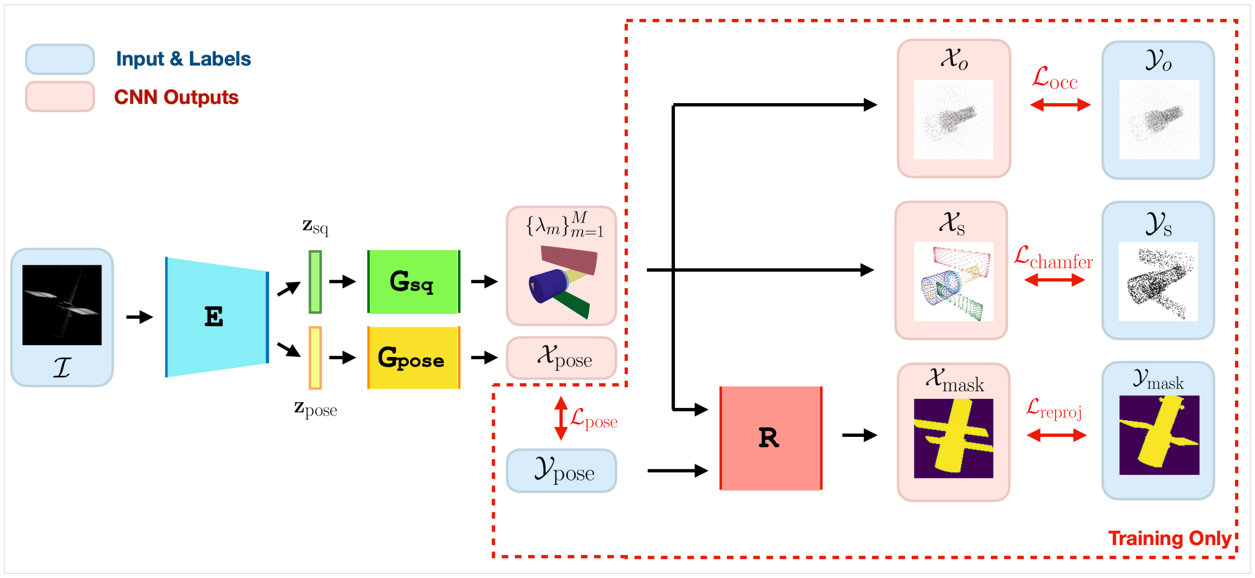


Fig. 5 CNN architecture and training pipeline. Various combinations of the loss functions are explored during the experiments.

Again, the formulation is free of numerical instability when $\epsilon \in [0, 1]$. The accuracy of the dual inside-outside equation is visualized in Fig. 4. A special case of the above formulations is when $\epsilon_1 = \epsilon_2 = 1$, and one can show that the dual inside-outside function simply becomes

$$f_{\text{dual}}(\boldsymbol{x}; \boldsymbol{\alpha}, \mathbf{1}) = \sqrt{\left(\frac{x}{\alpha_1}\right)^2 + \left(\frac{y}{\alpha_2}\right)^2 + \left(\frac{z}{\alpha_3}\right)^2} = 1 \tag{10}$$

which is an equation for an ellipsoid as confirmed in Fig. 1.

IV. Methodology

A. CNN Architecture

The proposed CNN architecture and training pipeline are visualized in Fig. 5. Training the proposed CNN involves an input image ($\mathcal{I}$) and binary mask ($\boldsymbol{\mathcal{Y}}_{\text{mask}}$) of the target spacecraft corresponding to the ground-truth pose $\boldsymbol{\mathcal{Y}}_{\text{pose}} = (\boldsymbol{r}_{\text{c}}, \boldsymbol{t}_{\text{c}})$. The subscript c denotes that the pose is of the entire assembly with respect to the *c*amera in order to differentiate from the pose of each primitive relative to the assembly. For the supervised training, N_{surf} points are sampled from the surface of the ground-truth mesh of the same spacecraft, forming a point cloud label ($\boldsymbol{\mathcal{Y}}_{\text{s}}$). Furthermore, N_{occ} points are randomly sampled within a unit cube and labelled according to whether they lie inside the ground-truth mesh or not ($\boldsymbol{\mathcal{Y}}_{\text{o}}$).

The architecture consists of a feature encoder (E) and two generators ($\boldsymbol{G}_{\text{sq}}, G_{\text{pose}}$). The feature encoder is an EfficientNet-B0 [56] connected to two separate fully connected layers that each output a 256-dimensional feature vector for superquadrics z_{sq} and pose z_{pose}, i.e., $(z_{\text{sq}}, z_{\text{pose}}) = E(\mathcal{I})$. The pose generator G_{pose} is a 3-layer network which outputs $\boldsymbol{\mathcal{X}}_{\text{pose}} = (\tilde{\boldsymbol{r}}_{\text{c}}, \tilde{\boldsymbol{t}}_{\text{c}}) = G_{\text{pose}}(z_{\text{pose}})$, the pose of the complete assembly with respect to the camera. The superquadric generator $\boldsymbol{G}_{\text{sq}} = \{G_{\text{sq}}^{(i)}\}_{i=1,\ldots,M}$ consists of M copies of the same 3-layer network, where the m-th network is in charge of predicting the m-th superquadric parameters, i.e., $\boldsymbol{\lambda}_m = G_{\text{sq}}^{(m)}(z_{\text{sq}})$.

During training, the superquadric parameter outputs $\boldsymbol{\lambda}$, and $\boldsymbol{\mathcal{Y}}_{\text{pose}}$ are used to render a binary silhouette of the target spacecraft ($\boldsymbol{\mathcal{X}}_{\text{mask}}$) via a differentiable renderer (R), i.e., $\boldsymbol{\mathcal{X}}_{\text{mask}} = R(\boldsymbol{\lambda}, \boldsymbol{\mathcal{Y}}_{\text{pose}})$. Here, $\boldsymbol{\mathcal{Y}}_{\text{pose}}$ is used instead of $\boldsymbol{\mathcal{X}}_{\text{pose}}$ so that the network can focus on learning more accurate superquadric shapes. During rendering, all primitives are converted into a mesh by transforming the vertices of an icosphere via Eq. 4 for improved rendering quality. This work uses the differentiable renderer of PyTorch3D [57] to allow gradient-based learning. Note that the renderer R is only used during training to provide additional supervision in 2D space.

B. Loss Functions

The proposed CNN is trained via supervised learning using normalized 3D mesh models, pose labels $\mathcal{Y}_{\text{pose}} = (\boldsymbol{r}_{\text{c}}, \boldsymbol{t}_{\text{c}})$ and binary masks ($\mathcal{Y}_{\text{mask}}$) corresponding to input images ($\mathcal{I}$) as sources of supervisory signals. The total loss is a weighted sum of various loss functions defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{chamfer}}(\boldsymbol{\mathcal{X}}_{\text{s}}, \boldsymbol{\mathcal{Y}}_{\text{s}}) + \mathcal{L}_{\text{occ}}(\boldsymbol{\mathcal{X}}_{\text{o}}, \boldsymbol{\mathcal{Y}}_{\text{o}}, \boldsymbol{\lambda}) + \mathcal{L}_{\text{reproj}}(\boldsymbol{\mathcal{X}}_{\text{mask}}, \boldsymbol{\mathcal{Y}}_{\text{mask}}) + \mathcal{L}_{\text{pose}}(\boldsymbol{\mathcal{X}}_{\text{pose}}, \boldsymbol{\mathcal{Y}}_{\text{pose}}) + \mathcal{L}_{\text{reg}}(\boldsymbol{\mathcal{X}}_{\text{o}}, \boldsymbol{\lambda}) \tag{11}$$

In Section VI, various combinations of the supervised losses are considered for different training sessions. This section provides details on each loss function and variables in Eq. 11.

1. Chamfer Loss

The bi-directional Chamfer loss ($\mathcal{L}_{\text{chamfer}}$) measures the squared distance between two sets of point clouds (S_1, S_2) defined as [37]

$$\mathcal{L}_{\text{chamfer}}(S_1, S_2) = \frac{1}{|S_1|}\sum_{\boldsymbol{x}\in S_1}\min_{\boldsymbol{y}\in S_2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \frac{1}{|S_2|}\sum_{\boldsymbol{y}\in S_2}\min_{\boldsymbol{x}\in S_1}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2, \tag{12}$$

i.e., the loss encourages both point clouds to subsume one another by minimizing the distance between the closest pairs of points. In this work, the predicted superquadric parameters ($\boldsymbol{\lambda}$) are converted to point clouds ($\boldsymbol{\mathcal{X}}_{\text{s}}$) by sampling N_{sq} surface points from each primitives using the dual sampling method (Eqs. 6, 9). The labels ($\boldsymbol{\mathcal{Y}}_{\text{s}}$) are obtained by sampling N_{surf} surface points from the ground-truth 3D mesh.

