ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

SCIENZE E TECNOLOGIE AEROSPAZIALI

Ciclo 37

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

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

DEEP LEARNING FOR SPACECRAFT POSE ESTIMATION: FROM ALGORITHMS TO HARDWARE-IN-THE-LOOP TESTBED

Presentata da: Alessandro Lotti

Coordinatore Dottorato

Dario Modenini

Supervisore

Paolo Tortora

Co-supervisore

Dario Modenini

Esame finale anno 2026

Abstract

The rapid growth of the New Space economy is increasing demand for in-orbit servicing and active debris removal missions that rely on rendezvous and proximity operations. In this context, relative pose estimation between a chaser and its target is a key enabling technology, achievable with various sensors (e.g., cameras, LIDAR). Monocular vision is attractive for its minimal system budget footprint. However, the complexity of space imagery and onboard deployment constraints still hinder its adoption in space. Monocular satellite pose estimation methods, especially when paired with deep learning for image processing, still suffer from domain gap, high computational cost, and limited uncertainty quantification. This work investigates these issues through an extensive set of experiments.

First, a domain-adversarial, Vision Transformer-based pipeline is introduced to reduce the performance gap between synthetic and real imagery. This approach ranked fourth and fifth in ESA's second Satellite Pose Estimation Challenge. Architectural and data-level strategies, such as generative models and hybrid neural network architectures, are then investigated to further mitigate domain gap while lowering computational cost. A parametric study is conducted to scale image processing on Edge TPUs, showing real-time performance with minimal energy consumption. Additionally, Sparse Variational Gaussian Processes (SVGPs) are explored to provide uncertainty estimates from neural network feature maps, showing a moderate-to-strong correlation between predicted uncertainty and actual errors on synthetic images.

For unknown targets, the pipeline must begin with an inspection phase. To this aim, lightweight localization models are evaluated on previously unseen spacecraft at far range, followed by a lightweight deep-learning-based feature matcher that drives a frame-to-frame pose estimation pipeline to recover camera motion. Results highlight current dataset limitations for this task.

Finally, the development, software integration, and calibration of a HIL testbed at the University of Bologna are presented, together with a dataset generation pipeline featuring a nearly symmetric target.

Acknowledgments

My first acknowledgment goes to Professors Dario Modenini and Paolo Tortora, who gave me the opportunity to achieve this milestone. I thank them for their constant availability and support, especially during difficult times.
I also thank Professor Tat-Jun Chin for granting me the opportunity to carry out my research period abroad with his research group, and I extend my gratitude to Marcus and Yassir for their constant support and proactivity. Thanks also to the people who made the experience in Adelaide special, particularly Victor and Sofia.
A special thanks goes to the entire Tecnopolo group, in particular to Daniele, and to my friends. Above all, for the incredible effort and willingness shown in helping my family following the 2023 flood. Having everyone from PhD students to full professors in our backyard helping to shovel mud and recover pieces of a life upended by that exceptional event was a beautiful demonstration of solidarity that will never be forgotten.
I thank my family for their boundless support and those who were there through the highs and lows of these years. I dedicate my PhD to my grandfather Sauro. You left at the beginning of this journey, making my start an uphill climb. I missed being able to tell you about the small steps along this path, and now that it may be finished, I dedicate it to you.

Contents

Abstract iii

Acknowledgements iv

Contents vi

List of Tables x

List of Figures xiii

List of Acronyms xvii

List of Symbols xx

1 Introduction 1
- 1.1 Historical background and motivation . . . 2
- 1.2 Literature review and research directions . . . 3
  - 1.2.1 Satellite pose estimation . . . 3
  - 1.2.2 Camera pose estimation around unknown satellites . . . 9
  - 1.2.3 Robotic-arm based testbed facilities for HIL validation . . . 10
- 1.3 Thesis focus . . . 12
- 1.4 Document structure . . . 12

2 Preliminaries 15
- 2.1 Reference frames and relative poses . . . 15
  - 2.1.1 Active and passive rotations . . . 16
  - 2.1.2 Notation and composition rules . . . 17
  - 2.1.3 Homogeneous transformations . . . 18
  - 2.1.4 Camera-chaser and frame-to-frame relative poses . . . 19
- 2.2 Recurrent evaluation metrics . . . 21
- 2.3 Deep learning architectures for image processing . . . 22
- 2.4 Implementation framework . . . 24

3 Pose estimation of known uncooperative targets **27**
3.1 Domain gap in satellite pose estimation 28
3.1.1 A domain adversarial approach with Vision Transformers . 28
3.1.2 Comparison with CNNs and embedded deployment constraints 36
3.1.3 A Hybrid CNN-Transformer architecture for efficient inference 37
3.1.4 Narrowing domain gap at the data level exploring Generative Adversarial Networks 41
3.2 Real-time satellite pose estimation on Edge TPU 45
3.2.1 Motivation and hardware context 45
3.2.2 Detection model for the Edge TPU 46
3.2.3 Scaling landmark regressors 47
3.2.4 Training and TPU-Specific Optimization 48
3.3 Uncertainty estimation using SVGPs 54
3.3.1 Proposed methodology 54
3.3.2 Results and limitations 58
3.4 Concluding remarks 62

4 Pose estimation of unknown targets **65**
4.1 Far range detection 65
4.2 Pose estimation 69
4.2.1 Dataset processing 69
4.2.2 Deep learning-based feature matching 71
4.2.3 Relative pose estimation pipeline 74

5 Development of a robotic HIL testbed **77**
5.1 Overview of the facility 78
5.1.1 Custom Sun simulator 80
5.1.2 Spacecraft mockup 82
5.2 Software integration 83
5.2.1 Linear axis integration 85
5.2.2 ROS 2 system architecture 87
5.3 Calibration procedures 89
5.3.1 Camera calibration 91

5.3.2 Hand-eye calibration 92
5.3.3 Mockup pose calibration in the world frame 95
5.3.4 Repeatability assessment using OptiTrack motion capture . 96
5.4 ALMASat dataset generation pipeline 97
5.4.1 Synthetic images rendering 98
5.4.2 Real images acquisition 99

6 Conclusions **103**
6.1 Future perspectives 105

List of Tables

3.1 Swin Transformers variants 30
3.2 Architectural details of the main SPEC 2021 submissions (Lotti et al., 2023a) 34
3.3 Results of the main SPEC 2021 submissions (Lotti et al., 2023a) . . 34
3.4 Data augmentation strategy applied during training of solutions C and D 35
3.5 SPEC 2021 final Lightbox leaderboard (partial) (Park et al., 2023) 36
3.6 SPEC 2021 final Sunlamp leaderboard (partial) (Park et al., 2023) 36
3.7 Pose estimation errors across real domains for CNN-based (EfficientNet-B5) and Transformer-based (Swin-Tiny) architectures of similar capacity (Lotti et al., 2023a) 37
3.8 Pose estimation errors achieved by the hybrid CNN–Transformer architecture with progressive augmentations (Lotti et al., 2023a) . 40
3.9 Pose estimation errors of the hybrid CNN–Transformer architecture compared with the Transformer-only model (Lotti et al., 2023a) . . 40
3.10 Impact of GAN preprocessing on pose estimation performance (Lotti, 2023) 44
3.11 EfficientNet-Lite models specifications relative to Lite0 variant (Lotti et al., 2023b) 47
3.12 Edge TPU pipelines regression head structure (Lotti et al., 2023b) 48
3.13 Summary of training configuration for Edge TPU-optimized SDN and LRN 49
3.14 Pose estimation errors achieved by the Edge TPU-optimized pipeline variants, in TensorFlow float32 version (Lotti et al., 2023b) 49
3.15 IoU scores for detection networks in TensorFlow and TFLite implementations (Lotti et al., 2023b) 50
3.16 Edge TPU compilation statistics 52

3.17 Performance of Edge TPU-optimized models on Coral Dev Board Mini vs Coral Dev Board, demonstrating the impact of memory transfer protocol 53
3.18 Patch classification accuracy for CNN and SVGP models across synthetic and real domains (Lotti et al., 2025b) 59
3.19 Landmark regression error (MAE) on ground-truth ROIs comparing dense and SVGP regression heads (Lotti et al., 2025b) 59
3.20 Pose errors with and without uncertainty filtering (Lotti et al., 2025b) 60
3.21 Pearson correlation between SVGP-predicted pose uncertainties and actual pose errors (Lotti et al., 2025b) 61
3.22 Execution time breakdown for uncertainty-aware pose estimation pipeline on Edge TPU (Lotti et al., 2025b) 61

4.1 Repartitioning of the PoseBowl Detection track public dataset for local evaluation 66
4.2 IoU scores for detection pipelines on the validation set 68

5.1 Main components of the robotic testbed (Lotti et al., 2025a) 79
5.2 Sun simulator components 81
5.3 CANopen motion control commands, according to the CiA 402 Profile Position Mode 86
5.4 Estimated camera intrinsic parameters 92
5.5 Hand-eye calibration results for UR10e tool–camera optical axis transformation 94

List of Figures

1.1 Sample images from the SPEED+ dataset (Park et al., 2022), synthetic (left), lightbox (center), sunlamp (right). (Lotti et al., 2023a) 8

2.1 Pinhole camera model illustrating the camera frame (c), optical axis, and image plane (Bradski, 2000) 19

3.1 Three-stage pose estimation pipeline comprising satellite detection, landmark regression, and PnP-based pose estimation (Lotti et al., 2023a) 29
3.2 Domain adversarial training architecture adopted for SPEC 2021; a Swin-based encoder feeds both a regressor and a domain discriminator, with reversed gradients ($-\gamma$) forcing domain-invariant feature extraction (Lotti et al., 2023a) 31
3.3 Schedule of the adversarial loss weighting parameter γ, gradually increasing from epoch 1 to 40 to stabilize early training 32
3.4 TPU core batching strategy ensuring 50% synthetic and 50% real images in each core's sub-batch for balanced gradient contributions during adversarial training 33
3.5 Synthetic light artifacts (sunflare augmentation) applied during training to improve robustness to illumination conditions (Lotti et al., 2023a) 35
3.6 Hybrid CNN-ViT landmark regression pipeline; EfficientNet Lite 4 extracts spatial features, followed by a Tiny ViT for feature correlation and a regression head for keypoint prediction 38
3.7 Strong photometric transformations used for domain generalization; histogram equalization, color inversion, solarization, and posterization applied to a synthetic training sample (Lotti et al., 2023a) 39

3.8 GAN training scheme where the generator transforms synthetic images to match real domain appearance, while the discriminator learns to distinguish generated from real samples. Classification feedback drives both networks' updates 43
3.9 Synthetic renders transformed to Lightbox domain (left) and Sunlamp domain (right) using GAN-based image-to-image translation, reproducing characteristic illumination and appearance of each target dataset (Lotti, 2023) 44
3.10 Pose error vs. inference time for NN variants (Lite0-4) on Coral Dev Board Mini CPU showing how TFLite conversion and quantization progressively increase throughput, the latter at the cost of increased pose estimation errors 51
3.11 CPU vs. Edge TPU performance on Coral Dev Board Mini. Inference runtime for detection and regression networks (left), pose error vs. throughput for the full pipeline variants (right) (Lotti et al., 2023b) 53
3.12 Uncertainty-aware pose estimation pipeline; SVGP heads on ROI detection and landmark regression networks provide confidence estimates, enabling frame rejection when uncertainty exceeds thresholds ε_1 or ε_2. Accepted frames yield pose estimates with uncertainty bounds (Lotti et al., 2025b) 56
3.13 Image divided into grid cells with Segment Anything pseudo-labels (green mask) indicating satellite presence for binary patch classification 57
3.14 Epistemic uncertainty distributions from the SVGP classifier; synthetic validation images cluster at low uncertainty, Lightbox shows moderate increase, and Sunlamp exhibits substantially higher values reflecting domain shift (Lotti et al., 2025b) 60

4.1 Sample images with annotated bounding box from the PoseBowl Detection Track dataset (NASA and DrivenData, 2024) 66
4.2 Sliding window detection; when initial confidence is below threshold, inference is repeated on up to five sub-windows to reduce information loss from resizing, the bounding box with highest score is retained 67

4.3 Class imbalance mitigation; the single positive patch containing the satellite (left) is oversampled (and augmented) to balance training data (right) . . . 68
4.4 Bounding box diagonal distribution in the validation set (left), IoU vs. bounding box diagonal size showing localization accuracy degradation for smaller targets (right) . . . 69
4.5 Sample images from the PoseBowl Pose Estimation Track dataset, illustrating different satellite models (NASA and DrivenData, 2024) 70
4.6 Feature matching between original and 2D-augmented image pairs; yellow lines connect correspondences across viewpoint and photometric transformations . . . 74

5.1 View of the robotic testbed facility at the Microsatellites and Space Microsystems Laboratory; dark room with labeled hardware components (left), control area with workstations and control boxes (right) . . . 78
5.2 Custom Sun simulator with metal halide bulb and parabolic reflector, mounted on a mobile support structure . . . 80
5.3 Light spots captured through a 75 μm pinhole at two camera positions, approximately 4 cm left (left) and right (right) from the reflector center, demonstrating beam collimation with an estimated divergence of $\approx 0.52°$. The central shadow is caused by the bulb enclosure . . . 81
5.4 ALMASat mockup used as target spacecraft (left) and corresponding CAD model in Blender (material preview) for synthetic image generation (right) . . . 82
5.5 CAD model of the robotic testbed assembly, used to export meshes to build the URDF file and RViz scene . . . 84
5.6 RViz visualization of the robotic system showing the safe volume for collision-free path planning (orange), Sun simulator volume (red), and principal reference frames (laboratory, camera optical, ALMASat body). The left panel displays joint sliders for real-time control and the live camera stream . . . 84

5.7 Layered ROS 2 architecture of the robotic testbed, comprising user interface and visualization (top), motion planning via MoveIt 2 with three planning groups, data integration for joint state coordination, and hardware drivers (bottom). Arrows indicate the flow of commands and state feedback between layers 88
5.8 TF tree of the robotic testbed showing the hierarchy of coordinate frames from the lab reference frame to the camera optical frame and static elements 90
5.9 Checkerboard poses used for camera intrinsic calibration, distributed on a spherical grid within the camera field of view at three distances (left), and per-image reprojection errors with a mean of 0.07 px (right) 92
5.10 Wireframe model of the ALMASat mockup with labeled landmarks 95
5.11 Annotated mockup image with 2D landmarks, reprojection errors, and estimated body-fixed frame 96
5.12 Positional repeatability of the 7-DoF robotic system measured via OptiTrack; distribution of Euclidean errors from mean position (left) and XY plane dispersion with 3σ boundary (right) 97
5.13 Representative sample of rendered images from the ALMASat synthetic dataset (Lotti et al., 2025a) 99
5.14 Representative sample of real images captured with the robotic testbed (Lotti et al., 2025a) 101

List of Acronyms

ADCS	Attitude Determination and Control System
ADR	Active Debris Removal
AI	Artificial Intelligence
API	Application Programming Interface
ARM	Advanced RISC Machines
ASIC	Application-Specific Integrated Circuit
CAD	Computer Aided Design
CAN	Controller Area Network
CiA	CAN in Automation
CCT	Correlated Color Temperature
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
CPOs	Close Proximity Operations
CPU	Central Processing Unit
CUT	Contrastive Unpaired Translation
DoF	Degrees of Freedom
ELBO	Evidence Lower Bound
EPnP	Efficient Perspective-n-Point
fps	Frames Per Second
FOV	Field of View
GAN	Generative Adversarial Network
GeLU	Gaussian Error Linear Unit
GNC	Guidance Navigation and Control
GP	Gaussian Processes
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HQ	High Quality
HIL	Hardware-in-the-Loop
HMI	Hydrargyrum Medium-arc Iodide (lamp)

HRNet	High-Resolution Network
IK	Inverse Kinematics
IOS	In Orbit Servicing
IoU	Intersection over Union
IPPE	Infinitesimal Plane-based Pose Estimation
JSON	JavaScript Object Notation
KL	Kullback–Leibler
LIDAR	Light Detection and Ranging
LM	Levenberg–Marquardt
LRN	Landmarks Regression Network
MAE	Mean Absolute Error
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NN	Neural Network
NCE	Noise Contrastive Estimation
PCIe	Peripheral Component Interconnect Express
PnP	Perspective-n-Point
PPM	Profile Position Mode
QAT	Quantization-Aware Training
RANSAC	Random Sample Consensus
RGB	Red, Green, Blue
RoI	Region of Interest
ReLU	Rectified Linear Unit
ReLU6	Rectified Linear Unit 6 (capped at 6)
ROS	Robot Operating System
RViz	ROS Visualization
SA	Segment Anything
SDN	Satellite Detection Network
SFM	Structure from Motion
SLAM	Simulteous Localization and Mapping
SoC	System on Chip
SPEC	Satellite Pose Estimation Competition
SPEED	Spacecraft Pose Estimation Dataset
SSD	Single Shot Detector

SVGP	Sparse Variational Gaussian Processes
SPN	Spacecraft Pose Network
TM	Template Matching
TF	TensorFlow
TF2	`tf2` transform library (ROS 2)
TFLite	TensorFlow Lite
TPU	Tensor Processing Unit
URCap	Universal Robots Capability
URDF	Unified Robot Description Format
UV	Ultraviolet
ViT	Vision Transformer
XML	Extensible Markup Language

List of Symbols

$\mathbb{R}$	set of real numbers
$\mathbb{R}^{3\times3}$	set of 3×3 real matrices
$\mathbb{R}^{4\times4}$	set of 4×4 real matrices
$\mathbb{N}$	set of natural numbers
$SO(3)$	special orthogonal group (3D rotations)
$SE(3)$	special Euclidean group (rigid transformations)
$\mathbf{R}_{ij}$	rotation matrix from frame j to frame i
$\mathbf{T}_{ij}$	homogeneous transformation matrix from frame j to frame i
$\mathbf{t}_{i\to j\|k}$	translation vector from i to j expressed in frame k
$\mathbf{K}$	camera intrinsic matrix
$\mathbf{I}$	identity matrix
$\mathbf{0}$	zero vector
$\mathbf{S}$	similarity matrix
$\mathbf{A}_i$	relative robot motion in hand-eye calibration
$\mathbf{B}_i$	relative camera motion in hand-eye calibration
$\mathbf{X}$	unknown transformation in hand-eye calibration
$\mathbf{p}_i$	point/position vector in frame i
$\tilde{\mathbf{p}}_i$	point in homogeneous coordinates in frame i
$\mathbf{q}$	unit quaternion
q_w	scalar part of quaternion
$\mathbf{q}_v$	vector part of quaternion
q_x, q_y, q_z	quaternion vector components
$\hat{\mathbf{n}}$	unit axis vector
$\hat{\mathbf{x}}_B^A, \hat{\mathbf{y}}_B^A, \hat{\mathbf{z}}_B^A$	unit basis vectors of frame B expressed in frame A
$\mathbf{r}_i$	residual/regression vector
$\hat{\mathbf{O}}_A, \hat{\mathbf{O}}_B$	output features/embeddings
θ	rotation angle
s	projective scale factor

u, v	pixel coordinates
f_x, f_y	focal lengths
c_x, c_y	principal point coordinates
τ	temperature parameter
γ	adaptive weight parameter
e_t	translation error
e_q	rotation/quaternion error
E	total pose error
N_B	batch size
N_S	number of samples
$\mathcal{L}$	loss function
$\mathcal{L}_{\text{ELBO}}$	evidence lower bound loss
L_{ADV}	adversarial loss
L_{REG}	regression loss
$\mathcal{L}_{A \to B}$	directional contrastive loss
$\| \cdot \|$	Euclidean norm
$\| \cdot \|_1$	L1 norm
$\| \cdot \|_2$	L2 norm
$\det(\cdot)$	determinant
$\| \cdot \|$	absolute value or cardinality
$\langle \cdot \rangle$	mean/average
$\langle\langle \cdot \rangle\rangle$	median
est	estimated value (subscript)
GT	ground truth value (subscript)
^{T}	matrix transpose (superscript)
$^{-1}$	matrix inverse (superscript)
w	world frame
c	camera frame
t	target/object frame
b	base/robot base frame
g	gripper/end-effector frame
$\mathcal{V}_A, \mathcal{V}_B$	foreground points sets
$k(\cdot, \cdot)$	kernel function
m	variational mean

M	number of inducing points
M	millions of parameters (neural network size)
k_1, k_2	radial distortion coefficients
p_1, p_2	tangential distortion coefficients
μ	mean
σ	standard deviation
$\theta_E, \theta_R, \theta_D$	encoder, regressor, and discriminator parameters
$\mathbf{F}, \mathbf{G}$	feature map tensors
$\mathbf{z}$	inducing variables (SVGP)
$\cap$	set intersection
$\cup$	set union
$\mathcal{O}$	Big-O notation (computational complexity)

1

Introduction

Rendezvous and proximity operations are attracting growing attention as the New Space Economy drives demand for complex missions such as In-Orbit Servicing (IOS) and Active Debris Removal (ADR). Given the inherent collision risks in these activities, developing autonomous navigation capabilities is essential to ensure mission safety and success. ADR aims to reduce debris by actively removing defunct satellites or fragments from orbit, while IOS enables satellites to be refueled, relocated, upgraded, extending their operational life. Close Proximity Operations (CPOs), a core component of IOS and ADR, involve tasks such as inspection and docking with satellites, including potentially non-cooperative targets. These operations carry significant risks due to the possibility of collisions during manoeuvres. Given the importance and challenges of CPOs, particularly for non-cooperative scenarios, this work focuses on pose estimation techniques to support the safe and autonomous execution of such tasks. This introduction opens with a brief background and the motivation framing the work, followed by the formulation of the research questions that shaped the research. The section concludes with an outline of the document's structure.

1.1 Historical background and motivation

The first docking in space dates back to 1966 with the Gemini VIII mission, which successfully docked with the Agena Target Vehicle. In the following years NASA continued to develop docking technologies with the Gemini, Apollo, and Shuttle programs. 1969 marked the first rendezvous between two crewed spacecraft, Soyuz 4 and Soyuz 5, while in 1975 the Apollo-Soyuz mission marked the first international docking in space. All of these missions were crew-controlled and highlighted the challenges of docking in space, with many failures, from Gemini VIII itself, where a stuck thruster caused the docked craft to spin, to the Soyuz-Salyut docking failures. As a result, spacecraft design rapidly shifted toward semi-automated docking technologies and, eventually, to fully automated docking technologies. Since 2008, fully autonomous systems such as the ESA Automated Transfer Vehicle and Japan's H-II Transfer Vehicle have regularly serviced the International Space Station.

In 2020, Northrop Grumman's Mission Extension Vehicle performed the first commercial in-orbit servicing mission, but navigation still relied on multiple sensing modalities and on a cooperative client. By contrast, demonstration missions started to explore the more challenging uncooperative scenario. PRISMA (2010) (Persson et al., 2010) flew a two-satellite formation, Mango and Tango, to test autonomous rendezvous and proximity operations, evaluating relative navigation pipelines. RemoveDEBRIS (2018) (Forshaw et al., 2016) deployed surrogate debris and demonstrated vision-based navigation with onboard cameras and LIDAR. These efforts paved the way for autonomous proximity operations without transponders or fiducial markers, but also exposed unresolved challenges for hand-crafted, traditional image-processing pipelines in camera-based systems. As the following review will show, such methods suffer from poor robustness to low signal-to-noise ratio and rapidly changing illumination conditions, as well as requiring extensive manual tuning. For these reasons the research community is increasingly investigating AI-based approaches, which can in principle be made robust to these challenges through proper training, but introduce new open problems that are the focus of this thesis. Autonomous relative navigation with uncooperative targets is central to upcoming active debris removal missions, such as ClearSpace-1 (Vasconcelos et al., 2025) planned for 2029, making vision-based navigation a key enabler and, within it, 6-DoF pose estimation a core capability.

1.2 Literature review and research directions

The review is organised along three directions that mirror the structure of the document. First, an historical progression of satellite pose estimation methods is presented, from early techniques up to state-of-the-art solutions. Second, the case of unknown targets is addressed. Third, hardware-in-the-loop robotic testbeds are reviewed, as these facilities are essential for bridging the simulation-to-real gap and validating algorithms under representative space conditions.

1.2.1 Satellite pose estimation

The field of spacecraft pose estimation has evolved significantly over the past decade, as documented in several review papers.

1. Opromolla et al. (2017) provided a comprehensive taxonomy of pose determination techniques for close-proximity operations, classifying technological solutions based on sensor types (radio-frequency, satellite-navigation, and electro-optical) and target cooperation levels. Their analysis established that electro-optical sensors offer the best operational flexibility for close-proximity operations regardless of target cooperativeness, while highlighting the trade-offs between active sensors (LIDAR - more robust to illumination but heavier and power-consuming) and passive sensors (cameras - finer resolution and higher update rates but limited 3D information). The review identified critical challenges including the computational burden for initial pose acquisition and the need for autonomous failure detection mechanisms.

2. Pasqualetto Cassinis et al. (2019) examined the robustness and applicability of monocular pose estimation systems for relative navigation with uncooperative spacecraft. The review highlighted the potential of Convolutional Neural Network (CNN)-based methods using transfer learning to achieve accurate pose initialization with relatively small training databases, though performance degradation when the target is not fully in the field of view remains problematic. Additionally, they suggested that the coarse accuracy characterizing these neural networks could be compensated by including

a Perspective-n-Point (PnP) solver which uses the CNN solution as initial guess for the relative pose.

3. Song et al. (2022) surveyed deep learning techniques for 6-DoF relative pose estimation. Spaceborne contributions were found to be predominantly supervised. The survey compared end-to-end regressors with indirect pipelines that combine CNN keypoint detection with geometric PnP and nonlinear refinement, noting that the latter generally achieves higher accuracy and integrates naturally with navigation filters. The authors identified key gaps such as scarce public real-image datasets/benchmarks and few hardware-aware/on-board deployments, reinforcing the need for deployable models and metrics that reflect the synthetic-to-real domain gap.

4. Pauly et al. (2023) provided a comprehensive assessment of current limitations preventing deep learning-based pose estimation algorithms deployment in actual space missions. The review compared existing algorithms in terms of both performance and network architecture size to evaluate deployability constraints. The survey emphasized two critical challenges: the domain gap between synthetic training data and real spaceborne imagery, and the need for lightweight, deployable models suitable for on-board implementation.

Taken together, the reviews converge on electro-optical sensing as the most versatile option for close-proximity relative navigation and identify monocular imagery as the most mass/volume/power-efficient choice. However, this hardware economy shifts complexity to perception.

Classical Methods

When a target model is available onboard, pre-deep-learning era pose estimation methods relied on model-based pipelines that match sensor data to stored 3D geometry. Classical methods detect and match hand-engineered features (points, lines) between 2D images and 3D models, then run iterative solvers to minimize reprojection error, an approach that is sensitive to initialization and prone to local minima. To mitigate correspondence issues, algorithms such as SoftPOSIT (David et al., 2004) and RANSAC (Fischler and Bolles, 1981) were introduced, while Template Matching (TM) pursue correlation against pre-rendered views. Variants

using fiducial markers and TM extensions improved robustness but remained computationally heavy and sensitive to pose/FoV changes, limiting real-time use on small spacecraft. (D'Amico et al., 2014) leveraged perceptual organization (Sobel, Hough) to aid initialization, yet faced similar constraints: high computational burden, and fragility to illumination and background. Sharma et al. (2018b) introduced a method with Weak Gradient Elimination that improved background robustness and achieved order-of-magnitude speedups over RANSAC on PRISMA imagery, but still yielded few high-confidence solutions.

Deep Learning Approaches

The seminal work of Sharma et al. (2018a) introduced deep learning to satellite pose estimation by discretizing the pose space and formulating it as a classification problem using CNNs. Their approach reduced pose estimation to 4 DoF: three for attitude and one for distance. However, this classification-based architecture suffers from inherent limitations as its accuracy depends heavily on the granularity of training poses, and even with large training sets, the achieved accuracy remains limited. Consequently, the authors recommend CNNs primarily as initializers for classical feature-based methods rather than complete solutions. Sharma and D'Amico (2020) introduced the Spacecraft Pose Estimation Dataset (SPEED), comprising 15,000 synthetic images rendered in OpenGL, split into training (12,000) and validation (3,000) sets. SPEED was the first large-scale dataset of its kind for satellite pose estimation. They also proposed the Spacecraft Pose Network (SPN), a multi-head architecture with three heads: the first performs bounding box regression and classification, the second identifies the closest attitude classes from m predefined rotations uniformly distributed in SO(3), and the third outputs weights for the n nearest classes. During inference, the final attitude is estimated by computing weighted averages of the n nearest quaternions.

A significant boost to deep learning-based pose estimation methods came in 2019 when ESA's Advanced Concepts Team and Stanford University's Space Rendezvous Laboratory organized the Satellite Pose Estimation Challenge (SPEC) (Kisantal et al., 2020). This competition aimed to engage the space community in addressing this research challenge and establish a common benchmark for satellite pose estimation algorithms. The pose evaluation metrics introduced in Section 2.2 have since become standard in the literature. The competition

leveraged the SPEED dataset, with submissions ranked by minimizing the pose error $\langle E \rangle$ (see Section 2.2).
All top-ranked solutions exploited neural networks. Specifically, the competition winners, UniAdelaide (Chen et al., 2019), used a three-stage pose-estimation pipeline (see Figure 3.1) featuring an object-detection network, a heatmap-based landmark-regression network, a Perspective-n-Point (PnP) solver, and a final non-linear optimization step. The overall architecture is relatively heavy, leveraging HRNet backbones (Sun et al., 2019) with high input resolution (768×768 px). A similar approach was adopted by Park et al. (2019), which, however, sought to reduce computational cost by employing lightweight networks such as MobileNet (Sandler et al., 2018) and YOLO (Redmon and Farhadi, 2017), together with depth-wise separable convolutions. In contrast, Proença and Gao (2020) proposed an end-to-end pipeline: given an input image, a modified ResNet backbone (He et al., 2016) regresses translation via two fully connected layers, while orientation is obtained through a soft classification over a discretized quaternion space modeled with a Gaussian mixture; at test time, the quaternion is recovered via least-squares fitting. Despite using a $\sim$500M-parameter network, this model achieved only a $\sim$10% improvement over the previous approach while employing $\sim 45\times$ more parameters. Overall, the competition results indicate that:

- NNs are indispensable for perception but not uniformly optimal across all pipeline stages.
- Methods pairing PnP-based geometry with NN-detected keypoints achieved the lowest pose estimation errors.
- With high-resolution imagery and GPU memory limits that constrain input size, a separate localization step (cropping irrelevant regions) brought significant gains in pose accuracy.
- The three-stage pipeline, detection, keypoint regression, and PnP, was established as a strong baseline and remains widely used on state-of-the-art algorithms, including in the experiments of this thesis.

SPEC 2019 inspired improvements to spacecraft pose estimation along several directions. In Hu et al. (2021), the authors observed that identical 2D keypoint

errors can correspond to very different 3D pose errors depending on target distance and viewpoint; accordingly, they introduced a training loss on 3D keypoint positions rather than 2D projections, making the method invariant to target distance.

