Bridging Domain Gap for Flight-Ready Spaceborne Vision

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

This work presents Spacecraft Pose Network v3 (SPNv3), a Neural Network (NN) for monocular pose estimation of a known, non-cooperative target spacecraft. As opposed to existing literature, SPNv3 is designed and trained to be computationally efficient while providing robustness to spaceborne images that have not been observed during offline training and validation on the ground. These characteristics are essential to deploying NNs on space-grade edge devices. They are achieved through careful NN design choices, and an extensive trade-off analysis reveals features such as data augmentation, transfer learning and vision transformer architecture as a few of those that contribute to simultaneously maximizing robustness and minimizing computational overhead. Experiments demonstrate that the final SPNv3 can achieve state-of-the-art pose accuracy on hardware-in-the-loop images from a robotic testbed while having trained exclusively on computer-generated synthetic images, effectively bridging the domain gap between synthetic and real imagery. At the same time, SPNv3 runs well above the update frequency of modern satellite navigation filters when tested on a representative graphical processing unit system with flight heritage. Overall, SPNv3 is an efficient, flight-ready NN model readily applicable to a wide range of close-range rendezvous and proximity operations with target resident space objects. The code implementation of SPNv3 will be made publicly available.

I. Introduction

The autonomous Rendezvous, Proximity Operations and Docking (RPOD) capability with non-cooperative Resident Space Objects (RSO) is a core technological requirement for various future space missions. Aimed at sustainable space development, these missions include on-orbit servicing and refueling operations such as now-canceled OSAM missions by NASA [1, 2] and active debris removal such as RemoveDEBRIS [3] by Surrey Space Center, ADRAS-J [4] by Astroscale and ClearSpace-1 [5] by ClearSpace SA. The RPOD capability with such targets—defunct satellites, debris, etc.—requires accurate, real-time knowledge of the position and orientation (i.e., *pose*) of the target spacecraft with respect to the servicer spacecraft. In these scenarios, the *non-cooperative* nature of the target implies that it is without active communication links or aiding fiduciary markers on its surface to guide the rendezvous process, such that the servicer must be able to estimate and track the target's pose using the onboard sensors and computers only. Given such constraints, a monocular camera is an attractive choice of sensor due to its low Size-Weight-and-Power-Cost (SWaP-C), which is suitable for miniaturized systems such as SmallSats and CubeSats. Moreover, low SWaP-C implies additional sensor redundancy and fail-proofing compared to more complex sensor systems such as LIght Detection And Ranging (LIDAR) and stereovision which requires careful calibration.

Recent years have seen a significant breakthrough in Machine Learning (ML)-based monocular pose estimation of a known, non-cooperative spacecraft. Training of these ML models such as Convolutional Neural Networks (CNN) generally requires a large-scale dataset of images and pose labels. However, access to space is difficult and expensive, and a close-range rendezvous of two satellites is in itself a very rare occasion, making in-situ data collection and label annotation a challenging task for spaceborne applications. In response, the aerospace community has adopted a methodology whereby synthetic images are generated with rendering tools and gaming engines such as OpenGL and Unreal Engine in order to train the neural networks [6–9]. However, synthetic images have inherently dissimilar visual features compared to real spaceborne images characterized by low Signal-to-Noise Ratio (SNR) and harsh illumination conditions. In addition, the degradation of vision cameras and target RSOs due to operations in the space environment is difficult to model prior to the mission and through computer graphics. This results in the problem known as *domain gap*, which manifests as a drop in performance when the models are tested on the Out-Of-Distribution (OOD) data sampled from a statistical distribution different from that of the training data [10, 11]. In case the models are trained

*Ph.D. Candidate, Department of Aeronautics & Astronautics, 496 Lomita Mall; tpark94@stanford.edu. Student Member AIAA.
†Associate Professor, Department of Aeronautics & Astronautics, 496 Lomita Mall. Associate Fellow AIAA.

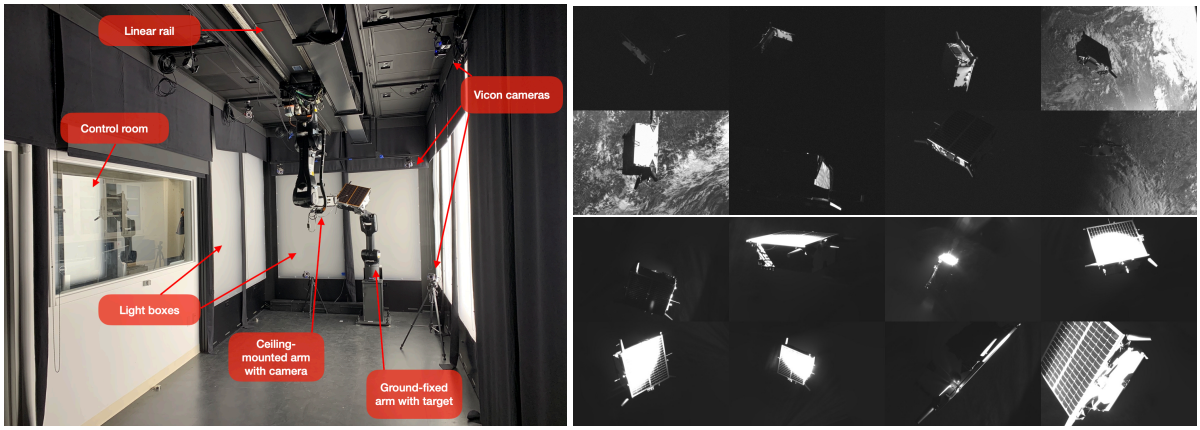


Fig. 1 (*left*) TRON simulation room and its components. Figure from Park et al. [14]. (*right*) Example HIL images from the `lightbox` (*top*) and `sunlamp` domains (*bottom*) of the SPEED+ dataset [15].

with synthetic images and tested on real-life images, it is also known as *reality gap* [12] or *sim2real gap* [13] primarily in robotics literature.

The inaccessibility of space poses a particularly challenging logistics problem to not only overcoming the domain gap but also verifying it during the stringent pre-flight verification processes typically required of space missions. The most prominent approach taken by the aerospace community is to utilize a robotic testbed to simulate the space environment on-ground. The idea is to physically stimulate the vision-based sensors using a mockup model of the target placed in a facility simulating the high-fidelity space-like illumination conditions. A well-calibrated testbed can then be used to re-create various RPOD scenarios, generating pose-annotated images at scale with minimal human intervention for the purpose of validating a NN's robustness across the sim2real gap. One such example is the Testbed for Rendezvous and Optical Navigation (TRON) facility at the Stanford's Space Rendezvous Laboratory (SLAB)* [14] which was used to create the so-called Hardware-In-the-Loop (HIL) images that constitute datasets such as SPEED+ [15] and SHIRT [16]. Figure 1 shows the TRON facility and a few example images from the two HIL domains of the SPEED+ dataset—`lightbox` and `sunlamp`. In contrast to SPEED+ which contains static images at random poses, the SHIRT dataset includes sequential `lightbox` images obtained during dynamic RPOD scenarios. This allows the testing of navigation filters with ML algorithms in the loop. Overall, these HIL images can then be used as on-ground surrogates of otherwise unavailable spaceborne images for the evaluation of NN and the navigation algorithm robustness across domain gaps. SPEED+ was used as the core dataset of the second Satellite Pose Estimation Competition (SPEC2021)† co-organized by SLAB and the Advanced Concepts Team (ACT) of the European Space Agency (ESA), in which the participants were tasked to predict the poses on SPEED+ HIL images while only having access to the labeled `synthetic` and unlabeled HIL domain images [17].

Given such tools and datasets, the authors have explored a strategy to fully close the domain gap between synthetic and spaceborne images. The strategy consists of three steps. The first is to train a NN model using only synthetic images to be as robust as possible to the unknown spaceborne images [18]. The robustness is evaluated on-ground using HIL domain images which are excluded from the training. Then, the NN is integrated as a measurement module into an Adaptive Unscented Kalman Filter (AUKF) which tracks the pose of the target [16]. While it was shown that a well-designed AUKF allows robust tracking of the target's pose despite the domain gap suffered by its NN measurement module, there is still a remaining gap between the HIL imagery used for on-ground evaluation and the spaceborne flight images. The primary reason is that HIL images use an inexpensive mockup model of the target spacecraft which has different material and surface properties compared to the real satellite. Therefore, Park and D'Amico [19] proposed to additionally fine-tune the NN weights using the flight images collected during in-space RPOD. The fine-tuning is done online in a supervised manner whereby the pose pseudo-labels are obtained from the most up-to-date state estimates of the onboard AUKF. This work has shown that Online Supervised Training (OST) can fully close the domain gap in the orientation prediction even when using a sub-optimally trained NN.

*`https://slab.stanford.edu/`
†`https://kelvins.esa.int/pose-estimation-2021/`

Despite the success of OST, there are benefits to further maximizing the robustness of onboard NN during the offline training on synthetic images. The first is that OST requires that the onboard ML-in-the-loop navigation filter be able to first converge to steady-states so that accurate pose pseudo-labels can be generated from the state estimates. The probability of failure to converge is minimized as the OOD robustness of the onboard NN is maximized prior to deployment. The second is that training a NN onboard the limited computing environment of satellite avionics could be expensive both in terms of the computational efficiency and power consumption. This could pose a logistical challenge to schedule the NN training amidst the nominal Guidance, Navigation and Control (GN&C) operations. These two reasons motivate

1) a training mechanism that renders the NN as robust as possible across domain gap from the offline training alone, which effectively minimizes the required number of OST steps performed in space, and
2) a NN design that is computationally efficient for both inference and online training while maintaining the maximum OOD robustness.

These two objectives—OOD robustness and computational efficiency—are at odd, since it is well known that larger NNs with higher capacity tend to perform better on both in-distribution and OOD data [20, 21].

In response to the above challenge, the main contribution of this work is Spacecraft Pose Network v3 (SPNv3), a NN model based on Vision Transformer (ViT) [22] for monocular pose estimation of a non-cooperative spacecraft. SPNv3 is designed and trained with onboard computational efficiency and robustness across the sim2real gap as top priorities. Extensive ablation studies of SPNv3 on the SPEED+ dataset demonstrate that ViT-based models without convolution operations dominate the computational efficiency front, and that their OOD robustness on HIL domain images can be improved by employing extensive data augmentation, enhanced transfer learning and increased input image resolution with minimal gain of computational overhead for inference. A comprehensive set of experiments reveals that the proposed SPNv3 can achieve state-of-the-art performances on the HIL domain images of the SPEED+ dataset while training exclusively on its `synthetic` images. Most importantly, a mid-size variant of SPNv3 would rank the first place on the `lightbox` category of SPEC2021 *without* any adversarial training or unsupervised domain adaptation, the methods which all winners employed by directly including the unlabeled test domain images into the training process. Such state-of-the-art robustness is achieved while taking no more than 40 ms per inference on an NVIDIA Jetson Nano 4GB [23], a representative, restricted compute environment similar to GPU-powered systems on satellite missions [24, 25].

This paper is organized as follows. Section II provides a brief overview of literature on ML-based monocular pose estimation and existing methods to bridge domain gap. Section III then goes over three key requirements that must be met by spaceborne ML models in addition to OOD robustness, two of which drive various design and training options for SPNv3 introduced in Section IV. Section V shows extensive experiments on the SPEED+ dataset to identify key elements of the design and training of SPNv3 that contribute the most to its robustness on HIL domain images and computational efficiency. The paper ends with conclusions and future works in Section VI.