2. Occupancy Loss

Another supervised loss employed during the training is the occupancy loss, which is defined for N_{occ} random points $\boldsymbol{\mathcal{X}}_o$ within a unit cube as [38]

$$\mathcal{L}_{\text{occ}}(\boldsymbol{\mathcal{X}}_{\text{o}}, \boldsymbol{\mathcal{Y}}_{\text{o}}, \boldsymbol{\lambda}) = \sum_{(\boldsymbol{x}_{\text{o}}, y_{\text{o}})\in(\boldsymbol{\mathcal{X}}_{\text{o}}, \boldsymbol{\mathcal{Y}}_{\text{o}})} \mathcal{L}_{\text{BCE}}\left[\sigma\left(s\mathcal{G}(\boldsymbol{x}_{\text{o}}; \boldsymbol{\alpha}, \boldsymbol{\epsilon})\right), y_{\text{o}}\right] \tag{13}$$

where $\mathcal{L}_{\text{BCE}}$ is the binary cross-entropy loss function, $y_{\text{o}} \in \{0, 1\} \in \boldsymbol{\mathcal{Y}}_{\text{o}}$ is the occupancy label indicating whether $\boldsymbol{x}_{\text{o}}$ lies inside the ground-truth 3D mesh ($y_{\text{o}} = 1$) or not ($y_{\text{o}} = 0$), and $\mathcal{G}(\boldsymbol{x}_{\text{o}}; \boldsymbol{\alpha}, \boldsymbol{\epsilon}) = \min_{m\in\{1,\ldots,M\}} g(\boldsymbol{x}_o; \boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m)$. Here, $g(\boldsymbol{x}_{\text{o}}; \boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m) : \mathbb{R}^3 \to [0, 1]$ denotes the *occupancy function*, where a greater output denotes a higher likelihood that the point $\boldsymbol{x}_{\text{o}}$ resides inside the primitive. The occupancy function is defined as [38]

$$g(\boldsymbol{x}_{\text{o}}; \boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m) = \sigma(s(1 - f_{\text{dual}}(\boldsymbol{x}_{\text{o}}; \boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m))), \tag{14}$$

where s is the sharpness parameter, and $\sigma(\cdot)$ is the sigmoid function. Note that the dual inside-outside function (f_{dual}) is always used for numerical stability. In summary, the occupancy loss in Eq. 13 makes the CNN classify whether a given 3D point that is internal to a ground-truth mesh is also inside the predicted primitive assembly or not.

3. Reprojection Loss

The reprojection loss compares the predicted silhouette $\boldsymbol{\mathcal{X}}_{\text{mask}} = R(\boldsymbol{\lambda}, \boldsymbol{\mathcal{X}}_{\text{pose}})$ with the ground-truth binary mask $\boldsymbol{\mathcal{Y}}_{\text{mask}}$ via Mean Squared Error (MSE) loss, i.e.,

$$\mathcal{L}_{\text{reproj}}(\boldsymbol{\mathcal{X}}_{\text{mask}}, \boldsymbol{\mathcal{Y}}_{\text{mask}}) = \|\boldsymbol{\mathcal{X}}_{\text{mask}} - \boldsymbol{\mathcal{Y}}_{\text{mask}}\|_F^2. \tag{15}$$

4. Pose Loss

The pose loss compares the predicted pose vectors $\boldsymbol{\mathcal{X}}_{\text{pose}} = (\tilde{\boldsymbol{r}}_{\text{c}}, \tilde{\boldsymbol{t}}_{\text{c}})$ and the labels $\mathcal{Y}_{\text{pose}} = (\boldsymbol{r}_{\text{c}}, \boldsymbol{t}_{\text{c}})$. This work uses the SPEED loss which was the official metric of the Satellite Pose Estimation Challenges [19, 20] and used in previous works as the loss function for pose regression task [10]. The SPEED loss is defined as

$$\mathcal{L}_{\text{pose}}(\boldsymbol{\mathcal{X}}_{\text{pose}}, \boldsymbol{\mathcal{Y}}_{\text{pose}}) = E_{\text{R}}(\tilde{\boldsymbol{R}}_{\text{c}}, \boldsymbol{R}_{\text{c}}) + E_{\text{t}}(\tilde{\boldsymbol{t}}_{\text{c}}, \boldsymbol{t}_{\text{c}})/\|\boldsymbol{t}_{\text{c}}\|, \tag{16}$$

where $\boldsymbol{R}_c$ is a direction cosine matrix recovered from the 6D rotation vector $\boldsymbol{r}_c$ [52], and

$$E_R(\tilde{\boldsymbol{R}}, \boldsymbol{R}) = \arccos \frac{\text{tr}(\boldsymbol{R}^\top \tilde{\boldsymbol{R}} - 1)}{2}, \tag{17a}$$

$$E_t(\tilde{\boldsymbol{t}}, \boldsymbol{t}) = \|\tilde{\boldsymbol{t}} - \boldsymbol{t}\|. \tag{17b}$$

In other words, $\mathcal{L}_{\text{pose}}$ combines the relative angle between two rotation matrices in radians and the translation error normalized by the ground-truth distance to the target.

5. Regularization Terms

The regularization loss $\mathcal{L}_{\text{reg}}$ consists of multiple loss terms and is added to the final loss function in order to prevent degenerate solutions and enforce different conditions to the final assemblies. The first loss, $\mathcal{L}_{\text{overlap}}$, penalizes overlapping primitives so that each primitive could be responsible for discrete parts of a spacecraft. This is done by taking the same points $\boldsymbol{X}_o$ used in the occupancy loss and assessing whether each point is internal to multiple primitives or not using the inside-outside function in Eq. 8. If the point is in the interior of more than β primitives, the loss is computed for the corresponding point, i.e., [58]

$$\mathcal{L}_{\text{overlap}}(\boldsymbol{X}_o, \boldsymbol{\lambda}) = \frac{1}{N_o} \sum_{\boldsymbol{x}_o \in \boldsymbol{X}_o} \max \left(0, \sum_{m=1}^{M} g(\boldsymbol{x}_o; \boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m) - \beta\right) \tag{18}$$

where $g(\boldsymbol{x}_o; \boldsymbol{\alpha}_m, \boldsymbol{\epsilon}_m)$ is the occupancy function in Eq. 14.

Finally, an additional regularization is introduced to the tapering parameter $\boldsymbol{k}$, i.e.,

$$\mathcal{L}_{\text{taper}} = \sum_{m=1}^{M} \|\boldsymbol{k}_m\|_2^2 \tag{19}$$

to reflect the fact that most structural components of spacecraft are regular shapes such as cuboids and cylinders. Therefore, the regularization loss is the weighted sum of

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{overlap}} + \mathcal{L}_{\text{taper}} \tag{20}$$

6. Remarks on the Number of Primitives

Previous works such as [37] additionally predict the probability of existence associated with each primitive in order to predict variable numbers of primitives depending on the complexity of the target objects in images. In contrast, the proposed CNN always predicts M number of primitives, because it was found that training for such probabilities for multiple supervised loss functions, particularly $\mathcal{L}_{\text{chamfer}}$ and $\mathcal{L}_{\text{occ}}$, and through differentiable rendering actually resulted in much more degraded reconstruction quality. Therefore, predicting assemblies with a variable number of primitives is left as future work.

V. Dataset

In order to train the CNN for 3D shape reconstruction of an unknown target, a large-scale dataset comprising different RSOs is required. For example, the ShapeNet dataset [42] consists of 51,300 unique 3D models of 55 common object categories, which enables improved generalization capability of trained NN models. On the other hand, satellites and spacecraft lack such quantity and diversity of available 3D models due to their proprietary nature. Despite such restrictions, this work presents a novel high-quality dataset comprising 64 unique spacecraft 3D models. The models are selectively acquired from the NASA 3D Resources [59] and ESA Science Satellite Fleet [60]. Each of these models is accompanied by 1,000 images, binary masks and corresponding pose labels to support simultaneous 3D structure characterization and pose estimation. The images and binary masks are rendered using the custom synthetic scene developed within the Unreal Engine (UE) 5. See Fig. 6 for a visualization of a few grayscale images and binary masks. The dataset will be made publicly available with a unique DOI.

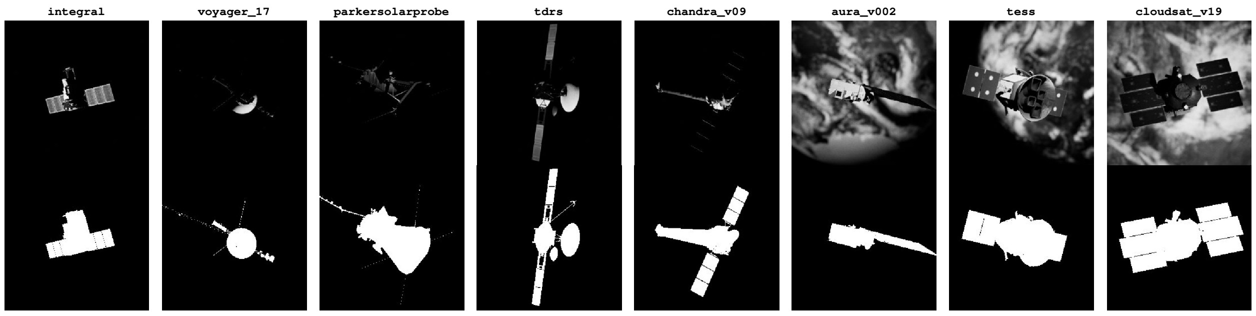


Fig. 6 Visualization of grayscale images and binary masks of select models in the dataset.

