POLITECNICO DI TORINO
Repository ISTITUZIONALE

Experimental Verification of a Monocular CNN-Based Pose Estimation Algorithm for the SROC Mission

Original
Experimental Verification of a Monocular CNN-Based Pose Estimation Algorithm for the SROC Mission / D'Ortona, A., Lovaglio, L., Visca, M., Stesina, F., Corpino, S.. - In: CEAS SPACE JOURNAL. - ISSN 1868-2502. - (In corso di stampa).

Availability:
This version is available at: 11583/3018318 since: 2026-11-23T12:24:07Z

Publisher:
Springer

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026

Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Experimental Verification of a Monocular CNN-Based Pose Estimation Algorithm for the SROC Mission

Antonio D'Ortona

antonio.dortona@polito.it

Polytechnic University of Turin

Lucrezia Lovaglio

Polytechnic University of Turin

Marco Visca

Tyvak International

Fabrizio Stesina

Polytechnic University of Turin

Sabrina Corpino

Polytechnic University of Turin

Research Article

Keywords: On-Orbit Servicing (OOS), rendezvous and docking, pose estimation, visual navigation, Convolutional Neural Network (CNN), hyperparameter optimization

Posted Date: January 21st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8545599/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Experimental Verification of a Monocular CNN-Based Pose Estimation Algorithm for the SROC Mission

Antonio D'Ortona[1], Lucrezia Lovaglio[1], Marco Visca[2], Fabrizio Stesina[1], Sabrina Corpino[1]

[1]Mechanical and Aerospace Engineering Department, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Turin, Italy.
[2]Tyvak International, Via Orvieto 19, 10149 Turin, Italy.

Abstract

This work presents a Convolutional Neural Network (CNN)-based framework for monocular pose estimation of spacecraft in the context of the ESA Space Rider Observer CubeSat (SROC) mission. The mission, led by a consortium of Italian institutions, involves autonomous proximity operations including inspection and docking maneuvers with the Space Rider vehicle. Given the constraints of CubeSat-class hardware, we explore an efficient and adaptable CNN-based solution for relative navigation using only optical input.

In order to address the challenges of domain gap between synthetic and real data, a high-fidelity synthetic dataset was generated using Blender with realistic lighting conditions and applying dataset augmentation techniques. Real image datasets were acquired at ESA's GRALS facility with a COTS camera to test the trained models with a 1:20 mockup of Space Rider. A hyperparameter optimization (HPO) was conducted using the Optuna framework to enhance training performance. Multiple training strategies were evaluated, including fine-tuning techniques and dataset mixing approaches, to enhance performances on a specific maneuver that requires greater accuracy.

The trained networks were deployed on the Payload Processing Unit using the ONNX Runtime framework for inference. Tests demonstrated consistent and reliable performance across multiple scenarios. These results confirm that, when supported by domain adaptation and hyperparameter optimization, the proposed CNN architecture can achieve accurate pose estimation under limited computational resources, making it a strong candidate for future autonomous CubeSat applications.

Keywords: On-Orbit Servicing (OOS), rendezvous and docking, pose estimation, visual navigation, Convolutional Neural Network (CNN), hyperparameter optimization

1 Introduction

1.1 State of the art

Spacecraft rendezvous and docking requires accurate relative pose estimation, which is the process of determining the position and orientation of a spacecraft using information from various sensors, such as IMU, GPS, vision, laser, and ultrasonic systems. In the CubeSat context, pose estimation must rely on highly miniaturised hardware, including both the sensor suite and the onboard embedded systems. Such components typically exhibit lower performance and technology readiness level (TRL) than those used on larger spacecraft. As a result, the available sensor suite often relies on active sensing techniques, in particular LIDAR [1] and time-of-flight (ToF) sensors [2].

Alternatively, passive vision-based systems have demonstrated strong potential for accurate pose estimation. When a collaborative target is available, predefined-pattern recognition techniques can be employed, such as using active illumination sources to detect fiducial light patterns [3], or relying on passive visual markers like black-and-white ArUco codes [4]. Given prior knowledge of the 3D positions of these markers, the Perspective-n-Point (PnP) algorithm [5] can be used to estimate the relative pose of the target with respect to the camera.

In scenarios where the use of visual markers has limited application, feature-based techniques provide an alternative for relative pose estimation. These algorithms typically employ scale and rotation-invariant descriptors extracting and matching distinctive features across multiple frames to infer 3D structure and relative motion, and in some cases, they also offer robustness to affine transformations [6]. Past space missions implemented some of the classical methods described before, e.g. in NASA CPOD (CubeSat Proximity Ops Demo) [7], edge detection and feature comparison algorithms performed on the camera outputs are then matched to the catalog to provide position and attitude information. In Surrey's RemoveDEBRIS (2018) [8], they computed the images captured on the ground to compute the relative pose with 2D/3D and 3D/3D matching techniques. Seeker (3U) [9, 10] implemented a similar method based on the combination of traditional feature-based SIFT and neural network.

Despite their effectiveness, such classical approaches can struggle in space applications, where the harsh environment significantly increases the uncertainty in sensor outputs. Their accuracy often decreases in low-texture or poorly illuminated scenes, and their high computational costs may exceed the capabilities of the onboard systems. In contrast, Deep Learning (DL) based methods such as Convolutional Neural Networks (CNNs) have emerged as a powerful alternative for monocular spacecraft pose estimation. By learning hierarchical spatial representations directly from image data, CNNs can achieve robust feature extraction even under challenging conditions. Their strong performance across diverse vision tasks—ranging from object detection to pose

estimation—combined with efficient inference capabilities, makes them particularly suitable for real-time processing on flight hardware [11].

Recent monocular spacecraft DL pose-estimation methods span both CNNs and transformers: the CNN SPN directly regresses 6-DoF pose from single images [12], while hybrid pipelines pair CNN detection (e.g., MobileNetV2+YOLOv3) with tracking and PnP [13–15]; the CNN-based model SPNv2 extends this with an EfficientDet multitask backbone predicting boxes, keypoints, and pose jointly [16, 17]. Other CNN variants infer pose via homography constraints [18] or fuse VAE (Variational Auto-Encoder)-learned features with geometric cues [19], and newer transformer-based approaches-such as SPTN [20], domain-gap-aware ViT pipelines [21], and multitask designs like PVSPE [22]-are emerging alongside model-based 6D pose frameworks using approximate 3D models [23].

Although DL architectures achieve strong pose-estimation performance, results depend heavily on training choices, especially hyperparameters and data quality. Hyperparameters such as learning rate, batch size, and network depth strongly affect convergence and generalization [24–26], yet manual tuning or grid search is inefficient; thus we adopt automated HPO with Optuna to efficiently search the space using probabilistic, history-guided optimization [27, 28]. However, even optimal hyperparameters require representative training data, since CNNs trained on synthetic images often degrade on real imagery due to domain gap effects [29, 30], motivating realistic, well-augmented datasets.

1.2 Domain gap

The domain gap between synthetic training images and real space imagery has been widely highlighted, for instance in the Satellite Pose Estimation Competitions (SPEC) 2019 and 2021 [29, 30]. Since large labeled in-orbit datasets are scarce or unavailable before launch, most CNNs are trained on synthetic renders, and mismatches in illumination, reflections, noise, and sensor artifacts can markedly degrade performance on real data. The winning solutions in SPEC partially addressed this by exploiting SPEED+, which includes HIL images acquired with the TRON testbed at SLAB [31, 32].

To mitigate the synthetic-to-real gap, two main families of techniques are commonly adopted. Unsupervised Domain Adaptation (UDA) leverages unlabeled target images during training, e.g. via style-transfer self-training, pseudo-label refinement, or test-time adaptation [33–35]. Domain randomization instead aggressively varies rendering conditions so that models learn domain-invariant features, often implemented through extensive data augmentation with software libraries such as the open-source Albumentations [36, 37].

1.3 Hardware implementation

Running deep learning algorithms on spacecraft on-board processors is constrained by limited power, harsh environmental conditions, and stringent reliability and radiation-tolerance requirements. Consequently, flight hardware must balance robustness, energy efficiency, and computational capability, while remaining compatible with ground-trained models.

Recent studies indicate that low-power accelerators can make on-board AI feasible. For example, [38] demonstrate state-of-the-art accuracy with a highly compact network running at 7.7 fps on a Coral Dev Board Mini (Edge TPU) with only 2.2 W power consumption, while [39] deploy a ViT-based network on a Jetson Nano with latencies below 40 ms per 224×224 image. Beyond commercial platforms, [40] presents an FPGA-based architecture for the ASI-funded ANHEO project, targeting nanosatellite-class hardware and achieving real-time inference with low power, underscoring the suitability of such solutions for CubeSat on-board processing.

1.4 The mission

This work is part of the ESA mission Space Rider Observer CubeSat (SROC), led by Tyvak International as the prime contractor of a fully Italian consortium. The consortium includes Politecnico di Torino, Università di Padova, and Stellar Project s.r.l. The Space Rider Observer Cubesat (SROC) [41] mission is a CubeSat mission designed to demonstrate key capabilities in the field of proximity operations, rendezvous, and docking within a safety-sensitive framework. As a payload of the European Space Agency's (ESA) Space Rider mission, SROC represents a significant step in the development and testing of novel technologies for nanosatellite-based inspection and autonomous operations in low Earth orbit (LEO). SROC consists of a nanosatellite equipped with a Docking System and a CubeSat dispenser for spacecraft deployment and retrieval. After being released from the cargo bay of the Space Rider vehicle, the CubeSat will execute a series of autonomous maneuvers to establish and maintain a safe distance from its mothership. The mission will then proceed to an observation phase, during which SROC will inspect Space Rider for potential damage, capturing high-resolution images to evaluate the vehicle's condition. Finally, the CubeSat will initiate a straight-line docking maneuver, demonstrating precise, controlled reattachment before Space Rider reenters Earth's atmosphere.

1.5 Objectives

The aim of this study is to validate the applicability of DL-based pose estimation for the ESA SROC CubeSat mission using a commercial off-the-shelf (COTS) camera, with a focus on real-time on-board execution during proximity operations. In particular, the proposed solution estimates the 6-DoF relative pose of the Space Rider target from a single monocular image acquired by a visible-spectrum optical payload. The mission imposes a set of requirements for the Visual Navigation during the straight-line final approach to docking.

To this end, a CNN-based framework is employed, leveraging the EfficientPose-inspired architecture proposed in [16], previously validated on the SPEED+ benchmark for robustness to domain-gap effects, attributes that make this algorithm particularly suitable to meet the mission requirements. This architecture was selected both for its demonstrated performance under synthetic-to-real domain shift and for the maturity of the CNN ecosystem (toolchains and deployment frameworks), which currently exceeds that of alternative DL paradigms such as Vision Transformers (ViTs). An additional advantage is the scalability inherited from EfficientDet: network depth

and width can be adjusted without altering the overall design, enabling systematic exploration of model sizes to satisfy on-board storage and memory constraints.

The model is trained exclusively on Blender-generated synthetic images that simulate realistic orbital illumination and SROC maneuver profiles. Generalization is boosted with strong augmentations (Albumentations [37]) and style augmentation [42], plus maneuver-aware loss weighting to emphasize operationally critical cases. Particular emphasis is placed on the final-approach regime: the network is fine-tuned on this subset, and training/evaluation pipelines are configured to prioritize translation and rotation accuracy where docking requirements apply.

Because CNN performance is sensitive to training choices, automated hyperparameter optimization (Optuna) is used to minimize pose loss. Building on [27], the search space is changed (e.g., learning rate/schedule, augmentation strength, loss weights, backbone/neck width and depth), and alternative training/fine-tuning strategies are compared to quantify gains in domain robustness and final-approach performance.

Training is implemented in PyTorch; the resulting model is exported to ONNX and executed with ONNX Runtime in a C++ pipeline, enabling platform-independent, efficient inference on the SROC Payload Processing Unit (PPU).

Finally, end-to-end performance is verified at ESA/ESTEC's GRALS facility [43], including inference on flight-representative hardware, providing a realistic assessment of mission readiness.

The general architecture developed is resumed in Figure 1.

