



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Deep Learning Solutions for Attitude Ambiguity in Relative Navigation with Unknown and Uncooperative Targets

TESI DI LAUREA MAGISTRALE IN
SPACE ENGINEERING - INGEGNERIA SPAZIALE

Author: **Matteo Rosa**

Student ID: 995531

Advisor: Michele Maestrini

Co-advisors: Pierluigi Di Lizia, Maria Alessandra De Luca, Niccolò Faraco

Academic Year: 2022-23

Abstract

This project investigates the challenging problem of relative navigation in scenarios involving unknown and uncooperative symmetric targets. The significance of this problem is growing with the increase in the number of space debris. To address this issue, two types of missions are gaining importance, On-Orbit Servicing and Active Debris Removal, which involve the ability to safely conduct proximity operations around even completely unknown targets.

CoMBiNa (COarse Model-Based relatIve NAvigation) and its reformulation represents a new navigation technique for autonomous inspection. CoMBiNa uses measurements from a stereo camera, and by combining Simultaneous Localization and Mapping (SLAM) techniques, Bayesian Coherent Point Drift (BCPD) algorithm, and Unscented Kalman filters (UKF), it estimates the relative pose and inertial properties of the target. The limitation of CoMBiNa is that it fails with symmetric targets. A reformulation has been proposed in which, in case of symmetric targets, the state variables are changed. This new formulation allows to estimate the relative position and the relative velocity between the chaser and the target and the orientation of the symmetry axis of the target body. Nevertheless it is not able to compute the complete attitude of the target and its angular velocity. Starting from this basis and taking advantage of artificial intelligence, in particular Convolutional Neural Networks (CNNs), a new formulation is proposed in this project, that aims to reconstruct the complete relative pose and the target angular velocity even for symmetric targets. Through CNN, the presence of specific features on the target will be detected, and after identifying them, this information will be combined with the already existing CoMBiNa algorithm to try to estimate the complete state of the target. The features allow us to derive attitude measurements of the target, even if it is symmetrical. The CoMBiNa pipeline is modified introducing different methods to derive the measurements, depending on the number of features available. To verify the validity of the algorithm, numerical simulations are carried out, producing promising results.

Keywords: Resident Space Objects, Space Debris, Proximity Operations, Pose and Inertia Estimation, Stereo-Vision, Uncooperative and unknown, Symmetry, Convolutional Neural Network, Object Detection

Sommario

Questo progetto studia il difficile problema della navigazione relativa in scenari che coinvolgono bersagli simmetrici, sconosciuti e non cooperativi. L'importanza di questo argomento sta crescendo con l'aumento del numero di detriti spaziali. Per affrontare questo problema, stanno acquisendo importanza due tipi di missioni, On-Orbit Servicing e Active Debris Removal, che prevedono la capacità di condurre in sicurezza operazioni di prossimità intorno a bersagli anche completamente sconosciuti.

CoMBiNa (COarse Model-Based relatIve NAvigation) e la sua riformulazione rappresentano una nuova tecnica di navigazione per l'ispezione autonoma. CoMBiNa utilizza le misure di una stereo camera e, combinando le tecniche di localizzazione e mappatura simultanea (SLAM), l'algoritmo Bayesian Coherent Point Drift (BCPD) e i Filtri di Kalman Unscented (UKF), stima la posa relativa e le proprietà inerziali del target. Il limite di CoMBiNa è che fallisce con bersagli simmetrici. È stata proposta una riformulazione in cui, in caso di bersagli simmetrici, le variabili di stato vengono modificate. Questa nuova formulazione consente di stimare la posizione relativa e la velocità relativa tra l'inseguitore e il bersaglio e l'orientamento dell'asse di simmetria del corpo del bersaglio. Tuttavia, non è in grado di calcolare l'assetto completo del bersaglio e la sua velocità angolare. Partendo da questa base e sfruttando l'intelligenza artificiale, in particolare le Reti Neurali Convolutionali (CNN), in questo progetto viene proposta una nuova formulazione che mira a ricostruire la posa relativa completa e la velocità angolare del bersaglio anche per bersagli simmetrici. Tramite una CNN, viene rilevata la presenza di caratteristiche specifiche sul bersaglio e, dopo averle identificate, queste informazioni vengono integrate con l'algoritmo già esistente per cercare di stimare lo stato completo del bersaglio. Le caratteristiche permettono di ricavare misure di assetto del bersaglio, anche se è simmetrico. La pipeline di CoMBiNa viene modificata introducendo diversi metodi per ricavare le misure, a seconda del numero di caratteristiche disponibili. Per verificare la validità dell'algoritmo, sono state effettuate simulazioni numeriche che hanno dato risultati promettenti.

Parole chiave: Oggetti Spaziali, Detriti Spaziali, Operazioni di Prossimità, Stima della Posa e dell'Inerzia, Visione Steroscopica, Non Cooperativo e Sconosciuto, Simmetria, Rete Neurale Convolutionale, Rilevamento di Oggetti

Contents

Abstract	i
Sommario	iii
Contents	v
List of Figures	vii
List of Tables	xi
Acronyms	xiii
1 Introduction	1
1.1 Literature review: Relative Navigation	2
1.1.1 Sensors	3
1.1.2 Cooperative Target	4
1.1.3 Uncooperative and Known Target	5
1.1.4 Uncooperative and Unknown Target	6
1.2 Thesis Objective	8
1.3 Thesis Outline	8
2 Introduction to CoMBiNa	9
2.1 CoMBiNa: Original Formulation	9
2.1.1 Reference Systems	9
2.1.2 Preliminary Phase	10
2.1.3 Operational Phase	10
2.2 CoMBiNa: Symmetric Target	19
2.2.1 Measurements Extraction	21
2.2.2 Filter Structure for Known Rigid Rotation Matrix	22
2.2.3 Filter Structure for Unknown Rigid Rotation Matrix	23

3	Computer Vision	25
3.1	Object Detection	26
3.1.1	Convolutional Neural Networks	26
3.1.2	Region Proposal Methods	27
3.1.3	One Stage Methods	29
3.1.4	Performance Metrics	30
3.2	Perspective-n-Point problem	31
3.2.1	Efficient Perspective-n-Point Solver	32
4	CoMBiNa: New Formulation	35
4.1	CoMBiNa: Symmetric Case with Features Detection	35
4.1.1	Features Choice	35
4.1.2	Algorithm's Update	38
4.1.3	Filter Structure for Unknown Rigid Rotation Matrix	46
4.2	Numerical Results	47
4.2.1	Problem Description & Ground Truth	47
4.2.2	Known Rigid Rotation	51
4.2.3	Monte Carlo Analysis	57
4.2.4	Unknown Rigid Rotation	61
5	Features Detection	69
5.1	Image Generation Process	69
5.2	Training Process	73
5.2.1	Training Results	75
5.3	Testing Results	79
6	Conclusions and Future Works	83
6.1	Conclusions	83
6.2	Future Works	84
A	Appendix A	85
B	Appendix B	89
	Bibliography	93
	Acknowledgements	101

List of Figures

2.1	Parallel axis stereo camera model adopted in CoMBiNa (courtesy of [45])	11
2.2	Extended CoMBiNa operational phase (courtesy of [20])	20
2.3	Rendering of the 3D symmetric body (courtesy of [45])	20
3.1	CNN architecture (courtesy of [25])	27
4.1	Pipeline for the CoMBiNa operational phase	36
4.2	Coarse model and features expressed in the target reference frame	37
4.3	Coarse model and features expressed in the camera reference frame	37
4.4	Visible e non visible features expressed in the image reference frame	38
4.5	Representation of the orientation of the reference frame of the feature in the inertial reference frame	43
4.6	Representation of the nominal inclined football orbit in the LVLH frame.	48
4.7	Component-wise evolution of the relative position and velocity in the LVLH frame	49
4.8	Component-wise evolution of the chaser's attitude: discontinuities in the MRP happen at switching location between nominal and shadow set	50
4.9	Evolution of the new attitude variables of the target	51
4.10	Evolution of attitude and inertia related quantities of the target	52
4.11	Convergence of the rotational filter (1 features and failure detection per- centage = 0%)	53
4.12	Convergence of the rotational filter (1 features and failure detection per- centage = 40%)	54
4.13	Convergence of the rotational filter (6 features and failure detection per- centage = 0%)	55
4.14	Convergence of the rotational filter (6 features and failure detection per- centage = 40%)	56
4.15	Convergence of the rotational filter (14 features and failure detection per- centage = 0%)	58

4.16	Convergence of the rotational filter (14 features and failure detection percentage = 40%)	59
4.17	Number of convergences varying the number of features	61
4.18	Convergence of the rotational filter with unknown rigid rotation matrix(1 features and failure detection percentage = 0%)	63
4.19	Convergence of the rotational filter with unknown rigid rotation matrix (1 features and failure detection percentage = 40%)	64
4.20	Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 0%)	65
4.21	Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 40%)	66
4.22	Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 0%)	67
4.23	Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 40%)	68
5.1	Model of the satellite	70
5.2	Model 5	71
5.3	File <i>.txt</i> used by YOLOv8	72
5.4	Comparison between different YOLO's architectures (courtesy of [37])	74
5.5	Losses and metrics of the training process	76
5.6	Confusion matrix	77
5.7	Precision-Recall curve	78
5.8	Output images generated by YOLOv8	79
5.9	Error distribution in bounding box centre detection	81
A.1	Convergence of the rotational filter with unknown rigid rotation matrix (1 features and failure detection percentage = 20%)	86
A.2	Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 20%)	87
A.3	Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 20%)	88
B.1	Convergence of the rotational filter with unknown rigid rotation matrix (1 features and failure detection percentage = 20%)	90
B.2	Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 20%)	91

B.3 Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 20%) 92

List of Tables

2.1	Reference frames	10
4.1	Stereo camera technical specifications	47
5.1	Features	70
5.2	Reference frames	73
5.3	Comparison between the different available model of YOLOv8	74
5.4	Average Precisions	78
5.5	Mean errors and standard deviations	80
5.6	Failure detection percentage for each class	81
A.1	Number of time steps without features and with the switch from EPnP to BCPD, by varying the number of features	85
B.1	Number of time steps without features and with the switch from EPnP to BCPD, by varying the number of features	89

Acronyms

A

ADR Active Debris Removal.

AP Average Precision.

B

BCPD Bayesian Coherent Point Drift.

C

CNN Convolutional Neural Network.

CoMBiNa Coarse Model-Based Relative Navigation.

CPD Coherent Point Drift.

E

EKF Extended Kalman Filter.

EPnP Efficient Perspective-n-Point.

F

FPN Feature Pyramid Network.

G

GNSS Global Navigation Satellite Systems.

I

IOS In-Orbit Servicing.

IoU Intersection over Union.

J

JINS Jins Is Not a Simulator.

L

LEO Low Earth Orbit.

LIDAR Light Detection and Ranging.

LVLH Local-Vertical Local-Horizontal.

M

mAP mean Average Precision.

MRP Modified Rodriguez Parameters.

P

PnP Perspective-n-Point.

R

RSO Resilient Space Object.

S

SLAM Simultaneous Localization and Mapping.

SRF Sensor Reference Frame.

SSD Single Shot Multibox Detector.

U

UKF Unscented Kalman Filter.

Y

YOLO You Only Look Once.

1 | Introduction

Space debris is a growing problem in recent years. The number of launches that are carried out each year is getting bigger, and consequently the number of space debris also increases. Different sources contribute to space debris, with the primary one being the fragmentation of spacecrafts and rocket bodies. This fragmentation often occurs as a result of explosions involving residual fuel or reactive chemicals within rocket engines and fuel tanks used during launches. Additionally, space debris includes inoperable or non-functional satellites. Even without additional launches, the increase of space debris is expected to continue and as the quantity of space debris increases, so does the likelihood of collisions, which grows exponentially. Subsequently, collision fragments further collide, generating smaller debris and amplifying the potential for future collisions. This phenomenon is recognized as the Kessler Syndrome [4]. Currently, this predicament is primarily concentrated in the Low Earth Orbit (LEO) region due to the extensive space activities conducted therein. In 2009, the first accidental in-orbit collision between two satellites occurred. The satellites involved were a privately owned American communication satellite and a Russian military satellite [4]. Two potential solutions mitigate the issue of space debris are Active Debris Removal (ADR) and In-Orbit Servicing (IOS). ADR involves the removal of existing debris from space, and its efficiency can be significantly enhanced by using specific criteria during the selection of debris for removal. The chosen debris should have a large mass, a high probability of collision, and it should be located at a high altitude [6]. An example of an ADR mission is Clearspace-1, in which the objective is to capture and safely bring down a large derelict object, the Vespa upper stage [5], on the other hand IOS refers to extending the life or functionalities of spacecrafts that are already in orbit. This can be done by performing maintenance, adjusting a spacecraft's orbit, changing the direction it is facing, providing more fuel, or even changing or upgrading the onboard instruments [7]. Both ADR and IOS activities require proximity operations to their target objects. Therefore, it is important to have precise information about the relative position and orientation of the target spacecraft, in relation to the chaser. The techniques to collect information about the relative pose are strongly influenced by the level of cooperation of the target. As a result, the capacity to classify the target according to its cooperation

level and the availability of information becomes crucial. This classification will facilitate the selection of the most suitable relative navigation strategy, tailored to the specific characteristics of the target.

1.1. Literature review: Relative Navigation

Relative navigation in space has been a critical aspect of space exploration since the earliest missions. It refers to the process of determining the position and velocity of a spacecraft relative to another object. Over the years, various techniques and technologies have been developed to enable accurate relative navigation, allowing for precise rendezvous, docking and proximity operations in space. Historically, a first example of proximity operations is represented by the Apollo program. Relative navigation played a crucial role, particularly during rendezvous, such as between the Lunar Module and the Command Module [8]. However, this and other missions that came later were performed by the crew on board. The Columbia accident in 2003, more research began toward autonomous on-board operations [45]. The first experiment of unmanned spacecraft was the ETS-VII. The mission consisted of two sub-tasks, which were the autonomous rendezvous/docking and robot experiments [73]. Then, proximity operations around uncooperative satellites were demonstrated by the Experimental Satellites System 10 (XSS-10) and 11 (XSS-11), in which the objectives were the autonomous rendezvous and inspection of multiple space objects [67]. In 2005 NASA's mission DART was launched to demonstrate autonomous proximity operations up to 5 meters to the target [59]. This mission was a partial failure but, in 2007, it still helped in the successful execution of the Orbital Express mission [45]. This mission demonstrates short range and long range autonomous rendezvous, capture and berthing, on-orbit electronics upgrades, on-orbit refueling, and autonomous fly-around visual inspection using a demonstration client satellite [3]. Another example was the PRISMA mission, which was launched in 2010 by the Swedish National Space Board. In this mission the servicer successfully performed semi-autonomous rendezvous maneuver with a non-cooperative target. NASA and the Canadian space agency instead demonstrated the TriDAR for performing relative navigation to an uncooperative but known target [45].

In these missions, terms related to the knowledge and co-operation of the target often recur. These terms are used to classify different types of targets. It is possible to start analyzing targets by considering the degree of cooperativeness they possess. They can be cooperative if they have active or passive equipment that allows them to communicate with the chaser, otherwise they are called uncooperative. In the second case there is a further subdivision with known targets on one side and unknown targets on the other. A

target can be defined as known if prior information about its shape, mass, inertia, and materials is available otherwise it is defined as unknown [71] [45]. In the last decades missions at cooperative targets, or uncooperative but known targets, were already successfully demonstrated using different methods of which a review will be presented in the course of this chapter. The field of relative navigation at uncooperative and unknown targets is more challenging and it will be the focus of this work. The subsequent subsections initially present various sensors employed for acquiring measurements. Subsequently, different relative navigation techniques are introduced based on the level of knowledge and cooperation of the targets.

1.1.1. Sensors

Before analyzing the methods that can be used to study different types of targets according to the level of cooperativeness and knowledge, it is important to present which kind of sensors are available to obtain relative measurements during operations. It is possible to identify three main families of sensors during proximity operations: radio frequency sensors, satellite navigation systems and optical sensors. One of the main drivers in the choice of the sensor is the level of cooperativeness of the target. If the target is actively cooperative, radio frequency sensors can be used to obtain range, range rate, line-of-sight and attitude measurements. However, radio frequency systems are characterized by high mass and power consumption, making them impractical for small satellites with strict mass and power budgets. Alternatively, cooperative satellites with a direct radio link can leverage Global Navigation Satellite Systems (GNSS), which is composed of a space segment and a user. The GNSS can provide information about the relative position and velocity, if the target is equipped with GNSS receivers, and it requires a lower mass and power budgets with respect to the radio frequency sensors. The main drawbacks are that it is not able to provide information about the attitude of the target, and its restricted use within LEO, due to the way the GPS constellation is made up [45].

When dealing with uncooperative targets, electro-optical sensors are currently the main technology for pose (i.e., relative position and relative attitude) determination. They can be either active or passive. The most common active technology is LIDAR, that can measure the distance to the target by illuminating it with an infrared source and then analyzing the back-scattered radiation. The information obtained is used to build a cloud of 3D points of the target that will be used for the pose estimation [48]. It is possible to classify this class of instrument according to the lights source used. Therefore, it is possible to have continuous-wave light source, that are applied in close range missions, and pulsed source, that instead can be used to compute bigger distances [45]. From a

historical point of view, some examples of the use of this technology can be seen in the Trajectory Control Sensor (TCS) on the Space Shuttle, the Videometer (VDM) on ESA's Automated Transfer Vehicle (ATV) and the RendezVous Sensor (RVS)/ Telegoniometer (TGM) on the HTV/ATV [14].

Among the passive optical sensors the most common are the monocular and the stereo camera. They can operate in a wavelength range from the visible to the infra-red. Monocular cameras provide a bi-dimensional information (i.e., the line of sight measurement), so it is possible to determine the relative position of an RSO only up to a scale factor. Conversely, stereo cameras can acquire 3D information about the target just like LIDARs, and typically the point cloud generated by them is much denser than the one of LIDARs. However, the maximum range at which the stereo camera performances is acceptable is limited by the baseline of the stereo configuration. In general, passive sensors are characterized by a lower cost, power consumption and hardware architecture complexity with respect to LIDARs. Despite that, the high sensitivity to the illumination condition and the potential presence of other celestial bodies in their field of view, make the use of these sensor challenging. The sensitivity to the illumination condition can be compensated by adding a light source within the sensor architecture, but it increases the power budget [45]. Three different generations of vision-based systems have been developed and tested. The Proximity Operation Sensor (PXS) was tested in the Engineering Test Satellite Program (ETS-Vii) mission [39]. The Advanced Video Guidance Sensor (AVGS) was used in the Demonstration of Autonomous rendezvous and Docking (DART) and Orbital Express (OE) [31]. The Visual Based System (VBS) was tested during the PRISMA [51] mission. These test demonstrate the ability to estimate the pose accurately [48].

1.1.2. Cooperative Target

Cooperative pose determination algorithms rely on extracting a limited number of features from acquired data-set to estimate the relative attitude and position of a Target Reference Frame (TRF) with respect to a Sensor Reference Frame (SRF). These features correspond to passive or active artificial markers whose 3D position in TRF is known since they are purposely mounted on the target. The pose determination process involves identifying and matching the detected markers with a stored catalogue [48]. In the case of 2D measurements, first of all the extracted features are matched to the real markers, resulting in a 2D-3D point correspondences. The algorithm that is commonly used to perform this task is the Perspective-n-Point (PnP) problem. An alternative to PnP is the Direct Least Square PnP solution [32] or the EPnP problem [40]. EPnP is a non-iterative method that guarantees a closed-form solution, still ensuring good accuracy of the solution. It

will be better described in the Chapter 3. Once the 3D positions of the markers in the SRF are known, the problem becomes similar to having 3D measurements, since it consists of determining the attitude and relative position given a series of 3D to 3D point correspondences [48]. If the number of matched features is 3, the relative rotation matrix can be reconstructed by implementing TRIAD [2]. If the number of features n is larger than three it is possible to distinguish potentially $(n^2 - n)/2$ unit vectors, and they can be used for relative attitude calculation using probabilistic approaches like the QUEST algorithm [2].

1.1.3. Uncooperative and Known Target

This section focuses on the scenario where the uncooperative target is assumed to be known, either through detailed geometry information or at least a simplified geometric model. In such cases, model-based algorithms are used for pose determination. These algorithms are very similar to those used in case of cooperative targets and they estimate the pose vector by matching data obtained from monocular or 3D sensors with corresponding information from the target model stored on-board. This task is accomplished by using feature-based approaches, which rely on the capability of extracting natural geometric features from the acquired data-set. Different techniques can be used depending on the type of available data [48]. The analysis starts by considering the monocular sensors, which provides 2D measurements. In this scenario feature-based algorithms can rely on either PnP solvers or Template Matching (TM) approaches. For PnP solvers the main challenge is finding the correct correspondence between the features from an acquired image and the target model, due to the absence of fiducial markers on the target surface. To solve this problem the SoftPOSIT [18] algorithm is used, which is an iterative but linear method used to estimate the pose. First it uses the soft-assign [30] techniques to determine image/model point correspondences, then the POSIT [21] algorithm is used to iteratively estimate the pose. An alternative solution to the soft-assign algorithm is given by RASCAN [26]. Once the matches have been determined, PnP methods can be used. TM in computer vision is the process of searching a monocular image for specific features and/or image sections which can be matched to an assigned template [48]. The matching is typically done using correlation based approaches, such as the sum of absolute differences [19], the normalized cross correlation [12] or the distance transform [27]. In pose determination, TM requires to generate a data-set of templates by sampling the 6 DOF pose space. Each template is created based on a set of relative position and attitude parameters, and the correlation function is used to evaluate the similarity between each template and the acquired image. The template with the best correlation provides

the unknown pose vector [48]. For monocular images, appearance-based methods provide an alternative to feature-based techniques. They analyze the shape and the texture of the acquired data-set by exploiting either Active Appearance Models (AAM) [15] or the Principle Components Analysis (PCA) [72].

In the last years, approaches have been developed that perform TM with Convolutional Neural Network (CNN) [53] [63]. These methods rely on an initial training phase on an available data-set of real or simulated images, and the trained CNN can be used to determine the pose from the images [20]. The main advantage lies in the reduced execution time for this application compared to traditional TM methods. Chapter 3 will examine the topic of neural networks in more detail, with a particular focus on the field of object detection.

If 3D sensors are considered instead of the 2D, pose acquisition can be achieved through algorithms that rely on global or local 3D feature descriptors. Alternatively to these feature-based approaches it is possible to use point based techniques that directly process 3D raw data, such as range images or point clouds. For pose tracking, it is possible to use also 3D data registration methods. The working principle of these algorithms is to find the best rigid transformation to align two data-sets (registration) by minimizing a given cost function which measures the similarity level between corresponding elements [48]. The Iterative Closest Point (ICP) algorithm is commonly used for this purpose with all its variants [45]. An alternative is represented by the Coherent Point Drift (CPD) algorithm for non-rigid point set registration [47]. This latter method has been generalized in [33] where the Bayesian Coherent Point Drift method is described. In this method the motion coherence theory has been replaced with Bayesian Inference, and so the matches of the points clouds is done in a statistical way, overcoming the issue of finding exact feature correspondences [20].

1.1.4. Uncooperative and Unknown Target

In the context of space application, the focus is often on dealing with unknown objects such as debris or asteroids. One approach to address this challenge is to build a model directly on board of the spacecraft using available measurements. There are two ways to accomplish this task. The first strategy is to build this model during a prior monitoring stage, in which the spacecraft is at a safe distance from the target object. This allows the application of reliable model-based algorithms for subsequent mission phases. The second strategy is to build the model simultaneously with the pose estimation and this is the object of study for Simultaneous Localization and Mapping (SLAM) [22]. SLAM is a real-time process by which an autonomous agent, moving within an unknown environment,

computes its own trajectory (localization) while simultaneously building a map of that environment (mapping) [48]. In recent years, this technique has been analyzed in several ways. The first approach to estimate the dynamic state, geometric shape, and mass model parameters of an unknown target simultaneously is proposed in [22]. This method uses a set of images acquired by a distributed set of 3D sensors, assumed to be uniformly positioned around the target. This method also exploits a Kalman Filter for the pose information. A feature-based SLAM algorithm that uses an Extended Kalman Filter (EKF) is proposed in [64]. The focus of this work is on the filtering aspect, assuming that a set of 3D features detected and tracked by a stereo-vision system is already available as input to the EKF. In [62], the authors combine the EKF-based SLAM filter described in [64] with a RASCAN-based surface reconstruction algorithm to generate a model of the unknown target which can be used for further refinement of the estimated pose. The key issue of these class of SLAM techniques is the matching between the on-board map and measurements. Indeed, filters updates work better with few accurate matched points rather than with many imprecise measurements. Moreover, the computational cost of these methods grows with the size of the map.

Graph-SLAM approaches offer an alternative solution where online navigation estimates are sacrificed in favor of improved mapping of the environment and better convergence [45]. An example of these methods is presented in [66], in which the algorithm uses 3D measurements from a stereo camera to estimate the complete relative state of the target, including its linear and angular velocities, center of mass, principal axes, and diagonal inertia matrix [48]. Another example is presented in [46], in which the ORB-SLAM algorithm is tested on real images.

CoMBiNa is a method for estimating the pose of RSO that was first presented in [45]. This method aims to estimate both the relative position and velocity between chaser and target, as well as the attitude and inertial parameters of the target. Good results are presented in [45] for the case of asymmetrical targets, whereas in the case of symmetry, it is shown that the results worsen considerably. In [20], the author proposes a strategy for dealing with the case where the target presents symmetry. This extension of CoMBiNa involves approximating the symmetric target to a gyroscopic structure and reconstructing only the orientation of the symmetry axis, dispensing with full attitude estimation, showing good results. Although some information on the attitude of the target can be obtained, it cannot yet be fully estimated.

1.2. Thesis Objective

The aim of the first part of this work is to modify the CoMBiNa algorithm to estimate the complete attitude of the symmetric target, rather than just the orientation of the axis of symmetry. This involves integrating CoMBiNa with an operational neural network, capable of supplying the precise locations of specific target features. Subsequently, in the second part of the work, the feasibility of using such a neural network to identify the locations of specific features on the target is demonstrated.

1.3. Thesis Outline

Chapter 2 first introduces the original version of CoMBiNa, providing a detailed analysis of its pipeline. Then, a description of the extension of CoMBiNa to the case where a symmetrical target is detected is provided, showing what modifications are required to obtain an estimate of the position of the spin axis of the target. In Chapter 3, a brief review on CNN is first made. Subsequently, several methods for carrying out object detection are presented, with a brief description, and typical methods for evaluating their performance. Finally, the Efficient Perspective-n-Point (EPnP) is presented, a method used to solve a typical problem in computer vision, the Perspective-n Point (PnP) problem. The new formulation of the CoMBiNa algorithm is presented in Chapter 4, where the different methods for deriving the target attitude are presented in detail. The results achieved through this new formulation are presented toward the end of the chapter. Afterwards, Chapter 5 explains the reasons for choosing YOLOv8 as the neural network in this work. It is also presented the procedure followed to train YOLOv8, including a description of the process for generating the images, a description of the training process, and a presentation of the results obtained. Finally, Chapter 6 contains the final conclusions and hints for possible future work.

2 | Introduction to CoMBiNa

2.1. CoMBiNa: Original Formulation

The Coarse Model-Based Relative Navigation (CoMBina) technique, introduced in [45], is designed for navigating in the proximity of unknown and uncooperative RSO. CoMBiNa carries out its operations through a two-phase approach. The initial phase, known as the preliminary phase, involves placing the chaser into a safe relative orbit and using a SLAM algorithm to construct a coarse model of the target. The second phase, named the operational phase, focuses on estimating relative pose and dynamics. During this phase, the process begins with the analysis of stereo images, which are then matched with the coarse model generated in the preliminary phase using the Bayesian Coherent Point Drift (BCPD) method, a statistical point-set registration technique [33]. Subsequently, the output from this process is input into two Unscented Kalman Filters (UKF) [60], allowing the recovery of both the target's pose and inertia properties. However, in the case of symmetrical targets, the method fails, as these types of targets have unobservable rotations that lead BCPD to converge to wrong solutions. Thus, in [20], a novel formulation of the state vector is proposed specifically for symmetric targets, enabling the acquisition of information regarding the symmetry axis.

2.1.1. Reference Systems

In Table 2.1, the reference systems that will be used and the symbols used to denote them throughout the work are presented.

Symbols	Reference Frame
\mathcal{I}	Earth Centred, Inertial
\mathcal{L}	Chaser Centred, Local-Vertical Local-Horizontal
\mathcal{C}	Chaser Centred, Chaser Fixed
\mathcal{S}	Camera Centred, Camera Fixed
\mathcal{T}	Target Centred, Target Fixed ("real")
$\hat{\mathcal{T}}$	Target Centred, Target Fixed ("estimated")

Table 2.1: Reference frames

2.1.2. Preliminary Phase

During this phase, the chaser remains in a safe orbit around the target to obtain images that will be used in a simultaneous localisation and mapping (SLAM) algorithm. In [45] the images are generated through simulation using Blender and a 3D model of the target RSO available online. Blender is able to render these models by simulating the movement of the camera in proximity to the target. The technique chosen SLAM to construct the preliminary model of the target is ORB-SLAM [46]. This system uses ORB features, which are robust against rotation, scale, and illumination variations. An example of the application of this technique in real-time operation can be found in [52]. After the execution of the ORB-SLAM algorithm, the point cloud is processed to remove the outliers. Firstly, an outlier rejection process is carried out using a k-Nearest Neighbor Graph (k-NNG) [23] approach. Secondly, to further enhance the construction of the model, a probabilistic feature-based space-carving approach [50] is used.

2.1.3. Operational Phase

Stereo Measurements

The initial stage involves extracting features from the images and determining their positions within the chase body frame, enabling the subsequent application of the point cloud registration technique. For optimal efficiency in executing this operation, a region of interest (ROI), containing the target, is identified in the image. Subsequently, SURF features [9] are extracted from this region. Following a further feature reduction process, the 3D coordinates of each feature are acquired in terms of their horizontal and vertical positions (u and v), along with the disparity value (α). It is now necessary to determine the relationship between the camera frame to the chaser body frame.

