POLITECNICO DI TORINO

MASTER DEGREE THESIS

Department or Control and Computer Engineering

Mechatronic Engineering

a.y. 2022/2023

AI-based Visual Pose Estimation for Space Applications

Supervisors:

Marcello CHIABERGE
(*Politecnico di Torino*)

Andrea MERLO
(*Thales Alenia Space Italia*)

Candidate:

Federico MOSCATO

November 2023

POLITECNICO DI TORINO

Abstract

Department or Control and Computer Engineering

Mechatronic Engineering

AI-based Visual Pose Estimation for Space Applications

by Federico MOSCATO

As the number of in-orbit satellites increases, the need for a way to service them becomes increasingly critical.
Recently the EU funded EROSS, a project with the purpose of providing a new range of services for in orbit satellites with consequent analysis for satellite design and life-cycle management. This initiative aims to enhance the availability of cost-effective and secure orbital services by assessing and validating the essential technological components of the Servicer spacecraft. The incorporation of robotic space technologies working on this project will lead to greater autonomy and safety in executing these services in space, requiring reduced ground-based supervision.

This master's thesis presents an innovative approach to pose estimation using deep learning and computer vision techniques. The research explores the development and implementation of a system for in-orbit satellites pose estimation. Delving into the complexities of rendezvous maneuvers, the system devised herein addresses the challenges associated with achieving and maintaining accurate pose estimations in the ever-changing and demanding conditions of space. Through a comprehensive exploration, this thesis contributes valuable insights and practical solutions to enhance the reliability and efficiency of satellite rendezvous processes.

A mono camera system is employed, reducing the hardware complexity and costs while maintaining performance. The camera captures pictures of the target satellite during the whole approach phase. A deep learning framework, based on a Convolutional Neural Network (CNN), is used to identify and track landmark features on the target satellite from captured images. This CNN-based approach provides high accuracy in feature recognition and tracking precision. A neural network-based regression model is introduced to map the 2D image coordinates or identified landmarks to their corresponding 3D coordinates with respect to the camera frame. This implementation permits to have a mono-camera instead of a stereo-camera system. Finally, incorporating the CPD algorithm, the system aligns the predicted 3D point clouds to the reference model, enabling accurate pose estimation and tracking.

The proposed system is tested through simulations. The results demonstrate the system's capability to estimate the pose of in-orbit satellites. This research contributes to the advancement of autonomous satellite operations, space debris management, and space exploration. Furthermore, it has the potential to enhance satellite rendezvous and capture capabilities.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
CAD	**C**omputer-**A**ided **D**esign
CNN	**C**onvolutional **N**eural **N**etwork
CPD	**C**oherent **P**oint **D**rift
CV	**C**omputer **V**ision
D	**D**imension(s)
DOF	**D**egrees **O**f **F**reedom
EM	**E**xpectation-**Maximization**
EROSS	**E**uropean **R**obotic **O**rbital **S**upport **S**ervices
EU	**E**uropean **U**nion
FAIR	**F**acebook's **AI** **R**esearch lab
FOV	**F**ield **O**f **V**iew
GEO	**G**eostationary **E**quatorial **O**rbit
GMM	**G**aussian **M**ixture **Model**
GPU	**G**raphics **P**rocessing **U**nit
HRNet	**H**igh-**Resolution** **Net**work
ICP	**I**terative **C**losest **P**oint
IOD	**I**n **O**rbit **D**emonstration
LEO	**L**ow **E**arth **O**rbit
ML	**M**achine **L**earning
MSE	**M**ean **S**quared **E**rror
NN	**N**eural **N**etwork
RANSAC	**Ran**dom **Sa**mple **C**onsensus
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
RPY	**R**oll, **P**itch, **Y**aw
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SURF	**S**peeded **U**p **R**obust **F**eature
MSER	**M**aximally **S**table **E**xtremal **R**egions
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
PnP	**P**erspective-**n**-**P**oint
SfM	**S**tructure **f**rom **M**otion
SPN	**S**pacecraft **P**ose **N**etwork
SWaP	**S**ize **W**eight **a**nd **P**ower

List of Symbols

Symbol	Name	Unit
E_{CNN}	Landmark Regression error	px
E_{NN}	Landmark Mapping error	cm
E_T	Translation error	cm
E_R	Rotation error	°
S_T	Translation score	-
S_R	Rotation error	rad

Chapter 1

Introduction

1.1 Thesis objective

In the expanse of space, satellite missions and on-orbit services have become critical assets, serving a myriad of applications including Earth observation, global communication, and scientific research.

The progressive introduction of AI algorithms into various environments, including space applications, represents a significant leap forward in technological advancement. In the context of pose estimation in space, the incorporation of AI brings a multitude of benefits that enhance the autonomy of satellite operations.

In recent years, we've witnessed a rapid proliferation of on-orbit satellites, driven by advancements in technology and the need for enhanced space services. As the number of these satellites continues to rise, the complexities associated with their safe and effective navigation, rendezvous, and scientific missions have grown in tandem. This is where AI shines, as it steps in to revolutionize the field of satellite pose estimation.

In the 2019 the EU funded EROSS [4], a project, with the goal of showcasing key European solutions for "Servicer" and "Client" vehicles, designed to be used in both low Earth orbit (LEO) and geostationary equatorial orbit (GEO). This initiative aims to enhance the availability of cost-effective and secure orbital services by assessing and validating the essential technological components of the Servicer spacecraft. These components are crucial for executing various tasks during satellite servicing operations, such as rendezvous, capture, grasping, berthing, and manipulation of a non-collaborative Client satellites. As a result, the incorporation of robotic space technologies will lead to greater autonomy and safety in executing these services in space, requiring reduced ground-based supervision.

AI algorithms, equipped with their machine learning capabilities, enable satellites to process vast amounts of data from onboard sensors with remarkable precision and efficiency. This means an elevated level of accuracy in determining a satellite's position, orientation, and trajectory. But the benefits go beyond mere precision.

AI algorithms, equipped with their machine learning capabilities, enable satellites to process vast amounts of data from onboard sensors with remarkable precision and efficiency. One remarkable development is the ability to estimate a satellite's position and orientation using just a single camera, eliminating the need for a stereocamera setup. This innovation not only enhances accuracy but also reduces hardware complexity, making satellite design more cost-effective. AI-driven monocular camera-based pose estimation empowers satellites to autonomously process visual data, adjust to dynamic orbital environments, and make informed decisions, even in the midst of complex maneuvers, ensuring the mission's success and safety.

Moreover, the increased autonomy provided by AI minimizes the need for constant human intervention and ground control. This not only reduces operational costs but also allows human operators to focus on more strategic aspects of the mission, enhancing productivity and mission efficiency. As we look to the future, AI algorithms promise to usher in a new era of space exploration and satellite operations.

In summary, the progressive introduction of AI algorithms in space applications, particularly in pose estimation, opens the door to enhanced accuracy, real-time adaptability, autonomy, and overall mission efficiency. This transformative technology propels us closer to unlocking the full potential of space exploration and satellite services.

The objective of this thesis is to implement the rendezvous of a collaborative satellite using AI algorithms, with a particular emphasis on their applications in mono camera-based visual pose estimation. The focus is specifically directed towards a detailed analysis of rendezvous operations within the 200-20cm distance range from a non-cooperative satellite. This project delves into the critical aspects of pose estimation throughout the entire trajectory of the rendezvous process, extending from the initial approach to the final berthing phase.

1.2 Dataset

The algorithm is designed around *TASI EROSS IOD Simulated Dataset N°2*, which consists of grayscale images of the satellite; see figure **1.1**.
The training dataset is composed of sixteen trajectories each of 900 images. Each trajectory covers the distance range from 200 to 20 cm to the target and the difference between each subsequent frame captured by each camera is 0.2 cm. Each image is of size 512x512 pixels and it's paired with ground truth 6DOF poses (position and orientation).

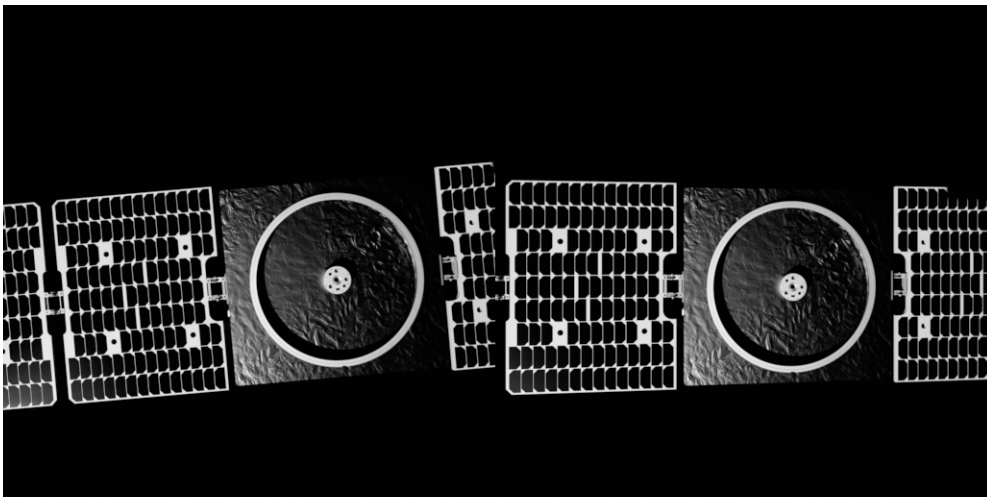

FIGURE 1.1: Sample images from the *TASI EROSS IOD Simulated Dataset N°2*

The data acquisition has been performed as follow:

- **Non Prepared scenario:** LAR view.
- **Natural Illumination:** Full illumination (sun @45°, 130k lux).
- **Illumination system:** ON (150lm x6 LEDs).
- **Simulated Camera Settings:**
  - Shutter speed: 20
  - ISO: 5
  - Aperture: 4
  - FOV: 67.8°
- **Reference System:**
  - Left-handed XYZ reference.
  - Origin [0,0,0]:
    * XY: zeroes on the vertical symmetry axis of the Target.
    * Z: Positive towards contact, zeroed on the lowest contact point of the LAR.
  - All units are in centimeters (cm).
- **Trajectories:**
  - all trajectories follow the same XYZ coordinates.
  - the rotation is considered with Euler angles as Pitch (around axis x), Yaw (around axis y) and Roll (around axis z), all positive counterclockwise. Trajectories' specifics are reported in table **1.1**.
  - Camera pointing XY in [-46, +20].

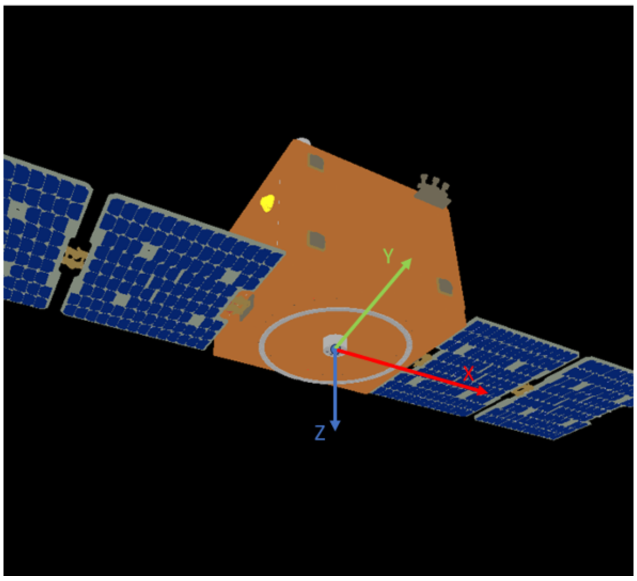

FIGURE 1.2: Model's UPS

TABLE 1.1: Specifics of the simulated training trajectories.

Trajectory	Roll(°)	Pitch(°)	Yaw(°)
TRAY_1	0	0	0
TRAY_2	0	0	-1
TRAY_3	0	0	-2
TRAY_4	0	0	-3
TRAY_5	0	0	-4
TRAY_6	0	0	-5
TRAY_7	0	1	0
TRAY_8	0	2	0
TRAY_9	0	3	0
TRAY_10	0	4	0
TRAY_11	0	5	0
TRAY_12	-1	0	0
TRAY_13	-2	0	0
TRAY_14	-3	0	0
TRAY_15	-4	0	0
TRAY_16	-5	0	0

The *TASI EROSS IOD Simulated Dataset N°3* along with the *Less_Difficult_Trajectory* and *Difficult_Trajectory* are used as the test dataset. The *TASI EROSS N°3* is composed of two trajectories, each of which was captured with two different camera positions. Each trajectory has 900 images.
The *TRAY_A* starts with +15° on Yaw and linearly converge toward 0 on contact, while *TRAY_B*, *Less_Difficult Trajectory* and *Difficult_Trajectory* present multiple errors on RPY converging toward 0 on contact as well.

1.3 Thesis structure

The thesis is structured in further five chapters:

Chapter 2 - *Background*:
This chapter provides a comprehensive overview of key concepts necessary for the correct understanding of this work, with a focus on monocular camera models, perspective projection, pose estimation, and a general introduction to deep learning models.

Chapter 3 - *State-of-art*:
This chapter delves into monocular pose estimation methods, covering classic approaches like RANSAC and SfM, and exploring modern techniques such as end-to-end learning with networks like PoseNet and Mask R-CNN. The chapter also introduces feature learning, emphasizing CNN-based methods like HRNet for predicting 2D landmark locations. Moreover, some studies about spacecraft pose estimation and their use of deep learning architectures are presented. The chapter also delves into point set alignments, highlighting the widely used and advanced algorithms like Coherent Point Drift (CPD) technique employed in the method for final pose estimation.

Chapter 4 - *Algorithms and Methods*:
This chapter delves into the methodology's core algorithms and techniques. It

outlines the offline architecture, detailing the 2D-3D correspondence process, landmark regression, and the neural network-based landmark mapping. The chapter then presents the online architecture, covering real-time processing and the Coherent Point Drift technique for pose estimation. Implementation challenges and dataset considerations are also discussed, providing a comprehensive overview of the applied methods.

Chapter 5 - *Implementation and Experiments*:
This chapter presents the tools and technologies employed for the project implementation and the evaluation metrics for pose estimation, Landmark Regression, Landmark Mapping are described. The chapter culminates in the assessment of both training and test datasets, showcasing the method's robustness and generalization across diverse scenarios. Overall, it provides comprehensive exploration of the research's implementation and experimentation phases.

Chapter 6 - *Discussions and Conclusions*:
The Chapter delves into challenges faced by on-board AI systems in space missions, focusing on verifiability and computational load. It emphasizes the significance of minimizing translation errors for accurate maneuvering in the proposed multi-model configuration. The section explores potential improvements, including enhanced landmark selection and strategies to fortify system robustness.

Chapter 2

Background

2.1 Camera Model

The monocular camera model, a foundational concept in computer vision and imaging, mimics the behavior of a pinhole camera to represent the process of capturing and projecting images. This model simplifies the intricate workings of an optical system to enable a comprehensive understanding of the relationship between the three-dimensional (3D) world and the resulting two-dimensional (2D) image.

In this model, we envision light passing through a minute aperture, analogous to a pinhole, projecting an inverted image onto a photosensitive surface, often a digital or analog image sensor. Key intrinsic parameters, including the focal length and principal point, characterize the camera's optical properties. The focal length dictates the scale of the captured scene, while the principal point marks the location where the optical axis intersects the image plane.

Coordinate systems play a pivotal role in this model, with the camera coordinate system centered at the optical center and the image coordinate system representing points on the image plane. The intrinsic parameters, encompassing the focal length and principal point, along with extrinsic parameters like rotation and translation, define the camera's pose in space.

A fundamental mathematical concept, perspective projection (described in depth in section **2.2**), captures the transformation from 3D points in the camera coordinate system to their 2D projections on the image plane. This equation involves the intrinsic matrix, encapsulating the focal lengths and principal points, and the coordinates of the 3D points.

While the monocular camera model serves as a fundamental abstraction, it often considers ideal conditions, neglecting real-world imperfections such as lens distortion. Distortion correction parameters may be introduced to refine the model, enhancing its accuracy.

Monocular cameras find extensive applications in various domains, from smartphones to surveillance cameras. They are integral to computer vision tasks, such as object recognition, pose estimation and structure-from-motion. The simplicity and versatility of the monocular camera model make it a cornerstone for comprehending the principles of image formation and interpretation in the broader field of computer vision.

The camera model displacement and orientation with respect to the world's reference system can be expressed as follow:

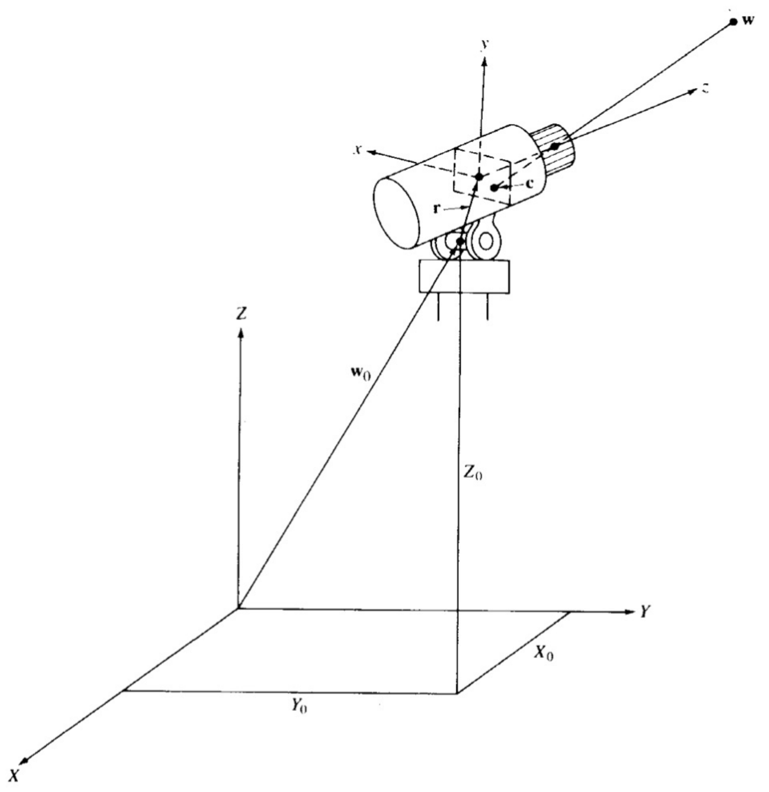

FIGURE 2.1: Camera Model

1. displacement $\mathbf{w}_0$ of the origin of the camera reference system.
2. Pan of x axis (rotation around x axis).
3. Tilt of z axis (rotation around z axis).
4. Displacement $\mathbf{r}$ of the image plane with respect to the center of the joint, on which the camera is mounted and around which can be rotated.

For the two linear displacements, their respective translation matrices are obtained as follow:

$$\mathbf{G} = \begin{bmatrix} \mathbb{1} & -\mathbf{w}_0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} \mathbb{1} & -\mathbf{r} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

The resultant rotation matrix, given the above rotations about x and z axis is:

$$\mathbf{R} = R_\alpha R_\theta = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 & 0 \\ -sin(\theta)cos(\alpha) & cos(\theta)cos(\alpha) & sin(\alpha) & 0 \\ sin(\theta)sin(\alpha) & -cos(\theta)sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

To compute the total transformation of a point from the 3D world to the image plane a perspective projection matrix $\mathbf{P}$ is needed. The next section is dedicated to the deep description of this concept.

The total transformation is computed as follows:

$$c_h = \mathbf{PCRG}w_h \tag{2.3}$$

2.2 Perspective Projection

Perspective projection is a fundamental concept in computer graphics and computer vision. It's a mathematical technique used to simulate how 3D scene or object appears when projected onto a 2D surface, such as a computer scene or image plane. The goal of perspective projection is to create a realistic representation of how objects in the 3D world would look from a particular viewpoint, taking into account the effects of distance and perspective. Some fundamental principles of perspective projections are:

- **Vanishing Point:** objects that are far away from the camera appear smaller and converge to a single point in the distance, called vanishing point. This effect creates a sense of depth and realism in the projected image.
- **Depth Perception:** Perspective projection accurately portrays the relative depth of objects, making objects closer to the camera larger and objects father smaller. This mimics the way human eye and camera lens perceive depth in the real world.
- **Foreshortening:** Perspective Projection results in foreshortening, where objects viewed from an angle are distorted in their shape and dimensions. This distortion is crucial for creating realistic images.
- **Depth Cues:** Perspective Projection includes depth cues, such as the overlap of objects, changes in size, and relative position of objects in the field of view., which help the viewer understand the spacial relationships between objects.