II. Related Work

A. ML Approaches to Spacecraft Pose Estimation

The first ML-based approach to pose estimation of a known target spacecraft was Spacecraft Pose Network (SPN) [6] which performs 1) relative attitude determination via a hybrid approach of attitude classification and regression and 2) translation estimation by exploiting the perspective transformation and geometric constraints. Since it is impossible to collect a pose-annotated image dataset from space, the same work introduced the Spacecraft PosE Estimation Dataset (SPEED) [26] which consists of 15,300 images of the Tango spacecraft from the PRISMA mission [27]. Specifically, the dataset comprises 1) 15,000 synthetic images rendered with OpenGL-based Optical Stimulator (OS) camera emulator software [28, 29] of SLAB multi-Satellite Software Simulator (S^3) [30] and 2) 300 *real* images of a mockup of the same target captured from the TRON facility at SLAB. The dataset was made publicly available as part of the first Satellite Pose Estimation Challenge (SPEC2019)[‡] [31] co-hosted by SLAB and ESA.

Many top-performing entries of SPEC2019 have adopted diverse CNN architectures and pose estimation strategies, such as probabilistic orientation estimation via soft classification [7] or estimation of 2D pixel coordinates of the designated keypoints on the spacecraft surface [32, 33]. Specifically, Chen et al. [33] and Park et al. [32], who respectively ranked first and fourth places in the challenge, independently proposed a three-stage architecture: 1) an

[‡]`https://kelvins.esa.int/satellite-pose-estimation-challenge/`

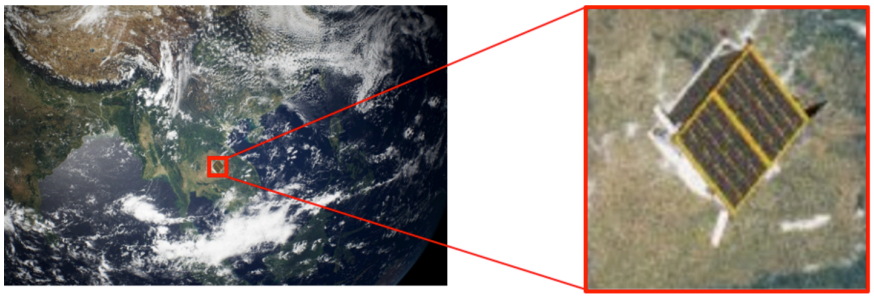


Fig. 2 Visualization of an image cropped around the far-away target spacecraft.

object detection CNN which is used to identify the Region-of-Interest (RoI) around the target, 2) a pose estimation CNN which takes in an image cropped around the detected RoI and outputs the 2D keypoint locations, and 3) a P*n*P module which solves for the full 6D pose based on the known correspondence of detected keypoint 2D locations and 3D model coordinates. Notably, Park et al. [32] directly regresses (x, y) coordinates of the keypoints, whereas Chen et al. [33] outputs a set of 2D heatmaps whose peaks correspond to the locations of the keypoints. They also showed that cropping the input image around the target using a detected bounding box is crucial to performing pose estimation of the far-away target. For example, the original resolution of the SPEED images is 1920×1200, while most CNNs for ImageNet [34] classification expect 224×224 inputs. Cropping around the target prior to such dramatic down-scaling helps preserve much of the detailed target features that would otherwise be lost due to large inter-spacecraft separation as visualized in Fig. 2. Many CNN models that followed SPEC2019 also adopt similar strategies as well [8, 35–38]. The readers are referred to Pasqualetto Cassinis et al. [39] for a more comprehensive review of monocular spacecraft pose estimation using both conventional and deep learning-based methods.

In addition to SPEED, a number of datasets for spacecraft pose estimation have been published and made publicly available in the literature. For example, Proença and Gao [7], who also ranked third place in SPEC2019, published the Unreal Rendered Spacecraft On-orbit (URSO) dataset which uses Unreal Engine 4 to render synthetic images of the Soyuz and Dragon spacecraft. Kaki et al. [37] renders synthetic images of the Cygnus spacecraft using Blender and its Cycles rendering engine. Dung et al. [40] also renders about 3,000 synthetic images of different satellites for various computer vision tasks such as bounding box prediction and satellite foreground and parts segmentation. Other authors also created their own synthetic datasets, such as those of the Envisat spacecraft rendered with Cinema 4D [38], the SwissCube dataset for spacecraft pose estimation from wide-depth-range images using the Mitsuba 2 renderer [36], the SPARK [8] dataset which contains images of 11 different spacecraft rendered with the Unity3D game engine, and Synthetic-Minerva II2 [41] which renders images of the Minerva-II2 rover from the Hayabusa2 mission [42] using SolidWorks' Photoview 360 renderer.

B. Algorithms to Bridge Domain Gap

The general approaches to tackling domain gaps can be put into two categories: domain adaptation and domain randomization. Specifically, Unsupervised Domain Adaptation (UDA) aims to close the gap by directly incorporating the *unlabeled* target domain images into the training phase [10, 11, 43–49]. Many different approaches have been proposed in the literature to close the said gap. One primary approach is to align the source and the target data distributions in the latent feature space by minimizing a certain divergence criterion representing the domain discrepancy, such as maximum mean discrepancy [45, 49–51], $\mathcal{H}$-divergence [10, 44], correlation via statistical moments [46, 52] or Wasserstein distance [53–55]. Another approach employs adversarial training [44, 48, 56] so that the predicted features become indistinguishable for both source and target domain inputs from the perspective of an auxiliary domain discriminator network. Note that the aforementioned approaches all aim to promote learning invariant representations by aligning both source and target domain data in the feature space.

One major shortcoming of UDA is that it requires simultaneous availability of the labeled source and unlabeled target domain data. This assumption violates the operational constraints of space missions as spaceborne images of the target do not become available until on-orbit rendezvous. Moreover, the processor and memory onboard the satellites are extremely limited for the full training session using large-scale synthetic source images and spaceborne target images collected during RPOD. Therefore, the conventional UDA approaches cannot be used for realistic space mission scenarios.

As opposed to UDA, domain randomization instead aims to randomize various aspects of the training images such that OOD test images would appear as another randomized instance of the training set [57–61]. For example, Sadeghi and Levine [58] and Tobin et al. [59] randomize lighting conditions, object textures and placements at rendering stage to train domain randomized reinforcement learning models. Jackson et al. [62] proposes style augmentation which randomizes the image texture [63] via neural style transfer [64]. While domain randomization does not require target domain images during the training phase, it also makes it difficult to provide any assurance that the CNN trained with domain randomization will work well on the target spaceborne images.

1. Source-Free Domain Adaptation

A different approach conducive to the operational constraints of space missions is *source-free* domain adaptation which does not require the availability of large-scale training images during the adaptation phase. Existing source-free algorithms leverage generative models for feature alignment [65–67] or pseudo-labeling and information maximization [68]. TENT [69] performs entropy minimization while updating only the affine parameters of the Batch Normalization (BN) layers [70]. However, methods based on entropy minimization require optimizations to be performed on batches of images in order to avoid trivial solutions, which could become computationally expensive on satellite avionics.

On the other hand, Test-Time Training (TTT) [71, 72] trains on a secondary Self-Supervised Learning (SSL) task from a shared feature encoder. During test time, the encoder is trained on SSL tasks, such as rotation prediction [73] or image colorization [74]. However, TTT generally requires a hand-designed task or a large batch of negative sample pairs (e.g., contrastive learning [75]). Finally, Lu et al. [76] self-trains an object pose estimator CNN using pseudo-labels from its own predictions. In order to improve the accuracy of pseudo-labels and mitigate outliers, they take a SLAM-based approach and solve the Pose Graph Optimization (PGO) problem to enhance the consistency of pseudo-labels across different images. However, PGO is an offline problem solving for a set of multiple poses satisfying the motion constraints, and collecting many images until pose pseudo-labels can be obtained via PGO could become computationally expensive on satellite avionics.

2. Spaceborne Applications

The approaches to bridging the domain gap in spaceborne applications have been explored recently after the introduction of SPEED+ [15] and SPEC2021 [17]. Two winners of the competition on respective HIL domain categories have both employed a generative-adversarial training procedure [77], introducing a discriminator network to the NN outputs to classify whether the predictions are made on the `synthetic` or HIL domain images. The winner of the `sunlamp` category further employed pseudo-labeling and self-training on the unlabeled HIL domain images [78]. Pérez-Villar et al. [79], who placed second place on both categories, also employed UDA and pseudo-labeling of the HIL domain images.

On the other hand, SPNv2 [18], one of the baseline models developed by the authors, designed a multi-scale, multi-task learning CNN architecture and trained with extensive data augmentations on the SPEED+ synthetic images only. The largest variant of SPNv2 would have ranked 3rd place in `lightbox` and 6th in `sunlamp` categories, respectively, without accessing the HIL domain images during training at all. Finally, EagerNet [80] performs dense predictions of pixel-wise object coordinates as opposed to heatmap predictions that many other methods adopted. They additionally predict the errors of predicted model coordinates which results in multiple pose hypotheses which are further refined using a probabilistic refinement. EagerNet trained with extensive data augmentation, including those specifically designed to target the HIL domain imagery, achieves state-of-the-art robustness without accessing the HIL domains, winning both HIL categories in the post-mortem competition.

As shown later, SPNv3 is able to achieve comparable robustness relative to EagerNet without access to the HIL domains as well. However, SPNv3 is a much simpler architecture which outputs heatmaps about known keypoints on the target surfaces, which can be provided directly to the onboard navigation filter [16] or be used to solve for the optimal pose solution via P*n*P [81].

III. Requirements of Spaceborne ML Models

Before introducing SPNv3, this section provides a brief overview of various requirements that must be met by not only SPNv3 but also ML models that are intended to run in space onboard satellite avionics. Some of these requirements, in addition to the OOD robustness across the sim2real gap, drive the design choices of SPNv3 in the next section.

A. Computationally Efficiency

The first requirement of a spaceborne ML model operating on satellite avionics is its computational efficiency. This is an obvious requirement for real-time operation of NNs in space whose runtime should ideally be within the update frequency of its overarching GN&C system. For example, Roscoe et al. [82] reports that the CubeSat Proximity Operation Demonstration (CPOD) mission launched in 2022 ran its RPO GN&C system at 0.5 Hz frequency with its star sensor and IMU data processed at up to 10 Hz for its Attitude Determination and Control System (ADCS) as part of an Extended Kalman Filter (EKF). However, the image processing algorithms often exceeded the onboard filter measurement update cycles, requiring a robust filter design that can handle out-of-sync and latent measurements. Taking the CPOD mission as a reference, this means the measurement processing of a NN, which includes not only its forward pass but also pre- and post-processing of its inputs and outputs, must be completed well within two seconds on a representative computing hardware with limited processing capabilities. Achieving this could be facilitated by the presence of an onboard GPU that can perform the NN inference and any other parallelizable operations, allowing the inference to be executed asynchronously from other GN&C operations running on Central Processing Units (CPU).

As explained in Section I, it may be desirable to perform online training of NNs in space to fully close the domain gap by directly incorporating the flight images that only become available during RPOD [19]. In this case, the NN must also be computationally efficient to run not only forward but also backward gradient propagation. Unlike inference, the training most certainly requires a GPU for the operation. However, the bright side is that the training need not run for every acquired image but only when there has been a substantial "change" in the imagery compared to the previous training round. The motivation is to avoid overfitting the NN to the specific scenery (e.g., Earth background) and view of the target since they do not change so abruptly within the short time frame in space especially at higher altitudes. Therefore, as long as there is a GPU onboard, the training latency is likely to be less of a computational bottleneck than inference which should be running in real time.

1. Remark on Flight Heritage GPU

Even nowadays, GPUs are still rarely found only in short-term missions which typically are not concerned with long-term radiation effects such as Total Ionising Dose (TID), or as part of a larger spacecraft which provides better protection of the onboard computing systems. Recent state-of-the-art report on small spacecraft technology by NASA has tabulated a number of GPU-based avionics [83, Table 8.1, §8.3], but many Commercially Off-The-Shelf (COTS) systems such as NVIDIA Jetson series are still not flight-proven or are radiation tolerant only against Single Event Effects (SEE) (e.g., bit flips).