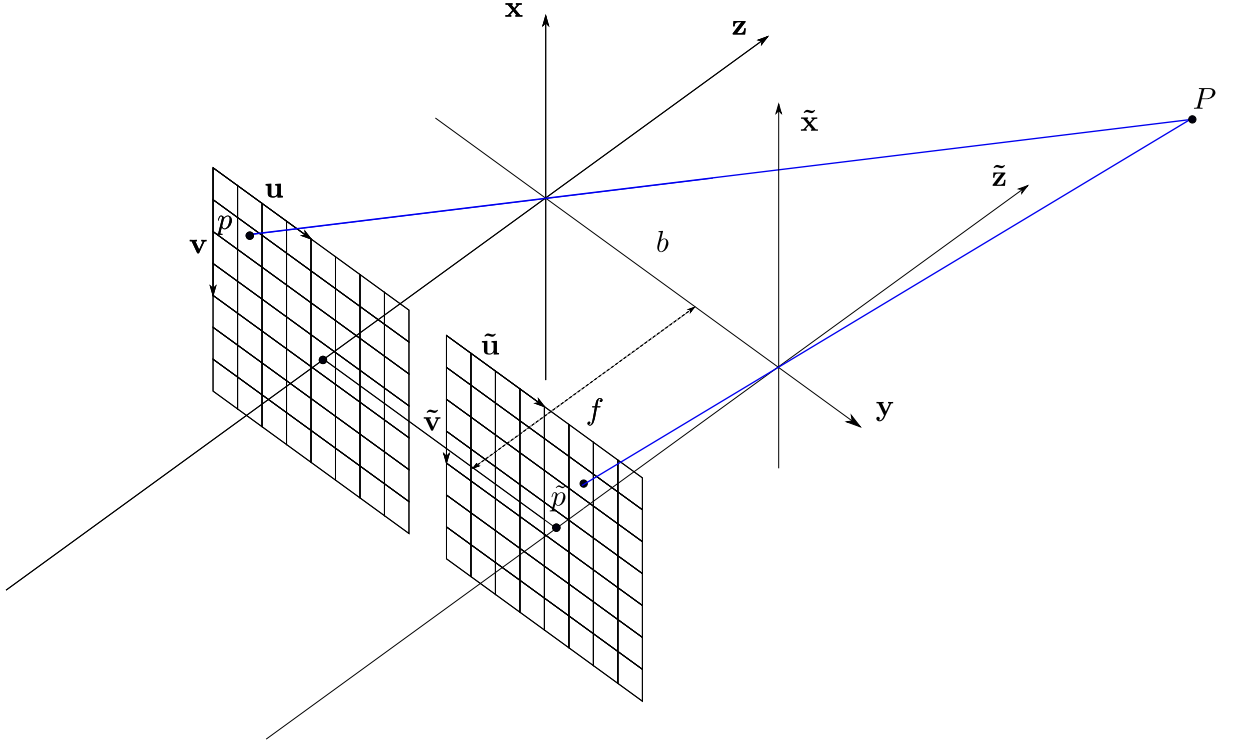


Figure 2.1: Parallel axis stereo camera model adopted in CoMBiNa (courtesy of [45])

In this work, it is assumed that a stereo camera is available, as it allows for a complete reconstruction of the position of the target from the measurements. Unlike a monocular camera, the stereo camera provides information about the range as well. The stereo camera consists of two monocular cameras with the same focal length and pixel density. Both cameras are oriented in the same way, with the only difference that the optical center of one is shifted by an amount b (baseline) in the direction \mathbf{y} . A schematic representation is shown in Fig. 2.1. At this point, it is possible to define the relationship between the points in the chaser system and those expressed in terms of pixel coordinates (u_i, v_i) .

A point in the chaser frame $\mathbf{p}_i^C = (p_x, p_y, p_z)$ can be expressed in terms of pixel coordinates of the first camera using the pinhole camera model as follows:

$$p_i^S = (u_i, v_i) = \left(u_0 - f d \frac{p_y}{p_z}, v_0 + f d \frac{p_x}{p_z} \right) \quad (2.1)$$

The expression of point \mathbf{p}_i^C with respect to the second camera is as follows:

$$p_i^{\tilde{S}} = (\tilde{u}_i, \tilde{v}_i) = \left(\tilde{u}_0 - f d \frac{p_y - b}{p_z}, \tilde{v}_0 + f d \frac{p_x}{p_z} \right) \quad (2.2)$$

(u_0, v_0) and $(\tilde{u}_0, \tilde{v}_0)$ are the coordinates of the optical centers of the first and second cameras, respectively. In this work, it is reasonable to assume that $v_0 = \tilde{v}_0$, while the difference between the horizontal coordinates can be defined as disparity α . Once the 3D coordinates of the i -th feature in terms of u_i , v_i and α_i are known, it is possible to determine the position of the feature in the chaser frame as follows:

$$\mathbf{p}_i^c = \left[-\frac{b}{\alpha_i}(v_i - v_0), \frac{b}{\alpha_i}(u_i - u_0), -\frac{fdb}{\alpha_i} \right]^T \quad (2.3)$$

Once all the extracted features are processed in this way, it is possible to obtain a 3D point cloud in the chaser-fixed, chaser-centered frame, $\mathcal{P}^c = \{\mathbf{p}_1^c, \dots, \mathbf{p}_M^c\} \subset \mathbb{R}^{3 \times M}$, which will be used in the subsequent steps of the operational phase.

Measurements Extraction from BCPD

In the point set registration phase, the goal is to align two distinct point sets: the first set is constructed during the preliminary phase, while the second set is obtained from the stereo measurements acquisition. The selected algorithm is known as Bayesian Coherent Point Drift (BCPD) [33], which is a variation of the Coherent Point Drift (CPD)[47] algorithm. The distinction between these two algorithms lies in their approach to determine points correspondences and the parameters of the transformation between one model and the other. CPD relies on the Expectation-Maximization algorithm, whereas in BCPD, optimization is based on Variational Bayesian Inference (VBI). Equation (2.4) shows the transformation model used in BCPD, which combines a similarity transformation and a non-rigid transformation.

$$\mathcal{T}(\mathbf{y}_k, \boldsymbol{\rho}, \mathbf{v}_k) = s\mathbf{R}(\mathbf{y}_k + \mathbf{v}_k) + \mathbf{t} \quad (2.4)$$

In this equation $s \in \mathbb{R}$ is a scaling factor, $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^{3 \times 1}$ is a translation vector. For simplicity these parameters are collected in a single set of transformation parameters $\boldsymbol{\rho}$. $\mathbf{v}_k \in \mathbb{R}^{3 \times 1}$ is a displacement vector that characterizes the non-rigid transformation.

In [45], BCPD is used to determine the aforementioned parameters. These parameters are used to align the coarse reconstructed model $\mathcal{M}^T = \{\mathbf{m}_1^T, \dots, \mathbf{m}_N^T\} \subset \mathbb{R}^{3 \times N}$, which is represented in the target reference frame, with the point cloud derived from stereo images \mathcal{P}^c , expressed in the chaser-centered, chaser-fixed reference frame. With an initial estimate of the attitude and position of the target from the filter state, it is possible to

project the point cloud \mathcal{P}^c into the estimated target reference frame $\tilde{\mathcal{T}}$ as follows:

$$\mathcal{P}^{\tilde{\mathcal{T}}} = \mathbf{C}_{\tilde{\mathcal{T}}c} \mathcal{P}^c - \mathbf{1}_M \otimes \mathbf{C}_{\tilde{\mathcal{T}}\mathcal{L}} \hat{\mathbf{r}} \quad (2.5)$$

In (2.5) $\hat{\mathbf{r}}$ represents the estimated relative position of the center of mass of the target with respect to the chaser, expressed in LVLH reference frame, while $\mathbf{C}_{\tilde{\mathcal{T}}c}$ and $\mathbf{C}_{\tilde{\mathcal{T}}\mathcal{L}}$ represent the rotation matrices from the chaser frame and LVLH frame respectively, to the estimated target frame. Having the coarse model of target \mathcal{M}^T available, it is possible to apply BCPD to these two models to determine the parameters of the transformation responsible for registering the measured point cloud with the coarse model. The transformation between the two point clouds is given by the following equation:

$$\mathcal{P}^T = s \mathbf{R} \mathcal{P}^{\tilde{\mathcal{T}}} + \mathbf{1} \otimes \mathbf{t} \quad (2.6)$$

where s is the scaling factor, \mathbf{R} is the rotation matrix and \mathbf{t} is the translation. The scaling factor should be very close to 1, indicating that the measurements have not altered the shape of the target. If this value deviates significantly from 1, the filter will reject the measurement. The rotation matrix \mathbf{R} is responsible for aligning the estimated target reference frame $\tilde{\mathcal{T}}$ with the real target reference frame \mathcal{T} , so it can also be expressed as $\mathbf{C}_{\mathcal{T}\tilde{\mathcal{T}}}$. Consequently, the rotation matrix from the inertial system to the target system can be derived as follows:

$$\mathbf{C}_{\mathcal{T}\mathcal{I}} = \mathbf{C}_{\mathcal{T}\tilde{\mathcal{T}}} \mathbf{C}_{\tilde{\mathcal{T}}\mathcal{I}} \quad (2.7)$$

It is then possible to derive from the $\mathbf{C}_{\mathcal{T}\mathcal{I}}$ matrix the set of Modified Rodriguez Parameters (MRPs) with norm $\|\boldsymbol{\sigma}_{\mathcal{T}}\| < 1$ describing the orientation of the target in the inertial reference system as follows [35]:

$$[\tilde{\boldsymbol{\sigma}}_{\mathcal{T}}] = \frac{\mathbf{C}_{\mathcal{T}\mathcal{I}}^T \mathbf{C}_{\mathcal{T}\mathcal{I}}}{\sqrt{\text{Tr}(\mathbf{C}_{\mathcal{T}\mathcal{I}}) + 1} (\sqrt{\text{Tr}(\mathbf{C}_{\mathcal{T}\mathcal{I}}) + 1} + 1)} \quad (2.8)$$

This equation encounters a singularity at an equivalent principal rotation of 180° , when both numerator and denominator become 0. Although this condition is very rare, it is possible to solve this issue by solving the following set of equations [34]:

$$\boldsymbol{\sigma}_{\mathcal{T}} \boldsymbol{\sigma}_{\mathcal{T}}^T = \frac{1}{2} (\mathbf{I} + \mathbf{C}_{\mathcal{T}\mathcal{I}}) \quad (2.9)$$

The second of information derived from BCPD concerns the position of the target in relation to the chaser. The position \mathbf{r} is expressed in the LVLH reference system while the translation term \mathbf{t} obtained from BCPD is expressed in the target reference frame. Consequently, it must be expressed in the LVLH system, and therefore the expression for obtaining \mathbf{r} is:

$$\mathbf{r} = \hat{\mathbf{r}} - \mathbf{C}_{\mathcal{L}\hat{\mathcal{T}}}\mathbf{C}_{\mathcal{T}\hat{\mathcal{T}}}^T\mathbf{t} \quad (2.10)$$

Summarising, the measurement vectors obtained from BCPD are:

$$\mathbf{z}_s = \mathbf{r} \quad \mathbf{z}_a = \boldsymbol{\sigma}_{\mathcal{T}} \quad (2.11)$$

These two vectors will be used in the update phase of the translational and rotational filter respectively.

Definition of the State Vector & Equation of Motion

Before delving into the two filters, it is essential to introduce the state vectors that will be provided to the filters, along with their respective equations of motion. Regarding the translational filter, its state vector $\hat{\mathbf{x}}_s$ includes the relative position $\hat{\mathbf{r}}$ and velocity $\hat{\mathbf{v}}$ expressed in the LVLH reference system. The dynamics of this state vector are defined in equation (2.13) and is described by the Nonlinear Equations of Relative Motion [61].

$$\hat{\mathbf{x}}_s = \begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \end{bmatrix} \quad (2.12)$$

$$\begin{aligned} \ddot{x} &= 2n\dot{y} + n^2x - \frac{\mu(R_t + x)}{[(R_t + x)^2 + y^2 + z^2]^{\frac{3}{2}}} + \frac{\mu}{R_t^2} \\ \ddot{y} &= -2n\dot{x} + n^2y - \frac{\mu y}{[(R_t + x)^2 + y^2 + z^2]^{\frac{3}{2}}} \\ \ddot{z} &= -\frac{\mu z}{[(R_t + x)^2 + y^2 + z^2]^{\frac{3}{2}}} \end{aligned} \quad (2.13)$$

In [45], it is assumed that the chaser does not perform any kind of translational manoeuvre and, for this reason, control accelerations are not included in the equations of motion.

Concerning the rotational filter, the state vector is composed by the Modified Rodriguez

Parameters $\hat{\boldsymbol{\sigma}}_{\hat{\tau}}$ [38, 61] of the target, the angular velocity $\hat{\boldsymbol{\omega}}_{\hat{\tau}}$ of the target and a set of two parameters $\hat{\mathbf{k}} = [\hat{k}_1, \hat{k}_2]$, that parameterise the inertia matrix of the target. Indeed, in the absence of external moments, the inertia matrix can be reconstructed as follows [66]:

$$\mathcal{J}_t = \begin{bmatrix} e^{\hat{k}_1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & e^{-\hat{k}_2} \end{bmatrix}, \quad \hat{k}_1 = \ln \frac{A}{B}, \quad \hat{k}_2 = \ln \frac{C}{B} \quad (2.14)$$

where A , B and C are the principal inertia moments. Therefore, the attitude state vector is:

$$\hat{\mathbf{x}}_a = \begin{bmatrix} \hat{\boldsymbol{\sigma}}_{\hat{\tau}} \\ \hat{\boldsymbol{\omega}}_{\hat{\tau}} \\ \hat{\mathbf{k}} \end{bmatrix} \quad (2.15)$$

The first three equations used for the propagation are provided in equation (2.16), which refers to the kinematics of the MRPs ([61], Chapter 3). Equations addressing the rotational motion of the target are presented in equation (2.17) ([61], Chapter 4). The two inertial terms, \hat{k}_1 and \hat{k}_2 , are assumed to be constant over the time.

$$\dot{\boldsymbol{\sigma}}_{\hat{\tau}} = \frac{1}{4} [(1 - \sigma_{\hat{\tau}}^2) \mathbf{I} + 2[\tilde{\boldsymbol{\sigma}}_{\hat{\tau}}] + 2\boldsymbol{\sigma}_{\hat{\tau}}\boldsymbol{\sigma}_{\hat{\tau}}^T] \boldsymbol{\omega}_{\hat{\tau}} \quad (2.16)$$

$$\mathcal{J}_t \dot{\boldsymbol{\omega}}_{\hat{\tau}} + [\tilde{\boldsymbol{\omega}}_{\hat{\tau}}] \mathcal{J}_t \boldsymbol{\omega}_{\hat{\tau}} = 0 \quad (2.17)$$

By leveraging this attitude parameterization, a director cosine matrix describing the rotation from the inertial frame \mathcal{I} to a general body frame \mathcal{B} can be computed as:

$$\mathbf{C}_{\mathcal{BI}} = \mathbf{I} + \frac{8[\tilde{\boldsymbol{\sigma}}][\tilde{\boldsymbol{\sigma}}] - 4(1 - \sigma^2)[\tilde{\boldsymbol{\sigma}}]}{(1 + \sigma^2)^2} \quad (2.18)$$

As the attitude approaches a 360° principal rotation, the norm of the MRP goes to infinity. Two sets of MRPs are used to solve this problem, the nominal $\boldsymbol{\sigma}$ and the shadow $\boldsymbol{\sigma}^S$ set. When $\|\boldsymbol{\sigma}\| = 1$ the switch between the two parameter sets occurs according to the following relationship:

$$\sigma^S = -\frac{\sigma}{\sigma^2} \quad (2.19)$$

Navigation Filters

Once the measurements have been obtained, two different filters are used, one for relative dynamics and one for attitude. The filters are used not only to reconstruct the state variables, but also to filter out noisy measurements [16]. Standard filters are based on two main steps. The first step (prediction) involves using the equations of motion to predict the mean and covariance of the state vector and the measurements at time step $k+1$. During the second step (update) the measurements are used to correct the predicted state vector.

Assume the equation of motion and the measurements equation are:

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k), \mathbf{u}(k), k] + \mathbf{v}(k) \quad (2.20)$$

$$\mathbf{z}(k+1) = \mathbf{h}[\mathbf{z}(k), \mathbf{u}(k), k] + \mathbf{w}(k) \quad (2.21)$$

where $\mathbf{f}[\cdot; \cdot; \cdot]$ is the process model, $\mathbf{x}(k)$ is the state vector at the time step k , $\mathbf{u}(k)$ is the vector of inputs and $\mathbf{v}(k)$ is the process noise perturbing the state, $\mathbf{z}(k+1)$ is the measurements at the time step $k+1$, $\mathbf{h}[\cdot; \cdot; \cdot]$ is the measurements function and $\mathbf{w}(k)$ is the measurements noise. \mathbf{v} and \mathbf{w} are assumed uncorrelated multivariate Gaussian processes with zero mean and $\mathbf{Q}(k)$, $\mathbf{R}(k)$ covariances respectively.

In [45], the Unscented Kalman Filter (UKF) [68], which relies on the Unscented Transformation (UT), was preferred over the commonly the Extended Kalman Filter (EKF). This choice was driven by the fact that the EKF is dependent on the linearization of the process around the current state. The accuracy of this linearization diminishes as the state becomes more nonlinear or when the measurement frequency is low.

The first UKF that will be presented is the one dedicated to the estimation of the relative dynamics (i.e., translational filter). First, the sigma points \mathcal{X} (2.22) and the weights (2.23) for the state at time step k are determined, using the mean $\hat{\mathbf{x}}_s(k|k)$ and the covariance $\mathbf{P}_s(k|k)$.

$$\begin{aligned}
\boldsymbol{\mathcal{X}}_s^{[0]}(k|k) &= \hat{\boldsymbol{x}}_s(k|k) \\
\boldsymbol{\mathcal{X}}_s^{[i]}(k|k) &= \hat{\boldsymbol{x}}_s(k|k) + \left(\sqrt{(n+\tau)\mathbf{P}_s(k|k)} \right)_i \quad i = 1, \dots, n \\
\boldsymbol{\mathcal{X}}_s^{[i+n]}(k|k) &= \hat{\boldsymbol{x}}_s(k|k) + \left(\sqrt{(n+\tau)\mathbf{P}_s(k|k)} \right)_{i+n} \quad i = 1, \dots, n
\end{aligned} \tag{2.22}$$

$$\begin{aligned}
\Omega_m^{[0]} &= \frac{\tau}{\tau+n} \\
\Omega_P^{[0]} &= \Omega_m^{[0]} - (1 - \gamma^2 + \beta) \\
\Omega_m^{[i]} &= \Omega_P^{[i]} = \frac{1}{2(\tau+n)} \quad i = 1, \dots, 2n
\end{aligned} \tag{2.23}$$

In (2.23) n indicates the number of states, whereas the subscript m is used to indicate the weights relative to the mean value, finally, P is used for those relative to the covariance. τ , β and γ are tuning parameters of the filter [68].

It is now possible to propagate the sigma points $\boldsymbol{\mathcal{X}}_s$ through the system dynamics (2.13), resulting in the sigma points at time step $k+1$. Consequently, this allows for the computation of the mean value of the state and its corresponding covariance as follows:

$$\hat{\boldsymbol{x}}_s(k+1|k) = \sum_{i=0}^{2n} \Omega_m^{[i]} \boldsymbol{\mathcal{X}}_s^{[i]}(k+1|k) \tag{2.24}$$

$$\begin{aligned}
\mathbf{P}_s(k+1|k) &= \sum_{i=0}^{2n} \Omega_P^{[i]} \left(\boldsymbol{\mathcal{X}}_s^{[i]}(k+1|k) - \hat{\boldsymbol{x}}_s(k+1|k) \right) \cdot \\
&\quad \cdot \left(\boldsymbol{\mathcal{X}}_s^{[i]}(k+1|k) - \hat{\boldsymbol{x}}_s(k+1|k) \right)^T
\end{aligned} \tag{2.25}$$

The sigma points are also passed through the measurements function to compute the predicted measurements $\boldsymbol{\mathcal{Z}}_s^{[i]}(k+1|k)$ for each sigma points.

$$\boldsymbol{\mathcal{Z}}_s^{[i]}(k+1|k) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \boldsymbol{\mathcal{X}}_s^{[i]}(k+1|k) \tag{2.26}$$

Now, mean expected measurements $\hat{\boldsymbol{z}}_s(k+1|k)$, measurements covariance $\mathbf{P}_{z,s}(k+1|k)$ and state measurements cross covariance $\mathbf{P}_{sz,s}(k+1|k)$ can be computed. These quantities are retrieved with the following equations:

$$\hat{\mathbf{z}}_s(k+1|k) = \sum_{i=0}^{2n} \Omega_m^{[i]} \mathbf{z}_s^{[i]}(k+1|k) \quad (2.27)$$

$$\begin{aligned} \mathbf{P}_{z,s}(k+1|k) &= \sum_{i=0}^{2n} \Omega_P^{[i]} \left(\mathbf{z}_s^{[i]}(k+1|k) - \hat{\mathbf{z}}_s(k+1|k) \right) \cdot \\ &\quad \cdot \left(\mathbf{z}_s^{[i]}(k+1|k) - \hat{\mathbf{z}}_s(k+1|k) \right)^T \end{aligned} \quad (2.28)$$

$$\begin{aligned} \mathbf{P}_{sz,s}(k+1|k) &= \sum_{i=0}^{2n} \Omega_P^{[i]} \left(\boldsymbol{\chi}_s^{[i]}(k+1|k) - \hat{\mathbf{x}}_s(k+1|k) \right) \cdot \\ &\quad \cdot \left(\mathbf{z}_s^{[i]}(k+1|k) - \hat{\mathbf{z}}_s(k+1|k) \right)^T \end{aligned} \quad (2.29)$$

Before applying the correction step one last check to reject outliers is performed. Indeed, if the scale factor s obtained from BCPD deviates too much from unity, the measurement is rejected, and the update step does not take place. Additionally, if the measurement obtained from BCPD has less than 99 % probability of being correct under its expected mean and covariance, then also in this case the update step is skipped. This particular condition is checked with the squared Mahalanobis distance, which can be computed as:

$$\text{MD}^2 = (\mathbf{z}_s(k+1) - \mathbf{z}_s(\hat{k}+1)) \mathbf{P}_{z,s}^{-1} (\mathbf{z}_s(k+1) - \mathbf{z}_s(\hat{k}+1))^T \quad (2.30)$$

and the threshold value is given by the inverse of the cumulative chi-square distribution function.

If the measurement is not rejected, the Kalman gain $\mathbf{K}(k+1)$ is computed (2.31):

$$\mathbf{K}(k+1) = \mathbf{P}_{sz,s}(k+1|k) \mathbf{P}_{z,s}^{-1}(k+1|k) \quad (2.31)$$

Finally, it is possible to update the state and covariance:

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{K}(k+1) (\mathbf{z}_s(k+1) - \hat{\mathbf{z}}_s(k+1|k)) \quad (2.32)$$

$$\mathbf{P}_s(k+1|k+1) = \mathbf{P}_s(k+1|k) - \mathbf{K}(k+1) \mathbf{P}_{z,s}(k+1|k) \mathbf{K}^T(k+1) \quad (2.33)$$

The second filter processes the attitude motion. The structure proposed for the first

filter is also repeated for the second filter. However, since the state of the second filter consists of $\hat{\sigma}_{\hat{\tau}}$, $\hat{\omega}_{\hat{\tau}}$ and \hat{k} , the sigma points calculated using (2.22)-(2.23) will be propagated by integrating the set of ODEs (2.16)-(2.17). The computation of the mean and the covariance after the propagation remains unchanged. All sigma points are then passed through the measurement equation, which now is:

$$\mathbf{z}_s^{[i]}(k+1|k) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3 \times 5} \\ \mathbf{0}_{5 \times 3} & \mathbf{0}_5 \end{bmatrix} \mathbf{x}_s^{[i]}(k+1|k) \quad (2.34)$$

The remaining steps of the procedure are the same as for the first filter.

Finally, an additional step is introduced for the rotational filter to ensure that the MRPs do not encounter a singularity. When the estimated norm of the MRPs reaches the unit value after the corrective step, the switch to the shadow set takes place. In this case, it is also necessary to transform the covariance matrix as follows [38]:

$$\mathbf{T} = 2 \frac{1}{\|\hat{\sigma}_{\hat{\tau}}\|^4} - 4 \hat{\sigma}_{\hat{\tau}} \hat{\sigma}_{\hat{\tau}}^T - \frac{1}{\|\hat{\sigma}_{\hat{\tau}}\|^2} \mathbf{I} \quad (2.35)$$

$$\mathbf{P}_{z,a}^S(k+1|k+1) = \begin{bmatrix} \mathbf{T} & \mathbf{0}_{3 \times 5} \\ \mathbf{0}_{5 \times 3} & \mathbf{I}_5 \end{bmatrix} \mathbf{P}_{z,a}(k+1|k+1) \begin{bmatrix} \mathbf{T} & \mathbf{0}_{3 \times 5} \\ \mathbf{0}_{5 \times 3} & \mathbf{I}_5 \end{bmatrix}^T$$

2.2. CoMBiNa: Symmetric Target

In [45], CoMBiNa is tested with both asymmetric and symmetric targets. However, with the original formulation of CoMBiNa, a problem arises when dealing with symmetrical targets. This problem comes from the fact that symmetrical targets have unobservable rotations, leading the Bayesian Coherent Point Drift (BCPD) algorithm to potentially converge to solutions where the two point cloud models have different rotations about the symmetry axis. To address this issue, in [20] an extension of CoMBiNa to the symmetrical case is presented. In this case, gyroscope theory is applied to the symmetrical body, allowing the estimation of the orientation of the symmetry axis of the target without the need to estimate the unobservable attitude and inertia properties. The pipeline of the operational phase of the CoMBiNa extension is shown in figure 2.2.

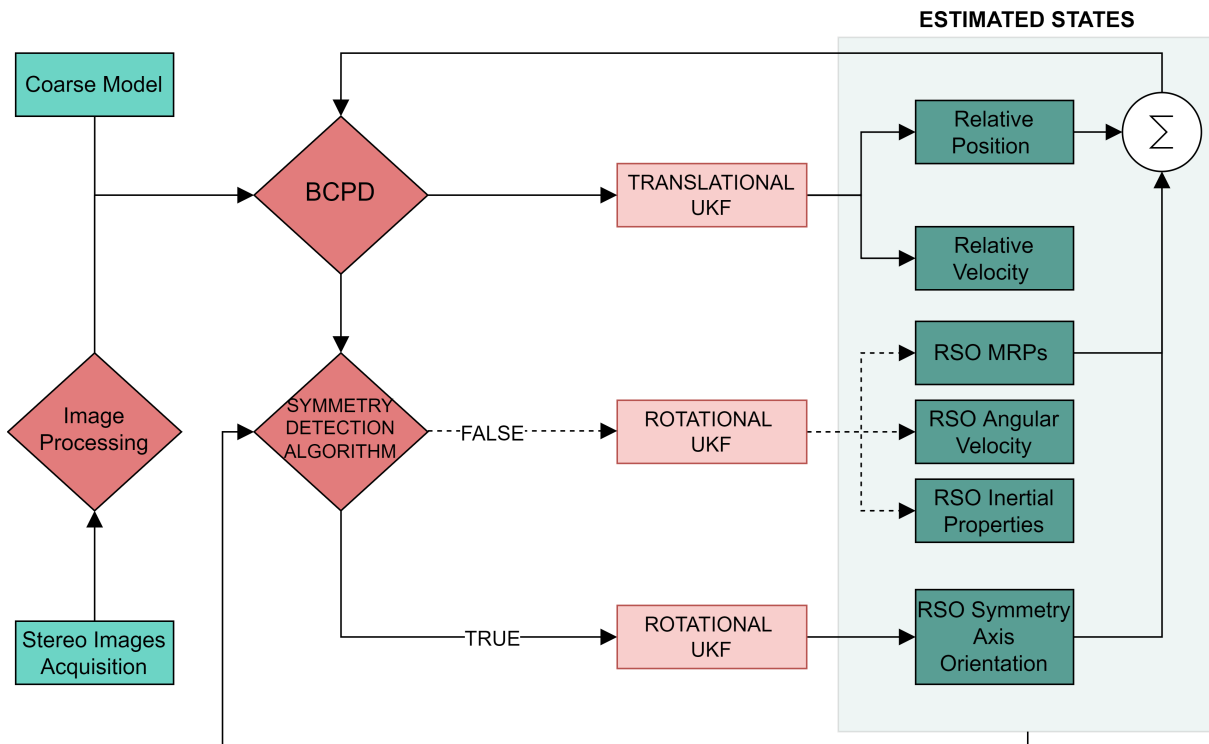


Figure 2.2: Extended CoMBiNa operational phase (courtesy of [20])

A rocket body was used as a symmetrical target, represented in Fig. 2.3.

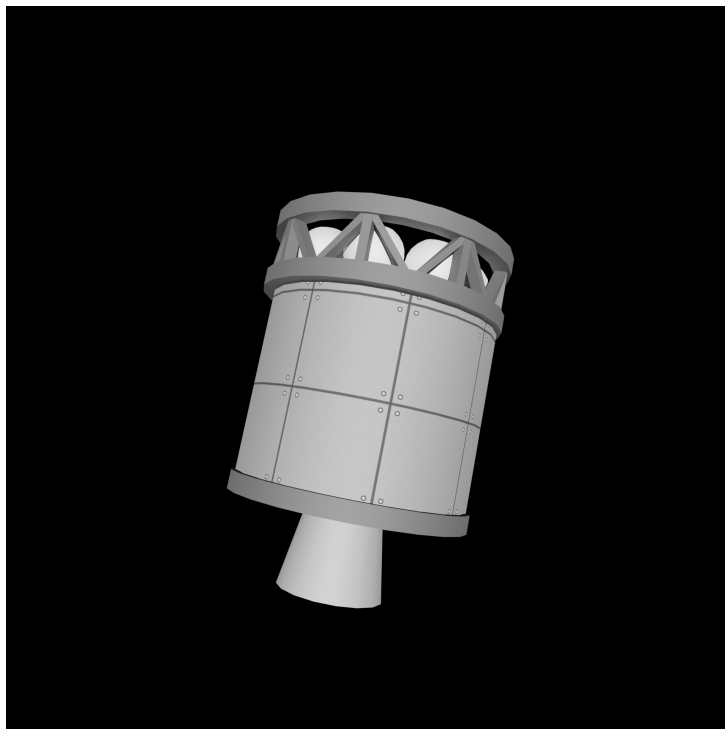


Figure 2.3: Rendering of the 3D symmetric body (courtesy of [45])

The structure of the algorithm is very similar to the original CoMBiNa. The operational phase starts with the acquisition of stereo images, which are then processed to obtain 3D point clouds. The point clouds are given as input to an operation module, which uses BCPD as a point set registration method to match them with the available coarse model, providing a set of measurements. These measurements are checked for consistency and treated in two UKF (translational and rotational). The main difference with respect to the original formulation lies in the definition of the quantities to be processed in the UKF. To change the definition of these quantities, the symmetrical RSO is approximated to a gyroscopic structure. The goal, is to be able to reconstruct the orientation of the spin axis \mathbf{J} of the RSO.