Several works then addressed computational burden limitations, since most of SPEC 2019 methods were evaluated only on high-end desktop GPUs and were not optimized for low-power embedded devices. Carcagnì et al. (2022) revisited the method of Chen et al. (2019) by replacing the landmark-regression backbone with a lighter network, while Posso et al. (2022) proposed a lightweight variant of the end-to-end approach in Proença and Gao (2020), though with accuracy below the top SPEC submissions. Piazza et al. (2022) focused on reducing the computational burden of the detector. Notably, Black et al. (2021) developed a light pipeline that achieves real-time inference on an Intel Joule 570X board consuming 3.7 W. Wang et al. (2022) pioneered the use of Transformers for satellite pose estimation, noting that CNN-based methods suffer from bias toward texture and lack long-range dependence modeling, improving robustness to scale and viewpoint changes.

Another recent turnaround in the field was the publication of the first hardware-in-the-loop (HIL) dataset, SPEED+ (Park et al., 2022). Designed to assess pose estimation methods across the synthetic-to-real domain gap, it comprises four partitions: rendered train (47,966 images) and validation (11,994 images), and two real test sets acquired from a spacecraft mock-up under different illumination, *sunlamp* (2,791 images) and *lightbox* (6,740 images). Pose distributions are consistent across partitions, with camera–target distances of about 2.5–10 m and uniform sampling of $SO(3)$ rotations; the Tango target does not always fit entirely within the image. A montage in Figure 1.1 highlights the domain gap: rendered and lightbox images exhibit more uniform lighting, whereas the sunlamp set features strong flares that cause occlusions.

SPEED+ has been adopted as a benchmark within the second ESA's sponsored satellite pose estimation challenge (Park et al., 2023). In this context participants were provided with the training and validation partitions, including images and associated poses, and submissions where ranked on the lightbox and sunlamp subsets. The purpose of the challenge was to engage the research community on the topic of domain gap in the context of satellite pose estimation.

Team *lava1302* (Wang et al., 2024) proposed a domain-agnostic, geometry-constrained framework: a CNN predicts 2D keypoint heatmaps and fine-grained

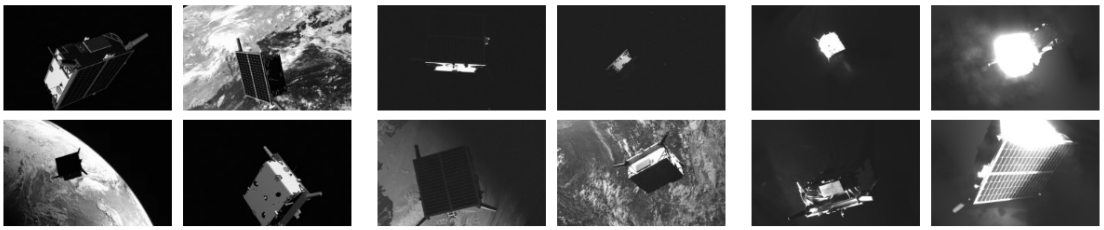

Figure 1.1: Sample images from the SPEED+ dataset (Park et al., 2022), synthetic (left), lightbox (center), sunlamp (right). (Lotti et al., 2023a)

segmentation, and PnP recovers the 6-DoF pose. Training treats poses as latent variables and alternates pseudo-label generation with network optimization (via self-training). A 3D Tango mesh provides supervision via projection with ground-truth (synthetic) or pseudo-poses (HIL), reducing the synthetic–real domain gap. Team *TangoUnchained* adopts the approach of (Chen et al., 2019) and applies adversarial training to improve its robustness across domains.

Overall, all top-ranked teams identified the adversarial training on unlabeled HIL test images at training time to be the crucial component of bridging the domain gap and the same applies to the solution proposed by the author that is introduced in Section 3.1.

Finally, Park and D'Amico (2024) proposed the SPNv2, based on EfficientPose (Bukschat and Vetter, 2020) and multi-task training, simultaneously learning detection, pose regression, keypoint heatmaps, and foreground segmentation. Extensive augmentation on SPEED+ synthetic images are introduced to increse cross-domain robustness. Further domain adaptation comes from unsupervised entropy minimization on unlabeled HIL images, updating the normalization-layer parameters.

While the research community has concentrated the efforts in providing more and more accurate models, uncertainty quantification, in the specific field of satellite monocular pose estimation, remains relatively underexplored. Li et al. (2022) revisited the landmark regression step as a detection task, using the network's detection confidence as a proxy for uncertainty. Dixon et al. (2025) recently adopted Deep Evidential Learning (Amini et al., 2020), which models uncertainty by placing a prior distribution over the likelihood function and learning its parameters directly from data. This enables single-pass uncertainty estimation by regressing distribution parameters rather than point predictions. However, in

both cases uncertainty is learned jointly with the main task, therefore there is no theoretical guarantee that it reflects true model performance, with no bias.

1.2.2 Camera pose estimation around unknown satellites

Most of the methods reviewed in the previous section, despite leveraging deep learning techniques, largely rely on the assumption that the target spacecraft geometry is known a priori. This dependency manifests primarily through their use of PnP algorithms, which require a 3D model to establish 2D-3D correspondences for pose estimation. This assumption was reasonable within the constraints of existing benchmark datasets: both SPEED and SPEED+, which have served as the primary benchmarks for the space research community, feature a single spacecraft model. This limited diversity enabled researchers to reconstruct accurate 3D models from the provided images and pose labels, even before the official wireframe model release, focusing their efforts on improving pose estimation accuracy within this known target framework.
However, in ADR and IOS missions, the target spacecraft may be unknown, damaged, or significantly altered. A 3D model may not be available prior to the mission, particularly when dealing with decades-old satellites, or spacecraft that have suffered structural damage. In these scenarios, PnP-based approaches become inapplicable, necessitating alternative strategies for pose estimation.
In this context, Simultaneous Localization and Mapping (SLAM) (Cadena et al., 2016), which estimates camera poses while simultaneously building an environment map, and Structure from Motion (SfM) (Schonberger and Frahm, 2016), which recovers camera trajectories from image sequences, represent natural approaches for handling unknown targets. These techniques excel at estimating relative camera motion without requiring prior knowledge of the observed scene geometry. However, traditional implementations of these pipelines rely on feature correspondence assumptions that break down in the space domain for the same reasons highlighted in the previous section, namely failure of traditional image processing algorithms on highly aberrated images and large computational burden of deep learning models.
Within an effort to overcome such challenges, in 2024 NASA sponsored two competitions named PoseBowl (NASA and DrivenData, 2024), featuring a detection and a pose estimation track where participants were tasked to develop

detection and camera pose estimation algorithms across different and unknown satellite models while keeping the computational burden constrained. This was a code competition with solutions run on DrivenData servers emulating a realistic space-grade platform featuring no GPU, 3 CPU cores, and 4GB RAM. The two datasets were generated in Blender and post-processed with the addition of complex backgrounds, noise, and blur, with images captured under varying illumination conditions.
The objective of the pose track was to estimate the pose of the camera with respect to the target within an image sequence where the target was considered stationary. Solutions had to exploit exclusively image data without relying on relative dynamics, since the poses were randomized, though with limits to simulate a realistic inspection phase.
The challenge proved significantly complex: only 8 teams managed to beat the trivial solution (predicting zero translation and no rotation for each frame), and the maximum improvement was limited to 5% of the competition pose score metric. This highlights the fundamental difficulty of generalizing pose estimation in space to unknown targets when computational resources are severely limited. While feature matching remains a good candidate method, and many deep learning solutions have been developed by the computer vision community (see for example DeTone et al. (2018), Sarlin et al. (2020), Sun et al. (2021)), to the best of the author's knowledge, no paper has yet explored the topic using deep learning models within the space field, accounting for its specificities, particularly the need for computational efficiency on space-grade hardware.

1.2.3 Robotic-arm based testbed facilities for HIL validation

Numerous testbed facilities have been developed across research institutes, universities, and industry to advance autonomous rendezvous technologies. From a high-level perspective, they can be divided into two categories: robotic-arm and floating-platform testbeds. This section briefly reports on robotic testbeds, which are typically best suited for validating vision-based navigation algorithms, as they offer precise 6-DoF motion control but a limited workspace, with at least some DoF constrained by the robot's reach, whereas floating platforms provide more realistic dynamics but generally fewer controllable DoF.
Germany's DLR operates several distinct large-scale facilities. The Testbed for

Robotic Optical Navigation employs a robotic arm mounted on a 10 m linear track and is used to simulate descent and landing scenarios (Krüger and Theil, 2010). The European Proximity Operations Simulator (version 2) (Benninghoff et al., 2017) is a large-scale facility that supports research on rendezvous and docking. The testbed features two industrial robot arms, one of which is mounted on a 25 m rail, and controllable background illumination to emulate both collaborative rendezvous and ADR missions.

ESA–ESTEC developed the GNC Rendezvous, Approach and Landing Simulator (GRALS) (Pasqualetto Cassinis et al., 2022), a dual-rail facility featuring a wall-mounted 33 m linear track with a KUKA robotic arm and a ceiling-mounted 5.6 m track with a second robotic arm. It also features space-representative lighting, and includes a Vicon Tracker system to provide ground-truth pose references.

GMV's *platform-art*© (Colmenarejo et al., 2019) also supports multi-robot setups. The facility (about 600 m^3) includes a dark room with black curtains and a Sun-like light source for space-representative illumination, and offers high-accuracy metrology. It comprises a 7-DoF KUKA robot on a 16 m linear rail and a 6-DoF Mitsubishi robot.

Stanford's Robotic Testbed for Rendezvous and Optical Navigation (Park et al., 2021) is specifically designed for validating vision-based navigation algorithms and features two 6-DoF KUKA robot arms, one of which is mounted on a ceiling-mounted linear rail, and a set of Vicon motion-tracking cameras. The facility includes multiple Earth-albedo light boxes and a sun lamp to recreate high-fidelity spaceborne illumination conditions.

Similarly, the Zero-G Laboratory at the University of Luxembourg (Olivares-Mendez et al., 2023) is designed to emulate rendezvous, docking, capture, and other interaction scenarios between spacecraft. It is equipped with space-representative lighting, a motion-capture system, an epoxy floor, and the capability to integrate on-board computers and mount large mockups. It also features two Universal Robots (UR) UR10e collaborative robots mounted on rails, one on the ceiling and one on the wall, with strokes of 3.2 m and 4.3 m, respectively. The experiment room measures 5 m $\times$ 3 m $\times$ 2.3 m (W $\times$ L $\times$ H) and is fully painted with non-reflective black paint.

The University of Würzburg (Regoli et al., 2013) operates a proximity-operations testbed with two KUKA KR 16 manipulators (6-DoF, 16 kg payload, 1611 mm reach) mounted 2.5 m apart with overlapping workspaces, using Polaris Spectra

optical tracking for ground-truth pose.
The Space Research Center of the Polish Academy of Sciences operates an air-bearing micro-gravity testbed for free-floating satellite–manipulator experiments (Rybus et al., 2013). The testbed's centerpiece is a 1.22 m, two-DoF robotic arm (6 kg). Supported by three air bearings, the arm achieves quasi-frictionless planar translation and rotation across a 2 x 3 m granite table, enabling realistic on-orbit manipulation tests.
Politecnico di Milano also operates a small-scale robotic testbed (Lunghi et al., 2017), primarily used to validate vision-based navigation for planetary landing. The platform offers 7-DoF, a 1.03 m reach, and supports payloads up to 10 kg.

1.3 Thesis focus

The motivation and research directions discussed above lead to a set of research questions that form the foundation of this thesis:

1. How to make deep learning-based monocular pose estimation algorithms robust to domain gap?
2. Is it possible to build deep learning-based monocular pose estimation algorithms that are compatible with onboard power levels?
3. How to get reliable uncertainty estimates out of them?

To answer these questions, at least in part, the research has been developed along two main tracks:

1. the development and validation of novel algorithms for monocular pose estimation, and
2. the installation, integration, and commissioning of a new robotic testbed at the Microsatellite and Space Microsystems Laboratory of the University of Bologna.

1.4 Document structure

This document is structured as follows. Chapter 2 introduces preliminaries and useful concepts to support the reading of the work. Chapter 3 focuses

on pose estimation of known uncooperative spacecraft, addressing the domain gap between synthetic and real data, the constraints of embedded deployment, and uncertainty estimation. Chapter 4 addresses unknown targets, presenting methods for detection, feature matching, and relative camera pose estimation. Chapter 5 describes the development of a robotic hardware-in-the-loop facility, including its design, software integration, calibration procedures, and the pipeline for generating a novel dataset featuring both synthetic and real imagery. Finally, Chapter 6 summarizes the main contributions and discusses limitations and future research directions.

2

Preliminaries

In this chapter, the main concepts and tools relevant to this work are introduced. It starts with the definition of pose and reference frames, followed by the evaluation metrics adopted to assess algorithms performance. Then, the main deep learning architectures employed, Convolutional Neural Networks, Transformers, and Vision Transformers, are introduced.

2.1 Reference frames and relative poses

A coordinate system (or reference frame) is defined by an origin point and three orthogonal axes. Throughout this thesis, only right-handed reference frames are considered, i.e., the cross product of the x and y axes yields the z axis: $\hat{\mathbf{x}} \times \hat{\mathbf{y}} = \hat{\mathbf{z}}$. The relative pose between two reference frames A and B refers to the position and orientation of frame B relative to frame A. The position is represented by the translation vector $\mathbf{t}_{A \to B} \in \mathbb{R}^3$, while the orientation can be described by expressing the unit vectors of frame B's axes in terms of frame A's coordinates. Let $\hat{\mathbf{x}}_B$, $\hat{\mathbf{y}}_B$, and $\hat{\mathbf{z}}_B$ denote the unit vectors along the axes of frame B. When expressed in frame A, these become:

$$\mathbf{R}_{AB} = \begin{bmatrix} | & | & | \\ \hat{\mathbf{x}}_B^A & \hat{\mathbf{y}}_B^A & \hat{\mathbf{z}}_B^A \\ | & | & | \end{bmatrix} \tag{2.1}$$

This matrix $\mathbf{R}_{AB}$ is a rotation matrix and represents the relative orientation between the two frames. The 9 entries in matrix $\mathbf{R}_{AB}$ must satisfy 6 constraints (orthonormality: $\mathbf{R}^T\mathbf{R} = \mathbf{I}$), meaning that only 3 of them are independent. There are therefore parametrizations that make use of fewer than 9 parameters. One such representation is the unit quaternion.
A unit quaternion $\mathbf{q}$ is a 4-dimensional vector that provides a singularity-free representation of rotations:

$$\mathbf{q} = [q_w, q_x, q_y, q_z]^T = [q_w, \mathbf{q}_v^T]^T \tag{2.2}$$

where $q_w \in \mathbb{R}$ is the scalar part and $\mathbf{q}_v = [q_x, q_y, q_z]^T \in \mathbb{R}^3$ is the vector part, satisfying the unit norm constraint:

$$\|\mathbf{q}\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} = 1 \tag{2.3}$$

A quaternion encodes a rotation of angle θ around a unit axis $\hat{\mathbf{n}}$ as:

$$\mathbf{q} = [\cos(\theta/2), \sin(\theta/2) \cdot \hat{\mathbf{n}}^T]^T \tag{2.4}$$

The rotation matrix can be recovered from a unit quaternion through the following conversion:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{2.5}$$

Quaternions offer several computational advantages: they avoid gimbal lock and require only 4 parameters with one constraint compared to 9 parameters for rotation matrices.

2.1.1 Active and passive rotations

Rotation matrices belong to the Special Orthogonal group $SO(3)$:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} : \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\} \tag{2.6}$$

The orthonormality constraint ensures that the columns form an orthonormal basis, while the determinant condition enforces the right-handedness of the coordinate system.
It is crucial to distinguish between two interpretations of rotations.

- **Passive rotation** (coordinate transformation): involves changing the reference frame in which an object is described, while the object's position and orientation remain unchanged. Passive rotations answer the question: *given the coordinates of an object in a coordinate system, what are its coordinates in a rotated coordinate system?*
- **Active rotation**: refers to the rotation of the object itself within a fixed coordinate system. The reference frame remains the same, but the object's orientation changes. Active rotations answer the question: *given the coordinates of an object in a coordinate system, what are the coordinates of the rotated object in the same coordinate system?*

In this thesis, passive transformations are estimated, which describe the geometric relationships between different coordinate systems.

2.1.2 Notation and composition rules

Rotation matrices can be composed to perform chains of coordinate transformations. The subscript cancellation rule provides an intuitive method for this composition: if the second subscript of the first matrix matches the first subscript of the second matrix, they cancel out:

$$\mathbf{R}_{AC} = \mathbf{R}_{A\cancel{B}}\mathbf{R}_{\cancel{B}C} \tag{2.7}$$

This rule reflects the chain of transformations from frame C to frame A through frame B.
The same cancellation principle applies to position vectors. For example, the position of frame C relative to frame A can be obtained by:

$$\mathbf{t}_{A\to C|A} = \mathbf{t}_{A\to B|A} + \mathbf{R}_{A\cancel{B}}\mathbf{t}_{B\to C|\cancel{B}} \tag{2.8}$$

Throughout this thesis, the following notation is adopted:

- $\mathbf{R}_{ij}$ is the rotation matrix that transforms coordinates from frame j to frame i. The columns of this matrix represent the unit vectors of frame j's axes expressed in frame i.
- $\mathbf{t}_{i\to j|k}$ represents the position vector from frame i to frame j, expressed in frame k.

In general, a point $\mathbf{p}_j$ expressed in frame j can be transformed to frame i using both rotation and translation:

$$\mathbf{p}_i = \mathbf{R}_{ij}\mathbf{p}_j + \mathbf{t}_{i\to j|i} \tag{2.9}$$

2.1.3 Homogeneous transformations

To simplify the composition of rotations and translations into a single linear operation, homogeneous coordinates are employed. A 3D point $\mathbf{p} = [x, y, z]^T$ is represented in homogeneous coordinates as $\tilde{\mathbf{p}} = [x, y, z, 1]^T$. This allows the combination of rotation and translation into a single matrix multiplication.
The homogeneous transformation matrix $\mathbf{T}_{ij} \in SE(3)$ encodes both the rotation and translation from frame j to frame i:

$$\mathbf{T}_{ij} = \begin{bmatrix} \mathbf{R}_{ij} & \mathbf{t}_{i\to j|i} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{2.10}$$

where $\mathbf{R}_{ij} \in SO(3)$ is the rotation matrix and $\mathbf{t}_{i\to j|i} \in \mathbb{R}^3$ is the translation vector from frame i to frame j expressed in frame i.
Using homogeneous coordinates, the transformation of a point from frame j to frame i becomes:

$$\tilde{\mathbf{p}}_i = \mathbf{T}_{ij}\tilde{\mathbf{p}}_j \tag{2.11}$$

Homogeneous transformations follow the same subscript cancellation rule as rotation matrices:

$$\mathbf{T}_{AC} = \mathbf{T}_{A\cancel{B}}\mathbf{T}_{\cancel{B}C} \tag{2.12}$$

This property makes it straightforward to compose multiple transformations along a kinematic chain, which is particularly useful in robotic applications where multiple coordinate frames are involved.

2.1.4 Camera-chaser and frame-to-frame relative poses

This work makes frequent use of the OpenCV (Bradski, 2000) library and therefore aligns with the OpenCV pinhole camera model for both the image and camera frames (see Figure 2.1). In OpenCV convention, the camera frame has its z-axis pointing forward (into the scene), x-axis pointing right, and y-axis pointing down.

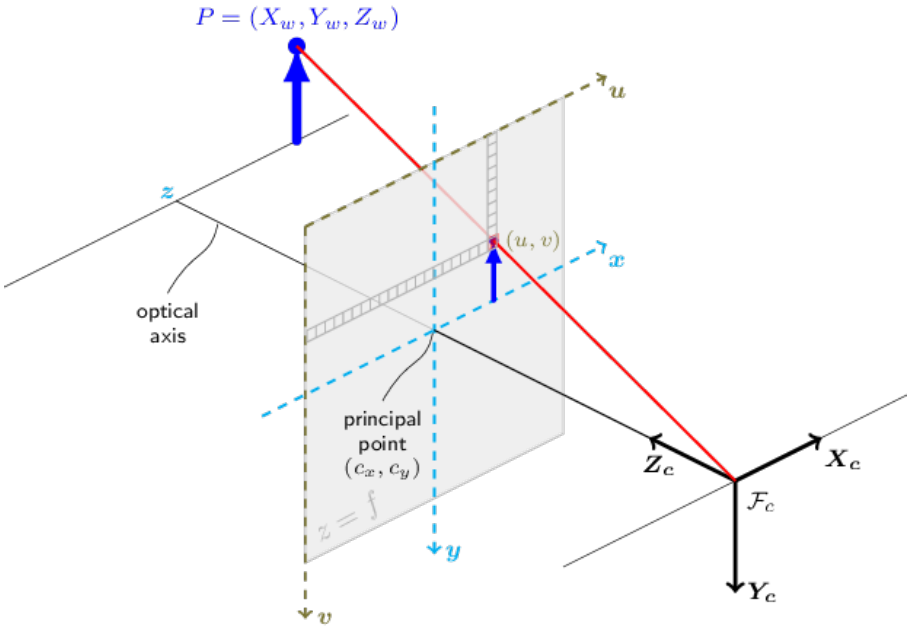

Figure 2.1: Pinhole camera model illustrating the camera frame (c), optical axis, and image plane (Bradski, 2000)

The camera-target (or chaser-target) pose is defined as the transformation $\mathbf{T}_{ct}$, which represents the orientation and position of the target's body frame (t) with respect to the camera frame (c):

$$\mathbf{T}_{ct} = \begin{bmatrix} \mathbf{R}_{ct} & \mathbf{t}_{c \to t|c} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.13}$$

The rotation component $\mathbf{R}_{ct}$ is the rotation matrix that aligns the target's body reference frame with the camera's reference frame. The translation component

$\mathbf{t}_{c\to t|c}$ is the relative position of the origin of the target's body reference frame with respect to the origin of the camera's reference frame, expressed in camera coordinates. This transformation maps points from the target's body frame to the camera frame:

$$\tilde{\mathbf{p}}_c = \mathbf{T}_{ct}\tilde{\mathbf{p}}_t \tag{2.14}$$

The projection onto the image plane then follows the pinhole camera model:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{R}_{ct} & \mathbf{t}_{c\to t|c} \end{bmatrix}\begin{bmatrix} \mathbf{p}_t \\ 1 \end{bmatrix} \tag{2.15}$$

where the camera intrinsic matrix $\mathbf{K}$ is defined as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

with f_x, f_y being the focal lengths in pixels, (c_x, c_y) the principal point coordinates, s a scale factor, and $\tilde{\mathbf{p}}_t = [\mathbf{p}_t^T, 1]^T$ the homogeneous coordinates of point $\mathbf{p}_t$. This notation and convention for the camera-target pose will be consistently used throughout Chapter 3.

Within the framework of the PoseBowl Pose Estimation Track challenge, the organizers provided sequences of images where the task is to estimate the transformation aligning each frame to a common reference. Specifically, for any frame i in the sequence, the required transformation aligns it to the first frame (frame 0), which is taken as the reference.

The challenge requires estimating two components:

- The rotation matrix $\mathbf{R}_{0i}$ that aligns frame i with the reference frame 0
- The translation vector $\mathbf{t}_{i\to 0|0}$, which represents the displacement from frame i to frame 0, expressed in the coordinates of frame 0. This vector indicates how the camera should be moved to return from its position at frame i to its position at the reference frame.

The frame-to-frame pose is therefore represented as the tuple $[\mathbf{R}_{0i}|\mathbf{t}_{i\to 0|0}]$. This notation and convention will be used in Chapter 4.

2.2 Recurrent evaluation metrics

Throughout this work, the following notation conventions are adopted:

- Subscript $_{\text{est}}$ denotes estimated values
- Subscript $_{\text{GT}}$ denotes ground truth values
- $\langle\cdot\rangle$ denotes average (mean) values
- $\langle\langle\cdot\rangle\rangle$ denotes median values
- $\pm$ precedes the standard deviation (also denoted as σ) unless otherwise specified

The translation error can be evaluated using two metrics. The absolute distance measures the Euclidean norm between estimated and ground truth positions:

$$e_t = \|\mathbf{t}_{\text{GT}} - \mathbf{t}_{\text{est}}\|_2 \tag{2.17}$$

However, since the significance of translation errors depends on the chaser-target distance, a normalized metric is also employed:

$$\bar{e}_t = \frac{e_t}{\|\mathbf{t}_{\text{GT}}\|} \tag{2.18}$$

This normalization accounts for the fact that a given translation error is more critical at short distances than at large distances.
The rotation error is measured using the geodesic distance, which represents the minimum rotation angle required to align the estimated orientation with the ground truth. For quaternion representations, this is computed as:

$$e_q = 2\arccos(|\mathbf{q}_{\text{GT}}^T \cdot \mathbf{q}_{\text{est}}|) \tag{2.19}$$

To facilitate comparison between different pose estimation methods, a combined score metric is defined by summing the two dimensionless error components:

$$E = \bar{e}_t + e_q \tag{2.20}$$

Detection models are evaluated in terms of Intersection over Union (IoU), which measures the overlap between estimated and ground truth bounding boxes:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} = \frac{|B_{\text{est}} \cap B_{\text{GT}}|}{|B_{\text{est}} \cup B_{\text{GT}}|} \tag{2.21}$$

where B_{est} and B_{GT} represent the estimated and ground truth bounding boxes respectively. The IoU metric ranges from 0 to 1, where 0 indicates no overlap and 1 indicates perfect alignment.

2.3 Deep learning architectures for image processing

Deep learning has revolutionized computer vision through computational models composed of multiple processing layers that learn representations of data with multiple levels of abstraction (LeCun et al., 2015). At its core lie neural networks, highly non-linear functions composed of nodes (or neurons) organized into layers that learn to map inputs to desired outputs. This learning occurs through a training process, typically supervised, where the network is exposed to labeled examples. During training, the network minimizes a loss function through backpropagation, an algorithm that computes gradients of the error with respect to each parameter and propagates these gradients backward through the layers to update the network's weights.

At the heart of many vision models lies the backbone which consists in the primary feature extraction network that processes raw input data into meaningful representations through these multiple levels of abstraction. Convolutional Neural Networks have emerged as the dominant backbone architecture for processing image data, utilizing convolutions, mathematical operations that slide learnable filters across input data to detect patterns. CNNs became the state-of-the-art approach with AlexNet (Krizhevsky et al., 2012), a deep convolutional network that won the ImageNet competition by a large margin, showing that these architectures, when combined with GPUs and large-scale datasets, could achieve unprecedented performance in visual recognition tasks. Unlike traditional fully connected networks where each neuron connects to all neurons in the previous layer, CNNs employ sparse connectivity (or local connectivity), where each neuron connects only to a small region of the input. This architectural choice significantly reduces the number of parameters (and computational cost) and captures the

spatial locality inherent in image data. These convolutions produce feature maps, which are the output representations at each layer encoding detected patterns with increasing abstraction: early layers typically capture low-level features like edges and textures, while deeper layers progressively learn more complex and task-specific features. Standard convolutions simultaneously capture spatial and channel-wise relationships across these feature maps, while depthwise separable convolutions decompose this operation into two steps: spatial filtering within each channel (depthwise) followed by cross-channel mixing (pointwise), further reducing computational cost. Transpose convolutions (also called deconvolutions) perform the inverse operation, upsampling feature maps by learning to expand spatial dimensions.

The advent of Transformers marked a paradigm shift in deep learning, initially revolutionizing natural language processing with the seminal "Attention is All You Need" paper (Vaswani et al., 2017). Transformers replaced recurrent architectures, which process sequences step by step, maintaining a hidden state that accumulates information over time, with an attention mechanism that enables direct global information exchange between all elements in a sequence. In NLP, these elements are typically words. The attention mechanism works by computing a weighted average of all these input elements, where the weights determine how much each element should "attend to" every other element in the sequence. For each element, the model generates queries (what information to look for), keys (what information is available), and values (the actual information content). Attention scores are computed by matching queries with keys, determining which elements are most relevant when computing the representation of a given position. Multiple attention heads operate in parallel, each learning different relationship patterns. The hidden dimension defines the internal representation size within the transformer blocks, typically larger than the input/output dimensions to provide sufficient expressiveness.

The success of Transformers in natural language processing inspired their adaptation to computer vision. Vision Transformers (ViTs) (Dosovitskiy et al., 2020) demonstrated that a pure transformer architecture could match or exceed CNN performance on image classification when trained on sufficient data. ViTs adapt the transformer architecture to visual tasks by treating images as sequences of patches (tokens), essentially converting the 2D spatial structure into a 1D sequence. Unlike CNNs' local receptive fields enforced by convolutional operations,

ViTs can model global dependencies from the first layer.
The ViT architecture consists of stacked transformer blocks, each containing two main components with residual connections. First, the self-attention layer enables each patch to exchange information with all other patches in the image, allowing the model to capture spatial relationships and global context. Second, the feedforward network or Multi-Layer Perceptron (MLP) processes each patch independently, transforming its feature representation through multiple layers. This design separates spatial reasoning (how patches relate to each other) from feature extraction (what each patch represents), allowing ViTs to learn both global image structure and local feature details.

2.4 Implementation framework

All neural networks developed in this thesis were trained on GPUs or on Cloud TPUs, which are Google's specialized hardware accelerators for deep learning workloads. As training is computationally expensive, it is essential to ensure that the input pipeline does not become a bottleneck. For instance, the SPEED+ dataset alone is about 17 GB, and inefficient data loading could lead to under-utilization of computational resources, increasing both training time and cost.
To address this, all datasets were converted into the TFRecords format, TensorFlow's native serialized binary representation. This enables fast sequential access and avoids the overhead of opening thousands of separate image and annotation files during training. For large datasets, multiple TFRecord files were generated. Splitting into smaller files allows TensorFlow's `tf.data` API to load data in parallel, significantly increasing throughput on TPUs and GPUs by overlapping dataset processing with model training.
During training, data loading was handled via the `tf.data` API, which supports parallel reading, caching, and prefetching. Unless otherwise specified, data augmentations such as geometric transformations, noise injection, and photometric adjustments were applied online through the `map` transformation, ensuring that augmented samples were generated dynamically without additional storage requirements. The resulting TFRecord files were then uploaded to a Google Cloud Storage bucket, providing the native filesystem interface required by TPUs and enabling direct high-bandwidth access during training.

Throughout this thesis, PnP problems were solved using OpenCV's `solvePnPRansac` function, which employs RANSAC to automatically discard potential outliers, with the method flag adjusted as needed and optionally followed by Levenberg–Marquardt (LM) refinement via `solvePnPRefineLM`. Unless otherwise stated, the default OpenCV parameters were adopted: 100 RANSAC iterations, a reprojection error threshold of 8.0 pixels, a confidence level of 99%, and termination criteria for the LM refinement set to a maximum of 20 iterations.