However, there is an increasing interest in deploying GPUs into deep space or long-term missions in LEO with existing flight heritage [25]. For example, Aitech Systems, Inc., a company that manufactures rugged computers for military and aerospace applications, utilizes the NVIDIA Jetson TX2i System-on-Module (SoM) in their S-A1760 system [24] which is designed for spacecraft and small satellites and has a flight heritage in LEO as part of a larger payload system [84]. The startup company Aethero[§] is also developing radiation-hardened edge computers for on-orbit data processing and autonomous decision-making in space. Its ECM-NxN utilizes an NVIDIA Jetson Orin processor, considered the best GPU edge processor available. While the ECM-NxN module has not directly flown in space, Aethero's use of this processor in their space computer indicates potential applications in space exploration. Finally, the Ingenuity Mars Helicopter used a GPU-equipped Qualcomm® Snapdragon™ 801 processor with great results, even though it operated in a "terrestrial" environment [85]. These are a few examples of global efforts to deploy edge GPU systems to space, and there is no doubt that more and more powerful space-grade GPUs will become commercially available to endure the harsh space environment for a prolonged period of time.

B. Number of Parameters

The number of parameters is closely related to the computational efficiency since more weights translate to more operations to be performed for both training and inference. However, the NN size is also tightly connected to the power consumption. For instance, the seminal work of Han et al. [86] in 2015 identified that, under the 45nm Complementary Metal-Oxide Semiconductor (CMOS) technology, 32-bit memory access to a Dynamic Random Access Memory (DRAM) consumes three orders of magnitude more power than accessing a Static Random Access Memory (SRAM) cache or performing a 32-bit floating point arithmetic. Considering miniature satellite systems such as CubeSats with limited power generation capability, it would be favorable to minimize the NN size so that the entire network can be

[§] `https://www.aethero.com/`

hosted on a local SRAM which is limited in memory size (typically up to 20MB for radiation hardened SRAM [83, Table 8.2, §8.3.4]).

The reduction in NN memory footprint can be achieved not just by minimizing the number of learnable parameters but also by quantizing the weights to lower-precision values such as IEEE half-precision floating point (FP16) [87] and 8-bit integer (INT8) [88]. More recent innovations in quantized training and inference include brain floating point (`bfloat16`) [89] for half-precision, 8-bit floating point (FP8) [90] and 4-bit floating point (FP4) [91], and even binary CNN whose weights and activations are constrained to $\{-1, +1\}$ [92]. Unfortunately, working with quantized weights often requires care to prevent over/underflowing of floating point values during gradient backpropagation and to minimize the loss of accuracy due to reduced bit resolution of individual weights. Moreover, many quantized floating point formats require specific hardware support to realize actual computational gain.

While power consumption of operating NNs on satellite avionics is an important engineering topic, this work does not immerse into the subject as realizing NN quantization is often an issue of software and hardware support on a specific system. Instead, it focuses on more general approaches to minimizing the total number of parameters while realizing the best OOD robustness on the SPEED+ dataset.

C. Batch-Agnostic Architecture

If any online training is to be done, it is desirable to have a batch-agnostic NN architecture, i.e., it does not contain any Batch Normalization (BN) layers [70]. The core of normalization (NORM) is to prevent internal covariate shifts of the feature maps and stabilize the training process. This is done by normalizing each feature map with a mean and a variance, for an i-th layer feature map,

$$\hat{\boldsymbol{X}}_i = \mathsf{NORM}(\boldsymbol{X}_i) \triangleq \frac{\boldsymbol{X}_i - \mathbb{E}[\boldsymbol{X}]}{\sqrt{\mathrm{Var}[\boldsymbol{X}_i]}} \tag{1}$$

where $\boldsymbol{X}_i$ is the feature map at the i-th layer. Unlike other methods, BN aims to compute the *batch-wise* feature map statistics across the entire training set. Since this is infeasible, in practice, the NORM layer statistics are computed across mini-batches, and the "global" statistics are updated using a running average as training progresses. Then, during inference, these running averages of the mean and variance are fixed and used for normalization.

The problem is that these approximations of the training data distribution in BN layers are simply incompatible with those of the test data that are drawn from a different distribution. Therefore, in this work, two methods are considered in search of a robust and batch-agnostic pose estimation NN architecture. One is to simply replace the BN layers in existing pre-trained architectures with batch-agnostic NORM layers such as Layer Normalization (LN) [93] and Group Normalization (GN) [94]. The other is to design or adopt a NN architecture that are inherently without BN layers such as ViT [22].

IV. Searching for Next Spacecraft Pose Network (SPNv3)

The goal of this section is two-fold. One is to introduce the overall pose estimation architecture given an image of the target at a visible distance in an RPOD scenario (Section IV.A). Then, different NN architectures (Section IV.B) and data augmentation techniques (Section IV.C) for training a pose estimation NN are explained which are combined in various configurations to study and explore which aspects of NN architecture and training algorithm contribute the most to the OOD robustness and reduced latency of a pose estimation NN (Section V). While SPN is the name of the first spacecraft pose estimation CNN introduced by Sharma and D'Amico [6], it is also used throughout this work as a legacy name referring to all NN architectures designed or adopted for spacecraft pose estimation by SLAB. When referring to the first original SPN, it is always accompanied by the reference to the relevant work [6].

A. Operational Scenario

In all vision-based RPOD scenarios, the servicer spacecraft begins tracking the non-cooperative target at kilometers of separations as shown in Fig. 3. For example, Angles-Only Navigation (AON) [95–98] obtains bearing angle measurements of the target from a Narrow Field-Of-View (NFOV) camera such as a star tracker which allows tracking of the target's relative orbital state via nonlinear filtering. AON continues until the inter-spacecraft distance becomes small enough such that the target appears resolved in the camera view, at which point the pose estimation algorithm kicks in.

Given the scenario, this work adopts the two-stage approach visualized in Fig. 3 as the common mechanism of the ML-based pose estimation algorithm which was first independently proposed by Park et al. [32] and Chen et al. [33] for

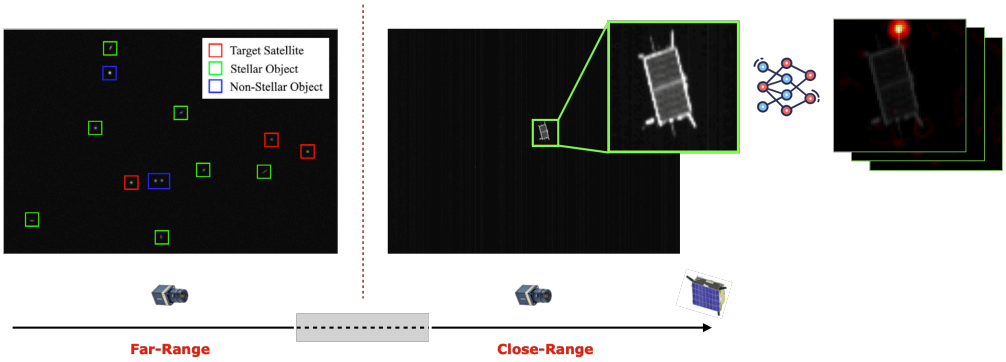


Fig. 3 Visualization of far-range AON [95] (*left*); close-range pose estimation on a cropped image (*middle*); and mechanism of feature/heatmap extraction by a NN (*right*). In general, the sensors are selected to ensure overlapping working distance to the target but it is not always possible based on the orbit and size of the target.

SPEC2019 [31]:

1) The first stage detects the target within the image frame using the state estimates from AON or a separate object detection NN. The latter may become especially useful in the case of the "lost-in-space" scenarios in which the navigation filter must reset all its state estimates. The detected bounding box or Region-of-Interest (RoI) from these sources is used to crop the image around the target spacecraft, and the cropped image is resized to lower-resolution inputs expected by the second-stage pose estimation NN (e.g., 224×224).
2) The second stage is the main pose estimation NN which takes the cropped low-resolution images and outputs K heatmaps centered at the 2D locations of known target surface keypoints. Knowing the 3D coordinates of these keypoints, the 2D keypoint locations extracted from the heatmap peaks can be used to solve PnP to compute the full 6D pose solution. Moreover, the advantage of dense predictions such as heatmaps is that their spread about the peak can be interpreted as uncertainties associated with each prediction [38]. These uncertainties can be leveraged to better perform state estimation via navigation filter [16] or even incorporated into the uncertainty-aware PnP algorithm [38, 99].

The heatmap-based pose estimation NN architecture is the most prominent approach, having won the first places in both SPEC2019 [31] and SPEC2021 [17]. Therefore, this work adopts the same heatmap detection architectures for NN. The remainder of this section explores various architectural designs and training algorithms to make the NN as robust as possible to OOD data.

B. Pose Estimation Architectures

This section explores three different heatmap detection architectures—EfficientDet [100], HRNet [101] and ViTPose [102]—visualized in Fig. 4. Note that this is not a comprehensive list of state-of-the-art pose estimation NN architectures. Nonetheless, they are chosen due to their simplicity and in order to keep the number of training sessions tractable while exploring vastly different architectural options.

1. EfficientDet

EfficientDet was originally proposed by Tan et al. [100] to enable computationally efficient yet high-performance object detection and semantic segmentation. EfficientDet builds upon the EfficientNet [20] backbone designed for the image classification task. At the heart of EfficientDet is the Bidirectional Feature Pyramid Network (BiFPN), a novel FPN architecture [103] which fuses and processes feature outputs from the backbone at multiple scales. By fusing features from low- to high-resolution and vice versa, BiFPN outperforms previous state-of-the-art FPN architectures on various benchmark object detection and semantic segmentation datasets. In the end, the feature outputs from BiFPN are processed by the task-specific prediction heads composed of convolution layers. For tasks that require multi-scale predictions (e.g., object detection), features from all resolutions are processed. In this work, the heatmap prediction head is attached only to the P2 level feature as shown in Fig. 4, where Pn indicates that the feature size is 2^{-n} of the input image resolution. Only the P2 features (i.e., heatmap has 1/4 of the input resolution) are used since the accuracy of heatmap prediction heavily depends on the heatmap pixel resolution.

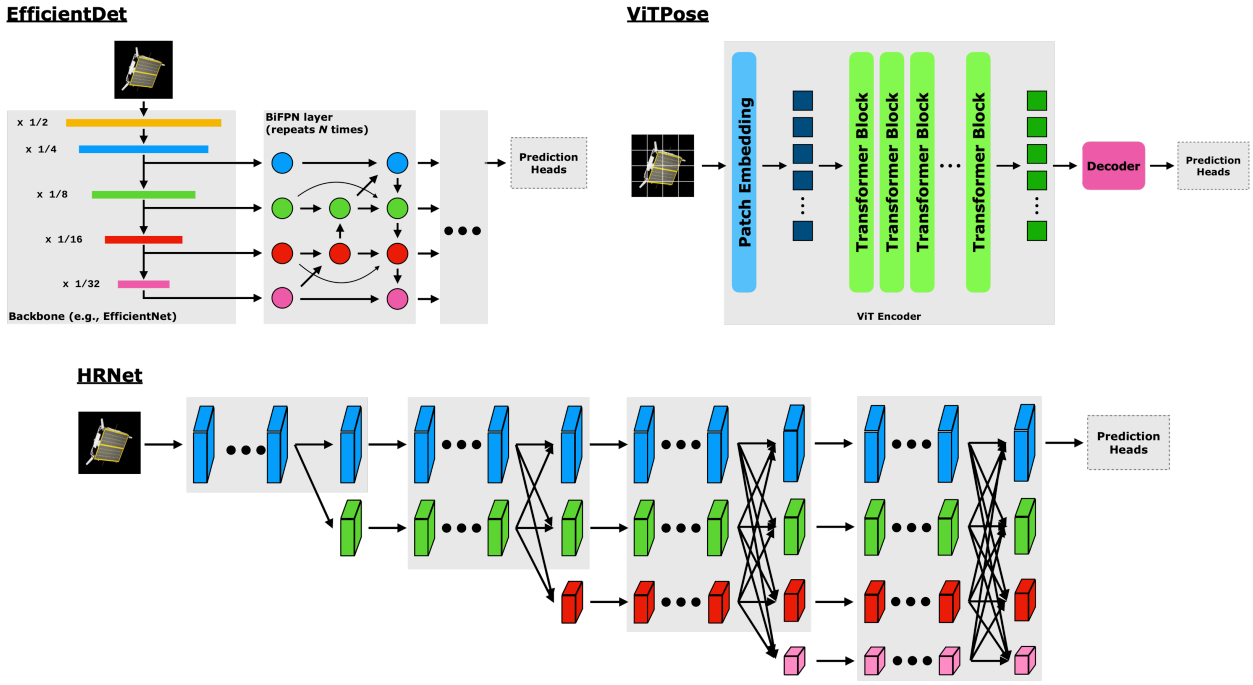


Fig. 4 Visualization of different pose estimation architectures: EfficientDet [100], HRNet [101] and ViTPose [102].

