

Every Move You Make: Inferring Intent of Non-Cooperative Satellite Rendezvous and Proximity Operations

THESIS MANUSCRIPT

Every Move You Make: Inferring Intent of Non-Cooperative Satellite Rendezvous and Proximity Operations

Thesis Manuscript

by

Calvin P. Grootenboer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on the 6th of October, 2026 at 10:45.

Student Number:	5208599	
Thesis Duration:	February 2, 2026 – October 6, 2026	
Thesis Supervisors:	Dr. S. Gehly,	TU Delft
	Ir. T. C. Goldman,	Royal NLR
	Ir. C. C. M. Pelt,	Royal NLR
Thesis Committee:	Dr. ir. J. Bouwmeester,	TU Delft, Chairperson
	Dr. C. Siemes,	TU Delft, External Examiner

Dedication

Voor Oma Ria, Opa Piet, en Ome Jan, die ons helaas hebben verlaten.

Voor kleine Femke, die gelukkig bij ons is gekomen.

Preface

All good things must come to an end. With this report, I close not only the thesis period but also my entire study time. It has been a long road getting from there to here, but my time is finally here.

Thank you, ma and pa. You know this period hasn't always been the easiest for me. Being able to go back home and not have to worry about life's chores has been a welcome break from it all, as has the mental support whenever I needed it.

I would like to thank my supervisors, Steve, Thomas, and Carlo, for their guidance. Their knowledge, insight, and the occasional epiphany they helped bring about would not have been possible without them. Thanks also to everyone else at NLR, for the tips on my thesis, the sharp questions during my presentations, and of course the table tennis and GeoGuessr games.

Thank you to everyone at Ecosmic. Not only was my internship there a great experience, but your continued interest and advice throughout my thesis were a real help.

Thank you, friends, for the struggles of studying on our theses together. The times spent last summer in EWI when the campus felt deserted, and the much-needed distractions outside of studying hours.

To whatever comes next — I shall make it a good one.

*Calvin P. Grootenboer
Delft, September 2026*

AI-statement

Throughout the course of this project, a large language model was used in a supporting role, both in the preparation of this document and in the implementation of the models and data generation it describes. Its use was mainly limited to rephrasing sentences and correcting grammar, assisting in navigating the literature, refactoring and optimising code, and generating code to plot figures. The main ideas, analyses, and conclusions presented in this thesis are entirely my own; all AI-assisted suggestions were critically reviewed and edited, and I take full responsibility for the final content of this document.

Abstract

Rendezvous and proximity operations (RPOs) are a standard class of space operation, increasingly characterised by dual-use and adversarial potential. When conducted non-cooperatively, the intent behind an RPO is unknown, complicating threat assessment for operators. This thesis presents a methodology for inferring RPO intent from observable trajectory data. Five intent classes are defined from representative mission objectives, and synthetic trajectories are generated for each via multi-objective optimisation, spanning both low Earth orbit and geosynchronous orbit regimes. An XGBoost classifier is trained on manoeuvre features extracted from this synthetic dataset to distinguish between intent classes, and is benchmarked against Random Forest and SVM alternatives. On synthetic data, the classifier reliably discriminates between intent classes, with accuracy improving as more manoeuvres, and thus more trajectory context, become available. Validation against real-world tracking data shows the same trend but at lower absolute accuracy, revealing a sim-to-real gap that is traceable to training data. This gap indicates that the synthetic training data does not fully capture the manoeuvre behaviour of real-world, potentially adversarial chasers, and that closing it will require closer alignment between simulated and observed RPO dynamics.

Contents

Dedication	i
Preface	ii
Summary	iv
Nomenclature	vii
1 Introduction	1
2 Literature Review	3
2.1 RPO Literature	3
2.2 Intent Inference Models	3
2.3 RPO Intent Inference	6
2.4 Literature Gaps	12
3 Research Questions and Thesis Scope	14
3.1 Research Question Formulation	14
4 Characterising Non-Cooperative RPOs	16
4.1 A Non-Extensive Overview of Non-Cooperative RPOs	16
4.2 A Closer Look: Analysis of Real-World RPOs	22
4.3 Intent Taxonomy and Justification	36
5 RPO Dataset Generation	39
5.1 Data Generation Background and Overview	39
5.2 Satellite Population and Initialisation	40
5.3 Mission Phase Optimisation and Terminal Conditions	41
5.4 Orbital Adjustment Phase	51
5.5 Real-World Data Acquisition	58
5.6 Comparison Synthetic and Real Datasets	62
6 Intent Classification Model	67
6.1 Input Data and Feature Engineering	67
6.2 Model Architecture	74
6.3 Model Evaluation	76
7 Model Robustness and Operational Realism	85
7.1 Degradation Background and Approach	85
7.2 Assessment on Degraded Synthetic Data	87
7.3 Real World Cases Assessment	90
7.4 Simulation-to-Real Failure Mode Shift	95
8 Discussion	100
8.1 Answer to Research Questions	100
8.2 Implications	102
8.3 Limitations	103
9 Conclusion	104
9.1 Recommendations for Future Work	105
References	106

A	Additional Information	114
A.1	Optimisers	114
A.2	Lead-up Optimisation Results	123
A.3	Noise Correlation	124
B	Verification	126
B.1	Simulation Framework Unit Tests	126
B.2	Optimisation Objective Unit Tests	128
B.3	Mission Optimisation Unit Tests	132
C	Thesis Planning	140

Nomenclature

Abbreviations

Abbreviation	Definition
ASAT	Anti-Satellite
CNN	Convolutional Neural Network
CW	Clohessey–Wiltshire
DE	Differential Evolution
DST	Dempster-Shafer Theory
FN	False Negative
SGA	Simple Genetic Algorithm
GAST	Greenwich Apparent Sidereal Time
GEO	Geostationary Orbit
GNN	Graph Neural Network
GNSS	Global Navigation Satellite System
GRA	Grey Relational Analysis
GRU	Gated Recurrent Unit
ISAR	Inverse Synthetic Aperture Radar
ISS	International Space Station
LEO	Low Earth Orbit
LSTM	Long Short-Term Memory
MEO	Medium Earth Orbit
MEV	Mission Extension Vehicle
ML	Machine Learning
MLP	Multilayer Perceptron
MPC	Model Predictive Control
MQ	Main Research Question
NORAD	North American Aerospace Defense Command
NSGA-II	Non-dominated Sorting Genetic Algorithm II
OOD	Out-of-Distribution
PSO	Particle Swarm Optimisation
RAAN	Right Ascension of the Ascending Node
RGB	Red-Green-Blue
RK	Runge-Kutta
RNN	Recurrent Neural Network
RPO	Rendezvous and Proximity Operations
RQ	Sub Research Question
RSO	Resident Space Object
SA	Simulated Annealing
SDP	Simplified Deep Space Perturbations (model)
SGP	Simplified General Perturbations (model)
SMA	Semi-Major Axis
SSA	Space Situational Awareness
SVM	Support Vector Machine
TLE	Two-Line Element (set)
TP	True Positive

Symbols

Symbol	Definition	Unit
a	Semi-Major Axis	[km]
D	Dish Diameter	[m]
D	Normalised Distance	[-]
d	Distance	[]
e	Eccentricity	[-]
f	Objective Function Evaluation	[-]
h	Angular Momentum	[]
H	Hypervolume	[m ^N]
i	Inclination	[deg, rad]
J	Objective	[-]
$\mathcal{K}$	Keplerian Transform	[-]
M	Mean Anomaly	[deg, rad]
n	Count	[-]
N	Count	[]
P	Penalty	[]
$\mathbf{r}$	Position Vector	[km]
R	Remaining Time	[s]
R	Excess Ratio	[-]
R	Radial	[km]
S	Along-track / In-track	[km]
T	Time	[s]
t	Epoch	[s]
t_f	Final Time	[s]
u	Mean Longitude / In-plane Angle	[deg, rad]
$\dot{u}$	Drift Rate	[deg/s, rad/s]
$\mathbf{v}$	Velocity Vector	[km/s]
V_{rm}	Reachability Metric	[m/s]
ΔV	Delta V	[m/s]
ΔV_λ	Lambert ΔV	[m/s]
w	Weight	[-]
W	Cross-track	[km]
x	Decision Vector	[-]
δ	Difference (threshold)	[km]
δ	Deviation	[m]
λ	Instantaneous Longitude	[deg, rad]
λ	Elevation	[deg, rad]
$\lambda(\cdot)$	Lebesgue Measure	[-]
λ	Wavelength	[m]
μ	Gravitational Parameter	[km ³ /s ²]
ν	True Anomaly	[deg, rad]
ω	Argument of Periapsis	[deg, rad]
Ω	Right Ascension of the Ascending Node	[deg, rad]
ϕ	Azimuth	[deg, rad]
σ	Standard deviation	[m]
Σ	Covariance Matrix	[-]
θ_{3dB}	3dB Half-Angle	[deg, rad]
θ_{eaves}	Eavesdropping Angle	[deg, rad]
θ_{phase}	Solar Phase Angle	[deg, rad]

"I am certain of nothing but the holiness of the heart's affections, and the truth of imagination."

John Keats, *Life, Letters and Literary Remains* - 1817

1

Introduction

Rendezvous and proximity operations (RPOs) are a well-established and routine class of orbital activities, encompassing a broad range of mission types including resupply to space stations, commercial life-extension services, and coordinated formation flying. However, a new subclass of RPOs is developing: non-cooperative RPOs, in which one of the actors involved does not actively participate in the encounter, or is unaware of and does not consent to the other actor's approach.

These non-cooperative RPOs sit within the broader context of space as a domain of both economic and military importance. Space provides key enabling technologies, such as global communication, positioning, and Earth observation data, and the downstream products of these applications form a significant part of the global economy: valued at 630.0 Billion US Dollars in 2023, and projected to reach 1,790 Billion US Dollars by 2035 [1]. Space is equally a military domain. It provides critical communication services for armed forces, supports intelligence gathering and target acquisition, and, through global navigation satellite systems (GNSS), provides the positioning and timing on which guided munitions and broader military navigation depend.

This dual importance makes space infrastructure an attractive target for hostile action, a risk compounded by its inherent vulnerability. Satellites face natural hazards intrinsic to the space environment, such as solar radiation, vacuum conditions, and orbital debris, as well as deliberate harmful actions from adversarial actors. The latter can take several forms, including kinetic physical attacks such as anti-satellite (ASAT) missiles, non-kinetic physical attacks such as high-powered microwave weapons, electronic warfare including spoofing and jamming, and cyber attacks such as injection attacks [2].

Non-cooperative RPOs are one such tool for hostile action. The action ultimately undertaken during an RPO can itself be a form of attack: for instance a cyberattack conducted at close range, or a non-kinetic disabling action such as dazzling. However, the RPOs documented to date are, for the most part, intelligence-gathering in nature: either supporting space domain awareness, by observing what satellites are present and characterising their capabilities, or supporting signal intelligence, by listening in on communications sent to a target satellite [3, 4].

One of the prominent features of these non-cooperative RPOs is the ambiguity around them. The mission of the satellite conducting the RPO (the chaser) is typically unknown to the operator of the target satellite. Although the operator of the chaser satellite may publish a mission statement, the observed behaviour of the spacecraft often does not align with the expected patterns of life associated with that mission type. In other cases, the mission description itself is vague, with broad terms like "technology demonstrator" being used, without specifying what the satellite is undertaking [4].

This ambiguity often leads to suspicion, which is further amplified by the nature of typical target satellites.

Some of the most common targets are geostationary (GEO) communications satellites and high-value intelligence or reconnaissance satellites [4, 3]. These systems are either military in nature or possess dual-use capabilities, which makes operators increasingly cautious when observing any unusual proximity operations.

Effort is thus put into characterising what is happening during these non-cooperative RPOs. Whereas the detection of an RPO event becomes feasible once the chaser spacecraft has begun manoeuvring, the timely characterisation of the intent underlying that manoeuvre remains a substantially more difficult problem. With the inventory of documented non-cooperative RPOs continuing to grow [3], there is an imperative to classify the intent of such encounters as early as possible. Early intent inference would enable the operators of targeted spacecraft to assess the nature and severity of the potential threat and, where deemed appropriate, undertake mitigating actions.

Space Situational Awareness (SSA) is the field that has increasingly turned towards resolving this ambiguity. SSA is a broad field, encompassing research areas such as sensor design and utilisation, conjunction analysis, re-entry prediction, and object characterisation; within it, a shift is underway away from simply maintaining catalogues of resident space objects (RSOs) and propagating their trajectories, towards understanding the behaviour, intent, and operational context of objects in orbit. This shift spans the general classification and characterisation of RSOs from radar, optical, spectral, and RF observations [5, 6, 7, 8, 9, 10], the detection and characterisation of manoeuvres [11, 12, 13, 14, 15, 16], and long-term pattern-of-life modelling [17, 18, 19, 20]. A subset of this work bears directly on non-cooperative RPOs, such as the estimation of manoeuvre magnitude from close-proximity observations [15] and the prediction of chaser manoeuvre timing [12].

This thesis is positioned within the field of SSA, addressing the need to understand not only that a non-cooperative RPO is occurring, but also what it plans to do: the intent behind it.

The purpose of this thesis is to investigate inferring the intent of non-cooperative RPOs at an early stage. To achieve this, the manuscript is structured as follows: Chapter 2 documents the literature on the topic of RPOs and of intent inference. Then, in Chapter 3, the research questions are formulated. Chapter 4 documents RPOs and non-cooperative RPOs alike, to define what classes there are and how they are executed. Chapter 5 elaborates on both the synthetic and real-world data considered for intent inference training and assessment. Chapter 6 introduces the intent inference data structure and models used to perform inference, as well as a preliminary assessment. Chapter 7 covers the process of matching real-world data conditions, model performance under these conditions, and the assessment on real-world data. Finally, Chapter 8 discusses the most important results, and the report is concluded in Chapter 9.

*"Basic research is what I am doing
when I don't know what I am doing."*

Wernher von Braun, *A Random
Walk in Science* - 1973

2

Literature Review

This chapter documents relevant literature to the field of non-cooperative RPO intent inference. First, literature related to RPOs in general is document in Section 2.1. This is followed by general intent inference literature in Section 2.2. This is followed by literature on RPO intent inference in Section 2.3 and the literature gaps on this topic in Section 2.4.

2.1. RPO Literature

Research into RPOs spans a wide field of sub-topics, not all directly relevant to intent inference for non-cooperative RPOs. Cis-lunar rendezvous, for example, is an established research area in its own right [21, 22], but beyond offering a potential future orbital domain in which non-cooperative RPOs might occur, it is not relevant.

More directly relevant is machine-learning-based pose estimation [23, 24, 25, 26, 27], which underpins a chaser's ability to determine a target's orientation from its own perspective – a prerequisite for RPOs requiring engagement with a specific part of the target, such as grappling. Research into orbital target-attacker-defender problems [28, 29, 30, 31, 32] is similarly relevant to both sides of a non-cooperative encounter, whether pursuing a fleeing target or evading a pursuing chaser.

Where the target is not fleeing, RPO trajectory optimisation techniques become directly applicable to the non-cooperative case. Machine learning is increasingly explored for this purpose [33, 34, 35], alongside convex optimisation [36, 37] and optimal control formulations such as model predictive control [38, 39], with direct methods further shown to be more robust to initial conditions than indirect solving [40]. These methods, however, can be computationally expensive [38] and often depend on a good initial guess to converge [41].

Metaheuristics offer a computationally cheaper alternative [42], including Simple Genetic Algorithms (SGAs) [43, 44], Particle Swarm Optimisation (PSO) [45, 44], and Differential Evolution (DE) [44]. These are further strengthened through hybridisation with one another or with numerical methods [41].

2.2. Intent Inference Models

RPO intent inference leans on the wider field of intent inference for its approaches. This field can be traced through three main periods, its traditional methods (Subsection 2.2.1), early machine learning (Subsection 2.2.2), and deep learning (Subsection 2.2.3).

2.2.1. Rule-Based and Statistical Approaches

Traditional methods include expert systems, which employ rule-based logic to infer underlying plans from observed facts [46, 47, 48, 49, 50]. The nature of expert systems means that their performance depends heavily on the implementation, which can require extensive manual tuning. Furthermore, they cannot take in information which is not part of the implementation of the expert system, and thus unprecedented or unconventional actions cannot be taken into account at all [50].

Template-based approaches [51] classify behaviour by matching observed actions against a library of predefined patterns. Their application is best suited to relatively simple, well-characterised contexts, and they tend to struggle once more intricate dynamics are introduced [50].

Statistical approaches treat observations as evidence for candidate plans, with different frameworks such as Bayesian conditional probability [52], Dempster-Shafer theory (DST) [53, 54], and Grey Relational Analysis (GRA) [55].

The strength of Bayesian networks lies in their robust causal reasoning capabilities. However, the construction and learning process of the network structures requires expert knowledge and extensive data. For complex plans, the scale and complexity of the network grows significantly, which can hinder development [52, 56, 50].

DST is similar to a Bayesian framework, but instead of observations being a conditional for a plan, observations are used as evidence for plans [50, 49]. Furthermore, the framework supports uncertain propositions and the combination of multiple (conflicting) sources of evidence easily [53, 54]. However, DST can produce inconsistent results under conflicting evidence and struggles to correlate evidence across time [50].

GRA, although developed as the framework for a certain type of control system [55], is also used for intent inference [57, 58, 50]. It works by trying to reconstruct underlying control inputs, and correlate the control inputs to intent. While GRA allows intent to be inferred from minimal data, it can overlook intricacies that distinguish between similar intents and needs a relation between intent and control input [50].

2.2.2. Classical Machine Learning

Early machine learning methods gained prominence over traditional methods for two main reasons; traditional methods scaled poorly for complex tasks requiring extensive manual configuration of the architectures used, and the ability to deal with noisy or otherwise sub-optimal data was lacking. Machine learning models proved to be more suited for these challenges, and could deal with more complex tasks, and thus a new avenue was opened [50]. Several types of early machine learning models exist, like Random Forests, Support Vector Machines (SVMs), and Clustering Methods. These methods fall under classification algorithms, which can be applied to intent characterisation but have other uses as well.

Random Forests are an ensemble of decision trees; Breiman [59] first developed them to overcome the limitations of a single decision tree, which can be sensitive to small changes in the input data. By taking the average result of multiple trees, all randomly and automatically constructed to consider different features of the input data, a more robust architecture is developed.

SVMs operate by taking data of pre-defined classes or categories, mapping that data to another dimensional space in which the features are more easily distinguished, and then finding a plane separating the classes [60].

XGBoost is a gradient boosting algorithm, and uses an ensemble of decision trees built sequentially, with each new tree trained to correct the errors of those preceding it, rather than being constructed independently and averaged as in Random Forests [61]. This sequential error-correction, combined with regularisation to control model complexity, allows XGBoost to achieve high predictive accuracy while remaining computationally efficient.

Clustering methods, like K-Means [62], Hierarchical [63], and DBSCAN [64], differ from Random Forests and SVMs in that they do not require pre-defined class labels. Instead, they rely on generating groupings themselves. The methods differ in how these groupings are made: K-means establishes a 'K'-

number of groupings based upon iterative calculation of the centroid, while DBSCAN identifies groupings based upon the density of data points, allowing for dynamic allocation of the number of groupings. Their usefulness for intent characterisation is limited, however, as no predefined intent labels can be assigned during training.

These methods can perform well on limited and structured data but struggle with larger inputs [50], but conversely can outperform more complex deep learning methods when the input is limited [65].

2.2.3. Deep Learning

Deep learning differs from classical machine learning in that the models are more complex, though no clear boundary exists [50].

MLPs consists of multiple layers of nodes, where each node from a layer is connected to all nodes from the next one. At each node, a weighted addition is performed based upon the connections, the weights for which are learned through training using backpropagation [66]. The issue with these types of neural networks is that they struggle with input data for which the order and duration is important, like time series [50]. Due to the limitations of MLPs, more advanced architectures are employed.

LSTMs and gated recurrent unit networks (GRUs) are recurrent neural nets designed to deal with the ordered nature of time-series. LSTMs utilise a three-gated cell design: input, output, and forget nodes, alongside hidden states [67]. The cells contain the information the network has about the input series up to that point, whilst the hidden state contains the current information for this time step. The forget gate in the cell decides what information from the last cell to forget, the input gate decides what information from the current step is considered, and the output gate decides how these are combined and outputted to the next layer of cells. GRUs combine the hidden states and the cells into one, simplifying the model [68], and it is shown by Chung et al. [69] that the performance of these two methods is practically the same, for reduced computational effort on the GRU side.

Convolutional networks add a pattern-recognition capability by encoding the input as an image-like array, learning filters that capture local structure which are then combined into higher-level features across successive layers [70].

Transformers dispense with recurrence and convolution entirely, instead using self-attention to weigh every element of the input sequence against every other element directly, rather than passing information forward step-by-step or through local filters. This allows global, long-range dependencies to be captured within a single layer, with the relationships between elements learned from the data rather than fixed by the architecture [71].

Improvements over base recurrent architectures are common in recent years. Bi-directionality extends the standard RNN by processing the input sequence in both forward and reverse temporal directions simultaneously, allowing the model to incorporate future context [72]. SA augmentation allows the recurrent model to directly attend to any prior state rather than relying solely on the compressed hidden vector passed between time-steps [72], alleviating the vanishing gradient problem and enabling the model to capture long-range dependencies. The addition of a convolutional step, typically applied before the recurrent layers, enables the model to extract local temporal features and patterns across a sliding window of time steps, providing richer input representations that the recurrent layers then process sequentially [73]. A transformer step, when incorporated alongside recurrent layers rather than replacing them, typically appears as a SA sublayer applied to the output sequence, combining the temporal locality and sequential inductive bias of the RNN with the global context aggregation of attention [74]. These enhancements are often combined, with architectures such as a convolutional, bi-directional LSTM with SA (CNN-BiLSTM-SA) [75]. It remains unclear whether these enhanced RNNs are fairly compared against Transformers, however, as equivalent architectural improvements remain comparatively underexplored for Transformer-based classification [76].

Whilst Transformers are able to assess the entire input at the same time, which has advantages over RNNs' sequential nature, it also limits the performance if certain temporal structures are present. RNNs need fewer parameters for capturing local temporal patterns, and have an inductive bias preferring recent information [77]. Transformers capture global patterns easily, but need to learn to capture the temporal aspect [78]. If recency matters, and if not enough training data is available for the transformer

to learn the temporal aspect of the data, RNNs will outperform Transformers.

2.3. RPO Intent Inference

The research on intent classification of RPOs is recent and ongoing, as can be observed in the overview of papers related to the topic in Table 2.1. The papers can broadly be defined by a few aspects: their definition of intent, how they classify said intent, and the data used for each model. The focus of papers differs as well, with some proposing entirely new approaches to intent inference, whilst others build on previous work, improving classification results through refined models.

Table 2.1: Overview of non-cooperative RPO intent recognition literature

Ref.	Paper	Year	Approach	Intent Definition	Orbital Domain
[79]	Automatic Intent Assignment through Estimation of Proximity Maneuvers	2026	DST	Mission Class	Agnostic
[80]	Intention Recognition of Space Non-cooperative Targets Using Large Language Models	2026	Machine Learning	Mission Class	GEO
[81]	Intention recognition for space non-cooperative targets using Kolmogorov-Arnold transformer	2026	Machine Learning	Motion Class	GEO
[82]	Intention inference for spacecraft proximity maneuver using cascaded neural network	2026	Machine Learning	Motion Class	GEO
[83]	Evidential Intent Assignment using Belief States in the Hill Frame	2025	DST	Mission Class	Agnostic
[31]	Detecting Satellite Maneuver Intent Using Game Theory	2025	Game Theory	Threat Index	Agnostic
[84]	Inferring Spacecraft Maneuver Intention via Inverse Optimal Control	2025	Inverse Optimal Control	Mission Class	Agnostic
[85]	Intention inference for space targets using deep convolutional neural network	2025	Machine Learning	Motion Class	GEO
[86]	Orbital motion intention recognition for space non-cooperative targets based on incomplete time series data	2025	Machine Learning	Motion Class	MEO, GEO
[73]	Orbital Behavior Intention Recognition for Space Non-Cooperative Targets Under Multiple Constraints	2025	Machine Learning	Motion Class	GEO
[75]	An intelligent hierarchical recognition method for long-term orbital maneuvering intention of non-cooperative satellites	2025	Machine Learning	Motion Class	MEO, GEO
[87]	Intention recognition method for spatial non-cooperative target based on improved random forest	2025	Machine Learning	Motion Class	GEO
[74]	BiGAT: A Model for Recognizing Motion Intentions of Space Non-cooperative Targets	2025	Machine Learning	Motion Class	GEO
[88]	Developments for the Characterization of Spacecraft Proximity Operations for Improved Space Situational Awareness	2024	Machine Learning	Mission Class	Agnostic
[89]	Spatial Non-cooperative Target Behavior Intent Recognition Based on Data Generation and Deep Neural Networks	2024	Machine Learning	Motion Class	LEO
[90]	Research on Intent Recognition Method for Non-Cooperative Satellite Based on LSTM Network	2023	Machine Learning	Motion Class	GEO
[91]	Deep neural network for non-cooperative space target intention recognition	2023	Machine Learning	Motion Class	GEO
[72]	An intention inference method for the space non-cooperative target based on BiGRU-Self Attention	2023	Machine Learning	Motion Class	GEO
[92]	Predicted Intent Inferred from Real-time Rendezvous and Proximity Behavior	2022	Machine Learning	Mission Class	GEO
[93]	Threat Analysis Methods of Actively Approaching Spacecraft	2022	Game Theory	Threat Index	GEO

2.3.1. Definitions of RPO Intent

The definition of intent used in the papers from Table 2.1 differs from paper to paper. These definitions can be categorised into three groups: Threat Index, Motion Class, and Mission Class.

The simplest of these is the Threat Index, utilised by He et al. [31] and Wu and Zhang [93]. Rather than classifying a specific behaviour or task, these papers reduce intent to a single scalar parameter. This parameter indicates the likelihood of an RPO occurring, based upon previously observed behaviour of a satellite relative to a target. It offers no further insight, such as when the RPO would occur or what the chaser satellite would do once it reaches the target. *Intent* is thus defined simply as intending to perform an RPO.

The most prevalent definition in the literature is that of Motion Classes, used by [81, 82, 85, 86, 73, 75, 87, 74, 90, 91, 72]. Under this definition, intent is characterised by the specific trajectory a chaser satellite follows relative to the target. Typical motion classes include *Circular Fly-around*, *Spiral Fly-around*, *Drip-style Fly-around*, *Approach and Interception*, *Distance-based Evasion*, and *Follow and Company* [87], though the exact set varies considerably among authors. The number of classes defined ranges from a handful up to 38 distinct categories [86]. Some authors further delineate their classes by incorporating additional parameters, such as the lighting conditions used by Yu et al. [81]. For these papers, intent is thus defined as the relative motion and is not linked directly to an action or purpose that motion could indicate.

The Mission Class definition of intent instead focuses on inferring the operational objective of a chaser spacecraft. The considered intents in this category differ per paper, with Balo et al. [79] and Balo, Kumar, and Soderlund [83] distinguishing between five categories: *Coverage*, *Docking*, *Signalling*, *Flyby*, and *Rendezvous*, whilst at the more extreme end Jing et al. [80] defines ten categories: *Detection*, *Surveillance*, *Interference*, *Negotiation*, *Observation*, *Assistance*, *Deterrence*, *Support*, *Experiment*, and *Attack*. Here, intent is thus defined as the actual purpose or task underlying the RPO.

Whilst all these definitions of intent have their own use, some are more useful than others. The Threat Index is good at indicating whether a satellite is approaching a target, but insights stop there: there is no indication of why a non-cooperative satellite approaches the target, nor any further characterisation of what the close approach might look like. Motion Classes offer more insight into what movement the satellite is performing, but not why it might be performing it. The usefulness of excessive distinction between slightly different motions (such as Forward Drip-style Fly-around versus Reverse Drip-style Fly-around in [86]), however, is not justified in the literature. This focus on minute differences has also been criticised by Chen et al. [73], who argues that such papers miss the point of intent inference because they are “focusing excessively on minor differences in actions while overlooking the possibility that such actions may share the same underlying intent”. Similarly, Balo et al. [79] criticises Motion Classes as a definition of intent because “the definition tends to conflate maneuvers and intents”, which is why their work instead utilises Mission Classes, as these provide the most insight among the three definitions.

Mission Classes themselves, however, are not without their own shortcomings. Depending on the intents included and the paper in question, it is often unclear why a particular behaviour is classified under a given intent. In the works of Balo et al. [79] and Balo, Kumar, and Soderlund [83], the authors manually link motions and approach angles to specific intents but do not justify these links against real-world observed cases. A similar issue arises in Jing et al. [80], where it remains unclear what meaningfully separates closely related categories such as *Detection* and *Surveillance*.

2.3.2. Approaches to RPO Intent Inference

The approaches to intent inference differ per paper and by the type of intent being classified. Once again, three broad categories arise: derived features, probabilistic methods, and machine learning.

Derived features represent the simplest approach to intent inference and are used exclusively by the Threat Index papers. This approach considers the theoretical time of flight and ΔV required for an RPO. He et al. [31] infer intent by tracking changes in the minimum Lambert rendezvous time achievable under a fixed ΔV budget, increasing the threat index after a manoeuvre that decreases time of flight. The threat index thus increases (indicating an RPO is more likely to occur) when the rendezvous time decreases per manoeuvre, and the method can account for multiple manoeuvres. Wu and Zhang [93] instead define the threat index as a weighted combination of the total ΔV and time of flight from two-impulse Lambert solutions over a bounded set of feasible rendezvous times, with the index increasing after any manoeuvre that reduces either quantity. Whilst these derived features are central to the Threat Index approach, they could equally be used as engineered features for other methods.

Motion Classes and Mission Classes are typically inferred using more complex approaches that allow for the incorporation of more data. One such approach is the use of probabilistic methods, under which DST and Bayesian neural nets fall.

DST [53] accumulates evidence from observed chaser motions to infer intent [83, 79]. Motions defined relative to the target spacecraft are treated as evidence supporting one or more Mission Classes, accommodating conflicting evidence and shared explanations across tasks. Balo, Kumar, and Soderlund [83] directly link motion classes to tasks and update belief based on motion persistence, although this formulation struggles to distinguish between tasks with overlapping underlying motions. Balo et al. [79] extend this by introducing support functions that incorporate additional attributes, including duration, orientation, and approach angle, to improve discrimination between closely related tasks. The main limitation of this approach is the need for manual linking of motion classes to intents; the support functions also require careful expert tuning, for which Balo et al. [79] suggest machine learning as a possible avenue of automation. In turn, Bala [88] demonstrate the use of Bayesian neural nets, which represent a combination of probabilistic methods with machine learning.

The most common approach in the literature is machine learning, with a wide array of architectures employed [92, 80, 81, 82, 85, 86, 73, 75, 87, 74, 90, 91, 72]. These generally employ deep learning architectures such as RNN variants and Transformers, trained on labelled datasets to classify intent; an overview of these model architectures is given in Table 2.2. Many of these papers share a significant common heritage: those focusing on motion classes via machine learning trace their origin to Wu and Zhang [93], a Threat Index paper. These papers build upon one another, differentiating themselves mainly by the machine learning architecture used and the number of intents distinguished. Overall, they demonstrate the considerable potential of machine learning models for classifying different intents, with high accuracies achieved on their synthetic datasets. The relative performance of each method within a paper can be observed in Table 2.2, with more advanced architectures generally performing better. What remains unclear from these results, however, is their applicability to real-world operations, and whether the synthetic datasets used are representative of real-world conditions.

A unique approach to intent inference is that of inverse optimal control by Goulet et al. [84]. In this paper, the optimal control policy underlying an approach is inferred from observations of that approach. The optimal control is determined for the following intents: *Rendezvous*, *Collision*, *Inspection*, *Departure*, and *V-Bar Perch*, where *V-Bar Perch* is defined as moving to the same orbit with a phase difference. Each intent has its own objective function, and trajectories are generated through an optimal control policy, with combinations of intents possible. Simulated observations are made of the trajectories, from which the optimal control policy is inferred in the form of weights for each trajectory.

2.3.3. Data used for RPO Intent Inference

The datasets used for both training and evaluating machine learning models are a fundamental aspect of intent inference. Owing to the limited availability of high-quality labelled data (see Chapter 4), all reviewed studies rely on synthetic datasets. However, these datasets differ considerably in their construction and underlying assumptions. Several characteristics distinguish the datasets used in the literature, including the dynamic model employed, the orbital regime considered, the simulated time frame, the source and type of observations, the quality of the generated data, and the relationship between the generated trajectories and the assigned intent labels.

The choice of dynamic model largely determines the realism of the simulated trajectories and the extent to which they represent real-world spacecraft behaviour. The most commonly used model is the Clohessy–Wiltshire (CW) equations, which linearise the relative equations of motion about a circular reference orbit. Their simplicity enables the generation of deterministic trajectories from a set of initial conditions, making them well suited for constructing Motion Classes. Furthermore, the CW framework allows for the formulation of derived dynamics that can be exploited by optimal control approaches [84, 75]. However, the applicability of the CW equations is limited, as the linearisation assumptions are only valid when the reference orbit is circular, the relative distances are small compared with the orbit radius, and the time-scales considered are short [94, 95]. Consequently, datasets generated using this model may not accurately capture long-duration or highly non-linear rendezvous scenarios.

The orbital regime represented in the datasets also varies across the literature. Most studies focus

Table 2.2: Machine learning methods used for intent recognition; column ranking indicates relative performance within the row's paper.

Ref.	Paper	Year	1 st	2 nd	3 rd	4 th	5 th	6 th
Jing et al. [80]	Intention Recognition of Space Non-cooperative Targets Using Large Language Models	2026	LLM					
Yu et al. [81]	Intention recognition for space non-cooperative targets using Kolmogorov-Arnold transformer	2026	KAT	Transformer	GRU	RNN	LSTM	
Zhang et al. [82]	Intention inference for spacecraft proximity maneuver using cascaded neural network	2026	Cascaded TCN-SA	TCN-SA	BiLSTM-SA	BiGRU-SA	TCN	Transformer
Li, Yang, and Luo [85]	Intention inference for space targets using deep convolutional neural network	2025	CNN					
Sun et al. [86]	Orbital motion intention recognition for space non-cooperative targets based on incomplete time series data	2025	BiGRU-SA	BiGRU	BiLSTM	GRU	LSTM	RNN
Chen et al. [73]	Orbital Behavior Intention Recognition for Space Non-Cooperative Targets Under Multiple Constraints	2025	CNN-LSTM-SA	LSTM-SA	CNN-LSTM	CNN	LSTM	SVM
Yuan et al. [75]	An intelligent hierarchical recognition method for long-term orbital maneuvering intention of non-cooperative satellites	2025	CNN-BiLSTM-SA	CNN-BiLSTM	BiLSTM-SA	BiLSTM	LSTM	
Zhao [87]	Intention recognition method for spatial non-cooperative target based on improved random forest	2025	Improved RF	CNN	GRU	LSTM	XGBoost	RF
Sun, Zhao, and Dang [74]	BiGAT: A Model for Recognizing Motion Intentions of Space Non-cooperative Targets	2025	BiGAT	Transformer	BiGRU-SA			
Yu et al. [89]	Spatial Non-cooperative Target Behavior Intent Recognition Based on Data Generation and Deep Neural Networks	2024	SAE-GRU	LSTM	GRU-FCN	SAE	MLP	
Bala [88]	Developments for the Characterization of Spacecraft Proximity Operations for Improved Space Situational Awareness	2024	BNN					
Zhang et al. [72]	An intention inference method for the space non-cooperative target based on BiGRU-Self Attention	2023	BiGRU-SA	BiGRU-Att	BiLSTM-Att	BiGRU	GRU	MLP
Sun and Dang [91]	Deep neural network for non-cooperative space target intention recognition	2023	MLP					
Yihan et al. [90]	Research on Intent Recognition Method for Non-Cooperative Satellite Based on LSTM Network	2023	LSTM					
Kelecyc et al. [92]	Predicted Intent Inferred from Real-time Rendezvous and Proximity Behavior	2022	Transformer					

exclusively on the GEO environment, while a smaller number additionally consider medium Earth orbit (MEO), and only a single study investigates intent inference in low Earth orbit (LEO), despite both GEO and LEO being operationally important environments for non-cooperative rendezvous and proximity operations. The reasons for this imbalance are not explicitly discussed in the literature.

The temporal extent of the simulated data is closely related to both the adopted dynamic model and the definition of intent. Because the validity of the CW equations is inherently limited in time, studies relying on this model typically simulate trajectories only over relatively short durations, typically hours (for example, four hours in [81]). As a result, approaches based on Motion Classes are generally limited to comparatively short prediction horizons. In contrast, the Threat Index papers evaluate intent over sequences of manoeuvres and therefore employ considerably longer simulation windows, with lead times extending from days to months (for example, cases of two days and three months in [31], with mixed results). The duration of the simulated trajectories directly determines how far in advance of the closest approach intent can be inferred.

Another important distinction lies in the simulated observation source. Some studies attempt to model the sensing process itself by generating measurements that correspond to realistic observation platforms. These platforms may either be ground-based sensors or sensors located on the target space-

craft. In the former case, both the chaser and target spacecraft are observed from the ground, with measurements corresponding to raw observations. In the latter case, the target spacecraft performs the observations, which naturally limits the available lead time because observations are typically assumed to become available only once the chaser is within a specified range. The resulting observables differ between studies and may include azimuth and elevation angles, range, relative velocity, or complete Cartesian relative position and velocity states. Other approaches remain observation-agnostic and instead assume that some form of processed state information is already available.