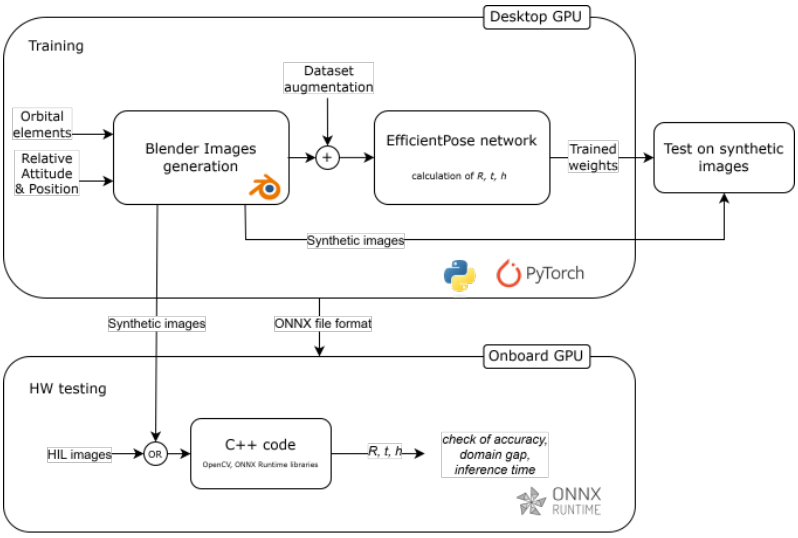


Fig. 1: System architecture

The main contributions of this work are:

- the development of a robust, tailored, and feasible convolutional neural network (CNN)–based algorithm for pose estimation, applied to an unprecedented dataset specifically generated for training and performance assessment;
- the definition of a verification strategy to evaluate pose-estimation performance against mission requirements under relevant conditions;
- the assessment of the proposed solution along different representative trajectories flown during the mission with a COTS camera and a low-budget target mockup;
- the implementation and verification on the flight hardware.

This paper is organized as follows. Section 2 presents the dataset generation pipeline, detailing the techniques employed to simulate realistic training data. Section 3 introduces the architecture of the proposed neural network. Section 4 discusses the hyperparameter optimization strategies adopted during the training phase. Section 5 outlines the experimental setup used to acquire the real dataset at the GRALS facility. Section 6 reports and analyzes the results obtained under different training strategies. Finally, Section 7 concludes the paper with final considerations and future perspectives.

2 Dataset creation

This chapter describes the generation of a synthetic image dataset reproducing the relative motion between SROC and Space Rider, used to develop, train, and verify CNN-based visual navigation algorithms for proximity operations. Blender, an open-source 3D modelling and rendering tool, is adopted for its realism and powerful Python scripting, despite not being a spectral renderer and not providing radiance outputs, which limits radiometric accuracy.

2.1 Image generation

To replicate orbital conditions, a Blender-based pipeline was developed using layered rendering of Earth, Space Rider, and scene lighting, later combined through node-based *compositing* [44]. High-resolution Earth textures, volumetric atmosphere, and a low-/high-poly modeling workflow with UVs [45] and UDIMs [46] enhanced geometric and textural realism. Realism was further improved with render passes (*Glossy Direct*, *Emission*) [47], glare effects, chromatic aberration, and sensor-like noise, reducing the domain gap with real images. Datasets were rendered using the Cycles engine with OptiX on an NVIDIA RTX 3090.

2.2 Python API

Blender's Python API is used to script the placement and motion of all objects. As shown in Fig. 2, the scene combines Earth and Space Rider models to reproduce the chaser–target maneuvers, with the key inputs being the relative position and attitude between SROC and Space Rider. The Blender World Reference Frame (WRF) is assumed coincident with the LVLH frame, so all objects are defined directly in LVLH, avoiding explicit inertial-to-local transformations and orbit propagation for the relative geometry. The LVLH frame is defined with the Z-axis pointing in the

radial direction toward Earth (nadir), the X-axis aligned with the spacecraft velocity (along-track), and the Y-axis completing a right-handed frame (cross-track) [48].

The scene is assembled from external files providing LVLH quantities: (a) SROC position; (b) Earth position (by definition $r_{\text{Earth}} = [0, 0, h_{\text{orbit}} + R_{\text{Earth}}]$); (c) Sun position; (d) Space Rider attitude quaternion; (e) SROC decentralization quaternion. Only the Sun position requires orbit propagation, which is computed from the mission duration and orbital parameters. Figure 2 summarizes the required inputs for image generation.

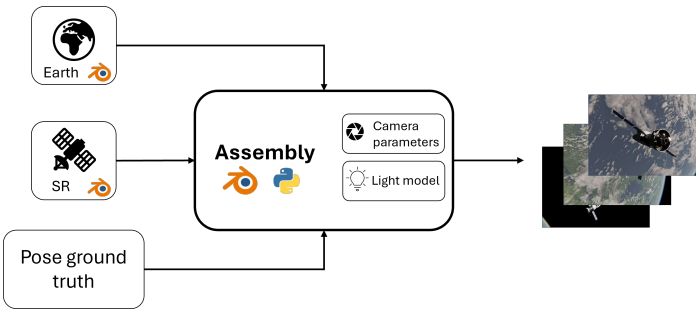


Fig. 2: Blender architecture

2.2.1 Observation dataset

The observation dataset represents medium-range inspection of Space Rider. SROC is placed on an elliptical relative trajectory around the target, with positions sampled in the Space Rider body frame: distances d and attitudes uniformly distributed to ensure diverse viewpoints. For the Blender setup, the Space Rider frame is initially aligned with the LVLH frame, so Sun and Earth geometry can be derived directly from the orbital parameters. Images are generated in Blender through the Python API, which controls object placement and attitude in the world (LVLH) frame (Fig. 3).

SROC is first oriented so that the camera boresight points to the origin of the Space Rider body frame. The camera Z-axis is $\mathbf{Z}_C = [0, 0, 1]$, and the relative position along the Space Rider axes, $\mathbf{r}_{\text{SROC}} \in \mathbb{R}^3$, is randomly sampled in the range above. Since SROC's initial attitude coincides with LVLH, the quaternion $\mathbf{q} \in \mathbb{R}^4$ that aligns $\mathbf{Z}_C$ with $-\mathbf{r}_{\text{SROC}}$ is

$$\begin{gathered} \alpha = \arccos\left(\mathbf{Z}_C \cdot \frac{-\mathbf{r}_{\text{SROC}}}{\|\mathbf{r}_{\text{SROC}}\|}\right), \quad \mathbf{a} = \frac{\mathbf{Z}_C \times (-\mathbf{r}_{\text{SROC}})}{\|\mathbf{Z}_C \times (-\mathbf{r}_{\text{SROC}})\|} \\ q_0 = \cos\tfrac{\alpha}{2}, \quad \mathbf{q}_V = \mathbf{a}\sin\tfrac{\alpha}{2} \\ \mathbf{q} = [q_0, \mathbf{q}_V] \end{gathered} \tag{1}$$

This quaternion is passed to Blender to orient SROC. The Space Rider's attitude is independently randomized by sampling and normalizing a quaternion $\mathbf{q}_{\text{SR}} \in \mathbb{R}^4$.

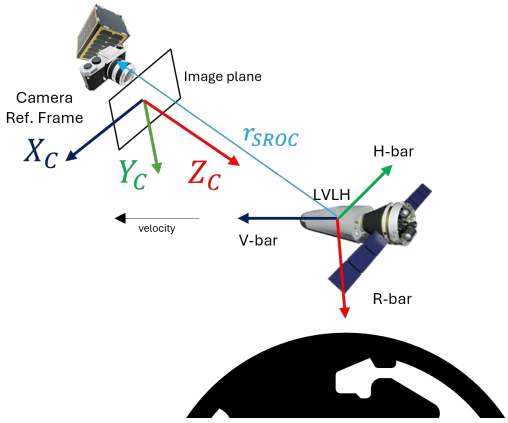


Fig. 3: Reference frames

A perfectly centered image corresponds to $\mathbf{r}_{\text{camera}} = [0,\, 0,\, d]$ in the camera frame. To generate off-center views while keeping Space Rider within the field of view, additional rotations about the camera axes X_C and Y_C are applied. Denoting these angles as ϕ (about X_C) and θ (about Y_C), their admissible values are obtained from the distance d, a reference target size L_{ref} (taken as the solar-panel span), and the camera model:

$$\phi = atan\left(\left(\frac{px_h}{2} - \frac{L_{\text{ref}} f}{d\, L_{px}}\right) L_{px} \frac{d}{f}\right), \quad \theta = atan\left(\left(\frac{px_w}{2} - \frac{L_{\text{ref}} f}{d\, L_{px}}\right) L_{px} \frac{d}{f}\right). \tag{2}$$

The corresponding inputs are summarized in Table 1.

Table 1: Inputs parameters for the observation dataset

Variable	Values
Pixels along sensor vertical axis (px_h)	1536
Pixels along sensor horizontal axis (px_w)	2048
Pixel size (L_{px})	2.2×10^{-6} m
Reference length (largest dimension of Space Rider) (L_{ref})	Solar panel span
Focal length (f)	70×10^{-3} m (Obs. Dataset) 3×10^{-3} m (FA Dataset)
SROC–Space Rider distance (d)	[230, 900] m (Obs. Dataset) [10, 42] m (FA Dataset)
Sun azimuth in LVLH	[13.77, 166.23] deg
Sun elevation in LVLH	[−76.23, 76.23] deg

As d increases, the allowable decentralization angles grow because the target occupies fewer pixels. Random off-centering is obtained by multiplying ϕ and θ by a

uniform factor in $[-1, 1]$, yielding physically consistent off-center views. The associated decentralization quaternion is

$$\mathbf{q}_x(\phi) = \left[\cos\tfrac{\phi}{2},\ \sin\tfrac{\phi}{2},\ 0,\ 0\right], \quad \mathbf{q}_y(\theta) = \left[\cos\tfrac{\theta}{2},\ 0,\ \sin\tfrac{\theta}{2},\ 0\right], \tag{3}$$

$$\mathbf{q}_{\text{dec}} = \mathbf{q}_y(\theta) \otimes \mathbf{q}_x(\phi), \qquad \mathbf{q}_{\text{final}} = \mathbf{q} \otimes \mathbf{q}_{\text{dec}}, \tag{4}$$

where $\otimes$ denotes quaternion multiplication. $\mathbf{q}_{\text{final}}$ is used as the final SROC attitude. Example images are shown in Fig. 4.



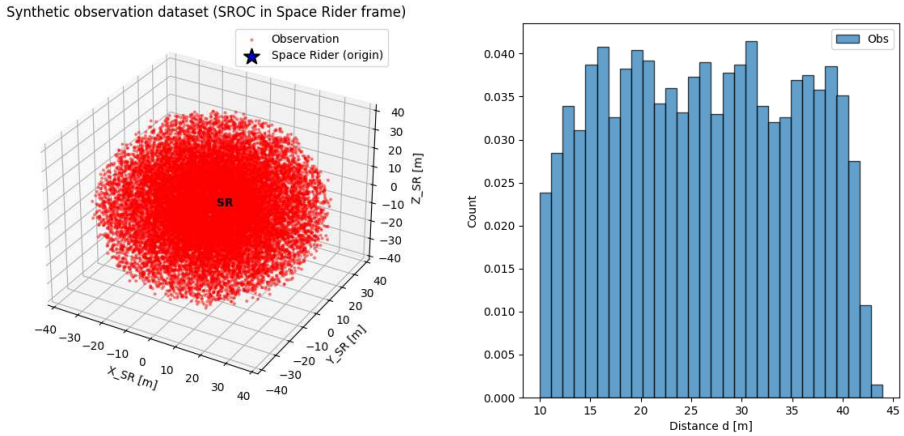
Fig. 4: Example images from the observation maneuver

2.2.2 Final approach dataset

The final approach dataset models the terminal docking phase. Unlike the observation dataset, Space Rider's attitude is kept fixed: the docking-port axis (body Z_{SR}) is aligned with the V-bar, while the spacecraft nose points toward Nadir. With X_{SR} directed from the CoG to the nose, this configuration is described by the quaternion $\mathbf{q}_{\text{SR}} = [0,\ 0.7071,\ 0,\ 0.7071]$, aligning the docking port with the positive V-bar.

