POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Deep Multi-Task Learning Based Monocular Relative Pose Estimation of Uncooperative Spacecraft

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Francesco Evangelisti**

Student ID: 968332
Advisor: Prof. Pierluigi Di Lizia
Co-advisors: Francesco Rossi (AIKO srl), Tobia Giani (AIKO srl)
Academic Year: 2022-23

Abstract

This dissertation introduces a Convolutional Neural Network (CNN) based pipeline for estimating the pose of an uncooperative spacecraft using a single grayscale monocular image. Close-proximity autonomous navigation about an uncooperative spacecraft is a crucial problem in the modern space industry for in-orbit servicing missions as well as Active Debris Removal (ADR) operations. Using a low Size-Weight-Power-Cost (SWaP-C) sensor like a monocular camera is preferred over heavier and more energy consuming sensors, but introduces the need of a very robust software to perform pose estimation.
The main contribution of this work is AIKO-NET, a multi-task, deep learning based CNN that leverages state-of-the-art architectures to enable the parallelization of diverse yet related tasks that synergically contribute to the main task: the proposed software is capable of localizing the target and extracting features such as keypoints Heatmap, Depthmap, Normalmap and Shadowmap from a single grayscale image. Ultimately, the relative pose estimation is performed by the means of two different methods: the direct approach consists in retrieving the 6D pose directly as the EfficientPose head output; the indirect one relies on the predicted heatmaps and exploits the Efficient Perspective-n-Point (EPnP) algorithm to deliver an estimate of the pose. Despite the Next-Generation Spacecraft PosE Estimation Dataset (SPEED+) is the benchmark dataset in this field, the data-centric nature of this project made it necessary to generate a new one from scratch: the Multi-Feature Spacecraft Pose Estimation Dataset (MFSPED) introduces three novel feature maps and fuels the aim of this research of testing a modular, extensively multi-task approach. Finally, AIKO-NET is embedded in a complete pipeline consisting of relative trajectory simulation, images sequence generation, pose estimation and position filtering through an Extended Kalman Filter (EKF). The work presented in this dissertation guaranteed a millimeter-level and degree-level accuracy on relative position and orientation predictions respectively, as resulted from testing on synthetic imagery from MFSPED. The work has been developed with an eye on future improvements of the whole pose prediction pipeline and implementation of techniques to overcome problems such as domain gap.
Keywords: spacecraft pose estimation, monocular camera, multi-task learning, deep learning, autonomous navigation

Abstract in lingua italiana

Questa tesi presenta una pipeline basata su una rete neurale convoluzionale (CNN) per la stima della posa di un satellite non cooperativo utilizzando un'unica immagine monoculare in scala di grigi.
La navigazione autonoma in prossimità di un satellite non cooperativo è un problema cruciale nell'industria spaziale moderna, che sta suscitando interesse per applicazioni quali missioni di servizio in orbita e operazioni di rimozione di detriti spaziali (ADR). L'uso di sensori leggeri, poco costosi e a basso consumo energetico come le fotocamere monoculari è preferibile rispetto a sensori più pesanti e ad alto consumo energetico, ma implica lo sviluppo di software più robusti per una stima della posa efficace e accurata.
Il contributo principale di questo lavoro è AIKO-NET, una CNN basata su logiche multi-task e su tecniche di deep learning. La rete sfrutta architetture allo stato dell'arte per consentire l'esecuzione parallela di compiti diversi ma correlati che contribuiscano sinergicamente alla stima della posa relativa. AIKO-NET è in grado di localizzare il target nell'immagine e di estrarre le seguenti feature: Heatmap relative a punti del satellite pre-selezionati, Depthmap, Normalmap e Shadowmap. Inoltre, esegue la stima della posa relativa attraverso due metodi: l'approccio diretto consiste nel recuperare la stima della posa 6D direttamente dall'output della "prediction head" EfficientPose; quello indiretto si basa sulle heatmap ed utilizza l'algoritmo Efficient Perspective-n-Point (EPnP) per fornire una stima accurata della posa.
Nonostante il "Next-Generation Spacecraft PosE Estimation Dataset" (SPEED+) sia il dataset di riferimento in questo campo, la natura datacentrica di questo progetto ha reso necessario costruire un nuovo dataset da zero: il "Multi-Feature Spacecraft Pose Estimation Dataset" (MFSPED) introduce tre nuove feature maps e rende possibile l'analisi di un approccio modulare e estensivamente multi-task.
Infine, AIKO-NET viene introdotta in una pipeline completa, composta dalla simulazione di traiettorie relativa, generazione delle rispettiva sequenze di immagini, stima della posa e applicazione di un "Extended Kalman Filter" (EKF) per la posizione. Il lavoro presentato in questa tesi ha dimostrato una precisione millimetrica nella stima della posizione e dell'ordine del grado nella stima dell'orientamento sulle immagini sintetiche di MFSPED.

Questo progetto è stato sviluppato tenendo in considerazione la possibilità di sviluppi futuri e l'implementazione di tecniche per affrontare problemi tipici di questo dominio come il domain gap.

Parole chiave: stima della posa di un satellite, immagine monoculare, approccio multi-task, deep learning, navigazione autonoma

Contents

List of Figures

List of Tables

Acronyms

AB Anchor Box. 22, 23, 28, 30

ADAM ADAptive Momentum. 15, 16

ADR Active Debris Removal. 1

AI Artificial Intelligence. 12

ANN Artificial Neural Network. vii, 12, 13, 16, 18

BB Bounding Box. 8, 22, 23, 28, 47, 67

BGD Batch Gradient Descent. 15

C-IoU Complete-Intersection over Union. 52

CAD Computer-Aided Design. 3

CNN Convolutional Neural Network. 3, 6, 7, 9–11, 18, 20–23, 26, 39, 41, 42, 51, 54, 62, 78, 82, 83

DCAI Data-Centered Artificial Intelligence. 41

DL Deep Learning. 13

ECI Earth Centered Inertial. 54

EKF Extended Kalman Filter. 10, 11, 37, 38, 53, 54, 59, 78, 79, 82

EOL End-Of-Life. 1

EPnP Efficient Perspective-n-Point. 32, 33, 52, 54, 67, 70, 82

ESA European Space Agency. 2, 3

FCC Federal Communications Commission. 1

FLOPS FLoating Operations Per Second. 26

FoV Field of View. 5, 44, 67

FPS Frames Per Second. 58

GDM Gradient Descent Method. 14, 15

GT Ground-Truth. 46, 60, 65, 67

HIL Hardware-In-the-Loop. 4, 9

LEO Low Earth Orbit. 1

LiDAR Light Detection And Ranging. 1, 2

LR Learning Rate. 14, 15, 62

ML Machine Learning. 5, 12, 13

MSE Mean Squared Error. 52

MTL Multi-Task Learning. 10, 16, 17, 39, 40, 42, 45, 51, 82

NASA National Aeronautics and Space Administration. 2

NMS Non-Maximum Suppression. 23

ODR Online Domain Refinement. 9, 40

OSAM-1 On-orbit Servicing, Assembly, and Manufacturing 1. 2

PnP Perspective-n-Point. 3, 32, 33, 82

ReLU Rectified Linear activation Unit. 20, 25

RMSProp Root Mean Square Propagation. 15

RoI Region of Interest. 6, 70

S/C Spacecraft. 1, 3

SGD Stochastic Gradient Descent. 15

SiLU Sigmoid Linear Units. 29, 30, 52

SLAB Space rendezvous LABoratory. 3, 4, 6, 40, 82

SOTA State-Of-The-Art. 2, 3, 82

SPEC Spacecraft Pose Estimation Challenge. 3, 4, 6, 8

SPEED Spacecraft PosE Estimation Dataset. 4, 6, 7, 9, 40

SPNv2 Spacecraft Pose Network v2. 6, 7, 9, 24, 40–42, 51, 52, 61, 82

SSA Space Situational Awareness. 1

STL Single-Task Learning. 16

SWaP-C Size-Weight-Power-Cost. 2

TLEs Two Line Elements. 54

TRON Testbed for Rendezvous and Optical Navigation. 4

UKF Unscented Kalman Filter. 37, 38

UT Unscented Transform. 38

YOLO You Only Look Once. 23

1 | Introduction

1.1. Problem statement and motivation

Pose estimation of uncooperative spacecrafts is a crucial problem in modern space operations [43]. An uncooperative spacecraft is one that lacks supportive means such as light-emitting markers and cannot establish a communication link, which makes it difficult to determine its position and orientation.
This dissertation aims to address this problem in a close-proximity scenario through a system capable of estimating the relative pose of an uncooperative S/C from a single grayscale monocular image.
In recent years, this approach has become increasingly interesting and important due to its potential to support various mission concepts aimed at ensuring the sustainability of near-Earth space, such as refueling space assets, Active Debris Removal (ADR), extending the functional lifetime of spacecrafts, performing in-situ Space Situational Awareness (SSA) or inspection to diagnose orbiting objects. By estimating the relative pose of the target spacecraft in real time, an autonomous servicer spacecraft can generate safe and fuel-efficient rendezvous and docking trajectories. Furthermore, a new rule adopted by the FCC on the 29^{th} of September 2022 addresses the growing risk of orbital debris requiring satellite operators in LEO to dispose of their satellites within 5 years of completing their missions [10], shortening the decades-old 25-year guideline for deorbiting satellites post-mission. This may also open other scenarios for the application of some of the solutions discussed in this dissertation, such as the removal of satellites that reach their end-of-life (EOL) and that are not pre-engineered for servicing.
Two main approaches exist to estimate the pose of an uncooperative target S/C: ground-based and spaceborne solutions. However, the use of the former approach is unfeasible for accurate estimation, because of the inherent high uncertainty when observing a small, far away object from a ground station on Earth, and because of the dependency on the S/C visibility [43]. On the other hand, using onboard sensors such as *Light Detection And Ranging* devices (LiDAR) and/or stereo cameras overcome these problems, but such instruments can be expensive, heavy and highly energy consuming. An optimal solution would

be found in a low Size-Weight-Power-Cost (SWaP-C) sensor, like a grayscale monocular camera, that also fits the limited onboard capacity of small spacecraft such as Cubesats. Of course, there are significant challenges associated with using a monocular camera for relative pose estimation, the first of which is the high complexity of the image process algorithms required to extract the necessary information and the weaker robustness to lighting conditions and variable backgrounds compared to LiDAR [43].
Despite these challenges, the estimation of the relative pose by means of a SWaP-C monocular camera is an appealing possibility, especially within the framework of on-orbit servicing missions. Among the most innovative missions in this framework, ESA ClearSpace-1 [2] is the first that will rendezvous with, capture and bring down a large derelict object: this is the first example of Active Debris Removal mission, and is planned to be launched in 2026. NASA OSAM-1 (On-orbit Servicing, Assembly, and Manufacturing 1) [5] servicer will rendezvous with, grasp, refuel, and relocate a US government-owned satellite to extend its life. This mission is scheduled for launch in 2024 and will demonstrate various world's firsts from the refueling of a satellite not designed to be serviced, to In-space robotic precision assembly. Between the private companies taking part in the new rising space economy, Northrop Grumman [25], Astroscale [11], D-Orbit [3], Infinite Orbits [4] and AIKO [1] are pioneering in-orbit servicing and space logistics.
In conclusion, accurate estimation of the relative pose of uncooperative spacecraft using minimal hardware is critical in various space scenarios. The use of a monocular camera for pose estimation represents a promising solution due to its low cost, light weight, and suitability for a range of space missions. However, significant challenges need to be overcome in terms of algorithm complexity and robustness. With the upcoming space missions and the growing importance of space operations, the development of accurate and cost-effective relative pose estimation methods will continue to be a critical research area in the future.

1.2. State of the art

In this section, the current state-of-the-art (SOTA) techniques used for spacecraft pose estimation from a monocular image are presented. All the image-processing algorithms share the common logic of recognizing and locating specific semantic features of the spacecraft within the image frame. These features are then used by some kind of subsystem to match these features to the prediction of the pose.
There are three main approaches for pose estimation: pure feature-based estimation, pure deep learning estimation, and hybrid methods [15]. The earliest methods proposed for relative pose estimation in space were feature-based, relying on hand-crafted features (like

corner [49] or edges [18]) extracted from a single image of the target, along with information from a 3D CAD model, to solve the Perspective-n-Points (PnP) problem. The first algorithm allowing proximity operations among uncooperative spacecraft was introduced by S. D'Amico [23] in 2014 and tested on spaceborne images from the PRISMA mission. In 2018, the Sharma-Ventura-D'Amico (SVD) algorithm [53] was proposed as an improvement on the 2014 algorithm, achieving SOTA performance in terms of robustness and efficiency. A further enhancement to the SVD was proposed in 2019, utilizing a three-stream image processing approach [19]: three parallel streams independently extract corners or edges, but in the end only the features that are common to all the streams are used to solve the PnP, compensating for possible false detections in one of the streams. Despite the slight improvement in robustness, this approach was not computationally feasible for real-world scenarios. On the other hand, deep learning-based methods utilize convolutional neural networks (CNNs) and can estimate the relative pose through either direct regression ([39, 44]), pure classification, or a hybrid regression-classification problem [52]. To attain a satisfactory level of pose accuracy, classification-based methods entail dividing the pose space into numerous pose labels. In contrast, direct regression-based methods necessitate cautious parameter selection to prevent unpredictable performance when learning the transformation from input 2-D pixel data to output regression parameters that describe the 6-D pose space [52]. The hybrid approach combines features extracted from a 3D CAD model with CNN-regressed landmarks or feature points to solve the PnP problem and estimate the relative pose.
Methods utilizing deep learning for computer vision applications, such as object detection, have exhibited noteworthy advancements in terms of precision and resilience when contrasted with feature-based techniques [52].

In this framework, the Space Rendezvous Laboratory (SLAB) [6] of Stanford University proposed various solutions ([40, 52]) throughout the last years, and has collaborated with the ESA to the organization of a pose estimation competition called Kelvin Spacecraft Pose Estimation Challenge (SPEC). There have been two editions of such challenge, in 2019 [8] and in 2021 [9]. The results of the 2019 ESA's Kelvins Pose Estimation Challenge indicate that the most effective approach to relative pose estimation via monocular images is using hybrid methods, where CNNs aid PnP solvers [15].

The main problem with CNNs is that they require a large number of labeled images to be trained, and such spaceborne images for the specific application are not available. Furthermore, the SOTA models need to be trained on images of the specific S/C that is the target of the pose estimation. This means that datasets have to be generated synthetically, taking care of the *domain gap* problem. The SLAB actively contributed to

the challenges by making available what have been the benchmark datasets for SPEC2019 and SPEC2021: SPEED [52] and SPEED+ [38], respectively.

1.2.1. Next-Generation Spacecraft PosE Estimation Dataset

SPEED+ (Spacecraft Pose Estimation Dataset +) is the evolution of SLAB's SPEED [52] and aims at bridging the gap between the current synthetic images and real, spaceborne images. The first version of SPEED consisted of 1500 synthetic as well as 300 actual camera images of a mock-up of the Tango spacecraft from the PRISMA mission, although the real spaceborne images are part of the validation and test sets only to evaluate the robustness and the domain adaptation capabilities of the pose estimation techniques. Synthetic images of a target spacecraft are easy to mass-produce but fail to resemble the visual features and illumination variability inherent to spaceborne images. In order to address this problem, SPEED+ introduces hardware-in-the-loop (HIL) images of a spacecraft mock-up model captured from the *Testbed for Rendezvous and Optical Navigation* (TRON) facility, which is a robotic testbed made to generate target images with accurate pose labels and high-fidelity spaceborne illumination conditions. SPEED+ comprises images from the `synthetic` domain, and from two simulated HIL domains with different sources of illumination as shown in Figure 1.1: `lightbox` and `sunlamp`.

(a) Tango mock-up illuminated by light boxes.

(b) Tango mock-up illuminated by a sun lamp.

Figure 1.1: TRON facility. (*credits to*: [38])

In this way, more than just hundreds of spaceborn images of a single target are available to comprehensively evaluate the robustness of a Machine Learning (ML) model. The dataset compositions for different domains and splits are reported in Table 1.1, while the parameters of the camera model used for simulating the images are reported in Table 1.2. Finally, Figure 1.2 shows some example images from the dataset.

Table 1.1: SPEED+ dataset composition for different domains and splits.

	`synthetic`	`lightbox`	`sunlamp`
Train	47966	-	-
Validation	11994	-	-
Test	-	6740	2791

Table 1.2: SPEED+ camera parameters.

Parameter	Value
Resolution ($N_u \times N_v$)	1920×1200 px
Focal length ($f_x = f_y$)	17.6 mm
Pixel pitch ($\rho_u = \rho_v$)	5.86 μm/px
Horizontal FoV	35.452°
Vertical FoV	22.595°

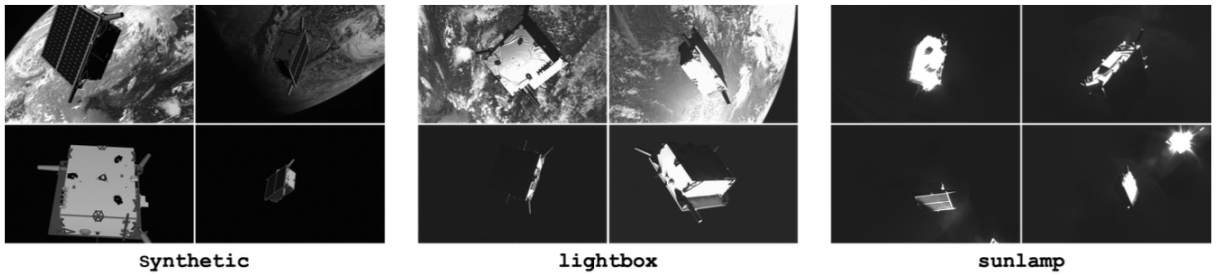


Figure 1.2: Example images from different domains of SPEED+.

While they were not public before, on January 12, 2023 also the lightbox and sunlamp domain test labels were made available. The complete dataset can be found in the *Stanford Digital Repository* [7].

The last work proposed by the SLAB is the *Spacecraft Pose Network v2* [39] (SPNv2), a multi-scale, multi-task CNN for spacecraft pose estimation that bases its training and performance evaluation on SPEED+. The new BiFPN-enabled, multi-scale features logic allows the network to parallelize different tasks such as object localization and pose estimation, differently from what was done in previous works. The SLAB's baseline for SPEC2019 [40] consisted in 2 subsystems working one after the other, as shown in Figure 1.3, with the output of the first one being the input for the second one.

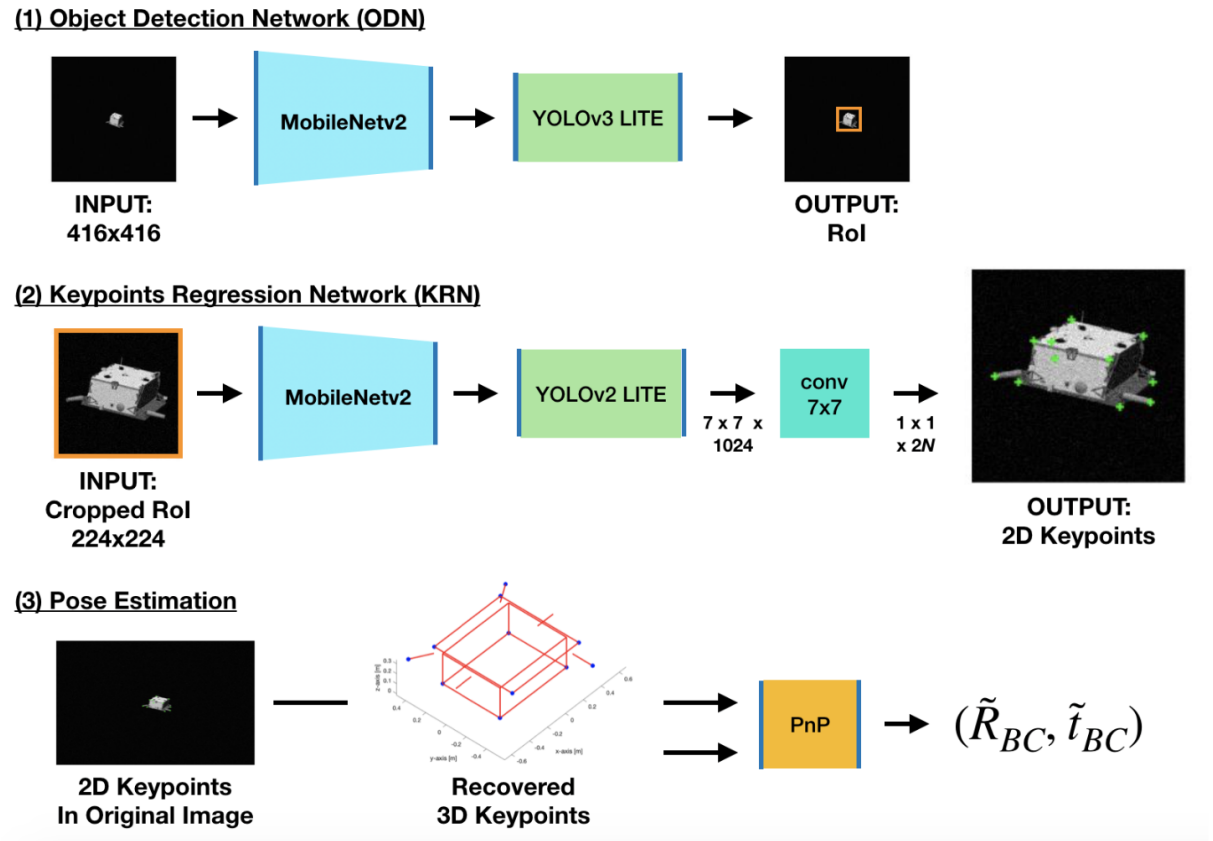


Figure 1.3: Architecture of SLAB's baseline for SPEC2019. (*credits to:* [40])

An *Object Detection Network* (ODN) was responsible for the bounding box prediction, which was used to crop the image into a Region of Interest (RoI) which would be passed to a *Keypoints Regression Network* (KRN) that output the 2D keypoints locations. This was necessary since the different scales of the satellite in the images did not make it possible to efficiently learn the keypoints related features, needed for estimating the relative pose.

On the other hand, the new SPNv2 architecture manages to perform object detection and pose prediction simultaneously, and is capable of generalizing feature maps at different scales.

1.2.2. Spacecraft Pose Network v2

Spacecraft Pose Network v2 [39] is a multi-scale, convolutional neural network (CNN) for pose estimation of uncooperative spacecraft based on a multi-task architecture consisting in a shared feature encoder and multiple prediction heads performing different tasks on a shared feature encoder. As depicted in Figure 1.4, the tasks are the prediction of pre-defined satellite keypoints, direct pose regression, and binary segmentation of the satellite foreground. While these tasks differ from each other, they share some fundamental information related to the target's geometry and pose. The "multi-scale" feature refers to the fact that the network can be scaled due to the nature of its EfficientNet [55] backbone: through a single parameter ϕ, compound scaling allows to optimize performance and accuracy changing the depth, width, and input resolution of the CNN following a scaling principle that will be later described in Section 2.2.1. For completeness, it has to be noticed that all the SPNv2 performance metrics that will be reported here are referred to a set scaling parameter $\phi = 3$, and to an input resolution of the images of 768×512. The GitHub repository is available at `https://github.com/tpark94/spnv2`.

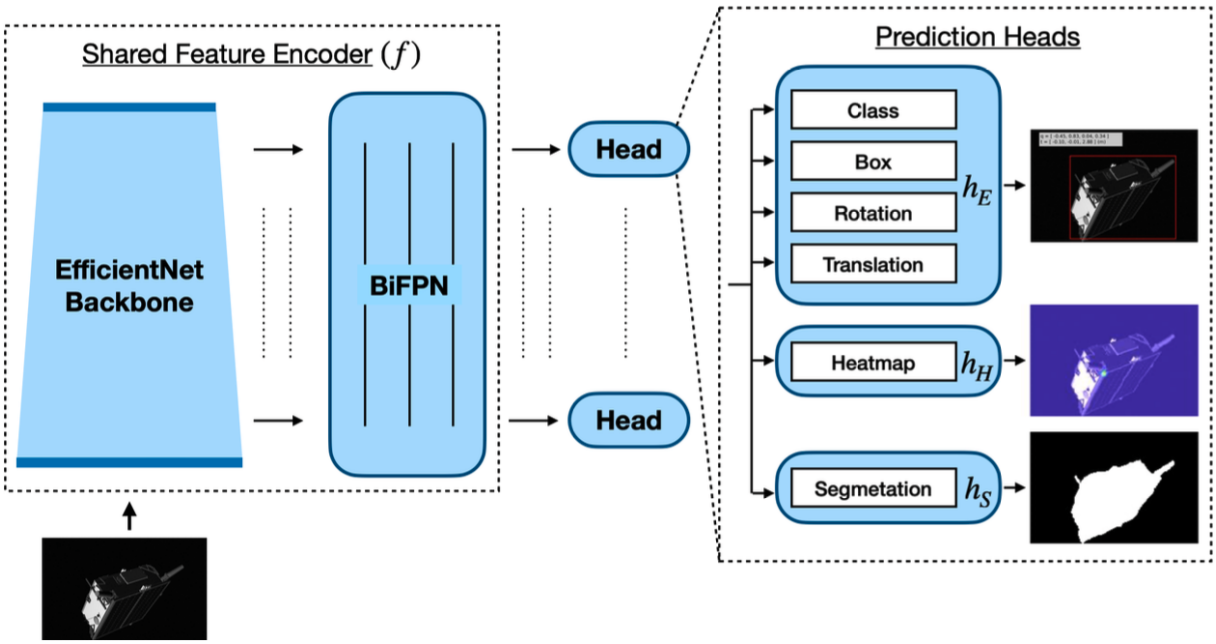


Figure 1.4: The SPNv2 architecture. (*credits to*: [39])

Since SPNv2 has also the objective of bridging the domain gap due to training on synthetic images, the previously introduced SPEED+ dataset is used. By jointly training on the different yet related tasks and by using extensive data augmentation on synthetic images only, the shared encoder is encouraged to "learn features that are common across image

domains that have fundamentally different visual characteristics compared to synthetic images" [39].

The proposed architecture is based on EfficientPose [17], which utilizes EfficientDet's backbone and neck as its feature encoder. This encoder includes the EfficientNet [55] backbone and a Bi-directional Feature Pyramid Network (BiFPN) that combines features from various scales. The shared feature encoder output is fed into different prediction heads, which are:

- EfficientPose head (h_{E})
- Heatmap head (h_{H})
- Segmentation head (h_{S})

The EfficientPose head includes subnets responsible for object presence binary classification, bounding box (BB) prediction, and target rotation and translation regression, and it will be detailed in Section 2.2.2; the Heatmap head predicts the 2D heatmaps of pre-defined spacecraft surface keypoints; the Segmentation head enables pixel-wise binary segmentation of the spacecraft foreground.

Training h_{E} focuses on the minimization of three losses:

- *focal loss* [33] for the classification subnet;
- *Intersection-over-Union loss* (IoU loss) [61] for the bounding box subnet;
- *SPEED score* [30] for the rotation and translation prediction subnets.

The SPEED score, or SPEED loss, is the official performance metric of the SPEC, and is defined as

$$E_{\mathrm{pose}} = E_{\mathrm{R}}(\hat{\mathbf{R}}, \mathbf{R}) + \frac{E_{\mathrm{T}}(\tilde{\mathbf{t}}, \mathbf{t})}{\|\mathbf{t}\|}, \tag{1.1}$$

with the ground-truth rotation matrix and translation vector are $(\mathbf{R}, \mathbf{t})$ and the respective predictions are $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$, and where the rotation error E_{R} and the translation error E_{T} are respectively defined as in Equation 1.2.

$$\begin{aligned} E_{\mathrm{R}}(\hat{\mathbf{R}}, \mathbf{R}) &= \arccos \frac{\mathrm{tr}(\mathbf{R}^{\top}, \hat{\mathbf{R}}) - 1}{2} \\ E_{\mathrm{T}}(\hat{\mathbf{t}}, \mathbf{t}) &= \|\hat{\mathbf{t}} - \mathbf{t}\| \end{aligned} \tag{1.2}$$

Finally, h_{H} minimizes pixel-wise mean squared error loss with respect to the ground-truth

heatmaps, and h_S minimizes the pixel-wise binary cross entropy loss. All these losses are associated with equal weights so that they have a balanced impact on the model training. Moreover, to address the issue of overfitting to synthetic images, SPNv2 is trained using extensive data augmentation techniques. The implementation details can be retrieved in the original paper [39].

Due to the architecture modularity, SPNv2 has been tested in different configurations, which are reported in Table 1.3 along with the respective performances.

Table 1.3: Bounding box and pose predictions of SPNv2 different head configurations. Bold numbers indicate the best performances.

Heads	Source	`lightbox`				`sunlamp`			
		IoU [-]	E_T [m]	E_R [°]	E_{pose} [-]	IoU [-]	E_T [m]	E_R [°]	E_{pose} [-]
E	E	0.878	0.400	21.921	0.446	0.901	0.293	32.066	0.608
H	H	-	0.368	13.577	0.298	-	0.424	19.739	0.414
E + H	E	0.915	0.181	8.757	0.183	0.911	0.251	14.175	0.288
	H	-	0.272	6.924	0.165	-	0.314	12.323	0.268
E + H + S	E	**0.918**	**0.175**	8.004	0.169	**0.919**	**0.225**	12.433	0.254
	H	-	0.271	**6.4779**	**0.158**	-	0.307	**11.065**	**0.245**

The letters E, H, and S stand for h_E, h_H, and h_S, respectively. Here, all the configurations were trained for 20 epochs but the one with h_H only, which was trained for 10 epochs only since heatmap prediction resulted easier to learn. Analyzing the performances, it is clear that having more prediction heads in the architecture improves the CNN robustness on the SPEED+ HIL domains. This interesting result will be taken as a starting point for the work presented in this dissertation, which has SPNv2 as a baseline and tries to extend its architecture by introducing new tasks to enhance the multi-task learning synergy.

SPNv2 also introduces Online Domain Refinement (ODR) to fine-tune the affine transformation parameters of the normalization layers of the feature encoder: since spaceborne images have never been integrated into the offline training procedure, the CNN performance during real missions lacks fundamental guarantee. This problem is addressed by making the SPNv2 architecture such that it can be refined on the target domain images on-board the spacecraft avionics. Although this approach is interesting and innovative, we will not go into its details in this dissertation since the aim of this study is to further analyze the multi-task nature of pose estimation CNNs to test whether it can provide better performance by enhancing the informative level of the dataset and the capability of the network itself to fully use it and minimize the estimation errors.