The ODEs presented in Chapter 3 of [20] allows to reconstruct the spin axis \mathbf{J} in a particular inertial system, \mathcal{I}_Γ , which has one of the axes aligned with the angular momentum $\mathbf{\Gamma}_{(0)}$ of the gyroscope. Therefore, a rigid rotation matrix to pass from \mathcal{I}_Γ to \mathcal{I} is introduced. In an initial analysis, the rigid rotation matrix is assumed to be known, while later this assumption is removed, and it is considered as an unknown.

2.2.1. Measurements Extraction

As already seen BCPD allows for the estimation of three terms: scaling factor, translation \mathbf{t} , and rotation matrix \mathbf{R} . The scaling factor should ideally be 1, indicating no change in size. The rotation matrix aligns the estimated target frame with the coarse model reference frame. In the symmetrical case, only the second row of this rotation matrix can be used, since it provides the coordinates of the symmetry axis \mathbf{J} . The first and third rows, due to the geometry of the target, provide a random orientation of the other two axes, and so they are discarded. Regarding the measurements of the expected position, it is correct in the symmetrical case and can therefore be used.

In conclusion, the measurement vectors obtained with BCPD are:

$$\mathbf{z}_s = \mathbf{r} \quad \mathbf{z}_a = \mathbf{J}|_{\mathcal{I}} \quad (2.36)$$

They represent the relative position in LVLH frame and the direction of gyroscopic axis in the inertial frame \mathcal{I} ; they will be used in the update phase of the translational and rotational filter respectively.

2.2.2. Filter Structure for Known Rigid Rotation Matrix

In the symmetrical case the state vector $\hat{\mathbf{x}}_s$ remains the same as in the asymmetrical case, while $\hat{\mathbf{x}}_a$ changes. Equation (2.37) shows the state vectors of the asymmetric case.

$$\hat{\mathbf{x}}_s = \begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \end{bmatrix} \quad \hat{\mathbf{x}}_a = \begin{bmatrix} \hat{\boldsymbol{\sigma}} \\ \hat{\boldsymbol{\omega}} \\ \hat{k}_1 \end{bmatrix} \quad (2.37)$$

In the symmetrical case, the MRPs describing the attitude of the target $\hat{\boldsymbol{\sigma}}_T$ are replaced with the three components of the symmetry axis \mathbf{J} expressed in the inertial reference frame \mathcal{I} . Then, in order to make the state observable, the angular velocity of the target $\hat{\boldsymbol{\omega}}_T$ is removed from the state itself. A more detailed analysis on the observability of the state is carried out in [20], where the procedure presented in [66] is adopted to carry out this analysis.

The new formulation of the state vector $\hat{\mathbf{x}}_a$ is the following:

$$\hat{\mathbf{x}}_a = \begin{bmatrix} \hat{J}_x \\ \hat{J}_y \\ \hat{J}_z \\ \frac{\hat{\Gamma}_{(0)}}{A} \end{bmatrix} \quad (2.38)$$

with equations of motion:

$$\begin{cases} \dot{J}_x = \frac{\Gamma_{(0)}}{A} J_z \\ \dot{J}_y = 0 \\ \dot{J}_z = -\frac{\Gamma_{(0)}}{A} J_x \\ \frac{d}{dt} \left(\frac{\Gamma_{(0)}}{A} \right) = 0 \end{cases} \quad (2.39)$$

Finally, since \mathbf{J} is a unit vector, it can be expressed in spherical coordinates as follows:

$$\begin{cases} J_x = r \cos \delta \cos \phi \\ J_y = r \cos \delta \sin \phi \\ J_z = r \sin \delta \end{cases} \quad (2.40)$$

and due to the fact that r has unit value, the state dimension can be further reduced, leading to the new formulation of the two state vectors, $\hat{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_a$ as follows:

$$\hat{\mathbf{x}}_s = \begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \end{bmatrix} \quad \hat{\mathbf{x}}_a = \begin{bmatrix} \hat{\phi} \\ \hat{\delta} \\ \frac{\hat{\Gamma}_{(0)}}{A} \end{bmatrix} \quad (2.41)$$

Therefore, it is not possible to compute directly the angular velocity of the target, but two different quantities can be computed a posteriori. The first is the absolute value of the equatorial component of the angular velocity, which turns out to be:

$$\|\mathbf{e}\| = \frac{\Gamma_{(0)}}{A} \sqrt{-\sin^2 \phi \cos^2 \delta + 1} \quad (2.42)$$

The second quantity is the square of the ratio between the angular velocity about the symmetry axis and the moment of inertia A about the other two axes, and it can be computed as follows:

$$\left(\frac{\omega_y}{A}\right)^2 = \left(\frac{\Gamma_{(0)}}{A}\right)^2 \sin^2 \phi \cos^2 \delta \quad (2.43)$$

Both of these quantities are constant over time due to gyroscopic properties of the body.

2.2.3. Filter Structure for Unknown Rigid Rotation Matrix

Equation (2.41) represents the two states, $\hat{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_a$, under the assumption of knowing the rigid rotation matrix between the general inertial system \mathcal{I} and the inertial system \mathcal{I}_Γ with an axis aligned with the axis of symmetry of the target. It is now possible to remove this assumption, and the rigid rotation matrix can be estimated using the filter. Therefore, the MRPs of the rigid rotation matrix $\hat{\boldsymbol{\sigma}}_{\mathcal{I}_\Gamma \mathcal{I}}$ are added to the state $\hat{\mathbf{x}}_a$, resulting in the following state vectors:

$$\hat{\mathbf{x}}_s = \begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \end{bmatrix} \quad \hat{\mathbf{x}}_a = \begin{bmatrix} \hat{\phi} \\ \hat{\delta} \\ \frac{\hat{\Gamma}_{(0)}}{A} \\ \hat{\boldsymbol{\sigma}}_{\mathcal{I}_\Gamma \mathcal{I}} \end{bmatrix} \quad (2.44)$$

Since $\hat{\boldsymbol{\sigma}}_{\mathcal{I}_\Gamma \mathcal{I}}$ represents a rotation between two inertial frames, its components do not change over time since the direction of the angular momentum remains fixed in an inertial frame in absence of external disturbances. Therefore, their equations of motion will be given by:

$$\begin{cases} \dot{\sigma}_{\mathcal{I}_\Gamma \mathcal{I},1} = 0 \\ \dot{\sigma}_{\mathcal{I}_\Gamma \mathcal{I},2} = 0 \\ \dot{\sigma}_{\mathcal{I}_\Gamma \mathcal{I},3} = 0 \end{cases} \quad (2.45)$$

3 | Computer Vision

The recent massive development of Artificial Intelligence (AI) application technologies has led to its widespread use in all fields of engineering and beyond. Machine Learning is a category of research and algorithms focused on finding patterns in data and using those patterns to make predictions. Machine Learning falls under the AI umbrella, which in turn intersects the broader field of knowledge discovery and data mining. It can be very difficult to extract high-level abstract features from raw data. Deep learning solves this central problem by exploiting a set of nonlinear cascaded units to extract characteristics and identify multiple levels of abstractions, called Neural Networks. Deep Neural Networks are often chosen as a tool to perform computer vision tasks. In particular, they can be leveraged to gain high level understanding of the environment from digital images or videos. The most important tasks of the computer vision are [25]:

- **Image classification:** the algorithm must recognise which kind of object is represented in the image.
- **Semantic Segmentation:** the algorithm is able to label each pixel in an image with a category, indicating what object or class it belongs to. However, it is not possible to distinguish objects belonging to the same category.
- **Object detection:** The algorithm is able to distinguish several objects in the same image, and identifies them with a bounding box.
- **Instance Segmentation:** it is a mix between object detection and semantic segmentation. The algorithm does what is done in semantic segmentation, but is also able to distinguish between different instances of the same type.

In this work, the aim is to train a neural network that is capable of performing the object detection task on images. For this reason, this task is analysed more in details in the following section, showing the current state-of-the-art for object detection methods.

3.1. Object Detection

The goal of the object detection is to detect all instances of the predefined classes, i.e. categories of objects we are interested in, and provide its coarse localization in the image by axis-aligned boxes. The detector should be able to identify all instances of the object classes and draw bounding box around it.

The methods used for object detection can be divided into three categories: region proposal (two-stage), one-stage and transformer based detectors. Region proposal detectors have a separate module for generating region proposals. These models try to find an arbitrary number of objects proposals in an image during the first stage and then classify and refine their localization in the second. One-stage detectors classify and localize semantic objects in a single shot using dense sampling [74]. The transformer-based detector category is less common, so it will not be discussed in detail in the following sections. Detection Transformer (DeTR) [13] and Swin transformer [44] are among the main architectures.

3.1.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specially designed networks optimized for processing image data. They drive the data-driven classification and also handle feature extraction. The CNN's architecture consists of multiple layers, and once the image is transformed into a vector shape, it is inputted into a traditional neural network for classification. A convolution is a linear transformation that employs filters to process the input. This is achieved by summing a set of parameters (the filter/kernel) with a segment of input of the same dimension, resulting in a scalar value. This operation is carried out at each pixel of the input image. Filters are learned from data, and maintaining their consistency is crucial.

The primary types of layers within a CNN are as follows:

- Convolutional layers: these layers blend all input components, producing a linear combination of values within a region of the input, considering all channels. This means that the activation map generated results from the outputs of neurons that are locally connected to the input through weights corresponding to the filter's parameters.
- Activation layers: these introduce non-linearities into the network to differentiate CNNs from linear classifiers. They are scalar functions that operate on each value of the volume, preserving its size.

- Pooling layers: these reduce the spatial dimensions of the volume while keeping the depth unchanged. A global average pooling (GAP) layer is a type of pooling layer used to reduce the spatial dimensions of a feature map. It works by calculating the average of all values in the feature map, producing a single output value for each feature map. GAP layers are often employed in CNNs to reduce the number of parameters, simplify the network's computational complexity, and enhance model generalization.
- Dense Layers: these contain the CNN network's learned features, and they are responsible for the final classification. Spatial dimensions are lost, and the CNN stacks the hidden layers. They are referred to as "dense" because each output neuron is connected to each input neuron.

CNNs are mentioned in this work because they are used in various object detection techniques. Their functioning involves assigning a fixed set of categories to an input image containing an unknown and variable number of instances, and then drawing bounding boxes around each object instance.

Fig. 3.1 shows the schematic architecture of the CNN.

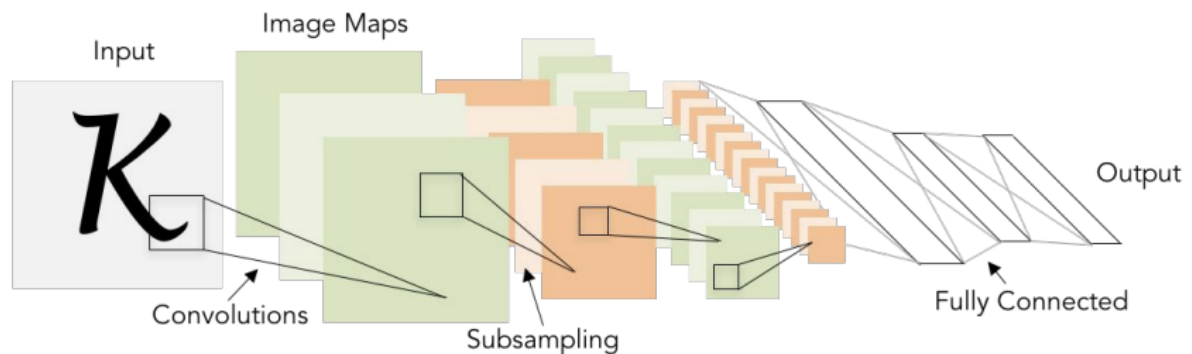


Figure 3.1: CNN architecture (courtesy of [25])

3.1.2. Region Proposal Methods

The main region proposal methods are presented in this section [53] [24] [74].

Region-based CNN (R-CNN)

This algorithm [29] is based on three steps. First, selective search is used to find the regions of the image where there is a higher probability of finding objects. Then the detected regions are scaled to a fixed size and are then given as input to a CNN to extract

the feature vectors. Finally, the feature vectors are passed into Support Vector Machine (SVM) to obtain the confidence scores.

The algorithm is computationally inefficient, since the selective search repeats same operation multiple times, making it unusable for real-time execution.

Fast R-CNN

Using Fast R-CNN [28], the image is first given to a CNN to obtain a feature map, then the proposal regions are extracted with selective search and a second activation map is produced. Finally, the activation map is passed to a single fully connected layer to obtain bounding boxes. The improvement of this method over the previous ones lies in the fact that selective search is no longer applied on the entire image, but only on the regions provided by the first CNN.

Faster R-CNN

Despite the improvements brought by Fast R-CNN, the selective search algorithm used to extract candidate regions represents a computational bottleneck. With Faster R-CNN [58], a fully convolutional Region Proposal Network (RPN) is introduced to extract candidate regions instead of the selective search, decreasing the computational time by one order of magnitude.

FPN

Feature Pyramid Network (FPN) [42], has a top-down architecture with lateral connections to construct high-level semantic features at various scales. FPN comprises two pathways: a bottom-up pathway that uses ConvNet to compute feature hierarchies at multiple scales and a top-down pathway that upsamples coarse feature maps to high-resolution features. Lateral connections, established through a 1x1 convolution operation, enhance semantic information in the features.

R-FCN

The Region-based Fully Convolutional Network (R-FCN) [17] is a novel approach that differs from traditional two-stage detectors. Instead of applying resource-intensive techniques to each proposal, R-FCN shares most computations within the network. It replaces fully connected layers with convolutional layers but addresses the challenge of translation-invariance in deeper convolutional layers by using position-sensitive score maps. These maps encode relative spatial information and are pooled to achieve precise localization.

R-FCN divides the region of interest into a grid and scores each cell for detection class features, later averaging these scores to predict the object class.

DetectoRS

Many modern two-stage object detectors employ a two-step approach, first proposing object candidates and then using these proposals to extract features for object detection. DetectoRS [54] takes this approach and applies it at both macro and micro levels of the network. At the macro level, they introduce a Recursive Feature Pyramid (RFP), while at the micro level, they introduce the Switchable Atrous Convolution (SAC). The combination of these two techniques, Recursive Feature Pyramid and Switchable Atrous Convolution, forms DetectoRS. DetectoRS is not suitable for real-time applications, as it can process only about 4 frames per second.

3.1.3. One Stage Methods

The main one stage methods are presented in this section [53] [24] [74].

YOLO

The You Only Look Once (YOLO) architecture is based on the idea of processing the entire image with a convolutional network. YOLO has several versions that have been developed over the years, with YOLOv8 being the most recent version in the current state of the art.

YOLOv1 [57], published in 2016, divides the image into an $S \times S$ grid and predicts B bounding boxes of the same class, along with its confidence for C different classes per grid element. Each bounding box prediction contains the confidence score, the coordinates of the bounding box's center, and its height and width. YOLOv2 (2017) [55] includes many improvements compared to the first version while still maintaining the same speed. In 2018, to be on par with other methods in the state-of-the-art, YOLOv3 [56] was developed, which shows significant changes and a larger architecture while keeping real-time performance. In the following years, several versions were developed, such as YOLOv4 [10], YOLOv5 [36], YOLOv6 [41], YOLOv7 [69], leading to YOLOv8 [37]. YOLOv8 is introduced by Ultralytics and offers five scaled versions, which differ in size: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large). YOLOv8 supports multiple vision tasks, including object detection, segmentation, pose estimation, tracking, and classification [65]. Over the years, other versions have also been developed, which are covered in [65], along with a more detailed description of the

aforementioned architectures.

SSD

Single Shot Multibox Detector (SSD) [43] was significantly faster and more accurate than state-of-the-art networks like YOLO and Faster R-CNN, as of 2016. SSD was built on VGG-16 [41], with additional auxiliary structures to improve performance. SSD detects smaller objects earlier in the network, while the deeper layers are responsible for offset of the default boxes and aspect ratio. The image and ground truth are fed to the network which extracts several feature maps of different dimensions; for each location in each activation map, a set of standard bounding boxes is proposed with different scales and aspect ratios and, for each of them, both the offset from the ground truth box and the confidence on each object category is computed.

3.1.4. Performance Metrics

The most common way to measure the accuracy of a detection is the Average Precision (AP) [49]. Before describing AP in detail, it is important to introduce a few concepts. A prediction, based on its correctness, can be classified as:

- True Positive (TP): it is a correct detection of a ground-truth bounding box.
- False Negative (FN): it is an undetected ground-truth bounding box.
- False Positive (FP): it is an incorrect detection of a nonexistent object or a misplaced detection of an existing object.
- True Negative (TN): it is not applicable

In order to give these definitions, it is necessary to establish what the correctness of a detection is. To do this, an index called intersection over union (IoU) is used. IoU measures the area overlap between the predicted bounding box BB_p and the ground truth box BB_{gt} , divided by their union.

$$\text{IoU} = \frac{\text{area}(BB_p \cap BB_{gt})}{\text{area}(BB_p \cup BB_{gt})} \quad (3.1)$$

To determine whether a detection is correct or incorrect, a threshold value is defined. If the IoU value is greater than the threshold, the detection is correct, otherwise it is incorrect. The assessment of object detection methods is mostly based on the precision P and recall R concepts, respectively defined as:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}} \quad (3.2)$$

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}} \quad (3.3)$$

The precision vs. recall curve can be seen as a trade-off between precision and recall for different confidence values associated to the bounding boxes generated by a detector. The area under this curve is defined as the AP and it is defined individually for each class in the data-set. Its definition is:

$$\text{AP} = \int_0^1 P(R) dR \quad (3.4)$$

If several classes are present, mean Average Precision (mAP) can also be introduced. mAP is used to define the accuracy of the detector over all the classes, and it can be computed as:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i \quad (3.5)$$

where N is the number of classes in the data-set.

3.2. Perspective-n-Point problem

The Perspective-n-Point problem is the problem of estimating the pose of an object, given a set of n points of the object itself with known 3D model coordinates, and given the corresponding 2D projections in the image. The definition of the problem statement is the following. Firstly, the reference frames of the camera and the target, which are referred to as C and B respectively, are defined. It is possible to express a point \mathbf{r}_B in the camera reference frame as follows:

$$\mathbf{r}_B = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \mathbf{R}_{B/C} \mathbf{r}_B + \mathbf{t}_{C/B} \quad (3.6)$$

In this context, $\mathbf{R}_{B/C}$ represents the rotation matrix between the target and the camera frames, while $\mathbf{t}_{C/B}$ represents the translation vector from the camera to the origin of the

target frame.

Subsequently, it is possible to introduce the pinhole camera model, which allows for the calculation of the projection onto the image plane of a point in the camera frame. Combining this model with equation (3.6), one can derive the perspective projection equations:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \frac{f}{\rho_u} & 0 & u_0 \\ 0 & \frac{f}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R}_{B/C} \mid \mathbf{t}_{C/B} \right] \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \mathbf{A}\mathbf{N} \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} \quad (3.7)$$

\mathbf{A} is the intrinsic camera matrix, which contains some known parameters such that the focal length f and the coordinates of the camera principal point. In (3.7), the unknown is the matrix \mathbf{N} . To find this matrix, there are various methods, both implicit and explicit, and in the next section, one of the explicit methods called EPnP will be analyzed. When distance information regarding the actual point is unavailable, the vector $[u; v; w]$ undergoes scaling in relation to the third component. So w is called scaling parameter. This commonly occur when employing monocular cameras that do not offer distance information. Conversely, the use of a stereo camera, which does provide distance information, may avoid the need for the scaling parameter.

3.2.1. Efficient Perspective-n-Point Solver

The Efficient Perspective-n-Point (EPnP) [40] is a non-iterative solution to the PnP problem. The PnP problem consists in estimating the pose of an object given a set of n points of the object with known 3D model coordinates and give the corresponding 2D projection detected in the image. This type of non-iterative solution provides better accuracy and lower computational complexity than other non-iterative methods, while it is faster but loses slightly in terms of accuracy compared to iterative methods. The idea behind the method is to express the n 3D points as a weighted sum of 4 virtual control points, and it requires at least 4 point correspondences to be used. The reference points are defined as \mathbf{p}_i (where $i = 1, 2, \dots, n$), while virtual control points are called \mathbf{c}_j (where $i = 1, 2, \dots, n$). The relationship linking the reference points to the virtual control points is:

$$\mathbf{p}_i = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j \quad (i = 1, 2, \dots, n) \quad (3.8)$$

where α_{ij} are homogeneous barycentric coordinates. This relationship holds for both the world coordinate system and the camera coordinate system. Let's now derive the matrix \mathbf{M} whose kernel contains the solution of the problem. The 2D projections of the reference points in the camera coordinate system, \mathbf{u}_i (where $i = 1, 2, \dots, n$), are obtained via the internal calibration matrix \mathbf{A} using the following equation:

$$w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{A} \mathbf{p}_i^c = \mathbf{A} \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^4 \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \quad \forall i \quad (3.9)$$

Substituting the third row of the equation (3.9) into the first two rows gives the following two equations:

$$\begin{aligned} \sum_{j=1}^4 \alpha_{ij} f_u x_j^c + \alpha_{ij} (u_c - u_i) z_j^c &= 0 \\ \sum_{j=1}^4 \alpha_{ij} f_v y_j^c + \alpha_{ij} (v_c - v_i) z_j^c &= 0 \end{aligned} \quad (3.10)$$

It is possible to rearrange the equations (3.10) in matrix form as follows:

$$\mathbf{M} \mathbf{x} = 0 \quad (3.11)$$

where \mathbf{x} is the vector containing the unknowns of the problem and \mathbf{M} is a $2n \times 12$ matrix. Therefore the solution of the problem lies in the kernel of \mathbf{M} and it is expressed as:

$$\mathbf{x} = \sum_{i=1}^N \beta_i \mathbf{v}_i \quad (3.12)$$

where the set \mathbf{v}_i are the columns of the right-singular vectors of \mathbf{M} corresponding to the N null singular values of \mathbf{M} . After calculating the initial coefficients β_i , the Gauss-Newton algorithm is used to refine them.

4 | CoMBiNa: New Formulation

In this chapter, modifications to the CoMBiNa algorithm for the full pose estimation of symmetrical targets are presented. In [20], the problem is addressed through adjustments to state variables and the reconstruction of the orientation of the symmetry axis of the target in the inertial reference frame. The current objective is to determine the MRPs $\sigma_{\mathcal{T}}$, angular velocity $\omega_{\mathcal{T}}$, and inertial terms \mathbf{k} , following a similar approach to that used for asymmetrical targets in [45]. This new process assumes the presence of features on the outer surface of the target, such as stickers representing various state flags, symbols from space agencies, or luminous LEDs. Furthermore, it is assumed that a neural network is available to detect the positions of the centers of these objects in the images provided by the stereo camera. Such a network has been trained in the course of this thesis and its performances are shown in Chapter 5

Fig. 4.1 shows the new CoMBiNa pipeline, in which the neural network is integrated into the pipeline presented in [20]. Two blocks are considered for BCPD. The first is called *BCPD standard*, and provides measurements of the relative position and MRPs of the target, while the block called *BCPD rotational* provides the measurement of the relative position and orientation of the axis of symmetry \mathbf{J} .

4.1. CoMBiNa: Symmetric Case with Features Detection

4.1.1. Features Choice

In this chapter, the objective is to establish that parameters can be determined by assuming the knowledge of the features locations. To achieve this, features were manually placed randomly on the coarse model, as illustrated in Fig. 4.2. This is performed because, during the preliminary phase in which the coarse model is build up, also the positions of the features on it are reconstructed. Their measurements, typically acquired through a neural network with stereo camera images as input (in a real-world application), are instead

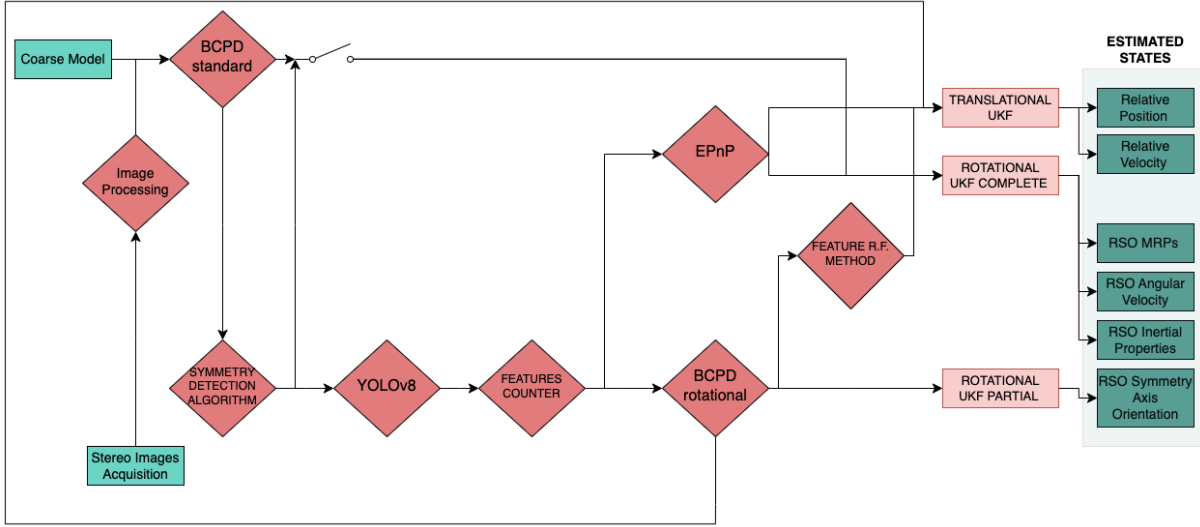


Figure 4.1: Pipeline for the CoMBiNa operational phase

simulated using the ground truth of the state vectors, since it is assumed that the neural network is available and functioning already during the preliminary phase of combina. This would allow to reconstruct a coarse model, while simultaneously characterizing some recognizable features. Fig. 4.3 illustrates the coarse model with the features expressed in the camera reference frame.

However, employing this method for simulating feature measurements results in not only obtaining measurements for the visible features depicted in the image captured by the stereo camera, but also for features that are not visible. This condition, being unrealistic, requires the introduction of a visibility check for each feature at every time instant (Fig. 4.4). Consequently, the measurement vector at a specific time instant i will only include measurements of features that are visible.

The visibility check is performed by taking into account the ground truth of the relative position, the ground truth of the orientation of the axis \mathbf{J} in the inertial system and simulated feature measurements. The check consists, for each feature and for each time step, of calculating the direction perpendicular to the orientation of the axis \mathbf{J} passing through the feature coordinates expressed in the target-centred inertial system. Then the direction of the relative position between chaser and target in the inertial system is calculated. Finally, having these two directions available, the angle between them is calculated using the Matlab command `atan2`. If $0 \leq \text{angle} \leq 90$, the feature is visible, but if $90 < \text{angle} < 180$, the feature is not visible and is discarded.

Furthermore, to closely replicate real-world conditions, a probability of failure in feature detection by the neural network is introduced. This detection failure means that even if

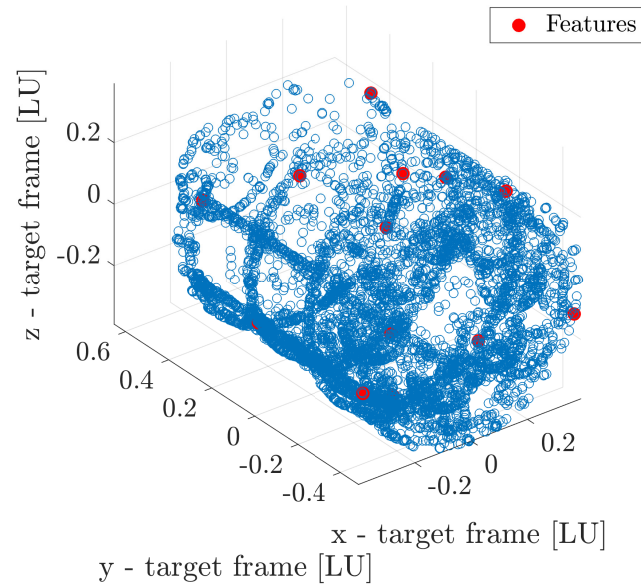


Figure 4.2: Coarse model and features expressed in the target reference frame

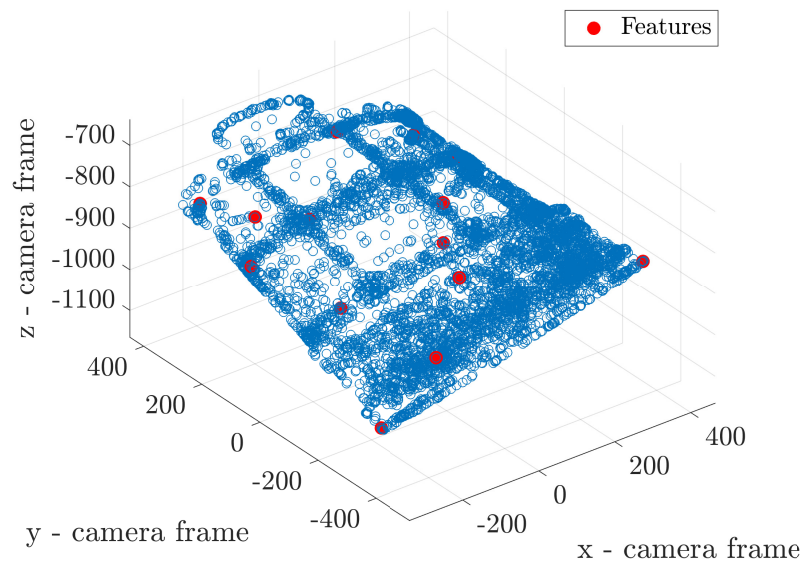


Figure 4.3: Coarse model and features expressed in the camera reference frame

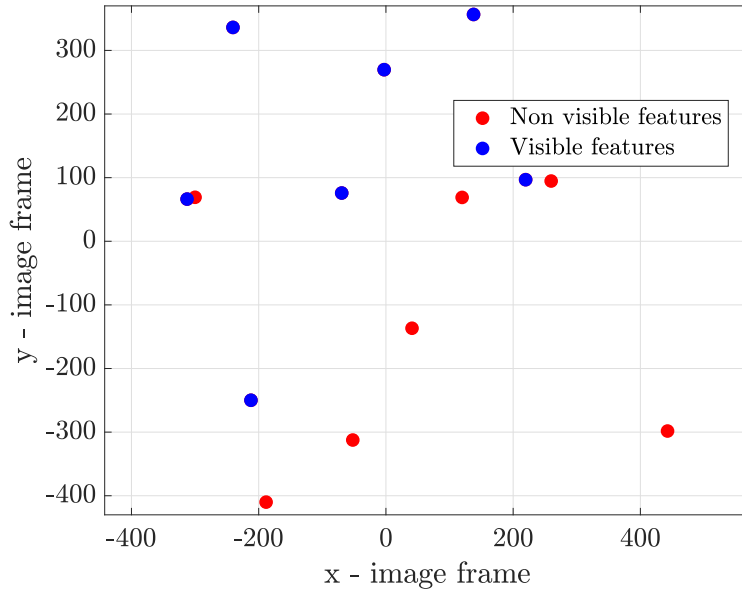


Figure 4.4: Visible e non visible features expressed in the image reference frame

a feature is visible, it might go undetected by the network, rendering its measurement unavailable. The probabilities considered in this work are 0% (indicating no failure), 20%, and 40%. This failure rate corresponds to the rate of false negatives of a network. As it will be seen in Chapter 5, these false negative rates are an envelop of the ones obtained by the trained network. Moreover, this work does not account for other failures which are characteristic of neural networks such as false positives (i.e., identification of features in wrong locations), detection of multiple features of the same type (i.e., introducing an ambiguity), and the possibility of wrong feature classification (i.e., wrong assignment of ID to an identified feature). The introduction of procedures to deal with these problems constitute an interesting future research direction. However, some preliminary conclusions about their impact can be drawn by looking at the performance of the trained CNN.