In this phase, SROC moves along the V-bar with distances $d \in [8, 50]$ m, so the LVLH relative position is $\mathbf{r}_{\text{SROC}} = [X,\ 0,\ 0]$. SROC's attitude is further decentralized as in the observation case to account for uncertainties in the docking-port location and possible attitude maneuvers (e.g., to optimize power). The camera parameters are identical except for the focal length, which is reduced during the final approach to keep the target's apparent size comparable at shorter ranges (Table 1). For training consistency, all synthetic images were rendered in Blender using a single pinhole model with fixed focal length ($f = 3$ mm) and no lens distortion. During evaluation on real data, images are first undistorted using the corresponding camera calibration (Sec. 5.2) and then rescaled to match the training intrinsics.

This configuration concentrates samples in the critical docking region, where accurate pose estimation is most important. Example images are shown in Fig. 6.



(a) Relative position in the synthetic observation dataset.

(b) Normalized distribution of target distance in the observation dataset.

Fig. 5: Synthetic observation dataset coverage in the Space Rider frame.

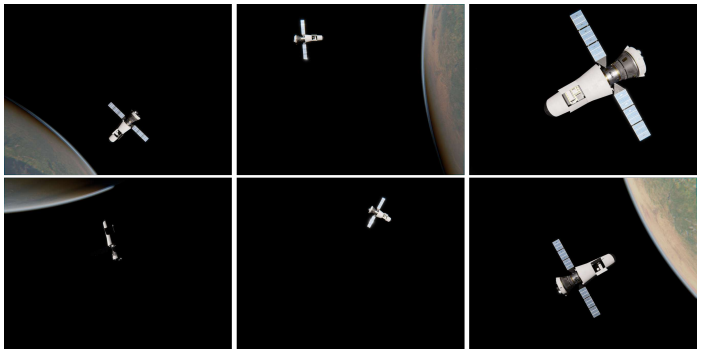
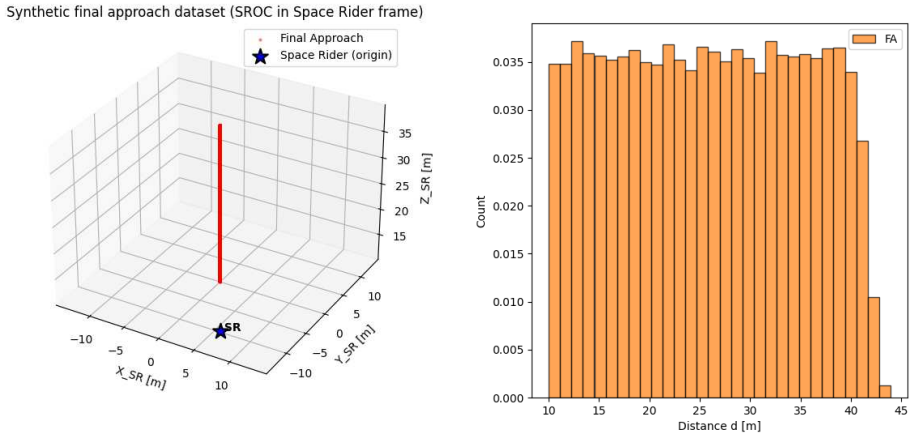


Fig. 6: Example images from the final approach maneuver

Figure 5 and 12 summarize the spatial properties of the synthetic observation and final approach dataset. The relative target positions in the Space Rider frame (left) show an approximately uniform and isotropic sampling around the spacecraft for the observation dataset, and a straight line shape for the final approach dataset. The normalized distance histogram (right) confirms balanced coverage across the full observation range, avoiding range-dependent bias in the training data.

2.2.3 Dataset Augmentation

To improve generalization and limit overfitting-particularly given the reliance on synthetic imagery-dataset augmentation is applied during training. We use the



(a) Relative position in the synthetic final approach dataset.

(b) Normalized distribution of target distance in the final approach dataset.

Fig. 7: Synthetic final approach dataset coverage in the Space Rider frame.

Albumentations library [37], which offers efficient, composable transformations. The adopted augmentations include (i) style-based operations (e.g., blur, texture and appearance changes) and (ii) geometric transformations (e.g., horizontal and vertical flips). In particular, a style-randomization strategy inspired by [42] is used to perturb texture, contrast, and color while preserving scene geometry, encouraging the network to focus on pose-relevant spatial cues and improving cross-domain robustness. All augmentations are activated with a probability of 50% during training. The full list of transformations is reported in Table 2.

Table 2: List of the Data Augmentation and equivalent commands applied [16]

Augmentation Techniques	**Functions**
Horizontal Flip	$A.HorizontalFlip$
Vertical flip	$A.VerticalFlip$
Noise	$A.OneOf(A.GaussNoise, A.ISONoise)$
Blur	$A.OneOf(A.MotionBlur,$ $A.MedianBlur, A.GlassBlur)$
Brightness and Contrast	$A.RandomBrightnessContrast$
Domain randomization	https://github.com/philipjackson/style-augmentation[42]

3 Pose Estimation CNN

Figure 9 illustrates the complete algorithm architecture, a customized implementation inspired by the EfficientDet-based Spacecraft Pose Network v2 (SPNv2) architecture

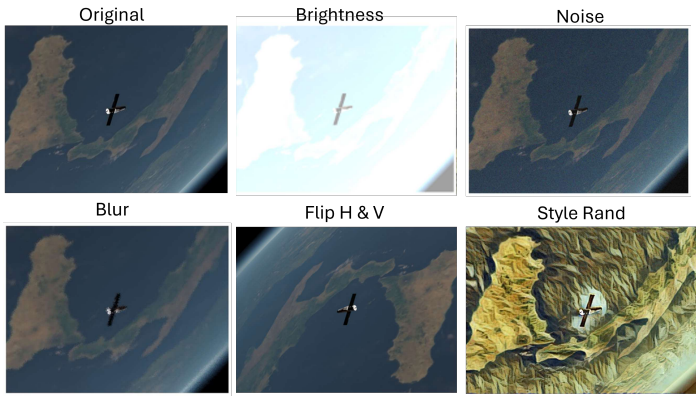


Fig. 8: An example image from the synthetic dataset with the augmentations listed in Table 2 applied.

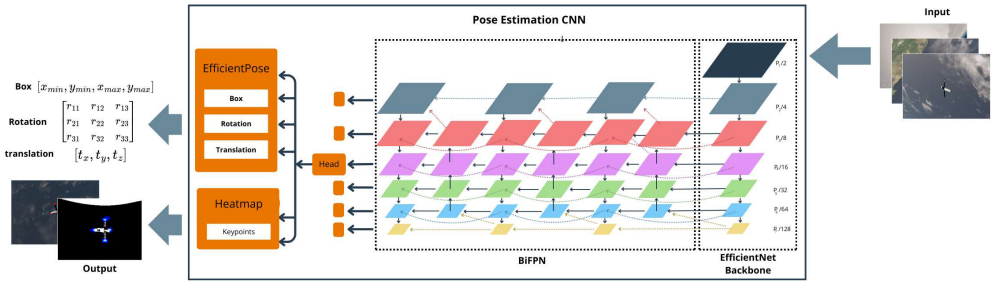


Fig. 9: CNN architecture

described in [16]. It leverages an EfficientDet [17] backbone, specifically utilizing an EfficientNet feature extractor combined with a Bidirectional Feature Pyramid Network (BiFPN) to produce multi-scale features, enabling robust feature representation across multiple resolutions. This network processes the dataset and outputs the target spacecraft's pose through two main prediction heads: one estimating the rotation matrix and translation vector (EfficientPose head), and the other generating heatmaps corresponding to predefined keypoints on the target. A third head is included for instance segmentation, which outputs a binary mask to enhance generalization and spatial precision. The network is trained jointly on all tasks — pose regression, keypoint localization, and segmentation — allowing the shared backbone to learn richer and more robust features. This multi-task setup improves convergence and enhances overall performance, particularly in cluttered or occluded scenes [49]. A distinguishing feature of SPNv2 is its scalability, which allows for adjustment of the model's depth and width via the EfficientDet compound scaling parameter ϕ [50], to meet specific performance and computational constraints. Thanks to this characteristic, the entire network structure can be easily modified by changing a single scaling parameter, effectively adjusting the computational load (in terms of FLOPS). This allows for

efficient experimentation with different configurations tailored to specific performance and resource constraints.

3.1 Backbone (EfficientDet)

The backbone follows EfficientDet and includes (i) an ImageNet-pretrained EfficientNet [51] for feature extraction and (ii) a Bidirectional FPN (BiFPN) for multi-scale fusion [17, 50]. As anticipated before, EfficientNet uses compound scaling ϕ to jointly scale depth, width, and input resolution [50]. Unlike the isotropic scaling originally proposed by EfficientNet [50], this study adopts a rectangular input resolution (see Table 1), maintaining a fixed, non-square aspect ratio due to application-specific constraints. The BiFPN performs top-down/bottom-up aggregation with learnable fusion weights; P2 features are included to support high-resolution auxiliary heads. Consistent with EfficientDet, BiFPN width and depth scale with ϕ [17].

3.2 Prediction heads

Three heads are employed to retrieve the outputs:

1. **EfficientPose (pose) head.** Operates on P3–P7 and comprises: (i) a binary objectness branch with focal loss [52]; (ii) a box regression branch with IoU-based loss (e.g., CIoU) [53]; (iii) and a pose branch using the continuous 6D rotation representation [54] and a PoseCNN-style translation parameterization (center+t_z) [11], with SPEED-style rotation/translation errors for evaluation [29].
2. **Heatmap (keypoints) head.** Predicts Gaussian heatmaps for predefined keypoints at higher resolution (P2), providing geometric cues that regularize pose.
3. **Segmentation head.** Produces a binary instance mask (P2), trained with synthetic labels, to sharpen object boundaries and suppress background clutter, benefiting overall pose stability.

4 Hyperparameter optimization

Hyperparameters strongly affect training efficiency and final accuracy, influencing convergence speed and generalization; poor choices can cause slow convergence or under/overfitting [26]. Rather than relying on subjective trial-and-error, hyperparameter optimization (HPO) provides a systematic way to explore the search space and improve performance.

In this work, Optuna [55] is used for hyperparameter optimization (HPO), adopting a Bayesian optimization strategy [56]. Optuna provides a lightweight and flexible framework, supporting efficient sampling and pruning strategies, and includes visualization tools for monitoring optimization progress. The HPO objective is to minimize the sum of pose estimation loss defined as the sum of the translation loss and rotation loss [29]:

$$E_{TN} = \left\| \frac{t_{pr} - t_{gt}}{t_{gt}} \right\|, \quad E_R = \operatorname{acos}\left(\frac{\operatorname{tr}(R_{gt}^T R_{pr}) - 1}{2} \right) \tag{5}$$

$$\mathcal{L}_{pose} = E_R + E_{TN}$$

This work builds on [27] but adopts a more targeted HPO strategy. Instead of treating the learning-rate (LR) scheduler as a categorical choice, a single scheduler family is selected and its internal parameters (initial LR, warm-up, decay rate/milestones) are jointly tuned, an aspect not explored in [27]. This avoids overlooking near-optimal configurations hidden within a scheduler's settings. Furthermore, the search space was reduced which curbs trial count and objective variance, and simplifies comparisons under fixed compute budgets [57]. Constraining the search thus improves sample efficiency and stability. Overfitting is addressed separately by capping the optimization budget and using a held-out test set.

The search space includes: (a) initial learning rate, (b) number of warm-up epochs, (c) polynomial decay exponent, and (d) total number of epochs. Alongside the search space, both the pruning strategy and sampling algorithm are specified. Early stopping is handled with the Asynchronous Successive Halving Algorithm (ASHA) [58], which extends the classical method [59] by allowing asynchronous trial evaluation, eliminating idle resources, and reallocating effort to promising configurations. For sampling, the Tree-structured Parzen Estimator (TPE) [60] is used, balancing exploration of new regions with exploitation of promising ones.