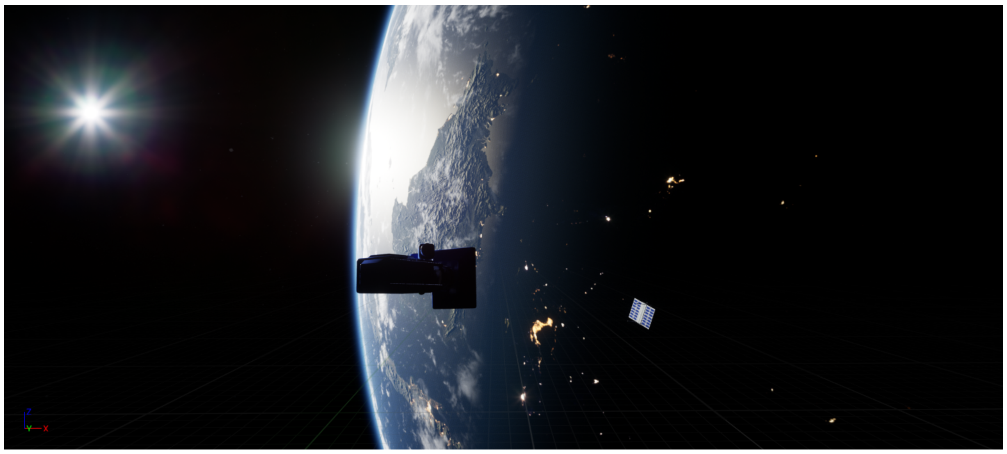


Fig. 7 The UE editor view of the scene. The scene consists of the camera, a CubeSat target, Earth and the Sun. Figure from Ahmed et al. [6]

A. Model Processing

The raw 3D models are first converted to watertight meshes using the method[‡] by Stutz and Geiger [61]. This removes any internal structures so that the sampled surface points ($\mathcal{Y}_s$) do not lie inside the 3D mesh. Moreover, the watertight-ness also allows evaluating the occupancy labels ($\mathcal{Y}_o$) to be used for the occupancy loss in Eq. 13. Once the model is made watertight, it is normalized to a unit size. The reason is that the goal of this work is 3D shape recovery from *single* images, which prohibits prediction of the target's absolute scale without a priori knowledge of its structure. Therefore, the models are normalized for scale-invariant shape recovery. The models are then centered about the center of mass assuming the mesh has uniform mass distribution. Finally, the models must be rotated so that they are aligned in a consistent manner. The challenge is that, unlike terrestrial objects such as chairs and planes in ShapeNet, satellites do not have canonical 'up' and 'forward' directions. Therefore, this work adopts a simple solution to align the 3D model's largest dimension with the x-axis, the second largest with the y-axis, and the smallest with the z-axis. Since the satellites generally have their largest dimensions along the extended solar panels, this method guarantees some level of geometric consistency among different satellite models after alignment.

B. UE Scene and Rendering

The processed models are used to render images using Unreal Engine (UE). The UE scene consists of various movable *actors* for the camera, the directional light simulating the Sun, and several static meshes for the target satellite, Earth with high-resolution texture, and the stellar background. Figure 7 visualizes an example UE scene containing the aforementioned actors. For more information on the constructed UE scene, the readers are referred to Ahmed et al. [6].

Once the models are imported to the UE scene, the images and masks are captured and rendered by the camera actor. For each sample, the camera and the target are placed at various locations around the Earth with random orientation

[‡]https://github.com/paschalidoud/mesh_fusion_simple

according to the pre-computed pose labels which consist of translation and 4D quaternion vectors. Similar to the previous datasets for poes estimation of a known spacecraft such as SPEED [3] and SPEED+ [4], the quaternion vectors representing the target's orientation with respect to the camera are randomly sampled using the subgroup algorithm [62]. The translation vector is instead sampled from a restricted range of distances so that the target nearly fills up the entire image. In this work, the normalized target spacecraft is placed at a random distance so that it nearly fills up the 256×256 images. Specifically, the same focal length and pixel sizes as the cameras for SPEED and SPEED+ are used for rendering.

C. Dataset Splits

Once 1,000 images and binary masks are generated for each of the 64 satellite models, the dataset is split into the training, validation and test sets. First, 7 satellite models and their associated images are randomly selected and reserved as the test set for the purpose of evaluating the performance of CNN on unseen images of the *unknown* models that do not participate in the training. For the remaining 57 models, 800 images per model are used for training, and the remaining 200 images are reserved for validation. In other words, unlike the test set, the validation set is to be used for the evaluation of unseen images of the *known* models already encountered during training. The information on the dataset splits will be made available along with the dataset.

VI. Experiment

A. Implementation

The proposed CNN model is trained for 100 epochs, where the model is trained on all training images per each epoch. The images are resized to 128×128 for more efficient training. The model is trained with the AdamW optimizer [63] with batch size 32, the initial learning rate of 1×10^{-4} which decays according to cosine annealing schedule [64]. The reconstructed assembly is set to have a fixed number of primitives (M). Unless specified otherwise, $M = 5$. The Chamfer loss ($\mathcal{L}_{\text{chamfer}}$) is evaluated using $N_{\text{surf}} = 2,000$ points randomly sampled from the ground-truth 3D mesh, and $N_{\text{sq}} = 642$ points are obtained per each primitive from the vertices of an icosahedron whose mesh is subdivided three times. The occupancy loss ($\mathcal{L}_{\text{occ}}$) instead uses N_{occ} = 10,000 points randomly sampled from within a unit cube. The model is trained on a single NVIDIA RTX 4090 24GB GPU.

During implementation, the range of predicted superquadric parameters is restricted. For example, the size and translation parameters are respectively restricted to $\boldsymbol{\alpha} \in [0.01, 0.5]^2$ and $\boldsymbol{t} \in [-0.5, 0.5]^3$ in order to fit the assembly within a unit cube as is done for the ground-truth 3D meshes. Furthermore, the shape parameters are restricted to $\boldsymbol{\epsilon} \in [0, 1]^2$. The shape parameters are generally lower-bounded with a small positive value in literature to prevent numerical instability when computing gradients of Eqs. 1, 2. However, this work can ignore such lower-bounds when using the Eqs. 7, 8 based on dual superquadrics. The primitives with $\epsilon > 1$ are also ignored since shapes in this regime (e.g., octahedra) are rarely found in spacecraft. The tapering parameters are also restricted to $\boldsymbol{k} \in [-1, 1]^2$.

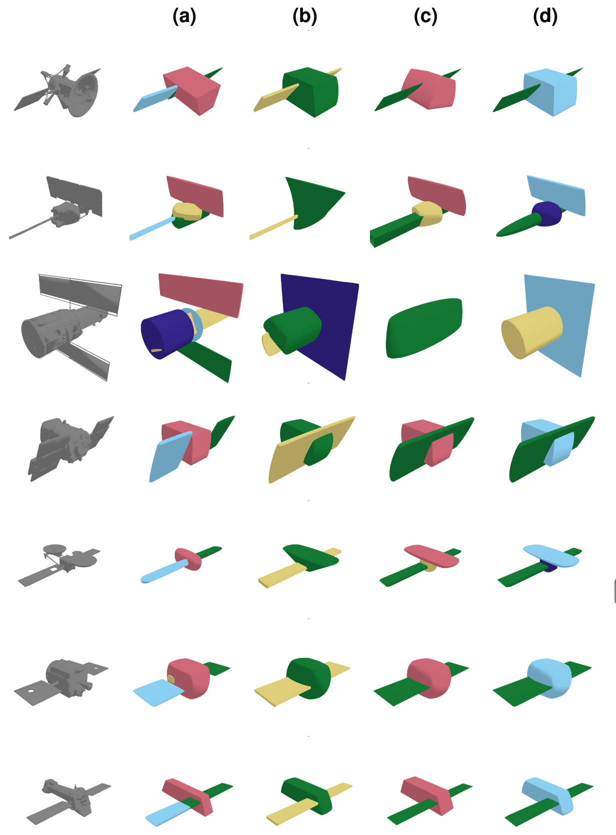
Finally, when sampling surface points for $\mathcal{L}_{\text{chamfer}}$, the points internal to any superquadric primitive are ignored in order to sample from the surface of an overall assembly. This is done using the occupancy function of Eq. 14 – if the point is deemed inside any of the primitive, that point is removed from the point cloud. Any visualization of the point clouds shown in this section follows the same process of removing overlapping points.