3

Pose estimation of known uncooperative targets

This chapter summarizes the most relevant satellite pose estimation pipelines and experiments that have been developed throughout the PhD to answer research questions 1-3. The material presented in this chapter previously appeared in 4 publications, specifically:

- Section 3.1 appeared in "Investigating Vision Transformers to Bridge Domain Gap in Satellite Pose Estimation" (Lotti et al., 2023a) and in "Improving Satellite Pose Estimation Across Domain Gap with Generative Adversarial Networks" (Lotti, 2023).
- Section 3.2 appeared in "Deep Learning for Real-Time Satellite Pose Estimation on Tensor Processing Units" (Lotti et al., 2023b).
- Section 3.3 appeared in "Exploring Stochastic Variational Gaussian Processes for Trustworthy Monocular Pose Estimation in Space" (Lotti et al., 2025b).

In the first part, the domain gap problem is addressed by comparing different solutions, starting with a domain adversarial approach that was proposed at SPEC

2021, and moving towards the investigation of methods that proved effective at bridging the domain gap without access to real images. The second part illustrates a parametric study to assess the accuracy-latency trade-off achievable on an embedded device using synthetic images. Finally, the last section introduces a method to get uncertainty estimates out of the pose estimation algorithm by exploiting Stochastic Variational Gaussian Processes (SVGP).

3.1 Domain gap in satellite pose estimation

Domain generalization refers to the ability of a neural network to perform well on data from domains that were not observed during training. It is an active research area in computer vision, driven by the need for models to remain reliable under changing acquisition conditions, sensor characteristics, or environmental factors. In satellite pose estimation, domain gap refers to the accuracy loss when neural networks trained on synthetic images are tested on real ones. At a high level, domain gap can be addressed by acting on the model architecture, on the training strategy, on the data, or on a combination of these elements. Methods belonging to all these categories are investigated in this thesis.

3.1.1 A domain adversarial approach with Vision Transformers

The first algorithm was developed during the SPEC 2021 competition and adopts a classical three-stage architecture (see Figure 3.1). The pipeline consists of a Satellite Detection Network (SDN), a Landmark Regression Network (LRN), and a PnP solver and tackles domain gap simultaneously from the model and training strategies points of view. In the first stage, the SDN localizes the target in the image, addressing the information loss that would result from direct image resizing. During inference, the predicted bounding box is enlarged by 15%. The second stage employs the LRN to regress landmark coordinates within this Region of Interest (ROI). Finally, the PnP solver computes the camera-target pose from the correspondences between detected landmarks and their 3D model coordinates. Specifically, the pipeline exploits EPnP (Lepetit et al., 2009) with RANSAC, using a maximum reprojection error threshold of 5 pixels, followed by Levenberg-Marquardt refinement. Since the 3D model of the Tango satellite was not available at the time of the competition, it was reconstructed using

MATLAB's `triangulateMultiview` with 12 images from the training dataset and their associated pose labels. This reconstruction served both for generating landmark projections used for NN training and for providing the 3D reference points required by the PnP solver in the final stage.

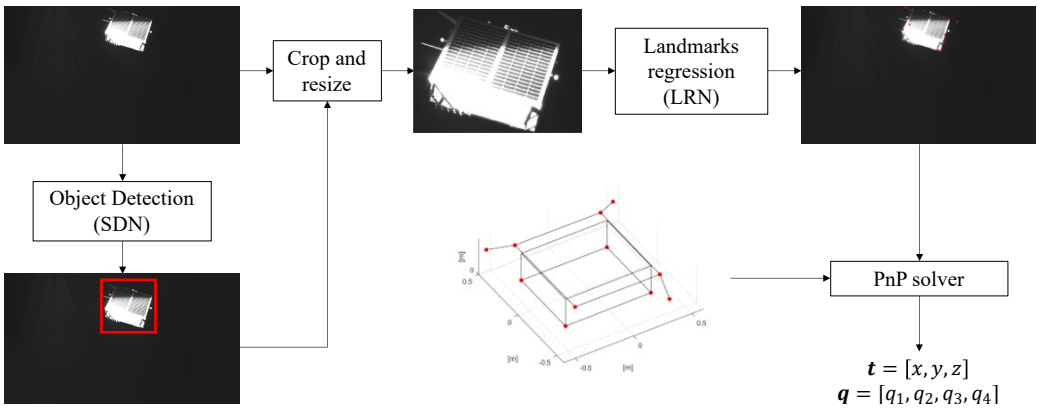

Figure 3.1: Three-stage pose estimation pipeline comprising satellite detection, landmark regression, and PnP-based pose estimation (Lotti et al., 2023a)

Since the competition focused solely on pose regression accuracy without imposing computational constraints, Vision Transformers were selected as the backbone architecture. This choice was motivated by their generalization capabilities, as demonstrated in Zhang et al. (2022). Among the many variants of the original implementation by Dosovitskiy et al. (2020), the Swin Transformer, proposed by Liu et al. (2021), was selected because it was designed as a general-purpose backbone for computer vision tasks. Its hierarchical structure resembles that of standard CNNs while incorporating the distinctive features of Transformers, and its computational cost scales linearly with input size, in contrast to the quadratic scaling of the original ViT. Swin Transformers features 4 hierarchical stages and come in different variants from "Tiny" to "Large", the main differences being the hidden dimension of the transformer blocks, the number of transformer blocks in each stage, and the number of attention heads, resulting in models ranging from 28M parameters (Tiny) to 197M parameters (Large) with corresponding trade-offs between computational efficiency and model capacity. The difference between backbone variants are summarized in Table 3.1.
Similarly to other top-performing methods in SPEC 2021, developed in the same timeframe, a pipeline based on domain adversarial training (Ganin et al., 2017) was developed. This technique employs a multi-headed network where one

Table 3.1: Swin Transformers variants

Model variant	Hidden dimension per stage	Heads per stage	Layers per stage
Tiny	96, 192, 384, 768	[3, 6, 12, 24]	[2, 2, 6, 2]
Small	96, 192, 384, 768	[3, 6, 12, 24]	[2, 2, 18, 2]
Base	128, 256, 512, 1024	[4, 8, 16, 32]	[2, 2, 18, 2]
Large	192, 384, 768, 1536	[6, 12, 24, 48]	[2, 2, 18, 2]

head acts as a domain discriminator. During training, the network processes both synthetic and real images, while the discriminator attempts to classify their domain of origin. The adversarial mechanism operates by reversing gradients before backpropagation to the shared encoder, forcing the encoder to maximize the discriminator's loss. Through this process, as the discriminator improves at domain classification, the encoder learns to extract domain-invariant features, enhancing cross-domain performance on the primary task.

Although requiring access to real images during training represents a practical limitation, this approach remains viable for real-world applications. Testing facilities such as those described in Section 1.2.3 and the facility illustrated in Chapter 5 serve a dual purpose: they enable the evaluation of algorithms under domain gap conditions and provide realistic training data that can be generated prior to actual space missions. Pre-deployment data generation allows domain adversarial methods to be effectively employed despite their dependency on real images. Both the SDN and the LRN employ an identical architecture within this domain adversarial framework (see Figure 3.2). Each network consists of a Swin Transformer backbone encoder (E) followed by global average pooling, which reduces the final feature map to a feature vector. This feature vector then branches into two heads: a discriminator head (D) for domain classification and a regressor head (R) for the primary task. The regressor head comprises three fully connected layers with GeLU activations (Hendrycks and Gimpel, 2023). Since all dataset images contain a single target, the detection task is simplified to localization. Therefore the SDN's regressor directly outputs the bounding box vertices, while the LRN's regressor produces a vector of normalized x-y coordinates corresponding to the landmarks shown in Figure 3.1 (highlighted in red), in a predefined sequence.

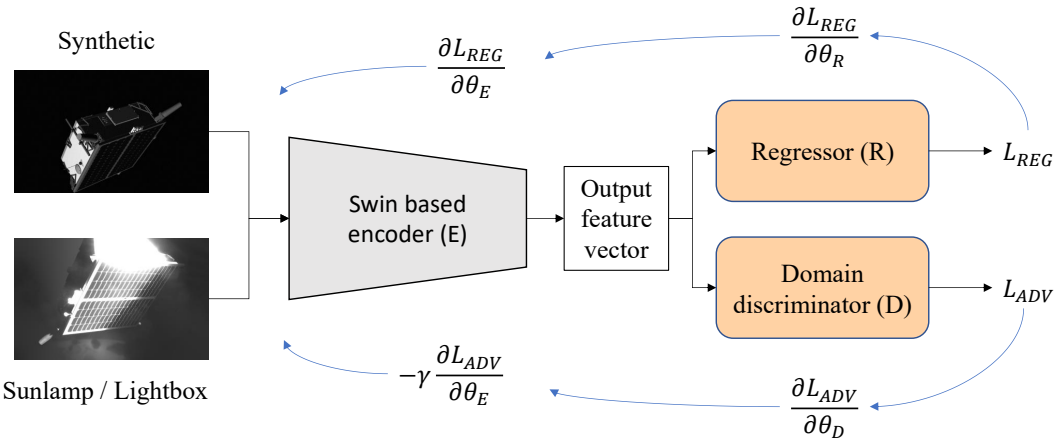

Figure 3.2: Domain adversarial training architecture adopted for SPEC 2021; a Swin-based encoder feeds both a regressor and a domain discriminator, with reversed gradients ($-\gamma$) forcing domain-invariant feature extraction (Lotti et al., 2023a)

The domain discrimination and regression losses are formulated as:

$$L_{ADV} = -\frac{1}{N_B}\sum_{i=1}^{N_B}\left[y_{D_i,\text{GT}}\log(y_{D_i,\text{est}}) + (1-y_{D_i,\text{GT}})\log(1-y_{D_i,\text{est}})\right] \tag{3.1}$$

$$L_{REG} = \frac{1}{N_S}\sum_{i=1}^{N_S}\|\mathbf{r}_{i,\text{est}} - \mathbf{r}_{i,\text{GT}}\|_1 \tag{3.2}$$

where L_{ADV} is the binary cross-entropy loss between estimated and ground truth domain labels y_D (with $y_D = 0$ for synthetic and $y_D = 1$ for real, treating both sunlamp and lightbox as real), $\mathbf{r}$ represents the regressed vector containing either bounding box vertices or landmark coordinates, and N_S and N_B denote the number of synthetic images in the batch and the total batch size, respectively. The adversarial training process encourages domain-invariant feature learning through competing objectives. Denoting by θ_D, θ_R, and θ_E the parameters of the discriminator, regressor, and encoder respectively, their optimization objectives are:

$$\hat{\theta}_D = \arg\min_{\theta_D}(L_{ADV}) \tag{3.3}$$

$$\hat{\theta}_R = \arg\min_{\theta_R}(L_{REG}) \tag{3.4}$$

$$\hat{\theta}_E = \arg\min_{\theta_E}(L_{REG} - \gamma L_{ADV}) \tag{3.5}$$

Since the discriminator minimizes L_{ADV} while the encoder minimizes $-\gamma L_{ADV}$, they have opposing objectives: the discriminator improves by distinguishing domains, while the encoder improves by making this distinction impossible. This forces the encoder to suppress domain-specific features and retain only those that are common across domains (Ganin et al., 2017).
The parameter γ in eq. (3.5) dynamically re-weights the contribution of the discrimination and regression losses during training. Since both the classification and the regressor heads are initially untrained, introducing the full adversarial signal from the beginning would destabilize training. Therefore, γ is gradually increased from epoch 1 to epoch 40 according to eq. (3.6), allowing the model to first learn meaningful representations before progressively incorporating domain-invariant constraints through the adversarial objective. Figure 3.3 reports how γ progresses over time.

$$\gamma = \frac{1}{50}\left(\frac{1}{1 - e^{\frac{10\,epoch}{total_epochs}}}\right) - 0.01 \tag{3.6}$$

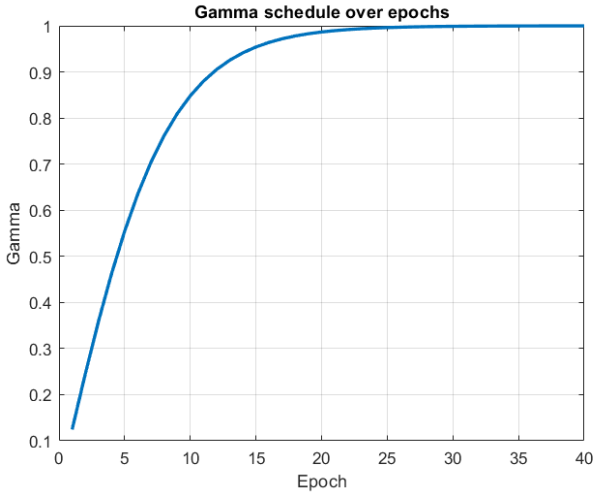

Figure 3.3: Schedule of the adversarial loss weighting parameter γ, gradually increasing from epoch 1 to 40 to stabilize early training

To address class imbalance arising from the different sizes of synthetic and real partitions, real images are repeated within each epoch to ensure batches maintain a 50% synthetic / 50% real distribution.
A key aspect of this training strategy is that regression loss is computed only

on synthetic images, whose pose labels are available, while discriminator loss utilizes all images in the batch. This distinction becomes particularly critical when training on Cloud TPUs, as the asymmetric loss computation must be properly handled in the distributed training implementation across TPU cores. TPUs consist of 8 cores, each processing 1/8 of the global batch during training. Each core performs a complete training step (forward and backward pass), after which gradients are synchronized across all replicas before weight updates. This sharding strategy requires careful data distribution: each TPU core must maintain the 50% synthetic / 50% real ratio within its local sub-batch, not just globally across the full batch, ensuring balanced gradient contributions from all cores. This balance is achieved through a specific data preparation pipeline: synthetic and real datasets are first organized into separate mini-batches of equal size, then zipped and concatenated to create balanced sub-batches containing exactly 50% synthetic and 50% real images. These balanced sub-batches are subsequently combined to form the global batch. Since this balancing occurs before TPU sharding, each core automatically receives a sub-batch that preserves the desired ratio, ensuring class balance at the TPU core replica level. A visualization of this concept is provided in Figure 3.4.

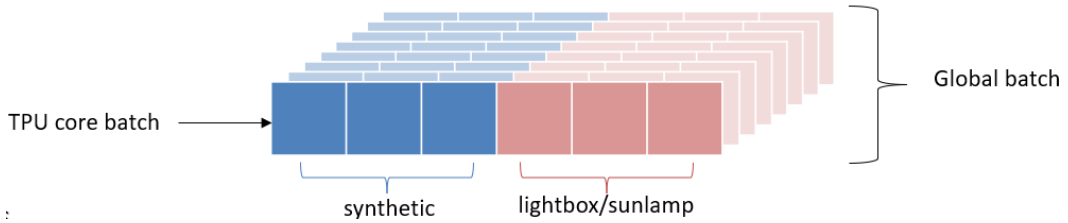

Figure 3.4: TPU core batching strategy ensuring 50% synthetic and 50% real images in each core's sub-batch for balanced gradient contributions during adversarial training

All models were trained for 40 epochs with a global batch size of 48, using AdamW (Loshchilov and Hutter, 2018) optimizer with cosine decay scheduling (initial learning rate: 5e-5). Both networks process squared inputs through different preprocessing pipelines. The SDN directly resizes full images to square format. The LRN, instead, operates on squared crops extracted from bounding boxes. Ground truth boxes are used for synthetic images while, since real images lack pose annotations, SDN-predicted boxes are employed for cropping. This approach ensures that real and synthetic images receive identical preprocessing despite the absence of ground truth annotations for real data, maintaining consistent

zoom levels and augmentations across domains for the LRN. These crops undergo augmentation with random scaling (up to 15%) and spatial shifts. Different backbone sizes, resolutions, and augmentations were tested while keeping the overall pose estimation pipeline and training algorithms fixed. The experiments employed two Swin Transformer variants: "Tiny" (pretrained on ImageNet) and "Base" (pretrained on ImageNet-22k). Tables 3.2 and 3.3 detail the architectures used in the main SPEC 2021 submissions and their achieved pose errors.

Table 3.2: Architectural details of the main SPEC 2021 submissions (Lotti et al., 2023a)

Sub-mission	**SDN**			**LRN**		
	Swin model	**Resolution [px]**	**# Parameters (M)**	**Swin model**	**Resolution [px]**	**# Parameters (M)**
A	Base	384	97.9	Tiny	224	37.0
B	Base	384	97.9	Base	384	97.9
C	Base	448	97.9	Base	448	97.9
D	Base	448	97.9	Base	448	97.9

Solutions A and B share the same SDN built on top of a Swin Base model at 384px resolution. Random brightness and contrast augmentations were applied to all images during both SDN and LRN training. Increasing the LRN backbone size and resolution from 224px (A) to 384px (B) reduced pose estimation error by 33% on lightbox and 48% on sunlamp domains, confirming that larger Transformers achieve better generalization.

Table 3.3: Results of the main SPEC 2021 submissions (Lotti et al., 2023a)

Submission	**Lightbox**			**Sunlamp**		
	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$
A	0.111	27.4	0.590	0.056	15.36	0.323
B	0.096	17.4	0.400	0.046	7.28	0.173
C	0.077	14.2	0.325	0.044	7.91	0.181
D	0.055	9.68	0.224	0.032	6.25	0.141

Resolution was further increased to 448px in variants C and D where the benefits of stronger photometric augmentations on synthetic images have been explored, with details reported in Table 3.4. In addition to these augmentations, artificial

sunflare was added offline to synthetic images (see Figure 3.5) to better simulate the sunlamp domain characteristics. Interestingly, despite targeting sunlamp appearance, the largest performance improvement was observed on the lightbox domain.

Table 3.4: Data augmentation strategy applied during training of solutions C and D

Augmentation	**Probability**	**Parameters**	**Description**
Rotation	100%	angle $\in [0°, 360°]$	Applies random rotation between $0°$ and $360°$
Random brightness	50%	Max delta = 50	Adjusts image brightness by $\pm 20\%$ of pixel range
Random contrast	50%	Lower = 0.1, Upper = 1.5	Modifies contrast with factor between 0.1x and 1.5x
Gaussian Blur	50%	$\sigma = 1.0$	Applies Gaussian blur
Gaussian Noise	50%	$\mu = 0$, $\sigma = 0.047$	Adds Gaussian noise with zero mean and standard deviation of 0.047

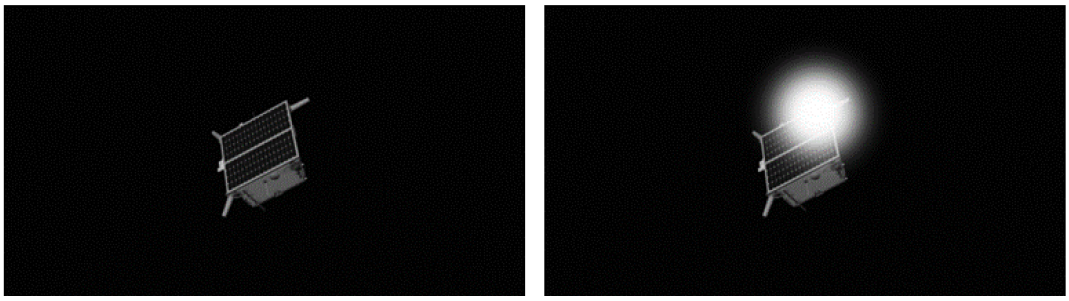

Figure 3.5: Synthetic light artifacts (sunflare augmentation) applied during training to improve robustness to illumination conditions (Lotti et al., 2023a)

The final SPEC 2021 leaderboards are reported in Tables 3.5 and 3.6, where submission D achieved 4th and 5th place in the sunlamp and lightbox rankings, despite significantly less submissions with respect to competitors.

Table 3.5: SPEC 2021 final Lightbox leaderboard (partial) (Park et al., 2023)

Rank	Team name	# Submissions	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$
1	TangoUnchained	35	0.0179	3.19	0.0735
2	VPU	47	0.0215	4.58	0.1014
3	Lava1302	47	0.0483	6.66	0.1646
4	haoranhuang_njust	68	0.0315	8.13	0.1734
5	**U3s_lab**	**13**	**0.0548**	**9.69**	**0.2240**

Table 3.6: SPEC 2021 final Sunlamp leaderboard (partial) (Park et al., 2023)

Rank	Team name	# Submissions	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$
1	Lava1302	47	0.0113	2.73	0.0589
2	VPU	47	0.0118	2.82	0.0611
3	TangoUnchained	35	0.0150	4.30	0.0900
4	**U3s_lab**	**13**	**0.0320**	**6.24**	**0.1409**
5	bbnc	68	0.0284	8.41	0.1751

3.1.2 Comparison with CNNs and embedded deployment constraints

While Vision Transformers demonstrated strong performance in the SPEC 2021 competition, it is important to evaluate how they compare against traditional CNN architectures. To provide this comparison, a parallel pipeline was developed using an EfficientNet-B5(Tan and Le, 2019), which has a comparable parameter count to Swin-Tiny. The algorithm pipeline and training setup remained identical to those described in Section 3.1.1, with one modification: a 1×1 convolution was added to the EfficientNet model to reduce channel dimensions to 768, matching the Swin-Tiny architecture. Both networks used 224px input resolution for this experiment. Table 3.7 presents this comparison, revealing the Vision Transformer's superior cross-domain generalization with pose estimation errors approximately 26% lower than its CNN-based counterpart.

Table 3.7: Pose estimation errors across real domains for CNN-based (EfficientNet-B5) and Transformer-based (Swin-Tiny) architectures of similar capacity (Lotti et al., 2023a)

SDN backbone	**LRN backbone**	$\langle E_{\text{sunlamp}} \rangle$	$\langle E_{\text{lightbox}} \rangle$	**# Parameters (SDN+LRN)**
Swin-Tiny	Swin-Tiny	0.258	0.373	74
EfficientNet-B5	EfficientNet-B5	0.442	0.406	79

These pipelines were deployed on a Coral Dev Board Mini, a single-board computer featuring a 1.5 GHz quad-core CPU with peak power consumption of approximately 3W, to assess their compatibility with space-graded hardware. The NNs were compiled with TensorFlow Lite (TFLite), which provides optimized implementations of neural network operations for ARM-based CPUs. The CNN pipeline achieved 0.22 fps, while the Transformer-based pipeline reached 0.15 fps, translating to about one image processed every 5 seconds. Such a latency can be inadequate for safety-critical close-proximity navigation, which requires continuous measurements. Furthermore, these domain adversarial models still depend entirely on real images during training, limiting their applicability when real images are unavailable before deployment.

3.1.3 A Hybrid CNN-Transformer architecture for efficient inference

To address these limitations, a hybrid architecture was developed combining CNN and Transformer components (see Figure 3.6). The backbone employs a CNN for initial feature extraction, followed by a compact Transformer module. The rationale for this design is twofold: CNNs efficiently process high-resolution inputs while being well-supported on mobile accelerators, whereas Transformer operations suffer from poor optimization on such devices. This architecture therefore allocates most computational burden to the CNN component while limiting the Transformer to feature correlation.

The CNN adopted for this experiment is an EfficientNet-Lite4, a mobile-optimized variant that differs from the original EfficientNet through: (1) removal of squeeze-and-excite blocks, and (2) replacement of Swish activations with ReLU6. The CNN reduces spatial resolution from 320×512px, in input, to a 10×16px feature map, which is then fed into the Transformer. The latter consists of a ViT-Tiny model

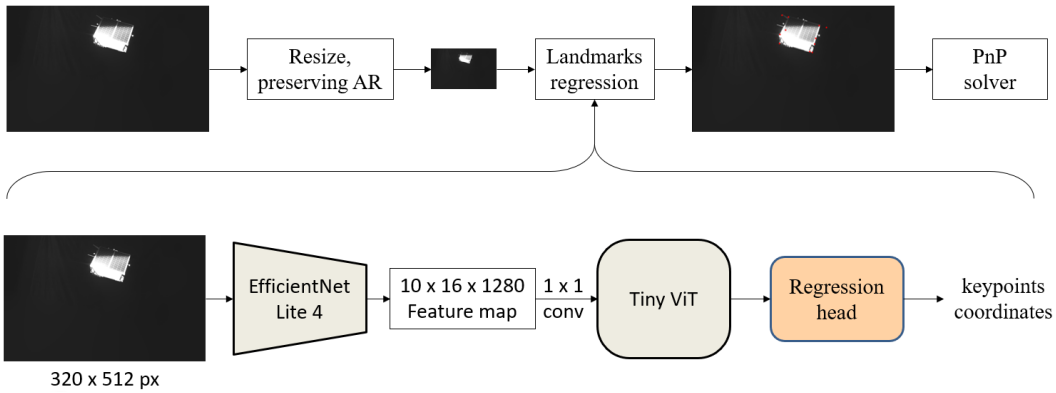

Figure 3.6: Hybrid CNN-ViT landmark regression pipeline; EfficientNet Lite 4 extracts spatial features, followed by a Tiny ViT for feature correlation and a regression head for keypoint prediction

(Zhang et al., 2022) featuring a single-layer encoder and 3 attention heads. This lightweight configuration is further justified by the fact that the CNN has already extracted high-level features and reduced spatial dimensions, allowing the Transformer to correlate all input patches without imposing significant computational overhead.

To further reduce computational cost, the detection stage was eliminated in favour of direct image resizing preserving the image aspect ratio. The model was trained for 80 epochs with a batch size of 64 using the AdamW optimizer. Weight decay was set to 1e-8, with learning rate following cosine decay scheduling from 1e-4. To address the second limitation, the dependence on real images during training, strong data augmentations were exploited to enhance domain generalization. The purpose is to expose the NN to diverse visual appearances, promoting the learning of domain-invariant features based on satellite geometry rather than textures. The training dataset was divided into three equal groups. Two groups were processed with different augmentation strategies: sunflare addition and style randomization (Jackson et al., 2019), while the third remained unaugmented to preserve original synthetic characteristics. Additionally, each input image undergoes random rotation followed by a comprehensive augmentation pipeline. This pipeline applies one strong transformation selected from equalize, invert, posterize, or solarize (Figure 3.7), combined with baseline transformations including random brightness, contrast, blur, and Gaussian noise, each applied with 50% probability. Training batches are built by sampling equally from all three augmentation groups

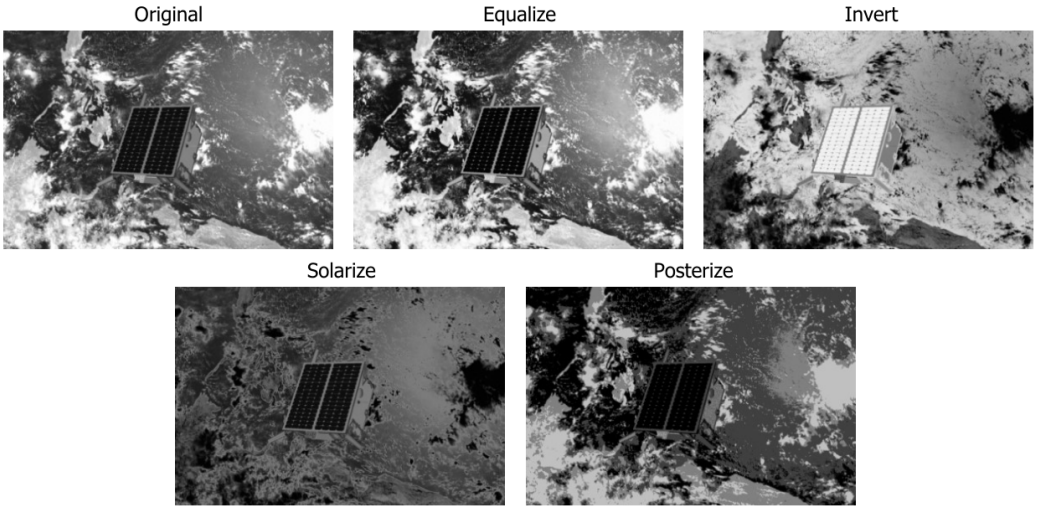

Figure 3.7: Strong photometric transformations used for domain generalization; histogram equalization, color inversion, solarization, and posterization applied to a synthetic training sample (Lotti et al., 2023a)

to ensure balanced exposure across synthetic variants. Table 3.8 demonstrates the cumulative impact of augmentation strategies on pose estimation accuracy. The results reveal significant improvements with each augmentation layer. Baseline transformations alone struggle with domain gap, particularly on sunlamp images ($\langle E \rangle = 1.245$). Adding sunflare and style randomization reduces errors by 19% on lightbox and 55% on sunlamp, suggesting these augmentations effectively capture domain-specific characteristics. The final layer of strong transformations (equalize, invert, posterize, solarize) provides additional gains, achieving additional error reductions of another 23% for lightbox and 30% for sunlamp. This progressive enhancement validates that comprehensive augmentation can substantially bridge the synthetic-to-real gap without requiring real images during training. The hybrid architecture achieved 2.5× faster inference on the Coral Board Mini's CPU (0.56 fps versus 0.22 fps), with an equivalent average pose error relative to the three-stage CNN pipeline from Section 3.1.2.

A final experiment was carried out to assess the validity of the hybrid architecture. Maintaining the dual-stage scheme of Figure 3.6, the hybrid CNN-Transformer was replaced with a Swin-Tiny model, followed by a 1×1 convolution to reduce feature channels to 192, matching the ViT-Tiny configuration for fair comparison.

Table 3.8: Pose estimation errors achieved by the hybrid CNN–Transformer architecture with progressive augmentations (Lotti et al., 2023a)

Augmentations applied	**Lightbox**			**Sunlamp**		
	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$
Baseline transformations	0.184	28.2	0.676	0.328	52.4	1.245
Addition of sunflare and style randomization (offline)	0.131	23.8	0.547	0.108	26.2	0.565
Addition of equalize, invert, posterize, solarize	0.098	18.6	0.422	0.077	18.1	0.393

Table 3.9 presents the results, showing that the hybrid architecture achieves an average pose error 37% lower than the Transformer-only pipeline, despite having less than half the parameters. This performance gap can be attributed, at least partially, to the absence of adversarial training, which exposes the data-hungry nature of Transformers and limits their effectiveness on small datasets.