4.1.2. Algorithm's Update

This section outlines the modifications made to the CoMBiNa algorithm in the context of symmetrical RSO. Two algorithms, the *Main Loop* 4.1 and the *Optimization Function* 4.2, have been modified in this work. The *Main Loop* includes the procedures for updating and propagating states and covariance matrices using the UKF. The *Optimization Function* describes the procedures for obtaining the measurements used in the filter.

Considering the *Main Loop*, in [45], the states employed are:

$$\hat{\mathbf{x}}_s = [\hat{\mathbf{r}}; \hat{\mathbf{v}}], \quad \hat{\mathbf{x}}_{a,2} = [\hat{\boldsymbol{\sigma}}; \hat{\boldsymbol{\omega}}; \hat{k}] \quad (4.1)$$

while in [20] $\hat{\mathbf{x}}_{a,2}$ is substituted with a new state vector:

$$\hat{\mathbf{x}}_{a,1} = \left[\hat{\phi}; \hat{\delta}; \frac{\hat{\Gamma}^{(0)}}{A} \right] \quad (4.2)$$

This substitution is necessary because in cases involving symmetrical targets, the estimation of the $\hat{\mathbf{x}}_{a,2}$ state is inaccurate. Nevertheless, by incorporating information about the position of the features in the image, acquired through the neural network, it becomes possible to accurately estimate $\hat{\mathbf{x}}_{a,2}$ even in symmetrical target scenarios. Consequently, in the reconfiguration of the main loop, the quantity of available features at a given time instant is very important.

In the reformulation of the *Main Loop* 4.1, the idea is to use $\hat{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_{a,2}$ as states, introducing $\hat{\mathbf{x}}_{a,1}$ only in the absence of features. If features are present in the image, $\hat{\mathbf{x}}_{a,1}$ is not reconstructed. In this case, the goal is to estimate only $\hat{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_{a,2}$. In cases where no features are detected within the image, $\hat{\mathbf{x}}_{a,1}$ is defined based on the last available information about $\hat{\mathbf{x}}_{a,2}$. In this scenario, the estimate of $\hat{\mathbf{x}}_{a,2}$ is only propagated by the filter, but its value is not corrected. Consequently, the longer the duration without available features, the less accurate the $\hat{\mathbf{x}}_{a,2}$ estimation becomes. The filter, therefore, operates on $\hat{\mathbf{x}}_s$ and $\hat{\mathbf{x}}_{a,1}$ in order to correctly estimate them, enabling the acquisition of at least the orientation of the symmetry axis of the RSO.

The most significant change, enabling the correct estimation of $\hat{\mathbf{x}}_{a,2}$ lies in the methodology for deriving measurements. In [20], it is emphasized that the $\mathbf{C}_{\mathcal{TI}}$ matrix provided by BCPD is incorrect for a symmetrical target. Only the information on the second line of this matrix, which describes the orientation of the symmetry axis in space, can be used, but it does not allow to derive measurements for MRPs. In the modification of the *optimization function* 4.2, the objective is to obtain accurate measurements regarding MRPs by leveraging information about the position of the features in the image, which is provided by the neural network.

The choice of methodology used depends on the number of features available at a certain instant in time.

MRPs Estimation Using EPnP

When there are at least four features in the same image, it becomes possible to employ EPnP to determine the $\mathbf{C}_{\mathcal{T}\mathcal{I}}$ matrix and, subsequently, the MRPs. In this study, EPnP with Gauss Optimization is used, and the corresponding code is provided in [1]. The function used, "*efficient_pnp_gauss*", requires as input a vector containing the homogeneous coordinates of the points in the target reference frame, a vector containing the homogeneous positions of the points on the image plane, and the intrinsic camera matrix defined as follows:

$$\mathbf{A} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

Among the outputs provided by the function, the most significant are the rotation $\mathbf{R}_{S\mathcal{T}}$ and translation $\mathbf{T}_{S\mathcal{T}}$ of the camera reference frame with respect to the target reference frame.

$\mathbf{C}_{\mathcal{T}\mathcal{I}}$ can now be computed using the consecutive rotation rule (Eq. 4.4).

$$\mathbf{C}_{\mathcal{T}\mathcal{I}} = \mathbf{R}_{S\mathcal{T}}^T \mathbf{C}_{CS}^T \mathbf{C}_{C\mathcal{I}} \quad (4.4)$$

To achieve this, two matrices need to be defined: the $\mathbf{C}_{C\mathcal{I}}$ matrix, computed using the chaser MRPs, and the rotation matrix of the chaser frame in relation to the camera frame, denoted as \mathbf{C}_{CS} . This matrix is determined by referring to Fig. 2.1, where it illustrates that the x-axis of the camera frame aligns with the y-axis of the chaser frame, and the y-axis of the camera frame aligns in the opposite direction of the x-axis of the chaser frame. Therefore, \mathbf{C}_{CS} is defined as:

$$\mathbf{C}_{CS} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

At this point, the MRPs of the target $\boldsymbol{\sigma}_{\mathcal{T}}$ can be derived from $\mathbf{C}_{C\mathcal{I}}$.

Regarding the measurement of the relative position between the chaser and target, it must be defined in the LVLH reference frame. Starting with the value of $\mathbf{T}_{S\mathcal{T}}$ obtained from EPnP, the relative position in the LVLH reference frame can be determined as follows:

$$\mathbf{r} = \mathbf{C}_{\mathcal{CL}}^T \mathbf{C}_{\mathcal{CS}} \mathbf{T}_{\mathcal{ST}} \quad (4.6)$$

where $\mathbf{C}_{\mathcal{CL}}$ is the rotation of the chaser reference frame with respect to the LVLH reference frame, computed as:

$$\mathbf{C}_{\mathcal{CL}} = \mathbf{C}_{\mathcal{CI}} \mathbf{C}_{\mathcal{LI}}^T \quad (4.7)$$

However, when the number of available points is equal to four, EPnP often yields incorrect results. On the other hand, increasing the number of points in the image leads to improved results; having five points available already provides much more accurate and precise outcomes [11].

For this reason, after acquiring measurements from EPnP, the validity of the result is verified by examining the squared Mahalanobis distance. This verification process is equal to the one previously described in Chapter 2. If the verification is successful, meaning that the squared Mahalanobis distance is less than the threshold value, the obtained measurements are:

$$\mathbf{z}_s = \mathbf{r} \quad \mathbf{z}_a = \boldsymbol{\sigma}_{\mathcal{T}} \quad (4.8)$$

otherwise, if the EPnP results do not pass the verification, measurements are obtained using the method described below.

The choice of using EPnP instead of BCPD, when sufficient features are present in the image, is justified by the fact that EPnP provides measurement estimation faster than BCPD.

MRPs Estimation Using Feature Reference Frame Method

In the cases where features are present in the image, but not in sufficient number to use EPnP, or when the measurements provided by EPnP are rejected, this new method, which starts by employing BCPD, is used to derive the full attitude measurements of the target. BCPD enables the accurate determination of the relative position measurement, \mathbf{r} , and the second row of the $\mathbf{C}_{\mathcal{T}\mathcal{I}}$ matrix, representing the orientation of the symmetry axis of the target \mathbf{J} in the inertial system. Subsequently, a feature is selected from those whose positions are provided in the image by the neural network and associated with its corresponding feature in the coarse model. Considering this information, the method aims to construct a reference system for the chosen feature.

By using the measurement of the position of the feature in the image, the reference system expressed in the inertial system is constructed. Simultaneously, using the feature in the coarse model, the reference system is expressed with respect to the target reference frame. The correct $\mathbf{C}_{\mathcal{I}\mathcal{I}}$ can then be derived from these two reference systems.

Here, a more detailed explanation of the process by which the reference systems for features are derived is reported.

Initially, the orientation of the symmetry axis \mathbf{J} of the target is considered as the orientation of the Y-axis, $\mathbf{Y}_{\mathcal{F}}^{\mathcal{I}}$, of the reference frame of the features in the inertial system. Subsequently, the idea is to establish a direction perpendicular to $\mathbf{Y}_{\mathcal{F}}^{\mathcal{I}}$, passing through the feature coordinates, expressed in the inertial reference frame. This direction represents the orientation of the X-axis, $\mathbf{X}_{\mathcal{F}}^{\mathcal{I}}$, of the feature reference system in the inertial system. Finally, by performing a vector product between $\mathbf{X}_{\mathcal{F}}^{\mathcal{I}}$ and $\mathbf{Y}_{\mathcal{F}}^{\mathcal{I}}$, it is possible to determine the orientation of the Z-axis, $\mathbf{Z}_{\mathcal{F}}^{\mathcal{I}}$.

The neural network provides the position of the feature in the image, which subsequently needs to be converted into the inertial system. To achieve this, the coordinates in the chaser reference frame, \mathbf{f}^c are computed using the equation 2.3. Then, by using the $\mathbf{C}_{c\mathcal{I}}$ matrix computed from the chaser MRPs, the coordinates of the feature in the inertial system, centered on the chaser, are determined (Eq.4.9). Finally, the inertial system is re-centered on the target, with the measurement obtained from BCPD for relative position, which is converted from the LVLH system to the inertial system using the $\mathbf{C}_{\mathcal{L}\mathcal{I}}$ matrix (Eq.4.10), obtaining the coordinates of the feature in the inertial reference frame centred on the target.

$$\mathbf{f}^{\mathcal{I}c} = \mathbf{C}_{c\mathcal{I}}^T \mathbf{f}^c \quad (4.9)$$

$$\mathbf{f}^{\mathcal{I}\tau} = \mathbf{f}^{\mathcal{I}c} - \mathbf{C}_{\mathcal{L}\mathcal{I}}^T \mathbf{r} \quad (4.10)$$

To determine the direction of the line perpendicular to the axis of symmetry \mathbf{J} that passes through point $\mathbf{f}^{\mathcal{I}\tau}$, the process starts by identifying the point \mathbf{p}_{perp} , which is the nearest point on \mathbf{J} to $\mathbf{f}^{\mathcal{I}\tau}$. The x,y and z components of \mathbf{J} are called a , b and c respectively, while the three components of the point $\mathbf{f}^{\mathcal{I}\tau}$ are called $\mathbf{f}_x^{\mathcal{I}\tau}$, $\mathbf{f}_y^{\mathcal{I}\tau}$ and $\mathbf{f}_z^{\mathcal{I}\tau}$. To locate this point, a scaling parameter, denoted as \mathbf{t}_s , is computed (Eq.4.11). This parameter is employed to scale the direction vector \mathbf{J} and obtain the coordinates of $\mathbf{p}_{perp} = [a\mathbf{t}_s, b\mathbf{t}_s, c\mathbf{t}_s]$.

$$\mathbf{t}_s = a\mathbf{f}_x^{\mathcal{I}\tau} + b\mathbf{f}_y^{\mathcal{I}\tau} + c\mathbf{f}_z^{\mathcal{I}\tau} \quad (4.11)$$

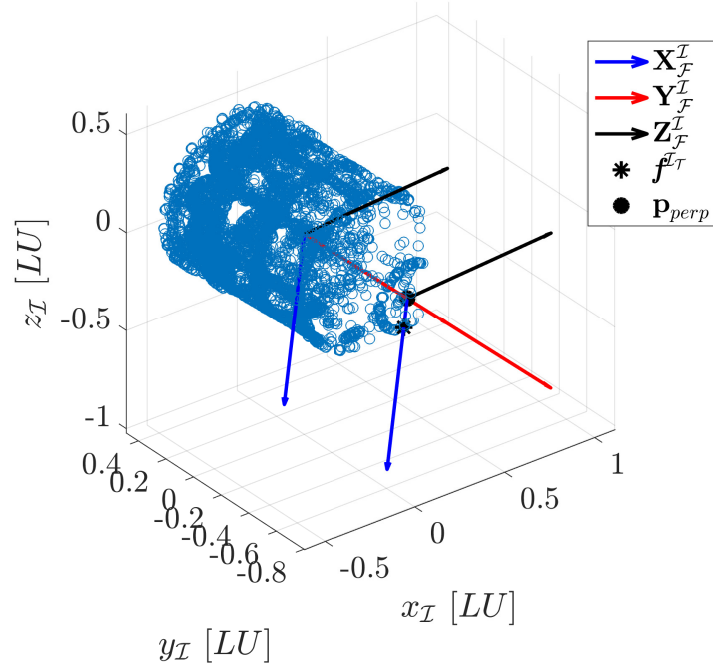


Figure 4.5: Representation of the orientation of the reference frame of the feature in the inertial reference frame

Subsequently, the components l , m , and n of the vector from $\mathbf{f}^{I\tau}$ to \mathbf{p}_{perp} are computed, and by normalizing these components, it is possible to obtain the direction of this vector, that corresponds to $\mathbf{X}_{\mathcal{F}}^I$.

Finally, after determining the value of $\mathbf{X}_{\mathcal{F}}^I$, $\mathbf{Z}_{\mathcal{F}}^I$ is calculated as follows:

$$\mathbf{Z}_{\mathcal{F}}^I = \mathbf{X}_{\mathcal{F}}^I \times \mathbf{Y}_{\mathcal{F}}^I \quad (4.12)$$

It is possible to derive the $\mathbf{C}_{\mathcal{F}I}$, which defines how the feature reference system is oriented with respect to the inertial reference system. The first row contains $\mathbf{X}_{\mathcal{F}}^I$, the second row contains $\mathbf{Y}_{\mathcal{F}}^I$, and the third row contains $\mathbf{Z}_{\mathcal{F}}^I$.

Figure 4.5 illustrates the results obtained by following this process. In this representation, a randomly selected feature is chosen, and the orientation of the axes of the reference frame of the feature in the inertial system is depicted. These axes are presented from two perspectives: one with point \mathbf{p}_{perp} as the origin and the other with the center of the target as the origin.

Now, the feature of the coarse model that corresponds to the one chosen from the image is taken into account. The features of the coarse model were reconstructed during the

preliminary phase, along with the model itself, and they are consequently available on board. The coordinates of the selected feature \mathbf{f}^T are expressed in the target reference frame. In this case, the reference system of the feature is constructed with respect to the target reference system. Consequently, the orientation of the axis of symmetry $\mathbf{J}|_{\mathcal{T}}$ will be:

$$\mathbf{J}|_{\mathcal{T}} = \mathbf{Y}_{\mathcal{F}}^T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \quad (4.13)$$

The procedure for calculating $\mathbf{X}_{\mathcal{F}}^T$ and $\mathbf{Z}_{\mathcal{F}}^T$, which represent the orientation of the x and z axes of the feature reference system in the target reference system respectively is the same as previously explained. Regarding the computation of $\mathbf{X}_{\mathcal{F}}^T$, it is derived as the orientation of the direction perpendicular to $\mathbf{Y}_{\mathcal{F}}^T$ passing through \mathbf{f}^T . Once $\mathbf{X}_{\mathcal{F}}^T$ is determined, $\mathbf{Z}_{\mathcal{F}}^T$ is computed through the vector product of $\mathbf{X}_{\mathcal{F}}^T$ and $\mathbf{Y}_{\mathcal{F}}^T$. The first row of the matrix $\mathbf{C}_{\mathcal{FI}}$ contains $\mathbf{X}_{\mathcal{F}}^T$, the second row contains $\mathbf{Y}_{\mathcal{F}}^T$ and the third row contains $\mathbf{Z}_{\mathcal{F}}^T$.

With knowledge of these two matrices, it is possible to calculate the $\mathbf{C}_{\mathcal{TI}}$ matrix by applying the rule of consecutive rotations. After obtaining $\mathbf{C}_{\mathcal{TI}}$, the MRPs for the target can be derived from it.

In conclusion, using this method the available measurements are:

$$\mathbf{z}_s = \mathbf{r} \quad \mathbf{z}_a = \boldsymbol{\sigma}_{\mathcal{T}} \quad (4.14)$$

No Features Available

In this scenario, where no features are available, the algorithm back falls into the situation previously outlined in [20]. Consequently, it becomes impossible to determine MRPs of the target, and it is only possible to derive the orientation of its axis of symmetry, resulting in the following measurements:

$$\mathbf{z}_s = \mathbf{r} \quad \mathbf{z}_a = \mathbf{J}|_{\mathcal{I}} \quad (4.15)$$

Algorithms

The pseudo-code of the two algorithms, *MainLoop* and *OptimizationFunction*, is shown in this paragraph. This helps to get a better view of what the two algorithms do by highlighting key steps.

Algorithm 4.1 Main Loop

```

1: Initialize all the variables. Define the state vectors  $\hat{\mathbf{x}}_s = [\hat{\mathbf{r}}; \hat{\mathbf{v}}]$ ,  $\hat{\mathbf{x}}_{a,2} = [\hat{\boldsymbol{\sigma}}; \hat{\boldsymbol{\omega}}; \hat{k}]$  and
    $\hat{\mathbf{x}}_{a,1} = \left[ \hat{\phi}; \hat{\delta}; \frac{\hat{\Gamma}^{(0)}}{A} \right]$  and their covariance matrices  $\mathbf{P}_s$ ,  $\mathbf{P}_{a,1}$  and  $\mathbf{P}_{a,2}$ 
2: for  $i = 1$  to  $N$  do
3:   Compute the number of visible features  $N_f$ , and store this value into a vector  $\mathbf{N}_f$ 
4:   Compute the sigma points for  $\hat{\mathbf{x}}_s$  and  $\hat{\mathbf{x}}_{a,2}$ 
5:   if  $N_f = 0$  then
6:     if  $i = 1$  or  $\mathbf{N}_f(i - 1) \neq 0$  then
7:       Compute  $\hat{\mathbf{x}}_{a,1}$  and  $\mathbf{P}_{a,1}$  from the knowledge of  $\hat{\mathbf{x}}_{a,2}$  and  $\mathbf{P}_{a,2}$ 
8:     end if
9:     Compute the sigma points for  $\hat{\mathbf{x}}_{a,1}$ 
10:  end if
11:  Run the Optimization Function to compute the measurements vector
12:  Store the measurements and pre-update the state vectors and covariance matrices
13:  if  $s \simeq 1$  then
14:    if  $N_f > 0$  then
15:      Update  $\hat{\mathbf{x}}_s$ ,  $\hat{\mathbf{x}}_{a,2}$ ,  $\mathbf{P}_s$  and  $\mathbf{P}_{a,2}$ 
16:      Define  $\hat{\mathbf{x}}_{a,1}$  and  $\mathbf{P}_{a,1}$  as NaN
17:    else
18:      Update  $\hat{\mathbf{x}}_s$ ,  $\hat{\mathbf{x}}_{a,1}$ ,  $\mathbf{P}_s$  and  $\mathbf{P}_{a,1}$ 
19:       $\hat{\mathbf{x}}_{a,2}$  and  $\mathbf{P}_{a,2}$  are not updated
20:    end if
21:  else
22:    All the variables are not updated
23:  end if
24:  Store state vectors and covariance matrices post update
25:  if  $N_f > 0$  then
26:    Propagate  $\hat{\mathbf{x}}_s$ ,  $\hat{\mathbf{x}}_{a,2}$ ,  $\mathbf{P}_s$  and  $\mathbf{P}_{a,2}$ 
27:    Define  $\hat{\mathbf{x}}_{a,1}$  and  $\mathbf{P}_{a,1}$  as NaN
28:  else
29:    Propagate all the variables
30:  end if
31: end for

```

Algorithm 4.2 Optimization Function

```

1: Definition of the number of available features  $N_f$ 
2: if  $N_f \geq 4$  then
3:   Use EPnP algorithm to compute the vector of measurements  $\mathbf{z}_{s,a} = [\mathbf{r}; \boldsymbol{\sigma}_{\mathcal{T}}]$ 
4:   Compute the Mahalanobis Distance MD
5:   if  $\text{MD}^2 \leq \text{threshold}$  then
6:     The measurements provided by EPnP can be used
7:   else
8:     Define a warning variable,  $\text{warning} = 0$ 
9:   end if
10: else
11:   Define a warning variable,  $\text{warning} = 0$ 
12: end if
13: if  $\text{warning} = 0$  then
14:   Run BCPD to obtain the vector of measurements  $\mathbf{z}_{s,a} = [\mathbf{r}; \mathbf{J}|_{\mathcal{I}}]$ 
15:   if  $N_f = 0$  then
16:     The vector of measurements is  $\mathbf{z}_{s,a} = [\mathbf{r}; \phi; \delta]$ 
17:   else
18:     Compute the direction cosine matrix  $\mathbf{C}_{\mathcal{T}\mathcal{I}}$ 
19:     The vector of measurements is  $\mathbf{z}_{s,a} = [\mathbf{r}; \boldsymbol{\sigma}_{\mathcal{T}}]$ 
20:   end if
21: end if

```

4.1.3. Filter Structure for Unknown Rigid Rotation Matrix

In cases where the rigid rotation matrix is unknown, the same procedure as outlined in [20] is applied. The MRPs of this matrix are introduced into the state vector $\hat{\mathbf{x}}_{a,1}$, resulting in an augmented state:

$$\hat{\mathbf{x}}_{a,1} = \begin{bmatrix} \hat{\phi} \\ \hat{\delta} \\ \hat{\Gamma}_{(0)} \\ A \\ \hat{\boldsymbol{\sigma}}_{\mathcal{I}\Gamma\mathcal{I}} \end{bmatrix} \quad (4.16)$$

The equations governing the dynamics of these variables are detailed in Chapter 2, specifically in Equation 2.45.

Within the algorithm, the MRPs of the rigid rotation matrix are treated as the parameters

responsible for determining the orientation of the \mathbf{J} axis. These parameters are computed based on the latest available information concerning the attitude of the target, but only if no features are detected in the image. Once computed, they are considered constant throughout periods of feature absence, because of their dynamics.

4.2. Numerical Results

In this section, a numerical analysis is performed to test the new formulation of CoMBiNa. Initially, the problem is outlined in a general context, providing a concise description of the orbital scenario and presenting the ground truth of the different variables. In a subsequent subsection, the outcomes achieved using the new algorithm formulation, assuming prior knowledge of the rigid rotation matrix, are presented. Additionally, the results obtained from a Monte Carlo analysis, conducted to assess the robustness of the algorithm, are described. Lastly, the section presents the results obtained when the rigid rotation matrix is unknown.

4.2.1. Problem Description & Ground Truth

To begin, it is essential to provide the specifications of the stereo camera used for capturing measurements in this work. The focal length f , pixel density d , and baseline b of the camera are detailed in Table 4.1.

Focal length f	Pixel density d	Baseline b
34 mm	48 pixels per unit length	1 m

Table 4.1: Stereo camera technical specifications

Nominal Relative Orbit

Regarding orbit, the chaser is considered to be on a circular orbit in LEO with a radius of 800 km. The mean motion in this case is $n \approx 0.001$ rad/s, whose inverse is used to provide the scaled time unit in all simulations, that is $\text{TU} = 963.65$ s. On the other hand, distances are scaled with the radius of the keep out sphere of the RSO, that is the maximum dimension of the RSO ($\text{LU} = 12$ m). Considering this parameters, the initial conditions of the nominal relative orbit are:

$$\mathbf{r} = [2; 0; 0.2]\text{LU}; \quad \mathbf{v} = [0; -4; 0]\text{LU/TU} \quad (4.17)$$

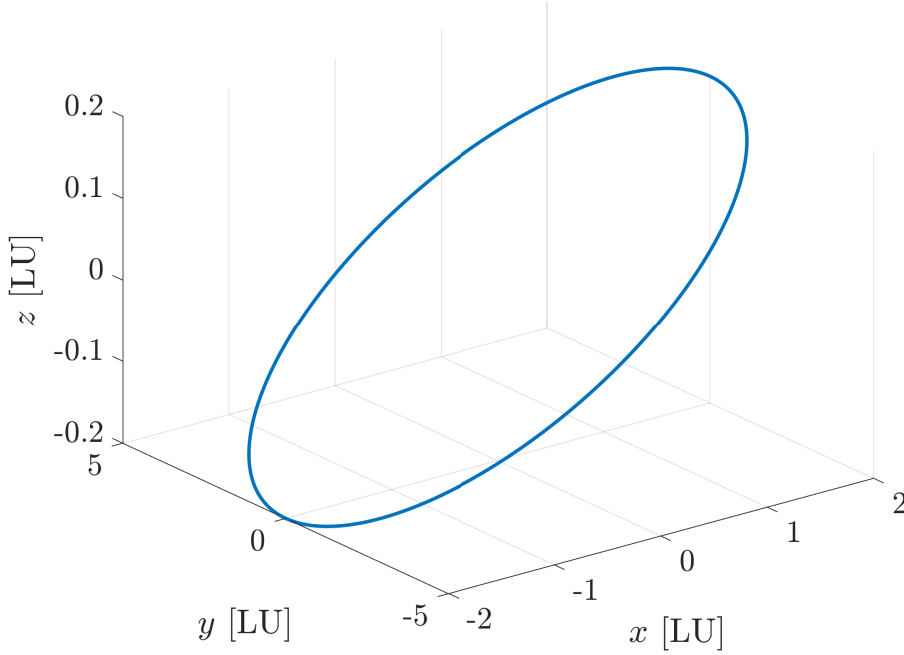


Figure 4.6: Representation of the nominal inclined football orbit in the LVLH frame.

This particular initial condition produces an inclined football orbit in the CW dynamics [70], and an approximate one in NERM. All simulations reported have been obtained by propagating the variables for an observation time of 10 [TU] and a time step of 0.01 [TU], obtaining 1000 time instants. The Fig. 4.6 represents the relative orbit, and the Fig. 4.7 shows the evolution of the relative position and velocity. The initial covariance matrix of the state vector $\hat{\mathbf{x}}_s$ is:

$$\mathbf{P}_{\mathbf{x}_s} = 0.1 \cdot \mathbf{I}_6 LU \quad (4.18)$$

The measurement noise \mathbf{R}_s is used as a tuning parameter of the filter and it is defined as:

$$\mathbf{R}_s = 0.01 \cdot \mathbf{I}_3 \quad (4.19)$$

The mean motion n is also used to compute the direction cosine matrix $\mathbf{C}_{\mathcal{LI}}$, expressing the orientation of the LVLH frame with respect to the inertial frame, as follows:

$$\mathbf{C}_{\mathcal{LI}} = \begin{bmatrix} \cos nt & \sin nt & 0 \\ -\sin nt & \cos nt & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.20)$$

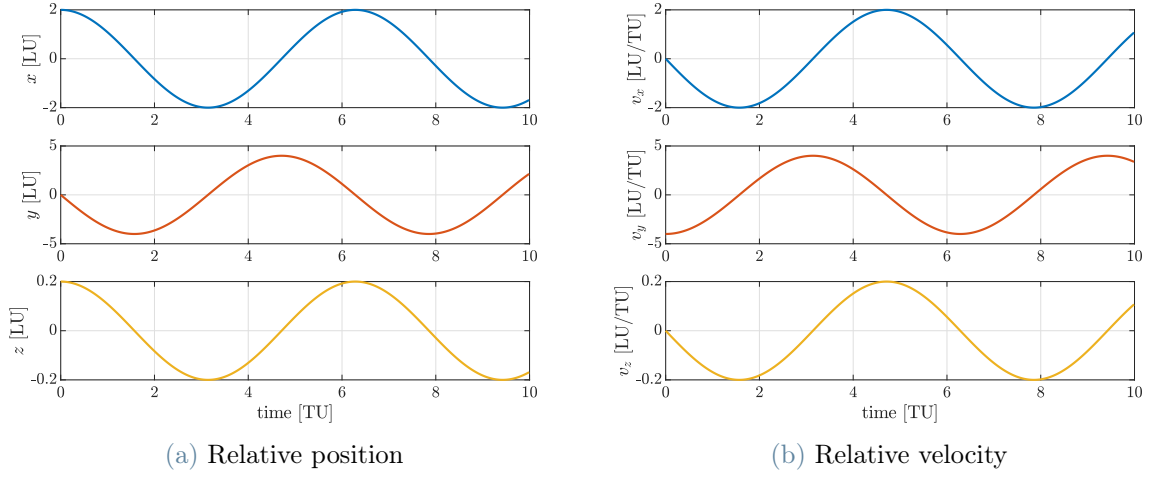


Figure 4.7: Component-wise evolution of the relative position and velocity in the LVLH frame

Chaser Attitude

The attitude of the chaser is initialize as follows:

$$\boldsymbol{\sigma}_C = [0; -0.4437; 0]; \quad \boldsymbol{\omega}_C = [0; 0; 1] \text{rad/TU} \quad (4.21)$$

the chaser vehicle is assumed to tumble according to Eq. 2.17 with a fully symmetric inertia matrix. This particular choice of attitude parametrization allows the RSO to be kept inside the FOV of the camera without having to maneuver the chaser spacecraft, while the angular velocity remains constant [20].

