



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
SCIENZE E TECNOLOGIE AEROSPAZIALI

Ciclo 38

Settore Concorsuale: 09/A1 - INGEGNERIA AERONAUTICA, AEROSPAZIALE E
NAVALE

Settore Scientifico Disciplinare: ING-IND/05 - IMPIANTI E SISTEMI
AEROSPAZIALI

MACHINE VISION FOR ROBUST AND RESOURCE-EFFICIENT
SPACECRAFT NAVIGATION

Presentata da: Roman Prokazov

Coordinatore Dottorato
Dario Modenini

Supervisore
Paolo Tortora

Co-supervisor
Dario Modenini
Andrea Merlo

Esame finale anno 2026

Abstract

The proliferation of space activities has substantially intensified the criticality of autonomous spacecraft navigation systems for both on-orbit servicing (OOS) and active debris removal (ADR) missions. This dissertation focuses on developing and validating model-based monocular vision algorithms employing deep learning (DL) architectures for spacecraft pose estimation in close-proximity operations, specifically designed for deployment on computationally constrained embedded processors. The research addresses three main challenges inherent to vision-based navigation systems: achieving real-time performance on resource-limited hardware, bridging the synthetic-to-real domain gap, and ensuring operational reliability through uncertainty-aware decision-making frameworks.

The first contribution proposes a DL-based Close-Range Pose Estimation Pipeline (CR-PEP) tailored to the European Robotic Orbital Support Services (EROSS) mission. The pipeline combines object detection, keypoint regression, and Perspective-n-Point (PnP) stages with an active switching module that transitions between algorithmic configurations to maintain continuous pose estimation even when the target moves partially outside the camera’s field of view. Validation with photorealistic synthetic datasets and embedded hardware demonstrates real-time performance and accuracy compatible with close-proximity mission requirements.

The second contribution presents inference acceleration strategies for DL-based models deployment on embedded systems. Through model conversion, quantization, and batch processing, representative lightweight architectures are optimised, achieving significant speedups with minimal accuracy loss — confirming their suitability for satellite avionics.

The third contribution tackles the domain gap challenge by employing a CycleGAN-based unsupervised image-to-image translation network. Experiments demonstrate that retraining models on CycleGAN-translated synthetic imagery, derived from the existing SPEED+ dataset, results in substantial improvements on HIL test sets, with marked reductions in both keypoint localization and pose estimation errors. These outcomes establish adversarial domain adaptation via CycleGAN as a practical solution for synthetic-to-real transfer when annotated real data are limited and supervised fine-tuning is not feasible.

The fourth contribution introduces the Spacecraft Trajectory Imagery for Docking Estimation (STRIDE) dataset, comprising synthetic and real HIL imagery from the Zero-G Lab of the University of Luxembourg. A baseline Safe Autonomous Fallback for Estimation of pose (SAFE-pose) algorithm is also presented, integrating epistemic uncertainty quantification with classical computer vision fallback mechanisms. During inference, it estimates per-keypoint uncertainty and automatically triggers a robust marker-based detection fallback whenever the network’s predictive confidence is compromised, thereby ensuring safe pose estimation performance under degraded network confidence.

Acknowledgements

I would first like to express my deepest gratitude to my supervisors, Prof. Paolo Tortora and Prof. Dario Modenini, for their continuous guidance, support, and encouragement throughout my PhD journey. Their expertise and mentorship have been fundamental to both my academic and personal growth.

A sincere thank you goes to Prof. Marco Zannoni, whose advice and motivation inspired me to embark on this PhD path. I am also deeply grateful to Dr. Alessandro Lotti for his technical guidance and valuable assistance during my doctoral studies. My warmest thanks to Elia, Daniele Fratm, Andrea, and all the colleagues from Technopolo di Forlì for their support and friendship over these years.

During my second year, I had the opportunity to carry out a research internship at Thales Alenia Space in Turin under the co-supervision of Andrea Merlo. I would like to express my sincere appreciation for his guidance and the enriching discussions we shared. My gratitude also extends to the entire Robotics Group at Thales Alenia Space — Alessandro, Bongio, Ciro, Marco, Paolo, Patrick, Davide, Genny, Giusy, Gabriele, and Alessandro — for welcoming me and creating such a stimulating and collaborative environment. A special thank you to my fellow PhD colleagues Giacomo, Giuseppe, and Simone for their friendship and teamwork during this period.

In the third year, I had the privilege to spend a research period abroad at the Zero-G Lab, University of Luxembourg, under the supervision of Prof. Miguel Olivares-Mendez. I deeply appreciate his support and the opportunity to develop my research within his team. My heartfelt thanks also go to Carol, Mohatahem, Raul, Matteo, Andrej, Soffia, Georges, Ricard, Aman and to the rest of the team for their kindness, collaboration, and shared experiences. I am equally grateful to colleagues from other groups at the University of Luxembourg — Spyros, Vittorio, Alex, and Niki — for the enjoyable moments spent together, especially during coffee breaks. Finally, I would like to thank my parents for their unwavering belief in me and for their constant support throughout this long journey. And above all, my deepest gratitude goes to my girlfriend, Sonya, for her patience, understanding, and encouragement through every step of this adventure.

Table of Contents

List of Tables	vii
List of Figures	ix
1 Introduction	1
1.1 Motivation	1
1.2 State of the Art	2
1.2.1 Feature-based Pose Estimation	2
1.2.2 DL-Based Pose Estimation	4
1.2.3 Deep Learning on Embedded Hardware	7
1.2.4 Domain Gap	8
1.2.5 Uncertainty estimation in Deep Learning	9
1.3 Research objectives	12
2 Preliminaries	13
2.1 Deep Learning theory	13
2.1.1 Neural Networks	14
2.1.2 Activation Functions	16
2.1.3 Loss Functions	18
2.1.4 Backpropagation and Optimization	19
2.1.5 Regularization	22
2.1.6 Convolutional Neural Networks	24
2.2 Deep Learning Architectures for Pose Estimation	28
2.2.1 SSD ResNet50 V1 FPN 640×640	28
2.2.2 EfficientNet-B0	30
2.2.3 YOLO (You Only Look Once)	32
2.2.4 CycleGAN	35
2.3 Uncertainty Estimation in Deep Learning	38
2.3.1 Types of Uncertainty	38
2.3.2 Methods for Uncertainty Estimation	38
2.3.3 Uncertainty Quantification for Pose Estimation	40
2.4 Model Optimization for Embedded Hardware	40
2.4.1 Quantization	40
2.4.2 Pruning Strategies	42
2.4.3 Deployment Frameworks	43
2.5 Perspective-n-Point Problem and Solvers	43

2.5.1	PnP Problem Formulation	43
2.5.2	Efficient Perspective-n-Point (EPnP)	44
2.5.3	Random Sample Consensus (RANSAC)	45
2.6	Evaluation Metrics	45
2.6.1	Object Detection	46
2.6.2	Keypoint Regression	47
2.6.3	Pose Estimation	48
2.6.4	Domain Adaptation	50
3	Pose estimation for the EROSS mission	51
3.1	EROSS mission overview	51
3.2	EROSS dataset	53
3.2.1	EROSS dataset with random spacecraft poses (Blender)	54
3.2.2	EROSS trajectories dataset (Unreal Engine 5)	55
3.3	Pose estimation for the EROSS mission	57
3.3.1	System requirements	57
3.3.2	Pipeline architecture	58
3.3.3	ODN and KRN overview and training process	60
3.4	Validation results on Jetson Nano	62
3.5	Chapter conclusion	66
4	Inference Acceleration on Embedded Systems	68
4.1	Models Training and Configuration	68
4.1.1	Dataset Overview	68
4.1.2	Object Detection Network: YOLOv8n	69
4.1.3	Object Detection Network: EfficientNet-B0	72
4.2	Inference Optimization Strategies	74
4.2.1	Model Format Conversion	74
4.2.2	Numerical Precision Reduction	75
4.2.3	Batch Processing	76
4.3	Experimental Results	76
4.3.1	YOLOv8n Performance	76
4.3.2	EfficientNet-B0 Performance	77
4.3.3	Comparative Analysis	78
4.4	Discussion and Limitations	80
4.4.1	Performance context	80
4.4.2	Identified Limitations	81
4.5	Chapter Conclusion	81

5	Addressing Domain Gap with CycleGAN	83
5.1	Models Training and Configuration	83
5.1.1	Dataset Overview	84
5.1.2	Object Detection Network: YOLO11n	84
5.1.3	Keypoint Regression Network: EfficientNet-B0	86
5.2	CycleGAN Domain Adaptation	89
5.2.1	Implementation Details	89
5.2.2	Quality Assessment	90
5.3	Experimental Results	92
5.3.1	Baseline Performance Assessment	92
5.3.2	Domain-Adapted Model Performance	94
5.4	Discussion and Limitations	95
5.4.1	Performance Contextualization	95
5.4.2	Identified Limitations	96
5.5	Chapter Conclusion	97
6	SAFE-pose validation on STRIDE Dataset	99
6.1	STRIDE Dataset	99
6.1.1	Zero-G Lab	99
6.1.2	Dataset Acquisition	101
6.1.3	Real subset Description	104
6.2	SAFE-Pose Estimation Pipeline	106
6.2.1	Validation Results	108
6.3	Chapter Conclusion	116
7	Conclusion	118
7.1	Review of Contributions	118
7.1.1	EROSS Mission Pose Estimation Pipeline	118
7.1.2	DL-models inference acceleration on Jetson Orin nano	119
7.1.3	CycleGAN-Based Domain Adaptation	120
7.1.4	STRIDE Dataset and SAFE-Pose Pipeline	121
	References	123

List of Tables

2.1	EfficientNet family specifications showing compound scaling progression from B0 to B7, with corresponding input resolution, parameters, and computational cost [1].	31
2.2	Performance comparison of YOLOv8 and YOLOv11 model variants at 640×640 input resolution, reporting accuracy-speed trade-offs across multiple model scales. All models are pre-trained on the COCO dataset [2] and evaluated on the validation set [3].	35
3.1	Examples of trajectory scenarios under different illumination conditions. . . .	56
3.2	System-level requirements for the EROSS pose estimation subsystem.	58
3.3	Dataset Division for Training, Validation, and Testing.	60
3.4	Comparison of object detection performance between ResNet50 and EfficientNet-B0 models on the Blender-based spacecraft test set.	61
3.5	Inference speed of each network module and overall pipeline on NVIDIA Jetson Nano.	63
3.6	Complete pipeline performance evaluation across different test scenarios. . . .	64
4.1	Dataset Configuration for Inference Optimization Experiments.	69
4.2	YOLOv8n Training Hyperparameters.	70
4.3	YOLOv8n Training Data Augmentation Parameters.	70
4.4	EfficientNet-B0 Training Hyperparameters.	72
4.5	EfficientNet-B0 Training Data Augmentation Parameters.	73
4.6	YOLOv8n Jetson Orin Nano inference performance across multiple precision and deployment formats.	77
4.7	EfficientNet-B0 Jetson Orin Nano inference results across varying precision and deployment configurations. The superscript denotes the adopted quantization operator representation: 1 for QDQ and 2 for QOperator.	78
4.8	Inference speed obtained on Jetson Nano in seconds [4].	80
5.1	SPEED+ Dataset Composition [5].	84
5.2	YOLO11n Data Augmentation Configuration.	84
5.3	YOLO11n Training Configuration.	85
5.4	Keypoint Regression Data Augmentation Pipeline.	87
5.5	Keypoint Regression Training Configuration.	88
5.6	CycleGAN Training Configuration.	90
5.7	Baseline Cross-Domain Performance Evaluation.	93

5.8	Domain Adaptation Results: Lightbox and Sunlamp Domains.	95
5.9	Performance Comparison with SPEC2021 Competition Results.	96
6.1	Mockup orientation with respect to camera reference frame (Cases 1-22). . . .	105
6.2	Object detection performance comparison between synthetic and CycleGAN retrained YOLO.	109
6.3	Keypoint detection and pose estimation performance (cases 1–15).	111
6.4	Object detection performance comparison between synthetic and CycleGAN retrained YOLO.	112
6.5	Keypoint detection and pose estimation performance (cases 16–22).	113
6.6	Model Performance Comparison on Nvidia Jetson Orin Nano	115
6.7	Monte Carlo Dropout computational cost for EfficientNet-B0	115
6.8	Component-wise inference latency breakdown	115

List of Figures

1.1	Example images from different domains of SPEED+ [5].	5
2.1	Visualization of the hierarchical relationship among artificial intelligence, machine learning, and deep learning [6].	14
2.2	Diagram of a fully connected feedforward neural network with three hidden layers, illustrating the dense connections between input, hidden, and output nodes [7].	15
2.3	Common activation functions used in neural networks.	17
2.4	Illustration of the gradient descent algorithm optimizing a loss surface with respect to model parameters, highlighting the iterative trajectory toward the global minimum [8].	20
2.5	Illustrative comparison demonstrating the impact of regularization on generalization performance.	22
2.6	Schematic overview of a convolutional neural network (CNN) architecture, illustrating the sequence of feature extraction through convolution and pooling layers followed by classification using fully connected layers [9].	25
2.7	A convolution operation using a 3×3 kernel, unit stride, and half padding on a 5×5 input, illustrating how padding and stride influence the output [10].	25
2.8	Hierarchical expansion of the receptive field across convolutional layers, illustrating how deep networks integrate local and global spatial context [11].	26
2.9	Comparison between max pooling and average pooling operations commonly found in convolutional architectures [12].	27
2.10	SSD ResNet50 V1 FPN architecture [13].	29
2.11	EfficientNet-B0 architecture illustrating the baseline network design with Mobile Inverted Bottleneck Convolution (MBConv) blocks, squeeze-and-excitation modules, and progressive spatial downsampling through nine stages from input to output [14].	31
2.12	YOLO architecture [15].	33
2.13	CycleGAN architecture showing the bidirectional generators G and F , discriminators D_X and D_Y , and the cycle consistency constraint that enforces $F(G(x)) \approx x$ and $G(F(y)) \approx y$ [16].	36
2.14	Illustration of pruning methods: (a) Non-structured weight pruning; (b) Three types of structured pruning [17].	42

2.15	Visualization showing the geometric setup of the PnP problem. The transformation from the body-fixed frame B to the camera frame C is parameterized by the rotation matrix $\mathbf{R}_{BC}$ and translation vector $\mathbf{t}_{BC}$. [18]	44
2.16	Illustration of the Intersection over Union (IoU) metric [19].	46
3.1	Artistic rendering of the EROSS spacecraft during close-range proximity operations, illustrating the robotic servicing arm and modular spacecraft components in orbit [20].	52
3.2	Development schedule for the EROSS mission [20].	53
3.3	Representation of Blender-based Earth model.	54
3.4	Example renderings of the Euclid satellite in randomly generated poses, used for illustrative purposes.	55
3.5	Top and side views of the servicer approaching target characterising α and β angles.	56
3.6	Example renderings depicting the Euclid spacecraft's motion along a trajectory sequence, used for illustrative purposes. The left image shows the furthest distance, while the right image shows the closest approach.	56
3.7	Trajectory profiles for two representative EROSS dataset sequences.	57
3.8	Close-range pose estimation pipeline.	59
3.9	Close-range pose estimation pipeline with active switch.	59
3.10	Schematic overview of the EfficientNet-B0-based detection and keypoint regression networks.	62
3.11	Visual examples of data augmentation transformations applied to images during training.	62
3.12	Validation pose errors across test trajectories. The frame range on the x-axis corresponds to a relative distance range between the chaser and target from 2 m to 20 cm. The green vertical line indicates the activation of the switching mechanism between satellite body and marker-based pose estimation.	65
4.1	Representative examples of data augmentation techniques employed in YOLOv8n training.	70
4.2	Training and validation loss evolution for YOLOv8n over 100 epochs.	71
4.3	Representative YOLOv8n validation predictions after training.	72
4.4	Representative EfficientNet-B0 validation predictions after training.	73
4.5	Training and validation loss evolution for EfficientNet-B0 over 100 epochs.	74
4.6	Inference time comparison between EfficientNet-B0 and YOLOv8n.	79
4.7	Accuracy comparison between EfficientNet-B0 and YOLOv8n.	79

5.1	YOLO11n training metrics: Training and validation losses (top), precision/recall curves (bottom left), and mean Average Precision metrics (bottom right) across 100 training epochs.	86
5.2	Augmentation pipeline examples: Visual demonstration of augmentation effects, including original image (left), Gaussian noise application (center), and physics-based solar flare simulation (right).	87
5.3	Training and validation RMSE loss curve evolution across 100 epochs.	89
5.4	Domain translation results for lightbox conditions: Comparison between original synthetic images (top row) and corresponding CycleGAN-translated outputs (bottom row).	91
5.5	Domain translation results for sunlamp conditions: Original synthetic renderings (top row) and CycleGAN-translated counterparts (bottom row).	91
5.6	Baseline performance visual examples across domains, clearly illustrating severe performance degradation under domain gap.	93
5.7	Representative translation failure examples demonstrating artifacts and distortions: lightbox domain (left) exhibits boundary discontinuities and spurious textures, while sunlamp domain (right) shows severe geometric distortions under extreme illumination conditions that compromise keypoint localization accuracy.	96
6.1	Zero-G lab facility [21].	100
6.2	STRIDE dataset acquisition setup.	101
6.3	Mockup, ArUco marker and RealSense D455 camera.	102
6.4	Comparison of the projected keypoints before and after calibration.	102
6.5	Blender setup comprising camera and mockup model.	103
6.6	Examples from the synthetic subset of the STRIDE dataset.	104
6.7	Examples from the real subset of the STRIDE dataset.	104
6.8	Top and side views of the chaser-camera approaching target characterizing α and β angles.	105
6.9	Examples from non-rectilinear cases, top panel—furthest, bottom panel—closest image.	106
6.10	SAFE-Pose pipeline architecture with fallback mechanism.	107
6.11	Representative examples of CycleGAN-based synthetic-to-real domain adaptation. Top row shows original synthetic images; bottom row displays corresponding GAN-translated images with enhanced photorealism.	109

6.12	Prediction examples across trajectory cases 1–15. Left column shows ODN bounding box detections with examples of both successful detections and fallback default boxes. Middle column displays KRN keypoint predictions with RMSE values, demonstrating both accurate predictions and failure cases. Right column presents ArUco-BCA marker detection results.	110
6.13	Comparison of SAFE-Pose performance for aligned trajectory cases (1–15) before and after CycleGAN-based domain adaptation. The plots show quaternion errors, position errors, and fallback usage statistics.	112
6.14	Prediction examples across trajectory cases 16–22. Left column shows ODN bounding box detections. Middle column displays KRN keypoint predictions. Right column presents ArUco-BCA marker detection results.	113
6.15	Comparison of SAFE-Pose performance for non-aligned trajectory cases (16–22) before and after CycleGAN-based domain adaptation. The plots show quaternion errors, position errors, and fallback usage statistics across cases.	114

Chapter 1

Introduction

1.1 Motivation

The ever-growing population of inactive satellites and debris in Low Earth Orbit (LEO) poses a significant threat to operational spacecraft [22]. Currently, over 40,000 objects are tracked, yet actual quantities of debris larger than 1 cm are estimated to exceed 1.2 million, with approximately 54,000 units above 10 cm in size. Researchers also estimate there are around 130 million smaller debris fragments between 1 mm and 1 cm, many of which cannot be individually tracked but nonetheless present hazards to active satellites. Fragmentation incidents, such as accidental collisions and satellite breakups, continue to add thousands of new debris objects every year and therefore increase the probability of igniting the Kessler syndrome [23] - a scenario in which the density of objects in orbit reaches a critical threshold, causing collisions to generate further debris in a self-sustaining cascade. This chain reaction can make certain orbital regions increasingly hazardous or even unusable for future space operations. Given this heightened risk, dedicated space debris removal and OOS missions are critically important. In this regard, efforts such as RemoveDEBRIS [24], Clearspace-1 [25], CRD2 [26], and EROSS [27] have been initiated in recent years. A fundamental requirement for these missions is the ability of the servicer satellite to quickly and accurately estimate the relative pose of the target space borne object. Currently, regardless of whether the target is cooperative, employing radio links or fiducials, or non-cooperative with known or unknown geometry, pose estimation typically involves cumbersome and expensive Light Detection and Ranging (LI-

DAR), which may impede the widespread adoption of space debris removal technologies. A promising alternative to LIDARs is the use of monocular cameras combined with DL-based computer vision algorithms [28], offering a cost-effective and lightweight solution. Nevertheless, vision-based navigation introduces added software complexity and demands robust performance under harsh illumination conditions present in the space environment. Examples of such conditions include direct sunlight causing intense glare, strong specular reflections from spacecraft surfaces, and variable albedo from Earth. Focusing on the very close-range during docking or pre-capture scenario, additional challenges emerge. As the servicer spacecraft approaches its target, the camera may experience partial or complete loss of visual coverage due to the target moving outside of its field of view (FOV), as well as substantial shadows caused by the relative geometry of the two spacecraft. Moreover, specifically DL-based pose estimation approaches still encounter significant reliability challenges in terms of domain adaptation to real mission data [29] and quantification of prediction uncertainty [30]. This can pose serious risks when such systems are embedded in autonomous control loops. Validation remains challenging, with few companies willing to test unproven algorithms on expensive satellites.

Addressing all aforementioned challenges, this research, co-funded by Thales Alenia Space Italia (TASI), aims to develop and validate DL-based pose estimation algorithms optimized for resource-efficiency and reliability during close proximity operations. The work focuses on uncooperative and cooperative target scenarios, with an emphasis on coping with shadow artifacts and limited camera viewpoints at very close range.

1.2 State of the Art

1.2.1 Feature-based Pose Estimation

In conventional feature-based pose estimation, salient image features are extracted from the two-dimensional (2D) image of the target spacecraft, matched to corresponding points on its known three-dimensional (3D) model, and used to solve the Perspective-n-Point (PnP) problem for estimating the six-degree-of-freedom (6-DoF) relative pose [31]. Over several decades, numerous feature extraction and correspondence algorithms have been proposed, initially for terrestrial computer vision applications and later adapted to the spaceborne operational environment [32, 33, 34].

Initial investigations in this domain predominantly employed handcrafted image feature detectors, including Harris corner detectors, Sobel and Canny edge operators, Hough line extractors, and scale- and rotation-invariant descriptors such as Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Binary Robust Independent

Elementary Features (BRIEF), and Oriented FAST and Rotated BRIEF (ORB) [35, 36, 37, 38, 39]. Following feature identification within the image, algorithms establish 2D-3D correspondences before minimizing a reprojection error function to determine the complete pose vector. The accuracy of this methodology is fundamentally contingent upon the quality and distinctiveness of extracted features, the robustness of the correspondence establishment stage, the resilience against background with harsh illumination conditions and correspondence outliers. Outlier rejection is typically handled by the Random Sample Consensus (RANSAC) [40] algorithm.

Within the aerospace domain, first contributions established closed-form solutions for aligning 2D edge features with 3D line segments through predictive search and verification strategies [41, 42, 43]. However, pose of the target in these approaches assumed roughly known in advance. Efforts to generalize feature-based pose estimation for complete pose initialization—independent of prior knowledge regarding target pose—have encompassed view-graph architectures [44] and template-matching strategies [45], wherein pre-rendered libraries of target views are compared against observed spacecraft contours. While demonstrating promise, these approaches exhibit limited scalability, as modest variations in view-point or illumination conditions can substantially alter target appearance. Further works introduced by Grompone [46] and D’Amico [33] incorporated corner and edge detection algorithms (e.g., Harris, Hough transform), and perceptual organization of Sobel edges, successfully applied to real imagery from missions including PRISMA [47].

Recent research [48] has focused on enhancing the robustness of feature detectors against sensor noise and environmental variability by implementing advanced algorithms for line segment detection (such as the Line Segment Detector [49]) and corner detection (such as Shi-Tomasi [50]). Lastly, in order suppress background interference, particularly Earth in the image field, Weak Gradient Elimination (WGE) methodology has been introduced by Sharma et al. [51].

In summary, feature-based pose estimation remains a foundational technique for spacecraft relative navigation, providing interpretable intermediate representations and initial pose hypotheses from monocular imagery. However, its reliance on carefully calibrated parameters, handcrafted feature engineering, sensitivity to background and challenging illumination conditions constrains its scalability across diverse spacecraft geometries and operational scenarios. These inherent limitations have motivated the transition toward learning-based and hybrid approaches that endeavor to generalize beyond the constraints of classical image processing pipelines.

1.2.2 DL-Based Pose Estimation

The application of DL techniques to spacecraft pose estimation has experienced rapid development in recent years, driven primarily by the unavailability of large-scale real spaceborne image datasets and the computational advantages offered by modern Convolutional Neural Networks (CNNs). This subsection surveys the key contributions in the field, examining the evolution of datasets and architectural approaches.

Dataset Evolution

The deployment of data-driven approaches for spacecraft pose estimation has been fundamentally enabled by the development of specialized datasets. The Spacecraft Pose Estimation Dataset (SPEED) [52] represents a first major contribution in this domain, comprising 15,300 images of the Tango spacecraft from the PRISMA mission [47]. The dataset includes 15,000 synthetic images generated in OpenGL-based software [53] and 300 real images captured at the Testbed for Rendezvous and Optical Navigation (TRON) facility of Stanford’s Space Rendezvous Laboratory (SLAB). SPEED was publicly available through the Satellite Pose Estimation Challenge (SPEC2019) [18], co-organized by SLAB and the European Space Agency’s Advanced Concepts Team.

Beyond SPEED, the research community has developed numerous complementary datasets. Musallam et al. introduced the SPARK dataset encompassing 11 different spacecraft rendered in Unity3D [54]. Proença and Gao [55] generated Spacecraft On-orbit (URSO) dataset leveraging Unreal Engine 4, which included 5,000 synthetic images of Soyuz and Dragon spacecraft. Kaki et al. [56] employed Blender’s Cycles rendering engine for Cygnus spacecraft imagery, while Dung et al. [57] created 3,117 synthetic images across multiple satellite configurations for various computer vision tasks. Additional notable contributions include the Envisat dataset rendered with Cinema 4D [58]. However, the main shortcoming of all aforementioned datasets lies in their predominant reliance on synthetic images. Until recently, no publicly available and comprehensive dataset included a sufficient amount of real-domain imagery to effectively assess model performance under the domain gap problem, which refers to the degradation in accuracy when models trained on synthetic data are applied to real-world observations.

Consequently, the SPEED+ dataset [5] was introduced to further advance research in spacecraft pose estimation, with a specific focus on mitigating the domain gap. Compared to its predecessor, SPEED+ is considerably larger and incorporates diverse illumination conditions that more accurately capture the visual complexity of on-orbit scenarios. The dataset comprises 59,960 synthetic images, including 47,966 for training and 11,994 for validation, along with a substantial set of real images acquired from two distinct domains

as shown in Figure 1.1. The first, referred to as the Lightbox domain, contains 6,740 images collected at SLAB using the TRON facility. The second, termed the Sunlamp domain, includes 2,791 images obtained under direct, high-intensity illumination. These two domains were designed to reproduce realistic optical phenomena by employing light boxes with diffuser plates to simulate surface albedo effects and a sun lamp to emulate solar illumination conditions. SPEED+ was made publicly available through the SPEC2021 [59] competition, establishing a standardized benchmark for evaluating and comparing DL-based spacecraft pose estimation models under realistic operational environments.

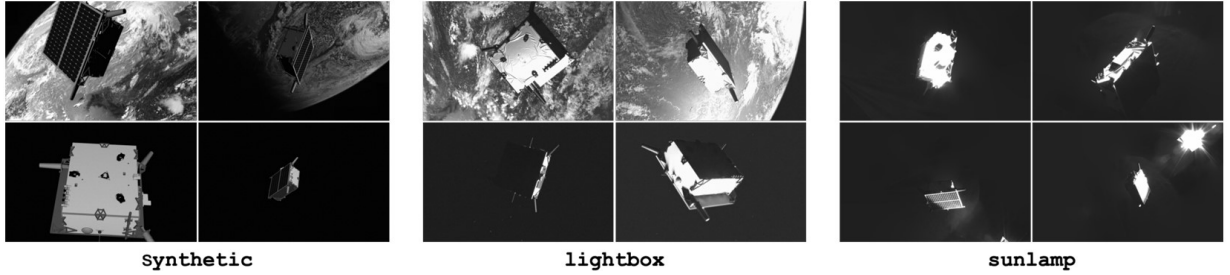


Figure 1.1: Example images from different domains of SPEED+ [5].

Architectural Evolution

The first DL-based spacecraft pose estimation algorithm proposed in the literature is the Spacecraft Pose Network (SPN) [60]. SPN is a hybrid approach for relative attitude determination, integrating both attitude classification and regression, as well as translation estimation based on perspective transformation and geometric constraints. The AlexNet [61] CNN was adopted for this purpose. Various versions of SPN were evaluated on a custom synthetic image superset consisting of 500,000 OpenGL-rendered images of the Tango spacecraft. These images encompass a wide range of poses, noise conditions, and target locations within the image frame, and ultimately provided the foundation for the development of the SPEED dataset.

With the official release of the SPEED dataset through the SPEC2019 competition, submissions from top-performing teams revealed a clear architectural convergence toward hybrid three-stage pipelines as opposed to direct end-to-end pose estimation. This commonly adopted pipeline proceeds as follows. First, an object detection network (ODN) is employed to localize the region of interest (ROI) around the spacecraft in the input image. Next, a keypoint regression network (KRN) is applied to the cropped region to directly regress the 2D pixel coordinates of predefined keypoints on the Tango spacecraft. Finally, given the known 3D model and the predicted keypoints, a Perspective-n-Point (PnP) algorithm is used to estimate the pose of the target. It is important to emphasize that ROI

cropping prior to network inference is critical, particularly when processing high-resolution imagery (1920×1200 px) with neural networks designed for ImageNet-standard [62] inputs (224×224 px). Without such preprocessing, substantial image detail may be lost during downsampling, impairing keypoint localization and overall pose estimation accuracy. In fact, Chen et al. [63] ranked first place in the competition by applying a three-stage architecture utilizing HRNet-based heatmap regression network for keypoint localization. Subsequent investigations have widely adopted similar architectural strategies [64, 58, 65].

Finally, The SPEC2021 [59] competition and released SPEED+ [5] dataset significantly advanced the state-of-the-art in spacecraft pose estimation by directly addressing the challenge of the domain gap. A key outcome of the competition was the demonstration that adversarial training and multi-task learning pipelines are critical to achieving high pose estimation accuracy across domain gaps. The two leading teams—TangoUnchained and lava1302—achieved remarkable accuracy on Hardware-In-the-Loop (HIL) test images without access to ground-truth pose labels during training. Both solutions yielded mean position errors below 11 cm and rotation errors under 4° for both lightbox and sunlamp domains. TangoUnchained combined object detection, adversarial HRNet-based keypoint regression, and a geometric PnP solver, with a particular emphasis on adversarially regularized landmark localization and multi-scale heatmap supervision. This approach proved especially effective for the challenging HIL-Lightbox domain, which featured low illumination and potential occlusion. In contrast, Wang et al. (lava1302 [66]) adopted a transformer-style backbone (HRFormer-S [67]) with dual heads for heatmap regression and fine-grained semantic segmentation, jointly optimized using adaptive wing [68] and cross-entropy losses. Their architecture was further strengthened by adversarial training on mask outputs and rounds of self-training leveraging pseudo-labels inferred from unlabeled test images. Lava1302’s method excelled in the HIL-Sunlamp domain, handling severe overexposure and surface glare. Notably, Lotti et al. (u3s_lab [69]), who ranked fourth and fifth in the sunlamp and lightbox domains respectively, utilized a pipeline that combined Swin Transformer-based [70] backbones for both object detection and landmark regression. Their approach focused on adversarial domain adaptation, but also explored lightweight hybrid CNN-ViT [71] models, placing emphasis on low-latency inference for embedded hardware.

Despite impressive gains of SPEC2021, ongoing challenges related to generalization, target diversity, and operational scalability remain open. Future efforts should prioritize adaptable, source-free (i.e., without requiring target domain access during adaptation) strategies that scale across diverse mission profiles and illumination conditions, ensuring that DL-based pose estimation methods continue to mature for deployment in complex, dynamic space environments.

1.2.3 Deep Learning on Embedded Hardware

The deployment of DL models on resource-constrained embedded hardware presents unique challenges for spacecraft applications, where computational resources, power consumption, and model size must be carefully balanced against accuracy requirements. This section reviews key approaches for deploying CNNs and DL-based spacecraft pose estimation pipelines on embedded platforms.

Model Compression

Model compression has emerged as a critical enabler for deploying deep neural networks (DNNs) on embedded systems. Han et al. [72] introduced Deep Compression, a three-stage pipeline combining pruning, trained quantization, and Huffman coding that achieved $35\times$ to $49\times$ compression ratios without accuracy loss. However, the approach requires custom hardware accelerators, such as EIE [73], to achieve full computational benefits. Building on this work, Iandola [74] proposed SqueezeNet, which achieves AlexNet-level accuracy with $50\times$ fewer parameters through architectural innovations including Fire modules and delayed downsampling, demonstrating that small models are inherently amenable to compression techniques. Finally, Jiang et al. [75] implemented single-shot pruning and quantization for embedded systems, achieving 69.4% model size reduction with $6-8\times$ speedup on NVIDIA Xavier NX, but the approach has not been validated for spacecraft applications where radiation tolerance is critical.

FPGA-Based Platforms

Field-Programmable Gate Arrays (FPGAs) are gaining more popularity for space applications due to their reconfigurability and power efficiency. Cosmas et al. [76] demonstrated FPGA-based CNN acceleration for spacecraft landmark localization using Xilinx Zynq Ultrascale+ MPSoC with Vitis AI, achieving an average Root Mean Square (RMS) error of 1.913 on the SPEED dataset; however, the implementation only processed the CNN backbone on FPGA while requiring CPU for post-processing, limiting end-to-end performance. Furthermore, Posso et al. [77] presented the first real-time spacecraft pose estimation on FPGA using mixed-precision quantization with the FINN library, achieving $7.7\times$ faster inference and $19.5\times$ better energy efficiency than previous work, though the quantization below 4 bits introduced accuracy degradation in pose estimation due to direct regression sensitivity. Lastly, Guo [78] developed an overlay accelerator for DeepLab CNN targeting spacecraft image segmentation on FPGA with a domain-specific instruction set, enabling reconfigurability without hardware re-burn, though the approach has not been extended to full pose estimation pipelines.

GPU-Based Embedded Platforms

GPU-based embedded platforms, such as NVIDIA Jetson series, offer alternative deployment options with different trade-offs. Swaminathan [79] benchmarked DNNs on Jetson Nano for real-time systems, finding 16.11% average speedup for optimized models over non-optimized counterparts. Alqahtani et al. [80] evaluated YOLOv8, EfficientDet, and SSD models on Raspberry Pi and Jetson Orin Nano platforms, demonstrating that Jetson Orin Nano achieved the most optimal balance of speed and accuracy for object detection, though its higher power consumption (5-15W) compared to FPGAs (1-3W) may limit suitability for small spacecraft with strict power budgets. Finally, Li et al. [81] investigated multi-model concurrent execution on Jetson Nano for autonomous systems, revealing significant scheduling challenges when running multiple CNNs simultaneously, which can be a common requirement in spacecraft operations.

1.2.4 Domain Gap

In general, the domain gap refers to the discrepancy between the distributions of the training and test datasets, which leads to a degradation in model performance when a model trained on one domain is applied to another [29]. In the context of space applications, the domain gap typically manifests as a synthetic-to-real transfer problem, where models trained exclusively on synthetic imagery exhibit degraded performance when applied to real images. This represents a fundamental challenge, as it is extremely difficult to accurately reproduce the harsh illumination conditions and diverse noise characteristics of the space environment during the synthetic data generation process [5]. Consequently, domain adaptation techniques must be employed to bridge this gap and enhance model generalization to real-world data.

One widely adopted approach to address this gap is domain randomization [82, 83]. Tobin et al. [84] introduced domain randomization strategy by introducing extensive variability in simulated environments, such as randomizing textures, lighting, object placement, and camera parameters, so that real-world data appears as another variation to the model. Yue et al. [85] proposed a domain randomization framework combined with pyramid consistency, which enforces feature invariance across multiple stylized versions of synthetic images. It is important to note that domain randomization is an attractive strategy since it does not require access to target domain data. However, in space applications, it offers no guarantee of accurate model performance on real imagery, as it often fails to capture the complex and subtle characteristics of the operational environment.

While domain randomization remains a popular strategy, adversarial domain adaptation approaches have gained traction for their ability to explicitly align feature distributions

between source and target domains [86]. The core idea is to train a domain discriminator adversarially against the feature extractor, encouraging the model to learn representations that are invariant to domain-specific cues. Domain adaptation methods can be classified as supervised [87], semi-supervised [88], or unsupervised [89]. Supervised approaches require labeled data in both source and target domains, semi-supervised methods use labeled source data and a small amount of labeled target data (along with many unlabeled target samples), and unsupervised methods require labeled data only from the source domain while relying solely on unlabeled data from the target domain. Canonical examples of Unsupervised Domain Adaptation (UDA) methods include Domain-Adversarial Neural Networks (DANN) [90], Adversarial Discriminative Domain Adaptation (ADDA) [91], and Multiple Source Domain Adversarial Networks (MDAN) [92], which have demonstrated strong performance across benchmarks in visual recognition and pose estimation tasks. Another notable example of UDA is CycleGAN [16] that has been widely utilized for unsupervised image-to-image translation, enabling synthetic images to more closely resemble the appearance of real images without paired supervision.

While significant progress has been made in the domain adaptation field, most proposed techniques require access to target domain data during either the training or adaptation phase, whether labeled or unlabeled. This requirement presents a fundamental challenge for spacecraft applications, as real on-orbit imagery is typically unavailable until actual Rendezvous and Proximity Operations (RPO) commence. Park and D’Amico [93] addressed this limitation through Online Domain Refinement (ODR), a source-free test-time adaptation method that refines normalization layer parameters via self-supervised entropy minimization on unlabeled target domain images during deployment. An alternative approach to mitigating domain gap effects involves the use of event-based cameras [94], which exhibit substantially higher dynamic range compared to conventional RGB cameras and demonstrate inherent robustness to harsh lighting conditions that typically induce domain shift. Jawaid et al. [94] demonstrated that sensor fusion combining RGB and event cameras can leverage the complementary strengths of both modalities to achieve robust pose estimation under extreme illumination. However, event-based methods for spacecraft pose estimation remain largely unexplored. Lastly, the unpaired image-to-image translation capability of CycleGAN, explored in this thesis, still offers a promising solution when limited real imagery can be acquired in-flight during early mission phases or from heritage missions, thereby enabling progressive refinement of pose estimation models.

1.2.5 Uncertainty estimation in Deep Learning

Reliable uncertainty quantification enables autonomous systems to recognize when predictions are unreliable, flag anomalous conditions, and defer critical operations to human

operators when necessary, thereby enhancing mission safety. The overall uncertainty typically comprises aleatoric and epistemic components, which are modeled by a Gaussian distribution over the predictions and by placing a distribution over the model weights (i.e., a Bayesian Neural Network [95]), respectively. The approaches for object detection and keypoint detection will be described, to the best of the author’s knowledge, as applied in computer vision and spacecraft pose estimation.

Uncertainty for Computer Vision Tasks

Object detection uncertainty has focused on bounding box localization. He et al. [96] introduced a novel loss function for learning bounding box localization variance alongside coordinate predictions, addressing annotation ambiguities in large-scale datasets. Their variance voting approach during non-maximum suppression merges neighboring boxes weighted by predicted uncertainties, significantly improving Average Precision On MS-COCO dataset [2]. Mun et al. [97] developed a separate uncertainty prediction network for monocular 3D object detection, demonstrating that occlusion information strongly correlates with localization uncertainty. For unsupervised 3D detection, Zhang et al. [98] employed dual detection branches to estimate fine-grained coordinate-level uncertainty, weighting pseudo-labels during iterative training to mitigate noise.

In the context of keypoint detection, Luo et al. [99] proposed scale-adaptive heatmap regression that predicts input-dependent standard deviations for Gaussian kernels, effectively modeling aleatoric uncertainty across varying scales and labeling ambiguities. The predicted scale maps capture localization difficulty under occlusion, blur, or ambiguous appearance. Furthermore, Lu and Koniusz [100] developed few-shot keypoint detection with explicit uncertainty modeling using multivariate Gaussian distributions to exploit correlations between neighboring keypoints. Their framework enables automatic filtering of unreliable detections, improving robustness for novel keypoint detection on unseen categories. Finally, Pavlakos et al. [101] employed stacked hourglass networks for semantic keypoint detection, where heatmap intensities indicate spatial prediction uncertainty, successfully recovering 6-DOF object pose with well-calibrated confidence.

Uncertainty in Spacecraft Pose Estimation

Regarding uncertainty quantification for spacecraft pose estimation, Li et al. [102] proposed a keypoint-based pipeline with explicit uncertainty prediction, where a keypoint detection network simultaneously localizes semantic features and predicts per-keypoint uncertainties. Their automatic keypoint selection strategy filters detections based on predicted uncertainty before pose estimation via EPnP [103], outperforming heatmap-based

and regression-based methods on the SPEED [52] benchmark. Yuan et al. [104] extended this approach with non-coplanar keypoint selection, analyzing how spatial configuration affects pose measurement accuracy. Their network combines uncertainty-based filtering with geometric constraints to ensure well-conditioned pose estimation, significantly enhancing accuracy and computational efficiency of the SPEC2019 [18] top entries.

Direct pose regression with uncertainty modeling has also shown promise. Yang et al. [105] introduced PVSPE, a pyramid vision transformer employing Matrix Fisher [106] and multivariate Gaussian distributions to model pose uncertainty. The learned uncertainties enable robust integration with Extended Kalman Filters for vision-based navigation, achieving centimeter-level translation and degree-level attitude accuracy under challenging illumination and occlusions on the SPEED+ dataset. Furthermore, benchmarking on the same dataset, Dixon et al. [107] employed evidential regression to estimate both epistemic and aleatoric uncertainties, demonstrating a median correlation of 0.996 between network prediction errors and associated uncertainty estimates for keypoint coordinates in synthetic images. They also showed that their uncertainty values of the predictions reliably increase when faced with out-of-distribution inputs or elevated image noise.