1.3. Development Framework

The work presented in this thesis was developed in collaboration with AIKO [1], an Italian company pioneering AI solutions for the space industry. The project involved the development of a deep learning model using Python[1] and PyTorch [2], an open-source machine learning library for Python that provides a wide range of functionalities for building and training deep neural networks. This project was my first experience working with Python and its libraries, and I had the opportunity to learn the language and its ecosystem through the development of the model. To access the computational resources required for training the deep learning model, a remote machine was used. The remote machine was accessed via Bash, which allowed me to interact with the machine's terminal and run commands. The development and training environments were set up using Docker[3], which is a containerization platform that enables the creation and deployment of portable environments. Docker allowed me to create an isolated environment with all the necessary dependencies and libraries required for the project, ensuring reproducibility and portability of the work. The project was managed using GitLab[4], which is a web-based Git repository manager that allowed for effective collaboration and version control among the team members. These tools provided the necessary infrastructure for the development, training, and deployment of the deep learning model.

1.4. Outline

This thesis is divided into 6 chapters.
Chapter 1 introduces the problem statement and motivation, highlighting the importance of accurate spacecraft pose estimation in various applications. The existing literature and the advancements made in spacecraft pose estimation are reviewed, including the Next-Generation Spacecraft Pose Estimation Dataset and the Spacecraft Pose Network v2. Chapter 2 will cover the mathematical background to understand the artificial intelligence (AI) techniques used in this thesis work, such as deep learning (DL), gradient-based learning, multi-task learning (MTL), and convolutional neural networks (CNNs). Additionally, it provides an overview of the EfficientDet and EfficientPose architectures and of the Perspective-n-Point problem. Finally, the relative orbital dynamics is faced and the equations of motion (EOM) are retrieved, concluding the chapter with extended Kalman filters (EKFs). In Chapter 3, the main work of this thesis, AIKO-NET, is presented

[1]Python. https://www.python.org/
[2]PyTorch. https://pytorch.org/
[3]Docker. https://www.docker.com/
[4]Gitlab. https://gitlab.com/

and analyzed in all its parts. AIKO-NET uses a custom dataset necessary to provide novel feature maps introduced as inputs for the CNN architecture. The full relative pose estimation pipeline is presented in Chapter 4, which includes trajectory generation, image sequence generation, and the implementation of the EKF. Finally, in Chapter 5, the results of the proposed AIKO-NET are evaluated through the presented error metrics, the predictions are visualized for an immediate understanding of the architecture potentialities, and the full pose prediction pipeline is tested. In Chapter 6, the work and the results are summarized, and future developments and improvement scenarios are presented.

2 | Theoretical Background

2.1. Artificial Intelligence

Artificial Intelligence (AI) is a rapidly growing field of computer science that aims to create intelligent machines capable of performing tasks that typically require human intelligence, such as learning, problem-solving, decision-making, and natural language processing. Artificial Neural Networks (ANNs) are a fundamental component of the field of Machine Learning (ML), a subset of AI that focuses on creating algorithms that enable computers to learn and improve from experience. ANNs are inspired by the structure and function of biological neurons in the human brain, as depicted in Figure 2.1, and are used to simulate complex decision-making processes. In ANNs, inputs are processed through interconnected layers of nodes [14], with each node using a mathematical function to transform the input before passing it on to the next layer. The weights and biases associated with each node are adjusted during training to optimize the network's performance in solving specific tasks.

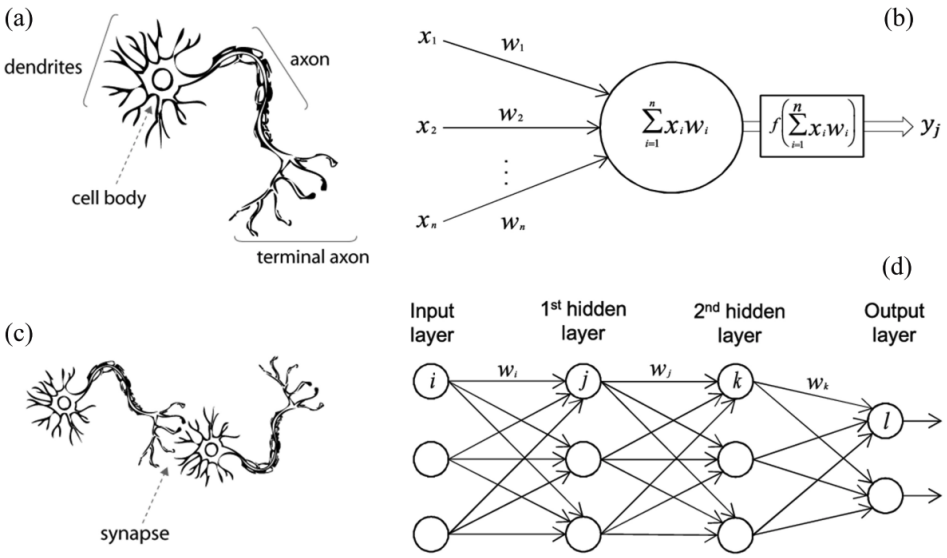


Figure 2.1: Biological neurons in comparison to an ANN. (a) human neuron; (b) artificial neuron; (c) biological synapse; (d) ANN synapses (*credits to:* [54])

A generic ANN is typically composed of one or more layers of neurons that process input data and produce output predictions. The architecture can be visualized as a directed acyclic graph, where each neuron in a layer is connected to all the neurons in the previous layer, and its output is passed as input to the next layer. The first layer of neurons is the *input layer* and receives the input data. The last layer is the *output layer*, which produces the final predictions. The layers between the input and output layers are called *hidden layers*. Each neuron in a layer applies an activation function to the weighted sum of its inputs and a bias term. The weights and biases of all the neurons in the network are the *learnable parameters* that are updated during training. The choice of activation function depends on the task and the nature of the data. During training, the input data is fed into the network, and the output predictions are compared to the true labels using a loss function. The goal of training is to adjust the network parameters to minimize the loss function. Once trained, the network can be used to make predictions on new data.

In this section, we present a theoretical background on artificial intelligence. Most of the material covered is based on two main reference books, "Neural Networks and Deep Learning: A Textbook" by Aggarwal (2019) [14] and "Deep Learning: A Practitioner's Approach" by Patterson and Gibson (2017) [41].

2.1.1. Deep Learning

One of the most significant developments in the field of ANNs has been the emergence of Deep Learning (DL), a type of ML that relies on deep neural networks. The "deep" architecture of these networks, meaning that they consist of multiple layers, allows them to model highly nonlinear relationships between inputs and outputs, making them particularly effective for tasks such as image and speech recognition, natural language processing, autonomous driving, and medical diagnosis.

The principal drawbacks of Deep Learning models are that they require large amounts of labeled data and computing resources to train effectively, and can be difficult to interpret and debug.

2.1.2. Gradient-based learning

In ML algorithms, the learning process relies on the minimization of a cost function to optimize the model parameters. Such cost function is defined as the sum of a loss function cost and a regularization cost. Let m be the number of training samples and L be the number of layers in a generic network.

$$J = \underbrace{\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(\mathbf{Y}_i, \hat{\mathbf{Y}}_i\right)}_{\text{loss function cost}} + \underbrace{\frac{\lambda}{2m}\sum_{l=1}^{L}\|\mathbf{W}^{[l]}\|_F}_{\text{regularization cost}} \tag{2.1}$$

The loss function cost is the average of the loss function $\mathcal{L}$ evaluated on the model's predictions $\hat{\mathbf{Y}}_i$ and the true output $\mathbf{Y}_i$ for each training sample i. The summation over m training samples represents the total cost of the model on the training data. Regarding the regularization cost, it is a penalty term related to the weights of the model to prevent overfitting. The standard technique used for this task is L_2-regularization and exploits the Frobenius norm of the weights $\|W^{[l]}\|$ at layer l. The term λ is the regularization hyper-parameter, and it can be tuned to provide the best performance. For a generic matrix $\mathbf{A}$ of size $(m \times n)$, the Frobenius norm is defined as:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{2.2}$$

Thus, the regularization cost adds a penalty to the overall cost for having large weight values in the model, encouraging it to prefer simpler solutions that generalize better to the new data. The iterative minimization of the cost function requires computing $\mathrm{d}\mathbf{W}^{[l]} = \frac{\partial J}{\partial \mathbf{W}^{[l]}}$ and $\mathrm{d}\mathbf{b}^{[l]} = \frac{\partial J}{\partial \mathbf{b}^{[l]}}$, namely the gradients of J with respect to the learned weights and biases, which is done by means of the backpropagation algorithm [21].

Gradient Descent Method

The most basic gradient-based learning algorithm is the *Gradient Descent Method* (GDM), which reduces the cost function by iteratively adjusting the model parameters in the direction of the negative gradient of the cost function, as shown in equations 2.3 for a model's weights and biases:

$$\begin{aligned} \mathbf{W}^{[l]} &\leftarrow \mathbf{W}^{[l]} - \alpha\ \mathrm{d}\mathbf{W}^{[l]} \\ \mathbf{b}^{[l]} &\leftarrow \mathbf{b}^{[l]} - \alpha\ \mathrm{d}\mathbf{b}^{[l]} \end{aligned} \tag{2.3}$$

where k represents the current step in the iterative process, and α is the learning rate that determines the step size of the updates. The learning rate (LR) is a hyperparameter that needs to be tuned carefully to ensure efficient and effective learning. Using a constant LR throughout the whole training process is not desirable: a lower LR used early on will

cause the algorithm to take too long to come even close to an optimal solution; a large initial LR will allow the algorithm to come reasonably close to a good solution at first, but will then make it oscillate around a local minimum for a very long time or diverge in an unstable way. A common approach is to use step decay, in which the LR is reduced by a particular factor every few epochs.

Gradient-based methods are not guaranteed to converge to the global optimum of a non-convex objective function. While these methods were initially limited to linear systems, it was later discovered that the presence of local minima did not pose a significant problem in practice [31].

Stochastic Gradient Descent

The *Stochastic Gradient Descent* (SGD) method is a variant of the GDM that randomly samples a subset of the training data (also known as a *mini-batch*) to compute the gradient of the cost function. The SGD method updates the model parameters based on the gradient computed from the *mini-batch*, rather than the entire training data set (what is done with a *Batch Gradient Descent* (BGD) approach), which leads to faster convergence and reduced computational complexity. In this way, the gradient-based optimization only has to be executed on a small subset of training examples prior to performing the parameter update. This is very common in deep-learning applications, where the size of training datasets can be very large.

ADAM optimization

The basic GDM is a simple optimization algorithm that updates the model parameters based on the gradients of the cost function. However, it is not guaranteed to converge to the global optimum of the objective function, which is generally non-convex. To overcome this limitation, more sophisticated gradient-based algorithms have been developed. One such algorithm is the ADAptive Momentum (ADAM) optimization [29], which combines the momentum update [45] with RMSProp [14] and bias correction. The GDM with momentum term accelerates convergence whenever mini-batches are used. Unlike the basic GDM which updates the parameters proportional to the gradients, this method instead employs an exponentially-weighted moving average of the gradients. As a result, the convergence is smoother, which allows for the use of larger learning rates. The RMSProp method reduces the oscillations that occur in high-curvature vector-space directions and promotes faster convergence in directions with smooth and consistent gradients. Thus, ADAM calculates a dynamic learning rate based on the normalized estimates of the first

and second moments of the gradients, which are approximated by exponentially-weighted moving averages of the past gradients. Normalization is necessary since the moments are usually biased towards zero due to their initialization. This is especially true during the early iterations of the optimization process. The algorithm is reported here below.

Algorithm 2.1 ADAM

Require: α: Learning rate
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for moment estimates
Initialize: $\mathbf{m}_{\mathrm{d}\mathbf{W}}, \mathbf{m}_{\mathrm{d}\mathbf{b}}, \mathbf{v}_{\mathrm{d}\mathbf{W}}, \mathbf{v}_{\mathrm{d}\mathbf{b}}$
for t in mini-batches **do**
  Backpropagation: compute $\mathrm{d}\mathbf{W}, \mathrm{d}\mathbf{b}$
  $\mathbf{m}_{\mathrm{d}\mathbf{W}} = \frac{1}{1-\beta_1^t}\beta_1 \mathbf{m}_{\mathrm{d}\mathbf{W}} + (1-\beta_1)\mathrm{d}\mathbf{W}$
  $\mathbf{m}_{\mathrm{d}\mathbf{b}} = \frac{1}{1-\beta_1^t}\beta_1 \mathbf{m}_{\mathrm{d}\mathbf{b}} + (1-\beta_1)\mathrm{d}\mathbf{b}$
  $\mathbf{v}_{\mathrm{d}\mathbf{W}} = \frac{1}{1-\beta_2^t}\beta_1 \mathbf{v}_{\mathrm{d}\mathbf{W}} + (1-\beta_2)\mathrm{d}\mathbf{W}^2$
  $\mathbf{v}_{\mathrm{d}\mathbf{b}} = \frac{1}{1-\beta_2^t}\beta_1 \mathbf{v}_{\mathrm{d}\mathbf{b}} + (1-\beta_2)\mathrm{d}\mathbf{b}^2$
  Update weights and biases:
  $\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\mathbf{m}_{\mathrm{d}\mathbf{W}}}{\sqrt{\mathbf{v}_{\mathrm{d}\mathbf{W}}}+\epsilon}$
  $\mathbf{b} \leftarrow \mathbf{b} - \alpha \frac{\mathbf{m}_{\mathrm{d}\mathbf{b}}}{\sqrt{\mathbf{v}_{\mathrm{d}\mathbf{b}}}+\epsilon}$
end for

Here, ϵ is a small number added to the denominator in order to prevent division by zero, typically set to ϵ =1e−8. The learning rate is adaptively computed by ADAM, but still requires manual tuning. While the remaining hyper-parameters β_1 and β_2 could be tuned, they are often set to the default values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as suggested by the authors of [29].

2.1.3. Multi-Task Learning

The Multi-Task Learning (MTL) [20] approach involves utilizing domain information present in the training signals of related tasks as an inductive bias to enhance generalization. This is achieved by simultaneously learning multiple tasks and utilizing a shared representation. The knowledge gained from one task can aid in the learning process of other tasks, resulting in improved performance. In Figure 2.2, four distinct ANNs are depicted, with each net being a function of identical inputs and having a single output. To train these nets, backpropagation is used in isolation for each net. As these nets are not interconnected, the knowledge gained by one net cannot aid in the learning of another net. This method of training is referred to as Single Task Learning (STL).

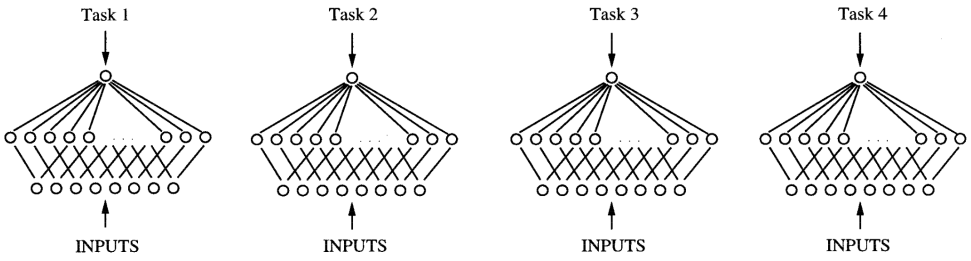


Figure 2.2: Single Task Backpropagation (STL) of four tasks with the same inputs. (*credits to:* [20])

Figure 2.3 shows a singular net which possesses the same inputs as the four nets in Figure 2.2. However, this net has four outputs, each corresponding to one of the tasks that the nets in Figure 2.2 were being trained on. It is important to notice that these four outputs are completely connected to a shared hidden layer. Backpropagation is performed concurrently on all four outputs of the MTL net. As the four outputs are linked to a common hidden layer, internal representations that are formed in the hidden layer for one task can be utilized by the other tasks. The fundamental concept behind multi-task learning is to share the knowledge gained from distinct tasks while they are being trained in parallel.

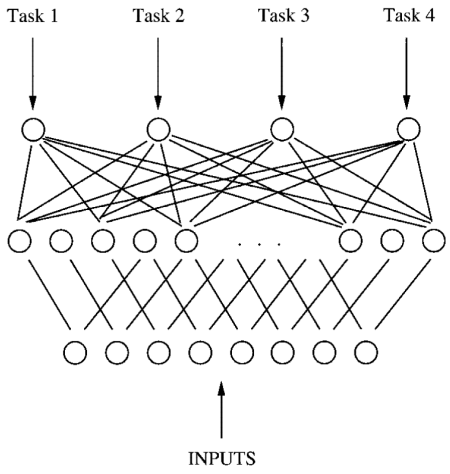


Figure 2.3: Multi-Task Backpropagation of four tasks with the same inputs. (*credits to:* [20])

Multi-task learning uses the training signals for related tasks as an inductive bias to improve generalization. Inductive bias is anything that causes an inductive learner to prefer some hypotheses over other hypotheses. For example, the addition of noise to backpropagation can occasionally enhance generalization: when tasks are uncorrelated,

their contribution to the combined gradient (which aggregates the errors fed back from the outputs of each layer) may be perceived as noise by other tasks. Thus uncorrelated tasks might improve generalization by acting as a source of noise.

2.1.4. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of ANNs that are specifically designed for computer vision applications. CNNs offer a significant advantage, compared to traditional ANNs. This is due to the utilization of two fundamental operations: convolution and pooling, which enable the network to efficiently analyze the input image data.

Architecture Overview

CNNs transform input data through a series of connected layers, ultimately producing a set of class scores at the output layer. While various architectures exist, all CNNs follow a specific pattern of layers, as shown in Figure 2.4.

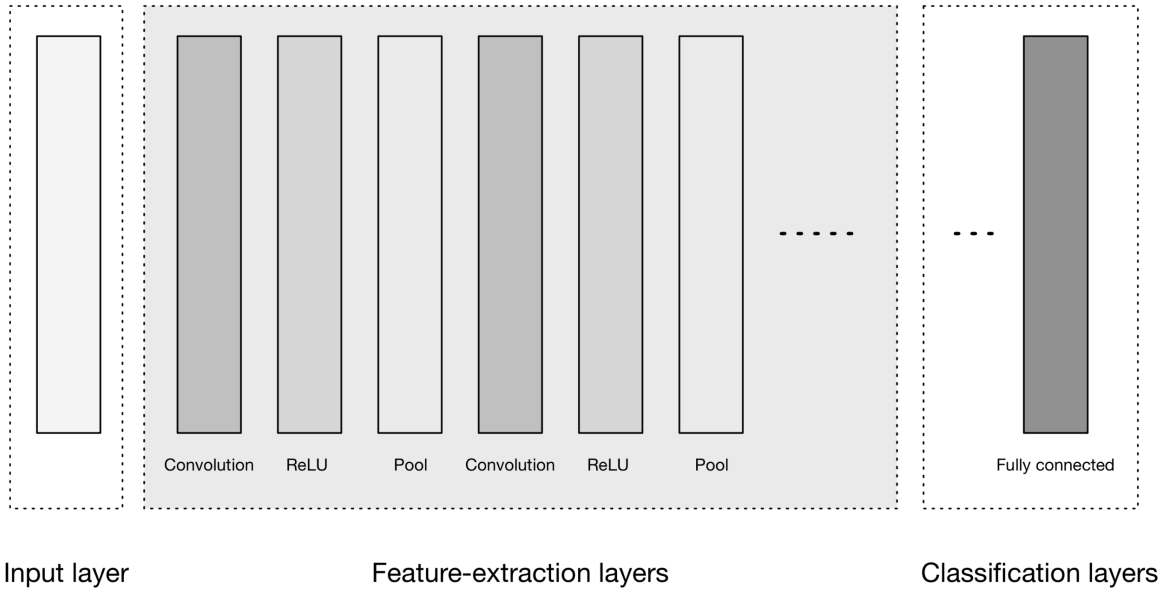


Figure 2.4: High-level general CNN architecture. (*credits to:* [41])

The three main parts depicted in the Figure 2.4 are: *input layer*, *feature-extraction layers*, and *classification layers*.

The *feature-extraction layers* contain operations that define CNNs: *convolution* and *pooling*, which progressively construct higher-order features by finding a number of features in the images. The process of automatically learning features is a key aspect of deep learning, as

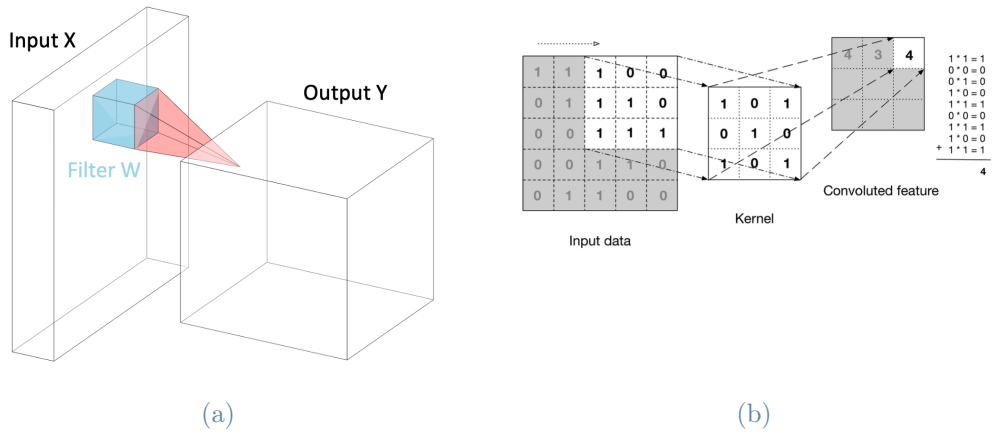
opposed to traditional hand-engineering methods.

Input Layer

The *input layer* typically accepts 3D input data, typically organized as a spatial grid with dimensions of width and height, corresponding to the image size, and a depth dimension representing the color channels.

Convolutional Layers

In convolutional layers, the input data undergoes a convolution operation which consists in sliding a small matrix called a kernel or filter over the input data, performing an element-wise multiplication between the filter and the corresponding part of the input data, and summing up the results to produce a single output value, as depicted in Figure 2.5. The filter is usually of a smaller size compared to the input data and is applied across the entire input volume to generate a feature map.



(a) (b)

Figure 2.5: Graphical representation of convolution layer and operation.

By sliding different filters over the input data, the convolutional layer produces multiple feature maps, each highlighting different patterns or features in the input data. The values making up the kernels are the weights to be learned by the net to extract features that are useful for the main task. To obtain multiple feature maps, the convolution operation is applied to the input tensor using multiple filters with different sets of weights. Each filter produces a separate feature map that captures a different aspect of the input data. These feature maps are stacked together to form the output tensor of the convolutional

layer. The convolution operation is a computationally efficient way to perform this feature extraction process, as it involves only a small number of learnable parameters and is easily parallelizable.

The expression

$$\mathbf{Y}_{ij} = \sum_m \sum_n \mathbf{X}_{mn} \mathbf{W}_{(i-m),(j-n)} \tag{2.4}$$

represents the convolution operation between an input tensor $\mathbf{X}$ and a kernel tensor $\mathbf{W}$, resulting in an output tensor $\mathbf{Y}$. The indices i and j correspond to the spatial position of the output tensor, while the indices m and n correspond to the spatial position of the kernel tensor. Specifically, for each position (i, j) in the output tensor, the kernel tensor is centered at position (m, n) in the input tensor, and the element-wise multiplication and summation are performed over all positions (m, n) that overlap with the kernel.

After obtaining the collection of feature maps from the convolutional layer, an activation function is applied to each element of each convolution output to introduce non-linearity into the model. This results in a set of "activation maps", where each activation map corresponds to a specific feature map. The most commonly used activation functions for convolutional layers are:

1. ReLU (Rectified Linear Unit): $g(z) = \max(0, z)$
2. Hyperbolic Tangent: $g(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
3. Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$
4. Leaky ReLU: $g(z) = \max(\alpha z, z)$, where α is a small positive value (e.g., 0.01)

By introducing non-linearity through activation functions, the convolutional neural network can learn more complex and abstract features from the input data.

In summary, convolutional layers play a crucial role in CNNs by extracting features from input data, and the convolution operation is performed using a filter that is slid over the input data. The output feature maps are then passed through activation functions, with ReLU being the most commonly used.

Pooling Layers

Pooling layers are used to reduce the spatial size of the feature maps and to introduce some amount of translation invariance to the learned features. The translational invariance is the

property that allows CNNs to recognize a feature regardless of its position in the image. A pooling layer takes a set of feature maps as input and outputs a smaller set of feature maps with reduced spatial resolution. The pooling process also utilizes a filter/kernel, albeit one without any elements. It essentially involves sliding this filter over sequential patches of the image and processing pixels caught in the kernel in some kind of way. This process is characterized by the filter size and the stride: this parameter determines how much a filter is shifted in either dimension when performing sliding window operations like convolution and pooling. There are different kinds of pooling operations. The most common type of pooling is max pooling, which divides the input feature map into non-overlapping rectangular regions and outputs the maximum value in each region, an example of which is depicted in Figure 2.6.

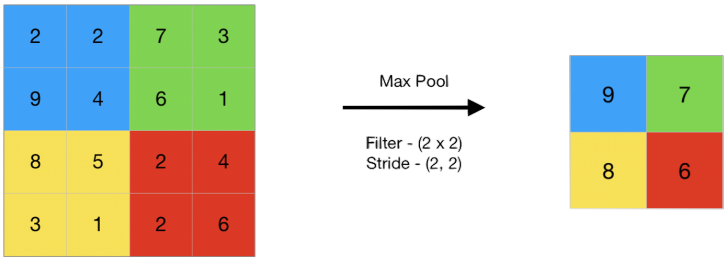


Figure 2.6: Max pooling example.

Other types of pooling include average pooling, which outputs the average value in each region, and L2 pooling, which outputs the L2 norm of the values in each region. The size of the pooling window is usually 2-3 pixels. This operation reduces the spatial size of the feature maps improving the computational efficiency of the network, and introduces some amount of translation invariance to the learned features making them more robust to small translations in the input image.

Classification Layers

The *classification layers* consist of one or more fully connected layers. These layers take the higher-order features and produce class probabilities or scores. As their name implies, these layers are fully connected to all neurons in the previous layer. The output of the fully connected layer is usually passed through a softmax activation function to ensure that the probabilities sum to one. During training, the weights of the fully connected layer are learned through backpropagation, and the cross-entropy loss is used to measure the difference between the predicted class probabilities and the true class labels. During inference, the final prediction is typically the class with the highest predicted probability.

The use of a classification layer allows the learned features to be transformed into a form that is suitable for making class predictions. The fully connected layer effectively computes a linear combination of the learned features, followed by a non-linear transformation through the softmax activation function. This allows the network to learn complex decision boundaries that can discriminate between the different classes in the dataset.

2.2. Object Detection and Pose Estimation

Object Detection

By *object detection* we refer to the process of localizing and classifying a *variable* number of objects of interest within an image, thus drawing *Bounding Boxes* (BBs) enclosing such objects. When the problem reduces to the task of finding and identifying a *fixed set* of objects, it is referred to as *object localization*. There are two main ways to tackle this problem: region proposal-based methods and one-stage methods. The first approach acts in 2 steps: in the first step, potential object regions are proposed (also called proposals); in the second step, the proposals are classified into object categories and refined to better localize the object. On the other hand, one-stage methods directly predict the presence and location of objects in a single step. One of the main drawbacks of region-based methods is that generating proposals can be computationally expensive [16].

In this work, although the object to be detected is just one, an object detection algorithm will be used as it is part of the architecture inherited from state-of-the-art CNNs. Furthermore, for the same reason, in this section we will focus on one-stage methods since the search for proposals would be excessive given the simplicity of the problem.

In this field, the use of *Anchor Boxes* (ABs) has become a standard technique. ABs are a mechanism used in object detection algorithms to help the model detect objects of different sizes and aspect ratios and are based on the idea of pre-defining a set of BBs with different sizes and aspect ratios, centered at various locations in an image. In order to more precisely localize objects in the image, during training, the object detection model learns to predict offsets to these pre-defined ABs. For each AB, the model learns to predict the offset values that minimize a loss function that assesses the discrepancy between the predicted and actual BB coordinates. The use of anchor boxes for object detection was introduced in 2015 by Ren, He, Girshick, and Sun in [48], where a two-stage object detection framework that uses a region proposal network (RPN) to generate object proposals using anchor boxes of different scales and aspect ratios is presented. However, the use of ABs has become a standard also in one-stage detection frameworks, the first

of which to adopt them was "SSD: Single Shot MultiBox Detector" [34], followed by the second version of the famous "You Only Look Once" network, YOLOv2 [47].

For ABs on a $S \times S$ grid, the object detection network output dimensionality would be $S \times S \times$ #ABs $\times$ (#classes + 5). This implies that for each grid cell, the neural network will produce a prediction represented by stacking up vectors like the following ${}^{AB_i}\mathbf{Y}_{ij}$ for each anchor box AB_i:

$$ {}^{AB_i}\mathbf{Y}_{ij} = \left[{}^{AB_i}p_c,\ {}^{AB_i}b_x,\ {}^{AB_i}b_y,\ {}^{AB_i}b_h,\ {}^{AB_i}b_w,\ {}^{AB_i}C_1,\ \ldots,\ {}^{AB_i}C_N\right]^\top \tag{2.5} $$

where p_c is the confidence score (i.e., the probability that the BB contains an object of interest), b_x and b_y are the BB center coordinates, and b_h and b_w are its height and width. The vector part from C_1 to C_N contains boolean variables and encodes the predicted class.

In order to eliminate overlapping detections and choose the most certain ones after an object detection model generates a set of candidate bounding boxes for each object instance, the post-processing step known as non-maximum suppression (NMS) is frequently used in object detection. This is how the algorithm operates:

1. Based on their confidence scores, the candidate bounding boxes are ordered.
2. All the other bounding boxes that have significant overlap (such as intersection over union (IoU) greater than a threshold) with the bounding box with the highest confidence score are removed.
3. Repeat step 2 until there are no more bounding boxes left.

The result of NMS is a set of non-overlapping bounding boxes that have high confidence scores and represent the detected objects in the image. By combining anchor boxes with non-maximum suppression, a highly effective and precise algorithm is produced, capable of detecting multiple objects within the same grid cell.

One of the most advanced methods for selecting anchor boxes involves performing K-means clustering [35] on the dataset.

Pose Estimation

Pose estimation is the task of estimating the position and the attitude (pose) of an object within an image. One way of facing pose estimation through CNNs is using a heatmap regression approach: this method involves predicting the probability of the presence of a specific keypoint in a specific location by generating a heatmap for each keypoint. Such

heatmaps are images in which the brighter the pixel, the more likely the presence of the associated keypoint. Another approach is to use a direct regression method: the keypoints coordinates are directly regressed without generating a heatmap. A third approach is to directly predict the pose in terms of orientation and translation, without using keypoint regression. The architecture of these networks typically consists of a backbone that extracts the features from the image, and one or a set of prediction heads (subnetworks) that predict the pose.

A state-of-the-art object detection and direct pose estimation method is EfficientPose [17], the approach used in SPNv2 and in the work presented in this dissertation. This architecture makes use of EfficientDet [56] as backbone, which will be presented here to better understand how the full configuration works.