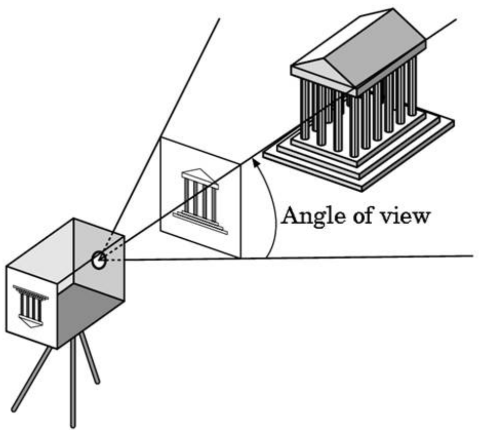

FIGURE 2.2: Perspective projection.

In computer graphics, the perspective matrix (or projection matrix) is used to transform 3D points in 2D coordinates on the screen. This projection is a crucial step in rendering 3D scenes in 2D images.[10]
The perspective matrix has several configurations depending on the specific conventions, camera parameters or coordinate systems used. The mono camera used in this project is considered ideal, with negligible distortion coefficient and squared field of view.

The field of view (FOV) is a fundamental concept in optics, computer graphics and computer vision. It refers to the extent of the observable world that can be seen through a particular device, such as camera, human eye or computer screen. The FOV determines the angle within which object or scenes are visible and it's usually specified as an angular quantity. In this setup, a squared field of view (FOV) implies that the vertical viewing angle is the same as the horizontal viewing angle.
The image below present a visual representation of the FOV:

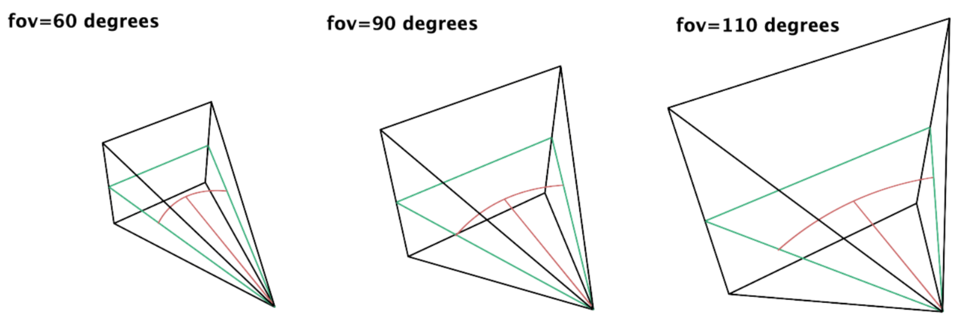

FIGURE 2.3: Visual FOV representation.

As discussed above, a simple implementation of the perspective matrix is enough for our purposes. The matrix is defined as follows:

$$\mathbf{P} = \begin{bmatrix} \frac{f}{a_r} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{(z_{far}+z_{near})}{(z_{near}-z_{far})} & \frac{2*z_{far}*z_{near}}{(z_{near}-z_{far})} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{2.4}$$

The f is the focal length of the camera and it's computed as $f = 1/tan(\frac{FOV}{2})$ with FOV expressed in radians. The a_r is the aspect ratio coefficient needed if the horizontal field of view is different from the vertical one, in this configuration it's 1. The z_{near} and z_{far} represent the distances to the near and far clipping planes, respectively. They define the range of distances of objects in the scene when projected from 3D space to 2D space.

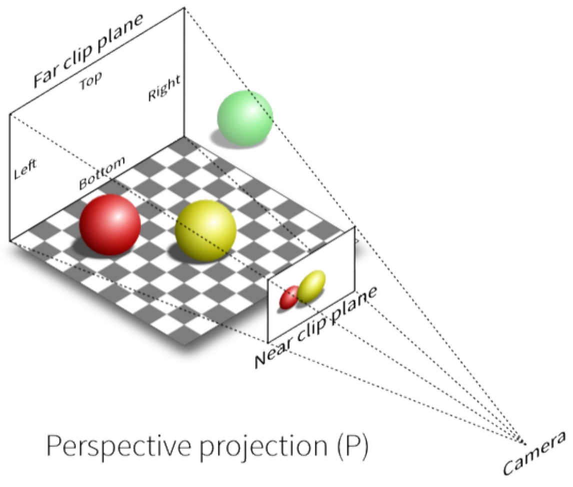

FIGURE 2.4: Object's perspective projection

2.3 Pose Estimation

Object pose estimation is a fundamental task in computer vision that involves determining the spatial orientation and position of an object in a given environment. The "pose" refers to the object's six degrees of freedom (6DOF), which include its translation (movement) along the x, y, and z axes and its rotation around these axes (roll, pitch, and yaw). The goal is to accurately understand how an object is positioned and oriented with respect of a reference coordinate system.

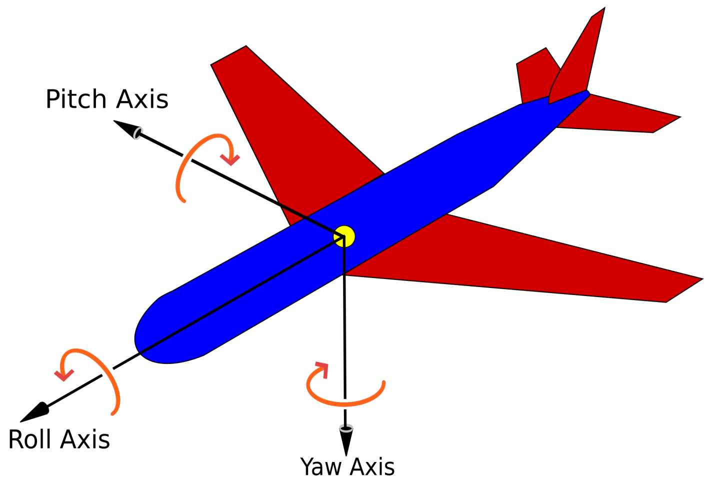

FIGURE 2.5: Roll, Pitch and Yaw rotations (RPY) of an object.

Key aspects of object pose estimation include:

- **Object Representation:** Objects are often represented by 3D geometric models or point clouds. These models describe the shape and structure of the object in a coordinate system.
- **Sensors:** Pose estimation relies on data acquired from sensors, such as cameras or depth sensors. Each sensor type has its strengths and limitations in capturing the necessary information for pose estimation.
- **Feature Extraction:** Features or keypoints (landmarks) are identifiable points on the object's surface that can be matched between the 3D model and the sensor data. These features serve as reference points for determining pose.
- **Matching and Correspondence:** The process involves finding correspondences between the features in the 3D model and those detected in the sensor data. Techniques such as feature matching and point cloud registration are employed for this purpose.
- **Pose Computation:** Once correspondences are established, algorithms calculate the pose parameters. This step involves identifying translation and rotation that best align the 3D model with the observed features in the sensor data.
- **Optimization:** Iterative optimization methods are used to refine the initial pose estimation. These methods aim to minimize the difference between predicted and observed feature locations.

- **Applications:** Object pose estimation is crucial in various applications, such as robotic manipulation, augmented reality, autonomous navigation, and quality control in manufacturing. It enables machines to interact with the environment and make informed decisions based on the perceived spatial relationships of objects.

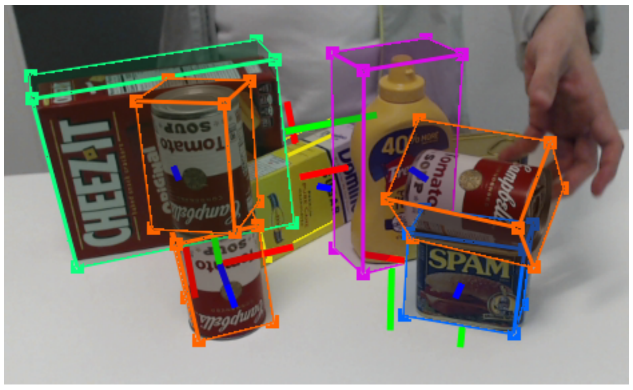

FIGURE 2.6: Example of applications of pose estimation.

Accurate object pose estimation is essential for tasks where knowing an object's precise location and orientation is critical for effective and safe interaction with the environment. Advances in computer vision, machine learning, and sensor technologies contribute to the ongoing improvement of object pose estimation methods.

2.4 From Machine Learning to Deep Learning

[1] In the vast realm of artificial intelligence, a crucial discipline emerges: machine learning (ML), a transformative approach to programming that defies conventional code-based paradigms. At its essence, machine learning empowers systems to evolve and improve through experiences, learning patterns from data rather than relying on explicit instructions. This paradigm shift opens the door to a new era of problem-solving, where algorithms become adept at making decisions, predictions, and inferences based on the information they ingest.

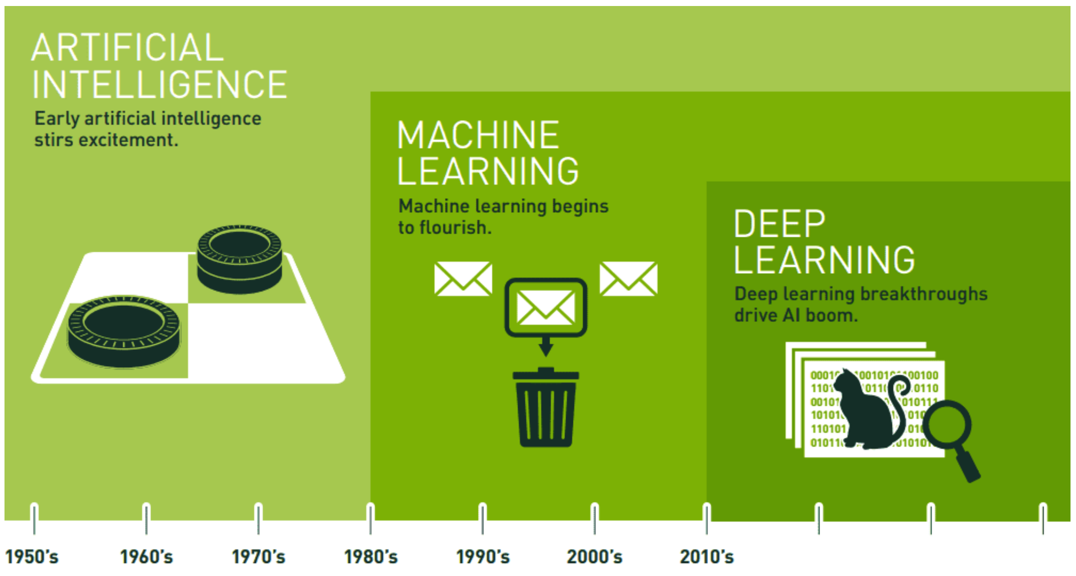
ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

FIGURE 2.7: Timeline of AI toward Deep Learning.

At the core of machine learning lies the intricate paradigm between algorithms and data. The learning process begins with a robust dataset, a collection of input features paired with corresponding outcomes. This data serves as the fodder for ML algorithms, mathematical constructs that decode patterns and relationships within the information. As the algorithm processes the data, it continually adjusts its internal parameters to better align its predictions with the ground truth.

Machine learning encompasses various learning paradigms, each tailored to specific challenges. In supervised learning, algorithms learn from labeled data, associating inputs with desired outputs. Unsupervised learning tackles unlabeled data, seeking inherent structures and relationships. Reinforcement learning introduces the element of interaction, where algorithms learn through trial and error, navigating an environment and adapting based on feedback.

As the capabilities of machine learning burgeoned, a more specialized branch emerged: deep learning. This paradigm shift brought about the rise of deep neural networks, inspired by the complexity of the human brain. Deep learning algorithms, structured as multi-layered neural networks, exhibit an unparalleled ability to automatically learn hierarchical representations from data.

[1]The *"T81-558: Applications of Deep Neural Networks"* course of the Washington University [9] and the *"CS231n: Deep Learning for Computer Vision"*[17] one are used as reference for the presentation of the topic.

Within the deep learning framework, neural networks delve into the intricacies of feature learning. These models, often identified as deep neural networks, boast multiple layers that autonomously extract increasingly complex features. The depth of these networks empowers them to uncover intricate patterns, making them particularly effective in tasks such as image and speech recognition.

The training process in machine learning and deep learning involves an iterative refinement of models. Algorithms, armed with backpropagation mechanisms, fine-tune their internal parameters to minimize the discrepancy between predicted and actual outcomes. This continual optimization results in models that not only perform well on training data but also generalize effectively to new, unseen data.

The applications of machine learning and deep learning span a multitude of domains, reshaping the landscape of technology and problem-solving. From image and speech recognition to natural language processing and autonomous systems, these methodologies have demonstrated unprecedented success. As we delve into the intricacies of machine learning and explore the depths of deep learning, we uncover the transformative power of algorithms that learn, adapt, and redefine the boundaries of artificial intelligence.

2.4.1 Neural Network (NN)

The neural network is one of the first deep learning model. It emulates how neurons function in the human brain using connected circuits to simulate the intelligent behaviour.
Neural networks accept as input a feature vector with fixed length and produces as output a vector of predicted values with fixed length as well. Usually changing the input or output vector length means redesigning the entire structure.
The term *"vector"* is usually referred to 1D arrays but with modern neural network it is increasingly common to find multiple dimensions arrays (as I will discuss later with the CNN).
The term *"dimension"* can be misleading in neural networks, since, used in sense of the dimension of a vector, it refers to the number of elements present in that vector (for instance, a neural network with ten input neurons has ten dimensions). However, in the case of CNN the input has multiple dimensions, so 2D input to a neural network with 128x128 pixels leads to a configuration of 16,368 input neurons. In the first example the neural network will be defined as 2D NN, in the second one 16,368D.

Regression and Classification

The neural network can functions in regression or classification. In the first case the output is a number predicted on the base of the input data, in the second one the identification of a specific class o category. It is important to note that the output of a regression or two-class classification model is a number (binary for the classification 1 for true value 0 or -1 for false value), but in case of multi-class classification the neural network has an output neuron for each category.

Neurons and Layers

The artificial neuron receives as input one or more sources from input data or previous layer neurons. It multiplies each of those inputs by a respective weight and

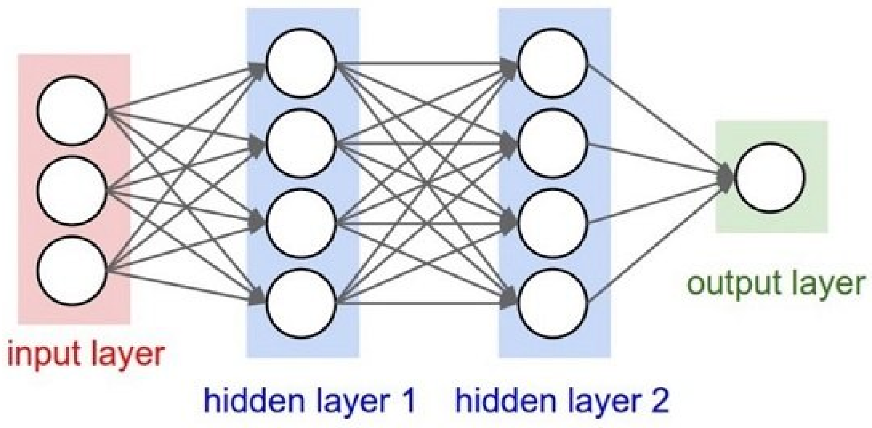

FIGURE 2.8: Neural Network's layer representation.

sums these multiplications together (sometimes also a bias factor is added). The result is passed to an activation function that determines the output of the neuron.

$$f(x, w) = \phi(\sum_i (w_i x_i)) \tag{2.5}$$

In the above equation the variables x and w represent the input and the weights of the neuron respectively, the $\phi(\cdot)$ the activation function and *f(x,w)* the output of the neuron.

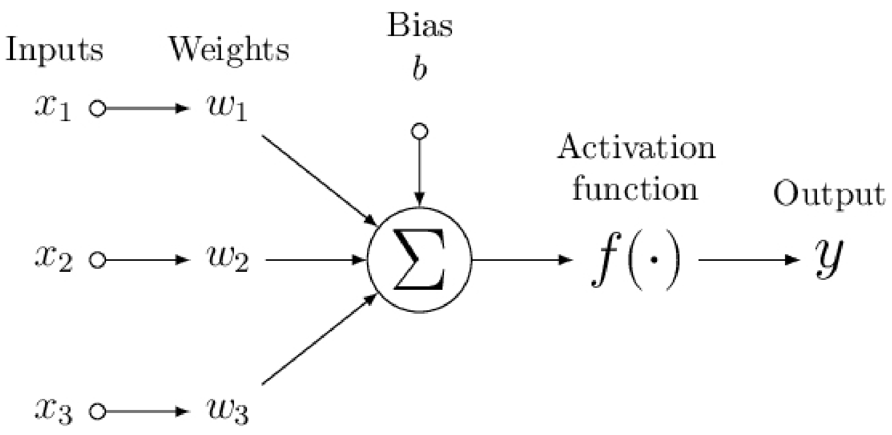

FIGURE 2.9: Artificial Neuron representation.

The neurons can be classified in four categories, depending on their position in the architecture:

- **Input Neurons:** each input neuron is mapped to an element in the feature vector.
- **Hidden Neurons:** intermediate neurons responsible of the abstraction of the neural networks to process the input into the output.
- **Output Neurons:** each output neuron calculates one part of the output.

- **Bias Neurons:** introduces an additional learnable parameter that improves the prediction's accuracy by shifting the activation threshold on the activation function.

Activation Functions

Activation functions, or transfer functions, have the role of computing the output of each layer of a neural network. Historically the more common are hyperbolic tangent, sigmoid, or linear activation functions. However, with modern deep learning models, specialized activation functions have been introduced to suit specific applications and tasks:

- **Rectified Linear Unit (ReLU):** use for the output hidden layers.
- **Softmax:** used for the output of classification neural networks.
- **Linear:** used for the output of regression neural networks.

In particular ReLU has gained rapid popularity in deep learning due to its ability to yield superior training results. Before the era of deep learning, the sigmoid function was the most prevalent activation function. However, as modern frameworks like PyTorch frequently train neural network using gradient descent optimization, computing partial derivatives of the sigmoid became a computationally challenging operation. The introduction of the Rectified Linear Unit significantly simplified this computation leading to improved performance and faster training.

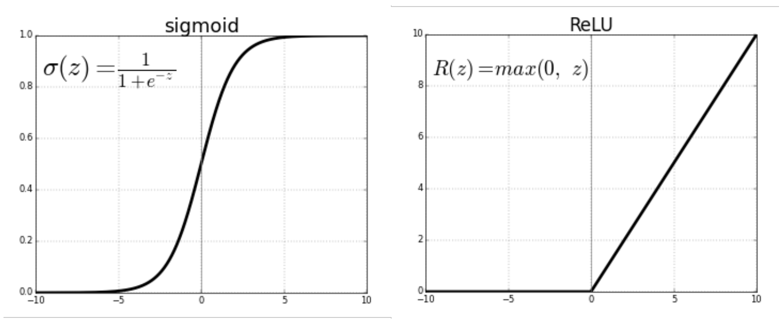

FIGURE 2.10: Sigmoid and ReLU activation functions graphic representation.

2.4.2 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) is a neural network technology that has deeply impacted the area of computer vision. The original concept of a CNN was introduced by Fukushima in 1980[7] and subsequent significant advancements were made by LeCun *et al.*[15].

The CNN, in contrast with most neural networks, has a specific order of the input elements and it is crucial to the training. The inputs are arranged into a grid which represents the input image, the highest-level unit. On the other hand each pixel is the smallest unit within the image and represents a scalar value denoting the intensity or color at a specific location. The images can be colored (expressed on three channels RGB) or grayscale with a single channel. Due to the amount of data, additional layers are needed to lighten the computation load. The layers works together to extract features from the input data and predict the output. The additional layers are:

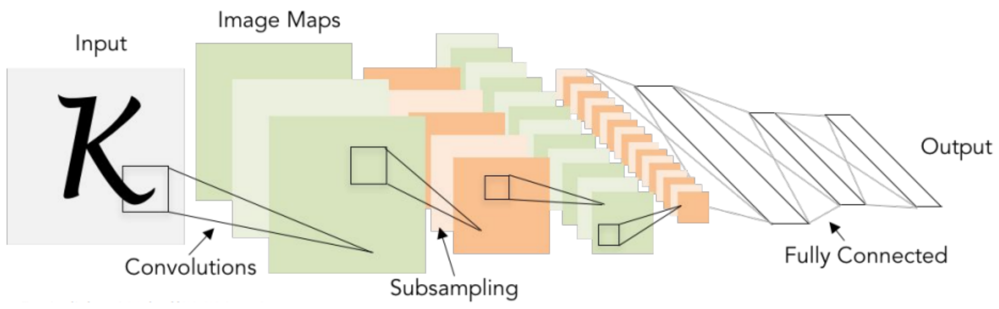

FIGURE 2.11: Convolutional Neural Network visual representation.

