AAS 24-027

SPEED-UE-CUBE: A MACHINE LEARNING DATASET FOR AUTONOMOUS, VISION-BASED SPACECRAFT NAVIGATION

Zahra Ahmed[*], Tae Ha Park[*], Abhijit Bhattacharjee[†], Reza Fazel-Rezai[‡], Russell Graves[§], Ossi Saarela[¶], Reece Teramoto[‖], Kautilya Vemulapalli[**], Simone D'Amico[††]

Vision-based navigation is a pivotal technology for future on-orbit operations, with Machine Learning (ML) approaches for pose estimation presenting a promising alternative to traditional computer vision techniques. Developing these ML algorithms requires large datasets for training and validation. Additionally, the difficulty of obtaining true spaceborne images drives the need for realistic, synthetic image datasets. This work presents SPEED-UE-Cube, a new machine learning dataset for pose estimation of a non-cooperative target to enable the commercial development of ML algorithms for vision-based navigation . SPEED-UE-Cube builds on previous datasets, such as the Spacecraft PosE Estimation Dataset (SPEED) but is rendered using Unreal Engine (UE) and employs a 3U CubeSat as the target. It comprises 30,000 randomized training images and 1,186 sequential images of a rendezvous trajectory between the target and a servicer spacecraft. The trajectory dataset is generated and evaluated using a new open-loop Rendezvous, Proximity Operations, and Docking (RPOD) simulation architecture that can be used to evaluate a pose estimation Convolutional Neural Network (CNN) online. Pose labels for the target CubeSat with respect to the servicer accompany all images in the training and trajectory datasets to facilitate supervised learning. The results from a comparative analysis between the SPEED-UE-Cube training subset and SPEED and the evaluation of the trajectory subset within the RPOD simulation framework indicate that SPEED-UE-Cube is a more challenging dataset for pose estimation than SPEED, particularly concerning the prediction of the relative orientation of the target.

INTRODUCTION

Autonomous, vision-based spacecraft navigation is a key enabler for future capabilities in areas such as Rendezvous, Proximity Operations, and Docking (RPOD) and In-Space Servicing, Assembly, and Manufacturing (ISAM) at scale. A critical step in vision-based navigation is determining the target's position and orientation, also known as its pose. Machine Learning (ML) algorithms such as Convolutional Neural Networks (CNN), have been recently shown to accurately and efficiently characterize the pose of a target spacecraft using images obtained with a monocular camera.[1–4] The low Size, Weight, Power and Cost (SWaP-C) requirements for monocular cameras

[*]PhD Candidate, Department of Aeronautics and Astronautics, Stanford University
[†]Principal Application Engineer, MathWorks.
[‡]Senior Science and Education Application Engineer, MathWorks
[§]Application Engineer, MathWorks
[¶]Space Segment Manager, MathWorks
[‖]Application Engineer, MathWorks
[**]Product Manager, MathWorks
[††]Associate Professor, Department of Aeronautics and Astronautics, Stanford University

combined with the high performance of these pose estimation CNNs make them an increasingly attractive technology for future RPOD and ISAM missions.

However, developing and deploying these ML approaches requires large training and evaluation datasets. The challenge of acquiring true spaceborne images also drives a reliance on synthetic image datasets. Only a few such datasets exist today, such as the Spacecraft PosE Estimation Dataset (SPEED[5] and SPEED+[6]), the Satellite Hardware-In-the-Loop Rendezvous Trajectory dataset (SHIRT),[7,8] the SPAcecraft Recognition leveraging Knowledge of space environment (SPARK) dataset[9] and the Unreal Rendered Spacecrafts On-Orbit (URSO) dataset,[10] which provide labeled synthetic and hardware-in-the-loop images as a surrogate of actual spaceborne imagery. Although these datasets are publicly available, most are prohibited for commercial use, limiting the development of ML algorithms in the commercial space sector.

This work presents SPEED-UE-Cube, a new commercially available machine learning dataset for monocular pose estimation of a non-cooperative target. SPEED-UE-Cube addresses the growing interest in ML approaches for autonomous spacecraft navigation by increasing the amount of synthetic spaceborne imagery available for training and evaluation. It was developed by the Space Rendezvous Laboratory (SLAB) at Stanford University in partnership with MathWorks. SPEED-UE-Cube differs from its predecessors, SPEED and SHIRT, in several ways beyond its commercial availability. First, the images are rendered using Unreal Engine (UE) 5 instead of OpenGL, allowing greater control over the space scene. Second, it employs a 3U CubeSat model created by MathWorks as the target, deviating from previous datasets that employed a model of the Tango spacecraft from the PRISMA mission.[11] Figure 1 shows the CubeSat's 3D CAD model. The CubeSat has two extended solar panels and three antennae that provide limited structural asymmetry, making it a realistic yet challenging target for ML pose estimation algorithms.

SPEED-UE-Cube consists of two distinct subsets: a randomly distributed training dataset of 30,000 images of a target spacecraft, and a trajectory dataset of 1,186 sequential images that model a rendezvous scenario between the target and a servicer spacecraft. The training dataset is randomized to capture a range of background, illumination conditions, and relative CubeSat positions and orientations. It is split into an 80:20 training/validation split. This work also presents a comparative analysis of a spacecraft pose estimation CNN to verify the performance of the training dataset. It is implemented in MATLAB®* R2023b using the Deep Learning Toolbox™and is trained and evaluated on SPEED-UE-Cube and SPEED. Conversely, the trajectory dataset is a sequential test set designed to evaluate pose estimation algorithms on realistic RPOD scenarios instead of single, randomized images. It is generated and evaluated using an open-loop RPOD simulation pipeline that can further evaluate the robustness of a pose estimation CNN on additional trajectories. The training and trajectory datasets are publicly available at `https://purl.stanford.edu/hw812wb1641`. The MATLAB implementation of the pose estimation CNN is also available at `https://github.com/tpark94/speed-ue-cube-baseline`

The remainder of the paper is as follows. First, the dataset generation section provides details about the UE scene and any rendering properties, followed by details on the pose labels for the training dataset. Next, the performance analysis section shows the results from a pose estimation CNN trained and evaluated on the training dataset and compares these results to those obtained with SPEED. Then, the RPOD simulation section introduces the open-loop simulation framework used

*MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

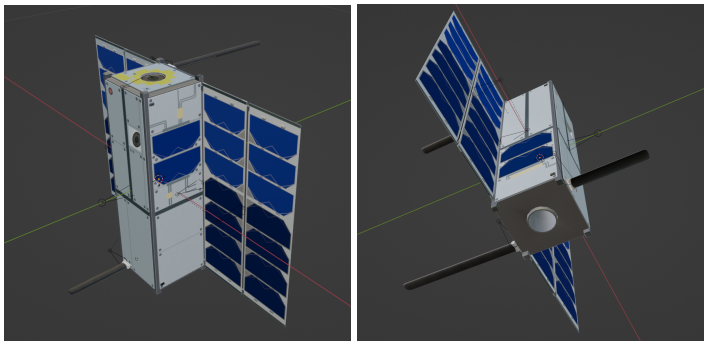


Figure 1: A 3D model of the target CubeSat created by MathWorks in collaboration with SLAB.

to generate and evaluate the trajectory dataset before providing CNN results when evaluated on the trajectory. Finally, the paper concludes with a contributions summary and describes how this work will be extended regarding the dataset and RPOD simulation.