2.2.1. EfficientDet

EfficientDet [56] is an object detection model introduced by Tan et al. in 2020, designed to achieve high accuracy and efficiency through its ability to scale the network architecture in a principled manner, and through the implementation of an efficient multi-scale feature fusion logic. The objective of multi-scale feature fusion is to integrate features at multiple resolutions. In other words, the aim is to find a transformation f that can effectively combine different features of a given list of multi-scale features $\vec{P}^{\text{in}} = (P^{\text{in}}_{l_1}, P^{\text{in}}_{l_2}, ...)$, where $P^{\text{in}}_{l_i}$ represents the feature at level l_i, to produce a new list of features $\vec{P}^{\text{out}} = f(\vec{P}^{\text{in}})$. The conventional top-down FPN shown in Figure 2.7(a) takes level 3-7 input features $\vec{P}^{\text{in}} = (P^{\text{in}}_3, ..., P^{\text{in}}_7)$ where P^{in}_i corresponds to a feature level with resolution of 2^{-i} of the input image. These multi-scale features are aggregated in a top-down manner:

$$\begin{aligned} P^{\text{out}}_7 &= Conv(P^{\text{in}}_7) \\ P^{\text{out}}_6 &= Conv(P^{\text{in}}_6 + Resize(P^{\text{out}}_7)) \\ &\dots \\ P^{\text{out}}_3 &= Conv(P^{\text{in}}_3 + Resize(P^{\text{out}}_4)) \end{aligned} \tag{2.6}$$

where $Resize$ is usually an upsampling or downsampling operation to match the features resolutions and $Conv$ is a convolution operation to process such features.

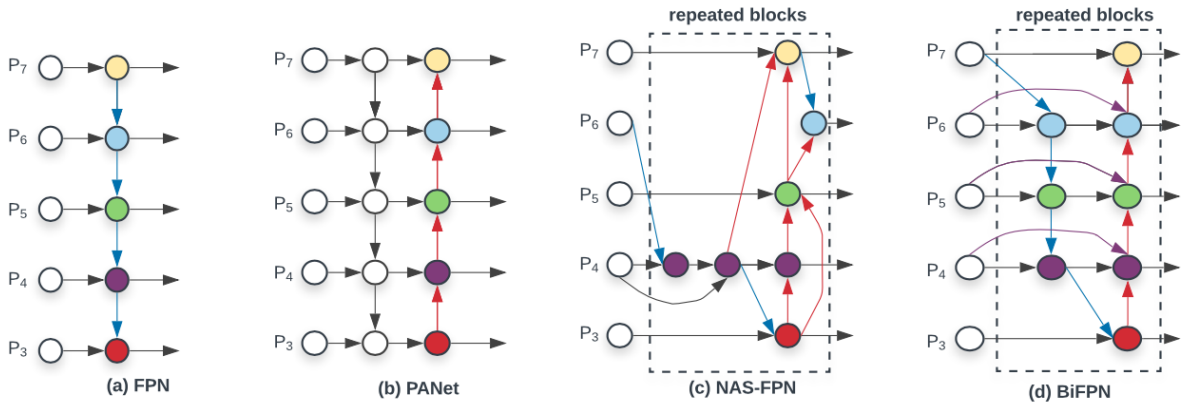


Figure 2.7: (a) FPN introduced a top-down pathway to fuse multi-scale features (P_3 - P_7); (b) PANet adds an additional bottom-up pathway on top of FPN; (3) NAS-FPN use neural architecture search to find irregular feature network topology and the repeatedly apply the same block; (d) BiFPN with better accuracy and efficiency trade-offs. (*credits to:* [56])

Non-conventional solutions presented before EfficientDet and sketched in Figure 2.7(b) and Figure 2.7(c) try to enhance the limited top-down approach typical of FPN. BiFPN takes the best of both architectures and implements a new flow of cross-scale connections between the features: the nodes with a single input edge are removed since they contribute less to the feature network, lowering the number of parameters and computations; furthermore, an extra edge is added from the original input to output node if they are at the same level, in order to fuse more features without adding much cost. Each bidirectional path is treated as one feature network layer and is repeated multiple times to enable more high-level feature fusion. Since different input features are at different resolutions, they usually contribute to the output feature unequally. Thus, BiFPN adopts *fast normalized fusion*, adding an additional weight for each input, and letting the network learn the importance of each input feature. As an example, here are described the two fused features at level 6 for BiFPN shown in Figure 2.7(d):

$$\begin{aligned} P_6^{\text{td}} &= Conv\left(\frac{w_1 \cdot P_6^{\text{in}} + w_2 \cdot Resize(P_7^{\text{in}})}{w_1 + w_2 + \epsilon}\right) \\ P_6^{\text{out}} &= Conv\left(\frac{w_1' \cdot P_6^{\text{in}} + w_2' \cdot P_6^{\text{td}} + w_3' \cdot Resize(P_5^{\text{out}})}{w_1' + w_2' + w_3' + \epsilon}\right) \end{aligned} \tag{2.7}$$

where $w_i \geq 0$ is ensured by applying ReLU to each w_i, and ϵ =1e−4 is to avoid numerical instability. In the proposed example, P_6^{td} is the intermediate feature at level 6 on the top-down pathway, and P_6^{out} is the output feature at level 6 on the bottom-up pathway.

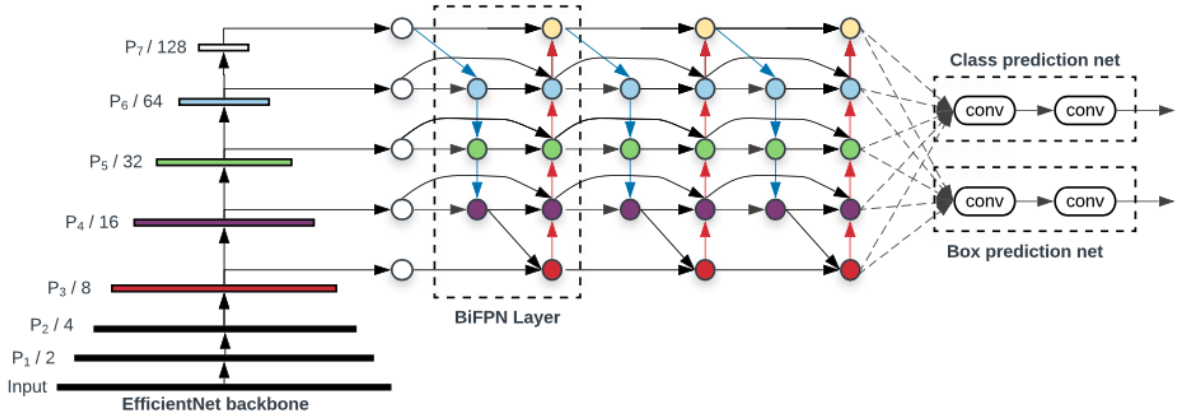


Figure 2.8: EfficientDet architecture. (*credits to:* [56])

The full EfficientDet architecture is presented in Figure 2.8: it employs EfficientNet [55] as the backbone network, and classification/bounding box prediction subnetworks. The peculiarity of EfficientNet is its ability to be scaled maintaining optimal accuracy and efficiency: this feature is called "compound scaling" and, unlike conventional practice that arbitrarily scales the network width, depth, and/or resolution of the input image, it is based on scaling these factors uniformly with a scaling coefficient ϕ. The principle on which the scaling is based is expressed in the following Equation 2.8:

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2.8}$$

The regular convolution operation computational cost expressed in FLOPS (*FLoating point Operations Per Second*) is directly proportional to d, w^2, and r^2. If we double the network depth, the FLOPS doubles, and if we double the network width or resolution, the FLOPS increases by a factor of four. Since convolution operations consume most of the computation cost in CNNs, scaling a CNN using Equation 2.8 will roughly increase the total FLOPS by $(\alpha \cdot \beta^2 \cdot \gamma^2)^{\phi}$. As stated in Equation 2.8, the original paper constrains $(\alpha \cdot \beta^2 \cdot \gamma^2) \approx 2$ so that, for any ϕ, the total FLOPS will increase by roughly 2^{ϕ}.

Table 2.1: EfficientNet-B0 baseline network.

Stage i	**Operator** $\hat{\mathcal{F}}_i$	**Resolution** $\hat{H}_i \times \hat{W}_i$	**#channels** $\hat{C}_i$	**#layers** $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 2.1 shows the architecture of the baseline architecture EfficientNet-B0, whose main building block is the mobile inverted bottleneck MBConv [50, 57], optimized by the squeeze-and-excitation method [26]. Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\hat{H}_i \times \hat{W}_i$ and output channels $\hat{C}_i$. In order to adapt EfficientDet to the same scaling flexibility of EfficientNet, the width, and depth of the BiFPN are scaled with the following equation [56]:

$$\begin{aligned} W_{\text{BiFPN}} &= 64 \cdot \left(1.35^{\phi}\right) \\ D_{\text{BiFPN}} &= 3 + \phi \end{aligned} \tag{2.9}$$

and the width of the box/class prediction network is fixed to be the same as BiFPN, while the depth of the box network is linearly increased as shown in Equation 2.10 [56].

$$\begin{aligned} W_{\text{pred}} &= W_{\text{BiFPN}} \\ D_{\text{box}} &= 3 + \lfloor \phi / 3 \rfloor \end{aligned} \tag{2.10}$$

Since BiFPN uses the features from levels 3 to 7, the network input resolution must be dividable by $2^7 = 128$, thus the resolution is scaled using the following equation [56]:

$$R_{\text{input}} = 512 + \phi \cdot 128 \tag{2.11}$$

Starting from Equations 2.9, 2.10, 2.11, a series of scaled networks can be developed. In Table 2.2, EfficientDet D0 to D4 are reported.

Table 2.2: Scaling configurations for EfficientDet D0-D4.

	Input size R_{input}	**Backbone Network**	**BiFPN** **#channels** W_{BiFPN}	**#layers** D_{BiFPN}	**Box/class #layers** D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4

2.2.2. EfficientPose

EfficientPose [17] is an approach for 6D pose estimation designed to be both fast and accurate. The architecture is made of an EfficientDet backbone extended through two subnetworks: these are analogous to the classification and bounding box regression subnetworks, but instead of predicting the class and BB for each AB, predict the rotation $\mathbf{R}$ and the translation $\mathbf{t}$ respectively. Since they share the input feature maps with the already existing subnets in EfficientDet, the additional computational cost is very low. Thus, EfficientPose can efficiently detect the

- Class
- 2D bounding box
- Rotation
- Translation

of one or more object instances and categories for a given image in a single shot. A high-level view of the architecture is presented in Figure 2.9. In order to maintain the scalability of EfficientDet, the size of the rotation and translation networks is also controlled by the scaling parameter ϕ.

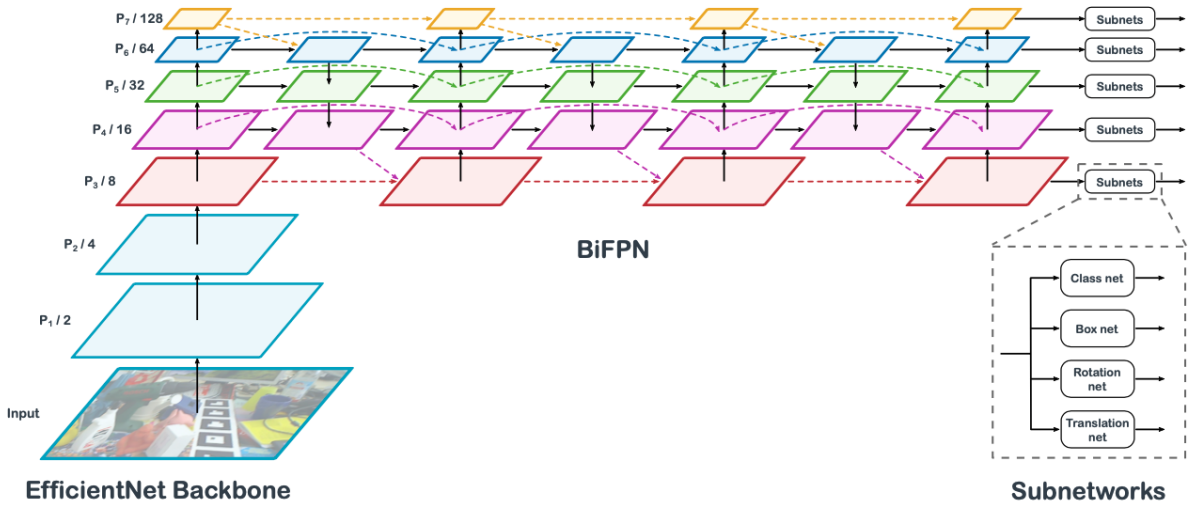


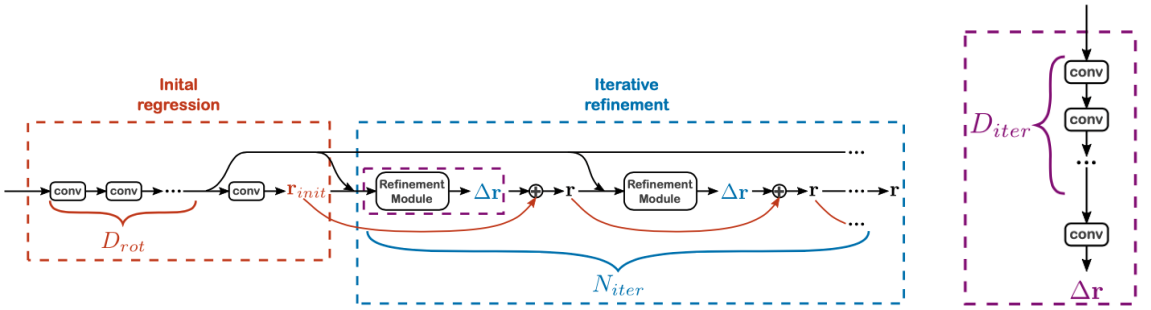
Figure 2.9: Schematic representation of EfficientPose architecture including the EfficientDet backbone and the prediction subnetworks. (*credits to:* [17])

Rotation Network

The rotation network is responsible for predicting the rotation of an object in 3D space and is designed to be efficient, accurate, and scalable. Throughout the description of this subnet, an axis-angle representation of the rotation $\mathbf{r} \in \mathbb{R}^3$ will be used. Yet, as stated in the original EfficientPose paper, this representation is not crucial for the developed approach and can be switched. The architecture is similar to the classification and box network in EfficientDet, but instead of using the output $\mathbf{r}_{\text{init}}$ directly as the regressed rotation, an iterative refinement module inspired by Kanazawa *et al.* [28] is added. This module takes the concatenation along the channel dimension of the current rotation $\mathbf{r}_{\text{init}}$ and the output of the last convolution layer prior to the initial regression layer which outputs $\mathbf{r}_{\text{init}}$ as the input and regresses $\Delta\mathbf{r}$ so that the final rotation regression is $\mathbf{r} = \mathbf{r}_{\text{init}} + \Delta\mathbf{r}$.

As depicted in Figure 2.10, the iterative refinement module consists of D_{iter} depthwise separable convolution layers, each followed by group normalization [58] and SiLU activation function [46]. Lastly, a single depthwise separable convolution layer with a linear activation function outputs $\Delta\mathbf{r}$. Both the number of times this module is applied N_{iter} and the single refinement module depth D_{iter} are scaled as in the following equation:

$$\begin{aligned} D_{\text{iter}} &= 2 + \lfloor \phi/3 \rfloor \\ N_{\text{iter}} &= 1 + \lfloor \phi/3 \rfloor \end{aligned} \tag{2.12}$$



(a) Rotation network architecture. (b) Ref. module.

Figure 2.10: EfficientPose rotation subnetwork architecture and refinement module. Each conv block consists of a depthwise separable convolution [22] layer followed by group normalization and SiLU activation. (*credits to:* [17])

Translation Network

The translation subnetwork is basically the same as the rotation one, but it outputs a translation $\mathbf{t} \in \mathbb{R}^3$ for each anchor box. Here, the approach of PoseCNN [59] is adopted: instead of directly regressing all the components of the translation vector $\mathbf{t} = (t_x, t_y, t_z)^\top$, the prediction of the 2D center point $\mathbf{p} = (p_x, p_y)^\top$ and of the depth coordinate t_z are treated as two different tasks. Assuming a pinhole camera with principal point $\mathbf{c} = (c_x, c_y)^\top$ and focal lengths (f_x, f_y), the missing translation components can then be retrieved using the following Equation 2.13.

$$\begin{aligned} t_x &= \frac{(p_x - c_x) \cdot t_z}{f_x} \\ t_y &= \frac{(p_y - c_y) \cdot t_z}{f_y} \end{aligned} \tag{2.13}$$

To predict the 2D center point, for each AB the subnet predicts the offset in pixels from the center of the AB to the center point of the corresponding object: these offsets are normalized with the stride of the input feature map from every level of the feature pyramid.

The approach of predicting the relative offset at each point in the feature maps is preferred over directly regressing the absolute coordinates p_x and p_y due to the translational invariance of the convolution. Because the convolution operation is translationally invariant, predicting the relative offset is more efficient and robust than predicting the absolute coordinates, which would require predicting the precise location of an object in the image.

2.3. Perspective-n-Point problem

The Perspective-n-Point problem is a computer vision problem that involves estimating the pose of an object from a set of n 3D points of a known model in its body frame and their corresponding 2D projections in an image taken by a calibrated camera, by mapping the 3D points to their 2D projections [24].

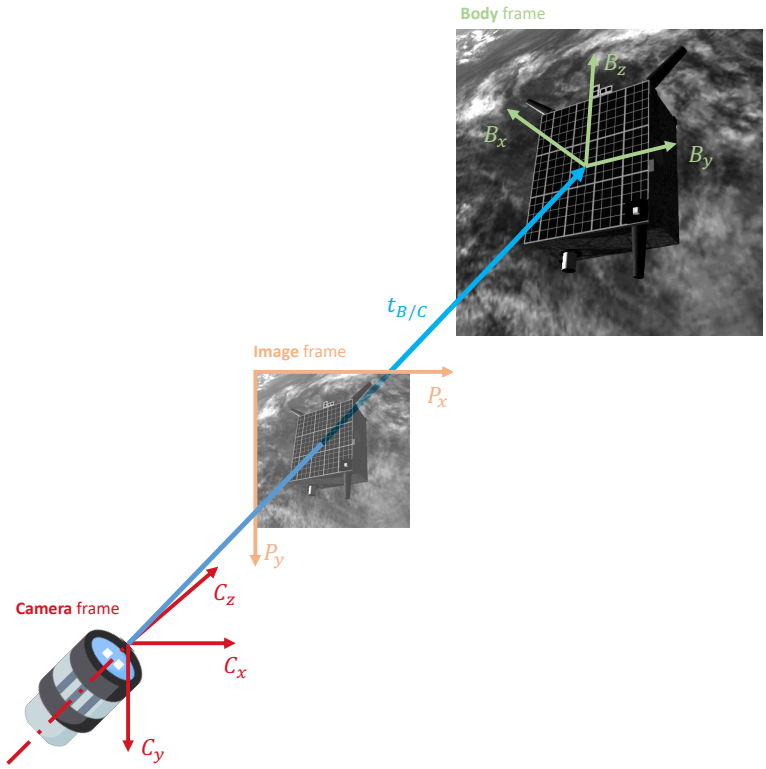


Figure 2.11: Reference frames involved in the PnP problem.

Taking Figure 2.11 as a reference, let's define as C the camera frame, P the image frame, and B the target body frame. C_z is aligned with the camera boresight, while C_x and C_y are parallel to P_x and P_y, respectively. Let $\mathbf{t}_{B/C}$ be the translation vector from the camera frame to the body frame in C, and let $\mathbf{R}_{B/C}$ be the DCM expressing the rotation to align B to C. Thus, the coordinates of a generic point $\mathbf{r}^B$ of the target can be expressed in the camera frame as:

$$\mathbf{r}^C = \begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} = \mathbf{R}_{B/C}\mathbf{r}^C + \mathbf{t}_{B/C} \tag{2.14}$$

By referring to the pinhole camera model, let's call (f_x, f_y) the camera focal lengths, and (c_x, c_y) the camera principal points. The generic target point expressed in the camera frame $\mathbf{r}^C$ can be projected onto the image frame P as:

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{x^C}{z^C} f_x + c_x \\ \\ \frac{y^C}{z^C} f_y + c_y \end{bmatrix} \tag{2.15}$$

The equations for perspective projection can be obtained by combining Equations 2.14 and 2.15, then rewriting them in terms of homogeneous coordinates:

$$\begin{bmatrix} uw \\ vw \\ w \end{bmatrix} = \mathbf{K} \underbrace{\begin{bmatrix} \mathbf{R}_{B/C} & \mathbf{t}_{B/C} \end{bmatrix}}_{\text{unknown: } \mathbf{P}} \begin{bmatrix} \mathbf{r}^B \\ 1 \end{bmatrix}, \tag{2.16}$$

where

$$\mathbf{K} := \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

is called the camera intrinsic matrix.
The matrix $\mathbf{P}$, which represents the problem's unknown, has 6 degrees of freedom. The relative attitude must be described by three parameters (e.g. Euler angles), and the relative translation must be described by three other parameters. Modern PnP solvers can be broadly categorized into two groups:

- iterative solvers, which minimize a measure of the fit error between the projected model points and the image points;
- multi-stage analytical methods, which use a linearized version of the projection equations.

Since the work presented in this dissertation uses the Efficient Perspective-n-Point (EPnP)

algorithm, which is a multi-stage analytical method, we will focus on this one.

2.3.1. EPnP: Efficient Perspective-n-Point

In a multi-stage analytical approach, the PnP problem is broken down into multiple sub-problems or stages, and each is solved analytically. These stages typically involve estimating the camera's internal parameters (intrinsic matrix) and its external parameters (rotation and translation), as well as minimizing the reprojection error between the observed 2D image coordinates and the predicted 2D coordinates of the 3D points based on the estimated camera pose. One popular multi-stage analytical method for solving the PnP problem is the EPnP (Efficient Perspective-n-Point) algorithm [32].

As summarized in [42], EPnP requires $n \geq 4$ point correspondences and its main idea is to express the n points as the weighted sum of 4 virtual control points $\{\mathbf{c}_j\}_{j=1,\dots,4}$ that become the unknowns of this formulation:

$$\mathbf{r}_i = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j \quad (i = 1, 2, \dots, n) \tag{2.18}$$

Equation 2.16 can be rewritten for a generic landmark i in terms of the 4 control points. Since every control point is described by 3 coordinates $\mathbf{c}_j = [c_j^x, c_j^y, c_j^z]^\top$, we end up with 12 control point coordinates:

$$\begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \\ \tilde{w}_i \end{bmatrix} = \mathbf{K} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} c_j^x \\ c_j^y \\ c_j^z \end{bmatrix} \tag{2.19}$$

By substituting the third equation into the other two, the following equations can be obtained for each model-point/image-point pair:

$$\begin{aligned} \sum_{j=1}^{4} \alpha_{ij} f_x c_j^x + \alpha_{ij}(u_0 - \tilde{u}_i) c_j^z = 0 \\ \sum_{j=1}^{4} \alpha_{ij} f_y c_j^y + \alpha_{ij}(v_0 - \tilde{v}_i) c_j^z = 0 \end{aligned} \tag{2.20}$$

Thus, Equation 2.20 yields $2n$ equations that can be arranged in matrix form as $\mathbf{A}\mathbf{x} = \mathbf{0}$, where $\mathbf{A}$ is $(2n \times 12)$ and the unknown vector $\mathbf{x}$ contains the 12 control point coordinates. Since the image points are estimated and are not perfect projections of the true model

points, **A** may have up to 4 linearly dependent columns, meaning that there could be 4 possible solutions. Among these, the one associated with the lowest reprojection error is selected.

2.4. Relative Orbital Dynamics

In this section, we will develop the general relative orbit Equations of Motion (EOM). The description of such dynamics and all the steps that will be shown are developed in "Analytical Mechanics of Space Systems" (Schaub & Junkins, 2003) [51].

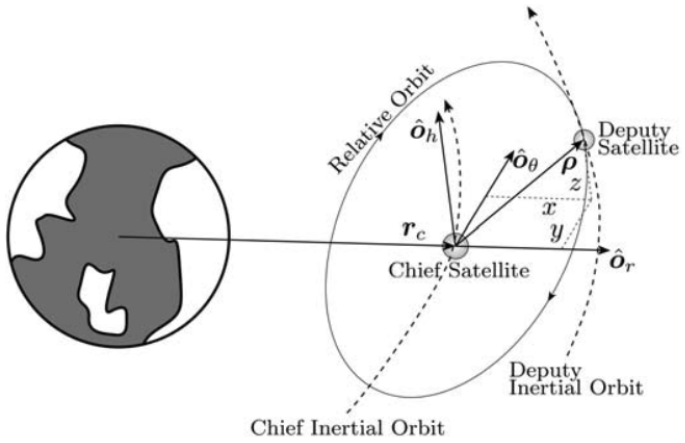


Figure 2.12: Illustration of a closed relative orbit. (*credits to:* [51])

In the framework shown in Figure 2.12, the problem is treated like a chief-deputy system, deriving from the formation flying nomenclature. Referring to the work presented in this dissertation, the chief can be seen as the satellite with the camera, and the deputy can be seen as the target spacecraft. The inertial chief position is $\mathbf{r}_c(t)$, while the inertial deputy position is given by $\mathbf{r}_d(t)$. To express the deputy relative orbit as seen from the chief, we introduce the Hill coordinate frame that we will call H. It is defined in Equation 2.21, where $\hat{\mathbf{o}}_r$ is a vector in the orbit radius direction, $\hat{\mathbf{o}}_h$ is parallel to the orbit momentum vector in the orbit normal direction, and $\hat{\mathbf{o}}_\theta$ completes the right-hand coordinate system.

$$\begin{aligned} \hat{\mathbf{o}}_r &= \frac{\mathbf{r}_c}{r_c} \\ \hat{\mathbf{o}}_\theta &= \hat{\mathbf{o}}_r \times \hat{\mathbf{o}}_h \\ \hat{\mathbf{o}}_h &= \frac{\mathbf{h}}{h} = \frac{\mathbf{r}_c \times \dot{\mathbf{r}}_c}{\|\mathbf{r}_c \times \dot{\mathbf{r}}_c\|} \end{aligned} \tag{2.21}$$

The relative orbit position vector $\boldsymbol{\rho}$ is expressed in H frame components as:

$$\boldsymbol{\rho} = \begin{bmatrix} x, & y, & z \end{bmatrix}^\top \tag{2.22}$$

The (x, y) coordinates define the relative orbit motion in the chief orbit plane, the z coordinate defines any motion out of the chief orbit plane. The deputy satellite position vector can be rewritten as:

$$\mathbf{r}_d = \mathbf{r}_c + \boldsymbol{\rho} = (r_c + x)\hat{\mathbf{o}}_r + y\hat{\mathbf{o}}_\theta + z\hat{\mathbf{o}}_h \tag{2.23}$$

Knowing that the angular velocity vector of the Hill frame H relative to the inertial frame N is given by $\boldsymbol{\omega}_{H/N} = \dot{\nu}\hat{\mathbf{o}}_h$ with ν being the chief frame true anomaly, the second time derivative of $\mathbf{r}_d$ in the inertial frame is given by:

$$\begin{aligned} \ddot{\mathbf{r}}_d = &(\ddot{r}_c + \ddot{x} - 2\dot{y}\dot{\nu} - \ddot{\nu}y - \dot{\nu}^2(r_c + x))\hat{\mathbf{o}}_r \\ &+ (\ddot{y} + 2\dot{\nu}(\dot{r}_c + \dot{x}) + \ddot{\nu}(r_c + x) - \dot{\nu}^2 y)\hat{\mathbf{o}}_\theta + \ddot{z}\hat{\mathbf{o}}_h \end{aligned} \tag{2.24}$$

The chief orbit angular momentum magnitude is given by $h = r_c^2\dot{\nu}$, and since we can assume Keplerian motion, $\dot{h} = 0$. Thus,

$$\dot{h} = 2r_c\dot{r}_c\dot{\nu} + r_c^2\ddot{\nu} = 0 \rightarrow \ddot{\nu} = -2\frac{\dot{r}_c}{r_c}\dot{\nu} \tag{2.25}$$

Then, we can express the chief acceleration vector as in Equation 2.26 by writing the chief satellite position as $\mathbf{r}_c = r_c\hat{\mathbf{o}}_r$ and by taking two time derivatives with respect to the inertial frame, where μ is the main attractor's gravitational parameter.

$$\ddot{\mathbf{r}}_c = (\ddot{r}_c - r_c\dot{\nu}^2)\hat{\mathbf{o}}_r = -\frac{\mu}{r_c^3}\mathbf{r}_c = -\frac{\mu}{r_c^2}\hat{\mathbf{o}}_r \tag{2.26}$$

Equating the vector components in Equation 2.26, the chief orbit radius acceleration is expressed as:

$$\ddot{r}_c = r_c\dot{\nu}^2 - \frac{\mu}{r_c^2} = r_c\dot{\nu}\left(1 - \frac{r_c}{p}\right) \tag{2.27}$$

where p is the chief orbit semi-latus rectum.
Substituting Equations 2.25 and 2.27 into Equation 2.24, the deputy acceleration is reduced to:

$$\begin{aligned}\ddot{\mathbf{r}}_d = &\left(\ddot{x} - 2\dot{\nu}\left(\dot{y} - y\frac{\dot{r}_c}{r_c}\right) - x\dot{\nu}^2 - \frac{\mu}{r_c^2}\right)\hat{\mathbf{o}}_r \\ &+ \left(\ddot{y} + 2\dot{\nu}\left(\dot{x} - x\frac{\dot{r}_c}{r_c}\right) - y\dot{\nu}^2\right)\hat{\mathbf{o}}_\theta + \ddot{z}\hat{\mathbf{o}}_h\end{aligned} \tag{2.28}$$

Let's now consider the orbit EOM for the two-body problem, for a generic orbit of radius $\mathbf{r}$:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{2.29}$$

By substituting the kinematic acceleration expression in Equation 2.28 into Equation 2.29, we retrieve the deputy satellite EOM:

$$\ddot{\mathbf{r}}_d = -\frac{\mu}{r_d^3}\begin{bmatrix} r_c + x \\ y \\ z \end{bmatrix}^H \tag{2.30}$$

with $r_d = \sqrt{(r_c + x)^2 + y^2 + z^2}$. Thus, by equating Equations 2.28 and 2.30, we obtain the exact nonlinear relative EOM for the deputy satellite in the Hill frame of the chief satellite as expressed in Equation 2.31.

$$\begin{aligned}\ddot{x} &= 2\dot{\nu}\left(\dot{y} - y\frac{\dot{r}_c}{r_c}\right) + x\dot{\nu}^2 + \frac{\mu}{r_c} - \frac{\mu(r_c + x)}{((r_c + x)^2 + y^2 + z^2)^{3/2}} \\ \ddot{y} &= -2\dot{\nu}\left(\dot{x} - x\frac{\dot{r}_c}{r_c}\right) + y\dot{\nu}^2 - \frac{\mu y}{((r_c + x)^2 + y^2 + z^2)^{3/2}} \\ \ddot{z} &= -\frac{\mu z}{((r_c + x)^2 + y^2 + z^2)^{3/2}}\end{aligned} \tag{2.31}$$

2.5. Kalman Filtering

In the field of navigation, filtering is used to estimate the state of a system based on measurements. One of the most commonly used filters is the Kalman filter, which is a mathematical algorithm that uses a series of measurements over time, along with knowledge of the system's dynamics and noise characteristics, to estimate the system's state with minimal error [27]. The Kalman filter is widely used in aerospace applications, including spacecraft navigation [43].