B. Evaluation Metrics

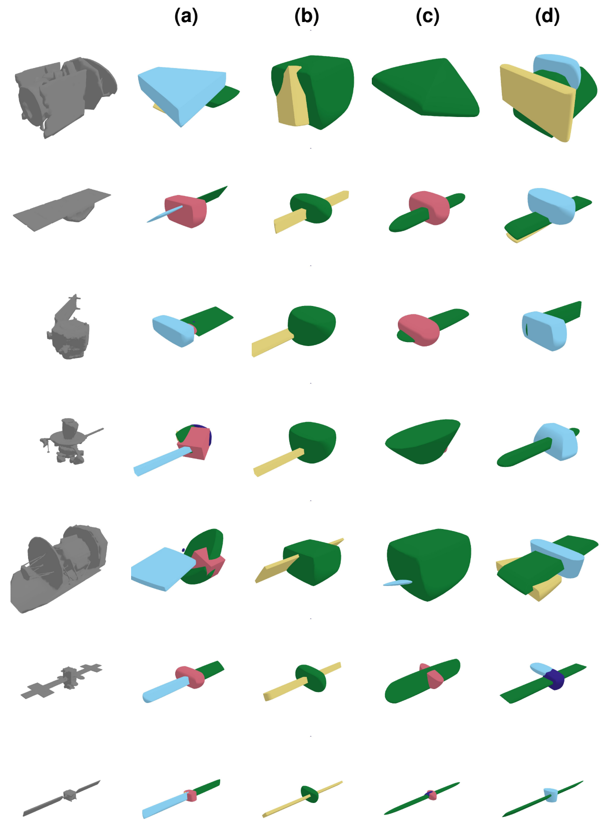
Apart from qualitative results, the quality of reconstructed assemblies is measured using two quantitative metrics. First, following the general practice in literature [38–40], this work reports the Chamfer-ℓ_1 distance which is simply equivalent to $\mathcal{L}_{\text{chamfer}}$ but with ℓ_1-norm. Second, the 2D IoU between the $\boldsymbol{\mathcal{Y}}_{\text{mask}}$ and $R(\boldsymbol{\lambda}, \boldsymbol{\mathcal{X}}_{\text{pose}})$ is reported. Note that unlike in training where $\boldsymbol{\mathcal{Y}}_{\text{pose}}$ is used for rendering, here the predicted pose $\boldsymbol{\mathcal{X}}_{\text{pose}}$ is used in order to measure the quality of both the assembly and the predicted pose simultaneously. Finally, the rotation error ($\boldsymbol{E}_{\text{R}}$) in Eq. 17a is reported in degrees. The translation error is not reported since the range of relative distance is very limited as explained in Section V.B.

Table 1 Quantitative evaluation of performance metrics on different training loss configurations. Arrows indicate the direction towards better performance. Bold faces indicate the best performance.

Config.	validation			test		
	Chamfer-ℓ_1 (↓)	E_R [°] (↓)	IoU (2D) (↑)	Chamfer-ℓ_1 (↓)	E_R [°] (↓)	IoU (2D) (↑)
(a) $\mathcal{L}_{\text{chamfer}} + \mathcal{L}_{\text{reproj}}$	**0.054** ± **0.026**	60.8 ± 56.1	0.594 ± 0.183	0.146 ± 0.084	127.1 ± 45.0	0.476 ± 0.151
(b) $\mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{reproj}}$	0.592 ± 0.081	60.3 ± 55.8	**0.599** ± **0.173**	0.663 ± 0.091	123.5 ± 45.8	**0.523** ± **0.149**
(c) $\mathcal{L}_{\text{chamfer}} + \mathcal{L}_{\text{occ}}$	0.085 ± 0.054	**59.4** ± **55.2**	0.583 ± 0.173	**0.139** ± **0.075**	**123.1** ± **46.7**	0.500 ± 0.154
(d) $\mathcal{L}_{\text{chamfer}} + \mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{reproj}}$	0.076 ± 0.038	59.6 ± 53.8	0.584 ± 0.181	0.143 ± 0.078	124.1 ± 44.7	0.484 ± 0.149



(a) Validation



(b) Test

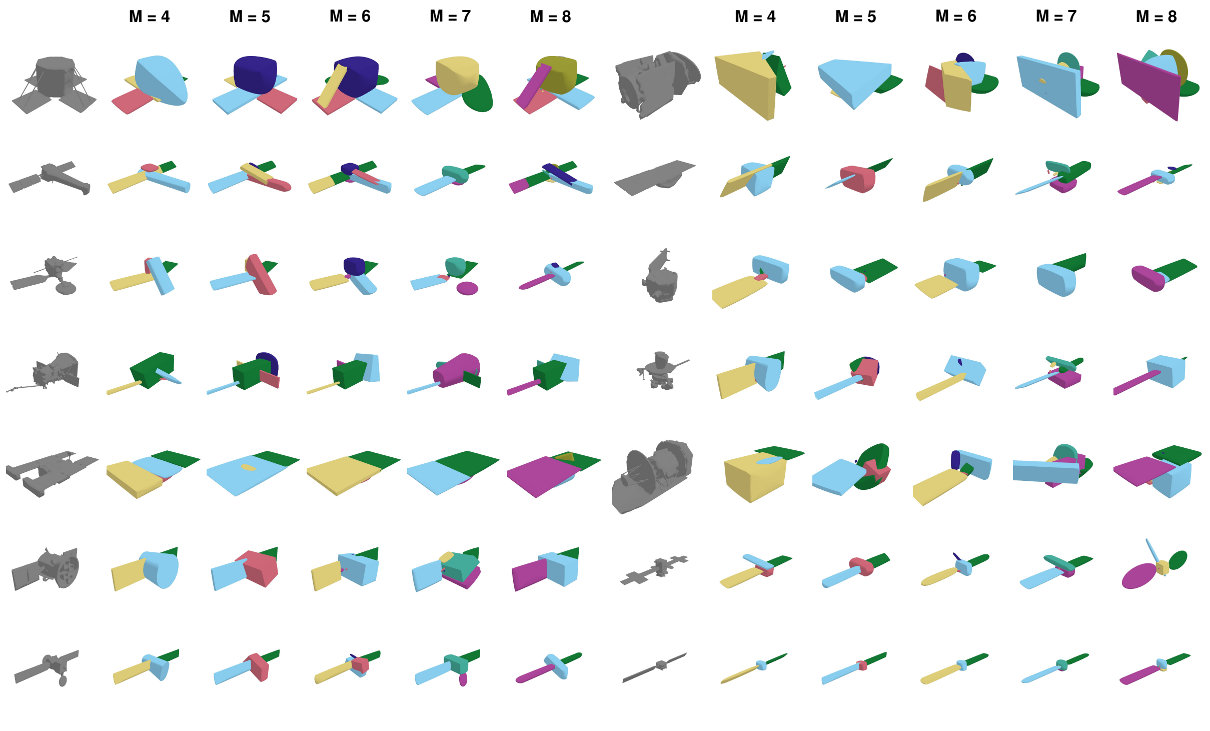
Fig. 8 Visualization of reconstructed assemblies using different training configurations.

C. Ablation Studies: Loss Functions

The experiment first begins with an ablation study where different configurations of supervised loss functions ($\mathcal{L}_{\text{chamfer}}$, $\mathcal{L}_{\text{occ}}$, $\mathcal{L}_{\text{reproj}}$) are used during training. Table 1 and Fig. 8 respectively show the quantitative and qualitative comparisons for different evaluation metrics. First, it is immediately obvious that removing the Chamfer loss is detrimental to the reconstruction quality as evidenced by large values of Chamfer-ℓ_1 distance in both validation and test sets. However, despite having the largest Chamfer distance by the factor of nearly seven compared to the runner-up in the validation set, configuration (b) without Chamfer loss still performs the best in terms of the IoU metric when reprojected to the 2D space using $\boldsymbol{X}_{\text{pose}}$. This suggests that the 2D IoU might not be the best metric for coarse shapes such as superquadrics compared to other representations such as meshes that can model fine-grained details.

Table 2 Quantitative evaluation of performance metrics on a different number of primitives (M). Arrows indicate the direction towards better performance. Bold faces indicate the best performance.

M	validation			test		
	Chamfer-ℓ_1 (↓)	E_R [°] (↓)	IoU (2D) (↑)	Chamfer-ℓ_1 (↓)	E_R [°] (↓)	IoU (2D) (↑)
4	0.057 ± 0.028	**59.2** ± **55.0**	0.590 ± 0.184	0.150 ± 0.085	123.7 ± 45.9	**0.477** ± **0.154**
5	0.054 ± 0.026	60.8 ± 56.1	0.594 ± 0.183	0.146 ± 0.084	127.1 ± 45.0	0.476 ± 0.151
6	0.053 ± 0.025	61.9 ± 55.6	0.585 ± 0.187	0.141 ± 0.078	**123.6** ± **46.1**	0.473 ± 0.149
7	0.052 ± 0.026	61.1 ± 55.8	0.590 ± 0.186	0.137 ± 0.074	124.0 ± 46.2	0.470 ± 0.148
8	**0.051** ± **0.025**	59.2 ± 55.1	**0.594** ± **0.186**	**0.137** ± **0.081**	125.3 ± 46.3	0.472 ± 0.146



(a) Validation **(b) Test**

Fig. 9 Visualization of reconstructed assemblies using different numbers of primitives.