The adopted optimization pipeline proceeds as follows: (1) **Initial trial:** a training with a random configuration of hyperparameters is completed, establishing a baseline validation score. (2) **Trials 2–10:** additional random samples are tested. Intermediate validation metrics are periodically reported; ASHA pruning is activated after a minimum number of validation steps to allow sufficient exploration before terminating unpromising trials. Regardless of pruning, the best validation score is recorded. (3) **From trial 11 onward:** TPE guides sampling based on prior results, while ASHA continues pruning underperforming trials asynchronously.

Both completed and pruned trials contribute to the surrogate model maintained by TPE, continuously refining the search toward promising hyperparameter regions. This staged, asynchronous strategy effectively balances exploration and exploitation while optimizing resource use. Finally, Optuna's visualization tools are used to analyze optimization history and identify the most influential hyperparameters.

5 GRALS Facility Setup

The experimental setup used at the GRALS facility is illustrated in Fig. 11. A comparable configuration is described in [43], where similar experiments were conducted in the same laboratory.

The testbed consists of the following components: (a) a 1:20 scale Space Rider mockup. The mockup was fabricated using additive manufacturing techniques. The main body structure was 3D printed with Polyethylene Terephthalate Glycol (PETG), selected for its durability and dimensional stability. The solar panel base structures were also printed in PETG, while the panel surfaces were covered with actual cuttings of solar cells. As illustrated in Fig. 4 and Fig. 13, the mockup presents a simplified geometry compared to the full Space Rider configuration. Certain structural details, such as the cargo bay and finer surface features, have been omitted. This design trade-off was made to balance visual realism and manufacturability while maintaining the

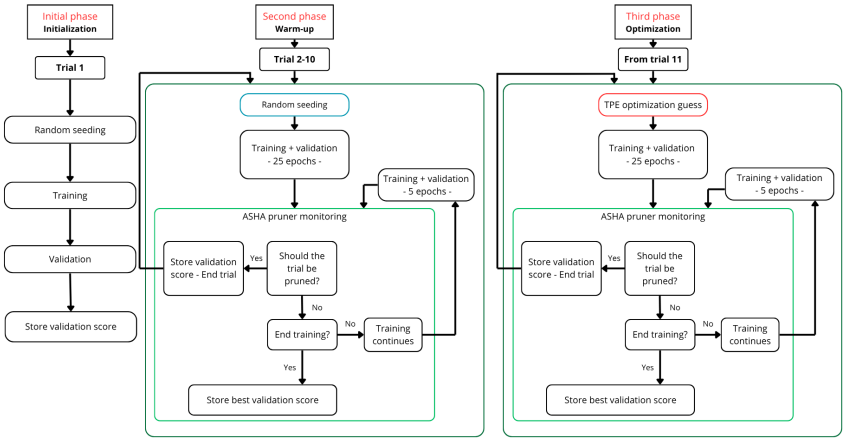


Fig. 10: General HPO pipeline

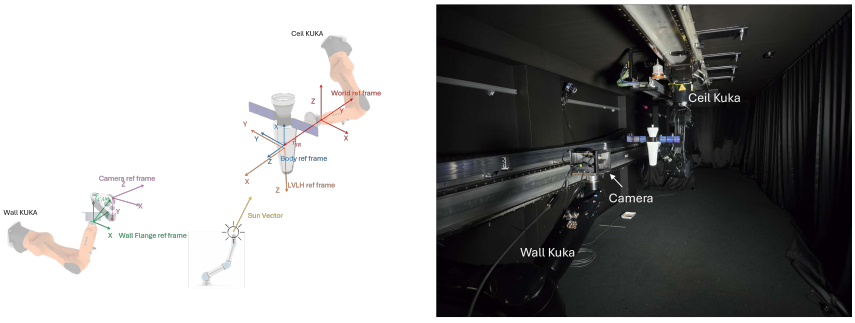


Fig. 11: Overview of the GRALS experimental setup.

geometric fidelity necessary for pose estimation tasks. (b) a camera assembly including both the camera and the Payload Processing Unit (PPU); (c) a wall-mounted KUKA robotic arm; (d) a ceiling-mounted KUKA robotic arm; (e) a collimated lamp mounted on a UR-5 robotic arm. The camera is attached to the wall-mounted KUKA arm, while the mockup is mounted on the ceiling arm. Both KUKA arms are equipped with 6 degrees of freedom and are mounted on linear rails, providing an additional translational degree of freedom (dof) and resulting in a total of 12 dof.

Illumination is provided by a collimated and dimmable lamp with a spectral output centered around 6000 K, mounted on the UR-5 arm. The lamp is directed at the mockup to simulate realistic lighting conditions during image acquisition.

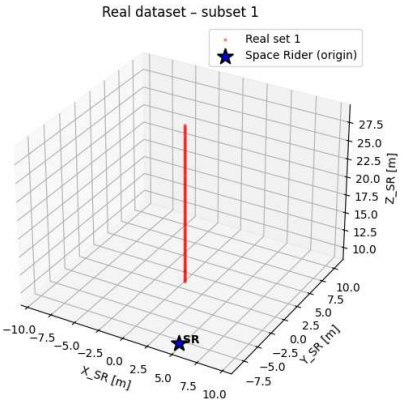
5.1 Dataset collection

To evaluate the performance and robustness of the proposed CNN-based pose estimation algorithm, three distinct experimental datasets were acquired at the GRALS facility. Each dataset was designed to replicate a different scenario of the mission involving varied relative positions and orientations between the camera and the Space Rider mockup. These experiments aim to assess the algorithm's effectiveness under conditions that replicate real in-orbit operations.

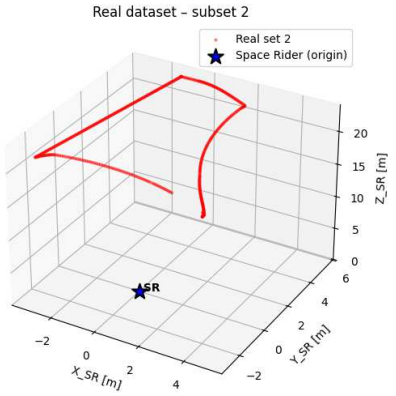
- **Dataset 1 – V-bar Approach:** The first dataset simulates a final approach phase along the V-bar (velocity vector) axis. In this experiment, the Space Rider mockup remains fixed while the camera follows a straight-line trajectory toward the docking interface, moving from a distance of 35 m down to 10 m (scaled to 1.75 m to 0.5 m in the facility). This setup replicates the final docking maneuver and tests the CNN's ability to estimate the pose across varying distances under aligned approach conditions. Total images captured: 860.
- **Dataset 2 – Top Inspection:** The second dataset represents a rotational inspection maneuver, where the camera maintains its boresight aligned with the docking port axis while the Space Rider mockup rotates around its own Z-axis. Furthermore, to simulate off-centered conditions, the robotic arm translated the camera within the XY plane of camera frame. This configuration is particularly relevant for scenarios where the approach axis is known (i.e., aligned with the camera boresight), but the relative roll angle is uncertain. The experiment evaluates the network's robustness in handling off-centered targets and varying in-plane rotations within the field of view. Figure 12b shows the robot-arm trajectory used during dataset acquisition. The initial plan was a square inspection path; however, due to workspace constraints, the trajectory was replanned, resulting in the non-square pattern shown. Total images captured: 435.
- **Dataset 3 – Observation Phase:** The third dataset simulates a general observation phase. In this case, the camera remains fixed in space while the robotic arm manipulates the mockup through two different rotational sequences around its joints. Figure 12c depicts the commanded trajectory of the Space Rider mockup during the acquisition campaign, designed to reproduce the relative motion profiles expected in the SROC inspection phase. This scenario is intended to assess the CNN's ability to handle arbitrary changes in attitude without translational motion, thus simulating complex pose variations observed during early mission observation phases. Total images captured: 800.

Illumination conditions were varied during the first two maneuvers to emulate in-orbit lighting changes. The Sun direction, parameterized by azimuth and elevation in the LVLH frame, is reported in Table 1. Due to kinematic constraints of the UR5 robotic arm used to orient the lamp, the full envelope could not be reproduced. The GRALS campaign therefore adopted a representative, physically feasible subset of Sun angles, namely $Az \in [15, 30]^\circ$ and $El \in [0, 15]^\circ$, which preserves realistic variability while remaining compatible with the facility setup.

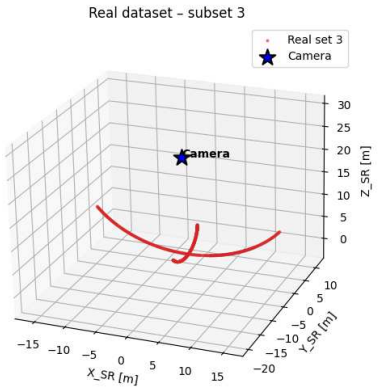
The image acquisition was conducted using black curtains to minimize ambient light interference and isolate the target scene. However, as shown in Fig. 13, certain



(a) Relative position in the real Dataset 1 - V-bar Approach



(b) Relative position in the real Dataset 2 – Top Inspection



(c) Relative position in the real Dataset 3 – Observation

Fig. 12: Real dataset relative position during the acquisition process

non-target elements-such as the robotic arm, the supporting rail on the left, and additional laboratory equipment-remain visible in the background. These could not be fully obscured due to safety constraints and facility limitations.

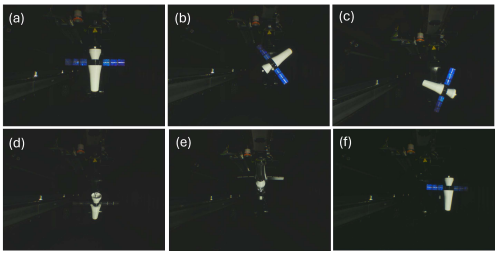


Fig. 13: (a) image from V-bar approach dataset; (b) and (c) from Top Inspection; (d), (e) and (f) from Observation phase

5.2 Camera Calibration

To accurately model the imaging geometry, the camera underwent an **intrinsic calibration procedure**. This process estimates the internal camera parameters, including the intrinsic matrix and distortion coefficients, which may deviate from nominal values due to lens imperfections, manufacturing tolerances, and sensor alignment errors. The calibration was performed using a standard checkerboard-based method. A sequence of images of a planar checkerboard pattern was captured from various viewpoints and orientations. These images were then processed using the OpenCV library [61] to extract corner points and solve for the camera parameters. This yielded the intrinsic camera matrix K_{calib} and distortion coefficients, including radial terms (k_1, k_2, k_3) and tangential terms (p_1, p_2).

During synthetic dataset generation, lens distortions were not modeled, the camera was assumed to follow an ideal pinhole model with nominal intrinsic parameters K_{train}. This discrepancy introduces a domain mismatch that can degrade CNN inference performance when applied to real images.

To bridge this gap, real images can be preprocessed to remove distortion and adjust the projection geometry to match that of the synthetic training domain. The OpenCV library provides tools to perform both distortion correction (in Python *cv.undistort* https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html) and geometric warping by providing the transformation matrix as input (*cv.warpPerspective* https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87).

Assuming a 3D point X is projected onto the image plane by both the training and calibrated cameras using the same extrinsic parameters $[R|t]$, the resulting image points are:

$$p_{\text{train}} = K_{\text{train}}[R|t]X, \quad p_{\text{calib}} = K_{\text{calib}}[R|t]X$$

This leads to the homography relation:

$$p_{\text{train}} = H\, p_{\text{calib}}, \quad \text{where } H = K_{\text{train}} K_{\text{calib}}^{-1}$$

The transformation matrix $H \in \mathbb{R}^{3\times3}$ allows remapping of the calibrated images into the geometry of the training setup, thereby improving consistency and reducing projection errors during CNN inference.

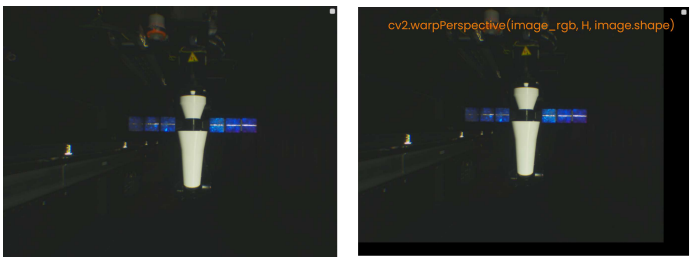


Fig. 14: Image warped after the transformation

6 Results

This section presents the results obtained with different training strategies, with the goal of maximizing pose-estimation performance during the final-approach phase. Two approaches are compared. In the first, the network is pretrained on the observation dataset-using the hyperparameters selected through the corresponding HPO-and subsequently fine tuned on the final approach dataset. This strategy aims to first learn general, domain-robust features from a broad set of viewing conditions, and then specialize the model for the task-critical docking geometry.