The Kalman filter operates in two stages: prediction and correction. In the prediction

stage, the filter uses the current estimate of the system state to predict its state at the next time step, based on the known system dynamics. In the correction stage, the filter updates the predicted state estimate with new measurements, using a weighting factor that depends on the relative uncertainty of the predicted state and the measurement.

In some cases, the system being estimated may be characterized by nonlinear dynamics, or the measurement model may be nonlinear. When this happens, an Extended Kalman filter (EKF) or an Unscented Kalman Filter (UKF) can be used, depending on the nonlinearity degree of the model. Since in the presented work the filtering is applied only to the translation predictions, and by referring to the underlying dynamics expressed in Section 2.4, only the EKF will be treated because it is considered enough to deal with such nonlinearity.

2.5.1. Extended Kalman Filter

The EKF linearizes the nonlinear functions around the current state estimate and measurement and applies the Kalman filter algorithm to the resulting linear system [36]. Let us consider a generic nonlinear model for the state $\mathbf{x}$ in Equation 2.32, where $\mathbf{u}_k$ represents the input at the time step k, $f(\cdot)$ is the state transition function, $h(\cdot)$ is the measurement function which maps the state to the measured quantity $\mathbf{y}$, $\mathbf{w}_k$ and $\mathbf{v}_k$ identify the process and measurement noise respectively.

$$\begin{cases} \mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right) \\ \mathbf{y}_k = h\left(\mathbf{x}_k, \mathbf{v}_k\right) \end{cases} \tag{2.32}$$

To linearize the state transition and the measurement functions, their Jacobian matrices can be computed as:

$$\mathbf{F} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}^+} \qquad\qquad \mathbf{H} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}^-} \tag{2.33}$$

The prediction phase consists of the propagation of both the state $\hat{\mathbf{x}}_k^+$ and the related covariance $\hat{\mathbf{P}}_k^+$ based on their values at the previous time step as follows:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1}^- &= f\left(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, 0\right) \\ \mathbf{P}_{k+1}^- &= \mathbf{F}\mathbf{P}_k^+\mathbf{F}^\top + \mathbf{Q}_k \end{aligned} \tag{2.34}$$

where $\mathbf{Q}_k$ represents the covariance matrix associated with process noise. The correction

phase, reported in Equation 2.35, exploits the incoming measurements and an optimal weighting factor called Kalman gain and $\mathbf{K}_{k+1}$.

$$\begin{aligned}
\mathbf{K}_{k+1} &\coloneqq \mathbf{P}_{k+1}^{-}\mathbf{H}^{\top}\left(\mathbf{H}\mathbf{P}_{k+1}^{-}\mathbf{H}^{\top}+\mathbf{R}_{k+1}\right)^{-1} \\
\hat{\mathbf{x}}_{k+1}^{+} &= \hat{\mathbf{x}}_{k+1}^{-}+\mathbf{K}_{k+1}\left(\mathbf{y}_{k+1}-h\left(\hat{\mathbf{x}}_{k+1}^{-},0\right)\right) \\
\mathbf{P}_{k+1}^{+} &= \left(\mathbf{I}-\mathbf{K}_{k+1}\mathbf{H}\right)\mathbf{P}_{k+1}^{-}
\end{aligned} \tag{2.35}$$

Note that and $\mathbf{R}_k$ is the covariance matrix associated with measurement noise. The filter consists in a recursive algorithm, so the outputs become the inputs of the successive iteration.

The application of an EKF fits the relative orbital dynamics introduced in Section 2.4 because the model does not present a high level of nonlinearity. When this is not the case, the EKF should be avoided because the linearization could lead to an inaccurate propagation of the state. Instead, the Unscented Kalman Filter (UKF) [37] overcomes these problems by introducing three tuning parameters and by relying on an Unscented Transform (UT) for the propagation of the mean and the covariance state. This type of filter will not be treated, because the pose estimation pipeline implemented in the work discussed in this dissertation introduces only filtering for the position of the target, thus an EKF is sufficient.

3 | AIKO-NET

In this chapter we present AIKO-NET, a Convolutional Neural Network (CNN) developed to enhance the current state-of-the-art MTL-based pose estimation of uncooperative spacecrafts through monocular images.

Recalling the problem statement, the objective of AIKO-NET is to estimate the position and attitude of the body frame B of the target spacecraft relative to the camera frame C. Figure 3.1 illustrates that $\mathbf{t}_{B/C}$ represents the relative position between the origins of the target's body reference frame and the camera's reference frame. Likewise, $\mathbf{q}_{B/C}$ denotes the quaternion that describes the rotation required to align the target's body reference frame with the camera's reference frame.

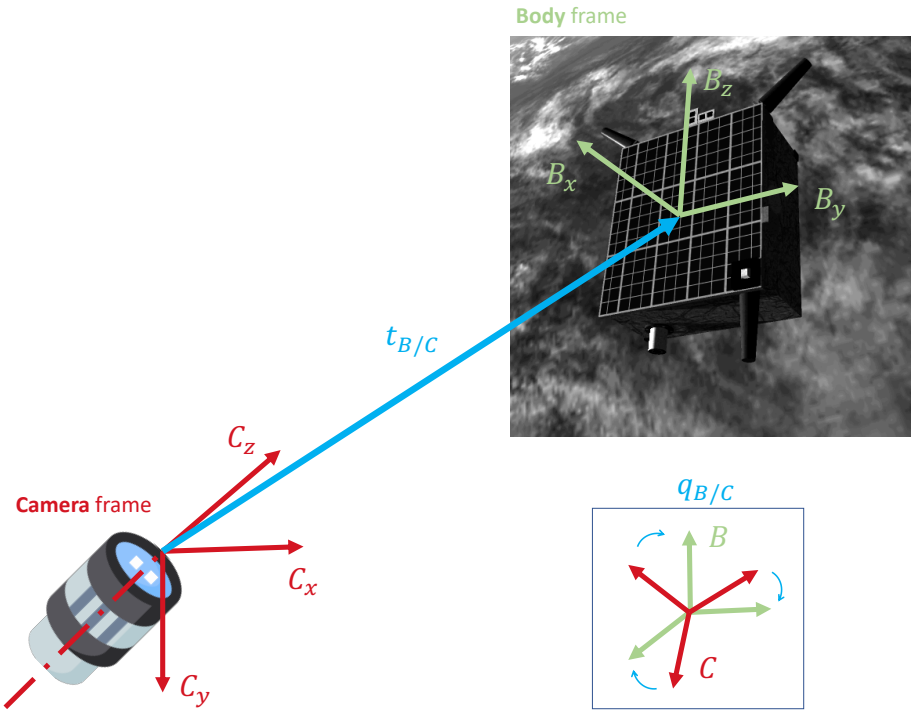


Figure 3.1: Definition of the reference frames, relative position, and relative attitude.

The main purposes of AIKO-NET are to demonstrate the reproducibility of the declared multi-task learning related improvements of SPNv2 in [39], and to push such MTL nature of the network to further improve the estimation accuracy by exploiting a researched synergy between different prediction heads. The presented architecture is a deep multi-task network that is built on top of the SPNv2, which serves as the baseline as discussed in Section 1.2.2. New features and the respective prediction heads are introduced, aiming at improving the network's accuracy by assigning it new parallel tasks to achieve, which may influence each other and ultimately lead to a better minimum with respect to a single-task network, as explained in Section 2.1.3.

It is important to note that the baseline has not only been modified by adding prediction heads, as the public code has a strong focus on inference, which means that it lacks many features for the evaluation of training performance. The work from SLAB is highly structured and interconnected, which has made it challenging to incorporate the necessary modules for training evaluation. Nonetheless, the architecture's flexibility has made it easy to implement new prediction heads and parallelize them with the main task.

Obviously, wanting to add tasks to solve, and therefore other features to identify, implies that ground-truths must exist for these features. This is why the SPEED+ dataset could not be used to train AIKO-NET. A completely customized dataset had to be generated to implement the multi-task approach, and also pre-processing and dataset generation pipelines were developed in order to build an extremely flexible and interconnected framework which finally led to ease of use and modularity of the whole network and training conditions. The dataset from AIKO takes Tango from the PRISMA mission as the target, but the underlying idea is that these datasets can be generated easily for any available 3D model.

The ODR technique used in SPNv2 is not considered in the presented AIKO-NET version because it goes beyond the scope of the current research.

3.1. Custom Dataset

Given the data-centric nature of this project and the need to propose an architecture that uses multi-task learning, a fully tailored dataset was developed and utilized to train, validate and test AIKO-NET. This dataset, namely the Multi-Feature Spacecraft Pose Estimation Dataset (MFSPED), was created especially for the needs of this investigation with Unity[1]. The MFSPED consists of 60,000 Tango synthetic images with unique metadata and feature

[1]All 3D simulations and visualization were performed in Unity version 2022.2.1 - https://unity.com/

maps, as well as 20,000 Earth background images. With MFSPED we will not refer to the dataset used for the training and evaluation of AIKO-NET: MFSPED contains the building blocks which can be used at will to generate the actual dataset used for the network. The full modularity of this project makes the dataset inherently flexible and tunable. The same MFSPED can be modified in order to tweak the camera parameters, change all the pose labels distributions, or even by changing the satellite model. This flexibility may help if different operative conditions are to be met, and to use the same dataset generation logic for other applications. In this context, this project can be included in a larger domain that exploits *data exploration* processes to efficiently train a network. Working with data-centric AI (DCAI) [60] and having complete control of a dataset, makes it possible to build accurate and robust models.

Building a dataset for multi-task learning purposes on a CNN involves brainstorming some feature maps that could help the network in solving its main task. In addition to the features used in SPNv2, three other features have been thought of:

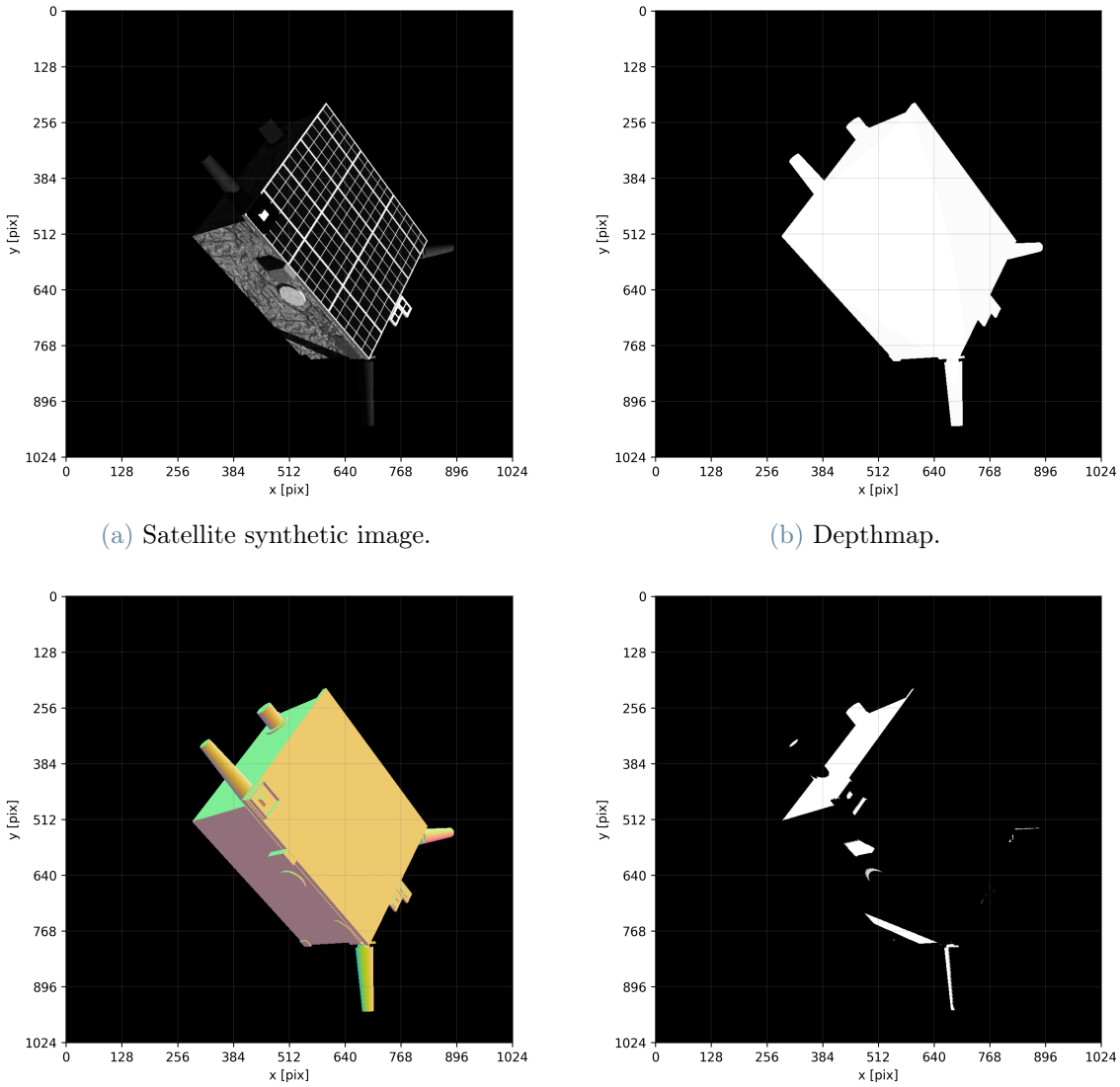
a) **Depthmap**: a segmentation mask of the satellite with additional information about the distance from the camera. The closer to the camera, the whiter the mask pixels; the further from it, the darkest.

b) **Normalmap**: a colormap related to the normal directions to the satellite surfaces in the target's body frame: the same surface results associated with the same color for every image in the dataset.

c) **Shadowmap**: an image where the shadowed part of the satellite corresponds to white pixels. This feature should help the net to better manage different lighting conditions.

An example of these maps, associated with an example image of the dataset, is shown in Figure 3.2.
Regarding the Depthmap, the effect of the gray scaling is not really noticeable with an object of small dimensions like Tango[2]. Thus, for this specific application, the Depthmap will be substituting the Segmentation mask that is used in SPNv2, but the additional information provided by this new feature map remains valuable for bigger and more complex objects.

[2] $[295 \times 570 \times 740]$ mm; data retrieved from: "Flight Results From PRISMA Formation Flying and Rendezvous Demonstration Mission" (*S. Persson et al.*, 2010). https://core.ac.uk/display/11152525



(a) Satellite synthetic image.

(b) Depthmap.

(c) Normalmap.

(d) Shadowmap.

Figure 3.2: An example of a synthetic image of Tango and the associated feature maps.

The dataset images are associated with a final feature derived from SPNv2, namely the **keypoints Heatmap**. This feature provides 2D heatmaps associated with each keypoint of the synthetic images. Differently from SPNv2, the selected keypoints here are 18 and are displayed on the model in Figure 3.3. The choice of doubling the number of keypoints to teach to the network aims at giving the CNN more reference points that could be useful in the context of MTL.

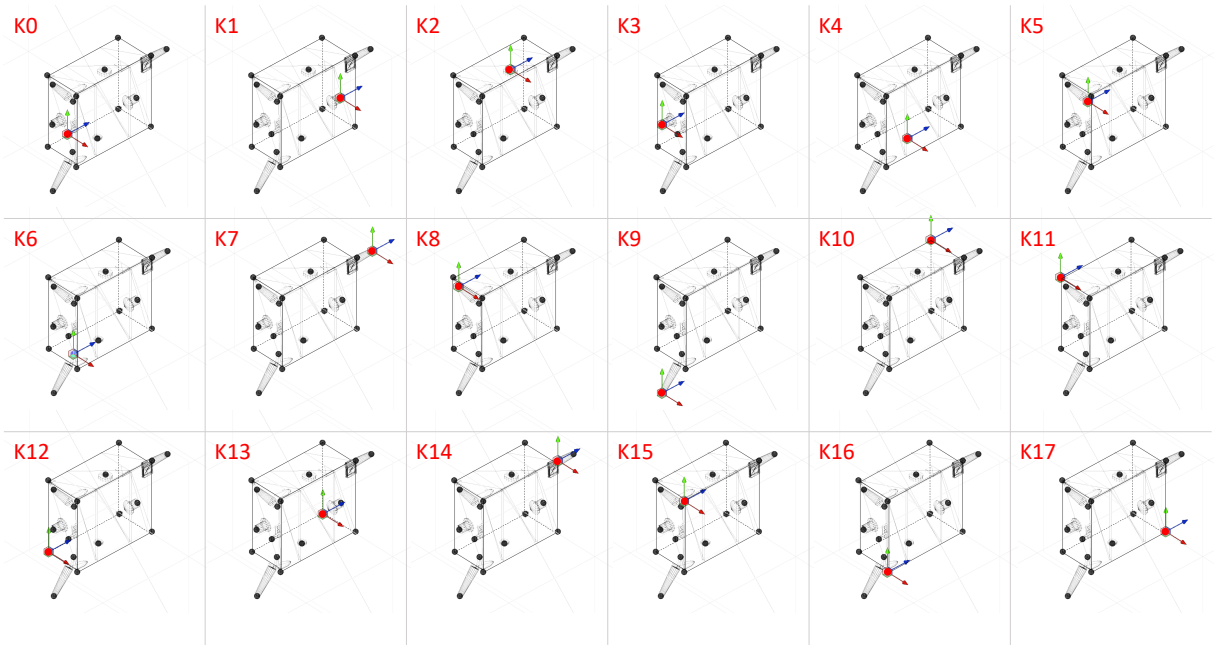
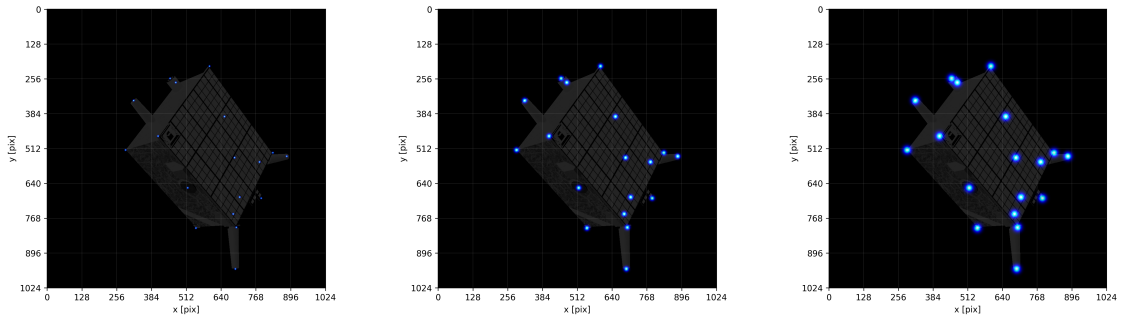


Figure 3.3: Model of Tango and the selected 18 keypoints.

The ground-truth heatmaps are generated as 2D normal distributions with means equal to the ground-truth locations of each keypoint, and a standard deviation σ_k. The standard deviation used in the current dataset is 2 pixels because this value has shown great performance across different ranges, but it can be tuned to modify the feature maps. Examples of heatmaps generated with different values of the standard deviation, derived from the metadata relative to the previously displayed satellite image, are shown in Figure 3.4.

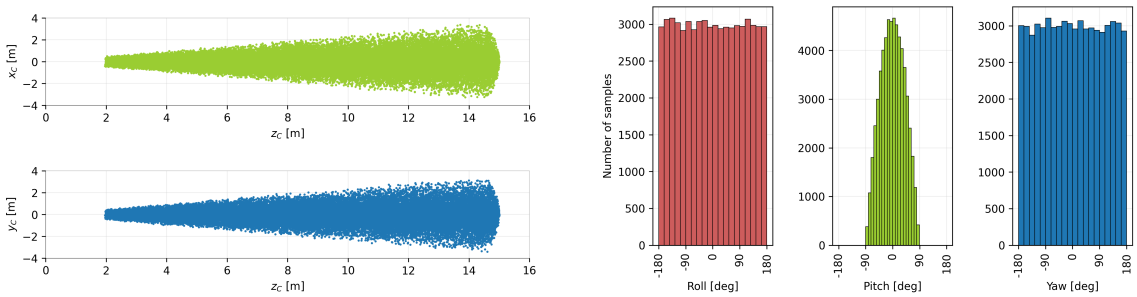


(a) $\sigma_k = 2$ pixels. (b) $\sigma_k = 6$ pixels. (c) $\sigma_k = 10$ pixels.

Figure 3.4: An example of keypoint heatmaps generated with different values of standard deviation. The superimposed satellite images is displayed just to give a visual reference for the keypoints locations, but are obviously not part of the ground-truths used during training.

3.1.1. Building the dataset

The satellite images are synthesized in Unity starting from pose data that is distributed as displayed in Figure 3.5: the distance from the target is considered to be normally distributed on a range between 2 and 15 meters, and a random divergence of the camera is considered to account for an imperfect pointing of the target. The parameters of the simulated camera used in Unity are provided in Table 3.1.



(a) Position labels distribution.

(b) Orientation labels distribution.

Figure 3.5: Position (a) and orientation (b) labels distributions in the dataset. The position label is represented in the camera frame C, whose z-axis is along the camera boresight, and the xy-axes form the image plane. The relative orientation distribution is parametrized as Euler angles.

Table 3.1: MFSPED camera parameters.

Parameter	Value
Resolution ($N_u \times N_v$)	1024×1024 px
Focal length ($f_x = f_y$)	39.47 mm
Pixel pitch ($\rho_u = \rho_v$)	5.86 μm/px
Horizontal FoV	35.0°
Vertical FoV	35.0°

The following Figure 3.6 depicts different levels of accuracy of the software module that simulates the appearance of objects in Unity. On the left, the simplest simulation considering only the different colors of the materials is shown. In the center, a simulation of both the colors and shading of the object based on the direction of light is presented. On the right, a more comprehensive simulation is displayed, which includes reflection effects

given by the reflective capacity of the materials that the object is made of. Even though the study presented in this dissertation focuses on the analysis of the performance of a MTL architecture of a neural network, it is important to notice that adding these basic layers of realism is really important in order to train the model on a dataset that best represents real pictures taken in space. The problem of training on synthetic images that are inherently different from real images is known as *domain gap*. In later development phases of AIKO-NET, *domain gap* will have to be strongly addressed in order to deliver a product that can be trained on synthetic images and work on real, spaceborne images without losing significant performance.

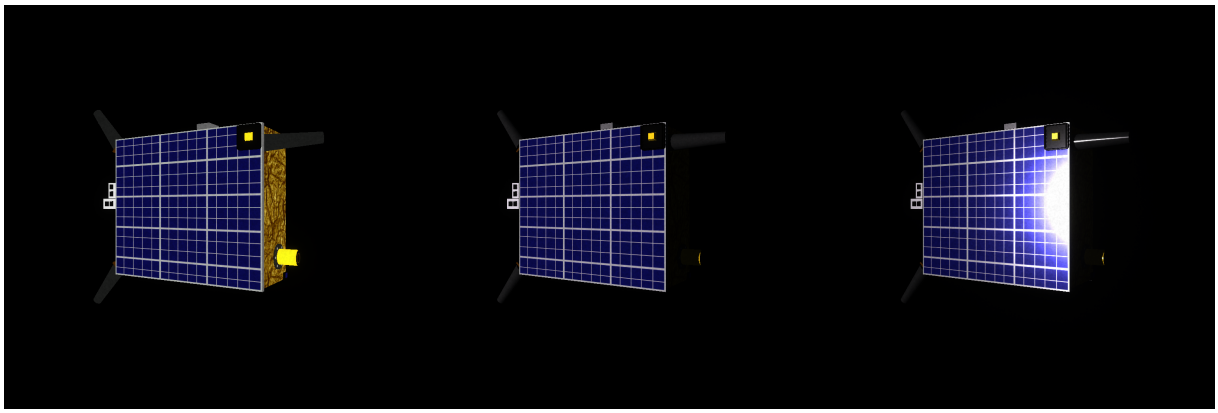


Figure 3.6: Synthetic images details

We will call *synthetic dataset* the Unity output, and *final dataset* the one that will be used for training, validation, and testing. In the *synthetic dataset*, the images of the satellite and of the Earth are generated separately and not in single scenes, thus it consists of satellite images with black background (SAT dataset) and Earth background images (BG dataset). Building the *final dataset* involves an editing step where satellite and background images are randomly chosen and merged to produce the final images for the model to be trained on. In the *synthetic dataset*, the position of the Sun is random and normally distributed such that the neural network can learn to generalize the prediction in any light condition. The fact that the satellite and background images are generated separately inherently implies that the *final dataset* will be made of final images for which the lighting is generally not consistent between the background and the foreground. In this way, the neural network can be trained with images including worst-case scenarios and enhance its robustness with respect to lighting conditions.

As shown in Figure 3.9, which describes the whole processing pipeline, the *synthetic dataset* needs to go through a "pre-processing" phase to make the "dataset generation" process possible. The dataset generation output will be the *final dataset*.

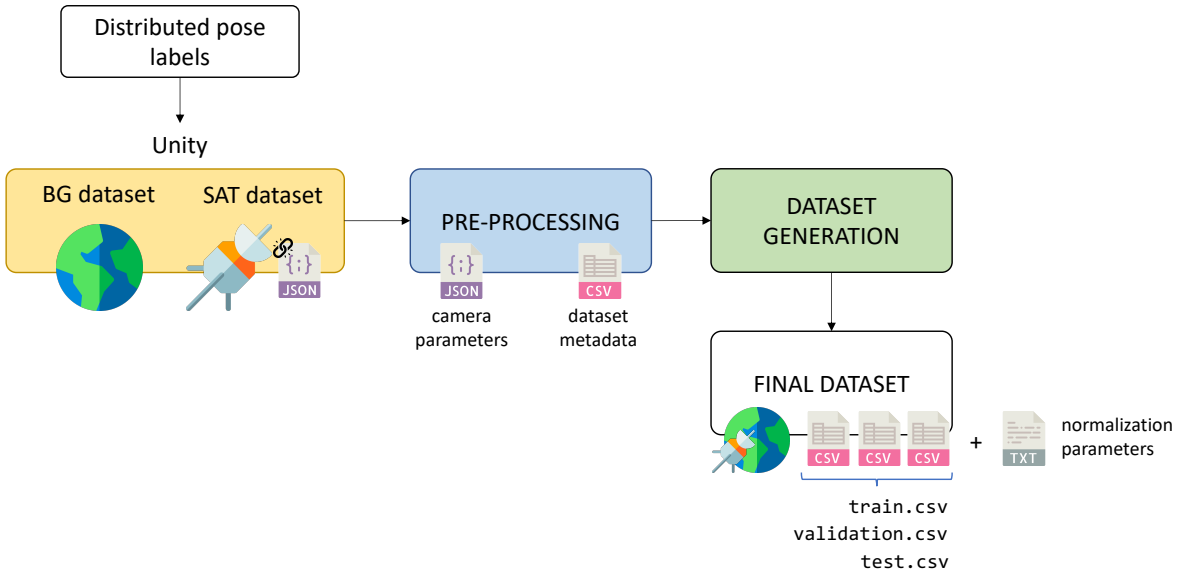


Figure 3.7: Full dataset pipeline.

The *synthetic dataset* comes with a `JSON` file containing the following metadata for each of the generated images of the satellite:

1. the intrinsic matrix $\mathbf{K}$ of the camera used in the simulated scene in Unity;
2. the ground-truths (GTs) of the relative pose: $\mathbf{t}_{C/B}$ and $\mathbf{q}_{C/B}$;
3. the normalized keypoints coordinates $\{x_k, y_k\}_{k=1,\dots,18}$;
4. the rotation quaternion from the Hill frame of the camera satellite to the camera frame $\mathbf{q}_{HC}$;
5. the paths to the relative feature maps.

The rotation quaternion $\mathbf{q}_{HC}$ is collected only if the pose labels used for the generation of the *synthetic dataset* derive from a trajectory simulation discussed in Section 4.1, and sets up the whole pose prediction pipeline to be integrated with a navigation filter.

This data is extracted during the pre-processing phase, which is extremely important not only to extract all the labels linked to each image but also to start an underlying tracking of the original synthetic images that will be used throughout the whole training, validation, and testing phases.

3.1.2. Pre-processing pipeline

The pre-processing of the *synthetic dataset* starts from collecting the metadata from the `JSON` files output by the Unity dataset generation step, and is displayed in Figure 3.8. This data is extracted for each image, and is collected in a `CSV` that will contain:

- the relative pose ground-truths;
- the normalized coordinates localizing the 2D positions of the keypoints;
- the quaternion representing the rotation from the Hill to the Camera frame;
- the normalized coordinates defining the Bounding Box;
- the image file name.

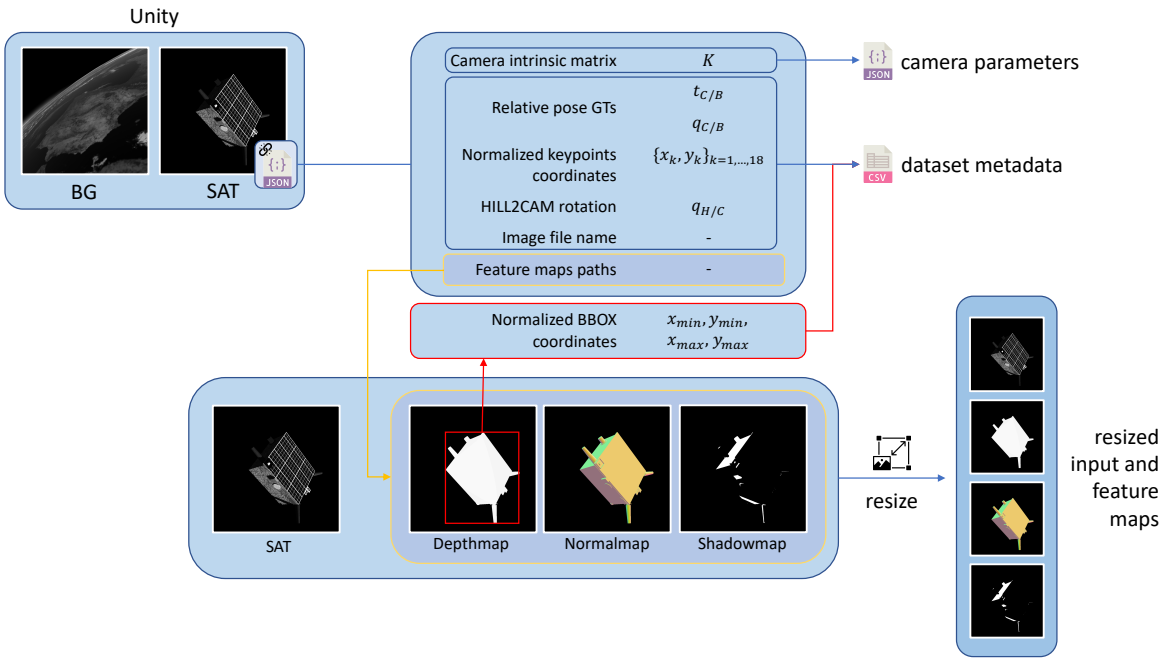


Figure 3.8: Synthetic dataset preprocessing pipeline.