Future Directions

Despite aforementioned advances, integration of uncertainty-aware pose estimation with downstream guidance, navigation, and control systems remains largely unexplored, requiring principled frameworks for uncertainty propagation through mission-critical decision pipelines to enable truly autonomous and safe space operations. Future research must establish standards for uncertainty evaluation in space applications, and create comprehensive benchmarks with real spaceborne imagery.

1.3 Research objectives

The principal aim of this research is to develop and validate DL-based spacecraft pose estimation algorithms for both cooperative and uncooperative targets with known 3D geometry, tailored for deployment on resource-constrained onboard processors in space environments. Specifically, this work pursues the following objectives:

- (a) Design a robust DL-based pose estimation framework suitable for close-proximity operations, with real-time inference capabilities on resources-constrained computational hardware.
- (b) Enhance the generalisation capability of vision-based pose estimators by mitigating the performance gap between synthetic and real spacecraft imagery through domain adaptation strategies.
- (c) Provide an imagery dataset to support the development and evaluation of DL-based spacecraft pose estimation algorithms.
- (d) Increase the robustness and safety of pose estimation systems through the integration of uncertainty quantification and fallback mechanisms, ensuring reliable operation under highly uncertain DL-based pose predictions.

The research objectives are addressed throughout the dissertation as follows.

Objective (a) is addressed in **Chapters 3** and **4**. Chapter 3 presents the design and validation of a DL-based pose estimation pipeline tailored to the EROSS on-orbit servicing mission, while Chapter 4 focuses on inference acceleration strategies for real-time inference on a Jetson Orin Nano embedded platform.

Objective (b) is tackled in **Chapter 5**, which explores domain adaptation through CycleGAN-based image-to-image translation to bridge the gap between synthetic and real spacecraft imagery.

Objectives (c) and **(d)** are covered in **Chapter 6**, which introduces the Spacecraft Trajectory Imagery for Docking Estimation (STRIDE) dataset and the Safe Autonomous Fallback for Estimation of Pose (SAFE-Pose) pipeline, implementing epistemic uncertainty quantification and classical computer vision algorithms as a fail-safe backup.

Chapter 2

Preliminaries

This chapter establishes the theoretical background essential for understanding the spacecraft pose estimation pipelines developed in this thesis. Section 2.1 introduces the mathematical foundations of deep learning, covering neural network architectures, training algorithms, and regularization techniques. Subsequently, Section 2.2 examines the specific CNN architectures employed in this research for object detection, keypoint regression, and domain adaptation tasks. Section 2.3 then addresses uncertainty quantification methods in deep learning, which enable confidence estimation for pose predictions under domain gap and challenging imaging conditions. Furthermore, Section 4.2 discusses model optimization techniques, such as quantization, pruning, and deployment frameworks, that facilitate efficient inference on resource-constrained spacecraft processors. Section 2.5 proceeds to formulate the Perspective-n-Point problem and describes the PnP algorithms utilized in this work. Finally, Section 2.6 defines the evaluation metrics used to quantify detection accuracy, keypoint localization precision, and overall pose estimation performance.

2.1 Deep Learning theory

Deep learning (DL) represents an advanced branch of machine learning (ML) that utilizes multilayered artificial neural networks to automatically learn hierarchical features from raw data, enabling the solution of highly complex tasks in fields such as computer vision and robotics [108]. Unlike traditional ML algorithms, which typically depend on manually engineered input features and may require substantial domain expertise [109], DL mod-

els extract increasingly abstract representations through layered transformations, thereby reducing the need for expert-guided feature selection and achieving state-of-the-art performance on a variety of unstructured datasets. Both DL and classical ML are encompassed within the broader discipline of artificial intelligence (AI) [110] as represented in Figure 2.1.

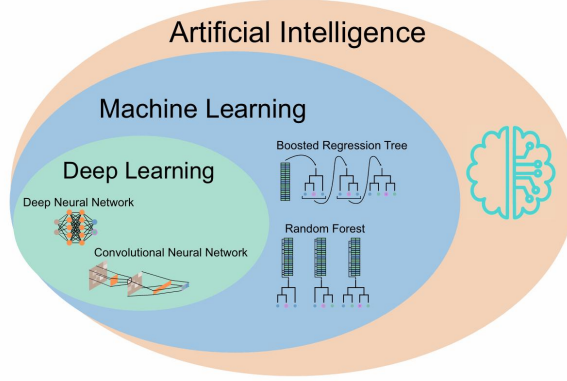


Figure 2.1: Visualization of the hierarchical relationship among artificial intelligence, machine learning, and deep learning [6].

2.1.1 Neural Networks

Artificial neural networks form the foundation of DL systems, drawing inspiration from the biological neural structures found in the human brain [108]. At the most fundamental level, an artificial neuron serves as the basic computational unit, processing multiple inputs to produce a single output through a mathematical transformation.

An artificial neuron, also known as a perceptron [111], computes a weighted sum of its inputs and applies a non-linear activation function to produce an output. Mathematically, for a neuron receiving n inputs $x_1, x_2, \dots, x_n$, the output y is computed as:

$$y = f\left(\sum_{i=1}^n w_i x_i + b\right) = f(\mathbf{w}^T \mathbf{x} + b) \quad (2.1)$$

where x_i are the input features, w_i are the learnable weights associated with each input, b is the bias term allowing the neuron to shift its activation threshold, $f(\cdot)$ is the activation function introducing non-linearity, and y is the neuron's output. The weights $\mathbf{w}$ and bias b are the learnable parameters that are adjusted during the training process to minimize the prediction error.

A neural network is constructed by organizing neurons into layers and connecting them to form a directed graph as represented in Figure 2.2. The standard feedforward neural network architecture [112] consists of three types of layers. The input layer receives the raw input data, with each neuron representing a single feature. Hidden layers perform

non-linear transformations on the data, learning increasingly abstract representations with early layers capturing low-level features and deeper layers learning high-level semantic concepts [62]. The term “deep” in deep learning refers to the presence of multiple hidden layers. The output layer produces the network’s predictions. For regression tasks like pose estimation, it directly predicts continuous values (e.g., position and orientation parameters), while for classification tasks, it typically uses a softmax activation to produce class probabilities.

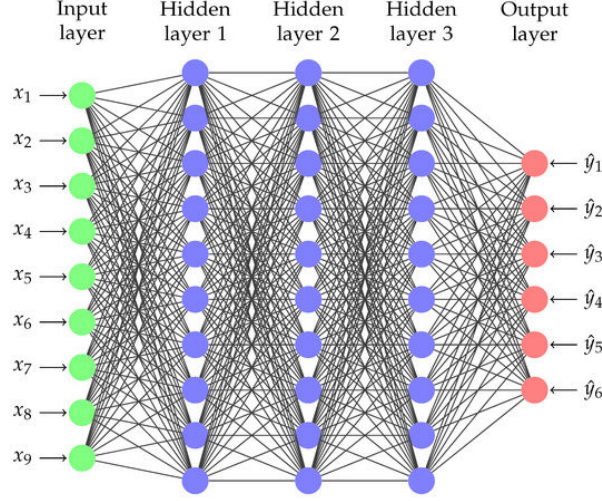


Figure 2.2: Diagram of a fully connected feedforward neural network with three hidden layers, illustrating the dense connections between input, hidden, and output nodes [7].

The process of computing a neural network’s output for a given input, known as forward propagation, involves the sequential flow of information from the input layer through the hidden layers to the output layer. For a network with L layers, the computation at each layer l is defined as:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \quad (2.2)$$

$$\mathbf{a}^{(l)} = f^{(l)}(\mathbf{z}^{(l)}) \quad (2.3)$$

where $\mathbf{a}^{(l)}$ denotes the activation (output) vector of layer l , $\mathbf{W}^{(l)}$ is the weight matrix connecting layer $l - 1$ to layer l , $\mathbf{b}^{(l)}$ is the bias vector of layer l , $\mathbf{z}^{(l)}$ represents the pre-activation value (weighted sum), $f^{(l)}(\cdot)$ is the activation function applied at layer l , and $\mathbf{a}^{(0)} = \mathbf{x}$ corresponds to the input vector. The weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ is determined by the number of neurons in consecutive layers, where n_l is the number of neurons in layer l . Each element $w_{ij}^{(l)}$ indicates the strength of the connection from neuron j in layer $l - 1$ to neuron i in layer l .

The final network output is obtained by recursively applying equations (2.2) and (2.3) across all layers:

$$\hat{\mathbf{y}} = f^{(L)}(\mathbf{W}^{(L)} f^{(L-1)}(\mathbf{W}^{(L-1)} \dots f^{(1)}(\mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)}) \dots + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)}) \quad (2.4)$$

This nested composition of linear transformations and nonlinear activation functions enables neural networks to learn hierarchical feature representations and approximate highly complex nonlinear mappings, as formalized by the universal approximation theorem [113, 114]. The network's depth (number of layers) and width (number of neurons per layer) jointly determine its representational capacity and its ability to model intricate patterns within the data.

2.1.2 Activation Functions

Activation functions introduce non-linearity into neural networks, enabling them to capture complex relationships between inputs and outputs. Without such non-linear components, even deep networks would reduce to a single linear transformation, severely limiting their representational capacity. The choice of activation function significantly affects training stability, convergence speed, and model generalization. The most commonly adopted activation functions are presented below and illustrated in Figure 2.3.

Sigmoid Function

The sigmoid activation function introduces smooth non-linearity and is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.5)$$

It maps any real-valued input to the range $(0, 1)$, making it suitable for binary classification tasks where outputs represent probabilities [115]. However, for large positive or negative inputs, gradients diminish rapidly, leading to the vanishing gradient problem and slower convergence in deep architectures [116].

Hyperbolic Tangent (tanh)

The tanh function is defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.6)$$

Unlike the sigmoid, $\tanh(x)$ produces zero-centered outputs within the range $(-1, 1)$, promoting more balanced gradient updates and often faster convergence during training [112]. Despite this advantage, tanh also suffers from gradient saturation for large input magnitudes, which can hinder learning in deeper networks.

Rectified Linear Unit (ReLU)

The ReLU [117] has become the standard activation function for most modern deep architectures. It is defined as:

$$\text{ReLU}(x) = \max(0, x) \quad (2.7)$$

ReLU offers several benefits: computational simplicity, non-saturating gradients for positive activations, and sparsity, which can improve model generalization [118]. However, ReLU units can become inactive when exposed to consistently negative inputs, leading to the “dying ReLU” issue, where neurons stop contributing to learning.

Leaky ReLU and Variants

The Leaky ReLU [119] modifies the ReLU function to preserve a small gradient for negative inputs:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{otherwise,} \end{cases} \quad (2.8)$$

where α is a small constant (typically 0.01). This ensures that neurons remain active and capable of recovery during training. Variants include the Parametric ReLU (PReLU) [120], in which α is a trainable parameter, and the Exponential Linear Unit (ELU) [121], which introduces an exponential term for negative activations to improve learning robustness and convergence behavior.

Softmax Function

For multi-class classification, the softmax function converts network logits into a normalized probability distribution:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (2.9)$$

where K denotes the number of output classes [122]. The softmax activation ensures that the sum of predicted probabilities equals one, making it well-suited for modeling categorical outcomes and probabilistic decision boundaries.

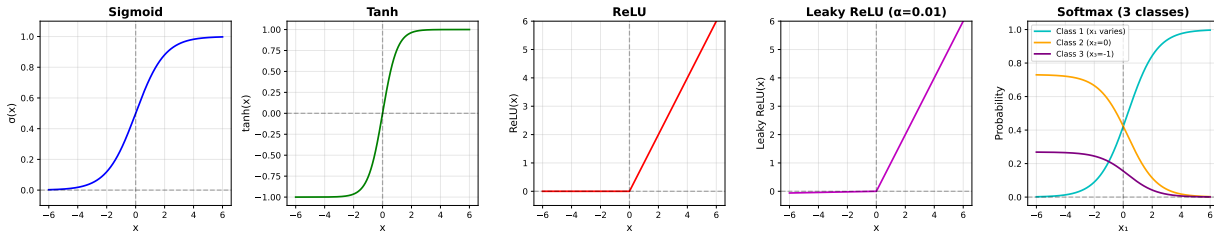


Figure 2.3: Common activation functions used in neural networks.

2.1.3 Loss Functions

Loss functions, also referred to as cost or objective functions, quantify the discrepancy between a network's predictions and the ground truth labels [123]. The selection of an appropriate loss function is crucial, as it defines the optimization objective and shapes the landscape that training algorithms explore. Different tasks, such as regression or classification, require distinct loss formulations to ensure proper learning behavior and performance.

Mean Squared Error (MSE)

For regression tasks, where the objective is to predict continuous values, the Mean Squared Error (MSE) is the most widely adopted loss function. It is defined as:

$$L_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.10)$$

where y_i denotes the true value, $\hat{y}_i$ the predicted output, and n the number of samples. MSE penalizes larger deviations quadratically, resulting in higher sensitivity to outliers. Its smooth and differentiable form makes it suitable for gradient-based optimization methods.

Mean Absolute Error (MAE)

A more robust alternative is the Mean Absolute Error (MAE), expressed as:

$$L_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.11)$$

MAE penalizes deviations linearly, providing resilience to outliers compared to MSE. However, its constant gradient near the optimal solution can slow convergence, particularly in models relying on fine-grained error adjustments during optimization.

Cross-Entropy Loss

For classification problems, Cross-Entropy (CE) loss [124], also referred to as log loss, is the standard choice. It is defined as:

$$L_{\text{CE}} = - \sum_{i=1}^K y_i \log(\hat{y}_i) \quad (2.12)$$

where y_i denotes the true label (often one-hot encoded) and $\hat{y}_i$ represents the predicted probability of class i . Cross-Entropy measures the divergence between the predicted and true probability distributions, penalizing confident yet incorrect predictions. This encourages the model to produce well-calibrated probabilistic outputs.

Binary Cross-Entropy (BCE)

In binary classification settings, Cross-Entropy simplifies to the Binary Cross-Entropy (BCE) formulation:

$$L_{\text{BCE}} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2.13)$$

where $y_i \in \{0, 1\}$ is the binary ground truth label, and $\hat{y}_i \in (0, 1)$ is the predicted probability. BCE effectively penalizes deviations between predicted and actual class probabilities, making it ideal for logistic regression and binary neural classification tasks.

2.1.4 Backpropagation and Optimization

Training a neural network entails adjusting its parameters (i.e., weights and biases) so that the loss function, which measures the discrepancy between the predictions and the ground truth labels, is minimized. This process is achieved through the combination of backpropagation and gradient-based optimization algorithms.

Backpropagation

Backpropagation [115], short for “backward propagation of errors,” is a fundamental algorithm used to compute the gradient of the loss function with respect to all model parameters efficiently. It operates by applying the chain rule of calculus recursively from the output layer back to the input layer. For a loss function L and a weight $w_{ij}^{(l)}$ connecting neuron j in layer $l - 1$ to neuron i in layer l , the corresponding gradient is expressed as:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} \cdot a_j^{(l-1)} \quad (2.14)$$

where $\delta_i^{(l)} = \frac{\partial L}{\partial z_i^{(l)}}$ represents the local error term at neuron i , and $a_j^{(l-1)}$ is the activation from the previous layer. The error term is propagated backward through the layers as:

$$\delta_i^{(l)} = \left(\sum_k w_{ki}^{(l+1)} \delta_k^{(l+1)} \right) f'^{(l)}(z_i^{(l)}), \quad (2.15)$$

where $f'^{(l)}(z_i^{(l)})$ denotes the derivative of the activation function. This backward recursion allows efficient gradient computation for all parameters without redundant calculations.

Gradient Descent

Gradient Descent [125] is the fundamental optimization method used to iteratively update parameters in the direction of the steepest loss decrease. The parameter update rule is

given by:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(\theta_t) \quad (2.16)$$

where θ denotes the vector of all network parameters, η is the learning rate controlling the step size, and $\nabla_{\theta} L$ represents the gradient of the loss with respect to the parameters. Choosing an appropriate learning rate is essential, values that are too large cause divergence due to overshooting, whereas excessively small values slow convergence [126]. Figure 2.4 provides a conceptual illustration of this optimization process.

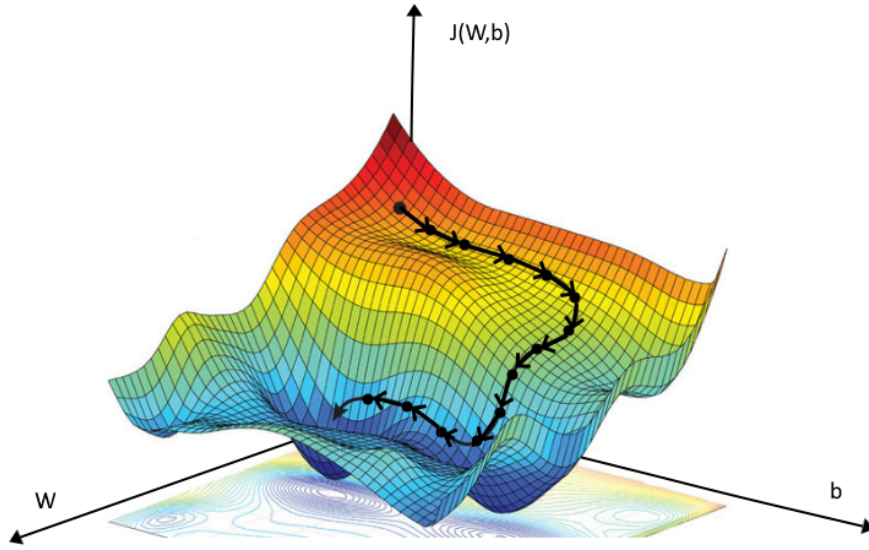


Figure 2.4: Illustration of the gradient descent algorithm optimizing a loss surface with respect to model parameters, highlighting the iterative trajectory toward the global minimum [8].

Advanced Optimization Algorithms

To accelerate convergence and improve training stability, several advanced optimization algorithms extend basic gradient descent by incorporating momentum and adaptive learning rate mechanisms. The most widely used methods are summarized below.

- **Momentum:** The Momentum [127] accumulates an exponentially weighted moving average of past gradients to smooth parameter updates and accelerate convergence along consistent gradient directions:

$$v_t = \beta v_{t-1} + \eta \nabla_{\theta} L(\theta_t), \quad (2.17)$$

$$\theta_{t+1} = \theta_t - v_t, \quad (2.18)$$

where β is a decay coefficient (typically 0.9). This formulation helps dampen oscillations and enhances stability in regions of steep curvature.

- **AdaGrad:** The Adaptive Gradient (AdaGrad) method [128] adjusts the learning rate for each parameter based on the historical accumulation of past gradient magnitudes. Parameters with large cumulative gradients receive smaller updates, while those with small gradients receive larger ones:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla_{\theta} L(\theta_t),$$

where G_t stores the sum of squared past gradients and ϵ prevents division by zero. Although adaptive, AdaGrad's learning rate decays excessively over long training runs, slowing convergence.

- **RMSProp:** RMSProp [129] improves upon AdaGrad by introducing an exponential decay factor to limit the effect of long-term gradient accumulation:

$$\begin{aligned} E[g^2]_t &= \gamma E[g^2]_{t-1} + (1 - \gamma)(\nabla_{\theta} L(\theta_t))^2, \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla_{\theta} L(\theta_t), \end{aligned}$$

where γ (typically 0.9) is the decay rate. RMSProp prevents the learning rate from diminishing too rapidly and is particularly effective on non-stationary or noisy objectives.

- **Adam (Adaptive Moment Estimation):** Adam [130] combines the benefits of both Momentum and RMSProp by maintaining exponentially decaying averages of both the first (m_t) and second (v_t) moments of the gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} L, \quad (2.19)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_{\theta} L)^2, \quad (2.20)$$

with bias-corrected estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \quad (2.21)$$

The parameter update rule is then:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t, \quad (2.22)$$

where $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ are typical default values. Adam adapts learning rates for each parameter individually, making it particularly effective for sparse, high-dimensional, or noisy optimization problems.

These optimizers represent significant advancements over standard gradient descent and are integral to the success of modern DL architectures.

2.1.5 Regularization

Regularization techniques are indispensable in mitigating overfitting — a condition in which a neural network captures noise or spurious correlations in the training data rather than learning the underlying data distribution (see Figure 2.5). Overfitting is particularly pervasive in deep architectures containing millions of trainable parameters trained on datasets of limited size or diversity [123]. By imposing explicit or implicit constraints on the learning process, regularization improves the model’s generalization capacity and stability during training.

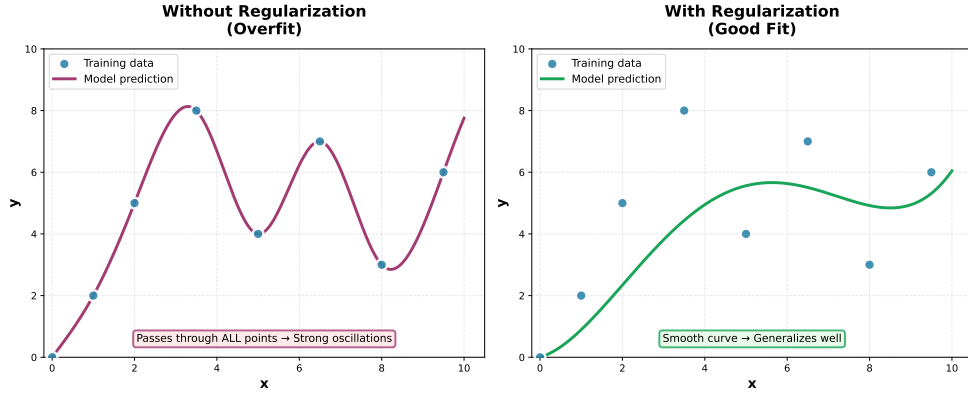


Figure 2.5: Illustrative comparison demonstrating the impact of regularization on generalization performance.

L2 Regularization (Weight Decay)

L2 regularization [131], also referred to as weight decay, introduces a penalty term proportional to the squared ℓ_2 -norm of the model parameters:

$$L_{\text{total}} = L_{\text{data}} + \lambda \sum_i w_i^2 = L_{\text{data}} + \lambda \|\mathbf{w}\|^2 \quad (2.23)$$

where $\lambda > 0$ controls the regularization strength. The inclusion of this penalty discourages excessively large parameter magnitudes, promoting smoother decision boundaries and improving generalization. During gradient-based optimization, the L2 term contributes an additional component to the gradient computation:

$$\frac{\partial L_{\text{total}}}{\partial w_i} = \frac{\partial L_{\text{data}}}{\partial w_i} + 2\lambda w_i, \quad (2.24)$$

which leads to a gradual decay of the weights toward zero unless continually reinforced by data-driven updates.

L1 Regularization

L1 regularization [132] enforces sparsity in the learned parameters by penalizing their absolute values rather than their squares:

$$L_{\text{total}} = L_{\text{data}} + \lambda \sum_i |w_i| = L_{\text{data}} + \lambda \|\mathbf{w}\|_1 \quad (2.25)$$

This formulation drives a subset of parameters exactly to zero, effectively performing feature selection. The resulting sparse weight structure yields a compact and interpretable model. Although L1 regularization enhances computational efficiency and interpretability, it is less prevalent in deep learning than L2 regularization due to the non-differentiability at zero and its potential to impede convergence stability in large-scale networks.

Dropout

Dropout [133] constitutes a stochastic regularization method specifically designed for deep networks. During training, each neuron is randomly deactivated with a probability p , producing a thinned subnetwork at each iteration:

$$\mathbf{a}^{(l)} = \mathbf{m} \odot f^{(l)}(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}), \quad (2.26)$$

where $\mathbf{m}$ is a binary mask with entries sampled independently from a Bernoulli distribution ($m_i \sim \text{Bernoulli}(1 - p)$), and $\odot$ denotes the element-wise (Hadamard) product. This random masking prevents neurons from co-adapting, compelling each to develop robust, complementary representations. At inference time, dropout is disabled, and activations are scaled by $(1 - p)$ to compensate for the higher neuron count. Conceptually, dropout acts as an implicit ensemble of models sharing parameters, which substantially enhances robustness and generalization to unseen data.

Batch Normalization

Batch Normalization (BN) [134] was initially introduced to mitigate internal covariate shift and accelerate optimization, yet it also functions as a form of regularization. BN normalizes the input activations within each mini-batch:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad (2.27)$$

where μ_B and σ_B^2 denote, respectively, the batch mean and variance:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i, \quad \sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2,$$

and ϵ ensures numerical stability. The normalized activations are subsequently scaled and shifted via learnable affine parameters:

$$y_i = \gamma \hat{x}_i + \beta, \quad (2.28)$$

where γ and β are optimized jointly with the network parameters. BN facilitates the use of higher learning rates, enhances gradient flow, and introduces stochasticity through mini-batch statistics, thereby implicitly regularizing the model. During inference, population-level mean and variance estimates, accumulated during training, are employed for consistent normalization.

Data Augmentation

Data augmentation [135] serves as an implicit regularization mechanism that artificially enlarges the training distribution via label-preserving transformations. For computer vision problems, particularly spacecraft pose estimation, typical augmentations encompass random rotations, translations, scaling, brightness and contrast variations, and noise injection. By augmenting training diversity, the model is compelled to learn invariant and robust features under varied imaging conditions, thereby improving performance generalization across challenging operational environments.

2.1.6 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a specialized architecture designed to process grid-structured data, particularly images [112]. Unlike fully connected networks where each neuron connects to all neurons in the previous layer, CNNs exploit the spatial structure of images through local connectivity and weight sharing. This architectural bias makes CNNs the dominant approach for visual tasks including spacecraft pose estimation from camera imagery. A typical CNN architecture, shown in Figure 2.6, consists of alternating convolutional and pooling layers followed by fully connected layers for final predictions.

The fundamental operation in CNNs is the discrete convolution, which applies learnable filters (also called kernels) to extract local features from the input. For a 2D input image $\mathbf{X}$ and a filter $\mathbf{W}$ of size $k \times k$, the convolution operation at spatial location (i, j) for output channel c is computed as:

$$y_{i,j,c} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \sum_{d=1}^D w_{m,n,d,c} \cdot x_{i+m,j+n,d} + b_c \quad (2.29)$$

where (i, j) are spatial coordinates, d indexes the D input channels, c indexes the output channel, $w_{m,n,d,c}$ are the filter weights, and b_c is the bias term. This operation slides the

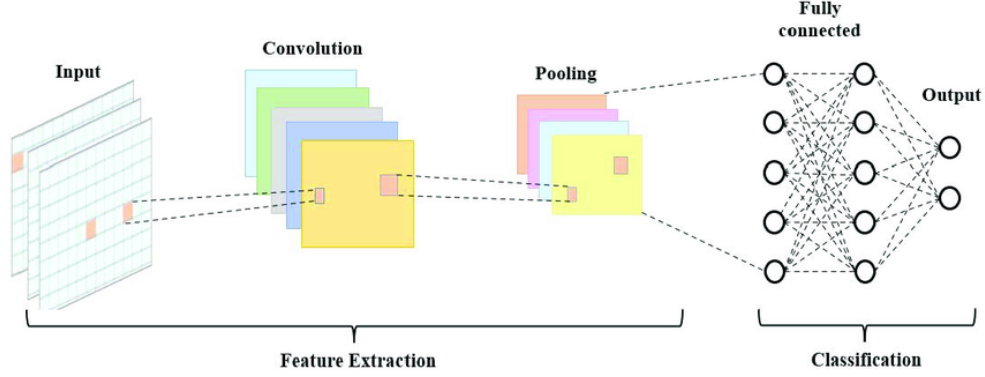


Figure 2.6: Schematic overview of a convolutional neural network (CNN) architecture, illustrating the sequence of feature extraction through convolution and pooling layers followed by classification using fully connected layers [9].

filter across the entire input, computing dot products at each position to produce a feature map. The stride parameter s controls how many pixels the filter moves at each step, while padding can be added to the input borders to control the output spatial dimensions. Figure 2.7 illustrates how a convolution operation is performed with a specific kernel size, stride, and padding configuration.

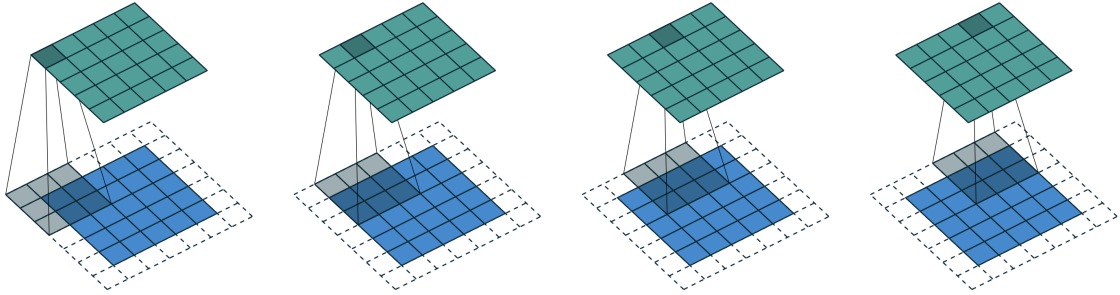


Figure 2.7: A convolution operation using a 3×3 kernel, unit stride, and half padding on a 5×5 input, illustrating how padding and stride influence the output [10].

Multiple filters are applied in parallel to extract different features, creating multiple output channels (feature maps). For an input of size $H \times W \times D$ (height, width, channels) convolved with C filters of size $k \times k$, stride s , and padding p , the output dimensions are:

$$H_{\text{out}} = \left\lfloor \frac{H + 2p - k}{s} \right\rfloor + 1, \quad W_{\text{out}} = \left\lfloor \frac{W + 2p - k}{s} \right\rfloor + 1, \quad C_{\text{out}} = C \quad (2.30)$$

The convolution operation embodies three key principles that make CNNs powerful for visual recognition:

- **Local Connectivity:** Each filter is locally receptive, processing a confined spatial region (receptive field) of the input. In deeper layers, receptive fields expand, en-

abling aggregation of local and global patterns. This progressive hierarchy facilitates abstraction from edges and textures to complex shapes and semantic structures. Figure 2.8 exemplifies hierarchical receptive fields across layers.

- **Parameter Sharing:** Identical filter weights are reused at every spatial location, drastically reducing the total number of parameters relative to fully connected architectures. This enforces translational invariance and encodes the inductive prior that visual features are likely to recur throughout the image domain.
- **Translation Equivariance and Invariance:** The shift-invariant property of convolutions, when combined with pooling, ensures that output activations respond coherently to shifted inputs. This confers robustness to minor spatial transformations and local deformations.

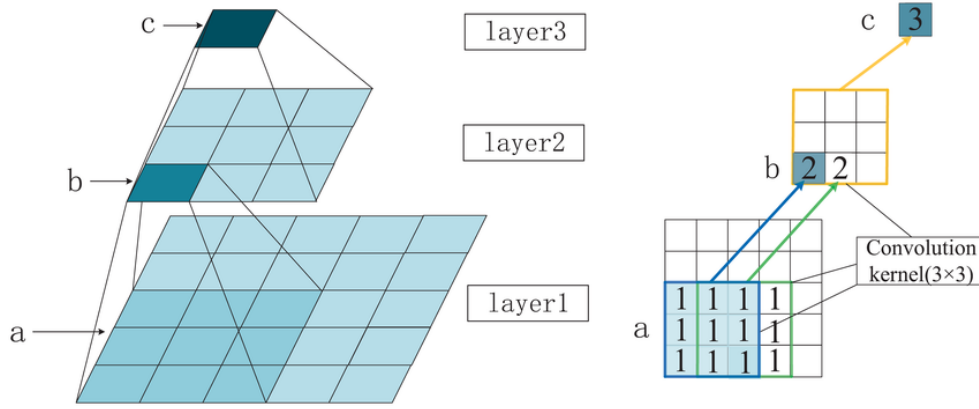


Figure 2.8: Hierarchical expansion of the receptive field across convolutional layers, illustrating how deep networks integrate local and global spatial context [11].

Following the convolutional layers, pooling layers perform spatial downsampling to reduce computational cost, provide limited translation invariance, and increase the receptive field of subsequent layers. The two most common pooling operations shown in Figure 2.9 are:

- **Max Pooling:** Selects the maximum value within each pooling window:

$$y_{i,j} = \max_{(m,n) \in R_{i,j}} x_{m,n} \quad (2.31)$$

This operation emphasizes salient features, introducing robustness to local distortions.

- **Average Pooling:** Computes the mean value over the pooling region:

$$y_{i,j} = \frac{1}{|R_{i,j}|} \sum_{(m,n) \in R_{i,j}} x_{m,n} \quad (2.32)$$

Average pooling yields smoother activations and is commonly employed in global pooling layers for final feature aggregation.

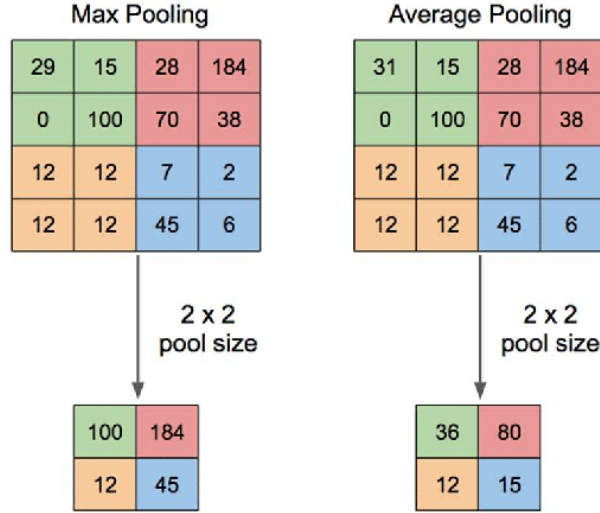


Figure 2.9: Comparison between max pooling and average pooling operations commonly found in convolutional architectures [12].

A typical CNN architecture follows a hierarchical pattern, progressively transforming the input image through multiple stages:

$$\text{INPUT} \rightarrow [\text{CONV} \rightarrow \text{RELU} \rightarrow \text{POOL}]^* \rightarrow \text{FC} \rightarrow \text{OUTPUT} \quad (2.33)$$

where the bracketed block is repeated multiple times. Early convolutional layers, operating on high-resolution feature maps with small receptive fields, learn low-level features such as edges, corners, and textures. Deeper layers, with larger receptive fields operating on spatially downsampled representations, combine these elementary features to recognize complex patterns like object parts and shapes. The fully connected (FC) layers at the end aggregate the spatial information and map it to the desired output (class probabilities or regression values).

For spacecraft pose estimation, CNNs can be employed in several configurations. In direct regression approaches, the CNN outputs pose parameters (translation and rotation) directly. The convolutional layers extract visual features from spacecraft imagery, while fully connected layers map these features to the 6-DOF pose representation. In keypoint-based methods, CNNs detect semantic keypoints on the spacecraft, which are then used to solve the PnP problem for pose estimation.

Notable CNN architectures that have influenced spacecraft pose estimation include LeNet-5 [112], the pioneering CNN for digit recognition; AlexNet [61], which demonstrated

the power of deep CNNs on large-scale image classification; VGGNet [136], emphasizing very deep networks with small filters; and more recent architectures such as ResNet [137] and EfficientNet [1], which will be discussed in detail in subsequent sections. These architectures have established design principles and building blocks that form the foundation of modern pose estimation systems.

2.2 Deep Learning Architectures for Pose Estimation

Deep neural networks employed in this thesis address three critical tasks in spacecraft pose estimation: object detection, keypoint regression, and domain adaptation. For object detection, architectures such as SSD ResNet50 V1 FPN 640×640, EfficientNet-B0, and YOLO are employed to localize spacecraft in images and provide initial ROI extraction. For keypoint regression, EfficientNet serves as the backbone network to directly predict 2D coordinates of semantic spacecraft keypoints, which are subsequently used for pose estimation via geometric algorithms. Finally, for domain adaptation, CycleGAN bridges the synthetic-to-real domain gap by translating rendered spacecraft images to photorealistic appearances, enabling training on abundant synthetic data while maintaining performance on real imagery.

2.2.1 SSD ResNet50 V1 FPN 640×640

The Single Shot MultiBox Detector (SSD) [138] with ResNet50 backbone and Feature Pyramid Network (FPN) [139] combines multi-scale feature extraction with efficient single-stage detection. Operating at 640×640 input resolution, this architecture is employed for spacecraft detection and localization, providing bounding boxes as regions of interest for subsequent pose estimation stages.

The SSD ResNet50 V1 FPN architecture, depicted in Figure 2.10, integrates three key components: the ResNet50 backbone for hierarchical feature extraction, the FPN for multi-scale feature fusion, and SSD detection heads for bounding box prediction at multiple pyramid levels.

ResNet50 Backbone

ResNet50 [137] employs residual learning through skip connections, enabling training of very deep networks. Residual blocks learn:

$$H(\mathbf{x}) = F(\mathbf{x}) + \mathbf{x} \quad (2.34)$$

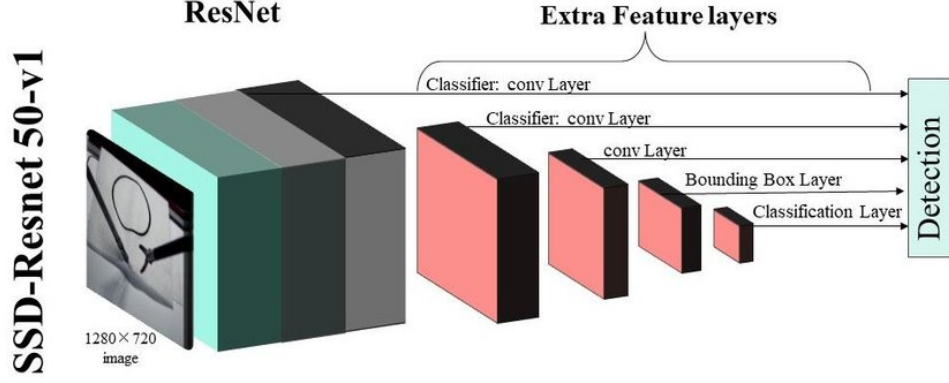


Figure 2.10: SSD ResNet50 V1 FPN architecture [13].

where $F(\mathbf{x})$ is the learned residual and the identity connection $\mathbf{x}$ allows direct gradient flow. The backbone processes input through five stages producing feature maps $\{\mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5\}$ with strides $\{4, 8, 16, 32\}$ at progressively coarser spatial resolutions with increasing semantic content.

Feature Pyramid Network

FPN constructs a multi-scale feature pyramid with lateral connections and top-down pathways:

$$\mathbf{M}_i = \text{Conv}_{1 \times 1}(\mathbf{C}_i) + \text{Upsample}(\mathbf{M}_{i+1}) \quad (2.35)$$

where $\mathbf{C}_i$ are ResNet50 feature maps and $\mathbf{M}_i$ are FPN pyramid levels. This enables multi-scale detection: small distant objects are detected using high-resolution features from early levels ($\mathbf{M}_2, \mathbf{M}_3$), while large close-proximity ones use semantically rich features from deep levels ($\mathbf{M}_4, \mathbf{M}_5$).

SSD Detection Heads

At each FPN level, SSD predicts bounding boxes and class confidences in a single forward pass. For each spatial location, the network predicts offsets $\Delta_{\text{box}} = (\Delta x, \Delta y, \Delta w, \Delta h)$ relative to predefined anchor boxes and class confidences $\mathbf{c} = (c_1, c_2, \dots, c_K)$. Predictions are decoded as:

$$\hat{x} = x_a + w_a \cdot \Delta x, \quad \hat{y} = y_a + h_a \cdot \Delta y, \quad \hat{w} = w_a \cdot \exp(\Delta w), \quad \hat{h} = h_a \cdot \exp(\Delta h) \quad (2.36)$$

where (x_a, y_a, w_a, h_a) are anchor parameters.

Loss Function

The training objective combines localization and classification losses:

$$L = \frac{1}{N}(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)) \quad (2.37)$$

where N is the number of matched anchors, L_{loc} uses Smooth L1 [140] for robust localization, and L_{conf} uses softmax cross-entropy with hard negative mining to address class imbalance.

2.2.2 EfficientNet-B0

EfficientNet, proposed by Tan and Le [1], introduces a principled approach to scaling neural networks that systematically balances network depth, width, and input resolution. While previous architectures typically scaled these dimensions arbitrarily or independently [136, 137], EfficientNet demonstrates that coordinated scaling across all three dimensions yields superior accuracy and efficiency trade-offs. The core innovation of EfficientNet is compound scaling, which uses a compound coefficient ϕ to uniformly scale network depth, width, and resolution:

$$\begin{aligned} \text{depth: } d &= \alpha^\phi \\ \text{width: } w &= \beta^\phi \\ \text{resolution: } r &= \gamma^\phi \end{aligned} \quad (2.38)$$

where α , β , and γ are constrained by:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \quad (2.39)$$

$$\alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1 \quad (2.40)$$

The constraint ensures that for any new ϕ , the total Floating Point Operations per second (FLOPs) increase by approximately 2^ϕ . The squared terms for width and resolution account for the fact that doubling network width increases FLOPs quadratically (both input and output channels double), and doubling resolution increases FLOPs quadratically (pixels increase by a factor of four for 2D images). The scaling coefficients α , β , and γ are determined through a small grid search on the baseline network, and then ϕ is varied to obtain different model variants [1]. For the EfficientNet family, the determined values are approximately $\alpha = 1.2$, $\beta = 1.1$, and $\gamma = 1.15$. The complete scaling progression and corresponding computational cost are summarized in Table 2.1.

The baseline architecture EfficientNet-B0, shown in Figure 2.11, is designed with the objective of maximizing accuracy under a computational budget [1]. The architecture consists of mobile inverted bottleneck convolution (MBConv) blocks, inspired by MobileNetV2 [141]. Each MBConv block incorporates several efficiency-enhancing components:

Table 2.1: EfficientNet family specifications showing compound scaling progression from B0 to B7, with corresponding input resolution, parameters, and computational cost [1].