Training exclusively on the final-approach dataset was also investigated; however, the resulting accuracy was consistently lower and is therefore not reported. This outcome further supports the role of a more general pretraining set in improving robustness and mitigating the synthetic-to-real domain gap.

In the second, the network is trained from scratch on a mixed dataset combining observation and final-approach samples. In the following sections, we present the inference software/hardware setup, summarize the hyperparameter optimization outcomes, and describe in detail the training strategies considered.

6.1 Transition from PyTorch to ONNX Runtime

The initial development and training of the neural network were carried out using PyTorch, which offers flexibility and a rich ecosystem for deep learning research. However, direct deployment of PyTorch models in a C++ environment introduced significant integration challenges due to limitations and compatibility issues with LibTorch within the existing software architecture.

To overcome these constraints, the ONNX Runtime [62] was adopted as an inference engine. ONNX (Open Neural Network Exchange) provides a standardized format for representing deep learning models, enabling interoperability across different platforms and frameworks. ONNX Runtime is a cross-platform, high-performance library that supports both C++ and Python, making it well-suited for deployment on embedded systems.

The following workflow was implemented to enable the transition: (a) **Model Export:** The trained PyTorch model was exported to the ONNX format using the `torch.onnx.export()` function, preserving the computational graph and network

parameters. (b) **Inference Integration:** ONNX Runtime was selected as the inference engine within the C++ environment due to its lightweight footprint, low latency, and broad hardware support. (c) **Hardware Acceleration:** To exploit the parallel processing capabilities of the NVIDIA Jetson Orin Nano, ONNX Runtime was configured with the CUDA Execution Provider, enabling GPU-accelerated inference. This pipeline enabled seamless integration of the trained CNN into the flight software stack while achieving high inference performance and maintaining compatibility with the mission's embedded computing constraints.

6.2 Hardware

To facilitate onboard execution of the spacecraft pose estimation model, the NVIDIA Jetson Orin Nano was selected as the Payload Processing Unit (PPU). This choice was driven by its balance of computational performance and power efficiency, making it suitable for CubeSat applications. The Jetson Orin Nano Developer Kit - utilized for testing - features are described in Table 3. These specifications provide ample resources for real-time inference tasks in space-constrained environments.

Table 3: Jetson Orin nano developer kit specs [63]

Spec	Value
GPU	Up to 1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
CPU	6-core Arm Cortex-A78AE v8.2 64-bit CPU
Memory	8 GB LPDDR5
AI Performance	Up to 67 TOPS (INT8) at 25W
Power Consumption	Configurable between 7W and 25W

The imaging system is built around a **wide field-of-view (WFOV), commercial off-the-shelf (COTS) camera**, equipped with a $3\,mm$ focal length lens and an aperture of $f/2$. The sensor is a COTS CMOS device with a maximum resolution of 2048×1536 pixels, integrated onto a Tyvak-custom printed circuit board assembly (PCBA). The exposure time was set to 40 ms after several adjustments to achieve a balanced illumination of the target.
The performance of the CNN deployed on the flight-representative hardware, using compound scaling $\phi = 2$ and an input resolution of 512×384, is reported in Table 4. The preprocessing time includes the operations applied to the input image prior to inference (e.g., resizing and warping). The full assembly was housed in a 3D-printed PLA enclosure, designed to be mounted on the robotic arm used during dataset acquisition.

6.2.1 Camera Calibration

The camera calibration procedure revealed a discrepancy from the nominal values used in synthetic dataset generation. Assuming a pixel size of $l_{px} = 2.2\,\mu$m, the intrinsic matrices are:

Table 4: Hardware performance metrics and input settings.

Spec	Value
Storage (ONNX file)	36.218 MB
Power (Jetson)	7 W
Preprocessing time	9.64 ms (mean)
Inference time	89.8 ms (mean), 95.4 ms (max), 88.99 ms (min)
Memory usage during inference	4075 MB (mean), 5511 MB (max)

$$K_{\text{train}} = \begin{bmatrix} 3/l_{px} & 0 & 1024 \\ 0 & 3/l_{px} & 768 \\ 0 & 0 & 1 \end{bmatrix}, \quad K_{\text{calib}} = \begin{bmatrix} 3.12/l_{px} & 0 & 1456 \\ 0 & 3.11/l_{px} & 828 \\ 0 & 0 & 1 \end{bmatrix}$$

The calibration process also yielded the following distortion coefficients: **radial** $k_1 = -0.439$, $k_2 = 0.253$, $k_3 = -0.0909$ and **tangential** $p_1 = 3.39 \times 10^{-4}$, $p_2 = 3.88 \times 10^{-4}$.

6.3 Hyperparameter optimization results

To identify optimal hyperparameters, which depend significantly on both the specific dataset and the CNN architecture under consideration, a constrained hyperparameter search was performed using the Tree-structured Parzen Estimator (TPE) algorithm. The optimization process was executed on a single NVIDIA RTX 3090 GPU, utilizing the observation dataset 2.2.1 both for training and validation. This dataset comprises a total of 24,397 images, with 80% allocated to the training set and 20% reserved for validation.

The image resolution was reduced to 512×384 pixels, preserving the original aspect ratio (Table 1). This resolution balances accuracy with onboard computational constraints, avoiding excessive memory load during inference or potential in-orbit fine-tuning. The compound scaling coefficient $\phi = 2$ (EfficientDet-D2) yields the parameter counts as in Table 5.

Table 5: Number of parameters for the selected configuration ($\phi = 2$).

Module	Parameters
Backbone (EfficientDet-D2)	8,058,362
Head 1 (heatmap)	135,190
Head 2 (EfficientPose)	276,008
Head 3 (segmentation)	134,065

The following hyperparameters were kept constant during the search: (a) the learning rate decay, the following polynomial decay was chosen $lr(t) = lr_0 \left(1 - \frac{t}{T}\right)^p$, where lr_0 is the initial learning rate, t is the current epoch, T is the total number of epochs, and p is the decay exponent; (b) batch size = 16; (c) optimizer set to AdamW [64]. 453 The hyperparameters explored are informed by (i) prior results in [27], (ii) standard

practice for pretrained EfficientNet/Det backbones, and (iii) limits of the hardware setup used for the HPO. Table 6 summarizes the search space and justification.

Table 6: HPO variables, ranges, and rationale.

Hyperparameter (range)	**Rationale**
Initial LR $\in [10^{-5}, 10^{-3}]$	Covers stable region for ImageNet-pretrained backbones with our batch size; initial screening trials showed divergence $> 10^{-2}$ and slow learning $< 10^{-5}$.
Warm-up epochs $\in [0, 5]$	Prevents early instability.
Decay exponent (p) $\in [0.5, 3]$	Spans slow-to-aggressive decay; initial screening trials found exponents outside this band underfit (too slow) or overfit late (too fast).
Total epochs (T) $\in [25, 40]$	Upper bound from observed validation reversal > 40 (overfitting); lower bound avoids underfitted model.

The optimization procedure was executed over 50 trials, with the first 10 trials using random sampling to support exploration before the TPE algorithm took over. The ASHA pruner was configured to defer trial pruning until after epoch 25, ensuring that potentially effective hyperparameter combinations had a sufficient number of epochs to demonstrate their potential. Validation was performed every 5 epochs to monitor performance and retain intermediate evaluation results. Note that all the trials were performed applying the dataset augmentation techniques described in 2.2.3 during the training procedure.

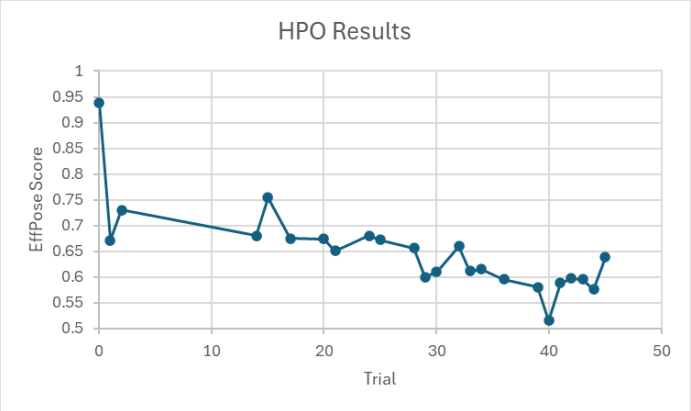


Fig. 15: Optimization history plot

Fig. 15 depicts the optimization history plot showing a clear improvement trend, with significant performance gains in the early trials, indicating effective initial

exploration. The best objective value is achieved at trial 40, after which further improvements become marginal, suggesting that Optuna has effectively identified near-optimal hyperparameters. The best set of Hyperparameters found is reported in 7.

Table 7: Best hyperparameter optimization trial

Trial	EffPose	End epoch	lr	warmup	Poly power
40	0.5157	39	5.5e-4	0	0.646

Optuna provides a feature importance analysis based on Functional ANOVA (fANOVA) decomposition [65], which quantifies how much each hyperparameter contributes to the variance in the objective value across all trials. The importances are represented by non-negative point numbers, where higher values mean that the parameters are more important. The results show that the initial learning rate has the highest impact (0.59), followed by the number of epochs (0.35), the polynomial decay exponent (*Poly power*) (0.03), and the number of warm-up epochs (0.02).

6.3.1 Analysis of the Trials

This section compares a subset of completed training trials, including a baseline configuration and models obtained through hyperparameter optimization. The hyperparameters used in each trial are summarized in Table 8. Although all completed trials were evaluated, only four representative networks are reported here. Network A corresponds to the initial baseline, built from standard, manually selected hyperparameters, whereas Networks B–D are trials generated by the HPO procedure.

Table 8: Hyperparameters comparison. Network A represents the first guess of the HPO proced, while Networks B–D correspond to trials from the HPO procedure, with D being the best result.

Network	End epoch	lr	warmup	Poly power
A	33	2e-4	1	1
B	26	4.4e-4	1	0.608
C	37	6e-4	2	0.821
D	39	5.5e-4	0	0.646

The performance of the EfficientPose and Heatmap heads is evaluated in terms of normalized translation error E_{TN} and rotation error θ_{err}, defined as 5.

The translation error for the Heatmap head (E_{TN_keypts}) was calculated by solving the PnP problem. Note that θ_{err} obtained via PnP-based attitude estimation was omitted, since its accuracy proved substantially inferior to that of the EfficientPose head. Each network (A–D) was pretrained on the *Observation Dataset* and then fine-tuned

on the *Final Approach Dataset*-composed of 19,172 images-using three strategies, with the fine-tuning phase specifically aimed at enhancing performance on this task-critical maneuver:

- **Strategy 1:** Fine-tuning for 5 epochs with fixed learning rate 1×10^{-5}.
- **Strategy 2:** Same as Strategy 1, but freezing the backbone to avoid weight updates.
- **Strategy 3:** Same as Strategy 2, but with higher learning rate 1×10^{-4}.

First, performance was evaluated on the synthetic observation test set and on the real observation dataset acquired at GRALS (Dataset 3 – Observation Phase, Section 5.1), without any fine-tuning. The results are reported in Table 9.

Table 9: Pose estimation results on the synthetic observation test set and the GRALS observation dataset (Dataset 3). E_{TN} is the normalized translation error, θ_{err} is the orientation error (deg), and **EffScore** is the overall efficiency score (lower is better). Model D is the best HPO trial

Domain	Model	E_{TN}	θ_{err} [deg]	EffScore
Synthetic	A	0.048	37.2	0.7
	B	0.042	21.43	0.42
	C	0.031	11.59	0.23
	D	0.034	10.45	**0.216**
Testbed	A	0.12	24.14	0.54
	B	0.11	17.24	0.42
	C	0.08	24.45	0.51
	D	0.043	14.25	**0.29**

Overall, the performance improves consistently from Model A to Model D, indicating the benefit of the HPO process. In both domains, Model D achieves the lowest EffScore, confirming it as the best configuration among the tested trials. As expected, errors increase on the real testbed data compared to the synthetic set, highlighting the remaining domain gap.