- **Convolutional Layers:** apply filters or kernels to perform convolution operations, extracting local features like edges and shapes.
- **Pooling Layers:** reduce the spacial dimensions of feature maps, making the network more robust to scale and orientation variation.
- **Dense Layers:** these layers connect all neurons to every neuron in the previous and subsequent layers, performing high-level feature extraction and decision-making.
- **Flattening Layer:** reshape the output from convolutional layers to 1D vector, allowing it to connect to fully connected (dense) layers.

Training and Testing Processes

The training process begins with the initialization of the network's weight and biases; usually these parameters are set to small random values.
The input is then fed through the network, the data propagates through each layer and predictions are produced. A loss function is used to quantify the error between the predicted and the ground truth values. The choice of the loss function depends on the specific task, the most common are MSE for regression and cross-entropy for classification. The backpropagation algorithm is employed to compute the gradients of the loss with respect to the network's parameters (weights and biases).
This process is essentially the chain rule from calculus, which calculates how small changes in each parameter affect the loss. Using the gradients computed during backpropagation, the network's weights and biases are updated through an optimization algorithm.

The goal is to adjust the parameters in a way that minimizes the loss function. The process is then iterated over the entire training dataset with a process called epoch. Multiple epochs are typically performed to improve the model's performance.

Periodically, the model's performance is evaluated on a separated validation dataset that the network has not seen during training. This method helps monitor the model's generalization capabilities.

After training and validation, the model is tested on a completely unseen test dataset to evaluate its performance on new, independent data.

Chapter 3

State of art

3.1 Monocular pose estimation

3.1.1 Classic Methods

Traditional pose estimation techniques usually use hand-crafted landmarks detectors and descriptors e.g. SIFT, SURF, MSER and BRIEF. These features serve as anchor points for establishing correspondences between 2D and 3D space, then estimate the pose using non-linear optimisation from the correspondence set. The landmarks are detected automatically and described using heuristic measures of geometric and photometric invariance.

Random Sample Consensus (RANSAC)

Robust algorithms like Random Sample Consensus (RANSAC) [6] play a pivotal role in filtering out outliers and estimating pose accurately. It is a robust algorithm widely used in computer vision and geometry computations for model fitting. As presented in [18], its primary objective is to estimate parameters of mathematical model from a set of observed data contaminated with outliers. RANSAC achieves this by iteratively selecting random subsets of data points, fitting a model to each subset, and identifying the consensus set of inliers that agree the model.

After the random selection of a minimal subset of data points, the model is fitted to the randomly selected subset. The type of model depends on the specific application, such as lines, planes or more complex structures. The model is then used to classify the remaining data points as inliers or outliers based on a predefined threshold. Data points that fit the model within the threshold are considered inliers.
The process is repeated from a fixed number of iterations, and the model with the largest consensus set (the set of inliers) is retained. With the consensus set, the model is refined using all inliers, and the final model parameters are obtained.

However, this method is susceptible to changes in lighting conditions and large variations in pose and texture. Nonetheless, the earlier research has given birth to effective and well-understood geometric algorithms (e.g. PnP solvers) that are able to estimate the pose accurately and robustly, given a reasonable correspondence set.

Structure from Motion (SfM)

SfM [32] represents a powerful paradigm for reconstructing the three-dimensional structure of a scene from a sequence of two-dimensional images. It assumes that the scene consists of static objects, and the motion is captured by the movement of the camera. SfM simultaneously estimates the camera poses and the 3D structure of the scene. Bundle Adjustment is a fundamental optimization technique within SfM, iteratively refining camera poses and the associated 3D structure. This approach is particularly effective in scenarios with sequential image captures, as seen in applications like photogrammetry.

Firstly distinctive features, such as keypoints, are extracted from each image in the sequence and matched together, establishing 2D-2D or 2D-3D correspondences. The camera poses for each image are estimated using techniques like Perspective-n-Point (PnP) algorithms [16]. The 3D coordinates of the scene points are mapped with a triangulation technique using the established correspondences and camera poses. Lastly the camera poses and 3D points are jointly refined to minimize the reprojection error across all images.

SfM is widely applied in applications such as creating 3D models of structures, objects, or scenes from a collection of images. It contributes to the alignment of virtual and real-world elements in augmented reality applications.

3.1.2 End-to-End Learning

Motivated by the success of deep learning in image classification and object detection, end-to-end learning methods for pose estimation have emerged ([30], [2], [11]). The advent of convolution neural networks (CNN) has revolutionized pose estimation by enabling the automatic learning of hierarchical features from images. These networks can be applied for end-to-end pose regression or integrated into larger systems. Their adaptability makes them suitable for real-time systems and object tracking.
These approaches utilize CNN architectures to learn complex non-linear mappings from input images to output poses. However, despite these end-to-end methods have demonstrated some success, they not achieved similar accuracy as geometry-based solutions.

PoseNet

PoseNet [12] is a pioneering deep learning architecture designed for estimating the camera pose directly from images. It reframes the pose estimation problem as a regression task, predicting both translation and rotation parameters.
The network architecture typically includes convolutional layers for feature extraction followed by fully connected layers for regression.

This approach is valuable in applications where real-time translation and rotation information are critical. However, the training process becomes more complex as diverse datasets are required for effective model generalization.

Mask R-CNN

Mask R-CNN [8] represents a fusion of object detection and instance segmentation, providing detailed pose information for each detected object. In additional to identifying objects in image, it provides detailed information about the shape and location of each instance, making it suitable for accurate pose estimation.

The model extends the Faster R-CNN architecture by incorporating an additional branch for predicting masks. This enables precise delineation of object boundaries, contributing to improved pose estimation, especially in scenario with complex object shapes.

This architecture is particularly effective in scenarios where precise delineation of object boundaries is essential. However, its computational demands are higher compared to simpler architectures.

Hand-Eye Coordination Networks

Networks designed for hand-eye coordination focus on tasks where precise pose estimation is critical for robotic manipulation. These networks contribute to the seamless interaction between robotic arms and surrounding environment.

These networks often involve architectures tailored for the specific requirements of robotic systems, combining vision-based pose estimation with control algorithms. They play a crucial role in applications such as robotic grasping and object manipulation.

Challenges often arise due to limited data availability for specific tasks, requiring careful consideration during model development.

3.1.3 Feature Learning

Even though the above described methods have demonstrated some success, they have not achieved similar accuracy as geometry-based solutions. Indeed, recent work [25] suggests that *"absolute pose regression approaches are more closely related to approximate pose estimation via image retrieval"*, thus they may not generalise well in practise.

While the keypoint matching problem can be solved with machine learning, feature learning methods based on deep convolutional neural networks (CNNs) typically fix 2D-3D keypoints associations and learn to predict the image locations of each corresponding 3D landmark. Examples include studies such as [23], [22] and [31], which mainly differ in model architecture and the choice of keypoints. For instance, [22] employs semantic keypoints, while [31] opts for vertices of the 3D bounding box.

In this spaceborne scenario, objects are typically not occluded and have relatively rich texture. As a result, it was opted for object surface landmarks in order to better relate them to strong visual features.

A notable commonality among these CNN-based methods is their shared characteristic of gradually transforming feature maps from high-resolution representations to low-resolution ones, then recovering them to high-resolution representations later in the process. Recent research emphasizes the significance of maintaining a high-resolution representation throughout tasks like object detection and human pose estimation [28][29]. Specifically, the High-Resolution Net (HRNet) [28], illustrated in figure **3.1**, upholds a high-resolution representation while exchanging information across parallel multi-resolution subnetworks, yielding superior spatial precision in landmark heatmaps. In the implemented satellite pose estimation framework, HRNet is leveraged to predict the locations of 2D landmarks in each image, contributing to achieving state-of-the-art accuracy.

HRNet

HRNet is a CNN with parallel high-to-low resolution subnetworks with repeated information exchange across multi-resolution subnetworks (multi-scale function) implemented for human pose estimation, aiming to detect the locations of K keypoints or parts of the human body from an image.

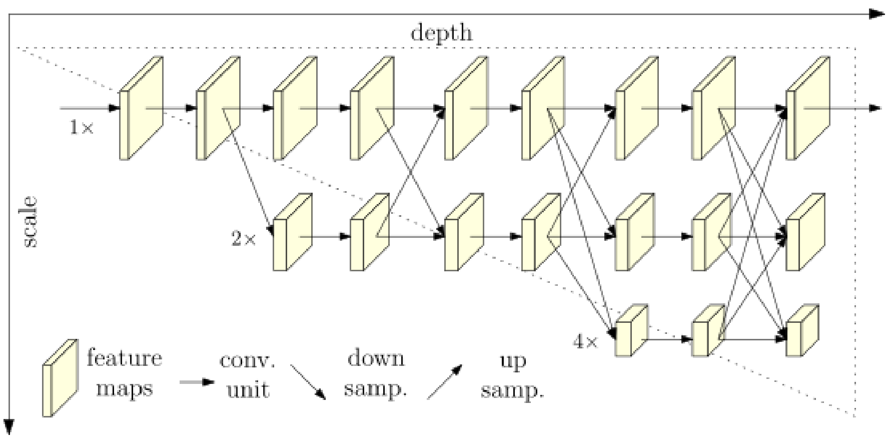

FIGURE 3.1: HRNet Architecture. The horizontal and vertical directions correspond to the depth of the network and the scale of the feature maps, respectively.

The architecture consists in two strided convolutions decreasing the resolution, a main body outputting the feature maps with the same resolution as its input feature maps, and a regressor estimating the heatmaps where the keypoints are chosen and transformed to the full resolution, see figure **3.1**.

Existing pose estimation networks are typically constructed by connecting subnetworks of varying resolutions in a sequential manner. Each subnetwork, representing a stage, comprises a series of convolutional layers, and there is a down-sampling layer between adjacent subnetworks to reduce the resolution by half. Let's denote N_{sr} as the subnetwork in the sth stage, with r indicating the resolution index (where its resolution is $\frac{1}{2^{(r-1)}}$ times lower than that of the first subnetwork). In a high-to-low network with S stages, such as 4 stages, it can be represented as:

$$N_{11} \to N_{22} \to N_{33} \to N_{44}. \tag{3.1}$$

HRNet proposes a parallel multi-resolution subnetwork approach, starting with a high-resolution subnetwork as the initial stage. Subsequently, high-to-low resolution subnetworks are gradually added one by one, creating new stages, and connecting these multi-resolution subnetworks in parallel. As a result, the resolutions for the parallel subnetworks in a later stage consist of the resolutions from the previous stage, along with an extra lower one.
An example network structure (with 4 parallel subnetworks) is gibven as follows:

$$\begin{tikzcd}
N_{11} \arrow[r] \arrow[dr] & N_{21} \arrow[r] & N_{31} \arrow[r] & N_{41} \\
 & N_{22} \arrow[r] \arrow[dr] & N_{32} \arrow[r] & N_{42} \\
 & & N_{33} \arrow[r] \arrow[dr] & N_{43} \\
 & & & N_{44}
\end{tikzcd} \tag{3.2}$$

Exchange units are introduced across parallel subnetworks, enabling each subnetwork to repeatedly receive information from other parallel subnetworks. An example illustrates this information exchange scheme. In this example, the third stage is divided into several (e.g., 3) exchange blocks, with each block composed of three parallel convolution units. These parallel units are connected through exchange units, as shown below:

$$\begin{tikzcd}
C_{131} \arrow[dr] & & C_{231} \arrow[dr] & & C_{331} \arrow[dr] & \\
C_{132} \arrow[r] & E_{13} \arrow[r] \arrow[ur] \arrow[dr] & C_{232} \arrow[r] & E_{23} \arrow[r] \arrow[ur] \arrow[dr] & C_{332} \arrow[r] & E_{33} \\
C_{133} \arrow[ur] & & C_{233} \arrow[ur] & & C_{333} \arrow[ur] &
\end{tikzcd} \tag{3.3}$$

Here, C_{bsr} represents the convolution unit in the r-th resolution of the b-th block in the s-th stage, and E_{bs} is the corresponding exchange unit.

The exchange unit aggregates the inputs, which are s response maps: $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_s\}$, to produce s response maps as outputs: $\{\mathbf{Y}_1, \mathbf{Y}_2, ..., \mathbf{Y}_s\}$, where the resolutions and widths remain consistent with the inputs. Additionally, an extra output map $\mathbf{Y}_{s+1}$ is generated by the exchange unit across stages: $\mathbf{Y}_{s+1} = a(\mathbf{Y}_s, s+1)$.

The function $a(\mathbf{X}_i, k)$ in this context includes upsampling or downsampling operations on $\mathbf{X}_i$ from resolution i to resolution k. Downsampling is achieved through strided 3×3 convolutions, such as a single 3×3 convolution with a stride of 2 for 2× downsampling, or two consecutive strided 3×3 convolutions with a stride of 2 for 4× downsampling. For upsampling, nearest neighbor sampling is followed by a 1 × 1 convolution to align the number of channels. When $i = k, a(\cdot, \cdot)$ represents an identity connection, and $a(\mathbf{X}_i, k)$ is equivalent to $\mathbf{X}_i$.

In the implemented method, the original architecture has been slightly modified in order to match the specifics of the analyzed case (grayscale 512x512 images).

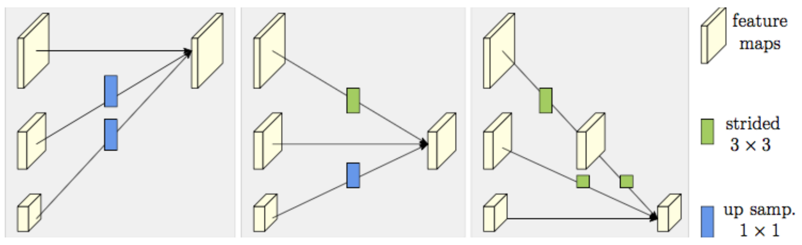

FIGURE 3.2: Exchange unit. Right legend: strided 3 × 3 = strided 3 × 3 convolution, up samp. 1×1= nearest neighbor up-sampling following a 1×1 convolution.

3.1.4 Spacecraft pose estimation

Monocular spacecraft pose estimation techniques typically embrace a model-based strategy. For instance, in studies like [5] and [27], the initial step involves image preprocessing and the utilization of feature detectors to identify salient features like line segments and basic geometric shapes. Subsequently, search algorithms are deployed to establish appropriate matches between the detected features and the 3D structure. Poses are then computed using Perspective-n-Point (PnP) solvers such as EPnP [16],

and refinement is carried out through optimization techniques.
As outlined in section **4**, the implemented approach also generates 2D-3D correspondences; however, along with the coordinates of 2D landmarks, they are regressed using a trained deep network.

The Spacecraft Pose Network (SPN) [26] represents a crucial contribution to the spacecraft project estimation problem using deep learning methods. SPN employs a hybrid of classification and regression neural networks for solving the pose estimation problem. Initially, it predicts the bounding box of the satellite in the image using an object detection sub-network. Subsequently, a classification sub-network retrieves the n most relevant base rotations from the feature map of the detected object. The regression sub-network learns a set of weights and produces the predicted rotation as a weighted average of the n base rotations. Finally, SPN determines the relative translation of the satellite by leveraging constraints derived from the predicted bounding box and rotation (for more comprehensive overview of spacecraft pose estimation, please read [3]).

3.1.5 Point Set Alignments

Iterative Closest Point (ICP)

Iterative Closest Point (ICP) [1] is a popular algorithm used for aligning two sets of 3D points through an iterative optimization process. It is commonly employed in scenarios where a reference 3D model needs to be aligned with a partially observed or reconstructed 3D scene. ICP iteratively refines the transformation between the two point sets to minimize the distance between corresponding points.

An initial transformation (translation and rotation) between the two point sets is estimated. the correspondences between the points in the reference and observed sets are established based on proximity, a weighted least-squares approach is used to minimize the distance between corresponding points, giving higher importance to reliable matches and the transformation is updated based on the registration results. The process is repeated iteratively until convergence, with a check for the change in transformation parameters.

Coherent Point Drift (CPD)

Coherent Point Drift (CPD) [20] is a sophisticated mathematical technique used in the field of point cloud registration. The primary goal of CPD is to align one point cloud with another, ensuring that they match as closely as possible. This alignment is particularly useful when working with objects or scenes that undergo only rigid transformations, meaning that they can be translated and rotated without deformation.

CPD takes a unique approach by treating the point clouds as probability distributions. Each point in the source and target clouds is associated with a probability density. In essence, the source points represent samples from one distribution, and the target points represent samples from another. This probabilistic perspective allows CPD to find optimal correspondences between points in the source and target clouds.

Given two point sets $X = \{x_1, x_2, \ldots, x_N\}$ and $Y = \{y_1, y_2, \ldots, y_M\}$ and initial point correspondences unknown, the estimation of the transformation is achieved through an optimization process with the objective of finding the optimal rigid transformation parameters to align X with Y.

Mathematical Steps:

- **Initialization:** the transformation parameters: R (rotation matrix) and t (translation vector) are initialized.
- **Expectation-Maximization (EM) Iterations:**
  - **E-step (Expectation Step):** computes the soft correspondences between points in X and Y using a Gaussian Mixture Model (GMM) with only rigid transformations. The probability of correspondence P_{ij} between x_i and y_j is calculated based on the distance between transformed x_i and y_j using the current R and t.

$$P_{ij} = \frac{\exp\left(-\frac{1}{2\sigma^2}\|R \cdot x_i + t - y_j\|^2\right)}{\sum_{k=1}^{M} \exp\left(-\frac{1}{2\sigma^2}\|R \cdot x_i + t - y_k\|^2\right)} \tag{3.4}$$

  - **M-step (Maximization Step):** Updates the rotation matrix R and translation vector t based on the soft correspondences. It uses a closed-form solution to compute the optimal rotation and translation.

$$R, t = \arg\min_{R,t} \sum_{i=1}^{N} \sum_{j=1}^{M} P_{ij} \|R \cdot x_i + t - y_j\|^2 \tag{3.5}$$

- **Convergence Check:** It checks for convergence by examining the change in the transformation parameters between consecutive iterations. If the parameters converge or a maximum number of iterations is reached, the algorithm terminates.
- **Output:** The final rotation matrix R and translation vector t represent the rigid transformation aligning X with Y.

The Gaussian Mixture Model (GMM) is used to assign soft correspondences, allowing the algorithm to handle cases where points in one set do not have clear one-to-one correspondences with points in the other set. The algorithm iteratively refines the transformation parameters until convergence.

As outlined in section **4.2.2**, this approach is used in the implemented method to refine the final pose by the predicted 3D landmarks position.

Chapter 4

Algorithms and methods

4.1 Offline Architecture

Figure **4.1** describes the overall offline pipeline of the implemented methodology, which consists of several main modules.
From the satellite CAD model nine landmarks have been manually selected (details about the selection criteria explained in section **4.3.1**). The 2D-3D correspondence of each landmark is used for training the *Landmark regression* module to predict the nine landmarks image position and the *Landmark Mapping* module to reconstruct the 3D position of each landmark from their image position. The implemented code can be accessed in [19].