The quality of the simulated observations also differs substantially between studies. Some studies train and evaluate their models using idealised datasets without measurement noise or observation gaps, whereas others explicitly incorporate sensor noise, missing observations, or incomplete measurement sequences. Since operational space surveillance data are inherently imperfect, accounting for these effects is essential when assessing the practical applicability of intent inference methods.

Finally, the relationship between the generated trajectories and the associated intent labels differs depending on the type of intent being inferred. For Motion Classes, this relationship is relatively straightforward, as the trajectory shape is directly determined by the initial conditions. Variations of these initial conditions can therefore be used to generate diverse examples belonging to the same motion class.

For Mission Classes, establishing this relationship is considerably more challenging. Approaches that infer mission intent through intermediate Motion Classes must define how combinations of motion patterns correspond to higher-level mission objectives, although the literature generally provides limited justification for these mappings. Studies that define Mission Classes independently of Motion Classes instead employ one of two approaches: objective-function-based trajectory generation, or matching to observed real-world behaviour. The latter is done by reproducing observed spacecraft behaviour while simulating the corresponding measurements [92]. A key limitation of this method is the very limited number of representative examples available for each intent class, which is likely to increase the risk of overfitting. In contrast, other approaches generate trajectories by optimising predefined objective functions [84]. Although the resulting trajectories are physically meaningful, the interpretation of the underlying objectives in terms of operational intent is not always straightforward. For example, combining a distance-minimisation objective, which promotes close approach, with a circling objective produces a trajectory whose optimisation objectives can be identified by an SSA operator, yet the operational meaning of this combination remains ambiguous.

2.3.4. Unique Contributions of Machine Learning RPO Intent Literature

Yuan et al. [75] employ Motion Classes, similar to several related studies, but extend this concept by introducing “long-term intent”, which corresponds to a higher-level categorisation analogous to Mission Classes. Rather than predicting Mission Classes directly, the long-term intent is inferred from combinations of Motion Classes. This hierarchical relationship is reflected in their machine learning architecture, where one network is trained to recognise Motion Classes, while a second network aggregates these predictions to infer the long-term intent.

To enable this hierarchical classification, the trajectory data must first be divided into pre- and post-manoeuvre segments so that the Motion Class following each manoeuvre can be identified. Although the authors describe the resulting prediction as “long-term” intent, the prediction accuracy decreases substantially with increasing prediction distance. The Motion Class classifier achieves less than 60% accuracy at a distance of 16 km, while the long-term intent classifier falls below 60% accuracy at approximately 14 km.

Sun et al. [86] investigate the performance of their machine learning model under conditions of incomplete observations. The authors argue that when both position and velocity measurements are available, the future orbit of a target can be determined analytically, eliminating the need for a machine learning-based approach. Consequently, they focus on scenarios in which only partial information is available, considering datasets containing exclusively position, angular, or range measurements. The study further evaluates the influence of the observation interval by considering three sampling strategies: equal time intervals, random time intervals, and marked time intervals. In the latter case, observations are collected at random intervals, but the elapsed time between consecutive measurements is included as an additional input feature.

Position- and angle-based observations achieve near-perfect classification accuracy when sampled at equal time intervals, whereas range-only observations perform poorly, never exceeding 17% classification accuracy. Overall, equal time interval sampling yields the highest performance, while random sampling significantly reduces classification accuracy. Including the elapsed time information largely compensates for this degradation, producing results that approach those obtained with uniformly sampled observations.

Furthermore, the authors introduce two retraining strategies, referred to as two-body training and noise training. In both cases, a model initially trained on noise-free trajectories propagated using the CW equations is fine-tuned for a limited number of additional epochs using a modified dataset. For two-body training, the fine-tuning dataset is generated using two-body dynamics, whereas noise training introduces measurement noise into the observations.

The results demonstrate that the proposed retraining strategies allow the models to adapt rapidly to the modified data distributions, requiring only 2–5 additional training epochs compared with nearly 100 epochs for the initial training. Following two-body retraining, the classification accuracy improves from the low-80% range to approximately 95% on a two-body testing dataset. Similarly, noise retraining increases the classification accuracy from roughly 35% to the low-70% range for a noisy testing dataset.

Yu et al. [89] generate synthetic trajectory data differently from most prior work. Their dataset consists of orbital trajectories representing three intent classes: *orbit maintenance*, *rendezvous*, and *inspection*¹. Rendezvous trajectories are generated using Lambert transfers to approach the target, inspection trajectories use relative orbital elements to establish a circumnavigation orbit around the target before transferring onto it, and orbit-maintenance trajectories are generated by temporarily offsetting the chaser's own orbit (through changes in altitude and/or inclination) before returning to the nominal orbit.

The trajectories are generated by randomly sampling physically plausible initial conditions. For the rendezvous and inspection classes, both the target and chaser orbital elements are sampled from predefined ranges in LEO to MEO. The transfer time is randomly selected between one-third and one-half of the orbital period, after which a Lambert transfer is computed. For the inspection class, an additional relative orbit is defined using relative orbital elements corresponding to a circular inspection orbit with a radius of 5 km to 10 km around the target. For the orbit-maintenance class, the same initial orbital element ranges are used, after which the chaser is subjected to a random altitude change of 3 km to 10 km or an inclination change of 0° to 10° before returning to its nominal orbit.

The resulting dataset contains a time series of the absolute state of both spacecraft. At each time step, the model receives the position, velocity, altitude, longitude, and latitude of both the target and chaser, yielding an 18-dimensional feature vector. This $(N, 18)$ vector is used for classification, with the entire sequence being used as input.

Bala [88] is not only unique for its use of a Bayesian neural net, but also for its use of exogenous variables. Exogenous variables in this case are properties of either the target or the chaser. In the research, Bala [88] consider the owner of the chaser satellite, the North American Aerospace Defense Command (NORAD) identifier (ID), and up to two payloads onboard the chaser. These exogenous variables aid in distinguishing intent, mostly due to the payload information. Payloads including robotic arms, cameras, or extra fuel tanks are the main driving information. The payload information allows for discerning intent for an already identified Motion Class, bringing the paper to Mission Class intent. What is not investigated, however, is whether the Motion Class does actually occur for the given Mission Class. Furthermore, how this payload information would be acquired in real-world operations is not investigated.

The intent inference approach from Goulet et al. [84] is unique among the body of literature considered. Optimal control for data generation is also used by Yuan et al. [75], but turning the optimal control problem around to use it for intent inference is not done elsewhere for non-cooperative RPOs, and this approach falls under the category GRA from Section 2.2. One issue with the approach, however, is its interpretability. Trajectories are generated according to the objective functions, but multiple objective

¹*Inspection* is sometimes translated as “fly-by”, but from the description of the trajectory it is a circular motion around the target.

functions can be combined together. There is, for example, the combination of the rendezvous and inspection intents, which produces an inward spiralling motion. An operator using this method would get, if the method worked, an indication that the control program had a weight for inspection and for rendezvous. But what this combination of weights means is up to the operator to discern. Furthermore, the method does not actually work for this case, with the weight for rendezvous not converging to a stable value.

2.4. Literature Gaps

A fundamental limitation of the current literature is the definition of intent itself. Most studies focus on Motion Classes, which, although useful as an intermediate representation, do not represent the true intent of a spacecraft. Rather than describing why a satellite performs a manoeuvre, Motion Classes merely characterise how it moves. Mission Classes provide a more meaningful representation of operational intent, yet the existing literature considers them only to a limited extent, with minimal justification as to why a given behaviour constitutes an intent. Evaluating the link between real-world observed behaviour and the intent behind it, and building a classifier that is trained on data that tries to mimic real-world behaviour, would constitute a significant addition to the literature.

A second limitation concerns the orbital regimes considered during model development. The existing literature is heavily concentrated on GEO. While some studies explicitly restrict their scope to GEO, others do so implicitly by generating training data using only GEO-representative altitude ranges. A small number of studies additionally consider MEO (5000 km altitude in [75] and 7000 km in [86]), and only one considers LEO [89]. This lack of literature in this regime is particularly noteworthy, as LEO contains the majority of operational satellites and therefore represents the environment in which non-cooperative rendezvous and proximity operations are most likely to occur. A plausible explanation is the increased complexity of LEO dynamics, where atmospheric drag, higher-order gravitational perturbations, shorter orbital periods, and a greater diversity of orbital planes could complicate both trajectory generation and model performance. Nevertheless, real-world examples such as the Russian NIVELIR programme (see Chapter 4) demonstrate that non-cooperative RPOs in LEO are operationally relevant. Extending intent recognition frameworks across domains therefore remains an important research gap.

Another important limitation lies in the generation of training data. Most studies assume that both detection and classification are performed onboard the target spacecraft, thereby restricting the available observations to close-range relative state measurements. This assumption excludes operationally relevant observation sources such as ground-based sensors and third-party space-based surveillance systems. Furthermore, the majority of studies generate trajectories using either the CW equations or two-body dynamics initialised with CW-derived conditions, neither of which fully captures the complexity of real orbital environments. As demonstrated by Sun et al. [86], models trained using CW-propagated trajectories exhibit a substantial reduction in performance when evaluated on trajectories generated using two-body dynamics due to the effects of the linearisation assumption in CW-equations, illustrating the limited generalisability of such datasets. Kelecý et al. [92] adopt a more realistic approach by basing their simulations on historically observed spacecraft behaviour. However, their dataset contains only a small number of representative examples with limited variability, increasing the likelihood of overfitting while also restricting the diversity of represented intent classes. Developing datasets that incorporate higher-fidelity dynamics and longer observation windows prior to closest approach would address several of these shortcomings simultaneously.

Furthermore, the current literature relies almost exclusively on relative state time series for intent inference, using the observed trajectory as the sole source of information. Although manoeuvres are implicitly reflected in these trajectories, no reviewed study explicitly incorporates manoeuvre detection or changes in orbital elements as model inputs, despite these quantities being directly informative of spacecraft intent. Moreover, exogenous information, including launch history, country of origin, previous operational behaviour, and known spacecraft capabilities, is almost entirely ignored. Such contextual information could provide valuable prior knowledge and improve classification performance, particularly during the early stages of an encounter when relatively little trajectory information is available. However, when it is implemented, the operational availability is questionable. Full information on the exact payloads of non-cooperative chasers is most likely not freely available in most cases.

Related to this is the limited attention given to early intent recognition. With the partial exception of Yuan et al. [75], which evaluates classification performance at detection distances of up to 16 km, none of the reviewed studies investigate how accurately intent can be inferred well before the closest approach. From an operational perspective, however, early warning is considerably more valuable than high-confidence classification shortly after an encounter. Future work should therefore characterise how classification performance evolves as a function of lead time and determine the minimum observation window required for reliable intent recognition.

"The only way of discovering the limits of the possible is to venture a little way past them into the impossible."

Arthur C. Clarke, *Profiles of the Future: An Inquiry into the Limits of the Possible* - 1973

3

Research Questions and Thesis Scope

This chapter defines the research questions of the thesis and their currently defined scope. Section 3.1 contains the formulation of the research questions and the aim of each.

3.1. Research Question Formulation

The research conducted in this thesis is guided by the following main research question (MQ).

Main Research Question

MQ: How can the intent of a non-cooperative satellite's rendezvous and proximity operation be inferred early?

The purpose of the thesis relates to the intent characterisation of non-cooperative RPOs. For this, three main things need to happen: non-cooperative RPOs need to be defined and characterised, representative data of non-cooperative RPOs needs to be generated, and a model to infer intent needs to be trained and assessed. Once this has been done to a sufficient level, the model itself can be further investigated, addressing questions such as what quality the input data needs to be and whether the model can be deliberately fooled. The research questions (RQs) relate to these topics and follow in part from the literature gaps identified in Chapter 2.

3.1.1. Characterisation of Non-Cooperative RPOs

Sub Research Question 1

RQ 1: What observable characteristics define non-cooperative rendezvous and proximity operations in Earth orbit?

The answer to this research question forms the basis for which intents to consider, how to simulate them, and how to obtain real-world validation data.

3.1.2. RPO Data Generation and Representation

Sub Research Question 2

RQ 2: How can synthetic datasets of non-cooperative RPO trajectories be generated and represented to enable intent inference?

This question arises from the observed literature, which trains models on synthetic data due to a lack of real-world data. However, the literature shows a gap for early intent inference, partly due to the training data used, which necessitates a different generation process and potentially a different representation of the data to enable early intent inference.

3.1.3. Machine Learning for Intent Prediction

Sub Research Question 3

RQ 3: Which machine learning model architectures are suitable for inferring intent in non-cooperative RPO scenarios?

Whilst the literature shows recurrent neural networks and transformer architectures performing well, this is in part due to their data input. With a potentially new synthetic data structure, an investigation into which model performs best on this structure is required.

3.1.4. Data Degradation

Sub Research Question 4

RQ 4: How does the performance of the intent prediction model degrade under non-ideal data conditions?

The literature shows that degradation does hinder classification performance, but this has not been investigated in the context of early intent inference. There is thus a need to understand the effects of degradation on such a model.

3.1.5. Real World Performance

Sub Research Question 5

RQ 5: How does the model perform on real world (non-cooperative) rendezvous and proximity operations?

The literature does not evaluate model performance on real-world data, cooperative or non-cooperative alike. This is partly due to a lack of real-world data availability, but assessing performance on even a limited number of real-world cases can give insight into how a model performs in practice and where improvements are required.

"If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle."

Sun Tzu, *The Art of War* - 5th century BC

4

Characterising Non-Cooperative RPOs

To infer the intent of a non-cooperative RPO, a model requires an understanding of what such an operation looks like in practice. This chapter builds that understanding in three steps: Section 4.1 reviews what has historically occurred, Section 4.2 examines specific RPO examples in detail, and Section 4.3 closes the chapter by combining these observations into the intent categories used by the model, along with the reasoning behind them.

4.1. A Non-Extensive Overview of Non-Cooperative RPOs

Information on non-cooperative RPOs is available in the open source, with extensive overviews provided by the Center for Strategic and International Studies [3, 96] and the Secure World Foundation [4]. However, these assessments rely on open-source analysis, observations from hobbyists, public statements, reports from government officials, and, analyses of open-source tracking data. This reliance limits the certainty of the assessments: what a satellite did, and how it did so, can be inferred from what is observed, but no public confirmation of intent is available.

Drawing on this open-source record, Table 4.1 presents an overview of non-cooperative RPOs to date. It covers both actual non-cooperative RPOs, cases where the target operator was not involved in planning the encounter, and domestic technology demonstrators, which either directly demonstrated non-cooperative RPO behaviour or showcased dual-use technology capable of enabling it.

Table 4.1: Known or suspected non-cooperative RPO satellites and programmes. Entries under *Other anomalous behaviour* did not conduct confirmed RPO against foreign assets but displayed activity consistent with inspector or counter-space roles, or have uncertain classification.

Name	Origin	Target(s)	Target Origin	Years Active	Description	Sources
<i>Russian GEO eavesdropping satellites</i>						
Luch / Olymp-K 1	Russia	Express AM-33, Intelsat 901, Intelsat 905, Intelsat 10-02, Eutelsat 10A, Intelsat ND, Paksat 1R, Turksat 4A, Eutelsat 28B, Turksat 4B, NSS-12, Intelsat 906, Intelsat 33E, Intelsat 17, Intelsat 20, Eutelsat 70B, Eutelsat 5W, Eutelsat 8W, ABS 3A, Azerspace 2, Intelsat 39, Intelsat 37E	Russian; Luxembourgish-American; French; Pakistani; Turkish; Luxembourgish; Hong Kong; Azerbaijani	2014–2025	First Russian GEO orbit-hopper. Repeatedly manoeuvred along the GEO belt, loitering in proximity to commercial communications satellites for extended periods. Visited approximately 30 GEO slots. Fragmented 2026.	[3, 4]
Luch / Olymp-K 2	Russia	Intelsat 39, Eutelsat KA-SAT (9A), Eutelsat 9B, SES 5, Astra 4A (Sirius 4), Eutelsat 3B, RASCOM-QAF 1R, Eutelsat KONNECT VHTS, Thor 7, Intelsat 10-02	Luxembourgish-American; French; Luxembourgish; Pan-African (45 countries); Norwegian	2023–present	Follow-on to Olymp-K 1. Exhibited identical loitering behaviour with a larger number of documented visits.	[3, 4]
<i>US GEO eavesdropping satellites</i>						
PAN / Nemesis-1 (USA-207)	USA	Yahsat 1B, others (unconfirmed)	United Arab Emirates; unknown	2009–2021	US signals intelligence (SIGINT) satellite with limited publicly available data. Observed manoeuvring along the GEO belt into proximity with commercial communications satellites, behaviour consistent with the interception of satellite communications downlinks in addition to a general SIGINT collection role, though the mission has not been officially confirmed. Presumed inactive after 2021.	[4]
Mentor 4 (USA-202)	USA	Thuraya 2, other	United Arab Emirates	2009–unknown	Part of the US Mentor/Advanced Orion GEO SIGINT programme, tasked with broad signals collection. Initially detected collecting signals over China before manoeuvring to the Middle East, where it loitered near Thuraya communications satellites, suggesting the interception of satellite communications alongside its wider SIGINT mission. Current status unknown.	[4]
<i>US GEO reconnaissance satellites</i>						

Continued on next page

(Table 4.1 continued)

Name	Origin	Target(s)	Target Origin	Years Active	Description	Sources
GSSAP (USA-253, 254, 270, 271, 324, 325; Hornet 7, Hornet 8)	USA	Reported: TJS-1, Express AM-8, Luch/Olymp-K 1, Paksat 1R, Nigcomsat 1R, Raduga-1M 3, Raduga-1M 2, Tianlian 2-01, B-2 G8, SJ-13, SJ-20 / Chinasat 6A, TJS-3, SY-12 01, SY-12 02, TJS-15, TJS-16, TJS-17	Chinese, Russian, Pakistani, Nigerian; others presumed	2014–present	US programme of dedicated GEO space-situational-awareness satellites. Operated similarly to Olymp-K, manoeuvring along the GEO belt to conduct close approaches to other satellites. Individual approach events are not officially disclosed, and the target list given here is drawn largely from Chinese open-source reporting, which is more detailed than Western sources but is unverified and may reflect the reporting party's own interests; the list should therefore be treated as indicative rather than confirmed. One satellite decommissioned to graveyard orbit.	[90, 4]
LDPE 3A	USA	SJ-23	Chinese	2023–unknown	LDPE satellites are host satellites, able to carry modular payloads and other smaller satellites. LDPE 3A hosted an SSA payload and approached SJ-23, presumably for inspection.	[4]
Mycroft (USA-287)	USA	EAGLE, S5	USA	2018–unknown	Technology demonstrator. Mycroft was deployed from EAGLE as a subsatellite and subsequently performed activities related to SSA. It later moved to S5, which had malfunctioned, manoeuvring to determine the status of S5's solar arrays and antennas.	[4]
<i>Chinese GEO reconnaissance satellites</i>						
SJ-17	China	Chinasat 5A, Chinasat 6A, Chinasat 20, Chinasat 1C, Chinasat 6B, Apstar 6, Yamal 202, Geofen 13, Luch 5V (not Olymp), Beidou-2 G4, Luch 5A (not Olymp), Horizons 3E, VENESAT-1, Yamal 300K	Chinese; Russian; Hong Kong; Luxembourgish-Japanese; Venezuelan	2016–present	Developed by CAST and launched to demonstrate electric propulsion and a robotic arm, per 2021 U.S. Space Command congressional testimony. Conducted robotic RPO demonstrations across the GEO belt: circumnavigated Chinasat-5A, carried out RPOs with Chinasat-6B (Jan. 2020), SJ-20 (Oct. 2020) and Chinasat-20, then executed a rapid inclination change to rendezvous with and follow the anomalous, drifting Chinasat-1C for about a week, likely to inspect it and monitor its recovery. Also conducted a brief RPO with the Venezuelan satellite VENESAT-1 (Nov. 2023) during an extended drift that carried it above and below the GEO belt before it resettled at 178°W in late 2024.	[4, 96]
SY-12 01 and 02	China	USA-270, USA-336, others	USA	2021–present	SY-12 01 and 02 are drifters in the GEO belt, occasionally stopping near satellites. During the approach by USA-336, SY-12 02 manoeuvred to obtain optimal lighting conditions to photograph the target; the same occurred during the approach of USA-270.	[4, 96]
<i>US counter-space satellites</i>						
Delta 180 Payload Assist System	USA	Delta 2/B	USA	1986	Technology demonstrator. The Payload Assist System intercepted the Delta 2/B rocket body, which was actively thrusting at the time.	[4]
<i>Russian suspected counter-space satellites (NIVELIR programme)</i>						

Continued on next page

(Table 4.1 continued)

Name	Origin	Target(s)	Target Origin	Years Active	Description	Sources
Cosmos 2542 & 2543	Russia	USA-245	USA	2019–present	Cosmos 2542 is the parent vehicle of subsatellite Cosmos 2543. Cosmos 2542 manoeuvred into close proximity with the US reconnaissance satellite USA-245; Cosmos 2543 subsequently released a projectile in its vicinity, raising concerns about kinetic ASAT capability. Believed to form part of the NIVELIR programme. Cosmos 2543 had previously approached Cosmos 2535 in a domestic proximity operation (see below).	[3, 4]
Cosmos 2558	Russia	USA-326	USA	2022–present	Inserted directly into an orbit co-planar with US reconnaissance satellite USA-326, then manoeuvred to maintain proximity and deployed a smaller sub-object. Remained near USA-326 for several years, leaving after June 2025. Believed to form part of NIVELIR.	[3, 4]
Cosmos 2576	Russia	USA-314	USA	2024–present	Launched directly into a co-planar orbit with US reconnaissance satellite USA-314. Ceased orbital synchronisation approximately February 2025, with a co-planar but lower orbit currently. Designated a “space weapon” by the US government. Believed to form part of NIVELIR.	[3, 4]
Cosmos 2588	Russia	USA-338	USA	2025–present	Behaviour consistent with Cosmos 2558 and 2576; manoeuvred into proximity with US reconnaissance satellite USA-338. Believed to form part of NIVELIR.	[4]
Cosmos 2519 (NIVELIR-L)	Russia	Cosmos 2521, Cosmos 2523	Russia (domestic)	2017–2018	Assessed to be the first NIVELIR pathfinder. Did not conduct RPO against a foreign asset, but demonstrated a nested deployment architecture: Cosmos 2519 released subsatellite Cosmos 2521, which in turn released Cosmos 2523.	[3, 4]
Cosmos 2543	Russia	Cosmos 2535	Russia (domestic)	2019–present	Cosmos 2543 approached Cosmos 2535 and released an object nearby. Cosmos 2543 would later conduct RPO with USA-245.	[4]
<i>Soviet counter-space satellites (IS programme)</i>						
Cosmos 185, 249, 252, 374, 375, 397, 404, 462, 804, 814, 843, 886, 910, 918, 961, 970, 1009, 1174, 1243, 1258, 1379	Soviet Union	Cosmos 248, 373, 394, 400, 459, 803, 839, 880, 909, 959, 967, 1171, 1241, 1375	Soviet Union	1963–1982	An anti-satellite weapon programme with tests spanning multiple decades. A chaser satellite would explode near its target to destroy it; some tests involved multiple chasers directed at a single target. Multiple successful tests occurred.	[4]
<i>Other anomalous behaviour</i>						

Continued on next page

(Table 4.1 continued)

Name	Origin	Target(s)	Target Origin	Years Active	Description	Sources
Victus Haze Puma	USA	Jackal-0004 (True Anomaly)	USA (domestic)	2026–present	US Space Force tactically responsive space (TacRS) demonstrator, built by Rocket Lab. Following a snap launch notice on 19 June 2026, Puma was deployed by Electron 16 h 42 min later; True Anomaly's Jackal-0004, already on orbit, located Puma from 2,000 km using onboard sensors and completed a rapid RPO intercept, circumnavigating and imaging it within 61 hours of the action order against a 72-hour target. In a second sortie tasked 14 July 2026, Puma manoeuvred away and Jackal-0004 pursued and imaged it, described as the first space-to-space pursuit of an evading target. Further sorties with the two spacecraft trading chaser and chased roles were planned over the following six months.	[97, 98]
Cosmos 2562	Russia	Resurs-P3	Russia (domestic)		Resurs-P3 had been considered inactive. Over approximately one year, Resurs-P3 manoeuvred to place itself in a close formation orbit with Cosmos 2562, believed to enable the latter to conduct an inspection. No activity observed since.	[3, 4]
Cosmos 2581, 2582 & 2583	Russia	Each other	Russia (domestic)	2025–present	Three satellites launched together that subsequently conducted mutual proximity operations of undisclosed purpose. One released an unidentified sub-object, generating concern regarding potential ASAT capability.	[3, 4]
TJS-3	China	TJS-3 AKM, Luch/Olymp-K, USA-233, USA-263, Chinasat 10, Chinasat 16, SJ-20, Chinasat 12, TJS-10	Chinese; USA; Russian	2018–present	Purpose unknown; potentially eavesdropping or reconnaissance. Demonstrated RPO behaviour in coordination with other Chinese satellites. TJS-3 AKM manoeuvred with TJS-3 to GEO and remained together for half a year. When TJS-3 began drifting along the GEO belt, TJS-3 AKM manoeuvred to occupy its original location, potentially complicating tracking of TJS-3. Multiple approaches confirmed with other satellites, but at distances of order 100 km.	[3, 96]
SY-24C-01, -02 & -03	China	SJ-6-05A; SJ-6-05B	China (domestic)	2023–present	Five Chinese satellites conducting mutual RPOs in 2024 and possibly beyond. SY-24C-01 to -03 performed corkscrew manoeuvres around SJ-6-05B; SY-24C-03 subsequently conducted close RPOs with SJ-6-05A. Purpose unknown.	[3]
SJ-12	China	SJ-06F	China (domestic)	2010–present	SJ-12 was launched with the stated purpose of being a technology demonstrator for environment probing, measurements, and communications. It manoeuvred close to SJ-06F, believed to be an electronic intelligence satellite; based on observed behaviour, the two satellites most likely collided. Exact purpose unknown.	[4]
SY-7, CX-3 & SJ-15	China	Each other	—	2013–2016	All three satellites were launched together as part of a technology demonstrator for “space maintenance”. SY-7 carried a robotic arm, CX-3 an optical surveillance payload, and SJ-15 RPO capabilities. SY-7 most likely captured one of the other two satellites.	[4]

Continued on next page

(Table 4.1 continued)

Name	Origin	Target(s)	Target Origin	Years Active	Description	Sources
SJ-21	China	Compass G2; SJ-25	China (domestic)	2021–present	Launched by SAST with the stated purpose of testing space debris mitigation technology. In January 2022, after a month of slow approach, SJ-21 grappled the defunct Chinese navigation satellite Compass G2 and pulled it into a graveyard orbit above GEO, prompting a 2022 U.S. Defense Intelligence Agency report to warn that robotic-arm technology could be used to grapple other satellites. SJ-21 later conducted extended RPOs with SJ-25, including an apparent docking from July to November 2025 followed by further RPOs until mid-January 2026.	[4, 96]

The behaviours documented in Table 4.1 have been grouped into a few categories: reconnaissance, eavesdropping, counter-space, and a broader category of "other" satellites.

For the reconnaissance satellites, the Geosynchronous Space Situational Awareness Program (GSSAP) stands out as a programme that has been active for over a decade, involves numerous satellites, and owing to its status as a semi-classified US programme, has relatively little public reporting associated with it. The programme is stated to track and characterise satellites in the GEO belt [4]. Other reconnaissance activity is more fragmented: SJ-17 has conducted a sustained series of close approaches across the Chinese GEO fleet, including an apparent inspection of the malfunctioning Chinasat-1C; SY-12 01 and 02 have manoeuvred to obtain favourable imaging geometry against US satellites; and LDPE 3A and Mycroft represent one-off inspection payloads hosted on or deployed from other US spacecraft, rather than dedicated, repeat-visit programmes in the mould of GSSAP or SJ-17.

Then there are the eavesdropping satellites, the prime examples being Luch/Olymp-K(-1) and Luch/Olymp-K-2, but also suspected activity from PAN/Nemesis-1, Mentor-4, and TJS-3. These satellites approach other satellites in the GEO belt and listen in on the communications being sent to them. It is worth noting that this behaviour occurring predominantly in the GEO belt makes sense: GEO is home to stationary telecommunications satellites, each of which constitutes a single, concentrated, high-value target carrying a large volume of traffic. In LEO, by contrast, communications satellites are far more dispersed across a constellation, so listening in on any one satellite yields comparatively less information. GEO satellites are also stationary relative to the ground, allowing an eavesdropping satellite to maintain a listening position over a specific geographic location.

Then there are the counter-space satellites: weapons that have demonstrated the potential to kinetically destroy a target satellite. Historically, the United States and Soviet Union conducted tests and demonstrations with such platforms. More recently, the Russian NIVELIR programme has also been accused of being a counter-space weapons programme: while it has not yet demonstrated the direct destruction of a target, it has demonstrated the capability to launch projectiles and sub-satellites from its on-orbit assets.

Finally, a broader "other" category captures behaviours that do not cleanly fit the three roles above, either because the purpose is unconfirmed or because the mission does not centre on reconnaissance, eavesdropping, or destruction. Victus Haze Puma demonstrates a rapid-response inspection capability rather than a sustained programme, while the target satellite's own evasive manoeuvring illustrates that non-cooperative RPO is not necessarily a one-sided activity. The SY-24C-01/-02/-03 and SJ-12/SY-7/CX-3/ SJ-15 clusters, along with TJS-3, involve Chinese satellites conducting RPOs of undisclosed purpose, in some cases against each other rather than against a foreign target; these are included because the demonstrated manoeuvring and, in some cases, robotic-arm or capture capability would be directly applicable to non-cooperative scenarios, even without confirmed use against a foreign asset. SJ-21 sits at the sharpest edge of this category: its 2022 grapple and relocation of the defunct Compass-G2 satellite is the clearest publicly documented instance of one satellite physically capturing another, and its extended RPO campaign with SJ-25, including an apparent docking, suggests the demonstrator role may be evolving into an operational one.

4.2. A Closer Look: Analysis of Real-World RPOs

Whilst the information in Table 4.1 gives an overview of the cases that have happened, insight into how they occurred and general trends in certain groupings is lacking. What follows is a deep dive of some real world RPOs, cooperative and non-cooperative alike.

4.2.1. Case Study: LUCH-1 and LUCH-2

As indicated in Table 4.1, LUCH Olymp K(-1) and LUCH Olymp K-2 are considered GEO eavesdropping satellites, meaning they listen in on communications sent to the target satellite they are near. This assessment mainly comes from the satellites they approach; these are communications satellites operated by either commercial or military operators [4, 3]. Their actual capabilities are unknown, as the Russian state has never publicly acknowledged their actual purpose.

Behaviour-wise, these satellites are unusual compared to the rest of the GEO belt. Due to their nature as eavesdropping satellites, they visit the orbital slots of other stationary satellites in the belt, resulting in

frequent movement: K-1 visited 29 targets before it was decommissioned, and K-2 has visited 8 targets to date. Across both vehicles, this movement follows a consistent, repeatable three-phase pattern: 1. An initial burn to deviate from the GEO belt in terms of semi-major axis (SMA), causing a relative drift to the belt. 2. A reduction in the difference of the chaser SMA with the GEO belt, causing a smaller drift rate. 3. Arrival at the GEO belt, where the chaser is at an offset in the in-track direction with its target.

Reconstructing the close-in behaviour (Phase 3) from two-line element (TLE) data has notable limitations. TLEs do not resolve manoeuvre timing precisely, so a manoeuvre executed between two TLE epochs can appear as a sudden, unphysical jump in relative position rather than a smooth transition. This is visible in several of the examples, most clearly in the Intelsat 39 (K-1) and Thor 7 (K-2) cases, where the reconstructed relative motion is erratic. This limitation affects the fine detail of the close-in trajectory but does not undermine the three-phase pattern itself, which is consistently observable in the approach geometry regardless of TLE quality.

The targets were determined from mentions of targets in public sources [4, 3], checked against the epochs at which the respective LUCH satellite's SMA matched the GEO belt, and thus did not drift relative to other GEO satellites. At these epochs, the instantaneous longitude of the LUCH satellite was computed via its true longitude,

$$u = \Omega + \omega + \nu \pmod{360^\circ} \quad (4.1)$$

where Ω , ω , and ν are the right ascension of the ascending node (RAAN), argument of pericentre, and true anomaly, respectively, and the closest target in terms of this instantaneous longitude was taken.

The movement history of K-1 is presented in Figure 4.1, with the targets listed in Table 4.2. Each start of an RPO is indicated by a vertical *Close approach* line.

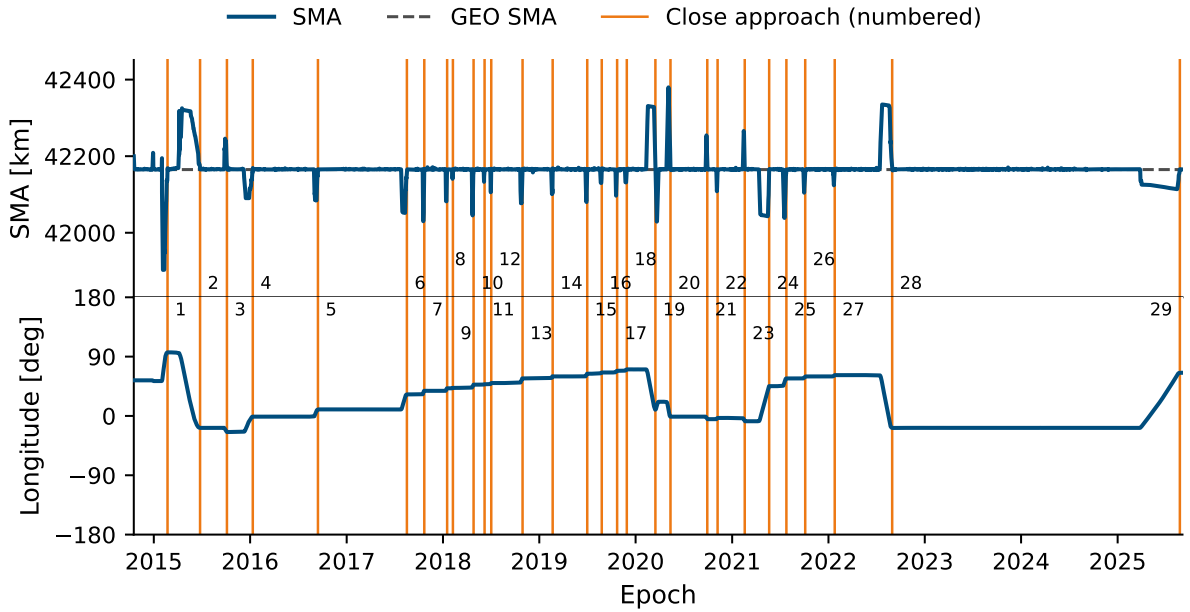


Figure 4.1: Overview of visits by LUCH Olymp K.

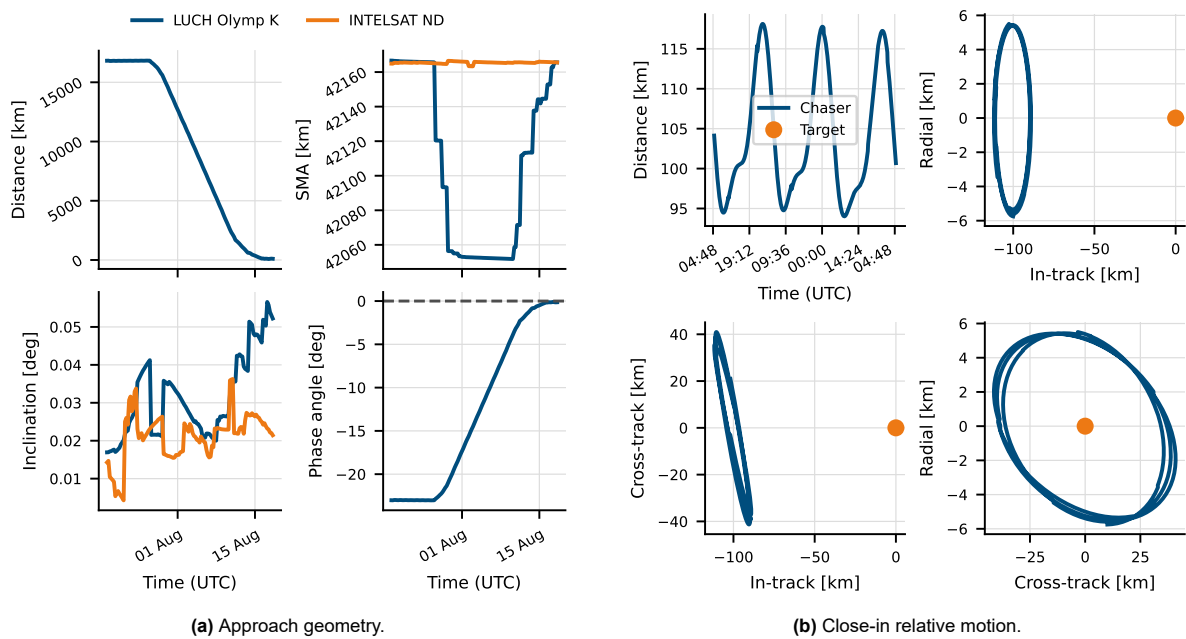
The time spent near a target does not follow a clear pattern: for some targets K-1 remains close for less than a month, whilst with Intelsat 37E it remained close for almost three years. The targets of each close approach are presented in Table 4.2.