Performances were also assessed on the testbed V-bar and Top Inspection datasets acquired at GRALS (Section 5.1) applying fine-tuning strategies explained above. The mean results for both datasets are reported in Table 10.

Fine-tuned models were evaluated on the observation datasets but results were not reported because fine-tuning on the final approach data led to a marked degradation in performance during observation phase conditions.

Without fine-tuning, Network C achieves the lowest EffScore and the best combined metric (H+E), indicating that the corresponding HPO trial generalizes better to the GRALS fine-tuning datasets than the other configurations. After fine-tuning, all strategies provide a substantial improvement with respect to the baseline case (no

Table 10: Comparison of training and fine-tuning strategies across different networks. Best values of H+E are highlighted in bold.

Strategy	Model	E_{TN_keypts}	E_{TN}	θ_{err} [deg]	EffScore	H+E
No Fine-Tuning	A	0.021	0.043	24.04	0.462	0.483
	B	0.024	0.04	14.97	0.302	0.326
	C	0.019	0.029	13.77	0.269	0.288
	D	0.02	0.026	14.76	0.284	0.304
Strategy 1	A	0.024	0.034	4.65	0.115	0.139
	B	0.021	0.030	4.03	0.100	0.121
	C	0.018	0.026	3.58	0.089	0.107
	D	0.018	0.036	4.07	0.107	0.125
Strategy 2	A	0.022	0.028	4.67	0.110	0.132
	B	0.023	0.027	4.29	0.101	0.124
	C	0.019	0.025	3.83	0.092	0.111
	D	0.020	0.033	4.49	0.111	0.131
Strategy 3	A	0.023	0.033	4.16	0.10	0.128
	B	0.023	0.021	4.00	0.09	0.114
	C	0.020	0.021	3.60	0.08	**0.104**
	D	0.020	0.022	3.95	0.09	0.111

fine-tuning), especially in terms of attitude error, which drops from $\sim$ 14–24 deg to $\sim$ 3.6–4.7 deg.

Among the investigated approaches, Strategy 3 yields the best overall performance, with the minimum H+E obtained by Network C. This suggests that, for the considered testbed data, allowing limited adaptation of the backbone (frozen backbone with higher learning rate on the heads) is beneficial, likely because it preserves the features learned during pretraining while still enabling fast specialization to the final approach visual conditions. A relevant outcome is that the best-performing model on the GRALS final-approach datasets is Network C rather than the best HPO trial, Network D. In contrast, in the real observation dataset that spans a wider set of viewpoints, attitudes, and illumination conditions, Network D achieves clearly superior performance. Overall, Network D—the best trial from the HPO—offers stronger generalization, while Network C—the second-best HPO trial—can be slightly more accurate under highly specialized conditions. Both configurations originate from the HPO process and represent strong choices compared to the initial baseline.

6.4 Double Dataset Training

This section evaluates an alternative strategy that avoids fine-tuning. The network is trained end-to-end on a composite dataset merging the observation and final-approach sets (38,688 images). This phase does not include a dedicated HPO; instead, we reuse the same polynomial learning-rate schedule and the hyperparameter configurations of Networks C and D, which provided the best performance in the previous analysis. To ensure a fairer comparison with the fine-tuning approach, the number of epochs is adjusted so that the total number of gradient updates remains comparable despite

the larger training set. Unless otherwise stated, the end epoch is set to 20.
Table 11 reports the results for Networks C and D, evaluated on the GRALS observation dataset and on the final-approach datasets. Final-approach values are reported as the mean across the V-bar approach and Top Inspection sequences.

Table 11: Training results with a double dataset strategy. The Network was tested on the testbed datasets.

Model	Dataset	E_{TN_keypts}	E_{TN}	θ_{err} [deg]	**EffScore**	**H+E**
Model C	Obs.	0.1	0.04	19.19	0.375	0.482
	FA	0.019	0.024	3.58	0.086	0.106
Model D	Obs.	0.068	0.069	33.14	0.648	0.716
	FA	0.021	0.023	3.44	0.083	0.104

Table 11 shows that training on the composite dataset yields a more balanced behavior across maneuvers. On the final-approach testbed set, the performance (H+E = 0.106) is close to the best fine-tuning result (Strategy 3, Model C: H+E = 0.104 in Table 10). However, unlike the fine-tuned models, the composite training maintains usable accuracy on the observation testbed set, indicating better retention of generalization. Overall, the double-dataset strategy improves maneuver-to-maneuver flexibility, even if it does not achieve the absolute best score on the final-approach scenario.

6.4.1 Influence of the Number of Training Epochs

To study the effect of training duration, we repeat the double-dataset training while keeping hyperparameters fixed and varying the number of epochs. Four settings are considered: 15, 20, 25, and 30 epochs. Results are summarized in Table 12.

Increasing the number of epochs has a strong impact on the observation-phase performance, especially for Network D, whose H+E improves markedly from 0.730 (15 epochs) to 0.349 (30 epochs). This suggests that longer training helps consolidate features needed for the more variable observation scenario. In contrast, final-approach performance is comparatively stable across epochs (H+E typically ∼0.10–0.11), with one notable exception: Network C at 30 epochs achieves the best final-approach score (H+E = 0.092), driven primarily by improved orientation accuracy. Overall, the epoch sweep indicates a trade-off: longer training tends to benefit observation generalization, while final-approach gains are smaller and may depend on the specific hyperparameter configuration.

6.5 Detailed distribution analysis

The following analysis compares training strategies using distribution-based diagnostics rather than only mean values. Translation errors are examined through axis-wise

Table 12: Results obtained with the double-dataset training strategy for different total numbers of epochs. Networks are evaluated on the GRALS testbed datasets.

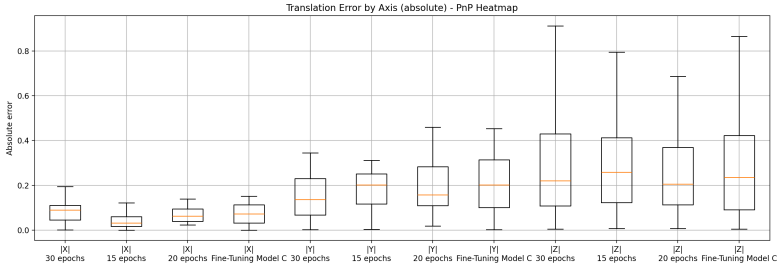
Epochs	Model	Dataset	E_{TN_keypts}	E_{TN}	θ_{err} [deg]	EffScore	H+E
15	Network C	Obs	0.075	0.074	35.26	0.689	0.765
		FA	0.018	0.021	3.46	0.082	0.100
	Network D	Obs	0.048	0.035	37.08	0.682	0.73
		FA	0.018	0.029	3.68	0.093	0.112
20	Network C	Obs	0.1	0.04	19.19	0.375	0.482
		FA	0.019	0.024	3.58	0.087	0.106
	Network D	Obs	0.068	0.069	33.14	0.648	0.716
		FA	0.021	0.023	3.44	0.083	0.104
25	Network C	Obs	0.086	0.048	18.55	0.372	0.458
		FA	0.019	0.027	3.59	0.089	0.109
	Network D	Obs	0.077	0.053	17.29	0.355	0.431
		FA	0.020	0.025	3.52	0.087	0.107
30	Network C	Obs	0.098	0.039	17.05	0.344	0.442
		FA	0.019	0.023	2.85	0.089	**0.092**
	Network D	Obs	0.074	0.037	13.68	0.275	**0.349**
		FA	0.017	0.023	3.71	0.088	0.105

boxplots (to highlight mean values and outliers) and through CDFs (Cumulative Distribution Functions) of the normalized translation error, while rotation performance is assessed with CDFs of the attitude error.
The boxplot of absolute translation error ($|X|, |Y|, |Z|$) summarizes how the position error is distributed along each axis. The median line indicates the typical error, the box shows the central 50% of samples, and whiskers/outliers reveal occasional large deviations. This plot makes it easy to identify whether the performance is uniform across axes or dominated by one component (e.g., a larger spread on $|Z|$ indicates less consistent estimation along that direction). The CDF provides a reliability-oriented view of the pose error. For any threshold t, the CDF value represents the fraction of samples with error $\leq t$. This allows reporting statements such as "95% of frames achieve error below t" and makes error tails visible, which is important for rendezvous scenarios where worst-case behavior matters.
This section analyzes the full error distributions for four training configurations: double-dataset training (15, 20, and 30 epochs) and the fine-tuned Model C (FT-C). The results below refer to the V-bar final-approach testbed dataset and the Top Inspection dataset, because mission requirements target top-view inspection.

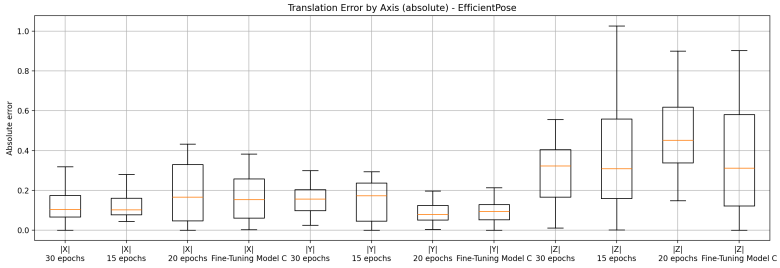
6.5.1 Axis-wise translation error (boxplots).

Figure 16 shows that the translation error has different values for the three components: the $|Z|$ component (typically aligned with the camera line-of-sight) exhibits the largest median and dispersion, whereas $|X|$ and $|Y|$ are generally tighter. For the EfficientPose head, 20 epochs (double-dataset) yields the largest Z median and

interquartile range, suggesting reduced robustness (more dispersion and more large-error cases). The FT-C and the 15-epoch double-dataset model yield lower typical $|Z|$ errors than the 20- and 30-epoch trainings, but they show larger spread. Conversely, the 30 epoch training presents an higher mean value but with a smaller box and less outliers. For the PnP heatmap estimate, the medians remain close across strategies, but $|Z|$ still dominates the variability and the FT-C shows the largest extreme outliers.



(a) PnP (heatmap) translation error on the V-bar dataset

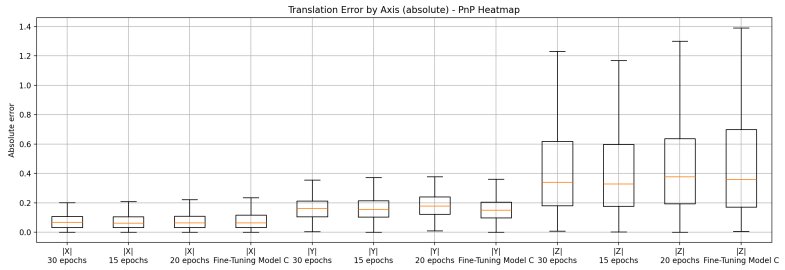


(b) EfficientPose translation error on the V-bar dataset

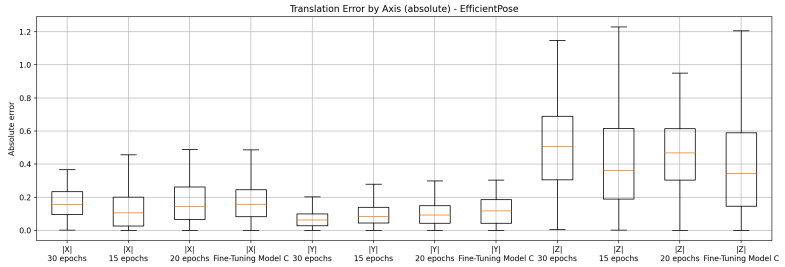
Fig. 16: Boxplots of absolute translation error components ($|X|, |Y|, |Z|$) evaluated on the V-bar final-approach testbed dataset.

In the Top Inspection testbed dataset (Figure 17), the heatmap head yields very similar error distributions across the three double-dataset trainings and the FT-C. Errors are consistently smallest along $|X|$, larger along $|Y|$, and dominated by $|Z|$, which also exhibits the widest spread and most outliers. This indicates that the PnP-from-heatmap translation is largely insensitive to the training strategy for this maneuver, with the depth/range axis remaining the main limiting component (with a slight advantage for the 15-epoch training).