Target Attitude

The attitude of the target is initialized with the following nominal values, which are used in the simulated environment to produce the images used by the algorithm:

$$\boldsymbol{\sigma}_T = [-0.083; 0.220; -0.500]; \quad \boldsymbol{\omega}_T = [-0.791; -1.1413; -1.631] \text{[rad/TU]} \quad (4.22)$$

Its inertial parameters, which remain constant in time, are defined as follows:

$$\mathbf{k}_T = [0.03; -0.03] \quad (4.23)$$

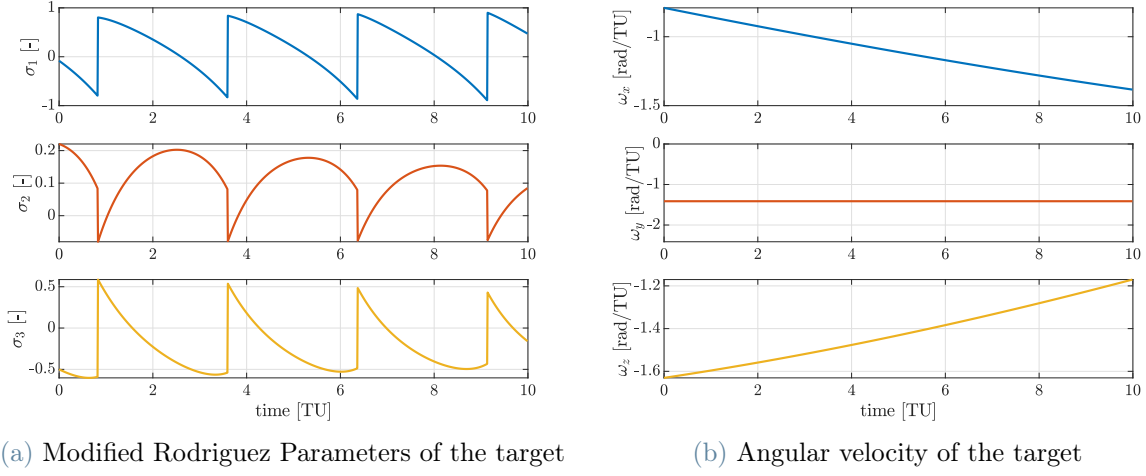


Figure 4.8: Component-wise evolution of the chaser's attitude: discontinuities in the MRP happen at switching location between nominal and shadow set

Fig. 4.8 shows the time evolution of Modified Rodriguez Parameters and angular velocity.

The initial covariance matrix and the measurements noise of the state vector $\hat{\mathbf{x}}_{a,2}$ are:

$$\mathbf{P}_{\mathbf{x}_{a,2}} = 0.01 \cdot \mathbf{I}_8 \quad \mathbf{R}_{a,2} = 0.01 \cdot \mathbf{I}_3 \quad (4.24)$$

In the reformulation of CoMBiNa proposed in [20], the variables of interest change, and it becomes necessary to specify their initial conditions. To maintain consistency with the initial conditions for the target's orientation as outlined previously, the initial values for the new state are derived from the values of the original state. Consequently, the initial conditions are:

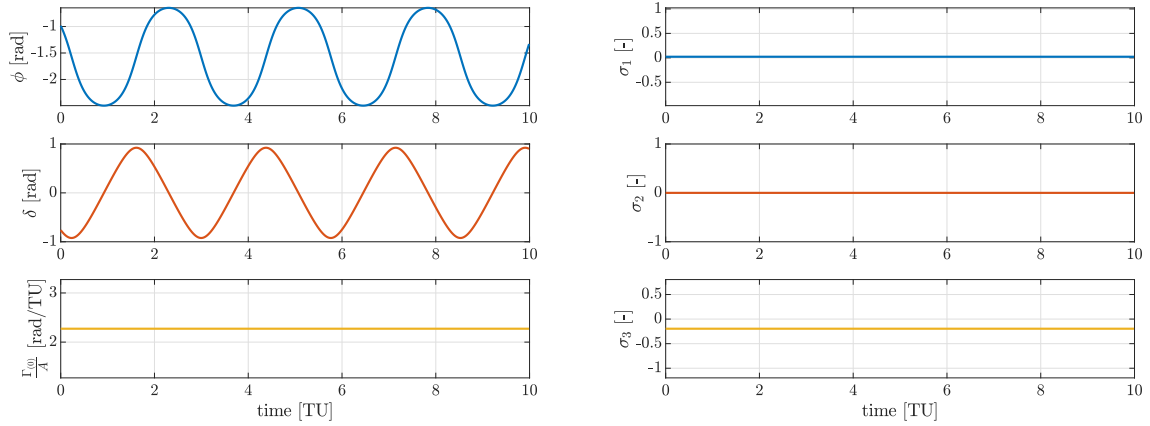
$$\phi = -0.986[\text{rad}] \quad \delta = -0.762[\text{rad}] \quad \frac{\Gamma_{(0)}}{A} = 2.2725[\text{rad/TU}] \quad (4.25)$$

The rigid rotation matrix can be expressed in terms of Modified Rodriguez parameters and by exploiting Eq. 2.8, it is possible to get:

$$\boldsymbol{\sigma}_{\mathcal{I}\mathcal{I}} = [0.0224; 0; -0.1960] \quad (4.26)$$

The evolution of these variables is represented in Fig. 4.9.

The Modified Rodrigues Parameters of the rigid rotation matrix remain unchanging over time, as they represent a rotation between two fixed inertial reference frames. $\frac{\Gamma_{(0)}}{A}$ remains



(a) Target's symmetry axis spherical coordinates in \mathcal{I}_Γ frame and $\frac{\Gamma_{(0)}}{A}$ ratio

(b) MRPs for rotation from \mathcal{I} to \mathcal{I}_Γ

Figure 4.9: Evolution of the new attitude variables of the target

constant due to the absence of external torque. The last quantities computed are the equatorial component of angular velocity \mathbf{e} and the square of the ratio between the axial component of angular velocity and the moment of inertia A . As expected, these values remain constant over time and are represented in Fig. 4.10.

The initial covariance matrix and the measurements noise of the state vector $\hat{\mathbf{x}}_{a,1}$ are:

$$\mathbf{P}_{\mathbf{x}_{a,1}} = NaN \quad \mathbf{R}_{a,1} = 0.01 \cdot \mathbf{I}_2 \quad (4.27)$$

4.2.2. Known Rigid Rotation

In this section, the new algorithm is tested under the assumption that the rigid rotation matrix is known. The algorithm is tested with different numbers of features on the target, starting with 1 and going up to 14. In the analysis different percentage of failure detection are used. Out of the 14 scenarios, three specific cases are reported, which correspond to the use of 1, 6, and 14 features. The scenario with 1 features is chosen because it represents the worst case considering the number of features. The case with 6 features is chosen because there are often 4 features available to use EPnP but, due to its poor performance with this number of points, its measurements are often rejected and BCPD is used. Lastly, the 14-feature case is selected because it relies solely on EPnP to determine measurements.

5 Since the problems highlighted in [45] and [20] are in the estimation of the attitude of a symmetrical target, while the position and relative velocity are estimated correctly, only

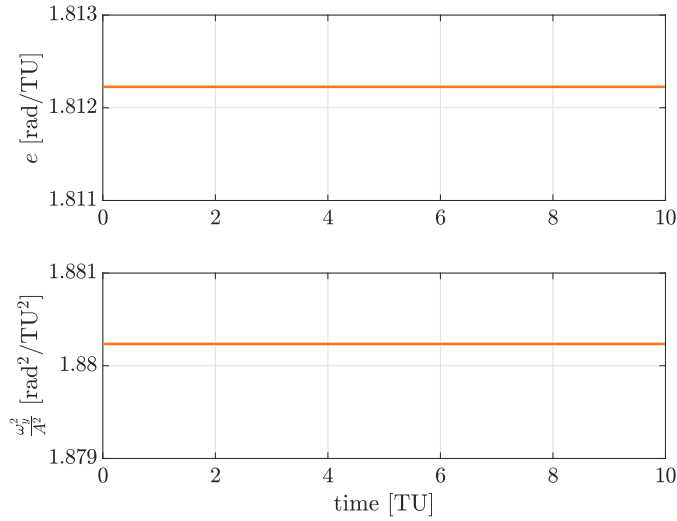


Figure 4.10: Evolution of attitude and inertia related quantities of the target

the convergence plots for attitude are shown below. The results obtained with a failure detection probability of 20% are given in Appendix A.

Results with 1 Feature

The initial case examined has a failure detection percentage of 0. With only one feature on the target, there are 423 time steps during which it remains unobservable in the image. The convergence plots presented in Fig. 4.11 indicate that, during these intervals, when the feature is not visible, the 3σ worsens for the MRPs σ and for the angular velocity ω , although for the inertial parameters k it remains constant. Despite these small worsenings at specific time steps, it is possible to achieve convergence.

In the context of a failure detection rate of 40%, there are 638 time steps where no features are visible in the image. The behavior of the convergence plots is the same as in the case described above (Fig. 4.12). In such scenarios, it is also very important to analyze the convergence plots for the state $\mathbf{x}_{a,1}$, as it often represents the sole source of information available for estimating, partially, the attitude of the target. It can be inferred that during the time intervals in which this state is estimated, the estimate tends to converge.

Results with 6 Features

In the scenario where the failure detection percentage is 0, it's ensured that at least one feature is visible in the images for each time step. As previously highlighted, when EPnP is used with limited features, there is a risk of obtaining imprecise measurements. Consequently, such measurements are discarded, and the transition to using those provided by

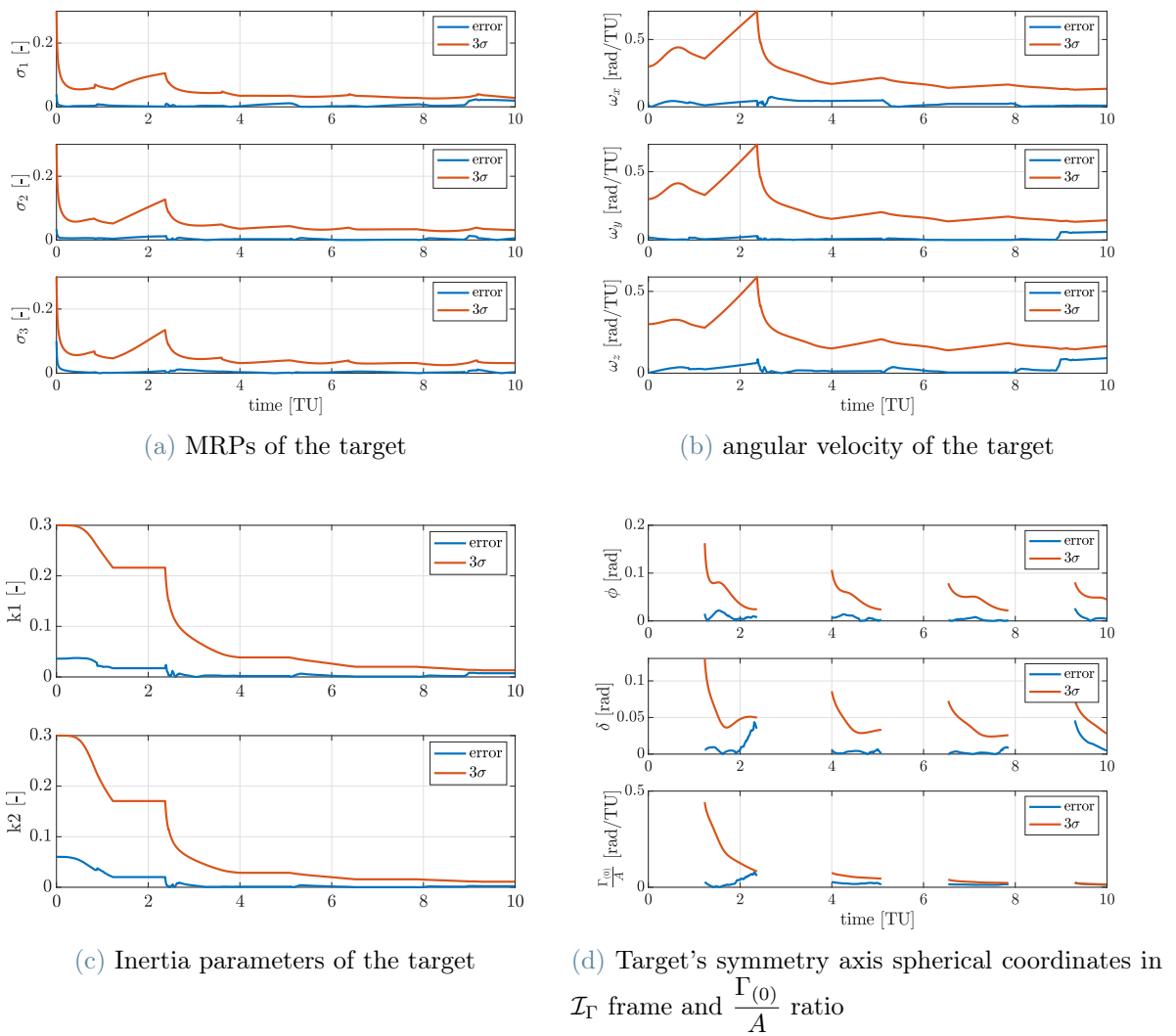
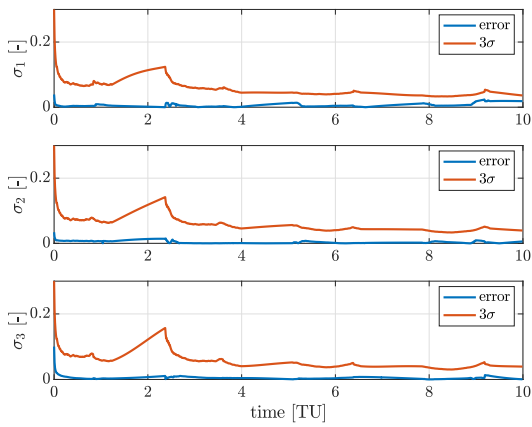
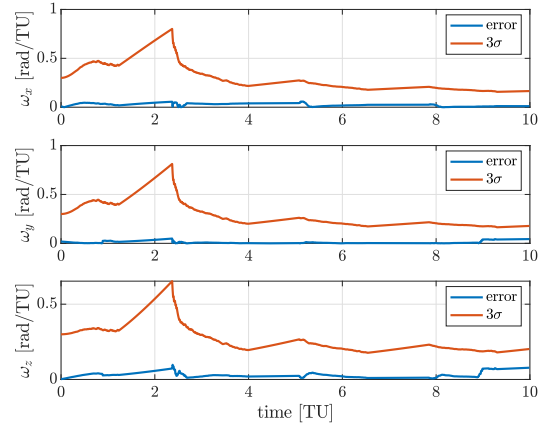


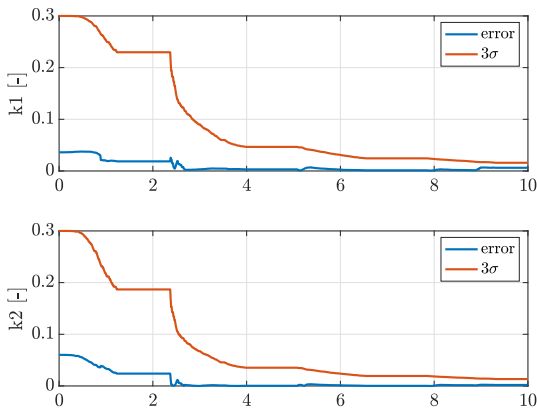
Figure 4.11: Convergence of the rotational filter (1 features and failure detection percentage = 0%)



(a) MRPs of the target



(b) angular velocity of the target



(c) Inertia parameters of the target

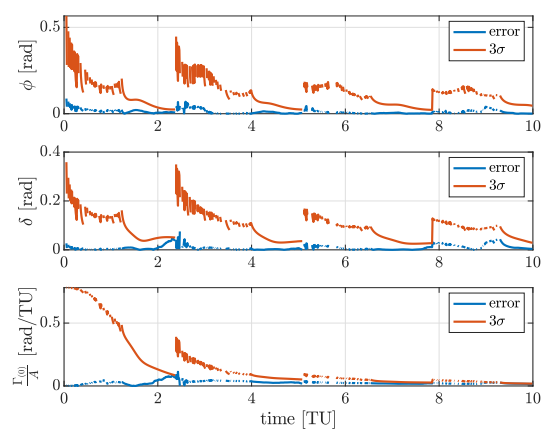
(d) Target's symmetry axis spherical coordinates in \mathcal{I}_T frame and $\frac{\Gamma(0)}{A}$ ratio

Figure 4.12: Convergence of the rotational filter (1 features and failure detection percentage = 40%)

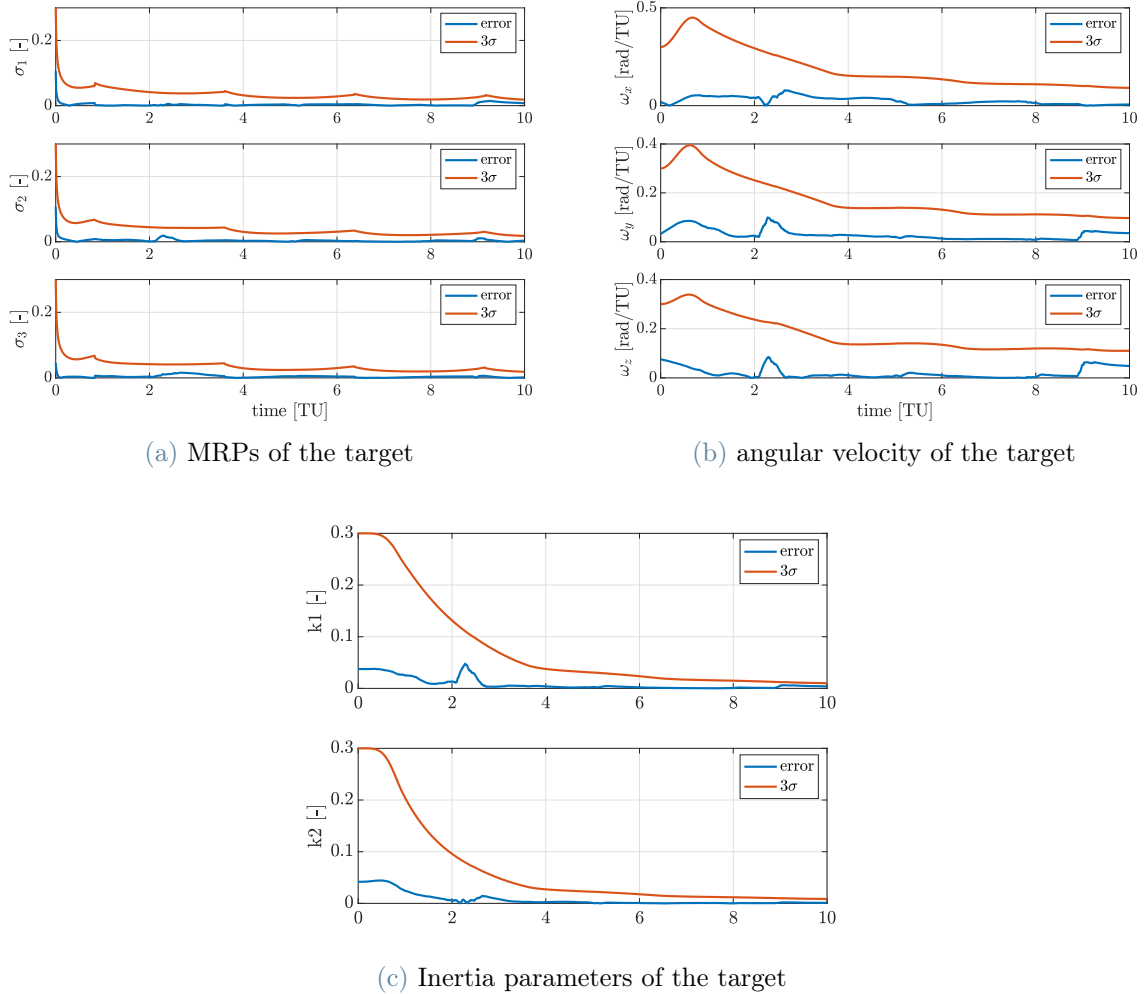


Figure 4.13: Convergence of the rotational filter (6 features and failure detection percentage = 0%)

BCPD takes place. This transition occurs 219 times. The convergence plots are depicted in Fig. 4.13.

Upon increasing the failure detection percentage to 40%, there are 59 time steps during which no features are available. At the same time, the occurrences of transitioning from EPnP to BCPD decrease from 219 to 29. This happens because in many situations where four features were present, having a failure in a detection leads to a reduction in the number of times that EPnP can be used. In this particular case, the convergence plots are displayed in Fig. 4.14.

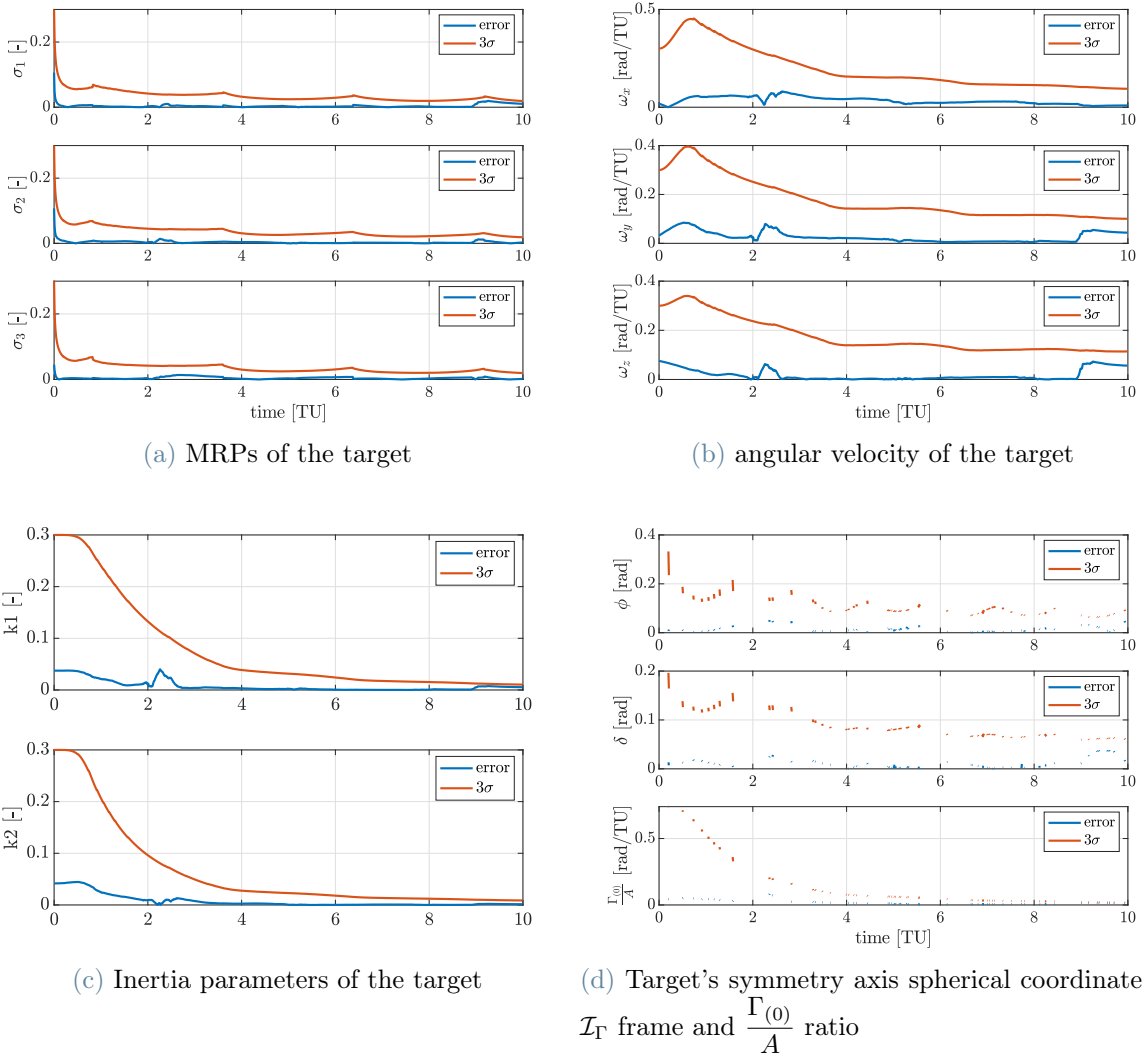


Figure 4.14: Convergence of the rotational filter (6 features and failure detection percentage = 40%)

Results with 14 Features

The presentation of the results starts by assuming a failure detection percentage of 0. In this scenario, as previously explained, features are always present in the image, and EPnP is employed throughout. As a result, the computation of the spherical coordinates for the symmetry axis and $\frac{\Gamma^{(0)}}{A}$ is omitted throughout the entire operational phase. Consequently, the graphical representation of these coordinates is not included. The convergence plots for the attitude and the inertia parameters of the target are displayed in Fig. 4.15. The next case under consideration is the one characterized by the worst failure detection rate, specifically 40%. In this particular scenario, there is one time step where no features are detected in the image, while there are 281 cases where EPnP measurements are inaccurate leading to the use of measurements provided by BCPD. In the only time step where no feature is present, the error of the $\mathbf{x}_{a,1}$ state with respect to its ground truth is less than the value of the 3σ . The convergence plots are shown in Fig. 4.16.

By looking at the plots it is possible to understand that the convergence is achieved for all the variables.

4.2.3. Monte Carlo Analysis

In this section, the results of the Monte Carlo analysis are presented, which is conducted to assess the robustness of the algorithm. This analysis involves the consideration of different initial conditions for the state $\hat{\mathbf{x}}_{a,2}$, as well as the variation of both the number of features (ranging from 1 to 14) and the value of the failure detection percentage. A total of 1000 samples are generated based on the initial state and its initial covariance. Subsequently, a selection of the 100 samples farthest from the mean value is made for their use in the Monte Carlo analysis. Within this analysis, statistical simulations are employed to replicate the measurements provided by BCPD. This approach is adopted to prevent the use of BCPD due to timing constraints.

Statistical Simulation of BCPD

The statistical simulation of BCPD measurements was conducted by employing 10 samples and taking into account the error between the measurements provided by the algorithm and the ground truth. Each measurements of the samples were divided into two groups: the first group encompassed the initial 500 measurements, while the second comprised the last 500. This division was implemented because it is expected that initial measurements provided by BCPD would exhibit higher inaccuracy compared to the later ones. This discrepancy derives from the initial inaccuracies in estimating the target's attitude and

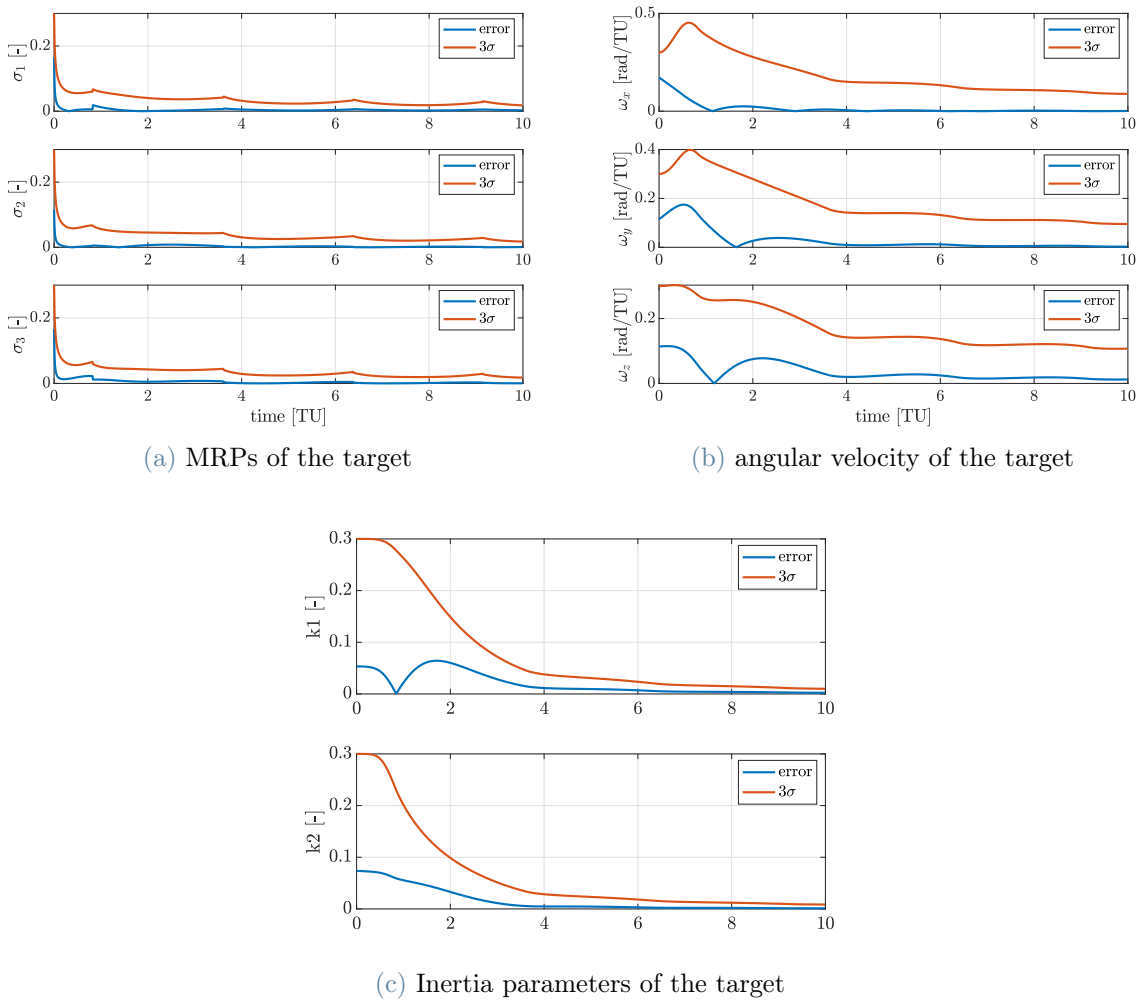
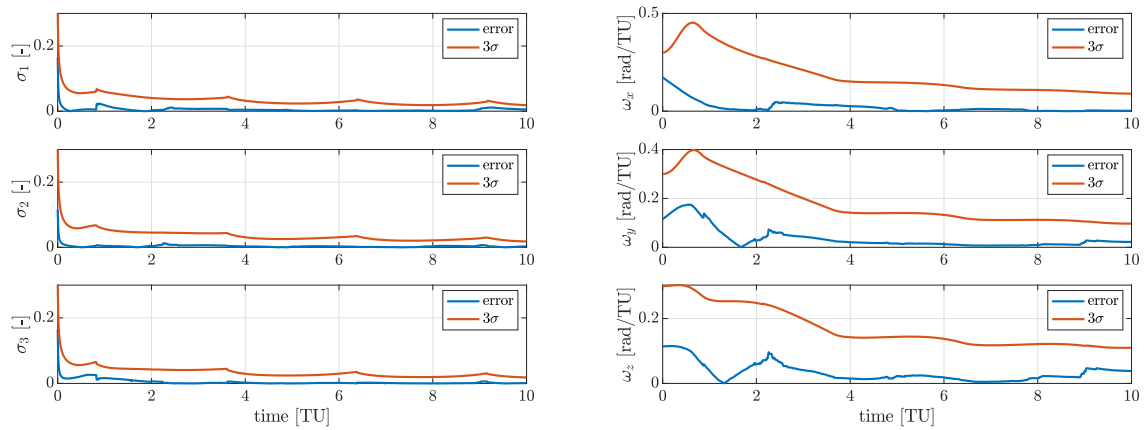
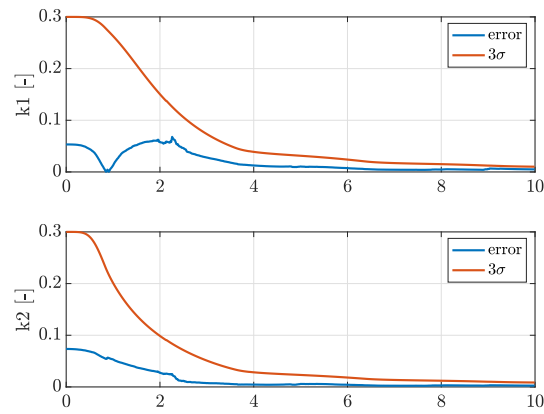


Figure 4.15: Convergence of the rotational filter (14 features and failure detection percentage = 0%)



(a) MRPs of the target

(b) angular velocity of the target



(c) Inertia parameters of the target

Figure 4.16: Convergence of the rotational filter (14 features and failure detection percentage = 40%)

relative position.