DATASET GENERATION

This section describes the rendering environment used to generate SPEED-UE-Cube and details the training subset properties regarding the pose label distribution.

Unreal Engine Scene

Unlike the previous datasets developed by SLAB such as SPEED[5] and SPEED+[6] which used OpenGL to render synthetic images, SPEED-UE-Cube uses Unreal Engine 5 as its renderer. OpenGL rendered accurate images of a spacecraft such that the pixel intensity distributions of both OpenGL-based synthetic and spaceborne images with identical pose labels matched with high accuracy.[12,13] This was achieved without high-fidelity physics-based rendering, allowing fast rendering speed. Moreover, the real Earth images captured by the Himawari-8 geostationary (GEO) meteorological satellite were inserted in the background for additional realism. This means it is impossible to control the appearance of the Earth background based on the satellite's altitude and viewing direction. However, the background affects ML datasets only marginally, since the goal of an ML algorithm for pose estimation is to be agnostic to image backgrounds.

Compared to the OpenGL-based renderer for SPEED, UE can render a high-fidelity world with intricate physically-based rendering models. Specifically, UE allows construction of a realistic space scene with Earth, Sun and stellar backgrounds as it is a game engine developed with the goal of realizing high Frames Per Second (FPS) and high-quality scenes .

To generate synthetic training data, it is necessary to use an Application Programming Interface (API) to connect to the UE renderer. Depending on the version of UE, this can be a MathWorks, Python, C++, or custom API. SPEED-UE-Cube adopted a C++ API to connect to UE 5.

Actors

The constructed scene is a 1/10 scale of the universe whose reference frame coincides with the Earth-Centered Inertial (ECI) frame. It consists of several *actors*, essentially UE entities that can

be manipulated and moved around. These include the `CameraActor`, the `DirectionalLight` which acts as the Sun, and various `StaticMeshActors` for the CubeSat, Earth, Moon, and stars. Some of these actors are visualized in Figure 2. The high-resolution surface textures for the Earth and its clouds are obtained from the NASA Blue Marble collection [†] [‡] [§] [¶] The cloud texture is applied as a transparent material to a separate sphere mesh slightly larger than the Earth mesh. Therefore, the clouds can be rotated around the Earth's surface. The Earth also includes the UE built-in `SkyAtmosphere` component[‖] to emulate realistic atmosphere effects such as Rayleigh and Mie scattering.

Figure 2: The camera, CubeSat, Earth and Sun actors. This image depicts the scene as viewed from the UE editor instead of a rendered scene via the camera actor for visualization purposes.

Camera Effects.

The camera is modeled as a Point Grey Grasshopper 3 with a Xenoplan 1.4/17mm lens. Specifically, the camera has a 1920 $\times$ 1200 pixels resolution and the horizontal field of view (FOV) of 35.6°. This is the same camera used to create SPEED[5] dataset images. Ref. 5 provides more camera model details.

Typically, the scenes rendered by game engines tend to prioritize a better gaming experience than scientific accuracy. Therefore, it is important to model the camera physics accurately to emulate the spaceborne imagery properly which is typically characterized by high contrast and low signal-to-noise ratio (SNR). In this work, the field depth is adjusted based on the distance to the target when capturing the scene through the camera actor. It is slightly perturbed when creating the training

[†] `https://visibleearth.nasa.gov/collection/1484/blue-marble`

[‡] Earth Topography Sources: Jesse Allen, NASA's Earth Observatory, using data from the General Bathymetric Chart of the Oceans (GEBCO) produced by the British Oceanographic Data Centre, Reto Stöckli, NASA Earth Observatory.

[§] Earth Color Map Source: NASA Earth Observatory images by Joshua Stevens, using Suomi NPP VIIRS data from Miguel Romàn.

[¶] Cloud Texture Source: NASA Goddard Space Flight Center Image by Reto Stöckli. Enhancements by Robert Simmon. Data and technical support: MODIS Land Group; MODIS Science Data Support Team; MODIS Atmosphere Group; MODIS Ocean Group.

[‖] `https://docs.unrealengine.com/5.1/en-US/sky-atmosphere-component-in-unreal-engine/`

dataset to add some blurring effect due to miscalibrated depth of field. Additionally, the image is post-processed in UE with built-in random film grain noise, chromatic aberration and vignette**.

Training Dataset Pose Labels

The training dataset comprises 30,000 RGB images with an 80:20 training/validation split. These images were created by generating 30,000 ground-truth labels containing the metadata necessary to render the images first, including the inter-spacecraft pose and the locations of the servicer's camera, Earth, Sun, and Moon.

The target's relative position labels are created by uniformly sampling the distance along the z-axis (i.e., camera boresight) within [1.5, 15] (m) and the position along the xy-axes (i.e., image plane) such that the entire CubeSat fits within the image frame. Theoretically, the CubeSat would still appear resolved beyond 15 m for the PointGrey camera model. However, heavy occlusion due to shadowing makes it nearly impossible to identify its presence at extreme distances for a wide range of incident light angles. Therefore, 15 m was chosen as an ad hoc maximum distance representative of RPOD terminal phases. The relative orientation labels are randomly sampled from the $SO(3)$ space using the subgroup algorithm,[14] similar to SPEED. The position and orientation labels distributions are visualized in Figure 3.

For each image, the camera is placed at an arbitrary altitude randomly sampled from a range between 700 km in Low-Earth Orbit (LEO) and Geostationary Orbit (GEO). It is important to ensure that (1) the Earth does not block the Sun and that (2) the Sun is not in direct view of the camera when placing the camera randomly in the UE scene. This is achieved by ensuring two conditions on the camera's absolute position and orientation:

1. The angle between the vectors from the Earth to camera ($\boldsymbol{r}_{E\to C}$) and from the Earth to the Sun ($\boldsymbol{r}_{E\to S}$) can never be larger than $60°$, and
2. The angle between the camera boresight ($\hat{\boldsymbol{z}}_C$) and the vector from the camera to the Sun ($\boldsymbol{r}_{C\to S}$) can never be smaller than $75°$.

The first condition ensures that the Sun is never blocked by the Earth and always illuminates the CubeSat. Theoretically, this angle varies depending on the servicer's altitude and can be larger than $90°$. However, the lack of Earth's albedo light made it increasingly likely that the CubeSat would be

**https://docs.unrealengine.com/5.1/en-US/post-process-effects-in-unreal-engine/

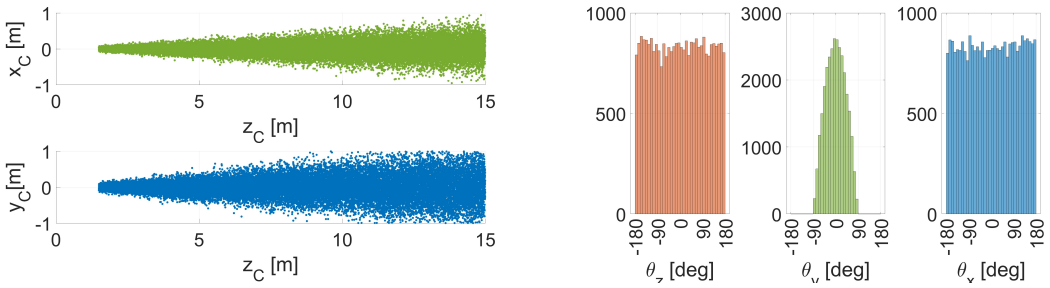


Figure 3: Position (*left*) and orientation (*right*) label distributions for SPEED-UE-Cube.