Table 3.9: Pose estimation errors of the hybrid CNN–Transformer architecture compared with the Transformer-only model (Lotti et al., 2023a)

Model	**Lightbox**			**Sunlamp**			**# Params**
	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$	$\langle \bar{e}_t \rangle$	$\langle e_q \rangle$ [deg]	$\langle E \rangle$	
Swin-Tiny + Dense head	0.185	30.7	0.721	0.117	26.9	0.587	27.7 M
EfficientNet-Lite4 + ViT-Tiny	0.098	18.6	0.422	0.077	18.1	0.393	12.5 M

3.1.4 Narrowing domain gap at the data level exploring Generative Adversarial Networks

The methods illustrated in the previous sections primarily focused on bridging domain gap from the architecture and training strategy perspectives. This section explores data-level domain adaptation using generative models. Specifically, Generative Adversarial Networks (GANs) are adopted to transform synthetic images into photorealistic representations that better approximate real-world visual characteristics. A preliminary disclaimer regards the applicability of this method in real contexts. Generative models require access to images of the target dataset. However, the number of images required, depending on the model, can be a fraction of the synthetic images one could create with rendering software, provided that the input images are sufficiently representative of the real-world characteristics that rendering software fails to simulate. Therefore, these methods become extremely valuable when used in combination with small-scale hardware-in-the-loop facilities, such as the one presented in Chapter 5. The physical limits of the facility allow only a partial exploration of the pose space. On the other hand, the pose space has no limitations in rendering software. Therefore, one could use a generative model to virtually increase the real dataset size, potentially overcoming the physical limits of the facility. GANs consist of two competing neural networks, a generator, that creates enhanced synthetic images, and a discriminator, that distinguishes between real and generated samples. Through adversarial training, the generator learns to map synthetic images to the target domain's visual distribution, producing images that become progressively indistinguishable from actual mockup pictures. The objective is to preserve the geometric accuracy of synthetic data while acquiring the photometric properties, artifacts, and aberrations present in real images. Generative adversarial networks for image translation fall into two categories based on their training requirements. Paired methods necessitate pixel-aligned image pairs between domains, where each source image has an exact corresponding target image. Unpaired methods, instead, learn mappings between domains using unaligned datasets without explicit correspondences. For this experiment, an unpaired approach was necessary for two reasons: first, while SPEED+ synthetic and real images share similar pose distributions, they do not contain exactly matching poses, making paired training impossible; second, only unpaired methods are applicable to the HIL facility sce-

nario described above, as the purpose is to re-create poses that are not physically reachable by the facility. For these reasons, the Contrastive Unpaired Translation (CUT) method presented in (Park et al., 2020) was adopted. This approach differs fundamentally from traditional unpaired methods like CycleGAN (Zhu et al., 2017), which rely on cycle-consistency losses. Cycle-consistency enforces that when an image is translated from source to target domain and then back, it should reconstruct the original image (i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$). While effective, this requires training two generators and discriminators simultaneously, doubling the computational burden.

CUT instead uses contrastive learning with a single generator-discriminator pair. Contrastive learning teaches the network through comparison of "positive" and "negative" patches. A positive pair consists of patches at the same spatial location in input and output images (e.g., top-left corner of input and top-left corner of output), while negative pairs are patches from different locations. The PatchNCE loss trains the network to bring positive pairs closer together in feature space while pushing negative pairs apart. The pipeline consists of a generator that transforms synthetic satellite images to look like real HIL images, and a discriminator that acts as a quality checker. During training, the discriminator learns to distinguish between real HIL images and generator outputs, providing an adversarial loss that backpropagates to the generator, pushing it to create more realistic images. The real HIL images serve as style references, the discriminator learns their visual characteristics (lighting, textures, sensor artifacts), and through the adversarial loss gradient, guides the generator to reproduce these characteristics (See Figure 3.8). This method only requires unpaired collections of synthetic and real images and achieves superior translation quality with significantly reduced computational resources compared to cycle-consistency strategies.

For this experiment, 1000 images were randomly sampled from the sunlamp and lightbox partitions (approximately 36% and 15% of each dataset, respectively) and used for training. The model was trained for 200 epochs with a batch size of 1, using a learning rate that linearly decays from 2e-4 to 0 starting from epoch 100. Separate models were trained for sunlamp and lightbox domains, on a 3090 NVIDIA GPU. Subsequently, all synthetic images from the SPEED+ training partition were processed through the respective generators, producing two sets of enhanced synthetic images, one adapted to sunlamp characteristics and one to lightbox. Example translations are shown in Figure 3.9.

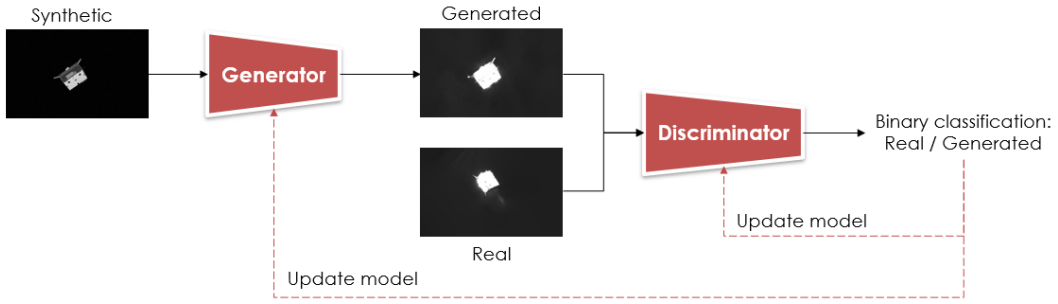

Figure 3.8: GAN training scheme where the generator transforms synthetic images to match real domain appearance, while the discriminator learns to distinguish generated from real samples. Classification feedback drives both networks' updates

While the enhanced synthetic images appear visually realistic, the true test of the translation's effectiveness lies in whether it successfully reduces the domain gap for pose estimation tasks. To evaluate this, a dual-stage pipeline was implemented following the progression of architectures explored in previous sections. Having demonstrated the superiority of Vision Transformers for domain generalization (Section 3.1.2) and the necessity of hybrid CNN-Transformer architectures for embedded deployment (Section 3.1.3), this experiment employs ConvFormer-S18 (Yu et al., 2024), a pure CNN that incorporates Transformer principles while remaining mobile-friendly. ConvFormer belongs to the MetaFormer family (Yu et al., 2022), which demonstrated that Transformers' effectiveness stems not from self-attention but from their overall architecture: token mixing and channel MLPs arranged in residual blocks. By replacing self-attention with depthwise separable convolutions as token mixers, ConvFormer achieves Transformer-like domain robustness while maintaining the computational efficiency of CNNs on embedded processors. The backbone connects to a lightweight 2-layer regression head comprising a depthwise separable convolution with ReLU activation followed by a fully connected layer. The network was trained for 40 epochs with a batch size of 48 using mean absolute error loss. Training was performed on Google Colab TPUs using Adam optimizer with cosine decay learning rate scheduling, starting from 7.5e-5. The backbone was initialized with ImageNet weights, and standard data augmentations were applied including random rotation, brightness and contrast adjustments, blur, and Gaussian noise. During inference, the regressed landmarks are processed by an EPnP solver to obtain the final pose estimate.
Table 3.10 compares the performance of the NN trained on GAN-enhanced syn-

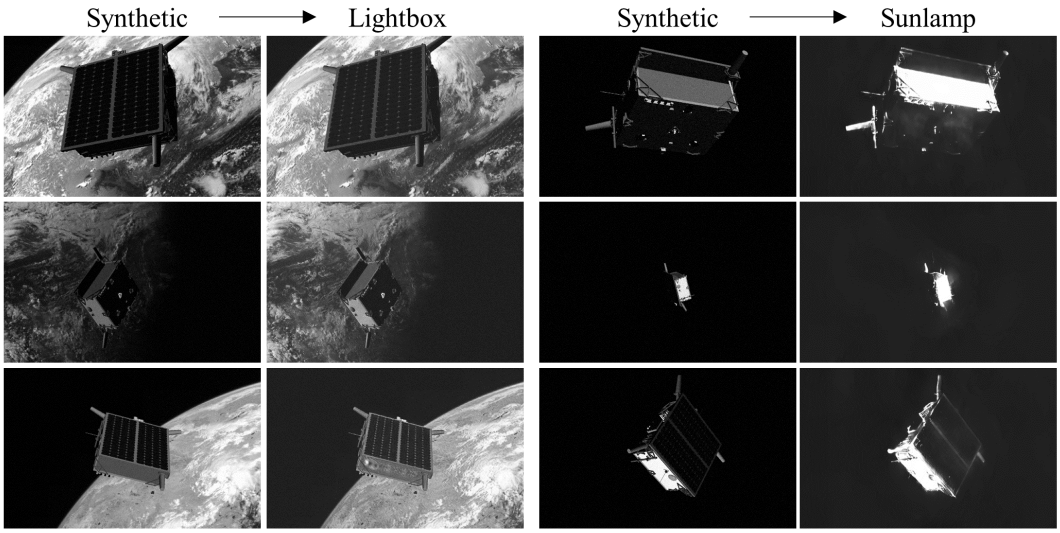

Figure 3.9: Synthetic renders transformed to Lightbox domain (left) and Sunlamp domain (right) using GAN-based image-to-image translation, reproducing characteristic illumination and appearance of each target dataset (Lotti, 2023)

Table 3.10: Impact of GAN preprocessing on pose estimation performance (Lotti, 2023)

GAN Processing	**Lightbox**				**Sunlamp**			
	$\langle\langle e_t \rangle\rangle$	$\langle\langle e_q \rangle\rangle$ [deg]	$\langle\langle E \rangle\rangle$	HQ	$\langle\langle e_t \rangle\rangle$	$\langle\langle e_q \rangle\rangle$ [deg]	$\langle\langle E \rangle\rangle$	HQ
With	**0.369**	**11.2**	**0.288**	**38.4%**	**0.131**	**5.08**	**0.116**	**67.6%**
Without	0.488	15.8	0.403	30.6%	0.580	25.5	0.597	15.8%

thetic images against training on original synthetic data, using median values to assess the improvement from GAN preprocessing. Since the objective is to measure the relative improvement from this augmentation rather than establish new accuracy benchmarks, medians provide a more robust metric that is less influenced by outliers. Beyond median errors, the analysis includes a high-quality (HQ) solution rate. Solutions qualify as HQ when meeting two thresholds: rotation error $< 5°$ (within 5 m) or $< 10°$ (beyond), and normalized position error < 0.1. The results reveal significant improvements, particularly for the sunlamp domain. Without GAN preprocessing, the sunlamp model exhibited poor performance with median translation error ($\langle\langle e_t \rangle\rangle$) of 0.580 and rotation error ($\langle\langle e_q \rangle\rangle$) of $25.5°$, achieving only 15.8% HQ solutions. GAN enhancement reduced these

errors by a factor of 5, bringing $\langle\langle e_t \rangle\rangle$ down to 0.131 and $\langle\langle e_q \rangle\rangle$ to $5.08°$, with HQ solutions jumping to 67.6%, a fourfold increase. The lightbox domain, while starting from better baseline performance, showed more modest improvements: median errors decreased by a factor of 1.4, with HQ solutions increasing from 30.6% to 38.4%. The superior performance on sunlamp images suggests that GAN preprocessing is particularly effective at modelling the complex lighting patterns and reflections characteristic of this domain, which hopefully is the primary effect simulated in the testbed setup presented in Chapter 5.

3.2 Real-time satellite pose estimation on Edge TPU

This section presents an adaptation of the pose estimation pipeline for real-time inference on embedded hardware. A central contribution of this work is a comprehensive parametric study that analyses the trade-off between pose estimation accuracy and inference latency on embedded hardware, an essential requirement for space applications with strict energy budgets. Given the focus on algorithm optimization, the SPEED dataset is adopted as a benchmark, using exclusively synthetic images.

3.2.1 Motivation and hardware context

The architectural explorations in previous sections revealed an inherent conflict between accuracy and computational efficiency. While the Vision Transformer pipelines from Section 3.1.1 achieved superior domain generalization, their computational demands remain incompatible with satellite processing constraints. Even the hybrid CNN-Transformer architecture (Section 3.1.3), despite its optimizations, achieved only 0.6 fps on embedded hardware, insufficient for real-time navigation. This challenge extends to all top-ranked methods from SPEC 2019 and 2021, reflecting a broader limitation of current AI approaches for space applications. This section therefore investigates how far accuracy can be maintained while meeting the stringent latency requirements of onboard deployment. To address this computational challenge, researchers at NASA proposed a smallsat board based on Edge Tensor Processing Units (Goodwill et al., 2021), designed to deliver high-performance, power-efficient AI capabilities for space missions. Building on this hardware platform, the pose estimation pipeline was redesigned

to exploit Edge TPU computational capabilities, enabling real-time inference under strict power constraints. The Edge TPU is an Application-Specific Integrated Circuit (ASIC) designed to accelerate machine learning inference, delivering high throughput at low power consumption. This study systematically explores the achievable trade-offs between inference speed, power consumption, and pose estimation accuracy, adopting a three-stage pipeline (Figure 3.1). Experiments were conducted once more on the Coral Dev Board Mini, but leveraging also its Edge TPU. The Edge TPU's operation set is inherently limited, supporting only fundamental neural network components: convolutional and fully connected layers with ReLU activations. These constraints preclude deployment of Transformer-based or other architectures requiring unsupported operations, directing the design toward simpler but TPU-compatible models.

3.2.2 Detection model for the Edge TPU

While previous sections prioritized detection accuracy through computationally intensive networks for bounding box regression, the need for Edge TPU optimization suggested a different approach. Rather than relying on model capacity for precise localization, the lightweight SSD MobileDet-EdgeTPU (Xiong et al., 2021) from the TensorFlow Object Detection API employs a structured detection framework, using predefined anchor boxes and multi-scale feature maps to achieve efficient detection with a model orders of magnitude smaller than the Transformers in Section 3.1.1. The NN processes 320×320x3 pixel inputs and is configured for single-class detection, as SPEED images contain only one satellite. The MobileDet backbone employs depthwise separable convolutions and ReLU6 activations, operations that are both computationally efficient and fully supported by the Edge TPU hardware. By replacing learned bounding box regression with structured prior knowledge about object scales and aspect ratios, the model maintains high detection performance while achieving real-time inference on the Edge TPU. Indeed, the detection framework employs a multi-scale anchor strategy across six feature layers, with anchor dimensions derived from k-means clustering on the SPEED dataset. Anchors range from 2% to 100% of image size across five aspect ratios (1:3, 1:2, 1:1, 2:1, 3:1), accommodating the full spectrum of satellite poses. Bounding box regression is performed using a Faster R-CNN-style box coder, while predictions are produced by a convolutional box

predictor that also leverages depthwise convolutions to reduce computational overhead. The final output is processed using non-maximum suppression with an Intersection over Union threshold of 0.6 to filter out redundant detections. With this configuration, the detection model comprises approximately 3.3 million parameters.

3.2.3 Scaling landmark regressors

For landmark regression, the EfficientNet family of neural networks is a suitable candidate for evaluating the trade-off between model accuracy and computational efficiency. EfficientNets are the result of a systematic model scaling study, where network depth, width, and input resolution are jointly optimized. Since standard EfficientNets contain TPU-incompatible operations, the previously introduced EfficientNet-Lite variants were adopted. Table 3.11 summarizes the specifications of the five Lite models, showing how width and depth coefficients scale systematically with input resolution from 224×224 (Lite0) to 300×300 pixels (Lite4). The width coefficient multiplies the number of channels in each layer, while the depth coefficient scales the number of layers in each network stage, together controlling model capacity and computational cost.

Table 3.11: EfficientNet-Lite models specifications relative to Lite0 variant (Lotti et al., 2023b)

Model	**Input size [px]**	**Width coefficient**	**Depth coefficient**
Lite0	224×224	1.0	1.0
Lite1	240×240	1.0	1.1
Lite2	260×260	1.1	1.2
Lite3	280×280	1.2	1.4
Lite4	300×300	1.4	1.8

Each model produces feature maps at 1/32 input resolution, which feed into the regression head detailed in Table 3.12. This head comprises four layers: an initial 1×1 convolution reducing to 128 channels, a depthwise separable convolution maintaining spatial context, and two final layers that regress the landmark coordinates. Feature reduction is therefore learned, rather than globally averaged as in the Transformer-based pipeline, using trainable operations instead

of fixed pooling to preserve more task-relevant information. Relative position and orientation are then computed using the EPnP algorithm, and successful pose estimates are further refined via Levenberg–Marquardt optimization.

Table 3.12: Edge TPU pipelines regression head structure (Lotti et al., 2023b)

#	Type	Activation	Kernel size	Padding	# filters
1	Standard	Relu	1	Valid	128
2	Separable	Relu	3	Same	128
3	Standard	-	1	Valid	kpts $\times$ 2
4	Standard	-	#3 resolution	Valid	kpts $\times$ 2

3.2.4 Training and TPU-Specific Optimization

The detection and regression networks were trained independently from pre-trained weights. The detection model was trained for 25,000 steps using a momentum optimizer with cosine decay scheduling, optimizing a combination of focal loss for classification and smooth Huber loss for bounding box regression. The regression networks were trained for 500 epochs using Adam optimizer with mean absolute error loss over normalized landmarks coordinates. Both pipelines employed data augmentation to improve generalization. Performance evaluation followed a k-fold cross-validation framework: the dataset was partitioned into k subsets, with models trained on k-1 partitions and evaluated on the hold-out one. This process was repeated k times, rotating the test partition, with results aggregated to compute robust performance metrics. Table 3.13 provides complete training configurations for all networks. Training was conducted on Google Colab TPUs.

Table 3.14 presents the pose estimation errors achieved by the pipeline variants using different EfficientNet-Lite backbones in their TensorFlow float32 baseline implementations. As expected, larger models demonstrate superior accuracy, with Lite4 achieving the lowest mean error ($\langle E \rangle = 0.0143$) using 15.4M parameters, while Lite0 shows higher error ($\langle E \rangle = 0.0186$) with only 6.9M parameters. The standard deviations reveal consistent performance across all variants, though larger models exhibit slightly better stability.

Table 3.13: Summary of training configuration for Edge TPU-optimized SDN and LRN

Parameter	**Detection Network**	**Regression Network**
Model architecture	SSD MobileDet-EdgeTPU	EfficientNet-Lite (0–4) + custom head
Pretrained weights	COCO	ImageNet
Optimizer	Momentum	Adam
Learning rate schedule	Cosine decay (0.015 $\rightarrow$ 0.45 warmup, then decay)	Linear decay ($2e^{-3}$ to $2e^{-5}$)
Loss function	Focal + Huber Loss	Mean absolute error
Batch size	200	250
Training steps / epochs	25,000 steps	500 epochs
Data augmentation	Crop, flip, brightness, contrast	Rotation, bounding box shift and scaling, random brightness and contrast
Cross-validation	k-fold (3,000 images per fold) on SPEED	k-fold (3,000 images per fold) on SPEED

Table 3.14: Pose estimation errors achieved by the Edge TPU-optimized pipeline variants, in TensorFlow float32 version (Lotti et al., 2023b)

Model	$\langle E \rangle$	$\langle e_q \rangle$ [deg]	$\langle e_t \rangle$ [m]	**Parameters** (SDN+LRN), M
Lite4	0.0143	0.57 ± 0.75	0.040 ± 0.167	15.4
Lite3	0.0152	0.61 ± 0.73	0.041 ± 0.106	10.5
Lite2	0.0163	0.66 ± 0.80	0.044 ± 0.109	8.4
Lite1	0.0166	0.67 ± 0.82	0.045 ± 0.124	7.7
Lite0	0.0186	0.76 ± 1.06	0.053 ± 0.201	**6.9**

To enhance pipeline efficiency, two key optimization techniques were then investigated:

- TensorFlow Lite (TFLite) conversion: enables lightweight deployment of neural networks on resource-constrained platforms by optimizing the computational graph.

- Weight quantization: reduces numerical precision from float32 to uint8, providing significant improvements in inference speed and memory usage.

Models were converted to TensorFlow Lite format (maintaining float32 precision) and deployed using tflite-runtime version 2.9.0 with XNNPACK, a library providing optimized neural network operators for ARM processors. All inference was performed on the Coral Dev Board Mini's CPU. Table 3.15 confirms that TFLite conversion preserves detection accuracy, with mean IoU remaining at 0.9332 and median IoU showing negligible change (0.9524 vs 0.9523) while Figure 3.10 presents the performance improvements achieved through TFLite deployment. Despite maintaining float32 precision, latency reductions ranged from 66% (Lite4) to 77% (Lite0), with smaller models showing greater relative speedup. These gains stem purely from graph optimization and XNNPACK's efficient operator implementations, while pose estimation accuracy remained identical to the TensorFlow baseline. Frame rates reported exclude image loading overhead, measuring only inference time.

Table 3.15: IoU scores for detection networks in TensorFlow and TFLite implementations (Lotti et al., 2023b)

SDN	$\langle$**IoU**$\rangle$	$\langle\langle$**IoU**$\rangle\rangle$
TF	0.9332	0.9524
TFLite	0.9332	0.9523

The next optimization phase employed Quantization Aware Training (QAT), a technique that prepares neural networks for low-precision deployment. Unlike post-training quantization, which simply converts a trained float32 model to uint8, often causing significant accuracy drops, QAT integrates quantization simulation directly into the training process. Fake quantization nodes are inserted throughout the network, mimicking the rounding and clipping effects of uint8 conversion while maintaining differentiability for backpropagation. This allows the network to learn robust representations that remain accurate despite reduced numerical precision. The quantization process transforms 32-bit floating-point values into 8-bit integers using scale and zero-point parameters that map the continuous range to discrete levels. This 4× reduction in precision translates to equivalent memory savings and enables the use of faster integer arithmetic units. For Edge

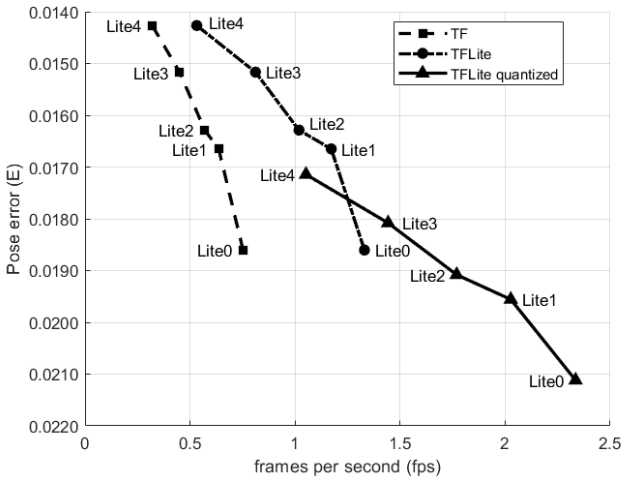

Figure 3.10: Pose error vs. inference time for NN variants (Lite0-4) on Coral Dev Board Mini CPU showing how TFLite conversion and quantization progressively increase throughput, the latter at the cost of increased pose estimation errors

TPU deployment, full quantization is mandatory as the hardware processes exclusively uint8 variables. Therefore, QAT becomes not just an optimization strategy but an essential bridge between model accuracy and hardware constraints. The entire cross-validation protocol was repeated with QAT-enabled training to ensure robust performance metrics under these deployment conditions. The Edge TPU compiler (v16.0) was used to prepare models for hardware acceleration. The Edge TPU features 8MB of on-chip memory for caching model parameters, a constraint that significantly impacts larger models. To optimize memory usage, the detection and regression networks were co-compiled, a strategy that allows the compiler to better allocate the limited on-chip memory between both networks, rather than treating them as independent models competing for the same resources. Table 3.16 summarizes the compilation results across all model variants. The detection network achieved near-perfect TPU mapping with 99.2% of operations (124 out of 125) successfully deployed on the accelerator. Only a single CUSTOM operation required CPU fallback. Similarly, regression networks showed excellent compatibility, with only QUANTIZE/DEQUANTIZE operations falling back to CPU execution.

Figure 3.11 compares CPU and Edge TPU performance, revealing remarkable improvements with hardware acceleration. The Edge TPU variants achieve up to 229% higher frame rates compared to CPU-only execution, demonstrating

Table 3.16: Edge TPU compilation statistics

Model	Input size [MB]	Output size [MB]	On-chip memory used [MB]	Off-chip memory [MB]	TPU operations	CPU operations
SDN - MobileDet	3.27	4.28	1.03-2.95	1.02-2.95	124	1
LRN - Lite0	3.98	4.43	3.87	0.45	68	2
LRN - Lite1	5.87	5.46	4.25	1.07	88	2
LRN - Lite2	5.58	6.57	4.62	1.72	88	2
LRN - Lite3	7.84	8.99	4.92	3.80	100	2
LRN - Lite4	12.88	14.75	4.60	9.81	124	2

the effectiveness of dedicated inference hardware. Beyond speed improvements, Edge TPU deployment significantly reduces power consumption and thermal load. Peak power consumption, measured with a benchtop supply, drops 27% when switching from CPU (3.0W) to TPU (2.2W). This efficiency gain manifests in thermal behaviour as well: average board temperature decreases from 85°C during CPU inference to 49°C with TPU acceleration, eliminating thermal throttling concerns for continuous operation. Regarding accuracy, QAT successfully mitigated quantization effects, limiting performance degradation to approximately 18% compared to float32 baselines. This represents an acceptable trade-off given the 2.3× speed improvement and substantial power savings, particularly for power-constrained space applications where every watt matters.

However Figure 3.11 reveals a concerning trend: inference speedup from Edge TPU acceleration diminishes significantly from Lite0 to Lite4 variants. This degradation correlates directly with the compilation statistics in Table 3.16, where larger models exceed the 8MB on-chip memory limit, requiring substantial off-chip memory streaming. For Lite4, with 9.81MB residing in off-chip memory, each inference requires continuous data transfer between the host and the TPU. The Coral Dev Board Mini's USB 2.0 interface appears to be the primary bottleneck rather than computational capacity. This hypothesis suggests that hardware acceleration benefits are constrained by the communication protocol, not the TPU's processing power. To validate this assumption, inference was repeated on a Coral Dev Board, which features a PCIe interface with its SoC, providing signif-

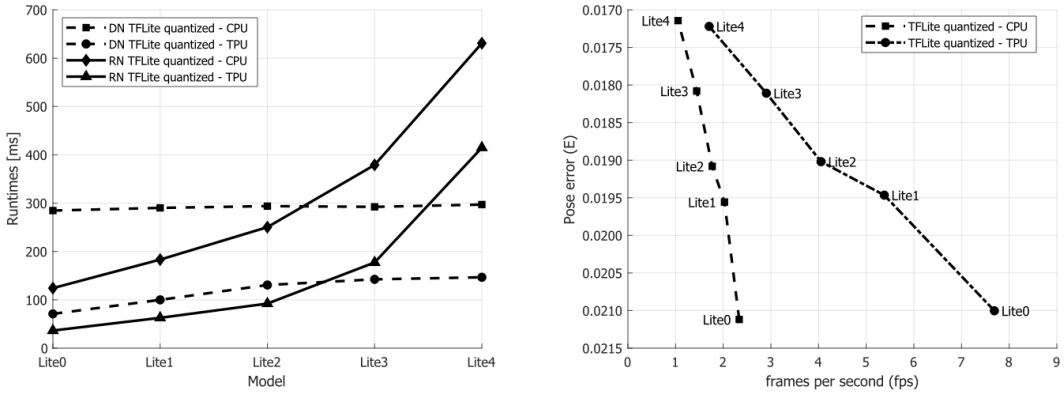

Figure 3.11: CPU vs. Edge TPU performance on Coral Dev Board Mini. Inference runtime for detection and regression networks (left), pose error vs. throughput for the full pipeline variants (right) (Lotti et al., 2023b)

icantly higher bandwidth. Table 3.17 presents evidence of this communication bottleneck.

Table 3.17: Performance of Edge TPU-optimized models on Coral Dev Board Mini vs Coral Dev Board, demonstrating the impact of memory transfer protocol

Pipeline variant	**Dev Board Mini fps**	**Dev Board fps**	**Improvement**
Lite0	7.7	27.0	+250%
Lite1	5.4	22.7	+320%
Lite2	4.0	18.4	+360%
Lite3	2.9	15.5	+434%
Lite4	1.7	11.6	+582%

The PCIe-equipped Dev Board delivers 3.5× to 6.8× higher frame rates across all model variants, with the performance gap widening significantly for larger models. Lite0, requiring minimal off-chip memory (0.45MB), shows a 250% improvement, while Lite4, with 9.81MB off-chip memory, achieves a remarkable 582% speedup. The Dev Board's consistent performance across model sizes, degrading from 27 to 11.6 fps rather than 7.7 to 1.7 fps, demonstrates that with adequate bandwidth, the Edge TPU can maintain reasonable inference speeds even for models exceeding on-chip memory capacity.

3.3 Uncertainty estimation using SVGPs

A critical limitation of the neural network approaches presented thus far is their inability to quantify prediction uncertainty. Neural networks are often overconfident in their predictions, providing deterministic outputs even in failure cases where the input lies far from the training distribution. This behaviour makes them unsuitable for safety-critical applications such as spacecraft proximity navigation, where the system must autonomously recognize its own limitations and flag unreliable predictions. This section presents preliminary results using Stochastic Variational Gaussian Processes (SVGP) to add uncertainty quantification to pose estimation. By combining neural network feature extraction with Gaussian Process (GP) predictions, the method provides confidence estimates alongside pose outputs.

3.3.1 Proposed methodology

Common practices for extracting uncertainty estimates from neural networks include:

- Monte Carlo Dropout (Gal and Ghahramani, 2016): applies different dropout during inference, treating each forward pass as a sample from an approximate posterior distribution.
- Bayesian Neural Networks (Blundell et al., 2015): replaces deterministic weights with probability distributions, learning both mean and variance parameters that are sampled at inference.
- Deep Ensembles (Lakshminarayanan et al., 2017): trains multiple independent models with different initializations, aggregating predictions to estimate uncertainty.

These methods can capture both aleatoric uncertainty (i.e. the inherent data noise) and epistemic uncertainty (i.e. model knowledge limitations). While dropout primarily captures epistemic uncertainty, Bayesian NNs model both types explicitly, and ensembles effectively estimate epistemic uncertainty through prediction disagreement. However, these traditional approaches do not meet the

tight computational requirements of space applications, as their reliance on multiple forward passes undermines efforts to keep models lightweight for onboard deployment. To mitigate this, an alternative approach is proposed that combines lightweight CNNs from the EfficientNet-Lite family with an inference pipeline that limits repeated passes. The pipeline replaces conventional regression heads with SVGPs that operate on CNN-extracted feature maps. The core principle is to leverage SVGPs' ability to quantify prediction reliability through feature space correlation. When presented with a new input, the SVGP measures similarity between its feature representation and those observed during training. High correlation with training features translates into confident predictions, while low correlation signals potential failure cases. This correlation-based confidence enables frame rejection when predicted uncertainty exceeds operational thresholds. This mechanism is useful under domain shift, where feature maps from out-of-distribution real images often differ from synthetic training embeddings. The SVGP identifies these anomalous patterns as low-correlation regions in the feature space and assigns them higher predictive uncertainty. For retained frames, uncertainty on regressed landmarks is propagated through the PnP solver to obtain pose uncertainty bounds. A high level overview of the uncertainty-augmented pipeline is illustrated in Figure 3.12. The training setup follows a three-stage approach: first, neural networks are trained with standard regression/classification heads to learn discriminative features. Second, these heads are removed and the trained backbones process the entire SPEED+ dataset to extract feature maps. Third, the extracted features are used to train two SVGP models, respectively for ROI detection and landmark regression, which learn feature-to-output mappings with predictive uncertainty.