Table 4.2: Close-approach events annotated in the NORAD 40258 overview (Figure 4.1).

#	Target	NORAD ID	Start Date	End Date	Start Longitude (Chaser/Target) [°]	End Longitude (Chaser/Target) [°]
1	EXPRESS AM-33	32478	2015-01-31	2015-02-21	52.90 / 96.46	96.41 / 96.46
2	INTELSAT 901	26824	2015-04-04	2015-06-25	95.69 / -18.05	-18.13 / -18.01
3	INTELSAT 905	27438	2015-09-24	2015-10-05	-18.07 / -24.50	-24.27 / -24.45
4	INTELSAT 1002	28358	2015-12-07	2016-01-11	-24.36 / -0.98	-1.09 / -0.98
5	EUTE 10A	34710	2016-08-29	2016-09-14	-1.08 / 10.03	9.88 / 9.95
6	INTELSAT ND	37392	2017-07-26	2017-08-17	9.79 / 32.81	32.67 / 32.81
7	PASAT 1R	37779	2017-10-16	2017-10-22	32.77 / 38.04	38.10 / 38.04
8	TURKSAT 4A	39522	2018-01-10	2018-01-16	38.06 / 42.06	41.92 / 41.98
9	TURKSAT 4A	39522	2018-02-05	2018-02-07	41.89 / 42.01	42.50 / 42.03
10	EUTE 28B	33460	2018-04-20	2018-04-26	42.57 / 48.07	47.46 / 48.10
11	EUTE 28B	33460	2018-06-04	2018-06-07	47.53 / 48.06	48.05 / 48.12
12	TURKSAT 4B	40984	2018-06-29	2018-07-02	48.00 / 49.98	49.91 / 49.98
13	NSS-12	36032	2018-10-20	2018-10-29	50.11 / 57.07	56.95 / 57.06
14	INTELSAT 33E	41748	2019-02-14	2019-02-20	57.04 / 60.02	59.93 / 59.99
15	INTELSAT 906	27513	2019-06-24	2019-07-01	59.92 / 64.19	64.05 / 64.15
16	INTELSAT 17	37238	2019-08-19	2019-08-25	64.07 / 66.02	65.92 / 65.96
17	INTELSAT 20	38740	2019-10-16	2019-10-22	66.00 / 68.51	68.47 / 68.50
18	EUTE 70B	39020	2019-11-21	2019-11-28	68.47 / 70.46	70.64 / 70.48
19	EUTE KA SAT 9A (KA SAT)	37258	2020-02-10	2020-03-16	70.55 / 8.99	9.38 / 8.97
20	INTELSAT 1002	28358	2020-04-26	2020-05-12	21.53 / -0.95	-1.21 / -1.02
21	EUTE 5W	44624	2020-09-22	2020-09-28	-1.07 / -5.03	-4.89 / -4.97
22	ABS 3A	40424	2020-11-01	2020-11-06	-4.82 / -2.94	-3.03 / -2.94
23	EUTE 8W	40875	2021-02-09	2021-02-17	-3.06 / -7.97	-8.08 / -8.02
24	AZERSPACE 2	43632	2021-04-12	2021-05-21	-8.08 / 45.13	45.19 / 45.13
25	NSS-12	36032	2021-07-12	2021-07-25	45.17 / 56.98	56.86 / 57.01
26	INTELSAT 33E	41748	2021-09-27	2021-10-04	56.97 / 60.01	59.92 / 60.04
27	INTELSAT 39	44476	2022-01-17	2022-01-24	59.94 / 61.94	61.84 / 61.95
28	INTELSAT 37E	42950	2022-07-13	2022-08-30	61.88 / -17.99	-18.32 / -18.02
29	INTELSAT 17	37238	2025-03-27	2025-08-24	-18.14 / 66.00	65.46 / 65.99

What does not become apparent from the behaviour in Figure 4.1 is the more detailed approach and close-in behaviour underlying the three-phase pattern. Two cases are examined below.

Figure 4.2 shows K-1's approach to Intelsat ND, which ended around 19 August 2017.

**Figure 4.2:** LUCH (Olymp-K) approach to Intelsat New Dawn, 19 August 2017.

As can be observed in Figure 4.2a, K-1 moves mainly by changing its SMA rather than its inclination, which remains close to zero and likely fluctuates due to measurement error or orbital perturbations. Consistent with Phase 1, the SMA is lowered gradually, over the course of a few days, to 100 km below the GEO belt. This gradual change is consistent with the electric thrusters LUCH Olymp K is stated to have [4]. The resulting drift decreases the distance and in-plane phase angle to the target, starting

from a 24° difference. Consistent with Phase 2, this drift is arrested by a climb back towards GEO, with a period spent about 40 km below the belt, before the SMA is increased once more to complete arrival (Phase 3). Around arrival, the distance is on the order of 100 km, with the phase angle reduced to near zero.

The close-in behaviour is shown in Figure 4.2b, a few days after arrival. A stable offset oscillating behaviour with an in-track separation of about 100 km is observed.

A second example is shown in Figure 4.3, which was with a different target.

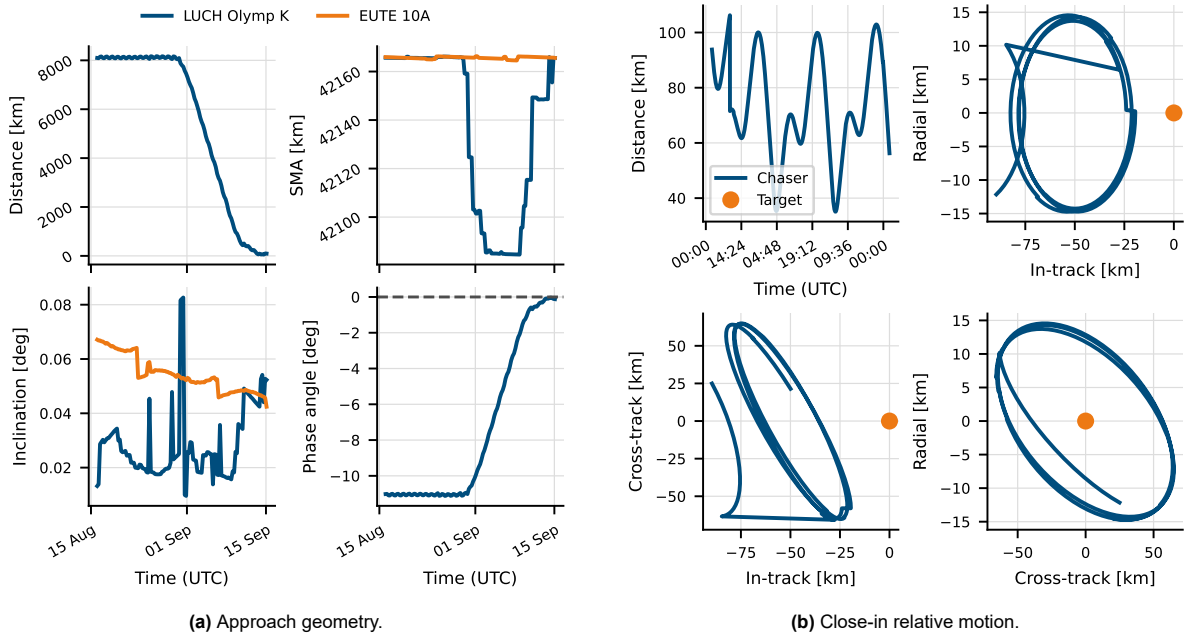


Figure 4.3: LUCH (Olymp-K) approach to EUTELSAT 10A, 16 September 2016.

The same phase structure is visible in Figure 4.3a, but with smaller magnitudes throughout: an SMA offset of about 60 km, correlating with a smaller phase-angle difference of 11° , and a slower drift rate at some 10 km below the GEO belt lasting almost a week. The in-plane difference at arrival is again less than a degree, at a distance on the order of 100 km. The close-in behaviour in Figure 4.3b shows a stable average in-track offset of 50 km.

The Intelsat 39 approach in Figure 4.4 illustrates the TLE limitation discussed above more severely. The approach itself (Phases 1–2) is unremarkable, travelling only 2° with an SMA offset of about 40 km. However, the reconstructed close-in behaviour (Figure 4.4b) is too erratic to represent a physically plausible orbit, and cannot be reliably attributed to either manoeuvring or TLE error alone given the available data.

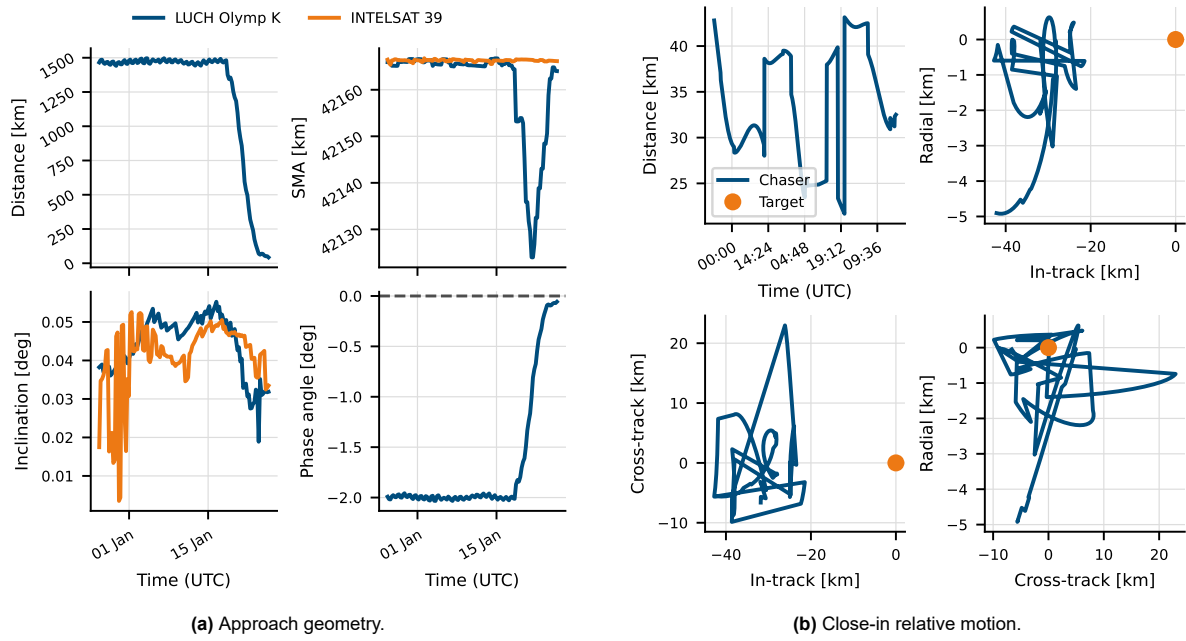


Figure 4.4: LUCH (Olymp-K) approach to INTELSAT 39, 25 January 2022. Example of unreliable data.

LUCH Olymp K-2 launched on 12 March 2023, almost 9 years after its predecessor, and the two satellites overlapped in operation for some time. K-2 replicates the same three-phase behaviour observed for K-1, with two notable differences: it has visited fewer targets (8 vs. 29), and its visits are longer on average (on the order of several months), reflecting both its shorter operational history to date and a tendency towards extended dwell periods at target slots. The visit history to date is presented in Figure 4.5.

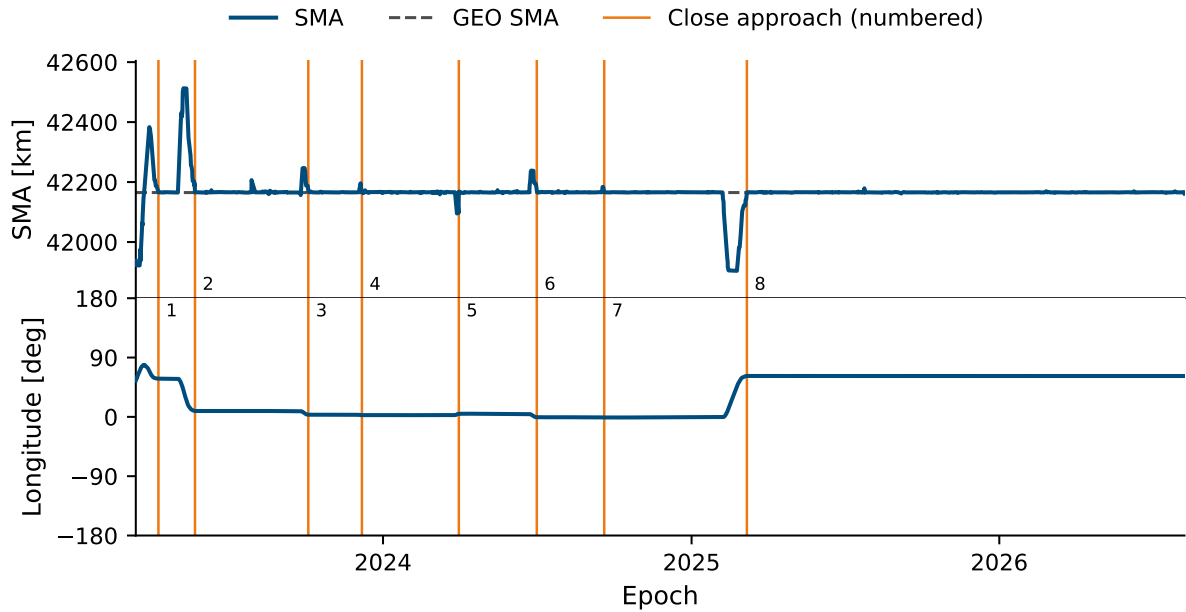


Figure 4.5: Overview of visits by LUCH Olymp K-2.

The targets were determined in the same way as for K-1, with the annotated events of Figure 4.5 listed in Table 4.3.

Table 4.3: Close-approach events annotated in the NORAD 55841 overview (Figure 4.5).

#	Target	NORAD ID	Start Date	End Date	Start Longitude (Chaser/Target) [°]	End Longitude (Chaser/Target) [°]
1	WGS F2	34713	2023-03-13	2023-04-09	54.97 / 57.49	58.00 / 57.50
2	EUTE 9B	41310	2023-05-02	2023-05-22	58.02 / 9.04	8.91 / 8.98
3	EUTE 3B	39773	2023-09-25	2023-10-04	8.98 / 3.12	3.26 / 3.08
4	EUTE KONNECT VHTS	53765	2023-12-03	2023-12-06	3.17 / 2.70	2.59 / 2.72
5	ASTRA 4A (SIRIUS 4)	32299	2024-03-26	2024-03-30	2.69 / 4.82	4.65 / 4.78
6	THOR 7	40613	2024-06-22	2024-07-01	4.69 / -0.62	-0.59 / -0.69
7	INTELSAT 1002	28358	2024-09-16	2024-09-19	-0.53 / -0.97	-0.92 / -0.99
8	INTELSAT 39	44476	2025-02-06	2025-03-07	-0.95 / 61.89	62.00 / 61.91

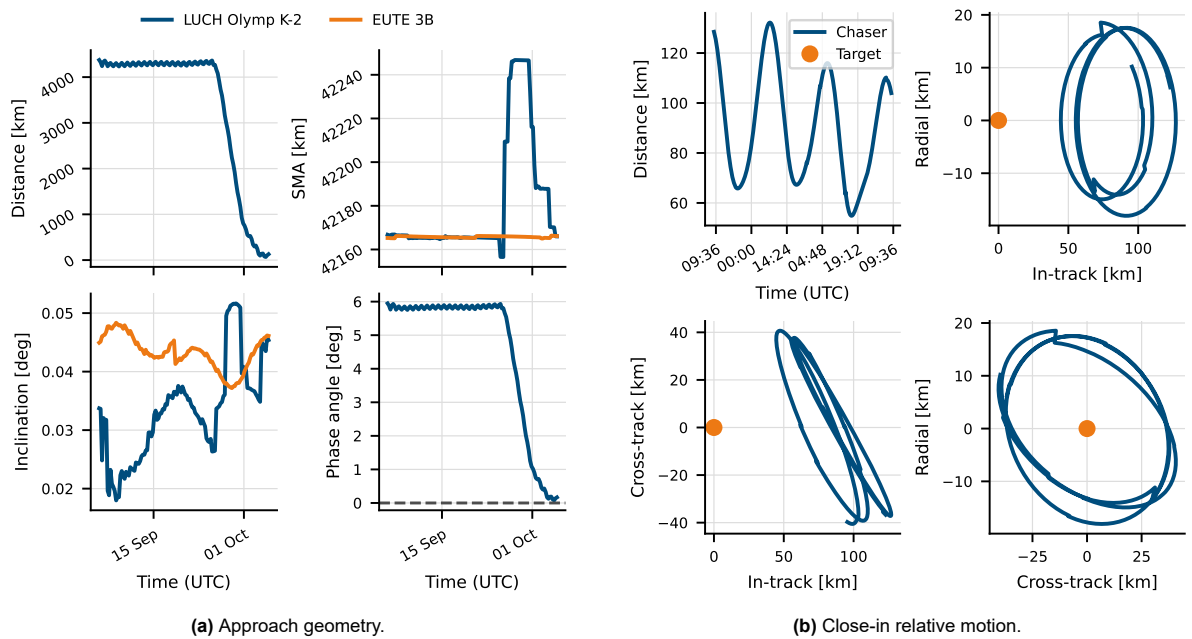


Figure 4.6: LUCH (Olymp-K2) approach to EUTELSAT 3B, 5 October 2023.

The approach and close-in behaviour for EUTELSAT 3B are presented in Figure 4.6, where around 6° of separation needed to be overcome, or approximately 4100 km. Consistent with Phase 1, this was done by increasing the SMA by some 80 km; consistent with Phase 2, it was then lowered step-wise, spending a few days some 30 km above the GEO belt before descending into the belt. The close-in behaviour (Phase 3) shows a stable offset oscillating behaviour, this time with an average in-track offset of some 90 km.

The visit to Intelsat 39, the second time this satellite has been visited by a LUCH satellite, is the longest-duration target of K-2, starting in March 2025 and ongoing as of writing.

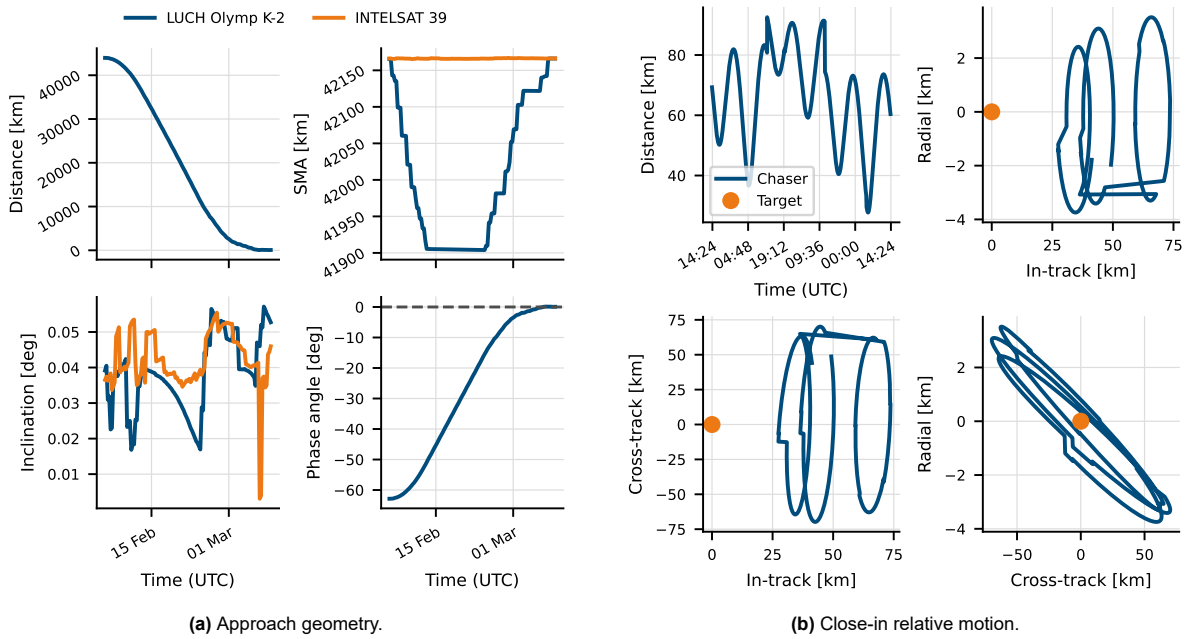


Figure 4.7: LUCH (Olymp-K2) approach to INTELSAT 39, 8 March 2025.

This approach is the furthest either satellite has moved along the GEO belt, requiring slightly less than a month and covering more than 60° of separation. The SMA was lowered by some 270 km over the course of about a week and remained there for more than a week before gradually increasing again; a clear Phase 2 dwell is observed at approximately 45 km below the belt, though the approach may include multiple smaller intermediate stops. The close-in behaviour in Figure 4.7b is consistent with the offset oscillating pattern, including a jump in the in-track direction comparable to that observed for Intelsat ND in Figure 4.2b.

K-2 shows the same TLE-reconstruction limitations as K-1, illustrated by the Thor 7 approach in Figure 4.8.

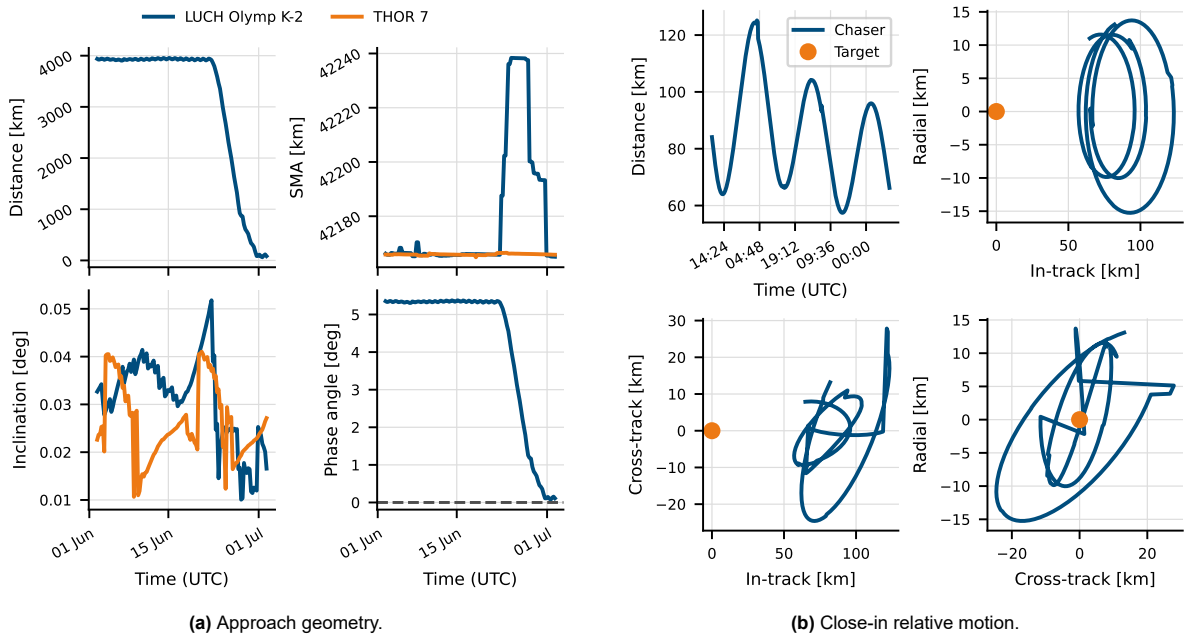


Figure 4.8: LUCH (Olymp-K2) approach to Thor 7, 2 July 2024. Example of unreliable data.

The close-in behaviour is dominated by jumps in the cross-track direction between TLEs, with smaller jumps in the radial and in-track directions. The approach itself (Phases 1–2) is consistent with the pattern described before.

Across both satellites, the observed behaviour consistently follows the three-phase pattern: an SMA offset to induce drift, an intermediate dwell, and arrival with sustained in-track station-keeping. The magnitude of the SMA offset and the duration of the drift phase scale with the required change in relative phase angle.

4.2.2. Case Study: LEO Inspections

Victus Haze, a United States tactically responsive space mission, provides an example of inspection behaviour in LEO. The mission paired Rocket Lab's Puma spacecraft, launched on 19 June 2026 under a rapid launch notice, with True Anomaly's Jackal-0004, already on orbit [97]. Rather than a single chaser-target pairing, the campaign was designed so that the two spacecraft would trade chaser and chased roles across a series of increasingly demanding sorties [99]: Jackal-0004 first located and imaged the newly deployed Puma, and in a later sortie pursued and imaged Puma as it manoeuvred away, described by the operator as the first space-to-space pursuit of an evading target [98]. The publicly stated objectives are the demonstration of space domain awareness and characterisation capabilities. The relative behaviour of the two spacecraft is shown in Figure 4.9, which for a given time-step takes the closest TLE (which may lie in the future), propagates that TLE to the epoch of the time-step, and converts the result to a Cartesian state. Multiple close approaches have occurred between the two spacecraft, and the two examples below illustrate this behaviour.

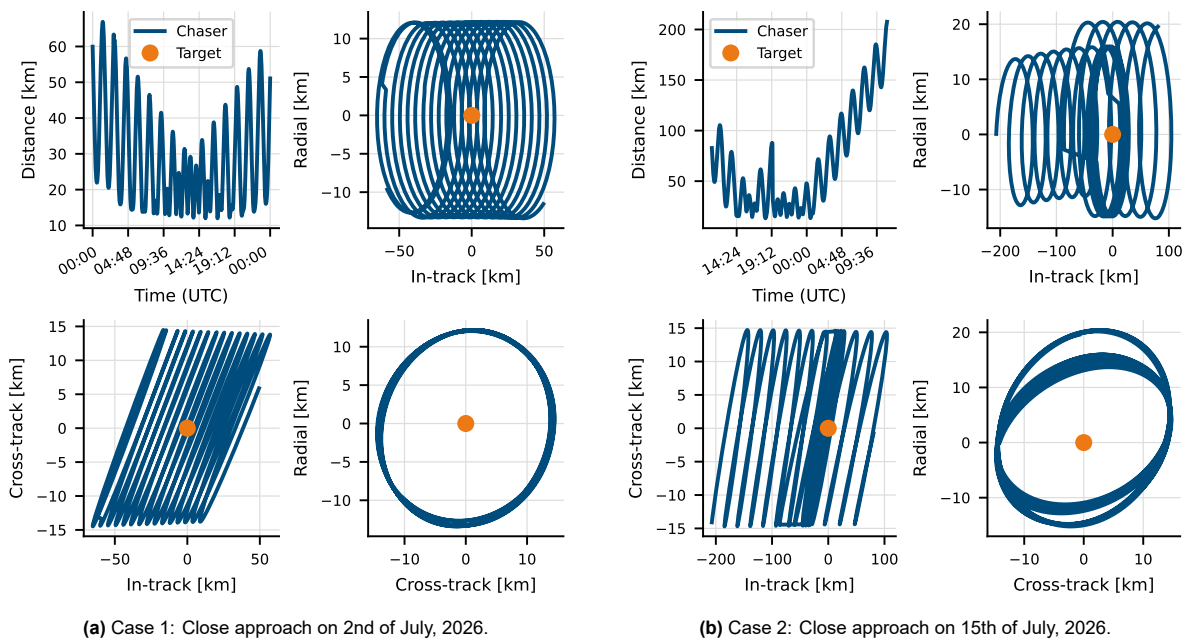


Figure 4.9: Victus Haze inspection behaviour in LEO.

A persistent circling drift in the in-track direction is clearly visible in both cases. The only difference between the two is that manoeuvres occur during the period, as indicated by the jumps in the plotted lines.

Further examples of LEO inspection behaviour come from Russian missions associated with the NIVELIR programme. Cosmos-2504 is suspected to have served as a technology demonstrator for the programme, conducting manoeuvres relative to the upper stage that launched it. Cosmos-2521, also attributed to NIVELIR, was deployed from Cosmos-2519, subsequently manoeuvred around it, and later released a further sub-satellite, Cosmos-2523. The close-in behaviour for these missions is shown in Figure 4.10.

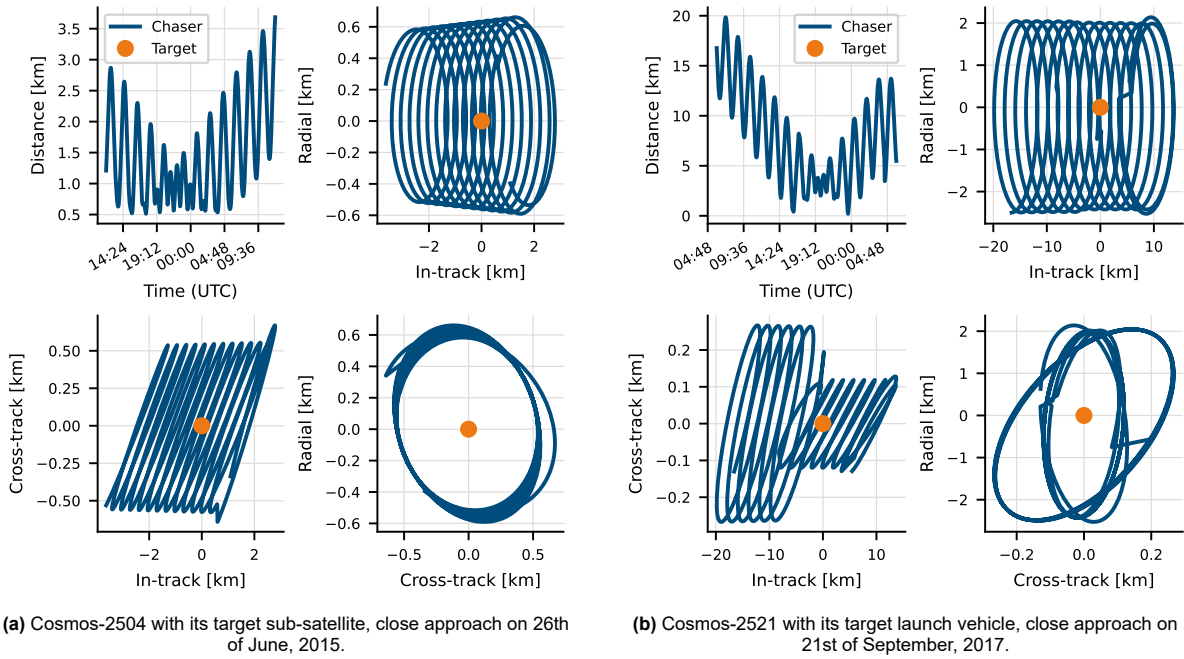


Figure 4.10: Inspection behaviour in LEO for Russian NIVELIR-programme missions.

The main difference from the Victus Haze case is the distance involved: these missions came within a few kilometres of their targets, rather than tens of kilometres, though the same circling drift is observed.

Cosmos-2504 and the Cosmos-2519/2521/2523 behaviours can be understood as pathfinders for the broader NIVELIR programme, whose later entries move beyond inspection towards more direct counter-space demonstrations. Cosmos-2542 and Cosmos-2543 replicate the same close-proximity approach behaviour as observed, with Cosmos-2543 going on to release a projectile near the US reconnaissance satellite USA-245. Cosmos-2558, 2576, and 2588 extend this further still: each was inserted directly into a co-planar orbit with a US reconnaissance satellite and maintained that position for an extended period, rather than departing after a single close approach. The stalking behaviour of the remaining NIVELIR satellites is harder to characterise, since their targets are classified and little to no data on them is available through Space-Track.

4.2.3. Case Study: GSSAP

The only publicly documented example of inspection behaviour in GEO comes from the GSSAP, although uncertainty remains on what exactly occurs during close approaches. GSSAP satellites are GEO intelligence assets operated by the United States Space Force, tasked with performing space situational awareness by visiting and observing other satellites within the geostationary belt [3, 4]. Their behaviour differs from signal intelligence satellites like the LUCH satellites, since they do not linger for long periods near their targets, and have been noted to visit satellites which are not communication satellites in GEO [4].

A significant complication in characterising GSSAP behaviour is that these are classified satellites for which TLE update rates are comparatively sparse, averaging on the order of 8 d between updates based on publicly available tracking data. This means trajectory accuracy near TCA can be poor: propagation to TCA can be based on a TLE almost a week old, and in some cases closer to a month old. For cases with a TLE only a few days old, the plots are shown in Figure 4.11.

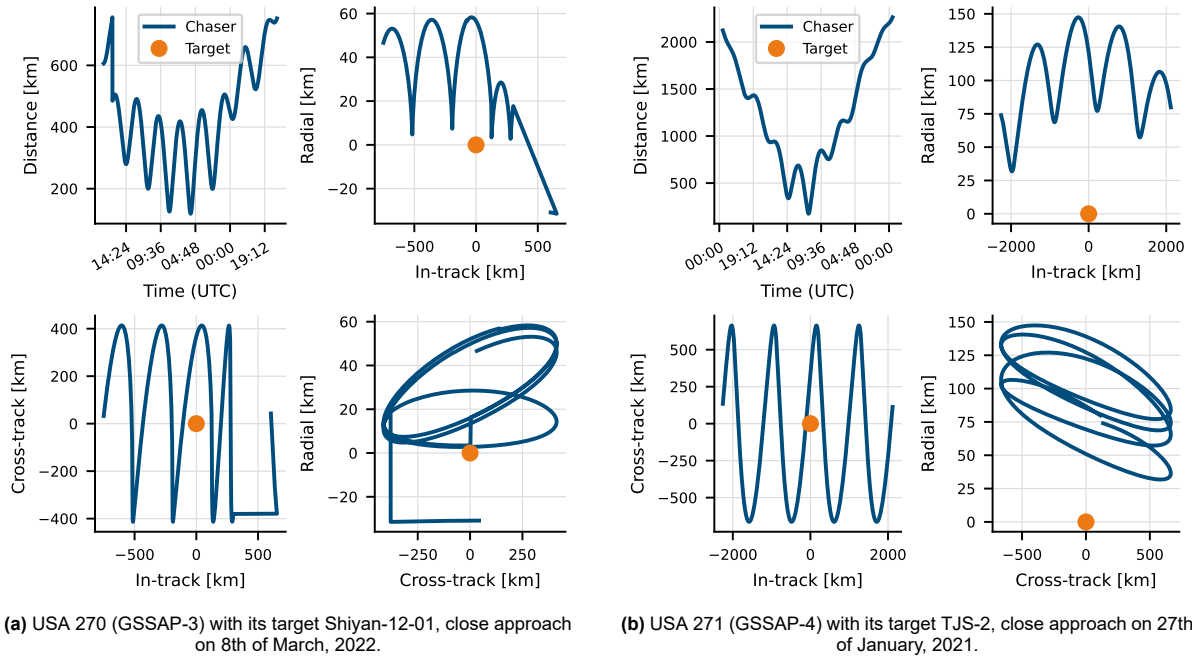


Figure 4.11: Inspection behaviour in GEO for U.S. GSSAP missions.

In Figure 4.11 it can be observed that GSSAP does not exhibit circling behaviour and distances remain in the hundreds of kilometres. In all cases, there may be a few moments during which a close approach is suitable for imaging, but less so than the circling cases in LEO. Notably, these do not match the open-source information either, where closer approaches are noted with distances around 10 km [4]. A significant part of this discrepancy is how stale the TLEs are compared to their close approaches, with typically several days between the close approach and the most recent TLE, as shown in Table 4.4, which reports the staleness for each close approach between target and chaser. For cases reported in the literature where no close approach below 500 km was found, no close approaches are recorded.

Table 4.4: Summary of TLE staleness across candidate GSSAP close-approach cases.

Case	N	Chaser staleness [h]			Target staleness [h]		
		Min	Mean	Max	Min	Mean	Max
GSSAP-3 / SY-12 01	6	2.3	432.8	1873.4	2.3	4.1	5.9
GSSAP-3 / SY-12 02	2	119.7	991.0	1862.4	2.6	4.0	5.4
GSSAP-1 / TJS-1	0	no close approach identified					
GSSAP-1 / LUCH	0	no close approach identified					
GSSAP-4 / TJS-2	4	161.8	304.2	445.2	0.1	2.4	4.7
GSSAP-4 / TJS-3	0	no close approach identified					
GSSAP-4 / TJS-15	1	365.2	365.2	365.2	3.4	3.4	3.4
GSSAP-5 / TJS-16	1	964.1	964.1	964.1	35.8	35.8	35.8
GSSAP-5 / TJS-17	1	967.1	967.1	967.1	38.7	38.7	38.7
GSSAP-4 / SKYNET 5A	1	87.1	87.1	87.1	2.4	2.4	2.4
GSSAP-5 / SYRACUSE 3A	0	no close approach identified					

4.2.4. Case Study: Docking Scenarios

Besides eavesdropping and inspection, docking is another mode of operation for non-cooperative rendezvous and proximity operations. As far as is publicly known, and as reflected in Table 4.1, no docking scenario has yet taken place in a non-cooperative manner between operators of different nations. However, demonstrators such as SJ-21 and SJ-17 highlight the potential for such scenarios to occur, through the use of robotic arms or other mechanisms capable of capturing a satellite.

Given the limited number of non-cooperative examples, cooperative docking scenarios are considered here for reference. A common setup is that of a crew or service module visiting a space station, like the International Space Station (ISS). An overview of a selection of these is presented in Table 4.5.

Table 4.5: Selected crewed docking missions to ISS and Tiangong.