Moreover, adding $\mathcal{L}_{\text{occ}}$ results in worse performance in the validation set as shown in the difference between (a) and (d). This can be explained by the fact that satellites include extremely thin structures such as solar panels, which means it is less likely that one can sample enough points internal to these structures compared to more common objects in ShapeNet such as chairs and planes. Therefore, the occupancy loss operates with highly imbalanced occupancy labels, potentially interfering with supervisory labels for other more reliable loss functions. On the other hand, adding $\mathcal{L}_{\text{reproj}}$ results in better reconstruction quality in the validation set as shown in the difference between (c) and (d) thanks to the additional supervision from the differentiable rendering. The same observation cannot be said for the test set; however, the Chamfer-ℓ_1 metrics are comparable for (a), (c) and (d) configurations with $\mathcal{L}_{\text{chamfer}}$. Considering that the test results are in general worse than the validation results due to the fact that the CNN is trained on an extremely restricted dataset and that the test models have not been observed during the training, the mismatching performance trends on the test set carry less importance than the validation set. Therefore, moving forward, the analyses focus on the models trained with the configuration (a) based on its superior performance on the validation set.

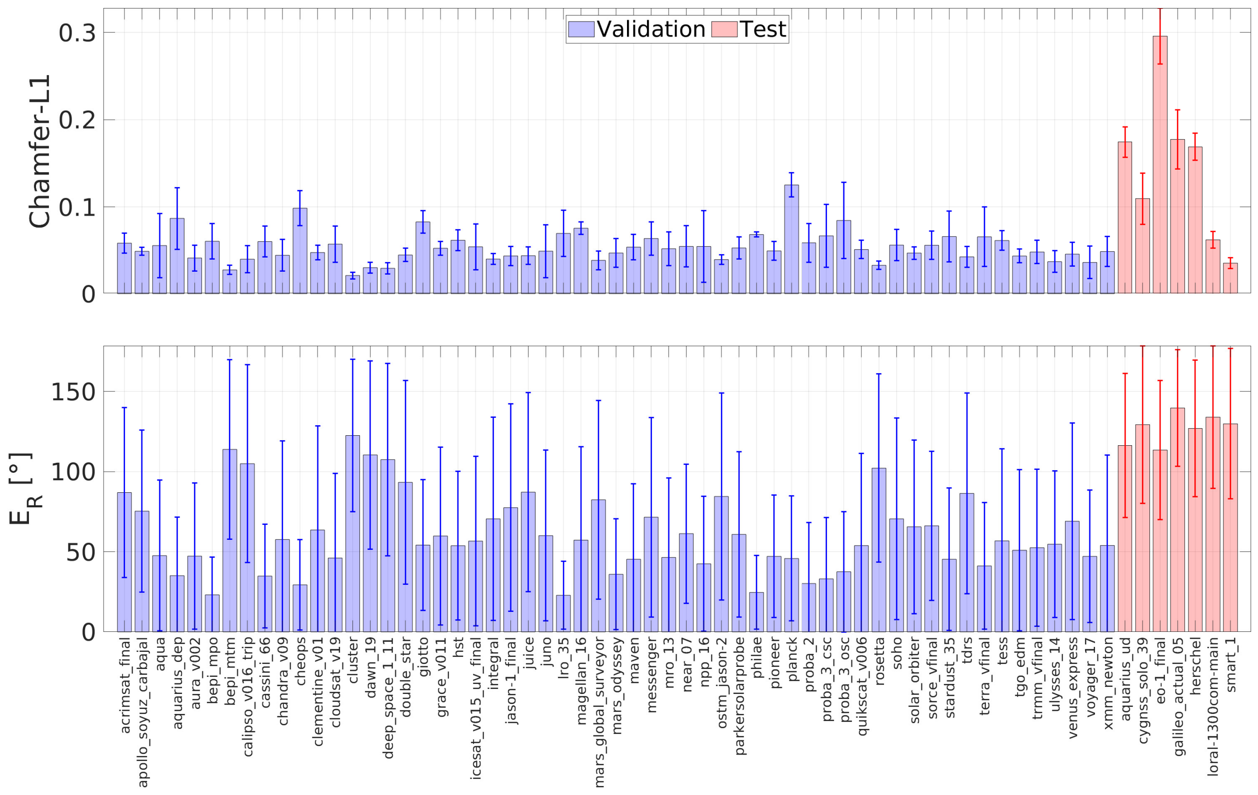


Fig. 10 Bar charts of mean performance for the Chamfer-ℓ_1 distance and rotation error ($E_{\mathbf{R}}$) metrics on each validation and test model. The standard deviations are noted with the error bars.

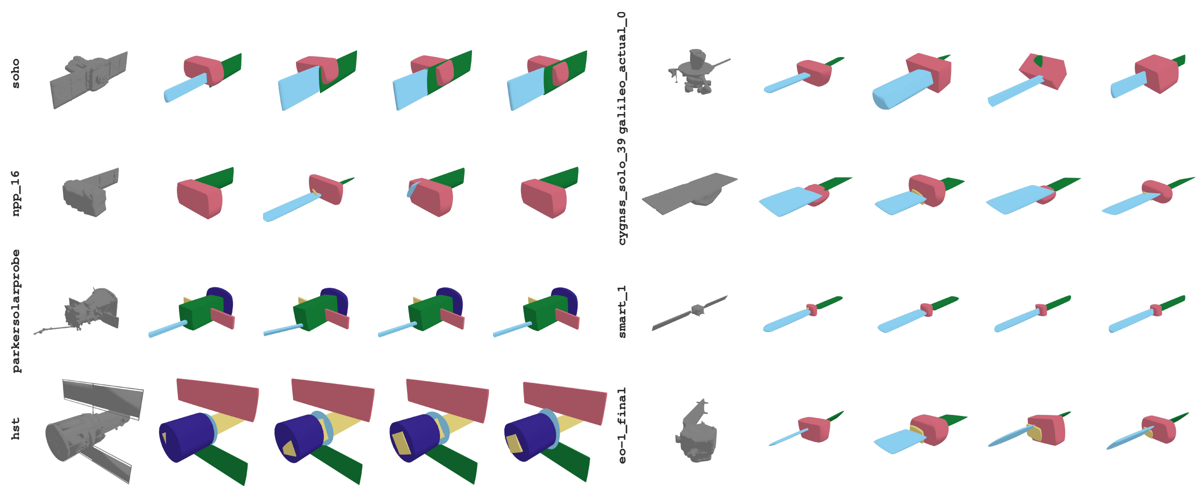
One can also make an interesting observation from Fig. 8 that the assemblies predicted with configuration (a) always associate one primitive for each solar panel on both validation and test models, whereas all other configurations tend to allocate a single primitive for both extended panels. It is likely that such a phenomenon can be attributed to the effect of the occupancy loss. Finally, while the performance on unknown models of the test set is generally worse, one can still see from Fig. 8b that the high-level structure of the target (e.g., one solar panel, two panels) can still be captured by the superquadric assemblies most of the time from single 2D images.

D. Ablation Studies: Number of Primitives

Next, the number of primitives (M) is varied from 4 to 8. Table 2 clearly shows that using more primitives in assemblies improves the Chamfer-ℓ_1 metric but with diminishing return as more primitives are added. This makes sense since more primitives allow for depicting complex structures with improved accuracy. The qualitative examples from Fig. 9 also support the same trend for the most part, as the primitives predicted with $M = 7$ or 8 tend to better associate each primitive with distinct parts of the target satellites in the validation. On the test set, Fig. 9b again indicates that the CNN can abstract high-level structural characteristics of the target satellites most of the time, but the predictions are very sensitive due to lack of training data.

E. Per-Model Performance Analyses

Figure 10 provides the mean and standard deviation of different metrics for each validation and test model. In terms of the Chamfer-ℓ_1 distance, there is a stark contrast in performance between validation and test sets. Interestingly, the distance is on par with those of the validation set for the `local-1300com-main` and `smart_1` test models which both have two symmetric extended solar panels. This is one of the most common types of structures in the training set, which makes it easier for CNN to predict the correct shapes for these two models despite having never observed them during training. On the other hand, the distance for the `eo-1_final` is particularly large because it has a unique structure (i.e., a single solar panel extended in angle) that no other models in the training set even remotely resemble. See Figs. 8b, 9b



(a) Validation **(b) Test**

Fig. 11 Visualization of reconstructed models on a few example models from both validation and test sets. Each predicted assembly corresponds to different image inputs for the same models.

for comparison as it visualizes the test models from top to bottom in the order presented in Fig. 10.