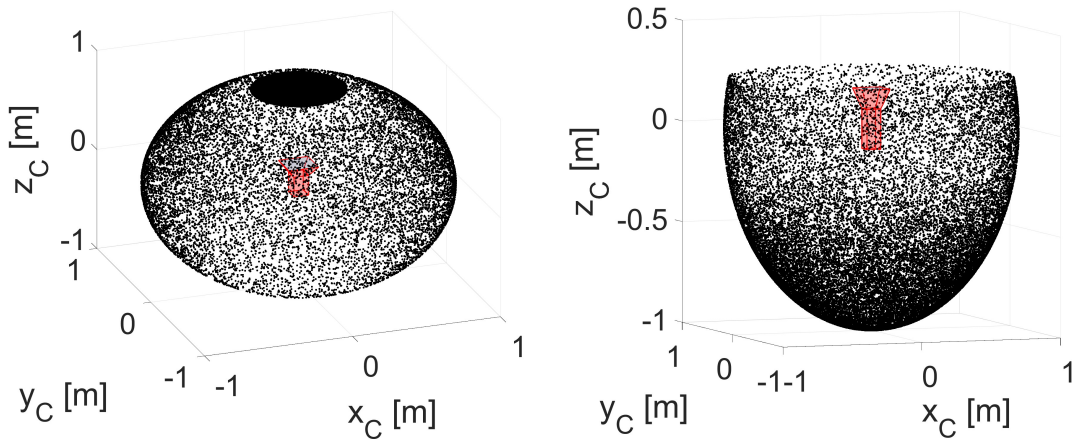


Figure 4: Earth (*left*) and Sun (*right*) normalized position distributions in the camera reference frame. Note that $+\hat{\boldsymbol{z}}_C$ coincides with the camera boresight.

under-illuminated at larger angular separations. Therefore, 60° was chosen as an ad hoc maximum angle between $\boldsymbol{r}_{E\to C}$ and $\boldsymbol{r}_{E\to S}$. The second condition ensures that the Sun is always outside the camera FOV. In fact, directly facing the Sun is not desired in real missions due to typical exclusion requirements and potential damage to the camera. The minimum 75° angular separation is also an ad hoc choice to ensure that the bare minimum of the CubeSat's surface is reflected back to the camera for visibility.

The training dataset is created while respecting the two conditions above. The first 15,000 images are rendered without the Earth in the background by forcing it outside the camera FOV. Conversely, the other 15,000 images are rendered forcing it *inside* the camera FOV. The dichotomy of the presence and absence of the Earth background is visualized in Figure 4 which shows the concentration of the Earth's normalized positions along the positive z-axis (i.e., the camera boresight). The figure also shows the distribution of the Sun's normalized positions which obey condition #2 (i.e., the Sun vector is never within the 75° angle from the camera boresight). Figure 5 shows sample images from the SPEED-UE-Cube training set depicting the variability in background, illumination conditions, and distance to the target.

PERFORMANCE ANALYSIS

This section describes the pose estimation performance studies used to evaluate SPEED-UE-Cube. The purpose of these studies was two-fold. First, they were used to understand the baseline performance of a Convolutional Neural Network (CNN) architecture trained on SPEED-UE-Cube compared to the similar, well-established SPEED dataset. Second, these analyses were used to identify any unacceptable outliers in SPEED-UE-Cube that should be manually removed. Unacceptable outliers were defined as any images where the CubeSat is completely occluded due to the illumination conditions and cannot be visually identified. Such images are distinct from acceptable outliers where the CubeSat is difficult to identify while some CubeSat features are still visible. The latter serve as challenging cases retained in the dataset to evaluate the robustness of the pose estimation algorithms. Overall, evaluating the outliers is important in ensuring that SPEED-UE-Cube is aligned with the boundary conditions under which a pose estimation CNN could be reasonably

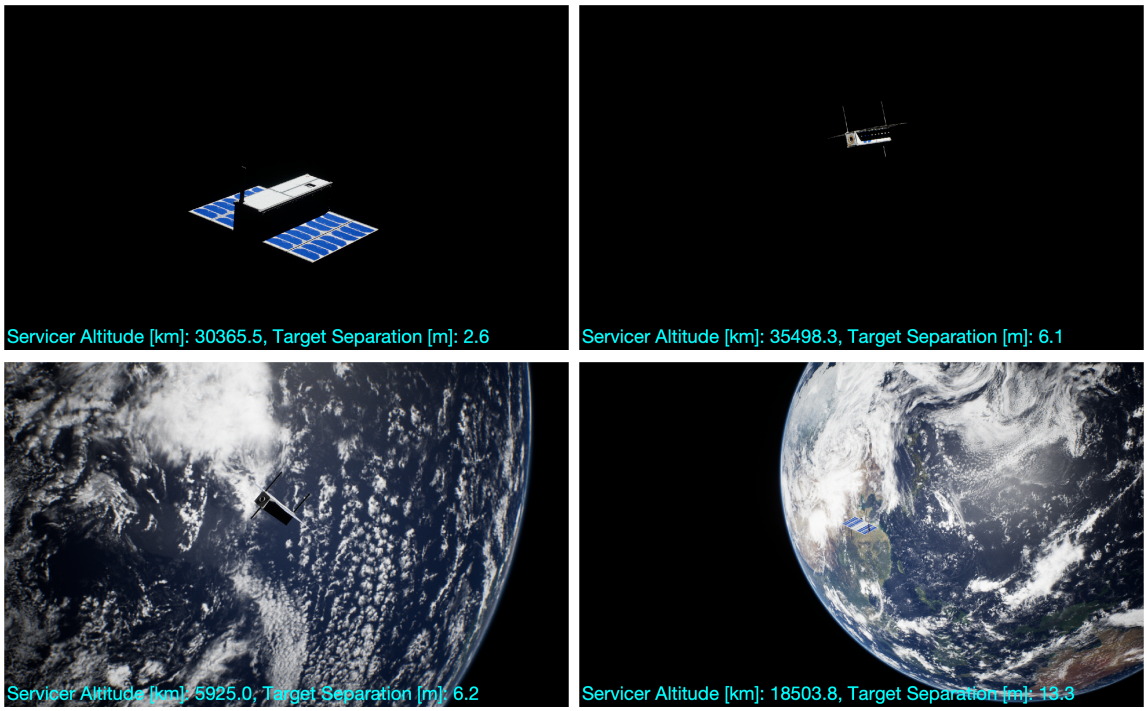


Figure 5: Sample images from SPEED-UE-Cube training dataset, which have been annotated with the orbital altitude of the servicer spacecraft and the separation between the servicer and target CubeSat. ‡,§, ¶

expected to operate.