The BB coordinates are retrieved by taking the minimum and maximum x and y coordinates of the pixels turned on in the depthmap. Both the BB and the keypoints coordinates are normalized with respect to the original size of the image, such that they can be used for any input size. The quaternion q_{HC} is stored to be accessible by the Kalman filter that will be introduced in Section 2.5. Furthermore, another `JSON` file containing exclusively the intrinsic camera matrix is generated, to be used by the prediction heads that need them.

3.1.3. Dataset generation

Once the data is pre-processed, the satellite and background images can be merged to obtain the *final dataset*. The number of images to be in the final dataset can be flexibly decided once the pre-processing is complete because all the necessary metadata is already stored and the images from the *synthetic dataset* are resized based on the selected input size for the neural network. Together with the number of images, also the percentages of the training, validation, and test splits can be chosen.

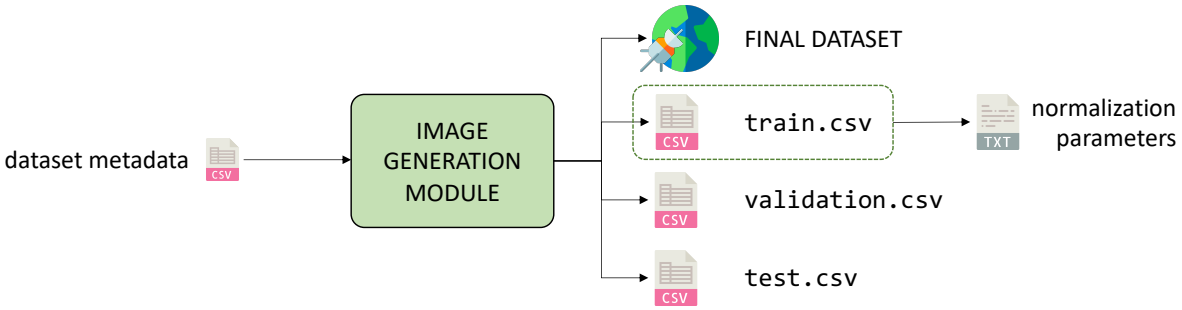


Figure 3.9: Dataset generation pipeline.

While a complete overview of this pipeline is depicted in Figure 3.9, the generation process for a single sample is sketched in Figure 3.10: this operation is repeated for each of the randomly selected images from the resized *synthetic dataset*. The entire process is built making sure that:

- the same satellite image is never repeated in the same split;
- the same satellite image is never present in more than one split;
- if the same background image has to be used multiple times, it is flipped and/or rotated such that no final image will have the same background;

This, of course, limits the maximum size of the final dataset to the number of synthetic satellite images produced with Unity. Obviously, these images cannot undergo any geometric transformation, because they have to stay consistent with the associated labels. The final images are exported as grayscale images.

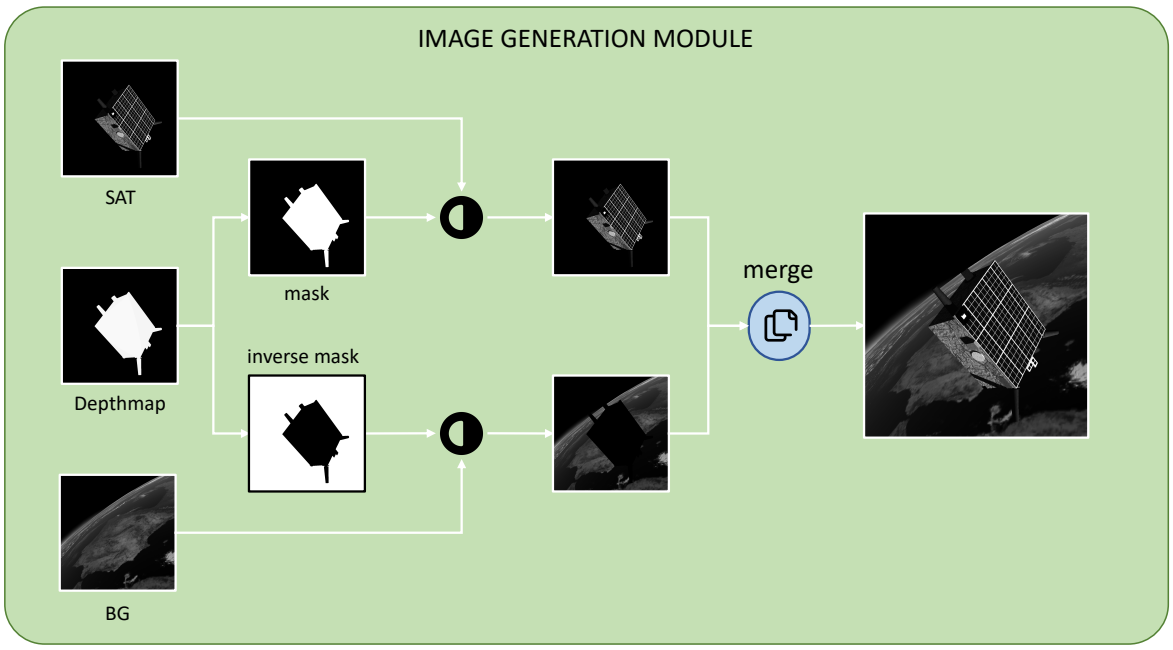


Figure 3.10: Image generation module.

Each of the generated images that are part of the *final dataset* is associated with a row in one of three `CSV` files (one for each split) containing the pose labels and all the available metadata, as well as the original image name from the *synthetic dataset* to be able to analyze the network performances in the best way possible, fitting the data exploration approach. The `CSV` files contain the names and metadata of the exported *final dataset*, and are built to be used by data loaders during the training, validation, and test phase respectively. At the end of the dataset generation process, the images in the training set are used to compute the normalization parameters to be used by AIKO-NET on the input images.

Twelve example images from the dataset are shown in Figure 3.11.

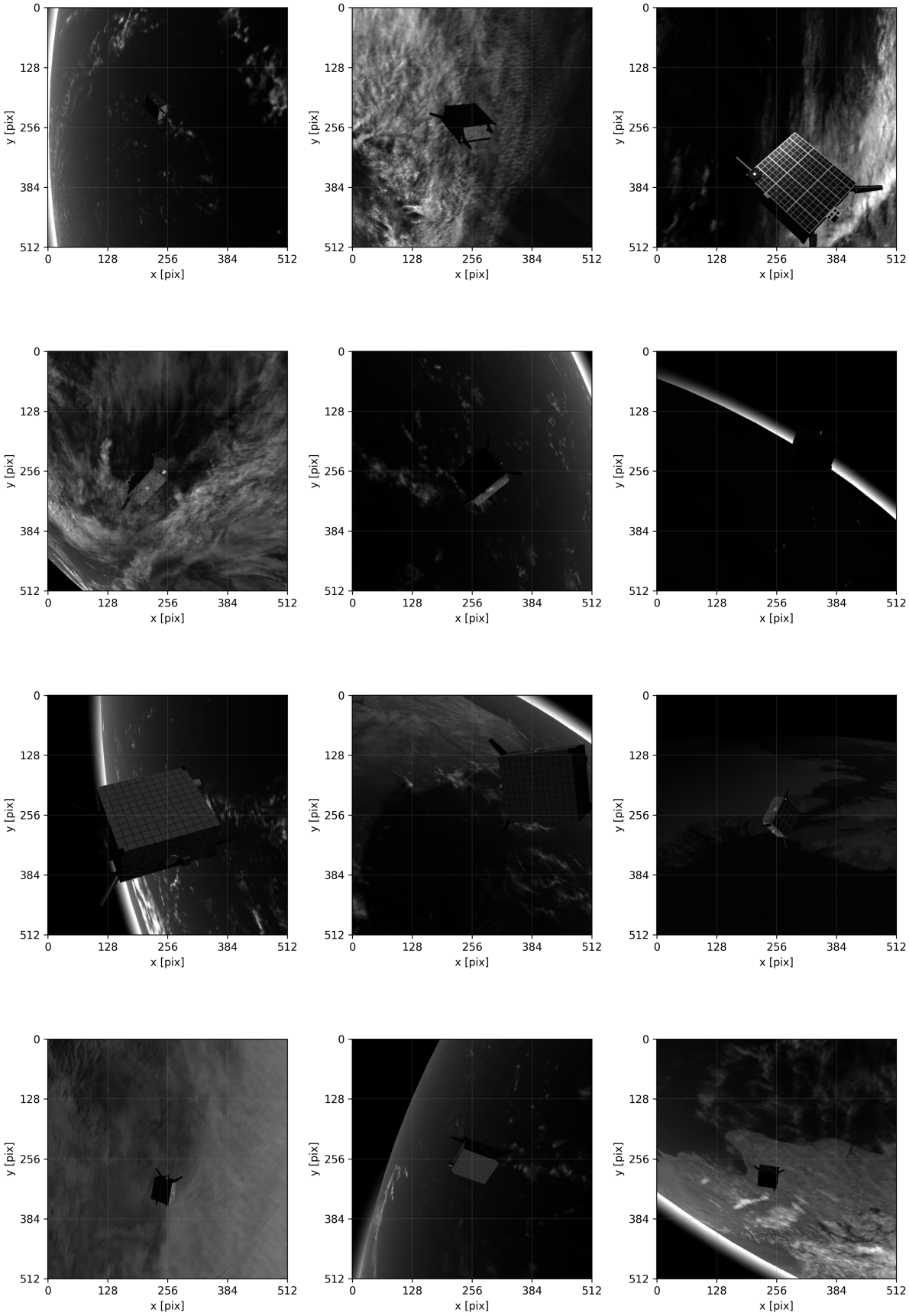


Figure 3.11: Example images with background from the generated dataset.

3.2. Architecture

The AIKO-NET architecture is an expansion of SPNv2 [39]. The "Shared Feature Encoder" is essentially an EfficientDet [56], which is made of a backbone and a neck that are EfficientNet [55] and the BiFPN, respectively. All these blocks on which the CNN is built are detailed in Section 2.2.

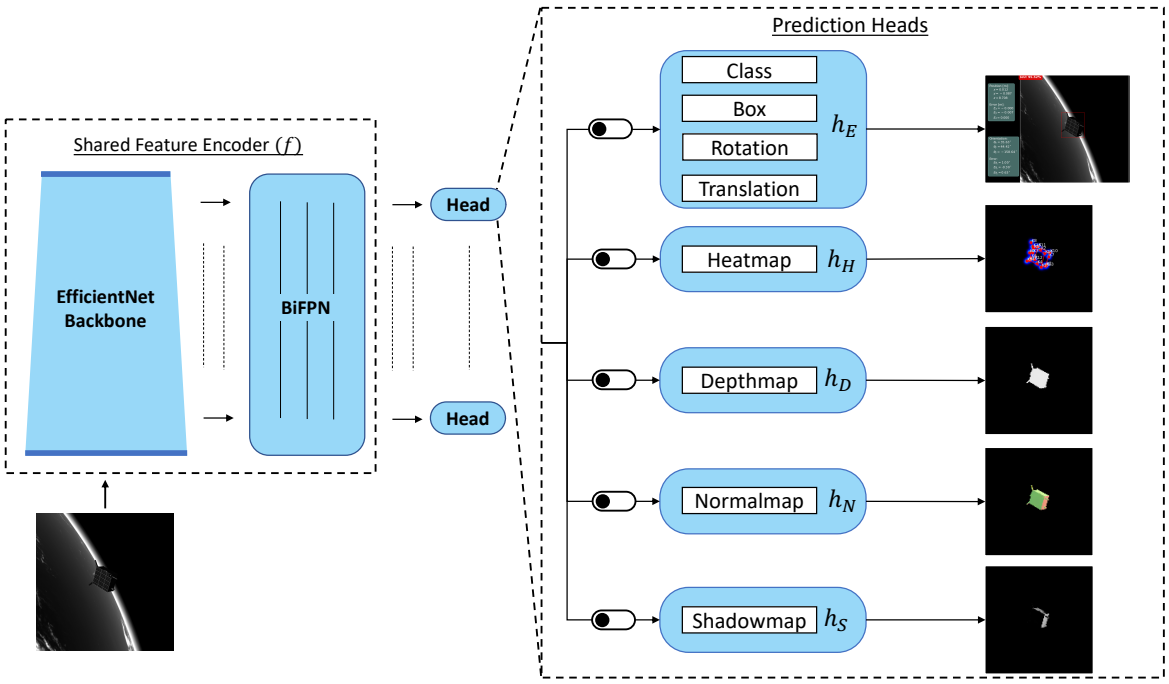


Figure 3.12: The AIKO-NET architecture.

The key difference with respect to SPNv2 here is the addition of several prediction heads that, as depicted in Figure 3.12, can be switched on or off with ease. This modular feature makes such that AIKO-NET is a valid test-bed for experimenting a MTL approach with different combinations. The following prediction heads are currently implemented:

1. EfficientPose [17] head h_{E}, that is responsible for a *direct pose estimation*;
2. Heatmap head h_{H}, responsible for an *indirect pose estimation*;
3. Depthmap head h_{D};
4. Normalmap head h_{N};
5. Shadowmap head h_{S}.

Obviously, one between the EfficientPose and the Heatmap heads has to be on in order to enable the neural network to output a pose prediction. The segmentation head used in the

original SPNv2 architecture is replaced by h_{D}: for the motivations discussed in Section 3.1, the two heads would provide a really similar output. Thus, only h_{D} is implemented because a depthmap is slightly more informational with respect to a segmentation map. Regarding the architecture of all the subnets, it resembles the one of EfficientPose [17], thus the convolution blocks consist of depthwise separable convolution [22] layers followed by group normalization and SiLU activation.
While the outputs from h_{D}, h_{N}, and h_S are not directly related to the main task of AIKO-NET, h_E performs a *direct* pose estimation as explained in 2.2.2, and h_H is responsible for an *indirect* pose estimation by outputting a prediction of the keypoints locations in the form of a heatmap and by letting them be processed by means of the EPnP algorithm introduced in Section 2.3.1.

3.2.1. Training losses

The EfficientPose head loss can be seen as the sum of a:

- *Focal* loss [33] for the classification task;
- *Complete Intersection over Union* (C-IoU) loss [61] for the object localization task;
- *SPEED* loss (introduced in Section 1.2.2) for the pose estimation task.

Since all the other heads output an image, they are associated with a *pixel-wise Mean Squared Error* (MSE) loss.
It is important to notice that the total loss $\mathcal{L}_{tot}$ that is backpropagated is a weighted sum of all the losses. Let $\mathcal{L}_{\mathrm{E}}$, $\mathcal{L}_{\mathrm{H}}$, $\mathcal{L}_{\mathrm{D}}$, $\mathcal{L}_{\mathrm{N}}$, $\mathcal{L}_{\mathrm{S}}$ be the losses of the single prediction heads identified by the subscripts. Then, the total loss is

$$\mathcal{L}_{tot} = w_{\mathrm{E}}\mathcal{L}_{\mathrm{E}} + w_{\mathrm{H}}\mathcal{L}_{\mathrm{H}} + w_{\mathrm{D}}\mathcal{L}_{\mathrm{D}} + w_{\mathrm{N}}\mathcal{L}_{\mathrm{N}} + w_{\mathrm{S}}\mathcal{L}_{\mathrm{S}} \tag{3.1}$$

and the EfficientPose head loss can be written as the weighted sum of the classification, localization, and pose losses as follows:

$$\mathcal{L}_{\mathrm{E}} = w_{\mathrm{E}}^{cls}\mathcal{L}_{\mathrm{E}}^{cls} + w_{\mathrm{E}}^{bbox}\mathcal{L}_{\mathrm{E}}^{bbox} + w_{\mathrm{E}}^{pose}\mathcal{L}_{\mathrm{E}}^{pose} \tag{3.2}$$

All these weights can be individually tuned to analyze the performance of the network in different configurations.

4 | Relative Pose Estimation Pipeline

In this chapter, we will introduce the full pipeline implemented to estimate the relative pose of an uncooperative target, which is illustrated in the image below.

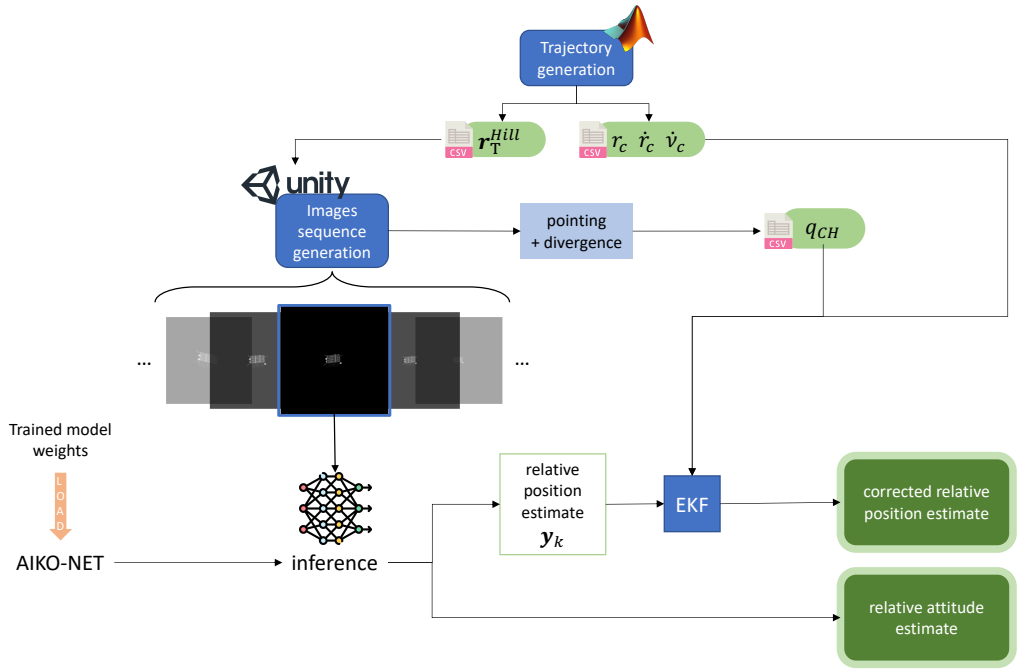


Figure 4.1: Full relative pose estimation pipeline.

The pipeline consists of four main steps:

1. Trajectory generation by means of relative orbital dynamics in MATLAB[1]: this step outputs data that will be used by Unity[2] for step 2 and by the EKF in step 4.

[1]MATLAB version 2021b was used for all simulations and data processing in this work - https://it.mathworks.com/products/matlab.html

[2]The Virtual Simulation department of AIKO, who I had to interface and collaborate with, was responsible for developing the Unity processes.

2. Processing of the trajectory data in Unity to produce a sequence of synthetic images representing the target in the camera frame. In this step, also the pointing plus a random divergence is set up. The outputs are the image sequence and a `JSON` file containing all the metadata described in Section3.1.2.
3. Pose estimation through AIKO-NET. The CNN takes the synthetic images as input and outputs two estimates of the relative pose between the spacecraft and the target: one directly from the EfficientPose head (*direct estimation*) and one from the Heatmap head's output processed by the EPnP algorithm (*indirect estimation*).
4. Filtering of the estimated position using an EKF.

In the following sections, we will describe each step of the pipeline in detail, including the implementation, challenges, and results.

4.1. Trajectory Generation

The trajectory generation process was built with the scope of setting up a framework to produce pseudo-random trajectories to be used as simulation scenarios in the full pose estimation pipeline. Before presenting the logic behind the orbit generation, it is important to note that only the relative translational dynamics is taken into account. This is due to the limited time during which the work presented in this dissertation was developed, but the relative rotational dynamics could be ideally implemented and embedded in this phase with the aim of filtering the whole relative pose in step 4 of the pipeline presented in the previous section.

For the sake of readability, from now on Tango will be the target spacecraft and AIKO-CAM will be the name of the chief satellite that we can virtually control, the state of which will be considered as exactly known. The subscript "t" will be referred to the target, while the subscript "c" will be for the chief. To generate realistic relative movement between AIKO-CAM and Tango, it is crucial to take into account the orbital parameters characterizing Tango's orbit. Therefore, we obtain the initial conditions for Tango from Two-Line Elements (TLEs) and then add a pseudo-random displacement to the state to obtain an initial state for AIKO-CAM. The TLEs of Tango have been retrieved from the *CelesTrak* [12] database and fed into the *NASA Horizons System* [13] tool, which returned the state of the satellite on a selected date. Thus, Tango state in the Earth Centered Inertial (ECI) frame is available.

$$\mathbf{x}_{t_0}^{ECI} = \begin{bmatrix} \mathbf{r}_{t_0}^{ECI\top}, & \dot{\mathbf{r}}_{t_0}^{ECI\top} \end{bmatrix}^\top \tag{4.1}$$

Then, a bounded random initial state displacement

$$\tilde{\mathbf{x}}^{ECI}_{t_0\rightarrow c_0} = \begin{bmatrix} \tilde{\mathbf{r}}^{ECI}_{t_0\rightarrow c_0}{}^{\top}, & \dot{\tilde{\mathbf{r}}}^{ECI}_{t_0\rightarrow c_0}{}^{\top} \end{bmatrix}^{\top} \tag{4.2}$$

is added to Tango's initial state $\mathbf{x}^{ECI}_{t_0}$ in order to retrieve an initial state for AIKO-CAM $\mathbf{x}^{ECI}_{c_0} = \begin{bmatrix} \mathbf{r}^{ECI\top}_{c_0}, & \dot{\mathbf{r}}^{ECI\top}_{c_0} \end{bmatrix}^{\top}$:

$$\mathbf{r}^{ECI}_{c_0} = \mathbf{r}^{ECI}_{t_0} + \tilde{\mathbf{r}}^{ECI}_{t_0\rightarrow c_0} \qquad\qquad \dot{\mathbf{r}}^{ECI}_{c_0} = \dot{\mathbf{r}}^{ECI}_{t_0} + \dot{\tilde{\mathbf{r}}}^{ECI}_{t_0\rightarrow c_0} \tag{4.3}$$

The randomness of these random displacements can be tuned to obtain different relative initial conditions resulting in different relative orbits. In order to implement the relative dynamics introduced in Section 2.4, we need to convert the initial relative state of the target in the chaser's Hill reference frame H, as done in Equation 4.4. To match the notation used in the description of the dynamics, $\boldsymbol{\rho}$ will be used to represent the target position in the chief's Hill frame.

$$\begin{aligned} \boldsymbol{\rho}_0 &= \mathbf{R}^{ECI\rightarrow H}\left(-\tilde{\mathbf{r}}^{ECI}_{t_0\rightarrow c_0}\right) \\ \dot{\boldsymbol{\rho}}_0 &= \mathbf{R}^{ECI\rightarrow H}\left(-\dot{\tilde{\mathbf{r}}}^{ECI}_{t_0\rightarrow c_0}\right) \end{aligned} \tag{4.4}$$

The matrix $\mathbf{R}^{ECI\rightarrow H}$ is the DCM representing the rotation between the two frames, and is a function of the AIKO-CAM orbital parameters as displayed in Equation 4.5, which are now known since its initial state in the ECI frame is defined.

$$\mathbf{R}^{ECI\rightarrow H} = \begin{bmatrix} \cos(\omega+\nu) & -\sin(\omega+\nu) & 0 \\ \sin(\omega+\nu) & \cos(\omega+\nu) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & -\sin i \\ 0 & \sin i & \cos i \end{bmatrix} \begin{bmatrix} \cos\Omega & -\sin\Omega & 0 \\ \sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

The rotation depends on the true latitude $\theta = \omega + \nu$ that is the sum of the argument of periapsis and the true anomaly, the chief's orbit inclination i, and the right ascension of the ascending node Ω.

By defining the target state $\mathbf{x}$ in H as

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\rho}^{\top} & \dot{\boldsymbol{\rho}}^{\top} \end{bmatrix}^{\top} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^{\top}, \tag{4.6}$$

we can write down the state space form of the model expressing the relative translational dynamics with the EOM in Equation 2.31 from Section 2.4, where f is the state transition function of the model.

$$\dot{\mathbf{x}} = f(\mathbf{x}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 2\dot{\nu}\left(\dot{y} - y\frac{\dot{r}_c}{r_c}\right) + x\dot{\nu}^2 + \frac{\mu}{r_c} - \frac{\mu(r_c+x)}{((r_c+x)^2+y^2+z^2)^{3/2}} \\ -2\dot{\nu}\left(\dot{x} - x\frac{\dot{r}_c}{r_c}\right) + y\dot{\nu}^2 - \frac{\mu y}{((r_c+x)^2+y^2+z^2)^{3/2}} \\ -\frac{\mu z}{((r_c+x)^2+y^2+z^2)^{3/2}} \end{bmatrix} \tag{4.7}$$

Note that this formulation needs to be completed with the orbital dynamics of the chaser in terms of true anomaly ν and orbital radius r_c, through the Equations 2.27 and 2.25 from Section 2.4 that are here reported for completeness.

$$\begin{aligned} \ddot{r}_c &= r_c\dot{\nu}^2 - \frac{\mu}{r_c^2} \\ \ddot{\nu} &= -2\frac{\dot{r}_c}{r_c}\dot{\nu} \end{aligned} \tag{4.8}$$

The images below show some examples of obtained trajectories for a full orbital period of the chief satellite. In the title of the plots, the operative range and the relative operational time are reported, highlighting the flexibility of the trajectory generation process.

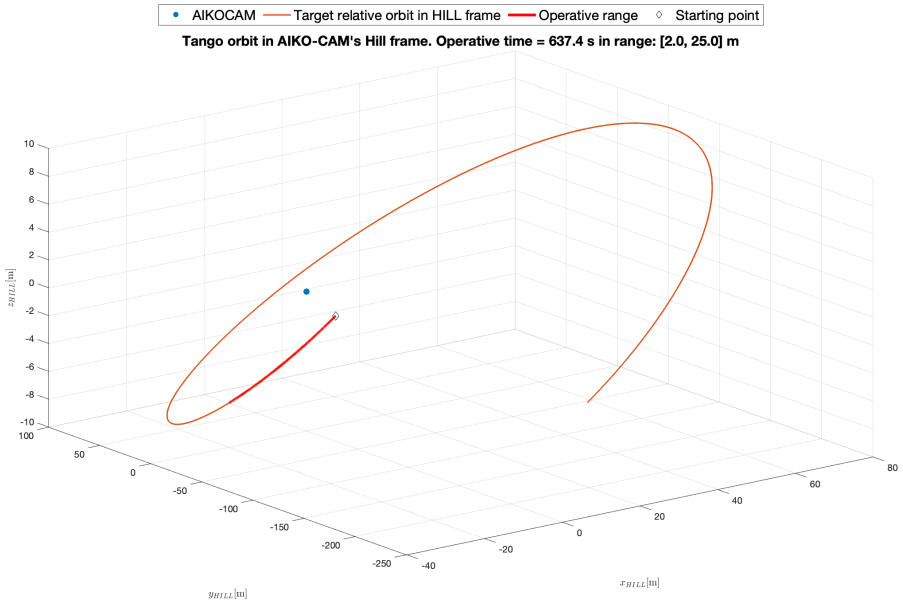


Figure 4.2: Generated trajectory: example 1.

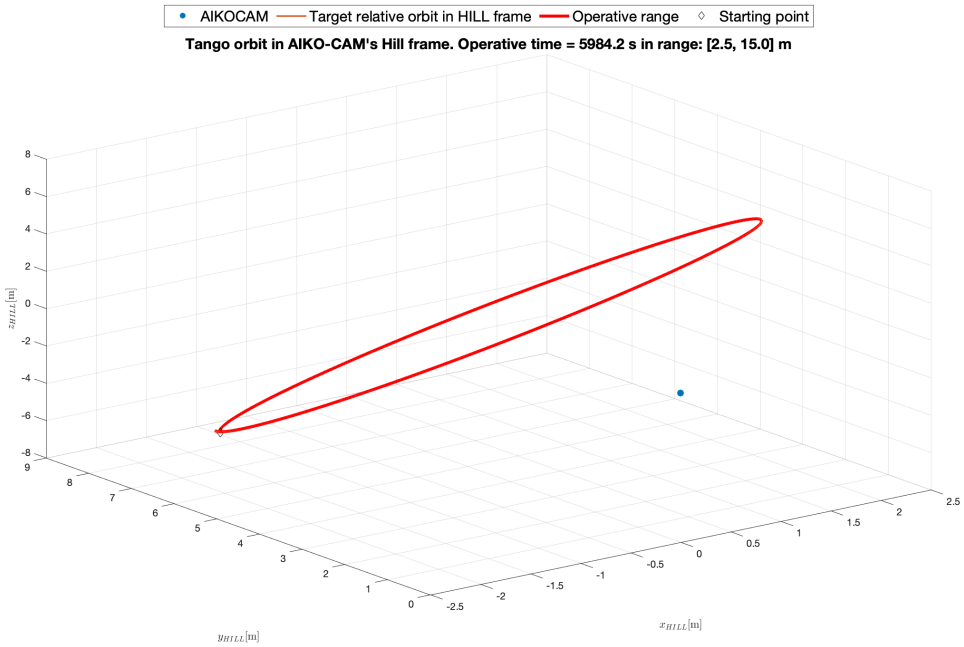


Figure 4.3: Generated trajectory: example 2.