2. HRNet

High Resolution Network (HRNet) [101] is also designed for performing tasks that require multi-scale predictions. Recall that EfficientDet builds upon a CNN backbone that is typically designed for the image classification task, resulting in an architecture which continuously reduces the feature resolution all the way down to 1/32 of the original input. On the other hand, HRNet is designed to maintain high-resolution features throughout the entire forward propagation, outputting the multi-resolution features by the nature of its design. To facilitate learning at different scales, HRNet first processes the input image to 1/4 resolution, then gradually adds lower-resolution features in parallel, continuously fusing information across different scales as visualized in Fig. 4.

HRNet has won SPEC2019 [31, 33] and the `lightbox` category of SPEC2021 [17], so it is considered as a candidate benchmark in this paper. Similar to EfficientDet, only the highest resolution P2 level features are processed for heatmap outputs.

3. ViTPose

Compared to EfficientDet and HRNet which are CNN architectures based on convolution operations, ViTPose [102] leverages the ViT backbone [22] which divides the input image into $P \times P$ patches then processes them in parallel through transformer blocks. ViTPose groups the output token embeddings according to their original spatial position in the input image. The resulting features have the resolution of $H/P \times W/P$, where (H, W) is the input image size. For example, the output features will have 1/16 resolution of the original image for $P = 16$. ViTPose processes them through 2 transposed convolution layers in the prediction head which upsamples the input feature to 1/4 resolution heatmap outputs, same as EfficientDet and HRNet.

C. Extensive Data Augmentation

Data augmentation is vital to training NNs that generalize well beyond their training data distribution and to preventing overfitting. As basic sets of augmentation, SPN heavily leverages the Albumentations library [104] for various image processing operations. For each training sample, N augmentation operations are randomly chosen from the set of designated operations in a manner similar to RandAugment [105]. Specifically, while RandAugment utilizes a single hyperparameter to control the constant level of magnitude of these operations (e.g., standard deviation of white

Table 1 List of data augmentations and equivalent commands in Albumentations v1.3.1 [104].

Augmentations	Commands
Brightness & Contrast	`RandomBrightnessContrast`
Blur	`OneOf(GaussianBlur, MedianBlur, Defocus)`
Noise	`OneOf(GaussNoise, ISONoise, MultiplicativeNoise)`
Bit Reduction	`Posterize`
Sharpen	`Sharpen`
Pixel Inversion	`Solarize`
Spatter	`Spatter`
Histogram Equalization	`Equalize`
Gamma Correction	`RandomGamma`
Solar Flare[†]	`RandomSunFlare`

[†]Modified so that the solar flare only appears within the ground-truth bounding box.

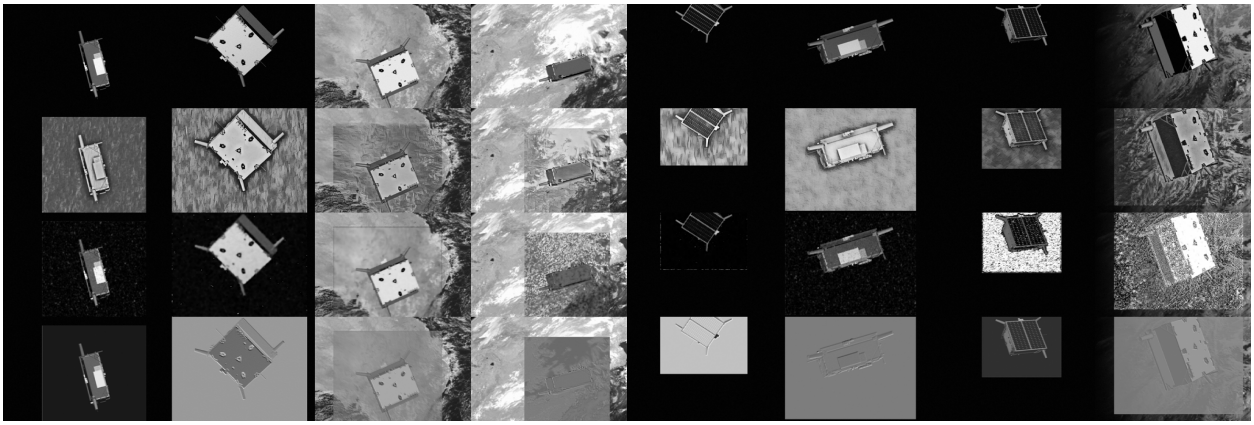


Fig. 5 Image augmentations. From top to bottom: SPEED+ `synthetic` images; StyleAugment [62]; DeepAugment [21]; RandConv [106].

Gaussian noise), this work simply picks each operation with a random magnitude within some pre-defined window. Then, these N operations are applied to the original image in sequence. The complete list of employed augmentations is provided in Table 1 along with their equivalent Albumentations commands. All commands are run with the default range of magnitudes.

In addition to the RandAugment-like augmentations which are composed of simple image processing operations such as Gaussian noise and contrast adjustment, three additional operations are explored that were designed specifically to address the domain gap issue.

1. Style Augmentation

Geirhos et al. [63] has empirically shown in 2019 that many state-of-the-art CNN architectures, such as ResNet [107] and VGG [108], exhibit strong bias toward object's texture when trained on ImageNet [34]. This finding is very counter-intuitive for humans who would, for example, classify an object or an animal based on its global or a set of local shapes, not based on its texture. To demonstrate such behavior, the authors have created Stylized-ImageNet (SIN) dataset by applying the AdaIN Neural Style Transfer (NST) pipeline [109] to each image in ImageNet with random styles from Kaggle's `Painter by Numbers` dataset[¶]. The result shows that when the same CNN architectures are trained on the SIN dataset, the networks not only exceed the performance of those trained on ImageNet, but they also show human-level robustness to previously unseen image distortions, such as noise, contrast change, and high- or low-pass

[¶]`https://www.kaggle.com/c/painter-by-numbers/`

filtering.

Following the same logic, Jackson et al. [62] devised style augmentation using the style transfer pipeline of Ghiasi et al. [110]. Specifically, in order to facilitate the style randomization process without resorting to individual "style images," they embedded different style images into the vector space using the intermediate feature output of an Inception CNN [111]. Then, during training, random style vectors are sampled and passed to the style transfer network to "stylize" the input image. This work adopts the style augmentation pipeline, using the same style embedding statistics as a source of randomization due to its ease of sampling different styles from the vector space embedding.

2. DeepAugment

DeepAugment [21] is another NN-based data augmentation technique that is done by perturbing the internal representations of a NN. These perturbations are randomly sampled from a set of hand-designed operations which include zeroing, negating, convolving, flipping, etc., of intermediate features and weights of a NN (e.g., an autoencoder) at random layers. For simplicity, this work follows the procedure identical to the one described in Hendrycks et al. [21] and uses the CAE architecture [112] to produce perturbed images.

3. RandConv

Finally, RandConv [106] applies a convolution operation with a $k \times k$ kernel composed of random weights. The motivation is similar to that of style augmentation – to distort the local texture of an input image. Following the original description, this work applies a convolution at the beginning with kernel size randomly sampled from $k = \{1, 3, 5, 7\}$, and the perturbed image is blended with the original image with random weights.

V. Trade-Off Analyses: OOD Robustness and Latency

This section performs the trade-off analyses between robustness across domain gap and training/inference latency for various configurations of pose estimation architectures (Section IV.B), data augmentation techniques (Section IV.C), and other aspects of NN training such as transfer learning and input resolution. The ultimate goal is to find the configuration that is both robust on SPEED+ HIL domain images and computationally efficient on a representative hardware, capable of operating above nominal GN&C update frequency (Section III.A) while also being batch-agnostic for potential online training (Section III.C).

A. Datasets

Following SPEC2021 [17], this section performs experiments using the SPEED+ dataset [15, 113] which consists of 59,960 `synthetic` images split in 80:20 ratio into the training and validation sets. It further includes two HIL image domains acquired from the TRON facility and a mockup model of the Tango spacecraft: 6,740 `lightbox` and 2,791 `sunlamp` images which are reserved solely for testing. In this section, all NN models are trained on the `synthetic` training set and evaluated on the `synthetic` validation set, `lightbox` and `sunlamp` test sets, and 25 labeled flight images from acquired in-space during RPO of the PRISMA mission (`prisma25`). Recall Fig. 1 for visualization of a few HIL domain images.

B. Implementations

In the original EfficientDet implementation [100], the input image resolution and the size of the BiFPN and prediction layers differ depending on the EfficientNet scaling parameter ϕ. In this section, for a fair comparison, the BiFPN layers are fixed to 4 convolution layers with 128 channels. The prediction heads for all pose estimation architectures consist of two convolution layers with 128 channels as well. Note that for most EfficientDet and HRNet implementations, the convolution operations in the prediction heads are depthwise separable convolutions [114] except ViTPose architectures that adopt full transposed 2D convolution operations with $\times 2$ upsampling and 128 channels. The NORM and activation layers for BiFPN and prediction heads all follow those used in the respective backbone NN. If the backbone consists of multiple types of NORM layers including BN, the following BiFPN and prediction heads adopt only BN. The 2D keypoint locations are extracted from each heatmap as the location with maximum pixel intensity then refined via Unbiased Data Processing (UDP) [115]. The keypoint coordinates are used to solve for the 6D pose solution via EPnP [81].

Given the ongoing interest in Artificial Intelligence (AI) for space applications, this work assumes that hardware

support for edge computing to run NN onboard satellites will increase in the upcoming years (see Section III.A.1). Therefore, this work adopts an NVIDIA Jetson Nano 4GB which contains a Quad-Core Arm® Cortex® A57 CPU and a 128-core NVIDIA Maxwell™ architecture GPU [23]. Nano is the most restrictive version of the commercially available Jetson family, which suits the limited processing power of onboard GPUs as evidenced in the NASA small spacecraft technology report [83]. Given its deprecated GPU architecture and CUDA language support, all NN training and inference on Jetson Nano are performed with full 32-bit Floating Point (FP32) precision using the C++ API of PyTorch 1.10.0.