Within this framework, the detection step was reformulated as a binary classification task on image patches. Input images are divided into non-overlapping patches that are individually processed by the CNN to determine satellite presence. This approach preserves full input resolution during ROI cropping while enabling efficient batch processing, as patches are stacked along the batch dimension for parallel inference. Training labels for the patch classifier were generated using the Segment Anything (SA) model (Kirillov et al., 2023) to produce segmentation masks (Figure 3.13). SA has not being fine-tuned on SPEED+, but segmentation was guided by providing Tango's landmark coordinates as positive examples, while 10 randomly sampled points outside the satellite's bounding

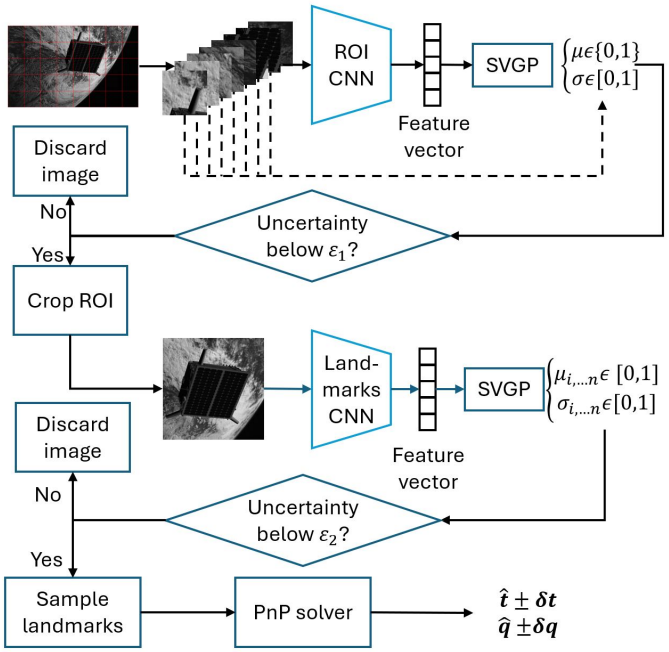

Figure 3.12: Uncertainty-aware pose estimation pipeline; SVGP heads on ROI detection and landmark regression networks provide confidence estimates, enabling frame rejection when uncertainty exceeds thresholds ε_1 or ε_2. Accepted frames yield pose estimates with uncertainty bounds (Lotti et al., 2025b)

box served as negative examples. This helped SA distinguish the satellite from complex backgrounds. The resulting segmentation masks were used to label each patch as positive (1) or negative (0). Images were divided into 40 patches of 240×240 pixels. To address class imbalance, patches were resampled during training to maintain exactly 50% positive and 50% negative examples per batch. Individual patches were augmented with random flips, brightness/contrast adjustments, Poisson noise, and cutout. The network was trained with binary cross-entropy loss for 100,000 steps, using a batch size of 40 (matching the patches per image) and a constant learning rate of 1e-3.

The second network follows the traditional approach by producing feature maps for landmark regression within the ROI. Training is performed with mean absolute error loss on landmark coordinates for 100 epochs with a batch size of 48. The learning rate decreases linearly from 1e-3 to 1e-5. Augmentation includes the same photometric transformations as the detection network, plus usual geometric variations: random rotation, ROI rescaling (1.15× to 2.0×), and translation (up to 25% of diagonal length). ROIs extracted from ground truth bounding boxes are

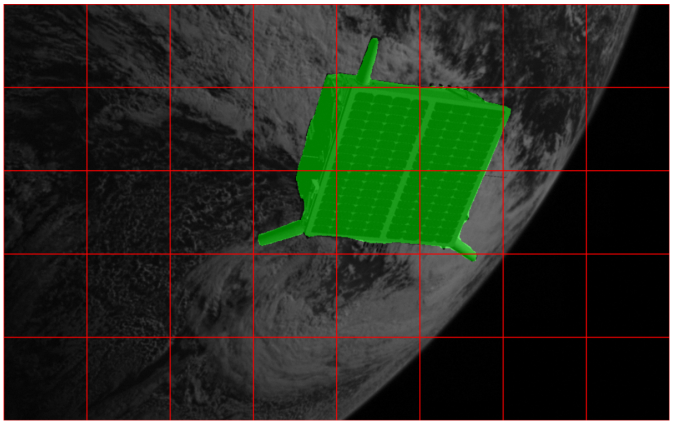

Figure 3.13: Image divided into grid cells with Segment Anything pseudo-labels (green mask) indicating satellite presence for binary patch classification

padded after transformation to prevent distortion during resizing. Both CNNs employ the Adam optimizer with ImageNet-pretrained initialization and incorporate quantization-aware training to ensure Edge TPU compatibility. After training, models are converted to TensorFlow Lite (full integer quantization) and compiled for Edge TPU deployment. The extracted feature maps from the CNNs serve as inputs to Gaussian Processes, which provide the uncertainty quantification missing from standard neural networks. GPs are Bayesian models that naturally produce predictions with associated confidence bounds by modelling correlations between data points through kernel functions. When presented with new features, the GP evaluates their similarity to training features: high similarity yields confident predictions, while dissimilar features produce high uncertainty. However, traditional GPs scale poorly with dataset size, requiring $\mathcal{O}(N^3)$ computations that quickly become intractable. Stochastic Variational Gaussian Processes overcome this limitation through sparse approximation: rather than computing correlations across all N training samples, they maintain only M inducing points $(M \ll N)$ that compactly represent the training distribution. This reduces computational complexity to $\mathcal{O}(M^2N)$. The inducing points are initialized via k-means clustering on extracted CNN features, to maximize feature space coverage. The training process optimizes the Evidence Lower Bound (ELBO) in Eq. (3.7).

$$\mathcal{L}_{\text{ELBO}} = \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \left[\log p(y_i \mid f_i)\right] - \text{KL}[q(z) \,\|\, p(z)] \tag{3.7}$$

where $f(x)$ represents the latent function value for input x, and $p(y|f(x))$ is the likelihood of observing label y given that latent value. The variational distribution $q(z)$ approximates the posterior over inducing variables z, while $p(z)$ represents their prior distribution. The second term, the Kullback-Leibler (KL) divergence, penalizes deviations between the approximate posterior $q(z)$ and the prior $p(z)$, ensuring the learned distribution remains close to the prior assumptions. In the proposed approach, CNNs map each input image x to a feature vector $\phi(x)$. The Gaussian Process operates on this learned representation rather than raw pixels:

$$f(\phi(\mathbf{x})) \sim \mathcal{GP}(0,\, k(\phi(\mathbf{x}),\, \phi(\mathbf{x}'))) \tag{3.8}$$

where $k(\cdot,\cdot)$ measures similarity between feature vectors. Two SVGP models are employed: the first classifies patches for satellite detection using a Bernoulli likelihood, while the second regresses continuous landmark coordinates using a Gaussian likelihood with a Squared Exponential kernel. Both models are trained on features extracted from SPEED+ synthetic images for 250 epochs using Adam optimizer (batch size: 64, learning rate: 1e-4) with 100 inducing points. SVGPs are implemented using the GPFlow library (Matthews et al., 2017).

3.3.2 Results and limitations

Results are presented in three parts: performance comparison between standard CNNs and SVGP models, uncertainty quantification analysis, and computational efficiency evaluation in terms of inference time. Table 3.18 presents patch classification accuracy across domains. Training and validation metrics were computed on 280,000 patches (about 7,000 images) from each partition. While the training set maintained class balance (50% positive/negative through resampling), validation and test sets preserved the natural class distribution to reflect real-world performance. Two key findings emerge. First, both models generalize well to lightbox images but show significant performance degradation on sunlamp data, confirming persistent domain gap challenges. Second, the SVGP achieves comparable accuracy to the standard CNN classifier, demonstrating that probabilistic modelling introduces minimal performance drop while adding uncertainty quantification capabilities.

Table 3.19 reports the mean absolute error for landmark regression on ground truth ROIs, comparing dense regression heads with SVGP models. On synthetic

Table 3.18: Patch classification accuracy for CNN and SVGP models across synthetic and real domains (Lotti et al., 2025b)

Model	Synthetic train	Synthetic validation	Lightbox	Sunlamp
CNN + dense head	96.3%	96.3%	87.3%	51.3%
CNN + SVGP	93.4%	95.7%	86.7%	45.2%

data, the SVGP shows higher error than the dense head (0.017 vs 0.009). However, this gap narrows considerably on real domains: only 13% degradation on lightbox (0.044 vs 0.039) and negligible difference on sunlamp (0.065 vs 0.064). This convergence suggests that dense heads overfit synthetic data, whereas SVGPs provide more consistent cross-domain performance.

Table 3.19: Landmark regression error (MAE) on ground-truth ROIs comparing dense and SVGP regression heads (Lotti et al., 2025b)

Model	Synthetic train	Synthetic validation	Lightbox	Sunlamp
CNN + dense head	0.009	0.009	0.039	0.064
CNN + SVGP	0.017	0.017	0.044	0.065

Figure 3.14 illustrates normalized epistemic uncertainty histograms from the SVGP classifier across validation, lightbox, and sunlamp domains. Image-level uncertainty is computed as the average uncertainty across all 40 patches, with histograms normalized to enable comparison between differently-sized partitions. The distributions reveal clear domain-dependent confidence patterns. Synthetic validation data clusters at low uncertainty values, indicating high model confidence on in-distribution samples. Lightbox images show moderate uncertainty increase, while sunlamp images exhibit substantially higher and more dispersed uncertainty values. This progression, directly correlates with the accuracy degradation observed in Table 3.18, validating that epistemic uncertainty effectively captures domain shift. The elevated sunlamp uncertainty indicates that extracted features deviate significantly from the synthetic training distribution, providing a reliable signal for detecting potential failure cases before they occur.
A moderate positive correlation (Pearson coefficient = 0.47) exists between SVGP epistemic uncertainty and pose error $\langle E \rangle$ on the validation set, supporting

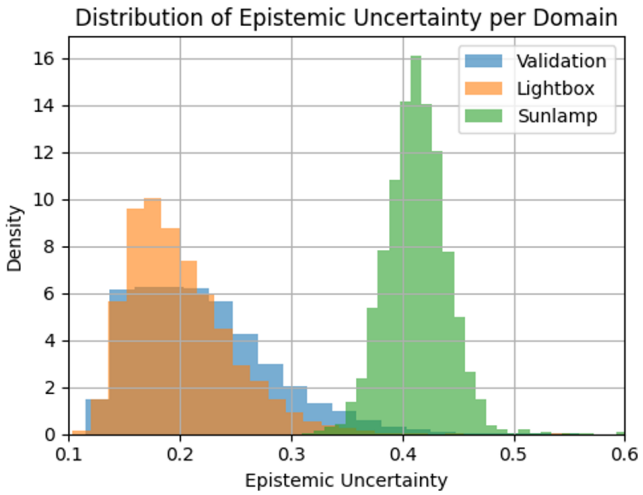

Figure 3.14: Epistemic uncertainty distributions from the SVGP classifier; synthetic validation images cluster at low uncertainty, Lightbox shows moderate increase, and Sunlamp exhibits substantially higher values reflecting domain shift (Lotti et al., 2025b)

uncertainty-based filtering. By rejecting predictions exceeding the validation set's 50th percentile uncertainty threshold, unreliable estimates can be eliminated before they propagate to the navigation system. Table 3.20 quantifies this filtering strategy's impact on pose estimation accuracy. The subscript indicates the percentage of rejected frames. Results exclude PnP failures and sunlamp data, where excessive misclassifications prevent meaningful pose estimation. The filtering demonstrates clear benefits on synthetic validation data: discarding 50.2% of high-uncertainty frames reduces mean translation error from 0.46 m to 0.39 m and rotation error from 13.3° to 7.51°. However, lightbox performance shows minimal improvement despite rejecting 36.6% of frames, suggesting that uncertainty correlates less strongly with pose error under real-world conditions. This domain-dependent behaviour indicates that uncertainty calibration learned on synthetic data may not fully transfer to real domains.

Table 3.20: Pose errors with and without uncertainty filtering (Lotti et al., 2025b)

Dataset	$\langle E \rangle$	$\langle e_q \rangle$ [deg]	$\langle e_t \rangle$ [m]
Synthetic validation	0.307	13.3	0.46
Synthetic validation – $\varepsilon_{1 \mid 50.2\%}$	0.186	7.51	0.39
Lightbox	0.80	35.0	1.04
Lightbox – $\varepsilon_{1 \mid 36.6\%}$	0.76	33.5	1.05

Following Li et al. (2022), pose uncertainty is estimated through Monte Carlo sampling. For each landmark, 250 (x,y) coordinate samples are drawn from normal distributions parameterized by SVGP-predicted means and variances. EPnP is used to compute a pose for each sampled landmark set, with the average deviation from the reference pose (computed using mean landmarks) serving as the uncertainty metric. Table 3.21 presents correlation coefficients between predicted uncertainties and true pose errors. Rotation uncertainties show moderate to strong correlation (0.73 synthetic, 0.58 lightbox), while position uncertainties exhibit weaker correlation (0.27 synthetic, 0.19 lightbox).

Table 3.21: Pearson correlation between SVGP-predicted pose uncertainties and actual pose errors (Lotti et al., 2025b)

Dataset	**Rotation**	**Position**
Synthetic validation	0.73	0.27
Lightbox	0.58	0.19

Table 3.22 details the computational breakdown of the pipeline. While individual components are efficient, CNNs require $\approx 50\,\mathrm{ms}$ and SVGPs $\approx 35\,\mathrm{ms}$, the patch-based detection strategy dominates execution time with 1.6 seconds for 40 patches. This bottleneck arises from the Edge TPU's lack of batch-inference support: the CNN backbone must process patches sequentially rather than in a single batched pass. This throughput, while below real-time requirements, demonstrates that uncertainty-aware pose estimation is approaching feasibility for edge deployment with further optimization.

Table 3.22: Execution time breakdown for uncertainty-aware pose estimation pipeline on Edge TPU (Lotti et al., 2025b)

Task	**Runtime [ms]**
CNN1 (40 patches)	1644.8
SVGP classifier	34.6
CNN2	50.7
SVGP regressor	34.2
Pose computation for 250 samples	151.0

While this preliminary study demonstrates the feasibility of uncertainty-aware pose estimation on edge devices, two main limitations remain.

1. The Edge TPU currently lacks support for batch inference, requiring sequential execution of the CNN backbone for each of the 40 patches. This constraint transforms what should be a single batched forward pass into 40 individual inferences, severely impacting runtime performance and partially undermining optimization efforts. This limitation, however, is specific to the Edge TPU architecture, lightweight GPUs and other accelerators support efficient batch processing, potentially reducing detection time by an order of magnitude.

2. Uncertainty calibration shows domain-dependent behavior: while effective on synthetic validation data, the correlation between uncertainty and actual errors weakens on real domains.

Despite these limitations, the framework establishes that, with further research, uncertainty quantification can be integrated into edge-deployable pose estimation systems, providing the self-awareness critical for safe spacecraft operations.

3.4 Concluding remarks

This chapter has presented three complementary research directions addressing the problem of pose estimation of known, uncooperative targets in space. Each direction targets a different limiting factor for the practical deployment of operational satellite pose estimation systems, namely:

1. Bridging the domain gap. Domain adversarial methods, hybrid CNN-Transformer architectures, and GAN-based data augmentation were explored to mitigate the performance drop between synthetic training data and real imagery. These approaches demonstrated that domain robustness can be improved without requiring large real datasets prior to launch.

2. Embedded deployment. A systematic parametric study on Edge TPU hardware quantified the trade-offs between accuracy, latency, and power consumption, highlighting that real-time performance is achievable on smallsat-class processors, though at the cost of architectural simplifications.

3. Uncertainty awareness. By integrating Stochastic Variational Gaussian Processes, the framework was extended to provide uncertainty estimates alongside pose predictions, enabling filtering of unreliable outputs.

The results suggest that practical onboard pose estimation systems can be both efficient and self-aware, though further work is needed to close residual gaps in domain transfer and uncertainty calibration.

4

Pose estimation of unknown targets

In the framework of ADR and IOS missions, an inspection phase is essential when approaching unknown targets. This phase aims to assess the target's state, structural integrity, and geometric configuration to enable safe proximity operations. Such operations depend on robust computer vision algorithms that can detect and localize previously unseen spacecraft while operating within the computational constraints of onboard systems. This section begins by presenting algorithms developed against the PoseBowl Challenge Detection Track dataset, for detection of previously unseen spacecraft in the far range. Subsequently, one of these solutions is integrated into an end-to-end pipeline that incorporates a custom feature matcher to estimate the frame-to-frame relative pose, to reconstruct the camera motion around the target, using the PoseBowl Pose Estimation Track dataset as a benchmark. However, pose estimation errors remain high, in line with competition results, highlighting the need for more comprehensive datasets to enable 3D viewpoint-supervised training.

4.1 Far range detection

The public dataset from the PoseBowl Detection Track competition comprises around 25,800 RGB images (1024 × 1280 px) featuring 15 different satellite

models with random backgrounds. The ground truth bounding boxes range from a few pixels to full image coverage and include thin appendages. Sample images are illustrated in Figure 4.1. In this context, several detection algorithms and pipelines have been set up to assess how well the lightweight models presented in the previous sections generalize to unseen spacecraft.

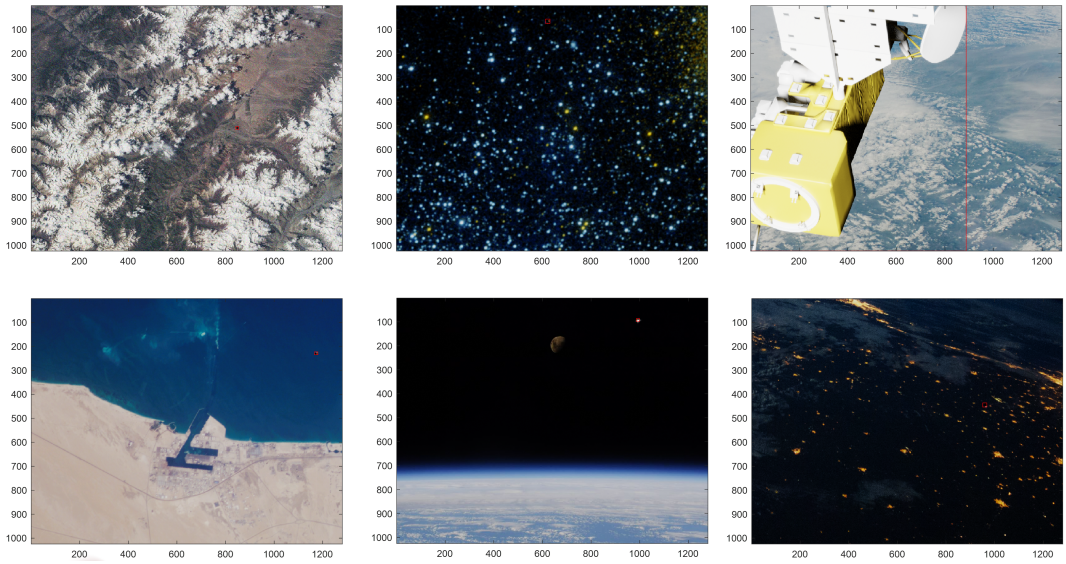

Figure 4.1: Sample images with annotated bounding box from the PoseBowl Detection Track dataset (NASA and DrivenData, 2024)

In order to assess the algorithms developed locally, the public dataset has been partitioned into training and validation sets, as illustrated in Table 4.1, ensuring no overlapping models between the two sub-partitions.

Table 4.1: Repartitioning of the PoseBowl Detection track public dataset for local evaluation

Dataset Split	**Train**	**Validation**
Images	20320	5479
Satellite Models	11	4

Three different neural networks were trained and combined into four different detection strategies. The first approach consists of retraining the MobileDet model used in Section 3.2.2. In this case, aspect ratios for the bounding boxes are not optimized using k-means clustering, as the models in the validation dataset are

different from those in the training set; therefore, default values from the API are retained. All satellite models are mapped to a generic "satellite" class. A sliding window approach is also implemented: whenever the detection threshold for the most confident bounding box falls below a given threshold, inference is repeated on a set of up to five sub-images, as shown in Figure 4.2. This multi-window approach reduces information loss during image resizing, and the bounding box with the highest confidence score is retained as the final prediction.

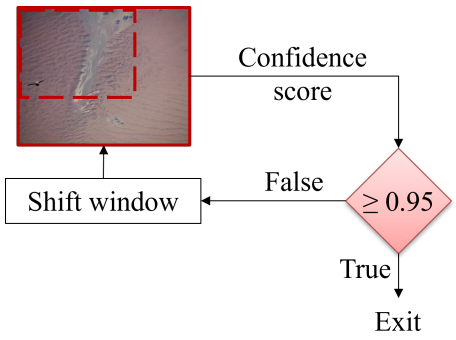

Figure 4.2: Sliding window detection; when initial confidence is below threshold, inference is repeated on up to five sub-windows to reduce information loss from resizing, the bounding box with highest score is retained

The second model consists of an EfficientNet-Lite0 CNN with a simple regression head. The head comprises a global average pooling layer followed by a dense layer with no activation. The model has an input resolution of 320×320 px, matching the MobileDet, and directly predicts bounding box coordinates. It is trained using mean absolute error, similar to the approach adopted in Section 3.1.1, but with a much lighter model. The third network is similar to the patch classifier introduced in Section 3.3, except that an EfficientNet-Lite0 is used to keep the pipeline lightweight. Input images are converted to black and white and partitioned into 256×320 px patches, then stacked along the batch dimension. In this case, class imbalance during training can be particularly pronounced since, as shown in Figure 4.1, the satellite might be extremely small and occupy only a single patch. Figure 4.3 shows an example of the mitigation strategy adopted, where the only positive patch is repeated and augmented.
Four pipelines have been tested. First, Models 1 and 2 were used independently. Then, the patch classifier was prepended to both models to isolate regions of interest before localization, thus avoiding direct image resizing. Results are illus-

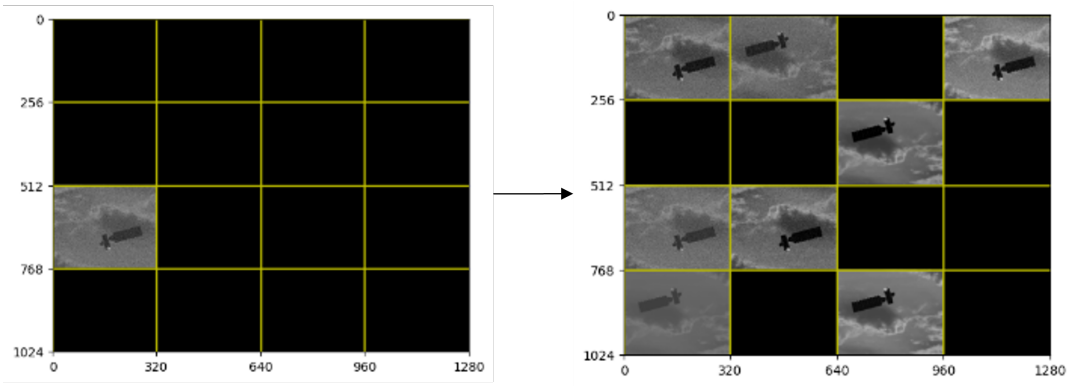

Figure 4.3: Class imbalance mitigation; the single positive patch containing the satellite (left) is oversampled (and augmented) to balance training data (right)

trated in Table 4.2, which shows that the patch classifier significantly increases the detection performance for both base models. The patch classification preprocessing step improves the average IoU by 0.08 for MobileDet and by 0.12 for the box regressor. The best performance is achieved by combining the patch classifier with the MobileDet, reaching an average IoU of 0.78. This demonstrates the effectiveness of the patch classifier in preserving image detail and enhancing localization accuracy compared to direct image resizing.

Table 4.2: IoU scores for detection pipelines on the validation set

Approach	$\langle$**IoU**$\rangle$	**Total NN parameters (M)**
1) MobileDet + Sliding Window	0.70 ± 0.24	3.3
2) EfficientNet-Lite0 bounding box regressor	0.64 ± 0.30	3.4
3) Patch classifier + MobileDet	0.78 ± 0.23	6.7
4) Patch classifier + EfficientNet-Lite0 bounding box regressor	0.76 ± 0.25	6.8

Figure 4.4 illustrates the scatter plot of IoU against ground truth bounding box size (diagonal) for the different satellites in the validation set, using pipeline #3. The plot shows that smaller bounding boxes lead to more missed or inaccurate detections. There is no clear trend across satellite models, and bounding box size remains the dominant factor affecting accuracy, with tiny boxes remaining

especially difficult.

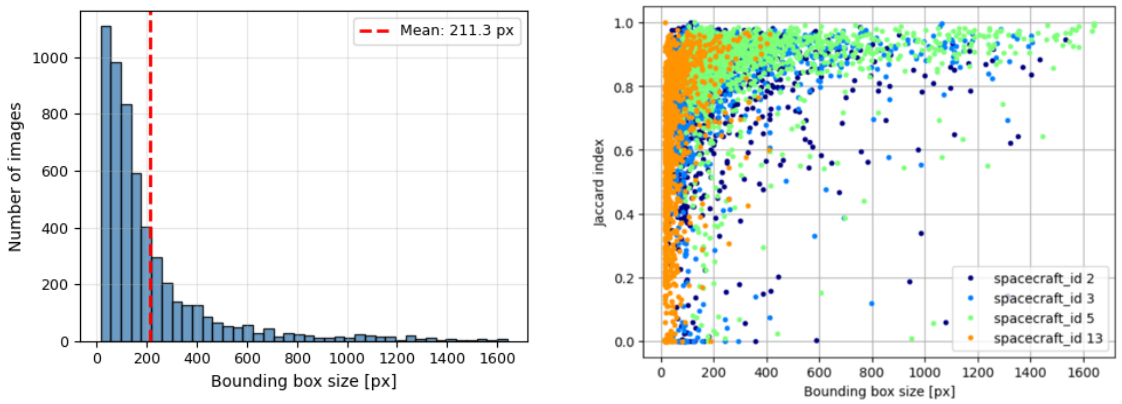

Figure 4.4: Bounding box diagonal distribution in the validation set (left), IoU vs. bounding box diagonal size showing localization accuracy degradation for smaller targets (right)

4.2 Pose estimation

The PoseBowl Pose Estimation Track challenged participants to recover camera poses around a stationary unkown target using images only. The competition supplies a dataset suitable for benchmarking computer vision algorithms that estimate camera motion during inspection maneuvers. Structure-from-Motion (SfM) is a straightforward approach, however, conventional feature-extraction methods often underperform on space imagery, and the complete SfM pipeline is computationally intensive. In this section, a pipeline is configured to evaluate the feasibility of an extremely lightweight deep learning–based feature extractor and matcher applied to sequential image pairs for unknown spacecraft pose estimation.

4.2.1 Dataset processing

The public dataset features 660 image chains containing 100 images each, depicting different satellite models. Unfortunately image chains of different satellites are shuffled and the naming does not reflect the content. Therefore, since the purpose of this experiment is to assess the capability of the network to adapt to previously unseen targets, the first step was to cluster chains based on the satellites they feature to keep different models separated during training and test.

This has been done by plotting one random image from the chains and manually grouping them into lists. After this process, 14 satellite models have been identified. However 2 chains ('1f5cc595f6', '972fc3cbe6') are deliberately excluded as the images appear extremely blurry and the target is not even distinguishable by a human eye. Figure 4.5 reports a montage of the identified models in the dataset.

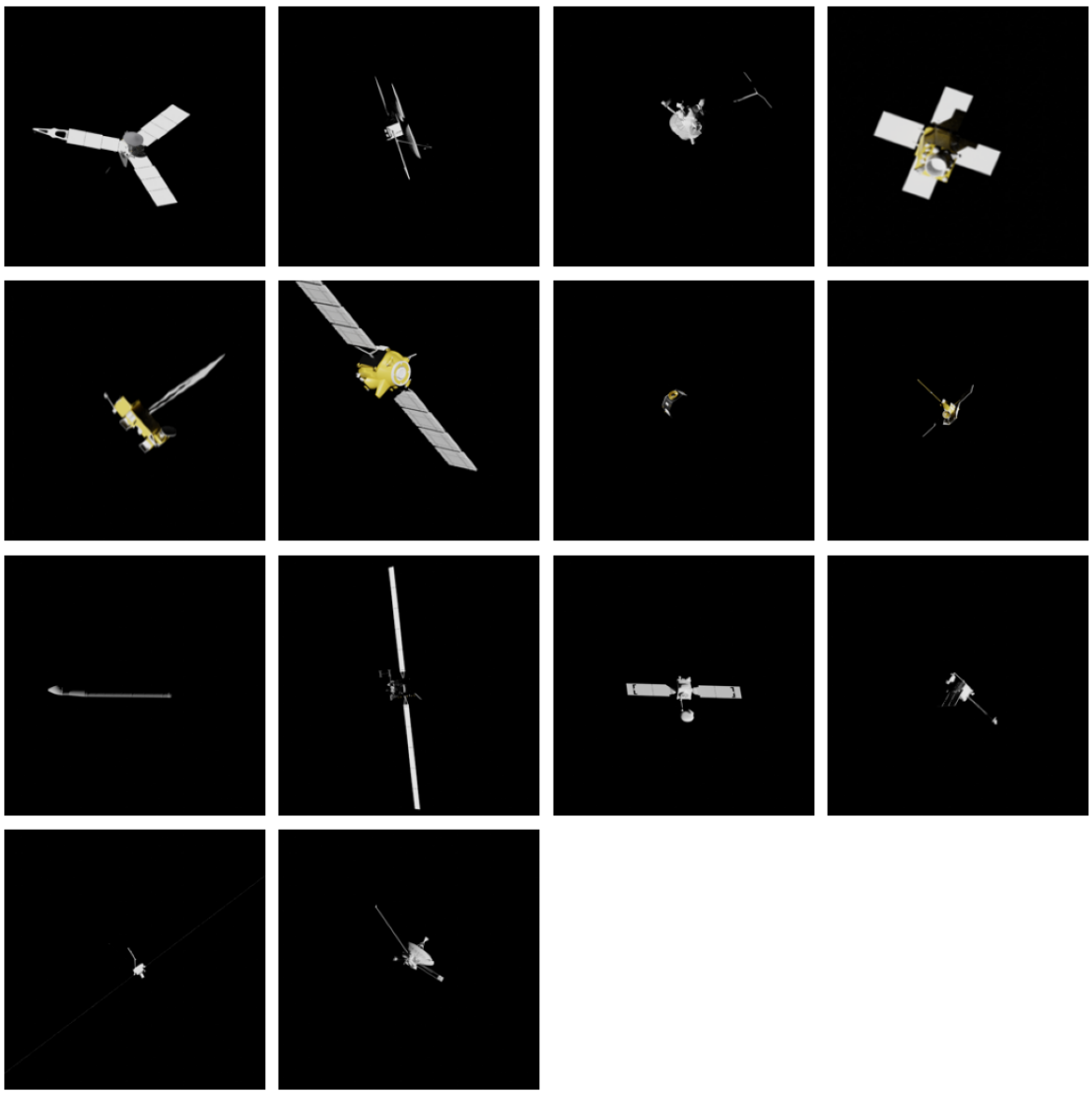

Figure 4.5: Sample images from the PoseBowl Pose Estimation Track dataset, illustrating different satellite models (NASA and DrivenData, 2024)

Chain lists were then processed on a per-satellite basis, and the resulting TFRecord files were named after the satellite model they feature. This facilitates keeping the training and validation sets separate by specifying the model keys assigned

to each partition. During processing, each image was passed through the patch classifier, trained on the Detection Track dataset, to obtain a zoomed-in view of the satellite, exposing more details. The resulting ROI was then fed to a Segment Anything model to produce a segmentation mask, which was stored in the record alongside the image and later used for guiding NN training.