MATLAB® Implementation

The training dataset was evaluated using the pose estimation pipeline introduced in Ref. 3. The pipeline consists of first cropping the images around the highest confidence bounding box detected using an object detection network (ODN). Then, the separate Keypoint Regression Network (KRN) is used to regress the locations of 11 pre-defined CubeSat keypoints. In this work, the ODN leverages the pre-trained YOLOv4 object detector from the Deep Learning Toolbox.[15] The KRN closely follows the original description, but its training hyperparameters follow those of the implementation described in Ref. 6 which was designed for training on the SPEED+ dataset. Finally, a Perspective-n-Point (PnP) algorithm-specifically EPnP[16]-is used to resolve the 6D pose of the CubeSat from the regressed keypoints and known 2D-3D keypoint correspondence. This entire pose estimation pipeline with the ODN, KRN, and EPnP will hereafter be referred to as ODN-KRN. The keypoints selected for this project represent the four corners of the solar panels, the four corners of the CubeSat body's top face, and the endpoints of the three antennae, as shown in Figure 6. Figure 7 shows a sample input image with the output of each step of the pipeline, namely the bounding box output of the ODN, the keypoints outputted from the KRN, and the satellite wireframe projected based on the estimated pose predictions from EPnP. Refs. 3 and 6 provide detailed information on ODN-KRN, data augmentation, and other hyperparameters.

Both YOLOv4 and EPnP have pre-built MATLAB implementations as part of the Deep Learn-

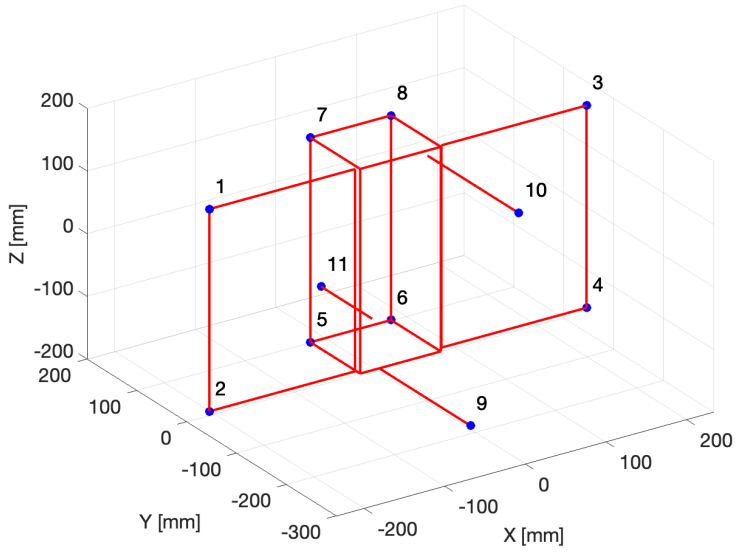


Figure 6: Keypoints labeled on CubeSat wireframe.

ing Toolbox and OpenCV-MATLAB interface, respectively. These pre-built features were used to perform the ODN and PnP steps of ODN-KRN. The KRN uses the pre-trained MobileNetv2[17] also available from the Deep Learning Toolbox. However, the prediction head was implemented from scratch using custom depth-wise convolution layers.[3] Its performance was compared to the 2022 PyTorch implementation[6] after both implementations were trained on the SPEED dataset[18] to verify the MATLAB implementation of the KRN with its custom layers . The readers are referred to Ref. 3 for the keypoints used for pose estimation of the Tango spacecraft. The data was pre-processed for this experiment using the ground truth bounding box to crop the images around the Region-of-Interest (RoI) instead of the ODN to compare the two KRN implementations directly. The performance was evaluated using two metrics: translation error (E_{T}) and orientation error (E_{R}), defined as[12]

$$E_{\mathrm{T}} = \|\tilde{\mathbf{t}} - \mathbf{t}\|_2 \tag{1}$$

$$E_{\mathrm{R}} = 2\arccos|<\tilde{\mathbf{q}}, \mathbf{q}>| \tag{2}$$

where $(\tilde{\mathbf{t}}, \tilde{\mathbf{q}})$ represents the predicted relative position and quaternion vectors of the target with respect to the camera, respectively, and $(\mathbf{t}, \mathbf{q})$ represent the corresponding ground truth values.

The KRN performance on SPEED for the PyTorch and MATLAB implementations is tabulated in Table 1. The mean values show good agreement, indicating that KRN is implemented correctly in MATLAB. The results are not identical due to different internal implementations of various operations within PyTorch and MATLAB, random seeds, and different EPnP implementations.

Training Dataset Performance

The entire ODN-KRN was trained and evaluated on SPEED-UE-Cube once the MATLAB implementation of the KRN was shown to be consistent with the original implementation. The errors

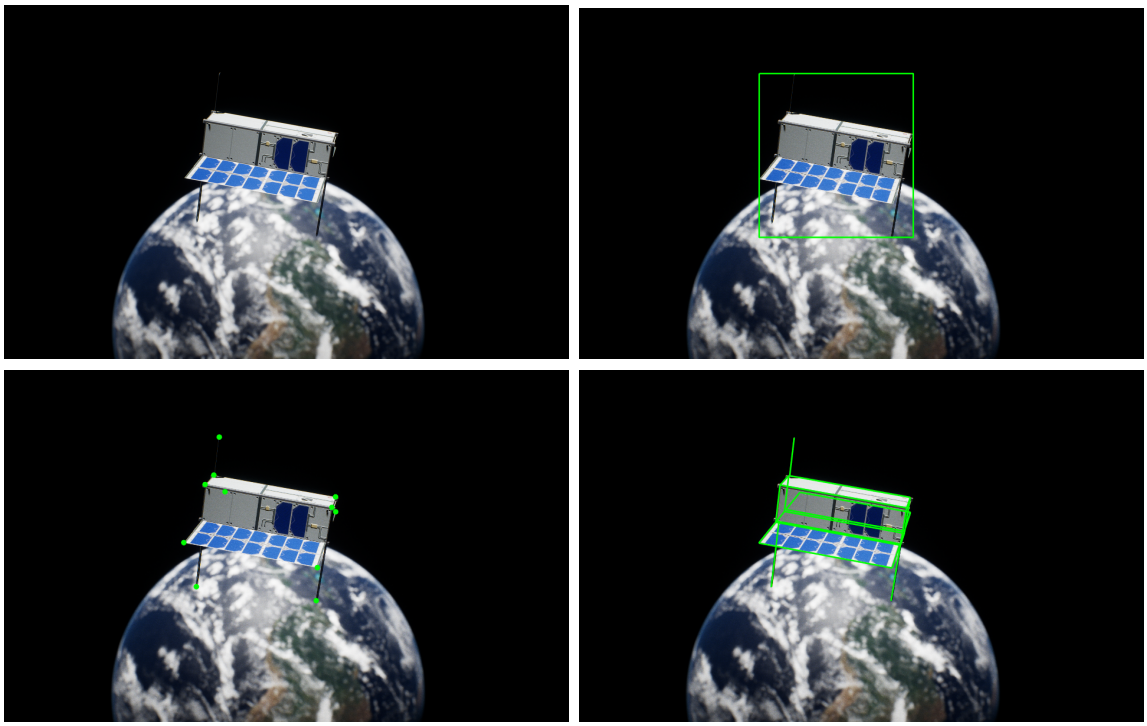


Figure 7: Sample input image (*top left*) with ground truth bounding box (*top right*), ground truth keypoints (*bottom left*) and projected wireframe (*bottom right*). ‡,§, ¶

for SPEED-UE-Cube were compared to those for SPEED to understand the relative difficulty of performing pose estimation on SPEED-UE-Cube.