1. Default Configuration

The default training configuration consists of the heatmap prediction head from the P2 level feature output and all data augmentations including style augmentation, DeepAugment and RandConv. The ground-truth heatmaps are Gaussian blobs with the standard deviation $\sigma = 1$ pixel around the 2D keypoint locations. The NN is trained with the pixel-wise Mean Squared Error (MSE) loss defined for the predicted heatmap $\hat{\boldsymbol{\mathcal{H}}}$ and the ground-truth $\boldsymbol{\mathcal{H}}$ as

$$\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{\mathcal{H}}}, \boldsymbol{\mathcal{H}}) = \frac{1}{K}\sum_{i=1}^{K} \|\hat{\boldsymbol{\mathcal{H}}}^{(i)} - \boldsymbol{\mathcal{H}}^{(i)}\|_F^2 \tag{2}$$

where K is the number of total heatmaps and $\|\cdot\|_F$ is a Frobenius norm of the heatmap matrix.

All training sessions excluding the experiments on the Jetson Nano are run on an NVIDIA RTX 4090 24GB GPU with `bfloat16` precision on PyTorch 2.1. All models are trained with the AdamW optimizer [116] for 30 epochs with the batch size 32, weight decay 1.0×10^{-4} and the initial learning rate 0.001 which linearly warms up during the first epoch then decays according to the cosine annealing schedule [117]. Note that the proposed training algorithm is not optimal for all different models considered in this study; nevertheless, adopting a common training method allows for a fair comparison of different metrics.

The default input image resolution after cropping is 224×224, identical to the ImageNet classification scenario. The input RoI is cropped using the ground-truth bounding box around the target following the procedure described in Park et al. [32]. Specifically, the ground-truth RoI is first corrected to a square-sized region with a size of $\max(h, w)$, where (h, w) are the height and width of the original RoI. This correction is implemented to ensure the aspect ratio of the target remains the same when resizing the cropped region. Then, during training, the new square RoI is enlarged and shifted by random factors in order to make the NN robust to object translation and misaligned RoI detection. During testing, the detected RoI is similarly converted into a square-sized region and enlarged by a fixed factor of 20% in order to ensure that the cropped region contains the entirety of the spacecraft.

Finally, all backbone NNs are obtained from the `timm` library available at HuggingFace [118], and they are all pre-trained on ImageNet-1K (IN-1K) [34] alone via supervised training unless noted otherwise. For simplicity, the architecture and backbone composition of a model is noted Architecture (*Backbone*) throughout this section.

2. Metrics

The performance metrics consist of the mean translation error (E_{t}), mean orientation error (E_{q}) and the mean pose error (E_{pose}) defined below for individual samples

$$e_{\text{t}} = \|\tilde{\boldsymbol{t}} - \boldsymbol{t}\|_2 \tag{3a}$$

$$\bar{e}_{\text{t}} = e_{\text{t}} / \|\boldsymbol{t}\| \tag{3b}$$

$$e_{\text{q}} = 2 \arccos | < \tilde{\boldsymbol{q}}, \boldsymbol{q} > | \tag{3c}$$

$$e_{\text{pose}} = e_{\text{q}} + \bar{e}_{\text{t}} \tag{3d}$$

Also note that for the SPEED+ HIL domains, the rotation and translation errors account for the calibration accuracy of the TRON facility. Specifically, for individual samples, the HIL errors are defined as [17]

$$\bar{e}_{\text{t}}^{*}(\tilde{\boldsymbol{t}}, \boldsymbol{t}) = \begin{cases} 0 & \text{if } \bar{e}_{\text{t}}(\tilde{\boldsymbol{t}}, \boldsymbol{t}) < \theta_t \\ \bar{e}_{\text{t}}(\tilde{\boldsymbol{t}}, \boldsymbol{t}) & \text{otherwise} \end{cases}, \quad e_{\text{q}}^{*}(\tilde{\boldsymbol{q}}, \boldsymbol{q}) = \begin{cases} 0 & \text{if } e_{\text{q}}(\tilde{\boldsymbol{q}}, \boldsymbol{q}) < \theta_q \\ e_{\text{q}}(\tilde{\boldsymbol{q}}, \boldsymbol{q}) & \text{otherwise} \end{cases} \tag{4}$$

The rotation threshold (θ_q) and the normalized translation threshold (θ_t) are derived from the calibration accuracy of TRON and are set to $\theta_q = 0.169^\circ$ and $\theta_t = 2.173 \times 10^{-3}$ [14]. Then, the HIL pose error for a sample is given as

$$e_{\text{pose}}^{*} = e_{\text{q}}^{*}(\tilde{\boldsymbol{q}}, \boldsymbol{q}) + \bar{e}_{\text{t}}^{*}(\tilde{\boldsymbol{t}}, \boldsymbol{t}) \tag{5}$$

Table 2 Ablation study on different pose estimation architectures and backbones. Latency and peak memory during training are measured on an NVIDIA RTX 4090 24GB GPU with batch size 1 and FP32 precision. Mean(std. dev.) across all samples for each domain are reported. The boldface denotes the best performance within each group.

Backbone	Norm.	Num. Param.	Mem. [MB]	Time [ms]		E_q / E_q^* [°]			
				Train	Test	synthetic	lightbox	sunlamp	prisma25
• MobileNetV3-L (×0.75) [119]	BN	2.3M	149	21.3	6.1	1.00(3.00)	10.48(29.97)	19.43(40.25)	3.66(7.33)
• HRFormer-T [120]	BN/LN	2.6M	264	58.5	20.4	1.22(4.30)	9.40(27.30)	**14.17(32.64)**	17.39(46.83)
• HRNet-W18-S [101]	BN	2.9M	**65**	**13.2**	**4.8**	1.55(5.77)	14.68(33.54)	26.13(43.51)	22.68(38.51)
• MobileNetV3-L (×1.0) [119]	BN	3.5M	168	20.4	6.1	0.91(2.82)	8.19(25.45)	14.29(32.96)	**2.46(1.76)**
• EfficientFormerV2-S0 [121]	BN/LN	3.7M	197	25.8	7.9	1.09(4.07)	10.76(30.10)	17.08(36.77)	2.91(3.90)
• EfficientNet-B0 [20]	BN	4.1M	212	23.1	6.6	**0.80(1.88)**	**7.75(25.41)**	14.70(35.11)	8.61(30.96)
• MobileViTv2-100 [122]	BN/LN	5.0M	262	21.8	6.9	0.81(2.28)	9.79(29.06)	17.89(38.72)	6.55(16.98)
• EfficientNetV2-B0 [123]	BN	6.1M	208	25.9	7.3	0.84(2.75)	7.25(24.30)	12.55(32.43)	2.81(4.38)
• ViT-T/16 [124]	LN	6.2M	**129**	**13.2**	**3.8**	1.01(2.60)	7.88(23.48)	14.49(32.25)	16.56(46.06)
• EfficientFormerV2-S1 [121]	BN/LN	6.2M	252	31.5	8.7	1.02(3.86)	9.50(27.77)	16.46(36.21)	17.21(41.00)
• EfficientNet-B1 [20]	BN	6.6M	274	28.4	7.7	**0.71(1.06)**	**6.07(21.58)**	**9.34(26.40)**	**2.50(2.79)**
• EfficientNetV2-B1 [123]	BN	7.1M	**236**	29.2	7.7	0.78(2.93)	5.87(21.20)	9.44(27.60)	3.53(6.99)
• MobileViTv2-125 [122]	BN/LN	7.5M	324	**21.4**	**6.8**	0.75(2.82)	7.92(25.56)	10.87(29.03)	3.48(6.66)
• EfficientNet-B2 [20]	BN	7.7M	292	27.6	7.9	**0.70(2.02)**	**5.21(19.44)**	**8.05(23.99)**	**1.98(1.52)**
• HRFormer-S [120]	BN/LN	7.9M	500	69.2	23.7	0.94(3.07)	6.77(22.17)	10.60(27.68)	37.88(62.67)
• EfficientNetV2-B2 [123]	BN	8.9M	**263**	30.3	8.1	0.73(2.18)	5.63(20.59)	8.48(25.36)	1.92(1.39)
• MobileViTv2-150 [122]	BN/LN	10.5M	393	**21.9**	**6.8**	0.71(2.27)	7.85(25.46)	10.19(27.72)	**1.88(1.09)**
• EfficientNet-B3 [20]	BN	10.6M	369	27.9	8.5	**0.66(2.16)**	**4.67(18.19)**	**7.08(21.24)**	2.07(1.78)
• HRNet-W18 [101]	BN	11.0M	**238**	42.4	13.7	0.88(2.79)	6.26(20.92)	10.52(28.54)	16.10(47.52)
• EfficientFormerV2-S2 [121]	BN/LN	12.6M	374	34.3	11.4	0.93(4.13)	6.94(23.33)	11.94(31.53)	2.13(1.63)
• EfficientNetV2-B3 [123]	BN	12.9M	335	**29.1**	**8.9**	0.69(2.18)	**5.11(19.08)**	**7.41(23.00)**	3.48(7.02)
• EfficientNet-B4 [20]	BN	17.2M	499	31.4	9.9	**0.69(1.47)**	5.67(20.27)	7.79(23.76)	**1.94(1.12)**
• EfficientNetV2-S [123]	BN	20.3M	468	32.9	10.3	0.62(1.72)	4.24(16.63)	6.10(18.91)	4.55(13.94)
• NFNet-L0 [125]	-	22.3M	497	24.3	8.3	0.64(2.06)	5.59(20.19)	7.91(23.41)	6.25(16.25)
• ViT-S/16 [124]	LN	22.7M	**377**	**11.8**	**3.3**	0.75(1.65)	5.12(17.75)	9.29(24.94)	2.32(1.36)
• HRNet-W30 [101]	BN	27.4M	493	44.8	13.6	0.73(2.09)	5.16(19.22)	8.19(24.52)	6.78(23.84)
• EfficientNet-B5 [20]	BN	27.8M	707	39.3	11.1	**0.58(1.37)**	**3.79(15.59)**	**5.26(16.63)**	**2.12(1.36)**
• ResMLP-S24 [126]	Affine	30.7M	**604**	16.4	4.2	1.08(3.21)	7.33(21.62)	18.65(37.93)	2.18(1.17)
• NFNet-L1 [125]	-	37.6M	799	33.5	12.7	0.54(1.82)	4.68(18.76)	6.53(21.80)	3.64(8.76)
• ViT-M/16 [124]	LN	39.6M	647	**11.5**	**3.4**	0.70(2.11)	4.32(15.77)	8.11(22.57)	2.18(1.07)
• EfficientNet-B6 [20]	BN	39.9M	944	39.9	11.9	**0.53(1.00)**	**3.65(15.14)**	**5.19(18.03)**	**1.99(1.06)**

• EfficientDet • HRNet • ViTPose

Note that in the overall pose estimation scenario (Section IV.A), the information on the relative position of the target is mostly included in the RoI detected from either the navigation filter or a separate object detection NN. Since the ground-truth RoIs are used for image pre-processing in this section, mean translation errors are expected to be very small throughout the experiments. Therefore, only the mean orientation errors (E_q / E_q^*) are reported for the majority of this section for brevity.