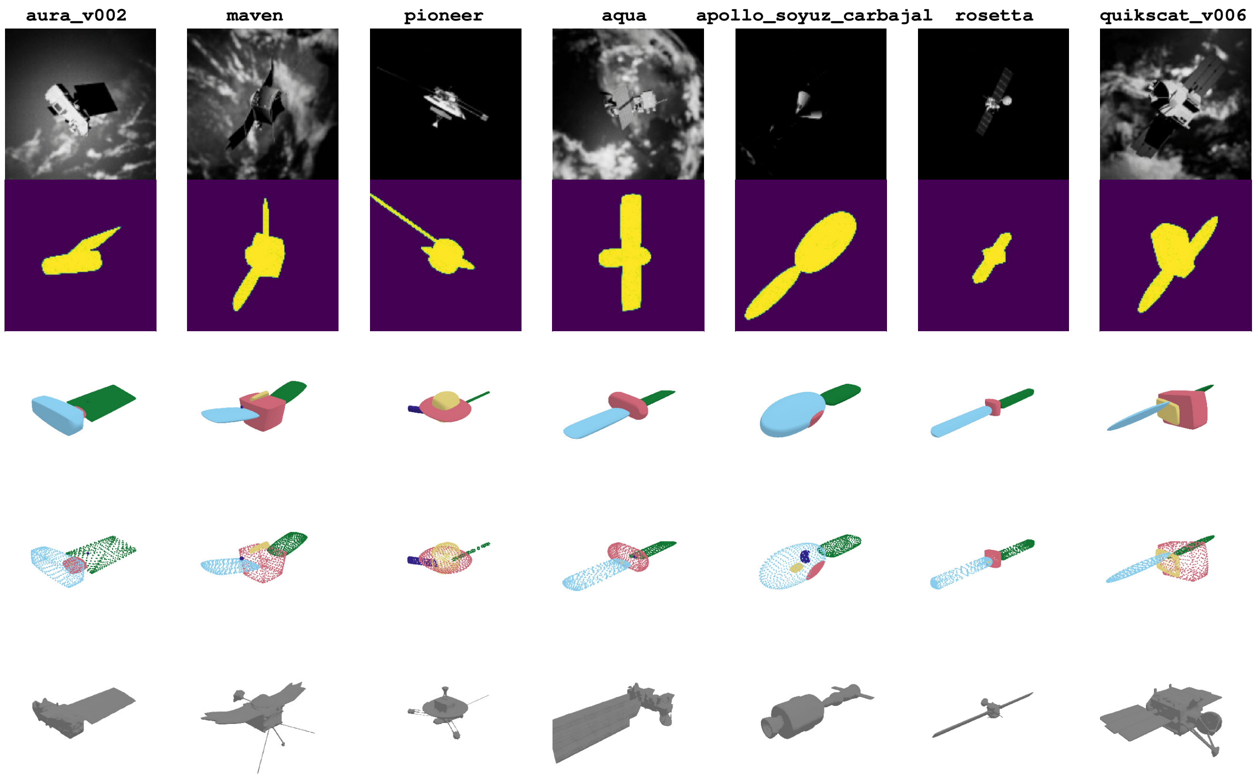
For the rotation errors, the mean performance on each model in the validation test varies significantly. However, investigating the individual models, it is found that many of those with the highest errors (e.g., `dawn_19`, `deep_space_1_11`, `rosetta`) are symmetric satellites with two extended solar panels. Even for known targets, it would be difficult to correctly disambiguate the viewpoint if the target is symmetric. On the other hand, the rotation errors for the test set show consistently high errors for all models.

Finally, more qualitative examples are provided in Fig. 12 which visualizes the predicted assemblies, reprojected silhouettes and the point clouds converted from the assemblies. Figure 11 instead shows the reconstructed assemblies for different input images of the same models. Note that the predicted assemblies are quite consistent across different inputs for the validation set. However, it is still possible that, depending on the view of the target and the illumination conditions in the input image, the CNN may predict a completely different structure (e.g., `npp_16`). On the other hand, the consistency is degraded on the test set. For example, the `galileo_actual_05` model is predicted correctly as having a single extended boom half of the time, but the other half of the predictions incorrectly reconstruct the assembly as having two extended solar panels. Again, such sensitivity is expected due to the limited quantity of available training models; however, it is still remarkable that the correct shape can be inferred most of the time from a single 2D image.

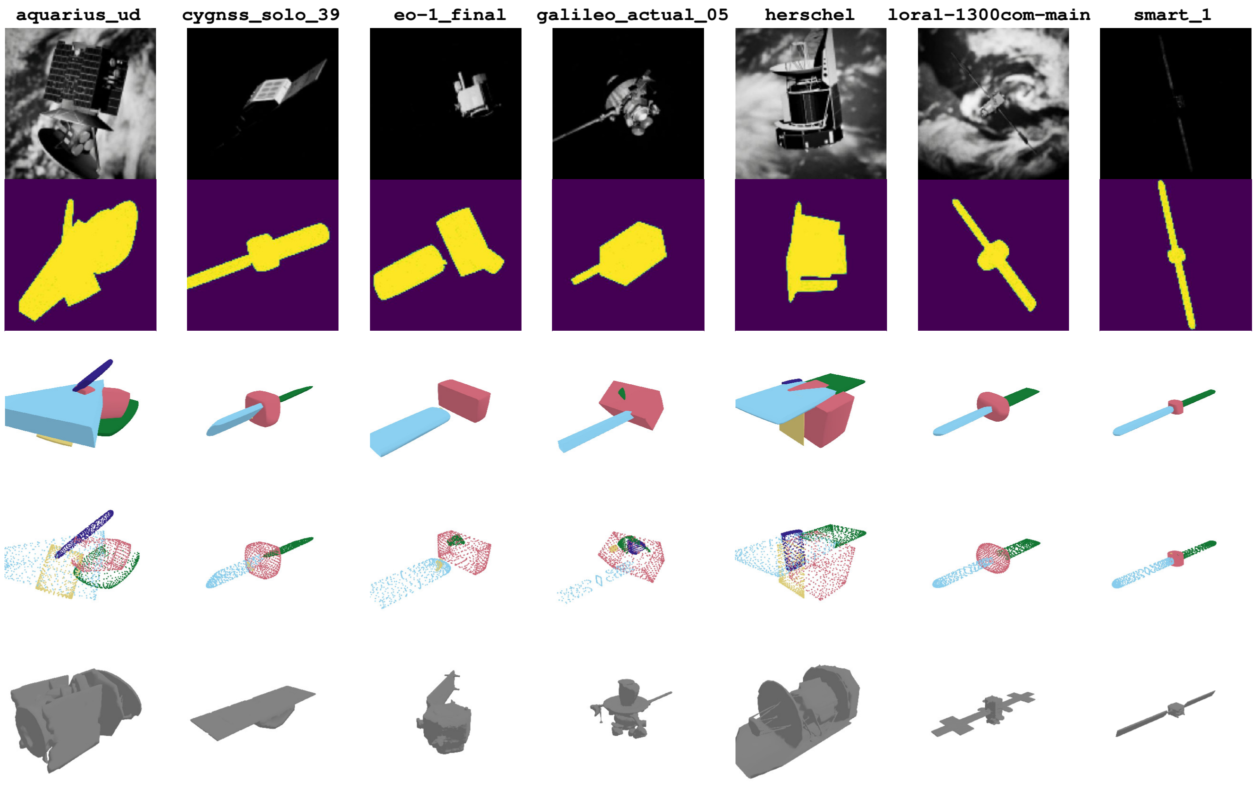
VII. Conclusion

This work introduces for the first time a Convolutional Neural Network (CNN) architecture and the training pipeline to simultaneously reconstruct the 3D structure of an unknown target spacecraft and estimate its pose with respect to the servicer. The 3D structure is represented as an assembly of a fixed number of superquadric primitives, allowing compact representation using only a handful of parameters. Novel formulations of the superquadrics are introduced to enable numerically stable evaluation of points on or within a superquadric primitive for all shape parameters and gradient-based learning. Furthermore, this work also introduces a novel dataset comprising 64 different satellite models and high-quality images rendered using the Unreal Engine. The proposed CNN is trained with the dataset, and the experimental results reveal that, while the proposed model performs poorly on unknown targets due to a lack of satellite models in the training dataset, it is still able to capture the correct high-level structures of the satellites (e.g., single solar panel, two symmetric solar panels) most of the time from single 2D images.

The results of this work pave the way forward for many different aspects of the proposed algorithm. First, the proposed architecture uses a fixed number of primitives, but it can be improved to predict a variable number of primitives depending on the complexity of the target model. The immediate next step would be to incorporate the proposed architecture into a sequential filter for Simultaneous Navigation and Characterization (SNAC) to either continuously refine the predicted superquadric primitive parameters or use the initially predicted assemblies to kick off subsequent



(a) Validation data.