Model	ϕ	Input Resolution	Parameters	FLOPs
EfficientNet-B0	0	224×224	5.3M	0.39B
EfficientNet-B1	1	240×240	7.8M	0.70B
EfficientNet-B2	2	260×260	9.2M	1.0B
EfficientNet-B3	3	300×300	12M	1.8B
EfficientNet-B4	4	380×380	19M	4.2B
EfficientNet-B5	5	456×456	30M	9.9B
EfficientNet-B6	6	528×528	43M	19B
EfficientNet-B7	7	600×600	66M	37B

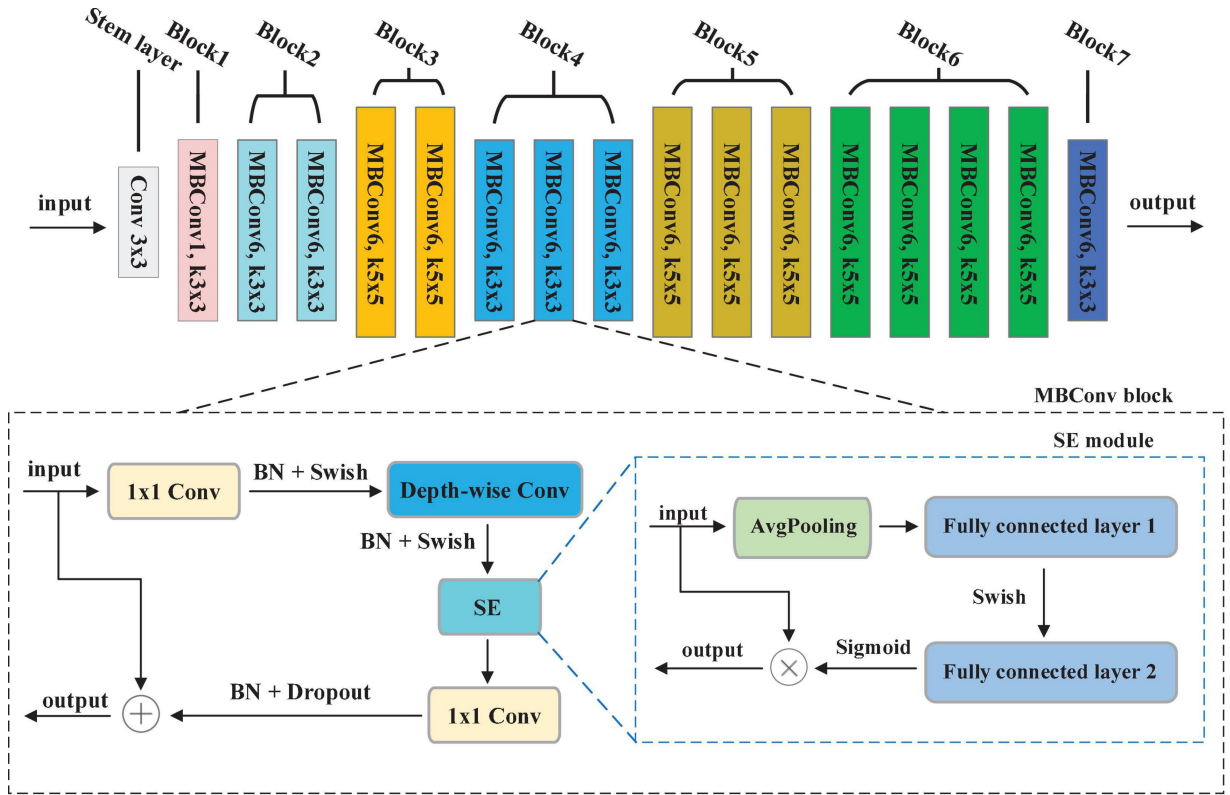


Figure 2.11: EfficientNet-B0 architecture illustrating the baseline network design with Mobile Inverted Bottleneck Convolution (MBCConv) blocks, squeeze-and-excitation modules, and progressive spatial downsampling through nine stages from input to output [14].

- **Depthwise Separable Convolutions:** Standard convolutions are factorized into depthwise and pointwise operations [142]:

$$\begin{aligned}
 \mathbf{z}_{\text{dw}} &= \text{DepthwiseConv}_{k \times k}(\mathbf{x}) \\
 \mathbf{z}_{\text{out}} &= \text{Conv}_{1 \times 1, c_{\text{out}}}(\mathbf{z}_{\text{dw}})
 \end{aligned} \tag{2.41}$$

A standard $k \times k$ convolution with c_{in} input channels and c_{out} output channels requires $k^2 \cdot c_{\text{in}} \cdot c_{\text{out}}$ parameters. Depthwise separable convolutions reduce this to $k^2 \cdot c_{\text{in}} + c_{\text{in}} \cdot c_{\text{out}}$, providing substantial computational savings [142].

- **Inverted Residual Structure:** Unlike ResNet’s bottleneck blocks [137] that compress then expand, MBConv blocks first expand the number of channels by an expansion factor t (typically 6), perform depthwise convolution on the expanded representation, then project back to the original dimensionality [141]:

$$\begin{aligned} \mathbf{z}_1 &= \text{Conv}_{1 \times 1, t \cdot c}(\mathbf{x}) \\ \mathbf{z}_2 &= \text{DepthwiseConv}_{k \times k}(\text{ReLU}(\mathbf{z}_1)) \\ \mathbf{z}_3 &= \text{Conv}_{1 \times 1, c}(\mathbf{z}_2) \\ H(\mathbf{x}) &= \mathbf{x} + \mathbf{z}_3 \quad (\text{if } c_{\text{in}} = c_{\text{out}}) \end{aligned} \tag{2.42}$$

The inverted residual structure maintains the residual connection principle [137] while operating in a higher-dimensional space during feature extraction, which has been shown to preserve information better than conventional bottleneck designs [141].

- **Squeeze-and-Excitation (SE) Blocks:** These channel attention mechanisms [143] adaptively recalibrate channel-wise feature responses:

$$\begin{aligned} \mathbf{s} &= \text{GlobalAvgPool}(\mathbf{z}) \\ \mathbf{w} &= \sigma(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot \mathbf{s})) \\ \tilde{\mathbf{z}} &= \mathbf{w} \odot \mathbf{z} \end{aligned} \tag{2.43}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are learned weight matrices, σ is the sigmoid function, and $\odot$ is element-wise multiplication. The SE block enables the network to emphasize informative features and suppress less useful ones by explicitly modeling interdependencies between channels [143].

The combination of these architectural innovations enables EfficientNet-B0 to achieve state-of-the-art accuracy with significantly fewer parameters and FLOPs compared to previous convolutional architectures [1]. This efficiency makes it particularly suitable for deployment on resource-constrained embedded systems such as spacecraft processors, where computational resources and power budgets are severely limited.

2.2.3 YOLO (You Only Look Once)

YOLO represents a state-of-the-art object detection network that directly predicts bounding boxes and class probabilities from full images in one forward pass [15]. Unlike tra-

ditional object detection approaches that apply classifiers to multiple regions of interest [144, 145], YOLO frames detection as a single regression problem, enabling real-time performance while maintaining competitive accuracy.

The core idea consists in dividing the input image into an $S \times S$ grid [15]. Each grid cell is responsible for detecting objects whose centers fall within that cell. For each grid cell, the network predicts B bounding boxes along with confidence scores and C conditional class probabilities. The confidence score reflects both the probability that the cell contains an object and how accurate the predicted box is:

$$\text{Confidence} = P(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}} \quad (2.44)$$

where IOU (Intersection over Union) measures the overlap between predicted and ground truth bounding boxes. If no object exists in a cell, the confidence should be zero; otherwise, it should equal the IOU.

Each bounding box consists of 5 predictions: $(x, y, w, h, \text{confidence})$, where (x, y) represents the box center coordinates relative to the grid cell bounds, (w, h) are the width and height relative to the entire image. Additionally, each grid cell predicts C conditional class probabilities $P(\text{Class}_i | \text{Object})$. At test time, these probabilities are multiplied by the box confidence to obtain class-specific confidence scores:

$$P(\text{Class}_i | \text{Object}) \times P(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}} = P(\text{Class}_i) \times \text{IOU}_{\text{pred}}^{\text{truth}} \quad (2.45)$$

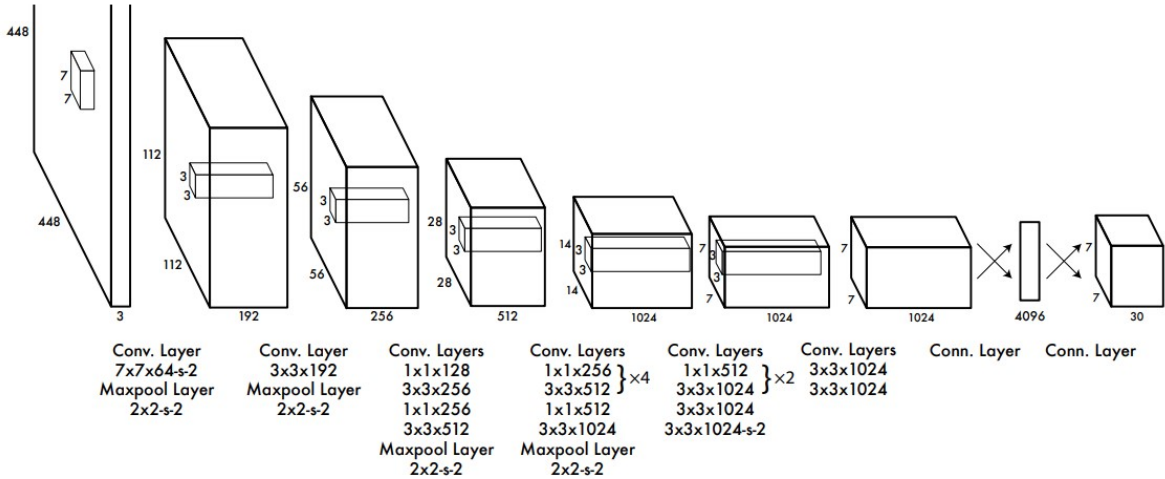


Figure 2.12: YOLO architecture [15].

The network architecture, illustrated in Figure 2.12, processes the input image through a series of convolutional layers inspired by GoogLeNet [146]. The final output is a $S \times S \times (B \times$

$5 + C$) tensor encoding all predictions across the grid. The network is trained end-to-end with a multi-part loss function that balances localization and classification objectives [15]:

$$\begin{aligned}
L = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{K}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{K}_{ij}^{\text{obj}} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{K}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{K}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{K}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{2.46}$$

where the indicator function $\mathbb{K}_{ij}^{\text{obj}}$ denotes the presence of an object in cell i assigned to bounding box j , while $\mathbb{K}_{ij}^{\text{noobj}}$ indicates its absence. The weighting coefficients $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = 0.5$ modulate the relative contributions of different loss components. The square root parameterization applied to bounding box dimensions (w, h) reduces the sensitivity to absolute errors in large boxes, thereby providing scale-invariant penalties. The loss formulation imposes coordinate regression penalties exclusively for grid cells containing objects ($\mathbb{K}_{ij}^{\text{obj}}$) while attenuating confidence penalties for background cells ($\lambda_{\text{noobj}} < 1$), effectively mitigating the class imbalance inherent in object detection tasks where negative samples vastly outnumber positive instances [15].

YOLOv8 & YOLOv11

The YOLO architecture has undergone continuous advancement since its inception, with each generation introducing architectural and algorithmic innovations to improve detection accuracy and computational efficiency [147, 148, 149]. Early YOLO versions laid the foundation for real-time, unified single-stage object detection. More recent developments emphasize anchor-free design principles, efficient backbone structures, and task-specific optimization strategies, culminating in the state-of-the-art models YOLOv8 and YOLOv11. Table 2.2 presents a comparative evaluation of YOLOv8 and YOLOv11 model variants on the COCO [2] dataset, illustrating the accuracy-speed trade-offs achievable across different model sizes.

Table 2.2: Performance comparison of YOLOv8 and YOLOv11 model variants at 640×640 input resolution, reporting accuracy-speed trade-offs across multiple model scales. All models are pre-trained on the COCO dataset [2] and evaluated on the validation set [3].

Model	Input (px)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed GPU TensorRT (ms)	Params (M)	FLOPs (B)
<i>YOLOv8 (A100 GPU)</i>						
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8
<i>YOLOv11 (T4 GPU)</i>						
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

2.2.4 CycleGAN

CycleGAN (Cycle-Consistent Generative Adversarial Network) is an unsupervised image-to-image translation framework introduced by Zhu et al. [16] that addresses the challenge of learning mappings between two visual domains. Unlike traditional supervised approaches that rely on aligned image pairs [150], CycleGAN learns to translate images from a source domain X to a target domain Y using only unpaired samples from each domain, making it particularly valuable for domain adaptation tasks.

Architecture

The CycleGAN architecture, illustrated in Figure 2.13, comprises two generator networks and two discriminator networks operating in a symmetric configuration. The generators $G : X \rightarrow Y$ and $F : Y \rightarrow X$ learn bidirectional mappings between domains, while discriminators D_X and D_Y distinguish between real and synthetically generated images in their respective domains. This bidirectional architecture enables the model to discover meaningful cross-domain correspondences without requiring paired training examples.

The generator G transforms images from source domain X to target domain Y , producing synthetic images $G(x)$ that are indistinguishable from authentic samples in Y . Conversely, generator F performs the inverse transformation from Y to X . The discriminators are trained adversarially to classify images as real or generated, thereby encouraging the generators to produce increasingly realistic outputs that conform to the target domain.

distribution [16].

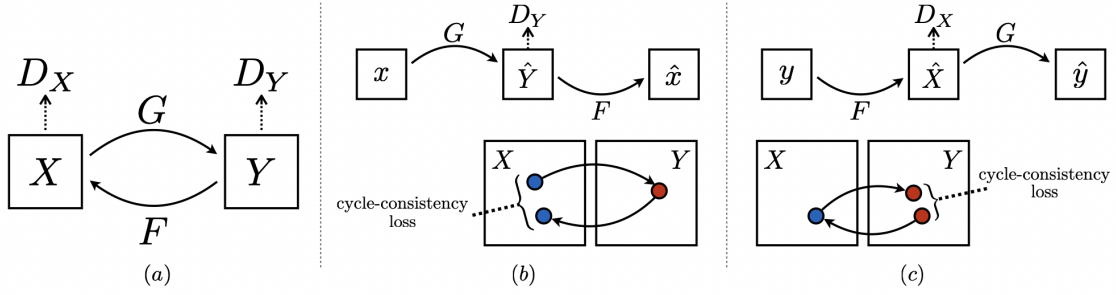


Figure 2.13: CycleGAN architecture showing the bidirectional generators G and F , discriminators D_X and D_Y , and the cycle consistency constraint that enforces $F(G(x)) \approx x$ and $G(F(y)) \approx y$ [16].

CycleGAN optimizes a composite objective function consisting of adversarial losses and cycle consistency losses to achieve effective unsupervised domain translation.

Adversarial Loss

The adversarial loss enforces that the generated images are statistically indistinguishable from real images in the target domain. For the forward mapping $G : X \rightarrow Y$ paired with discriminator D_Y , the adversarial objective follows the standard GAN formulation [151]:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))] \quad (2.47)$$

where the first term maximizes the discriminator's ability to correctly identify real images from domain Y , while the second term penalizes the discriminator when it successfully detects generated images $G(x)$ as fake. The generator G seeks to minimize this loss, while discriminator D_Y seeks to maximize it, establishing the adversarial minimax game. A symmetric adversarial loss $\mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$ governs the reverse mapping $F : Y \rightarrow X$ and its corresponding discriminator D_X .

Cycle Consistency Loss

The cycle consistency constraint constitutes the key innovation of CycleGAN, addressing the ill-posed nature of unsupervised domain translation by enforcing structural coherence in the learned mappings. Specifically, this constraint requires that sequentially applying both generators in a cycle should reconstruct the original input, thereby preventing mode

collapse and ensuring that semantic content is preserved during translation. The cycle consistency loss is expressed as:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1] \quad (2.48)$$

The first term enforces forward-backward cycle consistency, requiring that an image x from domain X , when translated to domain Y via G and subsequently back to domain X via F , should approximately recover the original input: $F(G(x)) \approx x$. The second term symmetrically enforces backward-forward cycle consistency: $G(F(y)) \approx y$. The L_1 norm is employed to encourage sharp reconstructions and discourage blurring artifacts [16].

Complete Objective

The complete training objective integrates both adversarial losses for realistic appearance and cycle consistency losses for structural preservation:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F) \quad (2.49)$$

where the hyperparameter λ controls the relative importance of cycle consistency relative to adversarial objectives, typically set to $\lambda = 10$ [16]. The generators and discriminators are optimized through the following minimax formulation [151]:

$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \quad (2.50)$$

This represents a two-player game where the generators (G, F) minimize the objective while the discriminators (D_X, D_Y) maximize it. In practice, this minimax optimization is implemented through alternating gradient descent, wherein discriminator and generator parameters are updated separately in each training iteration.

Relevance to Domain Adaptation

Models trained on extensive datasets generated by CycleGAN exhibit improved generalization in the real-world scenarios. In the context of spacecraft pose estimation, CycleGAN can bridge the domain gap between computer-generated spacecraft renderings and actual satellite imagery captured in space, where lighting conditions, sensor characteristics, and environmental factors differ significantly from simulation.

2.3 Uncertainty Estimation in Deep Learning

Uncertainty quantification is a critical component of reliable DL systems. Traditional neural networks provide point predictions without any measure of confidence, making it difficult to assess prediction reliability and identify potential failure cases. Uncertainty estimation enables models to express when they are uncertain about their predictions, allowing downstream systems to make informed decisions about whether to trust a prediction or request human intervention.

2.3.1 Types of Uncertainty

Uncertainty in deep learning can be decomposed into two fundamental categories as described below.

Aleatoric Uncertainty

Aleatoric uncertainty, also known as data uncertainty or irreducible uncertainty, captures the inherent noise and ambiguity in the data itself. This type of uncertainty arises from factors such as sensor noise, occlusions, motion blur, or inherent ambiguity in the input. Aleatoric uncertainty cannot be reduced by collecting more training data, as it represents the fundamental limitations of the observation process [30].

Epistemic Uncertainty

Epistemic uncertainty, also referred to as model uncertainty or reducible uncertainty, captures the uncertainty in the model parameters themselves. This uncertainty arises from limited training data and imperfect model capacity. Unlike aleatoric uncertainty, epistemic uncertainty can be reduced by observing more training data or using more expressive model architectures. In domain adaptation scenarios, epistemic uncertainty is particularly important as it tends to be higher for out-of-distribution samples, indicating when the model encounters inputs that differ from its training distribution [30].

2.3.2 Methods for Uncertainty Estimation

Several approaches have been developed to estimate uncertainty in deep neural networks.

Bayesian Neural Networks

Bayesian neural networks (BNNs) [95] provide a principled probabilistic framework for uncertainty estimation by placing probability distributions over network weights rather

than using point estimates. Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, BNNs maintain a posterior distribution over weights $p(w|\mathcal{D})$ and make predictions by marginalizing over this posterior:

$$p(y|x, \mathcal{D}) = \int p(y|x, w)p(w|\mathcal{D})dw \quad (2.51)$$

While theoretically elegant, exact inference in BNNs is intractable for modern deep networks, necessitating approximate inference techniques.

Monte Carlo Dropout

Monte Carlo Dropout (MC Dropout) [152] provides a practical approximation to Bayesian inference by interpreting dropout as a variational Bayesian approximation. By applying dropout at test time and performing multiple forward passes with different dropout masks, MC Dropout generates a distribution of predictions:

$$p(y|x, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^T p(y|x, w_t) \quad (2.52)$$

where w_t represents the network weights with dropout mask applied at iteration t . The variance of these predictions provides an estimate of epistemic uncertainty, while the predictive entropy captures total uncertainty.

Deep Ensembles

Deep ensembles [153] train multiple neural networks with different random initializations and combine their predictions to estimate uncertainty. For M ensemble members with parameters $\{w_1, w_2, \dots, w_M\}$, the ensemble prediction is:

$$p(y|x) \approx \frac{1}{M} \sum_{m=1}^M p(y|x, w_m) \quad (2.53)$$

Deep ensembles have been shown to provide well-calibrated uncertainty estimates and are particularly effective at capturing epistemic uncertainty. The disagreement between ensemble members indicates regions where the model is uncertain.

Evidential Deep Learning

Evidential deep learning [154] directly parameterizes a higher-order distribution over the predictive distribution, enabling single-pass uncertainty estimation. For classification tasks, the network outputs parameters of a Dirichlet distribution, while for regression, it outputs parameters of a Normal-Inverse-Gamma distribution. This approach provides both

aleatoric and epistemic uncertainty estimates without requiring multiple forward passes or ensemble training.

2.3.3 Uncertainty Quantification for Pose Estimation

For pose estimation tasks, uncertainty can be expressed at different levels:

- **Keypoint-level uncertainty:** Individual 2D or 3D keypoint predictions can be associated with uncertainty estimates, indicating confidence in specific feature localizations.
- **Pose-level uncertainty:** The final 6-DoF pose (3D position and 3D orientation) can be represented with associated uncertainty, typically as covariance matrices for position and orientation separately.
- **Multi-modal uncertainty:** Pose estimation may admit multiple plausible solutions due to symmetries or ambiguities, requiring mixture models or probabilistic representations.

2.4 Model Optimization for Embedded Hardware

Deploying DL models on embedded systems presents significant challenges due to strict constraints on computational resources, power consumption, memory bandwidth, and physical size. In spacecraft applications, these constraints are particularly severe, as onboard computers must operate in harsh space environments with limited power budgets, radiation-hardened processors with lower computational capabilities, and stringent reliability requirements. This section describes the principal optimization techniques that facilitate the deployment of DL models on resource-constrained hardware.

2.4.1 Quantization

Quantization is a widely used technique for reducing model size and accelerating inference by converting weights and activations from high-precision formats, such as 32-bit floating-point (FP32), to low-precision representations, most commonly 8-bit integers (INT8) [155]. While the strict definition of quantization refers to mapping continuous values to discrete integer levels, modern deployment pipelines often also utilize mixed-precision computation, where 16-bit floating-point (FP16) formats are employed to achieve a favorable balance between computational efficiency and retained model accuracy. The quantization process can be formalized as:

$$x_q = \text{round} \left(\frac{x}{S} \right) + Z \quad (2.54)$$

where x is the original floating-point value, x_q is the quantized integer value, S and Z are scale and zero-point quantization parameters. The dequantization process reverses this mapping:

$$x \approx S \cdot (x_q - Z) \quad (2.55)$$

Several quantization strategies are known, each offering distinct trade-offs in terms of computational efficiency and model accuracy, as outlined below:

- **Post-Training Quantization (PTQ):** PTQ [156] applies quantization to a pre-trained model without requiring retraining. This approach is computationally efficient but may result in accuracy degradation, particularly for aggressive quantization (e.g., INT4 or lower). PTQ typically requires a small calibration dataset to determine optimal scale factors and zero-points.
- **Quantization-Aware Training (QAT):** QAT [157] simulates the effects of quantization during training by inserting fake quantization operations in the forward pass. The backward pass uses the straight-through estimator to approximate gradients through the non-differentiable quantization operation:

$$\frac{\partial \mathcal{L}}{\partial x} \approx \frac{\partial \mathcal{L}}{\partial x_q} \quad (2.56)$$

QAT generally achieves better accuracy than PTQ, especially for low-bit quantization, at the cost of requiring model retraining.

- **Mixed-Precision Quantization:** Different layers or operations within a network may have varying sensitivity to quantization. Mixed-precision quantization assigns different bit-widths to different parts of the network based on their importance [158]. Sensitive layers (e.g., first and last layers) may retain higher precision, while less critical layers use aggressive quantization.

Quantization offers several practical benefits for embedded systems. First, it enables reduction in model size due to the lowered inference precision. This reduction in precision also decreases memory bandwidth requirements, as less data must be transferred between memory and compute units. Additionally, integer arithmetic operations are inherently faster and more energy-efficient than floating-point computations, resulting in improved computational speed. Finally, lower precision operations consume less power, which is especially important for battery-powered or solar-powered spacecraft systems.

2.4.2 Pruning Strategies

Pruning is a widely adopted and rigorously studied methodology for reducing the computational and memory footprints of DNNs, enabling efficient deployment on resource-constrained hardware while maintaining predictive performance. By systematically eliminating parameters deemed unimportant, pruning yields sparse models that offer substantial gains in inference efficiency and often improved generalization.

The leading strategies for pruning include [159, 160]:

- **Magnitude-based pruning:** Removes weights with the smallest absolute values under the assumption that such connections contribute least to model output. This approach is simple, hardware-agnostic, and effective for structured and unstructured sparsity.
- **Gradient-based pruning:** Assigns importance according to the gradient of the loss with respect to each parameter, targeting those whose removal minimally perturbs the loss landscape.
- **Iterative pruning:** Alternates cycles of pruning and fine-tuning to enable gradual adaptation to increasing sparsity. This sequential approach typically preserves accuracy better than aggressive, one-shot pruning.

The practical effectiveness of any pruning scheme is contingent upon both the inherent architecture and the degree of sparsity introduced. Figure 2.14 visualizes both unstructured and structured pruning paradigms, highlighting the distinction between arbitrary removal of parameters and the elimination of entire computational units.

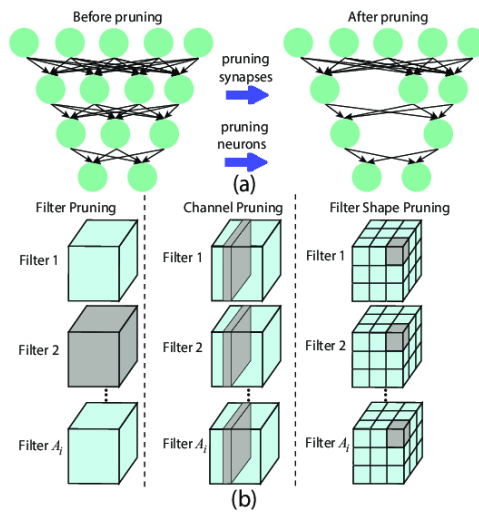


Figure 2.14: Illustration of pruning methods: (a) Non-structured weight pruning; (b) Three types of structured pruning [17].

2.4.3 Deployment Frameworks

A variety of specialized frameworks have been developed to facilitate the efficient deployment of neural networks on embedded and resource-constrained hardware. The following tools are among the most widely adopted for optimizing inference performance and enabling cross-platform compatibility:

- **TensorRT:** NVIDIA’s inference optimization framework offers advanced features such as automatic quantization, layer fusion, kernel auto-tuning, and support for mixed-precision inference. It is specifically designed for NVIDIA GPUs and embedded platforms, including the Jetson family [161].
- **TensorFlow Lite:** Tailored for mobile and embedded devices, TensorFlow Lite provides model quantization, pruning capabilities, and a suite of optimized kernels for ARM processors and various hardware accelerators [162].
- **ONNX Runtime:** This cross-platform inference engine supports a wide range of hardware backends and incorporates graph optimizations and quantization techniques to maximize performance and portability [163].
- **OpenVINO:** Intel’s toolkit is designed for optimizing and deploying DL models on Intel hardware, including CPUs, integrated GPUs, and Vision Processing Units (VPUs), enabling efficient inference across diverse edge devices [164].

2.5 Perspective-n-Point Problem and Solvers

This section describes the mathematical formulation of the PnP problem for spacecraft pose estimation, and presents the EPnP algorithm together with RANSAC framework used in this work.

2.5.1 PnP Problem Formulation

The Perspective- n -Point (PnP) [165] problem shown in Figure 2.15 aims to estimate the relative pose, comprising the rotation matrix $\mathbf{R}_{BC}$ and translation vector $\mathbf{t}_{BC}$, between a calibrated camera and an object, given a set of n 3D points $\mathbf{q}_{i,B} = [x_i, y_i, z_i]^T$ defined in the body-fixed frame B , and their corresponding image projections $\mathbf{p}_i = [u_i, v_i, 1]^T$ in the camera frame. The geometric relationship is given by:

$$\mathbf{r}_C = \mathbf{R}_{BC}\mathbf{q}_{i,B} + \mathbf{t}_{BC}, \quad \mathbf{p}_i = \begin{bmatrix} \frac{x_C}{z_C}f_x + c_x \\ \frac{y_C}{z_C}f_y + c_y \\ 1 \end{bmatrix}$$

where $\mathbf{r}_C = [x_C, y_C, z_C]^T$ denotes the coordinates of the target point in the camera reference frame, f_x and f_y are the focal lengths in pixels, and (c_x, c_y) are the pixel coordinates of the camera's principal point (optical center).

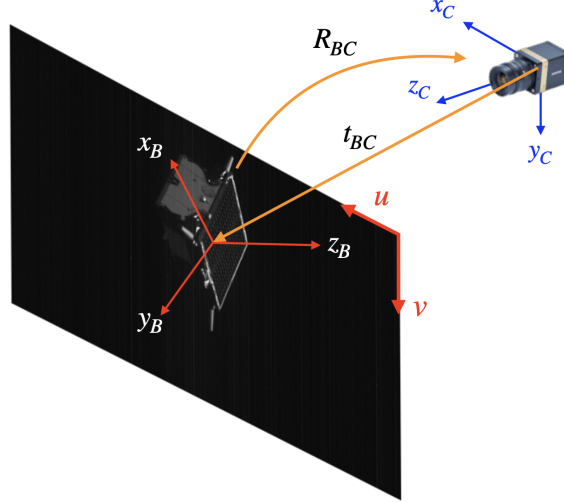


Figure 2.15: Visualization showing the geometric setup of the PnP problem. The transformation from the body-fixed frame B to the camera frame C is parameterized by the rotation matrix $\mathbf{R}_{BC}$ and translation vector $\mathbf{t}_{BC}$. [18]

This mapping can be compactly expressed in homogeneous coordinates as:

$$\begin{bmatrix} w_i u_i \\ w_i v_i \\ w_i \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} \mathbf{R}_{BC} & \mathbf{t}_{BC} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \quad (2.57)$$

Equation (2.57) describes the transformation from 3D object coordinates to 2D image points using the intrinsic camera matrix $\mathbf{K}$ and the extrinsic parameters $(\mathbf{R}_{BC}, \mathbf{t}_{BC})$. Six unknowns (three for rotation, three for translation) must be determined. While three non-collinear correspondences ($n = 3$) are theoretically sufficient, this minimal configuration can yield ambiguous solutions, especially for symmetric objects. To address these ambiguities, a variety of iterative and non-iterative PnP algorithms have been proposed [166, 167, 168, 103].

2.5.2 Efficient Perspective-n-Point (EPnP)

The EPnP algorithm provides a computationally efficient, non-iterative solution for pose estimation when at least four 3D–2D correspondences are given [103]. In EPnP, each 3D

point is expressed as a weighted sum of four virtual control points:

$$\mathbf{X}_{B,i} = \sum_{j=1}^4 \alpha_{ij} \mathbf{C}_j, \quad \sum_{j=1}^4 \alpha_{ij} = 1$$

where $\mathbf{C}_j$ are the control points in the object (body) frame, and α_{ij} are their barycentric coordinates. The same relationship holds in the camera frame:

$$\mathbf{X}_{C,i} = \sum_{j=1}^4 \alpha_{ij} \mathbf{C}_{C,j}$$

Once the control points $\mathbf{C}_{C,j}$ are estimated from the 3D–2D correspondences, the relative pose $(\mathbf{R}_{BC}, \mathbf{t}_{BC})$ is recovered by aligning the control points between the body and camera frames:

$$\mathbf{C}_{C,j} = \mathbf{R}_{BC} \mathbf{C}_j + \mathbf{t}_{BC}$$

EPnP achieves linear complexity in the number of correspondences, making it particularly attractive for dense keypoint regression scenarios [103]. However, EPnP assumes that all correspondences are inliers.

2.5.3 Random Sample Consensus (RANSAC)

To mitigate the influence of outliers among correspondences, the RANSAC algorithm is commonly used for robust pose estimation [40]. RANSAC repeatedly samples minimal sets of correspondences (four points for EPnP), computes candidate pose solutions, and evaluates each by counting inliers based on the reprojection error:

$$e_i = \left\| \mathbf{x}_{C,\text{img},i} - \pi \left(\mathbf{K}[\mathbf{R}_{BC}^{(k)} \mid \mathbf{t}_{BC}^{(k)}] \tilde{\mathbf{X}}_{B,i} \right) \right\|$$

where $\pi(\cdot)$ is the camera projection function, and an inlier is defined as $e_i < \tau$. The required number of RANSAC iterations K for a desired confidence level p and inlier ratio ϵ is:

$$K = \frac{\log(1 - p)}{\log(1 - \epsilon^4)}$$

Once the consensus set is identified, the final pose is refined using all inliers, typically with EPnP or further nonlinear optimization.

2.6 Evaluation Metrics

This section presents the evaluation metrics adopted to assess the performance of the DL-based pose estimation pipeline, which comprises object detection, keypoint regression and

pose estimation. For domain adaptation, the Fréchet Inception Distance (FID) [169] score is used to quantify the similarity between the distributions of translated and real target images, providing an objective measure of image realism and domain alignment.

2.6.1 Object Detection

The performance of object detection networks is evaluated using standard metrics that quantify localization accuracy [19].

Intersection over Union (IoU)

The IoU quantifies the degree of spatial overlap between a predicted bounding box B_p and the corresponding ground-truth box B_{gt} . It is defined as the ratio between the area of their intersection and the area of their union:

$$\text{IoU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|}. \quad (2.58)$$

An IoU value of 1 indicates perfect overlap, whereas a value of 0 indicates no overlap. Visual representation of the IoU formula is shown in Figure 2.16. To assess consistency across a dataset containing N samples, the mean and median IoU values can be computed.

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$

Figure 2.16: Illustration of the Intersection over Union (IoU) metric [19].

Average Precision (AP)

The AP metric measures the area under the precision–recall curve (PRC), summarizing the trade-off between detection precision and recall for a given IoU threshold t . Precision P and recall R are defined as:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad (2.59)$$

where TP , FP , and FN denote true positives, false positives, and false negatives, respectively. The Average Precision for a given IoU threshold t is then computed as the integral over the precision–recall curve:

$$AP_t = \int_0^1 P_t(R) dR, \quad (2.60)$$

where $P_t(R)$ represents the precision as a function of recall at threshold t . Commonly reported AP values include:

$$\{\text{AP@0.5}, \text{AP@0.75}, \text{AP@0.95}\}$$

which correspond to IoU thresholds of 0.5, 0.75, and 0.95, respectively, reflecting progressively stricter localization requirements.

Mean Average Precision (mAP@0.5:0.95)

The mAP is computed as the mean of Average Precision (AP) values across all object classes and multiple IoU thresholds. Specifically, mAP@0.5:0.95 averages AP over IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05:

$$\text{mAP@0.5:0.95} = \frac{1}{T} \sum_{t \in \mathcal{T}} \left(\frac{1}{C} \sum_{c=1}^C \text{AP}_{c,t} \right), \quad (2.61)$$

where $\mathcal{T} = \{0.5, 0.55, \dots, 0.95\}$, $T = |\mathcal{T}| = 10$ is the number of IoU thresholds, C is the number of object classes, and $\text{AP}_{c,t}$ denotes the Average Precision for class c at IoU threshold t . For single-class detection tasks, such as spacecraft or marker detection in this work, the class averaging simplifies to:

$$\text{mAP@0.5:0.95} = \frac{1}{T} \sum_{t \in \mathcal{T}} \text{AP}_t. \quad (2.62)$$

This metric provides a comprehensive measure of detection quality over varying degrees of localization strictness.

2.6.2 Keypoint Regression

Keypoint regression performance is evaluated using two complementary metrics that characterize localization accuracy at the pixel level: the Mean Root Mean Square Error (Mean RMSE) and the Percentage of Keypoints Below RMSE Threshold (PRMSE). These metrics jointly provide a detailed assessment of the model's ability to predict precise keypoint coordinates within the 2D image plane.

Mean Root Mean Square Error (Mean RMSE)

The Mean RMSE quantifies the average per-sample localization error between predicted and ground-truth keypoints. For each image, the RMSE is first computed across all keypoints, and these per-image values are then averaged over all samples:

$$\text{Mean RMSE} = \frac{1}{M} \sum_{m=1}^M \sqrt{\frac{1}{N} \sum_{i=1}^N \left\| \mathbf{p}_{i,m}^{\text{pred}} - \mathbf{p}_{i,m}^{\text{gt}} \right\|_2^2},$$

where M denotes the number of samples, N the number of keypoints per sample, and $\mathbf{p}_{i,m} = (x_{i,m}, y_{i,m})$ represents the 2D pixel coordinates of the i -th keypoint in the m -th image. This formulation ensures that each sample contributes equally to the final metric, providing a robust measure of average prediction quality across the dataset. Lower Mean RMSE values correspond to higher keypoint localization precision, with the squared error formulation emphasizing the impact of larger localization failures.

Percentage of keypoints Below RMSE Threshold (PRMSE@ τ)

The PRMSE at threshold τ is defined as

$$\text{PRMSE@}\tau = \frac{\#\{m \mid \text{RMSE} < \tau\}}{M},$$

that is, the number of data samples with RMSE below τ divided by the total number of samples M . The metric ranges from 0 to 1, with higher values indicating that a larger fraction of samples achieve at least the specified level of keypoint localization accuracy.

2.6.3 Pose Estimation

Pose estimation performance is evaluated following the metrics established by the Satellite Pose Estimation Competition 2021 (SPEC2021) [59], which defines standardized criteria for assessing spacecraft pose accuracy using the SPEED+ dataset [5].

Competition Metrics

For a single image the translation error (e_t) and rotation error (e_q) are defined as

$$e_t = \|\hat{t} - t\| \tag{2.63}$$

$$e_q = 2 \arccos(|\langle \hat{q}, q \rangle|) \tag{2.64}$$

where $(\hat{t}, \hat{q})$ and (t, q) are respectively the ground-truth and estimated translation and unit quaternion vectors, and $\langle \cdot, \cdot \rangle$ denotes a vector inner-product. Additionally, the normalized translation error ($\tilde{e}_t$) is defined as

$$\tilde{e}_t = e_t / \|t\| \tag{2.65}$$

which normalizes the translation error by the magnitude of ground-truth distance. Then, for a single sample, the pose error (e_{pose}) is given as

$$e_{\text{pose}} = e_q + \tilde{e}_t, \quad (2.66)$$

which is equivalent to the sum of rotation error in radians and the position error normalized by the ground-truth distance. Finally, for a test domain with N samples, the mean translation error (E_t), mean rotation error (E_q), and mean pose error (E_{pose}) are given as

$$E_t = \frac{1}{N} \sum_{i=1}^N e_t^{(i)}, \quad (2.67)$$

$$E_q = \frac{1}{N} \sum_{i=1}^N e_q^{(i)}, \quad (2.68)$$

$$E_{\text{pose}} = \frac{1}{N} \sum_{i=1}^N e_{\text{pose}}^{(i)}. \quad (2.69)$$

HIL Pose Error

Specifically, E_{pose} , also known as the SPEED score, served as the official metric of SPEC2019, as it effectively balanced the contributions of both position and rotation errors within a single measure.

In contrast to SPEC2019, which evaluated submissions using computer-generated synthetic images, SPEC2021 assesses pose estimates on HIL domain images, whose pose labels are derived from a calibrated robotic testbed. Consequently, the accuracy of these pose labels must be considered. The expected rotation and normalized position errors from calibration are $\theta_q = 0.169^\circ$ and $\theta_t = 2.173$ mm/m, respectively.

Thus, an estimated rotation or normalized translation is considered perfect when it falls below these thresholds, where the HIL rotation error (e_q^*) and HIL normalized translation error ($\tilde{e}_t^*$) are defined as

$$e_q^* = \begin{cases} 0 & \text{if } e_q < \theta_q \\ e_q & \text{otherwise,} \end{cases} \quad \tilde{e}_t^* = \begin{cases} 0 & \text{if } \tilde{e}_t < \theta_t \\ \tilde{e}_t & \text{otherwise} \end{cases} \quad (2.70)$$

Then, the HIL pose error (e_{pose}^*) for a single data is simply given as

$$e_{\text{pose}}^* = e_q^* + \tilde{e}_t^*, \quad (2.71)$$

and the official metric for SPEC2021, the mean HIL pose error (E_{pose}^*) for the HIL test domain with N samples, is given as

$$E_{\text{pose}}^* = \frac{1}{N} \sum_{i=1}^N (e_{\text{pose}}^*)^{(i)}. \quad (2.72)$$

This metric ensures that only significant errors above the calibration precision contribute to the final score, enabling robust and fair comparison of pose estimation methods. Lower values indicate better overall pose estimation accuracy.

2.6.4 Domain Adaptation

To quantitatively assess the quality of the translated images by CycleGAN and the effectiveness of domain adaptation, the Fréchet Inception Distance (FID) score [169] was employed as a primary evaluation metric.

The FID score measures the similarity between the distribution of generated images and real target domain images in a deep feature space. Lower FID values indicate that the generated images are more similar to the real target images in terms of both visual quality and diversity. Mathematically, the FID score is defined as:

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr} \left(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2} \right)$$

where μ_r and Σ_r are the mean and covariance of the feature representations of real target images, and μ_g and Σ_g are the mean and covariance of the feature representations of the generated images. The features are typically extracted from a pretrained Inception neural network.

The FID score provides an objective, widely accepted measure of the realism and diversity of images produced by generative models, and is particularly useful for evaluating the success of domain adaptation in unpaired image translation tasks.

Chapter 3

Pose estimation for the EROSS mission

This chapter presents the development and validation procedures for DL-based pose estimation pipelines within the framework of the EROSS mission. The activities were carried out during a PhD co-location period at Thales Alenia Space Italia (TASI). Section 3.1 introduces the mission, detailing its technical objectives and architectural context. Section 3.2 then describes the synthetic image dataset acquisition methodology employed in this work. Section 3.3 examines the design and implementation of the pose estimation pipeline, which is tailored to the specific operational requirements of the pre-capture phase in the EROSS mission. The subsequent section, 3.4, addresses the validation and deployment of the developed pipeline on embedded hardware platform provided by the industrial partner. The chapter concludes with a summary of key findings and future prospects in section 3.5.