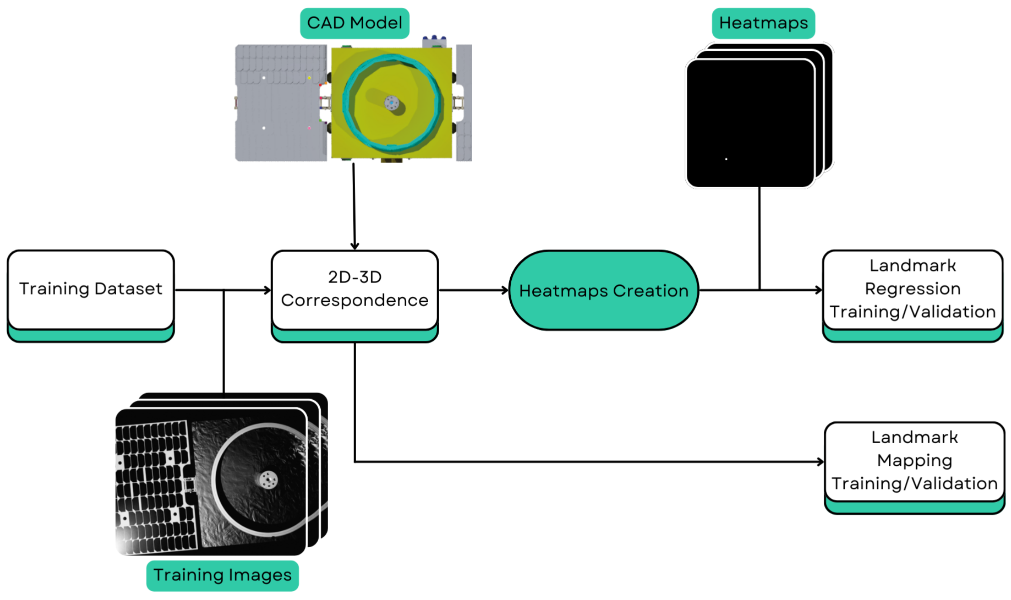

FIGURE 4.1: Offline pipeline of the implemented pose estimator.

4.1.1 2D-3D Correspondence

The computation of the 3D position of landmarks in the training images involves a transformation process that leverages the relative positions of the selected landmarks with respect to the satellite's reference system, as well as the position and orientation of the camera.

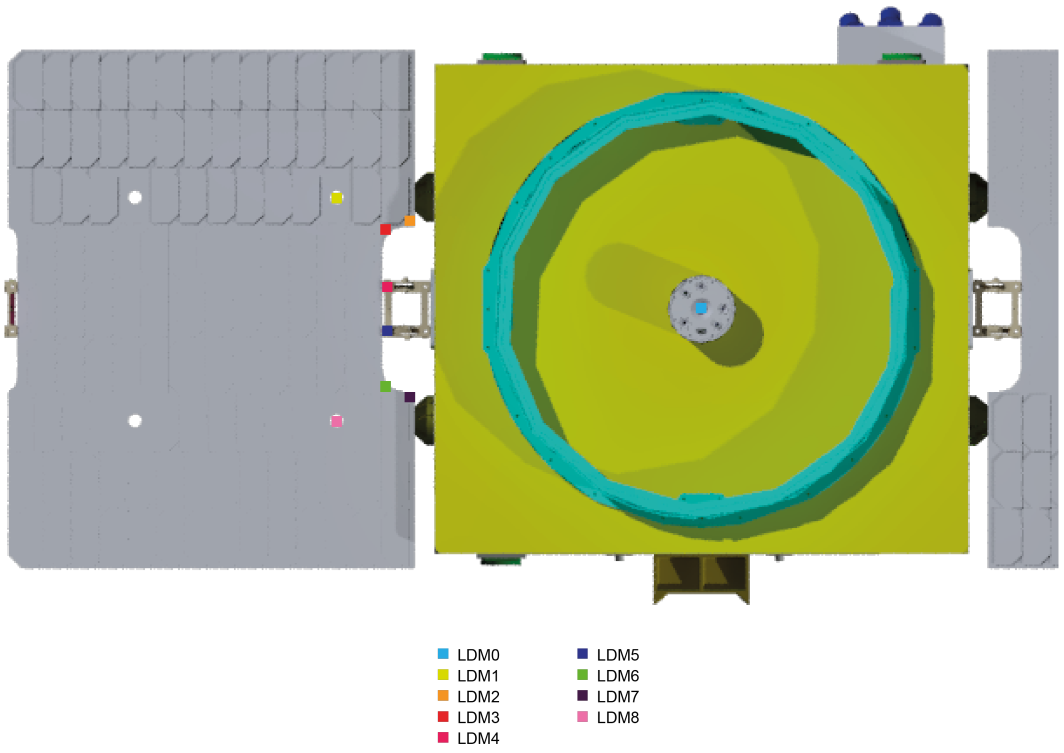

FIGURE 4.2: Selected Landmarks on the CAD Model.

The 3D landmarks are defined in the context of the satellite's reference system. These landmarks represent specific features on the satellite's structure. Their 3D coordinates are known relative to the satellite's own coordinate system thanks to the CAD model.

The camera that captures the training images is mounted on the wrist of a robotic arm on the servicer satellite and its position and orientation with respect to the satellite's reference system is paired with images in the training dataset.

To determine the 3D position of the landmarks in the camera's coordinate system, a simple operation is performed, transforming the landmarks' positions from the satellite's frame to the camera's frame by means of the systems' transformation matrix.

$$x_{cam,i} = \begin{bmatrix} \mathbf{R}^{\mathbf{T}} & t_{c,s} \\ 0 & 1 \end{bmatrix} x_{sat,i} \quad i = 1, ..., N \quad N : \text{number of landmarks} \tag{4.1}$$

Furthermore, perspective projection transformation (described in section **2.2**) is performed, relating 3D world coordinates $\{x_i\}$ to 2D image coordinates $\{z_i\}$ by means of the perspective matrix $\{\mathbf{P}\}$.

$$z_i = \mathbf{P} * x_{cam,i} \quad i = 1, ..., N \tag{4.2}$$

4.1.2 Landmark Regression

Each training image is first pre-processed (description of the pre-processing process in section **4.2.1**) and then coupled with a set of ground truth 2D landmarks $\{\mathbf{z}_i\}$, as described in section **4.1.1**. Those labels are used to supervise the training of the regression model to predict the 2D position of landmarks in the testing images. An additional label is introduced to handle images that capture only partially the satellite, the visibility coefficient $\{v_i\}$.

$$v_i = \begin{cases} 1, & \text{if } \mathbf{z}_i \text{ is inside image frame.} \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

The output of the model is a tensor of 9 heatmaps, one for each landmark $h(\mathbf{z}_i^p)$. The ground truth heatmaps $h(\mathbf{z}_i)$ are generated as 2D normal distributions with mean equal to the ground truth location of each landmark, and standard deviation of 1-pixel (code in appendix **A.1**).

The model is trained from scratch by minimizing the following customized loss:

$$l = \frac{1}{N} \sum_{i=1}^{N} v_i (h(\mathbf{z}_i^p) - h(\mathbf{z}_i))^2 \tag{4.4}$$

The loss function l is defined on a single image and in a mini batch l is simply averaged. The model is trained in 100 epochs with the *Adam optimizer*[13].

In order to select the landmark location in the predicted heatmap, an additional selection algorithm is introduced.
Each predicted heatmap is normalized over [-1,+1] range, where -1 represents a totally black pixel and +1 totally white. After an accurate study on the heatmaps that presents an accurately predicted landmark, the main result is that most of those images presented maximum value > -0.5, so a first threshold t_1 is introduced and set to that value. In only few cases the prediction of the landmark is more imprecise, due, for instance, to similarities with other features in the image. For those cases a second lower threshold is included $t_2 = -0.6$.

Firstly pixels in the images with value lower than t_2 are by default set to black (-1) to clear the image from impurities, then, for those heatmaps with maximum value over t_1, the position of their maximum value is taken as landmark position. On the other hand, for those who present a maximum value within the thresholds t_1 and t_2 an additional check on the heatmap variance is performed: only for those with a variance value over $2e^{-6}$ the landmark is considered within the image and its location is registered as above. For all the other heatmaps that do not follow these conditions, the landmark is considered out of the image frame (2D location set to default value [-1,-1]) and the visibility coefficient v_i is set to zero.

In case of visually similar landmarks, an additional check is introduced. The position of a landmark registered after recognition is compared with the previously registered landmarks. In case it has a position within a certain pixel range near to another landmark, the one with higher maximum value and variance is registered and the other is considered out of frame (code in appendix **A.2**). Figure **4.3** shows the algorithm's identification process.

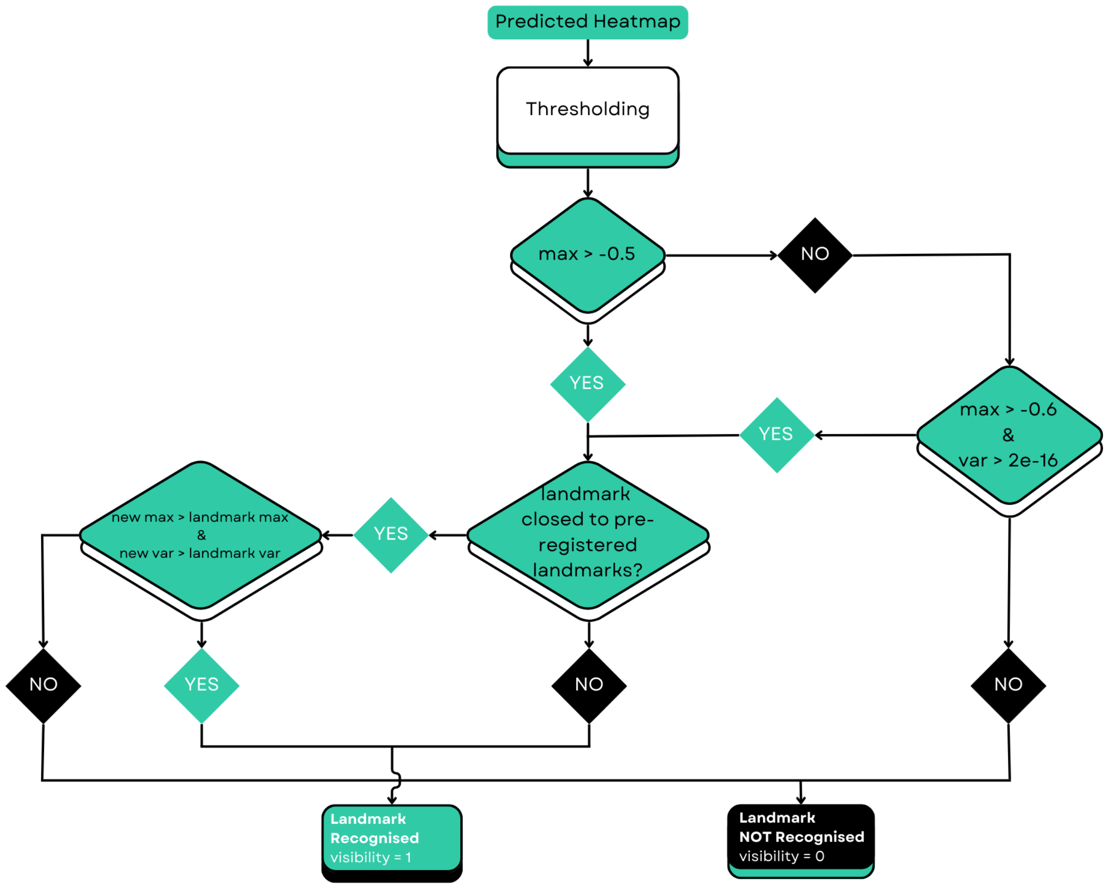

FIGURE 4.3: Landmark Identification Algorithm.

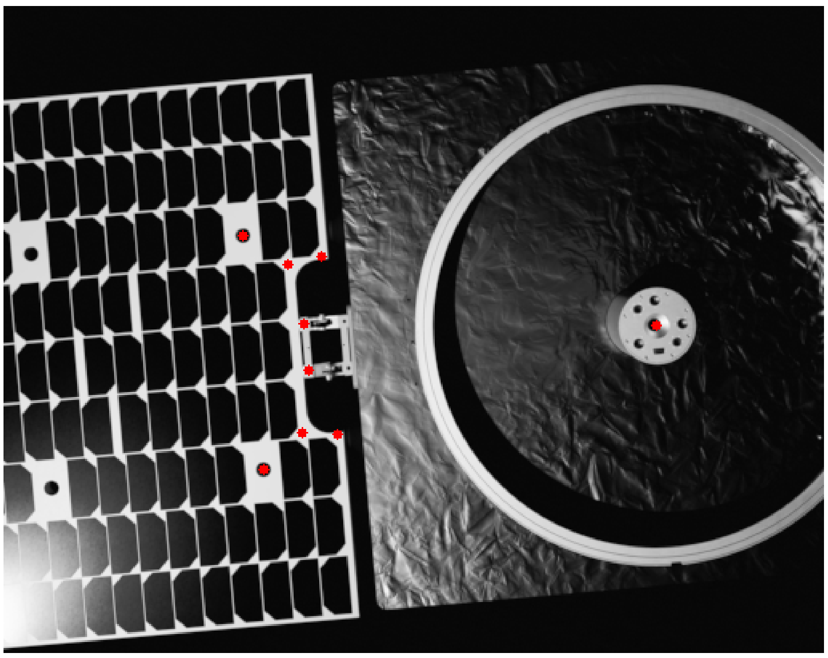

FIGURE 4.4: Landmarks identified by the Landmark Regression module.

4.1.3 Landmark Mapping

The Landmark Mapping is a neural network designed to estimate the 3D positions of landmarks using their 2D positions, mapping the 2D-3D relation.
The network takes as input the 2D positions of landmarks obtained from the Landmark Regression module (**4.1.2**) and predicts as output the respective 3D position.

The network has two hidden layers:

- The first hidden layer consists of 128 units. It processes the input data and learns complex patterns and relationships between the 2D and 3D coordinates.
- The second hidden layer has 64 units and further refines the features learned in the previous layer.

ReLU activation functions are applied after the first and second hidden layers. ReLU introduces non-linearity and helps the network capture complex relationships in the data.
The output layer is responsible for regressing the 3D positions of the landmarks. Each landmark is represented by a 3D coordinate (x, y, z). The output layer produces these 3D coordinates for all the landmarks.
The output of the network is reshaped to organize the predicted 3D coordinates for each landmark. This reshaping ensures that the output is in a format suitable for further processing.
The model is trained with the 2D-3D correspondences described in section **4.1.1**, but, before training, the 3D ground truth of landmarks out of image frame ($v_i = 0$) is set to default value $[0, 0, 0]$.
The model is trained from scratch by minimizing the following loss:

$$l = \frac{1}{N}\sum_{i=1}^{N} v_i(\mathbf{x}_i^p - \mathbf{x}_i)^2 \tag{4.5}$$

In the above equation the $\mathbf{x}_i^p$ represents the 3D position of the i-th landmark predicted by the model, while $\mathbf{x}_i$ is its ground truth position.
As in the previous model, the loss function l is defined on a single group of landmarks and in a mini batch it is simply averaged. The model is trained in a maximum of 150 batches with a *Adam optimizer*[13].

In order to improve the accuracy of the algorithm, three models are created, each of them specialized in a specific range of distance from the target. The training dataset is so split in three subdatasets depending on the distance on the z axis from the satellite with a 100 images superposition per model from each trajectory. The table below shows the specifications of each model.

Model Name	Covered Range (m)
M1	2.00-1.20
M2	1.40-0.50
M3	0.70-0.40

TABLE 4.1: Specifics of three models *M1*, *M2* and *M3*.

In order to prevent overfitting, an *Early Stopping* algorithm [9] has been introduced. Overfitting occurs when a neural network becomes too specialized in its understanding of the training data, to the point that it struggles to generalize to new, unseen data. The idea behind early stopping is to maintain under control the network's performance, particularly on a separate dataset called validation set, as it undergoes training. During the training process, the neural network works on refining its parameters using the training dataset. But instead of letting it train tirelessly until convergence, the training process is periodically paused and its performance are evaluated on the validation set. If, over a certain number of consecutive evaluations (determined by a parameter known as "patience"), the network's performance starts to worsen, it's an early warning sign. It suggest that the model is becoming too specialized. Once the point where the validation performance is consistently declining, the training phase is stopped. At this moment, the neural network is considered to have reached its optimal performance on unseen data. Its parameters are taken as the final model, capable to generalize to new unseen data.

4.2 Online Architecture

The online architecture operates in real-time. It processes the input data from the camera and produces as output the satellite pose estimation.

After being pre-processed, the image captured from the camera is passed to the *Landmark regression* module. The latter predicts the 2D location $\{\mathbf{z}_i\}$ of the 9 landmarks along with the visibility coefficient $\{v_i\}$ for each of them. The *Landmark Mapping* module then uses these data to compute the 3D position of each landmark with respect to the camera frame. The final 6DOF pose of the satellite is then computed from the CPD module. Figure **4.5** shows the online pipeline of the implemented methodology.

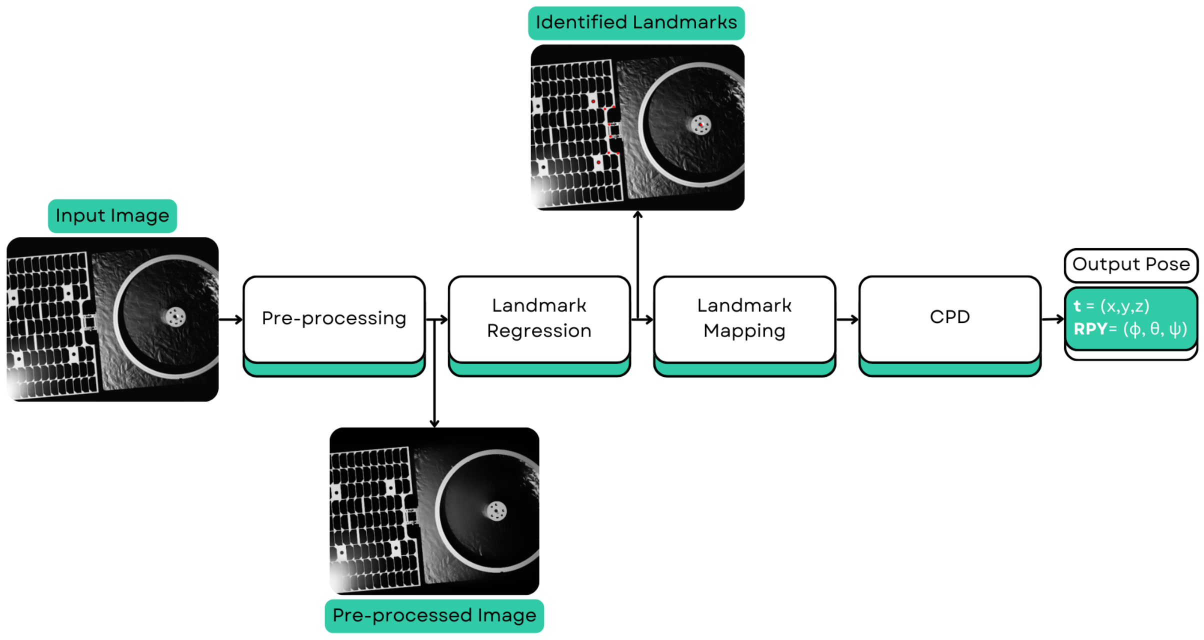

FIGURE 4.5: Online pipeline of the implemented pose estimator.

4.2.1 Pre-Processing

The image captured from the camera is pre-processed in the *Pre-Processing* module. It consists in a bilateral filter, which is a non-linear, edge-preserving smoothing filter that reduces noise while preserving the edges in an image.
The mathematical steps behind the bilateral filter involve computing weighted averages of pixel values within a local neighborhood.

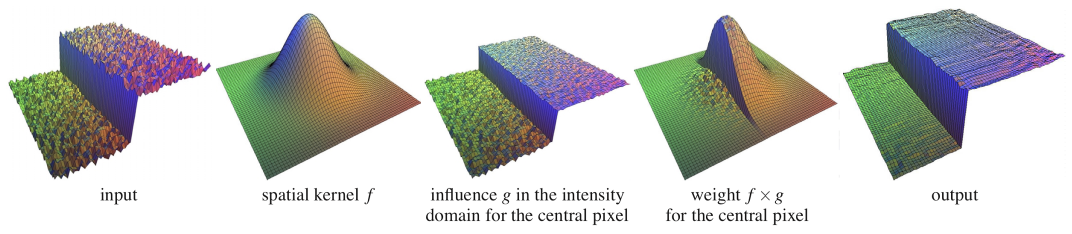
input
spatial kernel f
influence g in the intensity domain for the central pixel
weight $f \times g$ for the central pixel
output

FIGURE 4.6: Bilateral Filter mathematical steps.

The formula for the bilateral filter operation can be described as follows:

Given an input image $I(x, y)$ and the filter parameters:

- d (diameter of each pixel's neighborhood)
- σ_c (standard deviation of the color space)
- σ_s (standard deviation of the spatial space)

The filtered output image $B(x, y)$ can be computed using the following equation for each pixel (x, y):

$$B(x,y) = \frac{1}{W(x,y)} \sum_{(i,j)\in S} I(i,j) \cdot G_c(I(x,y), I(i,j), \sigma_c) \cdot G_s(x,y,i,j,\sigma_s) \tag{4.6}$$

Where:

- S is the neighborhood of pixel (x, y), defined by a window of diameter d.
- G_c is the color similarity term, which measures the similarity in color between pixels (x, y) and (i, j) in the color space.
- G_s is the spatial similarity term, which measures the spatial proximity between pixels (x, y) and (i, j).
- $W(x, y)$ is a normalization factor.