Mission	Docking Date	Chaser	Target
Shenzhou-8	2011-11-02	Shenzhou-8	Tiangong-1
Shenzhou-9	2012-06-18	Shenzhou-9	Tiangong-1
Shenzhou-10	2013-06-13	Shenzhou-10	Tiangong-1
Shenzhou-11	2016-10-18	Shenzhou-11	Tiangong-2
Soyuz MS-17	2020-10-14	Soyuz MS-17	ISS
Soyuz MS-18	2021-04-09	Soyuz MS-18	ISS
Crew-2	2021-04-24	Crew Dragon Endeavour	ISS
Soyuz MS-19	2021-10-05	Soyuz MS-19	ISS
Crew-3	2021-11-11	Crew Dragon Endurance	ISS
Soyuz MS-20	2021-12-08	Soyuz MS-20	ISS
Soyuz MS-21	2022-03-18	Soyuz MS-21	ISS
Crew-4	2022-04-27	Crew Dragon Freedom	ISS
Soyuz MS-22	2022-09-21	Soyuz MS-22	ISS
Soyuz MS-23	2023-02-26	Soyuz MS-23	ISS
Soyuz MS-24	2023-09-15	Soyuz MS-24	ISS
Soyuz MS-25	2024-03-25	Soyuz MS-25	ISS
Progress MS-27	2024-06-01	Progress MS-27	ISS

The lead-up and close-in behaviour of Crew Dragon *Endurance* with the ISS are presented in Figure 4.12, showing the approach geometry (Figure 4.12a) and the close-in relative motion (Figure 4.12b).

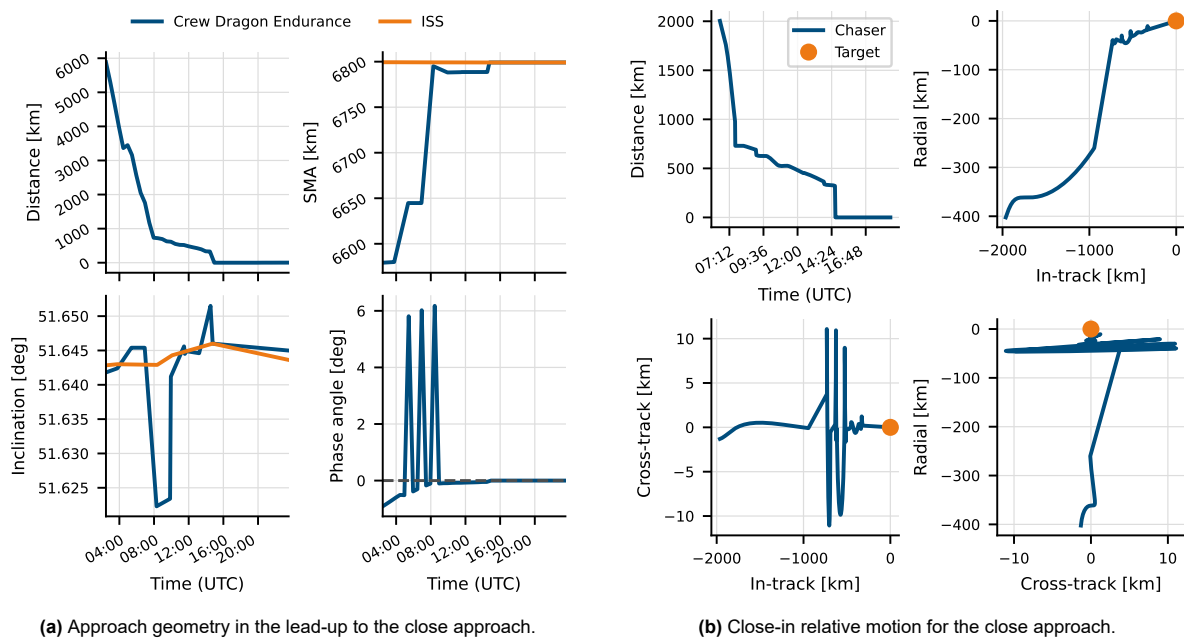


Figure 4.12: Crew Dragon *Endurance* approach to the ISS, 11 November 2021.

In the approach geometry (Figure 4.12a), the first available TLE places the Crew Dragon spacecraft at a semi-major axis (SMA) of 6550 km, where it remains for a few hours before increasing to 6650 km, while the ISS was at an SMA of about 6800 km. This difference allowed for a rapid change in the in-orbit phase angle between the two spacecraft. A further jump from 6650 km to close to 6800 km can then be observed; this smaller difference still permits some relative drift, which compensates for the remaining

difference in phase angle. After phasing for a few more hours, the SMA of Crew Dragon matches that of the ISS. Throughout this period, the inclination of the two spacecraft remained essentially identical. Overall, this behaviour took place in under 12 h.

The close-in behaviour, shown in Figure 4.12b, is difficult to observe in this case. The separation distance closes relatively quickly compared with the eavesdropping and inspection scenarios, decreasing from 2000 km to 0 km within a few hours, with no TLEs available at short range between the two spacecraft.

The approach and close-in behaviour of Shenzhou-8 approaching Tiangong-1 are shown in Figure 4.13, again split into the approach geometry (Figure 4.13a) and the close-in relative motion (Figure 4.13b).

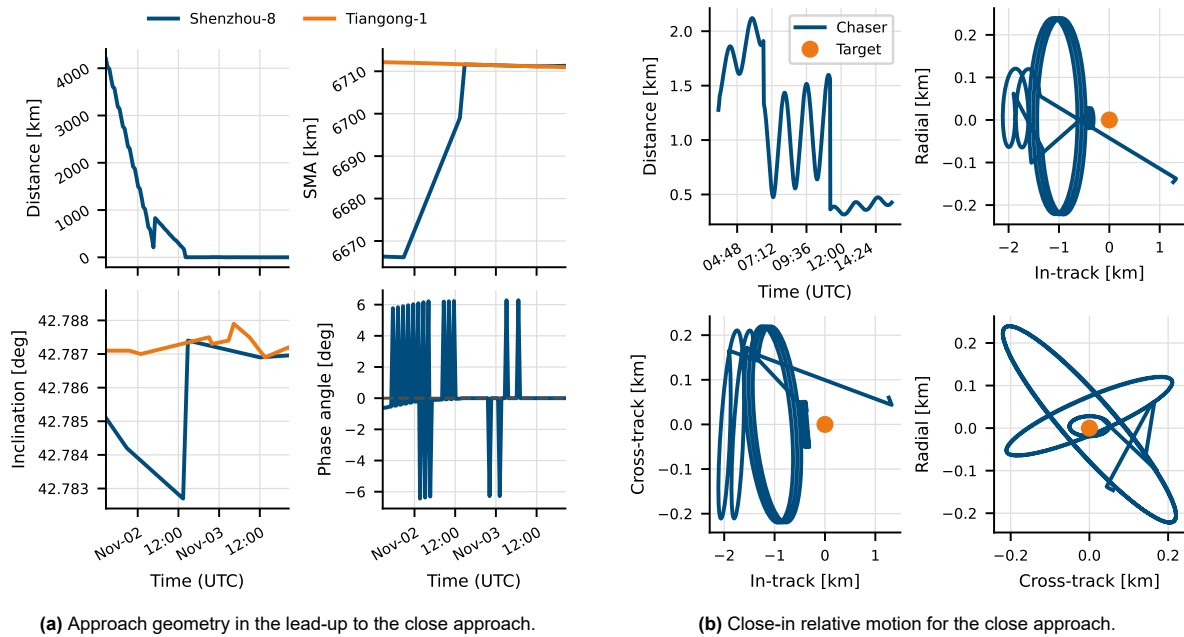


Figure 4.13: Shenzhou-8 approach to Tiangong-1, 3 November 2011.

In the approach geometry (Figure 4.13a), it can be observed that Shenzhou-8 was launched into an orbit with an SMA of 6650 km, compared with 6715 km for Tiangong-1. This difference enables a drift that decreases the phase angle between the two spacecraft. Within a few hours, the SMA becomes almost equal, resulting in a comparatively slower approach than in the Crew Dragon case. The inclination remained relatively similar over the entire period.

For the close approach, shown in Figure 4.13b, two TLEs were available before the docking TLE. In these, a stable offset in the in-track direction can be observed, with barely any oscillation in the radial or cross-track directions. This aligns with [100], which identifies a proximity hold as part of standard docking operations. Jumps between TLEs can also be observed, indicating manoeuvres.

A few points are worth noting regarding the behaviour of docking scenarios with space stations:

1. Often only a few TLEs exist for the entire RPO up to docking; 11 out of 18 cases have fewer than four TLEs before docking.
2. Satellites are launched into a co-orbital plane with their target space station and manoeuvre towards it over a time frame ranging from a few days to less than a day.
3. The launch brings the satellite to an SMA that can be a few hundred kilometres below that of its target, a difference that is then compensated for by the chaser vehicle.
4. Close-in behaviour, including the proximity hold, cannot always be observed due to limited TLE availability.

Other examples of docking scenarios do not include a space station, but are instead satellite-to-satellite. These include servicing missions like the Mission Extension Vehicles (MEVs) to the Intelsat geostationary satellites, non-cooperative demonstrators with a defunct satellite, or suspected dockings. An overview of these is included in Table 4.6.

Table 4.6: Satellite-to-satellite docking missions.

Mission	Docking Date	Chaser	Target
Orbital Express	2007-06-29	ASTRO	NEXTSat
MEV-1 $\times$ Intelsat 901	2020-02-25	MEV-1	Intelsat 901
MEV-2 $\times$ Intelsat 10-02	2021-04-12	MEV-2	Intelsat 10-02
SJ-21 $\times$ Compass-G2	2022-01-21	SJ-21	Compass-G2 (Beidou-2 G2)
SJ-21 $\times$ SJ-25	2025-07-02	SJ-21	SJ-25

The MEVs have been used to prolong the life of satellites in geostationary orbit. Two of them have been deployed, MEV-1 and MEV-2, servicing Intelsat 901 and Intelsat 10-02 respectively. It is worth noting that these missions occupy a middle ground between cooperative and non-cooperative RPOs: they were cooperative in the sense that the target operator had agreed to the servicing, but the target satellites themselves were not originally designed to be serviced and lacked any dedicated docking interface. As a result, the rendezvous and capture sequence relied on techniques closer to those used in non-cooperative dockings, namely grappling the apogee kick motor nozzle, rather than mating with a purpose-built docking port. However, the target did not take evasive action during this period.

The approach and close-in trajectories can be found in Figure 4.14. Reportedly, MEV-1 docked on February 25, 2020 at 7:15 UTC.

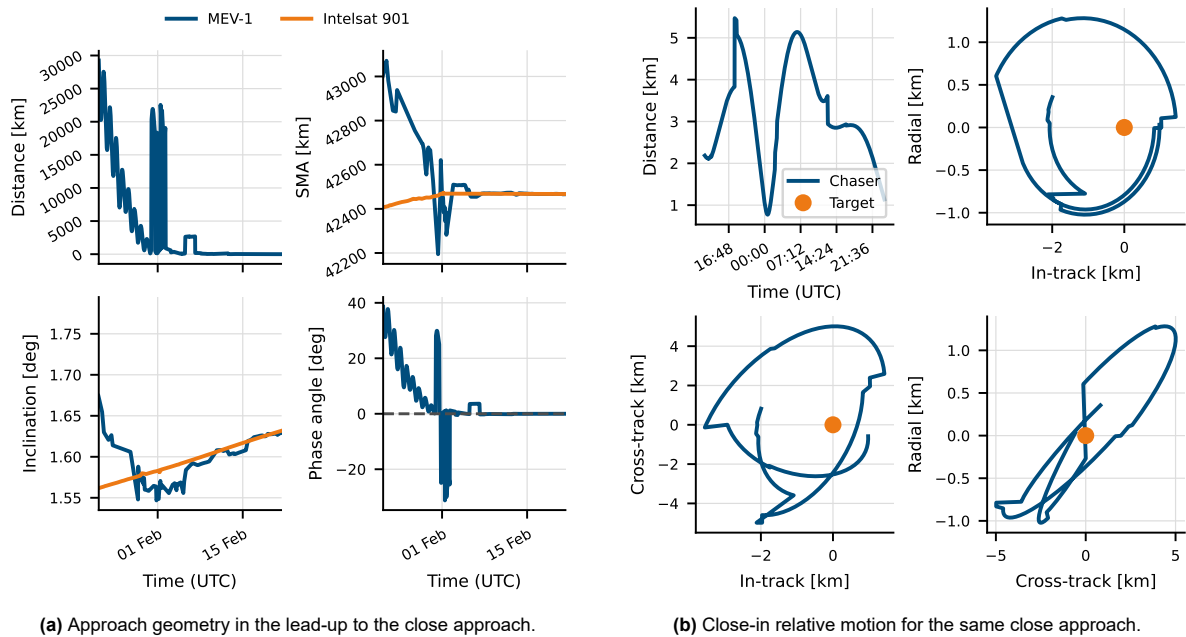


Figure 4.14: MEV-1 approach and docking with Intelsat 901, 25 February 2020.

As can be observed, MEV-1 spent time above the GEO belt for the approach with Intelsat 901, which reduced the phase angle. Around 1 February, TLE accuracy seems to cause issues with the relative distance, jumping both ahead of the target and then behind the target with a phase angle of 25° , or some 23 000 km, before both being reduced close to zero. A week later, a smaller jump occurs. These jumps indicate errors in the TLEs. This is further exemplified by the close-in behaviour. Whilst the reported docking date was the 25th of February, the closest approach found through the TLEs in this time frame is approximately 1 km, after which it starts to increase again. No proximity hold was observed.

Then there is the aforementioned SJ-21 non-cooperative docking with Compass-G2 through the use of a robotic arm. The approach and close-in trajectories can be found in Figure 4.15. SJ-21 is reported to have docked with Compass-G2 on the 22nd of January 2022, during a gap in commercial telescope coverage; the exact time of docking has not been publicly disclosed [4].

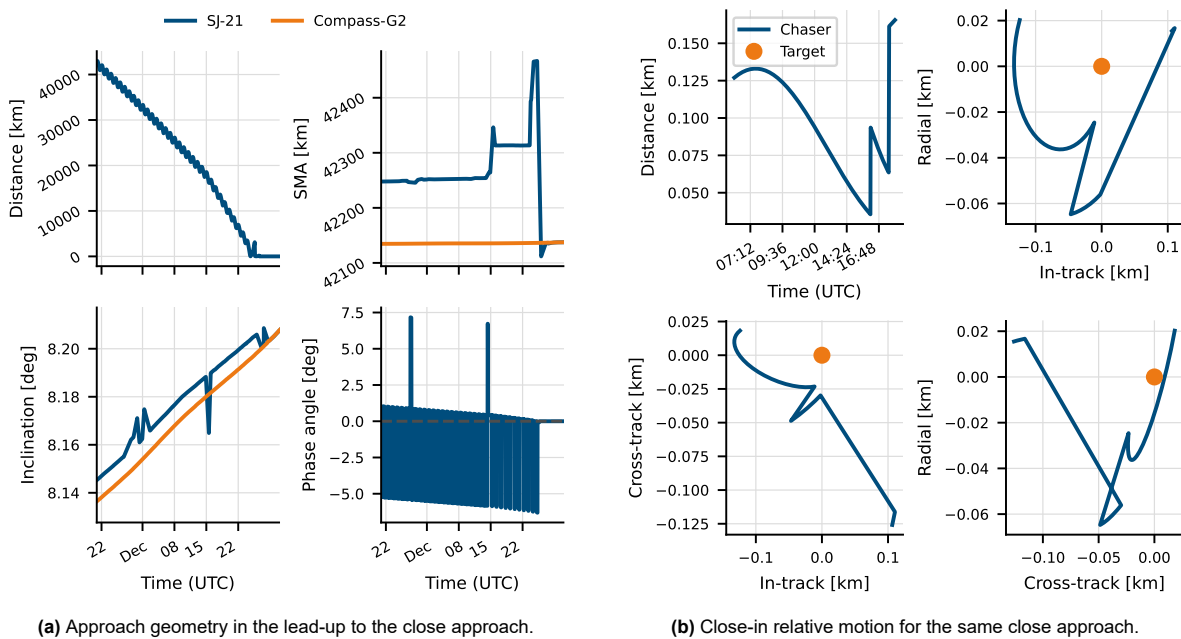


Figure 4.15: SJ-21 approach and docking with Compass-G2, late December 2021 – 22 January 2022.

SJ-21 approached Compass-G2 over a period of more than a month. It spent its initial drift at an SMA of 42 300 km, before increasing it to 42 480 km. It spent less than a week there, before reducing the deviation from the GEO belt over the period of a month. During this approach, SJ-21 overshot its target, requiring it to move below the GEO belt once more to reduce the remaining distance. It reached within a few kilometres of the GEO belt a few weeks before docking, and slowly drifted towards its target. It is hard to determine the actual docking date from the available TLE data alone; the reported date of 22 January 2022 is used here.

For these docking scenarios in GEO, it can be concluded that the approach can take on the order of months. Unlike LUCH or the LEO docking scenarios, for which the arrival back at the GEO SMA marked the start of the intent, here the arrival at GEO SMA, or close to it, marks the start of the slower drift. Why this slower drift occurs is unclear, but a possible explanation lies in the nature of the missions themselves: all of the GEO dockings analysed here were either technology demonstrators for a state actor or the first instance of this kind of manoeuvre performed by a commercial company. Most likely, an operator that wants to perform a non-cooperative docking manoeuvre has some incentive to complete it in a timely manner, so as not to give the operator of the target time to perform evasive burns.

4.2.5. General Trends

Overall, close-in behaviour can be broadly grouped by the task the satellite performs. Those that dock with a satellite or space station tend to come close to their target, those that listen in on communications stay near the target, whilst inspections perform fly-bys that tend to circle the target. However, these analyses are complicated by data quality. Exact docking times, for example, cannot be reliably determined from TLEs, in contrast to the docking dates that are typically published. Furthermore, as GSSAP shows, old TLEs can complicate matters further.

The approach phase is harder to quantify. The only grouping that shows a clear pattern is that of the two LUCH satellites. The analysis for these satellites is made easier by the fact that they perform all of their approaches in the GEO belt, meaning only phasing is required to move from a position to a target. This phasing exhibits a correlation between how fast they approach a target and how far away that

target is, as shown in Figure 4.16a. The drift rates also fall within some general bounds, as indicated in Figure 4.16b.

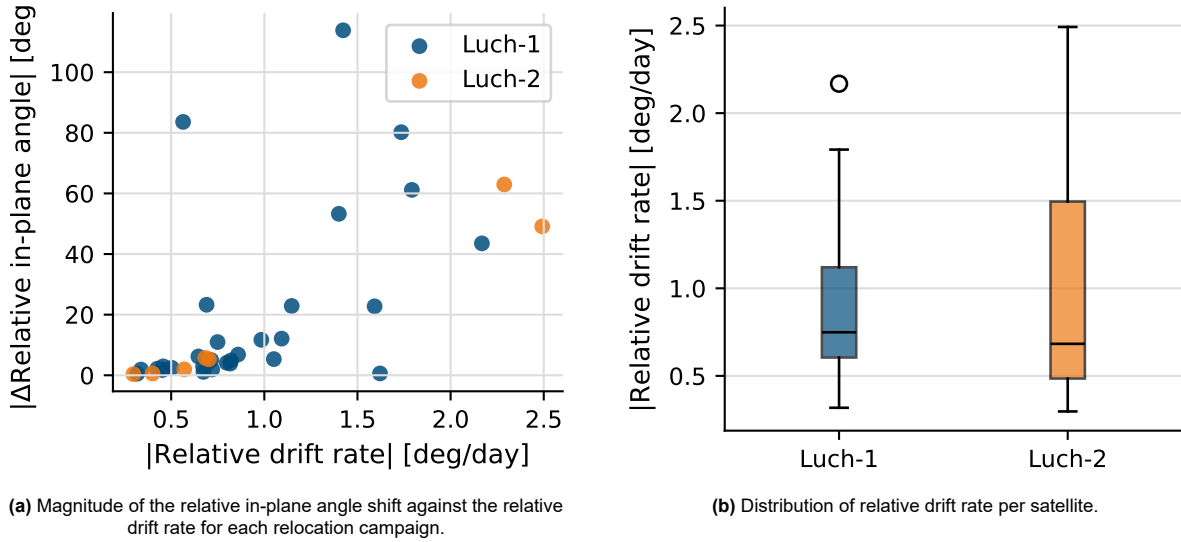


Figure 4.16: LUCH-1 and LUCH-2 relocation-campaign speed overview.

Similar groupings for other satellites are complicated by multiple factors. In GEO, the GSSAP satellites show no clear pattern, in part due to TLE staleness. In LEO, this entire analysis is complicated by the orbital domain itself; satellites are not in the same orbital plane and not only perform phasing manoeuvres, but also plane changes. Determining a relationship between starting conditions and time spent relocating thus cannot rely only on in-plane differences. Furthermore, determining when a relocation actually starts is complicated for satellites that have been in orbit for longer periods, as they might be performing station-keeping burns. For satellites that are newly launched, there is also the consideration of commissioning orbit, which can differ from the operational orbit of satellites already established there.

4.3. Intent Taxonomy and Justification

The concept of intent in the context of non-cooperative RPOs is not consistently defined in the literature. As discussed in Chapter 2, existing approaches to intent classification can generally be grouped into three categories: *Threat Index*, *Motion Class*, and *Mission Class*. These categories differ in both the information they convey and the complexity of the corresponding classification problem.

Threat indexes provide an indication of the likelihood that an RPO is going to occur. Their primary advantage lies in their simplicity, being a scalar index crafted through derived features. However, they provide little information regarding the purpose of the RPO, answering only whether an RPO is likely to occur or not.

Motion classes describe the relative trajectories between the target and the chaser spacecraft. By characterising the observed relative motion, they provide greater insight into how an RPO is executed. Nevertheless, the relationship between a particular motion pattern and the underlying operational objective is often ambiguous. The mapping between motion and intent is not one-to-one: a single motion class may be observed across missions with different intents, whilst a single intent may manifest through different motion classes. Consequently, motion alone is generally insufficient to uniquely determine intent.

Mission classes instead categorise an RPO according to its operational objective. Rather than describing whether an RPO occurs or how it is executed, mission classes seek to identify the purpose of the operation. As such, they provide the most informative representation of spacecraft intent. This increased expressiveness, however, comes at the cost of a more challenging classification problem,

since different mission objectives may exhibit similar observable behaviour. A summary of the differences between the three different approaches to intent definition is presented in Table 4.7,

Table 4.7: Comparison of intent taxonomy categories used in the literature.

Property	Threat Index	Motion Class	Mission Class
Output	Scalar likelihood	Relative trajectory classification	Operational objective classification
Basis	Derived features	Observed relative motion	Purpose of the operation
Information conveyed	Whether an RPO is likely to occur	How an RPO is executed	Why an RPO is being conducted
Link to underlying intent	None	Ambiguous	Direct
Classification complexity	Low	Moderate	High

For the purposes of this research, intent is inferred at the level of *Mission Classes*. Since the objective is to determine the purpose of a non-cooperative RPO, mission-level intent provides the most operationally meaningful information.

Although Mission Classes are covered by multiple papers in the literature, no universally accepted taxonomy currently exists. Different studies adopt differing sets of Mission Classes, often mixing operational objectives with specific trajectory geometries or spacecraft activities. For example, one taxonomy defines the classes *Rendezvous*, *Collision*, *Inspection*, *Departure*, and *V-Bar Perch* [84]. While the first four correspond to identifiable operational objectives, *V-Bar Perch* instead describes a particular relative configuration and therefore represents a trajectory rather than a mission. Similarly, another taxonomy defines the classes *Coverage*, *Docking*, *Signalling*, *Flyby*, and *Rendezvous* [79]. Although each represents a meaningful activity, the justification as to why a certain behaviour is consistent for a given intent is lacking.

A further limitation of existing intent taxonomies is that they are frequently only loosely connected to observed on-orbit behaviour. This is largely a consequence of the scarcity of publicly available data describing non-cooperative RPOs. As discussed in Chapter 4, publicly accessible catalogues such as Space-Track typically lack the temporal resolution required for accurate manoeuvre reconstruction, provide insufficient positional accuracy for analysing close-proximity dynamics, and do not include observations involving classified spacecraft. Consequently, many proposed intent definitions are necessarily derived from theoretical considerations rather than empirical evidence.

Nevertheless, historical non-cooperative RPOs demonstrate that a number of recurring mission objectives have been executed in orbit. As summarised in Table 4.1, these operations can broadly be grouped into several categories. Signal intelligence satellites perform prolonged proximity operations, predominantly in geostationary orbit, to intercept or monitor communications. Reconnaissance satellites conduct close inspection of resident space objects to obtain information regarding their physical characteristics, operational status, or capabilities. Counter-space systems exhibit two distinct behaviours: some demonstrate stalking or shadowing manoeuvres indicative of potential offensive capabilities, while others have intentionally collided with target spacecraft during ASAT demonstrations. Finally, several technology demonstrators have performed advanced proximity operations involving rendezvous, robotic capture, or docking. Although these missions are generally presented as technology demonstrations, the precise operational objectives are not always publicly disclosed.

This work will consider the commonly observed behaviours in orbit. Four mission classes are considered:

- **Inspection**, representing reconnaissance missions that seek to maximise observation of a target spacecraft.
- **Eavesdropping**, representing signal intelligence missions that maintain a favourable geometry for intercepting communications.

- **Docking**, representing servicing or capture missions that seek to establish physical contact with the target.
- **Collision**, representing counter-space missions whose objective is to destructively impact the target spacecraft.

In addition to these four mission classes, a *No RPO* class is included. This represents trajectories for which the chaser is not conducting proximity operations with respect to the target under consideration, for example when transiting towards another spacecraft. Including this class allows the intent inference framework to function as a binary threat index and prevents every observed trajectory from being assigned to one of the operational mission classes.

Table 4.8: Overview of the intent classes considered in this work.

Name	Description	LEO Example	GEO Example
Inspection	A reconnaissance mission that seeks to maximise observation of a target spacecraft from multiple viewing directions.	Victus Haze with Jackal-0004	GSSAP
Eavesdropping	A signal intelligence mission that maintains a favourable relative geometry to intercept the target's communications.	–	LUCH OLYMP K-1 and -2
Docking	A servicing or capture mission that seeks to establish physical contact with the target.	Docking with ISS	SJ-21 with Compass G2
Collision	A counter-space mission whose objective is to physically impact the target spacecraft.	Soviet IS Programme	–
No RPO	A trajectory in which the chaser is not conducting proximity operations with respect to the target under consideration.	NA	NA

"Dwell on the beauty of life. Watch the stars, and see yourself running with them."

Marcus Aurelius, *Meditations* - 2nd century AD

5

RPO Dataset Generation

This chapter details the data generation process of non-cooperative RPOs. Section 5.1 provides the necessary background and an overview of the modelling approach. Section 5.2 then explains the satellite population database used to define the trajectories' starting conditions. This is followed by the two optimisation stages: Section 5.3 covers the optimisation of the mission trajectories, and Section 5.4 covers the approach trajectories. Assessing real-world performance requires real-world data; the process of acquiring this is detailed in Section 5.5. Finally, Section 5.6 explores the match between the real-world and synthetic data.

5.1. Data Generation Background and Overview

Training and testing of the intent classification model requires representative data for each intent class. In machine learning, the requisite dataset size can range from thousands to millions of data-points, depending on the model architecture and the complexity of the classification problem [65]. For the models employed in Chapter 6, the estimated requirement is on the order of tens of thousands of data-points.

The use of synthetic rather than real-world data is necessitated by both the insufficient quantity and potential inadequacy of available observational records. As documented in [3, 4], the number of confirmed non-cooperative RPO events numbers in the dozens, several orders of magnitude below the threshold required for training. Moreover, even if a sufficient number of events existed, the associated data may not meet the requirements of the training process. As discussed in Chapter 4, such events can unfold faster than publicly available TLE update cycles, meaning that the underlying manoeuvres cannot be reliably reconstructed. In addition, operational data may be entirely inaccessible, particularly in cases involving classified satellites acting as either target or chaser.

These constraints necessitate training on simulated data, a choice consistent with the majority of studies surveyed in Table 2.1. However, as noted in Chapter 2, the existing literature does not address early-stage intent classification of non-cooperative RPOs, a limitation that is itself partly attributable to the nature of the data used in these studies. Prior work has predominantly focused on intent detection at close range, typically within tens of kilometres, and concerns itself with the trajectory following the final approach manoeuvre. With the exception of [75], the effect of preceding manoeuvres is largely disregarded. Phasing burns and orbital plane changes required to reach the vicinity of the target are not considered, despite the possibility that the approach trajectory itself carries discriminative information relevant to intent.

To enable early-stage detection of non-cooperative RPOs, the data generation framework is structured around two distinct phases: the *Mission Phase* and the *Orbital Adjustment Phase*. The Mission Phase

corresponds to the regime addressed in most of the existing literature, covering the period during which the chaser operates in proximity to the target and is actively executing its intended behaviour. The Orbital Adjustment Phase addresses the preceding segment, during which the chaser transitions from its initial orbit to the starting conditions of the Mission Phase; this phase is introduced to facilitate earlier inference of intent.

Each phase is driven by an optimisation of the chaser's behaviour for the intent. Not every intent requires both phases: for *Docking* and *Collision*, the target condition is a static terminal condition, fixed for all optimisations of the approach, and no Mission Phase optimisation is performed. For *Inspection* and *Eavesdropping*, by contrast, the Mission Phase optimiser minimises a multi-objective function defined with respect to the chaser's intent, yielding a mission trajectory that is optimal for the execution of that intent; the initial state of this trajectory is then adopted as the target condition. In both cases, whether a static terminal condition or the initial state of an optimised mission trajectory, this target condition is passed to the Orbital Adjustment Phase optimiser, which determines the optimal transfer to it by minimising a transfer objective that can differ per intent.

An overview of the terminal conditions at the end time (t_f) and objective functions (J) per intent is presented in Table 5.1. The definitions of the mission objectives and approach objectives are further elaborated on in Section 5.3 and Section 5.4, as well as the implementation of the objectives themselves.

Table 5.1: Overview of the target condition type, mission objective, and orbital adjustment objective defined for each intent.

Intent	Type	Target Condition	Mission Objective	Orbital Adjustment Objective
Docking	Static terminal	Zero relative position and velocity at t_f	–	$\min J_{\Delta V}$
Collision	Static terminal	Zero relative position at t_f ; relative velocity unconstrained	–	$\min(J_{\text{tof}} + J_{\Delta V})$
Inspection	Optimised trajectory	Initial state of the mission trajectory enabling multi-direction observation in close proximity	$\min\{J_{\text{dist}}, J_{\text{cov}}\}$	$\min J_{\Delta V}$
Eavesdropping	Optimised trajectory	Initial state of the stable mission trajectory within the target's communications cone	$\min\{J_{\text{stab}}, J_{\text{beam}}\}$	$\min J_{\Delta V}$

5.2. Satellite Population and Initialisation

Each optimisation phase requires as input satellite starting state(s). These starting states are not generated randomly, but instead based upon the current satellite population in Earth orbit, constructed as a database from publicly available sources. Space-Track was queried for the most recent TLEs of satellites, collected per region. Space-Track categorises each object into four categories: *Payloads* “P”, *Debris* “D”, *Rocket Body* “R”, and *Unknown* “U”; for the purposes of this research, only payloads were queried. For the relevant regions, the following filters were applied:

$$\begin{aligned} \text{LEO: } & 6678.1 \leq a \leq 8378.1[km], \\ & 0 \leq e \leq 0.25, \end{aligned}$$

$$\begin{aligned} \text{GEO: } & 41964 \leq a \leq 42364[-], \\ & 0 \leq e \leq 0.25, \end{aligned}$$

The retrieved TLEs were propagated to the then-current epoch using SGP4, via the SGP4 Python library [101], itself based on [102]. SGP4 is used as the propagator since the database contains the satellite states as TLEs; it comprises both SGP4 and SDP4 sub-models, with one selected internally depending

on the orbital period. It accounts for secular effects due to Earth's zonal gravitational harmonics and atmospheric drag, with SDP4 additionally including third-body perturbations from the Sun and Moon, and resonance effects for 12-hour and 24-hour orbits [102]. The propagator is initialised with the most recent TLE from each object, which is propagated to 26-06-2026 at 12:00:00 UTC. At this state, the osculating Keplerian elements of each object are obtained and serve as input for later propagators. For satellites in the GEO belt, the instantaneous longitude was additionally added.

To add the instantaneous longitude (λ) for satellites in the GEO database, the instantaneous longitude is calculated by Equation 5.1. Here, θ_G is the Greenwich meridian's angle with the Vernal Equinox.

$$\lambda = (\Omega + \omega + \nu - \theta_G) \bmod 360^\circ \quad (5.1)$$

To limit the angle to $\pm 180^\circ$ from the Greenwich meridian, Equation 5.2 is used. A positive angle would be east, and a negative one west, of the Greenwich meridian.

$$\lambda' = \begin{cases} \lambda - 360^\circ, & \lambda > 180^\circ, \\ \lambda, & \lambda \leq 180^\circ. \end{cases} \quad (5.2)$$

The database is queried using a seed, and any other potential filters. The seed determines the random number generation used in the selection of entries in the database, which ensures that for a given combination of seed and filters, the same satellite states are yielded.

Two relevant filters exist: a reachability metric V_{rm} for all orbital domains, and in GEO the addition of instantaneous longitude separation. The reachability metric is determined using Equation 5.3.

$$\Delta V_{rm} = \frac{||\vec{h}_1 - \vec{h}_2||}{\frac{1}{2}(a_1 + a_2)} \quad (5.3)$$

Equation 5.3 is a metric of the match between the orbital planes. Whilst its units are the same as a velocity, it is not a true ΔV required to change orbits. However, it is useful for filtering the database to eliminate chaser and target pairs with greater orbital differences, which is something not observed for the missions in Chapter 4 and does not occur in general according to Samson and Brett [4]. The instantaneous longitude filter is exclusively used in GEO, and based on the movements observed by both Luch satellites from Chapter 4. This filter limits the phasing that the satellites in GEO can perform. For the reachability metric, 20 m s^{-1} is used, whilst the maximum longitude difference is set to 60° .

Although these filters serve to match initial conditions from current non-cooperative satellite RPOs, in future scenarios, where potentially more propellant is available and greater orbital changes are possible, they might no longer hold and may need to be loosened.

The overall process to query the database is thus as follows. A domain, e.g. LEO, and seed (e.g. nr. 1) are specified. The seed and orbital domain give a fixed chaser satellite. For the chaser satellite, the reachability metric is calculated for every other satellite in the database of the domain. This database is then filtered with the standard specified max V_{rm} for the domain. If the domain is GEO, then the maximum instantaneous longitude separation is additionally used to filter the database. From the filtered database, a random satellite and its states are selected as target. In addition to this process, if another unique satellite for the given seed and domain is required, this is also possible. In this case, the target satellite is removed from the database, and another satellite is queried, if the database has non-zero entries remaining. This replacement is used for the *No RPO* scenario, to replace the original target.

5.3. Mission Phase Optimisation and Terminal Conditions

As indicated in Table 5.1, *Docking* and *Collision* are governed by a static terminal condition rather than a Mission Phase optimisation, whilst *Inspection* and *Eavesdropping* require an optimised mission trajectory. The terminal conditions for the former are given in Subsection 5.3.1, and the Mission Phase optimisation for the latter is described in Subsection 5.3.2 onwards.

5.3.1. Terminal Conditions

Docking

The terminal objective is a relative state between the target and chaser, requiring both the relative position and relative velocity to be zero at the terminal time. This represents a successful physical rendezvous suitable for docking or servicing operations.

The terminal conditions are defined as

$$\mathbf{r}_{\text{rel}}(t_f) = \mathbf{r}_c(t_f) - \mathbf{r}_t(t_f) = \mathbf{0}, \quad (5.4)$$

and

$$\mathbf{v}_{\text{rel}}(t_f) = \mathbf{v}_c(t_f) - \mathbf{v}_t(t_f) = \mathbf{0}, \quad (5.5)$$

where t_f is the terminal (docking) time, $\mathbf{r}_c$ and $\mathbf{r}_t$ are the chaser and target position vectors, respectively, and $\mathbf{v}_c$ and $\mathbf{v}_t$ are the corresponding velocity vectors.

Collision

The terminal objective is a relative state, defined as an intercept with the target, such that the relative position converges to zero at the time of impact. Unlike docking, matching the target's velocity is not required, allowing for a non-zero relative velocity at the terminal state.

The terminal condition is defined as

$$\mathbf{r}_{\text{rel}}(t_f) = \mathbf{r}_c(t_f) - \mathbf{r}_t(t_f) = \mathbf{0}, \quad (5.6)$$

where t_f is the terminal (impact) time, $\mathbf{r}_c$ is the chaser position vector, and $\mathbf{r}_t$ is the target position vector.

5.3.2. Mission Phase Optimisation

Inspection and *Eavesdropping* require a mission trajectory rather than a static terminal condition. These mission trajectories are determined for each seed and domain. Each case starts with the selection of the target satellite and a given duration to execute the mission. An optimiser then determines what the best relative position and velocity are at the start of the mission. This optimum is determined by the evaluation of the objective functions, which differ per intent, and results in trajectories of the chaser relative to the target.

For the satellite initialisation, the approach from Section 5.2 is used, taking only the target satellite. The chaser's initial state is then defined relative to the Target through six Keplerian offsets (Δa , Δe , Δi , $\Delta \omega$, $\Delta \Omega$, $\Delta \nu$), which form the design vector of the optimisation problem. These offsets are added element-wise to the Target's Keplerian elements to construct the Chaser's own Keplerian state. Propagation is done through Tudatpy [103], with a point mass gravity model and no other perturbations. Integration is done using the adaptive Runge-Kutta(RK)78 propagator, with a Hermite-spline interpolator to evaluate all optimisations on a fixed 10 000 uniform step size grid for the objective functions. Additional parameters saved depend on the intent simulated: for *Inspection*, the Sun position is saved for eclipse and lighting condition calculations. The RK78 propagator with interpolation was found to be faster than evaluating multiple `propagate_kepler_orbit` calls.