C. Ablation Studies

1. Architecture & Backbone

This section begins with the study on OOD robustness and latency of various pose estimation architectures and NN backbones. First, Table 2 tabulates the latency and pose accuracy of different pose estimation architectures with various

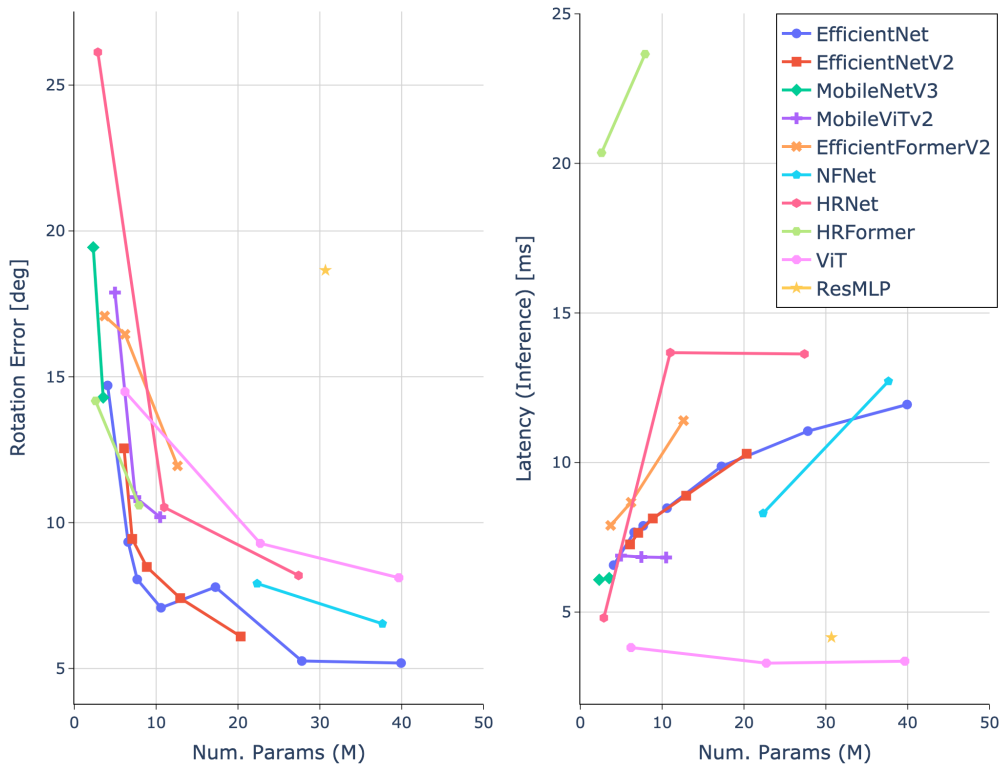


Fig. 6 Visualization of performance of different backbones tabulated in Table 2. Rotation error (E_q^*) is for the `sunlamp` domain.

backbone NNs of different sizes. These backbones cover a wide variety of NN architectures, including conventional CNNs with BN layers (e.g., EfficientNet [20], EfficientNetV2 [123], MobileNetV3 [119], HRNet [101]); NFNet [125] which has no separate NORM layers but performs weight standardization of the convolution weights [127]; Vision Transformers (ViT) [22] with self-attention modules [128]; architectures purely based on Multi-Layer Perceptrons (MLP) (e.g., ResMLP [126]); and hybrid architectures which both include convolution operations and self-attention layers for image patches as hinted by the use of both BN/LN layers (e.g., HRFormer [120], EfficientFormerV2 [121], MobileViTv2 [122].

In order to facilitate readability, Table 2 sorts different architectures according to their total number of parameters in ascending order and grouped into similar sizes. Overall, larger NNs tend to result in better OOD robustness as measured by the performance on the SPEED+ `lightbox` and `sunlamp` domains. Then, by investigating the models within each size group, it becomes immediately clear that the EfficientDet (*EfficientNet*) and EfficientDet (*EfficientNetV2*) consistently outperform all other models across different size groups in terms of robustness. They are closely followed by EfficientDet (*NFNet*) and ViTPose (*ViT*).

Figure 6 condenses the information in Table 2 by plotting the mean orientation error on the `sunlamp` domain and the inference latency against the number of parameters. Different families of backbones are plotted with different colors and marker shapes. First, the E_q^* plot in Fig. 6 (*left*) shows that EfficientDet (*EfficientNet*) and EfficientDet (*EfficientNetV2*) consistently outperform all other models in terms of OOD robustness. While EfficientDet (*NFNet*) models are closely behind them, there is no smaller variant of NFNet pre-trained on IN-1K that is publicly available. On the other hand, the latency plot in Fig. 6 (*right*) shows that ViTPose (*ViT*) and ViTPose (*ResMLP*) are clear winners. Both architectures consist solely of MLP layers, enabling a faster throughput compared to those with convolution operations. Unfortunately, ResMLP cannot perform well across domain gaps, which is no surprise since it is a pure MLP-based NN with no inductive bias of convolution operations or global spatial knowledge via self-attention layers. However, ViTPose (*ViT*) models closely follow EfficientDet (*EfficientNet*) in terms of OOD robustness.

Based on the OOD robustness and latency trade-off in Fig. 6, only EfficientDet (*EfficientNet*), EfficientDet (*EfficientNetV2*) and ViTPose (*ViT*) are considered for further analyses.

Table 3 Peak training memory and latency of select models from Table 2 are measured on an onboard GPU of NVIDIA Jetson Nano 4GB with PyTorch C++ API for batch size 1 and FP32 precision. The boldface denotes the best performance within each group.

Backbone	Norm.	Num. Param.	Mem. [MB]	Latency [ms]		E_q^* [°]	
				Train	Test	`lightbox`	`sunlamp`
• EfficientNet-B0	BN	4.1M	**212**	**446**	**57**	7.75(25.41)	14.70(35.11)
• EfficientNet-B1	BN	6.6M	274	595	72	6.07(21.58)	9.34(26.40)
• EfficientNet-B2	BN	7.7M	292	622	71	5.21(19.44)	8.05(23.99)
• EfficientNet-B3	BN	10.6M	369	758	75	4.67(18.19)	7.08(21.24)
• EfficientNet-B4	BN	17.2M	499	985	87	5.67(20.27)	7.79(23.76)
• EfficientNet-B5	BN	27.8M	707	1317	98	3.79(15.59)	5.26(16.63)
• EfficientNet-B6	BN	39.9M	944	1660	110	**3.65(15.14)**	**5.19(18.03)**
• EfficientNetV2-B0	BN	6.1M	**208**	**473**	**65**	7.25(24.30)	12.55(32.43)
• EfficientNetV2-B1	BN	7.1M	236	553	72	5.87(21.20)	9.44(27.60)
• EfficientNetV2-B2	BN	8.9M	263	598	74	5.63(20.59)	8.48(25.36)
• EfficientNetV2-B3	BN	12.9M	335	746	82	5.11(19.08)	7.41(23.00)
• EfficientNetV2-S	BN	20.3M	468	996	96	**4.24(16.63)**	**6.10(18.91)**
• ViT-T/16	LN	6.2M	**129**	**221**	25	7.88(23.48)	14.49(32.25)
• ViT-S/16	LN	22.7M	377	349	**22**	5.12(17.75)	9.29(24.94)
• ViT-M/16	LN	39.6M	647	546	23	**4.32(15.77)**	**8.11(22.57)**

• EfficientDet • ViTPose

2. Latency on NVIDIA Jetson Nano

The models chosen from the previous section are implemented on an NVIDIA Jetson Nano 4GB and tested for training and inference latency on its GPU. Specifically, all training and inference are performed with full FP32 precision for single images. The FP32 training and inference are due to the lack of support for FP16 or INT8 quantization on the Nano's GPU, whereas a single image training is due to the fact that it is undesirable for a satellite to wait and collect a batch of images of the target spacecraft during in-space RPOD, both logistically and computationally.

The latency measured on the Jetson Nano reported in Table 3 reaffirms the observation made in Table 2 and Fig. 6. For the same number of parameters, EfficientDet models dominate on the OOD robustness front compared to the ViTPose models. However, ViTPose (*ViT*) models are consistently faster for inference regardless of the size. In fact, the largest ViT backbone considered in this study—ViT-M/16 with nearly 40M parameters—runs nearly 2.5 times faster than the smallest variant of the EfficientNet backbone that is 1/10 in size, and its training latency is on par with that of EfficientNet-B1 which only has 6.6M parameters. It also requires at most about 650 MB of GPU virtual memory for each training session with a single image input. Considering the 4GB of GPU memory available on the Jetson Nano and assuming that most of the GN&C-related operations run on CPUs, 650 MB or less is completely within the range of operation excluding the possibility of running other image-processing algorithms on the GPU.

Recall from the earlier discussion that the training is expected to run only occasionally whenever there has been a substantial change in the scenery and the target's pose (Section III.A). Therefore, even the EfficientNet-B6 backbone could perform onboard online training asynchronously on a GPU given that there is enough power generated to support such a computationally heavy operation. In terms of the inference speed, all models considered in Table 3 can effectively run in real-time for 0.5 Hz GN&C updates of the CPOD reference mission [82] assuming that the pre- and post-processing of the NN inputs and outputs do not become the computational bottleneck.

3. Replacing BN Layers

Judging from the comparison on the NVIDIA Jetson Nano reported in Table 3, it appears that EfficientDet models are more OOD robust for the same number of parameters, but the larger ViTPose models are still more computationally efficient than small EfficientDet models, overcoming its comparative weakness in OOD robustness with increased size. There is one more reason that ViTPose is favored over EfficientDet, and that is due to the requirement that the onboard NNs must be batch-agnostic for online training (Section III.C). As evidenced in Table 2, most CNN backbones designed for ImageNet classification adopt BN layers, and even hybrid architectures always pair BN layers to each convolution

Table 4 EfficientDet architecture with the EfficientNet-B0 backbone evaluated with different NORM layers. The architecture's and the backbone's NORM layers during IN-1K pretraining are both noted. Mean(std. dev.) of average performances over 3 training sessions with different random seeds are reported. Latency is measured on an NVIDIA Jetson Nano 4G. The boldface denotes the best performance within each group.

NORM (Pre-train)	NORM (Current)	Training Epochs	Time [ms]		E_q / E_q^* [°]			
			Train	Test	synthetic	lightbox	sunlamp	prisma25
BN	BN	30	**446**	**57**	**0.81**(0.00)	7.34(0.27)	13.90(0.37)	7.54(3.96)
BN	GN	30	471	68	1.10(0.01)	12.44(0.39)	23.18(2.01)	12.24(1.32)
BN	GN	60	471	68	0.88(0.02)	9.76(0.34)	18.16(1.28)	11.47(4.33)
GN	GN	30	471	68	0.84(0.00)	**6.67**(0.49)	**12.20**(0.78)	**3.99**(2.50)

operation. Then, is it possible to replace those BN layers with other batch-agnostic layers (e.g., LN, GN) when the overwhelming majority of publicly available CNN backbones pre-trained on ImageNet come with BN layers?

Table 4 reports the orientation error of the EfficientDet (*EfficientNet-B0*) model (4.1M parameters) with different NORM layers during the IN-1K pre-training and the main training on SPEED+ `synthetic` images. The baseline is using BN layers for both sessions, which reports $E_q = 6.63°$ on `lightbox` and $E_q = 10.77°$ on `sunlamp` across 3 training sessions with different random seeds. When the BN layers of a pre-trained backbone is replaced with GN layers with group size 8, fully inheriting the trained affine parameters of the BN layers, the mean orientation error degrades by more than a factor of two. Training the network longer for 60 epochs instead of 30 does improve the overall performance, but it still does not reach the baseline level for all image domains. This was the conclusion of SPNv2 which also reported degradation in performance after replacing the BN with GN layers [18].

However, when the backbone is pre-trained on IN-1K with GN layers from the very beginning, the OOD performance shows consistent improvement on not only `lightbox` and `sunlamp` but also `prisma25`. The result indicates that it is possible to replace the BN layers in existing CNN architectures with GN or other batch-agnostic layers, but in order to preserve the OOD robustness, the backbone must be pre-trained on IN-1K or other large-scale datasets with those batch-agnostic layers from the very beginning. The problem is that ImageNet training is extremely expensive, both computationally and in terms of training time. For this study, EfficientNet-B0 was trained on 1M images of IN-1K loosely following the training Recipe B of Wightman et al. [129, Appendix F] for 350 epochs. Training this small NN on two NVIDIA RTX 4090 24GB GPUs took well over 100 hours. To train larger models would require even more computing power. If such heavy offline training can be afforded, one can use the EfficientNet-B0 backbone with GN layers that show OOD robustness comparable to the ViT-M/16 backbone with 40M parameters at the expense of increased inference time but with much less power consumption. However, if ImageNet pre-training cannot be afforded, it would be preferred to adopt ViT which is available for various ImageNet pre-trained models without BN layers.