(b) Test data.

Fig. 12 (*First row*) **Input image; (***Second row***) Reprojected silhouette based on predicted assembly and pose; (***Third row***) Predicted assembly; (***Fourth row***) Point clouds converted from the assemblies; (***Fifth row***) Ground-truth mesh.**

SNAC algorithms based on point clouds. Finally, for the proposed supervised learning approach to properly work, the dataset size must be increased by either reassembling existing satellite parts to create new spacecraft models or developing a generative model to output new spacecraft mesh models by training on existing sparse data.

Appendix: Derivation of Eq. 8

Recall that a point can be sampled on the surface of a superquadric with parameters $(\boldsymbol{\alpha}, \boldsymbol{\epsilon})$ using a dual approach, which entails first sampling points from two 2D superellipses each parametrized with $(1, \alpha_3, \epsilon_1)$ amd $(\alpha_1, \alpha_2, \epsilon_2)$ using Eq. 6 then performing a spherical product as shown in Eq. 7. Combining these two equations yield

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} = \begin{bmatrix} x/\gamma_1\gamma_2\alpha_1 \\ y/\gamma_1\gamma_2\alpha_2 \\ z/\gamma_1\alpha_3 \end{bmatrix} = \begin{bmatrix} (\cos^{\epsilon_1}\eta + \sin^{\epsilon_1}\eta)(\cos^{\epsilon_2}\omega + \sin^{\epsilon_2}\omega) \\ (\cos^{\epsilon_1}\eta + \sin^{\epsilon_1}\eta)(-\cos^{\epsilon_2}\omega + \sin^{\epsilon_2}\omega) \\ (-\cos^{\epsilon_1}\eta + \sin^{\epsilon_1}\eta) \end{bmatrix} \tag{21}$$

where $\boldsymbol{r}_{\text{dual}}(\eta, \omega) = [x\ y\ z]^\top$, ϵ_i denotes the *dual* shape parameter for the ease of notation, and $\gamma_i = 2^{\epsilon_i/2-1}$.

First, manipulating $(\bar{x}, \bar{y})$, one can obtain

$$\frac{\bar{x}+\bar{y}}{2} = (\cos^{\epsilon_1}\eta + \sin^{\epsilon_1}\eta)\sin^{\epsilon_2}\omega \tag{22}$$

$$\frac{\bar{x}-\bar{y}}{2} = (\cos^{\epsilon_1}\eta + \sin^{\epsilon_1}\eta)\cos^{\epsilon_2}\omega \tag{23}$$

$$\tag{24}$$

Then, it follows that

$$\left[\left(\frac{\bar{x}+\bar{y}}{2}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{\bar{x}-\bar{y}}{2}\right)^{\frac{2}{\epsilon_2}}\right]^{\frac{\epsilon_2}{2}} = \cos^{\epsilon_1}\eta + \sin^{\epsilon_1}\eta \tag{25}$$

due to the Pythagorean identity. Note that the left-hand side is equivalent to $\phi(\bar{x}, \bar{y}, \epsilon_2)$ as defined in Eq. 9. Now, one can repeat the same process for $\big(\phi(\bar{x}, \bar{y}, \epsilon_2), \bar{z}\big)$:

$$\frac{\phi(\bar{x}, \bar{y}, \epsilon_2) + \bar{z}}{2} = \sin^{\epsilon_1}\eta \tag{26}$$

$$\frac{\phi(\bar{x}, \bar{y}, \epsilon_2) - \bar{z}}{2} = \cos^{\epsilon_1}\eta \tag{27}$$

$$\tag{28}$$

and

$$\left[\left(\frac{\phi(\bar{x}, \bar{y}, \epsilon_2) + \bar{z}}{2}\right)^{\frac{2}{\epsilon_1}} + \left(\frac{\phi(\bar{x}, \bar{y}, \epsilon_2) - \bar{z}}{2}\right)^{\frac{2}{\epsilon_1}}\right]^{\frac{\epsilon_1}{2}} = 1 \tag{29}$$

again due to the same identity. The left-hand side is equivalent to $\phi(\phi(\bar{x}, \bar{y}, \epsilon_2), \bar{z}, \epsilon_1)$, which concludes the derivation.

Acknowledgement

Parts of this work have been supported by 1) SpaceWERX Orbital Prime - Stanford Subcontract Agreement No. SC-1778-01 US-RIPTIDE Phase 2 with Scientific Systems Company Inc. and 2) Redwire Space Corporation as Associate Member and Co-Founder of the Stanford's Center of AEroSpace Autonomy Research (CAESAR). The authors would like to thank the European Space Agency (ESA) for granting the authorization to use the 3D models from the ESA Science Satellite Fleet.

References

[1] D'Amico, S., Benn, M., and Jørgensen, J. L., "Pose estimation of an uncooperative spacecraft from actual space imagery," *International Journal of Space Science and Engineering*, Vol. 2, No. 2, 2014, p. 171. https://doi.org/10.1504/ijspacese.2014.060600.

[2] Giralo, V., and D'Amico, S., "Distributed multi-GNSS timing and localization for nanosatellites," *NAVIGATION*, Vol. 66, No. 4, 2019, pp. 729–746. https://doi.org/10.1002/navi.337.

[3] Sharma, S., and D'Amico, S., "Neural Network-Based Pose Estimation for Noncooperative Spacecraft Rendezvous," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 6, 2020, pp. 4638–4658. https://doi.org/10.1109/TAES.2020.2999148.

[4] Park, T. H., Märtens, M., Lecuyer, G., Izzo, D., and D'Amico, S., "SPEED+: Next-Generation Dataset for Spacecraft Pose Estimation across Domain Gap," *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 1–15. https://doi.org/10.1109/AERO53065.2022.9843439.

[5] Park, T. H., and D'Amico, S., "Adaptive Neural-Network-Based Unscented Kalman Filter for Robust Pose Tracking of Noncooperative Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 9, 2023, pp. 1671–1688. https://doi.org/10.2514/1.G007387.

[6] Ahmed, Z., Park, T. H., Bhattacharjee, A., Fazel-Rezai, R., Graves, R., Saarela, O., Teramoto, R., Vemulapalli, K., and D'Amico, S., "SPEED-UE-Cube: A Machine Learning Dataset for Autonomous, Vision-Based Spacecraft Navigation," *46th Annual AAS Guidance, Navigation and Control (GN&C) Conference*, 2024 [Accepted].

[7] Musallam, M. A., Gaudilliere, V., Ghorbel, E., Ismaeil, K. A., Perez, M. D., Poucet, M., and Aouada, D., "Spacecraft Recognition Leveraging Knowledge of Space Environment: Simulator, Dataset, Competition Design and Analysis," *2021 IEEE International Conference on Image Processing Challenges (ICIPC)*, 2021, pp. 11–15. https://doi.org/10.1109/ICIPC53495.2021.9620184.

[8] Price, A., and Yoshida, K., "A Monocular Pose Estimation Case Study: The Hayabusa2 Minerva-II2 Deployment," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 1992–2001. https://doi.org/10.1109/CVPRW53098.2021.00227.

[9] Park, T. H., Sharma, S., and D'Amico, S., "Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft," *2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, Maine*, 2019.

[10] Park, T. H., and D'Amico, S., "Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap," *Advances in Space Research*, 2023. https://doi.org/10.1016/j.asr.2023.03.036.

[11] Chen, B., Cao, J., Bustos, Á. P., and Chin, T.-J., "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 2816–2824. https://doi.org/10.1109/ICCVW.2019.00343.

[12] Proença, P. F., and Gao, Y., "Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6007–6013. https://doi.org/10.1109/ICRA40945.2020.9197244.

[13] Pasqualetto Cassinis, L., Fonod, R., Gill, E., Ahrns, I., and Gil-Fernández, J., "Evaluation of tightly- and loosely-coupled approaches in CNN-based pose estimation systems for uncooperative spacecraft," *Acta Astronautica*, Vol. 182, 2021, pp. 189–202. https://doi.org/10.1016/j.actaastro.2021.01.035.

[14] Black, K., Shankar, S., Fonseka, D., Deutsch, J., Dhir, A., and Akella, M., "Real-Time, Flight-Ready, Non-Cooperative Spacecraft Pose Estimation Using Monocular Imagery," *31st AAS/AIAA Space Flight Mechanics Meeting*, 2021.

[15] Sharma, S., Beierle, C., and D'Amico, S., "Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks," *2018 IEEE Aerospace Conference*, 2018, pp. 1–12. https://doi.org/10.1109/AERO.2018.8396425.

[16] Garcia, A., Musallam, M., Gaudilliere, V., Ghorbel, E., Ismaeil, K. A., Perez, M., and Aouada, D., "LSPnet: A 2D Localization-oriented Spacecraft Pose Estimation Neural Network," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2048–2056. https://doi.org/10.1109/CVPRW53098.2021.00233.