The ODN and KRN were trained on the 80% training split of SPEED-UE-Cube, and the entire pipeline was evaluated using the 20% training dataset validation split. The KRN was trained using the same hyperparameters described in[6] though adjusted for the different dataset size. For example, since SPEED-UE-Cube is half the size of SPEED+, its training epoch is halved to 150 epochs for the batch size of 48. The learning rate decay factor is likewise adjusted to 0.96 so that the final learning rates at the end of the training are similar for both datasets.

The results are shown in Table 2 and include three additional metrics, the mean Intersection-over-

Table 1: KRN performance on SPEED. The translation and orientation errors are reported by the mean and standard deviation for the dataset. Data was pre-processed using ground truth bounding boxes for both models.

Implementation	$\mathbf{E_T}$ [m]	$\mathbf{E_R}$ [°]
PyTorch	0.212 ± 0.275	2.942 ± 2.160
MATLAB	0.229 ± 0.490	2.980 ± 2.356

Table 2: ODN-KRN performance for SPEED and SPEED-UE-Cube Training Subset.

Dataset	**IoU**	$\mathbf{E_T}$ [m]	$\mathbf{E_R}$ [°]	$\tilde{\mathbf{E}}_\mathbf{T}$	$\tilde{\mathbf{E}}_\mathbf{R}$
SPEED	0.903	0.218 ± 0.612	3.004 ± 2.206	0.016 ± 0.017	0.303 ± 0.310
SPEED-UE-Cube	0.870	0.290 ± 3.269	5.183 ± 10.017	0.016 ± 0.108	0.437 ± 1.270

Union (IoU), the normalized translation error ($\tilde{E}_T$) and the normalized rotation error ($\tilde{E}_R$). The IoU measures the area of intersection between the ground truth and predicted bounding box divided by their union to quantify the error of the bounding boxes predicted by the ODN. The normalized translation and rotation error are defined as

$$\tilde{E}_\mathrm{T} = \frac{E_T}{d} \tag{3}$$

$$\tilde{E}_\mathrm{R} = \frac{E_R}{d} \tag{4}$$

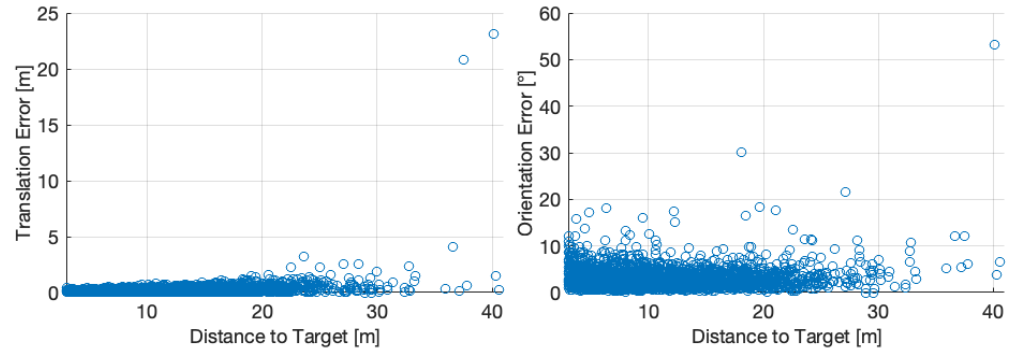
where d is the distance between the servicer and target normalized by the characteristic length of the target, or

$$d = \frac{\|\mathbf{t}\|}{l}. \tag{5}$$

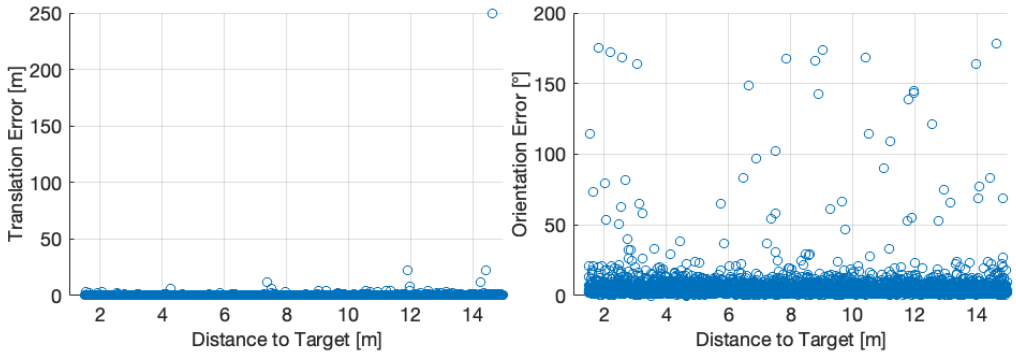
The normalized translation and rotation errors are important to account for the target distance and size variations between SPEED and SPEED-UE-Cube. While SPEED-UE-Cube limits the distance between the servicer and target to 15 meters, SPEED samples distances up to 50 meters. Additionally, the Tango model used in SPEED is larger than the 3U CubeSat used in SPEED-UE-Cube. The characteristic length for each target was taken as the width across the solar panels, 0.8 meters for Tango and 0.468 meters for the 3U CubeSat used in SPEED-UE-Cube.

The overall results indicate that it is slightly more difficult for ODN-KRN to perform pose estimation on SPEED-UE-Cube than on SPEED, particularly regarding the relative orientation. Moreover, the standard deviations indicate that the predictions on SPEED-UE-Cube suffer from more outliers than on SPEED, even when the results are normalized. This phenomenon is visualized in Figure 8 which plots the translation and orientation errors for each image in the validation split for SPEED and SPEED-UE-Cube as a function of the distance to the target. It does not appear that the outliers in the orientation error for SPEED-UE-Cube are correlated with the distance to the target. This indicates that these outliers could result from other factors such as the illumination conditions or ambiguities due to the symmetry of the CubeSat.

Another consideration is that the KRN with its regression-based keypoint detection pipeline is not a very robust pose estimation architecture. Therefore, it became necessary to verify whether the degradation of KRN's performance reported in Table 2 and Figure 8 is due to its own architectural inferiority or outlier dataset samples. A different CNN architecture based on SPNv2[19] was trained



(a) SPEED



(b) SPEED-UE-Cube – Training

Figure 8: Translation (*left*) and orientation (*right*) errors from the ODN-KRN plotted for SPEED (a) and the SPEED-UE-Cube training dataset (b).

on both SPEED and SPEED-UE-Cube to verify the integrity of the imagery. Specifically, the implemented architecture is a variant of SPNv2 with $\phi = 3$ and the heatmap prediction head. Additional details about various architectural choices can be found in Ref. 19. SPNv2 replaces the keypoint prediction step of the pose estimation pipeline only. Following the original SPNv2 implementation, the input is cropped around the ground-truth RoI then resized to 256×256 pixels instead of using bounding boxes predicted by an ODN. The training hyperparameters are modified from the original implementation such that SPNv2 is trained for 20 epochs on SPEED-UE-Cube and 40 epochs on SPEED using the AdamW optimizer.[20] The learning rate is linearly warmed up to 0.001 during the first epoch then decays every step according to the cosine annealing schedule for the rest of the training.[21]

The performance of SPNv2 is shown in Table 3. SPNv2 showed significant improvement in all error metrics and a reduction in the number of outlier images. This is unsurprising, since SPNv2 detects heatmaps associated with keypoints instead of directly regressing the numbers corresponding to the location of those keypoints which is less robust. However, the same trend observed for KRN is seen here. Namely, SPEED-UE-Cube suffers from more outliers than SPEED, especially for the rotation errors as evidenced by the order of magnitude higher standard deviations for SPEED-UE-Cube, even once normalized for the distance and target size. Overall, Table 3 verifies that the

degradation of CNN performance on SPEED-UE-Cube compared to SPEED is expected and not a defect in KRN design.