The simulation of relative position measurement starts by considering all initial 500 errors from each sample. Subsequently, the covariance is calculated based on these errors. The same procedure is followed for the last 500 errors of each sample. To simulate BCPD measurements of the relative position, the Matlab command *mvnrnd* is employed, where the mean value is zero and the previously calculated covariance is used. This random value is then added to the ground truth to generate the measurements. It is important to note that the covariance obtained from the initial 500 measurements of each sample is used for the first 500 simulated measurements, while the other calculated covariance is applied to the last 500 measurements.

Regarding the simulation of the orientation of the \mathbf{J} axis, the covariance and the average of the error angle between \mathbf{J} measured by BCPD and that of ground truth are computed. Again, a distinction is made between the first and second 500 time steps. For the simulation of the symmetry axis orientation measurement, an error angle is sampled for each time step using a Gaussian mixture distribution. Subsequently, a spherical cap is defined, centered around the ground truth of the orientation of \mathbf{J} and identified by the sampled error angle. Within this cap, any point on the circumference can be chosen by randomly sampling an angle between 0 and 2π . This chosen point dictates the direction of the measured \mathbf{J} .

Results of the Analysis

The objective is to examine whether altering the initial conditions of $\hat{\mathbf{x}}_{a,2}$ in a neighbourhood of the nominal value leads to convergence with the Ground truth. The convergence of a sample is assessed according to two different conditions. The first condition requires that the error between the estimated variables and the ground truth in the last 100 time steps must always be less than the value of the 3σ , while the second requires that the final error be at least one order of magnitude smaller than the initial error. The first condition is the most stringent one, establishing the failure condition of the method, while the second ensures error reduction, hence enhancing precision in the estimations. The first condition is always met except for the scenario where only one feature is available and a 40% detection failure rate is considered. In this case, the criterion is not satisfied as one time step in the last 100 of the simulation has an error larger than 3σ . Fig. 4.17 shows a graph that relates the number of converging samples with the number of features. This graph reveals that the convergence rate stabilises at 97%-98% for all failure detection percentage, and that these values are reached for all failure rates with at least 3 features.

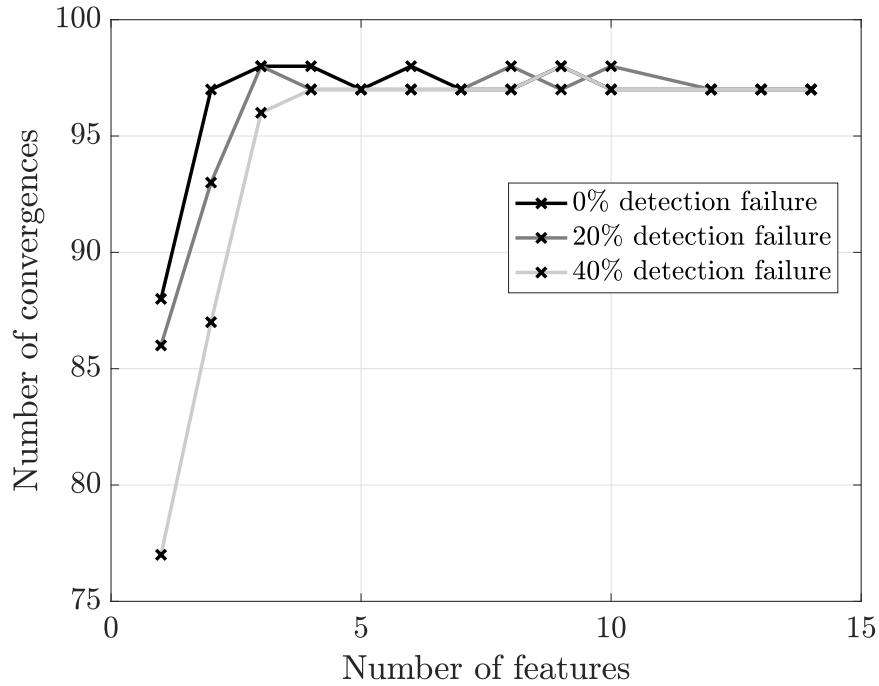


Figure 4.17: Number of convergences varying the number of features

4.2.4. Unknown Rigid Rotation

This section presents the analysis performed considering the rigid rotation matrix as an unknown, and adding its MRPs in $\hat{\mathbf{x}}_{a,1}$. As before, only the results considering 1, 6 and 14 features are shown.

The overall behavior of the plots aligns closely with that observed in the scenario involving a known rigid rotation matrix. Consequently, the considerations outlined earlier can be extended to these cases as well.

What distinguishes the situation involving an unknown rotation matrix is the representation of its MRPs. As previously indicated, MRPs are exclusively defined in the absence of features in the image, and during these periods, their values remain constant due to their dynamics (Eq. 2.45). From a graphical point of view, a singular point is plotted for each interval in which these MRPs are computed, enhancing the visualization of their convergence trend. Due to the construction of this rigid rotation matrix, $\sigma_{2\Gamma}$ consistently remains at zero. Consequently, the parameter plot associated with it is disregarded as it would yield no meaningful information, consistently showing an error of zero. The results obtained with a failure detection probability of 20% are given in Appendix B.

Results with 1 Feature

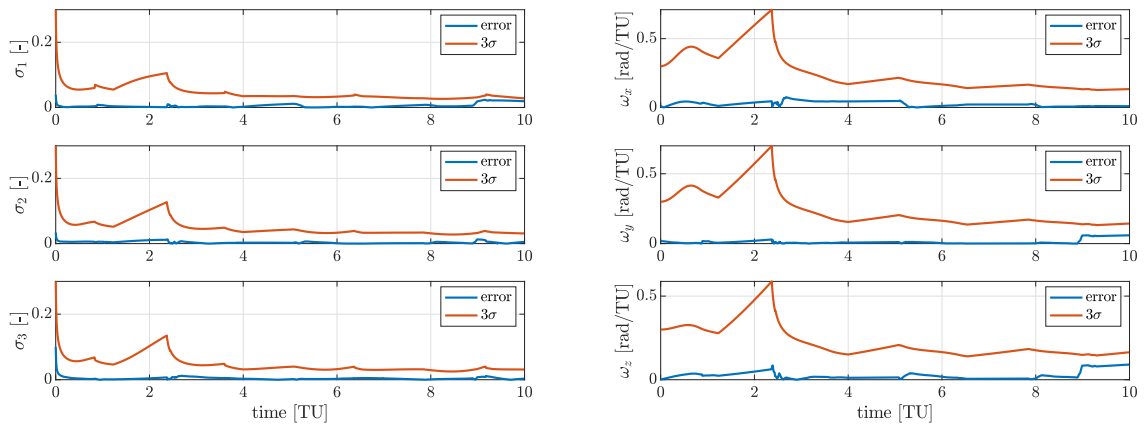
In the scenario where only one feature is present on the target, the utilization of EPnP is evidently not feasible. Considering the failure detection probability of 0% (Fig. 4.18), there are 423 time steps in which no features are available, allowing for the accurate reconstruction of only the orientation of the symmetry axis of the target. This count rises to 638 when the failure detection probability increases up to 40% (Fig. 4.19).

Results with 6 Features

Alternatively, in a situation with a 0% (Fig. 4.20) failure probability, features are consistently present in the images, although their limited availability results in frequent rejection of EPnP measurements, leading to a switch to BCPD. This transition occurs 219 times. Introducing a failure probability of 40% (Fig. 4.21) yields 59 instances where no features are detected in the image. Regarding EPnP, its use diminishes due to non-feature detection, resulting in a reduced number of transitions to BCPD, specifically 29 times. This happens because in many situations where four features were present, having a failure in a detection leads to a reduction in the number of times that EPnP can be used.

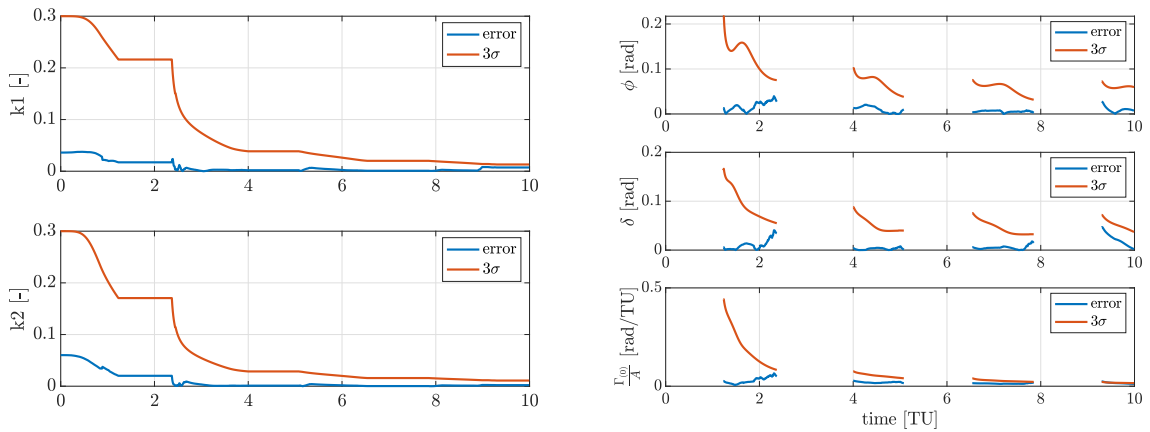
Results with 14 Features

This test case marks the threshold where, under a 0% (Fig. 4.22) probability of failure detection, features are consistently present, and only EPnP is employed for measurements. Increasing the failure detection probability to 40% (Fig. 4.23) introduces a single time instant without available features and 281 steps in which the switch between EPnP and BCPD takes place. During the singular time step where no features are available, the error between the estimated symmetry axis orientation and the corresponding ground truth remains below 3σ . This also holds true when evaluating the MRPs of the rigid rotation matrix.



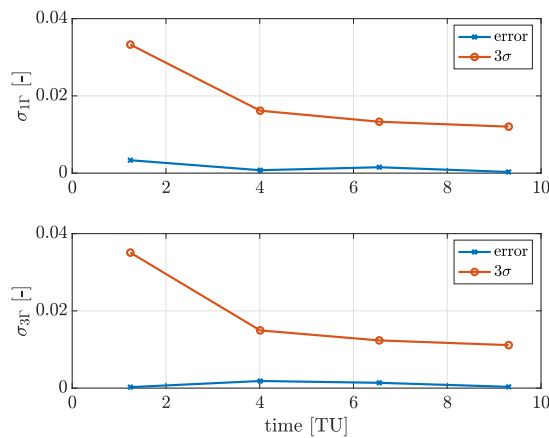
(a) MRPs of the target

(b) angular velocity of the target



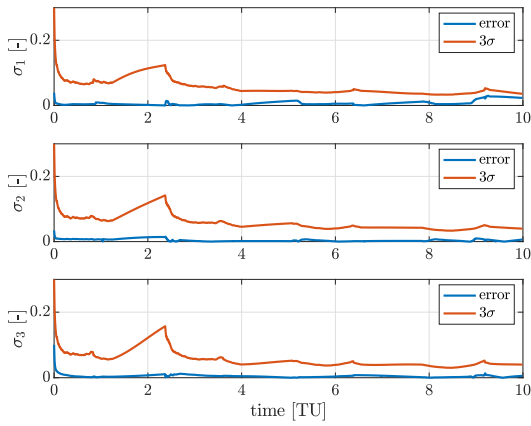
(c) Inertia parameters of the target

(d) Target's symmetry axis spherical coordinates in \mathcal{I}_Γ frame and $\frac{\Gamma(0)}{A}$ ratio

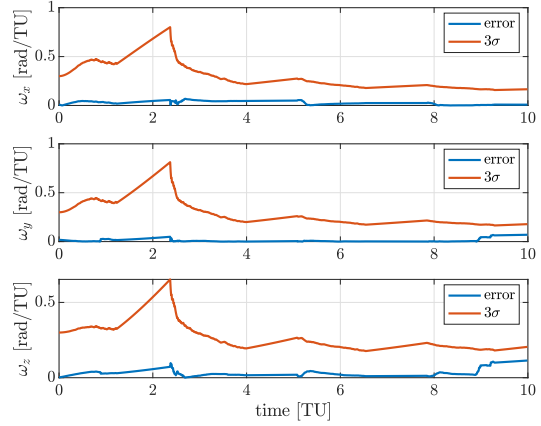


(e) MRPs of the rigid rotation matrix

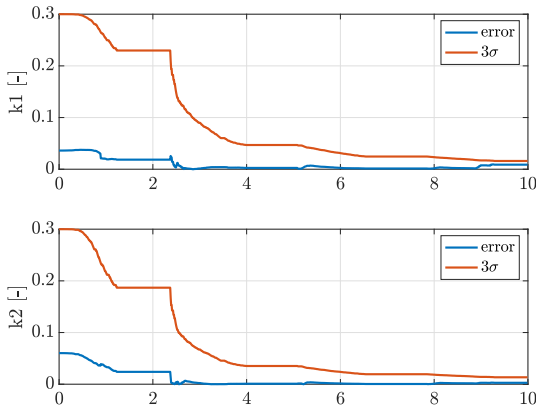
Figure 4.18: Convergence of the rotational filter with unknown rigid rotation matrix(1 features and failure detection percentage = 0%)



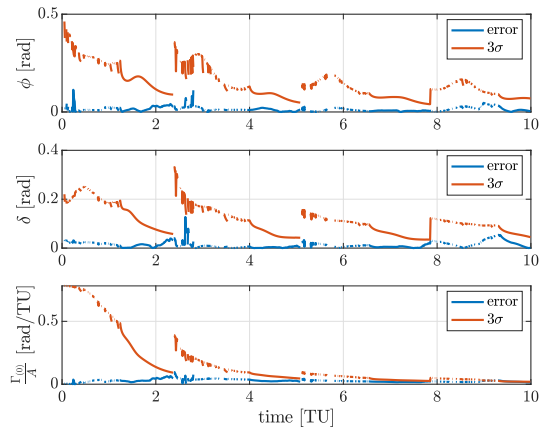
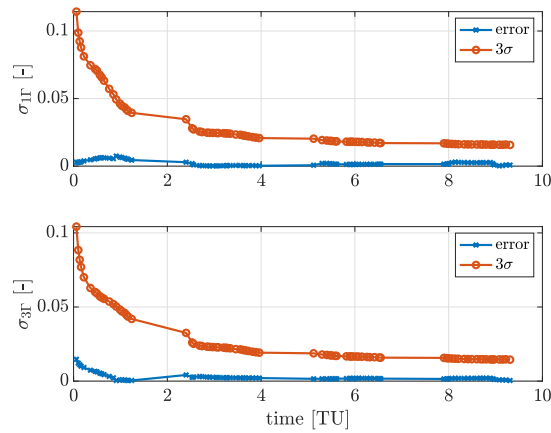
(a) MRPs of the target



(b) Angular velocity of the target



(c) Inertia parameters of the target

(d) Target's symmetry axis spherical coordinates in \mathcal{I}_T frame and $\frac{\Gamma(0)}{A}$ ratio

(e) MRPs of the rigid rotation matrix

Figure 4.19: Convergence of the rotational filter with unknown rigid rotation matrix (1 features and failure detection percentage = 40%)

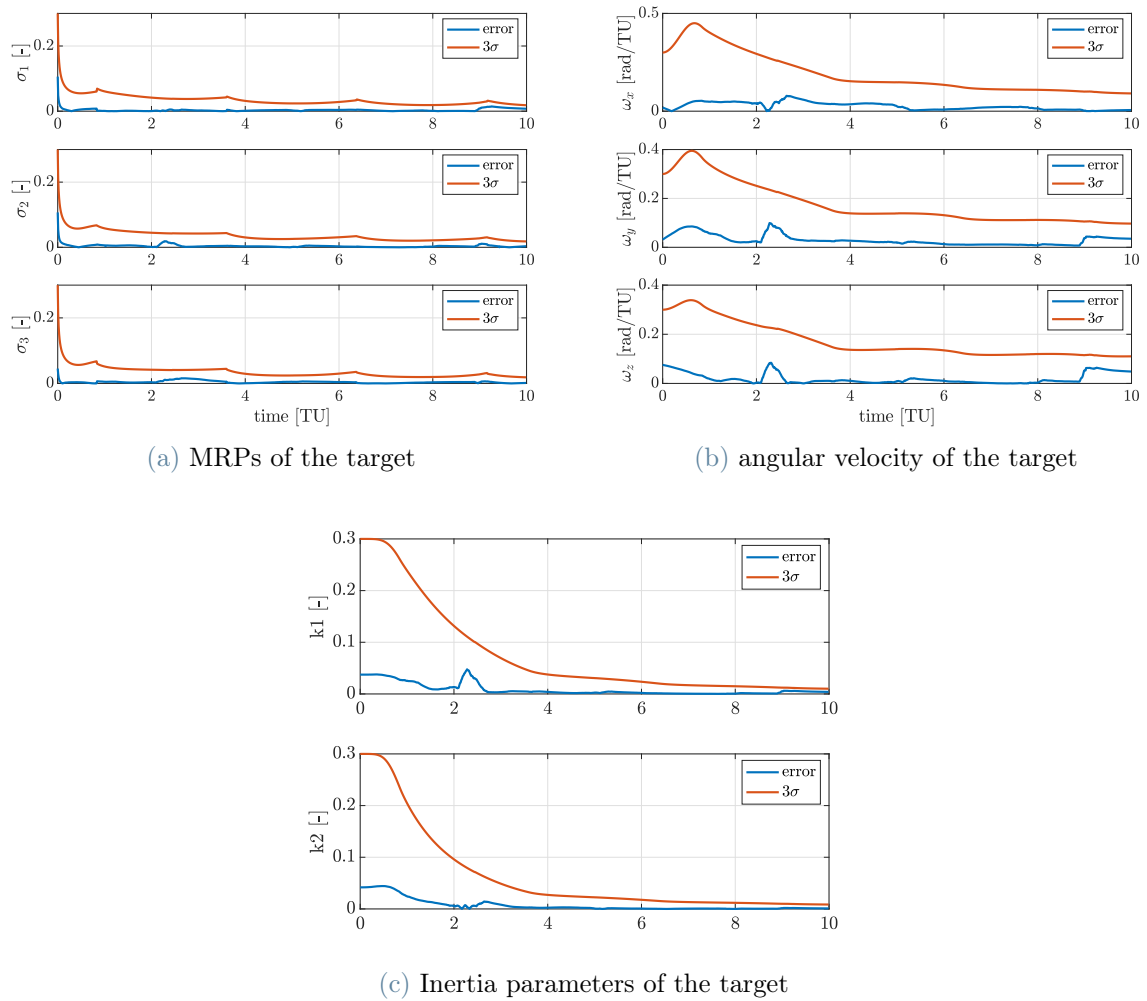
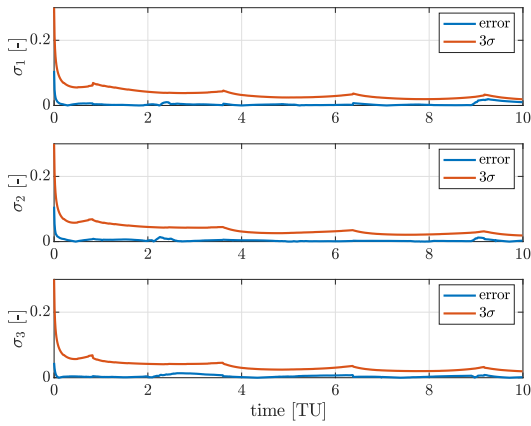
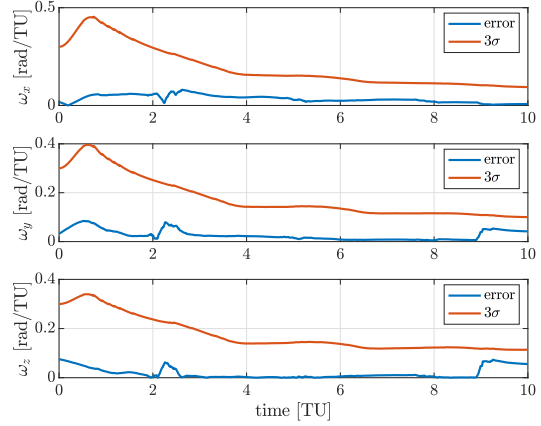


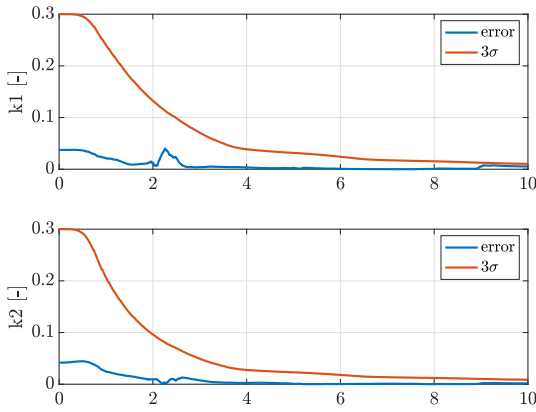
Figure 4.20: Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 0%)



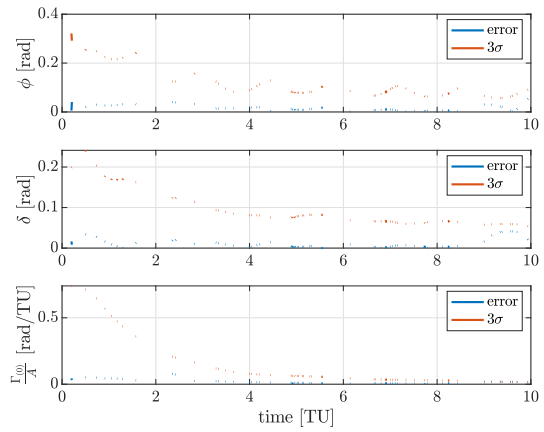
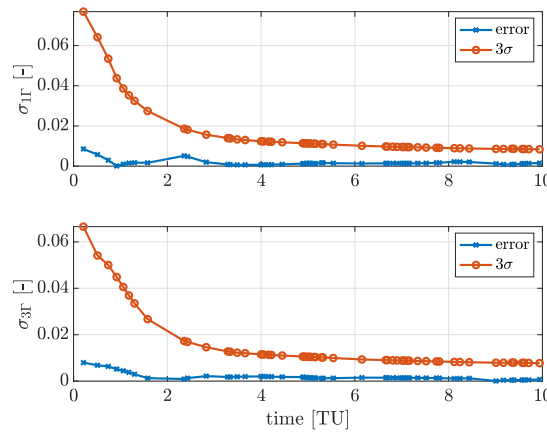
(a) MRPs of the target



(b) angular velocity of the target

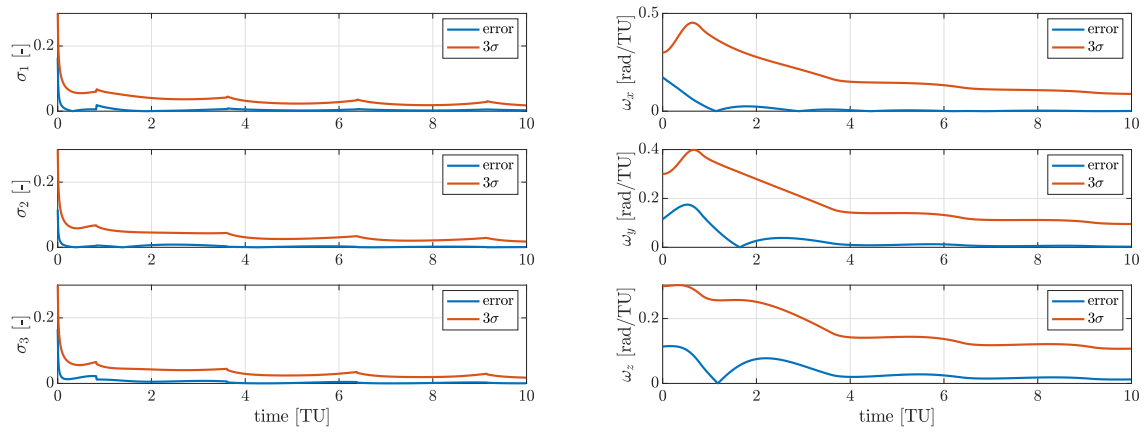


(c) Inertia parameters of the target

(d) Target's symmetry axis spherical coordinates in \mathcal{I}_T frame and $\frac{\Gamma(0)}{A}$ ratio

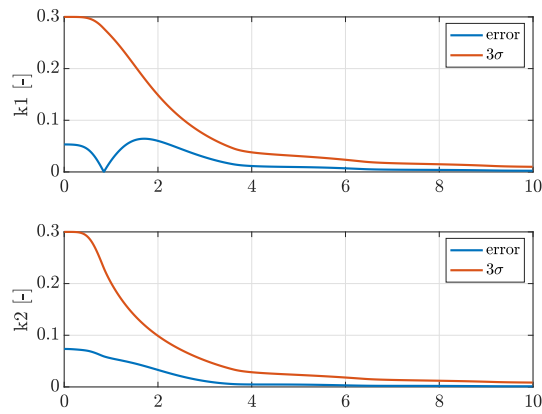
(e) MRPs of the rigid rotation matrix

Figure 4.21: Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 40%)



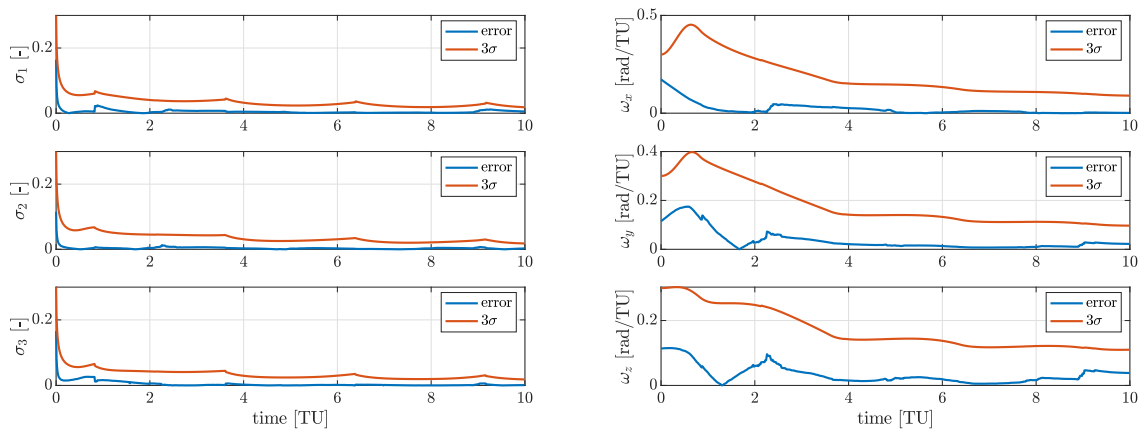
(a) MRPs of the target

(b) angular velocity of the target



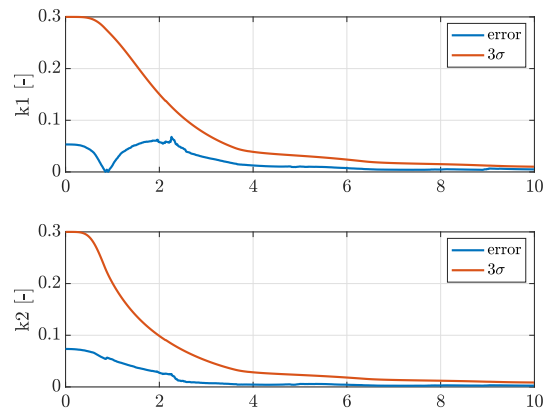
(c) Inertia parameters of the target

Figure 4.22: Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 0%)



(a) MRPs of the target

(b) angular velocity of the target



(c) Inertia parameters of the target

Figure 4.23: Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 40%)

5 | Features Detection

The objective of this chapter is to illustrate that through neural network training, it is possible to identify specific features, such as a NASA or ESA sticker, on a symmetrical target. These features correspond to the same ones that were previously considered as known and integrated into the algorithm discussed in the preceding chapter. The goal, therefore, is to generate output consisting of the coordinates of the centers of bounding boxes associated with these features. These coordinates can subsequently be used in the algorithm in the previous sections. Initially, the process of creating the image data-set, used for training and testing the neural network, is described in detail. Subsequently, the training process is described and its results are presented. Finally, an overview of the results obtained from the image test set is provided.

5.1. Image Generation Process

When training a neural network, the most effective approach to enhance its ability to confidently detect objects involves providing a vast data-set of images. Thus during the training process, the network will learn to detect objects under many different conditions. A CAD model of a satellite and Blender¹, which is a 3D computer graphics software toolset, are used to generate a sufficiently large data-set of images.

In this work, the satellite model shown in Figure 5.1 is used, which is different from the symmetrical RSO used in the Chapter 4. The reason why the neural network was trained on a different model with respect to the real one, is to simulate the case of an unknown target. Since the focus of this work is relative navigation around unknown targets, no prior knowledge of the target audience is available. Consequently, it is not possible to provide a model of the unknown target to the network for the training process.