Mission Phase Objectives

Each mission type is formulated as a multi-objective optimisation problem, with the objectives evaluated from parameters of the propagated trajectory. Candidate solutions that fail to propagate until the prescribed mission end time are assigned a large penalty value of 10^9 for both objectives.

Inspection The objective of an inspection mission is to obtain the most complete characterisation of the target spacecraft possible. Rather than converging to a single terminal state, the mission is defined by a terminal trajectory that enables observation from multiple viewing directions while remaining in close proximity to the target. The trajectory optimisation therefore considers two objectives. The first minimises the mean relative distance between the chaser and target throughout the inspection, ensuring that the target remains within an effective observation range. The second maximises the angular coverage of the target by encouraging observations from as many distinct viewing directions as

possible. Coverage is quantified by discretising the viewing sphere into azimuth–elevation bins and maximising the fraction of unique bins visited during the trajectory. To ensure physically meaningful observations, only viewing geometries with acceptable solar phase angles are considered, excluding configurations where the target is either fully eclipsed or viewed almost directly into the Sun.

The inspection mission intent is formulated as the following multi-objective optimisation problem,

$$\min_{\mathbf{u}(t)} \{J_{\text{dist}}, J_{\text{cov}}\}, \quad (5.7)$$

where J_{dist} minimises the mean relative distance between the chaser and target, and J_{cov} maximises the angular coverage of the target. Both objectives are evaluated directly from the propagated trajectory by post-processing the relative geometry between the Chaser, the Target, and the Sun.

1. Mean relative distance.

J_{dist} is computed as the mean of the relative separation over all propagated samples,

$$J_{\text{dist}} = \frac{1}{N} \sum_{k=1}^N \|\mathbf{r}_{\text{rel},k}\|, \quad (5.8)$$

where $\mathbf{r}_{\text{rel},k}$ denotes the Chaser–Target relative position vector at the k th sample. Since the optimisation minimises this quantity directly, trajectories maintaining a smaller average separation are preferred.

2. Angular coverage.

The second objective evaluates the diversity of viewing directions from which the Target is observed. A given sample only counts towards coverage if it represents a valid observation, meaning it satisfies both an illumination and an eclipse condition. This objective balances with the distance, since it promotes diverse viewing angles, which in part is constrained by the relative drift rate and time-of-flight (tof).

For the illumination condition, the relative position vector and the Sun–Target vector are normalised,

$$\hat{\mathbf{r}} = \frac{\mathbf{r}_{\text{rel}}}{\|\mathbf{r}_{\text{rel}}\|}, \quad \hat{\mathbf{s}} = \frac{\mathbf{r}_{\text{Sun}}}{\|\mathbf{r}_{\text{Sun}}\|}, \quad (5.9)$$

and the solar phase angle is computed from their dot product,

$$\theta_{\text{phase}} = \cos^{-1}(\hat{\mathbf{r}} \cdot \hat{\mathbf{s}}). \quad (5.10)$$

A sample is only retained for coverage evaluation if

$$\theta_{\min} < \theta_{\text{phase}} < \theta_{\max}. \quad (5.11)$$

The lower bound, $\theta_{\min}$, prevents the target from being observed under shadowed conditions, where insufficient illumination degrades visibility, as depicted in Figure 5.1. The upper bound, $\theta_{\max}$, avoids dazzling caused by (near) backlighting from the Sun, which can saturate the imaging sensor, as depicted in Figure 5.2. Acceptable observation conditions correspond to intermediate phase angles satisfying Equation 5.11, as depicted in Figure 5.3.

Beyond the phase-angle window, an eclipse condition on the target itself must also be satisfied, since the target may be shadowed by the Earth regardless of the chaser's viewing geometry. This is evaluated through Tudatpy's built-in shadow-function with a cannonball radiation-pressure model for the target, with Earth being the occulting body for the Sun. Only samples for which the target is illuminated are retained for the coverage analysis.

For each sample satisfying both the phase-angle and eclipse conditions, the viewing direction is converted into spherical coordinates,

$$\phi = \tan^{-1}\left(\frac{y}{x}\right), \quad (5.12)$$

$$\lambda = \sin^{-1}(z), \quad (5.13)$$

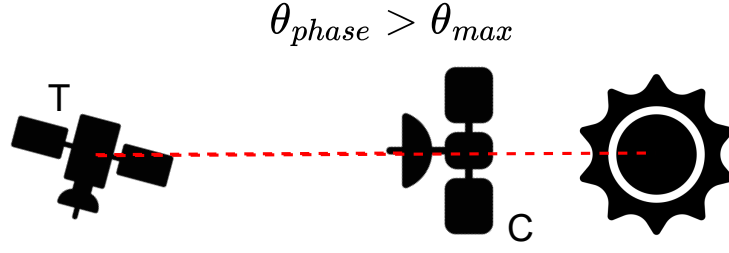


Figure 5.1: Target illumination for $\theta_{phase} < \theta_{min}$, where the target is insufficiently illuminated due to shadowing.

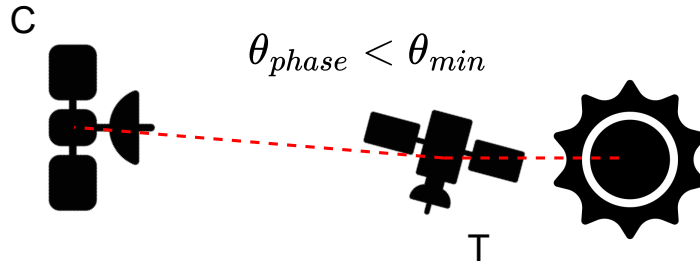


Figure 5.2: Target illumination for $\theta_{phase} > \theta_{max}$, where dazling degrades image quality.

where (x, y, z) are the components of the unit relative-position vector $\hat{\mathbf{r}}$. The azimuth and elevation are discretised into a uniform 48×24 grid,

$$i_\phi = \left\lfloor \frac{\phi + \pi}{2\pi} N_\phi \right\rfloor, \quad (5.14)$$

$$i_\lambda = \left\lfloor \frac{\lambda + \pi/2}{\pi} N_\lambda \right\rfloor, \quad (5.15)$$

with $N_\phi = 48$ and $N_\lambda = 24$. The number of unique bins visited is accumulated over the complete trajectory, and the objective is defined as

$$J_{cov} = -\frac{N_{visited}}{N_\phi N_\lambda}. \quad (5.16)$$

If no propagated samples satisfy both the illumination and eclipse conditions, the objective returns a penalty value of 10^6 .

Eavesdropping The objective of an eavesdropping mission is to maximise the opportunity for intercepting the target spacecraft's communications by maintaining a favourable relative geometry over an extended period. Rather than converging to a single terminal state, the mission is defined by a stable relative trajectory within the target's communications cone from a ground station. Two optimisation objectives are therefore considered. The first promotes trajectory stability. This encourages the chaser to maintain a nearly constant relative motion, reducing station-keeping requirements. The second objective maximises the time spent within the target's communications cone. This is achieved by maximising the fraction of the trajectory for which the chaser remains inside the specified antenna beam while simultaneously minimising the average off-axis angle (θ_{eaves}) within the beam. Consequently, trajectories that remain close to the beam centre for extended durations are preferred.

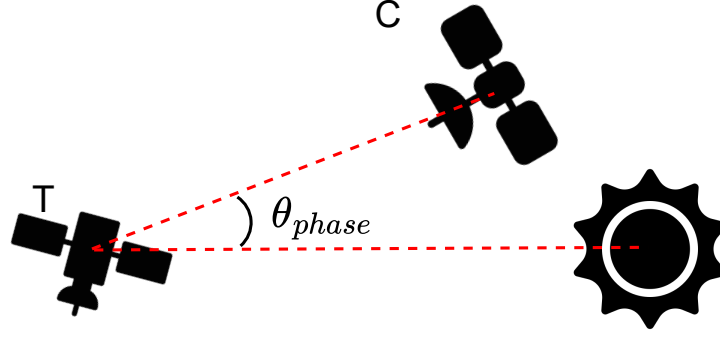


Figure 5.3: Target illumination for $\theta_{\min} < \theta_{\text{phase}} < \theta_{\max}$, providing suitable observation conditions.

The eavesdropping mission intent is similarly formulated as a multi-objective optimisation problem,

$$\min_{\mathbf{u}(t)} \{J_{\text{stab}}, J_{\text{beam}}\}. \quad (5.17)$$

where J_{stab} promotes a stable relative trajectory, and J_{beam} maximises the quality of the communications geometry. Both objectives are evaluated from the propagated Target and Chaser state histories.

1. Relative velocity stability.

J_{stab} minimises deviations from constant relative velocity in the LVLH frame, for the components in the R, S, and W-directions. The inertial relative velocity is first transformed into the Target-centred LVLH frame, with orthonormal basis vectors constructed as

$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{\|\mathbf{r}\|}, \quad (5.18)$$

$$\hat{\mathbf{w}} = \frac{\mathbf{r} \times \mathbf{v}}{\|\mathbf{r} \times \mathbf{v}\|}, \quad (5.19)$$

$$\hat{\mathbf{s}} = \hat{\mathbf{w}} \times \hat{\mathbf{r}}, \quad (5.20)$$

where $\mathbf{r}$ and $\mathbf{v}$ are the Target position and velocity vectors. The Chaser relative velocity is then projected onto this basis,

$$\mathbf{v}_{\text{LVLH}} = \begin{bmatrix} \hat{\mathbf{r}}^T \\ \hat{\mathbf{s}}^T \\ \hat{\mathbf{w}}^T \end{bmatrix} \Delta \mathbf{v}. \quad (5.21)$$

Each component is approximated by a least-squares linear fit,

$$v_i(t) = a_i t + b_i, \quad i \in \{\text{R, S, W}\}, \quad (5.22)$$

where a_i and b_i are obtained using first-order polynomial regression over the propagated samples. The objective is

$$J_{\text{stab}} = \sum_{i=1}^3 (w_s |a_i| + |b_i|), \quad w_s = 10^8, \quad (5.23)$$

which strongly penalises drift while also discouraging persistent velocity offsets. The weighting factor w_s is required to encourage the minimisation of $|a_i|$, which has much smaller magnitudes than $|b_i|$.

Two penalty functions are also defined, relating to the observed in-track offset from Chapter 4: an in-track offset requirement penalty and an in-track crossover penalty. The first encourages the

in-track offset to track d_{nominal} by penalising the RMS relative deviation of the absolute offset from this value, whilst the latter discourages sign changes in the in-track offset by penalising based upon the fraction of samples with an opposite sign to the majority of samples' sign of in-track separation. The d_{nominal} value is fixed per seed and determined by the random sample of the cone size per Subsection 5.3.3.

2. Communications beam interception.

J_{beam} favours trajectories that remain within the target's communications cone for extended periods while staying close to the beam centre. The Earth–Target (T) and Earth–Chaser (C) position vectors are normalised,

$$\hat{\mathbf{r}}_T = \frac{\mathbf{r}_T}{\|\mathbf{r}_T\|}, \quad \hat{\mathbf{r}}_C = \frac{\mathbf{r}_C}{\|\mathbf{r}_C\|}, \quad (5.24)$$

and the off-axis angle, illustrated in Figure 5.4, is computed as

$$\theta_{eavs} = \cos^{-1}(\hat{\mathbf{r}}_T \cdot \hat{\mathbf{r}}_C). \quad (5.25)$$

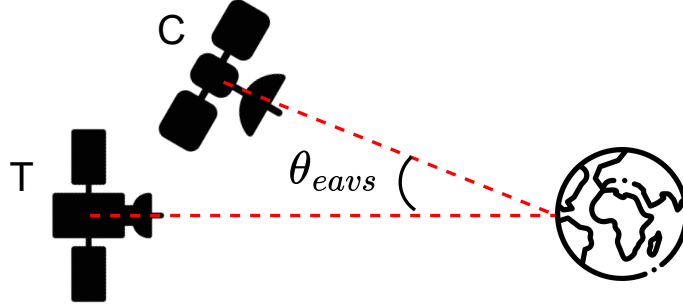


Figure 5.4: Illustration of target–Earth–chaser phase angle θ .

Samples satisfying

$$\theta_{eavs} < \theta_{3db} \quad (5.26)$$

are considered to lie inside the communications cone, where θ_{3db} is the ground station's 3 dB half-angle, illustrated in Figure 5.5.

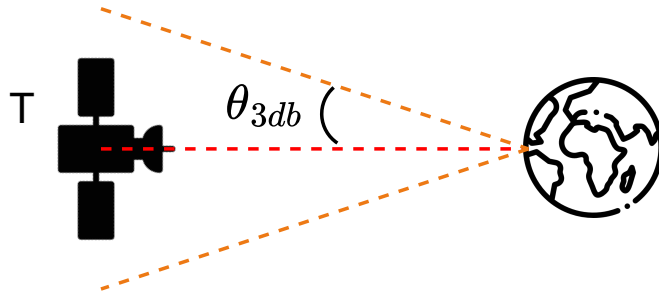


Figure 5.5: Illustration of the communications cone

Let

$$f_{\text{beam}} = \frac{N_{\text{inside}}}{N}, \quad (5.27)$$

denote the fraction of propagated samples inside the cone, and let $\bar{\theta}$ denote the mean off-axis angle of those samples. The objective is

$$J_{\text{beam}} = -f_{\text{beam}} \left(1 - \frac{\bar{\theta}}{\theta_{\text{beam}}} \right). \quad (5.28)$$

If no propagated samples fall within the communications cone, a penalty value of 10^6 is returned.

5.3.3. Mission Phase Inputs

For the mission simulation of the trajectories, a time-frame for each trajectory is required. As discussed in Chapter 4, the exact execution of a non-cooperative RPO is in all cases unclear, and this holds for the time-frame over which an intent is executed as well. For the two intents considered that require mission trajectories, *Inspection* and *Eavesdropping*, some indications are nonetheless available. The LUCH (OLYMP) satellites typically remain near a target for weeks to months [4]. Although multiple manoeuvres can occur within such a period, this indicates that a representative *Eavesdropping* trajectory should span a similar order of magnitude. For *Inspection* intents, GSSAP satellites are reported to “loiter” near other satellites, though no precise duration is given in open sources [3, 4].

Besides intent, the orbital regime is an important consideration for mission duration, for two reasons. First, a stable relative trajectory is harder to sustain in LEO than in GEO, since orbital perturbations, primarily atmospheric drag and J_2 , act far more strongly at lower altitudes [104]. A simulated trajectory in LEO would therefore require more frequent station-keeping manoeuvres for the same duration as in GEO. Second, lighting conditions change considerably faster in LEO than GEO, owing to the shorter orbital period (~ 1.5 h, versus ~ 24 h in GEO), which matters for intents that depend on sustained optical geometry, such as *Inspection*.

A mission duration of 3 days is set for the GEO domain. This value is scaled down for the lower regime to reflect their reduced trajectory stability and faster-changing lighting conditions, giving 1 day for LEO. Table 5.2 summarises the adopted durations alongside the orbital characteristics motivating them.

Table 5.2: Mission duration per orbital regime, with the orbital characteristics motivating the scaling.

Orbital regime	Orbital period	Dominant perturbations [104]	Mission duration
LEO	~ 1.5 h	Atmospheric drag, J_2	1 day
GEO	~ 24 h	3rd body Lunar and Solar, Solar Radiation Pressure	3 days

The *Eavesdropping* intent relies on receiving communications from the ground. For this, the 3 dB half-angle of the ground station’s emissions is used. For a given wavelength and dish diameter of a parabolic antenna, the 3 dB half-angle can be approximated using [105]:

$$\theta_{3\text{dB}} = 70 \frac{\lambda}{D} \quad (5.29)$$

Considering the UHF, S-band, and X-band frequency ranges across varying dish sizes, Figure 5.6 can be generated. The figure also includes several dishes used for communication with target satellites [106]. There is a range of realistic values, and, as with mission duration, this range differs between orbital domains. In GEO, where high-throughput communication satellites are common, higher frequencies and larger dishes are typical, such as the KSAT Svalbard X-band station with a 13 m dish diameter. In LEO, the situation differs: constellations provide the links, and with more satellites sharing the load, each satellite requires less throughput individually. This favours lower-frequency bands and smaller dishes.

Based on this, a half-angle range of 0.8° to 1.5° is considered representative for LEO, and 0.2° to 1.0° for GEO.

For the stand-off distances, a range is considered where the lower bound is 10 km for GEO and 2 km for LEO. The GEO lowerbound is closer than what LUCH satellites have been observed to do (Chapter 4), but within the same order of magnitude. The upper bound is scaled to the beam cone radius at the domain’s reference range, computed as $\text{range} \times \tan(\text{half-angle})$, giving a cone radius of

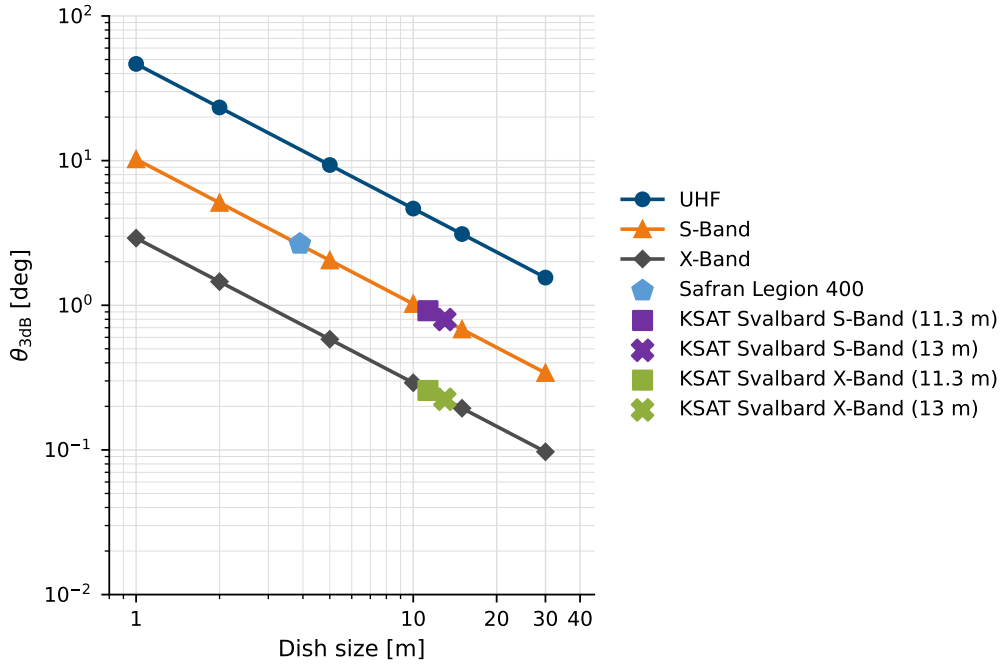


Figure 5.6: 3dB half-angle as a function of dish size for UHF, S-band, and X-band, with example ground-station dishes indicated [106].

7.0–13.1 km for LEO and 125–625 km for GEO across the stated half-angle bounds, from which the nominal ceiling is derived.

Lastly, the input for the *Inspection* intent requires information on Sun exclusion angles, since optical sensors can be dazzled if the Sun falls within their field of view. An overview of Sun exclusion angles for various commercial off-the-shelf sensors (both real and theoretical) is given in Table 5.3.

Table 5.3: Overview of commercial off-the-shelf optical sensors and their Sun exclusion angles (SEA).

Product	SEA (deg)	Application	Source
HICAM – 1 MP Space Detection & Navigation Camera	26	Space detection, navigation, SSA missions	[107]
AAC Clyde Space – 4 MP Optical Imager for Nanosatellites	30, 45	‘Imaging’	[108, 109]
AuriCam line-up – D35, D80, D8	38, 64, 45	‘Imaging’	[110]
Sapphire Sat	45	SSA	[111]

There is considerable variation between sensors, and the overview is not exhaustive; it nonetheless provides a reasonable indication of realistic values. Based on the observed range of values, the Sun exclusion angle for the *Inspection* intent is set in the range of 25° to 65°.

5.3.4. Optimiser Implementation

The multi-objective optimisation required by the multi-objective problems of *Inspection* and *Eavesdropping* is performed using the Non-dominated Sorting Genetic Algorithm II (NSGA-II). NSGA-II is a fast and elitist multi-objective evolutionary algorithm that combines computationally efficient non-dominated sorting with crowding-distance-based diversity preservation to approximate the Pareto-optimal front with reduced computational complexity [112]. During each generation, individuals are ranked into successive Pareto fronts, and selection, crossover, and mutation operators evolve the population while maintaining a diverse set of trade-off solutions.

NSGA-II is ran for a maximum of a hundred generations. To avoid running the full generation budget when no further improvement is being made, an early-stopping criterion based on hypervolume convergence is used. The reference point is fixed once, from the initial population at generation (g) 0, as the component-wise maximum of the objective values with a 10% margin,

$$\mathbf{r}_{\text{ref}} = 1.1 \cdot \max_i \mathbf{f}_i^{(0)}. \quad (5.30)$$

For each objective, either the minimum value for an objective is used, or the value of the reference point itself is taken for an objective if it is the minimum,

$$\mathbf{f}_i^{\text{clip}} = \min(\mathbf{f}_i, \mathbf{r}_{\text{ref}} - 10^{-12}), \quad \mathbf{f}_i \in \text{Pareto front}(g), \quad (5.31)$$

The hypervolume $H(g)$ is defined as the multi-dimensional volume of the region jointly enclosed between the reference point and the points of the Pareto front at generation g ,

$$H(g) = \lambda \left(\bigcup_i [\mathbf{f}_i^{\text{clip}}, \mathbf{r}_{\text{ref}}] \right), \quad (5.32)$$

where $\lambda(\cdot)$ denotes the Lebesgue measure and $[\mathbf{f}_i^{\text{clip}}, \mathbf{r}_{\text{ref}}]$ is the axis-aligned hyperrectangle spanned by a clipped Pareto-front point and the reference point.

Once at least 11 generations have been evaluated, the relative change in hypervolume over a trailing 10-generation window is computed,

$$\Delta_k = \frac{|H(g - k + 1) - H(g - k)|}{H(g - k)}, \quad k = 1, \dots, 10, \quad (5.33)$$

with the denominator replaced by 1 when $H(g - k) = 0$. The optimisation is considered converged, and terminates early, when

$$\max_k \Delta_k < \varepsilon, \quad \varepsilon = 1.0 \times 10^{-3}, \quad (5.34)$$

i.e. when every consecutive-generation relative change in hypervolume across the window falls below 0.1%, rather than continuing to the maximum budget of 100 generations. Across all runs, the maximum budget of a 100 generations was never reached.

The final solution is selected from the last generation by identifying the set of Pareto-optimal individuals. These are the solutions that are not dominated by any other solution in the population, meaning that no other individual performs at least as well in every objective while being strictly better in at least one objective.

Since multiple Pareto-optimal solutions generally exist in the front, a single representative solution is chosen using the distance-to-ideal-point criterion. First, the objective values of the Pareto front are normalised using min–max normalisation,

$$\hat{f}_{i,j} = \frac{f_{i,j} - f_j^{\min}}{f_j^{\max} - f_j^{\min}}, \quad (5.35)$$

where $f_{i,j}$ is the value of objective j for solution i , and $f_j^{\min}$ and $f_j^{\max}$ are the minimum and maximum values of objective j among the Pareto-optimal solutions. If $f_j^{\max} = f_j^{\min}$, the denominator is replaced by 1 to avoid division by zero. An illustration of the normalisation process is shown in Figure 5.7.

After normalisation, the ideal point corresponds to the origin, $\mathbf{0}$, since each objective is minimised independently. The Euclidean distance of each Pareto-optimal solution to the ideal point is then computed as

$$d_i = \|\hat{\mathbf{f}}_i\|_2 = \sqrt{\sum_{j=1}^m \hat{f}_{i,j}^2}, \quad (5.36)$$

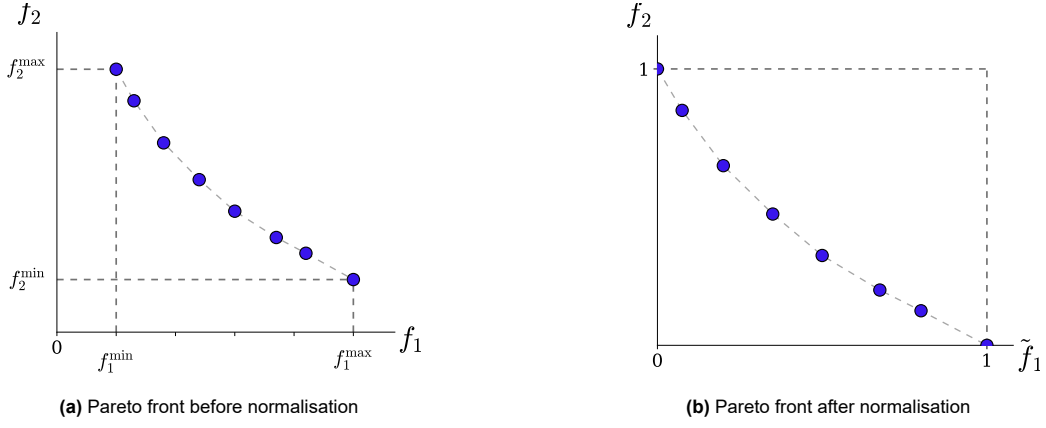


Figure 5.7: Effect of objective normalisation on the Pareto front.

where m is the number of objectives. The solution with the smallest distance,

$$i^* = \arg \min_i d_i, \quad (5.37)$$

is selected as the final solution. This approach provides a balanced compromise by selecting the Pareto-optimal individual that lies closest to the ideal point while ensuring that all objectives contribute equally through normalisation.

5.3.5. Mission Phase Results

The mission phase optimisation results for *Inspection* can be found in Figure 5.8, shown in the same format as the case studies for *Inspection* in GEO and LEO of Section 4.2.

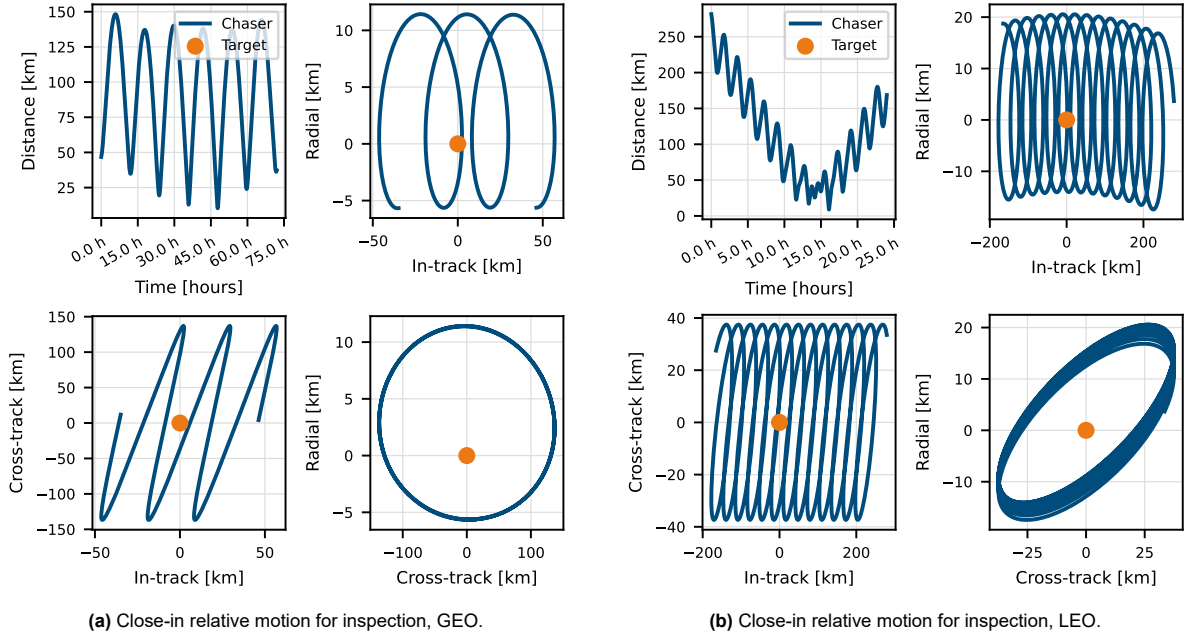


Figure 5.8: Example optimisation results for inspection intents.

The *Inspection* trajectories for both LEO and GEO in Figure 5.8 display a similar pattern: a drifting relative trajectory in which the chaser moves in an elliptical or circular path around the target. What differs is that in LEO, a larger relative drift rate and shorter orbital period can be observed, as indicated by higher frequency oscillations and the larger in-track distances.

As compared to the cases from Section 4.2, they display similar behaviour to that observed in LEO, whilst the case study for GSSAP in GEO showed considerably larger distances during close approach.

For *Eavesdropping*, these type of plots are shown in Figure 5.9 and can only be compared to the LUCH *Eavesdropping* cases in Section 4.2.

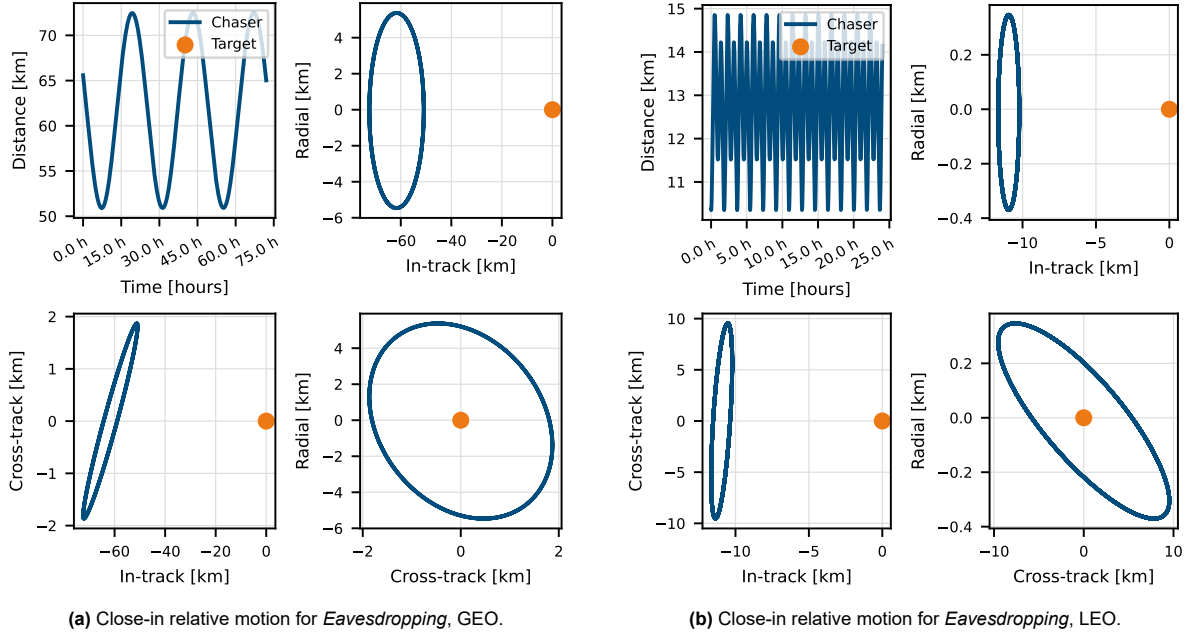


Figure 5.9: Example optimisation results for *Eavesdropping*.

In both LEO and GEO, the general behaviour of *Eavesdropping* is that of an off-set, stable, oscillating behaviour of the Chaser relative to the target. This is the same behaviour that can be observed for LUCH in GEO as per Section 4.2. The main differences between LEO and GEO is that of the distances involved; LEO *Eavesdropping* cases happen closer by, most likely due to the smaller communications cone due to being closer to Earth and a smaller θ_{3db} angle as per Subsection 5.3.3.

5.4. Orbital Adjustment Phase

The Orbital Adjustment Phase optimises the approach from starting conditions determined by the satellite chaser and target for a seed from the database (see Section 5.2). Two objectives are considered, depending on the intent; *Inspection*, *Eavesdropping*, and *Docking* all minimise only propellant through the proxy of ΔV , whilst *Collision* balances ΔV and tof.

5.4.1. Simulation Pipeline

The optimiser operates on a ten-element, normalised decision vector $\mathbf{x} \in [0, 1]^{10}$, decoded into physical quantities before being passed to the propagator.

The decision vector is structured as

$$\mathbf{x} = \left[\underbrace{x_0, x_1, x_2}_{\Delta V_1}, \underbrace{x_3, x_4, x_5}_{\Delta V_2}, \underbrace{x_6, x_7, x_8, x_9}_{\text{epoch fractions}} \right], \quad (5.38)$$

where the first six elements encode two three-dimensional ΔV vectors and the final four elements encode the manoeuvre and termination epochs as fractions of the available time window.

Each ΔV component in $[0, 1]$ is mapped to $[-\Delta V_{\max}, \Delta V_{\max}]$:

$$\Delta \mathbf{V}_1 = (\mathbf{x}_{0:3} - 0.5) \cdot 2 \Delta V_{\max}, \quad (5.39)$$

$$\Delta \mathbf{V}_2 = (\mathbf{x}_{3:6} - 0.5) \cdot 2 \Delta V_{\max}, \quad (5.40)$$

where $\Delta V_{\max}$ is a mission-specific bound on the magnitude of each component. Centring the encoding on zero delta-V gives the optimiser a symmetric search range in every axis, which improved optimiser performance. The maximum ΔV for a burn is thus $\sqrt{3} \Delta V_{\max}$.

The four epochs (t_i) are decoded using a shrinking-window fractional gap, which guarantees strict ordering $t_1 < t_2 < t_3 < t_4$ without an explicit ordering constraint in the objective. Let T denote the total available duration and $R_0 = T$ the initial remaining window. Then, for $i = 1, \dots, 4$,

$$\delta_i = x_{5+i} R_{i-1}, \quad (5.41)$$

$$R_i = R_{i-1} - \delta_i, \quad (5.42)$$

where δ_i is the i -th offset and R_i the remaining window after it is consumed. The absolute epochs follow by cumulative summation from the mission start epoch t_0 :

$$t_1 = t_0 + \delta_1, \quad (5.43)$$

$$t_2 = t_1 + \delta_2, \quad (5.44)$$

$$t_3 = t_2 + \delta_3, \quad (5.45)$$

$$t_4 = t_3 + \delta_4, \quad (5.46)$$

with t_1, t_2, t_3 the three manoeuvre epochs and t_4 the termination epoch. By construction, the ordering $t_1 < t_2 < t_3 < t_4 \leq t_0 + T$ holds for any $\mathbf{x} \in [0, 1]^{10}$.

The decoded manoeuvre parameters drive the propagator, which advances both the target and chaser from t_0 to t_4 in four legs. The target coasts passively throughout; the chaser receives the two prescribed impulsive burns $\Delta \mathbf{V}_1, \Delta \mathbf{V}_2$, followed by two Lambert-solved burns $\Delta \mathbf{V}_3, \Delta \mathbf{V}_4$ that bring it to the required end condition.

Each leg is propagated analytically rather than numerically integrated: the Cartesian state $\mathbf{X}$ at the start of the leg is converted to osculating Keplerian elements ($\mathcal{K}(\mathcal{X})$), advanced by the leg duration via Tudatpy's `propagate_kepler_orbit` [103], and converted back to Cartesian coordinates,

$$\mathbf{X}(t_{k+1}) = \mathcal{K}^{-1}(\mathcal{K}(\mathbf{X}(t_k)) + \Delta t_k), \quad \Delta t_k = t_{k+1} - t_k, \quad (5.47)$$

where $\mathcal{K}(\cdot)$ converts a Cartesian state to Keplerian elements and $\mathcal{K}^{-1}(\cdot)$ converts back, after the mean anomaly has been advanced by Δt_k under two-body dynamics. This applies to the target across all four legs ($t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$) and to the chaser on the legs between impulsive events.

The chaser's terminal target is defined relative to the target's end-of-mission state $\mathbf{X}_T(t_4) = [\mathbf{r}_T, \mathbf{v}_T]$, the state determined by the mission optimisation of Section 5.3, rather than as an absolute state in the inertial frame. The rotation matrix $\mathbf{R}_{\text{RSW} \rightarrow I}(\mathbf{r}_T, \mathbf{v}_T)$ from the radial-along-track-cross-track (RSW/LVLH) frame to the inertial frame maps the prescribed relative offset $(\delta \mathbf{r}, \delta \mathbf{v})$ into an inertial target state:

$$\mathbf{r}_C^{\text{target}} = \mathbf{r}_T + \mathbf{R}_{\text{RSW} \rightarrow I} \delta \mathbf{r}, \quad \mathbf{v}_C^{\text{target}} = \mathbf{v}_T + \mathbf{R}_{\text{RSW} \rightarrow I} \delta \mathbf{v}. \quad (5.48)$$

The Lambert leg is solved for this target state, not for the target satellite's state itself.