The G_c term is defined as a Gaussian function in the color space:

$$G_c(I(x,y), I(i,j), \sigma_c) = \exp\left(-\frac{\|I(x,y) - I(i,j)\|^2}{2\sigma_c^2}\right) \tag{4.7}$$

The G_s term is a Gaussian function in the spatial space:

$$G_s(x,y,i,j,\sigma_s) = \exp\left(-\frac{\|(x,y) - (i,j)\|^2}{2\sigma_s^2}\right) \tag{4.8}$$

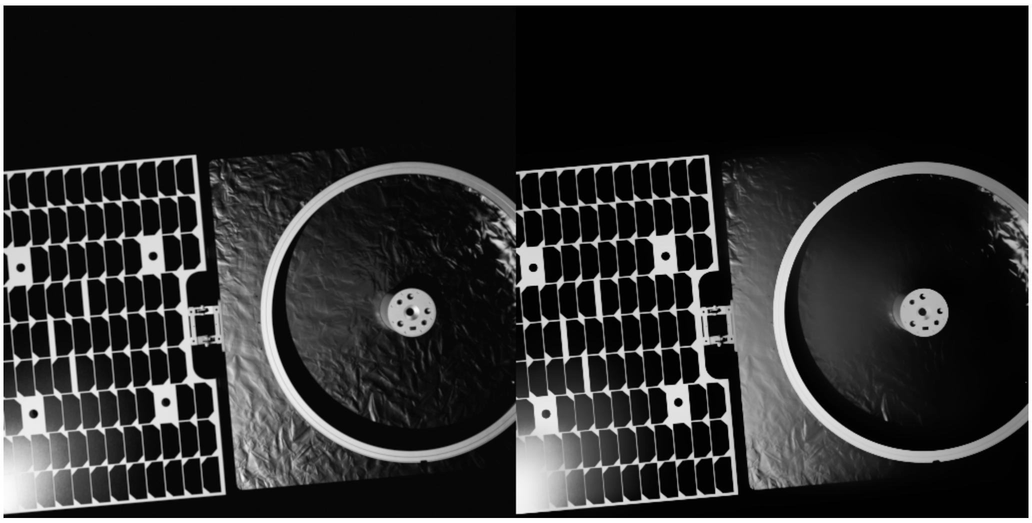
FIGURE 4.7: On the left the original sample image from the training dataset, on the right the resultant image after the application of the bilaterial filter ($d = 40, \sigma_c = 40, \sigma_s = 200$).

The bilateral filter operates by applying these Gaussian weighting functions to compute the weighted average of pixel values within the defined neighborhood, both in color and spatial spaces, resulting in a smoothed image that preserves edges and details. The filter helps reduce noise while keeping the important structures in the image intact.

4.2.2 Coherent Point Drift (CPD)

As the *Landmark Mapping* module predicts the position of each landmark independently one from the other and so each landmark has its own error over the three axis, the resultant cloud of points is misaligned with respect to the rigid reference position given by the CAD model.
In order to estimate the 6DOF pose of the satellite and in the meantime overcome this misalignment problem a mathematical technique is used: Coherent Point Drift.

Two sets of 3D points are present: one is the "target", which represents the 3D position of the landmarks in the camera frame supposing no translation and no rotation, and the other is "source", which is the 3D position of the predicted landmarks. The "source" point set is only composed by landmarks identified in the image frame ($v_i = 1$). It is important to know that the algorithm's performances strongly depend on the number of points in the set so, with the camera approaching the target, some landmarks are cut from the image frame and consequently the pose estimation accuracy reduces.

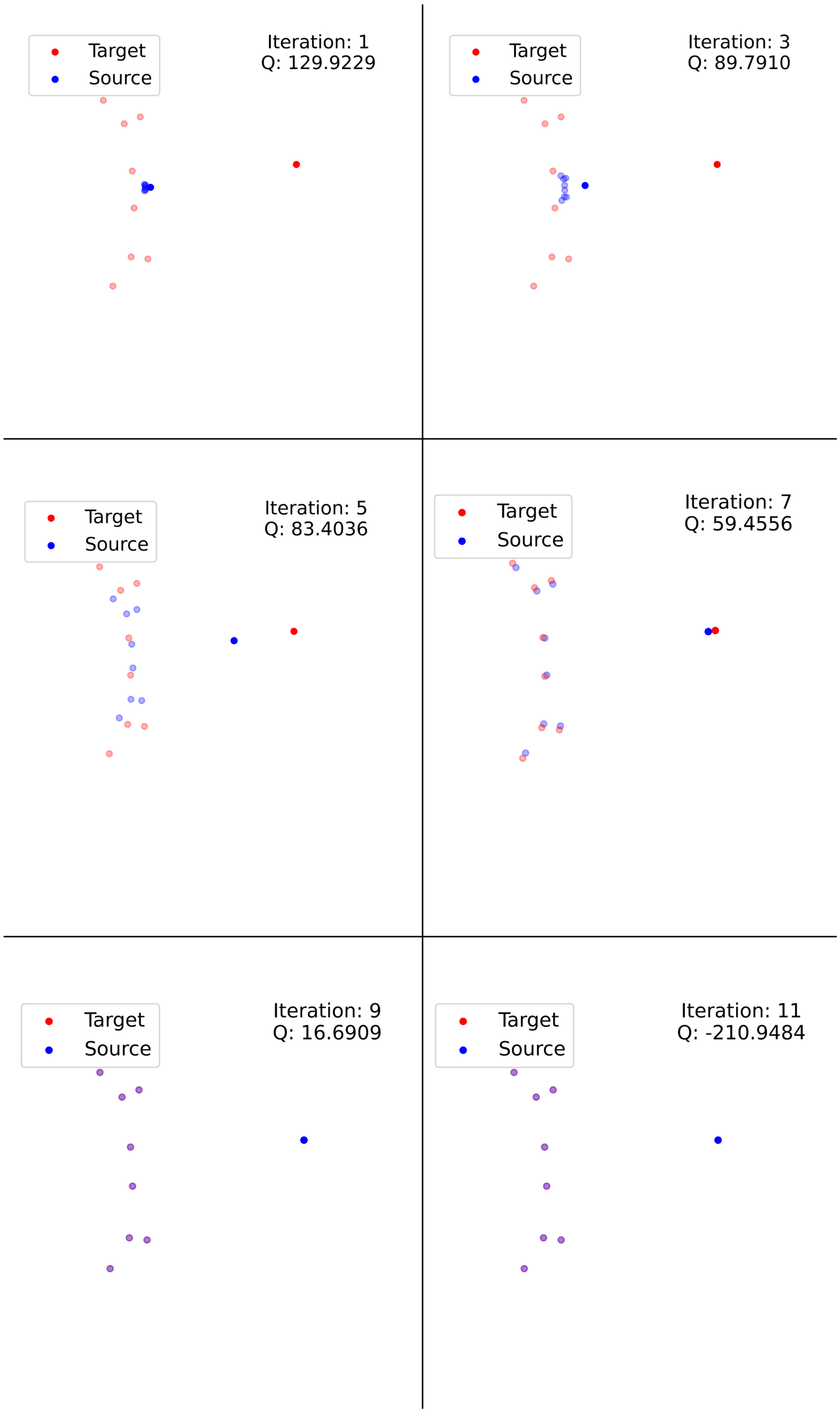

FIGURE 4.8: Iterations of the CPD optimization process.

4.3 Implementation problems and Technical choices

4.3.1 Landmarks Selection

The main problem in the implementation of the *Landmark regression* module (**4.1.2**) is the selection of the landmarks. Selecting meaningful landmarks is a critical first step, requiring a accurate understanding of the satellite's structure. These landmarks must possess distinct characteristics that remain invariant under varying conditions, such as changes in lighting, orientation, or potential occlusions.

The first performed attempt in the selection of the landmarks was composed of 11 landmarks with relevant features in the satellite's structure. Most of those visual features were similar to each others and the resultant heatmap predicted by the CNN for a single landmark was ambiguous among multiple ones. This led to a complicated recognition of the landmark location in the image with complex and heavier algorithms for the 2D position identification.

In both cases the set of landmarks has been selected near the approach target to limit the number of out of frame landmark in closer positions.

4.3.2 Dataset availability

Another notable challenge stems from the limited size of available datasets. A smaller dataset poses a risk of overfitting, potentially hindering the model's ability to generalize across diverse scenarios. Addressing this issue requires careful consideration of data augmentation techniques, introducing various transformations to enhance the model's adaptability.

The dataset used for training present five different orientations on each axis of the satellite during the whole approaching range. The main limitation given by the used dataset is the lack of combined rotations over multiple axis and a wider range of rotation on single axis.

Even though the *Landmark regression* module training is strictly related to the availability of the images, the *Landmarks Mapping* one is totally independent. The 2D-3D correspondence of landmarks used to train the model requires the only relative position of the landmarks from the CAD Model and the perspective matrix to be performed. This means that a possible further step to expand the dataset is to perform several simulations with rotations over multiple axis to create new wider datasets and improve the model performances.

Chapter 5

Implementation and Experiments

5.1 Tools and Technologies

In conducting this research, careful consideration was given to the selection of tools and technologies that would best support the implementation, experimentation, and analysis processes. This section provides an overview of the programming languages, machine learning frameworks, data processing tools, and other technologies employed throughout the research.

5.1.1 Programming Language

The primary programming language for this research was Python, chosen for its versatility, rich ecosystem, and widespread use in machine learning. Python's readability and extensive libraries facilitated efficient coding and experimentation.

5.1.2 Machine Learning Frameworks

PyTorch

PyTorch [21] was selected as the primary deep learning framework due to its flexibility which aligned well with the nature of the tasks involving customized model architectures and strong community support which contributed to a fluid development experience.

PyTorch is an open-source machine learning framework developed by Facebook's AI Research lab (FAIR). Its design philosophy emphasizes flexibility, enabling researchers and practitioners to tailor datasets and training procedures to specific requirements. The flexibility offered by PyTorch manifests in two key areas: custom datasets and custom training/validation phases. PyTorch supports automatic differentiation, making it easier to implement and experiment with complex neural network architectures. Its community, extensive documentation, and seamless integration with hardware accelerators like GPUs contribute to its popularity in both research and production.

PyTorch facilitates the creation of the custom dataset through the *torch.utils.data. Dataset* class.
By inheriting from this class and implementing the *__len__* and *__getitem__* methods, it's been possible to define datasets tailored to data structure and format used. This capability is invaluable when working with diverse data types, such as images or time-series, allowing seamless integration into PyTorch's data loading utilities.

PyTorch's flexibility extends to the training and validation phases, enabling users to define custom training loops, loss functions, and evaluation metrics. This is crucial for experimenting with novel architectures, incorporating domain-specific metrics, or implementing advanced training techniques. The ability to seamlessly integrate custom logic into the training process empowers researchers to push the boundaries of model development.

The deep understanding of this framework and its functionalities has been of primary importance for using third-party models like *HRNet*[28] integrating with a customized dataset and ad hoc training and validation phase.

5.1.3 Data Processing and Analysis Tools

Pandas

Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames that facilitate the handling of structured data. Pandas excels in data cleaning, manipulation, and exploration tasks, offering a plurality of functions for indexing, merging, grouping, and aggregating data. Its integration with other libraries, such as NumPy, makes it a go-to choice for working with labeled data and time series.

Pandas DataFrames offer a convenient and versatile way to handle tabular data, making them an excellent choice for storing and preprocessing data before creating datasets in PyTorch. The integration between Pandas and PyTorch simplifies the transition from data exploration to model training.

NumPy

NumPy is a fundamental library for numerical operations in Python. It provides support for large, multi-dimensional arrays and matrices, along with an assortment of high-level mathematical functions to operate on these arrays.

Moreover, NumPy arrays and PyTorch tensors share several similarities, making them interchangeable in many contexts. These similarities contribute to a smooth integration between the two libraries, facilitating data manipulation and interoperability.
The compatibility between NumPy and PyTorch simplifies data exchange and promotes a cohesive workflow in mixed-library environments.

5.1.4 Visualization Libraries

Matplotlib and Seaborn

Matplotlib is a versatile 2D plotting library for Python. Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of generating complex visualizations with concise syntax. It is particularly useful for exploring relationships in datasets through specialized plots for categorical data, distribution plots, and regression plots.
All the plots presented in the *Evaluation of Training Dataset* (**5.2.2**) and in the *Evaluation of Test Datasets* (**5.2.3**) sections are created using this library.

5.2 Experimentation and Evaluation

5.2.1 Metrics

The final pose estimation scores are defined according to [14] score definition, which consists in the identification of translation and rotational errors and their respective scores.
The estimated pose of each image is evaluated using a rotation error E_R and a translation error E_T. Lets consider q^* the rotation quaternion ground truth of an image and q its estimation and, analogously, $\mathbf{t}^*$ the ground truth translation of an image and $\mathbf{t}$ the respective estimation.
The orientation error E_R is calculated as the angular distance between the predicted, q^* and ground truth true q unit quaternions, i.e., the magnitude of the rotation that aligns the target body frame with the camera reference frame.

$$E_R = 2cos^{-1}(|z|) \tag{5.1}$$

where z is the real part of the Hamilton product between q^* and the conjugate of q, i.e.: $z + c = q^* \cdot \bar{q}$, and c is the vector part of the Hamilton product. The translation errors is defined as the magnitude (2-norm) of difference between the ground-truth (**t***) and estimated (**t**) position vectors from the origin of the camera reference frame to that of the target body frame:

$$E_T = \|\mathbf{t}^* - \mathbf{t}\|_2 \tag{5.2}$$

The rotation score S_R is the same as E_R, but in radians, while the translation score S_T is defined as the translation error E_T normalized by the ground truth translation:

$$S_T = \frac{\|\mathbf{t}^* - \mathbf{t}\|_2}{\|\mathbf{t}^*\|_2} \tag{5.3}$$

which penalizes the position errors more heavily when the target satellite is closer.
The final score is the sum of contributions of both scores.

$$S = S_R + S_T \tag{5.4}$$

Additionally, the implemented method is subjected to iterative assessments after the execution of each module. This approach allows to discern and analyze the specific impact and weight that each individual module exerts on the final error. By dissecting the performance at each stage, we gain valuable insights into the contributions of each module, facilitating a comprehensive understanding of the overall system dynamics and optimization potential.

Landmark Regression Evaluation Metric
The *Landmark Regression* module is assessed computing the mean error values, among the set of landmarks identified in the image. The predicted pixel coordinates are compared with the ground truth ones.

$$E_{2D} = \frac{\sum_{i=1}^{N}(\mathbf{z}_i^* - \mathbf{z}_i)}{N} \quad \text{with } i = 1, \ldots, N \text{ landmarks with } v = 1 \tag{5.5}$$

where $\mathbf{z}_i^*$ is the i-th landmark's ground truth position, $\mathbf{z}_i$ is the respective prediction and v it is the visibility coefficient.

The overall module error is defined as the L2 norm of the computed mean error:

$$E_{CNN} = \|E_{2D}\|_2 \tag{5.6}$$

Landmark Mapping Evaluation Metric
The *Landmark Mapping* module is assessed similarly to the previous one, computing the L2 norm on the mean 3D position error of landmarks marked with visibility coefficient $v = 1$. The predicted 3D coordinates are compared with the ground truth ones.

$$E_{3D} = \frac{\sum_{i=1}^{N}(\mathbf{x}_i^* - \mathbf{x}_i)}{N} \quad \text{with } i = 1, \ldots, N \text{ landmarks with } v = 1 \tag{5.7}$$

where $\mathbf{x}_i^*$ is the ground truth position of the i-th landmark, $\mathbf{x}_i$, is the respective prediction and v it is the visibility coefficient.
The overall module error is defined as the L2 norm of the computed mean error:

$$E_{NN} = \|E_{3D}\|_2 \tag{5.8}$$

5.2.2 Evaluation on Training Dataset

Two distinct experiments are conducted to comprehensively evaluate the performance of the implemented method. In the first experiment, the method underwent rigorous testing on the training dataset, where it is exposed to known data. This initial experiment serves as a crucial phase for fine-tuning and optimizing the model's parameters.

Landmark Regression

As predictable, as far as the camera gets closer to the target region, the landmark location prediction becomes more granular and accurate.

It is important to notice that the performed experiment takes into account 40 cm as minimum distance from the target. This is due to an accurate results analysis which reports that in closer positions the number of landmarks in the camera frame is not sufficient to predict correctly the pose.

The *Landmark Regression* module reports an average 2D error $E_{CNN} = 1.38 \pm 0.06$ pixels over the sixteen training trajectories, corresponding to the 0.0027% considering $512x512$ images.

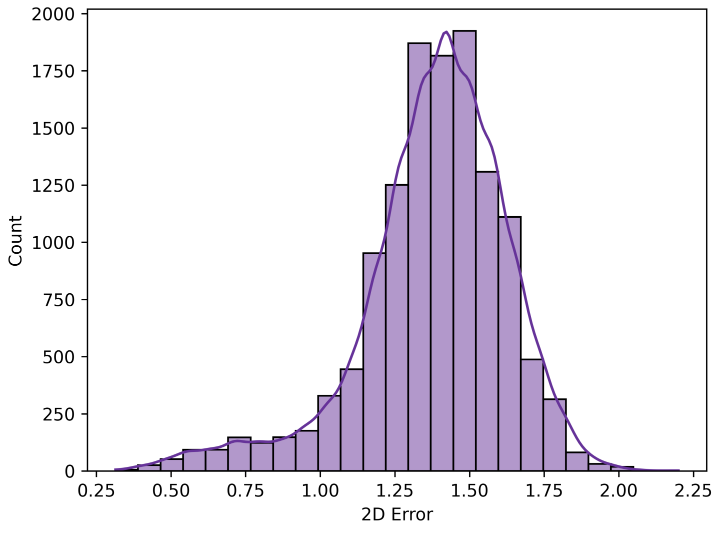

FIGURE 5.1: 2D error over the training dataset.

As it's possible to notice in figure **5.2**, the performances are quite the same for each training trajectory, due to the fact that the *Landmark Regression* module predictions are not affected by the orientation of the satellite if this not introduces occlusion. On the other hand the module's performances are strictly affected by the quality and light of the captured image.

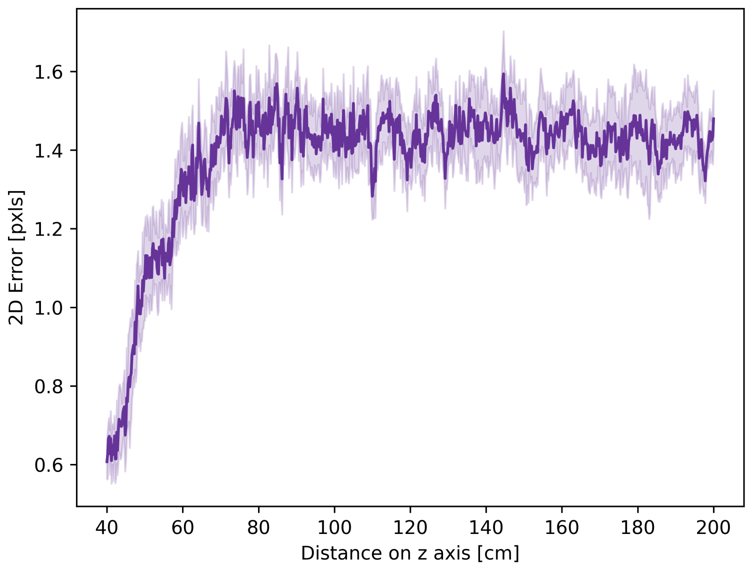

FIGURE 5.2: Landmark Regression error over the trajectory range (z axis) in pixels.

The following table report the mean 2D error and its variance for each trajectory in the train dataset. It's important to notice that the error grows along with the RPY rotations in the respective trajectories (see the trajectories specifics at **1.1**).