Due to obligations under a Non-Disclosure Agreement between the University of Bologna and TASI, the dataset images themselves are not publicly available; for illustration purposes, images of the Euclid spacecraft [170] are included where appropriate.

3.1 EROSS mission overview

The European Robotic Orbital Support Services (EROSS) mission constitutes a pioneering European initiative aimed at demonstrating advanced on-orbit servicing capabilities, with a specific focus on the capture, refueling, and technological enhancement of client



Figure 3.1: Artistic rendering of the EROSS spacecraft during close-range proximity operations, illustrating the robotic servicing arm and modular spacecraft components in orbit [20].

satellites within a LEO environment [171]. This demonstration mission seeks to validate, in operational conditions, a comprehensive suite of robotic and visual navigation technologies integral to future commercial orbital service operations, including satellite life extension, end-of-life management, and in-orbit upgrades. Central to the EROSS mission is the objective of advancing the maturity of rendezvous and autonomous robotic systems, which are essential for the sustainable and economically viable utilization of space assets. The project architecture involves a dedicated servicer satellite equipped with a dexterous robotic arm (see Figure 3.1), capable of executing both cooperative and non-cooperative capture maneuvers, payload manipulation, and precise servicing actions under visual guidance and pose estimation frameworks [172]. Furthermore, the project serves as a strategic platform to test the integration of modular and reconfigurable interfaces — key enablers for future space ecosystem adaptability and commercial exploitation. Through the systematic validation of core technologies in both ground-ground and simulated orbital scenarios, EROSS sets the foundation for a paradigm shift towards intelligent, adaptive, and reusable orbital infrastructure, firmly positioning European space stakeholders at the forefront of the evolving space economy [173].

The EROSS timeline illustrated on Figure 3.2 presents the sequence of key phases and milestones for a space mission development project between 2021 and 2026. The process is

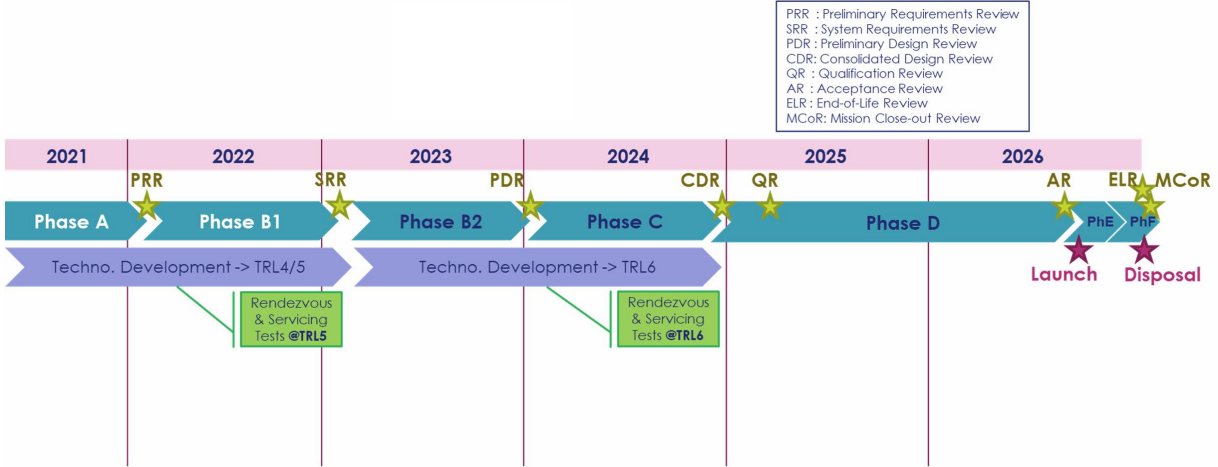


Figure 3.2: Development schedule for the EROSS mission [20].

split into distinct, color-coded phases — labeled A, B1, B2, C, and D — each corresponding to progressive steps of technical development and system readiness. Every phase is marked by formal technical reviews designed to assess requirements, design maturity, and qualification status, such as the Preliminary Requirements Review, System Requirements Review, Preliminary Design Review, Consolidated Design Review, Qualification Review, and Acceptance Review. During the initial phases, technology development advances from technology readiness level 4 to 5, and later to level 6, with specialized testing campaigns for rendezvous and servicing taking place at each stage. As the project evolves, further comprehensive reviews and tests are scheduled, ultimately leading to mission acceptance, launch, end-of-life review, and final mission close-out.

During the development of the pose estimation pipeline the project’s position was in Phase B2, where only synthetic datasets were available for the validation.

3.2 EROSS dataset

The synthetic datasets used for training and validation of the EROSS pose estimation pipeline algorithms were obtained from two distinct sources. The first dataset, featuring spacecraft poses randomly distributed against an Earth background, was generated using Blender to validate a hybrid three-stage approach adapted from the current literature [174]. The second dataset, consisting of trajectories produced in Unreal Engine 5 and provided by TASI, was used to train and validate the pose estimation pipeline in very close-range scenarios where the target partially leaves the camera’s FOV.

3.2.1 EROSS dataset with random spacecraft poses (Blender)

Blender is a free and open-source 3D computer graphics suite renowned for its extensive capabilities in modeling, texturing, rendering, and animation across multiple platforms, including Windows, Linux, and macOS. It is widely used in scientific visualization, animation, and engineering applications due to its physically based rendering engine (Cycles), robust Python scripting interface, and flexibility for custom workflows.

For the present study, Blender was utilized to generate a synthetic dataset featuring 10,000 images of a spacecraft in randomly distributed poses. The Cycles renderer, which employs ray tracing for high-fidelity realism, was used to model scenes according to established references [175] and [55]. The simulation environment incorporated a multi-layer Earth model constructed with concentric spheres representing the surface, cloud layer, and atmosphere (see figure 3.3) and a CAD representation of the EROSS client satellite provided by TASI. High-resolution textures from NASA’s Blue Marble dataset were applied to represent color, topography, ocean coverage, and cloud structures. Realism was further enhanced using specialized shaders for cloud effects and simulated atmospheric scattering via Blender’s compositing mode. The lighting was configured with a sun lamp to reproduce the spectral properties of direct solar illumination.



Figure 3.3: Representation of Blender-based Earth model.

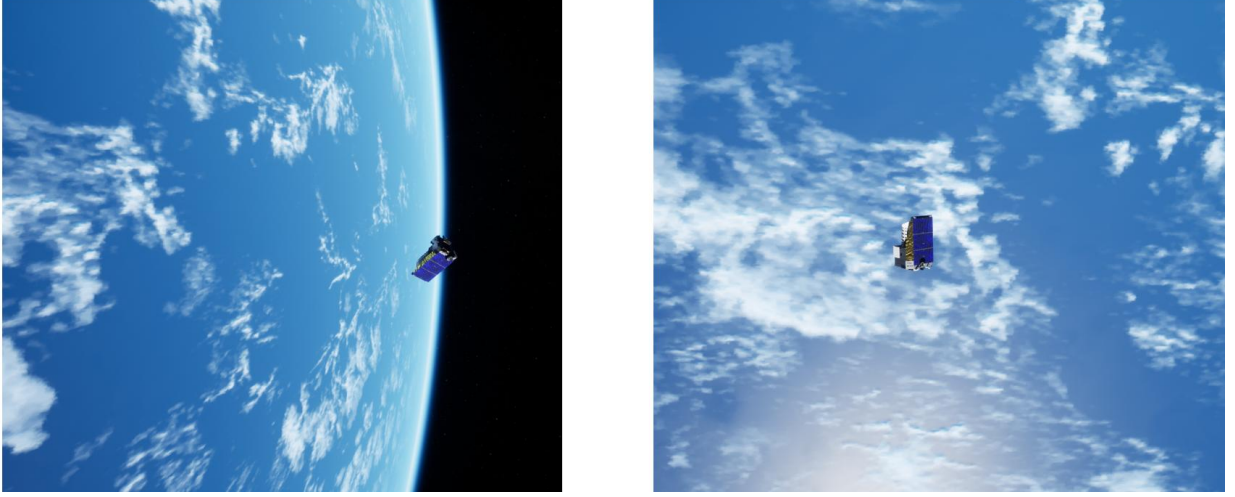


Figure 3.4: Example renderings of the Euclid satellite in randomly generated poses, used for illustrative purposes.

Automated control over scene configuration, rendering, and annotation was achieved through Blender’s Python API, supplemented by the Starfish Python library [176], which enabled efficient generation of annotated images with either prescribed or randomly sampled spacecraft poses and allowed systematic variation of scene backgrounds and lighting. Generated examples are displayed in Figure 3.4. Annotation enrichment involved projecting known 3D keypoint coordinates onto the image plane using the perspective projection equation 3.1 for downstream training. [177]:

$$\begin{bmatrix} u_i w_i \\ v_i w_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R(\mathbf{q}_{\mathbf{BC}}) \mathbf{t}_{\mathbf{BC}}] \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \quad (3.1)$$

where x_B, y_B, z_B are the 3D coordinates of the spacecraft keypoint, $[R(q_{BC}) \mathbf{t}_{BC}]$ is the roto-translation vector, f_x, f_y, c_x, c_y are the parameters of the intrinsic matrix and $u_i w_i, v_i w_i$ are the pixel coordinates of the spacecraft keypoint. These keypoints serve as the ground-truth labels for training object detection and keypoint regression models. Synthetic images were rendered with the camera parameters configured to a resolution of 512×512 px and a FOV of 67.8°, and were subsequently converted to grayscale.

3.2.2 EROSS trajectories dataset (Unreal Engine 5)

The spacecraft trajectories image dataset was generated using Unreal Engine 5 — a state-of-the-art, real-time 3D graphics platform that incorporates a comprehensive physics-based rendering engine for accurate simulation of lighting and cinematic visual effects [178]. Ray-tracing features were employed to achieve physically realistic diffusion, reflection, and diff-

Table 3.1: Examples of trajectory scenarios under different illumination conditions.

	$T_{e,h}^1$	$T_{e,h}^2$	$T_{e,h}^3$	$T_{e,h}^4$	$T_{e,h}^5$	$T_{e,h}^6$	$T_{e,h}^7$	$T_{e,h}^8$	$T_{e,h}^9$	$T_{e,h}^{10}$	$T_{e,h}^{11}$	$T_{e,h}^{12}$	$T_{e,h}^{13}$	$T_{e,h}^{14}$
$\alpha_{(deg)}$	10	10	10	40	40	40	70	70	70	120	120	120	150	150
$\beta_{(deg)}$	0	120	240	0	120	240	0	120	240	0	120	240	0	120

$T_{e,h}$ = Trajectory (easy, hard)

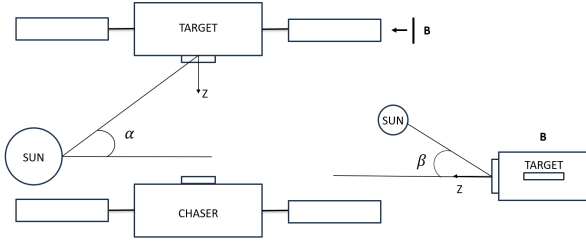
Figure 3.5: Top and side views of the servicer approaching target characterising α and β angles.

Figure 3.6: Example renderings depicting the Euclid spacecraft's motion along a trajectory sequence, used for illustrative purposes. The left image shows the furthest distance, while the right image shows the closest approach.

raction of light within each synthetic scene. Lighting was managed with Unreal Engine 5's Lumen system, which supports dynamic and scalable simulation crucial for the faithful representation of Earth, Moon, servicer, and target spacecraft, ensuring correct proportions and orbital distances as seen in LEO. High-fidelity scene rendering was accomplished using full path-tracing for a limited number of high-accuracy acquisitions. Due to the computational cost, large-scale dataset generation utilized directional lighting, optimized for the correct radiative flux and angular distribution to balance physical realism and efficiency.

The final dataset comprises 14 individual trajectory sequences, each containing 903 images depicting the servicer spacecraft's approach from an initial separation of 2 m to a terminal range of 20 cm, corresponding to the grasping point of the servicer's gripper (see Figure 3.6). Eight of the sequences represent "easy" nominal rectilinear approaches, while the remaining six are "hard" trajectories in which the servicer deviates from the nominal path and performs rotations up to 2–5 degrees. Trajectory profile examples from "easy" and "hard" cases are represented in Figure 3.7. Illumination conditions vary substantially across sequences, as detailed in Table 3.1 and Figure 3.5. All images were rendered with camera settings fixed at 512×512 px resolution and a field of view of 67.8° and converted to grayscale.

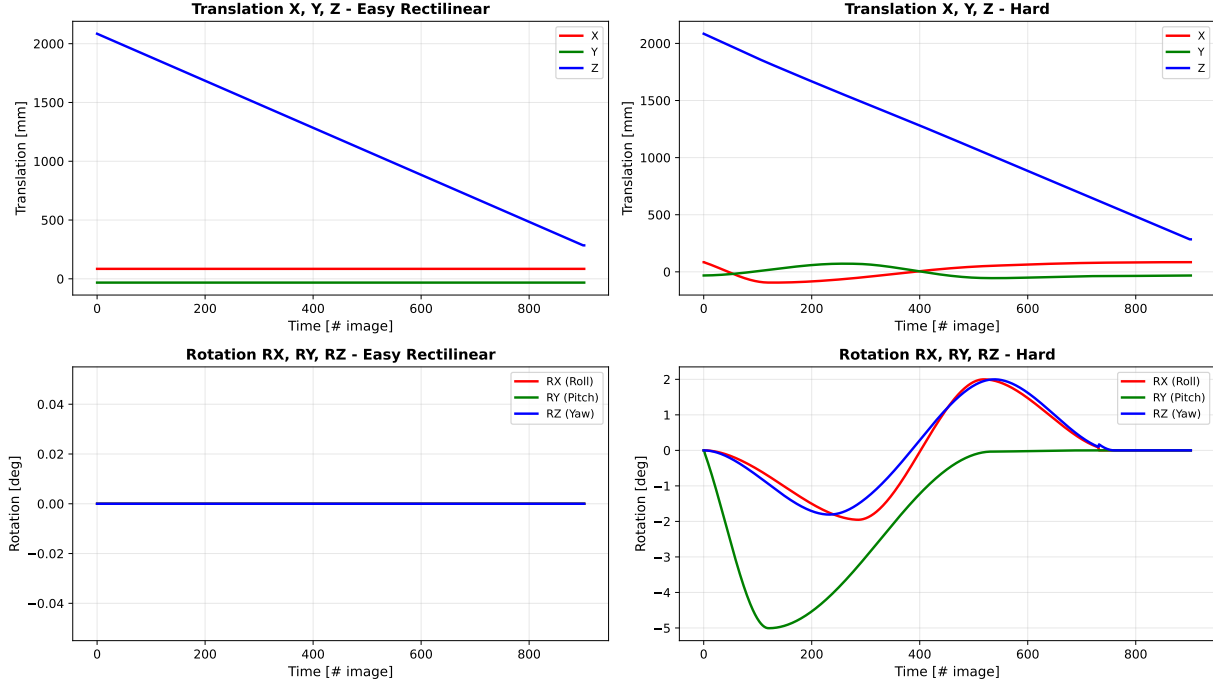


Figure 3.7: Trajectory profiles for two representative EROSS dataset sequences.

3.3 Pose estimation for the EROSS mission

A comprehensive overview of the pose estimation pipeline developed for the EROSS mission details the DL models utilized throughout the process, the training procedures, and the evaluation results. The discussion begins with a presentation of the system-level requirements that govern the design and operational constraints of the pose estimation subsystem. Subsequently, the pipeline architecture and its integration within the mission context are outlined. The section then presents the specific neural network architectures selected for both object detection and keypoint regression, along with justification for these choices based on model efficiency, accuracy, and suitability for deployment in space applications. The training methodology, including dataset preparation and optimization strategies, is then described. Finally, quantitative results are reported to demonstrate the performance and reliability of the proposed system on the embedded hardware provided by the company.

3.3.1 System requirements

The design and validation of the CRPEP is governed by a comprehensive set of operational requirements derived from the mission-level specifications established during EROSS Phase B2. These requirements define the performance envelope within which the pose estimation system must operate to support autonomous capture and servicing operations. Table 3.2 summarizes the key system-level requirements for the pose estimation subsystem. Due to

industrial confidentiality agreements between the University of Bologna and TASI, specific numerical values cannot be disclosed; however, the requirement categories and their associated metrics provide the necessary context for evaluating the pipeline performance presented in subsequent sections.

Table 3.2: System-level requirements for the EROSS pose estimation subsystem.

Requirement	Specification
Power consumption	Max. wattage per camera (one active, one cold redundant)
Sensor frame rate	Min. frame rate (Hz) for monocular camera
Relative target motion	Residual GNC velocity (mm/s) and max. angular rate (deg/s) at operation start
Pose accuracy	3σ error bounds at grasping point: position (mm), orientation (deg)
RPO duration	Prescribed max. arm motion speed (cm/s); start distance varies by scenario

Specific numerical values undisclosed due to industrial confidentiality.

The validation results presented in Section 3.4 are evaluated against these requirements to assess the operational readiness of the developed CRPEP for integration into the EROSS flight system.

3.3.2 Pipeline architecture

The close-range pose estimation pipeline (CRPEP) was initially developed for the EROSS project using a state-of-the-art, hybrid three-stage approach. Validation was conducted utilizing a Blender-based test set covering relative distances from 10 to 50 m. The general workflow is depicted in Figure 3.8. The ODN operates directly on the raw input image, identifying the ROI, which is then cropped, resized, and processed by the KRN to predict the predefined 2D pixel coordinates of the spacecraft keypoints. The final spacecraft pose is subsequently estimated via a RANSAC-based EPNP solver using the predicted image keypoints in conjunction with the known 3D model of the target. The CRPEP pipeline demonstrates robust performance when the spacecraft is fully contained within the camera’s FOV, allowing all required keypoints to be detected for accurate pose estimation. However, in the final phase of the capture or docking procedure, the standard CRPEP becomes unsuitable for pose estimation due to significant limitations associated with very close-

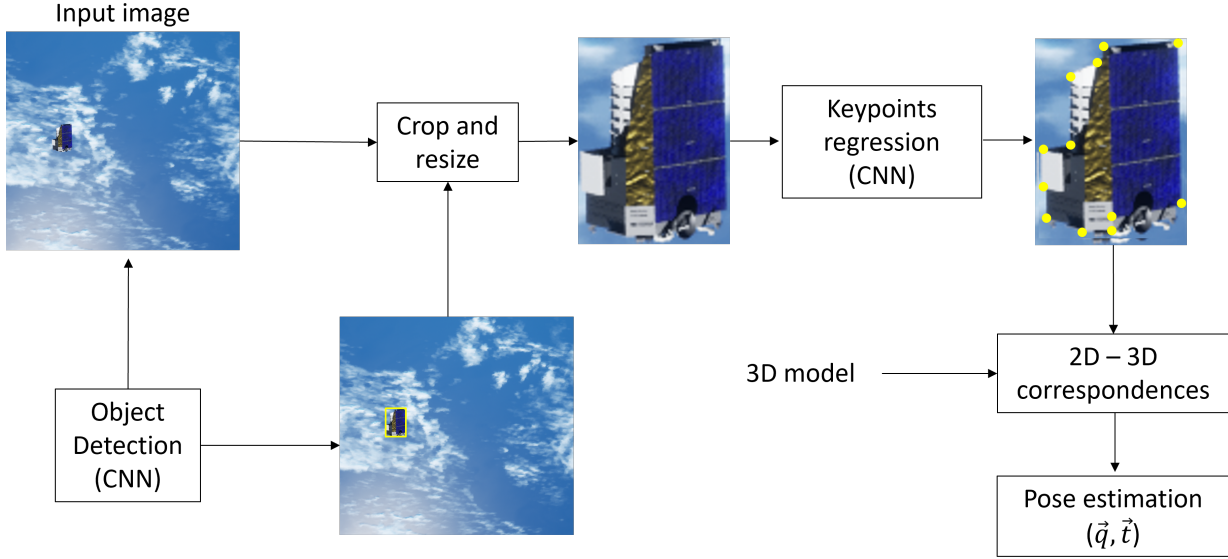


Figure 3.8: Close-range pose estimation pipeline.

range visual conditions. As the servicer spacecraft approaches at minimal separation, the target gradually moves out of the camera's FOV, resulting in an absence of detectable keypoints. In such scenarios, only partial features, such as the side of the spacecraft, may remain visible. Since EPNP algorithms necessitate a minimum of four distinct keypoints for accurate pose recovery, reliable estimation becomes infeasible under these conditions. To address this challenge and ensure robust performance throughout the entire rendezvous sequence, an enhanced CRPEP variant incorporating an active switching mechanism is introduced in figure 3.9.

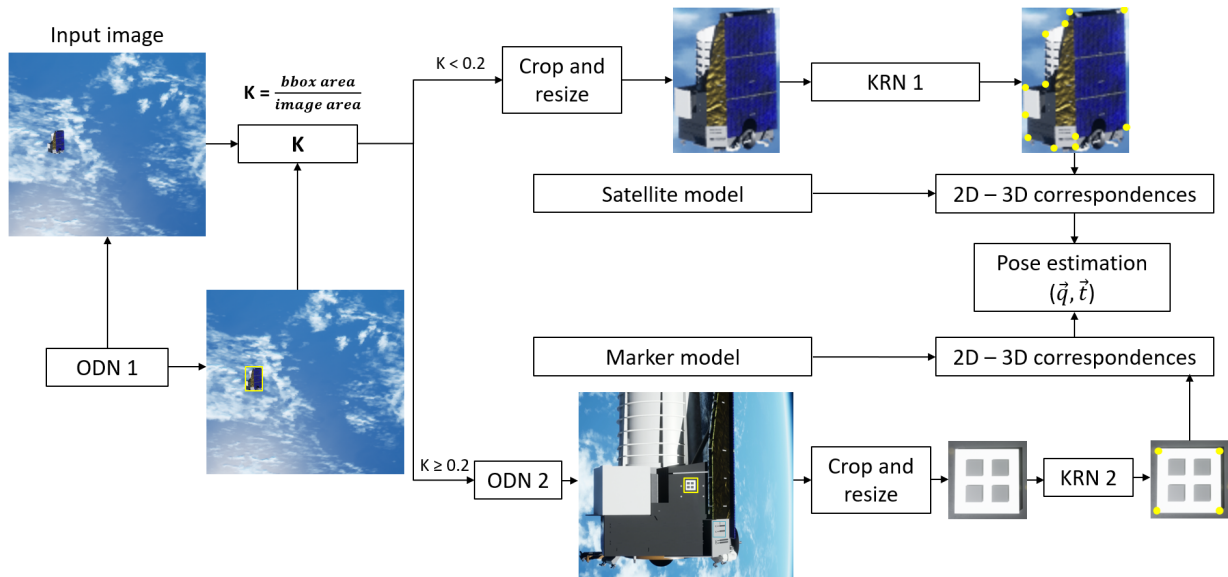


Figure 3.9: Close-range pose estimation pipeline with active switch.

The following steps outline the procedure. Initially, ODN 1, trained to detect the entire spacecraft, is applied to the raw input image. At this stage, the area ratio K between the ROI identified by ODN 1 and the total image area is computed. When this ratio remains below the predefined threshold of 0.2, the pipeline estimates the pose of the full spacecraft. If K exceeds the threshold, indicating that only a portion of the spacecraft is visible, ODN 2, which is trained to detect a specific smaller feature on the spacecraft, is subsequently activated. This is followed by deployment of KRN 2, which regresses the keypoints associated with the detected feature, and the pose of that element is then estimated. Notably, both the feature and the spacecraft share the same reference center in their respective 3D models; thus, the estimated element pose corresponds directly to the spacecraft pose. Note, while the diagram shown in Figure 3.9 uses a visual marker as the illustrative feature, the architecture is agnostic to the nature of the detected object. Consequently, any identifiable spacecraft component can be utilized, making this pipeline generally applicable, including for non-cooperative targets.

3.3.3 ODN and KRN overview and training process

To present the data preparation strategy, each synthetic dataset used in this study is divided into training, validation, and test subsets, as reported in Table 3.3. This allocation provides sufficient data for model training while also maintaining separate sets for independent evaluation of models performance.

Two distinct ODNs are evaluated for the detection stage. The first approach leverages the TensorFlow 2 Object Detection API [179] - a robust framework developed by Google for the design, training, and deployment of object detection models. This platform provides an extensive suite of tools, pre-trained models, and a user-friendly interface, thereby streamlining the implementation of state-of-the-art detection algorithms. A key strength of the API is its support for transfer learning, which enables efficient adaptation of pre-trained models from large-scale datasets to new tasks with limited annotated data.

Table 3.3: Dataset Division for Training, Validation, and Testing.

Dataset	Training	Validation	Testing
Blender	8000	1000	1000
Unreal Engine 5	9030	1806	1806

Among the available architectures, the SSD ResNet50 V1 FPN 640x640 is selected for its demonstrated accuracy and robustness. The second approach employs EfficientNet-B0 [1],

a convolutional neural network architecture originally designed for image classification tasks. This model is characterized by a compound scaling method that uniformly balances network depth, width, and resolution, facilitating high performance under restricted computational resources. For object detection, EfficientNet-B0 is adapted to regress the spacecraft bounding box coordinates by modifying its final classification layer.

A comparative evaluation of SSD ResNet50 V1 FPN 640x640 and EfficientNet-B0 in Table 3.4 on the Blender test set shows that EfficientNet-B0 achieves slightly superior performance across all metrics. The ResNet50 architecture, which employs a deep residual network with approximately 25 million parameters involves substantial computational overhead. In contrast, EfficientNet-B0 leverages compound scaling and depthwise separable convolutions, reducing the number of parameters to 5.3 million while maintaining high detection accuracy and robustness.

Table 3.4: Comparison of object detection performance between ResNet50 and EfficientNet-B0 models on the Blender-based spacecraft test set.

Metric	ResNet50	EfficientNet-B0
Mean IoU	0.913	0.931
Median IoU	0.926	0.945

Given EfficientNet-B0’s consistent and stable localization results, along with its superior efficiency, only this model is retained for further evaluation and integration within the pose estimation pipelines. As detailed in Figure 3.10, all 4 networks used in the CRPEP with active switch are implemented based on EfficientNet-B0 and perform keypoint regression task. However, ODN 1 and ODN 2 assigned to object detection while KRN1 and KRN 2 to direct keypoint regression, with their respective outputs serving as inputs for the final EPNP-based pose estimation.

EfficientNet-B0 backbones were initialized using weights pre-trained on ImageNet [62]. Training and validation datasets were loaded with custom PyTorch dataloaders, which handled keypoint annotations and implemented the specified augmentation strategies. The models were trained with a batch size of 64 and optimized with the Adam optimizer at a learning rate of 10^{-3} . Each training run consisted of 100 epochs, and model performance was monitored using the root mean squared error (RMSE) metric after each epoch. To enhance robustness and prevent overfitting, data augmentation as shown in Figure 3.11 was performed during training using the Albumentations library [180]. Input images were resized to 224×224 px and subjected to random rotations with a limit of 15° . Brightness and contrast were adjusted by up to 50%. Noise and other image distortions, including

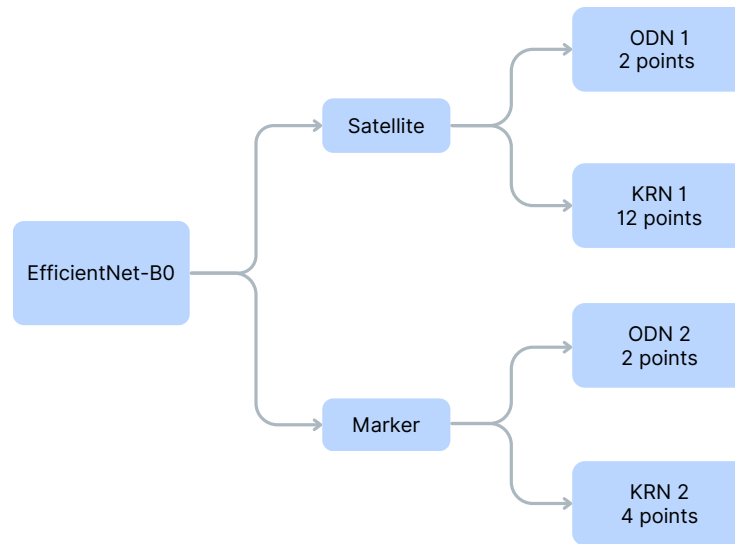


Figure 3.10: Schematic overview of the EfficientNet-B0-based detection and keypoint regression networks.

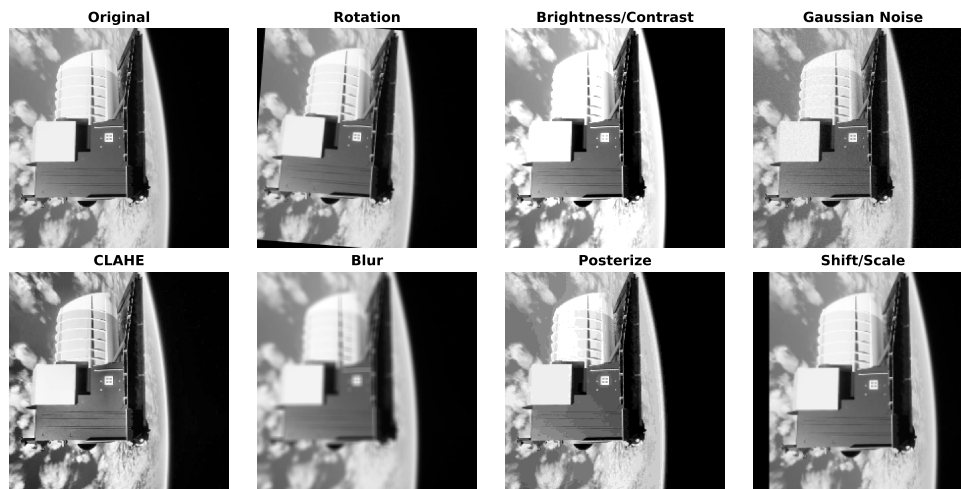


Figure 3.11: Visual examples of data augmentation transformations applied to images during training.

Gaussian noise, CLAHE, random gamma correction, and compression artifacts were introduced to emulate variable real-space conditions. Further transformations, such as shift, scaling, and minor rotations, were applied to diversify the training data. All training procedures were executed on an NVIDIA GeForce RTX 3090 GPU.

3.4 Validation results on Jetson Nano

The CRPEP with active switch was deployed on the NVIDIA Jetson Nano platform [181] using JetPack 4.6. All model components were packaged and executed within Docker

containers to ensure reproducibility and portability in the embedded environment. No inference optimization strategies such as TensorRT acceleration or quantization were implemented; instead, all model evaluations were performed using standard PyTorch implementations at float32 precision. Performance assessment was conducted by measuring inference speed in FPS for each neural network component as well as for the complete pipeline.

Table 3.5 summarizes the observed inference rates across all pipeline components. Individual network modules (ODN 1, ODN 2, KRN 1, and KRN 2) achieved inference speeds ranging from 15 to 16 FPS when evaluated independently. However, when integrated into the complete pipeline architecture, the end-to-end inference rate decreased to 5 FPS due to sequential processing overhead. While this performance satisfies the minimum requirements for guidance, navigation, and control (GNC) systems in proximity operations, which typically operate at 1–10 Hz update rates, further optimization through model quantization and framework-level acceleration could substantially improve throughput for more demanding real-time applications.

Table 3.5: Inference speed of each network module and overall pipeline on NVIDIA Jetson Nano.

Module	Inference FPS
ODN 1	15
KRN 1	16
ODN 2	15
KRN 2	16
Full pipeline	5

Table 3.6 presents a comprehensive quantitative evaluation of the complete pipeline across three distinct test scenarios: random poses from the Blender dataset ($\text{Satellite}_{\text{rp}}$), trajectory sequences with the full spacecraft visible ($\text{Satellite}_{\text{tr}}$), and close-range sequences requiring marker-based pose estimation ($\text{Marker}_{\text{tr}}$). Object detection performance remained consistently high across all scenarios, with mean IoU values ranging from 0.931 to 0.972 and median IoU values between 0.945 and 0.981, confirming robust spacecraft localization under diverse viewing geometries and illumination conditions.

Keypoint detection accuracy, quantified by RMSE, exhibited substantial variation depending on scenario complexity. The random pose dataset yielded the highest RMSE of 8.65 px, attributed to the wide distribution of spacecraft orientations, scales, and Earth background variations that challenge keypoint localization consistency. The trajectory-based full spacecraft scenario achieved significantly improved performance with an RMSE of 5.40 px. The marker-based close-range scenario demonstrated exceptional keypoint pre-

cision with an RMSE of only 0.90 px, reflecting the higher visual contrast and geometric simplicity of the fiducial marker relative to the full spacecraft structure.

Table 3.6: Complete pipeline performance evaluation across different test scenarios.

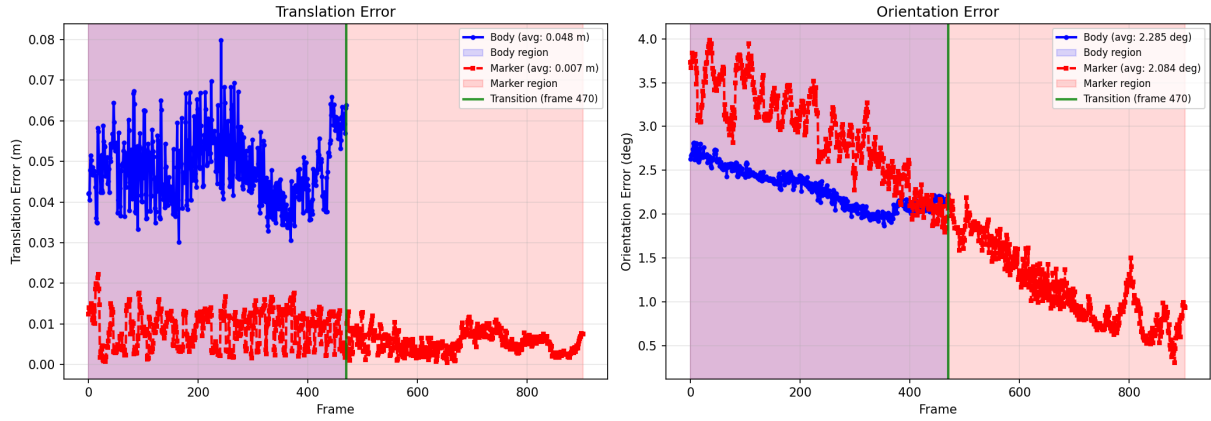
Metric	Satellite_{rp}	Satellite_{tr}	Marker_{tr}
Mean IoU	0.931	0.965	0.972
Median IoU	0.945	0.975	0.981
RMSE (px)	8.65	5.40	0.90
Mean Position Error (m)	1.981	0.047	0.007
Mean Orientation Error (°)	13.809	2.147	2.112

rp: random poses, tr: trajectories

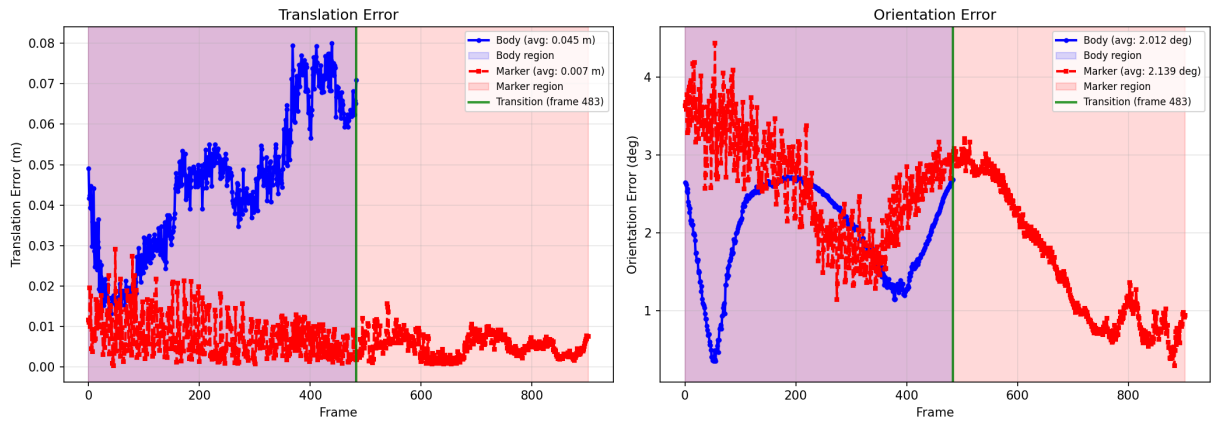
The trajectory-based full spacecraft configuration achieved centimeter-level translational accuracy (47 mm), which is compatible with mid-range proximity operations but falls short of the sub-centimeter precision typically required for final approach and capture phases. The marker-based configuration demonstrated exceptional translational performance with a mean error of 7 mm, approaching the accuracy levels necessary for terminal docking operations. However, orientation errors remained elevated at 2.147° and 2.112° for the trajectory and marker scenarios, respectively. These angular errors are attributable to several factors: (1) the low image resolution (512×512 px) limits the precision of key-point localization, particularly for small angular variations; (2) challenging illumination conditions in certain trajectory sequences degrade feature detectability; and (3) the limited number of keypoints used for EPnP solving (4 points for the marker) constrains the geometric conditioning of the pose recovery problem.

Figure 3.12 illustrates the temporal evolution of translation and orientation errors along two representative test trajectories — one nominal rectilinear approach (easy, Figure 3.12a) and one perturbed trajectory with attitude deviations (hard, Figure 3.12b). The x-axis represents frame number, corresponding to a continuous decrease in relative distance from 2 m to 20 cm. The green vertical line in each plot marks the activation of the switching mechanism, where the pipeline transitions from full spacecraft body pose estimation to marker-based pose estimation.

For the easy trajectory (Figure 3.12a), the body-based pose estimation maintains relatively stable translational errors between 30–70 mm prior to the switch activation, with orientation errors gradually decreasing from approximately 3.5° to 2.0° as the spacecraft approaches and occupies a larger fraction of the image. Following the transition to marker-based estimation at frame 470, translational performance improves dramatically, with errors consistently remaining below 10 mm throughout the remainder of the trajectory. Orientation errors similarly decrease to approximately 1.0–1.5°, demonstrating the effec-



(a) Trajectory 13 (easy)



(b) Trajectory 14 (hard)

Figure 3.12: Validation pose errors across test trajectories. The frame range on the x-axis corresponds to a relative distance range between the chaser and target from 2 m to 20 cm. The green vertical line indicates the activation of the switching mechanism between satellite body and marker-based pose estimation.

tiveness of the switching strategy in maintaining continuous, high-quality pose estimates across the complete approach profile.

The hard trajectory (Figure 3.12b) exhibits substantially more challenging behavior due to the non-rectilinear motion and spacecraft attitude variations. Body-based translational errors fluctuate between 30–80 mm with significantly higher variance relative to the nominal case, while orientation errors reach 4.5° with pronounced oscillations reflecting the difficulty of accurate pose recovery under non-aligned viewing geometries. Notably, a dramatic improvement in both translation and orientation is observed immediately following the switch activation at frame 470. Marker-based estimation reduces translational errors to below 15 mm and orientation errors to approximately 1.0 – 2.0° .

3.5 Chapter conclusion

This chapter has presented the development and validation of a DL-based pose estimation pipeline for the EROSS mission. The approach leveraged synthetic datasets generated in Blender, encompassing 10,000 images with randomly distributed spacecraft poses, and Unreal Engine 5, contributing 12,642 trajectory images spanning approach distances from 2 m to 20 cm under systematically varied illumination conditions. To address visibility challenges at close operational distances, particularly when the target spacecraft moves outside the camera’s FOV, a CRPEP with active switching mechanism was implemented. The pipeline incorporated two ODNs and two KRNs based on the EfficientNet-B0 architecture, enabling robust detection and pose estimation across the complete rendezvous sequence.

Comprehensive evaluation on the NVIDIA Jetson Nano platform demonstrated performance compatible with proximity operations requirements. The complete pipeline achieved an end-to-end inference rate of 5 FPS using standard PyTorch FP32 implementations without optimization, satisfying the typical 1–10 Hz update rate requirements for GNC systems in on-orbit servicing missions. Quantitative results varied substantially across operational scenarios: the random pose test set yielded mean position errors of 1.981 m with orientation errors of 13.809° , reflecting the challenges of unconstrained viewing geometries and diverse Earth backgrounds. The trajectory-based evaluation with full spacecraft visibility achieved significantly improved performance, with mean position errors of 47 mm and orientation errors of 2.147° , demonstrating centimeter-level accuracy compatible with mid-range proximity operations. Most notably, the marker-based close-range configuration achieved exceptional precision with mean position errors of 7 mm and orientation errors of 2.112° , approaching the accuracy requirements for terminal docking phases.

Temporal analysis of pose estimation errors along representative trajectories confirmed the effectiveness of the active switching mechanism. For nominal rectilinear approaches, body-based pose estimation maintained translational errors between 30–70 mm, with the marker-based configuration reducing errors to below 10 mm following switch activation. Even under challenging non-rectilinear trajectories with attitude perturbations, the switching strategy maintained operational continuity with post-switch translational errors consistently below 15 mm. Object detection performance remained robust across all scenarios, with mean IoU values ranging from 0.931 to 0.972, confirming reliable spacecraft localization under diverse operational conditions.

Despite these promising results, several limitations warrant attention in future research. Orientation errors of $2\text{--}4^\circ$ remain elevated for precision-critical applications, particularly under challenging illumination conditions and non-aligned viewing geometries. Contributing factors include the limited image resolution (512×512 px), constrained keypoint con-

figurations for EPnP solving, and insufficient augmentation strategies to fully replicate the diversity of real operational conditions. Future work should prioritize: (1) implementation of TensorRT-based model optimization and INT8 quantization to improve inference throughput while maintaining acceptable accuracy; (2) comprehensive augmentation techniques incorporating domain randomization and adversarial training to enhance robustness under extreme lighting and perspective distortions; (3) exploration of alternative network architectures such as vision transformers [71], which may yield improved pose estimation accuracy albeit with increased computational demands; and (4) hardware integration of dual-camera systems with complementary wide and narrow FOVs, enabling seamless coverage across extended operational distance ranges without reliance on active switching mechanisms. The developed CRPEP framework provides a validated foundation for future DL-based navigation systems in autonomous on-orbit servicing missions.