4.2.2 Deep learning-based feature matching

The feature matcher adopts a hybrid CNN-Transformer architecture inspired by the LoFTR model (Sun et al., 2021), while remaining lightweight. It consists of a dense feature matcher, meaning it produces a descriptor for every pixel in the final feature map, enabling pixel-wise correspondence search between the two feature maps. Two consecutive RGB frames of size $300 \times 300 \times 3$ are processed by a shared EfficientNet-Lite4 backbone pre-trained on ImageNet and truncated at the `block5a_activation` layer to preserve higher spatial resolution. The extracted features are reduced to 64 channels via a 1×1 convolution and then upsampled through a learned transposed convolution (kernel size 4, stride 2) followed by ReLU and $\mathrm{LayerNorm}$, producing feature maps $\mathbf{F}_A, \mathbf{F}_B \in \mathbb{R}^{38\times38\times64}$.

To enhance spatial awareness, a 2D positional encoding is added to these features. The encoding combines multi-frequency Fourier features, sine and cosine functions at exponentially increasing frequencies applied to normalized spatial coordinates, with geometric channels that capture radial distance and angular position. This rich positional information is linearly projected to match the feature dimension and normalized, then added to the features to create position-aware representations $\mathbf{G}_A$ and $\mathbf{G}_B$.

The spatial feature maps are then flattened into sequences of dimension 1444×64 and processed through two transformer layers. Each layer first applies self-attention independently to both image features, allowing each position to aggregate context from its own image. This is crucial for understanding local structure before matching. Subsequently, symmetric cross-attention enables features from one image to query and attend to features from the other image bidirectionally. This cross-modal interaction refines the features by incorporating information about potential correspondences, making matched points more similar in the embedding space while pushing non-matches apart.

After the transformer processing, the refined sequences are reshaped back to

spatial grids and upsampled once more via a transposed convolution with layer normalization, yielding dense descriptors $\mathbf{O}_A, \mathbf{O}_B \in \mathbb{R}^{76\times76\times64}$. The entire architecture contains approximately 2.1M parameters, making it suitable for real-time applications.

The model is trained in a self-supervised manner without requiring ground-truth correspondence annotations, which are not available. The training leverages known geometric transformations to create synthetic training pairs. Given an input image $\mathbf{I}$, a second view $\mathbf{I}'$ is generated by applying a combination of random transformations: rotation $\theta \sim \mathcal{U}(-30°, 30°)$ with 80% probability, scaling $s \sim \mathcal{U}(0.8, 1.2)$ with 50% probability, and shear $\gamma \sim \mathcal{U}(-0.2, 0.2)$ with 30% probability. These transformations establish a known mapping between pixels, where each location (i, j) in the original image corresponds to location $(i', j') = \mathcal{T}(i, j)$ in the transformed image, providing dense supervision for every valid pixel.

Both images undergo photometric augmentations, including random brightness, contrast, Gaussian blur, and additive noise, each applied independently with 25% probability to improve robustness to appearance variations. After processing through the feature matcher, the output descriptors $\mathbf{O}_A, \mathbf{O}_B \in \mathbb{R}^{76\times76\times64}$ are flattened and L2-normalized to obtain $\hat{\mathbf{O}}_A, \hat{\mathbf{O}}_B \in \mathbb{R}^{N\times64}$ where $N = 76^2$, with each row representing the descriptor of a spatial location. A similarity matrix is computed by taking the dot product between all descriptor pairs:

$$\mathbf{S} = \frac{1}{\tau}\hat{\mathbf{O}}_A\hat{\mathbf{O}}_B^T \tag{4.1}$$

where $\mathbf{S}_{ij}$ represents the similarity between the i-th spatial location in image A and the j-th location in image B, and $\tau = 0.1$ is a temperature parameter that sharpens the distribution. The training objective is formulated as a cross-entropy loss where the similarity matrix $\mathbf{S}$ provides the logits and the known geometric transformation provides the labels. For the forward direction (A to B), each row i of $\mathbf{S}$ is treated as a classification problem with label $j^* = \mathcal{T}(i)$, the known correspondence index:

$$\mathcal{L}_{A\to B}^{(i)} = -\log \frac{\exp(\mathbf{S}_{i,j^*})}{\sum_{k=1}^{N}\exp(\mathbf{S}_{i,k})} \tag{4.2}$$

A symmetric loss $\mathcal{L}_{B\to A}$ is computed using $\mathbf{S}^T$ as logits and the inverse transformation indices as labels. The segmentation masks obtained during preprocessing

are incorporated to force the model to focus on learning pixel matches only within the target object, ignoring the background. The bidirectional losses are computed separately and masked according to their respective valid pixels. The final loss is averaged over the total number of valid pixels from both directions:

$$\mathcal{L} = \frac{1}{|\mathcal{V}_A| + |\mathcal{V}_B|} \left(\sum_{i \in \mathcal{V}_A} \mathcal{L}^{(i)}_{A \to B} + \sum_{j \in \mathcal{V}_B} \mathcal{L}^{(j)}_{B \to A} \right) \tag{4.3}$$

where $\mathcal{V}_A$ and $\mathcal{V}_B$ denote the sets of valid pixels in the segmented regions of images A and B, respectively. The network is trained with a batch size of 8, for 50 epochs with Adam optimizer and linear learning rate decay from 1e-3 to 3e-5. Training is carried out on 10 satellite models while the image chains from the remaining 4 models are kept out for testing.

At inference, the model processes pairs of images without requiring known transformations between them. Given two input images, they are passed through the feature matcher to obtain dense descriptors $\mathbf{O}_A, \mathbf{O}_B \in \mathbb{R}^{76 \times 76 \times 64}$. These descriptors are flattened and L2-normalized, yielding $\hat{\mathbf{O}}_A, \hat{\mathbf{O}}_B \in \mathbb{R}^{N \times 64}$ where $N = 76^2$.

The similarity matrix is computed using Equation (4.1). To ensure robust matching, mutual nearest neighbors are identified: for each spatial location i in image A, the best match in image B is found as $j^* = \arg\max_j \mathbf{S}_{ij}$, and similarly for each location j in image B, the best match in image A is $i^* = \arg\max_i \mathbf{S}_{ij}$. A correspondence (i, j) is retained only if both conditions hold: $j = j^*$ and $i = i^*$, ensuring bidirectional consistency. Since the feature maps have resolution 76×76 while the input images are 300×300, each feature location represents a receptive field of approximately 4×4 pixels in the original image. To obtain pixel-level correspondences, the matched coordinates are mapped back to image space by scaling: a feature at position (x_f, y_f) corresponds to pixel position $(x_f \cdot 300/76, y_f \cdot 300/76)$ in the input image. The mutual correspondences are ranked by their similarity scores, and the top-k matches are retained.

Figure 4.6 presents a montage of image pairs with their associated pixel correspondences from four sequences in the holdout validation set. Each pair consists of an original image and its 30° rotated counterpart, with the top-15 mutual matches displayed. The results demonstrate that the segmentation mask effectively guides the model to focus on foreground regions, as evidenced by the concentration of

correspondences within the object boundaries while ignoring the background.

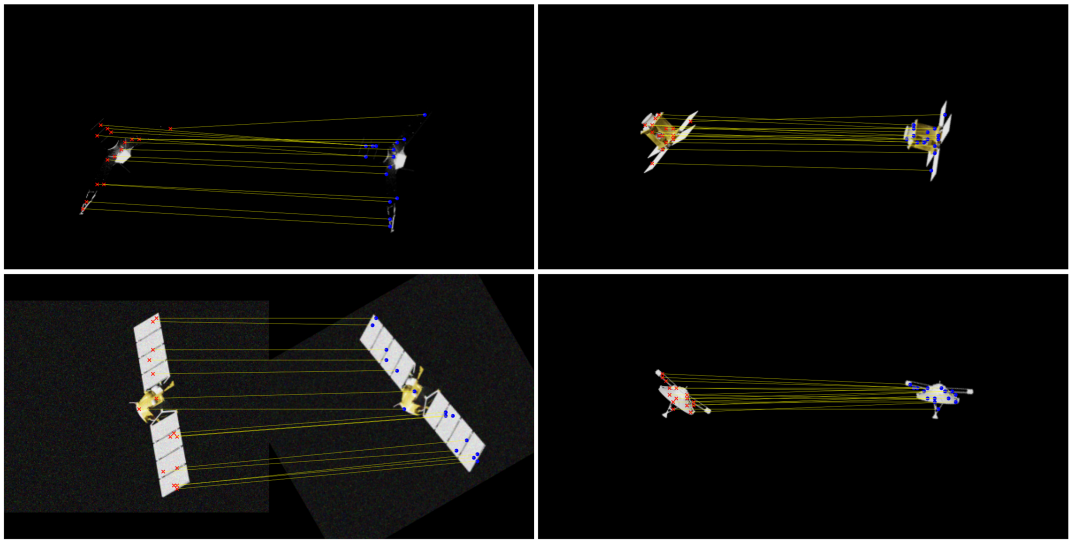

Figure 4.6: Feature matching between original and 2D-augmented image pairs; yellow lines connect correspondences across viewpoint and photometric transformations

4.2.3 Relative pose estimation pipeline

The patch classifier and feature matcher are integrated to build a frame-to-frame pose estimation pipeline to be tested on the validation dataset image chains. Given the top-k matched feature pairs between consecutive images, the essential matrix is computed using OpenCV's `findEssentialMat` function with RANSAC and the camera intrinsic matrix $\mathbf{K}$. The essential matrix encodes the epipolar geometry between two camera views.
The `recoverPose` function then extracts the relative pose between consecutive frames. This function returns the transformation $\mathbf{T}_{10}$ that maps points from frame 0 to frame 1:

$$\mathbf{T}_{10} = \begin{bmatrix} \mathbf{R}_{10} & \hat{\mathbf{t}}_{1\to 0|1} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{4.4}$$

However, the problem can be solved up to scale, therefore the translation vector $\hat{\mathbf{t}}_{1\to 0|1}$ is a unit vector ($\|\hat{\mathbf{t}}_{1\to 0|1}\| = 1$) that indicates only the direction of motion. By chaining these frame-to-frame transformations, the pose of the reference frame 0 relative to frame i is obtained through sequential composition:

$$\mathbf{T}_{i0} = \mathbf{T}_{i,i-1} \cdot \mathbf{T}_{i-1,0} \tag{4.5}$$

Since the competition requires the transformation from frame i to frame 0 (rather than from 0 to i), the final pose components are computed as:

$$\mathbf{R}_{0i} = \mathbf{R}_{i0}^{T} \tag{4.6}$$

$$\hat{\mathbf{t}}_{i\to 0|0} = \mathbf{R}_{0i} \cdot \hat{\mathbf{t}}_{i\to 0|i} \tag{4.7}$$

where $\hat{\mathbf{t}}_{0\to i|i}$ is extracted from $\mathbf{T}_{i0}$.
The resulting pose errors are high, in line with the competition results, where top-performing methods have achieved only 5% improvement over the trivial baseline solution that predicts no pose changes. The sequential nature of pose composition accumulates errors from each individual transformation, and when the matcher fails at a given point in the sequence, all subsequent pose estimates become unreliable yet still contribute to the average. Despite these limitations, which are partly inherent to the dataset's nature that constrains validation possibilities, the feature-matching results are promising and the approach provides a foundation for further refinement and fine-tuning.

5

Development of a robotic HIL testbed

This section introduces a new robotic testbed, installed at the Microsatellite and Space Microsystems Laboratory of the University of Bologna as part of this work, and provides an overview of the software integration, commissioning, and calibration procedures. The testbed offers a realistic environment for spacecraft pose estimation with hardware-in-the-loop testing and plays a key role in the laboratory's development of the vision-based navigation research area, opening new opportunities for collaboration in this field. The final sections describe an automated dataset generation pipeline implemented as the first application of the testbed. By featuring an almost perfectly symmetric target, reflecting the design of many satellites, this mockup adds an extra element of complexity to pose estimation compared to benchmarks datasets like SPEED+. The material reported in this chapter appeared in "Development of a Robotic Testbed and Initial Testing for Machine Learning-Based Spacecraft Pose Estimation" (Lotti et al., 2025a).

5.1 Overview of the facility

The facility design is the result of a trade-off between technical requirements, available space, which constrained the maximum footprint, and available budget. The testbed includes a collaborative robot, featuring 6 degrees of freedom and a 1.3m reach, mounted on a motorized linear axis with a 1.8-meter stroke, resulting in a 7-DoF robotic setup. An OptiTrack motion capture system with four cameras is installed to provide ground-truth pose measurements. The facility also includes an optical table, a spacecraft mockup, a sun simulator, an RGB camera with a 8mm focal length lens, and a dark room measuring 4.4×2.0×2.2m (length, width, height), covered by a polyester blackout curtain with an acrylic backing. A detailed list of the components is reported in Table 5.1, a view of the facility is provided in Figure 5.1, and a CAD assembly is shown in Figure 5.5.

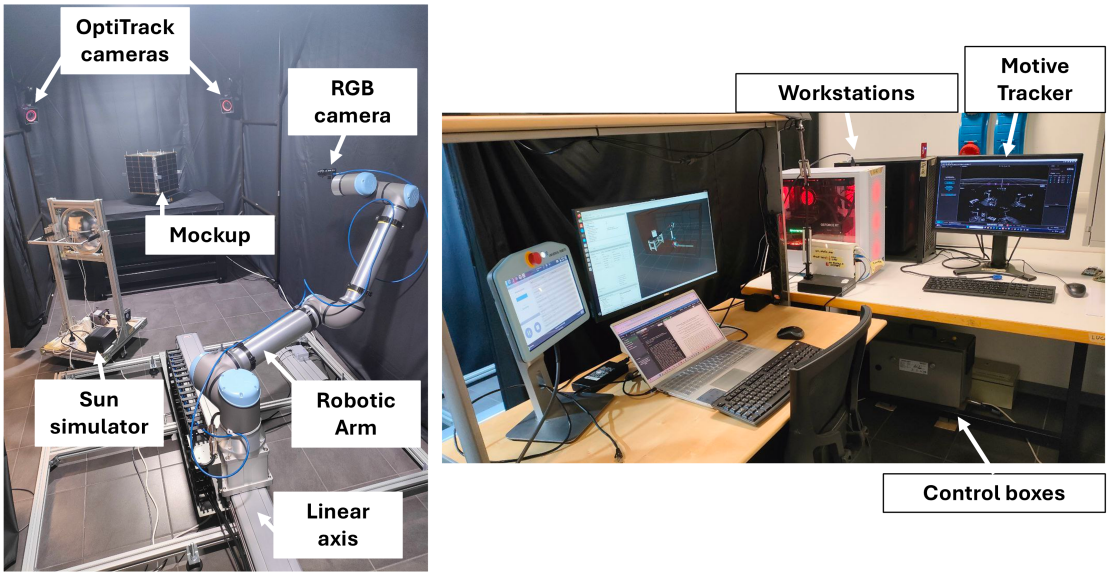

Figure 5.1: View of the robotic testbed facility at the Microsatellites and Space Microsystems Laboratory; dark room with labeled hardware components (left), control area with workstations and control boxes (right)

While collaborative robots are designed for use with humans in the loop, and this application primarily relies on scripted programming, a UR collaborative robot was still chosen for three main reasons:

1. **Safety**: given the educational context of the facility, safety is a key aspect. Collaborative robots are inherently safe, as they are designed to stop upon

contact with obstacles or people. Although the 7th axis is not collaborative and does not stop in case of impact, its movement range is limited to a single degree of freedom.

2. **International collaboration**: several partner universities already use or are adopting the same cobot. This common platform is supposed to facilitate international collaboration and exchange.

3. **Integration with ROS**: the UR robot is supported by stable and well-documented drivers, which simplify integration into ROS-based workflows.

Table 5.1: Main components of the robotic testbed (Lotti et al., 2025a)

Component	**Model**	**Specifications**
Robotic arm	UR10e (Universal Robots)	6-DoF, 1.3 m reach, 12.5 kg payload, repeatability ± 0.05 mm (3σ)
7th axis (linear)	Slidekit 2.0 (Ewellix)	Ball screw drive, 1.8 m stroke, repeatability ± 0.1 mm (3σ)
Optical table	PFR60120-7 frame + B60120AX breadboard (Thorlabs)	$1.2 \times 0.6 \times 0.76$ m, high rigidity frame with damped optical surface
Spacecraft mockup	ALMASat engineering model	$33 \times 33 \times 31.5$ cm
Sun simulator	Custom	Metal halide lamp (HMI)
Camera	Blackfly S BFS-U3-23S3C-C (Teledyne FLIR)	IMX392 CMOS sensor, 1920×1200 px, 8 mm lens
Motion capture cameras	(4×) PrimeX 22 (OptiTrack)	Sub-millimeter accuracy, 6-DoF pose tracking
Dark room	Custom (OptoSigma)	$4.4 \times 2.0 \times 2.2$ m, polyester blackout curtain with acrylic backing

The dark room is designed to block environmental light, allowing finer control of lighting conditions, and providing a uniform background which is easier to handle in image post-processing. For the same reason, the optical table has been covered with a black aluminum foil to minimize secondary reflections.

5.1.1 Custom Sun simulator

Realistic illumination is provided by a custom-built Sun simulator (Figure 5.2), which consists of a metal halide bulb and a parabolic reflector to emulate the collimated light of the Sun. The Sun simulator was assembled using testing and development materials previously employed in the characterization of components and ray-tracing simulations realized for the NanoDyna ADCS testbed facility (Curatolo et al., 2023), where the author was responsible for the development of the Sun simulator. The main components of the current setup are listed in Table 5.2. Light collimation is achieved by placing the bulb's arc at the focus of the parabolic reflector, located at the its open end. The divergence angle, measured using an industrial camera and a 75 μm pinhole, is estimated in 0.52° at approximately 4 cm from the center. Figure 5.3 shows that the light spot remains fixed as the camera is moved. The circular shadow is due to the light bulb's enclosure, which reduces irradiance on the frontal portion.

Figure 5.2: Custom Sun simulator with metal halide bulb and parabolic reflector, mounted on a mobile support structure

While the reflector geometry is not optimized for uniform light distribution and it lacks high-performance optical coating, the simulator still fits its purpose for photographic rendering. This is mainly due to the bulb's color temperature of 6200K and the absence of UV filtering. Based on previous design heritage, this

configuration remains one of the most cost-effective solutions for approximating the spectral characteristics of real sunlight. The irradiance at 1.5m is about 1200 W/m^2 as measured by an ISM 400 light meter.

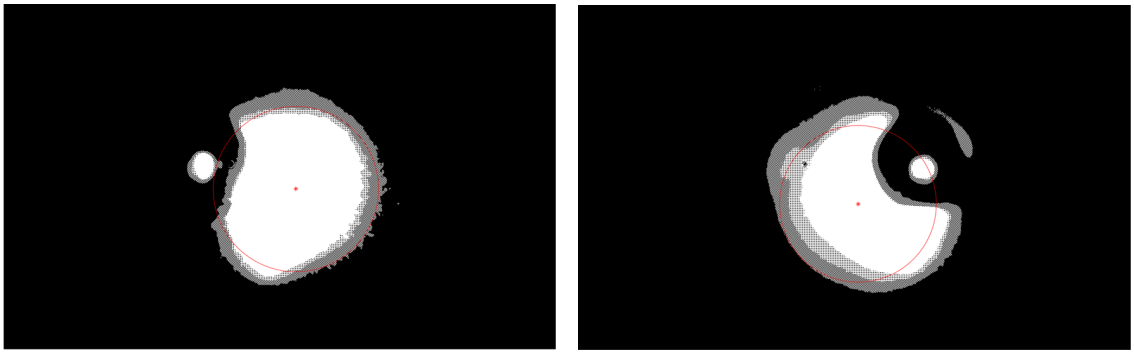

Figure 5.3: Light spots captured through a 75 μm pinhole at two camera positions, approximately 4 cm left (left) and right (right) from the reflector center, demonstrating beam collimation with an estimated divergence of $\approx 0.52°$. The central shadow is caused by the bulb enclosure

The simulator is mounted on a mobile support structure equipped with wheels, allowing it to be repositioned within the dark room. The reflector plane can be tilted to adjust the illumination direction, resulting in four degrees of freedom: two translations and two rotations. The bulb is positioned at approximately 85 cm from the ground.

Table 5.2: Sun simulator components

Component	**Model / Description**	**Key Specifications**
Bulb	BA 400 SE (Sylvania)	HMI, 400 W, 6200 K CCT, no UV filter
Reflector	Parabolic Reflector (Edmund Optics)	Aluminum-coated, 12” diameter
Ballast	Magnetic ballast for HMI lamps (Amlux)	Up to 575 W, 230 V, 6.8 A
Ignitor	NI 1000 LE lamp ignitor (B,A,G electronics)	For 400–1000 W lamps, 3–5 kV, 10 A
Lamp holder	(Amlux)	GX9.5
Capacitor	(Amlux)	60 μF
Support structure	Custom mobile base with tilt adjustment	4 DoF: 2 translations + 2 rotations, bulb at $\sim$85 cm height

5.1.2 Spacecraft mockup

The ALMASAt engineering model (Tortora, 2009) is adopted as the spacecraft mockup. The microsatellite measures approximately 33 × 33 × 31.5 cm and has a nearly perfectly symmetric design. The main bus is made of aluminum and includes an attitude control subsystem that extends beyond the bus structure. However, the mockup lacked realistic solar panels. To simulate them, aluminum composite panels (alupanels) were used and coated with a copper backing to replicate the visual appearance of printed circuits. The solar cells were emulated using glossy black vinyl, applied manually with the aid of a 3D-printed positioning mask. The solar cells size and geometry match the ones of the real satellite. The panel borders were finished with aluminum tape and Kapton tape to enhance realism. A CAD model of the mockup was reconstructed based on the physical assembly (see Figure 5.4).

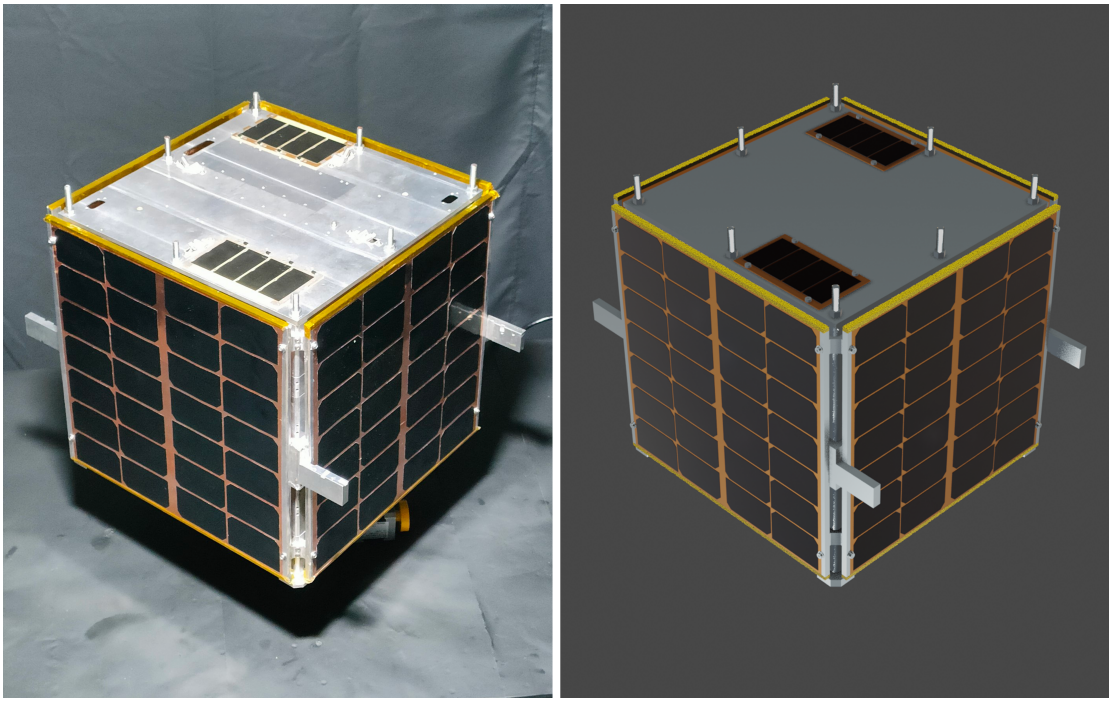

Figure 5.4: ALMASat mockup used as target spacecraft (left) and corresponding CAD model in Blender (material preview) for synthetic image generation (right)

In the standard configuration, the cobot's end-effector holds the camera, allowing it to move around the dark room environment while the ALMASat mockup is fixed on the optical breadboard. However, the setup also supports an inverted

configuration: since the mockup weighs approximately 8.4 kg, well within the UR10e's 12.5 kg payload capacity, it can be mounted on the end-effector while the camera is fixed on the optical table. This arrangement enables the simulation of complex target dynamics.

5.2 Software integration

A central workstation running Ubuntu 24 coordinates the robotic system using ROS 2 Jazzy (Macenski et al., 2022). It interfaces with the robotic arm, the linear axis, and the camera, enabling synchronized control and data acquisition. A secondary workstation running Windows operates the OptiTrack's Motive software. Specifically, the main workstation is connected to:

- the robot control box via Ethernet,
- the linear axis via a CAN-USB adapter,
- the camera via USB 3.2,
- the Windows workstation via Ethernet, streaming motion capture data using the NatNet protocol.

The robot control box also shares the CAN connection to the linear axis, allowing both the workstation and the robot to access it. The seven joints, i.e., the 6-DoF robotic arm combined with the linear axis, are coordinated using MoveIt 2 (Coleman et al., 2014), a motion-planning software that handles Inverse Kinematics (IK), trajectory planning, obstacle avoidance, and motion execution within the ROS 2 ecosystem. The components of the facility, along with their spatial arrangement, were modeled using SolidWorks (see Figure 5.5). They were then exported in a URDF (Unified Robot Description Format) file, an XML-based format in ROS used to describe a robot's kinematic structure, geometry, and physical properties, enabling motion planning, collision free path planning, and simulation. A visual representation of the real-time scene in RViz is shown in Figure 5.6.

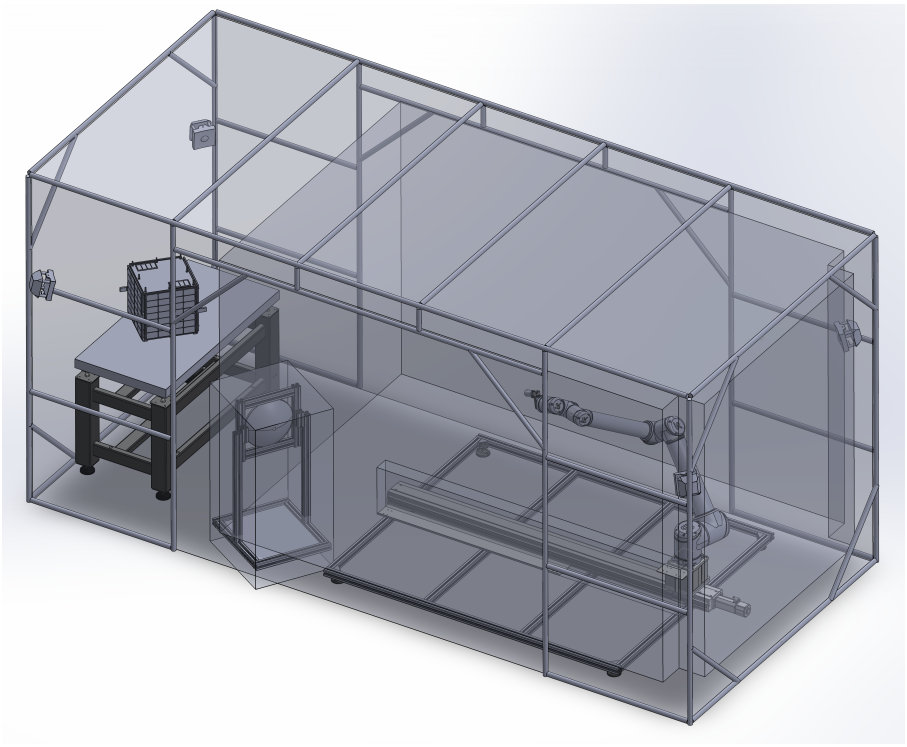

Figure 5.5: CAD model of the robotic testbed assembly, used to export meshes to build the URDF file and RViz scene

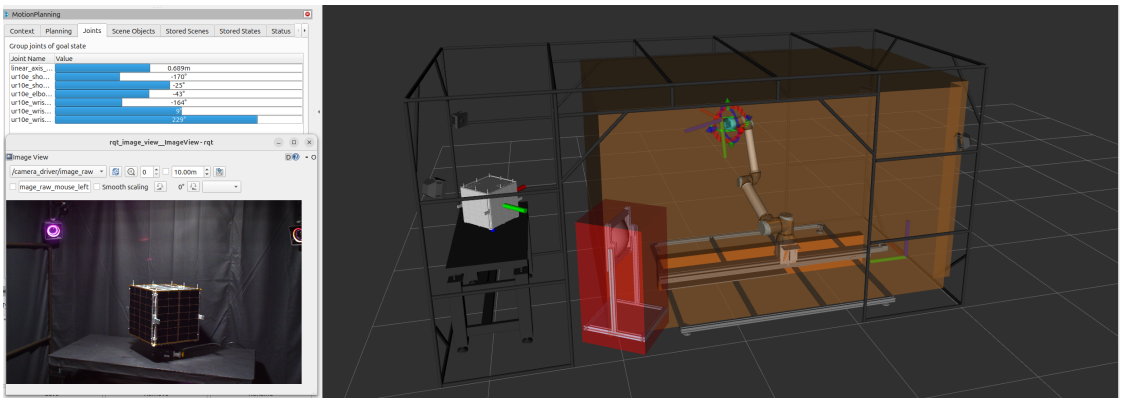

Figure 5.6: RViz visualization of the robotic system showing the safe volume for collision-free path planning (orange), Sun simulator volume (red), and principal reference frames (laboratory, camera optical, ALMASat body). The left panel displays joint sliders for real-time control and the live camera stream

5.2.1 Linear axis integration

While the UR cobot can be easily integrated into ROS using its driver and the external control plugin[1], the integration of the seventh axis required a custom implementation. The Ewellix linear axis interfaces well with UR software through the URCap ecosystem. This integration is effective in industrial environments, where repetitive tasks can be programmed via the teach pendant by setting waypoints, speeds, and accelerations from the GUI. However, this setup is not ideal for the testbed, where the linear axis must be coordinated with the robot and therefore must be directly visible to ROS. Furthermore, when using URCap, linear axis movements block the cobot and vice versa, which may limit future development of the facility. Hopefully, the axis features a BG 65SX50 MI Dunkermotoren drive that can be commanded via CANopen, following the CiA 402 standard (CAN in Automation (CiA), 2022). The first step was therefore to split the CAN connection to reach the workstation. Position is commanded in terms of "counts". Intuitively, counts can be seen as the encoder's "steps", where the number of counts per revolution is set by the encoder's resolution, and each count corresponds to the smallest angular movement that can be measured. The conversion factor between the linear position of the guide and the motor encoder counts was determined experimentally, yielding a value of 3276.8 counts/mm. The position control sequence, detailed in Table 5.3, consists of three steps: first, the target position is written to the drive's position register (object 0x607A) in encoder counts; second, the drive is enabled and prepared for motion; and finally, the movement is triggered by setting the new setpoint flag in the controlword. At the current stage of development, only position commands are sent through ROS. The axis remains configurable through the robot's teach pendant, which can still be used to set axis speed and acceleration parameters and controls a relay that switches the motor drive on or off together with the cobot's control box.
To integrate the linear axis into the ROS 2 ecosystem, a dedicated node was developed that acts as a bridge between CANopen communication and the standard ROS interface. The node subscribes to position commands through the `/linear_axis_controller/command topic`, which accepts Float64 messages representing the target position in meters. For low-level CAN communication, the node utilizes system calls to `cansend` for transmitting commands and `candump` for reading re-

[1] UR ROS2 Documentation: Robot setup

Table 5.3: CANopen motion control commands, according to the CiA 402 Profile Position Mode

Command	CAN Frame	Object	Value	Function	Description
Set Target Position	67F#237A6000 [POSITION]	0x607A	Variable	Target Position	Sets desired position in counts. [POSITION]: 4 bytes, little-endian, signed int32
Enable Operation	67F#2B4060000 F000000	0x6040	0x0F	Controlword	Enables drive operation and clears bit 4 (new setpoint flag)
Start Movement	67F#2B4060001 F000000	0x6040	0x1F	Controlword	Sets bit 4 to trigger movement execution to target position

sponses. Position feedback is obtained by periodically querying the drive's actual position (CANopen object 0x6064). The axis state is broadcasted on two ROS topics:

- `/linear_joint_states_real` (`sensor_msgs/JointState`): publishes the current position in meters, maintaining compatibility with standard ROS joint representations.
- `/linear_axis/status` (`geometry_msgs/Vector3`): provides operational state information where: x indicates movement status (1.0 if moving, 0.0 if idle), y shows whether the target position has been reached within tolerance, and z reports the absolute position error in millimeters.