The chaser is propagated and burned sequentially, with each burn instantaneously changing velocity but not position. Superscripts pre, i and post, i denote the chaser's state immediately before and after the i -th burn:

$$\mathbf{X}_C^{\text{pre},1} = \text{propagate}(\mathbf{X}_C(t_0), t_0 \rightarrow t_1), \quad \mathbf{X}_C^{\text{post},1} = \mathbf{X}_C^{\text{pre},1} + [0, \Delta \mathbf{V}_1], \quad (5.49)$$

$$\mathbf{X}_C^{\text{pre},2} = \text{propagate}(\mathbf{X}_C^{\text{post},1}, t_1 \rightarrow t_2), \quad \mathbf{X}_C^{\text{post},2} = \mathbf{X}_C^{\text{pre},2} + [0, \Delta \mathbf{V}_2], \quad (5.50)$$

$$\mathbf{X}_C^{\text{pre},3} = \text{propagate}(\mathbf{X}_C^{\text{post},2}, t_2 \rightarrow t_3), \quad (5.51)$$

$\mathbf{X}_C^{\text{pre},3}$ is the departure state handed to the Lambert solver.

Given the departure state $\mathbf{X}_C^{\text{pre},3} = [\mathbf{r}_{\text{dep}}, \mathbf{v}_{\text{dep}}]$ at t_3 , the arrival position $\mathbf{r}_C^{\text{target}}$ and $\mathbf{v}_C^{\text{target}}$ at t_4 , and the time of flight $\Delta t_{34} = t_4 - t_3$, Lambert's problem is solved for the transfer orbit connecting the two positions in the prescribed time (subscripts dep and arr throughout refer to the departure and arrival ends of this transfer). Izzo's zero-revolution formulation [113] is used as the primary solver, returning the radial and transverse velocity components at both endpoints of the transfer arc, $(v_{r,\text{dep}}, v_{t,\text{dep}})$ and $(v_{r,\text{arr}}, v_{t,\text{arr}})$. Izzo's method is used first because it is generally fast and robust [113]; Gooding's formulation serves as a fallback because it remains valid for co-planar transfers, where Izzo's solver is ill-defined [114].

These scalar components are expressed in the RSW (Radial–Along-track–Cross-track) frame and must be rotated into the inertial frame. The cross-track unit vector is obtained from the departure and arrival position vectors,

$$\hat{\mathbf{W}} = \frac{\mathbf{r}_{\text{dep}} \times \mathbf{r}_{\text{arr}}}{\|\mathbf{r}_{\text{dep}} \times \mathbf{r}_{\text{arr}}\|}, \quad (5.52)$$

with the radial and along-track unit vectors at each endpoint given by $\hat{\mathbf{R}} = \mathbf{r}/\|\mathbf{r}\|$ and $\hat{\mathbf{S}} = \hat{\mathbf{W}} \times \hat{\mathbf{R}}$. The transfer velocities follow as

$$\mathbf{v}_{\text{transfer}} = v_R \hat{\mathbf{R}} + v_S \hat{\mathbf{S}}. \quad (5.53)$$

If $\mathbf{r}_{\text{dep}}$ and $\mathbf{r}_{\text{arr}}$ are (near-)collinear, $\|\mathbf{r}_{\text{dep}} \times \mathbf{r}_{\text{arr}}\| \rightarrow 0$ and the cross-track direction is ill-defined. In this degenerate case the solver falls back to Gooding's formulation and instead determines the orbit plane from the departure position and velocity, $\hat{\mathbf{W}} = (\mathbf{r}_{\text{dep}} \times \mathbf{v}_{\text{dep}})/\|\mathbf{r}_{\text{dep}} \times \mathbf{v}_{\text{dep}}\|$, which remains well-defined provided the departure state is not itself purely radial.

The two impulsive burns follow from the difference between the transfer velocity and the actual (pre- and post-transfer) velocities:

$$\Delta \mathbf{V}_3 = \mathbf{v}_{\text{transfer,dep}} - \mathbf{v}_{\text{dep}}, \quad \Delta \mathbf{V}_4 = \mathbf{v}_{\text{arr}}^{\text{target}} - \mathbf{v}_{\text{transfer,arr}}. \quad (5.54)$$

With $\Delta \mathbf{V}_3$ applied, the resulting state is propagated forward under Keplerian dynamics from t_3 to t_4 using the same analytic method as the earlier legs, and $\Delta \mathbf{V}_4$ is applied at arrival to complete the transfer.

The total delta-V cost reported to the objective is the sum of all burn magnitudes, $\Delta V_{\text{tot}} = \sum_{i=1}^4 \|\Delta \mathbf{V}_i\|$, for all intents except *Collision*, where only the first three are summed ($\Delta V_{\text{tot}} = \sum_{i=1}^3 \|\Delta \mathbf{V}_i\|$). The overall pipeline is summarised in Figure 5.10. The propagator returns

$$(\Delta V_{\text{tot}}, t_f, \mathbf{X}_T^{\text{events}}, \mathbf{X}_C^{\text{pre}}, \mathbf{X}_C^{\text{post}}) \quad (5.55)$$

where $\{\Delta V_i\}_{i=1}^4$ are the magnitudes of the two prescribed and two Lambert-solved burns; t_f is the total transfer time; $\mathbf{X}_T^{\text{events}}$ is the (4×6) array of the target's Cartesian state at each of the four manoeuvre epochs $t_1, \dots, t_4$; and $\mathbf{X}_C^{\text{pre}}, \mathbf{X}_C^{\text{post}}$ are the corresponding (4×6) arrays of the chaser's state immediately before and after each burn. Any failed run is caught and penalised, keeping the optimiser away from infeasible regions of the search space.

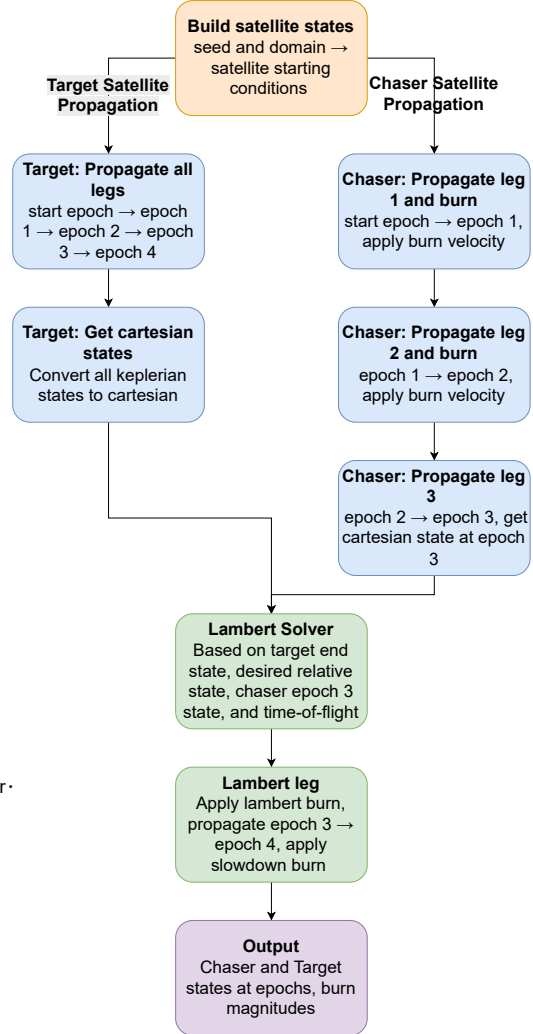


Figure 5.10: Propagation pipeline

The scalar objective combines the total delta-V budget with several penalty terms:

$$J(\mathbf{x}) = \Delta V_{\text{tot}} + P_v + P_{\text{peri}} + P_{\text{order}} + \begin{cases} t_f/10^3, & \text{collision scenario,} \\ 0, & \text{otherwise,} \end{cases} \quad (5.56)$$

where $\Delta V_{\text{tot}} = \sum_{i=1}^4 \Delta V_i$; P_v penalises delta-V budget violations; P_{peri} penalises perigee-altitude violations evaluated on $\mathbf{X}_C^{\text{post}}$; and P_{order} penalises epoch-ordering violations (in practice unreachable given the shrinking-window encoding, but retained for robustness). A failed propagation returns $J(\mathbf{x}) = 10^{10}$, driving the optimiser away from that region of the search space.

5.4.2. Optimiser Selection

As discussed in Section 2.1, several different options for the optimisation of the orbital adjustment phase are available. For the purposes of this thesis, a meta-heuristic algorithm is used due to its flexibility, ease of implementation, and documented proof of viability for multi-burn rendezvous (see Chapter 2). Of meta-heuristic algorithms, the SGA, SADE, PSO, and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithms were selected due to their prevalence in the literature. All of these algorithms work with a population of individuals and evolve them based upon different rule sets. All implementations of these algorithms are done using the pygmo library [115] and further explanation is present in Appendix A.

SGA

The SGA algorithm is one of the first evolutionary algorithms developed and, as the name implies, one of the simplest considered [43]. The implementation follows a generational cycle: binary tournament selection fills a mating pool by repeatedly drawing two individuals and keeping the fitter one, after which simulated binary crossover (SBX) combines pairs of parents into offspring, with a distribution index controlling how closely offspring cluster around their parents. Polynomial mutation then perturbs each decision variable independently with a small probability, maintaining genetic diversity and helping prevent premature convergence. Population replacement follows a pure elitist strategy: parent and offspring populations are pooled and ranked by fitness, with only the best retained, guaranteeing the best solution found so far is never lost, at the cost of a higher risk of premature convergence since fitter individuals can dominate the mating pool across many consecutive generations.

SADE

SADE extends classic differential evolution by self-adapting its control parameters, allowing the mutation scale factor and crossover rate to evolve alongside the population rather than being fixed [116]. For each individual, a mutant vector is formed from the weighted difference between two other randomly selected individuals, added to a third; a trial vector is then built by copying a contiguous block of components from this mutant into the parent, with the crossover rate controlling how much of the mutant is drawn in. The trial vector replaces the parent only if it improves on its fitness. Each individual carries its own mutation and crossover settings, occasionally reinitialised at random; settings attached to successful trial vectors are passed on to the next generation, biasing the population over time towards settings that have empirically worked well. The algorithm terminates early once the population's spread in both fitness and decision variables falls below the specified tolerances, indicating effective convergence.

PSO

PSO models the population as a swarm of particles moving through the search space, each carrying a position and velocity rather than undergoing genetic operators [117]. At every generation, each particle's velocity is updated by combining an inertia term with a cognitive term pulling it towards its own best-known position and a social term pulling it towards the best position found across the swarm, with randomised weighting introducing stochasticity into the update. The inertia weight governs the exploration–exploitation trade-off: a higher value favours continued movement in the current direction, while a lower value lets the cognitive and social terms dominate. The updated velocity is clamped to a maximum magnitude to prevent particles overshooting the search space in a single step, after which the particle's position is updated accordingly. Personal and global bests are refreshed only once all particles have moved for the current generation, rather than incrementally, so an improved position only becomes visible to other particles in the following generation; this cycle then repeats for the specified number of generations.

CMA-ES

CMA-ES departs from the other algorithms in that it maintains no population between generations; instead, it samples an entire offspring population directly from a multivariate Gaussian distribution parameterised by a mean and covariance matrix [118]. After ranking the offspring by fitness, only the best are retained for recombination, and the new mean is formed as a rank-weighted average of these. Two evolution paths are then updated: a step-size path, which accumulates the direction of successive mean shifts and adapts the step size accordingly (increasing it when steps point in similar directions, indicating overly cautious movement, and decreasing it when steps cancel out, indicating overshoot), and a covariance path, which accumulates

The testing of these algorithms was performed using the *docking* intent over 40 seeds. The initial assessment involved running all four algorithms over 2000 generations g . This resulted in the evolutionary behaviour shown in Figure 5.11a for GEO as the domain, and Figure 5.11b for LEO, with the average runtimes presented in Table 5.4.

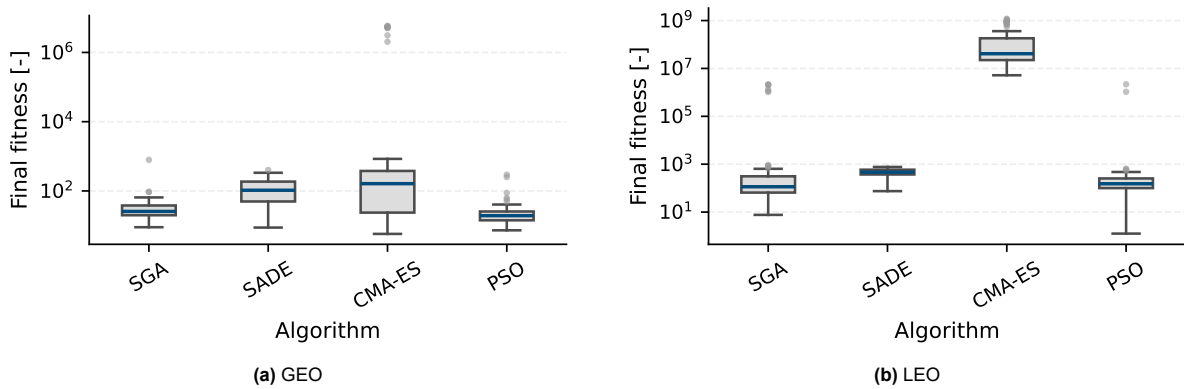


Figure 5.11: Convergence of the four base genetic algorithms on the docking intent, across all seeds, in the GEO and LEO domains.

Table 5.4: Mean optimiser runtime per domain, averaged over 40 seeds.

Domain	SGA [s]	SADE [s]	CMA-ES [s]	PSO [s]
LEO	106.8	120.1	0.9	120.9
GEO	91.5	104.6	62.7	105.6

There is no clear single best-fit solution. In GEO, PSO and SGA both risk failing to reach an optimal solution, as suggested by their high means, whilst SADE's higher median suggests it struggles to approach the global optimum; on balance, SGA performs best on mean cost in GEO, whilst PSO performs best in LEO. Furthermore, for some seeds stagnation seems to occur, with no improvement over hundreds of generations; in these cases, early termination would be preferable. Because of this, alternatives were investigated, constructed by taking the three genetic algorithms that performed well, SGA, PSO, and SADE, and altering them.

First, all three algorithms get a custom early-stopping condition. As noted in Chapter 2, genetic algorithms can be good at covering the search space and nearing a global optimum, but bad at converging to it once they are close. Stopping the algorithm once it is close to the optimum and handing over the best solution to another algorithm is a common way to improve convergence speed [41]. In this case, each algorithm is handed over to a Nelder–Mead optimiser, which is a numerical method that constructs a simplex on the search space and expands, contracts, or reflects its vertices to refine the solution. This combination of early-stopping and Nelder–Mead refinement is applied throughout the remaining tests for all algorithms.

Stagnation is investigated with different stagnation conditions, which combine the number of generations the algorithm is allowed to stagnate for and the tightness of the stagnation threshold. Table 5.5 contains the different combinations of stagnation conditions considered.

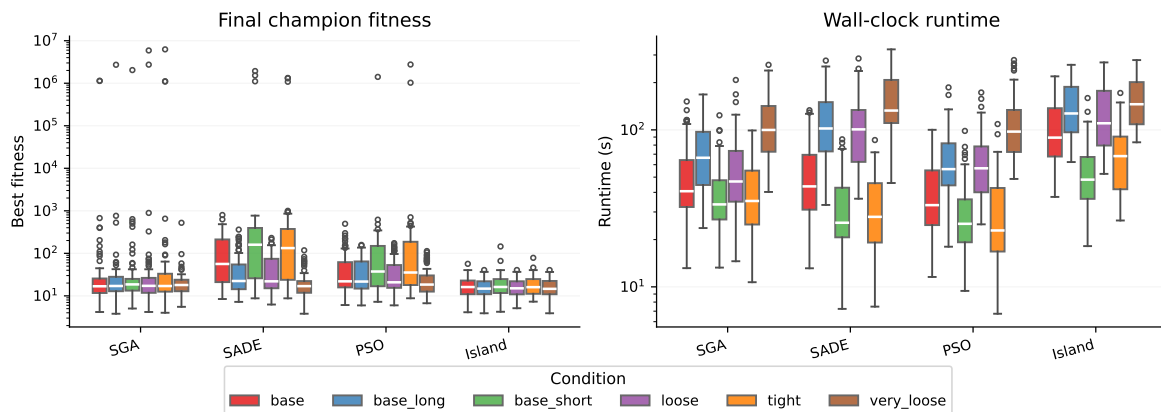
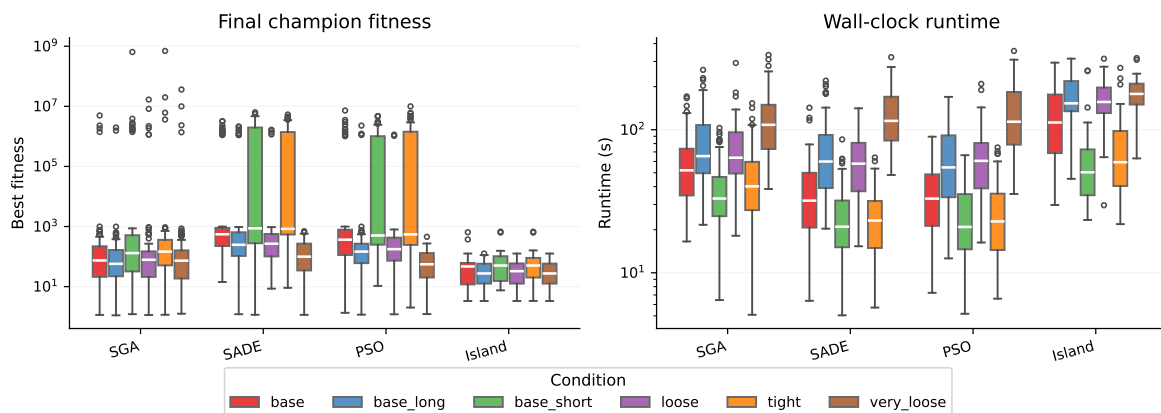
Table 5.5: Stagnation-detection conditions tested for early termination.

Condition	Stagnation window [gen]	Stagnation threshold [%]
Base	100	1.0
Base Long	200	1.0
Base Short	50	1.0
Tight	50	0.5
Loose	200	2.0
Very Loose	500	5.0

In Table 5.5, the *Base* condition considers stagnation to have occurred if the algorithm does not improve the objective function by one percent or more for 100 generations. When this condition is met, the optimiser hands the optimum fitness over to Nelder-Mead, does local refinement.

Furthermore, an island optimiser is also considered, using the same set of stagnation conditions. Island optimisers work by combining different base algorithms and running them in parallel. Once every set number of generations (in this case, 50), the best individuals from each algorithm are migrated across to the other algorithm. This allows the beneficial behaviours of different algorithms to be combined. For this version, SGA and SADE are combined in an island configuration. One island is equivalent to the base algorithm it consists of, meaning the combined islands have increased computational load per generation as compared to the base algorithms.

The optimisation results in terms of runtime and convergence are shown as boxplots with their medians in Figure 5.12 for GEO, and Figure 5.13 for LEO.

**Figure 5.12:** Runtime and convergence of the SGA–SADE island configuration compared to the base algorithms, GEO domain.**Figure 5.13:** Runtime and convergence of the SGA–SADE island configuration compared to the base algorithms, LEO domain.

In terms of convergence behaviour, across all stagnation conditions, the island optimiser performs best. Across the different stagnation conditions, the convergence of the island optimisers is about the same, meaning the choice of stagnation condition does not have much influence on the end results. This is also the case for SGA, but not for PSO and SADE, which can vary with stagnation conditions. However, looking at the tail ends of the distribution, other algorithms can score better for a given seed than the island optimiser. This means that global convergence is not guaranteed.

The difference between stagnation conditions for the island optimiser appears in the runtime comparison. The fastest is *Base Short*, whilst the longest median runtime comes from *Very Loose*. This is also reflected in the convergence behaviour, where the island optimisers that took longer also score slightly better. Notably, the island optimiser with the *Tight* stagnation condition performs similarly to *Very Loose*, which performed best, whilst being one of the faster conditions.

For this reason, the island optimiser is used with the *Tight* stagnation condition. As with a mean runtime of 74 s in GEO, the projected runtime for this domain across 40 000 runs is 34 days, whilst for *Very Loose* it would be almost 70 days. Reduced run-time is achieved by running the seeds in parallel, but still lasts about 4 days per domain.

5.4.3. Adjustment Phase Results

The results of the orbital adjustment phase are what the intent inference model is trained on, and thus there need to be features that allow for discerning one intent from another.

The lead-up to *Inspection* and *Eavesdropping* for the same starting seed in GEO is presented in Figure 5.14. Since both have the same starting condition and lead-up objective, difference between the two approaches arise from the difference in target conditions.

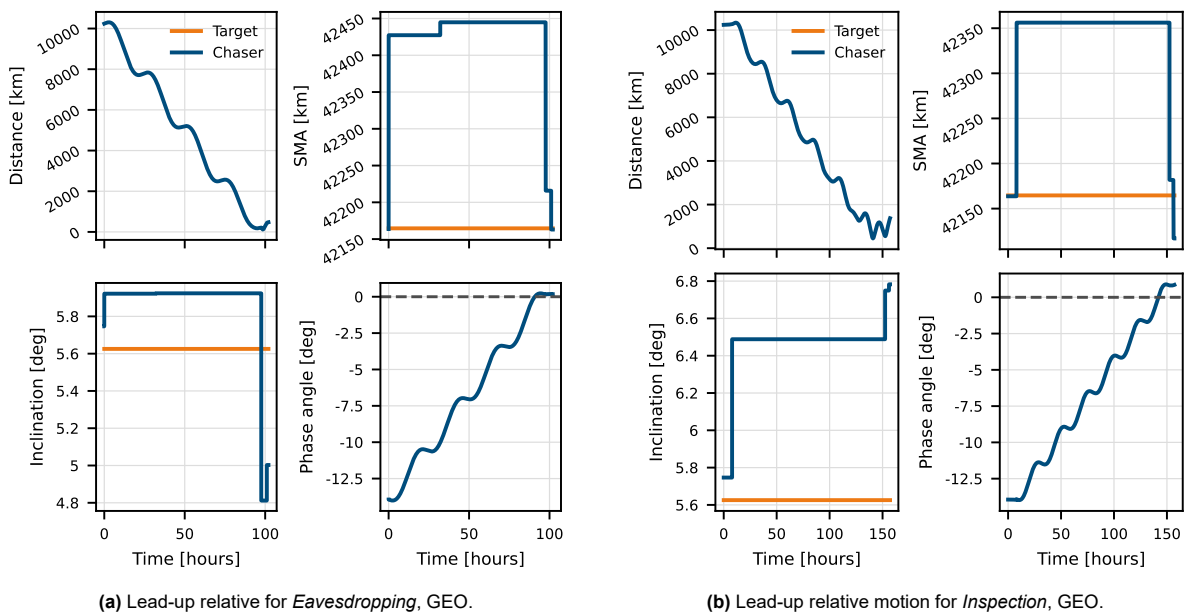


Figure 5.14: Example lead-up optimisation results for GEO.

Two main differences can be observed: for *Eavesdropping* the approach is faster than for *Inspection*, and the main observable difference between the *Inspection* and *Eavesdropping* ending conditions is that for *Eavesdropping* the SMA is equal to that of the target, whilst for *Inspection* it is not. The former is a result of the orbital adjustment optimiser, which can end the transfer faster than the maximum allotted time. The latter is a result of the mission phase optimisation, where *Eavesdropping* in general has a chaser orbit that does not drift relative to the target, whilst *Inspection* does.

The same plots for *Docking* and *Collision* are presented below. Unlike the *Inspection*/*Eavesdropping* pair, whose difference can from the mission phase differences, the difference between *Docking* and

Collision arises only from the orbital adjustment phase.

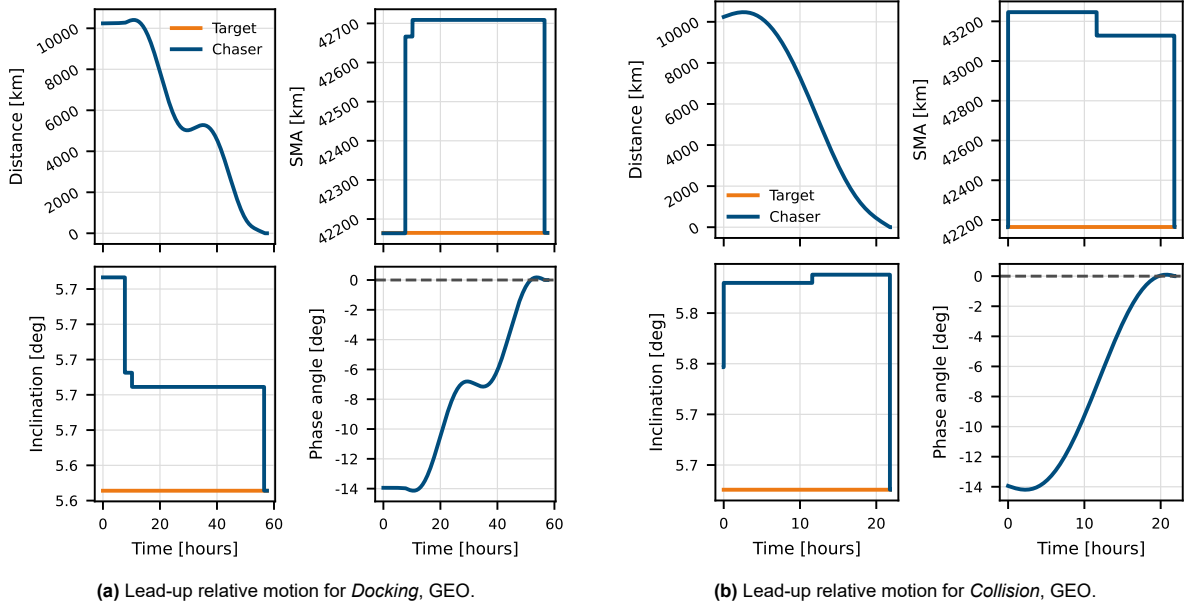


Figure 5.15: Example lead-up optimisation results for GEO.

The main observable difference between the *Docking* and *Collision* intents is the time the approach takes, which is about a third for *Collision* than it is for *Docking*. This is due to the objective difference between the two, where *Collision* is incentivised to minimise tof, along with the ΔV .

For LEO, these same plots are presented in Appendix A. They show the same general behaviour.

5.5. Real-World Data Acquisition

All real-world data are acquired through Space-Track's TLE catalogue. Two approaches are used to detect manoeuvres: a method tailored specifically to the LUCH-1 and LUCH-2 spacecraft, and a general-purpose method applicable more broadly. This split is necessary for large-scale acquisition of orbital data, since manual inspection of trajectories and manoeuvres is infeasible at scale. The LUCH satellites are treated separately because they employ low-thrust manoeuvring, for which the general method is not configured; their consistent behaviour (see Section 4.2) makes the specific algorithm possible for this pair.

5.5.1. LUCH-1 and -2 Algorithm

The file-generation procedure for LUCH relies on the consistent behaviour of the LUCH satellites, whose manoeuvres towards a target are driven by changes in SMA that induce a drift relative to the GEO belt. Both LUCH satellites used this drift to move between the targets of successive eavesdropping periods, as shown in Chapter 4's Figure 4.1 and Figure 4.5. Both satellites use electric propulsion, so a single manoeuvre is not (near-)instantaneous and may last several days, with multiple TLEs falling between its start and end. An algorithm determines the burns before each eavesdropping episode by clustering the burn TLEs, and taking the pre- and post-burn TLEs for the pre- and post-manoeuver states.

The algorithm proceeds in two phases: first, the target of each eavesdropping period is identified; second, the four manoeuvres leading up to the start of that period are detected.

An eavesdropping period is defined as a time interval during which a LUCH satellite's SMA, a , remains at the GEO value, $a_{\text{GEO}} = 42,164.14$ km, and therefore does not drift relative to the belt. A sample at epoch t_i is classified as GEO-resident if

$$|a(t_i) - a_{\text{GEO}}| \leq \Delta a_{\text{thresh}}, \quad (5.57)$$

and the start and end of an eavesdropping period are taken as the first and last consecutive epochs satisfying Equation 5.57. The threshold value a_{thresh} for SMA deviation is used to negate the effect of noise, which causes small changes in the SMA between TLEs, and was determined qualitatively as 5 km.

At the onset of an eavesdropping period, the target is identified by comparing the instantaneous longitude of the LUCH satellite to that of every candidate target at the same epoch. The longitude λ of a satellite is obtained from its orbital elements as

$$\lambda = (\Omega + u - \theta_{\text{GAST}}) \bmod 360^\circ, \quad (5.58)$$

where Ω is the RAAN, $u = \omega + \nu$ is the argument of latitude (the sum of the argument of perigee ω and true anomaly ν), and θ_{GAST} is the Greenwich Apparent Sidereal Time (GAST) angle at that epoch, and is required to get a reference from the vernal equinox and the Greenwich meridian

The target is then selected as the candidate k minimising the angular separation to the LUCH satellite's longitude λ_{LUCH} :

$$k^* = \arg \min_k \Delta \lambda_k, \quad (5.59)$$

where

$$\Delta \lambda_k = \min(|\lambda_{\text{LUCH}} - \lambda_k|, 360^\circ - |\lambda_{\text{LUCH}} - \lambda_k|), \quad (5.60)$$

with the wrapped form of $\Delta \lambda_k$ in Equation 5.60 accounting for the discontinuity at $\pm 180^\circ$. Furthermore, a maximum separation of $\Delta \lambda_{\text{man}}$ of 1.0° is used, if no target below this threshold value is found, the episode is neglected. This algorithm produced the selected targets in Figure 4.1 and Figure 4.5, which matches the open source information [4].

For each identified eavesdropping period, the four manoeuvres preceding its start are identified by examining SMA evolution before the return to GEO SMA. A burn is defined as a continuous increase or decrease in SMA, and burns are identified as the period between stable SMA plateaus, or in other words the period between the times where the SMA is not changing. A plateau is determined by checking the difference between subsequent samples:

$$|a(t_{i+1}) - a(t_i)| < \Delta a_{\text{plateau}}, \quad (5.61)$$

where a sample-to-sample difference satisfying Equation 5.61 is classified as stable, and $\Delta a_{\text{plateau}}$ is a threshold set to distinguish genuine SMA change away from stability and orbit-determination noise deviation from stability. It is set to 1 km. Consecutive stable samples are grouped into a single plateau. Furthermore, a condition that a plateau must last at least 6 hours is implemented to distinguish between TLEs with shorter update frequency, which might fall under the $\Delta a_{\text{plateau}}$ threshold.

The burns are thus defined as the periods between stable SMA plateaus. However, a burn may be split into separate sub-burns if a reversal in the direction of SMA change occurs during the window.

At the start of a burn period, no trend is assumed. Once the SMA has moved away from the initial value by at least a threshold δ_{min} , a trend is established as either rising or falling:

$$\text{trend} = \begin{cases} +1 \text{ (rising)}, & a_i - a_0 \geq \delta_{\text{min}} \\ -1 \text{ (falling)}, & a_i - a_0 \leq -\delta_{\text{min}} \end{cases} \quad (5.62)$$

where a_i is the SMA at sample i and a_0 is the SMA at the start of the burn window.

For the remainder of the burn, the extreme a_{ext} of the SMA is tracked, which is the maximum value reached so far during a rising trend, or the minimum during a falling trend. Each new sample is compared against this extreme. If the SMA falls back from a_{ext} by at least δ_{min} , opposite to the current trend direction, the trend is deemed reversed and the burn is split at the extreme sample:

Following a confirmed reversal, the trend flips and the running extreme resets to the current sample, so the process can detect further reversals within the same burn window. This threshold-based detection prevents single-sample noise from being registered as a reversal.

Examples of the burn clustering algorithm can be found in Figure 5.16.

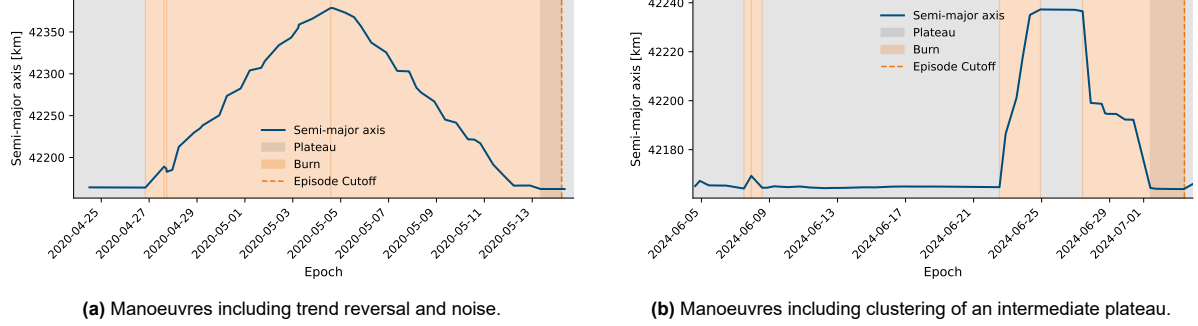


Figure 5.16: Manoeuvre clustering examples of LUCH.

Whilst the examples in Figure 5.16 show that the general algorithm functions as intended, they also highlight its limitations. In Figure 5.16a, the reversal mechanism correctly splits the manoeuvre at the peak of the SMA excursion. However, either due to noise or another phenomenon, a further trend reversal is found early during the subsequent increase in SMA, despite the SMA continuing to increase afterwards as well. In Figure 5.16b, the algorithm correctly identifies the major SMA excursions and the plateau during which the maximum drift rate is achieved; it also shows that the intermediate plateau is clustered with the rest of the SMA reduction, since it exhibits a slight downward trend of its own. Having the last burn split at the beginning and end of the plateau could potentially give an indication of the intent already at the plateau itself, but now it is neglected for the input.

General Algorithm

The general algorithm works differently from the LUCH specific one. Instead of finding the target dynamically, it needs to have chaser and target specified. For a specified time frame, the close approaches between the target and chaser are identified using the `find_peaks` function from `SciPy` [119], applied to the negative of the relative distance so that local minima of $d(t)$ are returned as peaks:

$$t_i \in \{t : -d(t) \text{ is a local maximum}\}, \quad i = 1, \dots, N_{\text{approach}} \quad (5.63)$$

where $d(t)$ is the chaser-target relative distance as a function of time and N_{approach} is the number of candidate close approaches found. For intents where multiple close approaches are possible, such as *Inspection*, all identified minima t_i are kept. For intents where only a single close approach is expected, such as *Docking*, only the global minimum is kept:

$$t^* = \arg \min_{t_i} d(t_i) \quad (5.64)$$

The candidate close approaches are further filtered by the minimum distance achieved, $d_{\text{min}} = d(t_i)$. A candidate is retained only if it falls below an intent-specific threshold d_{thresh} :

$$d_{\text{min}} \leq d_{\text{thresh}} \quad (5.65)$$

For *Docking*, $d_{\text{thresh}} = 1$ km, whilst for *Inspection* it is set to $d_{\text{thresh}} = 500$ km in GEO, and $d_{\text{thresh}} = 50$ km in LEO. This filtering step is applied to exclude cases where a given intent was reported but no correspondingly close approach is observed in the publicly available chaser and target data.

For each close approach, all TLEs leading up to the close approach are considered when scanning for manoeuvres. Manoeuvre detection is performed by propagating the state of a given TLE forward to the epoch of the subsequent TLE. At this common epoch, the Euclidean distance between the propagated state and the state given by the subsequent TLE is evaluated:

$$\Delta r_k = \|\mathbf{r}_{k \rightarrow k+1} - \mathbf{r}_{k+1}\| \quad (5.66)$$

where $\mathbf{r}_{k \rightarrow k+1}$ is the position obtained by propagating TLE k to the epoch of TLE $k + 1$, and $\mathbf{r}_{k+1}$ is the position given directly by TLE $k + 1$ at that epoch. A manoeuvre is deemed to have occurred between TLEs k and $k + 1$ if this distance exceeds a threshold value Δr_{thresh} :

$$\Delta r_k > \Delta r_{\text{thresh}} \quad (5.67)$$

The threshold is set to $\Delta r_{\text{thresh}} = 1 \text{ km}$.

For this TLE propagation, the SGP4 python library is used [101], which in turn is based on [102].

The minima identification logic and subsequent manoeuvre scan are shown in Figure 5.17 and Figure 5.18.

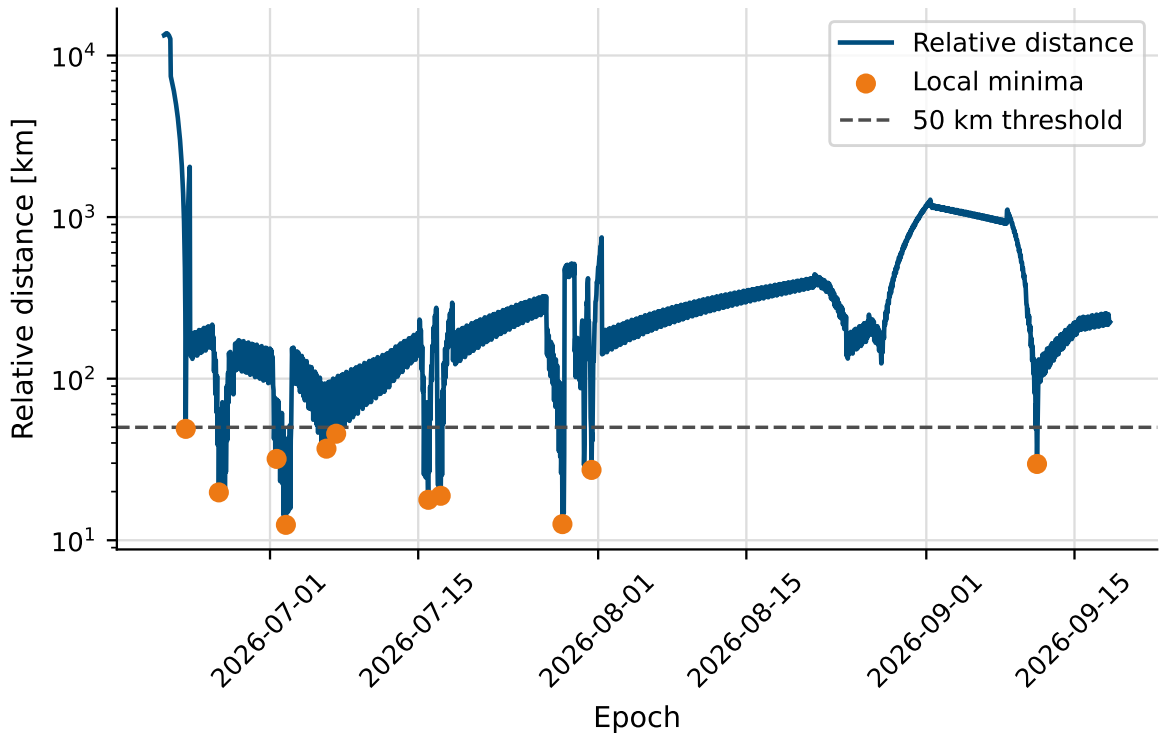


Figure 5.17: Minima identification for Victus Haze and Jackal inspections.