Trajectory	E_{CNN} [pixels]
TRAY_1	1.37 ± 0.05
TRAY_2	1.39 ± 0.05
TRAY_3	1.35 ± 0.06
TRAY_4	1.34 ± 0.06
TRAY_5	1.37 ± 0.07
TRAY_6	1.38 ± 0.07
TRAY_7	1.37 ± 0.06
TRAY_8	1.43 ± 0.06
TRAY_9	1.33 ± 0.06
TRAY_10	1.34 ± 0.06
TRAY_11	1.42 ± 0.07
TRAY_12	1.41 ± 0.05
TRAY_13	1.38 ± 0.05
TRAY_14	1.38 ± 0.05
TRAY_15	1.38 ± 0.05
TRAY_16	1.37 ± 0.05
Overall	1.38 ± 0.06

TABLE 5.1: Landmark Regression module results.

Landmark Mapping

The evaluation of the performances of *Landmark Mapping* module is performed on both multi-models configuration and single model configuration.

In the first case, each model is trained to cover a specific range of distances from the target. This strategic division aimed to capitalize on the unique strengths of each model within its designated proximity band. The switch between one model to the following one is performed as the predicted distance from the target overcome a threshold for five times.

The following table shows the used thresholds for the models switching:

Models transition	Threshold (m)
M1 $\rightarrow$ M2	1.30
M2 $\rightarrow$ M3	0.60

TABLE 5.2: Thresholds of the multi-model configuration.

The Landmark Mapping module takes the predicted 2D positions of landmarks as input, and as a result, its output is influenced by any residual errors propagated from the preceding module. Notably, the errors in predicting the 3D positions tend to be more pronounced for target locations that are farther from the camera. This happens because a pixel error in the 2D positions at greater distances corresponds to a relatively larger 3D error compared to positions that are closer to the camera.

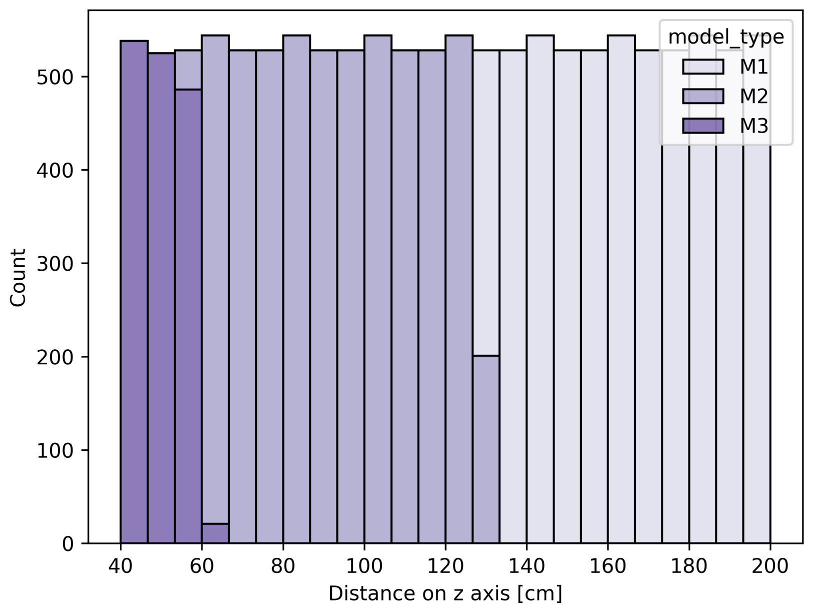

FIGURE 5.3: Models Distribution over the*z* axis

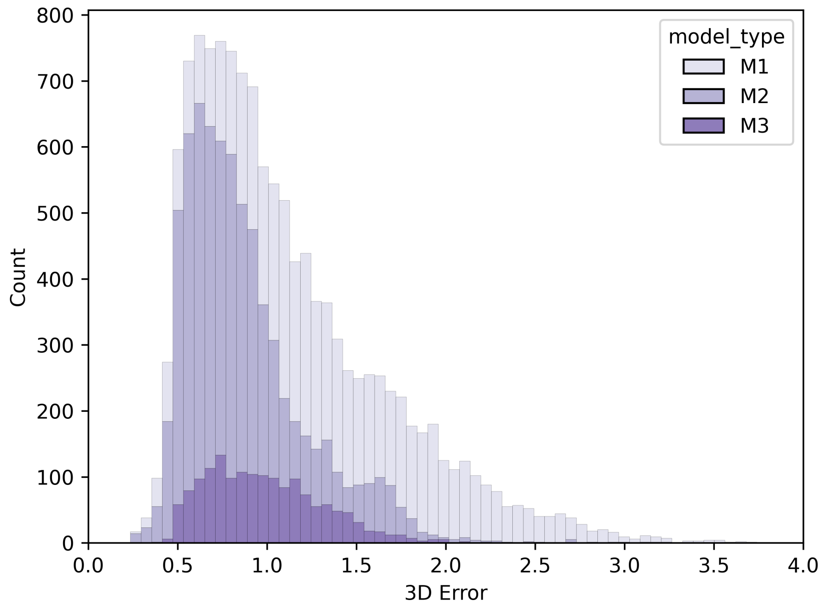

FIGURE 5.4: Landmark Mapping error for the different models of the multi-model configuration.

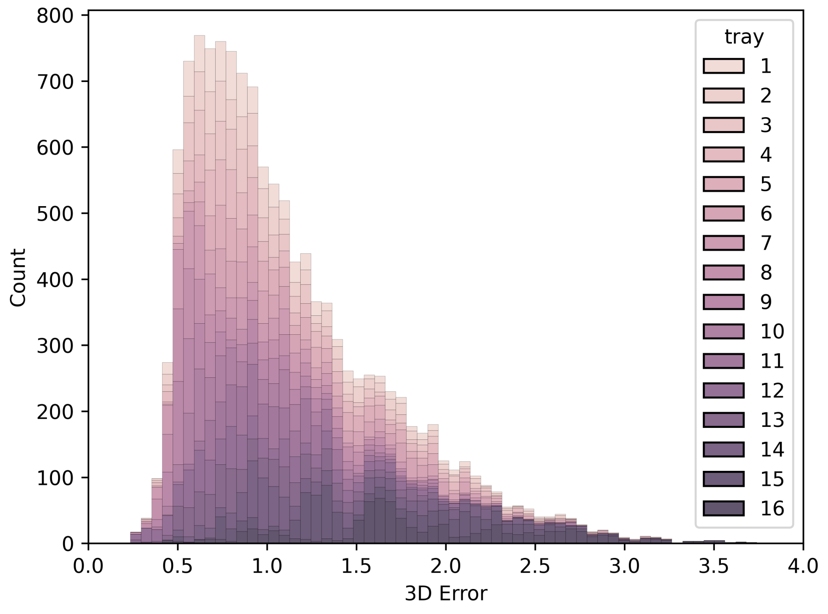

FIGURE 5.5: 3D error over the training dataset in the multi-model configuration.

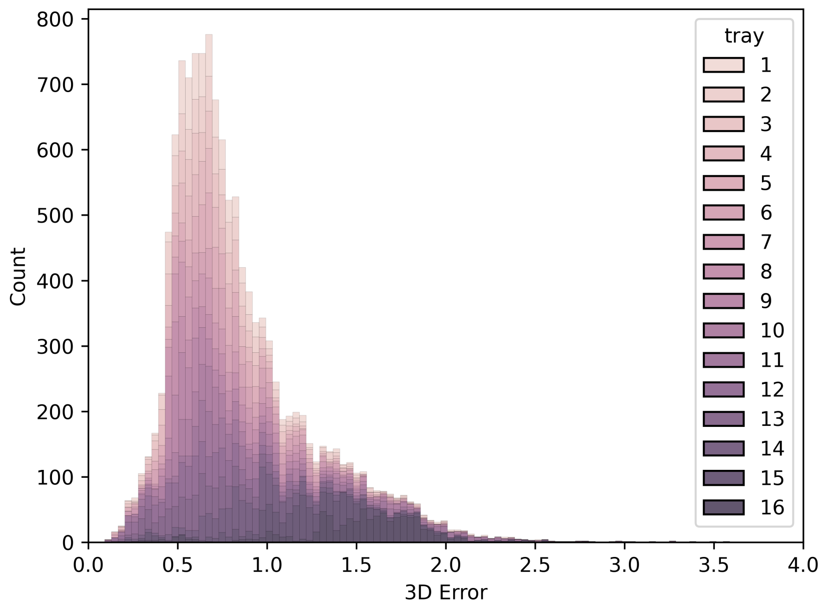

FIGURE 5.6: 3D error over the training dataset in the single-model configuration.

As it's evident in figures **5.7** and **5.8**, the single-model configuration is more consistent than the multi-model one. This is due to the fact that the training phase is performed with a wider dataset than the other configuration, but of diversified data.
The multi-model configuration presents drastic jumps in the transition from one model to another, giving to the switching model algorithm too weight in 3D position evaluation.

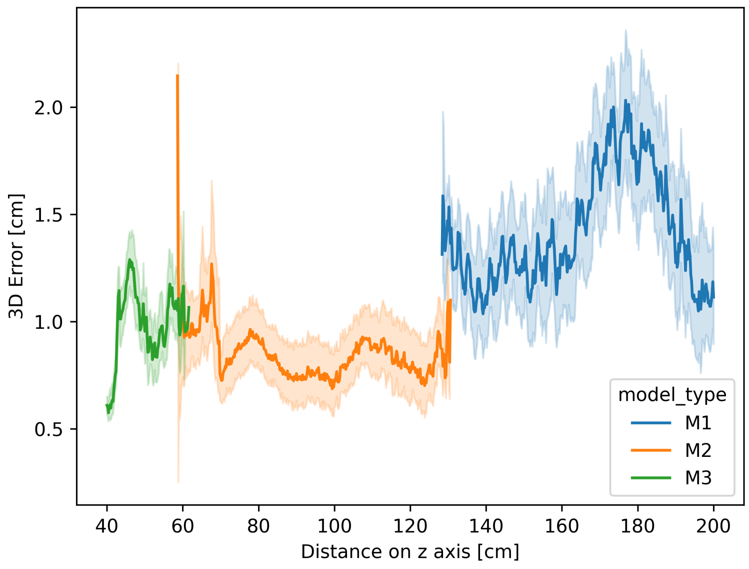

FIGURE 5.7: Landmark Mapping error over the trajectory range (*z* axis) in the multi-model configuration.

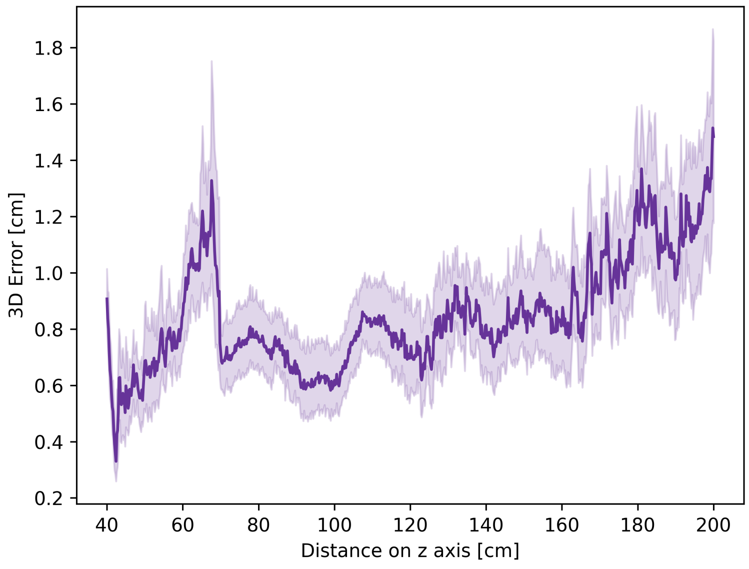

FIGURE 5.8: Landmark Mapping error over the trajectory range (*z* axis) in the single-model configuration.

Overall the single model configuration presents better performances in the 3D landmarks position prediction. The main problem of the multi-model configuration is the amount of data used to train the three models. With a wider training dataset and a more accurate decision algorithm in the transition from a model to another, this configuration could introduce more robustness in the landmarks position prediction.

In the following table are compared the performances of the two analyzed configuration for each training trajectory.

Trajectory	E_{NN} multi-model	E_{NN} single model
TRAY_1	(1.02 ± 0.22) cm	(0.76 ± 0.22) cm
TRAY_2	(1.06 ± 0.22) cm	(0.74 ± 0.22) cm
TRAY_3	(1.02 ± 0.23) cm	(0.73 ± 0.23) cm
TRAY_4	(1.05 ± 0.23) cm	(0.69 ± 0.23) cm
TRAY_5	(1.16 ± 0.24) cm	(0.74 ± 0.24) cm
TRAY_6	(1.30 ± 0.32) cm	(0.90 ± 0.30) cm
TRAY_7	(0.87 ± 0.18) cm	(0.72 ± 0.18) cm
TRAY_8	(0.82 ± 0.13) cm	(0.72 ± 0.13) cm
TRAY_9	(0.86 ± 0.19) cm	(0.76 ± 0.19) cm
TRAY_10	(0.91 ± 0.15) cm	(0.84 ± 0.15) cm
TRAY_11	(1.03 ± 0.07) cm	(0.95 ± 0.07) cm
TRAY_12	(0.94 ± 0.16) cm	(0.66 ± 0.17) cm
TRAY_13	(1.03 ± 0.17) cm	(0.70 ± 0.17) cm
TRAY_14	(1.31 ± 0.27) cm	(0.92 ± 0.27) cm
TRAY_15	(1.59 ± 0.28) cm	(1.20 ± 0.28) cm
TRAY_16	(1.90 ± 0.42) cm	(1.52 ± 0.42) cm
Overall	(1.12 ± 0.48) cm	(0.85 ± 0.35) cm

TABLE 5.3: Landmark Mapping module results with single and multi-model configurations.

Final Pose Evaluation

The *CPD* module introduces an ulterior error due to the alignment of the reference points set with the predicted one.

It's important to notice that the two errors E_R and E_T are deeply affected by the number of points in the predicted point set.

In cases of a sparse point set, CPD struggles to discern a coherent structure, leading to decreased accuracy in the alignment process. The algorithm's ability to effectively capture the global patterns and deformations decreases, resulting in sub-optimal alignments. Therefore, maintaining an adequate number of points in the input sets is crucial for CPD to deliver robust and reliable performance, ensuring the successful alignment of point clouds in various applications.

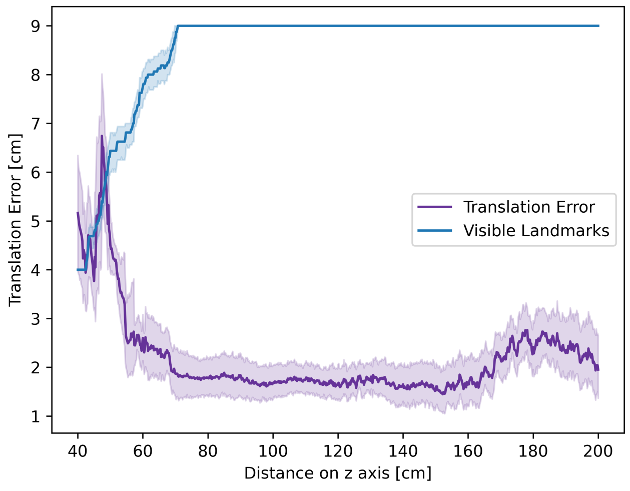

FIGURE 5.9: Translation error over the trajectory range (z axis) in multi-model configuration.

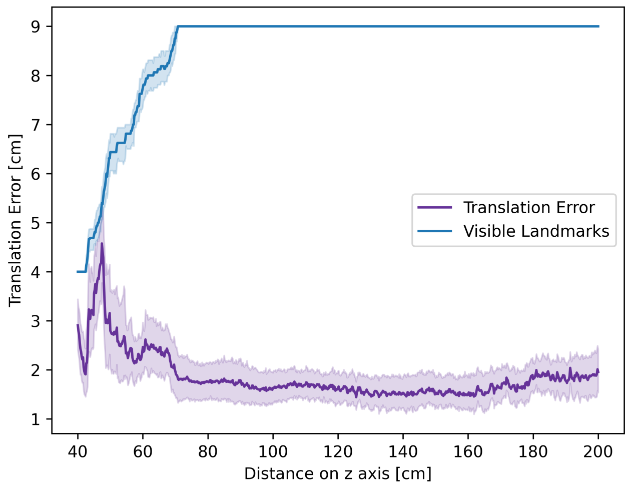

FIGURE 5.10: Translation error over the trajectory range (z axis) in single-model configuration.

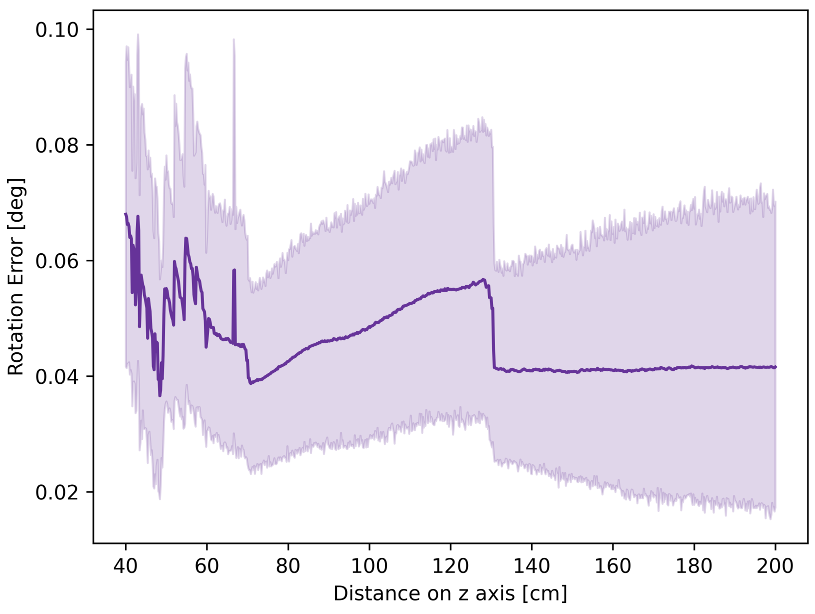

FIGURE 5.11: Rotation error over the trajectory range (z axis) in multi-model configuration.

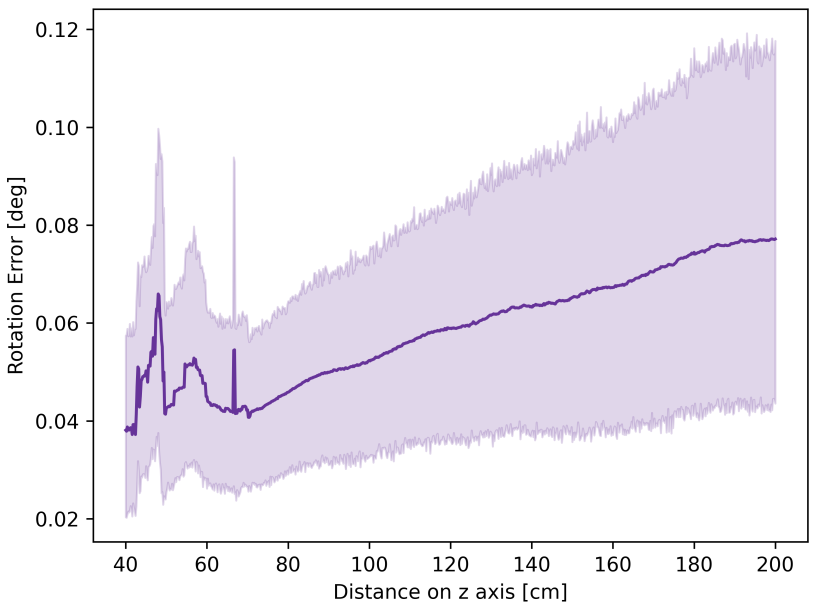

FIGURE 5.12: Rotation error over the trajectory range (z axis) in single-model configuration.

As shown in figures **5.9** and **5.10**, the algorithm performances vary drastically as the number of predicted landmarks becomes six or less.