[17] D'Amico, S., Bodin, P., Delpech, M., and Noteborn, R., "PRISMA," *Distributed Space Missions for Earth System Monitoring Space Technology Library*, Vol. 31, edited by M. D'Errico, 2013, Chap. 21, pp. 599–637. https://doi.org/10.1007/978-1-4614-4541-8_21.

[18] Park, T. H., Bosse, J., and D'Amico, S., "Robotic Testbed for Rendezvous and Optical Navigation: Multi-Source Calibration and Machine Learning Use Cases," *2021 AAS/AIAA Astrodynamics Specialist Conference, Big Sky, Vitrual*, 2021.

[19] Kisantal, M., Sharma, S., Park, T. H., Izzo, D., Märtens, M., and D'Amico, S., "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 5, 2020, pp. 4083–4098. https://doi.org/10.1109/TAES.2020.2989063.

[20] Park, T. H., Märtens, M., Jawaid, M., Wang, Z., Chen, B., Chin, T.-J., Izzo, D., and D'Amico, S., "Satellite Pose Estimation Competition 2021: Results and Analyses," *Acta Astronautica*, Vol. 204, 2023, pp. 640–665. https://doi.org/10.1016/j.actaastro.2023.01.002.

[21] Harris, C., and Stephens, M., "A combined corner and edge detector," *Proceedings of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[22] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, 2004, p. 91–110. https://doi.org/10.1023/b:visi.0000029664.99615.94.

[23] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: An Efficient Alternative to SIFT or SURF," *Proceedings of the 2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. https://doi.org/10.1109/ICCV.2011.6126544.

[24] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D., "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, Vol. 31, No. 5, 2015, pp. 1147–1163. https://doi.org/10.1109/TRO.2015.2463671.

[25] Dor, M., and Tsiotras, P., *ORB-SLAM Applied to Spacecraft Non-Cooperative Rendezvous*, ???? https://doi.org/10.2514/6.2018-1963.

[26] Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S., "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction," *Computer Vision – ECCV 2016*, edited by B. Leibe, J. Matas, N. Sebe, and M. Welling, Springer International Publishing, Cham, 2016, pp. 628–644. https://doi.org/10.1007/978-3-319-46484-8_38.

[27] Xie, H., Yao, H., Sun, X., Zhou, S., and Zhang, S., "Pix2Vox: Context-Aware 3D Reconstruction From Single and Multi-View Images," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2690–2698. https://doi.org/10.1109/ICCV.2019.00278.

[28] Tatarchenko, M., Dosovitskiy, A., and Brox, T., "Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2107–2115. https://doi.org/10.1109/ICCV.2017.230.

[29] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X., "O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis," *ACM Trans. Graph.*, Vol. 36, No. 4, 2017. https://doi.org/10.1145/3072959.3073608.

[30] Qi, C. R., Yi, L., Su, H., and Guibas, L. J., "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[31] Fan, H., Su, H., and Guibas, L., "A Point Set Generation Network for 3D Object Reconstruction from a Single Image," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2463–2471. https://doi.org/10.1109/CVPR.2017.264.

[32] Jiang, L., Shi, S., Qi, X., and Jia, J., "GAL: Geometric Adversarial Loss for Single-View 3D-Object Reconstruction," *Computer Vision – ECCV 2018*, edited by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Springer International Publishing, Cham, 2018, pp. 820–834.

[33] Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L., "Learning Representations and Generative Models for 3D Point Clouds," *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause, PMLR, 2018, pp. 40–49. URL https://proceedings.mlr.press/v80/achlioptas18a.html.

[34] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G., "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," *Computer Vision – ECCV 2018*, edited by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Springer International Publishing, Cham, 2018, pp. 55–71.

[35] Gao, J., Chen, W., Xiang, T., Jacobson, A., McGuire, M., and Fidler, S., "Learning Deformable Tetrahedral Meshes for 3D Reconstruction," *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Curran Associates, Inc., 2020, pp. 9936–9947. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

[36] Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J., "Learning Shape Abstractions by Assembling Volumetric Primitives," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1466–1474. https://doi.org/10.1109/CVPR.2017.160.

[37] Paschalidou, D., Ulusoy, A. O., and Geiger, A., "Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10336–10345. https://doi.org/10.1109/CVPR.2019.01059.

[38] Paschalidou, D., Van Gool, L., and Geiger, A., "Learning Unsupervised Hierarchical Part Decomposition of 3D Objects From a Single RGB Image," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1057–1067. https://doi.org/10.1109/CVPR42600.2020.00114.

[39] Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., and Tagliasacchi, A., "CvxNet: Learnable Convex Decomposition," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 31–41. https://doi.org/10.1109/CVPR42600.2020.00011.

[40] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A., "Occupancy Networks: Learning 3D Reconstruction in Function Space," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4455–4465. https://doi.org/10.1109/CVPR.2019.00459.

[41] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ravi, R., and Ng, R., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," Springer International Publishing, 2020, pp. 405–421.

[42] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F., "ShapeNet: An Information-Rich 3D Model Repository," Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[43] Müller, T., Evans, A., Schied, C., and Keller, A., "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Trans. Graph.*, Vol. 41, 2022. https://doi.org/10.1145/3528223.3530127.

[44] Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F., "D-NeRF: Neural Radiance Fields for Dynamic Scenes," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10313–10322. https://doi.org/10.1109/CVPR46437.2021.01018.

[45] Mergy, A., Lecuyer, G., Derksen, D., and Izzo, D., "Vision-Based Neural Scene Representations for Spacecraft," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021, pp. 2002–2011.

[46] Mahendrakar, T., Caruso, B., Nguyen, V. M., White, R. T., and Steffen, T., "3D Reconstruction of Non-cooperative Resident Space Objects using Instant NGP-accelerated NeRF and D-NeRF," , 2023.

[47] Schönberger, J. L., and Frahm, J.-M., "Structure-from-Motion Revisited," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[48] Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M., "Pixelwise View Selection for Unstructured Multi-View Stereo," *European Conference on Computer Vision (ECCV)*, 2016.

[49] Yen-Chen, L., Florence, P., Barron, J. T., Rodriguez, A., Isola, P., and Lin, T.-Y., "iNeRF: Inverting Neural Radiance Fields for Pose Estimation," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1323–1330. https://doi.org/10.1109/IROS51168.2021.9636708.

[50] Rosinol, A., Leonard, J. J., and Carlone, L., "NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields," 2022. URL http://arxiv.org/abs/2210.13641.

[51] Barr, A. H., "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, Vol. 1, No. 1, 1981, pp. 11–23. https://doi.org/10.1109/MCG.1981.1673799.

[52] Zhou, Y., Barnes, C., Lu, J., Yang, J., and Li, H., "On the Continuity of Rotation Representations in Neural Networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5738–5746. https://doi.org/10.1109/CVPR.2019.00589.

[53] Pilu, M., and Fisher, R. B., "Equal-Distance Sampling of Superellipse Models," *Proceedings of the 1995 British Conference on Machine Vision (Vol. 1)*, 1995, p. 257–266.

[54] Liu, W., Wu, Y., Ruan, S., and Chirikjian, G. S., "Robust and Accurate Superquadric Recovery: a Probabilistic Approach," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 2666–2675. https://doi.org/10.1109/CVPR52688.2022.00270.

[55] Jaklič, A., Leonardis, A., and Solina, F., *Superquadrics and Their Geometric Properties*, Springer Netherlands, Dordrecht, 2000, pp. 13–39. https://doi.org/10.1007/978-94-015-9456-1_2.

[56] Tan, M., and Le, Q., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov, PMLR, 2019, pp. 6105–6114. URL https://proceedings.mlr.press/v97/tan19a.html.

[57] Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., and Gkioxari, G., "Accelerating 3D Deep Learning with PyTorch3D," , 2020.

[58] Paschalidou, D., Katharopoulos, A., Geiger, A., and Fidler, S., "Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3203–3214. https://doi.org/10.1109/CVPR46437.2021.00322.

[59] "NASA 3D Resources – 3D Models," Available at https://nasa3d.arc.nasa.gov/models (2023/11/22), 2023.

[60] "ESA Science Satellite Fleet," Available at https://scifleet.esa.int/ (2023/11/22), 2023.

[61] Stutz, D., and Geiger, A., "Learning 3D Shape Completion Under Weak Supervision," *International Journal of Computer Vision*, Vol. 128, 2020, pp. 1162–1181. https://doi.org/10.1007/s11263-018-1126-y.

[62] Shoemake, K., "III.6 - Uniform Random Rotations," *Graphics Gems III (IBM Version)*, edited by D. Kirk, Morgan Kaufmann, San Francisco, 1992, pp. 124 – 132.

[63] Loshchilov, I., and Hutter, F., "Decoupled Weight Decay Regularization," *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[64] Loshchilov, I., and Hutter, F., "SGDR: Stochastic Gradient Descent with Warm Restarts," *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.