Chapter 4

Inference Acceleration on Embedded Systems

This chapter investigates computational optimization strategies for deploying DL-based spacecraft detection systems on resource-constrained orbital platforms. The investigation evaluates model compression techniques, precision reduction methodologies, and hardware acceleration frameworks. Section 4.1 details the baseline network configurations and training protocols. Section 4.2 examines model format conversion, quantization strategies, and execution provider optimization on NVIDIA Jetson Orin Nano hardware. Section 4.3 presents comprehensive performance benchmarks encompassing inference speed, throughput metrics, and detection accuracy degradation analysis. Section 4.4 contextualizes the obtained findings, compares performance benchmarks, and identifies key technical limitations. The chapter concludes with Section 6.3.

4.1 Models Training and Configuration

4.1.1 Dataset Overview

For the inference optimization investigations presented in this chapter, a selected subset from the synthetic partition of the SPEED+ dataset [5] was employed. Table 4.1 presents the dataset configuration utilized for all optimization experiments. The intentional reduction from the full SPEED+ synthetic dataset, which contains 47,966 available training images, was justified by the observation that this subset is sufficient to train robust object

Table 4.1: Dataset Configuration for Inference Optimization Experiments.

Partition	Images Used	Source
Training	10,000	SPEED+ Synthetic
Validation	1,000	SPEED+ Synthetic
Test (Synthetic)	1,000	SPEED+ Synthetic

detection models capable of achieving satisfactory performance on the test set. The primary objective of this phase was to investigate model optimization, rather than to maximize detection accuracy. Cross-domain generalization, including experiments involving training on the complete dataset with domain adaptation strategies, is addressed in the following chapter.

4.1.2 Object Detection Network: YOLOv8n

The first evaluated detection framework is the YOLOv8n, selected for its favorable trade-off between computational efficiency and detection accuracy. The nano variant, comprising approximately 3.2 million parameters, offers a notably reduced model complexity, rendering it well-suited for embedded deployment under stringent resource constraints.

Training configuration

The YOLOv8n training process was configured to balance robust model learning with computational efficiency, following the default training hyperparameters provided by the Ultralytics library [3]. The configuration, presented in Table 4.2, includes a training duration of 100 epochs with a batch size of 16 and a base learning rate of 1×10^{-2} . Optimization was performed using stochastic gradient descent (SGD) with momentum, complemented by a weight decay factor of 5×10^{-4} to mitigate overfitting. The training loss formulation incorporates a composite of Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) for classification, together with the Distribution Focal Loss (DFL) and Complete Intersection over Union (CIoU) components for bounding box regression, enabling balanced optimization between localization precision and detection confidence.

In conjunction with the training configuration, the YOLOv8n pipeline employed a comprehensive data augmentation strategy to improve model generalization and robustness under varying illumination, geometric, and background conditions. The augmentation scheme, summarized in Table 4.3 integrates photometric, geometric, and advanced transformations. Photometric augmentations include random adjustments in hue, saturation, and brightness to emulate sensor and lighting variability present in on-orbit imaging. Geometric transformations such as translation, rescaling, and horizontal flipping promote spatial

Table 4.2: YOLOv8n Training Hyperparameters.

Parameter	Setting
Training Epochs	100
Batch Size	16
Base Learning Rate	1×10^{-2}
Weight Decay	5×10^{-4}
Class Loss	BCEWithLogitsLoss
Box Loss	DFL + CIoU
Distribution Focal Loss	DFL
Optimizer	SGD with Momentum

invariance, while advanced methods, such as mosaic augmentation and random erasing, further enhance dataset diversity by synthesizing composite images and simulating partial occlusions. Some augmentation examples are presented in Figure 4.1.

Table 4.3: YOLOv8n Training Data Augmentation Parameters.

Augmentation Type	Parameter	Default Value
Photometric	HSV Hue (hsv_h)	0.015
	HSV Saturation (hsv_s)	0.7
	HSV Value/Brightness (hsv_v)	0.4
Geometric	Translation	0.1 ($\pm 10\%$)
	Scale	0.5 ($\pm 50\%$)
	Horizontal Flip (fliplr)	0.5 probability
Advanced	Mosaic	1.0 (enabled)
	Random Erasing	0.4 probability
	Crop Fraction	1.0 (full image)

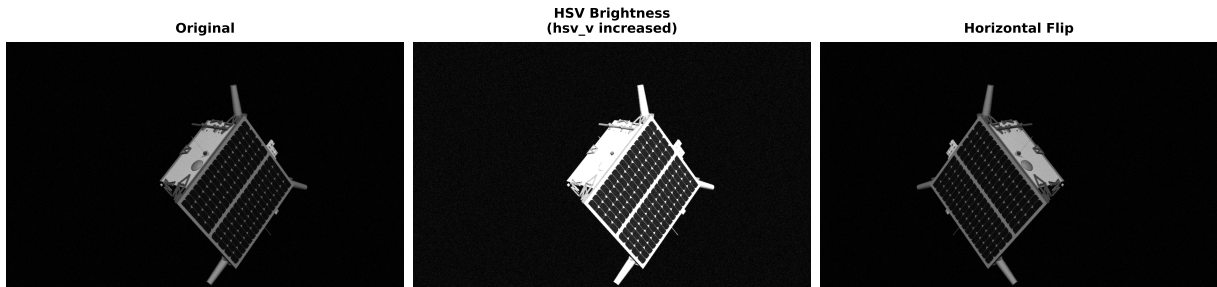


Figure 4.1: Representative examples of data augmentation techniques employed in YOLOv8n training.

Training dynamics

During training the model demonstrated stable convergence and consistent performance improvements across both training and validation metrics. The loss evolution revealed a characteristic rapid reduction during the initial epochs, followed by progressive refinement and stabilization beyond approximately epoch 60, indicating efficient optimization without signs of overfitting. At the end of training, the model achieved a mean average precision (mAP@0.5:0.95) of 0.994, with corresponding Box Loss, Classification Loss, and DFL values of 0.176, 0.098, and 0.856, respectively. The evolution of the loss components throughout the training process is depicted in Figure 4.2, highlighting smooth convergence behavior and effective balance between detection accuracy and localization precision. Qualitative inspection of representative validation predictions, shown in Figure 4.3, further illustrates the model’s ability to accurately localize the spacecraft.

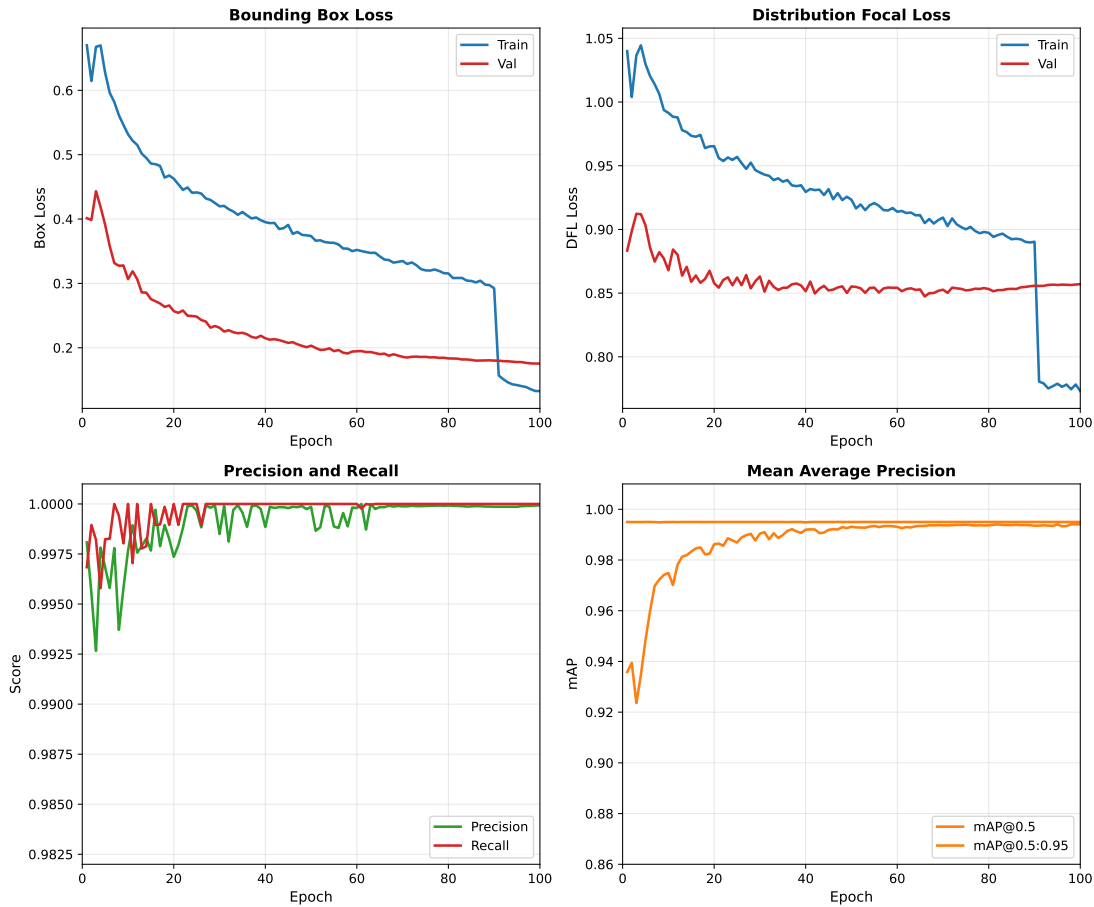


Figure 4.2: Training and validation loss evolution for YOLOv8n over 100 epochs.

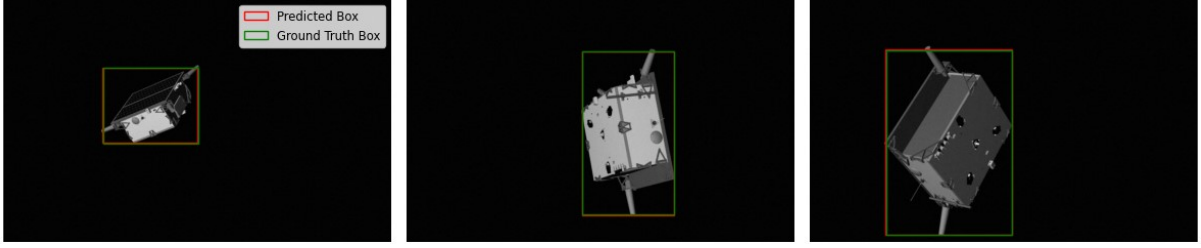


Figure 4.3: Representative YOLOv8n validation predictions after training.

4.1.3 Object Detection Network: EfficientNet-B0

EfficientNet-B0 was selected as the second network for comparison with YOLOv8n due to its compound scaling methodology, which systematically balances representational capacity and computational efficiency. In this study, the EfficientNet-B0 backbone was adapted to perform direct regression of spacecraft bounding box coordinates.

Training Configuration

The model was trained for 100 epochs using the Adam optimizer and a base learning rate of 1×10^{-3} , with a batch size of 16 and weight decay regularization of 5×10^{-4} . The set of training hyperparameters employed in this study is summarized in Table 4.4. The optimization objective was defined using the SmoothL1Loss [140], selected for its hybrid quadratic-linear formulation that reduces sensitivity to outliers. This property is particularly advantageous in coordinate regression tasks, where occasional large prediction deviations may arise due to partial target occlusions or extreme viewing geometries.

Table 4.4: EfficientNet-B0 Training Hyperparameters.

Parameter	Setting
Training Epochs	100
Batch Size	16
Loss Function	SmoothL1Loss
Optimizer	Adam
Base Learning Rate	1×10^{-3}
Weight Decay	5×10^{-4}

To enhance network generalization and mitigate overfitting, a diverse augmentation pipeline was applied during training. The augmentation set, presented in Table 4.5, includes a combination of photometric perturbations to simulate variations in illumination and contrast, geometric transformations to promote spatial invariance, and degradations designed to replicate sensor and compression artifacts typically present in spaceborne imagery.

Table 4.5: EfficientNet-B0 Training Data Augmentation Parameters.

Augmentation Type	Parameter	Probability
Photometric	RandomBrightnessContrast	0.2
	GaussNoise	0.8
	CLAHE	0.8
	RandomGamma	0.8
Degradations	ImageCompression	0.8
	Posterize	0.8
	Blur	0.8
Geometric	Rotate	0.8
	Affine	0.2

Training dynamics

The model performance was evaluated using Root Mean Square Error (RMSE) and mean Average Precision (mAP) metrics to ensure a balanced assessment of localization precision and detection consistency. Throughout training, the network exhibited a typical learning trajectory characterized by a steep initial decline in loss values, followed by gradual refinement as convergence was approached. The close alignment between training and validation loss curves indicated stable optimization and effective regularization, with no evidence of overfitting (Figure 4.5).

At convergence, the network attained a validation RMSE of 4.94 px and a mean Average Precision (mAP@0.5:0.95) of 0.721, reflecting robust regression and detection capabilities under the established training configuration in Table 4.4. The use of SmoothL1Loss contributed to stable learning behavior by mitigating the influence of extreme prediction deviations, particularly in scenarios involving partial occlusions or challenging geometries. Representative qualitative results, shown in Figure 4.4, demonstrate consistent bounding box predictions.

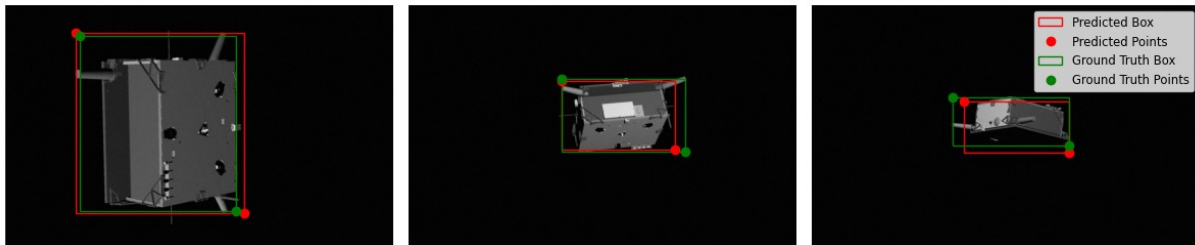


Figure 4.4: Representative EfficientNet-B0 validation predictions after training.

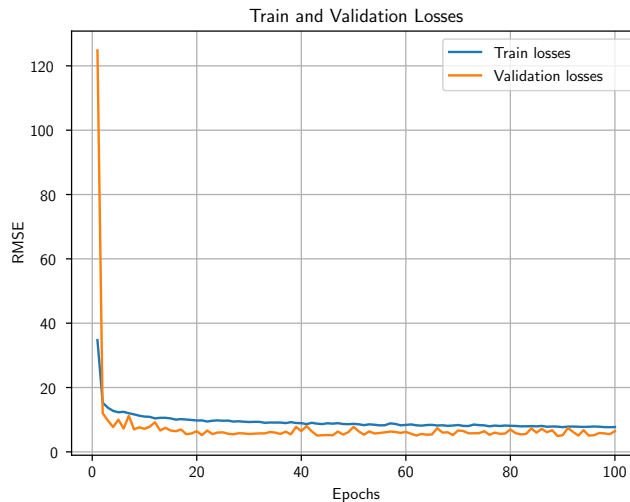


Figure 4.5: Training and validation loss evolution for EfficientNet-B0 over 100 epochs.

4.2 Inference Optimization Strategies

To enable deployment on computationally constrained spacecraft processors, systematic investigation of model compression and acceleration techniques was conducted. This section details format conversion methodologies, numerical precision reduction strategies, and hardware-specific optimization frameworks evaluated on the NVIDIA Jetson Orin Nano embedded platform.

4.2.1 Model Format Conversion

Three distinct model representation formats were investigated: native PyTorch serialization, Open Neural Network Exchange (ONNX) intermediate representation, and NVIDIA TensorRT optimized execution format. Each format provides different trade-offs between portability, optimization potential, and hardware-specific acceleration capabilities.

ONNX Conversion

ONNX provides a hardware-agnostic intermediate representation enabling deployment across heterogeneous execution environments. Models were exported from PyTorch to ONNX format using the `torch.onnx.export` API with opset version 13, maintaining full operator compatibility. The ONNX Runtime inference engine with CUDA execution provider was employed for GPU-accelerated inference.

TensorRT Optimization

TensorRT applies graph optimization transformations including layer fusion, kernel auto-tuning, and precision calibration to maximize throughput on NVIDIA hardware. Models were converted to TensorRT engine format using the `trtexec` utility with dynamic shape support for the input tensor dimensions. A warmup procedure using 50 calibration images was executed prior to performance benchmarking to ensure complete kernel compilation and optimization.

4.2.2 Numerical Precision Reduction

Quantization techniques reduce computational complexity and memory bandwidth requirements by representing weights and activations using lower-precision numerical formats. Three precision configurations were systematically evaluated: FP32 (full precision baseline), FP16 (half-precision floating-point), and INT8 (8-bit integer quantization).

FP16 Mixed Precision

Half-precision inference reduces memory footprint by 50% and enables utilization of Tensor Core acceleration units on modern NVIDIA architectures. FP16 conversion was performed automatically by TensorRT during engine compilation with `-fp16` flag specification.

INT8 Quantization

Integer quantization provides maximum compression ratio but requires calibration to determine appropriate scaling factors mapping floating-point value ranges to discrete integer representations. Two quantization schemes were evaluated:

1. **Quantize-Dequantize (QDQ) representation:** Explicit quantization/dequantization operations inserted into the computational graph, enabling per-layer precision control.
2. **QOperator representation:** Fused quantized operators replacing standard floating-point operations throughout the network.

Post-training quantization was performed using 500 calibration samples from the synthetic validation set. The calibration algorithm was employed for determining optimal quantization parameters.

4.2.3 Batch Processing

Batch inference amortizes preprocessing overhead and enables parallelization opportunities across multiple input samples. Experiments evaluated batch sizes of 1 (online inference) and 16 (batch processing) to assess throughput scalability versus latency constraints for different operational scenarios.

4.3 Experimental Results

Comprehensive performance evaluation was conducted across accuracy preservation and computational efficiency dimensions. All benchmarks were executed on NVIDIA Jetson Orin Nano hardware (8GB variant, 1024-core NVIDIA Ampere GPU, 6-core Arm Cortex-A78AE CPU) operating in 15W power mode.

4.3.1 YOLOv8n Performance

Inference benchmarking of the YOLOv8n architecture was conducted across different model representations, precision levels, and optimization frameworks, with results summarized in Table 4.6. The experiments revealed consistent performance trends across all configurations, with TensorRT-based deployments demonstrating the most significant acceleration benefits compared to PyTorch and ONNX baselines.

In particular, TensorRT inference using FP16 achieved an inference speed of 11.65 ms, corresponding to 85.85 FPS, while preserving full accuracy parity with the FP32 configuration (IoU = 0.945). Quantization to INT8 precision yielded the highest throughput, attaining 7.74 ms (129.19 FPS), though at the expense of a measurable accuracy reduction (IoU decrease from 0.946 to 0.886, equivalent to a 6.3% degradation). This trade-off highlights the architecture’s sensitivity to lower-precision arithmetic when operating under aggressive quantization.

The results also emphasize the notable efficacy of hardware-level acceleration, as TensorRT-optimized inference at FP16 precision provided a $2.3\times$ speedup over the PyTorch FP32 GPU baseline. Batch inference experiments further demonstrated the scalability of the YOLOv8n model, with batch execution in FP32 mode achieving speed of 141.1 FPS. However, this improvement comes with increased latency in per-frame processing contexts, where real-time requirements prioritize immediate predictions. Overall, these findings confirm that YOLOv8n maintains robust inference performance even under reduced precision constraints, establishing it as an optimal candidate for real-time, on-board vision applications.

Table 4.6: YOLOv8n Jetson Orin Nano inference performance across multiple precision and deployment formats.

Format	Precision	Inference Speed (ms)	FPS	IoU	Pre Speed (ms)	Post Speed (ms)	# Images
PyTorch	fp32	26.684	37.47	0.9460	7.20	431.36	1000
ONNX CUDA	fp32	28.661	34.89	0.945	7.784	408.189	1000
	fp16	33.84	29.55	0.945	8.50	456.19	1000
TensorRT	fp32	20.42	48.97	0.9453	9.710	499.30	1000
	fp16	11.647	85.85	0.945	8.634	432.71	1000
	int8	7.74	129.19	0.886	7.028	343.18	1000
TensorRT Batch	fp32	7.085	141.14	0.946	5.804	282.86	1000

4.3.2 EfficientNet-B0 Performance

The inference performance of the EfficientNet-B0 was evaluated under multiple computation backends and precision settings to analyze optimization gains and trade-offs between accuracy and execution speed. The results presented in Table 4.7 indicate that GPU acceleration markedly improved computational throughput relative to CPU-based execution. Specifically, PyTorch GPU inference achieved a $51\times$ speedup compared to the CPU baseline, reducing latency from 1841 ms to 35.55 ms per image. This dramatic improvement reflects the strong parallelization efficiency of convolutional operations within the EfficientNet-B0 structure when executed on CUDA-enabled devices.

Among all configurations, TensorRT deployment using FP16 precision achieved the highest single-image performance, delivering an inference time of 7.35 ms (136.05 FPS) while maintaining an IoU of 0.850. This represents negligible degradation compared to the full-precision FP32 baseline, confirming that half-precision arithmetic can be safely adopted for embedded inference without compromising detection quality. The FP16 TensorRT configuration yielded a $250\times$ speedup relative to CPU inference and improved performance by approximately $4.8\times$ over the PyTorch GPU execution, demonstrating the benefits of precision scaling and hardware-specific optimization.

Further experiments incorporating batch inference strategies revealed that processing 16 images simultaneously enhanced per-sample throughput to 4.37 ms (229.09 FPS). Although this result underscores the scalability and efficiency of the TensorRT framework, it also introduces latency that is unsuitable for real-time systems. The quantized INT8 configurations, while offering moderate speed benefits, suffered from substantial accuracy degradation (IoU below 0.45), attributable to quantization incompatibilities with certain EfficientNet operations. Therefore, FP16 precision represents the most effective trade-off between performance and numerical stability for embedded deployment of this architecture.

Table 4.7: EfficientNet-B0 Jetson Orin Nano inference results across varying precision and deployment configurations. The superscript denotes the adopted quantization operator representation: 1 for QDQ and 2 for QOperator.

Format	Precision	Inference Speed (ms)	FPS	IoU	Pre Speed (ms)	Post Speed (ms)	# Images
PyTorch CPU	fp32	1841	0.543	0.845	27.11	0.110	1000
PyTorch GPU	fp32	35.546	28.13	0.845	16.021	0.346	1000
ONNX CPU	fp32	61.872	16.16	0.845	20.35	0.1302	1000
ONNX CUDA	fp32	21.081	47.43	0.849	16.59	0.1337	1000
	fp16	17.520	57.07	0.849	14.33	0.119	1000
	int8 ^[1]	29.854	33.49	0.436	18.72	0.1242	1000
	int8 ^[2]	24.075	41.53	0.436	18.824	0.122	1000
TensorRT	fp32	10.774	92.81	0.845	17.13	0.1174	950
	fp16	7.350	136.05	0.850	14.25	0.1061	950
	int8 ^[1]	11.151	86.88	1.71×10^{-6}	16.435	0.148	950
	int8 ^[2]	28.031	35.67	0.436	17.866	0.125	950
TensorRT Batch	fp32	4.365	229.09	0.850	17.95	0.1123	976

4.3.3 Comparative Analysis

Direct architectural comparison reveals that optimized EfficientNet-B0 (TensorRT FP16: 7.35 ms, 136.05 FPS) marginally outperforms quantized YOLOv8n (TensorRT INT8: 7.74 ms, 129.19 FPS) in raw throughput by 6.86 FPS, while incurring only a 4.1% accuracy penalty relative to YOLOv8n’s baseline performance.

The comparative distribution of inference speeds, shown in Figure 4.6, demonstrates that both networks benefit considerably from TensorRT deployment. While TensorRT FP16 configurations deliver near-peak performance for both architectures, INT8 quantization has a more pronounced detrimental impact on EfficientNet-B0. However, it lead to a reduction in IoU to sub-optimal levels as represented in Figure 4.7. YOLOv8n, on the other hand, exhibits better quantization resilience, retaining acceptable detection accuracy even under INT8 precision, making it particularly favorable for mission profiles prioritizing power efficiency without severe accuracy trade-offs.

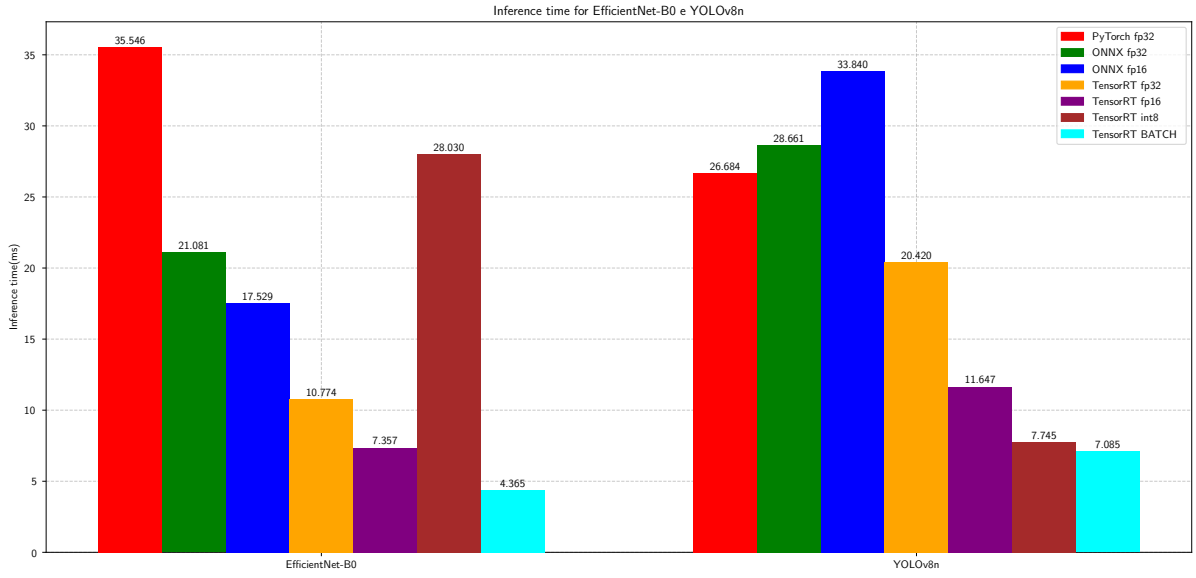


Figure 4.6: Inference time comparison between EfficientNet-B0 and YOLOv8n.

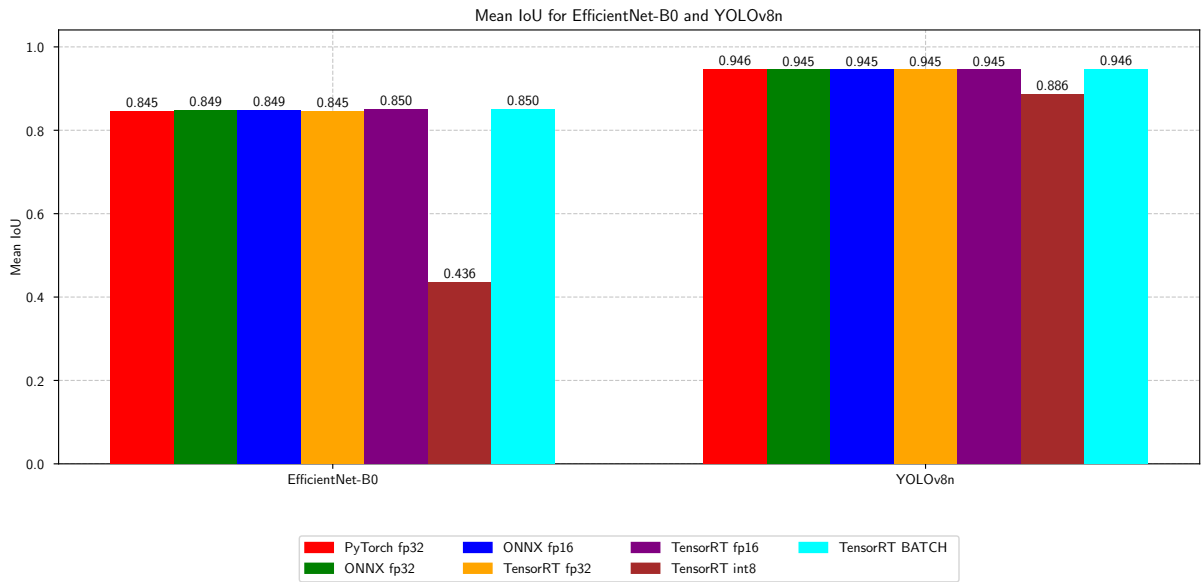


Figure 4.7: Accuracy comparison between EfficientNet-B0 and YOLOv8n.

4.4 Discussion and Limitations

4.4.1 Performance context

To assess the practical significance of the proposed inference optimization pipeline, achieved results were compared against established benchmarks from existing space-qualified and embedded vision systems. This contextual evaluation demonstrates that the reported performance metrics align favorably with operational requirements of onboard spacecraft autonomy and real-time visual navigation.

RemoveDEBRIS Mission Validation

The RemoveDEBRIS mission [24] employed vision-based navigation utilizing digital camera and LiDAR sensors for debris observation and rendezvous operations. The system demonstrated processing times below 100 ms per frame for determining critical parameters including range and angular rates essential for proximity operations. The optimized architectures presented herein achieve 7-8 ms core inference times (15-30 ms including preprocessing), representing 3-13 $\times$ improvement over RemoveDEBRIS performance benchmarks while providing equivalent detection capabilities.

Comparative Embedded System Performance

Lofqvist et al. [4] investigated object detection acceleration on Jetson Nano hardware using Single Shot MultiBox Detector (SSD) and Region-based Fully Convolutional Network (R-FCN) architectures pre-trained on the DOTA aerial imagery dataset [182]. Their optimized systems achieved inference times of 1.24-5.21 seconds depending on image resolution and compression strategy as represented in Table 4.8. In the referenced paper, the term "Large" corresponds to images with a resolution of $7,500 \times 7,500$ px, whereas "Small" refers to images with a resolution of 350×350 px.

Table 4.8: Inference speed obtained on Jetson Nano in seconds [4].

	Large image		Small image	
	Original	Compressed	Original	Compressed
SSD	5.21	3.07	1.48	1.24
R-FCN	7.32	6.16	5.39	5.21

The optimized architectures evaluated in this work at approximately 8 ms inference speed achieve a 169 $\times$ acceleration compared to the fastest configuration reported in [4] (1.24s), underscoring the substantial performance gains. It is important to note that the DOTA

dataset used in the referenced paper contains a substantially higher number of objects per frame, which likely contributed to the increased computational load observed in their evaluation.

4.4.2 Identified Limitations

INT8 quantization of EfficientNet-B0 encountered significant technical challenges related to operation compatibility with ONNX Runtime and TensorRT execution providers. Specifically, quantized models utilizing either QDQ or QOperator representations exhibited frequent CPU fallback for unsupported operations (QLinearMul, QLinearGlobalAveragePool, QLinearSigmoid, QLinearConv), resulting in performance degradation.

QDQ representation produced semantically incorrect outputs (IoU approaching zero), indicating improper handling of dequantization operations during TensorRT conversion. While QOperator representation maintained correct functional behavior, unsupported quantized operations forced CPU execution, negating potential acceleration benefits. These findings suggest that alternative quantization-aware training methodologies or direct TensorRT native quantization pipelines should be explored in future work.

4.5 Chapter Conclusion

This chapter systematically analyzed inference optimization strategies for deploying DL-based spacecraft detection and localization systems on Jetson Orin Nano hardware. The YOLOv8n and EfficientNet-B0 architectures were trained using the synthetic subset of the SPEED+ dataset and subsequently optimized through model format conversion, numerical precision reduction, and hardware-specific acceleration techniques.

The results demonstrated substantial acceleration potential, with TensorRT optimization at FP16 precision achieving a $5\times$ speedup for EfficientNet-B0 (136 FPS) and $5\times$ speedup for INT8-quantized YOLOv8n (129 FPS) relative to GPU baselines, thus enabling real-time on-orbit operations. Quantization efficacy was found to be architecture-dependent: while INT8 quantization proved effective for YOLOv8n with a moderate 6.3% accuracy degradation, it led to systematic failures for EfficientNet-B0 due to unsupported quantized operations, highlighting the need for alternative quantization strategies or architectural modifications.

Half-precision inference yielded negligible accuracy-speed trade-offs, maintaining near-identical accuracy compared to FP32 baselines (IoU difference $< 0.5\%$) while providing significant throughput improvements, thereby establishing FP16 as the most suitable precision configuration for embedded deployment. Batch processing further enhanced through-

put, with 16-image batches achieving 229 FPS. However, this approach introduced latency penalties in applications requiring real-time inference.

The optimized models also demonstrated strong alignment with existing literature, delivering performance improvements of 3–13 $\times$ over reported RemoveDEBRIS processing times and up to 169 $\times$ acceleration relative to prior Jetson Nano implementations. Collectively, these results confirm the feasibility of real-time, on-board inference for spacecraft detection tasks, exceeding operational frame rate requirements by a margin of two to four times.

While the optimization strategies presented in this chapter focused exclusively on the object detection module, these findings can be generalized to complete pose estimation pipelines integrating multiple neural network components. A detailed evaluation of a full multi-stage pose estimation pipeline deployed on embedded hardware is presented in Chapter 6.

Chapter 5

Addressing Domain Gap with CycleGAN

This chapter provides a comprehensive experimental evaluation of the hybrid three-stage pose estimation pipeline, with particular emphasis on assessing the capabilities of CycleGAN-based domain adaptation for mitigating the synthetic-to-real domain gap. Section 5.1 documents the baseline model architectures, training procedures, and hyperparameter configurations. Section 5.2 describes the implementation of CycleGAN, including a detailed account of its network architectures, training strategies, and the metrics used for translation quality assessment. Section 5.3 presents experimental results that encompass baseline cross-domain performance and evaluation of domain-adapted models. Section 5.4 discusses the identified limitations, notably translation artifacts and sensitivity to illumination conditions. Finally, Section 6.3 synthesizes the key findings and articulates the principal conclusions drawn from this investigation.

5.1 Models Training and Configuration

This section details the architectural choices, training procedures, and hyperparameter configurations employed for the two principal components of the spacecraft pose estimation pipeline: the object detection network and the keypoint regression network. Both models were trained on the synthetic partition of the SPEED+ dataset, as outlined in Table 5.1.

5.1.1 Dataset Overview

The SPEED+ dataset consists of synthetic and Hardware-In-the-Loop (HIL) images captured under two distinct illumination conditions, as shown in Table 5.1. The synthetic domain provides 47,966 training and 11,994 validation images, while the sunlamp and lightbox domains serve exclusively as test sets with 2,791 and 6,740 images, respectively. This configuration facilitates evaluation of cross-domain generalization capabilities under realistic operational conditions.

Table 5.1: SPEED+ Dataset Composition [5].

	Synthetic	Sunlamp	Lightbox
Train	47,966	–	–
Validation	11,994	–	–
Test	–	2,791	6,740

5.1.2 Object Detection Network: YOLO11n

The YOLO11n architecture was selected for the object detection stage based on its favorable trade-off between computational efficiency and detection accuracy. The nano variant offers reduced model complexity suitable for inference in resource-constrained environments while maintaining robust detection performance.

Data Augmentation Strategy

To enhance model robustness and mitigate overfitting to synthetic training data, a comprehensive augmentation strategy was implemented encompassing both geometric and photometric transformations. Table 5.2 summarizes the augmentation parameters applied during training.

Table 5.2: YOLO11n Data Augmentation Configuration.

Augmentation Type	Parameter	Value
Geometric	Rotation	$\pm 10^\circ$
	Translation	$\pm 12\%$
	Scale	$\pm 7\%$
	Shear	$\pm 5^\circ$
Photometric	HSV Saturation (hsv_s)	0.13554
	HSV Value/Brightness (hsv_v)	0.73636

Geometric transformations simulate variations in spacecraft orientation and camera positioning, while photometric adjustments account for illumination variability inherent in orbital environments. The relatively conservative augmentation magnitudes were chosen to preserve the structural integrity of spacecraft features while introducing sufficient diversity to improve generalization.

Training Configuration

The YOLO11n detector was trained using the AdamW optimizer with cosine annealing learning rate scheduling, as detailed in Table 5.3. This training strategy promotes stable convergence and prevents overfitting through explicit L2 regularization.

Table 5.3: YOLO11n Training Configuration.

Parameter	Value
Epochs	100
Batch Size	32
Image Size	[400, 640] px
Initial Learning Rate	0.00269
Final Learning Rate Ratio	0.0288
Weight Decay	0.00015
Learning Rate Schedule	Cosine Annealing
Optimizer	AdamW
Warmup Epochs	3
Early Stopping Patience	20
Save Period	Every 5 epochs
Random Seed	42

The learning rate schedule incorporated a three-epoch warmup phase to stabilize initial training dynamics. Early stopping with a patience of 20 epochs was employed to prevent overtraining and reduce computational overhead. All experiments utilized a fixed random seed to ensure reproducibility.

Training Dynamics

Figure 5.1 illustrates the training progression across all 100 epochs. The bounding box loss and distribution focal loss exhibit characteristic rapid descent during initial epochs, followed by gradual refinement. Validation metrics stabilize after the midpoint of training, indicating effective convergence and the absence of significant overfitting. Both precision and recall achieve values exceeding 0.98, while mean Average Precision at IoU threshold

0.5 (mAP@0.5) reaches 0.99, indicating excellent detection performance on the validation set.

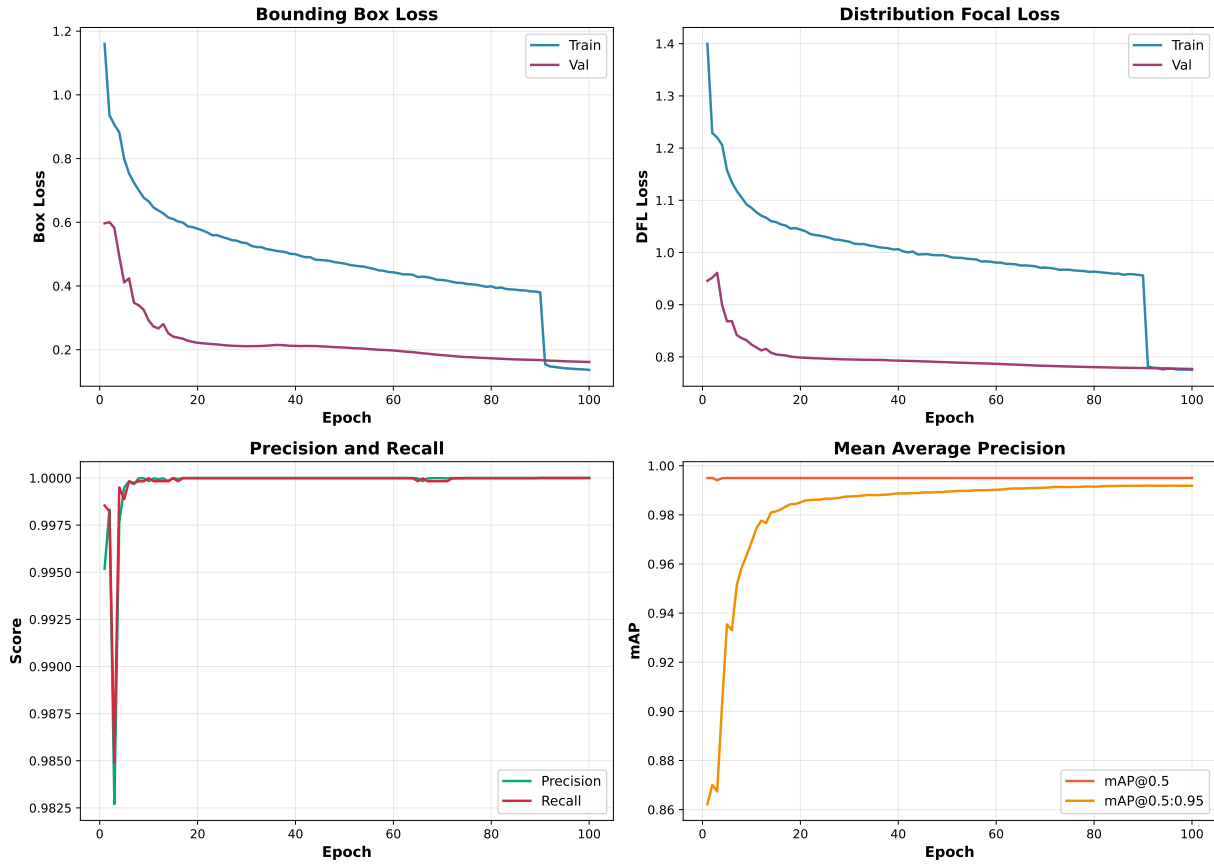


Figure 5.1: YOLO11n training metrics: Training and validation losses (top), precision/recall curves (bottom left), and mean Average Precision metrics (bottom right) across 100 training epochs.

5.1.3 Keypoint Regression Network: EfficientNet-B0

For keypoint localization, an EfficientNet-B0 backbone was employed due to its optimal balance between representational capacity and computational efficiency. The network architecture was adapted for direct coordinate regression, outputting 2D pixel coordinates for predefined spacecraft keypoints.

Data Augmentation Strategy

Given the sensitivity of keypoint localization to photometric and geometric variations, an extensive augmentation pipeline was implemented using the Albumentations library [180].

Table 5.4 presents the complete augmentation configuration, encompassing geometric transformations, photometric adjustments, and image degradation techniques.

Table 5.4: Keypoint Regression Data Augmentation Pipeline.