4. Transfer Learning

The results so far indicate that ViTPose models are superior in terms of computational efficiency and batch-agnostic by nature, satisfying both Requirements #1 and #3 of Section III. However, they still fall short in OOD robustness when compared to EfficientDet models of comparable sizes. Therefore, it would be favorable to improve the OOD robustness of ViTPose models without substantial sacrifice of computational advantages. One potential remedy is to improve the transfer learning from the ImageNet pre-training. The ViT backbones of the ViTPose architecture so far are all pre-trained on IN-1K according to the DeiT III recipe [124]. This is an improved training method compared to DeiT [130] which is, in turn, an improved training method compared to the vanilla ViT [22]. In this section, the effect of ImageNet pre-training for ViT backbones is analyzed in detail by comparing the two different pre-training methods (DeiT vs. DeiT III) and datasets in Table 5. First, using random initialization without any pre-training on a large-scale dataset does not train well on SPEED+. For ViT-T/16 and ViT-S/16, using DeiT III leads to improved performance over DeiT when using only IN-1K for supervised pre-training. When the backbone is trained instead on a much larger ImageNet-22K (IN-22K) [34] which contains nearly 14M images and 22K class labels then fine-tuned on IN-1K, there is a marginal improvement in OOD robustness for both ViT-S/16 and ViT-M/16 models. The results overall indicate that better generalization of the backbone NN on a large-scale image classification dataset leads to a noticeable improvement in the downstream pose estimation task across the sim2real gap, though it seems the size of the pre-training dataset has

Table 5 Performance of the ViTPose pose estimation architecture with three backbones (ViT-T/16 & ViT-S/16 & ViT-M/16) with different pretraining datasets (None & ImageNet-1K (IN-1K) & ImageNet-22K (IN-22K)) and methods (DeiT [130] & DeiT III [124]). Mean(std. dev.) of average performances over 3 training sessions with different random seeds are reported. The boldface denotes the best performance within each group.

Backbone (Pre-Train Method)	Pre-Train Dataset	Num. Param.	IN-1K top-1 (%)	E_q / E_q^* [°]			
				`synthetic`	`lightbox`	`sunlamp`	`prisma25`
• ViT-T/16	-	6.2M	-	16.12(1.14)	59.72(0.82)	83.34(0.71)	66.46(7.75)
• ViT-T/16 (DeiT)	IN-1K	6.2M	72.17	1.13(0.02)	8.92(0.38)	18.90(0.53)	17.67(3.21)
• ViT-T/16 (DeiT III)†	IN-1K	6.2M	75.12	**1.02(0.01)**	**8.27(0.09)**	**14.23(0.47)**	**10.51(4.80)**
• EfficientNetV2-B0	IN-1K	6.2M	78.37	0.84	7.25	12.55	2.81
• ViT-S/16	-	22.7M	-	6.40(1.35)	41.63(3.86)	65.74(2.57)	39.39(7.96)
• ViT-S/16 (DeiT)	IN-1K	22.7M	79.86	0.87(0.02)	6.11(0.15)	13.20(0.89)	8.79(0.36)
• ViT-S/16 (DeiT III)	IN-1K	22.7M	81.36	0.75(0.00)	4.98(0.14)	8.81(0.29)	4.26(3.08)
• ViT-S/16 (DeiT III)	IN-22K	22.7M	83.07	**0.74(0.02)**	**4.84(0.01)**	**8.29(0.16)**	**2.24(0.08)**
• EfficientNetV2-S	IN-1K	20.3M	83.91	0.62	4.24	6.10	4.55
• ViT-M/16	-	39.6M	-	36.19(1.85)	79.61(1.13)	95.43(0.98)	82.80(9.73)
• ViT-M/16 (DeiT III)	IN-1K	39.6M	83.10	0.71(0.01)	4.46(0.20)	8.36(0.15)	**2.19(0.06)**
• ViT-M/16 (DeiT III)	IN-22K	39.6M	84.56	**0.67(0.00)**	**4.20(0.11)**	**7.73(0.11)**	4.18(2.01)
• EfficientNet-B6	IN-1K	39.9M	84.12	0.53	3.65	5.19	1.99

• EfficientDet • ViTPose † Trained by the authors

Table 6 Performance of ViTPose (*ViT-S/16*) pre-trained via DeiT III on IN-22K. The input image resolution and the standard deviation (σ) of the ground-truth heatmaps are varied. Mean(std. dev.) of average performances over 3 training sessions with different random seeds are reported. Latency is measured on an NVIDIA Jetson Nano 4GB. The boldface denotes the best performance.

Input Res.	σ [pix]	Mem. [MB]	Latency [ms]		E_q / E_q^* [°]			
			Train	Test	`synthetic`	`lightbox`	`sunlamp`	`prisma25`
224 × 224	1	**377**	**349**	**22**	0.74(0.02)	4.84(0.01)	8.29(0.16)	2.24(0.08)
288 × 288	1	401	566	23	0.58(0.01)	3.88(0.12)	6.70(0.22)	5.79(2.82)
384 × 384	1	498	1046	27	0.47(0.01)	3.30(0.12)	5.61(0.40)	2.22(0.17)
384 × 384	2	498	1046	27	0.51(0.00)	2.99(0.11)	5.11(0.08)	1.86(0.01)
448 × 448	2	585	1500	34	**0.47(0.01)**	**3.16(0.13)**	**4.53(0.16)**	**1.78(0.11)**
EfficientDet (*EfficientNet-B6*)								
224 × 224	1	944	1660	110	0.53	3.65	5.19	1.99

little effect on the OOD robustness beyond the scale of IN-1K.

5. Input Resolution

Another potential remedy to improve the OOD robustness of ViTPose models is to increase the input image resolution. The effect of input image resolution of ViTPose on the OOD robustness and latency on the Jetson Nano is reported in Table 6. Unsurprisingly, increasing the resolution of the input image cropped around the target leads to improved performance across the board but at the cost of increased peak training memory and latency. Specifically, doubling the input resolution from 224 × 224 to 448 × 448 nearly quadruples the training time for each input. This is expected, since doubling the input resolution but maintaining the same patch size (16 × 16) leads to double the length of patch tokens fed into the transformer blocks, and the memory footprint of the scaled dot-product operation of self-attention modules grows quadratically to the token length. Even though the training latency is tantamount to that of EfficientDet (*EfficientNet-B6*) operating on 224 × 224 inputs with nearly 40M parameters, the inference latency is still

Table 7 Performance of the ViTPose-S/16 different different data augmentation configurations. Mean(std. dev.) of average performances over 3 training sessions with different random seeds are reported. The boldface denotes the best performance.

SA	DA	RC	E_q / E_q^* [°]			
			`synthetic`	`lightbox`	`sunlamp`	`prisma25`
-	-	-	**0.64(0.01)**	6.49(0.22)	13.18(0.62)	36.45(5.28)
✓	-	-	0.69(0.01)	6.16(0.08)	9.80(0.38)	2.40(0.21)
✓	✓	-	0.71(0.01)	5.66(0.35)	10.30(0.75)	**2.20(0.16)**
-	✓	✓	0.73(0.01)	4.84(0.18)	9.22(0.08)	3.58(0.88)
✓	✓	✓	0.72(0.01)	**4.68(0.27)**	**8.50(0.15)**	2.22(0.09)

about a third and the training memory nearly half of EfficientDet (*EfficientNet-B6*) while performing better on all image domains. The result suggests that it could be beneficial to use ViT-based models with higher input resolutions as long as the training time does not become the bottleneck.

6. Data Augmentation

The experiments so far used all three data augmentation techniques—Style Augmentation, DeepAugment and RandConv. In order to verify the individual contribution of these augmentations, Table 7 reports the mean orientation errors for different data augmentation configurations. The default baseline only consists of 5 random augmentations from the Albumentations library implemented in the style of RandAugment [105] (Section IV.C).

Note that adding just the style augmentation to the training significantly improves the performance on `sunlamp`, and incrementally adding DeepAugment and RandConv leads to consistent performance improvement on `lightbox`. While it does not bear much statistical significance, an interesting observation is that the performances on 25 flight images of `prisma25` dramatically improve and stabilize (as observed from the standard deviation) once the style augmentation and other techniques are implemented during the training.

D. Summary

The previous section investigates different aspects of designing and training a robust pose estimation NN. Under the same training conditions with identical input image resolutions, EfficientDet (*EfficientNet*) consistently outperforms all other models in terms of the OOD robustness as measured on the SPEED+ HIL domains. On the other hand, ViTPose (*ViT*) dominates the training and inference latency for different model sizes but with inferior OOD robustness. However, various ablation studies reveal that the ViTPose models can be made as robust as EfficientDet with improved pre-training of the ViT backbone, extensive data augmentation and increased input image resolution. Specifically, even with twice higher image resolution, the inference of ViTPose (*ViT-S/16*) is still faster than that of EfficientDet (*EfficientNet-B0*) that is 1/5 in size. Doubling the input resolution does lead to the increase of training latency by a factor of 4, but as stated in Requirement #1 (Section III.A), the increased training time is less likely to become as mission-critical as the inference time on a satellite avionic with an onboard GPU. Moreover, ViTPose satisfies Requirement #3 by design (Section III.C), completely lacking BN layers that could interfere with accurate online training on single images.

Moving forward, the ViTPose models trained with the data augmentation proposed in Section IV.C are referred to as SPNv3 for convenience. The default SPNv3 configuration is the ViT backbone pre-trained via DeiT III [124] on IN-22K then fine-tuned on IN-1K, patch size $P = 16$, and input resolution 448×448.

1. Comparison on SPEED+

Table 8 compares the final performances of SPNv3-S, a mid-size variant that takes the ViT-S/16 backbone (22.7M parameters), against the top entries of SPEC2021 [17] and other state-of-the-art models introduced during the post-mortem competition following SPEC2021 and others. It can be seen that, with a higher input resolution at 448×448, SPNv3-S already outperforms the top entries of the `lightbox` category. Moreover, ensembling the output heatmap predictions from three models trained with different random seeds further improves the overall performance, achieving $E_q = 2.69°$ on `lightbox` and $E_q = 3.89°$ on `sunlamp`. Some of the pose predictions by the ensemble are visualized in Fig. 7 via reprojection of the Tango wireframe model. Note that the winning teams for respective HIL

Table 8 Performance of SPNv3 models and state-of-the-art. Mean(std. dev.) denotes the average performance over 3 training sessions with different random seeds. The boldface denotes the best performance in each group.