Additionally, to enable MoveIt integration, a separate controller node was developed that implements the `FollowJointTrajectory` action interface. The `LinearAxisMoveItController` exposes a `FollowJointTrajectory` action server at the

topic `/linear_axis_controller/follow_joint_trajectory`. This is the standard interface expected by MoveIt for trajectory execution, allowing the linear axis to be integrated into multi-axis motion planning.
The controller implements two callbacks:

- Goal callback: accepts new trajectory requests only when no execution is in progress, preventing command conflicts.
- Execute callback: processes the trajectory and manages the execution lifecycle.

When MoveIt sends a trajectory:

1. The controller extracts the final target position from the trajectory points. Since at this stage, the linear axis node only supports position control, it simplifies the trajectory to a point-to-point movement.
2. The target position is published to `/linear_axis_controller/command` as a Float64 message.
3. The controller subscribes to `/linear_axis/status` to receive real-time feedback about: movement state, position reached status, and current position error in millimeters.

The controller returns FollowJointTrajectory results code: `SUCCESSFUL`, if the target position is reached within tolerance (0.1 mm), `GOAL_TOLERANCE_VIOLATED`, and `INVALID_GOAL`.

5.2.2 ROS 2 system architecture

The system's ROS 2 architecture can be described using a layered model, as illustrated in Figure 5.7.

- **User interface and 3D visualization**: the topmost layer provides the human-machine interface through RViz. This layer abstracts system complexity behind an intuitive 3D graphical interface where operators can monitor the robot configuration, and execute commands. The camera viewer provides real-time visual feedback from the testbed.

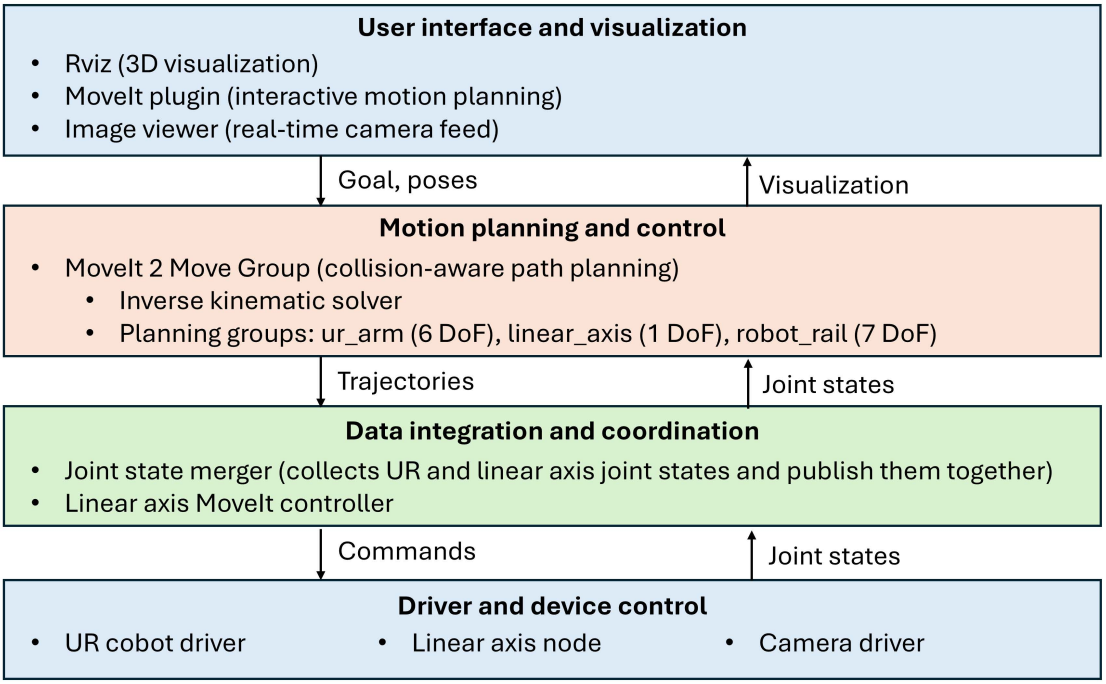

Figure 5.7: Layered ROS 2 architecture of the robotic testbed, comprising user interface and visualization (top), motion planning via MoveIt 2 with three planning groups, data integration for joint state coordination, and hardware drivers (bottom). Arrows indicate the flow of commands and state feedback between layers

- **Motion planning and control layer**: MoveIt 2 operates and coordinates the 7 joints, accounting for the robot's kinematic constraints, joint limits, and obstacle avoidance. Three planning groups are configured to control the linear axis and the robotic arm both independently and jointly.
- **Data integration layer**: this layer bridges the gap between heterogeneous hardware and the homogeneous interfaces expected by the planning layer. Specifically, a ROS node called `joint_state_merger` collects joint states from the UR cobot driver (published on `/joint_states`) and from the linear axis node (published on `/linear_joint_states_real`), merges them, and republishes them as `/joint_states_combined`, as required by the MoveIt planner.
- **Driver layer**: implements ROS nodes for each hardware component, handling device-specific protocols and communication.

The robotic system is started through a shell script which orchestrates the initial-

ization sequence through the following steps:

- Verifies connectivity to the UR10e robot.
- Executes a Python script to power on the UR10e and release the brakes.
- Configures the can0 interface at 125kbps bitrate for linear guide communication.
- Sends a sequence of CANopen commands to wake up the Dunkermotoren drive executing the CiA 402 state machine sequence (shutdown, switch on, enable operation) to enable the motor.
- Starts the ROS 2 UR driver node.
- Switches the UR controller to active state via ROS 2 service call.
- Launches the custom node that handles CANopen communication for the linear axis.
- Starts the joint state merger node.
- Initializes the controller that adapts MoveIt trajectory actions to linear axis position commands.
- Launches MoveIt move groups for inverse kinematics, and 7-DOF collision-free path planning.
- Starts RViz with MoveIt plugin for 3D visualization and interactive control.
- Loads the URCap external control program on the robot to enable ROS command execution.
- Activates the FLIR camera publisher for image acquisition and the image viewer for visual feedback.

5.3 Calibration procedures

MoveIt allows setting target poses for the end effector and automatically computes the joint values needed to reach them via inverse kinematics. It relies on TF2

(Foote, 2013), a ROS 2 library that tracks coordinate frame relationships, to transform poses between different reference frames. In the context of this facility, it is of interest to define the relative pose between the camera and the target. This requires MoveIt to know precisely how the camera is mounted on the robot. Figure 5.8 illustrates the kinematic chain used in this ROS 2 setup. Since the linear axis structure may be subject to small displacements or misalignments with respect to the lab frame, a fixed reference frame called `robot_home` was introduced. The linear axis is rigidly constrained to its support structure, and `robot_home` is defined on this structure, perfectly coincident with the robot base link when the slider is at position zero. From this point forward, references to the "world" frame indicate this `robot_home` frame rather than the `lab_frame`. Consequently, the pose of the satellite mockup was also estimated relative to `robot_home`, ensuring consistent spatial referencing throughout the system.

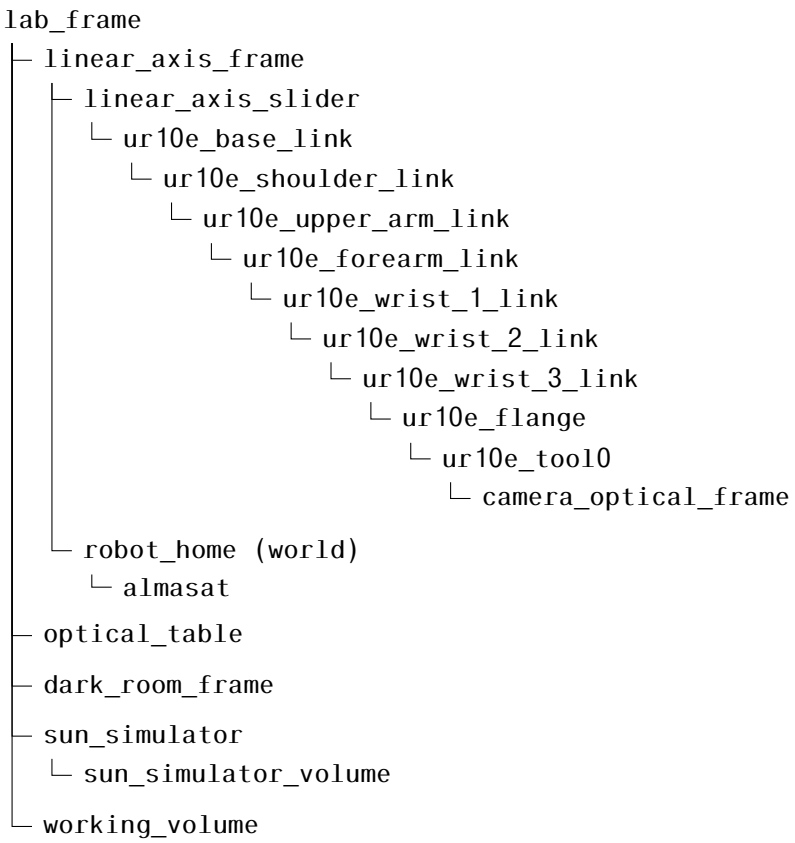

Figure 5.8: TF tree of the robotic testbed showing the hierarchy of coordinate frames from the lab reference frame to the camera optical frame and static elements

TF2 maintains a tree of coordinate frame transformations. The `robot_state_-`

publisher node parses the URDF model and subscribes to joint states, publishing the resulting transforms to TF2, which allows querying the pose of any frame relative to any other. However, the initial URDF, created by manually measuring distances after the facility was assembled, with the help of the SolidWorks model, is subject to errors, in the order of centimeters and degrees. While such errors are generally acceptable for high level modelling and obstacle avoidance (with sufficient clearence), they are too large when accurate camera poses are required. Several calibrations are therefore needed to ensure that all key frames are correctly referenced to the common robot_home frame. The main calibrations performed include:

- Camera intrinsic calibration.
- Hand-eye calibration.
- Calibration of the mockup pose within the world frame.

At the end of calibration, the world-to-almasat and ur10e_tool0-to-camera_optical_frame transformations are updated and the environment is re-compiled.

5.3.1 Camera calibration

Intrinsic calibration was performed using a 6×9 checkerboard with 25 mm squares. The checkerboard was mounted on the robotic arm's end effector, while the camera remained fixed (see Figure 5.9). To ensure comprehensive coverage of the camera's field of view at multiple distances, poses were generated automatically using a spherical grid sampling strategy. A grid of 4×3 angular positions (horizontal × vertical) was sampled at three distances (1.3m, 1.8m, 2.5m) from the camera, with random checkerboard tilts applied at each position to improve distortion estimation. Each candidate pose was validated through inverse kinematics and collision checking via MoveIt before inclusion. This resulted in 33 validated poses. The joint positions associated with these poses were recorded and subsequently replayed by a script that sequentially executed each pose while capturing and saving the corresponding image.
The images were then processed using MATLAB's Camera Calibration Toolbox, which returned the intrinsic parameters listed in Table 5.4. The resulting mean per-image reprojection error is approximately 0.07 px, with similar errors across all images.

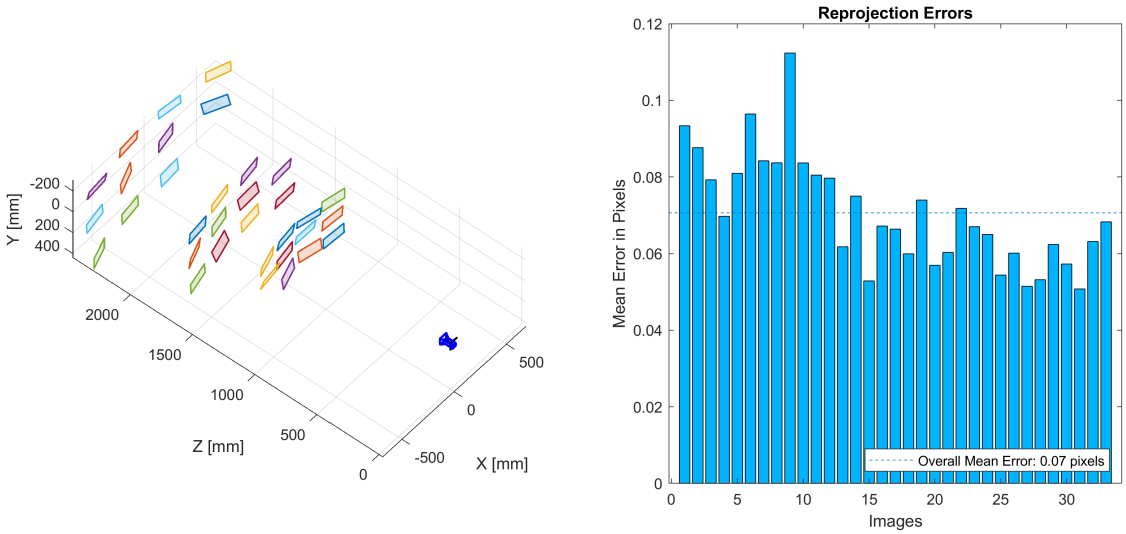

Figure 5.9: Checkerboard poses used for camera intrinsic calibration, distributed on a spherical grid within the camera field of view at three distances (left), and per-image reprojection errors with a mean of 0.07 px (right)

Table 5.4: Estimated camera intrinsic parameters

Parameter	**Value**	**Std. Error (1σ)**
Focal length (f_x, f_y) [px]	[2395.8, 2397.9]	[7.53, 7.45]
Principal point (c_x, c_y) [px]	[1002.5, 581.1]	[2.94, 3.16]
Radial distortion (k_1, k_2)	[-0.1152, 0.2560]	[0.0020, 0.0087]
Tangential distortion (p_1, p_2)	[-0.0024, 0.0003]	[0.0002, 0.0003]

5.3.2 Hand-eye calibration

Hand-eye calibration consists of estimating the transformation between the camera's optical frame (c) and the robot's end effector (g):

$$X = \mathbf{T}_{gc} = \begin{bmatrix} \mathbf{R}_{gc} & \mathbf{t}_{g \to c|g} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{5.1}$$

This is done by collecting multiple paired measurements of:

- the robot's pose (using forward kinematics) in a base frame (b), and
- the camera's pose relative to a fixed calibration target (t)

From these measurements, relative motions between successive time steps i and i+1 are computed:

$$A_i = \mathbf{T}_{bg,i}^{-1}\mathbf{T}_{bg,i+1} \tag{5.2}$$

$$B_i = \mathbf{T}_{ct,i}\mathbf{T}_{ct,i+1}^{-1} \tag{5.3}$$

Where A_i is the end-effector motion between two time steps i and $i+1$, expressed in the end-effector frame at time i, and B_i is the camera motion between the same time steps, expressed in the camera frame at time i. These relative motions are used to formulate the $AX = XB$ equation:

$$A_i\mathbf{X} = \mathbf{X}B_i \tag{5.4}$$

This expresses a frame-consistency condition: starting in the camera frame at time $i+1$, applying the fixed transformation X to the end-effector frame and then the end-effector motion A_i must yield the same result as first applying the camera motion B_i and then transforming to the end-effector frame via X. This ensures that the camera and end-effector motions are consistent through the fixed, unknown transformation X. This formulation holds for the eye-in-hand configuration, i.e. when the camera is mounted on the end effector and is looking at a fixed calibration target.

Specifically, the calibration process is implemented according to the following steps, using a dedicated Python script:

- The robotic system is moved to a target pose by setting a goal in MoveIt and captures an image of the chessboard.
- Checkerboard corners are detected using OpenCV's `findChessboardCorners` and refined with `cornerSubPix`.
- For each robot pose, the target-to-camera transformation is computed using `solvePnP` with the IPPE algorithm (Collins and Bartoli, 2014), accounting for the chessboard's planarity. Simultaneously, the gripper-to-robot_-home transformation is obtained through TF2, and both transformations are recorded.
- Hand-eye calibration is performed using the OpenCV's function `calibrateHandEye`, based on the Tsai algorithm (Tsai and Lenz, 1989) through

OpenCV's `calibrateHandEye`.

- Calibration accuracy is verified by computing the consistency of the checkerboard pose in the world frame (w). Since the checkerboard remains stationary throughout the calibration, all computed world-to-target transformations should theoretically be identical:

$$\mathbf{T}_{wt,i} = \mathbf{T}_{wg,i} \cdot \mathbf{T}_{gc} \cdot \mathbf{T}_{ct,i} \tag{5.5}$$

- The average pose is computed, and the deviation of each candidate pose from the mean is evaluated in terms of translation error and geodesic angular error to check for outliers.

Hand-eye calibration poses were generated automatically using a spherical grid sampling strategy at three distances from a fixed checkerboard, with the camera always pointing toward the checkerboard center to ensure it remains within the field of view. This configuration provides perspective diversity from different viewpoints while additional roll variation around the optical axis ensures sufficient rotational diversity, which is critical for hand-eye calibration. Each candidate pose was validated through inverse kinematics and collision checking via MoveIt, resulting in 26 validated poses. The joint positions were recorded and replayed by an automated script that executed each pose while capturing the corresponding image. Hand-eye calibration was repeated three times to assess repeatability. Translation statistics were computed by direct averaging, while rotation statistics were computed using quaternion averaging. The results are reported in Table 5.5. After calibration, the `tool0_to_camera_optical` transformation was updated in the URDF file.

Table 5.5: Hand-eye calibration results for UR10e tool–camera optical axis transformation

	[mm]		**[deg]**
X	-1.23 $\pm$ 1.71	Roll	0.78 $\pm$ 0.05
Y	54.75 $\pm$ 4.20	Pitch	0.74 $\pm$ 0.06
Z	71.94 $\pm$ 5.05	Yaw	179.89 $\pm$ 0.10

5.3.3 Mockup pose calibration in the world frame

Once the transformation chain from the world frame to the camera frame was established, the camera was used to localize the mockup within the same reference frame. This ensures that whenever a target mockup–camera pose is requested by MoveIt, TF2 computes the accurate transformation through the common world frame. It then returns the corresponding tool pose to MoveIt, allowing it to handle the trajectory planning. A straightforward calibration procedure was adopted. The robotic system was commanded to a specific pose that ensured the mockup was clearly visible to the camera, with at least two faces in view to avoid coplanar points, which make the PnP problem more susceptible to error, and an image was captured. This image was manually annotated by selecting a set of visible landmarks, which were matched with 3D points extracted from the CAD model (see Figure 5.10).

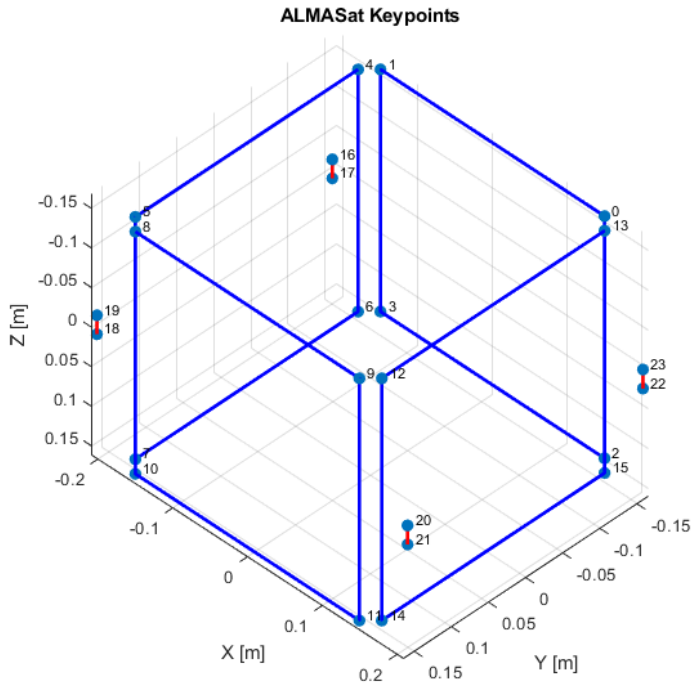

Figure 5.10: Wireframe model of the ALMASat mockup with labeled landmarks

The mockup's pose in the camera frame was estimated by solving the PnP problem. Its pose in the world frame was then obtained by composing the camera-to-world

transformation with the mockup-to-camera transformation:

$$\mathbf{T}_{wt} = \mathbf{T}_{wc} \cdot \mathbf{T}_{ct} \tag{5.6}$$

Figure 5.11 shows the annotated image, the selected landmarks, the corresponding reprojection errors, and the estimated body-fixed frame associated with the mockup.

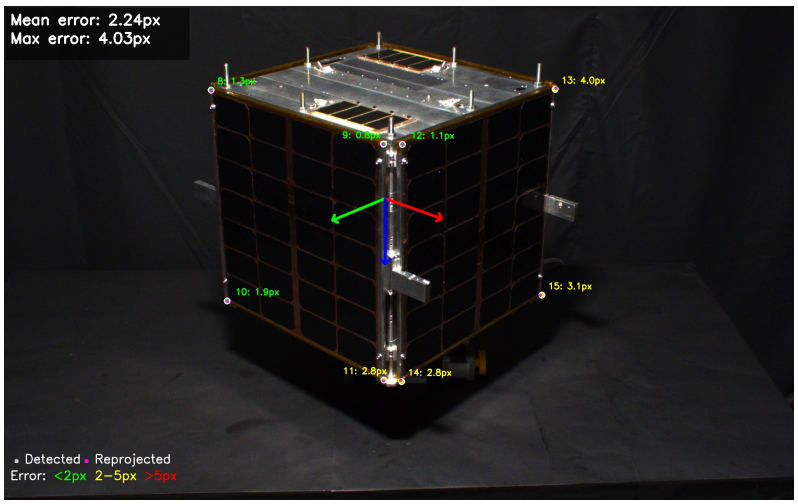

Figure 5.11: Annotated mockup image with 2D landmarks, reprojection errors, and estimated body-fixed frame

5.3.4 Repeatability assessment using OptiTrack motion capture

The positional repeatability of the 7-DoF robotic system was measured using the OptiTrack motion capture system as ground truth. The latter was calibrated using the Motive software, achieving a 3D calibration error of 0.125 mm. A single reflective marker was mounted on the robot end-effector, ensuring visibility from the OptiTrack cameras when the robot was in the target reference pose. The test followed a cyclic procedure: the system moved to a randomly generated joint configuration, then returned to the target pose. The marker position was recorded only when the robot reached a stable pose (after a 2-second stabilization period), as the marker could become temporarily occluded during motion due to varying end-effector orientations. A total of 33 successful trials were completed, with starting poses spanning a workspace of approximately $1.5 \times 0.6 \times 2.0$ m. Motion planning was performed using MoveIt with tight joint tolerances (in

this paragraph, the symbols $\pm$ indicate minimum / maximum values) for the target pose (± 0.1 mm for the linear axis and $\pm 0.0046°$ for the UR10e joints), while relaxed tolerances (± 2 mm and $\pm 0.57°$ respectively) were used for the random starting poses. Velocity and acceleration were limited to 20% of maximum values to ensure smooth motion. Figure 5.12 (left) presents a histogram of the Euclidean error from the mean target position, with a mean of 0.124 mm and standard deviation of 0.056 mm, yielding a $\mu + 3\sigma$ bound of 0.291 mm. The maximum observed error is 0.242 mm. These values approach the intrinsic accuracy of the OptiTrack system (0.125 mm calibration error). Figure 5.12 (right) provides a top view (XY plane) of the target pose dispersion, centered on the mean position. Each point is color-coded by its 3D error magnitude. The dashed circle represents the $\mu + 3\sigma$ boundary for the 2D distribution.

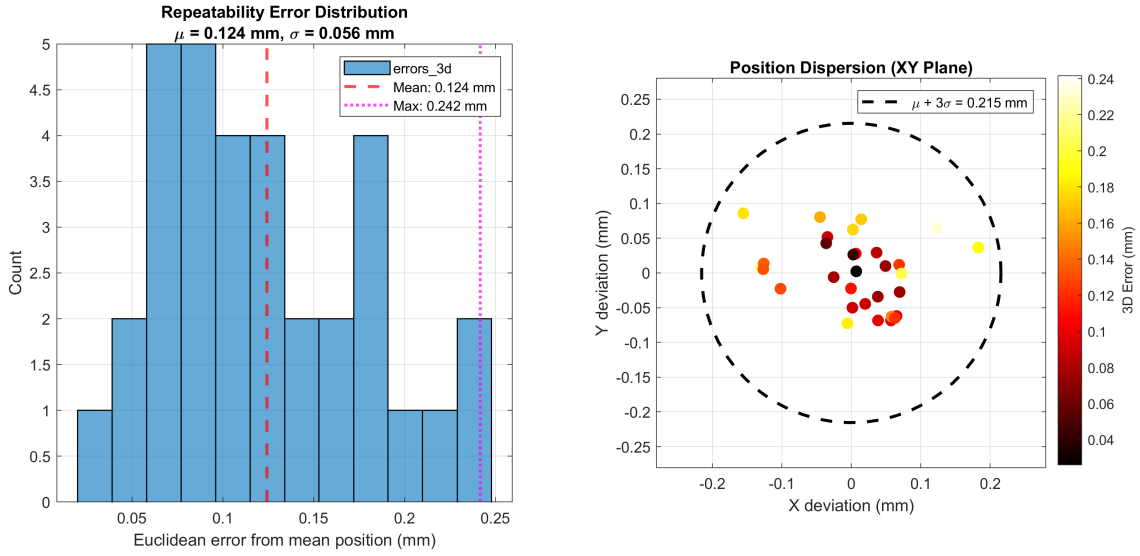

Figure 5.12: Positional repeatability of the 7-DoF robotic system measured via OptiTrack; distribution of Euclidean errors from mean position (left) and XY plane dispersion with 3σ boundary (right)

5.4 ALMASat dataset generation pipeline

The first application of the robotic testbed will consist in the development of a HIL dataset for validating pose estimation algorithms. The dataset consists of two partitions. The synthetic partition was generated in Blender using the CAD model (Figure 5.4). The real partition will be created by leveraging the testbed to

capture images of the ALMASat mockup from varied viewpoints with associated relative pose labels. The acquisition pipeline has been validated but full dataset capture will proceed after integrating also the OptiTrack cameras into the ROS workflow.

5.4.1 Synthetic images rendering

The synthetic dataset generation procedure is aligned with the one adopted by the author to develop the COSMO Photorealistic Dataset (Lotti et al., 2023b), except that no Earth background is rendered in this case. Sunlight is emulated by a Blender sun lamp, which provides light of constant intensity emitted in a single direction from infinitely far away. The intensity was set to 1062 W/m^2, the value reported in literature (Mehta et al., 2018) for the strength of sunlight in the visible spectrum impacting Earth-orbiting objects. Additionally, a blackbody node was used to convert the Sun's blackbody temperature of 5778 K to the corresponding RGB values. The image generation pipeline is controlled through Blender's Python API using the Starfish library (Schubert et al., 2021). Sensor and lens properties are adjusted to match the camera intrinsics in Table 5.4. The direction of sunlight with respect to the camera has been randomized across the dataset, subject to constraints that guarantee minimum illumination of the satellite, thus avoiding completely dark images. To this end, at least one illuminated surface of the box-shaped ALMASat mockup must be in view of the camera. A surface is considered illuminated if the sunlight forms an angle smaller than 80° with its normal. Similarly, a surface is considered in view of the camera if the angle between the surface normal and the camera position in the satellite's frame is smaller than 70°. For each scene setup, the sunlight is randomly rotated until both requirements are satisfied. The dataset consists of 20,000 images of the ALMASat spacecraft in different poses. To automate the image generation pipeline, the Starfish "sequence" class has been employed. This enables the creation of a set of poses and iterative scene setup by looping through the sequence. The camera-target distance is sampled from $\mathcal{N}(3.5\,\mathrm{m}, 1\,\mathrm{m})$, rejecting all values above 6 m and below 1 m. The attitude is randomly sampled from a uniform distribution of rotations in the SO(3) space. For each frame in the sequence, keyframes are inserted for the position and rotation quaternion of the satellite, camera, and sun lamp, enabling batch rendering of the entire sequence. Images have been

rendered using the ray-tracing Cycles render engine on an NVIDIA RTX 3090 GPU through command-line execution. A montage of images from the synthetic dataset is shown in Figure 5.13.