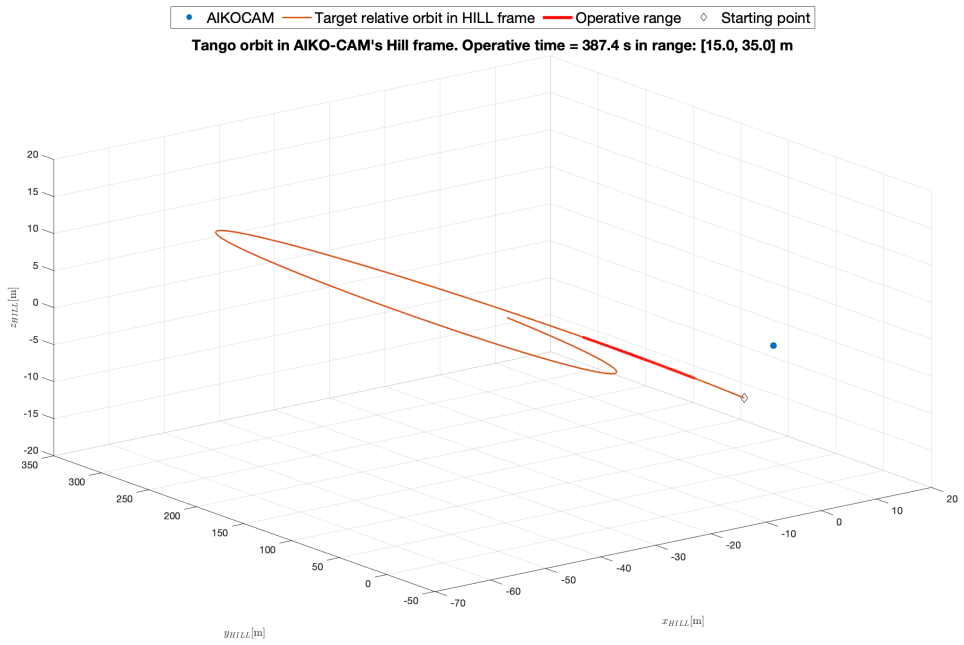


Figure 4.4: Generated trajectory: example 3.

4.2. Images sequence generation

As this process was developed by the Virtual Simulation team within AIKO, we will not go into detail. Nevertheless, it is useful to briefly recap the inputs and outputs of this block and to describe the relation between the reference frames characterizing them. Unity receives as input the trajectory in the Hill reference frame H of AIKO-CAM, and outputs:

- a sequence of images at 2 FPS;
- the associated relative pose ground truths in the Camera reference frame C;
- the quaternion q_{HC} for each frame, expressing the rotation from H to C, considering an imperfect target pointing calibrated with respect to the camera parameters.

Figure 4.5 gives a complete overview of the frames involved in the problem.

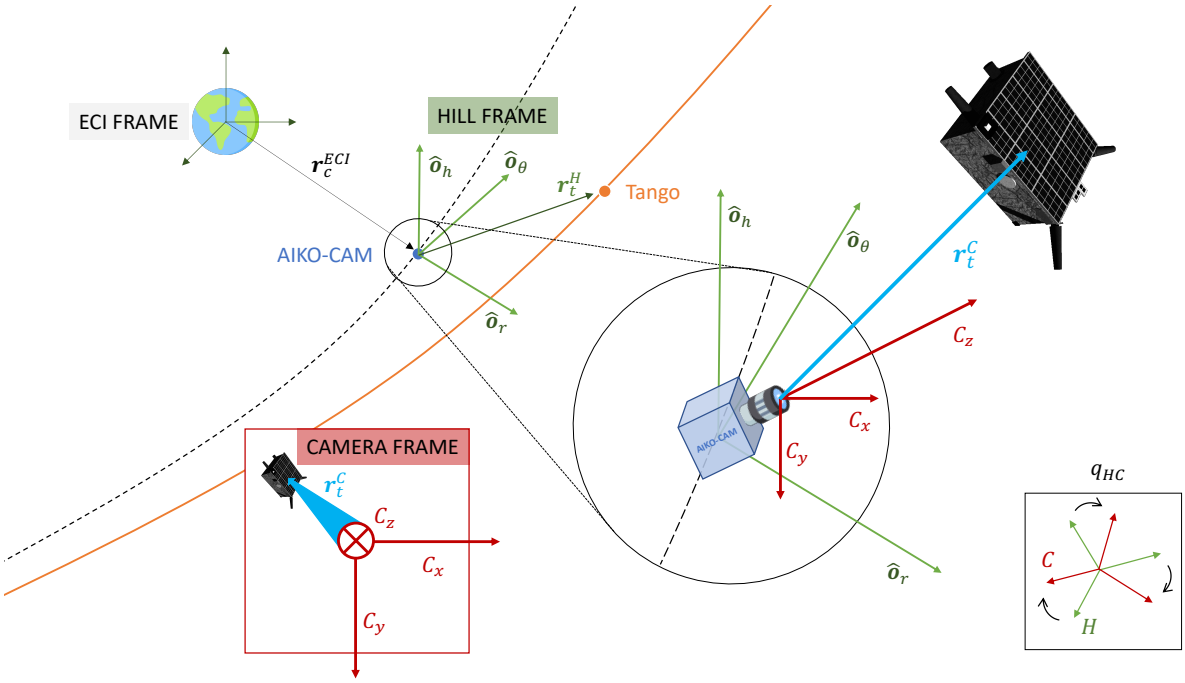


Figure 4.5: Overview on reference frames.

4.3. Extended Kalman Filter implementation

Let us recall the EKF algorithm in Equation 4.9.

$$\begin{aligned}
\hat{\mathbf{x}}_{k+1}^{-} &= f\left(\hat{\mathbf{x}}_{k}^{+}, \mathbf{u}_k, 0\right) \\
\mathbf{P}_{k+1}^{-} &= \mathbf{F}\mathbf{P}_{k}^{+}\mathbf{F}^{\top} + \mathbf{Q}_k \\
\mathbf{K}_{k+1} &\coloneqq \mathbf{P}_{k+1}^{-}\mathbf{H}^{\top}\left(\mathbf{H}\mathbf{P}_{k+1}^{-}\mathbf{H}^{\top} + \mathbf{R}_{k+1}\right)^{-1} \\
\hat{\mathbf{x}}_{k+1}^{+} &= \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{K}_{k+1}\left(\mathbf{y}_{k+1} - h\left(\hat{\mathbf{x}}_{k+1}^{-}, 0\right)\right) \\
\mathbf{P}_{k+1}^{+} &= \left(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}\right)\mathbf{P}_{k+1}^{-}
\end{aligned} \tag{4.9}$$

We already defined the state transition function f in Equation 4.7, and it is easy to retrieve the analytical form of its Jacobian

$$\mathbf{F} = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial y} & \frac{\partial \ddot{x}}{\partial z} & 0 & \frac{\partial \ddot{x}}{\partial \dot{y}} & 0 \\
\frac{\partial \ddot{y}}{\partial x} & \frac{\partial \ddot{y}}{\partial y} & \frac{\partial \ddot{y}}{\partial z} & \frac{\partial \ddot{y}}{\partial \dot{x}} & 0 & 0 \\
\frac{\partial \ddot{z}}{\partial x} & \frac{\partial \ddot{z}}{\partial y} & \frac{\partial \ddot{z}}{\partial z} & 0 & 0 & 0
\end{bmatrix} \tag{4.10}$$

as well as the Jacobian of the measurement function [43].

$$\mathbf{H} = \frac{\partial h}{\partial \mathbf{x}} = \begin{bmatrix}
 & & & 0 & 0 & 0 \\
\mathbf{R}_{CH}(\mathbf{q}_{CH}) & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0
\end{bmatrix} \tag{4.11}$$

The noise measurement matrix $\mathbf{R}$ and the process matrix $\mathbf{Q}$ have been tuned to ensure a good performance of the filter. Regarding the inputs to the EKF block depicted in Figure 4.1, it is important to notice that all the parameters needed for the computation of the Jacobians, namely r_c, $\dot{r}_c$, $\dot{\nu}_c$ and $\mathbf{q}_{CH}$, are considered as exactly known as they derive from the state of AIKO-CAM, which is virtually under our control. This is an assumption that is not realistic since such parameters should derive from a number of measurements made on and off-board, thus they would be affected by some degree of error and uncertainty. However, this work aims just at implementing a basic form of Kalman filtering to demonstrate the validity of the process and of the whole pipeline.

5 | Results

5.1. Error metrics

In order to evaluate the performance of AIKO-NET, it is necessary to define and use appropriate error metrics. These metrics are critical for understanding the accuracy of the network and determining its effectiveness in real-world scenarios.
Taking inspiration from the approach presented in [42], both the mean and the median of each error will be reported along with the standard deviation and the interquartile range, respectively. For the translation, we will call $\mathbf{E}_t$ the vector containing the absolute error on the (x, y, z) components, and E_t its norm, as expressed in Equation 5.1 where $\hat{\mathbf{t}}_{C/B}$ represents a prediction of the translation vector and $\mathbf{t}_{C/B}$ is the corresponding GT.

$$E_t = \|\mathbf{E}_t\| = \| \left(|\hat{\mathbf{t}}_{C/B} - \mathbf{t}_{C/B}|\right) \| \tag{5.1}$$

Then, e_t is the normalized translation error, and is defined as the absolute translation error divided by the GT distance:

$$e_t = \frac{E_t}{\|\mathbf{t}_{C/B}\|} \tag{5.2}$$

Regarding the error relative to the rotation, it will be measured in two different ways: in terms of quaternion error, to represent the overall attitude error with a single scalar value, as

$$E_q = 2 \cdot \arccos |\mathbf{q} \cdot \hat{\mathbf{q}}|, \tag{5.3}$$

where $\hat{\mathbf{q}}$ represents a prediction of the relative orientation and $\mathbf{q}$ is the GT, and in terms of Euler angles as

$$\mathbf{E}_\theta = \left[|\hat{\theta}_x - \theta_x|, \quad |\hat{\theta}_y - \theta_y|, \quad |\hat{\theta}_y - \theta_y|\right], \tag{5.4}$$

where $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)$ are predictions and $(\theta_x, \theta_y, \theta_z)$ are the GTs. The overall pose error is expressed

using the SPEED score introduced in Section 1.2.2:

$$\text{SPEED} = e_t + Eq \tag{5.5}$$

5.2. AIKO-NET Performance

Since AIKO-NET is based on a multi-scale, multi-feature architecture, there are some important assessments to be made before presenting the performances.
The compound scaling parameter used for all the presented models is $\phi = 3$, because this value showed the best trade-off between the accuracy of the net and computational cost in SPNv2 [39]. Furthermore, the chosen input size for the network is 512×512. The generated dataset is made of 40000 images, and the splits used for the training, validation, and testing phases are presented in Table 5.1.

Table 5.1: Generated dataset splits.

	`training`	`validation`	`testing`
Split	70%	20%	10%
N_{images}	28000	8000	4000

Exploiting the modularity of the architecture, the different configurations reported in Table 5.2 have been trained, validated, and tested.

Table 5.2: Prediction heads configurations for 9 versions of the network.

	`V0`	`V1`	`V2`	`V3`	`V4`	`V5`	`V6`	`V7`	`V8`
h_E	✓	✓	✓		✓	✓	✓	✓	✓
h_H	✓	✓		✓	✓	✓	✓	✓	✓
h_D	✓	✓				✓			✓
h_N	✓	✓					✓		✓
h_S	✓	✓						✓	✓
BGs		✓	✓	✓	✓	✓	✓	✓	✓

The `V0` configuration was used for the first tests, and the background images are excluded from the final dataset to assess the performance of AIKO-NET in full configuration and optimal scenario

conditions. The aim of this process was to find suitable configurations that may over-perform others. It is very important to remember that the configurations from `V0` to `V7` were trained using loss functions that equally concur to the total loss. This decision was taken to initially identify how much each prediction head enhances the "benchmark performance" associated with `V0`.
The last version, `V8`, has the same configuration as `V1` but the losses weights are modified based on the previous results.

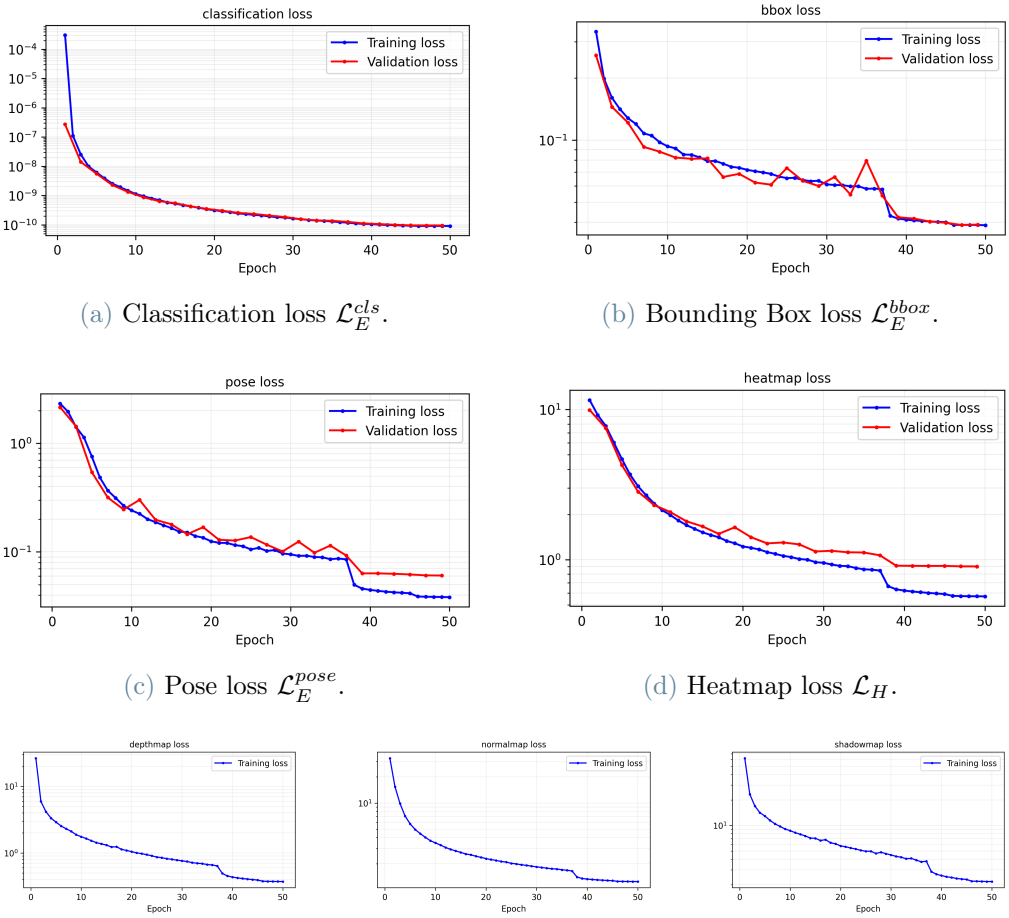
Each configuration is trained for 50 epochs with a batch size of 10 and validated every 2 epochs. A learning rate of 5e−4 is used, which is scaled by a factor of 1e−1 at the 75% and 90% of the training process. For clarity of the exposition, these values are reported in Table 5.3, where f_{val} denotes the validation frequency.

Table 5.3: Training parameters.

Epochs	f_{val}	batch size	LR	LR steps	LR factor
50	2	10	5e−4	75% − 90%	1e−1

5.2.1. Benchmark: AIKO-NET `V0`

In Figure 5.1, the evolution of the losses of `V0` to be used as a benchmark. Tables 5.4 and 5.5 show the mean, standard deviation, median and interquartile range (IQR) relative to the results of testing the final model on the whole testset of 4000 images, from both the *direct* and *indirect* estimations. The training and validation losses for multiple heads are plotted over the epochs, and it can be observed that the validation losses closely follow the training losses. This indicates that the CNN does not underfit or overfit the data. This result is likely due to the narrow domain of the images used in this study, which results in macroscopic similarities between the training, validation, and test data splits. Furthermore, it is also noticeable how the network benefits from the LR steps, and it is clear that the LR and the LR factor used correspond to a good training behavior since the CNN continues to learn even after scaling the LR at the 75% and 90% of the training epochs. Since the images on which AIKO-NET is trained contain just one object, the classification loss is the easiest task and because of this it is useful as a benchmark to check that the CNN training is behaving fine despite the multi-task architecture. For the bounding box task, even though the validation loss is often lower than the training one, the global evolution of the losses on the two splits confirms the validity of the dataset by converging to really close values.



(a) Classification loss $\mathcal{L}_E^{cls}$.

(b) Bounding Box loss $\mathcal{L}_E^{bbox}$.

(c) Pose loss $\mathcal{L}_E^{pose}$.

(d) Heatmap loss $\mathcal{L}_H$.

(e) Depthmap loss $\mathcal{L}_D$.

(f) Normalmap loss $\mathcal{L}_N$.

(g) Shadowmap loss $\mathcal{L}_S$.

Figure 5.1: Evolution of the losses during the training of AIKO-NET `V0`.

Analyzing the losses related to more difficult tasks, such as the pose and the heatmap, the validation loss results in reaching a slightly higher value with respect to the training one. As has been already said, this is probably due to the higher level of complexity of such tasks. Specifically for the heatmap loss, the validation loss looks to stay constant during the last epochs, while the training loss continues to go down: this might be a signal of starting overfitting, so training the heatmap head for more than 50 epochs in these conditions might not be helpful for the network.

The results reported in the tables below show an overall good performance of AIKO-NET `V0`, with a generally better accuracy for the EfficientPose head.

Table 5.4: Global performance of VO - mean values and STDs.

	EfficientPose		Heatmap + EPnP	
	Mean	**STD**	**Mean**	**STD**
$\mathbf{E}_t$ [mm]	$[9.0, 9.0, 50.6]$	$[8.6, 9.1, 66.3]$	$[23.8, 22.7, 188.5]$	$[19.6, 17.6, 137.0]$
E_t [mm]	52.2	67.5	191.3	139.3
e_t	0.007	0.007	0.028	0.027
$\mathbf{E}_\theta$ [deg]	$[1.9, 1.2, 2.0]$	$[2.6, 1.9, 2.5]$	$[2.2, 1.5, 2.6]$	$[5.3, 4.1, 6.0]$
E_q [deg]	2.6	2.3	3.5	7.2
SPEED	0.05172	0.04259	0.08308	0.10281

Table 5.5: Global performance of VO - median values and IQRs.

	EfficientPose		Heatmap + EPnP	
	Median	**IQR**	**Median**	**IQR**
$\mathbf{E}_t$ [mm]	$[6.9, 6.7, 28.6]$	$[8.9, 8.8, 54.3]$	$[20.7, 20.1, 167.3]$	$[17.7, 16.9, 182.1]$
E_t [mm]	30.2	55.7	169.7	183.7
e_t	0.005	0.005	0.021	0.026
$\mathbf{E}_\theta$ [deg]	$[1.3, 0.9, 1.4]$	$[1.7, 1.2, 1.8]$	$[0.9, 0.5, 0.9]$	$[1.6, 0.9, 1.8]$
E_q [deg]	2.2	1.6	1.4	1.9
SPEED	0.04457	0.03042	0.05618	0.04537

These results derive from an error analysis excluding some outliers: inputs are considered as attitude outliers when

$$\text{the error on any of the Euler angles is } > 100^\circ,$$

and position outliers when

$$\text{the error on the normalized range is } > 0.2.$$

The number of the resulting outliers and their kind, associated to each pose prediction method are shown in Table 5.6 as percentages of the total testset.

Table 5.6: Percentages of outlier images in the testset. Values are categorized with respect to the estimation method and estimated state.

	EfficientPose	Heatmap + EPnP
Position outliers	0.0%	1.9%
Attitude outliers	1.8%	2.0%

In Figure 5.2, all the losses are displayed on the same plot to compare their order of magnitude. It is easy to notice that there are some losses that have a more significant impact on the total loss, meaning that the heads they are associated with are forcing the backpropagation to update the network weights to better perform their task. Since these losses are the ones related to the feature maps, they are making it harder for the main task loss to reach a better minimum. The classification loss is left out of this comparison since it ranges between ∼1e−4 and ∼1e−10. We can find the motivation for these different behaviors in the nature of the feature maps: estimating the shadowmap is a really challenging task since the geometrical features of the satellite get completely altered; the normalmap is a RGB image, thus the error is high when compared to other feature maps as expected; the depthmap is easier to estimate with respect to the aforementioned features because, with the used dataset, it can be approximated to a simple segmentation mask.

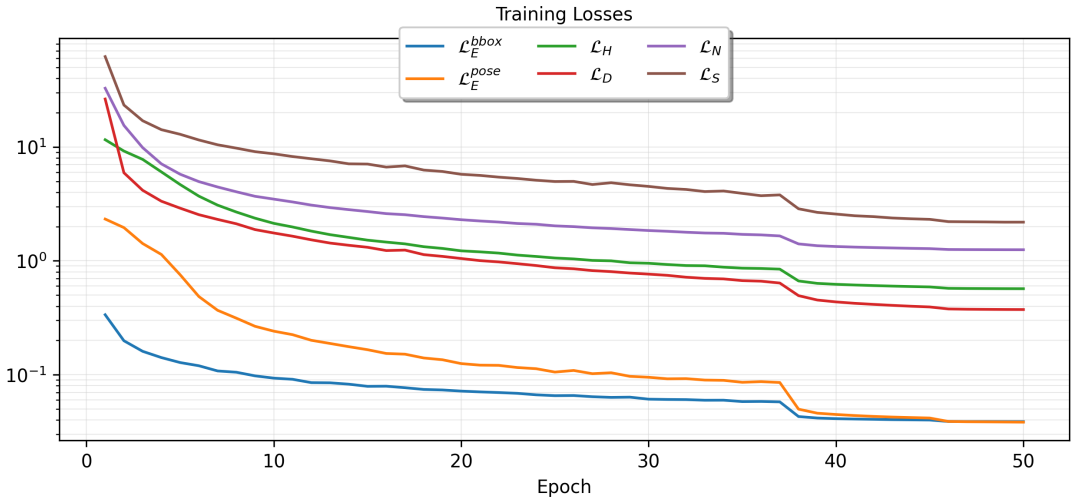
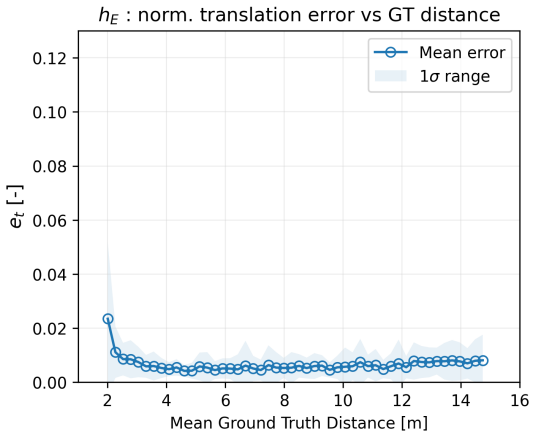
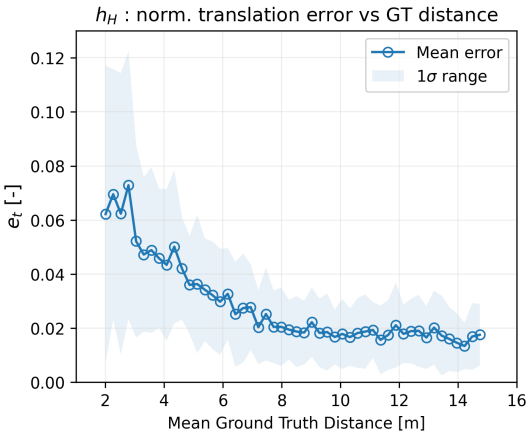


Figure 5.2: Comparison of `VO` training loss orders of magnitude. All the losses apart from the classification loss are reported.

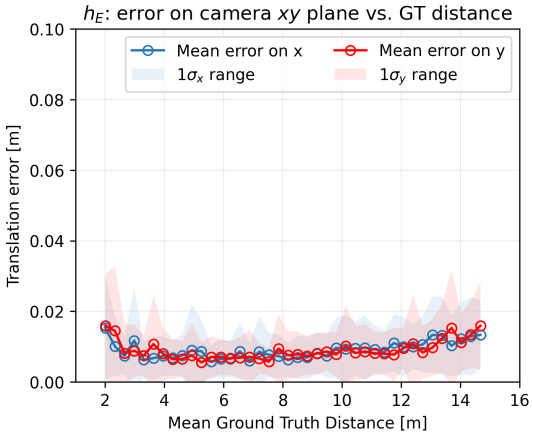
It is also interesting to analyze the accuracy of the model with respect to the GT distance from the target.



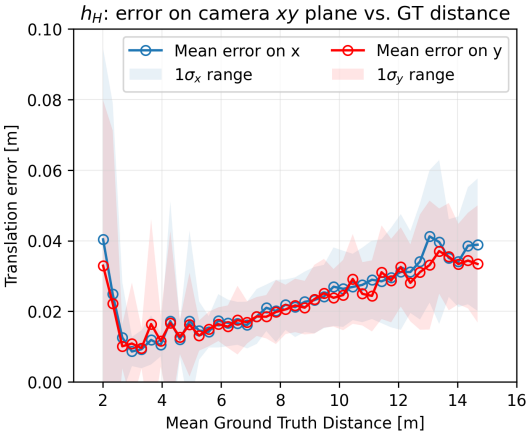
(a) $h_E : e_t$



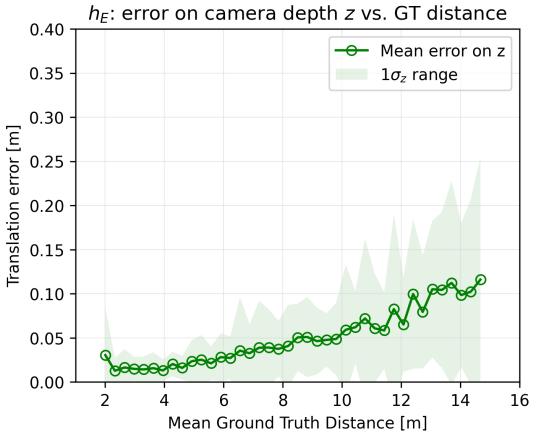
(b) $h_H + EPnP : e_t$



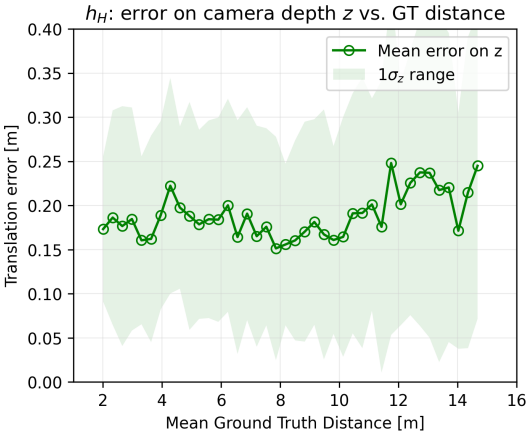
(c) $h_E : E_x, E_y$



(d) $h_H + EPnP : E_x, E_y$



(e) $h_E : E_z$



(f) $h_H + EPnP : E_z$

Figure 5.3: Effect of relative distance on translation errors. Results from direct estimation through the EfficientPose head on the left (a)-(c)-(e), and from indirect estimation through the Heatmap head + EPnP on the right (b)-(d)-(f).

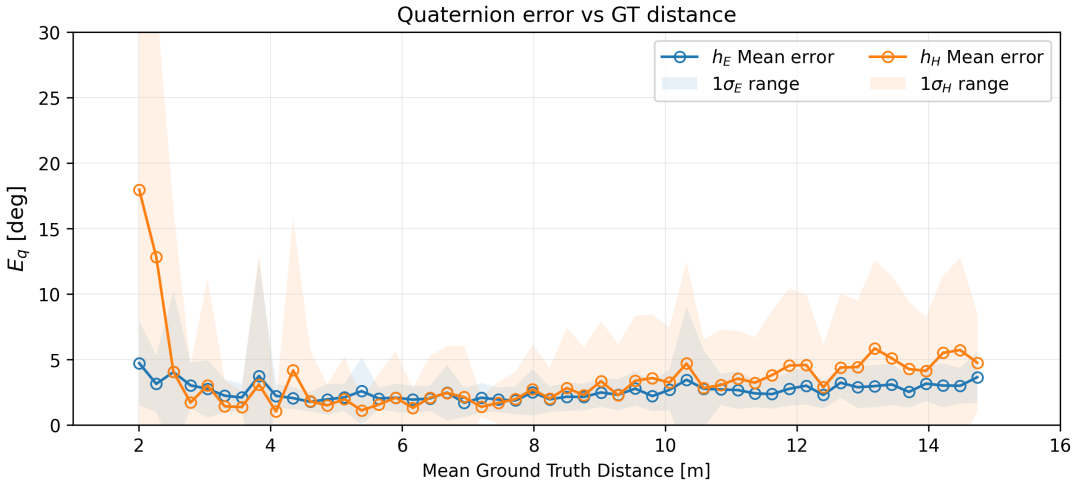


Figure 5.4: Error on the quaternion vs. relative distance

In order to perform this analysis, the test images were grouped by their associated distance GT, and for each group, the corresponding mean performance is plotted against the mean distance. Figure 5.3 shows the results of the analysis for the translation errors, while in Figure 5.4 the quaternion error is taken into consideration.

Starting from the translation errors in Figure 5.3, the results deriving from the direct estimation made by the EfficientPose head are reported on the left column, while the ones deriving from the indirect estimation, i.e. Heatmap head + EPnP, are placed on the right. In order to extract the most information from the data, the plots relative to the same quantities, so on the same row, have the same limits on the y axis. This way, not only the single head's performance can be appreciated, but also an efficient comparison between the two estimations results immediately available.

At a first glance, it is obvious how the EfficientPose head has an overall better performance, as already seen in Tables 5.4 and 5.5. However, both estimation approaches show an increase in translation errors with the distance from the target. This is likely to be due to the fact that the features of the satellite become more and more difficult to extract as the target gets smaller and smaller in the input images that are already low resolution ones. In particular, the depth component z in the camera frame results more difficult to estimate with respect to x and y: first of all, the position components on the camera xy plane are more bounded due to the nature of the problem, and it would be interesting to see how this behavior changes with different FoVs; moreover, the BB detection task surely helps in the x and y components estimation, while has a lower impact on the prediction of the z coordinate. Small distances also result in less accurate position predictions: while the errors for the EfficientPose head remain contained, the EPnP finds it really hard to precisely estimate the position of the target when it is really close to the camera. This is probably due to keypoints that might be off the FoV of the camera, or to the

sigma value chosen for the heatmaps ground-truths generation discussed in Section3.1.

Moving on to quaternion errors, the results from both estimation approaches are plotted together in Figure 5.4. It can once again be noticed how the overall EfficientPose predictions accuracy is more stable, while the indirect estimation presents issues with both smaller and bigger distances. The analysis for the pitch, roll, and yaw angles is not reported since the results obtained are similar for each of the three Euler angles for both estimation approaches.

5.2.2. Configurations performance comparison

As introduced at the beginning of this chapter, different configurations were tested to understand if and how the architecture is capable of solving different tasks in parallel and if such a parallelization can, in some cases, enhance the performance of the principal tasks that are the pose estimation through both the direct and indirect approach.

It is important to recall that all the configurations apart from `V0` are trained, validated, and tested on a dataset made of satellite images with background. Figures 5.5, 5.6 and 5.7 depict the boxplots of the normalized translation error, quaternion error, and SPEED scores respectively, obtained for each configuration tested on the whole testset, and for both the estimation methods.