For the EfficientPose head, the training strategy has a clearer impact. The $|Z|$ component still dominates, but the 30-epoch double-dataset training shows a higher median and a higher lower bound of the box, whereas 15 epochs and the FT-C reduce the typical $|Z|$ error at the cost of a broader dispersion.



(a) Boxplots for the heatmap head on the Top Inspection testbed dataset.



(b) Boxplots for the EfficientPose head on the Top Inspection testbed dataset.

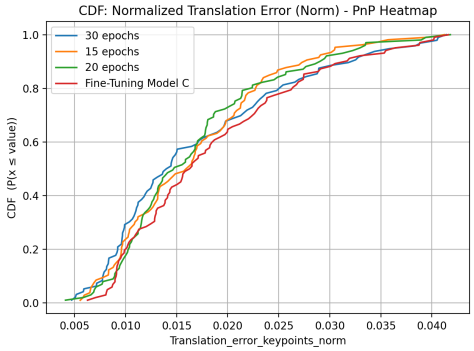
Fig. 17: Boxplots of absolute translation error components ($|X|, |Y|, |Z|$) evaluated on the Top Inspection testbed dataset.

6.5.2 Normalized translation and rotation errors (CDFs).

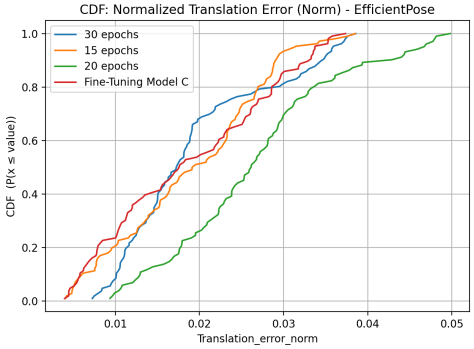
Figure 18 reports the CDFs on the V-bar dataset. For the PnP heatmap translation ($E_{TN_{\text{keypts}}}$), the curves largely overlap, indicating limited sensitivity to the training strategy; only small differences appear in the mid-to-high percentiles. For EfficientPose translation, the 20-epoch model is generally right-shifted (worse), while 15 epochs and the FT-C achieve lower errors for a larger fraction of samples. Rotation CDFs show the clearest separation: 20 epochs is right-shifted, 30 epochs is left-shifted (lower errors across most percentiles), but for higher CDF value-more than 0.6-15 epochs and FT-C remain close and more competitive.

Figure 19a and Figure 19b show the CDFs for the *Top Inspection* dataset. For the PnP translation from the heatmap head (Figure 19a), the curves nearly overlap, indicating that keypoint-based PnP is only marginally influenced by the training strategy in this maneuver.

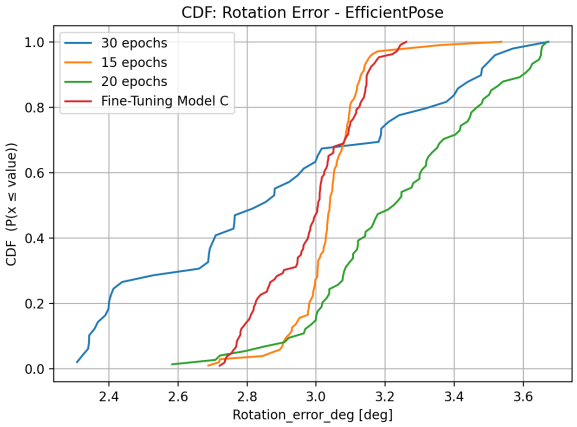
Conversely, the EfficientPose translation CDF (Figure 19b) shows clearer separation: the 15-epoch double-dataset model is consistently left-shifted, achieving lower translation errors for a larger fraction of samples. Fine-tuned Model C also performs well, outperforming the 20- and 30-epoch models. The strongest difference appears in rotation (Figure 19c), where the 30-epoch double-dataset model yields the most



(a) CDF of $E_{TN_{\text{keypts}}}$ (PnP heatmap) - V-bar



(b) CDF of E_{TN} (EfficientPose translation) - V-bar



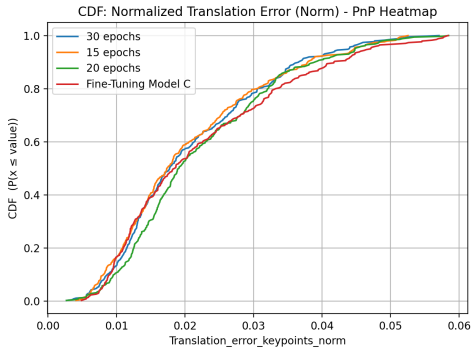
(c) CDF of θ_{err} (EfficientPose rotation) - V-bar

Fig. 18: CDFs of translation and rotation errors on the V-bar final-approach testbed dataset.

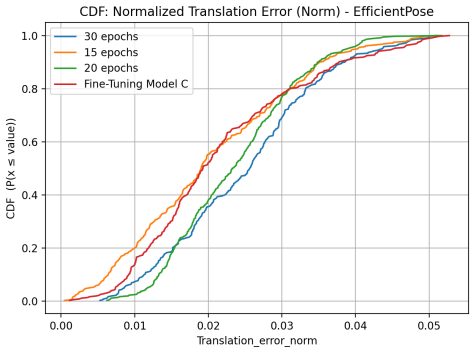
left-shifted curve, suggesting that performance on this dataset is primarily driven by improvements in the rotation head.

6.5.3 Discussion

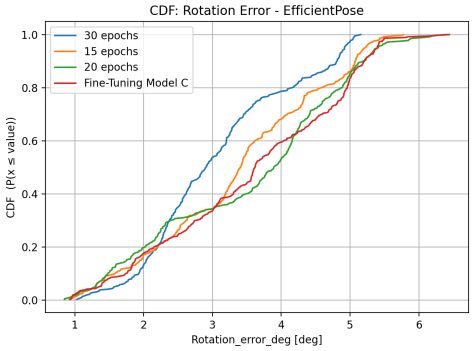
Overall, the Heatmap/PnP head is relatively insensitive to the training strategy: the CDFs are largely overlapping, with about 90% of samples achieving E_{TN_keypts} below ≈ 0.03 in V-bar and ≈ 0.035 in Top Inspection. In contrast, EfficientPose is more affected by the training recipe. For translation (E_{TN}), the 15-epoch double-dataset training yields the most reliable high-percentile behavior (lighter tail), even when the mean slightly favors the 30-epoch setting. For rotation, the best strategy depends on the maneuver: longer training (30 epochs) can improve attitude accuracy in the



(a) CDF of $E_{TN_{\text{keypts}}}$ (PnP heatmap) - Top Inspection



(b) CDF of E_{TN} (EfficientPose translation) - Top Inspection



(c) CDF of θ_{err} (EfficientPose rotation) - Top Inspection dataset

Fig. 19: CDFs of translation and rotation errors on the Top Inspection testbed dataset.

Top-Inspection case, whereas in V-bar the 15-epoch / fine-tuned model remains more consistent across percentiles.

7 Conclusions

This work presented a complete solution for deploying a flight-ready visual navigation system based on Convolutional Neural Networks (CNNs) through the application on the ESA's Space Rider Observer CubeSat (SROC) mission. The study focused on pose estimation using a single monocular image acquired by a wide field-of-view (WFOV) COTS camera, targeting autonomous proximity operations and docking maneuvers.

The proposed pipeline includes the generation of a synthetic training dataset using a Blender based rendering engine, accounting for realistic illumination and orbital

conditions. Robust data augmentation techniques, such as style and geometric transformations, have been employed to reduce the domain gap between synthetic and real imagery. A key contribution of this work is the integration of automated Hyperparameter Optimization (HPO), leveraging the Optuna framework to fine-tune model performance using a domain-randomized training set. The CNN model was trained in PyTorch and exported to ONNX format for deployment on the Payload Processing Unit (PPU), ensuring platform-agnostic inference suitable for real-time onboard applications.

To validate the approach, a high-fidelity real world dataset was acquired at ESA's GRALS facility. The image data was corrected for lens distortions using camera calibration procedures based on checkerboard images, ensuring consistency between synthetic and real-world projections. This correction proved essential for minimizing projection errors and enhancing inference reliability.

Several training strategies were explored, with particular emphasis on improving performance during the final approach phase, where accuracy is critical. The results demonstrated that a mixed training dataset combining both observation and final approach maneuvers yielded the most balanced performance across translation and rotation metrics, with strong generalization across multiple trajectories. Strategies involving fine-tuning with a frozen backbone showed robustness in specific cases but lacked flexibility in broader conditions.

Experimental results revealed that the EfficientPose head consistently provided better rotation accuracy, particularly in challenging lighting conditions. However, the Heatmap head offered more stable translation performance.

Notably, the observation maneuver presented the greatest challenge for both heads due to wider viewpoint diversity, increased feature variability, and the presence of clutter within the field of view, such as robotic arms and rails, that were difficult to mask due to safety constraints. Similar clutter was present even in the final approach dataset; however, training the CNN with a dataset specifically tailored to this maneuver significantly improved its robustness in cluttered environments. This confirms the importance of task-specific data in reducing the effects of environmental noise and visual ambiguity.

Results could be further improved by adopting stronger augmentation schemes, such as Random Convolutions, and more specialized synthetic-to-real techniques. In addition to Albumentations and the current style randomization, a promising direction is *GAN-based style transfer* (e.g., CycleGAN) to map Blender renders toward the testbed/flight visual appearance while preserving scene geometry, thereby reducing the domain gap directly at the image level.

The GRALS testbed dataset is also not fully representative of flight imagery: the scene contains clutter and non-flight elements (e.g., robotic arm visibility and other objects), and some maneuvers-especially the observation phase-could be executed with improved coverage and repeatability. A cleaner and more mission-representative acquisition would enable a more reliable assessment of performance.

In conclusion, this study confirms that CNN-based monocular pose estimation, when coupled with appropriate training strategies, dataset design, and hardware-aware deployment can provide reliable, real-time visual navigation performance for CubeSat missions.

Acknowledgments

This study is part of a PhD program in Aerospace Engineering at Politecnico di Torino. The program is part of the project PNRR-NGEU, which has received funding from the MUR and Tyvak International s.r.l. - DM 352/2022. The GRALS experiments were supported by the Discovery Programme through the EISI Agreement – Visiting Researcher 4000146000. The first author would like to thank Massimo Casasco, Irene Huertas Garcia, Joris Belhadj, Sebastian Rimmer, and Ane Morkemo for their invaluable support during the test campaign at ESTEC.