Trajectory	E_{CNN} (pxls)	E_{NN} (cm)	E_T (cm)	E_R (°)	S_T	S_R	S
TRAY_1	1.37 ± 0.05	1.02 ± 0.22	2.41 ± 2.09	0	0.0228	0	0.0228
TRAY_2	1.39 ± 0.05	1.06 ± 0.22	2.17 ± 1.67	0.010	0.0200	0.0001	0.0201
TRAY_3	1.35 ± 0.06	1.02 ± 0.23	1.92 ± 0.98	0.030	0.0180	0.0005	0.0185
TRAY_4	1.34 ± 0.06	1.05 ± 0.23	1.89 ± 0.67	0.058	0.0170	0.0010	0.0180
TRAY_5	1.37 ± 0.07	1.16 ± 0.24	2.08 ± 0.86	0.095	0.0196	0.0016	0.0212
TRAY_6	1.38 ± 0.07	1.30 ± 0.32	2.50 ± 4.91	0.137	0.0245	0.0024	0.0269
TRAY_7	1.37 ± 0.06	0.87 ± 0.18	1.96 ± 1.50	0.006	0.0185	0.0001	0.0186
TRAY_8	1.43 ± 0.06	0.82 ± 0.13	1.65 ± 1.33	0.022	0.0160	0.0004	0.0164
TRAY_9	1.33 ± 0.06	0.86 ± 0.19	1.64 ± 1.48	0.048	0.0158	0.0008	0.0166
TRAY_10	1.34 ± 0.06	0.91 ± 0.15	1.71 ± 1.94	0.083	0.0167	0.0014	0.0181
TRAY_11	1.42 ± 0.07	1.03 ± 0.07	1.91 ± 3.00	0.127	0.0190	0.0022	0.0212
TRAY_12	1.41 ± 0.05	0.94 ± 0.16	1.67 ± 1.88	0.006	0.0158	0.0001	0.0159
TRAY_13	1.38 ± 0.05	1.03 ± 0.17	1.55 ± 1.60	0.014	0.0138	0.0002	0.0140
TRAY_14	1.38 ± 0.05	1.31 ± 0.27	2.30 ± 1.06	0.023	0.0195	0.0004	0.0199
TRAY_15	1.38 ± 0.05	1.59 ± 0.28	3.21 ± 0.70	0.032	0.0271	0.0005	0.0276
TRAY_16	1.37 ± 0.05	1.90 ± 0.42	4.23 ± 0.60	0.042	0.0357	0.0007	0.0364
Overall	1.38 ± 0.06	1.12 ± 0.48	2.18 ± 2.09	0.046	0.0200	0.0008	**0.0208**

TABLE 5.4: Results on training set with multi-model configuration.

Trajectory	E_{CNN} (pxls)	E_{NN} (cm)	E_T (cm)	E_R (°)	S_T	S_R	S
TRAY_1	1.37 ± 0.05	0.76 ± 0.22	2.14 ± 2.09	0	0.0196	0	0.0196
TRAY_2	1.39 ± 0.05	0.74 ± 0.22	1.89 ± 1.67	0.010	0.0168	0.0001	0.0169
TRAY_3	1.35 ± 0.06	0.73 ± 0.23	1.72 ± 0.98	0.032	0.0152	0.0005	0.0157
TRAY_4	1.34 ± 0.06	0.69 ± 0.23	1.66 ± 0.67	0.065	0.0152	0.0011	0.0163
TRAY_5	1.37 ± 0.07	0.74 ± 0.24	1.72 ± 0.86	0.110	0.0164	0.0019	0.0183
TRAY_6	1.38 ± 0.07	0.90 ± 0.30	1.98 ± 4.91	0.166	0.0197	0.0029	0.0226
TRAY_7	1.37 ± 0.06	0.72 ± 0.18	1.93 ± 1.50	0.007	0.0174	0.0001	0.0175
TRAY_8	1.43 ± 0.06	0.72 ± 0.13	1.79 ± 1.33	0.028	0.0159	0.0005	0.0164
TRAY_9	1.33 ± 0.06	0.76 ± 0.19	1.64 ± 1.48	0.062	0.0145	0.0011	0.0155
TRAY_10	1.34 ± 0.06	0.84 ± 0.15	1.55 ± 1.94	0.109	0.0131	0.0019	0.0150
TRAY_11	1.42 ± 0.07	0.95 ± 0.07	1.54 ± 3.00	0.168	0.0126	0.0029	0.0155
TRAY_12	1.41 ± 0.05	0.66 ± 0.17	1.17 ± 1.88	0.011	0.0112	0.0002	0.0114
TRAY_13	1.38 ± 0.05	0.70 ± 0.17	0.85 ± 1.60	0.023	0.0083	0.0004	0.0087
TRAY_14	1.38 ± 0.05	0.92 ± 0.27	1.65 ± 1.06	0.036	0.0152	0.0006	0.0158
TRAY_15	1.38 ± 0.05	1.20 ± 0.28	2.67 ± 0.70	0.049	0.0240	0.0009	0.0249
TRAY_16	1.37 ± 0.05	1.52 ± 0.42	3.75 ± 0.60	0.064	0.0335	0.0011	0.0346
Overall	1.38 ± 0.06	0.85 ± 0.35	1.85 ± 1.19	0.058	0.0168	0.0010	**0.0178**

TABLE 5.5: Results on training set with single-model configuration.

Tables **5.4** and **5.5** highlight that the single-model configuration demonstrates superior overall performance (S = 0.0178) compared to the multi-model configuration (S = 0.0208). However, it's noteworthy that the multi-model configuration has slightly better performance in the rotation aspect despite the overall advantage of the single-model setup.

5.2.3 Evaluation on Test Datasets

The second experiment is carried out on a separate test dataset composed of four new and unseen trajectories. This test dataset is intentionally kept independent from the training data, simulating real-world scenarios and ensuring the model's generalization capabilities. The performance on the test dataset provides insights into the method's ability to extrapolate learned patterns and accurately handle novel data instances. Together, these two experiments facilitate a comprehensive assessment of the method's robustness, effectiveness, and generalization across different datasets.

The first two trajectories (A and B) present an image quality and light conditions similar to the training images, while the other two ($Less\_Difficult\_Trajectory$ and $Difficult\_Trajectory$) present light condition too far from the training images and so the system struggles to predict the correct position of each landmarks in the image due to the fact that the *Landmark Mapping* module is quite sensitive to the output of the *Landmark Regression* one. Indeed, if the landmarks that are supposed to be present in the image frame are not all recognised, the 3D landmark position is predicted with high error.
So the system highly relies on the correct identification of each landmark in the image by the *Landmark Regression* (a deepened analysis of the problem and possible improvements is discussed at section **6.3**).

Due to this, with the ladder dataset the evaluation is performed only on the *Landmark Mapping* and *CPD* modules, assuming exact the prediction of the landmarks position in the image (*2D Error = 0*).

As shown is figures **5.13** and **5.14** the 2D landmarks location is predicted with a good accuracy also on the test set, with an average 2D error $E_{CNN} = 1.35 \pm 0.04$ pixels (corresponding to the 0.0026% for 512x512 images) and a error variation on the z axis coherent with the training results.

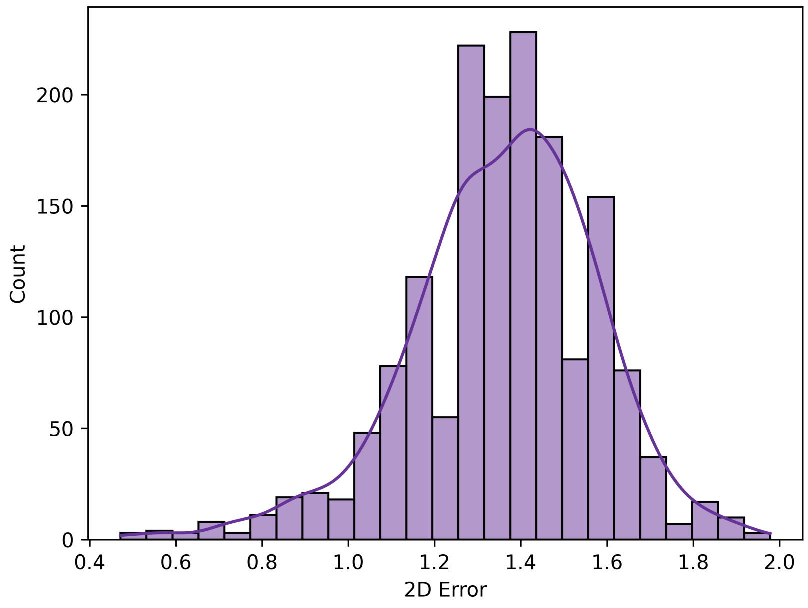

FIGURE 5.13: 2D error over the test dataset.

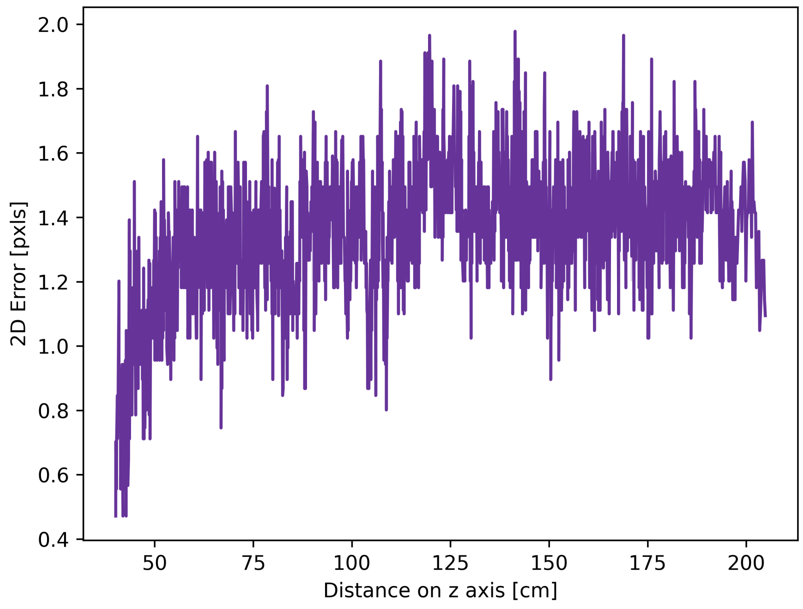

FIGURE 5.14: Landmark Regression error over the trajectory range (z axis) on the test dataset.

In contrast with the training results, the multi-model configuration of the *Landmark Mapping* module express better performances on the test set than the single-model configuration on each test trajectory.

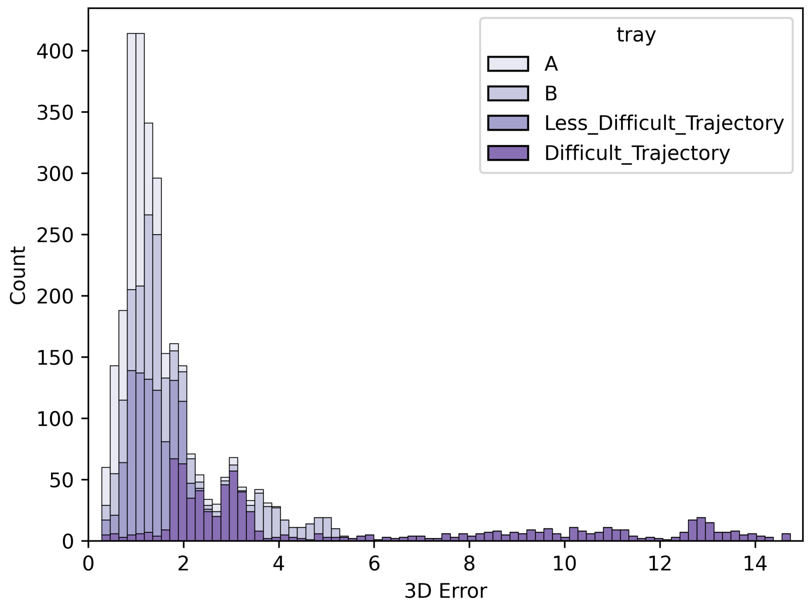

FIGURE 5.15: Landmark Mapping error in multi-model configuration on test dataset.

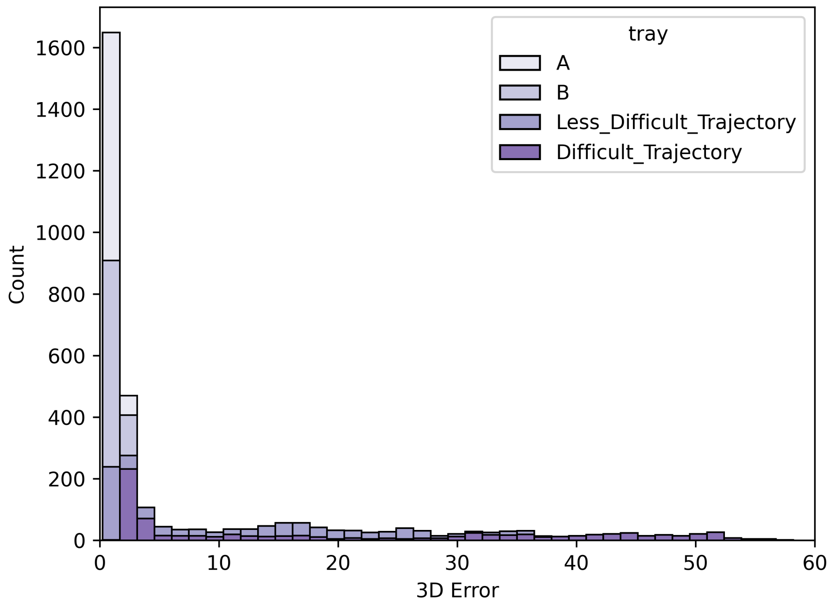

FIGURE 5.16: Landmark Mapping error in single-model configuration on test dataset.

In both configurations the *Difficult Trajectory* is predicted with noticeable error in particular in its first part where its RPY rotations are considerable, even beyond the range of rotations known by the models (see figure **5.17** and **5.18**).

The EROSS project [24] sets a stringent guideline, aiming to a less than 2-5 centimeters 3D error, reflecting the project's emphasis on precision in spaceborne applications. The achieved 3D error, measured in the experiments, consistently falls within the specified range, showcasing the system's capability to deliver accurate and reliable results, crucial for the success of rendezvous maneuvers and other space missions.

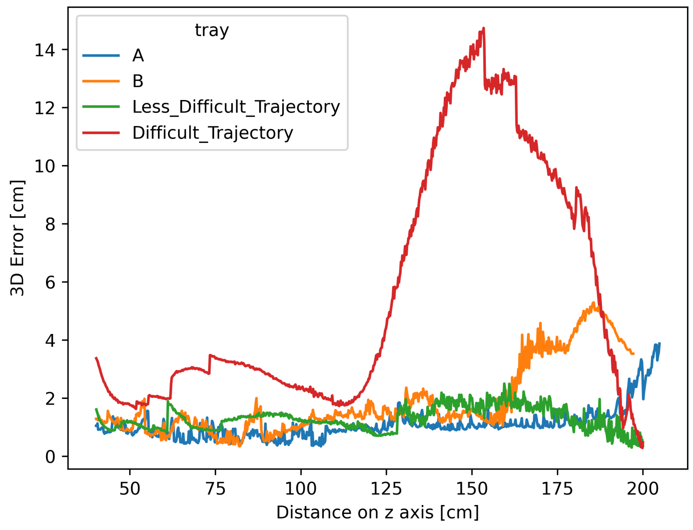

FIGURE 5.17: Landmark Mapping error over the trajectory range (z axis) in the multi-model configuration on test dataset.

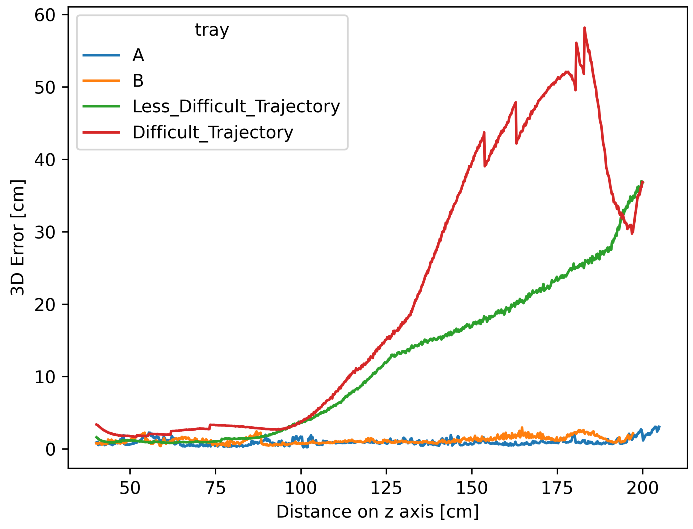

FIGURE 5.18: Landmark Mapping error over the trajectory range (z axis) in the single-model configuration on test dataset.

As previously discussed, the final translation and rotation errors E_T and E_R are affected by the number of landmarks in the predicted point set. In figures 5.19 and 5.20, in positions closed to the target, it's possible to see that the translation error increases as far as the number of visible landmarks in the image frame decreases.

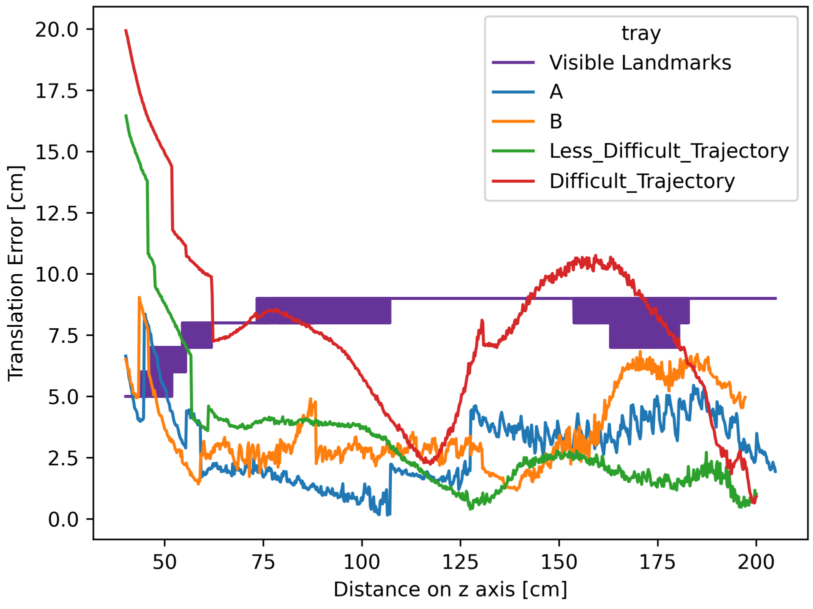

FIGURE 5.19: Translation error over the trajectory range (z axis) in multi-model configuration on test dataset.

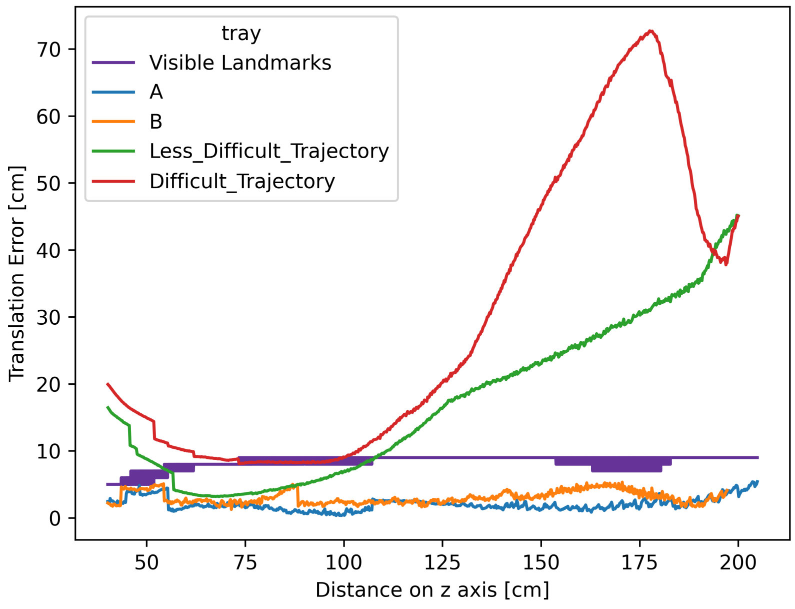

FIGURE 5.20: Translation error over the trajectory range (z axis) in single-model configuration on test dataset.

The multi-model configuration exhibits noticeable performance jumps during the transitions between models. These abrupt shifts highlight the challenge associated with integrating different models, suggesting the need for further refinement in the transition logic to ensure smoother and more consistent predictions across varying proximity bands.