Once the CAD model was loaded into Blender, features were manually incorporated onto its external surface. The features were chosen to represent reality as much as possible. It was decided to add flags of different states, the symbols of some space agencies and LEDs. The first two categories of objects are detailed in table 5.1, and were uploaded

¹Blender official site: <https://www.blender.org>



Figure 5.1: Model of the satellite

to Blender as images. As for the LEDs, on the other hand, they were modelled directly within the Blender environment, with two distinct colors, red and green.






Features	Figure
Italy flag	
USA flag	
Germany flag	
NASA symbol	
ESA symbol	

Table 5.1: Features

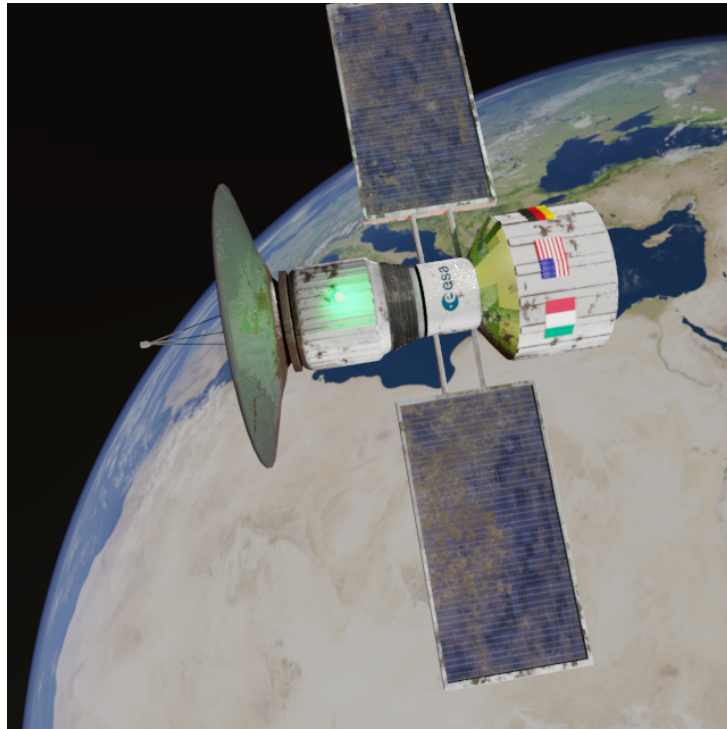


Figure 5.2: Model 5

Five different models were created to establish different conditions for the objects. On each model, the arrangement, quantity, and size of the objects are different. The five models created, with their respective features, are shown below:

- **Model 1:** 1 green led, 1 red led, 1 USA flag, 1 NASA symbol and 1 ESA symbol
- **Model 2:** 1 green led, 1 Germany flag, 1 USA flag, 1 Italy flag and 1 ESA symbol
- **Model 3:** 1 Germany flag, 1 Italy flag, 1 USA flag, 2 NASA symbol and 1 ESA symbol
- **Model 4:** 1 green led, 1 red led, 1 Germany flag, 1 NASA symbol and 1 ESA symbol
- **Model 5:** 1 green led, 1 red led, 1 Germany flag, 1 USA flag, 1 Italy flag, 1 NASA symbol and 1 ESA symbol

Figure 5.2 shows one of these models.

JINS, introduced in [24], is used to obtain the images. The tool is able to automate the rendering of images, and is composed of three different steps. After a pre-processing phase, which is the first step, in which the spacecraft model needs to be adjusted in order to be compliant with the requisites of the successive piece of software, the code generates

```

5 0.456250 0.404687 0.015625 0.015625
4 0.607812 0.392188 0.028125 0.062500
2 0.760156 0.433594 0.042188 0.057813
0 0.728906 0.289844 0.051562 0.023438
1 0.749219 0.350781 0.048438 0.057813

```

Figure 5.3: File *.txt* used by YOLOv8

the different images with the respective masks of the objects in the image. These masks delineate the regions within the image where the objects are located. Finally, using the images and the masks, a second script generates an annotation *.json* file containing information about the image and the objects in the image.

JINS employs a monocular camera for image generation. Conversely, CoMBiNa adopts a stereo camera. Nevertheless, as previously stated, the objective of this chapter is to demonstrate the possibility of detecting specific objects on the satellite’s outer surface. Hence, it is possible to use any type of camera for this purpose.

Now, using another code, the *.json* file is converted into the format required by YOLOv8, which is the network that is used in this work. This format is a *.txt* file that has a line for each object contained in the image, in which the class number and the coordinates of the bounding box enclosing the object are given. For YOLOv8, the coordinates of the bounding box are its centre, width and height. These coordinates must also be normalised with respect to the size of the image. An example of a *.txt* file is shown in Figure 5.3.

After generating 1200 images, 240 for each of the models presented above, and 1200 *.txt* files containing the bounding boxes, they are uploaded in Roboflow². It is a toolset designed for computer vision and image processing tasks. Roboflow was used because it allows, in a very simple way, to divide the entire data-set into train, validation and test set, and to do data augmentation. Another advantage is that, once the data-set has been generated, Roboflow outputs lines of code that can be used to download the data-set, directly from the site itself, during the training process.

Out of the 1200 uploaded images, 51 are discarded because they are “not annotated”. The remaining 1149 images are divided into training, validation and test set as follows:

- **Training set:** 806 images (70%)
- **Validation set:** 227 images (20%)
- **Test set:** 116 images (10%)

²Roboflow official site: <https://roboflow.com/>

It is useful to analyse the data-set in more detail to see whether the object classes are well represented. Among the 1,149 images, there are a total of 3,178 annotations distributed across seven distinct classes. Table 5.2 shows the numbers of annotations present for each class. The data reveals a slight under representation of the FlagITA and LEDRed classes.

Class	Number of annotations
FlagGER	492
FlagITA	323
FlagUSA	443
LEDGreen	486
LEDRed	315
SymbolESA	606
SymbolNASA	513

Table 5.2: Reference frames

The final procedure that can be executed in Roboflow, before generating the entire data-set, involves data augmentation. This method is employed to expand the quantity of images in the training set by implementing various transformations on the initially available images. In this work, both horizontal and vertical flips were applied to the entire image as well as to the bounding boxes.

5.2. Training Process

One-stage detection methods are preferred over region proposal methods due to their superior computational efficiency, a critical requirement for enabling real-time operations in this specific application. In this study, the YOLO family of one-stage methods was selected. As discussed in Chapter 3 different versions of YOLO exist, and Figure 5.4 provides a comparison of some of them. The figure on the left illustrates how the mAP changes in relation to the number of parameters in the YOLO version, while the figure on the right shows the mAP variations with respect to the speed of the method. It's evident that the most recent version, YOLOv8, stands out as the optimal choice, and is consequently adopted for this work.

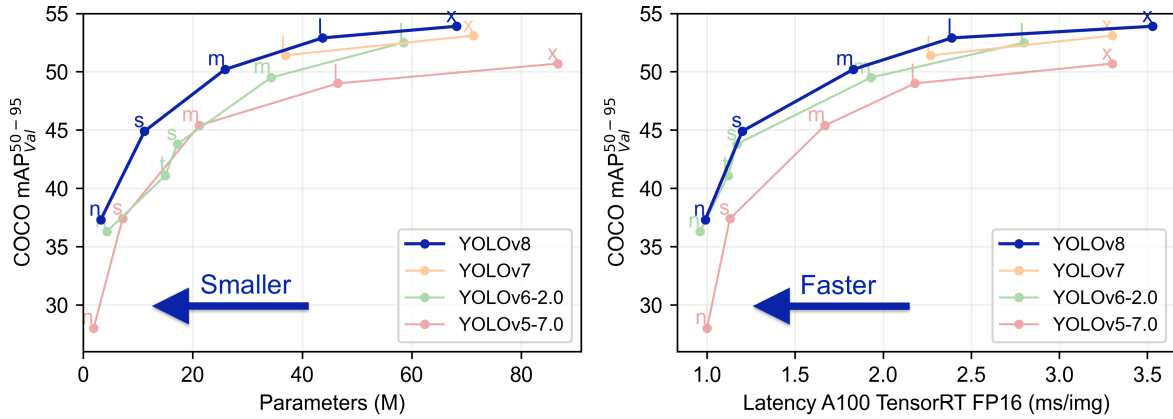


Figure 5.4: Comparison between different YOLO's architectures (courtesy of [37])

As previously mentioned, Ultralytics introduced five scaled versions of YOLOv8. They are shown in Table 5.3, highlighting their variations in size and performance metrics achieved by evaluating these models on the COCO data-set. For the purposes of this work, the pre-trained model employed is YOLOv8s, as it represents a good trade-off between mAP and execution speed. YOLOv8s demonstrates a significant enhancement in mAP compared to the smaller model, with only a moderate increase in execution time. Conversely, when considering the larger architectures, the improvements in mAP become progressively smaller, while the corresponding increases in execution time grow in a more significant way.

Model	size (pixels)	mAP_{50-95}^{val}	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	0.99	3.2	8.7
YOLOv8s	640	44.9	1.20	11.2	28.6
YOLOv8m	640	50.2	1.83	25.9	78.9
YOLOv8l	640	52.9	2.39	43.7	165.2
YOLOv8x	640	53.9	3.53	11.2	257.8

Table 5.3: Comparison between the different available model of YOLOv8

The training process was carried out using a number of epochs equal to 150. Initially, 100 epochs were employed, but the trends observed in the loss functions indicated that further reductions in losses were possible, potentially leading to improved network performance.

Consequently, the number of training epochs was extended to 150, and the behavior of validation losses begins to stabilize.

During training, YOLOv8 uses a generalisation technique for neural networks called early stopping, which aims to prevent overfitting. Among the parameters characterising early stopping, the most important is the patience. This parameter determines after how many epochs the training process must be stopped if the losses do not decrease. The standard value of the patience used by YOLOv8, when not explicitly specified in the input, is typically set at 50. However, for a more robust prevention against overfitting, a value of 20 is chosen.

5.2.1. Training Results

The training of YOLOv8 provides as output some charts that are useful to better understand the training process and the expected accuracy of the model. The first chart analysed shows some key metrics tracked by YOLOv8 (Fig. 5.5). Specifically, the focus is on the Box Loss and Class Loss for both the training and validation data-sets. Fig. 5.5 shows that the behaviour of this loss is correct and that the model is converging. The trends observed in the Box and Class Loss metrics suggest that these values decrease as the training epochs progress.

The second chart analysed is the Confusion Matrix, which shows how the model handles different classes. From this matrix, it is possible to see how often an object of a certain class is correctly detected, how often it is incorrectly detected as an object belonging to another class and how often it is confused with the background and therefore not detected. Fig. 5.6 shows the confusion matrix for the training process at hand. The confusion matrix is diagonal, which means that the classes are detected correctly, and they are not confused with other classes. There is only one term not on the diagonal indicating that in 2% of the cases the flag of Italy is confused with the one of the USA. This is an example of wrong feature classification. It should not represent a problem since the confidence level of this detection is quite low, and by increasing the threshold value of the confidence level it is discarded. However, this type of failure represents a possible direction for future research.

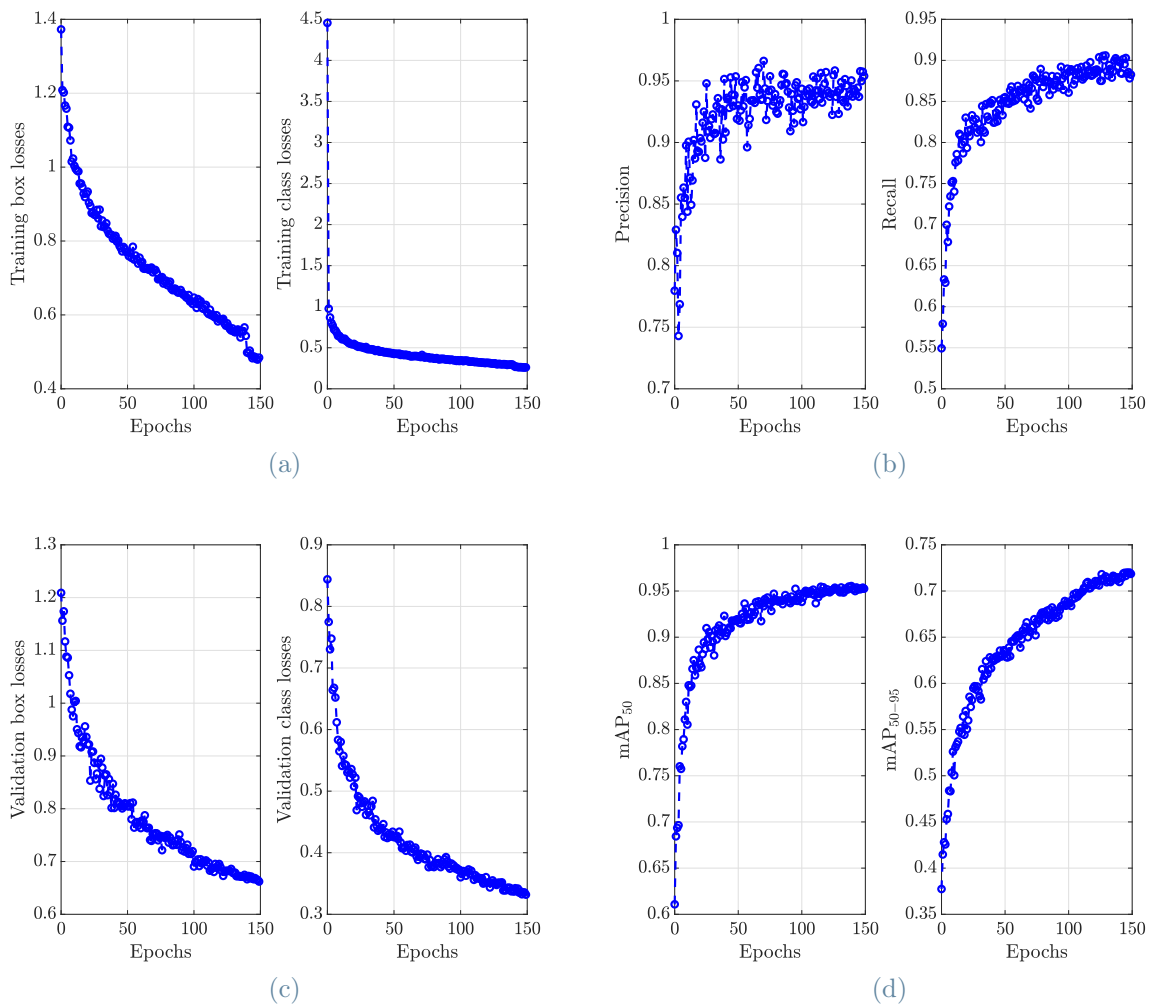


Figure 5.5: Losses and metrics of the training process

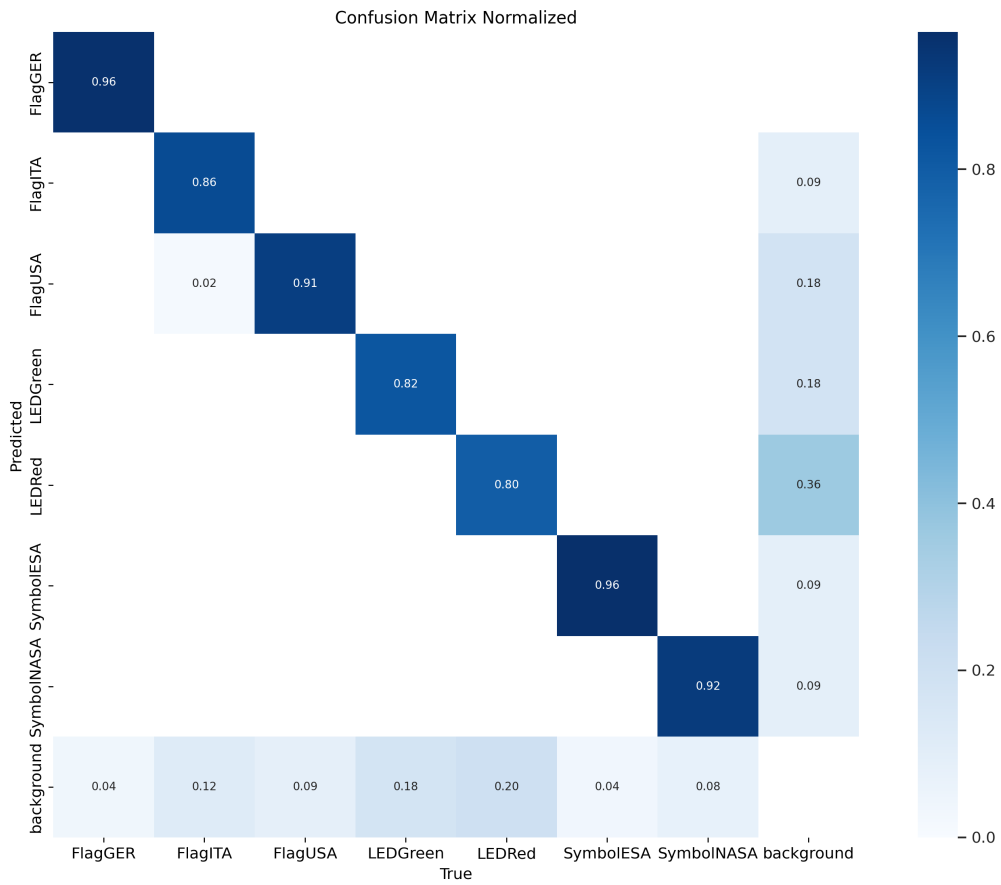


Figure 5.6: Confusion matrix

Another chart that provides useful information for understanding the performance of the network is the precision-recall curve, shown in Fig. 5.7. This curve shows the trade-off between precision and recall for an IoU threshold of 50. In Fig 5.7, the area under the curve has a very large value, representing both high precision and recall. High precision relates to a low false positive rate, and high recall relates to a low false negative rate.

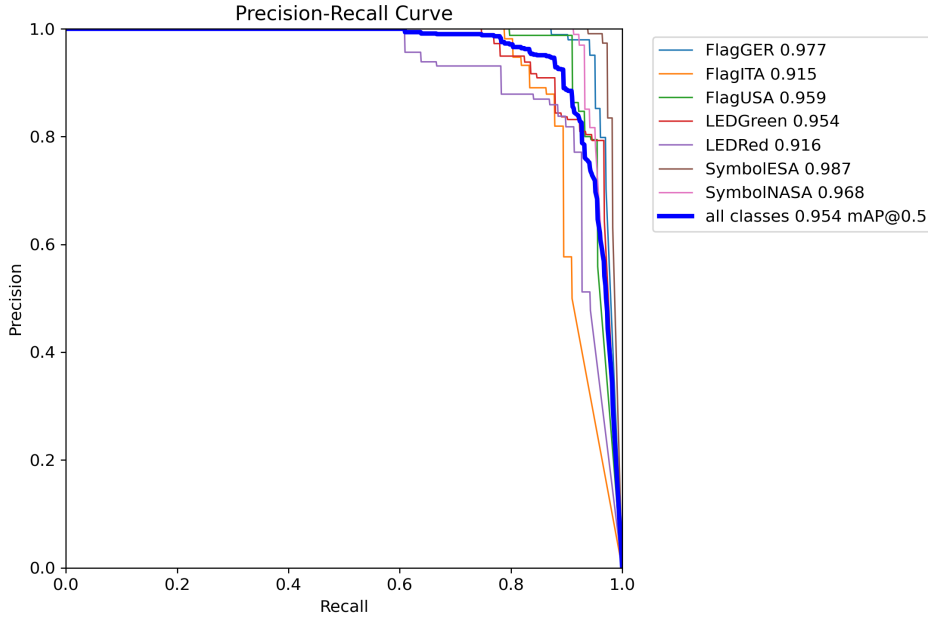


Figure 5.7: Precision-Recall curve

Finally, the trained model is subjected to validation using a set of images not previously used during YOLOv8's training. Through validation, the AP values for each class and the mAP are computed. The evaluation is carried out at different IoU thresholds. Initially, AP_{50} and mAP_{50} are computed, where IoU threshold is set to 50. Subsequently, this analysis is extended to AP_{50-95} and mAP_{50-95} . In this case, 10 different APs and mAPs are computed, each using a different IoU threshold (ranging from 50% to 95% with a 5% increment). Finally, the values of AP_{50-95} and mAP_{50-95} are determined as the averages of these 10 calculated values. The AP computed are presented in Tab. 5.4 and the mAP are reported in 5.1.

Class	AP_{50}	AP_{50-95}
FlagGER	0.977	0.823
FlagITA	0.905	0.737
FlagUSA	0.959	0.779
LEDGreen	0.959	0.597
LEDRed	0.914	0.527
SymbolESA	0.986	0.778
SymbolNASA	0.968	0.792

Table 5.4: Average Precisions

$$\text{mAP}_{50} = 0.953 \quad \text{mAP}_{50-95} = 0.719 \quad (5.1)$$

5.3. Testing Results

The trained network is evaluated using a test set of images, allowing for an assessment of its performance on previously unseen images. In this initial analysis, the satellite model employed to create the test set is equal to the one on which the network was trained.

This section assesses several metrics, including the IoU, the pixel-wise error between the center of the predicted bounding box and the ground truth, and the percentage of objects that exist in the ground truth but were not identified by the network.

Fig. 5.8 shows two representations of how YOLOv8 generates output images. In these images, a bounding box is delineated around the detected object, the corresponding label is assigned to it, and the detection's confidence level is indicated. The confidence level is a measure of the network's confidence that the assigned label is accurate for the detected object. Consequently, the higher the value of the confidence level, the more reliable the detection. The confidence level chosen in this work is 0.4. This means that all the detection that have a confidence level lower than 0.4 are discarded.

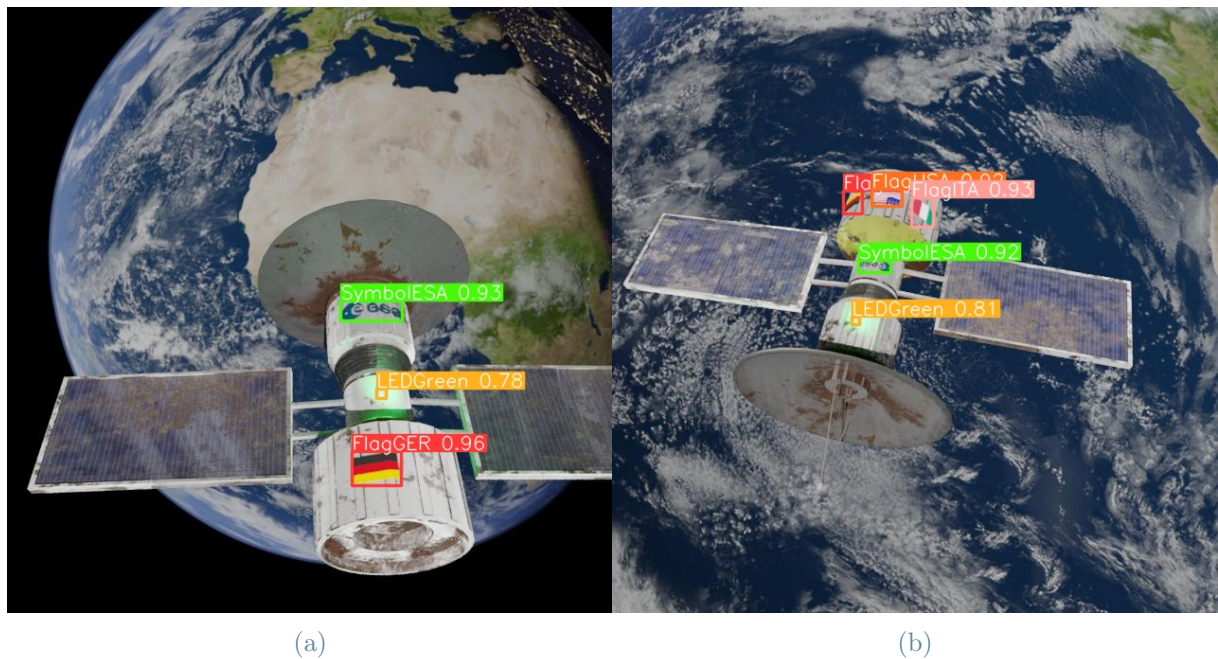


Figure 5.8: Output images generated by YOLOv8

The first metric that is analysed is the IoU. For every object detected within the image,

this metric is computed using Eq. 3.1. Subsequently, the average value of these IoU indices is determined, resulting in the following:

$$\text{Mean IoU} = 85.8203\% \quad (5.2)$$

The obtained value indicates that, on average, the predicted bounding boxes of the network overlap with those of the ground truth to a great extent.

The second metric under examination is the pixel-wise error between the predicted bounding boxes and the ground truth. This error is analyzed because the neural network provides as input to CoMBiNa the position of the features in the image, specifically in terms of the center of the bounding box. Hence, minimizing the error between the predicted and true centers is crucial. For each predicted bounding box, errors along both the x-axis of the image (err_x) and the y-axis (err_y) are computed. Subsequently, the mean error and standard deviation concerning both the x and y axes of the image are determined, leading to the following results:

	Mean Value (pixels)	Standard Deviation (pixels)
err_x	0.7342	1.0906
err_y	0.4758	0.9052

Table 5.5: Mean errors and standard deviations

The mean errors for both the x and y axes of the image are less than one pixel. This indicates that the network accurately predicts the center of the bounding box within the image and, consequently, CoMBiNa receives highly precise feature information, thereby enhancing its ability to accurately determine the target's orientation.

Graphically, Fig. 5.9 displays histograms illustrating the distribution of these errors.

Finally, an assessment regarding the failure to detect objects is conducted, which accounts for cases where an object exists within the image, but the network fails to identify it. This analysis is carried out individually for each object class, and the non-detection percentage is calculated as the ratio between the number of times an object of a class is not detected and the total number of times the object of that class appears in all images. The outcomes are presented in Tab. 5.6. The evaluation of non-detection percentages by the network is used to validate the correctness of the introduced detection failure probabilities outlined in Chapter 4 (0%, 20%, 40%). The failure percentage presented in the Tab. 5.6 for different object classes are compliant with the values employed during the assessments in Chapter 4.

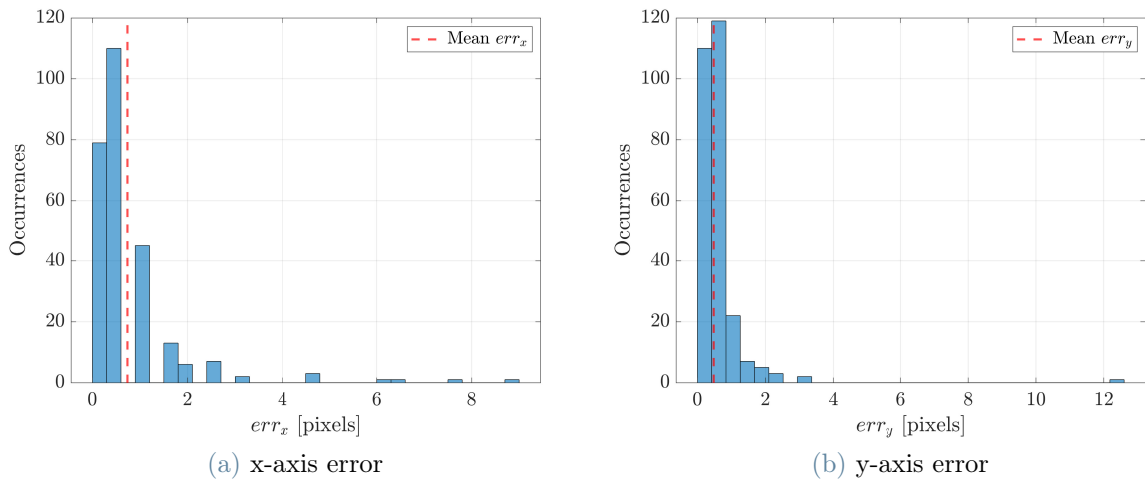


Figure 5.9: Error distribution in bounding box centre detection

Class	Failure detection percentage
FlagGER	9.7561%
FlagITA	3.3333%
FlagUSA	4.6512%
LEDGreen	36.3636%
LEDRed	19.3548%
SymbolESA	6.7797%
SymbolNASA	8.4746%

Table 5.6: Failure detection percentage for each class

Another type of issue is the identification of an object that doesn't exist in the image. To determine its occurrence, the total number of objects that are detected by YOLOv8 is calculated and this value is added to the total number of miss detections. The result obtained is then compared to the total number of objects in the test set. If the two numbers are equal, this failure condition does not occur, otherwise it does. In this scenario, the quantity of labels predicted by the network exceeds the number of labels present in the test set. As a result, YOLOv8 produces 4 additional detections, implying that 4 objects are being identified even though they don't exist in reality.

To address this problem, the proposed solution is to increase the confidence threshold of the network for object detection. This is because when the network confuses two objects, the confidence value is likely to be low, and increasing the threshold can help reduce the identification of non-existing objects. This condition is very rare, which is why it was not

dealt with in Chapter 4. Nonetheless, it represents a starting point for possible future study.

6 | Conclusions and Future Works

This chapter contains a brief summary of what is contained in this work, also presenting some ideas for possible future work.

6.1. Conclusions

CoMBiNa, presented in [45], consists of a new technique to tackle the complicated problem of relative navigation around unknown and uncooperative targets. CoMBiNa is designed to handle any types of RSOs, both asymmetrical and symmetrical. Regarding asymmetric targets CoMBiNa works well and provides very good results, while considering symmetric targets the method fails. The reason is that symmetric RSO have unobservable rotations around the symmetry axis, which cause the BCPD algorithm to converge incorrectly. To address this problem, a reformulation of CoMBiNa is presented in [20], where the variables within the attitude state vector are changed, approximating the RSO to a gyroscopic structure. In this way, the orientation of the symmetry axis of the target in the inertial system can be derived. This can be done since in the matrix describing the orientation of the target in the inertial system derived with BCPD, the second row, which describes the orientation of the axis of symmetry, remains accurate.

This study aims to achieve a complete determination of the attitude, even in the case of symmetrical targets. To accomplish this, an integration of a neural network into CoMBiNa is implemented, enabling the detection of features positioned on the surface of the target in images obtained from a stereo camera. This process effectively breaks the symmetry of the target, making possible the reconstruction of its complete attitude.

Chapter 4 outlines the modifications made to the CoMBiNa algorithm to incorporate the information derived from the neural network. It assumes the availability and functionality of the neural network, and consequently the measurements of the position of the features are simulated. The effectiveness of the proposed method is assessed towards the end of the chapter.