References

[1] Nocerino, A., Opromolla, R., Fasano, G., Grassi, M.: Lidar adaptive parameters selection and target pointing control for close-proximity space operations. Aerospace Science and Technology **155**, 109646 (2024). https://doi.org/10.1016/j.ast.2024.109646

[2] Sun, D., Hu, L., Duan, H., Pei, H.: Relative Pose Estimation of Non-Cooperative Space Targets Using a TOF Camera. Remote Sensing **14**(23) (2022). https://doi.org/10.3390/rs14236100

[3] Pirat, C., Ankersen, F., Walker, R., Gass, V.: Vision based navigation for autonomous cooperative docking of cubesats. Acta Astronautica **146**, 418–434 (2018). https://doi.org/10.1016/j.actaastro.2018.01.059

[4] Vela, C., Fasano, G., Opromolla, R.: Pose determination of passively cooperative spacecraft in close proximity using a monocular camera and aruco markers. Acta Astronautica **201**, 22–38 (2022). https://doi.org/10.1016/j.actaastro.2022.08.024

[5] Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate o(n) solution to the pnp problem. International Journal of Computer Vision **81** (2009). https://doi.org/10.1007/s11263-008-0152-6

[6] Tareen, S.A.K., Saleem, Z.: A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–10 (2018). https://doi.org/10.1109/ICOMET.2018.8346440

[7] Roscoe, C.W.T., Westphal, J.J., Mosleh, E.: Overview and gnc design of the cubesat proximity operations demonstration (cpod) mission. Acta Astronautica **153**, 410–421 (2018). https://doi.org/10.1016/j.actaastro.2018.03.033

[8] Forshaw, J.L., Aglietti, G.S., Navarathinam, N., Kadhem, H., Salmon, T., Pisseloup, A., Joffre, E., Chabot, T., Retat, I., Axthelm, R., Barraclough, S., Ratcliffe, A., Bernal, C., Chaumette, F., Pollini, A., Steyn, W.H.: Removedebris: An in-orbit active debris removal demonstration mission. Acta Astronautica **127**, 448–463 (2016) https://doi.org/10.1016/j.actaastro.2016.06.018

[9] Banker, B., Askew, S.: Seeker 1.0: Prototype robotic free flying inspector mission overview. In: 33rd Annual AIAA/USU Conference on Small Satellites (2019). https://digitalcommons.usu.edu/smallsat/2019/all2019/151/

[10] Sullivan, J., Gambone, E., Kirven, T., Pedrotty, S., Wood, B.: Rapid development of the seeker free-flying inspector guidance, navigation, and control system. In: AAS Guidance and Control Conference (2019)

[11] Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In: Robotics: Science and Systems (2018). https://doi.org/10.48550/arXiv.1711.00199. http://www.roboticsproceedings.org/rss14/p19.html

[12] Sharma, S., D'Amico, S.: Pose Estimation for Non-Cooperative Rendezvous Using Neural Networks. https://doi.org/10.1109/AERO.2018.8396425 (2019). https://arxiv.org/abs/1906.09868

[13] Park, T.H., Sharma, S., D'Amico, S.: Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft. https://doi.org/10.48550/arXiv.1909.00392 (2019). https://arxiv.org/abs/1909.00392

[14] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474 (2019). https://arxiv.org/abs/1801.04381

[15] Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement. https://doi.org/10.48550/arXiv.1804.02767 (2018). https://arxiv.org/abs/1804.02767

[16] Park, T.H., D'Amico, S.: Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap. Advances in Space Research **73**(11), 5726–5740 (2024). https://doi.org/10.1016/j.asr.2023.03.036

[17] Tan, M., Pang, R., Le, Q.V.: EfficientDet: Scalable and Efficient Object Detection. https://doi.org/10.48550/arXiv.1911.09070 (2020). https://arxiv.org/abs/1911.09070

[18] Chen, S., Yang, W., Wang, W., Mai, J., Liang, J., Zhang, X.: Spacecraft homography pose estimation with single-stage deep convolutional neural network. Sensors

24(6) (2024) https://doi.org/10.3390/s24061828

[19] LIU, Y., ZHOU, R., DU, D., CAO, S., QI, N.: Feature-aided pose estimation approach based on variational auto-encoder structure for spacecrafts. Chinese Journal of Aeronautics **37**(8), 329–341 (2024) https://doi.org/10.1016/j.cja.2024.03.017

[20] Gui, Y., Qi, Y., Xiao, X., Lin, B., Cui, H., Huang, X.: Sptn: Transformer-based spacecraft pose estimation network for space objects tracking. Astrodynamics **9**(5), 713–725 (2025) https://doi.org/10.1007/s42064-024-0260-2

[21] Park, T.H., D'Amico, S.: Bridging the Domain Gap for Flight-Ready Spaceborne Vision. arXiv (2024). https://doi.org/10.48550/ARXIV.2409.11661

[22] Yang, H., Xiao, X., Yao, M., Xiong, Y., Cui, H., Fu, Y.: Pvspe: A pyramid vision multitask transformer network for spacecraft pose estimation. Advances in Space Research **74**(3), 1327–1342 (2024) https://doi.org/10.1016/j.asr.2024.05.011

[23] Ulmer, M., Durner, M., Sundermeyer, M., Stoiber, M., Triebel, R.: 6D Object Pose Estimation from Approximate 3D Models for Orbital Robotics. arXiv (2023). https://doi.org/10.48550/ARXIV.2303.13241

[24] Yu, T., Zhu, H.: Hyper-Parameter Optimization: A Review of Algorithms and Applications. CoRR **abs/2003.05689** (2020) 2003.05689. https://doi.org/10.48550/arXiv.2003.05689

[25] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, ??? (2016). http://www.deeplearningbook.org

[26] Feurer, M., Hutter, F.: Hyperparameter optimization. In: Automated Machine Learning, pp. 3–33. Springer, ??? (2019). https://doi.org/10.1007/978-3-030-05318-5_1

[27] Lovaglio, L., D'Ortona, A., Stesina, F., Corpino, S.: Cnn-based visual navigation: Optimization strategies for monocular pose estimation in proximity operations, pp. 1470–1480 (2024). https://doi.org/10.52202/078368-0127

[28] Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 25, pp. 2951–2959 (2012). https://doi.org/10.48550/arXiv.1206.2944

[29] Kisantal, M., Sharma, S., Park, T.H., Izzo, D., Märtens, M., D'Amico, S.: Satellite pose estimation challenge: Dataset, competition design, and results. IEEE Transactions on Aerospace and Electronic Systems **56**(5), 4083–4098 (2020). https://doi.org/10.1109/TAES.2020.2989063

[30] Park, T.H., Märtens, M., Jawaid, M., Wang, Z., Chen, B., Chin, T.-J., Izzo, D., D'Amico, S.: Satellite pose estimation competition 2021: Results and analyses. Acta Astronautica **204**, 640–665 (2023). https://doi.org/10.1016/j.actaastro.2023.01.002

[31] Park, T.H., Martens, M., Lecuyer, G., Izzo, D., D'Amico, S.: Speed+: Next-generation dataset for spacecraft pose estimation across domain gap. In: 2022 IEEE Aerospace Conference (AERO), pp. 1–15. IEEE, ??? (2022). https://doi.org/10.1109/aero53065.2022.9843439 . https://doi.org/10.1109/AERO53065.2022.9843439. http://dx.doi.org/10.1109/AERO53065.2022.9843439

[32] Park, T., Bosse, J., D'Amico, S.: Robotic Testbed for Rendezvous and Optical Navigation: Multi-Source Calibration and Machine Learning Use Cases. https://doi.org/10.48550/arXiv.2108.05529

[33] Wang, Z., Chen, M., Guo, Y., Li, Z., Yu, Q.: Bridging the Domain Gap in Satellite Pose Estimation: a Self-Training Approach based on Geometrical Constraints. https://doi.org/10.48550/arXiv.2212.12103 (2022). https://arxiv.org/abs/2212.12103

[34] Pérez-Villar, J.I.B., García-Martín, A., Bescós, J.: Spacecraft Pose Estimation Based on Unsupervised Domain Adaptation and on a 3D-Guided Loss Combination, pp. 37–52. Springer, ??? (2023). https://doi.org/10.1007/978-3-031-25056-9_3

[35] Pérez-Villar, J.I.B., García-Martín, A., Bescós, J., SanMiguel, J.C.: Test-time adaptation for keypoint-based spacecraft pose estimation based on predicted-view synthesis. IEEE Transactions on Aerospace and Electronic Systems **60**(5), 6752–6764 (2024) https://doi.org/10.1109/taes.2024.3410956 . https://doi.org/10.1109/TAES.2024.3410956

[36] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. https://doi.org/10.1109/IROS.2017.8202133 (2017). https://arxiv.org/abs/1703.06907

[37] Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: Fast and flexible image augmentations. Information **11**(2), 125 (2020) https://doi.org/10.3390/info11020125 . https://doi.org/10.3390/info11020125

[38] Lotti, A., Modenini, D., Tortora, P., Saponara, M., Perino, M.A.: Deep learning for real-time satellite pose estimation on tensor processing units. Journal of Spacecraft and Rockets **60**(3), 1034–1038 (2023) https://doi.org/10.2514/1.A35496. https://doi.org/10.2514/1.A35496

[39] Park, T.H., D'Amico, S.: Bridging Domain Gap for Flight-Ready Spaceborne Vision. https://doi.org/10.48550/arXiv.2409.11661 (2024). https://arxiv.org/abs/2409.11661

[40] Capuano, G.M., Capuano, V., Napolano, G., Opromolla, R., Severi, M., Strollo, A., Petra, N., Cuciniello, G., Zaccagnino, E., varacalli, G.: Fpga hardware acceleration for deep learning-based satellite pose estimation. (2025). https://doi.org/10.2514/6.2025-2715

[41] Ingiosi, F., Ammirante, G., Taiano, G., Pennacchia, A., Corpino, S., Stesina, F., Branz, F., Caon, A., Lion, L., Imperatrice, M., Sansone, F., Girardi, G., Peruffo, M., Eynde, J., Pirat, C., Francesconi, A., Campioli, S., D'Ortona, A., La Bella, E., Deva, L.: The Space Rider Observer Cube (SROC) Cubesat Mission, pp. 201–212 (2024). https://doi.org/10.52202/078365-0023

[42] Jackson, P.T., Atapour-Abarghouei, A., Bonner, S., Breckon, T.P., Obara, B.: Style augmentation: Data augmentation via style randomization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 83–92 (2019). https://doi.org/10.48550/arXiv.1809.05375

[43] Pasqualetto Cassinis, L., Menicucci, A., Gill, E., Ahrns, I., Sanchez-Gestido, M.: On-ground validation of a cnn-based monocular pose estimation system for uncooperative spacecraft: Bridging domain shift in rendezvous scenarios. Acta Astronautica **196**, 123–138 (2022). https://doi.org/10.1016/j.actaastro.2022.04.002

[44] Blender Documentation Team: Blender 4.5 Manual, Compositing. https://docs.blender.org/manual/en/latest/compositing/introduction.html

[45] Blender Documentation Team: Blender 4.5 Manual, UVs. https://docs.blender.org/manual/en/latest/modeling/meshes/uv/uv_texture_spaces.html

[46] Blender Documentation Team: Blender 4.5 Manual, UDIMs. https://docs.blender.org/manual/en/latest/modeling/meshes/uv/workflows/udims.html

[47] Blender Documentation Team: Blender 4.5 Manual, Passes. https://docs.blender.org/manual/en/latest/render/layers/passes.html

[48] Fehse, W.: Automated Rendezvous and Docking of Spacecraft. Cambridge University Press, ??? (2003). https://doi.org/10.1017/CBO9780511543388

[49] Caruana, R.: Multitask learning. Machine Learning **28**(1), 41–75 (1997). https://doi.org/10.1023/A:1007379606734

[50] Tan, M., Le, Q.V.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1905.11946 (2020). https://arxiv.

org/abs/1905.11946

[51] Wightman, R.: PyTorch Image Models. GitHub (2019). https://doi.org/10.5281/zenodo.4414861

[52] Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal Loss for Dense Object Detection. https://doi.org/10.1109/TPAMI.2018.2858826 (2018). https://arxiv.org/abs/1708.02002

[53] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. https://doi.org/10.48550/arXiv.1911.08287 (2019). https://arxiv.org/abs/1911.08287

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

[55] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019). https://doi.org/10.1145/3292500.3330701

[56] Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian Optimization of Machine Learning Algorithms (2012). https://arxiv.org/abs/1206.2944

[57] Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., Lindauer, M.: Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges. https://doi.org/10.1002/widm.1484 (2021). https://arxiv.org/abs/2107.05847

[58] Li, L., Jamieson, K.G., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., Talwalkar, A.: Massively parallel hyperparameter tuning. CoRR (2018) 1810.05934. https://doi.org/10.48550/arXiv.1810.05934

[59] Jamieson, K., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. In: Gretton, A., Robert, C.C. (eds.) Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 51, pp. 240–248. PMLR, Cadiz, Spain (2016). https://doi.org/10.48550/arXiv.1502.07943. https://proceedings.mlr.press/v51/jamieson16.html

[60] Watanabe, S.: Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. https://doi.org/10.48550/arXiv.2304.11127 (2023). https://arxiv.org/abs/2304.11127

[61] Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)

[62] ONNX Runtime developers: ONNX Runtime. https://onnxruntime.ai/ (2021)

[63] NVIDIA: Jetson Orin Nano Super Developer Kit, The Most Affordable Generative AI Supercomputer. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/

[64] Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. https://doi.org/10.48550/arXiv.1711.05101 (2019). https://arxiv.org/abs/1711.05101

[65] Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 754–762. PMLR, Bejing, China (2014). https://proceedings.mlr.press/v32/hutter14.html