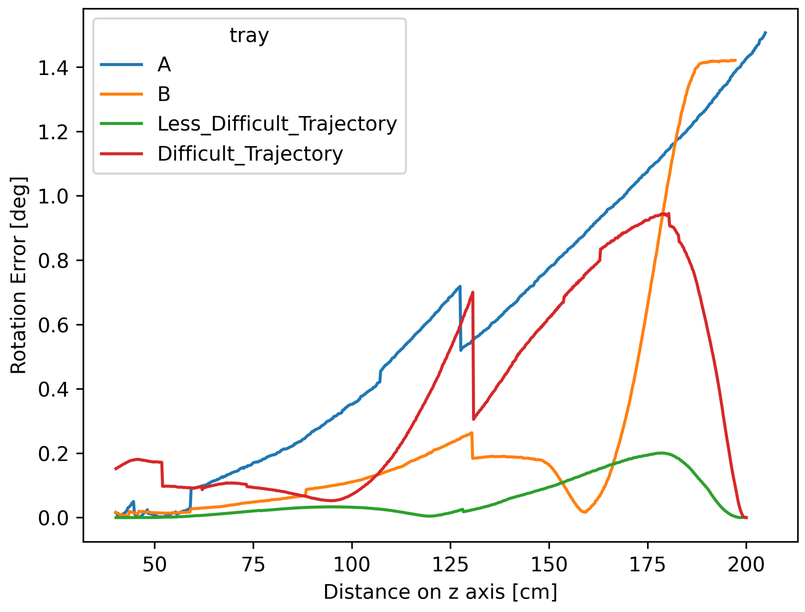

FIGURE 5.21: Rotation error over the trajectory range (z axis) in multi-model configuration on test dataset.

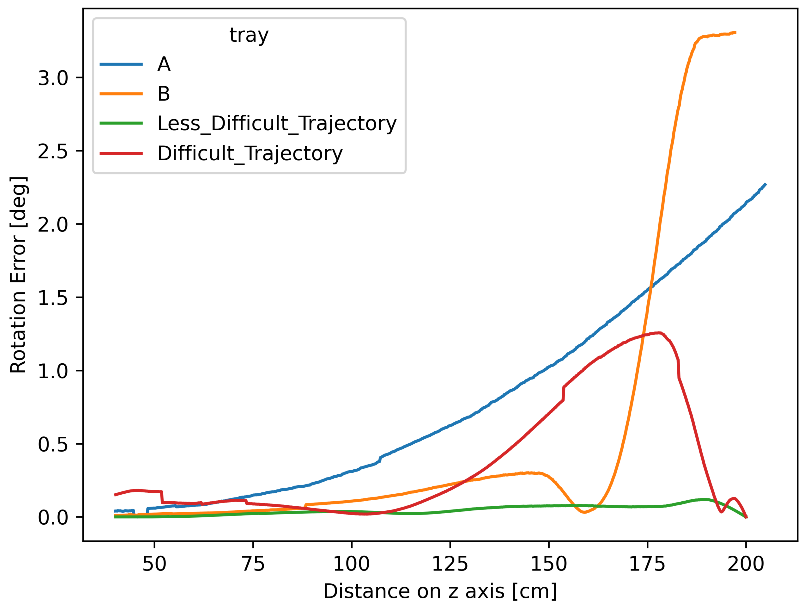

FIGURE 5.22: Rotation error over the trajectory range (z axis) in single-model configuration on test dataset.

In the following tables are reported the analysed errors and scores for each test trajectory and in both analyzed configurations: multi-model and- single-model.

In contrast to the training results, the single-model configuration has lower performance both on translation and rotation, meaning that compared to the multi-model one demonstrates an overfitting behavior.

Overall, considering the multi-model configuration, the testing performance, both on rotation (S_R) and translation (S_T), is not so far from the training one and quite encouraging, demonstrating the ability of this configuration to generalize when dealing with unknown data.

Trajectory	E_{CNN} (pxls)	E_{NN} (cm)	E_T (cm)	E_R (°)	S_T	S_R	S
TRAY_A	1.34 ± 0.05	1.08 ± 0.29	2.96	0.601	0.0238	0.0105	0.0343
TRAY_B	1.37 ± 0.05	1.89 ± 1.60	3.62	0.281	0.0287	0.0049	0.0336
Less_D_Tray	0.00 ± 0.00	1.27 ± 0.15	3.32	0.057	0.0337	0.0010	0.0347
Difficult_Tray	0.00 ± 0.00	5.58 ± 17.96	7.73	0.375	0.0697	0.0066	0.0763
Overall	1.35 ± 0.04	2.46 ± 8.38	4.41	0.328	0.0390	0.0057	**0.0447**

TABLE 5.6: Results on test set with multi-model configuration.

Trajectory	E_{CNN} (pxls)	E_{NN} (cm)	E_T (cm)	E_R (°)	S_T	S_R	S
TRAY_A	1.34 ± 0.05	0.95 ± 0.29	2.04	0.781	0.0165	0.0136	0.0301
TRAY_B	1.37 ± 0.05	1.21 ± 1.60	3.06	0.544	0.0246	0.0095	0.0341
Less_D_Tray	0.00 ± 0.00	11.76 ± 0.15	17.19	0.045	0.1166	0.0008	0.1174
Difficult_Tray	0.00 ± 0.00	20.69 ± 17.96	29.36	0.373	0.2010	0.0065	0.2075
Overall	1.35 ± 0.04	8.67 ± 180.0	12.9	0.435	0.0897	0.0076	**0.0973**

TABLE 5.7: Results on test set with single-model configuration.

Chapter 6

Discussion and conclusions

6.1 Challenges in On-Board AI Systems for Space Missions

AI algorithms in on-board space applications encounter significant challenges related to both verifiability and computational load, crucial factors for the success and safety of space missions.

6.1.1 Verifiability Issues

AI algorithms, particularly those employing deep learning, are characterized by intricate architectures and numerous parameters. The complexity of these models makes it challenging to provide comprehensive assurance of their correctness. In space applications, where system failures are not an option, ensuring the verifiability of AI algorithms becomes predominant.

Many AI models, including deep neural networks, lack inherent explainability. Understanding the decision-making process within these "black box" models is essential to verify their reliability. Achieving transparency in AI decision logic is critical in scenarios where the basis for decision-making must be interpretable, such as during critical space maneuvers.

Space environments are dynamic and may exhibit uncertainties. AI algorithms designed for adaptability and learning might introduce challenges in predicting their behavior accurately. Verifying the robustness of adaptive AI systems in the face of unforeseen conditions is a persistent concern.

6.1.2 Computational Issues

On-board space systems typically operate with limited computational resources due to factors such as size, weight, and power constraints (SWaP). Implementing AI algorithms with demanding computational requirements may strain available resources, affecting the overall efficiency of the system.

Certain space applications, such as autonomous navigation or hazard avoidance, demand real-time decision-making. AI algorithms with high computational loads may struggle to meet these stringent timing constraints. Delays in processing could lead to missed opportunities or, in critical situations, mission failure.

In addition to computational power, energy efficiency is a crucial consideration. Prolonged missions and the reliance on energy-harvesting sources necessitate AI algorithms that balance computational complexity with energy consumption, ensuring sustained and reliable operation.

Addressing these challenges requires a multidisciplinary approach involving AI researchers, space engineers, and mission planners. Techniques such as formal verification, explainable AI, and hardware optimization are essential to enhance the verifiability and efficiency of AI algorithms in on-board space applications.

6.2 Results Analysis

In the broader context, the multi-model configuration emerges as the more robust and adaptable option, showcasing superior performance on the test set and demonstrating effective generalization capabilities to previously unseen data. The overall system score on the test dataset is $S = 0.0447$, primarily affected by the translation component $S_T = 0.0390$, as opposed to the relatively lower contribution from the rotation aspect, $S_R = 0.0057$. The noteworthy aspect is the necessity of prioritize the minimization of translation errors, since, in proximity to the target, precise translation is crucial for accurate maneuvering. Moreover, rotation pose can be more effectively predicted with the incorporation of a navigation filter. This strategic integration allows for a corrective mechanism, compensating for rotational discrepancies and enhancing the system's overall precision in navigating close quarters.

6.3 Possible Improvements

6.3.1 Landmark Selection

As already discussed at section **4.3.1**, the selection of the number and position of landmarks in the model is crucial for the resulting performances of the system. Since the final translation and rotation errors (E_T, E_R) are strictly related to the number of landmarks identified in the image frame, which reduces in positions closer to the target, a suitable improvement would be the increase of the initial number of landmarks.

Another option would be using multiple models also for the *Landmark Regression* module, which are able to identify a target set of landmarks in farther positions and a second set in closer distances to the target, in order to keep the number of identified points in the image frame as greater as possible.
This implementation would also lead to a more accurate pose estimation in positions closer to the minimum distance analyzed in the experiments (from $40cm$ to $20cm$). A well-performed pose estimation in positions very close to the target would help to minimize any errors introduced by camera distortions.

6.3.2 Landmark Mapping Sensitivity

The assessment of trajectories such as *Less_Difficult_Trajectory* and *Difficult_Trajectory* is conducted with a noteworthy consideration: the assumption of zero prediction error for the points' location in the image. This assumption is made out of necessity since the *Landmark Regression* module encounters difficulties in identifying all the landmarks expected to be present in the image frames. Consequently, the input data for the *Landmark Mapping* module is marked by a heightened sensitivity, as the accuracy of its predictions is contingent upon the successful identification of landmarks by the preceding module.

A potential avenue for improvement involves an expansion of the training dataset to incorporate instances where certain landmarks remain unidentified due to inherent challenges in their recognition. This strategy aims to enhance the model's resilience to scenarios where specific landmarks pose persistent issues during identification. By exposing the model to a more diverse range of challenges and including cases of landmark ambiguity, it is anticipated that the trained model will develop a more robust understanding, leading to improved performance, particularly in situations mirroring real-world complexities. This adjustment aligns with the overarching goal of fortifying the system's adaptability and generalization capabilities, addressing challenges posed by varying environmental conditions and unforeseen factors during autonomous space applications.

Moreover, to fortify the robustness of the system, there is a prospect to introduce a more sophisticated pre-processing system. This advanced system would be designed to mitigate the impact of varying image light conditions on landmark identification. By incorporating techniques such as adaptive image enhancement, contrast normalization, or even exploring deep learning-based methods for illumination invariance, the model could become less susceptible to fluctuations in lighting. Such enhancements would foster greater reliability in landmark identification by the *Landmark Regression* module, subsequently improving the overall accuracy of the *Landmark Mapping* module. This proactive approach anticipates and addresses challenges associated with real-world scenarios where illumination conditions can be unpredictable, ensuring the model's effectiveness across diverse operational environments in on-board space applications.

6.4 Conclusions

This project presents a dedicated monocular pose estimation framework designed for spaceborne objects, emphasizing its applicability to satellite rendezvous maneuvers. The framework capitalizes on the strengths of deep neural networks, seamlessly integrating feature learning and establishing robust 2D-3D correspondence mapping. Notably, the incorporation of HRNet, known for its high-resolution image representation, significantly contributes to the precision of pose predictions and the subsequent refinement process. The framework further demonstrates its efficiency by employing geometric optimization techniques, ensuring accurate alignment of point sets and enhancing the overall robustness of the pose estimation system.

Appendix A

Support Code

A.1 Ground Truth Heatmaps

```python
import numpy as np

def createHeatmap(landmark, vp, hmap_w, hmap_h, sig=1):
    hmap = np.zeros((hmap_height + 3, hmap_width + 3))
    x, y = landmark
    if vp:
        for i in range(y - 3*sig, y + 3*sig):
            for j in range(x - 3*sig, x + 3*sig):
                hmap[i,j]+=np.exp(-((i-y)** 2+(j-x)**2)/(2*sig**2))

    hmap = hmap[1:-2, 1:-2]

    return hmap

def coord2Heatmap(landmarks, vis, hmap_w=512, hmap_h=512, sig=1):
    hmaps = []

    for landmark, vp in zip(landmarks, vis):
        hmaps.append(createHeatmap(landmark,vp,hmap_w,hmap_h,sig))

    hmaps = np.array(hmaps).squeeze()

    return hmaps
```

LISTING A.1: Ground Truth Heatmaps

A.2 Landmark Location Selection

```python
import numpy as np

def get_landmarks(landmarks_images, threshold1=-0.5, threshold2=-0.6):
    landmarks2D = []
    for img in landmarks_images:
        Lnd_found = False
        lndx = -1
        lndy = -1
        mask1 = img < threshold2
        img[mask1] = -1

        if np.max(img) > threshold1:
            Lnd_found = True
        elif np.max(img) > threshold2 and np.var(img) > 2e-6:
            Lnd_found = True

        if Lnd_found:
            p_lndy,p_lndx = np.where(img == np.max(img))
            lndx = np.round(np.mean(p_lndx))
            lndy = np.round(np.mean(p_lndy))
            if landmarks2D is not None or len(landmarks2D) > 0:
                for data in landmarks2D:
                    if data[2] == 1 and abs(lndx - data[0]) <= 3
                        and abs(lndy - data[1]) <= 3:
                        if np.max(img)>data[3] and np.var(img)>data[4]:
                            data[0] = -1
                            data[1] = -1
                            data[2] = 0
                            n_lnd=[lndx,lndy,1,np.max(img),np.var(img)]
                        else:
                            n_lnd = [-1,-1,0,np.max(img),np.var(img)]

            n_lnd = [lndx,lndy,1,np.max(img),np.var(img)]
        else:
            n_lnd = [-1,-1,0,np.max(img),np.var(img)]

        landmarks2D.append(n_lnd)

    return np.array(landmarks2D)
```

LISTING A.2: Landmark Location Selection

Bibliography

[1] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[2] Samarth Brahmbhatt et al. "MapNet: Geometry-Aware Learning of Maps for Camera Localization". In: *CoRR* abs/1712.03342 (2017). arXiv: 1712.03342. URL: http://arxiv.org/abs/1712.03342.

[3] Pasqualetto Cassinis. "Delft University of Technology Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft". In: 2019.

[4] cordis.europa.eu CORDIS. *European Robotic Orbital Support Services: Eross project: Fact sheet: H2020: Cordis: European Commission*. Nov. 2018. URL: https://cordis.europa.eu/project/id/821904.

[5] Simone D'Amico, Mathias Benn, and John Leif Jørgensen. "Pose estimation of an uncooperative spacecraft from actual space imagery". In: 2014. URL: https://api.semanticscholar.org/CorpusID:1537971.

[6] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24 (1981). URL: https://api.semanticscholar.org/CorpusID:972888.

[7] Kunihiko Fukushima. "Neocognition: A self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: (1980).

[8] Kaiming He et al. "Mask R-CNN". In: (2017), pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[9] Jeff Heaton. *T81-558:Applications of Deep Neural Networks*. 2023. URL: https://sites.wustl.edu/jeffheaton/t81-558/ (visited on 11/27/2023).

[10] John F. Hughes et al. *Computer Graphics: Principles and Practice*. 3rd ed. Upper Saddle River, NJ: Addison-Wesley, 2013.

[11] Alex Kendall and Roberto Cipolla. "Modelling Uncertainty in Deep Learning for Camera Relocalization". In: *CoRR* abs/1509.05909 (2015). arXiv: 1509.05909. URL: http://arxiv.org/abs/1509.05909.

[12] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "Convolutional networks for real-time 6-DOF camera relocalization". In: *CoRR* abs/1505.07427 (2015). arXiv: 1505.07427. URL: http://arxiv.org/abs/1505.07427.

[13] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[14] Mate Kisantal et al. "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results". In: *CoRR* abs/1911.02050 (2019). arXiv: 1911.02050. URL: http://arxiv.org/abs/1911.02050.

[15] Yann Lecun and Y. Bengio. "Convolutional Networks for Images, Speech, and Time-Series". In: *The Handbook of Brain Theory and Neural Networks* (Jan. 1995).

[16] Vincent Lepetit, Francesc Moreno-Noguer, and P. Fua. “EPnP: An Accurate O(n) Solution to the PnP Problem”. In: *International Journal of Computer Vision* 81 (2009), pp. 155–166. URL: https://api.semanticscholar.org/CorpusID:207252029.

[17] Fei-Fei Li. *CS231n: Deep Learning for Computer Vision*. 2023. URL: http://cs231n.stanford.edu (visited on 11/27/2023).

[18] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60 (2004), pp. 91–110. URL: https://api.semanticscholar.org/CorpusID:174065.

[19] Federico Moscato. *satellite-visual-pose-estimator*. https://github.com/JMFede/satellite-visual-pose-estimator. 2023.

[20] Andriy Myronenko and Xubo B. Song. “Point-Set Registration: Coherent Point Drift”. In: *CoRR* abs/0905.2635 (2009). arXiv: 0905.2635. URL: http://arxiv.org/abs/0905.2635.

[21] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *CoRR* abs/1912.01703 (2019). arXiv: 1912.01703. URL: http://arxiv.org/abs/1912.01703.

[22] Georgios Pavlakos et al. “6-DoF Object Pose from Semantic Keypoints”. In: *CoRR* abs/1703.04670 (2017). arXiv: 1703.04670. URL: http://arxiv.org/abs/1703.04670.

[23] Sida Peng et al. “PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation”. In: *CoRR* abs/1812.11788 (2018). arXiv: 1812.11788. URL: http://arxiv.org/abs/1812.11788.

[24] Maximo A. Roa et al. “EROSS: In-Orbit Demonstration of European Robotic Orbital Support Services”. In: (2023).

[25] Torsten Sattler et al. “Understanding the Limitations of CNN-based Absolute Camera Pose Regression”. In: *CoRR* abs/1903.07504 (2019). arXiv: 1903.07504. URL: http://arxiv.org/abs/1903.07504.

[26] Sumant Sharma and Simone D’Amico. “Pose Estimation for Non-Cooperative Rendezvous Using Neural Networks”. In: *CoRR* abs/1906.09868 (2019). arXiv: 1906.09868. URL: http://arxiv.org/abs/1906.09868.

[27] Sumant Sharma, Jacopo Ventura, and Simone D’Amico. “Robust Model-Based Monocular Pose Initialization for Noncooperative Spacecraft Rendezvous”. In: *Journal of Spacecraft and Rockets* (2018). URL: https://api.semanticscholar.org/CorpusID:125936812.

[28] Ke Sun et al. *Deep High-Resolution Representation Learning for Human Pose Estimation*. 2019. arXiv: 1902.09212 [cs.CV].

[29] Ke Sun et al. “High-Resolution Representations for Labeling Pixels and Regions”. In: *CoRR* abs/1904.04514 (2019). arXiv: 1904.04514. URL: http://arxiv.org/abs/1904.04514.

[30] Martin Sundermeyer et al. “Implicit 3D Orientation Learning for 6D Object Detection from RGB Images”. In: *CoRR* abs/1902.01275 (2019). arXiv: 1902.01275. URL: http://arxiv.org/abs/1902.01275.

[31] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. “Real-Time Seamless Single Shot 6D Object Pose Prediction”. In: *CoRR* abs/1711.08848 (2017). arXiv: 1711.08848. URL: http://arxiv.org/abs/1711.08848.

[32] Bill Triggs et al. “Bundle Adjustment - A Modern Synthesis”. In: *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*. ICCV ’99. Springer-Verlag, 2000, pp. 298–372. ISBN: 3-540-67973-1. URL: http://dl.acm.org/citation.cfm?id=646271.685629.

Acknowledgements

Desidero esprimere i miei più sentiti ringraziamenti a tutte le persone che hanno contribuito al completamento di questa tesi.

Innanzitutto, desidero ringraziare il mio relatore, il prof. Marcello Chiaberge, e correlatore, l'ing. Andrea Merlo, per avermi dato la possibilità di intraprendere questo progetto. Ringrazio l'ing. Marco Lapolla per la sua guida, disponibilità e dedizione nell'aiutarmi a sviluppare e perfezionare il mio lavoro.

Un ringraziamento speciale va alla mia famiglia che ha sempre sostenuto e incoraggiato il mio percorso accademico. Il loro sostegno e supporto sono stati la spinta necessaria per superare le sfide e raggiungere questo traguardo.

Ringrazio i compagni dell'università: Vale, Ali, Morgan, Matte, Nicoli, Franco, Luca e Nicco, che hanno alleviato le mie giornate e contro cui ho perso innumerevoli partite a bodriga.

Un ringraziamento agli amici di FORO, mia casa nell'ultimo periodo, con cui ho condiviso molte pause caffè.

Agli amici di una vita e compagni di mille avventure: Gian, Pier, Ale, Stol, Lollo e Chiara che, nonostante le nostre strade stiano prendendo direzioni diverse, sono e saranno sempre presenti al mio fianco.

Infine, il ringraziamento più importante va a Nau, mia compagna di viaggio, che mi ha sostenuto in questo percorso, credendo in me anche quando io non l'ho fatto.

Grazie mille a tutti.

Fede