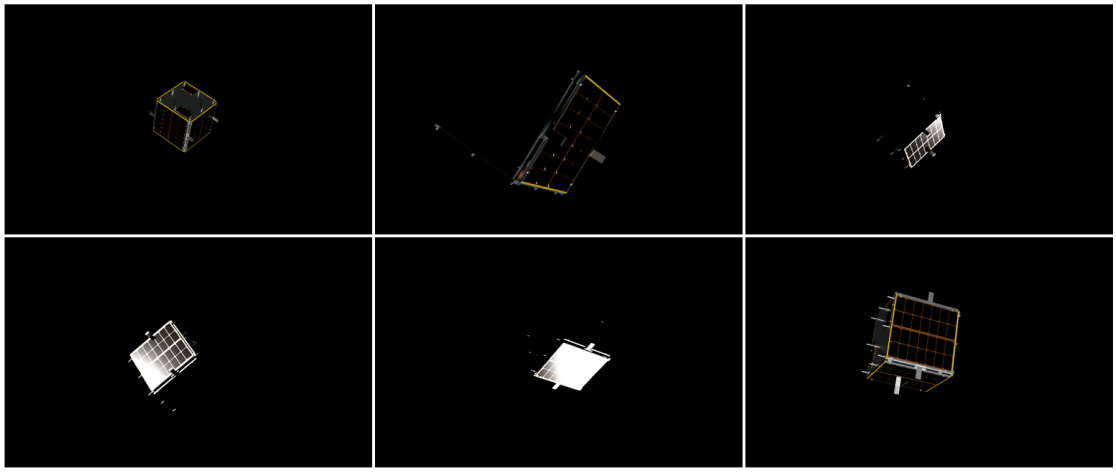

Figure 5.13: Representative sample of rendered images from the ALMASat synthetic dataset (Lotti et al., 2025a)

5.4.2 Real images acquisition

The process to collect real images involves three main steps: offline pose generation, online execution with image capture, and post-processing. The pose generation step is designed to maximize pose diversity and space exploitation, while keeping valid and feasible poses only. The workspace is divided into a grid structure where each cell centre point serves as a reference position for pose generation. For each grid cell, the algorithm generates multiple camera poses starting from a "look-at" configuration, where the camera optical axis is pointed towards the origin of ALMASat's body frame.

To create this look-at pose, the algorithm first retrieves the ALMASat's pose in the world frame ($\mathbf{T}_{wt}$) through a TF2 query. Using this target position and the grid cell centre as the (virtual) camera position, it constructs a transformation matrix ($\mathbf{T}_{wc}$) that places the camera at the cell centre with its Z-axis (optical axis) pointing toward the target.

From this central look-at pose, the algorithm verifies that all model keypoints (Figure 5.10) are visible within the camera's field of view. This validation requires multiple coordinate transformations: first, the keypoints defined in AlmaSat's

local coordinate system ($\mathbf{p}_t$) are transformed to world coordinates using the target-to-world transformation ($\mathbf{T}_{wt}$), then from world coordinates to the camera's coordinate system using the the world-to-camera transformation ($\mathbf{T}_{cw}$), as expressed by eq. (5.7).

$$\tilde{\mathbf{p}}_{c,i} = \mathbf{T}_{wc}^{-1} \cdot \mathbf{T}_{wt} \cdot \tilde{\mathbf{p}}_{t,i} \tag{5.7}$$

Once in camera coordinates, each 3D point is projected onto the 2D image plane through perspective projection (Eq. (2.15)). The algorithm checks that all projected points fall within the image bounds with a 100px margin. Then it verifies that the desired pose is kinematically reachable by calling MoveIt's inverse kinematics service. The IK request specifies the desired camera pose in the world frame, with the target link set to `camera_optical_frame`, collision checking enabled, and using the current joint states as the initial seed configuration. The solver attempts to find joint values for the 7-DoF system that would place the camera at the requested pose. If the initial attempt returns no solution, the algorithm retries with a set of predefined seeds, each representing different arm configurations that might provide alternative solution paths. The IK solver has a 1-second timeout per attempt and returns either a valid joint configuration that achieves the desired camera pose, or an error code indicating why the pose cannot be reached. Then, random variations from this look-at pose are generated by adding a small jitter in the camera position and randomly sampling a rotation axis and angle. Each pose undergoes the same validation pipeline. Upon completion, the algorithm saves two JSON files: one containing the camera-to-target relative poses, and one with the associated joint configurations ready for robot control.

Image capture is handled by a second script that loads the joint configurations. The script commands joint positions with extremely stringent tolerances to ensure maximum precision in image acquisition and repeatability. Joint tolerances are set to 0.1 mm for the linear axis and 0.0046° for the UR10e joints. For each pose in the sequence, a movement is initiated through MoveIt, configured to operate at 20% of the robot's nominal velocity and acceleration to minimize vibrations and overshoot. The script implements a robust retry strategy, allowing up to 5 attempts to reach each pose, with a 2-second interval between attempts. Once the movement is completed, the system samples joint states at 10 Hz, continuously

verifying that all values remain within the specified tolerances. The position is considered stable only when it remains within limits for at least 3 consecutive samples. After achieving stability, the script waits an additional second before proceeding with image capture to ensure further stabilization. At the moment of capture, TF2 queries are executed to determine the actual transformation between the camera and the ALMASat mockup, exporting both the raw image and the pose. This pose re-computation strategy after movement ensures cancellation of any residual error between commanded poses and achieved poses.
The final step consists of background removal which is implemented using the Segment Anything model (Kirillov et al., 2023). The network is guided through the ground truth bounding boxes around the target, which are retrieved by projecting the wireframe model (Figure 5.10) on each image using the ground truth pose. Sample images obtained with this procedure are illustrated in Figure 5.14.

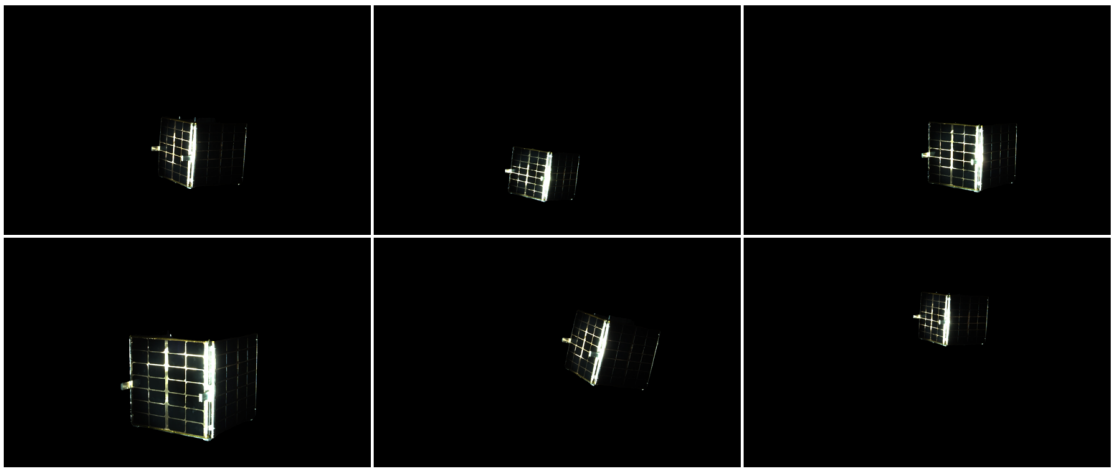

Figure 5.14: Representative sample of real images captured with the robotic testbed (Lotti et al., 2025a)

6

Conclusions

This thesis presented the outcomes of PhD research on monocular satellite pose estimation for close-proximity operations. The work starts by addressing three major challenges that hinder the deployment of deep learning-based pose estimation systems to space:

1. The domain gap between synthetic training data and real imagery.
2. The computational limitations of spaceborne hardware.
3. The lack of consolidated methods for uncertainty quantification in this context.

These challenges have been investigated in Chapter 3 through three complementary lines of research. First, domain adversarial training, hybrid CNN–Transformer architectures, and GAN-based augmentation were employed to mitigate the gap between synthetic training sets and real imagery, with a progressive reduction in the reliance on real images, up to configurations based solely on image augmentations. Second, a systematic study of embedded deployment on Edge TPU hardware quantified the trade-offs between accuracy, latency, and power consumption on synthetic imagery, and detailed the contribution of individual optimization steps. The results indicate that real-time, state-of-the-art performance (7.7 fps) can be achieved on smallsat-class boards (Goodwill et al., 2021) by

shifting the heavy image processing through neural networks onto dedicated co-processors. This is possible with a power consumption of only 2.2 W on an Edge TPU, although at the cost of architectural simplifications. Finally, Section 3.3 introduced Stochastic Variational Gaussian Processes to estimate uncertainty from neural feature map similarity. The method revealed a moderate-to-strong correlation between predicted uncertainty and actual errors on synthetic data, although this correlation diminished on real imagery. Despite these limitations, the SVGP classifier was still able to associate higher uncertainty values with Sunlamp images, which contained a large number of failures.

Chapter 4 extended the focus to unknown targets, demonstrating that the light-weight localization algorithms originally designed for known spacecraft could generalize to previously unseen satellites with only minimal architectural modifications. In addition, a complete pipeline for feature extraction, feature matching between image pairs, and camera pose estimation in inspection scenarios was developed, although pose-regression accuracy throughout the chain remains low. This is primarily attributed to failures of the feature matcher, currently trained on 2D image transformations that do not generalize well to 3D viewpoint changes. This is also a limitation of the dataset employed, which provides neither 3D models nor initial sequence poses, making robust 3D training unfeasible. This observation is confirmed by the results of the NASA PoseBowl competition (pose estimation track), where participants achieved only marginal improvements over the trivial solution.

Chapter 5 complemented these contributions with the development and commissioning of a 7-DoF robotic hardware-in-the-loop testbed for validating satellite pose estimation algorithms. The facility includes a purposely built Sun simulator, developed as part of this work, and a full-scale spacecraft mockup, finalized during this work to achieve a photorealistic appearance. A ROS 2-based control architecture was implemented, and the procedures for camera, hand-eye, and mockup pose calibration were described, together with a preliminary assessment of the repeatability of the system using OptiTrack cameras. As a first application, the testbed was employed to develop a pipeline to generate a dataset, which combines synthetic renderings and real images. This dataset will provide a challenging benchmark, featuring an almost symmetric target. Full image capture will be completed as soon as the ground truth system is fully integrated into the workflow, and the dataset will then be shared with the research community.

6.1 Future perspectives

Looking ahead, several research directions emerge from the work presented in this thesis. Concerning satellite pose estimation, future developments should aim at improving the calibration of uncertainty estimates on real imagery, as current results demonstrated a degradation compared to synthetic data. Additionally, the proposed uncertainty estimation procedure should be tested within a complete relative navigation architecture that includes a state estimation algorithm, where the estimated covariance must be consistent with actual measurement errors to ensure proper filter tuning. Furthermore, the current sampling-based approach for propagating keypoint uncertainty to pose estimates could be replaced by analytical covariance propagation through the PnP solver, potentially reducing computational cost.

In relation to camera pose estimation around unknown targets, there remains substantial room for improvement. First, the feature matcher must be made more robust to 3D viewpoint changes. This could be achieved by training on matches between different frames rather than relying on a self-supervised approach. However, as mentioned, this is not feasible within the PoseBowl dataset because the labels provide limited information, including the absence of 3D models and of the initial camera-satellite pose in each image chain. The creation of a dataset with ground-truth feature correspondences, enabling supervised learning of matches and the extraction of more robust features under varying perspectives and orientations, is therefore an area to explore. Second, both the backend and the validation must be improved. Regarding validation, the limitations are once more associated with the constraints of the dataset. Regarding the backend, the inclusion of bundle adjustment could strengthen overall accuracy, although its computational cost for space-qualified hardware must be carefully assessed. A further priority is the validation of the pipeline on real images.

Regarding the robotic testbed, given the relevance of this new facility, further developments are already planned. The OptiTrack camera system will be fully integrated within the ROS workflow to provide ground-truth pose data. While the ROS-MoveIt workflow developed in this work is convenient for setting target poses, and the overall system repeatability has been proven to be high, the current setup requires mounting and dismounting the camera to invert the camera-

checkerboard configuration whenever intrinsic or hand-eye calibration must be repeated, for example, if the focus flange is moved to change the depth of field. This process proved to be time-consuming. A straightforward solution is to accept residual errors in the hand-eye calibration (for instance by avoiding repeating it), use the last calibrated optical frame in the URDF to set target poses, and rely on the OptiTrack cameras to get ground truth poses by cancelling out the residual bias. It is also worth mentioning that the mockup is already mounted on a PI precision rotation stage, kept fixed during the development presented in this thesis, which could be integrated into the ROS workflow to provide an additional degree of freedom and single-axis spinning dynamics for the target. Moreover, there are already plans to procure diverse mockups. Additional cameras, already procured, will also be tested. These include a near-infrared camera and an event-based camera that could help reduce the domain gap by acting at the sensor level. In addition, GANs may be explored to generate a third partition of the dataset, labeled `sim2real`, allowing the virtual reproduction of the full range of relative poses that cannot be physically achieved within the facility's spatial constraints.

Bibliography

A. Amini, W. Schwarting, A. Soleimany, and D. Rus. Deep Evidential Regression. In *Advances in Neural Information Processing Systems*, volume 33, pages 14927–14937. Curran Associates, Inc., 2020.

H. Benninghoff, F. Rems, E.-A. Risse, and C. Mietner. European Proximity Operations Simulator 2.0 (EPOS) - A Robotic-Based Rendezvous and Docking Simulator. *Journal of large-scale research facilities JLSRF*, 3:A107–A107, Apr. 2017. ISSN 2364-091X. doi: 10.17815/jlsrf-3-155.

K. Black, S. Shankar, D. Fonseka, J. Deutsch, A. Dhir, and M. R. Akella. Real-time, flight-ready, non-cooperative spacecraft pose estimation using monocular imagery. In *Proceedings of the 31st AAS/AIAA Space Flight Mechanics Meeting*. AAS / AIAA, Feb. 2021.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Network. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1613–1622. PMLR, June 2015. ISSN: 1938-7228.

G. Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, Nov 2000.

Y. Bukschat and M. Vetter. EfficientPose: An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach, Nov. 2020. arXiv:2011.04307.

C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec. 2016. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2016.2624754.

CAN in Automation (CiA). CiA 402: CANopen device profile for drives and motion control. Technical Report CiA 402 version 4.1.0, CAN in Automation e.V., Nuremberg, Germany, 2022.

P. Carcagnì, M. Leo, P. Spagnolo, P. L. Mazzeo, and C. Distante. A Lightweight Model for Satellite Pose Estimation. In S. Sclaroff, C. Distante, M. Leo, G. M. Farinella, and F. Tombari, editors, *Image Analysis and Processing – ICIAP 2022*, volume 13231, pages 3–14. Springer International Publishing, Cham, 2022. ISBN 978-3-031-06426-5 978-3-031-06427-2. doi: 10.1007/978-3-031-06427-2_1. Series Title: Lecture Notes in Computer Science.

B. Chen, J. Cao, A. Parra, and T.-J. Chin. Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 2816–2824, Seoul, Korea (South), Oct. 2019. IEEE. ISBN 978-1-7281-5023-9. doi: 10.1109/ICCVW.2019.00343.

D. T. Coleman, I. A. Sucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of Software Engineering for Robotics*, 5(1):3–16, 2014.

T. Collins and A. Bartoli. Infinitesimal Plane-Based Pose Estimation. *International Journal of Computer Vision*, 109(3):252–286, Sept. 2014. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-014-0725-5.

P. Colmenarejo, M. Graziano, G. Novelli, D. Mora, P. Serra, A. Tomassini, K. Seweryn, G. Prisco, and J. G. Fernandez. On ground validation of debris removal technologies. *Acta Astronautica*, 158:206–219, May 2019. ISSN 0094-5765. doi: 10.1016/j.actaastro.2018.01.026.

A. Curatolo, D. Modenini, G. Curzi, A. Lotti, D. Pecorella, E. Strollo, A. Locarini, A. Hyslop, and D. Oddenino. The nanosatellite-class attitude control facility at esa-estec. In *Aerospace Europe Conference 2023 – 10th EUCASS – 9th CEAS*, page 12, 2023. doi: 10.13009/EUCASS2023-760.

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*, 2(2):171, 2014. ISSN 2048-8459, 2048-8467. doi: 10.1504/IJSPACESE.2014.060600.

P. David, D. DeMenthon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous Pose and Correspondence Determination. *International Journal of Computer*

Vision, 59(3):259–284, Sept. 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000025800.10423.1f.

D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 337–33712, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00060.

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*, 160:110055, May 2025. ISSN 1270-9638. doi: 10.1016/j.ast.2025.110055.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations (ICLR 2021), Virtual Event*, Oct. 2020.

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*, 24(6):381–395, June 1981. ISSN 0001-0782, 1557-7317. doi: 10.1145/358669.358692.

T. Foote. tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, Woburn, MA, USA, Apr. 2013. IEEE. doi: 10.1109/TePRA.2013.6556373.

J. L. Forshaw, G. S. Aglietti, N. Navarathinam, H. Kadhem, T. Salmon, A. Pisseloup, E. Joffre, T. Chabot, I. Retat, R. Axthelm, S. Barraclough, A. Ratcliffe, C. Bernal, F. Chaumette, A. Pollini, and W. H. Steyn. RemoveDEBRIS: An in-orbit active debris removal demonstration mission. *Acta Astronautica*, 127:448–463, Oct. 2016. ISSN 0094-5765. doi: 10.1016/j.actaastro.2016.06.018.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International*

Conference on Machine Learning, pages 1050–1059. PMLR, June 2016. ISSN: 1938-7228.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. In G. Csurka, editor, *Domain Adaptation in Computer Vision Applications*, pages 189–209. Springer International Publishing, Cham, 2017. ISBN 978-3-319-58346-4 978-3-319-58347-1. doi: 10.1007/978-3-319-58347-1_10. Series Title: Advances in Computer Vision and Pattern Recognition.

J. Goodwill, G. Crum, J. MacKinnon, C. Brewer, M. Monaghan, T. Wise, and C. Wilson. NASA SpaceCube Edge TPU SmallSat Card for Autonomous Operations and Onboard Science-Data Analysis. In *Proceedings of the 35th Annual Small Satellite Conference (SSC21-VII-08)*, 2021.

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.90.

D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs), June 2023. arXiv:1606.08415 [cs].

Y. Hu, S. Speierer, W. Jakob, P. Fua, and M. Salzmann. Wide-Depth-Range 6D Object Pose Estimation in Space. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15865–15874, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.01561.

P. T. Jackson, A. Atapour-Abarghouei, S. Bonner, T. P. Breckon, and B. Obara. Style Augmentation: Data Augmentation via Style Randomization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 83–92, 2019.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment Anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages

3992–4003, Paris, France, Oct. 2023. IEEE. ISBN 979-8-3503-0718-4. doi: 10.1109/ICCV51070.2023.00371.

M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märtens, and S. D'Amico. Satellite Pose Estimation Challenge: Dataset, Competition Design, and Results. *IEEE Transactions on Aerospace and Electronic Systems*, 56(5):4083–4098, Oct. 2020. ISSN 1557-9603. doi: 10.1109/TAES.2020.2989063.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

H. Krüger and S. Theil. TRON - Hardware-in-the-Loop Test Facility for Lunar Descent and Landing Optical Navigation. *IFAC Proceedings Volumes*, 43(15): 265–270, Jan. 2010. ISSN 1474-6670. doi: 10.3182/20100906-5-JP-2022.00046.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. Publisher: Nature Publishing Group.

V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, Feb. 2009. ISSN 1573-1405. doi: 10.1007/s11263-008-0152-6.

K. Li, H. Zhang, and C. Hu. Learning-Based Pose Estimation of Non-Cooperative Spacecrafts with Uncertainty Prediction. *Aerospace*, 9(10):592, Oct. 2022. ISSN 2226-4310. doi: 10.3390/aerospace9100592. Publisher: Multidisciplinary Digital Publishing Institute.

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)*, pages 9992–10002, Montreal, QC, Canada, Oct. 2021. IEEE. ISBN 978-1-6654-2812-5. doi: 10.1109/ICCV48922.2021.00986.

I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR) 2019, Poster/Abstract*, Sept. 2018.

A. Lotti. Improving satellite pose estimation across domain gap with generative adversarial networks. In *Materials Research Proceedings, Vol. 33, III Aerospace PhD-Days*, pages 376–381, Oct. 2023. doi: 10.21741/9781644902677-55.

A. Lotti, D. Modenini, and P. Tortora. Investigating Vision Transformers for Bridging Domain Gap in Satellite Pose Estimation. In C. Ieracitano, N. Mammone, M. Di Clemente, M. Mahmud, R. Furfaro, and F. C. Morabito, editors, *The Use of Artificial Intelligence for Space Applications*, pages 299–314, Cham, 2023a. Springer Nature Switzerland. ISBN 978-3-031-25755-1. doi: 10.1007/978-3-031-25755-1_20.

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*, 60(3):1034–1038, 2023b. ISSN 0022-4650. doi: 10.2514/1.A35496. Publisher: American Institute of Aeronautics and Astronautics.

A. Lotti, D. Modenini, and P. Tortora. Development of a robotic testbed and initial testing for machine learning-based spacecraft pose estimation. In *Proceedings of the 28th AIDAA Conference – 10th CEAS Aerospace Europe Conference*, Turin, Italy, Dec. 2025a.

A. Lotti, D. Modenini, and P. Tortora. Exploring Stochastic Variational Gaussian Processes for Trustworthy Monocular Pose Estimation in Space. In *2025 IEEE 12th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 430–435, Naples, Italy, June 2025b. IEEE. ISBN 979-8-3315-0152-5. doi: 10.1109/MetroAeroSpace64938.2025.11114617.

P. Lunghi, M. Ciarambino, L. Losi, and M. Lavagna. A new experimental facility for testing of vision-based gnc algorithms for planetary landing. In *Proceedings of the 10th International ESA Conference on Guidance, Navigation & Control Systems (GNC 2017)*, 2017.

S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot Operating

System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66): eabm6074, May 2022. ISSN 2470-9476. doi: 10.1126/scirobotics.abm6074.

A. G. d. G. Matthews, M. v. d. Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian Process Library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017. ISSN 1533-7928.

P. M. Mehta, R. Linares, and A. C. Walker. Photometric Data from Nonresolved Objects for Improved Drag and Reentry Prediction. *Journal of Spacecraft and Rockets*, 55(4):959–970, July 2018. ISSN 0022-4650, 1533-6794. doi: 10.2514/1.A33825.

NASA and DrivenData. Pose Bowl: Spacecraft Detection and Pose Estimation Challenge (Pose Estimation Track). link, May 2024. Retrieved May 2025.

M. Olivares-Mendez, M. R. Makhdoomi, B. C. Yalçın, Z. Bokal, V. Muralidharan, M. Ortiz Del Castillo, V. Gaudilliere, L. Pauly, O. Borgue, M. Alandihallaj, J. Thoemel, E. Skrzypczyk, A. Rathinam, K. R. Barad, A. E. R. Shabayek, A. M. Hein, D. Aouada, and C. Martinez. Zero-G Lab: A multi-purpose facility for emulating space operations. *Journal of Space Safety Engineering*, 10(4):509–521, Dec. 2023. ISSN 2468-8967. doi: 10.1016/j.jsse.2023.09.003.

R. Opromolla, G. Fasano, G. Rufino, and M. Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93:53–72, Aug. 2017. ISSN 03760421. doi: 10.1016/j.paerosci.2017.07.001.

T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu. Contrastive Learning for Unpaired Image-to-Image Translation. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, volume 12354, pages 319–345. Springer International Publishing, Cham, 2020. ISBN 978-3-030-58544-0 978-3-030-58545-7. doi: 10.1007/978-3-030-58545-7_19. Series Title: Lecture Notes in Computer Science.

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*, 73 (11):5726–5740, June 2024. ISSN 0273-1177. doi: 10.1016/j.asr.2023.03.036.

T. H. Park, S. Sharma, and S. D'Amico. Towards robust learning-based pose estimation of noncooperative spacecraft. In *2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, Maine, USA*, 2019.

T. H. Park, J. Bosse, and S. D'Amico. Robotic testbed for rendezvous and optical navigation: Multi-source calibration and machine learning use cases. In *2021 AAS/AIAA Astrodynamics Specialist Conference, Big Sky, Virtual, USA*, Dec. 2021.

T. H. Park, M. Märtens, 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)*, pages 1–15, Mar. 2022. doi: 10.1109/AERO53065.2022.9843439. ISSN: 1095-323X.

T. H. Park, M. Märtens, 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*, 204:640–665, Mar. 2023. ISSN 0094-5765. doi: 10.1016/j.actaastro.2023.01.002.

L. Pasqualetto Cassinis, R. Fonod, and E. Gill. Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft. *Progress in Aerospace Sciences*, 110:100548, Oct. 2019. ISSN 03760421. doi: 10.1016/j.paerosci.2019.05.008.

L. Pasqualetto Cassinis, A. Menicucci, E. Gill, I. Ahrns, and M. Sanchez-Gestido. On-ground validation of a CNN-based monocular pose estimation system for uncooperative spacecraft: Bridging domain shift in rendezvous scenarios. *Acta Astronautica*, 196:123–138, July 2022. ISSN 0094-5765. doi: 10.1016/j.actaastro.2022.04.002.

L. Pauly, W. Rharbaoui, C. Shneider, A. Rathinam, V. Gaudillière, and D. Aouada. A survey on deep learning-based monocular spacecraft pose estimation: Current state, limitations and prospects. *Acta Astronautica*, 212:339–360, Nov. 2023. ISSN 00945765. doi: 10.1016/j.actaastro.2023.08.001.

S. Persson, S. D'Amico, and J. Harr. Flight results from prisma formation flying and rendezvous demonstration mission. In *Proceedings of the 61st International Astronautical Congress (IAC)*, pages 1–12, Prague, Czech Republic, 2010. International Astronautical Federation (IAF).

M. Piazza, M. Maestrini, and P. Di Lizia. Monocular Relative Pose Estimation Pipeline for Uncooperative Resident Space Objects. *Journal of Aerospace Information Systems*, 19(9):613–632, Sept. 2022. ISSN 2327-3097. doi: 10.2514/1.I011064.

J. Posso, G. Bois, and Y. Savaria. Mobile-URSONet: an Embeddable Neural Network for Onboard Spacecraft Pose Estimation. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 794–798, May 2022. doi: 10.1109/ISCAS48785.2022.9937721. ISSN: 2158-1525.

P. F. Proença and Y. Gao. Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6007–6013, May 2020. doi: 10.1109/ICRA40945.2020.9197244. ISSN: 2577-087X.

J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017.

L. Regoli, C. Herrmann, K. Ravandoor, and K. Schilling. New Testing Facility for Proximity Operations in Space. *IFAC Proceedings Volumes*, 46(19):353–358, Jan. 2013. ISSN 1474-6670. doi: 10.3182/20130902-5-DE-2040.00030.

T. Rybus, T. Barcinski, J. Lisowski, J. Nicolau-Kuklinski, K. Seweryn, M. Ciesielska, K. Grassmann, J. Grygorczuk, M. Karczewski, M. Kowalski, et al. New planar air-bearing microgravity simulator for verification of space robotics numerical simulations and control algorithms. In *Proceedings of the 12th ESA Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2013.

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, Salt Lake City, UT, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00474.

P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning Feature Matching With Graph Neural Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4937–4946, Seattle,

WA, USA, June 2020. IEEE. ISBN 978-1-7281-7168-5. doi: 10.1109/CVPR42600.2020.00499.

J. L. Schonberger and J.-M. Frahm. Structure-From-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

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)*, pages 1–15, Mar. 2021. doi: 10.1109/AERO50100.2021.9438232. ISSN: 1095-323X.

S. Sharma and S. D'Amico. Neural Network-Based Pose Estimation for Noncooperative Spacecraft Rendezvous. *IEEE Transactions on Aerospace and Electronic Systems*, 56(6):4638–4658, Dec. 2020. ISSN 1557-9603. doi: 10.1109/TAES.2020.2999148.

S. Sharma, C. Beierle, and S. D'Amico. Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks. In *2018 IEEE Aerospace Conference*, pages 1–12, Mar. 2018a. doi: 10.1109/AERO.2018.8396425.

S. Sharma, J. Ventura, and S. D'Amico. Robust Model-Based Monocular Pose Initialization for Noncooperative Spacecraft Rendezvous. *Journal of Spacecraft and Rockets*, 55(6):1414–1429, Nov. 2018b. ISSN 0022-4650, 1533-6794. doi: 10.2514/1.A34124.

J. Song, D. Rondao, and N. Aouf. Deep learning-based spacecraft relative navigation methods: A survey. *Acta Astronautica*, 191:22–40, Feb. 2022. ISSN 00945765. doi: 10.1016/j.actaastro.2021.10.025.

J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. LoFTR: Detector-Free Local Feature Matching with Transformers. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8918–8927, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.00881.

K. Sun, B. Xiao, D. Liu, and J. Wang. Deep High-Resolution Representation Learning for Human Pose Estimation. In *2019 IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 5686–5696, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-7281-3293-8. doi: 10.1109/CVPR.2019.00584.

M. Tan and Q. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, May 2019. ISSN: 2640-3498.

P. Tortora. Almasat-1 launch onboard the vega maiden flight. In *Atti del XX Congresso Nazionale AIDAA*, pages 1–14, Milano, Italy, 2009. Associazione Italiana di Aeronautica e Astronautica.

R. Tsai and R. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5 (3):345–358, June 1989. ISSN 2374-958X. doi: 10.1109/70.34770.

J. Vasconcelos, S. Gaggi, T. Amaral, C. Bakouche, A. Cotuna, and A. Friaças. Close-Proximity Operations Design, Analysis, and Validation for Non-Cooperative Targets with an Application to the ClearSpace-1 Mission. *Aerospace*, 12(1): 67, Jan. 2025. ISSN 2226-4310. doi: 10.3390/aerospace12010067. Publisher: Multidisciplinary Digital Publishing Institute.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Z. Wang, Z. Zhang, X. Sun, Z. Li, and Q. Yu. Revisiting Monocular Satellite Pose Estimation With Transformer. *IEEE Transactions on Aerospace and Electronic Systems*, 58(5):4279–4294, Oct. 2022. ISSN 1557-9603. doi: 10.1109/TAES.2022.3161605.

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*, 60(3):2500–2514, June 2024. ISSN 1557-9603. doi: 10.1109/TAES.2023.3250385.

Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh, and B. Chen. MobileDets: Searching for Object Detection Architectures for Mobile Accelerators. In *2021 IEEE/CVF Conference on Computer Vision*

and Pattern Recognition (CVPR), pages 3824–3833, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.00382.

W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. MetaFormer is Actually What You Need for Vision. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10819, June 2022. doi: 10.1109/CVPR52688.2022.01055. ISSN: 2575-7075.

W. Yu, C. Si, P. Zhou, M. Luo, Y. Zhou, J. Feng, S. Yan, and X. Wang. MetaFormer Baselines for Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):896–912, Feb. 2024. ISSN 1939-3539. doi: 10.1109/TPAMI.2023.3329173.

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 (CVPR)*, pages 7277–7286, 2022.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Venice, Oct. 2017. IEEE. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.244.