Type	Transform	Configuration	p
Geometric	Rotation	$\pm 15^\circ$ range	0.3
	Scale	Factor: (0.85, 1.15)	0.4
	Shear	$\pm 5^\circ$ maximum	0.4
	Translation	$\pm 15\%$ of dimensions	0.4
Photometric	Brightness/Contrast	$\pm 30\%$ variation	0.7
	HSV Shifts	Value: (0, 100)	0.6
	Solar Flare	Physics-based simulation	0.6
Degradation	Gaussian Noise	Per-channel disabled	0.1
	CLAHE	Clip: 4.0, Grid: 8×8	0.5
	Compression	Quality: (10, 100)	0.5
	Gamma Adjustment	Range: (80, 120)	0.5
	Blur	Kernel limit: 1	0.5
Normalization	ImageNet Stats	$\mu=[0.485, 0.456, 0.406]$	1.0

All transformations were implemented with keypoint-awareness to ensure spatial consistency between augmented images and corresponding annotations. Particular attention was devoted to simulating realistic imaging artifacts, including solar flare effects and compression artifacts, which are prevalent in space-based optical systems. Figure 5.2 provides representative examples of augmented training samples.

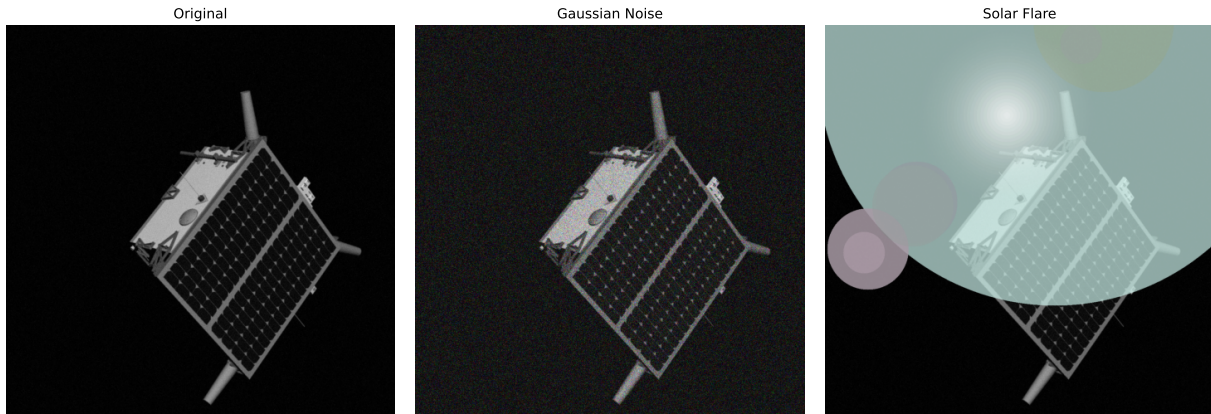


Figure 5.2: Augmentation pipeline examples: Visual demonstration of augmentation effects, including original image (left), Gaussian noise application (center), and physics-based solar flare simulation (right).

Training Configuration

The keypoint regression network was trained using SmoothL1Loss, a robust variant of L1 loss that exhibits reduced sensitivity to outliers through quadratic behavior near zero. This loss function is particularly well-suited for coordinate regression tasks where prediction errors may occasionally be large due to occlusions or challenging viewing geometries. Table 5.5 summarizes the complete training configuration. The AdamW optimizer was employed with cosine annealing scheduling, mirroring the approach adopted for the detection network to maintain methodological consistency.

Table 5.5: Keypoint Regression Training Configuration.

Parameter	Setting
Training Epochs	100
Batch Size	32
Loss Function	SmoothL1Loss
Optimizer	AdamW
Base Learning Rate	1e-4
Weight Decay	1e-5
LR Scheduler	CosineAnnealingLR
Early Stopping Patience	20 epochs
Minimum LR	1e-6

Training Dynamics

Model performance was quantitatively assessed using RMSE metric. Figure 5.3 depicts the training and validation loss curve evolution of RMSE throughout training. The rapid initial descent in the metric indicates effective learning, while the convergence of training and validation curves around epoch 40 suggests appropriate regularization. The final RPMSE stabilizes below 5 px on the validation set, demonstrating high-precision keypoint localization on synthetic data.

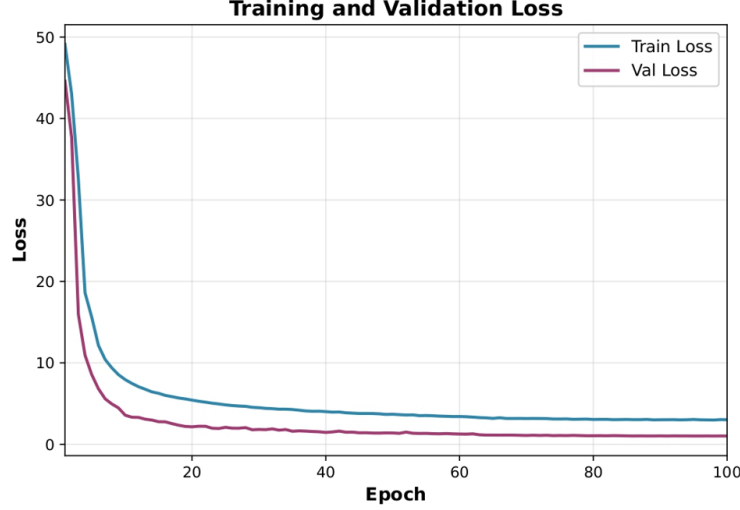


Figure 5.3: Training and validation RMSE loss curve evolution across 100 epochs.

5.2 CycleGAN Domain Adaptation

5.2.1 Implementation Details

Separate CycleGAN models were independently trained for the lightbox and sunlamp target domains to systematically transform synthetic satellite imagery into realistic visual counterparts exhibiting domain-specific photometric and textural characteristics while preserving underlying geometric structure essential for accurate pose estimation.

Data Preparation: Training datasets were carefully balanced between synthetic source images and real target domain images — 2,791 image pairs for the sunlamp domain and 6,740 pairs for the lightbox domain. All images underwent standardized preprocessing, being resized to 400×640 px and normalized to the $[-1, 1]$ range required by the tanh output activation. Stochastic horizontal flipping (probability $p = 0.5$) and vertical flipping ($p = 0.2$) were applied as effective augmentation strategies to enhance translation robustness.

Network Architecture: Generator networks employed ResNet-style encoder-decoder architectures [183] incorporating 9 residual blocks providing sufficient representational capacity for complex image-to-image translation while maintaining reasonable computational requirements. The encoder pathway contained an initial 7×7 convolution layer with reflection padding and instance normalization, succeeded by two progressive downsampling blocks exponentially expanding channel dimensions from 64 to 256 through strided convolutions. Following residual transformation, two transposed convolutional layers systematically upsampled feature representations back to 64 channels before a final 7×7

convolution with tanh activation produced single-channel grayscale output. Discriminator networks employed PatchGAN architectures [150] featuring progressive 4×4 convolutions expanding from 64 to 512 channels, thereby focusing discriminative capacity on local texture authenticity rather than global image coherence.

Table 5.6: CycleGAN Training Configuration.

Hyperparameter	Value
Training Epochs	200
Batch Size	1
Image Resolution	400×640 px
Learning Rate	2e-4
Optimizer	AdamW
β Parameters	(0.5, 0.999)
Weight Decay	1e-5
λ_{cycle}	10.0
$\lambda_{identity}$	0.0
Gradient Clipping	1.0
LR Decay Start	Epoch 100

Training Strategy: The standard CycleGAN protocol [16] was followed with cycle consistency weight $\lambda_{cyc} = 10.0$ to enforce bidirectional translation consistency and preserve structural information. Identity loss was deliberately disabled ($\lambda_{identity} = 0$) as color preservation proves irrelevant for grayscale satellite imagery where only intensity distributions require adaptation. Critical stabilization techniques included gradient clipping with maximum norm 1.0 to prevent training collapse, maintenance of a 50-image replay buffer to reduce discriminator oscillation and mode collapse, and halving of discriminator loss magnitudes to balance adversarial learning rates between generator and discriminator updates. Learning rates underwent linear decay to zero commencing at epoch 100 to promote final convergence refinement. The summary of all hyperparameters is shown in Table 5.6.

5.2.2 Quality Assessment

Translation quality was comprehensively evaluated through both qualitative visual inspection and quantitative Fréchet Inception Distance (FID) score [169, 184], a perceptual metric measuring distributional similarity between generated and real image features within deep feature spaces. Translated images successfully maintained structural integrity and geometric fidelity while effectively adopting target domain visual characteristics including texture patterns, noise distributions, and illumination properties. The lightbox translation achieved an exceptionally low FID score of 19.62, indicating excellent domain alignment

approaching human-imperceptible quality thresholds (FID scores below 20 represent high-fidelity adaptation by established convention). Sunlamp translations yielded a notably higher FID score of 59.93, reflecting the increased difficulty of translating extreme illumination conditions characterized by harsh shadows and specular highlights. Translation examples for lightbox and sunlamp domains are illustrated in Figure 5.4 and Figure 5.5 respectively.

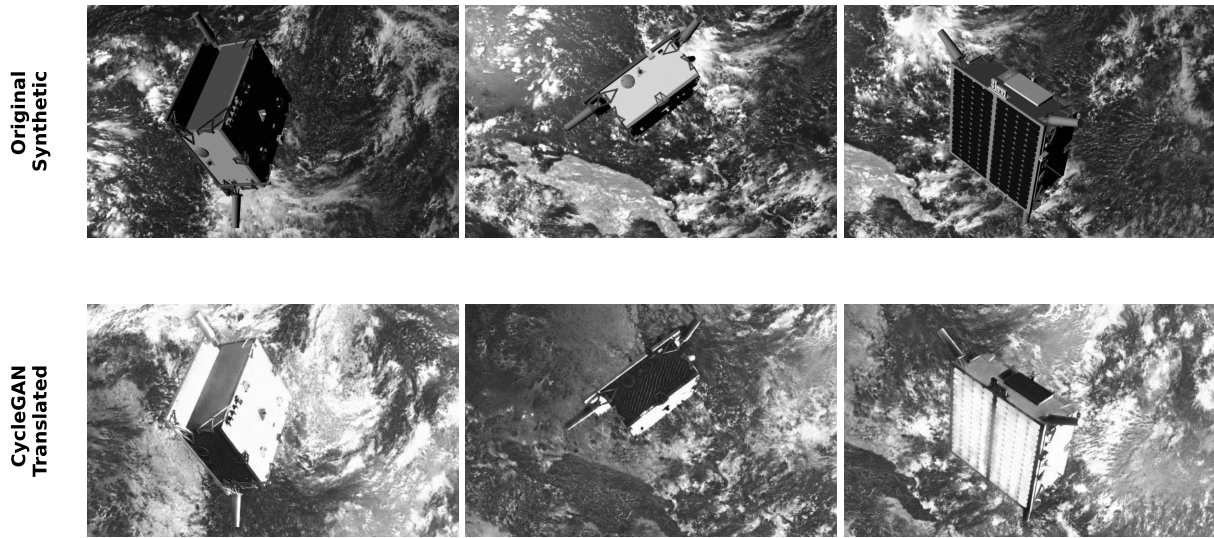


Figure 5.4: Domain translation results for lightbox conditions: Comparison between original synthetic images (top row) and corresponding CycleGAN-translated outputs (bottom row).

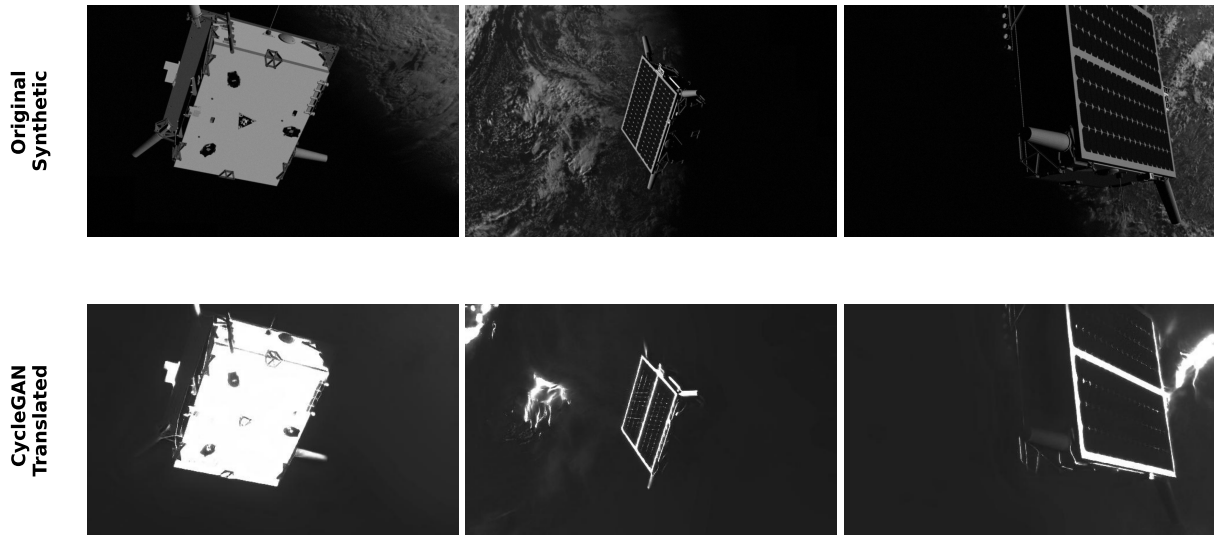


Figure 5.5: Domain translation results for sunlamp conditions: Original synthetic renderings (top row) and CycleGAN-translated counterparts (bottom row).

5.3 Experimental Results

5.3.1 Baseline Performance Assessment

Baseline models trained exclusively on synthetic SPEED+ data comprising 47,966 training images established comprehensive performance benchmarks quantifying the magnitude of the domain gap as shown in Table 5.7. Both the YOLO11n object detector and EfficientNet-B0 keypoint regressor demonstrated stable training convergence over 100 epochs under cosine annealing learning rate schedules, achieving near-optimal performance on synthetic validation set.

Object Detection Performance: On synthetic test data, the YOLO11n achieved near-perfect performance with $\text{mAP}@0.5:0.95$ of 0.989 and mean IoU of 0.973, confirming excellent architectural capacity for the detection task. However, direct transfer to HIL images revealed substantial performance degradation attributable to domain gap. Lightbox domain evaluation showed mAP declining precipitously to 0.607 representing a 39% performance reduction, while IoU fell to 0.748 indicating a 23% degradation in localization precision. Sunlamp conditions exhibited further degradation with mAP of 0.582 and IoU of 0.721, likely attributable to increased lighting variations and specular reflections. Most critically, $\text{AP}@0.95$ — the stringent precision metric most relevant for downstream pose estimation, plummeted from 0.906 to 0.169 for lightbox and 0.145 for sunlamp, indicating severely compromised high-precision localization essential for accurate keypoint extraction.

Keypoint Localization Performance: The domain gap manifested even more severely for keypoint regression, representing the pipeline’s critical bottleneck. RMSE underwent dramatic escalation, increasing 13-fold to 28.52 px for lightbox and 15-fold to 34.18 px for sunlamp compared to the baseline 2.24 px achieved on synthetic data. The $\text{PRMSE}@10$ metric, measuring the proportion of keypoints localized within 10-pixel radius, dropped from 0.746 to merely 0.135 for lightbox and 0.089 for sunlamp. The stringent $\text{PRMSE}@2$ metric fell drastically from 0.122 to 0.009 and 0.005 respectively, indicating near-complete failure of precise keypoint localization under domain gap.

End-to-End Pose Estimation: These cascading errors propagated destructively through the EPnP solver, precipitating dramatic pose accuracy degradation rendering the baseline system essentially non-functional for real-world deployment. Orientation error increased dramatically from 1.79° to 38.45° for lightbox and 48.76° for sunlamp, representing 21-fold and 27-fold performance deterioration respectively. Position error similarly degraded from 0.057 m to 0.642 m for lightbox and 0.834 m for sunlamp, representing 11-fold and 15-fold increases. The composite pose score increased from 0.041 to 0.773 for lightbox and 0.981 for sunlamp, representing 19-fold and 24-fold degradation, confirming the severe practical limitations of direct synthetic-to-real transfer without domain adaptation.

Table 5.7: Baseline Cross-Domain Performance Evaluation.

Component	Metric	Synthetic	Lightbox	Sunlamp
Object Detection	Detection Rate (%)	100.00	99.01	98.67
	Mean IoU	0.973	0.748	0.721
	AP@0.5	1.000	0.837	0.805
	AP@0.75	0.999	0.642	0.598
	AP@0.95	0.906	0.169	0.145
	mAP@0.5:0.95	0.989	0.607	0.582
Keypoint Detection	RMSE (px)	2.24	28.52	34.18
	PRMSE@2	0.122	0.009	0.005
	PRMSE@5	0.466	0.048	0.028
	PRMSE@10	0.746	0.135	0.089
	PRMSE@20	0.896	0.295	0.218
Pose Estimation	Position Error (m)	0.057	0.642	0.834
	Orientation Error ($^{\circ}$)	1.79	38.45	48.76
	Pose Score	0.041	0.773	0.981

The severity of cross-domain performance degradation is visually evident in Figure 5.6, which contrasts model predictions across synthetic, lightbox, and sunlamp domains.

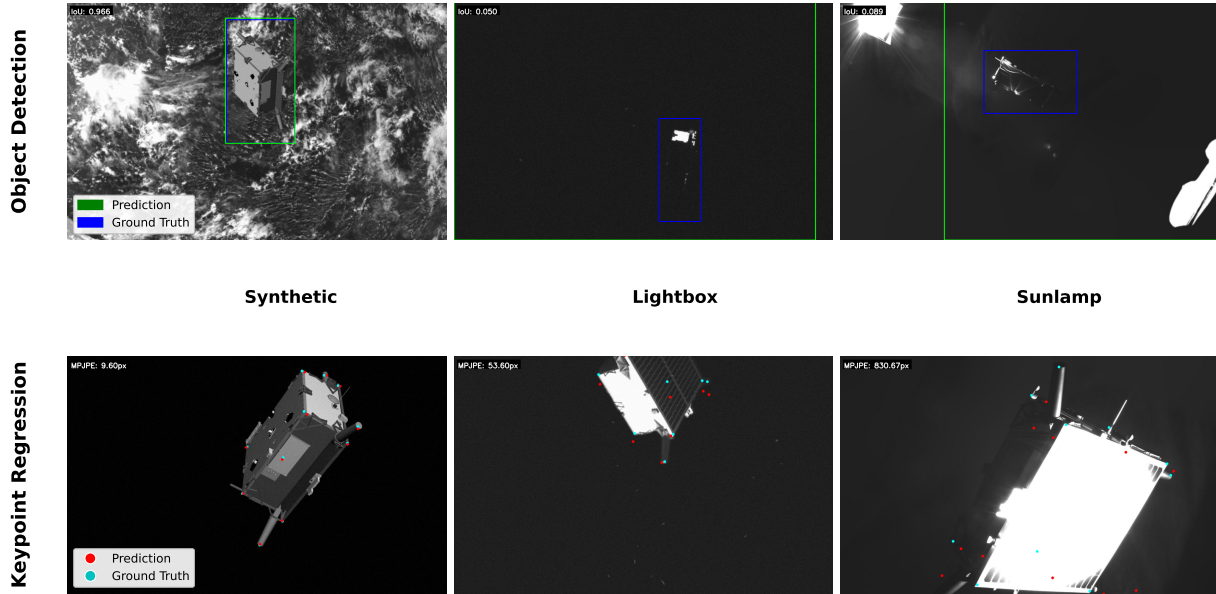


Figure 5.6: Baseline performance visual examples across domains, clearly illustrating severe performance degradation under domain gap.

5.3.2 Domain-Adapted Model Performance

Two distinct retraining strategies were systematically evaluated utilizing CycleGAN-translated datasets: training from random initialization (Strategy 1) and fine-tuning from baseline model weights (Strategy 2), enabling assessment of transfer learning benefits versus complete retraining. The detailed evaluation results are shown in Table 5.8.

Lightbox Domain Results: Both training strategies achieved substantial performance improvements validating the domain adaptation hypothesis. Fine-tuning (Strategy 2) proved marginally yet consistently superior, reducing orientation error dramatically from 38.45° to 19.85° representing a remarkable 48% improvement, while simultaneously reducing position error from 0.642 m to 0.305 m indicating a 52% enhancement. The pose score improved substantially from 0.773 to 0.398, representing a 49% reduction in overall pose error. Object detection mAP increased substantially to 0.876 restoring performance to within 11% of synthetic baseline, while keypoint RMSE decreased markedly to 16.28 px, still elevated compared to synthetic performance yet representing a 43% improvement over unadapted baseline. The minimal performance differential between training strategies of 1.6° in orientation error, 0.030 m in position error, and 0.038 in pose score suggests that domain-adapted imagery itself provides the predominant performance benefit rather than the specific training initialization methodology, indicating that translation quality constitutes the primary determinant of adaptation success.

Sunlamp Domain Results: Based upon lightbox experimental findings only Strategy 2 (fine-tuning) was applied to the sunlamp domain to optimize computational resource allocation. Results revealed markedly limited adaptation effectiveness reflecting the severe translation challenges posed by extreme illumination conditions. Orientation error improved modestly by 10% from 48.76° to 43.85° , while position error improved by 19% from 0.834 m to 0.674 m. The pose score improved marginally from 0.981 to 0.857, representing only a 13% enhancement. Object detection mAP declined marginally from 0.582 to 0.565, though the critical high-precision AP@0.95 metric improved significantly by 68% from 0.145 to 0.244. Keypoint localization showed minimal improvement with RMSE decreasing only slightly by 7% from 34.18 px to 31.75 px. These limited improvements suggest that the challenging sunlamp illumination regime characterized by harsh shadows and extreme contrast approaches the limits of CycleGAN’s effective translation capabilities, necessitating more sophisticated domain adaptation methodologies for such extreme conditions.

Table 5.8: Domain Adaptation Results: Lightbox and Sunlamp Domains.

Task	Metric	Lightbox			Sunlamp	
		Baseline	Strat. 1	Strat. 2	Baseline	Strat. 2
Object Detection	Detection Rate (%)	99.01	99.99	99.99	98.67	98.45
	Mean IoU	0.748	0.910	0.910	0.721	0.745
	AP@0.5	0.837	0.991	0.993	0.805	0.825
	AP@0.75	0.642	0.947	0.946	0.598	0.642
	AP@0.95	0.169	0.407	0.413	0.145	0.244
	mAP@0.5:0.95	0.607	0.873	0.876	0.582	0.565
Keypoint Detection	RMSE (px)	28.52	17.54	16.28	34.18	31.75
	PRMSE@2	0.009	0.015	0.021	0.005	0.008
	PRMSE@5	0.048	0.092	0.115	0.028	0.042
	PRMSE@10	0.135	0.268	0.315	0.089	0.125
	PRMSE@20	0.295	0.542	0.598	0.218	0.285
Pose Estimation	Position Error (m)	0.642	0.335	0.305	0.834	0.674
	Orientation Error (°)	38.45	21.45	19.85	48.76	43.85
	Pose Score	0.773	0.436	0.398	0.981	0.857

5.4 Discussion and Limitations

5.4.1 Performance Contextualization

Comparative analysis with SPEC2021 competition results presented in Table 5.9 provides essential context for evaluating the domain adaptation framework. The adapted lightbox system achieved performance positioning between the 6th and 7th ranked competition entries, demonstrating competitive capability relative to established benchmarks. Specifically, orientation error of 19.85° and position error of 0.305 m fall within the performance envelope of mid-tier competition systems, while remaining distant from top-tier results (3.187° and 0.105 m respectively). The sunlamp domain presented considerably greater challenge reflecting the fundamental difficulty of extreme illumination adaptation. Orientation error of 43.85° and position error of 0.674 m indicate substantial remaining improvement potential relative to top entries of the competition.

The principal contribution of this work resides in demonstrating that CycleGAN-based domain adaptation achieves practically meaningful synthetic-to-real transfer utilizing exclusively unpaired imagery, without requiring complex multi-component systems or training on target domain with pseudo labels. However, adversarial training approach adopted by the SPEC2021 competition winner has proven to be the most effective for bridging the domain gap. In this method, the landmark regression model is trained in alternating steps: the discriminator is updated to better classify whether feature heatmaps originate

from the synthetic or real domain, while the generator is updated both to minimize the heatmap loss and to fool the discriminator by making its outputs domain-invariant. This adversarial interaction robustly aligns the feature distributions between domains and leads to superior cross-domain performance.

Table 5.9: Performance Comparison with SPEC2021 Competition Results.

Domain	Metric	1st Place	Present Approach
Lightbox	Position Error (m)	0.105	0.305
	Orientation Error ($^{\circ}$)	3.187	19.85
	Pose Score	0.073	0.398
Sunlamp	Position Error (m)	0.065	0.674
	Orientation Error ($^{\circ}$)	2.728	43.85
	Pose Score	0.059	0.857

5.4.2 Identified Limitations

It is important to note that, for some images, the CycleGAN networks experienced translational problems that produced visually perceptible artifacts concentrated at the image boundaries and in high-contrast regions, directly compromising keypoint localization precision. Figure 5.7 illustrates representative failure cases where CycleGAN produces spurious texture patterns and boundary discontinuities. The sunlamp domain particularly suffers when the synthetic-to-real visual discrepancy exceeds CycleGAN’s effective translation capacity, resulting in geometric distortions that severely degrade downstream pose estimation performance.

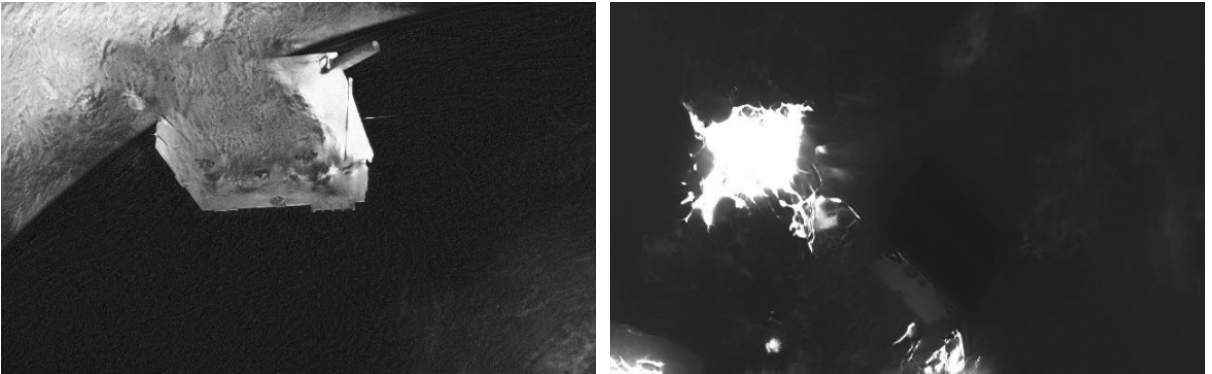


Figure 5.7: Representative translation failure examples demonstrating artifacts and distortions: lightbox domain (left) exhibits boundary discontinuities and spurious textures, while sunlamp domain (right) shows severe geometric distortions under extreme illumination conditions that compromise keypoint localization accuracy.

5.5 Chapter Conclusion

This investigation systematically quantified the synthetic-to-real domain gap for spacecraft pose estimation and demonstrated that CycleGAN-based adaptation substantially mitigated this gap through unsupervised image-to-image translation while preserving geometric structure essential for pose recovery.

Baseline cross-domain transfer produced severe degradation, with orientation error escalating from 1.79° to 38.45° for lightbox and 48.76° for sunlamp. Position error degraded from 0.057 m to 0.642 m for lightbox and 0.834 m for sunlamp. Keypoint localization error increased 13-fold to 28.52 px for lightbox and 15-fold to 34.18 px for sunlamp. Composite pose score degraded from 0.041 to 0.773 for lightbox and 0.981 for sunlamp.

CycleGAN-based adaptation achieved substantial performance recovery for the lightbox domain, demonstrating 48% orientation error reduction (38.45° to 19.85°), 52% position error improvement (0.642 m to 0.305 m), 49% pose score enhancement (0.773 to 0.398), and 43% keypoint localization improvement (28.52 px to 16.28 px). These improvements elevated system performance to position between the 6th and 7th ranked SPEC2021 competition entries, validating that unsupervised translation meaningfully reduced domain gap.

The sunlamp domain presented fundamental challenges that approached CycleGAN’s translation capability limits. Orientation error improved modestly by 10% (48.76° to 43.85°), position error improved 19% (0.834 m to 0.674 m), pose score improved 13% (0.981 to 0.857), while keypoint localization showed minimal 7% enhancement (34.18 px to 31.75 px). These limited improvements indicated that CycleGAN-based adaptation alone proved insufficient for extreme domain gaps, necessitating more sophisticated approaches such as adversarial feature alignment.

While the absolute performance remained below top-tier competition methods, this research demonstrated that a practically meaningful synthetic-to-real transfer can achieve mid-tier competitive results on the lightbox domain. Moreover, the lightweight pipeline architecture (YOLO11n + EfficientNet-B0, 7.9 M parameters), although contributing to the reduced network performance, was deliberately chosen to ensure computational efficiency suitable for resource-constrained spacecraft processors — a factor not considered in SPEC2021.

The domain adaptation evaluation presented in this chapter was conducted on the SPEED+ dataset, which comprises discrete laboratory acquisitions rather than contiguous RPO trajectory sequences. Realistic on-orbit operations introduce additional complexities including temporal correlation between consecutive observations, constrained viewing geometries during approach trajectories, varying illumination conditions throughout the rendezvous, and time-critical computational constraints that impact both adaptation

feasibility and pose estimation performance. Chapter 6 addresses these operational considerations by evaluating pose estimation performance on trajectory-based datasets that represent realistic RPO scenarios.

Chapter 6

SAFE-pose validation on STRIDE Dataset

This chapter presents the experimental validation of the Safe Autonomous Fallback for Estimation of Pose (SAFE-pose) pipeline using the Spacecraft Trajectory Imagery for Docking Estimation (STRIDE) dataset. The objective is to evaluate the accuracy, reliability, and operational robustness of the proposed DL-based pose estimation approach under conditions where the network exhibits uncertainty in its predictions. Section 6.1 introduces the STRIDE dataset, outlining the data acquisition methodology, key characteristics, and configuration of the Zero-G Lab facility. Subsequently, the SAFE-pose architecture and its hierarchical fallback mechanism are described in detail, followed by a quantitative evaluation of the pipeline’s performance in Section 6.2. Chapter concludes with Section 6.3.

6.1 STRIDE Dataset

6.1.1 Zero-G Lab

The Zero-G Lab [21] at the University of Luxembourg represents a state-of-the-art facility specifically designed for emulating orbital mechanics and spacecraft dynamics under controlled terrestrial conditions. The facility, shown in Figure 6.1, comprises a $7\text{ m} \times 6\text{ m} \times 2.3\text{ m}$ enclosed environment that integrates multiple robotic systems, precision instrumentation, and advanced sensing capabilities to replicate space operations with unprecedented fidelity. The laboratory infrastructure consists of two Universal Robots UR10e [185] robotic

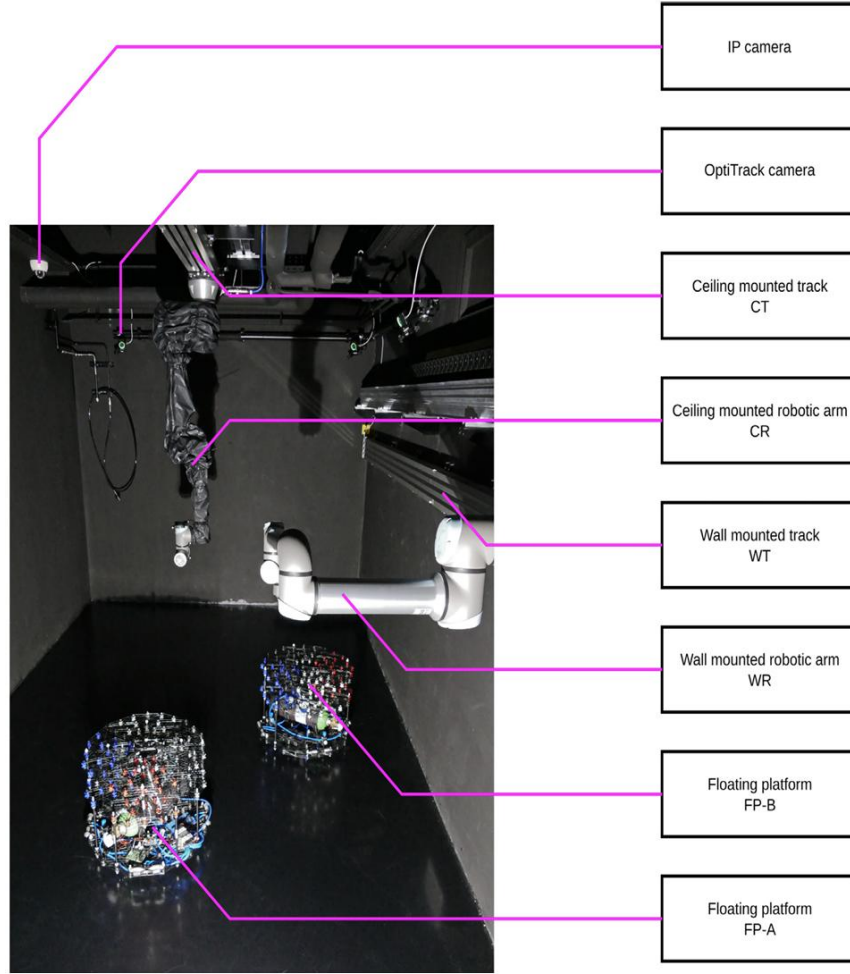


Figure 6.1: Zero-G lab facility [21].

manipulators, each providing six degrees of freedom with a working radius of 1300 mm. These manipulators are mounted on independent linear rail systems (one ceiling-mounted with a stroke length of 3204 mm and one wall-mounted with 4330 mm), effectively providing a seventh degree of freedom for extended workspace coverage. This configuration enables precise emulation of complex relative motion between spacecraft during proximity operations, accommodating both chaser and target dynamics simultaneously.

Central to the facility’s micro-gravity emulation capability is a $5\text{ m} \times 3\text{ m}$ super-flat epoxy floor, engineered to micron-scale precision with a flatness tolerance maintained within $25\text{ }\mu\text{m}$ across the entire surface. This exceptional surface quality enables frictionless motion of pneumatic floating platforms through air-bearing technology, effectively reproducing two-dimensional orbital dynamics. Although this air-bearing floor is part of the available laboratory infrastructure, it is not employed in the experimental campaign presented in this chapter and is instead considered for use in future investigations.

The facility incorporates an OptiTrack Prime 13W [186] motion capture system com-

prising fourteen cameras strategically positioned throughout the operational volume. Operating at 240 Hz with sub-millimeter position tracking accuracy, this system provides real-time localization of all robotic systems, floating platforms, and experimental apparatus within the laboratory space. The high-frequency, high-precision tracking capability enables closed-loop control of dynamic experiments and provides ground truth measurements for validation purposes; however, in the specific spacecraft configuration with large solar panels considered in this chapter, occlusions prevented sufficient visibility of the markers defining the complete 3D model of the mockup, and the system was therefore not used, but it remains a valuable asset for future work.

To replicate the unique illumination conditions encountered in the space environment, the Zero-G Lab features a Godox SL-60W [187] sun simulator capable of producing collimated light at 5600 K color temperature, closely approximating the spectral characteristics of unfiltered solar radiation. This illumination system, with an irradiance capability approaching 1.35 kW/m^2 , can be mounted either statically on dedicated rails or dynamically on the robotic manipulators, enabling simulation of varying solar angles and the harsh lighting contrasts characteristic of orbital operations.

6.1.2 Dataset Acquisition

Real subset

The STRIDE dataset acquisition leverages the comprehensive infrastructure of the Zero-G Lab to generate real imagery representative of spacecraft rendezvous and docking scenarios under diverse operational conditions. The experimental setup, illustrated in Figure 6.2, employs a systematic approach to capture imagery while maintaining precise ground truth measurements. The target spacecraft mockup, equipped with a high-contrast ArUco [188] fiducial marker for validation purposes, is mounted on robotic arm 2, whereas the camera

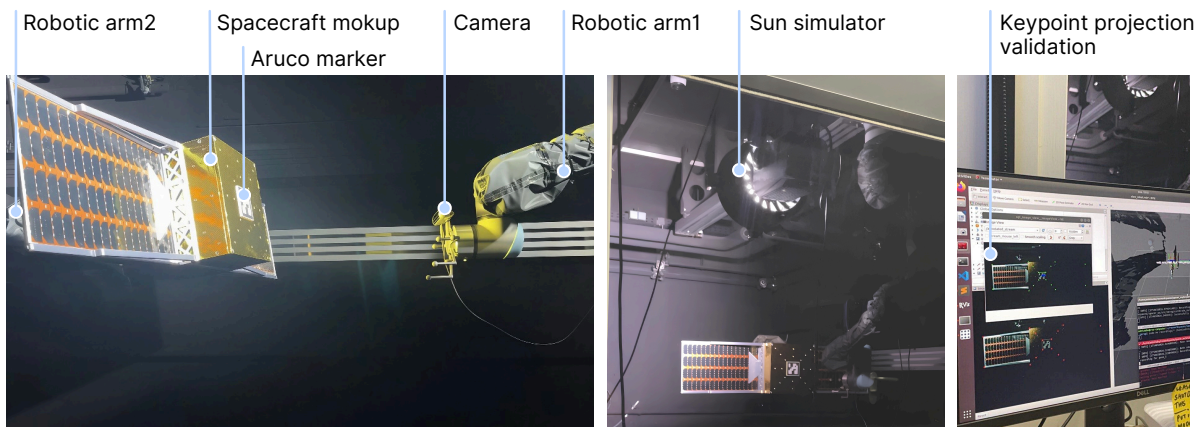


Figure 6.2: STRIDE dataset acquisition setup.

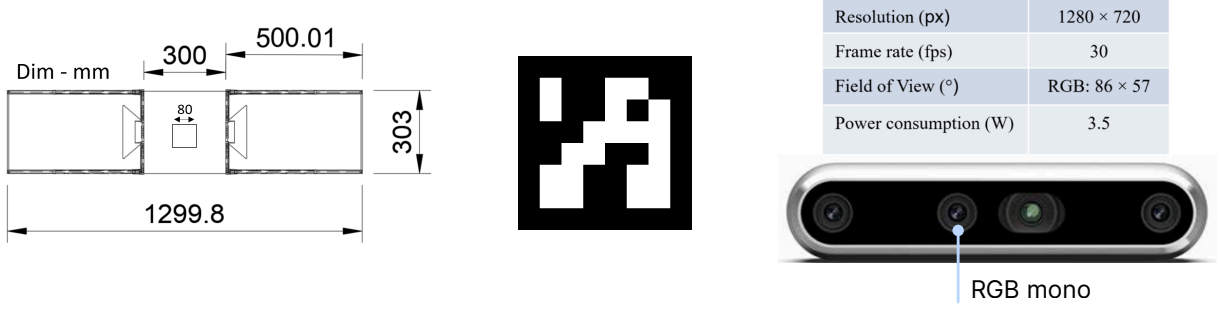


Figure 6.3: Mockup, ArUco marker and RealSense D455 camera.

is attached to robotic arm 1. During image acquisition, both the sun simulator and the keypoint projection assessment module are activated. The adopted camera mounting configuration enables precise positioning with respect to the target mockup by exploiting the robotic manipulator’s kinematic chain. Details regarding the target model, the ArUco marker dimensions, and the camera specifications are provided in Figure 6.3. For real image acquisition, an Intel RealSense D455 [189] camera is employed as the primary vision sensor. It is configured with a focal length of 1.93 mm and operates at a resolution of 1280×720 px. Although the camera features stereo imaging capabilities, it is used in monocular mode for the collection of this dataset.

Camera calibration was performed using a two-stage process: the Moveit [190] calibration tool provided hand-eye calibration parameters, while the ROS RealSense [191] package yielded intrinsic camera parameters. This comprehensive calibration ensures accurate projection of 3D spacecraft keypoints onto the image plane, with residual calibration errors quantified and compensated during post-processing as represented in figure 6.4. A small static offset remains visible for keypoints located at the outer regions of the solar panels, possibly due to reduced intrinsic calibration accuracy near the image borders. However, since this work focuses on the four marker keypoints at the spacecraft’s core structure, the impact of this residual offset on the overall pose accuracy is limited.

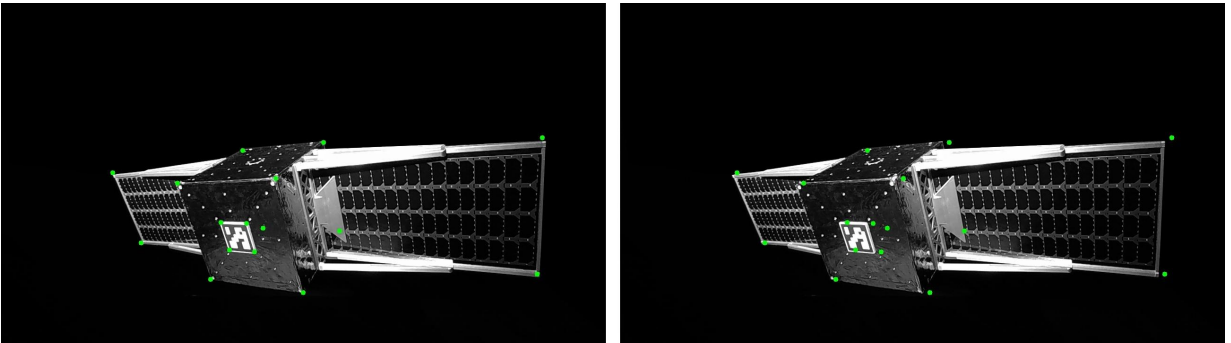


Figure 6.4: Comparison of the projected keypoints before and after calibration.