No RPO scenarios

The No RPO scenarios are generated using the same cases as the other intents, but replaces the target's states by that of a non-target. Non-targets are selected as those that are 'close' to the real target. In the case of Luch, the logic Equation 5.59 is used. Instead of taking the target that is closest, the second closest one from the list is taken that still satisfies $\Delta \lambda_{\text{man}}$.

For the general algorithm, a different approach is used to identify replacement targets. Candidate satellites are queried from the database based on an offset in inclination and SMA relative to the reference target:

$$a \in [a_0 - \Delta a, a_0 + \Delta a], \quad i \in [i_0 - \Delta i, i_0 + \Delta i] \quad (5.68)$$

where a_0 and i_0 are the SMA and inclination of the reference target at the epoch of interest, and Δa , Δi are search margins. If no candidates are returned, the margins are doubled and the query is repeated

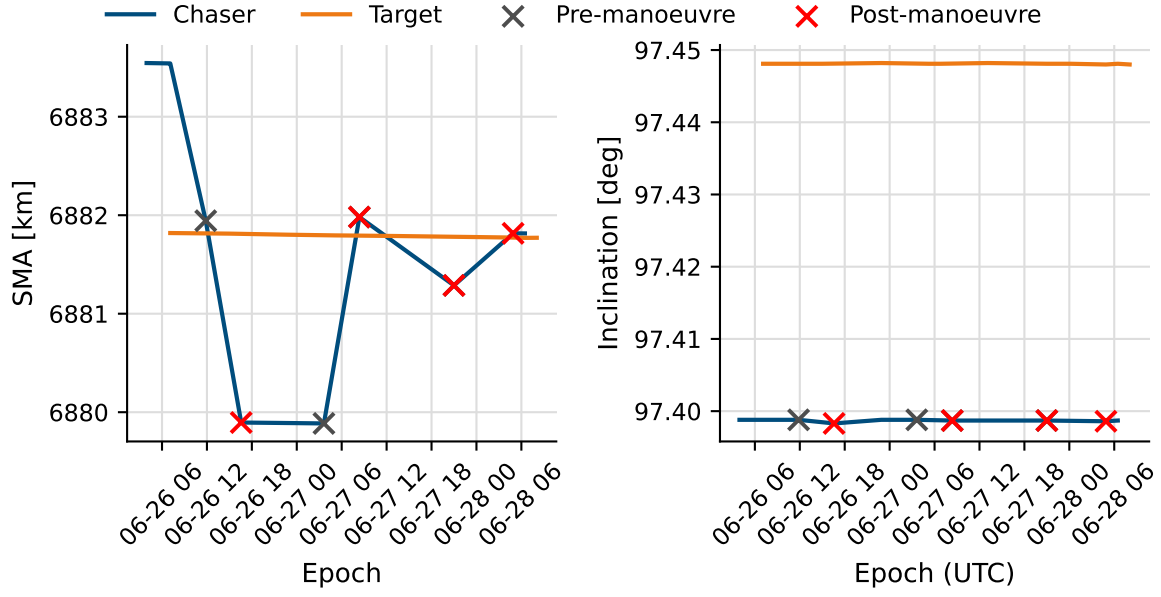


Figure 5.18: Manoeuvres detection leading up to Victus Haze and Jackal inspections.

until at least one candidate is found. If more than 20 candidates remain, the set is further narrowed by longitude proximity to the reference target, keeping only the 20 candidates with the smallest angular longitude difference, similar to Equation 5.58.

For each remaining candidate, its Keplerian state at the epoch of closest approach is compared directly against that of the reference target. Both states are obtained by converting the Cartesian position at that epoch into Keplerian elements, $[a, e, i, \Omega, \omega, \theta]$, where angular differences are wrapped in their respective range.

The six resulting differences are then normalised by their own standard deviation across all candidates, so that elements with different scales and spreads contribute comparably to the overall distance metric:

$$\widehat{\Delta x_j} = \frac{\Delta x_j}{\sigma(\Delta x_j)}, \quad j \in \{a, e, i, \Omega, \omega, \theta\} \quad (5.69)$$

An additional weight w_a is applied to the normalised SMA difference to emphasise its contribution relative to the other elements. The candidates are then ranked by the weighted Euclidean norm of the normalised differences:

$$D = \sqrt{(w_a \widehat{\Delta a})^2 + \widehat{\Delta e}^2 + \widehat{\Delta i}^2 + \widehat{\Delta \Omega}^2 + \widehat{\Delta \omega}^2 + \widehat{\Delta \theta}^2} \quad (5.70)$$

and the n candidates minimising D are selected as replacement targets. Depending on the case, n is set to 2 or 5.

5.5.2. Overview of Real World Cases

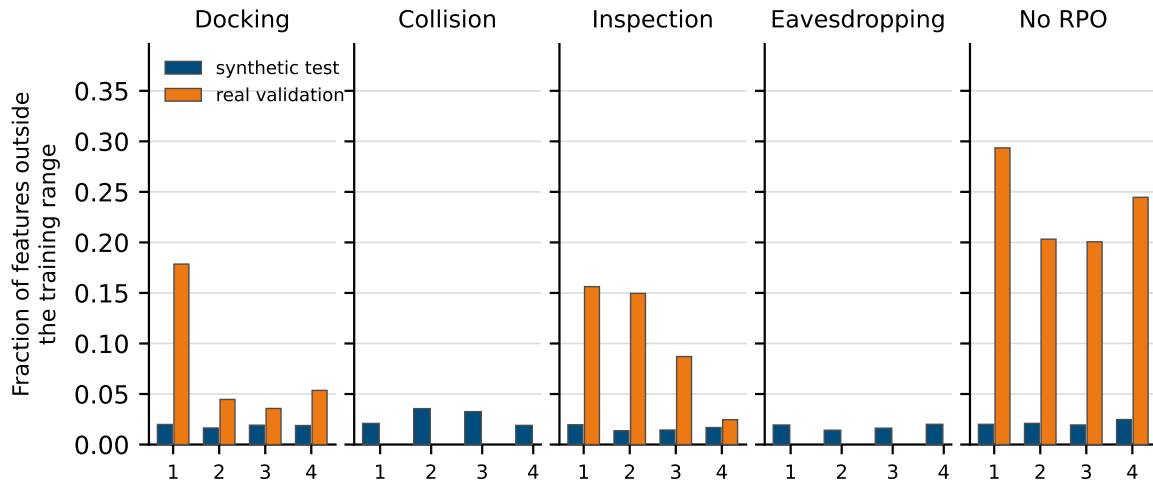
Real world cases considered for the algorithms are covered in Chapter 4, and put through their respective algorithms to extract manoeuvres before RPO. This resulted in a total of 122 scenarios in GEO, and 259 in LEO. The breakdown of the intents per chaser is shown in Table 5.6, whilst more information on each case can also be found in Chapter 7.

5.6. Comparison Synthetic and Real Datasets

Model training is performed on synthetic data, which means that for the model to perform in real-world operational scenarios, the synthetic data needs to match real-world data. The comparison of these two sets is done in two ways: distribution analysis, in which it is evaluated whether the raw inputs of the real-world data fall within the distribution of synthetic data, per input itself; and a clustering method,

Table 5.6: Source breakdown per intent, split by orbital regime

Intent	GEO	LEO
Docking	SJ-21 (2), MEV-1 (1), MEV-2 (1)	ASTRO 4 (1), Crew Dragon Endurance (1), Progress MS-27 (1), Shenzhou-8 (1), Shenzhou-9 (1), Shenzhou-10 (1), Shenzhou-11 (1), Soyuz MS-23 (1)
Collision	–	–
Inspection	USA270/GSSAP-3 (8), SJ-17 (6), USA271/GSSAP-4 (6), USA324/GSSAP-5 (2)	Cosmos 2521 (21), Cosmos 2504 (6), Victus Haze (5)
Eavesdropping	LUCH Olymp K-1 (29), LUCH Olymp K-2 (8)	–
No RPO	LUCH Olymp K-1 (22), MEV-2 (18), LUCH Olymp K-2 (6), USA271/GSSAP-4 (6), USA324/GSSAP-5 (3), MEV-1 (2), USA270/GSSAP-3 (2)	Cosmos 2521 (168), Victus Haze (45), Cosmos 2504 (6)

**Figure 5.19:** Out-of-Distribution analysis of real and synthetic test sets, compared to LEO training data.

which uses `scikit-learn`'s `NearestNeighbors` [120] per intent to investigate whether real-world intents cluster near synthetic-world intents.

This comparison is not a direct validation of the synthetic data generation, as the real-world manoeuvre extraction adds a layer of potential misalignment between synthetic and real data.

The features considered are the ones from Chapter 7 which have undergone noise and cadence degradation. Further explanation on the features can also be found in Chapter 6.

5.6.1. Distribution Analysis

For the out-of-distribution (OOD) analysis, the 1%–99% range of values is considered for the baseline feature range. The expected fraction of values falling OOD for a perfect match is thus 2%, whilst higher fractions indicate a shift between the training data and the set under comparison.

The per-intent, per-manoeuvre split is presented in Figure 5.19 for LEO and Figure 5.20 for GEO, showing OOD for both the synthetic test set and the real validation set.

Three patterns emerge from Figure 5.19 and Figure 5.20. First, the overall match between real and synthetic data is domain-dependent and not uniformly better or worse for one domain over the other, as it depends on the intent. *No RPO*, for example, has a similar OOD score to synthetic data in GEO, but in LEO this is not the case. This can potentially be explained by the difference in how *No RPO* targets are

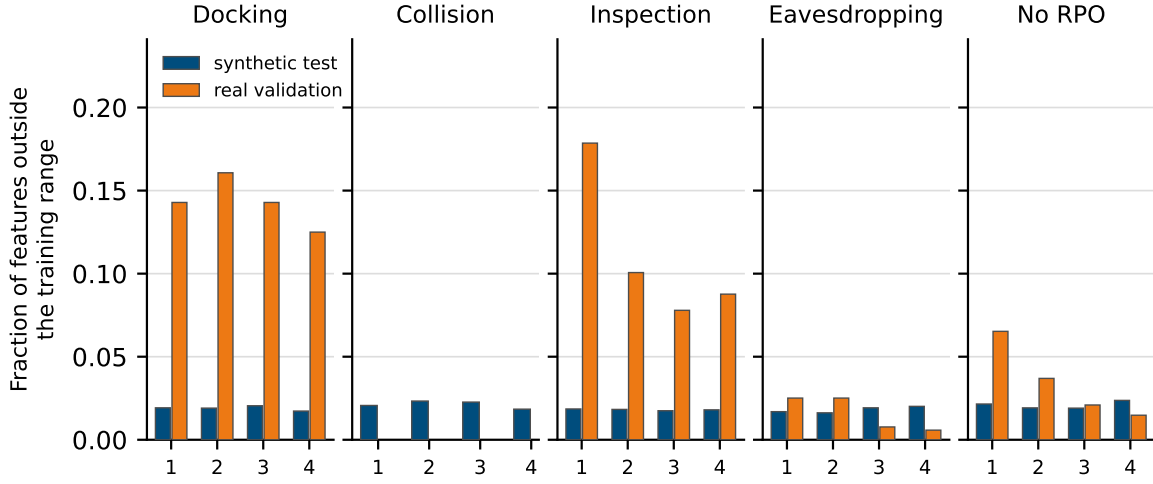


Figure 5.20: Out-of-Distribution analysis of real and synthetic test sets, compared to GEO training data.

generated between synthetic and real data: for synthetic data the replacement target is selected based on similarity with the real target at the start of the simulation, whereas for real data this comparison is made at closest approach by the chaser. The difference between these two approaches is potentially greater in LEO than in GEO, where the latter environment most likely has a smaller difference due to less variance from perturbations and generally similar orbits, being all within the GEO belt. *Docking*, by contrast, matches somewhat worse in GEO than in LEO, though the GEO real sample for this intent is very thin ($n = 4$) and this should be read as indicative rather than confirmed.

Moreover, later manoeuvres are a better match than earlier ones. In LEO this is visible for *Docking* and *Inspection*; in GEO for *Inspection*, *Eavesdropping*, and *No RPO*. The two exceptions are *No RPO* in LEO, discussed before, and *Docking* in GEO, where no comparable improvement across the manoeuvre sequence is observed. For this latter scenario, no definite conclusions can be made since it is only supported by four data points.

There is thus a genuine, intent- and domain-dependent shift between real and synthetic data. Earlier manoeuvres show a greater mismatch than later ones in most but not all intent/domain combinations, and *Eavesdropping* has the best simulated match with real data.

5.6.2. Clustering Analysis

Whilst an OOD score does give an indication of whether the data generation matches reality, it does not give the full picture. An intent can have a high OOD score, meaning there is a mismatch for the features, but it could still match the patterns or combination of features for its intent. Analysis of a sample's nearest neighbours within the training data reveals whether the underlying patterns or combination of features matches what the model is trained on, and could potentially indicate that whilst features are OOD, the pattern underlying them does match the data generation. For each episode, 25 of its nearest neighbours are determined. The mean composition of these neighbours' intents is presented in Figure 5.21 and Figure 5.22, showing how both the synthetic test data and the real data cluster with the training data.

For the clustering in both LEO and GEO, *Docking* stands out as having more *Docking* neighbours for real data than synthetic, meaning the real data points are more easily discernible than their synthetic counterparts. For *Inspection*, the clustering story differs between LEO and GEO. Whilst both domains had similar OOD scores, the clustering in LEO shows close proximity to *Docking* neighbours, which is also significantly greater than for the synthetic set. In GEO, on the other hand, there is significant clustering with *Inspection*, *Eavesdropping*, and *No RPO*, indicating it does not clearly fit into the classes of synthetic data. *Eavesdropping* has almost identical clustering between real and synthetic data, so not only is the OOD score similar, but the underlying patterns are as well, and good classification performance is expected. The *No RPO* clustering trend in LEO and GEO is reversed: whilst targets were better discernible for GEO synthetic data, they are more often clustered closer to other intents

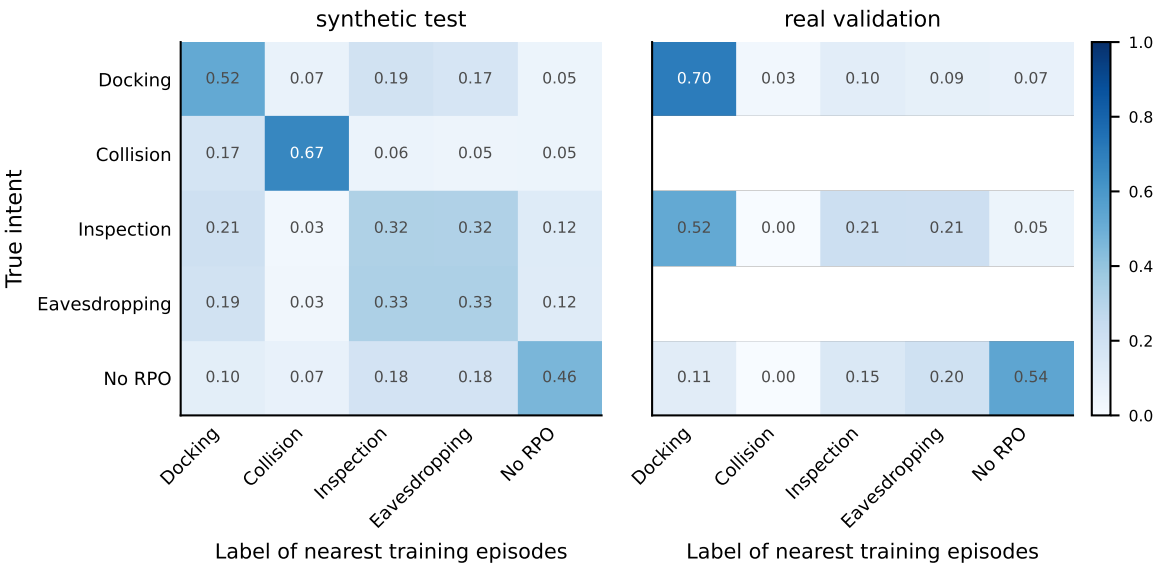


Figure 5.21: Neighbourhood distribution of real and synthetic test sets, compared to LEO training data.

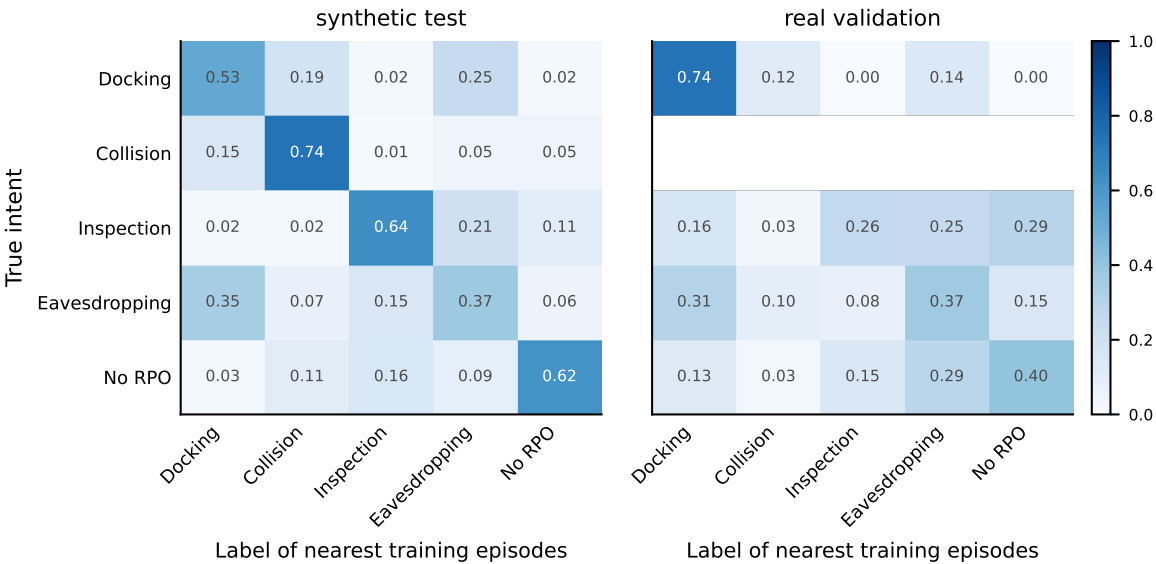


Figure 5.22: Neighbourhood distribution of real and synthetic test sets, compared to GEO training data.

in real data; in LEO, conversely, the model is trained on targets that are less discernible than the real data.

Taken together, the OOD and clustering results point to *Inspection* as the intent where the model is most likely to struggle in operation, in both LEO and GEO, though for different underlying reasons in each domain (further explored in Chapter 7). *No RPO* is a secondary concern, but only in LEO, where both its OOD mismatch and its clustering with other classes are worse than in GEO. *Docking* and *Eavesdropping* appear to transfer well, though the former is based on a very thin real sample and should be treated cautiously. More broadly, since the mismatch between real and synthetic data is largest for early manoeuvres and shrinks over the course of a campaign, classification confidence is expected to be least reliable when only the first one or two manoeuvres are observed.

"I'm sorry, Dave. I'm afraid I can't do that."

HAL 9000, *2001: A Space Odyssey*
- 1968

6

Intent Classification Model

This chapter documents the model used for intent classification. Section 6.1 details the reasoning and structure of the model inputs. Then, Section 6.2 explains the model architectures and their training. Finally, Section 6.3 covers the classification performance of the models.

The process behind this chapter was highly iterative with not only sections within this chapter, but also performance on real world data with the degraded training data presented in Chapter 7. In short, the best performing model is XGBoost, as determined in Section 6.3. The best input features for this XGBoost model are Keplerian deltas with a distance metric, as determined in Section 6.1 which in turn relies on real world data generated in Section 5.5 and further details on these real world cases is available in Section 7.3. Whenever it is not specified, it is these inputs and model that is being evaluated.

6.1. Input Data and Feature Engineering

In the literature, intent inference is typically treated as a time-series classification problem (see Chapter 2). This approach follows naturally from the classification method most commonly adopted, in which the relative trajectory of the chaser around the target is analysed throughout the execution of the intent. Whilst this offers the richest source of information for inferring intent, it is fundamentally reactive: intent can only be inferred once the relative motion that reveals it has already taken place. Early intent inference, by contrast, must begin before this execution phase. In other words, intent inference must come from data that occurs during the approach trajectory that precedes the execution of intent.

The question that follows is how such an approach trajectory should be represented. Treating it as a time-series problem, in line with the existing literature, remains possible; however, the information density of the time series itself is likely to be sparse during this phase. While a satellite is under natural motion, with no manoeuvring taking place, there is little in the trajectory itself to indicate intent. It is only the execution of a manoeuvre and the change in the trajectory that follows that provides direct evidence of what the intent might be. This suggests that a representation built around discrete manoeuvre events, rather than continuous trajectory evolution, may be better suited to the goal of early inference.

For this reason, the initial models are built on the information available at the manoeuvres executed during the approach. As Goldman, Geib, and Miller [121] explain, plan recognition is the field of study concerned with inferring the underlying goal behind an agent's actions from the actions themselves; here, the relevant actions are the manoeuvres.

6.1.1. Base Feature Input

The input data is constructed from instantaneous pre- and post-manoeuvre state information at the exact time of each manoeuvre. Two options are considered, the pure Keplerian elements ($a, e, i, \Omega, \omega, v$) of both the target and chaser satellites, and the difference (delta) between these elements. With pure Keplerian elements, there are four feature sets per manoeuvre; for both target and chaser the elements pre- and post-manoeuvre. For Keplerian deltas, this reduces to two feature sets, the difference between target and chaser, pre- and post manoeuvre. Both options have their potential advantages and disadvantages. With the pure Keplerian elements, a model has the most information about the situation. It could, in theory, derive all the Keplerian deltas from this information, and retain the context around them. For example, whilst a 1 km difference in the SMA (a) is indicative of a drift between the two objects, the relative drift rate will depend also on the actual SMAs of the two objects. However, the use of Keplerian deltas has the advantage that the model has the pre- and post-manoeuvre differences directly, and thus it is easier for the model to learn the effect of the manoeuvre.

The accuracy of a model trained on the pure or delta Keplerian elements is presented in Figure 6.1. Accuracy is shown on the synthetic data set (Figure 6.1a), and real data (Figure 6.1b). This comparison is intended only to motivate the choice of base input, ahead of the model and training details discussed in Section 6.3 and Chapter 7, and the accuracy does not reflect final model performance.

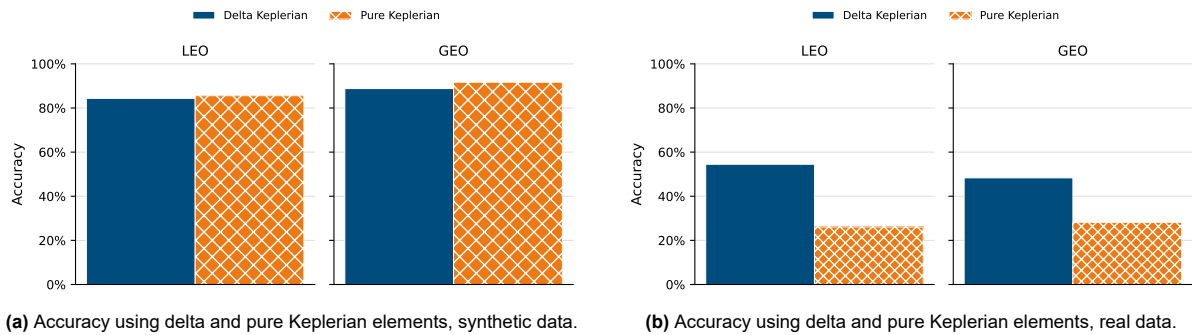


Figure 6.1: Classification performance of the model using delta and pure Keplerian elements.

As can be observed in Figure 6.1a, the model performs well regardless of which data type is used as input, when evaluated on synthetic data. There is a marginal difference between the two, with pure Keplerian elements scoring slightly better in both LEO and GEO. However, the real change comes when evaluated on real data. Here, deltas perform better, with a significant difference in accuracy between the two in favour of using the Keplerian deltas.

Despite showing marginally better performance on synthetic data, pure Keplerian elements struggle to generalise to real-world cases. This suggests that pre-processing the data into delta Keplerian elements is beneficial for the model, as it focuses training on the relative geometry of the approach rather than the absolute orbital context. For this reason, delta Keplerian elements are used as the base input for the models. However, the deltas by themselves also do not perform perfectly and there is room for improvement with additional features, derived from the orbital state.

6.1.2. Derived Feature Input

Several engineered features are considered to aid the model with intent inference to improve the performance of using just the Keplerian deltas. These fall into four categories: 1. distance, 2. plane alignment, 3. relative drift, 4. potential required velocity change. The four categories potentially reveal more about how the RPO is occurring; some intents might happen closer by or further away, in which case the category 1 is of use. Category 2 says something about the relative geometry of the RPO, which can be a differentiator between intents. Both categories 3 and 4 say something about if an RPO is actually being performed, but through different metrics. With drift rate the model potentially learns patterns in the timing of when an RPO is supposed to happen, whilst with required velocity change it can learn about the burn patterns that might lead to an RPO.

Distance is the simplest metric and is the distance between the chaser and target satellite at certain points in time, which in this case corresponds to the manoeuvres.

The metrics related to plane alignment give an indication of the match between the general geometry of the two satellites' orbits. What they do not indicate is phasing. The orbital plane is defined through the angular momentum vector, which is calculated as

$$\vec{h} = \vec{r} \times \vec{v}. \quad (6.1)$$

The angle between two vectors is then determined as

$$\theta_h = \arccos \left(\frac{\|\vec{h}_c\| \cdot \|\vec{h}_t\|}{\|\vec{h}_c\| \|\vec{h}_t\|} \right). \quad (6.2)$$

Both the angles pre- and post-manoeuve are considered, as well as the change in the angle itself.

For phasing, three metrics are considered: the in-plane angle difference, which assumes the orbital planes of the chaser and target are aligned, the drift rate, and the time-to-alignment. The mean longitude (u) is calculated as

$$u = \Omega + \omega + M. \quad (6.3)$$

It is thus determined using mean anomaly (M), and not true anomaly (ν) to make use of the mean motion to calculate relative drift rate and time of alignment. The difference is then

$$u_d = u_c - u_t. \quad (6.4)$$

The relative drift rate (du/dt , or $\dot{u}$) is then calculated using the mean drift rates

$$\dot{u} = \sqrt{\frac{\mu}{a_c^3}} - \sqrt{\frac{\mu}{a_t^3}}, \quad (6.5)$$

where μ is the gravitational parameter. For the time-to-alignment, the synodic period is first calculated. This is the period between alignments and depends on the drift rate $\dot{u}$. It is defined as

$$T_{syn} = \frac{2\pi}{\dot{u}}. \quad (6.6)$$

This is used for the time-to-alignment:

$$T_{align} = \frac{-u_d}{\dot{u}} \bmod T_{syn}. \quad (6.7)$$

For the potential required delta-v metrics, the Lambert solver from Chapter 5 is re-used to evaluate the pre- and post-manoeuve states. This metric is similar to some Threat Index papers (see Chapter 2); however, for computational reasons, only one solution is considered, rather than a bounded set of potential combinations of time-of-flight and ΔV . A time-of-flight is required for the Lambert solver and is set to a quarter of the orbital period of the chaser satellite,

$$tof_\lambda = \frac{\pi}{2} \sqrt{\frac{a_c^3}{\mu}}. \quad (6.8)$$

In addition to the Lambert-based transfer cost, a lower-bound benchmark is computed using a two-impulse Hohmann-type transfer between the two orbits, ignoring phasing entirely. This isolates the geometric (non-phasing) contribution to the manoeuvre cost, since it represents the delta-v floor that the true rendezvous cost approaches given unlimited time to phase. For circular orbits with semi-major axes a_t and a_c , and plane angle θ between the two orbital planes, the transfer orbit semi-major axis is

$$a_{trans} = \frac{a_t + a_c}{2}. \quad (6.9)$$

The circular and transfer-orbit velocities at each end of the transfer are then

$$v_t = \sqrt{\frac{\mu}{a_t}}, \quad v_c = \sqrt{\frac{\mu}{a_c}}, \quad (6.10)$$

$$v_{trans,t} = \sqrt{\mu \left(\frac{2}{a_t} - \frac{1}{a_{trans}} \right)}, \quad v_{trans,c} = \sqrt{\mu \left(\frac{2}{a_c} - \frac{1}{a_{trans}} \right)}. \quad (6.11)$$

The first burn injects the chaser onto the transfer orbit and is assumed to be coplanar, so its cost is simply

$$\Delta V_1 = |v_{trans,t} - v_t|. \quad (6.12)$$

The second burn both circularises the transfer orbit and folds in the full plane change θ , combined vectorially as

$$\Delta V_2 = \sqrt{v_{trans,c}^2 + v_c^2 - 2 v_{trans,c} v_c \cos \theta}. \quad (6.13)$$

The total floor cost is then $\Delta V_f = \Delta V_1 + \Delta V_2$. In the case where the target and chaser are at the same semi-major axis a , the cost is defined as

$$\Delta V_f = 2V \sin\left(\frac{\theta}{2}\right), \quad V = \sqrt{\frac{\mu}{a}}. \quad (6.14)$$

Comparing this floor value to the delta-v obtained from the fixed-time-of-flight Lambert solution gives a measure of how much of the Lambert cost is attributable to phasing constraints rather than to the underlying orbital geometry. However, this does assume the target spacecraft is in a near circular orbit.

A summary of these considered features and their definitions is presented in Table 6.1.

Table 6.1: Summary of engineered features considered for RPO intent classification.

Symbol	Name	Definition	Units
d	Distance	$d = \ \mathbf{r}_c - \mathbf{r}_t\ $	m
$\theta_{h,pre}$	Angular momentum angle, pre	angle between $\mathbf{h}_{c,pre}$ and $\mathbf{h}_{t,pre}$	rad
$\theta_{h,post}$	Angular momentum angle, post	angle between $\mathbf{h}_{c,post}$ and $\mathbf{h}_{t,post}$	rad
$\Delta\theta_h$	Change in angular momentum angle	$\theta_{h,post} - \theta_{h,pre}$	rad
u	In-plane angle	argument of latitude difference, chaser minus target	rad
$\dot{u}$	Drift rate	du/dt	rad/s
T_{align}	Time to alignment	$T_{align} = (-u/\dot{u}) \bmod T_{syn}$, the smallest $t \geq 0$ at which $u(t) = 0$	s
$\Delta V_{\lambda,pre}$	Lambert ΔV , pre	$F_{\lambda}(\mathbf{r}_c, \mathbf{r}_t, \text{ToF})$	m/s
$\Delta V_{\lambda,post}$	Lambert ΔV , post	$F_{\lambda}(\mathbf{r}_{c,post}, \mathbf{r}_t, \text{ToF})$	m/s
$\Delta V_{f,pre}$	ΔV floor, pre	$F(\mathbf{r}_c, \mathbf{r}_t)$	m/s
$\Delta V_{f,post}$	ΔV floor, post	$F(\mathbf{r}_{c,post}, \mathbf{r}_t)$	m/s
ΔV_{red}	ΔV reduction	$\Delta V_{\lambda,pre} - \Delta V_{\lambda,post}$	m/s
$R_{\Delta V}$	Excess ratio, post	$\Delta V_{\lambda,post} / \Delta V_{f,post}$	—

For the first evaluation of the metrics, the influence of distance is evaluated stand-alone since preliminary testing showed the feature to have good potential to improve performance. A model trained on only Delta Keplerian features and one with the *distance* feature added at each manoeuvre is evaluated in Figure 6.2.

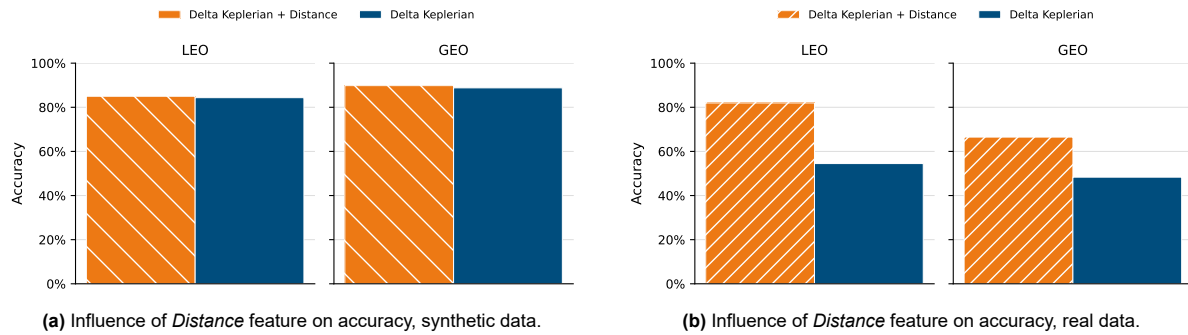


Figure 6.2: Influence of distance on classification performance of the model.

Both models perform well on synthetic data, with the addition of *distance* having negligible effect (under 1%) on the evaluation accuracy. The real improvement comes from the evaluation on real data, where both LEO and GEO accuracy improve significantly. The distance feature is thus considered for base model input. Note that the performance in LEO on real data is similar to that of synthetic; this is slightly misleading due to the real data set being fundamentally different and further information can be found in Chapter 7.

The other features, from groups 2 to 4, are evaluated together against the baseline. Three models are compared in Figure 6.3: the base model, trained on the delta Keplerian features (no distance metric), the candidate addition, trained only on the new features from groups 2 to 4 (New Features), and a model combining both feature sets (Combined).

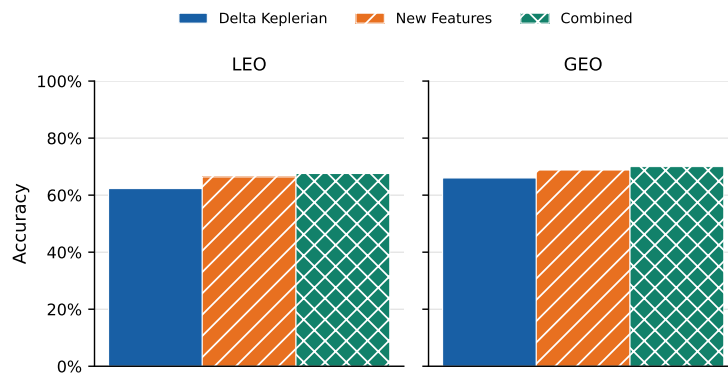


Figure 6.3: Model comparison on different training feature sets, synthetic data.

It can be observed in Figure 6.3 that all models perform well regardless of the feature set they are trained on. Notably, the new feature set on its own already forms a complete basis for intent inference, slightly outperforming the base Keplerian deltas. Combining the two sets yields the best performance overall, as would be expected given that each set carries information the other lacks. This improvement, however, comes with some redundancy between the two sets, so rather than appending the new feature set in full, a selection of it is added to the base set to avoid redundant input parameters.

For this, the importance of the features is investigated. Feature importance is assessed using `scikit-learn`'s `permutation_importance` [120] function. Permutation importance measures the effect a feature has on performance by randomly shuffling that feature within a given dataset.

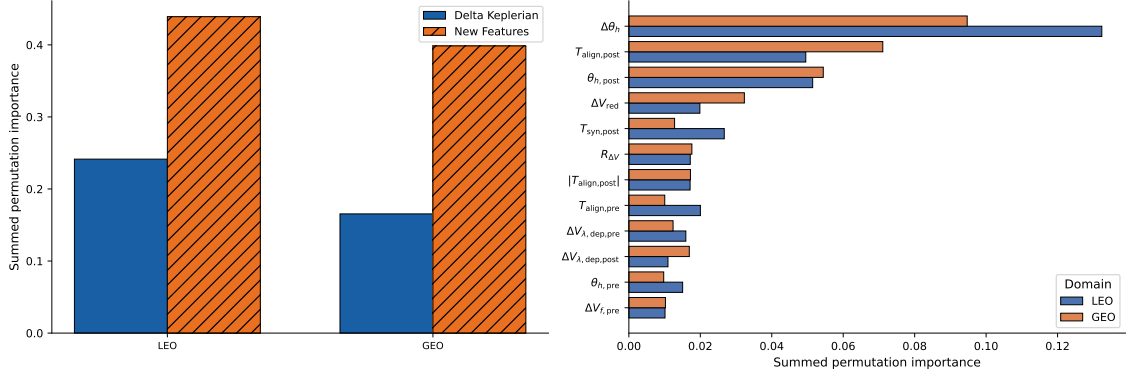


Figure 6.4: Feature permutation importance of combined model, summed per set and top 12 of new features

Figure 6.4 shows the summed feature importance for the base and new feature sets, for the combined model. It can be observed that the performance of the combined model is affected more by the permutation of the new features than the base features. It should be noted, however, that the new feature set has more features in it than the base set, so some individual features from the base set can be more important than those from the new set.