Table 3: SPNv2 performance when trained and evaluated on SPEED-UE-Cube. Data were pre-processed using ground-truth bounding boxes for both models.

Dataset	$\mathbf{E_T}$ [m]	$\mathbf{E_R}$ [°]	$\tilde{\mathbf{E}}_\mathbf{T}$	$\tilde{\mathbf{E}}_\mathbf{R}$
SPEED	0.068 ± 0.085	0.972 ± 0.674	0.003 ± 0.002	0.109 ± 0.062
SPEED-UE-Cube	0.055 ± 0.144	1.252 ± 6.527	0.005 ± 0.004	0.098 ± 0.103

Finally, to distinguish the SPEED-UE-Cube outliers as either acceptable or unacceptable, the images with the highest orientation errors returned by SPNv2, show in Figure 9, were manually evaluated. All these outlier images were instances where Earth was outside of the camera's FOV and where the CubeSat antennae were at least partially occluded. However, these images were determined to include enough visible features, such as the edges of the CubeSat, to be acceptable and were therefore retained in the dataset.

RPOD SIMULATION

In order to ensure the robustness of any pose estimation CNN for future RPOD missions, it is important that the algorithms be evaluated on sequential trajectory data in addition to single, randomized images. To this end, this section first introduces an open-loop simulation framework that serves as a testing environment for evaluating a pose estimation CNN online given a desired RPOD scenario. Then, this section introduces the trajectory subset of SPEED-UE-Cube generated and evaluated using this framework. Thus, the trajectory dataset demonstrates the feasibility of the simulation framework and serves as a sequential test set to complement the randomly distributed training dataset introduced earlier.

Open-Loop System Architecture

The open-loop simulation framework is authored in System Composer™, which enables system and software architecture definition with native Simulink®integration. The framework consists of three main components, shown in Figure 10. First, it requires a translational and rotational spacecraft dynamics model for the servicer and target spacecraft. Given an initial orbit for each spacecraft, this system must output the relative pose of the target with respect to the servicer, the absolute pose of the servicer, as well as the orientation of the Earth and Sun in the inertial frame. These outputs can then be provided to the second component, an Unreal Engine interface that takes in the position and orientation of all actors in the scene and produces a rendered image. Finally, the rendered image can be passed to the pose estimation CNN to predict the target pose. This process can be repeated for the duration of an arbitrary RPOD trajectory and the errors in the predicted pose can be used to evaluate the CNN's performance.

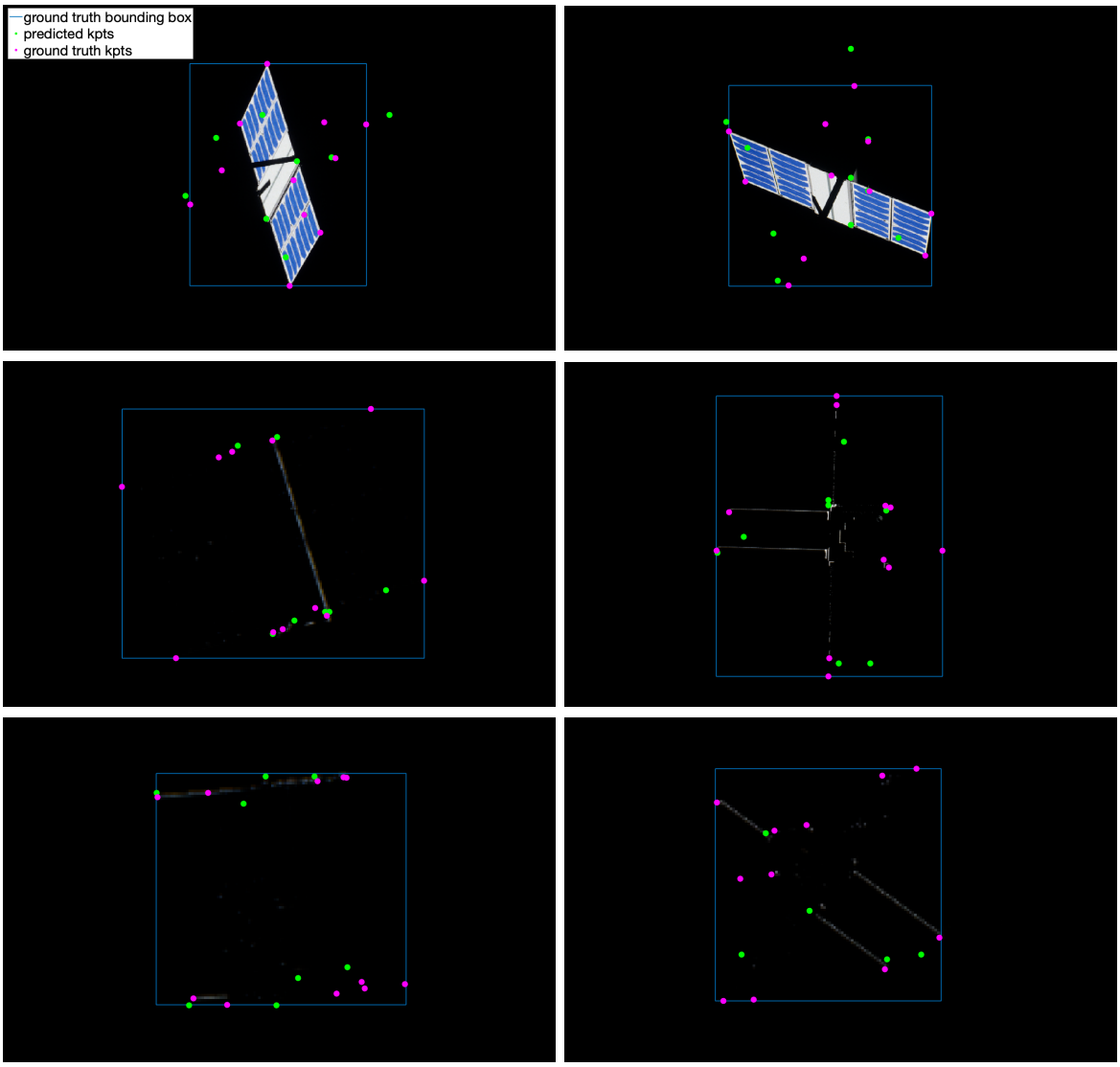


Figure 9: Instances with the highest rotation error from SPNv2. Images are shown cropped closer to the ground truth bounding box with predicted and ground truth keypoints.

Trajectory Dataset

The pipeline in Figure 10 was implemented using the SLAB Satellite Software (S^3)[22] to model the spacecraft dynamics for the trajectory subset of SPEED-UE-Cube. The images were rendered using the same C++ UE API used for the training dataset. Finally, the trajectory was evaluated on both the ODN-KRN and SPNv2 pose estimation CNNs trained using the SPEED-UE-Cube training subset.