Synthetic subset

A synthetic data generation process utilizes Blender. The basic scene with the camera and spacecraft models is shown in Figure 6.5. The camera settings were configured with a focal length of 1.93 mm and a resolution of 1280×720 px to match the parameters of the Intel RealSense D455 camera used for capturing real validation images of the mockup. A CAD model of the spacecraft, provided by the Zero-G Lab, was equipped with an ArUco marker to enable validation. To further enhance variability and promote generalization in the DL models, the strength of the normal map texture of spacecraft multi-layer insulation (MLI) was randomly altered between 0.1 and 3 for each rendering. This procedure accounts for the challenge of replicating the exact MLI shape present in real-world conditions. For each image, the spacecraft’s position and orientation relative to the camera were randomized, with the relative distance to the camera set between 0.33 and 1 m. Therefore, the minimum distance between the marker and the camera was constrained to 15 cm, which could correspond to the typical length of the end-effector or gripper of a robotic arm on an active servicing chaser. Each frame was retained only if several requirements were satisfied: the marker was at least partially within the camera’s field of view (verified via frustum culling); the marker was not occluded by the spacecraft itself, confirmed through ray casting from the camera to the marker; and at least 2 marker corner-keypoints were illuminated by direct simulated sunlight, also checked using ray casting. Rendering was performed with the physically based Cycles engine to achieve photorealistic results through ray tracing. For each validated frame, camera pose parameters were exported as ground-truth labels in JSON format. A synthetic dataset 40,000 grayscale images featuring spacecraft in random poses and under different illumination conditions was procedurally generated. Some examples are show in Figure 6.6.

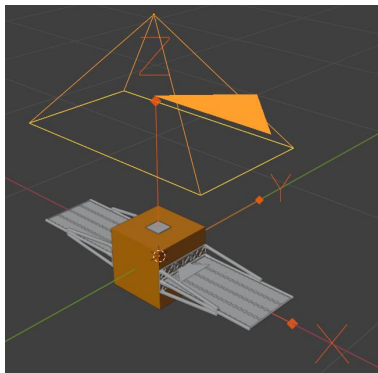


Figure 6.5: Blender setup comprising camera and mockup model.

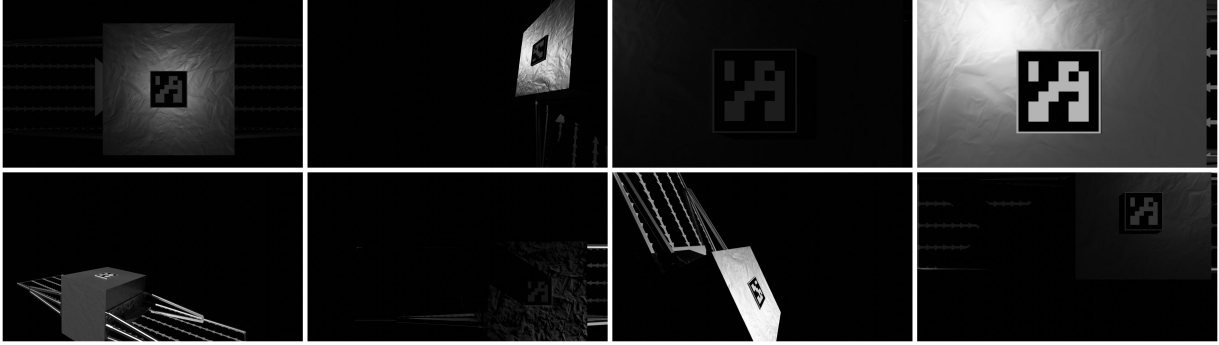


Figure 6.6: Examples from the synthetic subset of the STRIDE dataset.

6.1.3 Real subset Description

The real subset comprises 1,044 grayscale HIL images across 22 trajectory cases with respect to different illumination conditions. Some examples from the dataset are shown in Figure 6.7. Target-to-camera distances, measured from the geometric center of the spacecraft mockup, range from 0.33 m to 0.79 m. The minimum marker-to-camera distance of 0.15 m was selected to account for the potential presence of the servicer’s robotic arm gripper that may be used in a real on-orbit capturing scenario. The maximum distance was empirically determined to ensure that the entire spacecraft mockup fits within the camera’s FOV at the beginning of each image acquisition trajectory.

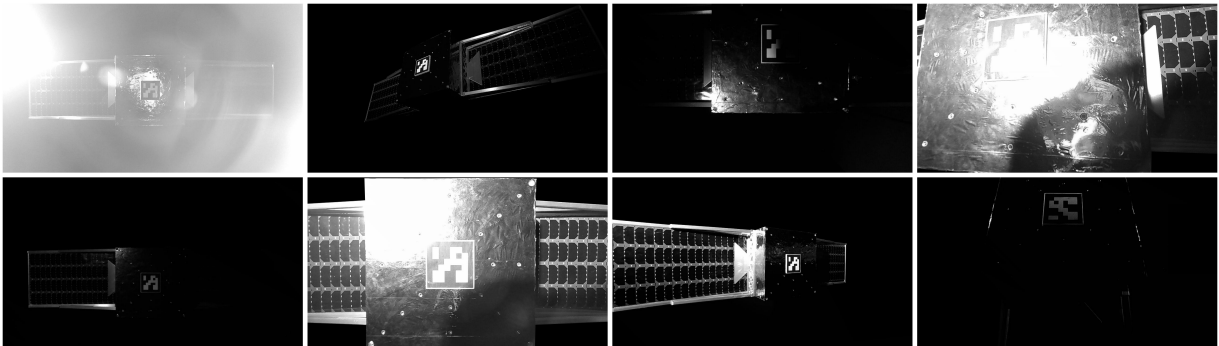


Figure 6.7: Examples from the real subset of the STRIDE dataset.

Systematic variation of illumination conditions was achieved through controlled adjustment of the sun simulator’s position, orientation and intensity. The illumination parameters varied include the azimuth angle (α) ranging from -90° to $+90^\circ$, the elevation angle (β) from 0° to 75° , the displacement of the sun simulator position along the z axis of the global reference frame (see Figure 6.8), and intensity modulation from 10% to 100% of maximum output. This comprehensive coverage ensures that the dataset captures the full spectrum of lighting conditions encountered during orbital operations, from direct solar illumination causing sensor saturation to oblique lighting creating deep shadows and partial occlusions.

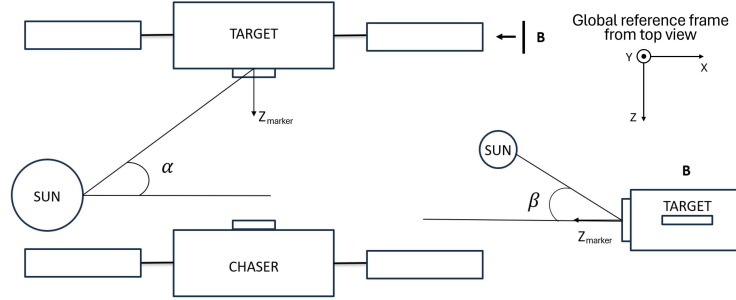


Figure 6.8: Top and side views of the chaser-camera approaching target characterizing α and β angles.

Table 6.1 summarizes the systematic orientation variations applied across the dataset cases. Cases 1–5 and 7–15 maintain near-zero spacecraft rotations relative to the camera to emulate nominal docking approaches, during which the servicer and target spacecraft are required to be precisely aligned. In contrast, Cases 6 and 16–22 introduce static attitude deviations to replicate conditions where perfect alignment cannot be achieved due to operational constraints or limitations in the servicer’s approach trajectory.

Table 6.1: Mockup orientation with respect to camera reference frame (Cases 1-22).

Metric	Case										
	1	2	3	4	5	6	7	8	9	10	11
num. imgs	99	42	42	42	42	45	100	42	42	42	42
x (deg)	0.59	0.34	0.30	0.31	0.32	0.05	0.37	0.30	0.30	0.31	0.29
y (deg)	-0.65	-0.67	-0.68	-0.67	-0.67	14.55	-0.65	-0.67	-0.67	-0.68	-0.67
z (deg)	0.26	0.28	0.28	0.28	0.28	-1.36	0.31	0.29	0.29	0.30	0.29
Metric	Case										
	12	13	14	15	16	17	18	19	20	21	22
num. imgs	42	42	42	100	50	42	47	44	46	43	42
x (deg)	0.30	0.30	0.30	0.34	19.43	-9.88	49.72	-29.02	-50.41	0.57	0.20
y (deg)	-0.67	-0.68	-0.69	-0.66	25.97	9.12	-20.70	26.01	16.21	-30.66	-23.39
z (deg)	0.29	0.29	0.29	0.31	-2.90	-1.28	-22.40	-13.37	-84.89	-0.08	-89.78

Figure 6.9 shows the first and last images of each case, arranged vertically from top to bottom, and captured along the image acquisition sequence. Case 17 corresponds to the most challenging illumination condition, where direct sunlight completely obscures the marker visibility for part of the initial distance range. In such situations, a marker-based pose estimation method becomes ineffective, and other model-based solutions, such as the CRPEP framework with an active switching mechanism described in Chapter 3, can be employed to estimate the full spacecraft pose. Similarly, in cases such as 18 and 20, the marker is visible only at the initial stage of the image acquisition process. For these scenarios, a

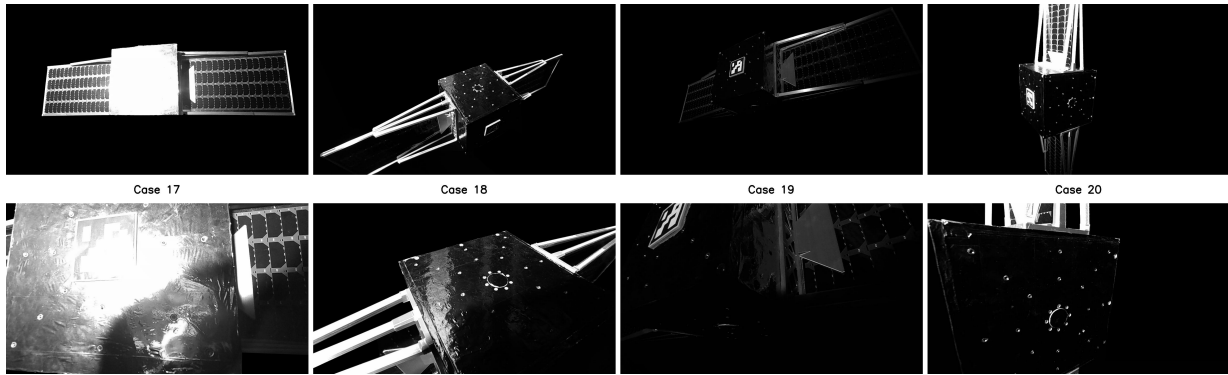


Figure 6.9: Examples from non-rectilinear cases, top panel—furthest, bottom panel—closest image.

marker-based approach is also not feasible, necessitating the adoption of alternative pose estimation strategies.

6.2 SAFE-Pose Estimation Pipeline

Pipeline Architecture

The Safe Autonomous Fallback for Estimation of pose (SAFE-pose) pipeline implements a hierarchical processing strategy that integrates DL-based keypoint detection with classical computer vision fallback mechanisms. This hybrid architecture, illustrated in Figure 6.10, addresses the need for robust and reliable spacecraft pose estimation approach, particularly in scenarios where the DNN exhibits high uncertainty in its predictions.

The overall procedure is outlined as follows. Initially, a YOLO11n [3] ODN processes the full resolution input image to localize the spacecraft, generating a bounding box that defines the ROI. This lightweight network, with only 2.6 million parameters, provides rapid initial detection with moderate accuracy degradation relative to more parameter-rich network variants.

Following ROI extraction, the cropped and resized image region is processed by an EfficientNet-B0-based [1] KRN that directly predicts pixel coordinates for predefined spacecraft keypoints. The network architecture, comprising 5.3 million parameters, was selected to balance model capacity against inference speed requirements.

Central to the pipeline’s reliability is the implementation of Monte Carlo Dropout [192] uncertainty quantification. During inference, the KRN performs 5 forward passes with dropout probability of 0.3 maintained active, generating a distribution of keypoint predictions. From this distribution, both mean predictions and uncertainty estimates are computed, with the latter serving as a reliability metric for triggering fallback mechanisms.

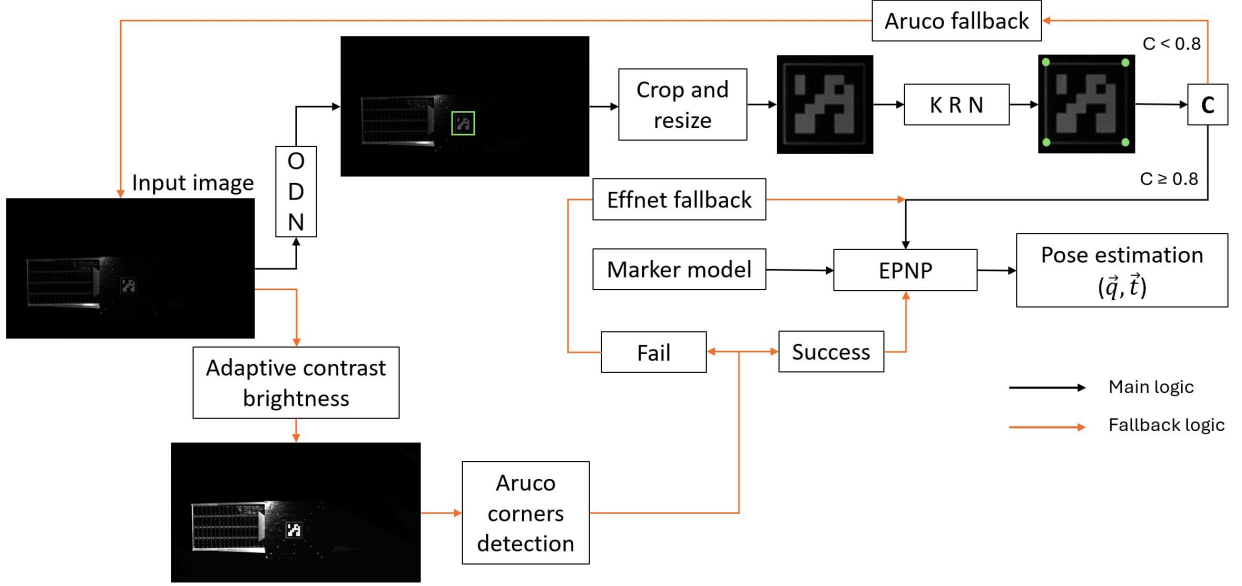


Figure 6.10: SAFE-Pose pipeline architecture with fallback mechanism.

The epistemic [30] uncertainty captured through this process reflects the model’s confidence in its predictions given the input image characteristics.

The hierarchical fallback system activates when uncertainty metrics exceed predefined thresholds: individual keypoint uncertainty exceeding 7 px or mean uncertainty across all keypoints surpassing 5 px. Note that the uncertainty value presented in the Figure 6.10 is normalized with respect to a maximum threshold of 20 px to obtain a percentile-based confidence measure. Upon activation, the fallback pathway applies Brightness and Contrast Adjustment (BCA) algorithm in terms of adaptive histogram equalization with 26% clipping. The ArUco detection algorithm then attempts to identify fiducial marker corners using the OpenCV implementation.

When KRN uncertainty remains below the threshold, neural network predictions are utilized directly. If uncertainty triggers fallback activation and ArUco detection succeeds, classical detection results supersede neural predictions. In cases where both uncertainty exceeds thresholds and ArUco detection fails, the system defaults to neural network predictions while flagging reduced confidence for downstream processing.

Final pose recovery employs the EPnP algorithm, utilizing the detected keypoints along with the known 3D marker model.

Training procedure

A synthetic subset of the STRIDE dataset comprising 40000 synthetic spacecraft images with randomized poses was divided into 35000 for training and 5000 for validation. The training process for EfficientNet-B0 backbone used ImageNet initialization [62], with cus-

tom PyTorch dataloaders handling keypoint annotations and augmentations. Training proceeded with a batch size of 128, an Adam optimizer with a learning rate of 10^{-3} , and mean squared error loss over 100 epochs, with performance monitored via RMSE on the validation set. To ensure robustness to varying illumination conditions, the training augmentation pipeline included geometric transforms (resizing, random rotation, affine scaling/translation, perspective distortion), challenging occlusion (coarse dropout), motion/defocus blur, lighting simulations (random brightness/contrast shifts, shadows, over-exposure), and noise effects (additive noise, CLAHE, random gamma, compression). Validation images underwent only resizing and normalization. YOLO11n was trained using the Ultralytics [3] default pipeline, which incorporates advanced augmentations such as mosaic and mixup, random horizontal and vertical flips, HSV color-space perturbations, random scaling, translation, and rotation. Key hyperparameters included a batch size of 128, an input image size of 640, and a learning rate of 0.01, optimized with either Adam or SGD. All models were trained on an NVIDIA GeForce RTX 4090 GPU.

6.2.1 Validation Results

Validation of the SAFE-Pose algorithm on the STRIDE dataset was carried out using two configurations of the ODN and KRN modules. The first configuration was trained exclusively on the Blender-generated synthetic subset, whereas the second employed images produced by a CycleGAN networks to mitigate the domain gap between synthetic and HIL imagery. The computational optimisation of the domain-adapted models for deployment on the Jetson Orin Nano is presented at the end of this subsection.

Domain adaptation

In Chapter 5, the domain adaptation analysis was performed on the SPEED+ dataset, whose HIL component is built from discrete viewpoints and therefore represents a set of static snapshots rather than temporally contiguous observations. While this setting is well suited to studying the synthetic-to-real domain gap under controlled illumination, it does not explicitly capture the sequential nature of a rendezvous and docking manoeuvre.

The STRIDE dataset introduced in this chapter is designed to address this limitation by providing complete docking trajectories acquired in hardware-in-the-loop conditions. Each case consists of a time-ordered image sequence with varying relative distance, non-rectilinear attitudes, and systematically modified illumination produced by the sun simulator, thus embedding the temporal correlations and pose-lighting biases that characterise realistic RPO scenarios. Examples of images generated by the CycleGAN-based domain adaptation method are shown in Figure 6.11.

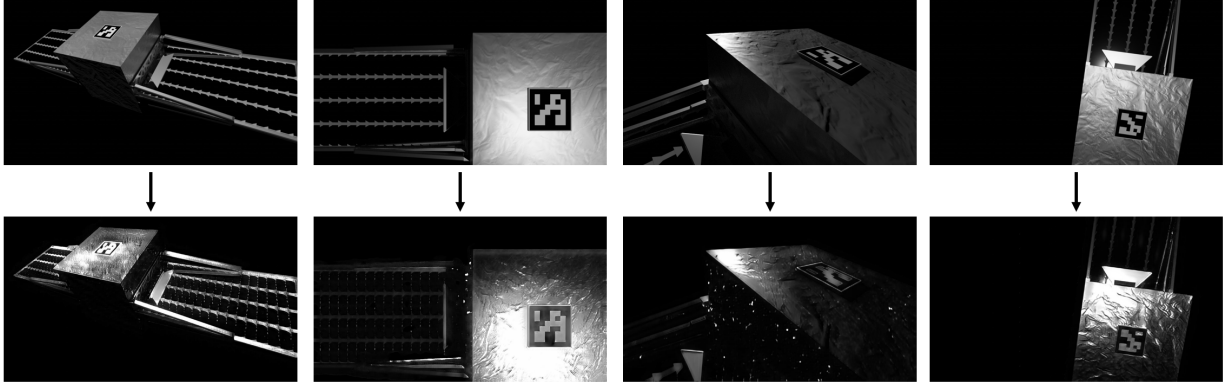


Figure 6.11: Representative examples of CycleGAN-based synthetic-to-real domain adaptation. Top row shows original synthetic images; bottom row displays corresponding GAN-translated images with enhanced photorealism.

Performance on Trajectory Cases 1–15

The evaluation results for trajectory cases 1–15, which maintain near-zero spacecraft rotations, are presented in Tables 6.2 and 6.3. These cases represent nominal docking approaches where the ArUco marker remains fully inside of the image frame throughout the rendezvous trajectory.

The YOLO detection module trained exclusively on the Blender-generated synthetic dataset achieved a mean IoU of 0.877 but exhibited a significant number of false positives (57 FP) and occasional missed detections (2 FN). In these failure cases, the SAFE-Pose pipeline deploys a default bounding box sufficiently large to encompass the marker region. Domain adaptation via CycleGAN retraining yielded dramatic improvements across all detection metrics: mean IoU increased to 0.960, false positives were reduced to only 3 instances, and no missed detections occurred. The average precision metrics similarly demonstrated substantial gains, with AP@0.5:0.95 improving from 0.889 to 0.960, confirming the effectiveness of adversarial domain adaptation in bridging the synthetic-to-real domain gap for spacecraft detection tasks.

Table 6.2: Object detection performance comparison between synthetic and CycleGAN retrained YOLO.

Method	Mean IoU	AP@0.5	AP@0.75	AP@0.5:0.95	FP	FN
Synth	0.877	0.928	0.925	0.889	57	2
CycleGAN	0.960	0.995	0.995	0.960	3	-

Regarding the KRN module, the synthetically trained EfficientNet-B0 backbone achieved a mean keypoint localization RMSE of 3.094 px, with 91.2% of predictions falling within a 5-pixel error threshold. Following CycleGAN-based domain adaptation, keypoint accuracy

improved markedly, with mean RMSE decreasing to 2.483 px. This enhanced localization precision directly translated to improved pose estimation accuracy: mean position error was reduced from 0.025 m to 0.010 m, while mean orientation error decreased from 8.797° to 3.301° — representing a 60% reduction in translational error and a 62% reduction in rotational error. Both the synthetically trained and domain-adapted KRN configurations maintained 100% inference success rate, confirming the robustness of the regression-based approach even under challenging illumination conditions. Visual examples of the synthetically trained models’ predictions and detections are illustrated in Figure 6.12.

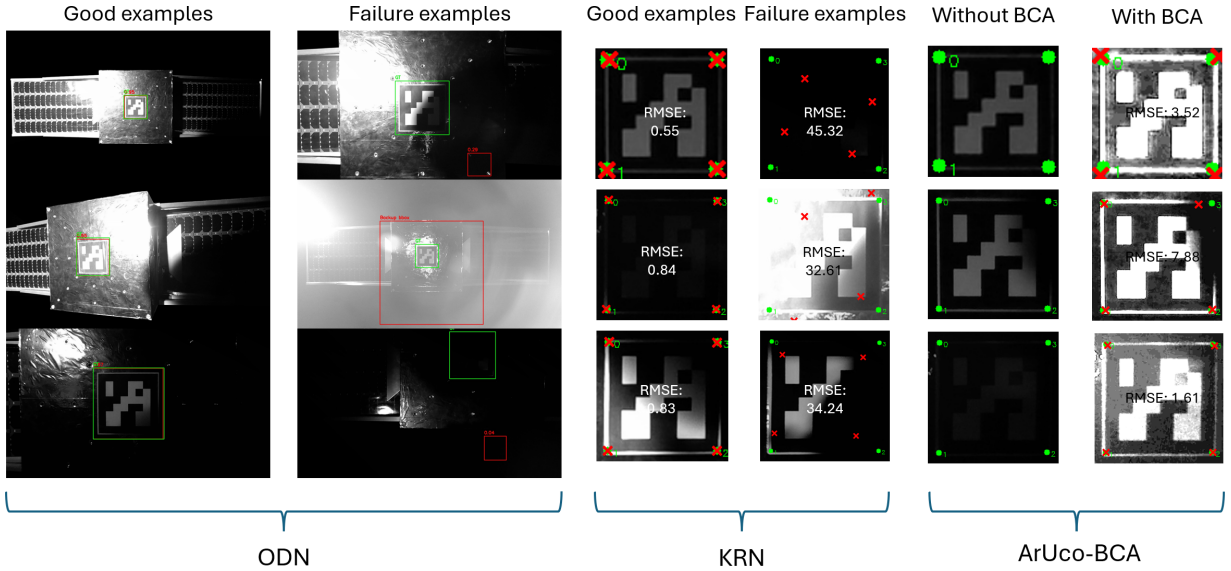


Figure 6.12: Prediction examples across trajectory cases 1–15. Left column shows ODN bounding box detections with examples of both successful detections and fallback default boxes. Middle column displays KRN keypoint predictions with RMSE values, demonstrating both accurate predictions and failure cases. Right column presents ArUco-BCA marker detection results.

The enhanced ArUco-BCA detection algorithm demonstrated a 12.37% improvement in marker detection rate over the standard OpenCV implementation, achieving a 68.9% success rate compared to 56.6% for the baseline approach. This improvement is particularly significant given the extreme illumination variations present in the images, including instances of direct solar saturation and deep shadowing. However, the marker-based approach exhibited higher sensitivity to lighting conditions, as evidenced by the substantially lower success rate compared to the DL-based methods. When ArUco-BCA successfully detected the marker, it achieved a mean position error of 0.029 m and orientation error of 6.071° , demonstrating competitive accuracy but reduced reliability relative to neural network-based approaches.

The SAFE-Pose integrated pipeline, which hierarchically combines neural and classical

Table 6.3: Keypoint detection and pose estimation performance (cases 1–15).

Metric	Effnet		ArUco		SAFE-Pose	
	Synth.	GAN	Raw	BCA	Synth.	GAN
Mean RMSE [px]	3.094	2.483	3.026	3.282	-	-
PRMSE@1px	0.310	0.117	0.089	0.040	-	-
PRMSE@2px	0.657	0.619	0.430	0.410	-	-
PRMSE@5px	0.912	0.935	0.888	0.849	-	-
PRMSE@10px	0.973	0.982	1.000	0.996	-	-
Success rate (%)	100	100	56.6	68.9	-	-
Mean Position Error (m)	0.025	0.010	-	0.029	0.029	0.018
Mean Orientation Error ($^{\circ}$)	8.797	3.301	-	6.071	8.335	5.167

detection methods with uncertainty-driven fallback mechanisms, achieved mean pose estimation errors of 0.029 m (translation) and 8.335° (rotation) for the synthetically trained configuration. Following domain adaptation, these errors were reduced to 0.018 m and 5.167° , respectively. Figure 6.13 presents a comprehensive comparison of the pipeline’s performance across all 15 trajectory cases before and after CycleGAN-based retraining. The quaternion and position error plots reveal that domain adaptation consistently reduces pose estimation errors across the majority of test cases, with particularly pronounced improvements observed in cases 2, 6, and 15 where challenging illumination or distance variations are present. The fallback usage statistics, displayed in the bottom panels of Figure 6.13, indicate that the domain-adapted configuration reduces reliance on the ArUco-BCA fallback mechanism due to increased neural network confidence, while maintaining robust performance through strategic activation of classical vision methods when epistemic uncertainty exceeds predefined thresholds. Notably, cases 3, 4, and 7 exhibit elevated fallback activation rates for both configurations, correlating with imagery containing extreme lighting conditions that degrade neural network predictive confidence.

Performance on Trajectory Cases 16–22

Trajectory cases 16–22 introduce static attitude deviations ranging from 10 to 50 degrees in roll, pitch, and yaw to replicate conditions where perfect alignment cannot be achieved due to operational constraints or limitations in the servicer’s approach trajectory. These cases present substantially increased difficulty compared to the aligned scenarios, as the non-rectilinear approach angles cause partial marker occlusions, perspective distortions, and reduced marker visibility throughout significant portions of the trajectory. The evaluation results are presented in Tables 6.4 and 6.5.

The YOLO detection module trained on synthetic data exhibited a substantial performance degradation, achieving a mean IoU of only 0.473 with 32 false positive detections.

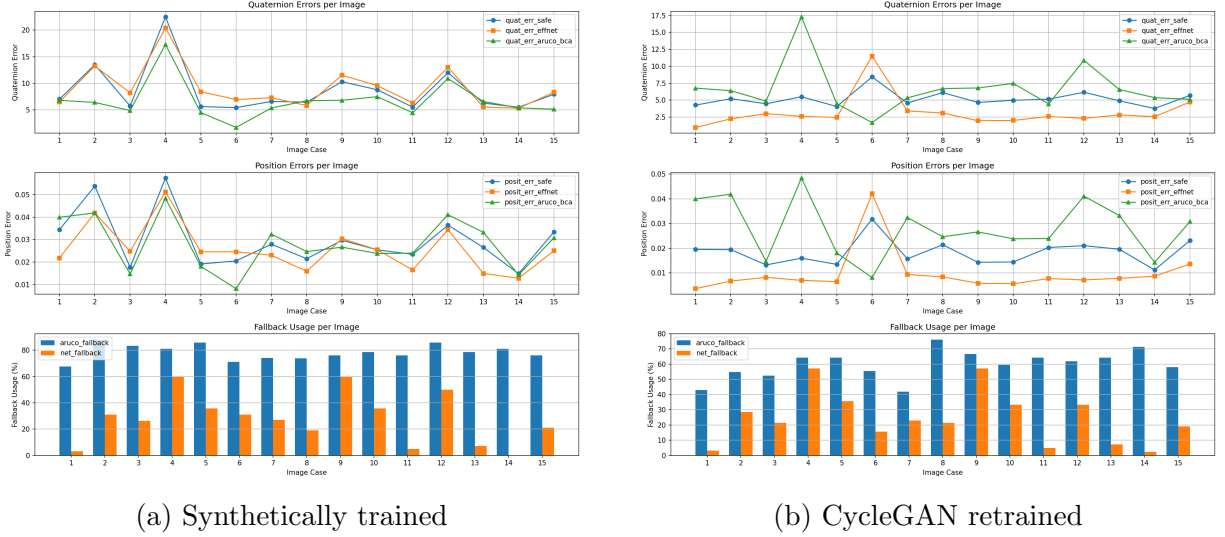


Figure 6.13: Comparison of SAFE-Pose performance for aligned trajectory cases (1–15) before and after CycleGAN-based domain adaptation. The plots show quaternion errors, position errors, and fallback usage statistics.

However, CycleGAN-based domain adaptation demonstrated remarkable improvements, increasing mean IoU to 0.795 and reducing false positives to 6 instances. The AP@0.5:0.95 metric improved from 0.437 to 0.672, representing a 54% relative improvement. Notably, the domain-adapted model produced 2 false negatives, likely corresponding to extreme illumination cases.

Table 6.4: Object detection performance comparison between synthetic and CycleGAN retrained YOLO.

Method	Mean IoU	AP@0.5	AP@0.75	AP@0.5:0.95	FP	FN
Synth	0.473	0.489	0.488	0.437	32	0
CycleGAN	0.795	0.830	0.757	0.672	6	2

The KRN module performance degraded significantly under non-aligned viewing geometries. The synthetically trained EfficientNet-B0 achieved a mean RMSE of 15.902 px — more than five times higher than the 3.094 px observed in aligned cases. Domain adaptation reduced the mean RMSE to 13.601 px, representing a 14.5% improvement, though accuracy remained substantially lower than in aligned scenarios. The percentage of predictions within 5 px remained at only 50% for both configurations. Visualization examples of the SAFE-Pose pipeline performance for these scenarios are presented in Figure 6.14.

The marker-based detection performance was particularly affected by non-aligned geometries. The standard ArUco detection achieved only a 34.39% success rate, while the enhanced ArUco-BCA algorithm improved this to 43.63% — representing a 27% relative increase but remaining substantially below the 68.9% achieved in aligned cases. The reduced

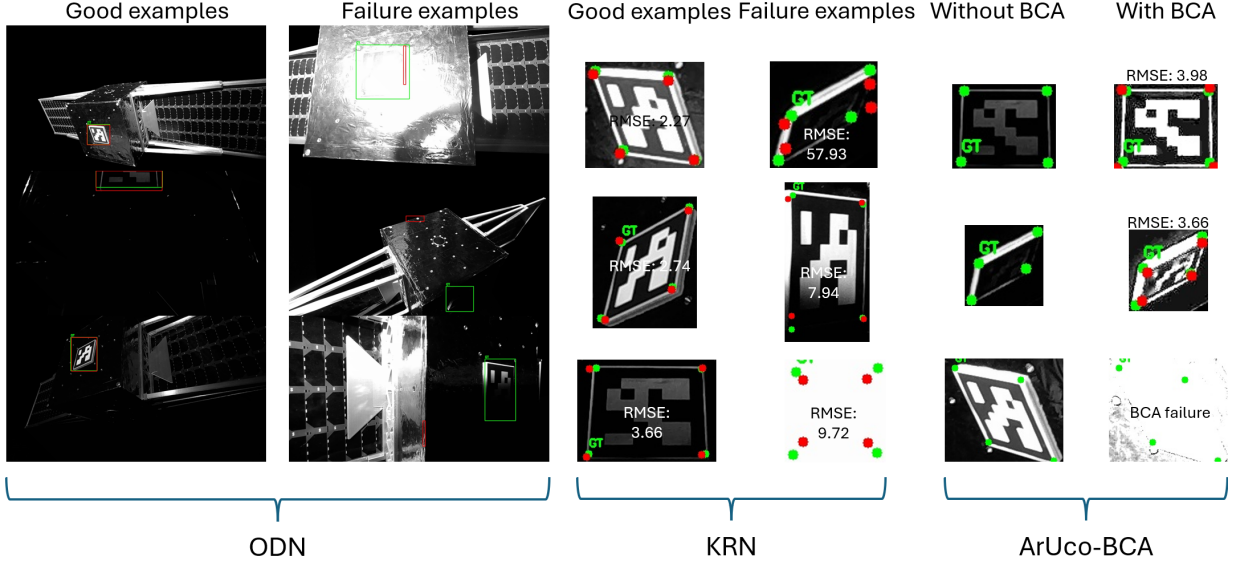


Figure 6.14: Prediction examples across trajectory cases 16–22. Left column shows ODN bounding box detections. Middle column displays KRN keypoint predictions. Right column presents ArUco-BCA marker detection results.

detection rate is primarily attributable to partial marker occlusions caused by spacecraft structural elements, saturated solar reflections, and deep shadows. However, when successful, the ArUco-BCA algorithm demonstrated exceptional pose accuracy, achieving a mean orientation error of only 0.024° — significantly outperforming the neural network-based approaches.

Table 6.5: Keypoint detection and pose estimation performance (cases 16–22).

Metric	Effnet		ArUco		SAFE-Pose	
	Synth.	GAN	Raw	BCA	Synth.	GAN
Mean RMSE [px]	15.902	13.601	2.180	3.119	-	-
PRMSE@1px	0.000	0.000	0.000	0.007	-	-
PRMSE@2px	0.150	0.045	0.491	0.255	-	-
PRMSE@5px	0.500	0.500	1.000	0.898	-	-
PRMSE@10px	0.621	0.720	1.000	1.000	-	-
Success rate (%)	100	100	34.39	43.63	-	-
Mean Position Error (m)	0.126	0.099	-	2.686	0.129	0.096
Mean Orientation Error ($^\circ$)	30.899	14.724	-	0.024	27.750	15.364

The integrated SAFE-Pose pipeline achieved mean pose estimation errors of 0.129 m (translation) and 27.750° (rotation) for the synthetically trained configuration. Domain adaptation reduced these errors to 0.096 m and 15.364° , respectively — representing improvements of 26% in position and 45% in orientation. Figure 6.15 presents a comprehensive comparison across all seven non-aligned trajectory cases. The quaternion error plots reveal substantial variability between cases, with case 18 exhibiting the highest orientation

errors (exceeding 50° for the synthetically trained model) due to the 49.72° roll deviation combined with challenging illumination. Domain adaptation consistently reduces errors across all cases, with particularly dramatic improvements in cases 18, 19, and 20 where non-rectilinear geometries are most severe. The fallback usage statistics demonstrate increased reliance on the ArUco-BCA mechanism relative to aligned cases, with the synthetically trained configuration activating fallback in 40–60% of frames across most cases. The domain-adapted configuration reduces fallback activation frequency while maintaining the safety-critical redundancy when neural network uncertainty is elevated. Cases 17 and 21 show near-complete fallback activation for both configurations, correlating with extreme solar illumination that saturates the marker region and degrades neural network confidence to unacceptable levels.

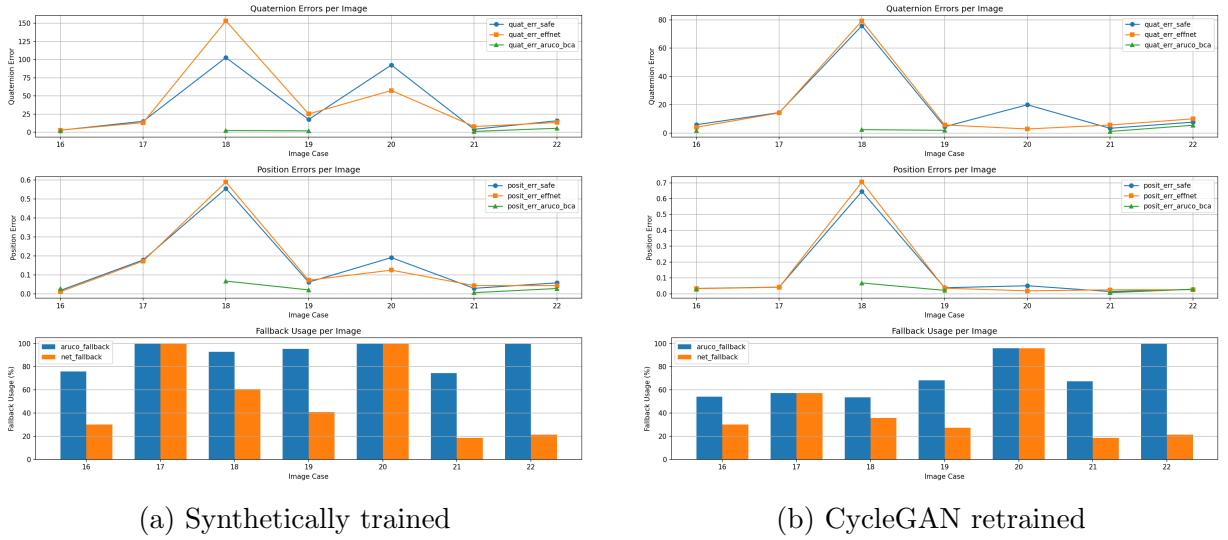


Figure 6.15: Comparison of SAFE-Pose performance for non-aligned trajectory cases (16–22) before and after CycleGAN-based domain adaptation. The plots show quaternion errors, position errors, and fallback usage statistics across cases.

SAFE-Pose Optimization

To evaluate the computational feasibility of deploying the SAFE-Pose pipeline on resource-constrained embedded hardware, comprehensive optimization experiments were conducted on the NVIDIA Jetson Orin Nano platform. Table 6.6 presents a systematic comparison of model performance across multiple optimization strategies, including framework conversion (PyTorch, ONNX, TensorRT) and precision quantization (FP32, FP16, INT8).

For the EfficientNet-B0 KRN module, TensorRT FP16 quantization achieved the optimal balance between inference speed and accuracy, delivering 103.4 FPS with a mean RMSE of 3.944 px — representing only a 0.2% accuracy degradation relative to the base-

Table 6.6: Model Performance Comparison on Nvidia Jetson Orin Nano

Model	EfficientNet-B0				YOLO11n			
	Size (MB)	Time (ms)	FPS	RMSE	Size (MB)	Time (ms)	FPS	IOU
PyTorch FP32	46.46	31.6	31.6	3.936	5.20	29.32	34.1	0.904
PyTorch FP16	46.46	33.70	29.7	3.940	5.20	29.65	33.7	0.905
ONNX FP32	15.35	20.58	48.6	3.938	10.11	32.80	30.5	0.905
ONNX FP16	7.72	17.33	57.7	3.939	10.11	32.53	30.7	0.905
TensorRT FP32	16.84	15.24	65.6	3.965	11.87	33.19	30.1	0.905
TensorRT FP16	9.42	9.67	103.4	3.944	8.41	23.15	43.2	0.905
TensorRT INT8	5.77	7.39	135.3	11.816	5.24	15.51	64.5	0.673

line PyTorch FP32 implementation while providing a $3.3\times$ speedup. Further quantization to INT8 yielded 135.3 FPS but resulted in substantial accuracy loss (RMSE: 11.816 px), rendering this configuration unsuitable for precision-critical pose estimation tasks. The YOLO11n ODN module demonstrated similar optimization characteristics, with TensorRT FP16 achieving 43.2 FPS while maintaining an IoU of 0.905, identical to the FP32 baseline. Model size was reduced from 46.46 MB to 9.42 MB for EfficientNet-B0, but increased from 5.20 MB to 8.41 MB for YOLO11n through TensorRT FP16 conversion. However, Monte Carlo Dropout for epistemic uncertainty quantification is incompatible with quantized model architectures, as dropout layer behavior becomes undefined when network weights are represented in reduced precision formats. Consequently, the complete SAFE-Pose pipeline was evaluated using standard PyTorch FP32 implementations for both the ODN and KRN modules to enable uncertainty-aware fallback mechanisms. Tables 6.7 and 6.8 present the computational cost analysis of Monte Carlo Dropout and the component-wise latency breakdown of the complete pipeline, respectively.

Table 6.7: Monte Carlo Dropout computational cost for EfficientNet-B0

Num. passes	Time (ms)	FPS
1	10.91	31.6
3	83.65	12.0
5	135.24	7.4
15	409.46	2.4
30	786.5	1.3

Table 6.8: Component-wise inference latency breakdown

Component	Time (ms)	FPS
Image loading	10.91	91.7
YOLO FP32	30.43	32.9
EffNet FP32	135.24	7.4
ArUco-BCA	298.78	3.3
PnP	1.21	823.1
Total	167.55	6.0

The relationship between inference time and the number of stochastic forward passes is approximately linear, with 5 passes requiring 135.24 ms and 30 passes demanding 786.5

ms. Based on this analysis, 5 forward passes were selected as the operational configuration, providing sufficient statistical confidence in uncertainty estimates while maintaining acceptable inference latency. The component-wise breakdown reveals that the KRN module with 5-pass Monte Carlo Dropout constitutes the primary computational bottleneck, consuming 135.24 ms (80.7% of total pipeline time). The ODN module requires 30.43 ms, while image loading contributes 10.91 ms of overhead. Notably, the ArUco-BCA fallback mechanism exhibits substantially higher latency (298.78 ms) than neural network inference, justifying its deployment only when epistemic uncertainty exceeds predefined thresholds. The complete pipeline achieves an inference rate of 6.0 FPS (167.55 ms per frame), which satisfies frame rate requirements for vision-based navigation systems in on-orbit servicing and active debris removal missions.