Team/Model	Num. Params.	lightbox			sunlamp		
		$\bar{E}_t^*$ [-]	E_q^* [°]	E_{pose}^* [-]	$\bar{E}_t^*$ [-]	E_q^* [°]	E_{pose}^* [-]
SPEC2021 Top-3 [17]							
TangoUnchained [17]*	-	**0.018**	**3.187**	**0.073**	0.015	4.299	0.090
VPU [79]*	-	0.022	4.577	0.101	0.012	2.827	0.061
lava1302 [78]*	-	0.048	6.664	0.165	**0.011**	**2.728**	**0.059**
Post-mortem & misc.							
SPNv2 [18]	52.5M	0.025	5.577	0.122	0.026	9.788	0.197
Chen et al. [131]*	-	0.018	2.865	0.068	0.014	2.750	0.062
FA-VAE [132]	41.6M[1]	0.027	4.939	0.114	0.028	5.185	0.118
EagerNet [80]	88.6M[2]	**0.009**	**1.748**	**0.039**	**0.013**	**2.664**	**0.059**
Proposed							
SPNv3-S	22.7M	$0.016_{(0.001)}$	$3.044_{(0.234)}$	$0.069_{(0.004)}$	$0.020_{(0.000)}$	$4.371_{(0.314)}$	$0.097_{(0.006)}$
SPNv3-S†	22.7M	0.017	2.689	0.064	0.020	3.883	0.088
SPNv3-M	39.6M	$0.014_{(0.000)}$	$2.595_{(0.123)}$	$0.060_{(0.002)}$	$0.018_{(0.001)}$	$4.158_{(0.254)}$	$0.090_{(0.005)}$
SPNv3-M†	39.6M	0.014	2.396	0.056	0.017	3.765	0.082
SPNv3-B	86.3M	$0.013_{(0.001)}$	$2.183_{(0.129)}$	$0.051_{(0.003)}$	$0.016_{(0.001)}$	$3.612_{(0.164)}$	$0.079_{(0.003)}$
SPNv3-B†	86.3M	**0.012**	**2.033**	**0.047**	**0.015**	**3.400**	**0.074**

* Included HIL domain images into training

† From ensemble of heatmap predictions from the models of three training sessions

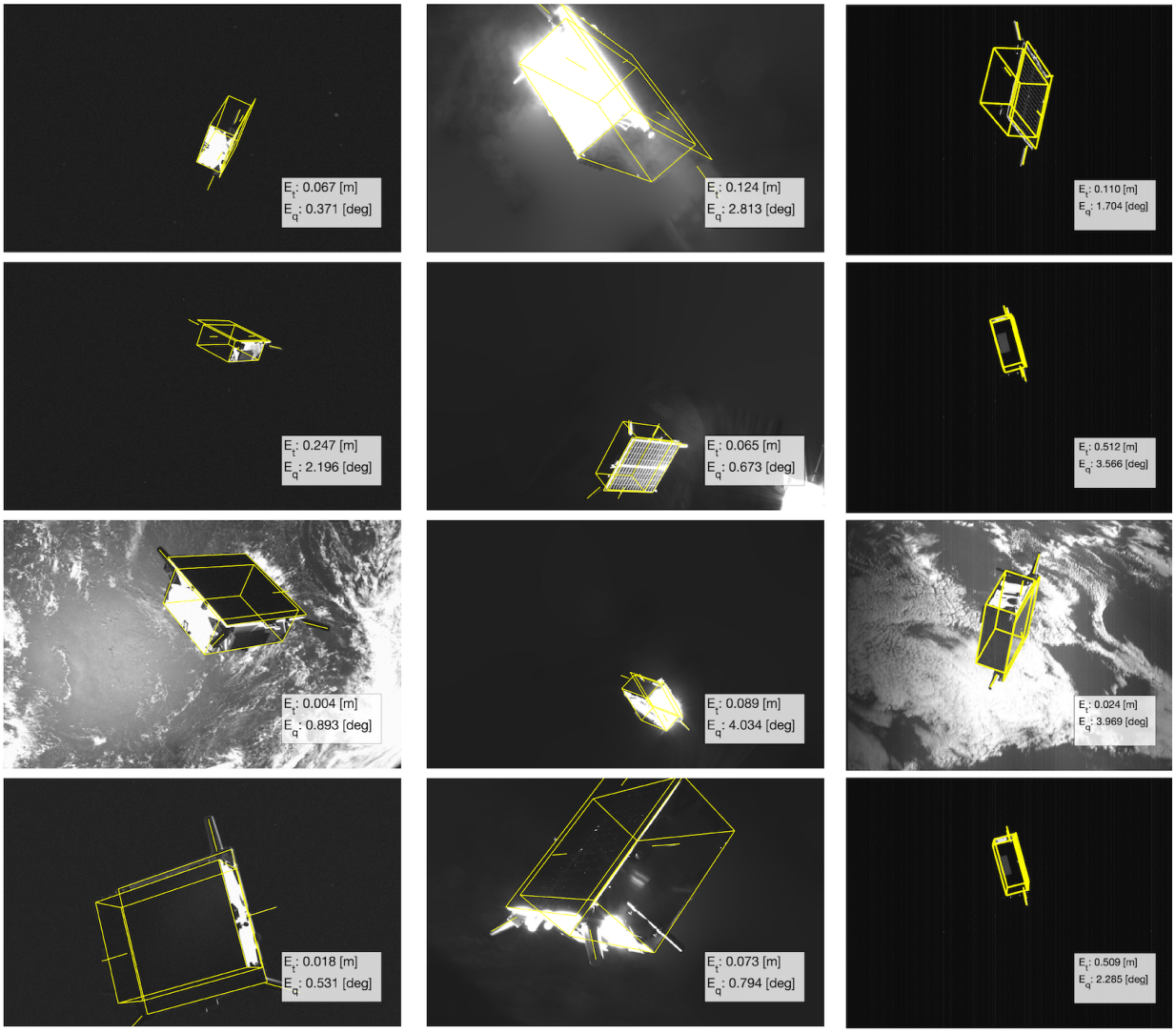
[1] Approx. from DarkNet-53 backbone [133] [2] Approx. from ConvNeXt-B backbone [134]

categories—TangoUnchained and lava1302— adopted adversarial training, directly utilizing the unlabeled images of the HIL domains. On the other hand, the proposed SPNv3 does *not* utilize any information about the HIL domain during the training phase. The performance of SPNv3 is comparable to that of EagerNet [80] which also does not involve the HIL domain images during training. Note that EagerNet uses a pre-trained ConvNeXt-Base network with 89M parameters and generates multiple pose hypotheses from dense predictions of model coordinates and predicted errors, which are further refined iteratively using a region-based approach. On the other hand, SPNv3-S is smaller and simpler, predicting keypoint locations from lower-resolution heatmaps after just a single forward propagation.

In the end, the ensemble performance with 448×448 inputs would rank SPNv3-S at first place in the `lightbox` category and third place in the `sunlamp` category of SPEC2021 [17]. Indeed, running an ensemble of multiple models may be computationally expensive especially onboard the satellite avionics. However, considering that the inference of SPNv3-S with 448×448 inputs only takes about 34 ms on a GPU of an NVIDIA Jetson Nano (see Table 6), it may be possible to run an ensemble of a few models in real-time during RPOD. The largest variant of equivalent size, SPNv3-B with 86.3M parameters, marginally improves the accuracy, nearly matching that of EagerNet on `lightbox` yet still remaining in third place on `sunlamp`. However, such a difference in performance between SPNv3-S and SPNv3-B would likely not yield significant improvement that justifies a near quadruple increase in size when processed by an onboard navigation filter.

E. Limitations

The proposed SPNv3 based on ViTPose is not without limitations. One major limitation is its dependence on a pre-trained ViT backbone. More specifically, Table 5 reveals that the OOD robustness of the downstream pose estimation task depends not just on what dataset the ViT backbone is pre-trained on but also on how the pre-training was conducted. This observation implies that those who cannot afford the computational power nor the expertise of ImageNet pre-training are restricted in exploring and fine-tuning different architectural options. In fact, the dependence on ImageNet pre-training prevents this work from exploring other aspects of ViT backbones, e.g., varying the patch sizes (P). Furthermore, the optimally pre-trained backbone may not exist, in which case the sub-optimally pre-trained alternative may have to be used for training on SPEED+. In response to such cases, the authors' previous work [19] shows how one can further train the sub-optimal NN onboard the satellite avionics in real-time during RPOD. However,



(a) `lightbox` **(b) `sunlamp`** **(c) `prisma25`**

Fig. 7 Projection of the Tango wireframe model based on the poses predicted by the ensemble of SPNv3-S@448 on the sample SPEED+ HIL and `prisma25` images.

if a NN can be trained according to the recipe analyzed in this section to be as OOD robust as possible, the online training on satellite avionics may be minimized or skipped altogether.

VI. Conclusion

This paper presented SPNv3, a computationally efficient and robust Neural Network (NN) model based on the Vision Transformer (ViT) architecture. SPNv3 is trained exclusively on computer-generated synthetic images to achieve state-of-the-art robustness on unknown spaceborne imagery. Then, it is tested exhaustively on the hardware-in-the-loop images from a robotic facility that can simulate high-fidelity spaceborne illumination conditions. Extensive analyses are performed to identify data augmentation, transfer learning, NN size and input image resolution as key aspects of the design and training that contribute the most to its Ouf-Of-Distribution (OOD) robustness. In the end, a mid-size SPNv3-S with 22.7M parameters can perform inference on a larger 448×448 input images at nearly 30Hz on a restrictive GPU environment, which is well above the nominal update frequency of modern satellite guidance, navigation and control systems. Therefore, this paper successfully demonstrated that SPNv3 is flight-ready, capable of achieving state-of-the-art

robustness while simultaneously meeting the prescribed computational requirements. Moreover, additional experiments reveal that SPNv3 can be trained to operate at a distance as far as where the target spacecraft occupies merely 10×10 pixels region albeit with more outliers.

The immediate next step is to address the requirement of power consumption by minimizing the memory footprint of NN while maintaining the OOD robustness of larger SPNv3 variants. Candidate methods include NN weight pruning and quantization to reduced-precision formats. Furthermore, the OOD robustness evaluation in this work relies on the SPEED+ dataset which is limited to a 10 m separation between the camera and the target due to the hardware restrictions. Therefore, the OOD robustness must be evaluated at a farther distance which predicates on improved calibration of the robotic testbed.

Funding Sources

This work is supported by the US Space Force SpaceWERX Orbital Prime Small Business Technology Transfer (STTR) contract number FA8649-23-P-0560 awarded to TenOne Aerospace in collaboration with SLAB.

References

[1] NASA, "On-orbit Servicing, Assembly, and Manufacturing 1 (OSAM-1)," `https://www.nasa.gov/mission/on-orbit-servicing-assembly-and-manufacturing-1/`, 2024. Accessed: 2023/12/11.

[2] NASA, "On-orbit Servicing, Assembly, and Manufacturing 2 (OSAM-2)," `https://www.nasa.gov/mission/on-orbit-servicing-assembly-and-manufacturing-2-osam-2/`, 2023. Accessed: 2023/12/11.

[3] Aglietti, G., Taylor, B., Fellowes, S., Ainley, S., Tye, D., Cox, C., Zarkesh, A., Mafficini, A., Vinkoff, N., Bashford, K., and et al., "RemoveDEBRIS: An in-orbit demonstration of technologies for the removal of space debris," *The Aeronautical Journal*, Vol. 124, No. 1271, 2020, p. 1–23. https://doi.org/10.1017/aer.2019.136.

[4] Astroscale, "ADRAS-J," https://astroscale.com/missions/adras-j/, 2024. Accessed: 2024/04/26.

[5] ESA, "ClearSpace-1 Mission," https://www.eoportal.org/other-space-activities/clearspace-1#european-industry-leads-debris-removal, 2020. Accessed: 2023/12/11.

[6] 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.

[7] 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.

[8] 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.

[9] Park, T. H., and D'Amico, S., *Rapid Abstraction of Spacecraft 3D Structure from Single 2D Image*, 2024. https://doi.org/10.2514/6.2024-2768.

[10] Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F., "Analysis of Representations for Domain Adaptation," *Advances in Neural Information Processing Systems*, 2007.

[11] Peng, X., Usman, B., Kaushik, N., Wang, D., Hoffman, J., and Saenko, K., "VisDA: A Synthetic-to-Real Benchmark for Visual Domain Adaptation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 2102–21025. https://doi.org/10.1109/CVPRW.2018.00271.

[12] Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., and Birchfield, S., "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *Proceedings of The 2nd Conference on Robot Learning*, Proceedings of Machine Learning Research, Vol. 87, edited by A. Billard, A. Dragan, J. Peters, and J. Morimoto, PMLR, 2018, pp. 306–316. URL https://proceedings.mlr.press/v87/tremblay18a.html.

[13] Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D., "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" *IEEE Robotics and Automation Letters*, Vol. 5, No. 4, 2020, pp. 6670–6677. https://doi.org/10.1109/LRA.2020.3013848.