RPOD Trajectory Generation

The rendezvous trajectory is simulated similarly to the SHIRT dataset.[7] First, the absolute orbit

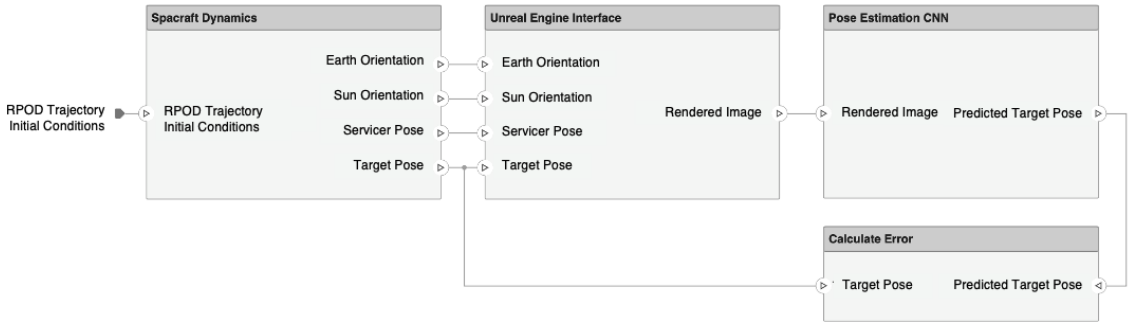


Figure 10: Open-Loop RPOD Simulation Architecture.

shown in Table 4 resembles that of SHIRT with the identical initial epoch at 2011/07/18 01:00:00 UTC. However, its inclination is modified to 67° so that the vector from the Earth to the Sun is approximately normal to the orbital plane and that the CubeSat is always illuminated throughout the trajectory. Table 4 also shows the initial quasi-nonsingular relative orbital elements (ROE)[23] indicating that the target is in an e/i-vector-separated passively safe relative orbit.[24] The trajectory lasts for 1 full orbit with a 5 seconds measurement interval, resulting in 1,186 images total. The evolution of the 3D relative Cartesian position vector in the servicer's Radial-Tangential-Normal (RTN) frame is visualized in Figure 11. The trajectory was simulated with S^{3}[22] with high-fidelity environmental force and torque models identical to those used for the SHIRT dataset. The only difference is that the parameters for the target, such as mass and ballistic coefficients, are adjusted to a 3U CubeSat. The force and torque models and the CubeSat parameters used to generate the trajectory are summarized in Tables 5 and 6. However Section V of Ref. 7 contains additional details.

Similar to SHIRT, the target's initial relative attitude is set to $\boldsymbol{q}_o = [1/\sqrt{2} \;\; 1/\sqrt{2} \;\; 0 \;\; 0]^\top$ which rotates at 1°/sec about its z-axis. However, unlike SHIRT, the camera's viewpoint is manually fixed to point at the CubeSat without attitude control.

Table 4: Initial mean absolute orbital elements of the servicer and relative orbit elements of the target with respect to the servicer.

Servicer Mean OE						**Target Mean ROE [m]**					
a [km]	e [-]	i [°]	Ω [°]	ω [°]	M [°]	$a\delta a$	$a\delta\lambda$	$a\delta e_x$	$a\delta e_y$	$a\delta i_x$	$a\delta i_y$
7078.135	0.001	67	189.9	0	0	0	0	0	7	0	3

RPOD Trajectory Pose Estimation Results

The trajectory was evaluated on the ODN-KRN and SPNv2 pose estimation CNNs after rendering the images given by the RPOD trajectory using the same UE scene as for the training dataset. The results in Table 7 show an increase in the translation and orientation errors as compared to the

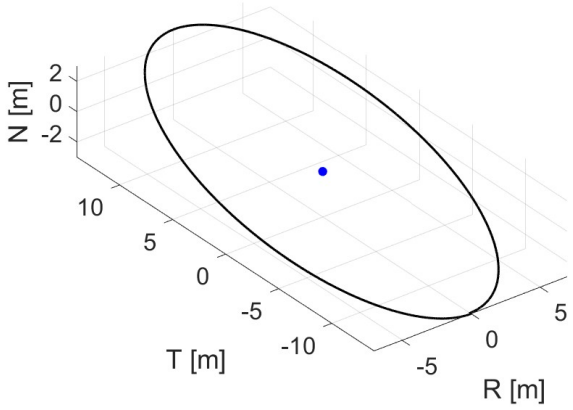


Figure 11: Relative position in the servicer's RTN frame.

Table 5: RPOD simulation force and torque models, recreated from Ref. 7.

Force Models	
Geopotential field (degree x order)	GGM05S (120 x 120)
Atmospheric density	NRLMSISE-00
Solar radiation pressure	Cannon-ball, conical Earth shadow
Third-body gravity	Analytical Sun & Moon
Relativistic effect	1st order
Torque Models	
Gravity gradient	Analytical
Atmospheric density	NRLMSISE-00
Solar radiation pressure	Conical Earth shadow
Geomagnetic field (order)	IGRF-13 (10)

validation split of the SPEED-UE-Cube training dataset shown in Table 2. Similar to the training dataset, the performance across all error metrics improves when the trajectory dataset is evaluated on SPNv2 instead, also shown in Table 7, showing good agreement with the SPEED-UE-Cube mean values shown in Table 3. However, SPNv2 still shows an increase in standard deviations for translation and rotation errors when evaluated on the trajectory dataset, corroborating the ODN-KRN performance trend. This increase may be attributed to the difference in the distribution of images between the training and trajectory dataset. While the training dataset was distributed equally between Earth inside and outside the camera's FOV, the trajectory dataset distribution is driven by the spacecraft dynamics and the orbit properties instead. The Earth is outside of the camera's FOV more than half of the time for the given trajectory, which typically results in higher errors especially when the target is oriented such that it can only be viewed edge-on, as shown by the instances of highest rotation error in Figure 9.

Table 6: 3U CubeSat Parameters.

Force Model Evaluation	
Spacecraft mass [kg]	4.5
Cross-sectional area (drag) [m^2]	0.10
Cross-sectional area (SRP) [m^2]	0.10
Aerodynamic drag coefficient	2.25
SRP coefficient	1.2
Torque Model Evaluation	
Number of faces	4
Principal moment of inertia [kg·m^2]	diag(0.050, 0.038, 0.021)
Direction Cosine Matrix (DCM) from body to principal frame	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Magnetic dipole moment [A·m^2]	$[0, 0, 5.677 \times 10^{-7}]^{\mathrm{T}}$

Figure 12 plots the individual components of translation and orientation errors from the ODN-KRN and SPNv2 for the trajectory subset as a function of time. Here, the Euler angles computed from the error rotation matrix represent the orientation error components. Importantly, the translation error is dominated by the z-component which coincides with the distance along the camera boresight. This observation also agrees with the analysis of the original KRN performed in Ref. 3. Furthermore, Figure 12 shows how the number of outliers decreases significantly between the ODN-KRN and SPNv2, highlighting the role of a robust CNN architecture in minimizing pose estimation errors. While the purpose of the open-loop RPOD framework from Figure 10 is to evaluate the CNN performance on its own, it is important to note that the outliers in Figure 12 can be further reduced by incorporating a navigation filter, such as an unscented Kalman filter (UKF), into the loop. In this case, the CNN predictions would serve as a course initialization into the filter. This approach has been shown to improve pose estimation performance even when the test data originates from a different domain beyond synthetically rendered images.[7]

Overall, the trajectory dataset provides a challenging test set for SPEED-UE-Cube beyond the randomly distributed training images and highlights the importance of evaluating pose estimation algorithms on sequential data indicative of real-world RPOD scenarios.