6.3 Chapter Conclusion

This chapter has presented the comprehensive experimental development and validation of the STRIDE dataset and the SAFE-Pose algorithm. The dataset comprises 40,000 grayscale synthetic images of a spacecraft mockup model provided by the Zero-G Lab, generated across randomized spacecraft poses and varying illumination conditions within a distance range of 0.33–1.00 m. The real subset, acquired at the Zero-G Lab facility, consists of 1,044 grayscale images distributed across 22 distinct trajectory cases. These trajectories were captured under systematically varied illumination conditions with both aligned (cases 1–15) and non-aligned (cases 16–22) spacecraft orientations, spanning distances from 0.33 m to 0.79 m and encompassing extreme lighting scenarios representative of on-orbit operations.

The SAFE-Pose pipeline, incorporating KRN epistemic uncertainty estimation via Monte Carlo Dropout and an uncertainty-triggered fallback mechanism grounded in an enhanced ArUco-BCA pose estimation approach, demonstrated substantial robustness across diverse operational scenarios. The integration of CycleGAN-based domain adaptation proved critical to bridging the synthetic-to-real domain gap, yielding dramatic performance improvements across all pipeline components. For the ODN module, domain adaptation increased mean IoU from 0.877 to 0.960 for aligned cases and from 0.473 to 0.795 for non-aligned cases, while simultaneously reducing false positive detections by 95% and 81%, respectively.

Quantitative results for aligned trajectory cases (1–15) indicate that the domain-adapted SAFE-Pose pipeline achieved a mean position error of 0.018 m and orientation error of 5.167° , representing 60% and 62% reductions relative to the synthetically trained baseline. The ArUco-BCA fallback mechanism demonstrated a 12.37% improvement in detection

rate over standard OpenCV implementation, achieving a 68.9% success rate with competitive pose accuracy (position error: 0.029 m, orientation error: 6.071°) when detection succeeded. For non-aligned trajectory cases (16–22), characterized by 10–50 degree attitude deviations, the domain-adapted configuration achieved mean errors of 0.096 m in position and 15.364° in orientation — representing 26% and 45% improvements over the synthetic baseline despite substantially degraded viewing geometries.

The analysis revealed complementary operational domains for neural network-based and classical vision approaches: DL-based regression maintained continuous operation with 100% inference success rate across all scenarios but exhibited reduced accuracy under non-rectilinear viewing geometries, while the ArUco-BCA method achieved exceptional precision (orientation error: 0.024°) when successful but suffered from reduced reliability (43.63% success rate in non-aligned cases) due to sensitivity to marker occlusions and extreme illumination. The hierarchical integration of these methods through uncertainty-aware fallback mechanisms enabled the SAFE-Pose pipeline to maintain operational continuity while leveraging the respective strengths of each approach.

Computational optimization experiments on the NVIDIA Jetson Orin Nano platform confirmed the feasibility of embedded deployment. Although TensorRT FP16 quantization achieved 103.4 FPS for the KRN module with negligible accuracy degradation (0.2%), the incompatibility of Monte Carlo Dropout with quantized architectures necessitated the use of PyTorch FP32 implementations for uncertainty quantification. The complete pipeline, configured with 5 Monte Carlo Dropout passes, achieved an end-to-end inference rate of 6.0 FPS (167.55 ms per frame), satisfying the typical guidance, navigation, and control update rate requirements for vision-based navigation systems in on-orbit servicing and active debris removal missions.

The developed SAFE-Pose algorithm is established as the baseline validation solution for the STRIDE dataset, providing a quantitative benchmark for future pose estimation research employing sequential trajectory imagery under realistic illumination conditions. The STRIDE dataset will be made publicly available with the subsequent journal publication, providing the research community with a new pose estimation benchmark that explicitly captures the temporal correlations and domain-specific challenges characteristic of spacecraft rendezvous and docking operations.

Chapter 7

Conclusion

This study aimed to develop and validate deep learning (DL)-based spacecraft pose estimation algorithms suitable for resource-constrained embedded platforms, supporting on-orbit servicing (OOS) and active debris removal (ADR) missions. The research addressed three fundamental challenges in vision-based navigation: (i) achieving real-time inference on limited hardware, (ii) mitigating the synthetic-to-real domain gap, and (iii) ensuring robust operation through uncertainty-aware decision frameworks. All developments assumed monocular imagery and a known 3D model of the cooperative or uncooperative target spacecraft.

7.1 Review of Contributions

7.1.1 EROSS Mission Pose Estimation Pipeline

As detailed in Chapter 3, a DL-based pose estimation pipeline was developed, specifically designed to meet the operational requirements of the European Robotic Orbital Support Services (EROSS) mission. The activities were carried out during a PhD co-location period at Thales Alenia Space Italia (TASI). The proposed Close Range Pose Estimation Pipeline (CRPEP) with active switch adopts a dual three-stage architecture: an object detection network (ODN) for spacecraft localization and a keypoint regression network (KRN) for predicting keypoints subsequently used by a PnP solver for final 6D pose recovery. Four networks are utilized in total: the initial set (ODN1 and KRN1) estimates the pose of the full spacecraft, while the second set (ODN2 and KRN2) activates to process marker

features at very close range, particularly when the spacecraft partially exits the camera’s FOV. All networks leverage the EfficientNet-B0 backbone, offering an optimal trade-off between computational efficiency and accuracy.

Two synthetic datasets support algorithm training and evaluation. The first, generated in Blender, comprises 10,000 images spanning randomized spacecraft poses across diverse viewpoints. The second, created in Unreal Engine 5, contains 12,642 trajectory-based images simulating representative rendezvous conditions with varying illumination scenarios, including direct sunlight and shadows coming from the servicer itself.

Experimental results on the NVIDIA Jetson Nano embedded platform confirm that the CRPEP with active switch achieves the requisite inference speed of 5 FPS for space operational needs, with translation accuracies at the millimeter level and achievable attitude accuracy at the degree level. Future work should integrate a dual-camera configuration with large and small FOVs, thereby extending the operational range of the system.

7.1.2 DL-models inference acceleration on Jetson Orin nano

As detailed in Chapter 4, inference optimization strategies were investigated for deploying DL-based spacecraft detection systems on Jetson Orin Nano hardware. The YOLOv8n and EfficientNet-B0 architectures were trained using the synthetic subset of the SPEED+ dataset and subsequently optimized through model format conversion, numerical precision reduction, and hardware-specific acceleration techniques.

Experimental results demonstrated significant acceleration potential. TensorRT optimization with FP16 precision enabled EfficientNet-B0 to reach a fivefold speedup, achieving 136 FPS, while INT8-quantized YOLOv8n obtained a $3.5\times$ speedup with 129 FPS compared to their respective GPU baselines. Both results exceed the performance requirements for real-time on-orbit operations. However, quantization efficacy was observed to be architecture-dependent. Specifically, INT8 quantization caused only a 6.3% accuracy degradation for YOLOv8n, while EfficientNet-B0 exhibited systematic failures due to unsupported quantized operations (QLinearMul, QLinearGlobalAveragePool, QLinearSigmoid, QLinearConv) in ONNX Runtime and TensorRT. Quantized EfficientNet-B0 models often reverted to CPU execution, with QDQ representations producing invalid outputs ($\text{IoU} \approx 0$), while QOperator preserved numerical correctness but incurred significant performance degradation from frequent CPU–GPU memory transfers. Half-precision inference provided a practical compromise, yielding negligible accuracy loss relative to FP32 (IoU difference $< 0.5\%$) while substantially improving throughput. Batch processing further increased inference speed, with 16-image batches reaching 229 FPS, though at the cost of higher latency — an important factor for real-time applications. The optimized models were consistent with findings in existing literature, achieving 3–13 $\times$ acceleration compared

to reported RemoveDEBRIS benchmarks and up to $169\times$ improvement over earlier Jetson Nano implementations. These results demonstrate the feasibility of real-time, on-board inference for spacecraft detection, surpassing the operational requirements of modern satellite avionics.

Overall, the results underscore the strong potential of FP16 hardware acceleration for spaceborne deep learning, while emphasizing the limitations of INT8 quantization workflows across certain architectures. Future work should explore quantization-aware training, pruning techniques, and TensorRT-native quantization pipelines to fully realize the performance and efficiency gains of DL-based models.

7.1.3 CycleGAN-Based Domain Adaptation

Chapter 5 addressed the domain gap problem in spacecraft pose estimation by implementing CycleGAN image-to-image translation networks on the SPEED+ dataset. A hybrid three-stage architecture was employed, comprising a YOLOv11-based ODN, an EfficientNet-B0-based KRN, and an EPnP solver for final 6D pose recovery. For a baseline solution both networks were trained on synthetic images of the SPEED+ dataset. However, cross-domain transfer on the Hardware-In-the-Loop (HIL) images produced severe degradation. Therefore, to address the domain gap, separate CycleGAN instances were independently trained on the lightbox and sunlamp domains using ResNet-style generators with nine residual blocks and PatchGAN discriminators.

Translation quality assessment through Fréchet Inception Distance (FID) score demonstrated successful domain alignment, with the lightbox translation achieving FID of 19.62, indicating excellent perceptual similarity approaching human-imperceptible quality thresholds. The sunlamp translation yielded FID of 59.93, reflecting increased difficulty of translating extreme illumination conditions characterized by harsh shadows and specular highlights.

Domain-adapted models retrained on CycleGAN-translated synthetic imagery demonstrated substantial performance improvement with respect to baseline solution when evaluated on real HIL images. The lightbox domain adaptation reduced orientation error from 38.45° to 19.85° and position error from 0.642 m to 0.305 m, with RMSE decreasing from 28.52 px to 16.28 px. These results demonstrated competitive mid-tier performance in SPEC2021 competition rankings.

The sunlamp domain exhibited more modest improvements, with orientation error reducing from 48.76° to 43.85° and position error improving from 0.834 m to 0.674 m, reflecting the fundamental difficulty of extreme illumination scenarios that approached CycleGAN's translation capability limits. These outcomes suggest that translation artifacts present in CycleGAN-generated images may differentially affect various components of the

pipeline, warranting further investigation into alternative adaptation architectures, such as adversarial feature alignment employed by the top entries of the SPEC2021 competition.

Nevertheless, the quantitative improvements demonstrated that CycleGAN-based domain adaptation mitigated significantly the synthetic-to-real gap. Future work should prioritize source-free domain adaptation methods that do not require access to target domain imagery. This need becomes particularly critical in the absence of dedicated on-ground validation testbeds, where real images may not be available prior to rendezvous and proximity operations.

7.1.4 STRIDE Dataset and SAFE-Pose Pipeline

Chapter 6 introduced the Spacecraft Trajectory Imagery for Docking Estimation (STRIDE) dataset and presented the development of the Safe Autonomous Fallback for Estimation of Pose (SAFE-Pose) algorithm, which integrates epistemic uncertainty quantification with classical computer vision fallback mechanisms to address the need for robust and reliable DL-based pose estimation under degraded network confidence.

The STRIDE dataset comprises images from synthetic and HIL domains. The synthetic subset, generated in Blender, contains 40,000 grayscale images of a spacecraft mockup with randomized poses spanning distances from 0.33 m to 1.00 m under systematically varied illumination conditions. The real subset consists of 1,044 grayscale images acquired across 22 trajectory cases at the Zero-G Lab of the University of Luxembourg, with target-to-camera distances ranging from 0.33 m to 0.79 m. The trajectory cases are divided into aligned scenarios (cases 1–15) maintaining near-zero spacecraft rotations to emulate nominal docking approaches, and non-aligned scenarios (cases 16–22) introducing static attitude deviations of 10–50 degrees to replicate operationally constrained approach trajectories. Real image acquisition employed an Intel RealSense D455 camera in monocular mode, with illumination conditions systematically varied using a Godox SL-60W sun simulator to emulate extreme solar lighting, deep shadows, and sensor saturation.

The SAFE-Pose pipeline employs a hybrid architecture integrating YOLO11n for object detection, EfficientNet-B0 for keypoint regression, and EPnP for pose recovery. Central to the pipeline’s reliability is Monte Carlo Dropout uncertainty quantification with 5 forward passes, generating epistemic uncertainty estimates that trigger an enhanced ArUco-BCA fallback mechanism when individual keypoint uncertainty exceeds 7 px or mean uncertainty surpasses 5 px. To address the synthetic-to-real domain gap, CycleGAN-based unsupervised image-to-image translation was employed, yielding substantial performance improvements across all pipeline components.

Quantitative validation on aligned trajectory cases (1–15) demonstrated that domain adaptation reduced mean pose estimation errors from 0.029 m and 8.335° (synthetically

trained) to 0.018 m and 5.167° (CycleGAN retrained), representing 60% and 62% reductions in translational and rotational errors, respectively. The enhanced ArUco-BCA fallback achieved a 68.9% detection success rate with mean errors of 0.029 m and 6.071° when successful, demonstrating a 12.37% improvement over standard OpenCV implementation. For non-aligned trajectory cases (16–22), the domain-adapted configuration achieved mean errors of 0.096 m and 15.364° , representing 26% and 45% improvements over the synthetic baseline despite substantially degraded viewing geometries. ArUco-BCA success rate decreased to 43.63% due to partial marker occlusions and extreme illumination, though it achieved exceptional precision (0.024° orientation error) when detection succeeded.

Deployment on the NVIDIA Jetson Orin Nano platform confirmed computational feasibility for embedded systems, with the complete pipeline achieving 6.0 FPS using PyTorch FP32 implementations to maintain Monte Carlo Dropout compatibility. Although TensorRT FP16 quantization of individual modules (KRN: 103.4 FPS, ODN: 43.2 FPS) demonstrated significant acceleration potential, the incompatibility of dropout layers with reduced precision formats necessitated full-precision inference for uncertainty-aware operation. The achieved 6.0 FPS throughput satisfies typical GNC update rate requirements (1–10 Hz) for vision-based navigation in on-orbit servicing and active debris removal missions.

The developed SAFE-Pose algorithm establishes a baseline validation solution for the STRIDE dataset, demonstrating that hierarchical integration of neural network-based and classical vision methods through uncertainty-driven fallback mechanisms enables robust pose estimation across diverse operational scenarios. Future work should focus on joint modeling of epistemic and aleatoric uncertainties within the KRN, integration of object detection uncertainty into fallback decision logic, and exploration of quantization-compatible uncertainty estimation techniques to enable optimized embedded deployment. The STRIDE dataset will be made publicly available to the research community with the subsequent journal publication, providing a new benchmark for pose estimation research employing sequential trajectory imagery under realistic illumination conditions.

References

- [1] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.
- [2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [3] “Ultralytics yolo documentation,” <https://docs.ultralytics.com/>, Ultralytics, (Accessed: 2025-09-12).
- [4] M. Lofqvist and J. Cano, “Accelerating deep learning applications in space,” *arXiv preprint arXiv:2007.11089*, 2020.
- [5] T. H. Park, M. Mörtens, G. Lecuyer, D. Izzo, and S. D’Amico, “Speed+: Next-generation dataset for spacecraft pose estimation across domain gap,” in *2022 IEEE aerospace conference (AERO)*. IEEE, 2022, pp. 1–15.
- [6] S. Christin, Hervet, and N. Lecomte, “Machine learning and deep learning—a review for ecologists,” *Methods in Ecology and Evolution*, vol. 13, no. 8, pp. 1647–1662, 2022.
- [7] J. F. Rodríguez, J. Macías, M. J. Castro, M. de la Asunción, and C. Sánchez-Linares, “Use of neural networks for tsunami maximum height and arrival time predictions,” *GeoHazards*, vol. 3, no. 2, pp. 323–344, 2022.
- [8] A. Amini, A. Soleimany, S. Karaman, and D. Rus, “Spatial uncertainty sampling for end-to-end control,” *arXiv preprint arXiv:1805.04829*, 2018.

- [9] M. Abdulaal, M. Ibrahim, M. Mahmoud, J. Khalid, A. Aljohani, A. Milyani, and A. Abusorrah, “Real-time detection of false readings in smart grid ami using deep and ensemble learning,” *IEEE Access*, vol. 10, pp. 1–1, 01 2022.
- [10] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” *arXiv preprint arXiv:1603.07285*, 2016.
- [11] F. Ni, J. Wang, J. Tang, W. Yu, and R. Xu, “Side channel analysis based on feature fusion network,” *Plos one*, vol. 17, no. 10, p. e0274616, 2022.
- [12] M. Yani, B. Irawan, and C. Setiningsih, “Application of transfer learning using convolutional neural network method for early detection of terry’s nail,” in *Journal of Physics: Conference Series*, vol. 1201, no. 1. IOP Publishing, 2019, p. 012052.
- [13] F. R. Fathabadi, J. L. Grantner, I. Abdel-Qader, and S. A. Shebrain, “Box-trainer assessment system with real-time multi-class detection and tracking of laparoscopic instruments, using cnn,” *Acta Polytechnica Hungarica*, vol. 19, no. 2, pp. 7–27, 2022.
- [14] W. Zhou, J. Ji, Y. Jiang, J. Wang, Q. Qi, and Y. Yi, “Eards: Efficientnet and attention-based residual depth-wise separable convolution for joint od and oc segmentation,” *Frontiers in neuroscience*, vol. 17, p. 1139181, 2023.
- [15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
- [16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.
- [17] X. Ma, S. Lin, S. Ye, Z. He, L. Zhang, G. Yuan, S. H. Tan, Z. Li, D. Fan, X. Qian *et al.*, “Non-structured dnn weight pruning—is it beneficial in any platform?” *IEEE transactions on neural networks and learning systems*, vol. 33, no. 9, pp. 4930–4944, 2021.
- [18] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märkl, and S. D’Amico, “Satellite pose estimation challenge: Dataset, competition design, and results,” vol. 56, no. 5, 2020, pp. 4083–4098.
- [19] R. Padilla, W. L. Passos, T. L. Dias, S. L. Netto, and E. A. Da Silva, “A comparative analysis of object detection metrics with a companion open-source toolkit,” *Electronics*, vol. 10, no. 3, p. 279, 2021.

- [20] S. Behar-Lafenetre and P. Dandré, “Eross mission progress status,” October 17 2023, presentation at Clean Space Industry Days. [Online]. Available: https://indico.esa.int/event/450/contributions/8912/attachments/5734/9490/20231017_S.Behar-P.Dandr%C3%A9_EROSS_Mission_progress_status.pdf
- [21] M. Olivares-Mendez, M. R. Makhdoomi, B. C. Yalçın, Z. Bokal, V. Muralidharan, M. O. Del Castillo, V. Gaudilliere, L. Pauly, O. Borgue, M. Alandihallaj *et al.*, “Zero-g lab: A multi-purpose facility for emulating space operations,” *Journal of Space Safety Engineering*, vol. 10, no. 4, pp. 509–521, 2023.
- [22] European Space Agency, “Space debris by the numbers,” https://www.esa.int/Space_Safety/Space_Debris/Space_debris_by_the_numbers, 2024, accessed: 2024.
- [23] D. J. Kessler and B. G. Cour-Palais, “Collision frequency of artificial satellites: The creation of a debris belt,” *Journal of Geophysical Research: Space Physics*, vol. 83, no. A6, pp. 2637–2646, 1978.
- [24] G. S. Aglietti *et al.*, “The active space debris removal mission removedebris. part 2: In orbit operations,” *Acta Astronautica*, vol. 168, pp. 310–322, 2020.
- [25] R. Biesbroek, S. Aziz, A. Wolahan, S. Cipolla, M. Richard-Noca, and L. Piguet, “The clearspace-1 mission: Esa and clearspace team up to remove debris,” in *Proc. 8th eur. conf. Sp. debris*, 2021, pp. 1–3.
- [26] T. Yamamoto, J. Matsumoto, H. Okamoto, R. Yoshida, C. Hoshino, and K. Yamanaka, “Pave the way for active debris removal realization: Jaxa commercial removal of debris demonstration (crd2),” in *8th European Conference on Space Debris*, 2021, p. 200.
- [27] A. Behar *et al.*, “EROSS: In-orbit demonstration of European robotic orbital support services,” in *2024 IEEE Aerospace Conference*, 2024, pp. 1–14.
- [28] L. Pauly, W. Rharbaoui, C. Shneider, A. Rathinam, V. Gaudilliere, and D. Aouada, “A survey on deep learning-based monocular spacecraft pose estimation: Current state, limitations and prospects,” *Acta Astronautica*, vol. 212, pp. 339–360, 2023.
- [29] A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia, “A brief review of domain adaptation,” *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pp. 877–894, 2021.
- [30] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” *Advances in neural information processing systems*, vol. 30, 2017.

- [31] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [32] E. Salahat and M. Qasaimeh, “Recent advances in features extraction and description algorithms: A comprehensive survey,” in *2017 IEEE international conference on industrial technology (ICIT)*. IEEE, 2017, pp. 1059–1063.
- [33] S. D’Amico, M. Benn, and J. L. Jørgensen, “Pose estimation of an uncooperative spacecraft from actual space imagery,” *International Journal of Space Science and Engineering* 5, vol. 2, no. 2, pp. 171–189, 2014.
- [34] C.-H. Chien and K. Baker, “Pose estimation for servicing of orbital replacement units in a cluttered environment,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*, vol. 5. IEEE, 2004, pp. 5141–5146.
- [35] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [36] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*. Springer, 2006, pp. 404–417.
- [37] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European conference on computer vision*. Springer, 2010, pp. 778–792.
- [38] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.
- [39] D. Bojanić, K. Bartol, T. Pribanić, T. Petković, Y. D. Donoso, and J. S. Mas, “On the comparison of classic and deep keypoint detector and descriptor methods,” in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*. IEEE, 2019, pp. 64–69.
- [40] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [41] M. R. Leinz, C.-T. Chen, P. Scott, W. Gaumer, P. Sabasteanski, and M. Beaven, “Modeling, simulation, testing, and verification of the orbital express autonomous rendezvous and capture sensor system (arcss),” in *Sensors and Systems for Space Applications II*, vol. 6958. SPIE, 2008, pp. 75–87.

- [42] A. Cropp, *Pose estimation and relative orbit determination of a nearby target microsatellite using passive imagery*. University of Surrey (United Kingdom), 2001.
- [43] S. Zhang and X. Cao, “Closed-form solution of monocular vision-based relative pose determination for rvd spacecrafts,” *Aircraft Engineering and Aerospace Technology*, vol. 77, no. 3, pp. 192–198, 2005.
- [44] A. Petit, E. Marchand, and K. Kanani, “Vision-based detection and tracking for space navigation in a rendezvous context,” in *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS*, 2012.
- [45] M. Kisantal, R. Volden, M. Griffa, B. Tweddle, S. Parkes, and M. Romano, “A model-based 3D template matching technique for pose acquisition of an uncooperative space object,” *Sensors*, vol. 15, no. 3, pp. 6360–6382, 2015.
- [46] A. A. Grompone, “Vision-based 3d motion estimation for on-orbit proximity satellite tracking and navigation,” 2015.
- [47] M. D’Errico, *Distributed space missions for earth system monitoring*. Springer Science & Business Media, 2012, vol. 31.
- [48] V. Capuano, S. R. Alimo, A. Q. Ho, and S.-J. Chung, “Robust features extraction for on-board monocular-based spacecraft pose acquisition,” in *AIAA Scitech 2019 Forum*, 2019, p. 2005.
- [49] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.
- [50] J. Shi *et al.*, “Good features to track,” in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.
- [51] S. Sharma, J. Ventura, and S. D’Amico, “Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous,” *Journal of Spacecraft and Rockets*, vol. 55, no. 6, pp. 1414–1429, 2018.
- [52] S. Sharma, T. H. Park, and S. D’Amico, “Spacecraft pose estimation dataset (speed),” Stanford Digital Repository, 2019. [Online]. Available: <https://doi.org/10.25740/dz692fn7184>
- [53] C. Beierle and S. D’Amico, “Variable-magnification optical stimulator for training and validation of spaceborne vision-based navigation,” *Journal of Spacecraft and Rockets*, vol. 56, no. 4, pp. 1060–1072, 2019.

- [54] M. A. Musallam, V. Gaudilliere, E. Ghorbel, K. Al Ismaeil, M. D. Perez, M. Poucet, and D. Aouada, “Spacecraft recognition leveraging knowledge of space environment: Simulator, dataset, competition design and analysis,” in *2021 IEEE International Conference on Image Processing Challenges (ICIPC)*. IEEE, 2021, pp. 11–15.
- [55] P. F. Proença and Y. Gao, “Deep learning for spacecraft pose estimation from photorealistic rendering,” pp. 6007–6013, 2020.
- [56] S. Kaki, J. Deutsch, K. Black, A. Cura-Portillo, B. A. Jones, and M. R. Akella, “Real-time image-based relative pose estimation and filtering for spacecraft applications,” *Journal of Aerospace Information Systems*, vol. 20, no. 6, pp. 290–307, 2023.
- [57] H. A. Dung, B. Chen, and T.-J. Chin, “A spacecraft dataset for detection, segmentation and parts recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2012–2019.
- [58] L. P. Cassinis, R. Fonod, E. Gill, I. Ahrns, and J. Gil-Fernández, “Evaluation of tightly-and loosely-coupled approaches in cnn-based pose estimation systems for uncooperative spacecraft,” *Acta Astronautica*, vol. 182, pp. 189–202, 2021.
- [59] T. H. Park, M. Mörtens, M. Jawaid, Z. Wang, B. Chen, T.-J. Chin, D. Izzo, and S. D’Amico, “Satellite pose estimation competition 2021: Results and analyses,” *Acta Astronautica*, vol. 204, pp. 640–665, 2023.
- [60] S. Sharma, C. Beierle, and S. D’Amico, “Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks,” in *2018 IEEE Aerospace Conference*. IEEE, 2018, pp. 1–12.
- [61] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” *arXiv preprint arXiv:1803.01164*, 2018.
- [62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, vol. 25, 2012.
- [63] B. Chen, J. Cao, A. Parra, and T.-J. Chin, “Satellite pose estimation with deep landmark regression and nonlinear pose refinement,” in *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019, pp. 0–0.

- [64] Y. Hu, S. Speierer, W. Jakob, P. Fua, and M. Salzmann, “Wide-depth-range 6d object pose estimation in space,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 870–15 879.
- [65] A. Garcia, M. A. Musallam, V. Gaudilliere, E. Ghorbel, K. Al Ismaeil, M. Perez, and D. Aouada, “Lspnet: A 2d localization-oriented spacecraft pose estimation neural network,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2048–2056.
- [66] Z. Wang, M. Chen, Y. Guo, Z. Li, and Q. Yu, “Bridging the domain gap in satellite pose estimation: A self-training approach based on geometrical constraints,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 60, no. 3, pp. 2500–2514, 2024.
- [67] Y. Yuan, R. Fu, L. Huang, W. Lin, C. Zhang, X. Chen, and J. Wang, “Hrformer: High-resolution vision transformer for dense predict,” *Advances in neural information processing systems*, vol. 34, pp. 7281–7293, 2021.
- [68] X. Wang, L. Bo, and L. Fuxin, “Adaptive wing loss for robust face alignment via heatmap regression,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6971–6981.
- [69] A. Lotti, D. Modenini, and P. Tortora, “Investigating vision transformers for bridging domain gap in satellite pose estimation,” in *International Conference on Applied Intelligence and Informatics*. Springer, 2022, pp. 299–314.
- [70] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10 002.
- [71] C. Zhang, M. Zhang, S. Zhang, D. Jin, Q. Zhou, Z. Cai, H. Zhao, X. Liu, and Z. Liu, “Delving deep into the generalization of vision transformers under distribution shifts,” in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2022, pp. 7277–7286.
- [72] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.
- [73] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, “Eie: Efficient inference engine on compressed deep neural network,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

- [74] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [75] B. Jiang, J. Chen, and Y. Liu, “Single-shot pruning and quantization for hardware-friendly neural network acceleration,” *Engineering Applications of Artificial Intelligence*, vol. 126, p. 106816, 2023.
- [76] K. Cosmas and K. Kenichi, “Utilization of fpga for onboard inference of landmark localization in cnn-based spacecraft pose estimation,” *Aerospace*, vol. 7, no. 11, p. 159, 2020.
- [77] J. Posso, J.-P. Diguët, and A. L. Ekpo, “Real-time spacecraft pose estimation using mixed-precision quantized neural network on cots reconfigurable mpso,” *arXiv preprint arXiv:2407.06170*, 2024.
- [78] Z. Guo, K. Liu, W. Liu, X. Sun, C. Ding, and S. Li, “An overlay accelerator of deeplab cnn for spacecraft image segmentation on fpga,” *Remote Sensing*, vol. 16, no. 5, p. 894, 2024.
- [79] T. P. Swaminathan, C. Silver, T. Akilan, and J. Kumar, “Benchmarking deep learning models on nvidia jetson nano for real-time systems: An empirical investigation,” *Procedia Computer Science*, vol. 260, pp. 906–913, 2025.
- [80] D. K. Alqahtani, M. A. Cheema, and A. N. Toosi, “Benchmarking deep learning models for object detection on edge computing devices,” in *International Conference on Service-Oriented Computing*. Springer, 2024, pp. 142–150.
- [81] P. Li, X. Wang, K. Huang, Y. Huang, S. Li, and M. Iqbal, “Multi-model running latency optimization in an edge computing paradigm,” *Sensors*, vol. 22, no. 16, p. 6097, 2022.
- [82] F. Sadeghi and S. Levine, “Cad2rl: Real single-image flight without a single real image,” *arXiv preprint arXiv:1611.04201*, 2016.
- [83] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, “Structured domain randomization: Bridging the reality gap by context-aware synthetic data,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7249–7255.
- [84] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,”

- in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.
- [85] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, “Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2100–2110.
 - [86] G. Csurka *et al.*, *Domain adaptation in computer vision applications*. Springer, 2017, vol. 2.
 - [87] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4068–4076.
 - [88] T. Yao, Y. Pan, C.-W. Ngo, H. Li, and T. Mei, “Semi-supervised domain adaptation with subspace learning for visual recognition,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 2142–2150.
 - [89] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, “Deep reconstruction-classification networks for unsupervised domain adaptation,” in *European conference on computer vision*. Springer, 2016, pp. 597–613.
 - [90] Y. Ganin and V. Lempitsky, “Domain-adversarial training of neural networks,” in *Journal of Machine Learning Research*, vol. 17, no. 59, 2016, pp. 1–35.
 - [91] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *CVPR*, 2017, pp. 7167–7176.
 - [92] H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon, “Adversarial multiple source domain adaptation,” in *NeurIPS*, 2018, pp. 8559–8570.
 - [93] T. H. Park and S. D’Amico, “Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap,” *Advances in Space Research*, vol. 73, no. 11, pp. 5726–5740, 2024.
 - [94] M. Jawaid, M. Märtens, and T.-J. Chin, “Event-rgb fusion for spacecraft pose estimation under harsh lighting,” *Aerospace Science and Technology*, p. 111039, 2025.
 - [95] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

- [96] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, “Bounding box regression with uncertainty for accurate object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2888–2897.
- [97] J. Mun and H. Choi, “Uncertainty prediction for monocular 3d object detection,” *Sensors*, vol. 23, no. 12, p. 5395, 2023.
- [98] R. Zhang, J. Dong, C. Ma, R. Shen, and S. Wang, “Harnessing uncertainty-aware bounding boxes for unsupervised 3D object detection,” *arXiv preprint arXiv:2408.00619*, 2024.
- [99] Z. Luo, Z. Wang, Y. Huang, L. Wang, T. Tan, and E. Zhou, “Rethinking the heatmap regression for bottom-up human pose estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 264–13 273.
- [100] C. Lu and P. Koniusz, “Few-shot keypoint detection with uncertainty learning for unseen species,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 416–19 426.
- [101] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, “6-DoF object pose from semantic keypoints,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2011–2018.
- [102] K. Li, H. Zhang, and C. Hu, “Learning-based pose estimation of non-cooperative spacecrafts with uncertainty prediction,” *Aerospace*, vol. 9, no. 10, p. 592, 2022.
- [103] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Ep n p: An accurate o (n) solution to the p n p problem,” *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.
- [104] H. Yuan, H. Chen, J. Wu, and G. Kang, “Non-cooperative spacecraft pose estimation based on feature point distribution selection learning,” *Aerospace*, vol. 11, no. 7, p. 526, 2024.
- [105] H. Yang, X. Xiao, M. Yao, Y. Xiong, H. Cui, and Y. Fu, “Pvspe: A pyramid vision multitask transformer network for spacecraft pose estimation,” *Advances in Space Research*, vol. 74, no. 3, pp. 1327–1342, 2024.
- [106] D. Mohlin, J. Sullivan, and G. Bianchi, “Probabilistic orientation estimation with matrix fisher distributions,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 4884–4893, 2020.

- [107] T. O. Dixon, S. A. Giles, and A. A. Gorodetsky, “Predicting uncertainty in vision-based satellite pose estimation using deep evidential regression,” *Aerospace Science and Technology*, vol. 160, p. 110055, 2025.
- [108] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [109] Z.-H. Zhou, *Machine learning*. Springer nature, 2021.
- [110] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2020.
- [111] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain,” *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [112] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [113] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [114] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [115] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [116] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [117] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.
- [118] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
- [119] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. icml*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.

- [120] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [121] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, vol. 4, no. 5, p. 11, 2015.
- [122] J. S. Bridle, “Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition,” in *Neurocomputing: Algorithms, architectures and applications*. Springer, 1990, pp. 227–236.
- [123] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.
- [124] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.
- [125] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.
- [126] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.
- [127] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.
- [128] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.” *Journal of machine learning research*, vol. 12, no. 7, 2011.
- [129] T. Tieleman, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, p. 26, 2012.
- [130] D. P. Kingma, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [131] C. Cortes, M. Mohri, and A. Rostamizadeh, “L2 regularization for learning kernels,” *arXiv preprint arXiv:1205.2653*, 2012.

- [132] M. Schmidt, G. Fung, and R. Rosales, “Fast optimization methods for l1 regularization: A comparative study and two new approaches,” in *European Conference on Machine Learning*. Springer, 2007, pp. 286–297.
- [133] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [134] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. pmlr, 2015, pp. 448–456.
- [135] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.
- [136] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.
- [137] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [138] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [139] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
- [140] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, “Wing loss for robust facial landmark localisation with convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2235–2245.
- [141] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [142] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

- [143] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
- [144] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [145] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [146] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [147] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.
- [148] —, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [149] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas,” *Machine learning and knowledge extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.
- [150] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1125–1134.
- [151] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014.
- [152] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *International Conference on Machine Learning*. PMLR, 2016, pp. 1050–1059.
- [153] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [154] M. Sensoy, L. Kaplan, and M. Kandemir, “Evidential deep learning to quantify classification uncertainty,” in *Advances in Neural Information Processing Systems*, 2018, pp. 3179–3189.

- [155] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.
- [156] R. Banner, Y. Nahshan, and D. Soudry, “Post training 4-bit quantization of convolutional networks for rapid-deployment,” in *Advances in Neural Information Processing Systems*, 2019, pp. 7950–7958.
- [157] S. A. Tailor, J. Fernandez-Marques, and N. D. Lane, “Degree-quant: Quantization-aware training for graph neural networks,” *arXiv preprint arXiv:2008.05000*, 2020.
- [158] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, “Haq: Hardware-aware automated quantization with mixed precision,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.
- [159] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Gutttag, “What is the state of neural network pruning?” *Proceedings of machine learning and systems*, vol. 2, pp. 129–146, 2020.
- [160] H. Cheng, M. Zhang, and J. Q. Shi, “A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [161] NVIDIA Corporation, “NVIDIA TensorRT,” <https://developer.nvidia.com/tensorrt>, 2024.
- [162] Google, “TensorFlow Lite,” <https://www.tensorflow.org/lite>, 2024.
- [163] Microsoft, “ONNX Runtime,” <https://onnxruntime.ai/>, 2024.
- [164] Intel Corporation, “OpenVINO Toolkit,” <https://docs.openvino.ai/>, 2024.
- [165] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, “Complete solution classification for the perspective-three-point problem,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 8, pp. 930–943, 2003.
- [166] D. F. DeMenthon and L. S. Davis, “Model-based object pose in 25 lines of code,” *International journal of computer vision*, vol. 15, no. 1, pp. 123–141, 1995.
- [167] C.-P. Lu, G. D. Hager, and E. Mjølness, “Fast and globally convergent pose estimation from video images,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 6, pp. 610–622, 2000.

- [168] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi, “Revisiting the pnp problem: A fast, general and optimal solution,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2344–2351.
- [169] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” *Advances in neural information processing systems*, vol. 30, 2017.
- [170] L. M. G. Venancio, L. Carminati, J. Amiaux, L. Bonino, G. Racca, R. Vavrek, R. Laureijs, A. Short, T. Boenke, and P. Strada, “Status of the performance of the euclid spacecraft,” in *Space telescopes and instrumentation 2020: optical, infrared, and millimeter wave*, vol. 11443. SPIE, 2020, pp. 45–60.
- [171] V. Dubanchet, J. B. Romero, K. N. Gregertsen, H. Austad, J. Gancet, K. Natusewicz, J. Vinals, G. Guerra, G. Rekleitis, I. Paraskevas *et al.*, “Eross project—european autonomous robotic vehicle for on-orbit servicing,” in *International Symposium on Artificial Intelligence, Robotics and Automation in Space, (i-SAIRAS’20). USA: Pasadena, California*, 2020.
- [172] M. A. Roa, A. Beyer, I. Rodríguez, M. Stelzer, M. de Stefano, J.-P. Lutze, H. Mishra, F. Elhardt, G. Grunwald, V. Dubanchet *et al.*, “Eross: In-orbit demonstration of european robotic orbital support services,” in *2024 IEEE Aerospace Conference*. IEEE, 2024, pp. 1–9.
- [173] V. Dubancheta, J. Bejar Romero, I. Paraskevas, P. Lopez Negro, A. Cuffolo, F. Maye, A. Rodríguez Reina, P. Romero Manrique, M. Alonso, S. Torralbo Dezainde *et al.*, “Eross project—ground validation of an autonomous gnc architecture towards future european servicing missions,” in *Proceedings of the 72nd International Astronautical Congress (IAC)*, 2021, pp. 25–29.
- [174] T. H. Park, S. Sharma, and S. D’Amico, “Towards robust learning-based pose estimation of noncooperative spacecraft,” *arXiv preprint arXiv:1909.00392*, 2019.
- [175] A. Lotti, D. Modenini, P. Tortora, M. Saponara, and M. A. Perino, “Deep learning for real-time satellite pose estimation on tensor processing units,” *Journal of Spacecraft and Rockets*, vol. 60, no. 3, pp. 1034–1038, 2023.
- [176] C. Schubert, K. Black, D. Fonseka, A. Dhir, J. Deutsch, N. Dhamani, G. Martin, and M. Akella, “A pipeline for vision-based on-orbit proximity operations using deep learning and synthetic imagery,” in *2021 IEEE Aerospace Conference (50100)*. IEEE, 2021, pp. 1–15.

- [177] G.-Q. Wei and S. Ma, “A complete two-plane camera calibration method and experimental comparisons,” in *1993 (4th) International Conference on Computer Vision*. IEEE, 1993, pp. 439–446.
- [178] D. Conde, J. Balado, M. Soilán, J. Martínez, and P. Arias, “Lidar data processing for digitization of the castro of santa trega and integration in unreal engine 5,” *International Journal of Architectural Heritage*, pp. 1–21, 2023.
- [179] “Tensorflow object detection api tutorial (latest),” 2025, accessed: 2025-10-09. [Online]. Available: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/>
- [180] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: fast and flexible image augmentations,” *Information*, vol. 11, no. 2, p. 125, 2020.
- [181] NVIDIA Corporation, “Jetson nano,” <https://developer.nvidia.com/embedded/jetson-nano>, 2025, accessed: 2025-10-09.
- [182] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, “Dota: A large-scale dataset for object detection in aerial images,” *CVPR*, 2018.
- [183] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” *CoRR*, vol. abs/1603.08155, 2016. [Online]. Available: <http://arxiv.org/abs/1603.08155>
- [184] M. Seitzer, “pytorch-fid: FID Score for PyTorch,” <https://github.com/mseitzer/pytorch-fid>, August 2020, version 0.3.0.
- [185] “Universal robots ur10e collaborative robot – product specifications,” <https://www.universal-robots.com/products/ur10e/>, Universal Robots, (Accessed: 2025-09-12).
- [186] “Optitrack prime-13w camera,” <https://www.optitrack.com/cameras/prime-13w/>, OptiTrack, 2025, accessed: 2025-10-16.
- [187] “Led video light godox sl-60w daylight — store godox.pl,” <https://store.godox.pl/en/led-lamps/160-led-video-light-godox-sl-60w-daylight-6952344210307.html>, Godox Photo Equipment Co., Ltd., (Accessed: 2025-09-12).
- [188] O. Kedilioglu, T. M. Bocco, M. Landesberger, A. Rizzo, and J. Franke, “Arucoe: Enhanced aruco marker,” in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, 2021, pp. 878–881.

- [189] “Intel® realSense™ depth camera d455 - product specifications,” <https://www.intel.com/content/www/us/en/products/sku/205847/intel-realsense-depth-camera-d455/specifications.html>, Intel Corporation, (Accessed: 2025-09-12).
- [190] “Hand-eye calibration - moveit documentation,” https://moveit.picknik.ai/humble/doc/examples/hand_eye_calibration/hand_eye_calibration_tutorial.html, PickNik Robotics, (Accessed: 2025-09-12).
- [191] I. Corporation, “Intel realSense ros driver,” <https://github.com/IntelRealSense/realsense-ros>, (Accessed: 2025-09-12).
- [192] D. Milanés-Hermosilla, R. Trujillo Codorniú, R. López-Baracaldo, R. Sagaró-Zamora, D. Delisle-Rodriguez, J. J. Villarejo-Mayor, and J. R. Nunez-Alvarez, “Monte carlo dropout for uncertainty estimation and motor imagery classification,” *Sensors*, vol. 21, no. 21, p. 7241, 2021.



Finanziato dall'Unione europea – Next Generation EU,
Missione 4, Componente 2, Investimento 3.3 (D.M. 117/2023)
CUP J33C22001550009