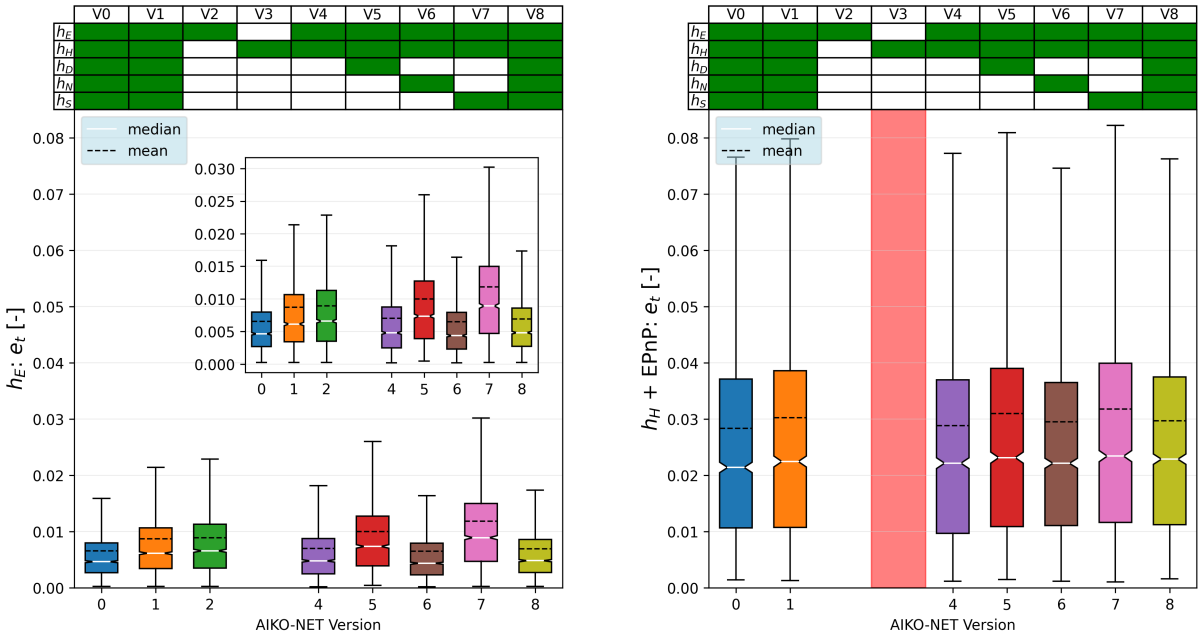


Figure 5.5: AIKO-NET versions normalized translation error comparison. Direct estimation on the left, indirect estimation on the right. Config. `V3` errors are replaced by a red-shaded region because they are notably higher.

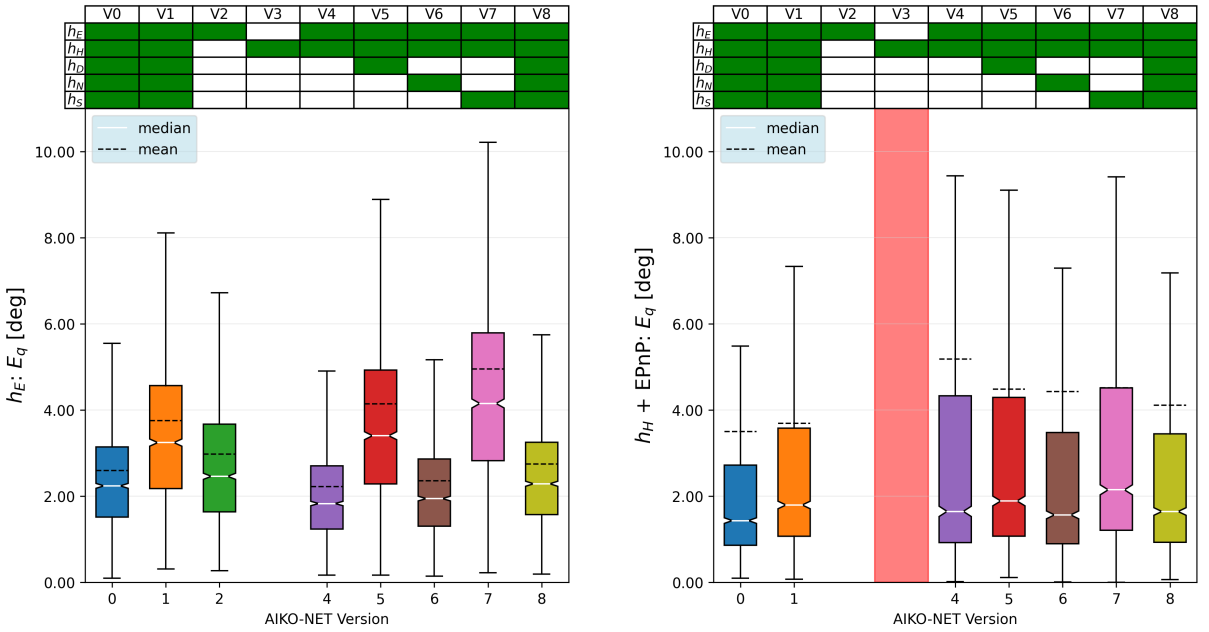


Figure 5.6: AIKO-NET versions quaternion error comparison. Direct estimation on the left, indirect estimation on the right. Config. `V3` errors are replaced by a red-shaded region because they are notably higher.

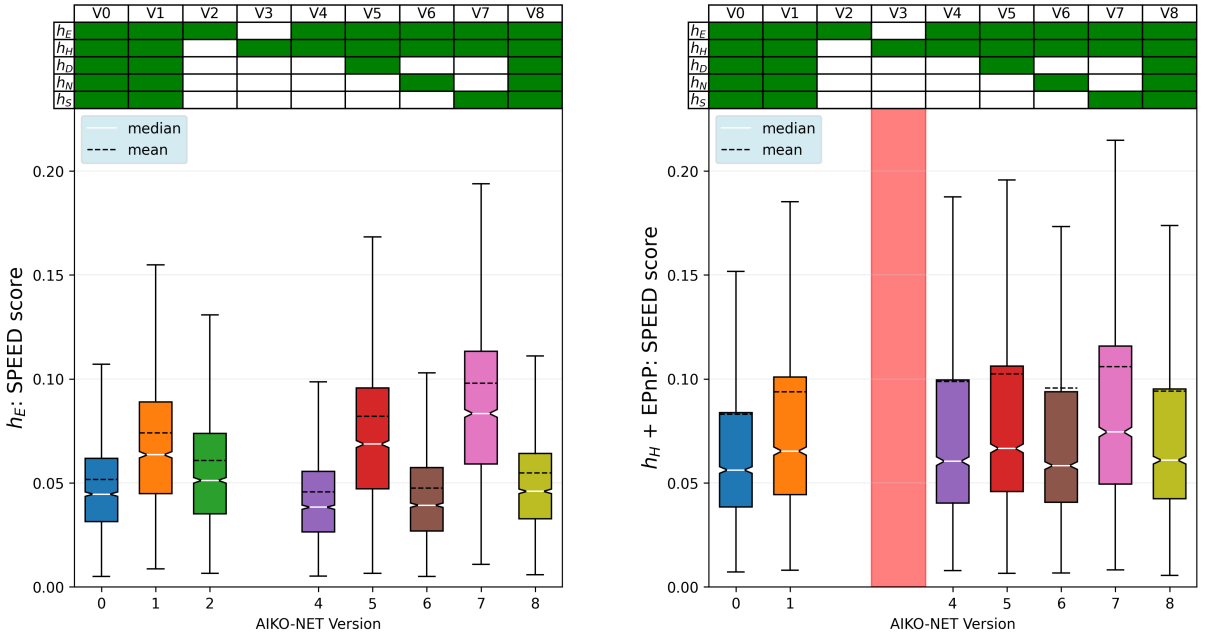


Figure 5.7: AIKO-NET versions SPEED score comparison. Direct estimation on the left, indirect estimation on the right. Config. `V3` errors are replaced by a red-shaded region because they are notably higher.

AIKO-NET V1 is directly comparable to V0 since they share the same configuration except for the backgrounds on the images. The experiments show that the performance of AIKO-NET V1 is slightly worse than V0, as the increased difficulty of the problem affects the overall performance. Nonetheless, the architecture demonstrates robustness to different datasets, which is evidenced by the results convincing quality. In addition, the lighting conditions in the background images do not match the foreground scene, which is a more challenging condition than a realistic one. This mismatch can be seen as a limitation of the dataset, which aims to cover a wide range of diverse cases rather than being representative of real-world scenarios only. The boxplots for V3 are not included in the plots and are replaced by a red-shaded region due to their high errors, which is primarily because the architecture is not optimized solely for keypoint detection. Specifically, the heatmap head does not work on a cropped RoI as in [40], and no keypoint outlier detection method is applied. As a consequence, the performance of this configuration is considerably lower compared to the other ones. It is noteworthy to observe that the performance of the Heatmap head + EPnP method improves significantly when another parallel head, such as EfficientPose, is added to the architecture, as demonstrated by the V4 configuration. The indirect estimation method appears to benefit from the tasks performed by the head responsible for the direct estimation, as the bounding box detection is likely to enhance the quality of the keypoints detection process. It should be noted that the indirect estimation method may not perform well if the architecture is not designed to support it, and if there are no parallel, object-detection related tasks operating. Also EfficientPose benefits from the parallelization with the keypoints heatmap estimation task: its predictions in V4 result more accurate than V2 both for position and attitude. The two heads result to be synergic. Despite the fact that the configurations from V5 to V7 exhibit similar levels of performance, it is interesting to note that the worst results are obtained with V7, which involves using the shadowmap feature in addition to h_E and h_H. The inferior performance of V7 can be attributed to the greater difficulty posed by the shadowmap feature. When compared to V4, both the direct and indirect methods of V5 show worse performance for both the position and attitude estimations. This can be attributed to the high depthmap loss magnitude and its low informative level. On the other hand, V6 with the normalmap head shows slightly better performance than V4, despite the high loss magnitude associated with it. This could be due to the high informative level of the RGB feature, which seems to aid the network in its primary tasks. The boxplots depicting the errors associated with the Heatmap head + PnP method predictions (right column) indicate that the interquartile ranges (IQRs) are typically more spread out. Additionally, in some instances, the range from the median to the third quartile (Q3) is wider than the range from the median to the first quartile (Q1), which indicates that the distribution of errors is skewed towards higher values. It is also worth noting that the means occasionally lie outside of the IQR. This behavior suggests that the model may have difficulty handling certain cases, and it may be worth investigating those cases in more detail to improve the overall performance.

We will now briefly discuss V8, which was set up to be an optimized version of V1. The loss

weights used by `V1` and `V8` are summarized in Table 5.7. The reference nomenclature can be found in Section 3.2.1.

Table 5.7: Loss weights on different AIKO-NET configurations.

	w_{E}	w_{E}^{cls}	w_{E}^{bbox}	w_{E}^{pose}	w_{H}	w_{D}	w_{N}	w_{S}
`V1`	1	1	1	1	1	1	1	1
`V8`	1	0.1	0.5	1	1	0.3	0.5	0.2

The reason behind choosing the specific loss weights is based on the following observations:

- the classification loss is consistently low during the entire training process, and hence, it is considered less important than the other losses;
- the bounding box prediction loss has a lower priority than the pose loss in the EfficientPose head;
- the losses associated with the primary tasks should carry the most significant weights;
- the shadowmap loss has the highest value and should be lowered relatively to focus on more critical tasks;
- among the three additional features, namely depthmap, normalmap, and shadowmap, the normalmap is the most informative and beneficial for the network, and thus, it should have the highest priority among the three.

The test results reveal that `V8`'s EfficientPose head exhibits improved performance in predicting both the relative position and attitude as expected when compared to `V1`. However, there is no significant improvement observed in the relative pose estimation through the indirect method. Finally, the overall performance metrics of `V8` are reported in Tables 5.8 and 5.9 and show a slight worsening with respect to `V0`.

Table 5.8: Global performance of `V8` - mean values and STDs.

	EfficientPose		**Heatmap + EPnP**	
	Mean	**STD**	**Mean**	**STD**
$\mathbf{E}_t$ [mm]	$[9.2, 8.8, 57.8]$	$[9.9, 8.7, 80.0]$	$[24.3, 23.6, 200.2]$	$[23.2, 22.8, 159.3]$
E_t [mm]	59.2	81.1	203.1	162.6
e_t	0.007	0.008	0.030	0.028
$\mathbf{E}_\theta$ [deg]	$[2.1, 1.2, 2.2]$	$[3.1, 1.8, 3.6]$	$[2.7, 1.7, 3.2]$	$[6.0, 4.1, 6.9]$
E_q [deg]	2.7	2.9	4.1	8.0
SPEED	0.05476	0.05413	0.09404	0.11634

Table 5.9: Global performance of `V8` - median values and IQRs.

	EfficientPose		**Heatmap + EPnP**	
	Median	**IQR**	**Median**	**IQR**
$\mathbf{E}_t$ [mm]	$[6.6, 6.6, 29.6]$	$[9.4, 8.6, 59.9]$	$[20.5, 20.2, 177.5]$	$[18.2, 17.9, 188.8]$
E_t [mm]	31.0	61.2	179.8	190.5
e_t	0.005	0.006	0.023	0.026
$\mathbf{E}_\theta$ [deg]	$[1.3, 0.9, 1.5]$	$[1.8, 1.2, 1.9]$	$[1.0, 0.6, 1.1]$	$[2.1, 1.2, 1.4]$
E_q [deg]	2.3	1.7	1.6	2.5
SPEED	0.04593	0.03142	0.06091	0.05277

To provide a final and clear overview of the comparison between all the tested configurations, Figure 5.8 displays the performance from both the direct and indirect methods of AIKO-NET `V0-V8` (`V3` is excluded because the errors are orders of magnitude higher) in a median e_t vs. median E_q plot. Circular markers refer to results from direct estimations, while triangular markers refer to those from indirect ones. The size of the markers is directly related to the SPEED score IQR.

The overall accuracy on the normalized translation is higher for the direct method, while the indirect approach provides a more stable estimation of the relative orientation: generally, the predictions made by the EfficientPose head are more dependant on the network configuration with respect to the ones deriving from the Heatmap head. Looking at the circular markers, we can say that the better the overall prediction quality, the smaller their size. This means that low median errors correspond to low IQRs for the EfficientPose head: as the performance gets better, also the robustness of the network increases, resulting in an enhanced generalization capability.

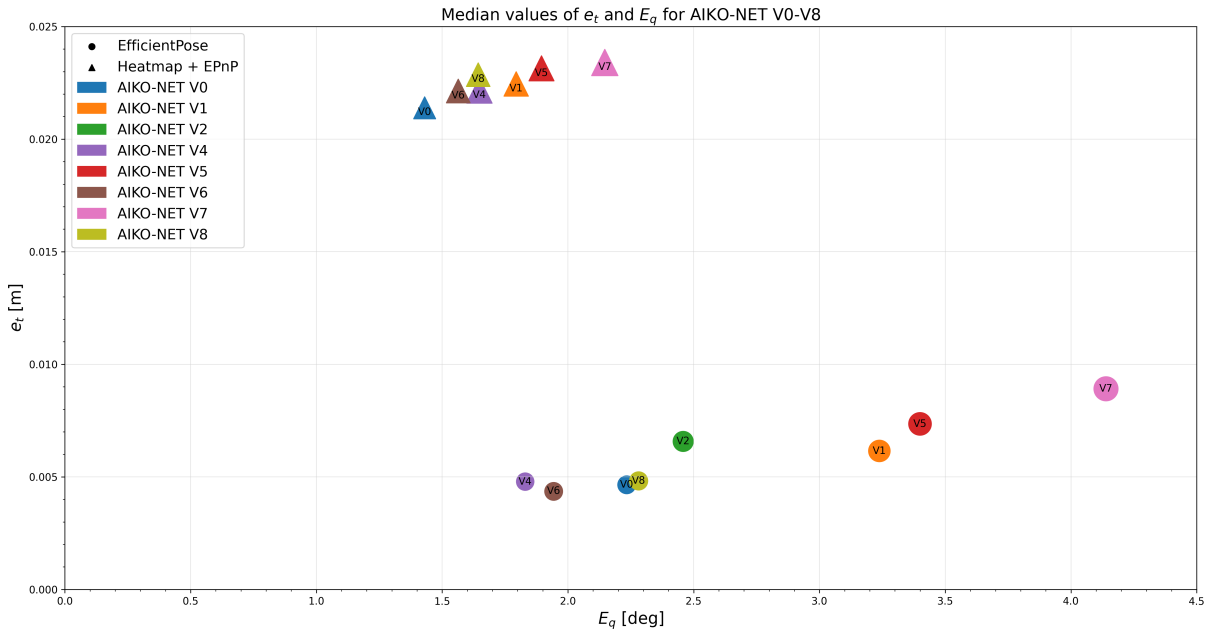


Figure 5.8: AIKO-NET versions: e_t vs. E_q. Marker size depends on the SPEED score IQR. Errors from `V3` are not displayed since they are notably higher with respect to the other versions.

5.2.3. Prediction visualization

In this section, we will visualize the predictions made by our object detection network. We will show the output of the network on sample test images from three different distance ranges: close range (0-5 meters), medium range (5-15 meters), and long range (10-15 meters). For each test image, we will display the input image with overlapped predicted bounding boxes and the output heatmap in the first row. The Tango wireframe will be superimposed on the heatmap to make the visualization more readable and immediate. On the second row, the depthmap, normalmap and shadowmap are reported in this order. This visualization will help us gain a better understanding of how the network is making its predictions and will allow us to identify any areas for improvement. Figures 5.9 and 5.10 show AIKO-NET `V8` predictions of close range images. Although only the EfficientPose head pose prediction is reported, it is interesting to see how the keypoint detection task suffers from high uncertainty, denoted by the halos around the predicted landmarks. This might be one of the issues that cause the indirect method to result in worse performance with respect to the direct one. It has to be noticed that all the feature maps outputs, namely the heatmap, depthmap, normalmap and shadowmap, have a 128×128 resolution. This is because the BiFPN feeds inputs to the prediciton heads only from its 3^{rd} to 7^{th} levels. Consequently, the maximum resolution input available to any prediction head is one-third of the original input 512×512, which corresponds to a 128×128 image.

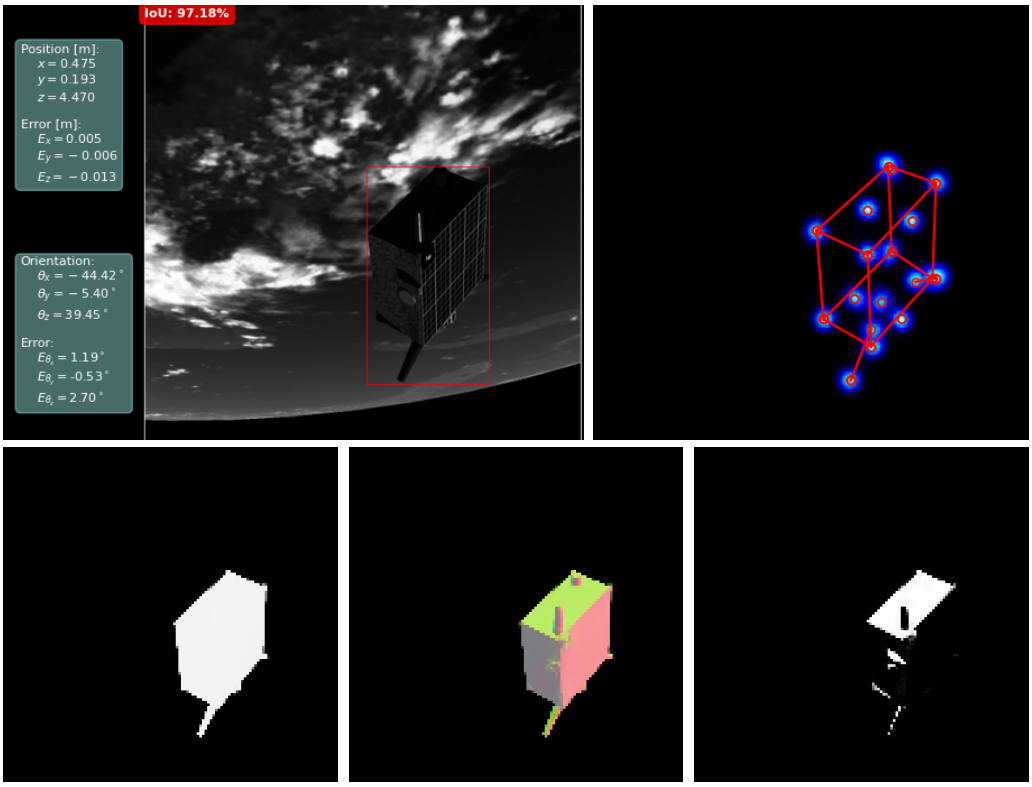


Figure 5.9: Prediction visualization for the close range image *rgb_36434.png*.

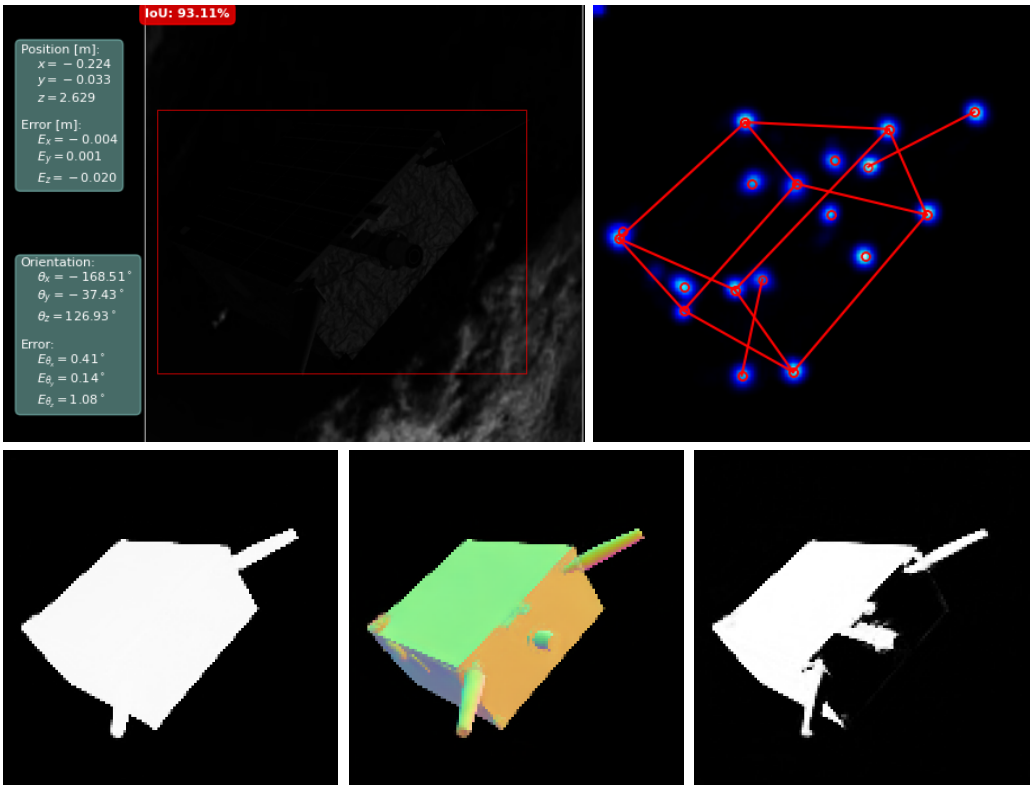


Figure 5.10: Prediction visualization for the close range image *rgb_36369.png*.

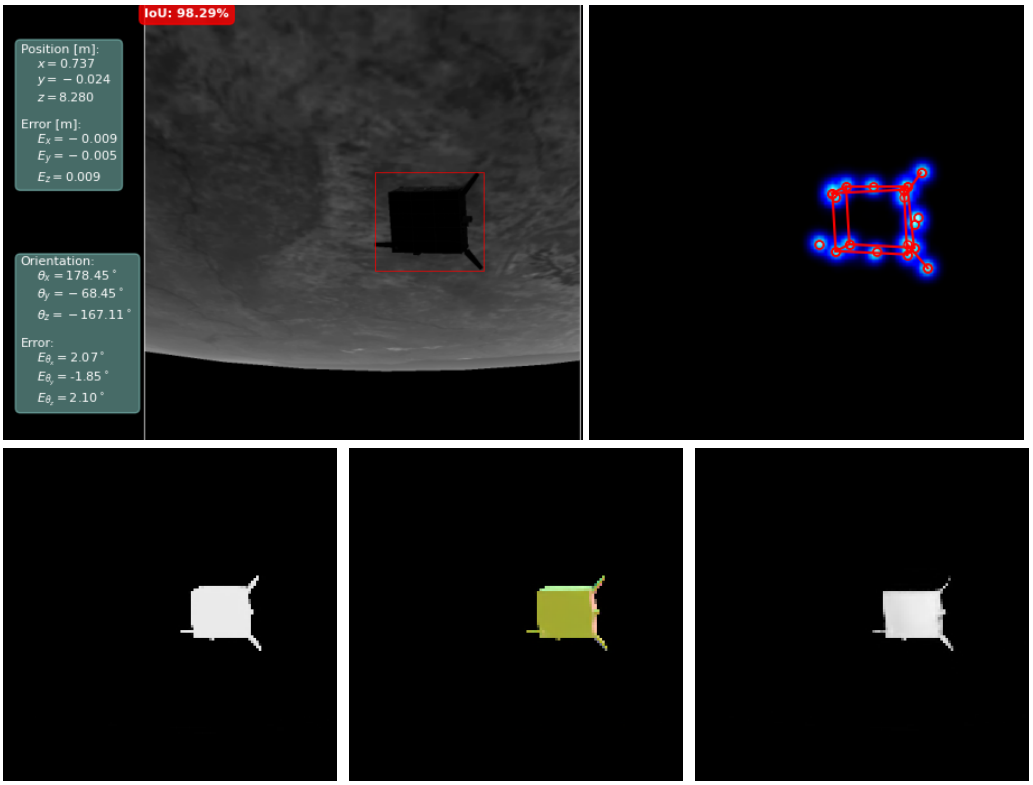


Figure 5.11: Prediction visualization for the medium range image *rgb_36002.png*.

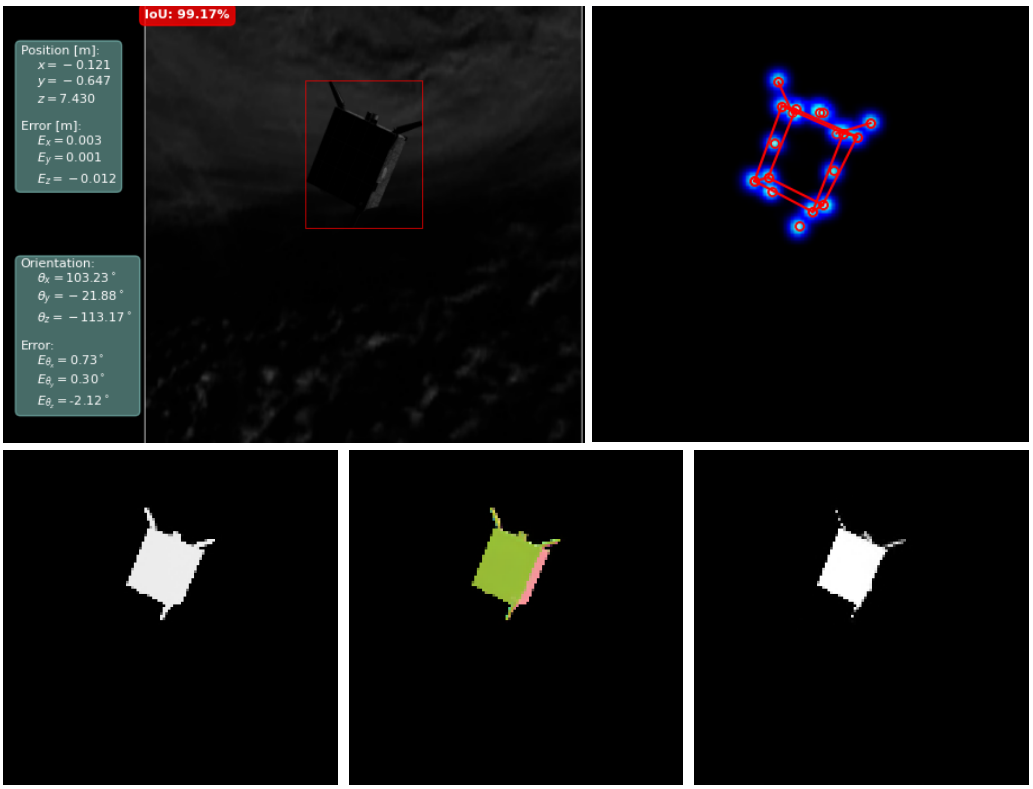


Figure 5.12: Prediction visualization for the medium range image *rgb_36166.png*.

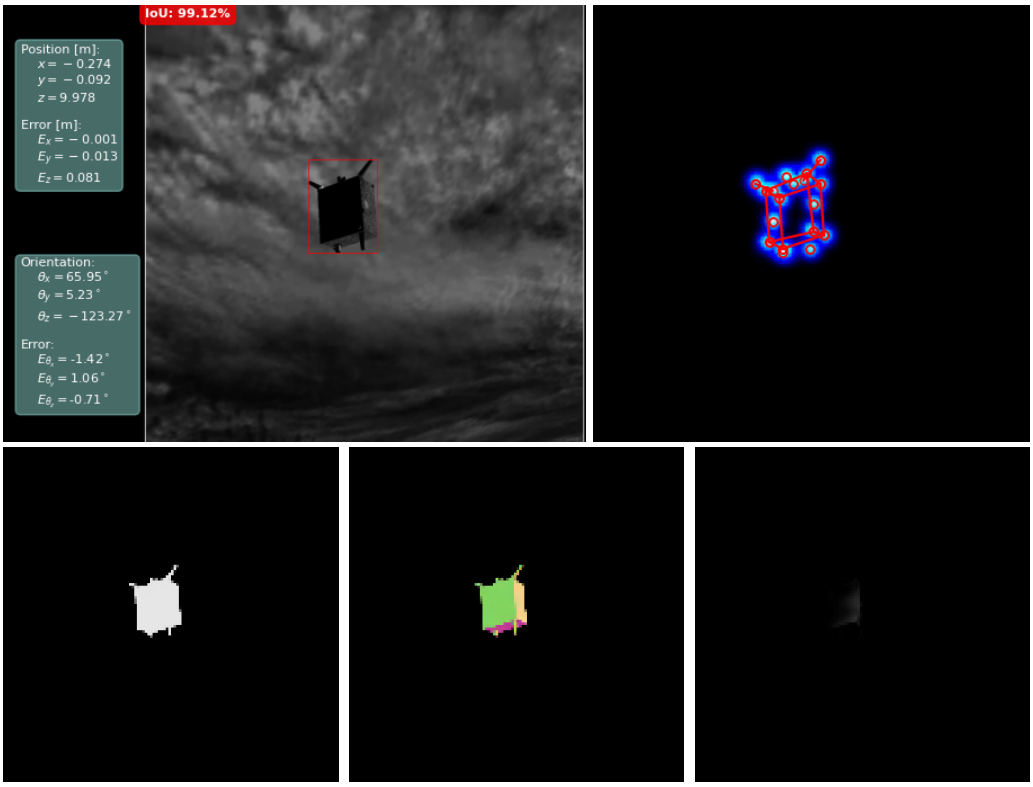


Figure 5.13: Prediction visualization for the long range image *rgb_36171.png*.