Table 7: ODN-KRN and SPNv2 performance for trajectory subset of SPEED-UE-Cube. For SPNv2, the data is pre-processed using ground-truth bounding boxes.

CNN	IoU	$\mathbf{E_T}$ [m]	$\mathbf{E_R}$ [°]	$\tilde{\mathbf{E}}_\mathbf{T}$	$\tilde{\mathbf{E}}_\mathbf{R}$
ODN-KRN	0.842	0.689 ± 7.002	13.212 ± 24.198	0.028 ± 0.236	0.643 ± 1.299
SPNv2	–	0.106 ± 0.691	1.846 ± 8.768	0.004 ± 0.024	0.088 ± 0.492

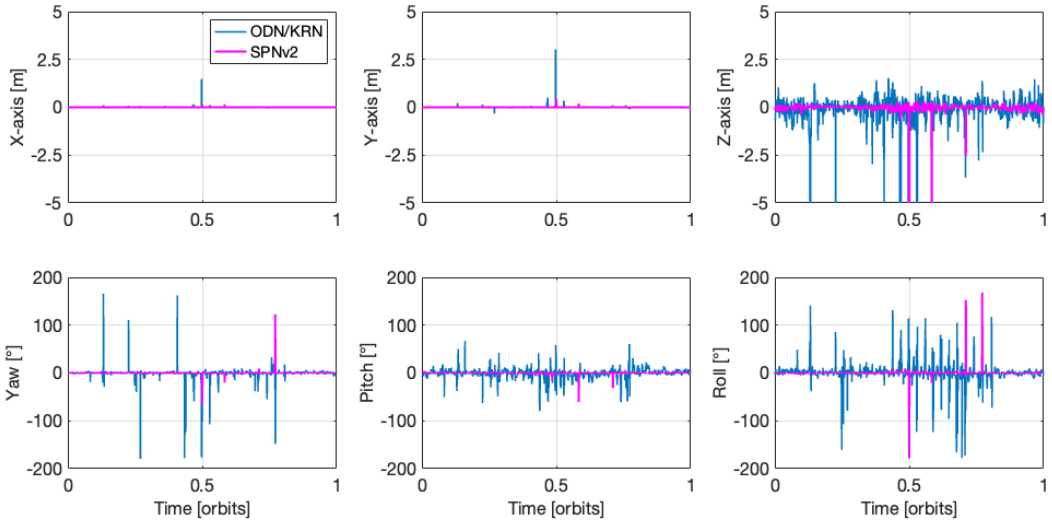


Figure 12: Translation (*top*) and orientation (*bottom*) error components for the trajectory subset of SPEED-UE-Cube as a function of time when evaluated on the ODN-KRN and SPNv2. Translation errors are given in the servicer's camera frame, where the boresight is along the z-axis.

CONCLUSION

This work describes the generation of SPEED-UE-Cube, the Unreal Engine (UE)-based Spacecraft PosE Estimation Dataset (SPEED) of a 3U CubeSat. SPEED-UE-Cube was developed by Stanford University's Space Rendezvous Lab (SLAB) in partnership with MathWorks and includes two distinct sub-sets: a training dataset composed of 30,000 images and a trajectory dataset that consists of 1,186 images in a rendezvous scenario. All images are accompanied with the position and orientation (i.e., pose) labels of the target CubeSat with respect to the camera. The report also provides performance analyses of a Convolutional Neural Network (CNN) model when trained and tested on the training subset of SPEED-UE-Cube. Specifically, the integrity of both SPEED-UE-Cube and the MATLAB implementation of a CNN model was verified via a comparative study with respect to a benchmark dataset and CNN model. The trajectory dataset was generated using a novel Rendezvous, Proximity Operations, and Docking (RPOD) open-loop simulation that can be used to evaluate a trained pose estimation CNN. The comparative analyses and evaluation of the trajectory dataset via the RPOD framework reveal that SPEED-UE-Cube has more challenging outlier

images than the benchmark SPEED dataset. This observation was corroborated by the performance analyses of two different CNN models.

In the future, the UE rendering capability will be extended to improve the realism of imagery and better capture the visual characteristics inherent to spaceborne imagery. Furthermore, it will be expanded to render additional outputs, such as semantic masks, depth fields, etc., to support a wide range of spaceborne computer vision tasks beyond pose estimation. Additionally, the RPOD open-loop simulation framework will be fully implemented and tested in MATLAB and Simulink to allow for continuous evaluation of the CNN given an arbitrary RPOD trajectory. Finally, this open-loop RPOD framework will be extended to a closed-loop simulation incorporating a navigation filter and guidance and control.

Ultimately, SPEED-UE-Cube and the RPOD simulation framework presented in this work are important tools that will enable the development and increase accessibility of robust autonomous, vision-based navigation algorithms using machine learning.

ACKNOWLEDGMENT

This work is partially funded by MathWorks and is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

REFERENCES

[1] S. Sharma, C. Beierle, and S. D'Amico, "Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks," *2018 IEEE Aerospace Conference*, 2018, pp. 1–12.

[2] P. F. Proença and Y. Gao, "Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6007–6013.

[3] T. H. Park, S. Sharma, and S. D'Amico, "Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft," 2019.

[4] B. Chen, J. Cao, A. Parra, and T.-J. Chin, "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement," *ICCVW*, 2019.

[5] 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, 2020, pp. 4638–4658.

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

[7] 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, 2023, pp. 1671–1688.

[8] T. H. Park and S. D'Amico, "SHIRT: Satellite Hardware-In-the-loop Rendezvous Trajectories Dataset," Stanford Digital Repository, 2022. Available at `https://purl.stanford.edu/zq716br5462`.

[9] A. Rathinam, V. Gaudilliere, M. A. Mohamed Ali, M. Ortiz Del Castillo, L. Pauly, and D. Aouada, "SPARK 2022 dataset : Spacecraft detection and trajectory estimation," 2022.

[10] P. F. Proença and Y. Gao, "Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6007–6013.

[11] S. D'Amico, P. Bodin, M. Delpech, and R. M.Sc, *PRISMA*, pp. 599–637. 08 2013.

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

[13] S. Sharma, *Pose estimation of uncooperative spacecraft using monocular vision and deep learning*. PhD thesis, 2019.

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

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[16] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, Vol. 81, 02 2009.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
[18] S. Sharma and S. D'Amico, "Pose Estimation for Non-Cooperative Rendezvous Using Neural Networks," 2019.
[19] T. H. Park and S. D'Amico, "Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap," *Advances in Space Research*, mar 2023.
[20] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *International Conference on Learning Representations*, 2019.
[21] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," *International Conference on Learning Representations*, 2017.
[22] V. Giralo and S. D'Amico, "Development of the Stanford GNSS Navigation Testbed for Distributed Space Systems," *Institute of Navigation, International Technical Meeting, Reston, Virginia*, January 29 - February 1 2018.
[23] A. W. Koenig, T. Guffanti, and S. D'Amico, "New State Transition Matrices for Spacecraft Relative Motion in Perturbed Orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1749–1768.
[24] S. D'Amico and O. Montenbruck, "Proximity Operations of Formation-Flying Spacecraft Using an Eccentricity/Inclination Vector Separation," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 3, 2006, pp. 554–563.