In Chapter 5, it is demonstrated that a neural network can accurately detect the position of features, such as the NASA or ESA symbol, on the outer surface of the target. YOLOv8

is trained and tested on a different target model from the one used in Chapter 4, resulting in highly satisfactory outcomes.

6.2. Future Works

When addressing the navigation aspect, it becomes essential to incorporate the neural network into CoMBiNa. This would lead to using, no longer the simulated measures, but the real ones provided by YOLOv8 and to diversify the failure detection probabilities, which instead in this work are assumed equal for the different feature classes. Subsequently, this adjustment allows for a realistic assessment of the new formulation of CoMBiNa.

This work also exhibits certain limitations related to the training and testing of YOLOv8. The training process was conducted using a single model, which was then utilized for the testing phase. However, given the consideration of unknown RSOs, it would be advantageous to evaluate the performance of the network on images containing a target different from the one used during training. In the training phase, the incorporation of different satellite models aims to introduce as much variability as possible into the image conditions.

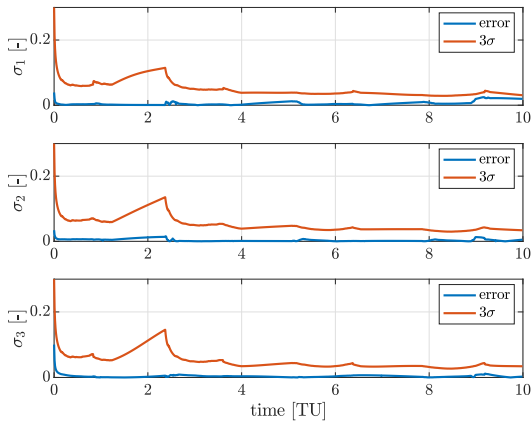
Another problem, which arises in the feature detection part, concerns the identification of features when two of them belong to the same classes, e.g., in case there are two NASA symbols. In this work this case is dealt with in a post-processing part of the results, however, it is necessary to find a way to identify them a priori in order to then correctly associate them with those stored on the coarse model. Two other types of failure, which are not discussed in this paper since they occur rarely, are false positives (i.e., identification of features in wrong locations), and the possibility of wrong feature classification (i.e., wrong assignment of ID to an identified feature). However, understanding the potential influence of these two failure conditions on performances would be an interesting future research direction.

A | Appendix A

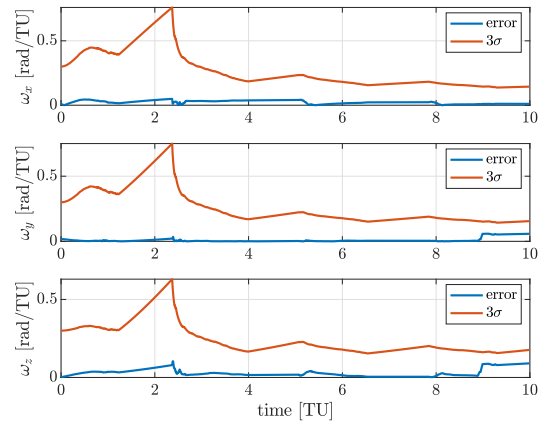
This appendix shows the results obtained using a detection failure rate of 20%, varying the number of features, in the case of known rigid rotation matrix.

	Number of time steps without features	Number of switch from EPnP to BCPD
1 feature	516	0
6 features	6	96
14 features	0	193

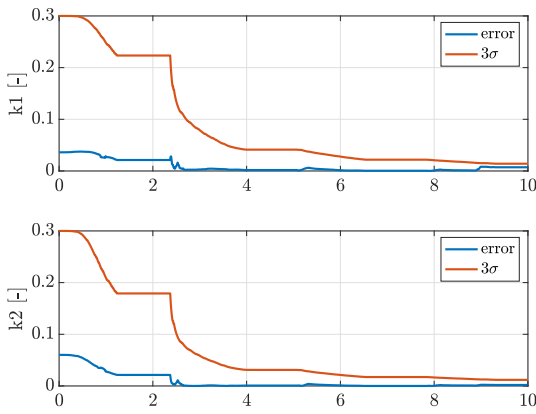
Table A.1: Number of time steps without features and with the switch from EPnP to BCPD, by varying the number of features



(a) MRPs of the target



(b) angular velocity of the target



(c) Inertia parameters of the target

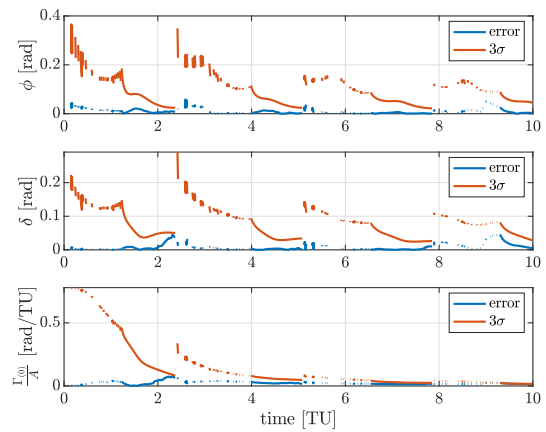
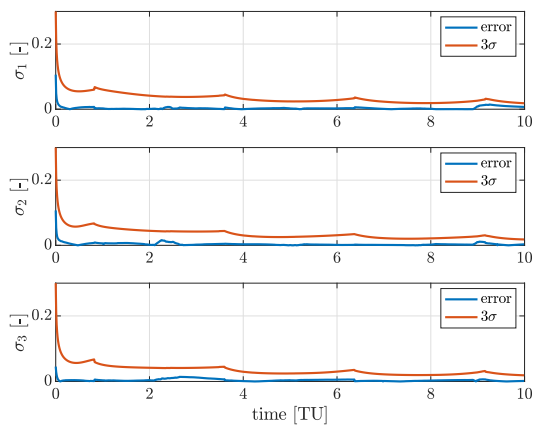
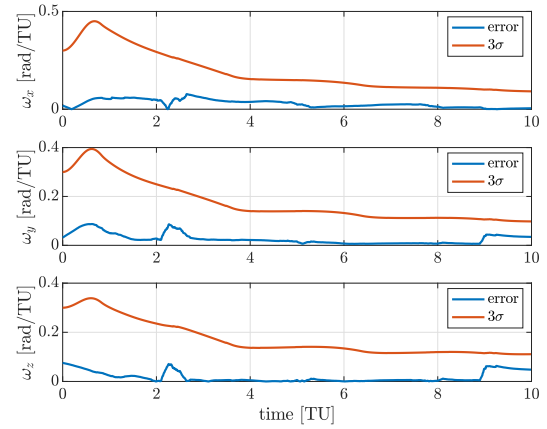
(d) Target's symmetry axis spherical coordinates in \mathcal{I}_r frame and $\frac{\Gamma(0)}{A}$ ratio

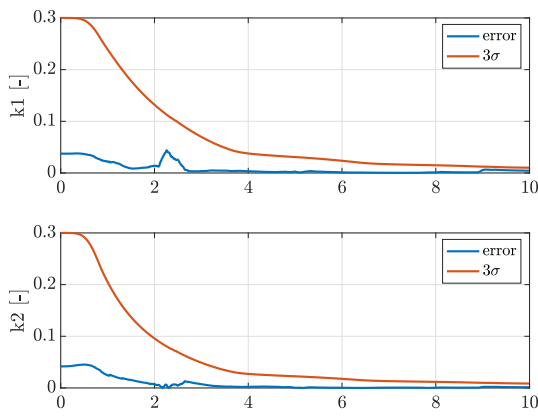
Figure A.1: Convergence of the rotational filter with unknown rigid rotation matrix (1 features and failure detection percentage = 20%)



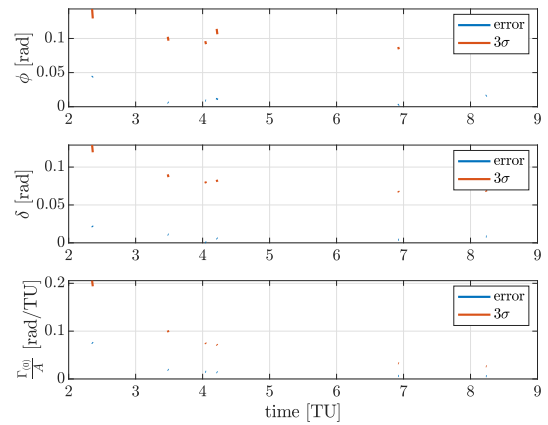
(a) MRPs of the target



(b) angular velocity of the target

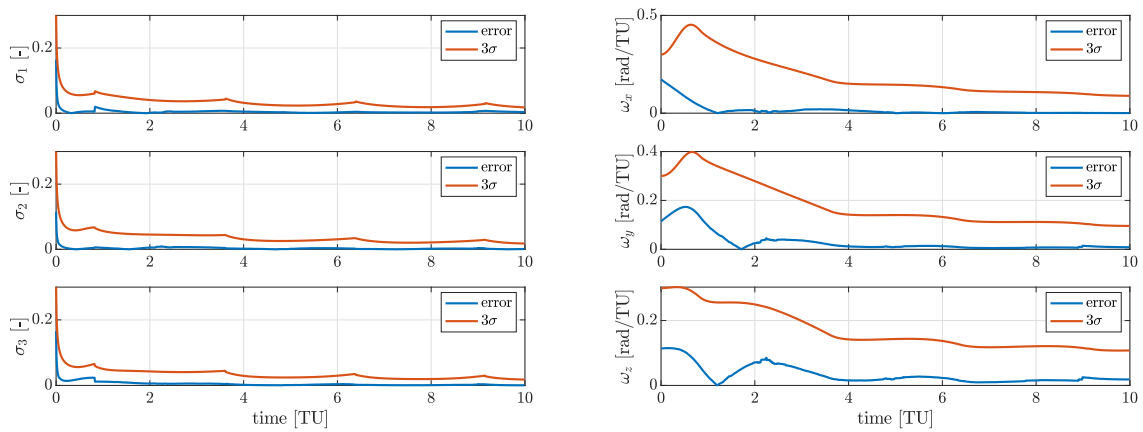


(c) Inertia parameters of the target



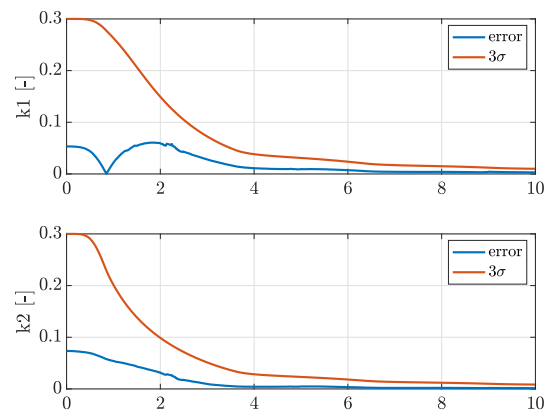
(d) Target's symmetry axis spherical coordinates in \mathcal{I}_Γ frame and $\frac{\Gamma^{(0)}}{A}$ ratio

Figure A.2: Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 20%)



(a) MRPs of the target

(b) angular velocity of the target



(c) Inertia parameters of the target

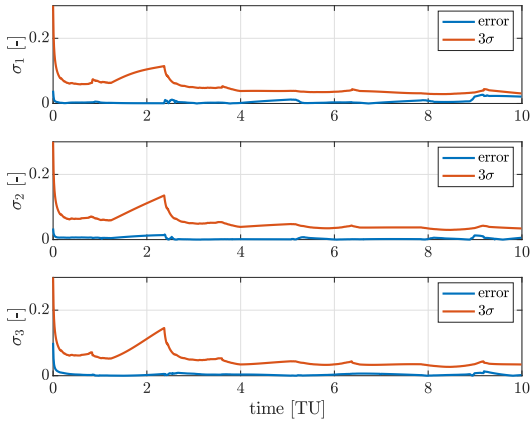
Figure A.3: Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 20%)

B | Appendix B

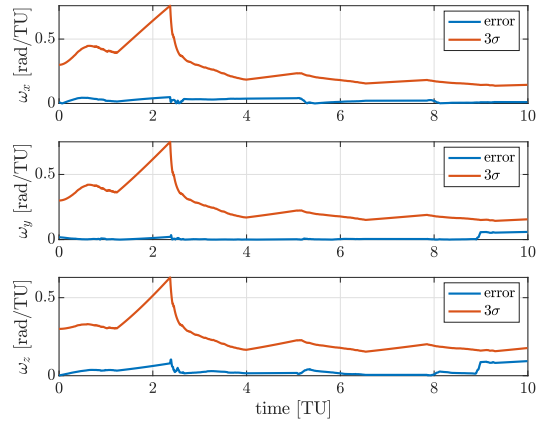
This appendix shows the results obtained using a detection failure rate of 20%, varying the number of features, in the case of unknown rigid rotation matrix.

	Number of time steps without features	Number of switch from EPnP to BCPD
1 feature	516	0
6 features	6	96
14 features	0	193

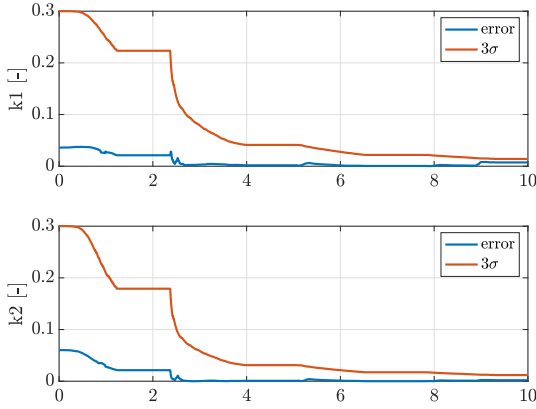
Table B.1: Number of time steps without features and with the switch from EPnP to BCPD, by varying the number of features



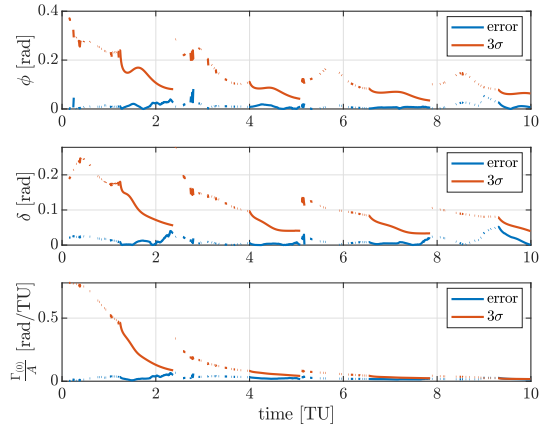
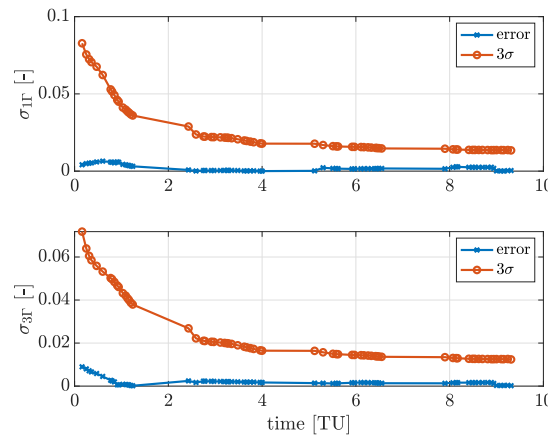
(a) MRPs of the target



(b) angular velocity of the target

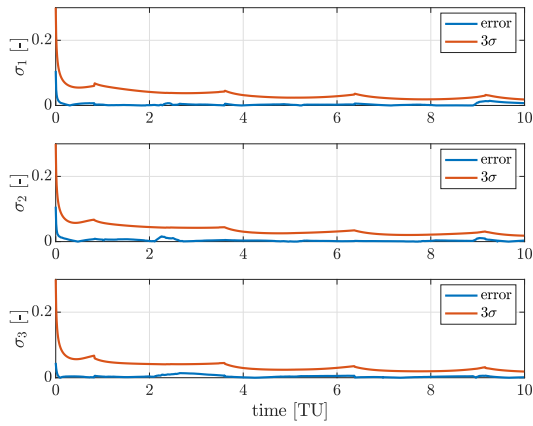


(c) Inertia parameters of the target

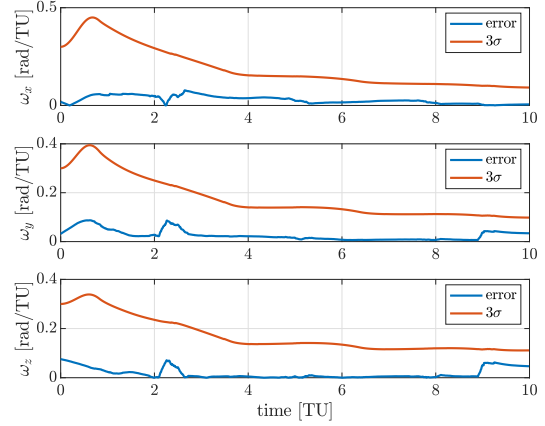
(d) Target's symmetry axis spherical coordinates in \mathcal{I}_T frame and $\frac{\Gamma^{(0)}}{A}$ ratio

(e) MRPs of the rigid rotation matrix

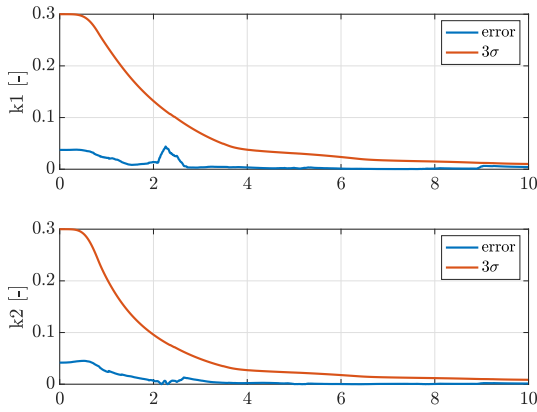
Figure B.1: Convergence of the rotational filter with unknown rigid rotation matrix (1 features and failure detection percentage = 20%)



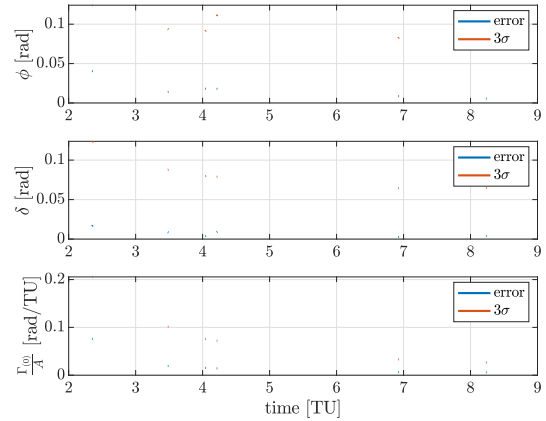
(a) MRPs of the target



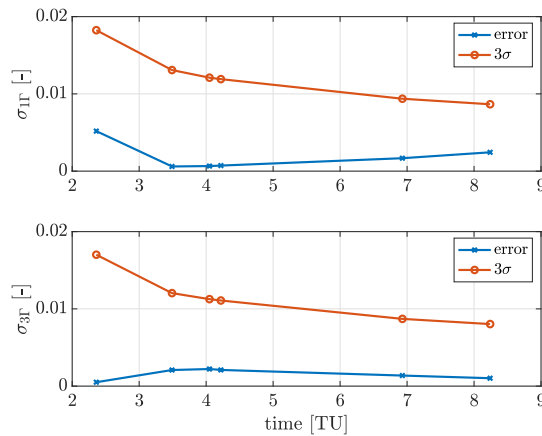
(b) angular velocity of the target



(c) Inertia parameters of the target

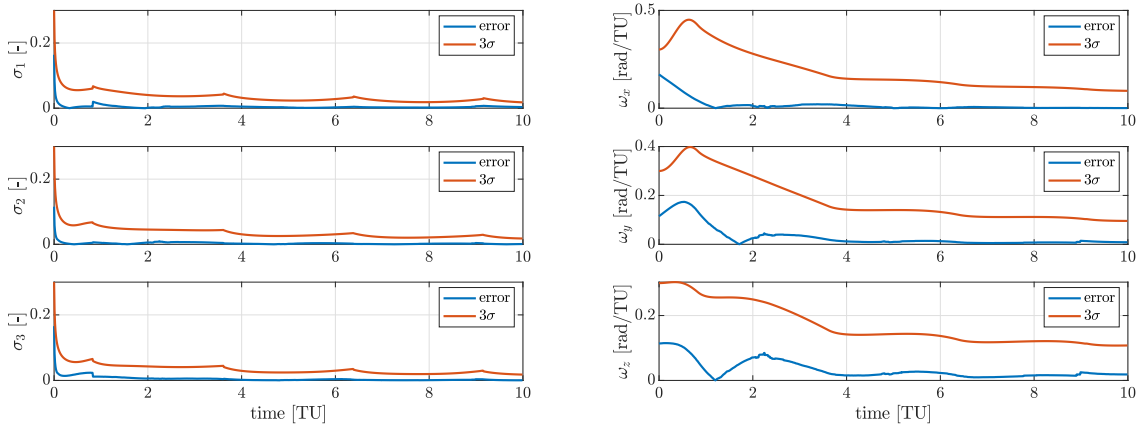


(d) Target's symmetry axis spherical coordinates in \mathcal{I}_Γ frame and $\frac{\Gamma(0)}{A}$ ratio



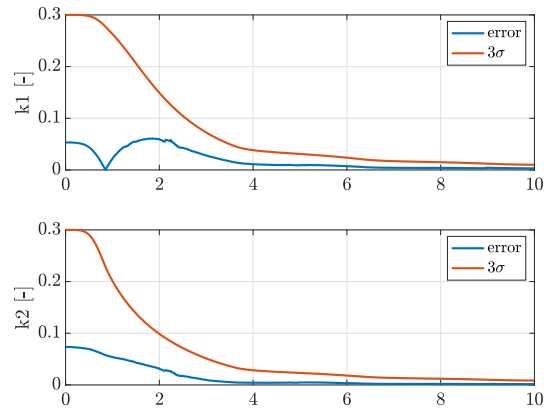
(e) MRPs of the rigid rotation matrix

Figure B.2: Convergence of the rotational filter with unknown rigid rotation matrix (6 features and failure detection percentage = 20%)



(a) MRPs of the target

(b) angular velocity of the target



(c) Inertia parameters of the target

Figure B.3: Convergence of the rotational filter with unknown rigid rotation matrix (14 features and failure detection percentage = 20%)

Bibliography

- [1] EPNP. URL <https://github.com/cvlab-epfl/EPnP.git>.
- [2] Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, 4(1):70–77, 1981.
- [3] D. A. R. P. Agency. Orbital express, . URL <https://www.darpa.mil/about-us/timeline/orbital-express>.
- [4] E. S. Agency. About space debris, . URL https://www.esa.int/Space_Safety/Space_Debris/About_space_debris.
- [5] E. S. Agency. clearspace-1, . URL https://www.esa.int/Space_Safety/ClearSpace-1.
- [6] E. S. Agency. Active debris removal, . URL https://www.esa.int/Space_Safety/Space_Debris/Active_debris_removal.
- [7] E. S. Agency. Esa moves ahead with in-orbit servicing missions, . URL https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/ESA_moves_ahead_with_In-Orbit_Servicing_missions2.
- [8] J. D. Alexander and R. W. Becker. Apollo experience report: Evolution of the rendezvous-maneuver plan for the lunar-landing missions. Technical report, 1973.
- [9] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7–13, 2006. Proceedings, Part I 9*, pages 404–417. Springer, 2006.
- [10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [11] D. BORONI. Sviluppo di un sistema di navigazione e controllo per manovre di prossimità basato su sensore visivo. 2011.
- [12] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross

- correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–102. SPIE, 2001.
- [13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.
- [14] J. A. Christian and S. Cryan. A survey of lidar technology and its use in spacecraft relative navigation. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4641, 2013.
- [15] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.
- [16] J. L. Crassidis and J. L. Junkins. *Optimal estimation of dynamic systems*. CRC press, 2011.
- [17] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29, 2016.
- [18] P. David, D. Dementhon, R. Duraiswami, and H. Samet. Softposit: Simultaneous pose and correspondence determination. *International Journal of Computer Vision*, 59:259–284, 2004.
- [19] N. N. Dawoud, B. B. Samir, and J. Janier. Fast template matching method based optimized sum of absolute difference algorithm for face localization. *International Journal of Computer Applications*, 18(8):0975–8887, 2011.
- [20] M. A. De Luca. Relative navigation at unknown uncooperative symmetric objects. 2022.
- [21] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *International journal of computer vision*, 15:123–141, 1995.
- [22] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.
- [23] D. Eppstein, M. S. Paterson, and F. F. Yao. On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17:263–282, 1997.
- [24] N. Faraco, M. Maestrini, and P. Di Lizia. Instance segmentation for feature recognition on noncooperative resident space objects. *Journal of Spacecraft and Rockets*, 59(6):2160–2174, 2022.

- [25] L. Fei-Fei, L. Yunzhu, and G. Ruohan. Stanford course cs231n: Convolutional neural networks for visual recognition. URL <http://cs231n.stanford.edu/>.
- [26] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [27] D. M. Gavrila and V. Philomin. Real-time object detection using distance transforms. In *Proc. Intelligent Vehicles Conf*, volume 998, 1998.
- [28] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [29] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [30] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching: pose estimation and correspondence. *Pattern recognition*, 31(8):1019–1031, 1998.
- [31] A. Heaton, R. Howard, and R. Pinson. Orbital express avgs validation and calibration for automated rendezvous. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6937, 2008.
- [32] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390. IEEE, 2011.
- [33] O. Hirose. A bayesian formulation of coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 43(7):2269–2286, 2020.
- [34] P. C. Hughes. *Spacecraft attitude dynamics*. Courier Corporation, 2012.
- [35] J. E. Hurtado. Interior parameters, exterior parameters, and a cayley-like transform. *Journal of Guidance, Control, and Dynamics*, 32(2):653–657, 2009.
- [36] G. Jocher. Yolov5 by ultralytics. URL <https://github.com/ultralytics/yolov5>.
- [37] G. Jocher, A. Chaurasia, and J. Qiu. Yolo by ultralytics. URL <https://github.com/ultralytics/ultralytics>.
- [38] C. D. Karlgaard and H. Schaub. Nonsingular attitude filtering using modified Rodrigues parameters. *The Journal of the Astronautical Sciences*, 57(4):777–791, 2009.

- [39] T. Kasai, M. Oda, and T. Suzuki. Results of the ets-7 mission-rendezvous docking and space robotics experiments.
- [40] V. Lepetit, F. Moreno-Noguer, and P. Fua. Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, 81:155–166, 2009.
- [41] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.
- [42] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [43] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.
- [44] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
- [45] M. Maestrini. Satellite inspection of unknown resident space objects. 2022.
- [46] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.
- [47] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.
- [48] R. Opromolla, G. Fasano, G. Rufino, and M. Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93:53–72, 2017.
- [49] R. Padilla, S. L. Netto, and E. A. Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.
- [50] Q. Pan, G. Reitmayr, and T. Drummond. Proforma: Probabilistic feature-based on-line rapid model acquisition. In *BMVC*, volume 2, page 6, 2009.

- [51] S. Persson, P. Bodin, E. Gill, J. Harr, and J. Jørgensen. Prisma-an autonomous formation flying mission. 2006.
- [52] E. Piazza, A. Romanoni, and M. Matteucci. Real-time cpu-based large-scale three-dimensional mesh reconstruction. *IEEE Robotics and Automation Letters*, 3(3):1584–1591, 2018.
- [53] M. Piazza. Deep learning-based monocular relative pose estimation of uncooperative spacecraft. 2020.
- [54] S. Qiao, L.-C. Chen, and A. Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10213–10224, 2021.
- [55] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
- [56] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [58] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [59] T. E. Rumford. Demonstration of autonomous rendezvous technology (dart) project summary. In *Space Systems Technology and Operations*, volume 5088, pages 10–19. SPIE, 2003.
- [60] H. Schaub and J. L. Junkins. *Analytical mechanics of space systems*. Aiaa, 2003.
- [61] H. Schaub and J. L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA, 2nd edition, October 2009. doi: 10.2514/4.867231.
- [62] F. Schnitzer, K. Janschek, and G. Willich. Experimental results for image-based geometrical reconstruction for spacecraft rendezvous navigation with unknown and uncooperative target spacecraft. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5040–5045. IEEE, 2012.
- [63] S. Sharma and S. D’Amico. Neural network-based pose estimation for noncooperative

- spacecraft rendezvous. *IEEE Transactions on Aerospace and Electronic Systems*, 56(6):4638–4658, 2020.
- [64] A. Sonnenburg, M. Tkocz, and K. Janschek. EKF-slam based approach for spacecraft rendezvous navigation with unknown target spacecraft. *IFAC Proceedings Volumes*, 43(15):339–344, 2010.
- [65] J. Terven and D. Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.
- [66] B. E. Tweddle. *Computer vision-based localization and mapping of an unknown, uncooperative and spinning target for spacecraft proximity operations*. PhD thesis, Massachusetts Institute of Technology, 2013.
- [67] J. Ventura. *Autonomous proximity operations for noncooperative space targets*. PhD thesis, Technische Universität München, 2016.
- [68] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.
- [69] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.
- [70] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science: Graz, Austria, June 20–21, 1991 Proceedings*, pages 359–370. Springer, 2005.
- [71] J. R. Wertz and R. Bell. Autonomous rendezvous and docking technologies: status and prospects. *Space Systems Technology and Operations*, 5088:20–30, 2003.
- [72] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
- [73] K. Yoshida, K. Hashizume, D. Nenchev, N. Inaba, and M. Oda. Control of a space manipulator for autonomous target capture-ets-vii flight experiments and analysis. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4376, 2000.
- [74] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. A survey

of modern deep learning based object detection models. *Digital Signal Processing*, 126:103514, 2022.

Acknowledgements

First of all, I wanted to thank Professor Pierluigi Di Lizia for giving me the opportunity to do this work, which I was passionate about and involved in. Together with him, I also want to thank my advisor Michele and my two co-advisors Alessandra and Niccolò for being by my side and providing me with support throughout the work.

Then I wanted to thank my girlfriend Alice who stood by me with love and wisdom throughout my university career, spurring me on to give my best. I also wanted to thank her for all the good times we shared together, from the days we spent together studying to the wonderful trips we took.

I want to thank my whole family, who have always been there to push me forward. My dad and my mum for their valuable advices and my sister for her support, in her unique way. I also thank my grandparents Liana, Mario and Marina for the delicious lunches together and the days in Genoa.

Finally, I would like to express my gratitude to all my friends, with whom I have shared holidays and wonderful evenings, which have been useful to take my mind off my study load.