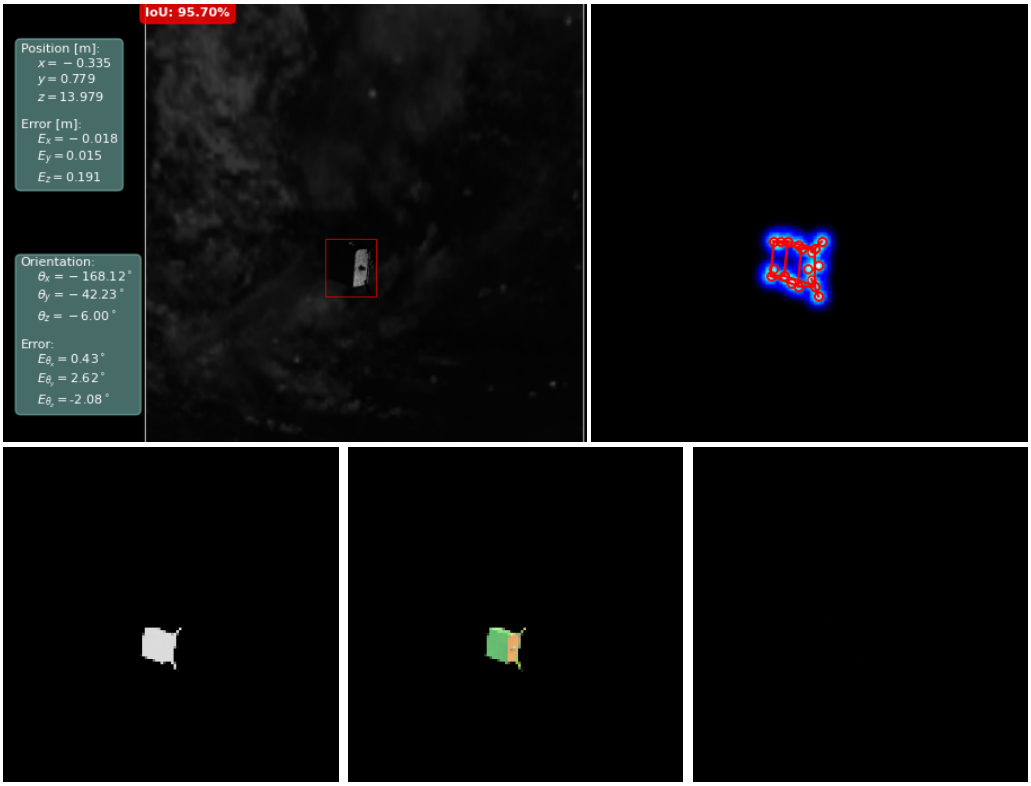


Figure 5.14: Prediction visualization for the long range image *rgb_37300.png*.

In both medium and long range prediction visualizations it can be noticed that, even though EfficientPose can still provide very good pose predictions, the predicted keypoints heatmap feature starts to be a bit more chaotic. It can be observed that the halos around the predicted keypoints start to be more impactful with respect to the 2D relative positions between the single keypoints. As a result, the performance of the indirect method deteriorates as the distance from the target increases. This issue can be attributed mainly to the fixed standard deviation used for generating the heatmap ground-truths. Also, the shadowmap prediction results to be poorly performed in long range scenarios like in Figures 5.13 and 5.14: this feature map gets more difficult to generalize when the target is small, a condition which worsens the already hard task due to the geometric inconsistency with respect to the other feature maps.

5.3. Pose Estimation Pipeline Performance

In order to evaluate the whole pipeline, an example of the full processing of a sequence of images is shown in this section. This test was performed on the orbit displayed in Figure 5.15, generated through the procedure explained in Section 4.1.

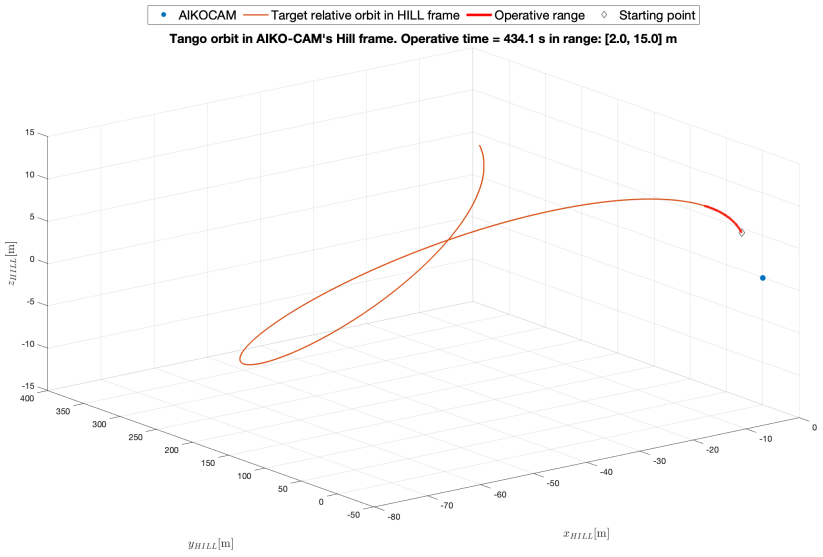


Figure 5.15: Tested relative orbit.

The trajectory data was passed to Unity, and the output sequence of images is briefly presented in the mosaic in Figure 5.16. The ground-truths on the images represent their position in the camera frame. This test was done by assuming a perfect pointing without any divergence so that the x and y ground-truths in the camera frame are always zero. Moreover, the backgrounds are excluded from the scene to work with a simple case, so that the overall results are cannot be affected by any unexpected issue. Thus, the architecture used for prediction is AIKO-NET `V0`.

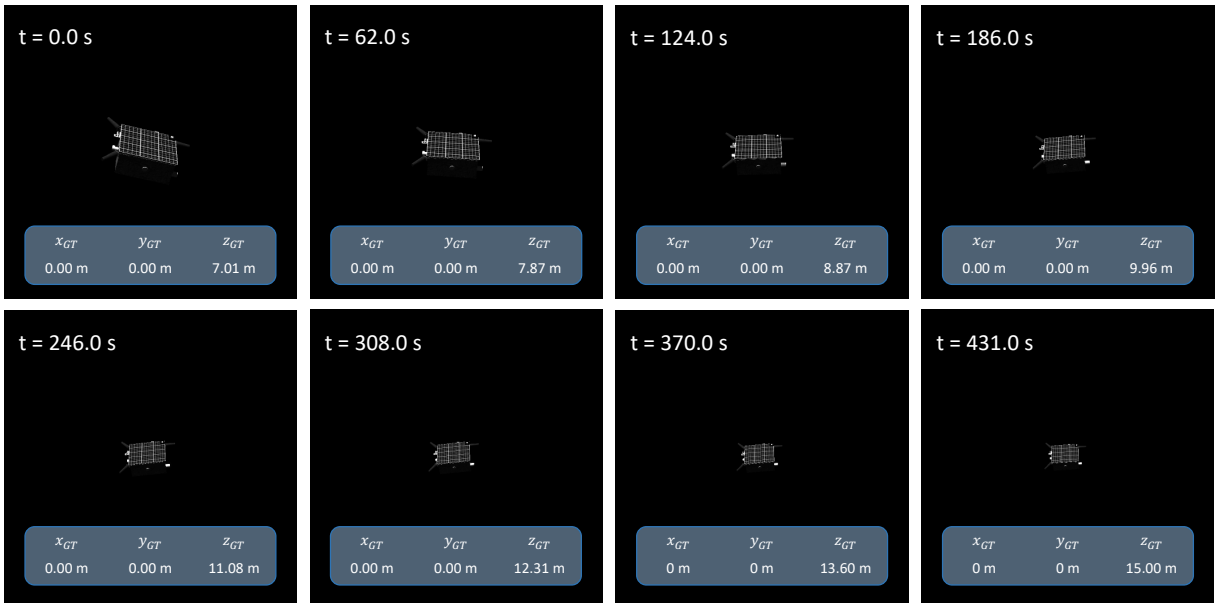


Figure 5.16: Mosaic of images: output sequence from Unity.

In order to enhance the clarity of the data, the filtered and raw measurements were overlaid on the graphs and a logarithmic scale was used. Also, the plots report the 3σ limit: the black line derives from the propagation of uncertainties on the position and defines the range within which the 99.73% of the data points are expected to fall. Error values above the 3σ limit identify outliers. As a first observation, it is good to notice that the CNN predictions on a dynamic, simulated scene result consistent: the errors do not present strange behaviors or sudden spikes, meaning that the proposed architecture robustly handles the pose prediction problem. This is a promising result as it suggests a first step towards the applicability of AIKO-NET to real-world scenarios. Although the predictions made by the CNN are already very accurate, with errors in the order of centimeters and millimeters, Figure 5.17 demonstrates the effectiveness of the filtering action. In this specific test, the EKF may not provide significant additional benefits from the error point of view, but its filtering action would be useful in a more complete application for control purposes by removing excessive noise in the measured states.

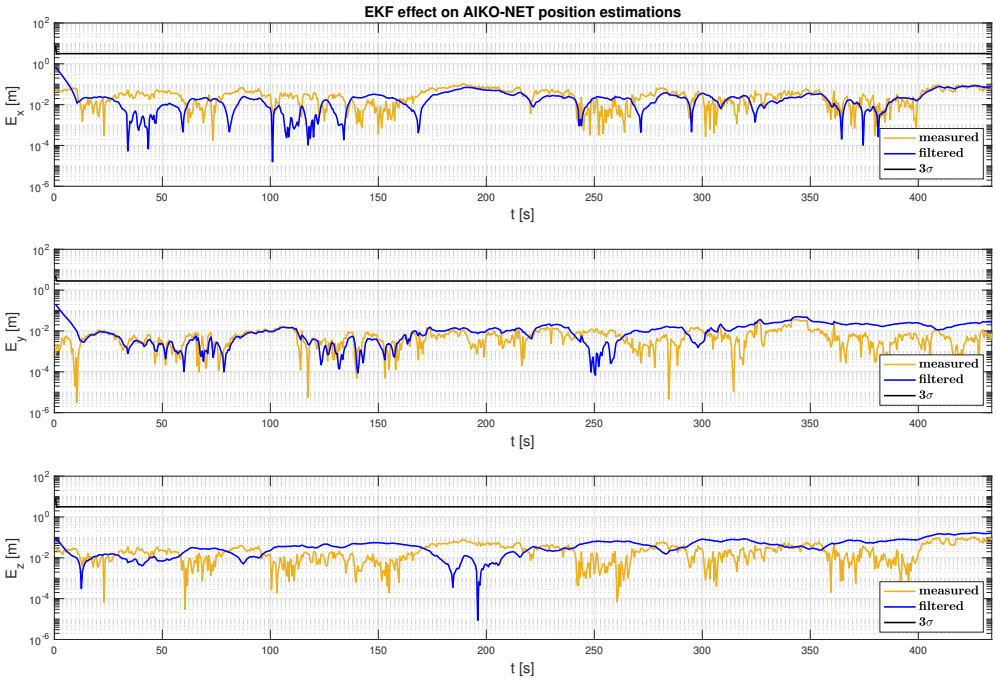


Figure 5.17: Effect of EKF on AIKO-NET position predictions.

To perform a more rigorous evaluation of the EKF's performance, additional noise was deliberately added to the measurements, and the noise measurement matrices were fine-tuned accordingly. Specifically, two levels of disturbances were introduced: the first level was in the order of centimeters, while the second level was in the order of decimeters. The results for each magnitude of disturbance are presented in Figures 5.18 and 5.19, respectively. With these more inaccurate measurements, the beneficial effect of the EKF is clear: the filtering often manages to lower the errors by an order of magnitude. Based on the results, it is clear that the EKF has demonstrated robustness in dealing with inaccurate inputs. Despite the initial inaccuracy, the EKF was able to converge quickly to low error levels, which are comparable to those obtained in the previous test case presented in Figure 5.17.

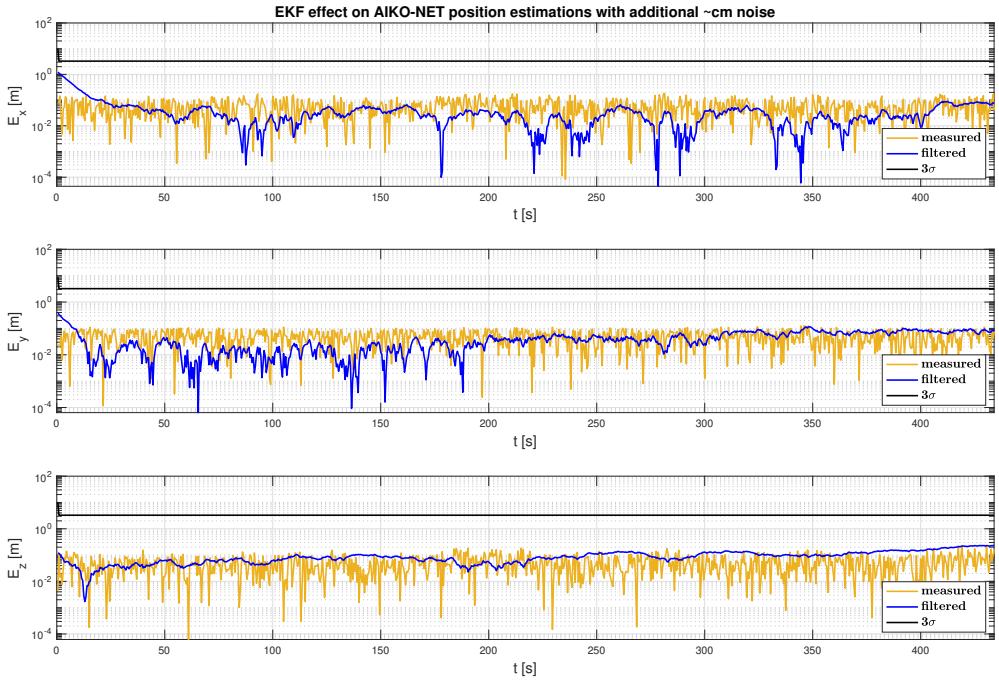


Figure 5.18: Effect of EKF on AIKO-NET disturbed ($\sim cm$) position predictions.

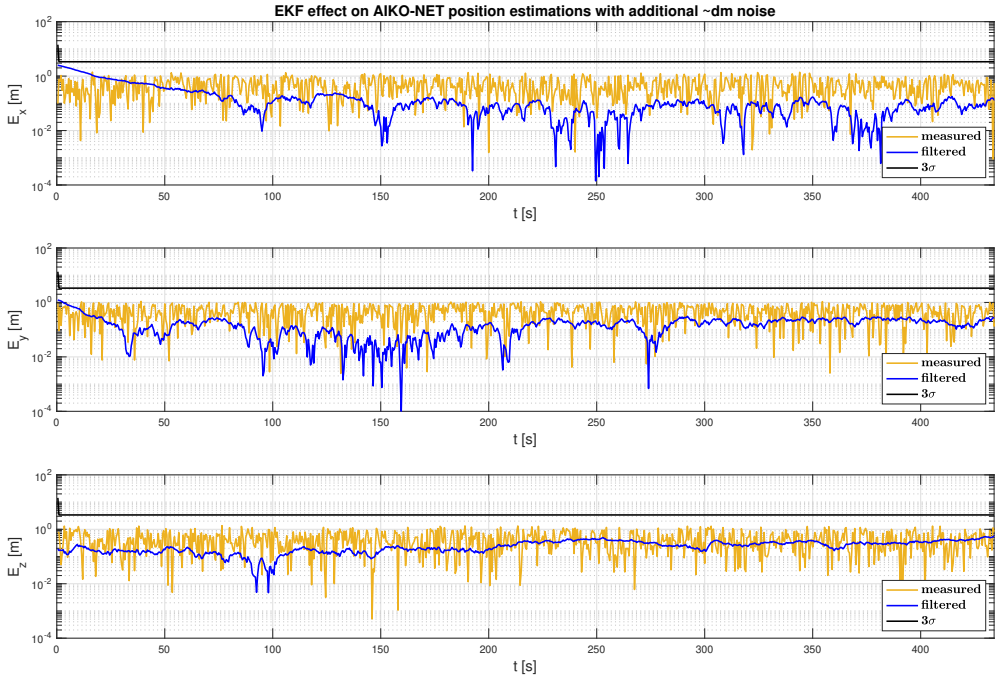


Figure 5.19: Effect of EKF on AIKO-NET disturbed ($\sim dm$) position predictions.

Thus, the development of such a tool can be useful for lowering measurements errors in scenarios where the predictions are not as accurate as the ones provided by AIKO-NET `VO`: this opens scenarios that may consider pose estimation on satellites beyond the trained range, or the application of a lighter, smaller network for which the quality of the predictions may deteriorate with respect to the tested benchmark configuration.

6 | Conclusions and Future work

6.1. Conclusions

This thesis investigates the parallelization capabilities of an AI architecture for relative pose estimation of a known, uncooperative satellite from greyscale, monocular camera images. The study contributes by demonstrating the reproducibility of SLAB's SPNv2 SOTA results and by developing and exploring different configurations of AIKO-NET. AIKO-NET is a MTL based, modular architecture made of a total of 5 prediction heads. Three of such heads do not explicitly solve the pose estimation task but try to help the network in being more accurate. The aim is to assess the architecture's ability to solve multiple tasks simultaneously in order to improve the principal tasks' performance, namely, pose estimation using both the direct and indirect approach. The direct approach involves estimating the 6D pose of a target directly from the output of the EfficientPose head. The indirect approach, on the other hand, relies on the Heatmap head output as input for a PnP solver (EPnP) to output another pose estimate.

We began our investigation by conducting a comprehensive review of the most advanced techniques for pose estimation, focusing on those based deep learning approaches, and paying special attention to the strategies employed by *Park et al.* in their "Spacecraft Pose Estimation Network v2" [39]. Building upon the foundation laid by this baseline work, we developed a complete relative pose estimation pipeline. As AIKO-NET introduces novel feature maps, it was necessary to create a new dataset to ensure a fully adaptable framework. This led to the development of the Multi-Feature Spacecraft Pose Estimation Dataset (MFSPED), which provided us with the necessary resources to train AIKO-NET in all of its configurations. Our study explored nine distinct configurations and tested them on 4000 test images, ultimately showcasing state-of-the-art performance and the benefits of a modular architecture that can be trained with a virtually unlimited dataset.

The CNN developed in this work was integrated into a comprehensive pipeline that begins with the custom generation of trajectories to test the network's performance. This data is then used to generate synthetic images that simulate relative trajectory scenarios. The pipeline concludes with the improvement of pose predictions through the application of an Extended Kalman Filter (EKF), which has shown robustness in handling errors even higher than those provided by AIKO-NET. The aforementioned results highlight the potential of the EKF in effectively

filtering out high errors, which opens up avenues for several possible solutions. For instance, a simpler and smaller network with lower performance but also lower computational costs could be considered. Additionally, simpler backbones could be used, and training for less than 50 epochs could be explored. Furthermore, the modularity of AIKO-NET can be leveraged to modify the network based on real-world scenarios.

6.2. Future work

There is certainly a lot of work to be done and many directions to explore that could make the work presented in this thesis more complete and suitable for real-world scenarios. The most relevant observations and hypotheses on future work are listed here below:

1. Addressing the *domain gap* problem is crucial: the domain gap problem in computer vision refers to the performance degradation of a model trained on data that does not correctly represent the domain of real application. Training on synthetic images a software that should work with real, spaceborne images introduces a series of biases that may lead to inaccurate real-world estimations. To face this problem, one approach would be to improve the training dataset by using, for example, better digital assets which can provide higher-quality images. Data augmentation is a typical method used to face the domain gap: by introducing augmentations to the synthetic imagery, one can increase the diversity of the training set and improve the model's robustness to different conditions.

2. The standard deviation used for generating the ground-truth heatmaps for the keypoints prediction could be optimized based on the relative distance from the target, introducing a dynamic selection of such quantity. This would allow for a more precise representation of the keypoints, which would lead to more accurate pose estimation by indirect method.

3. Exploring other loss types would be interesting: the current loss functions used for the different prediction heads in AIKO-NET could be replaced with other types of loss functions that might be more suitable for the task at hand.

4. Applying a dynamically weighted loss function could be a great optimization method for AIKO-NET: changing the heads' losses weights throughout the training by ensuring, for example, a balanced total loss would be extremely beneficial for the CNN and could make the prediction heads even more synergic.

5. Implementing the Kalman filtering also for the relative orientation measurements would make the prediction pipeline more complete.

Finally, future research directions should pave the way for the deployment of a complete vision-based relative navigation architecture for space systems.

Bibliography

[1] Aiko Space | Aiko. URL https://aikospace.com/.

[2] ClearSpace-1. URL https://www.esa.int/Space_Safety/ClearSpace-1.

[3] D-ORBIT. URL https://www.dorbit.space/.

[4] Home - Infinite Orbits. URL https://www.infiniteorbits.io/.

[5] OSAM-1 | NASA's Exploration In-space Services. URL https://nexis.gsfc.nasa.gov/osam-1.html.

[6] Space Rendezvous Laboratory. URL https://slab.stanford.edu/.

[7] Next Generation Spacecraft Pose Estimation Dataset (SPEED+). URL https://purl.stanford.edu/wv398fc4383.

[8] European Space Agency: Spacecraft pose estimation challenge 2019, 2019. URL https://kelvins.esa.int/satellite-pose-estimation-challenge/.

[9] European Space Agency: Spacecraft pose estimation challenge 2021, 2021. URL https://kelvins.esa.int/pose-estimation-2021/.

[10] FCC Adopts New '5-Year Rule' for Deorbiting Satellites, 9 2022. URL https://www.fcc.gov/document/fcc-adopts-new-5-year-rule-deorbiting-satellites.

[11] Astroscale, Securing Space Sustainability, 2 2023. URL https://astroscale.com/.

[12] CelesTrak, 2023. URL https://celestrak.org/.

[13] NASA Horizons System, 2023. URL https://ssd.jpl.nasa.gov/horizons/app.html#/.

[14] C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, softcover reprint of the original 1st ed. 2018 edition, 1 2019.

[15] M. Bechini, G. L. Civardi, M. Quirino, A. Colombo, and M. Lavagna. Robust Monocular Pose Initialization via Visual and Thermal Image Fusion. *73rd International Astronautical Congress (IAC), Paris, France*, 9 2022. URL https://re.public.polimi.it/bitstream/11311/1221785/1/BECHM02-22.pdf.

[16] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv (Cornell University)*, 4 2020. URL https://arxiv.org/pdf/2004.10934v1.

[17] Y. Bukschat. EfficientPose: An efficient, accurate and scalable end-to-end 6D..., 11 2020. URL https://arxiv.org/abs/2011.04307.

[18] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 6 1986. doi: 10.1109/tpami.1986.4767851.

[19] V. Capuano, S. R. Alimo, A. F. W. Ho, and S.-J. Chung. Robust Features Extraction for On-board Monocular-based Spacecraft Pose Acquisition. *AIAA Scitech 2019 Forum*, 1 2019. doi: 10.2514/6.2019-2005.

[20] R. Caruana. Multitask learning. *Learning to learn*, pages 95–133, 5 1998.

[21] Y. Chauvin and D. E. Rumelhart. *Backpropagation*. Psychology Press, 2 2013.

[22] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 10 2016. URL https://arxiv.org/abs/1610.02357v3.

[23] S. D'Amico, M. Benn, and J. B. Jørgensen. Pose estimation of an uncooperative spacecraft from actual space imagery. *International journal of space science and engineering*, 2(2):171, 4 2014. doi: 10.1504/ijspacese.2014.060600.

[24] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Elsevier eBooks*, pages 726–740, 1 1987. doi: 10.1016/b978-0-08-051581-6.50070-2.

[25] N. Grumman. SpaceLogistics - Northrop Grumman, 3 2023. URL https://www.northropgrumman.com/space/space-logistics-services/.

[26] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-Excitation Networks. *arXiv (Cornell University)*, 6 2018. doi: 10.1109/cvpr.2018.00745. URL http://arxiv.org/pdf/1709.01507.

[27] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 3 1960. doi: 10.1115/1.3662552.

[28] A. Kanazawa, M. J. Black, D. R. Jacobs, and J. Malik. End-to-End Recovery of Human Shape and Pose. *Computer Vision and Pattern Recognition*, 6 2018. doi: 10.1109/cvpr.2018.00744.

[29] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv (Cornell University)*, 12 2014. URL https://www.arxiv.org/pdf/1412.6980.

[30] M. Kisantal, S. Sharma, T. G. 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, 11 2019. doi: 10.1109/taes.2020.2989063.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1 1998. doi: 10.1109/5.726791.

[32] 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, 2 2009. doi: 10.1007/s11263-008-0152-6.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. *arXiv (Cornell University)*, 8 2017. doi: 10.1109/iccv.2017.324. URL `http://arxiv.org/pdf/1708.02002`.

[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. M. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science*, pages 21–37, 10 2016. doi: 10.1007/978-3-319-46448-0.

[35] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1:281–297, 1 1967.

[36] O. Montenbruck and E. Gill. *Satellite Orbits*. Springer Science Business Media, 12 2012.

[37] O. Montenbruck and E. Gill. *Satellite Orbits*. Springer Science Business Media, 12 2012.

[38] T. G. Park, M. Märtens, G. Lecuyer, D. Izzo, and S. D'Amico. SPEED+: Next-Generation Dataset for Spacecraft Pose Estimation across Domain Gap. *arXiv (Cornell University)*, 10 2021. doi: 10.1109/aero53065.2022.9843439. URL `http://arxiv.org/pdf/2110.03101`.

[39] T. H. Park. Robust Multi-Task Learning and Online Refinement for Spacecraft..., 3 2022. URL `https://arxiv.org/abs/2203.04275`.

[40] T. H. Park, S. Sharma, and S. D'Amico. Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft. *arXiv (Cornell University)*, 9 2019. URL `http://export.arxiv.org/pdf/1909.00392`.

[41] J. Patterson and A. Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, 1 edition, 9 2017.

[42] M. Piazza. Deep learning-based monocular relative pose estimation of uncooperative spacecraft, 2020. URL `http://hdl.handle.net/10589/170228`.

[43] 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, 5 2022. doi: 10.2514/1.i011064.

[44] P. F. Proença and Y. Gao. Deep Learning for Spacecraft Pose Estimation from Photorealistic Rendering. *International Conference on Robotics and Automation*, 5 2020. doi: 10.1109/icra40945.2020.9197244.

[45] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1 1999. doi: 10.1016/s0893-6080(98)00116-6.

[46] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions. *arXiv (Cornell University)*, 10 2017. URL `https://arxiv.org/pdf/1710.05941.pdf`.

[47] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *Computer Vision and Pattern Recognition*, 7 2017. doi: 10.1109/cvpr.2017.690.

[48] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 6 2017. doi: 10.1109/tpami.2016.2577031.

[49] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. *International Conference on Computer Vision*, 11 2011. doi: 10.1109/iccv.2011.6126544.

[50] M. Sandler, A. W. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv (Cornell University)*, 6 2018. doi: 10.1109/cvpr.2018.00474. URL `http://arxiv.org/pdf/1801.04381`.

[51] H. Schaub and J. L. Junkins. *Analytical Mechanics of Space Systems*. AIAA, 1 2003.

[52] 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, 6 2020. doi: 10.1109/taes.2020.2999148.

[53] 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, 6 2018. doi: 10.2514/1.a34124.

[54] K. Suzuki. *Artificial Neural Networks*. IntechOpen, 1 2013.

[55] M. Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 5 2019. URL `https://arxiv.org/abs/1905.11946`.

[56] M. Tan. EfficientDet: Scalable and Efficient Object Detection, 11 2019. URL `https://arxiv.org/abs/1911.09070`.

[57] M. Tan, B. Chen, R. Pang, V. K. Vasudevan, M. Sandler, A. W. Howard, and Q. V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *arXiv (Cornell University)*, 6 2019. doi: 10.1109/cvpr.2019.00293. URL `http://arxiv.org/pdf/1807.11626`.

[58] Y. Wu and K. He. Group Normalization. *Lecture Notes in Computer Science*, pages 3–19, 1 2018. doi: 10.1007/978-3-030-01261-8.

[59] Y.-T. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv (Cornell University)*, 6 2018. doi: 10.15607/rss.2018.xiv.019. URL `http://arxiv.org/pdf/1711.00199`.

[60] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, and X. Hu. Data-centric ai: Perspectives and challenges, 2023.

[61] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of the ... AAAI Conference on Artificial Intelligence*, 34(07):12993–13000, 4 2020. doi: 10.1609/aaai.v34i07.6999. URL `https://ojs.aaai.org/index.php/AAAI/article/download/6999/6853`.

Acknowledgements - Ringraziamenti

Ringrazio il mio relatore Pierluigi Di Lizia per l'impegno dimostrato nel supporto costante dall'inizio di questo percorso, e per la gentilezza che lo contraddistingue. Grazie ad AIKO, che mi ha dato l'opportunità di essere protagonista di un progetto stimolante, in un ambiente in cui mi sono sentito sempre ben accolto. Merito soprattuto di Ilaria, Francesco e Tobia, persone di rara capacità ed empatia che mi hanno guidato ed aiutato durante questo percorso impegnativo e divertente. Grazie a Tobia per tutto il fantastico lavoro sul dataset che mi ha permesso di portare avanti la tesi nel migliore dei modi. Un grazie speciale a Francesco, che è stato per me mentore, collega e amico, e che non mi ha mai fatto mancare supporto umano e tecnico.

Grazie ai miei genitori e a mio fratello, che mi hanno dato ogni abbraccio, mezzo e supporto necessari per poter arrivare a questo traguardo, e che hanno sempre creduto in me. Grazie a tutta la mia famiglia per avermi sempre fatto sentire a casa e per avermi dato la carica per continuare a studiare. Grazie a Cix, Lorenzo e Paloma per avermi fatto rimanere bambino.
Alle mie fantastiche coinquiline, ormai come sorelle.
Ai miei amici di San Benedetto, a quelli di Milano e di Siviglia, per tutti i momenti felici durante questi anni di studio.
Ad Antonio e Imane, con i quali ho intrapreso un percorso di regressione mentale che ha trasformato questi ultimi anni universitari in un continuo di risate e battute di infimo, vergognoso livello. Grazie per la vostra amicizia sincera.
Grazie a Lavinia per il suo supporto sempre discreto, per il suo amore, e per la serenità che continua a donarmi.

Questa tesi, così come l'impegno che c'è dietro, è dedicata a Nonno Rino e Nonna Dina. Custodisco nel mio cuore l'orgoglio di aver avuto nella vita due persone così belle che, con cura, mi hanno insegnato i valori dell'amore, del gioco e del lavoro.
Oggi spero che la mia felicità vi raggiunga.