From the new feature set, the most important features are $\Delta\theta_h$, $\theta_{h,post}$, $T_{align,post}$ and ΔV_{red} , as can be observed in Figure 6.4. There is some difference between LEO and GEO, with the angular momentum angle change being much more important in LEO than in GEO, which most likely comes from the planar alignment of most satellites in the GEO belt, reducing the signal that could potentially be given. In turn, ΔV_{red} is more important in GEO, which is potentially an indication of the phasing difference between the two domains.

It should be noted that the most easily derived of these features are $\Delta\theta_h$, $\theta_{h,post}$, and $T_{align,post}$. For this reason, and due to their permutation importance, they are considered for a new model. Furthermore, $\theta_{h,pre}$ and $T_{align,pre}$ are also considered as they are related to these features. This selected feature set is evaluated in Figure 6.5, where it is compared to the Keplerian deltas with distance on real data.

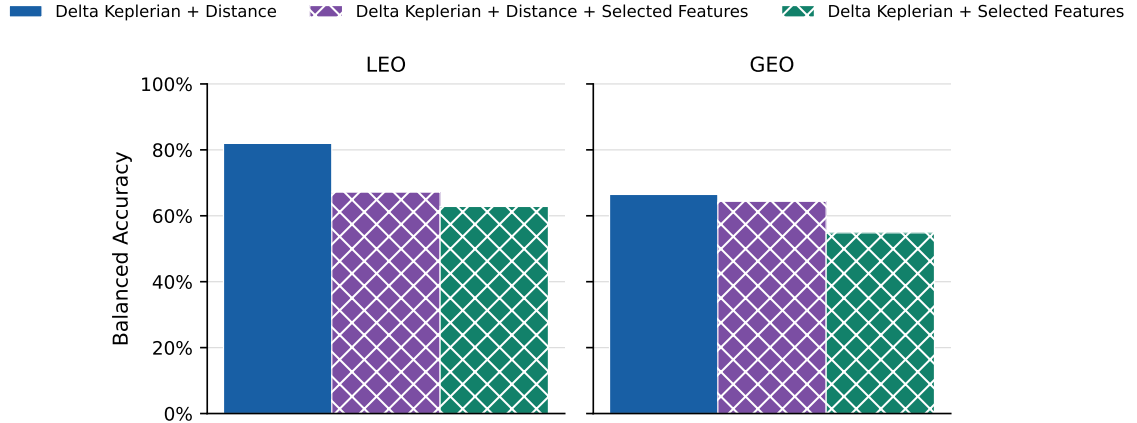


Figure 6.5: Model comparison on different training feature sets, real data.

The selected feature set combined with the delta Keplerians performs the worst on real data of all the feature sets considered in Figure 6.5, though there is an improvement over the delta Keplerians alone in Figure 6.2b. The distance metric improves performance more than the selected features alone, but notably the set without the selected features but only the distance metric performs the best in LEO and GEO. In LEO the difference is significantly more than in GEO, and is potentially driven by the greater mismatch of synthetic and real features in LEO than in GEO as shown in Section 5.6.

The final input feature set is thus the Keplerian deltas with the distance metric at each manoeuvre.

6.1.3. Training Input

The training input sequence per sample is presented in Figure 6.6 and Figure 6.7. The full input sequence, shown in Figure 6.6, reflects the features found to be best for intent inference. To allow early intent inference, some or all of the later manoeuvres are omitted from the sequence, allowing intent to be inferred before all four manoeuvres have taken place. Figure 6.7 illustrates such a partial input sequence, in which one or more manoeuvres are absent and the corresponding feature entries are zero-valued.

Data Input per case							
Manoeuvre 1		Manoeuvre 2		Manoeuvre 3		Manoeuvre 4	
pre	post	pre	post	pre	post	pre	post
Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}
Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}
Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}
ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}
$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$
Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}
Distance		Distance		Distance		Distance	

Figure 6.6: Full input sequence considering all four manoeuvres.

Data Input per case - missing manoeuvres							
Manoeuvre 1		Manoeuvre 2		Manoeuvre 3		Manoeuvre 4	
pre	post	pre	post	pre	post	pre	post
Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	0	0	0	0
Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	0	0	0	0
Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	0	0	0	0
ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	ΔQ_{rc}	0	0	0	0
$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	0	0	0	0
Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	0	0	0	0
Distance		Distance		0		0	

Figure 6.7: Partial input sequence considering fewer than four manoeuvres.

To allow the models to distinguish between full and partial input, an additional integer-valued feature indicating the number of manoeuvres present (ranging from 1 to 4) is appended to the input vector. Without this indicator, a genuine absence of manoeuvre data would be numerically indistinguishable from a case in which the Keplerian element deltas happen to be zero, potentially allowing the models to misinterpret missing information as matching orbits. The use of *NaN* values was considered for this purpose, but was rejected in favour of reserving *NaN* exclusively for data that is genuinely unavailable. In other words, zero-padding is used for manoeuvres that have not yet occurred, whereas *NaN* is used for data that is potentially unavailable.

To allow for early intent inference, the model should at least be able to perform accurately on information up till manoeuvre #3. Whilst for the *Inspection* and *Eavesdropping* manoeuvre #4 corresponds to the start of the mission trajectory, for *docking* and *ramming* it represents the end of those intents.

The dataset is balanced per intent class to ensure that the models do not over-represent any single intent during training. Without class balancing, the model is trained predominantly on the majority class, and can minimise its loss simply by predicting that class most of the time, regardless of the input. Within each balanced class, 90% of the samples are used for training and validation, with the remaining 10% held out for testing. The 90% is split further, with 85% used for training and 15% for validation, giving

an overall split of 76.5% training, 13.5% validation, and 10% test. This balancing and split is shown in Figure 6.8a for LEO and Figure 6.8b for GEO.

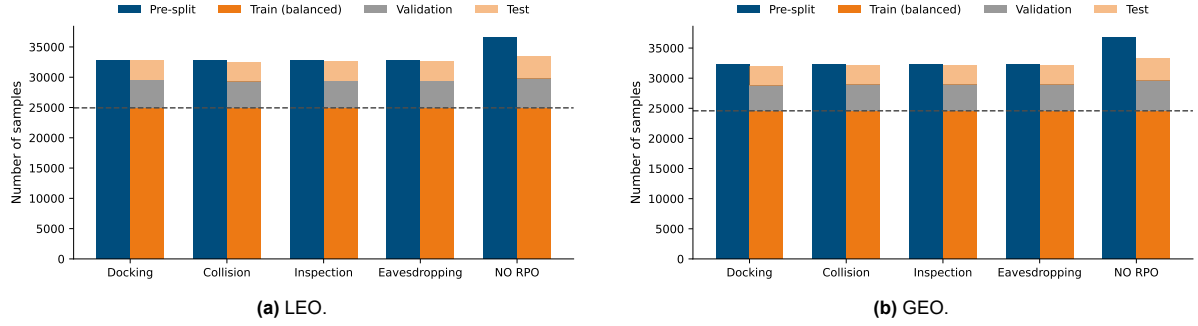


Figure 6.8: Balancing of intent classes and train–test split.

Models are trained separately on their domain, which means there is a LEO and a GEO classifier model.

6.2. Model Architecture

Three supervised classification algorithms were selected as candidates for the intent classification task, all with different approaches: SVM, Random Forest, and XGBoost. This diversity was intentional, as the three approaches make different assumptions about decision boundary structure and feature interactions, allowing their relative performance to indicate which inductive bias best matches the structure of the delta-Keplerian feature space described in Section 6.1. All model implementations were done using the `scikit-learn` library [120].

The SVM constructs an optimal separating hyperplane that maximises the margin between classes in a (possibly transformed) high-dimensional feature space [60]. For linearly inseparable data, the kernel trick implicitly maps input features into a higher-dimensional space via a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, avoiding explicit computation of the transformation. In this work, a linear SVM is used, due to the difficulty of non-linear SVMs with large dataset ($>1,000$ samples) [120]. The regularisation parameter C is tuned to balance margin width against misclassification tolerance. Multi-class classification is handled via a one-versus-rest scheme, in which a separate binary classifier is trained to distinguish each intent class from all others, and the final label is assigned to the class whose classifier produces the highest decision-function score. SVMs are well suited to problems with a clear margin of separation [60].

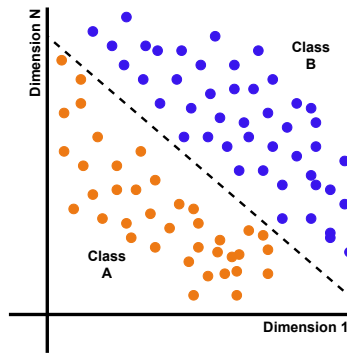


Figure 6.9: Example architecture of an SVM classifier.

Random Forest is a bagging-based ensemble method that trains a large number of decision trees on subsets of the training data [59]. At each split within a tree, only a random subset of features is considered, which decorrelates individual trees and reduces the variance of the ensemble relative to a single decision tree. The final classification is obtained by averaging the predicted class probabilities across all trees and selecting the class with the highest mean probability. Key hyperparameters include the

number of trees, maximum tree depth, and the number of features considered at each split, all of which govern the bias-variance trade-off of the ensemble. Random Forests are attractive for this application because they are relatively insensitive to feature scaling, naturally handle non-linear interactions, and provide a built-in measure of feature importance that can be used to interpret which orbital elements are most discriminative for a given manoeuvre intent.

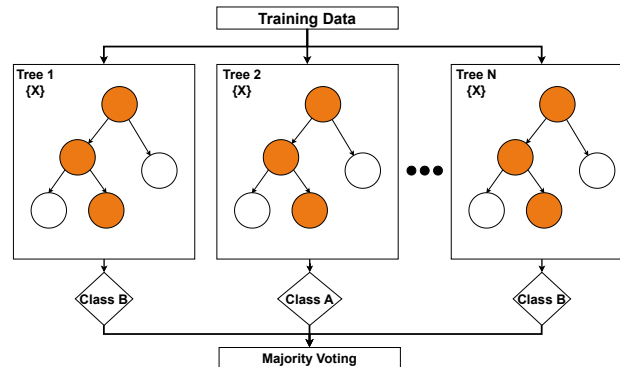


Figure 6.10: Example architecture of a Random Forest classifier.

XGBoost is a gradient boosting algorithm that builds an ensemble of decision trees sequentially, where each new tree is trained to correct the residual errors of the ensemble formed by its predecessors [61]. Unlike Random Forest, in which trees are grown independently and in parallel, boosting is a sequential process in which the loss function is minimised using gradient descent. XGBoost extends standard gradient boosting with regularisation terms on tree complexity, a second-order (Newton) approximation of the loss function, and efficient handling of sparse data, which together tend to improve generalisation and training speed relative to earlier boosting implementations. Relevant hyperparameters include the learning rate, maximum tree depth, number of boosting rounds, and L1/L2 regularisation coefficients. Because boosting focuses successive trees on previously misclassified samples, XGBoost is expected to perform well in distinguishing intent classes with subtle or overlapping feature signatures, such as Inspection versus Eavesdropping sequences.

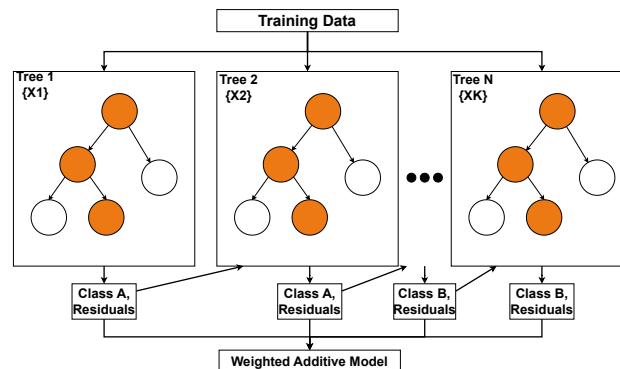


Figure 6.11: Example architecture of an XGBoost classifier.

6.2.1. Hyper-parameter Tuning

For the Random Forest, a few key hyper-parameters are considered for tuning: the number of trees, which sets the ensemble size; the maximum tree depth, which bounds the number of splits a tree may make; the minimum samples required to split a node and to form a leaf, which control how small a subset of data a tree may partition on; and the number of features considered at each split, which governs the randomness injected between trees. For XGBoost, the number of boosting rounds sets how many trees are added sequentially; the learning rate scales each tree's contribution to the ensemble, trading off convergence speed against over-fitting; maximum tree depth and `min_child_weight` constrain individual tree complexity; `subsample` and `colsample_bytree` randomly subsample rows and columns for

each tree to decorrelate the ensemble; `gamma` sets the minimum loss reduction required for a further split; and `reg_alpha/reg_lambda` apply L1/L2 penalties to leaf weights. For the linear SVM, `C` controls the trade-off between maximising the margin and tolerating misclassified training points, with smaller values giving a wider margin at the cost of more misclassification, while the maximum number of solver iterations sets the convergence budget.

Hyper-parameters for all three candidate models are tuned using `RandomizedSearchCV` from `scikit-learn` [120], with 5-fold cross-validation, optimising for mean validation accuracy across folds. Randomised search is preferred over an exhaustive grid search because it samples a fixed number of parameter combinations from specified distributions, allowing a wider hyper-parameter space to be explored within a reasonable computational time. For each model, the best-performing configuration is refit on the full training set and evaluated on the held-out test set.

The Random Forest search space is constrained to reduce the risk of over-fitting. Tree depth is limited to 6–15 rather than allowed to grow unrestricted, the minimum samples required to split a node is set to 15–40, and to form a leaf to 10–30, forcing each leaf to represent a non-trivial subset of the training data rather than fitting to individual samples or noise. The number of trees is varied between 200 and 500, and the number of features considered at each split is toggled between the $\sqrt{n}$ and $\log_2 n$ heuristics, where n is the number of features. Thirty search iterations are used.

XGBoost is tuned over a substantially larger hyper-parameter space, using 30 search iterations, reflecting the greater number of tunable parameters. The number of boosting rounds (400–1200) is combined with early stopping (patience of 30 rounds, monitored on the held-out evaluation set), allowing the effective number of trees to be determined adaptively rather than fixed by the search alone. Tree depth is restricted to a shallower range (3–8) than for Random Forest, since over-fitting compounds across sequential stages, and `min_child_weight` (3–12) provides an additional constraint on leaf size. The learning rate is sampled from 0.01–0.10. Stochastic row and column sampling (`subsample`, `colsample_bytree`) decorrelate successive trees, while L1 and L2 regularisation (`reg_alpha`, `reg_lambda`) and a minimum split-loss threshold (`gamma`) further penalise model complexity.

For the linear SVM, the hyper-parameter space is narrower than for the other two models, reflecting that the regularisation strength `C` is the dominant parameter governing model behaviour. `C` is sampled from a log-uniform distribution spanning five orders of magnitude (10^{-3} to 10^2). The maximum number of solver iterations is also varied (2000–8000), to ensure convergence is not a confounding factor rather than to influence generalisation directly. Thirty search iterations are used to ensure adequate coverage of the log-scaled `C` range.

For all three models, the final selected configuration is the one achieving the highest mean cross-validation accuracy; its performance is reported on both the training and test sets to assess the degree of over-fitting introduced by the tuning process.

6.3. Model Evaluation

Model performance is assessed using standard classification metrics¹ [122]. For a given class c , four quantities are defined: true positives (TP_c), the number of samples of class c correctly classified as c ; false positives (FP_c), the number of samples from other classes incorrectly classified as c ; false negatives (FN_c), the number of samples of class c incorrectly classified as another class; and true negatives (TN_c), the number of samples correctly classified as not belonging to c .

Precision quantifies how many of the samples predicted as class c actually belong to that class, and is defined as

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}. \quad (6.15)$$

A low precision indicates that the model frequently mislabels other intent classes as c , generating false alarms for that manoeuvre type.

¹Because all classes are trained and assessed on the same sample size, weighted metrics are not required for assessment on the synthetic dataset, as these will be the same as the unweighted ones.

Recall (also referred to as sensitivity or the true positive rate) quantifies how many of the actual samples of class c are correctly identified, and is defined as

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}. \quad (6.16)$$

A low recall indicates that the model frequently fails to detect class c when it is truly present, misclassifying it as another intent.

Precision and recall capture complementary failure modes and are typically in tension: increasing one tends to decrease the other, depending on where the model's decision boundary is drawn. The F1-score combines both into a single metric via their harmonic mean,

$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}. \quad (6.17)$$

The harmonic mean is used rather than the arithmetic mean because it penalises large imbalances between precision and recall more heavily, so a high F1-score can only be achieved when both are high.

Finally, overall accuracy is reported as the proportion of all samples correctly classified, over all classes,

$$\text{Accuracy} = \frac{\sum_{c=1}^N TP_c}{\text{Total number of samples}}. \quad (6.18)$$

The three models were evaluated on the testing set, splitting the data per manoeuvre count. The accuracy per manoeuvre, for both LEO and GEO can be found in Figure 6.12, which shows the evolution of the three model's classification performance as context increases.

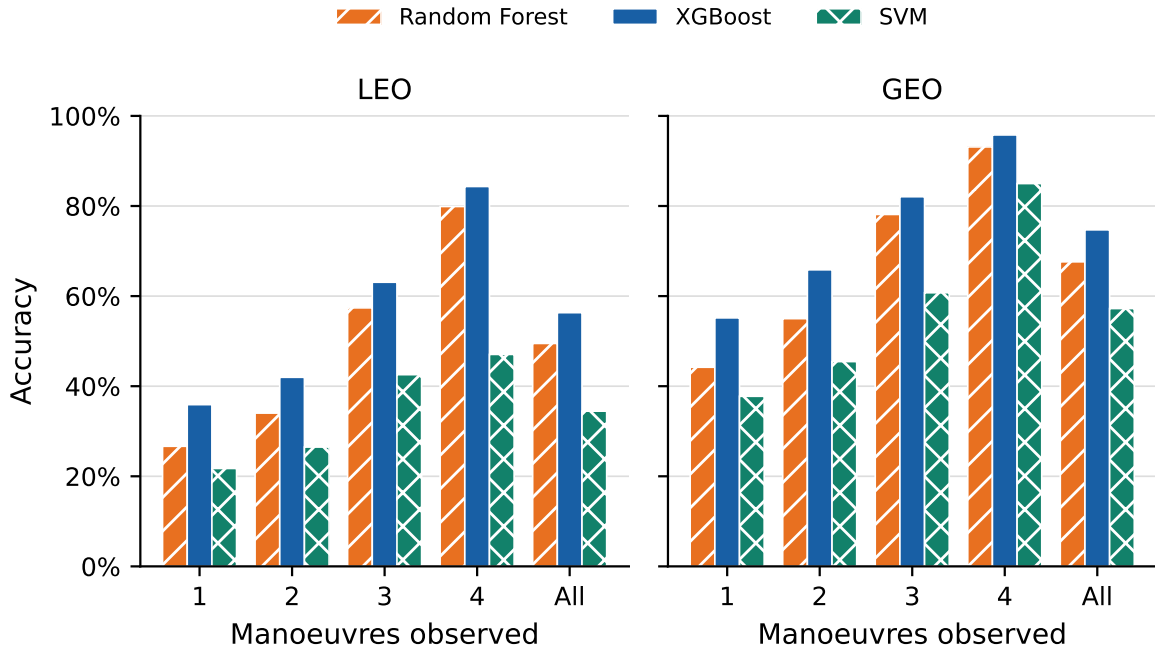


Figure 6.12: Classification accuracy by manoeuvre sequence length on the test sets, GEO and LEO.

A few things can be observed in Figure 6.12. XGBoost stands out as the best classifier, achieving the highest accuracy in both GEO and LEO across every manoeuvre count. In both domains, the largest single-step gains occur later rather than earlier in the manoeuvre sequence rather than diminishing monotonically, suggesting that early intent inference is hard regardless of the model.

Random Forest comes second, while SVM performs worst throughout. The classification models still perform better in GEO than in LEO overall, and the Random Forest–XGBoost gap narrows as more manoeuvres are observed in both domains. These results mirror the rankings established in Table 2.2, where XGBoost outperforms Random Forest, which in turn outperforms SVM. Furthermore, the accuracy achieved at four manoeuvres approaches the maximum reported in the literature (see Chapter 2) for GEO, though LEO does lack behind these values.

The higher accuracy in literature is achieved with more complex model architectures, like Transformers or improved RNNs. Whilst the higher accuracy can in part be explained by the more complex architectures, another notable difference is the approach to intent inference. In part, lower performance is expected since the intent is inferred using less context than literature, which uses time-series information of the Chaser–Target relative dynamics for execution of the intent. In contrast, here the tabular information of only the manoeuvres leading up to the intent are used.

To figure out where the mislabelling occurs, the confusion matrices of each model, at full context (only $n=4$ for manoeuvre count), are presented for LEO in Figure 6.14, Figure 6.13, and Figure 6.15, and for GEO in Figure 6.17, Figure 6.16, and Figure 6.18.

LEO

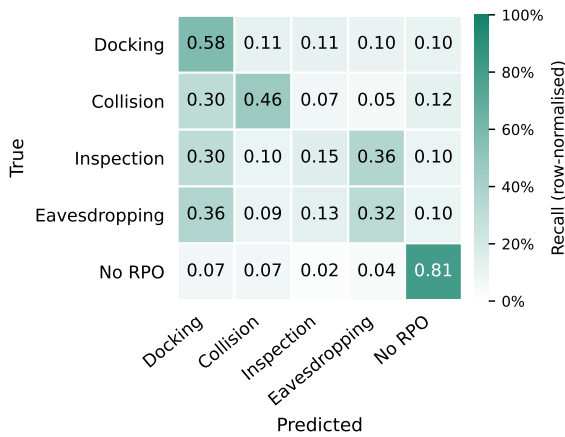


Table 6.2: Precision, recall and F1 per intent ($n=4$ manoeuvres) — SVM, LEO.

Intent	Precision	Recall	F1
Docking	0.355	0.578	0.440
Collision	0.549	0.458	0.500
Inspection	0.308	0.146	0.198
Eavesdropping	0.368	0.319	0.342
No RPO	0.688	0.809	0.743
Average	0.454	0.462	0.445

Figure 6.13: Confusion matrix for SVM in LEO, at full context ($n=4$ manoeuvres).

SVM is considerably weaker in LEO than either of the other two models. *Docking* is a broad sink that absorbs misclassifications from every other RPO class, while *No RPO* remains comparatively well isolated. *Inspection* is by far the worst-performing class on both precision and recall, meaning the model is neither good at finding true inspection episodes nor reliable when it predicts the label. *Docking*'s recall is inflated by this sink effect but its precision is correspondingly low, confirming that although it absorbs a large share of misclassified episodes, it is also frequently wrong when predicted. *No RPO*, by contrast, achieves the highest recall, precision, and F1 of any class by a wide margin, making it clearly the model's best-performing intent in LEO. Taken together, this is the weakest of the six GEO/LEO model combinations, driven primarily by the near-total failure to recall inspection and the model's heavy reliance on *Docking* and *No RPO* as default predictions, rather than any genuine separation of the RPO intents. Furthermore it suggests that separating RPOs from non-RPOs is an easier task than inferring the intent of specific RPOs.

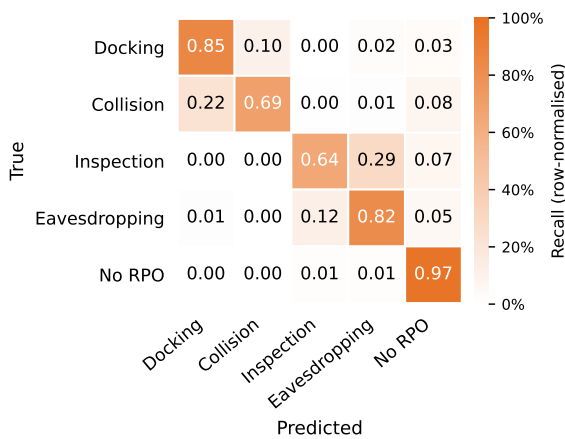


Table 6.3: Precision, recall and F1 per intent (n=4 manoeuvres) — Random Forest, LEO.

Intent	Precision	Recall	F1
Docking	0.788	0.850	0.818
Collision	0.866	0.693	0.770
Inspection	0.820	0.645	0.722
Eavesdropping	0.713	0.819	0.763
No RPO	0.830	0.965	0.892
Average	0.803	0.795	0.793

Figure 6.14: Confusion matrix for Random Forest in LEO, at full context (n=4 manoeuvres).

Random Forest performs reasonably well in LEO, though unevenly across intents. *No RPO* scores highest, followed by *Docking* and *Eavesdropping*, while *Collision* and *Inspection* lag behind (Figure 6.3). Two mutual confusion blocks are present, *Docking–Collision* and *Inspection–Eavesdropping*, are both present and asymmetric: *Collision* is drawn toward *Docking* considerably more often than the reverse, and *Inspection* is drawn toward *Eavesdropping* considerably more often than the reverse. *No RPO* has the highest F1 of the five, with recall exceeding precision, showing the model is not conservative about predicting it. Compared to the SVM, which was not able to discern between RPO intents, Random Forests prove that classifying between RPOs is possible, however it does highlight that discerning between intents with similar profiles can still be an issue.

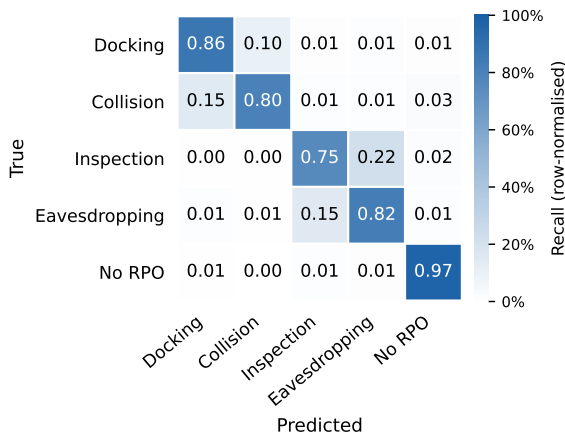


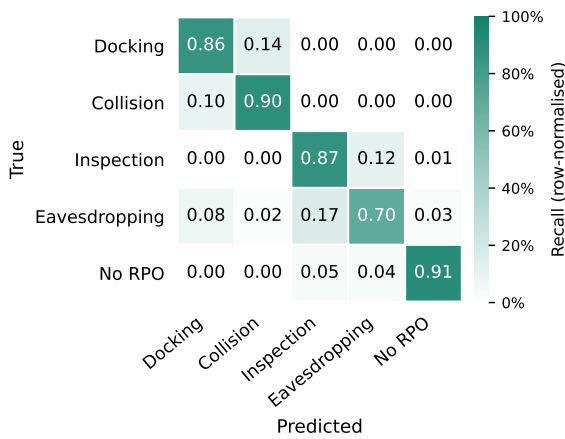
Table 6.4: Precision, recall and F1 per intent (n=4 manoeuvres) — XGBoost, LEO.

Intent	Precision	Recall	F1
Docking	0.836	0.863	0.849
Collision	0.871	0.802	0.835
Inspection	0.802	0.753	0.776
Eavesdropping	0.762	0.817	0.788
No RPO	0.937	0.967	0.952
Average	0.841	0.840	0.840

Figure 6.15: Confusion matrix for XGBoost in LEO, at full context (n=4 manoeuvres).

XGBoost improves clearly on Random Forest in LEO. The confusion patterns follow the same two structural blocks as Random Forest, *Docking–Collision* and *Inspection–Eavesdropping*, but the model recovers a meaningfully larger share of the confused classes in both directions, and false positives into No RPO stay modest for every class. An XGBoost intent inference models should be able to infer intent between classes in LEO, when trained on representative data, although some misclassification between overlapping classes might still occur.

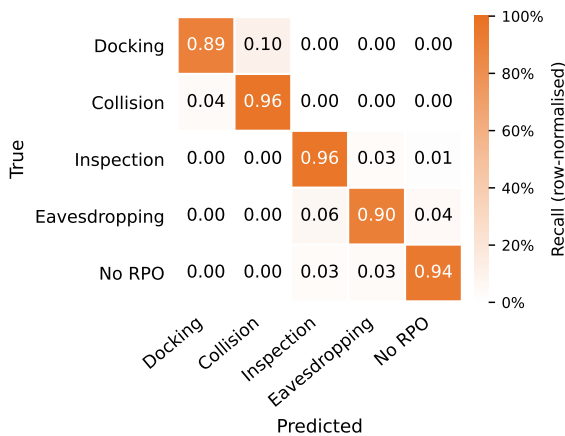
GEO

**Table 6.5:** Precision, recall and F1 per intent (n=4 manoeuvres) — SVM, GEO.

Intent	Precision	Recall	F1
Docking	0.825	0.862	0.843
Collision	0.852	0.898	0.874
Inspection	0.798	0.870	0.832
Eavesdropping	0.803	0.700	0.748
No RPO	0.961	0.911	0.935
Average	0.848	0.848	0.847

Figure 6.16: Confusion matrix for SVM in GEO, at full context (n=4 manoeuvres).

SVM is the weakest performer in GEO (as it was in LEO), however it is able to discern between RPO intent much better than it did in LEO. *No RPO* retains well predicted as it was in LEO, but for the other intents there is better classification. *Eavesdropping* is comparatively the weakest class, it is still able to be discerned from the other classes with some accuracy. The *Docking–Collision* and *Inspection–Eavesdropping* blocks are both also present, mirroring the better performing results from LEO. However, a confusion not seen in LEO at all emerges, with *Eavesdropping* additionally misclassified as *Docking*. *Collision*, *Inspection*, and *Docking* each show recall exceeding precision, indicating the model over-predicts each of these relative to their true occurrence. *Eavesdropping* is the only class where the reverse holds, with a substantial share of true episodes instead drawn away into the other three classes. Compared to LEO, SVM in GEO at least manages a genuine, if imperfect, separation between the RPO intents, rather than collapsing into a single dominant sink class. This suggests that GEO intent inference is an easier task, at least on synthetic data.

**Table 6.6:** Precision, recall and F1 per intent (n=4 manoeuvres) — Random Forest, GEO.

Intent	Precision	Recall	F1
Docking	0.961	0.891	0.925
Collision	0.902	0.963	0.932
Inspection	0.912	0.960	0.935
Eavesdropping	0.931	0.902	0.916
No RPO	0.953	0.939	0.946
Average	0.932	0.931	0.931

Figure 6.17: Confusion matrix for Random Forest in GEO, at full context (n=4 manoeuvres).

Random Forest performs solidly in GEO, with recall fairly balanced across intents. The *Docking–Collision* and *Inspection–Eavesdropping* blocks are both present and asymmetric, skewed toward *Docking* being mistaken for *Collision* in the first case and toward *Eavesdropping* being mistaken for *Inspection* in the second. *No RPO*'s confusion is small and concentrated, mainly drawing from *Eavesdropping*. *Collision* has the highest recall and the lowest precision of the five, which for a safety-critical class is arguably the preferable failure mode, since a missed collision is more costly than a false alarm.

Compared to SVM, Random Forest keeps both confusion blocks considerably tighter, confirming that the two class pairs are genuinely separable given a more capable model, even if they remain the two hardest distinctions in GEO.

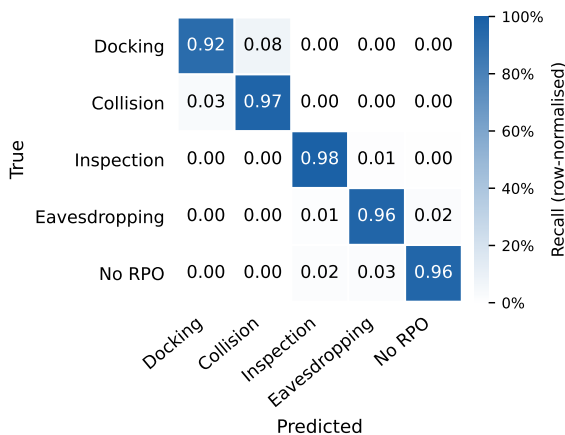


Table 6.7: Precision, recall and F1 per intent (n=4 manoeuvres) — XGBoost, GEO.

Intent	Precision	Recall	F1
Docking	0.964	0.924	0.944
Collision	0.927	0.965	0.946
Inspection	0.964	0.984	0.974
Eavesdropping	0.960	0.960	0.960
No RPO	0.973	0.955	0.964
Average	0.958	0.958	0.958

Figure 6.18: Confusion matrix for XGBoost in GEO, at full context (n=4 manoeuvres).

XGBoost is clearly the strongest model in GEO, a result reflected in both its overall accuracy and its per-intent recall. *Docking* is now the weakest class, but such a high F1 score it indicates that all intents can be reasonably well discerned. The confusion patterns are far more contained than for either of the other two models: the *Docking–Collision* and *Inspection–Eavesdropping* blocks are still present in principle, but at much lower rates in both directions, and *Eavesdropping* is the only intent misclassified as *No RPO* to any notable degree. Performance is broadly balanced across precision and recall for every class; *Eavesdropping* now has precision and recall that are exactly equal, the most balanced class in the table and no longer the weakest, as it was under both other models. *Inspection* has the highest F1 of the five, with *Docking* the lowest, but the spread between them is much narrower than under either Random Forest or SVM. An XGBoost intent inference model should be able to infer intent between classes in GEO with only limited confusion between the two hardest pairs, indicating that the remaining errors reflect genuine ambiguity in the underlying trajectories rather than a limitation of the classifier itself.

XGBoost: LEO versus GEO

XGBoost's performance is weaker in LEO than in GEO across every class, though the degradation is not uniform. *Inspection* has the lowest F1 of the five in LEO, driven by the lowest recall of any class; *eavesdropping* is a close second-lowest by F1, but for a different reason, driven by the lowest precision rather than recall. The confusion patterns follow the same two blocks seen in GEO, docking–collision and inspection–eavesdropping, but both operate at considerably higher rates than their GEO counterparts, while false positives into *No RPO* remain modest in both directions. *No RPO*'s F1 is essentially flat between the two domains and remains the strongest-performing class in LEO by a wide margin, though this is not a straightforward continuation of the GEO ranking, since *inspection* held the highest F1 in GEO before becoming the weakest class in LEO. *Docking*'s precision falls by more than its recall moving from GEO to LEO, so, unlike *No RPO*, it is not comparably stable across domains.

6.3.1. XGBoost Metrics

Diving into the performance of XGBoost, the model itself can provide more insight into the classification process. First, the reliability of the model's confidence estimates is examined in Figure 6.19: for each predicted intent, samples are binned by the model's reported confidence, and the actual accuracy of those predictions within each bin is compared against that confidence. Confidence here refers to the class probability returned by `predict_proba`, obtained by passing the sum of each tree's contribution to a class's score through a softmax function; a higher confidence value indicates that the accumulated tree scores favour one class over the others more strongly, and can serve as an indication of the certainty of a given classification.

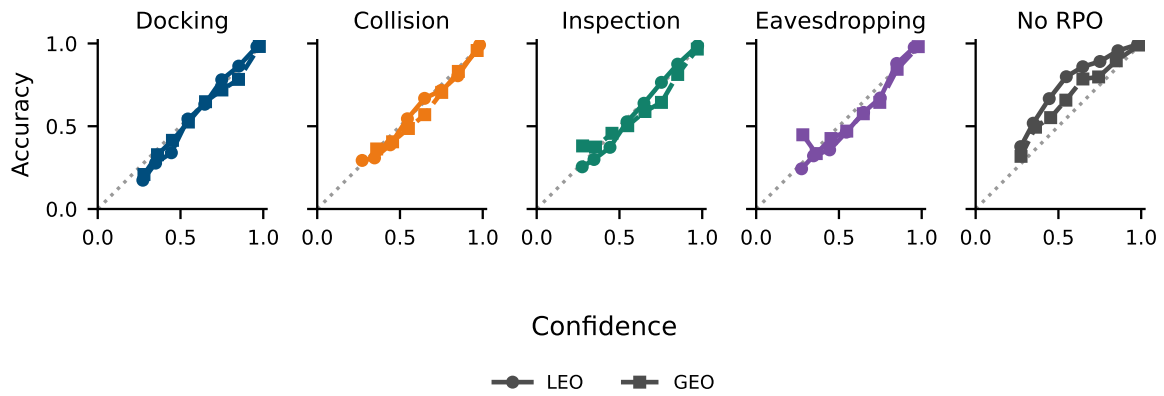


Figure 6.19: Reliability of XGBoost's predicted-class confidence per intent, for LEO and GEO. The dotted diagonal indicates perfect calibration; points above the diagonal indicate underconfidence, where true accuracy exceeds the reported confidence.

Figure 6.19 shows a strong positive relationship between confidence and accuracy for every intent, in both LEO and GEO. However, every curve around the diagonal, indicating that the model is not over or under confident: for a given predicted intent, the true accuracy of the model's predictions tends to be as the confidence it reports. This effect is not seen for the No RPO scenario, where the model is slightly more accurate than confident. This implies that the model's confidence is a good indicator of the reliability of the classification, at least on the synthetic data.

Figure 6.12 did already indicate that performance improves with manoeuvre count, more context means better performance. However, this performance is not uniform per intent, as shown in Figure 6.20.

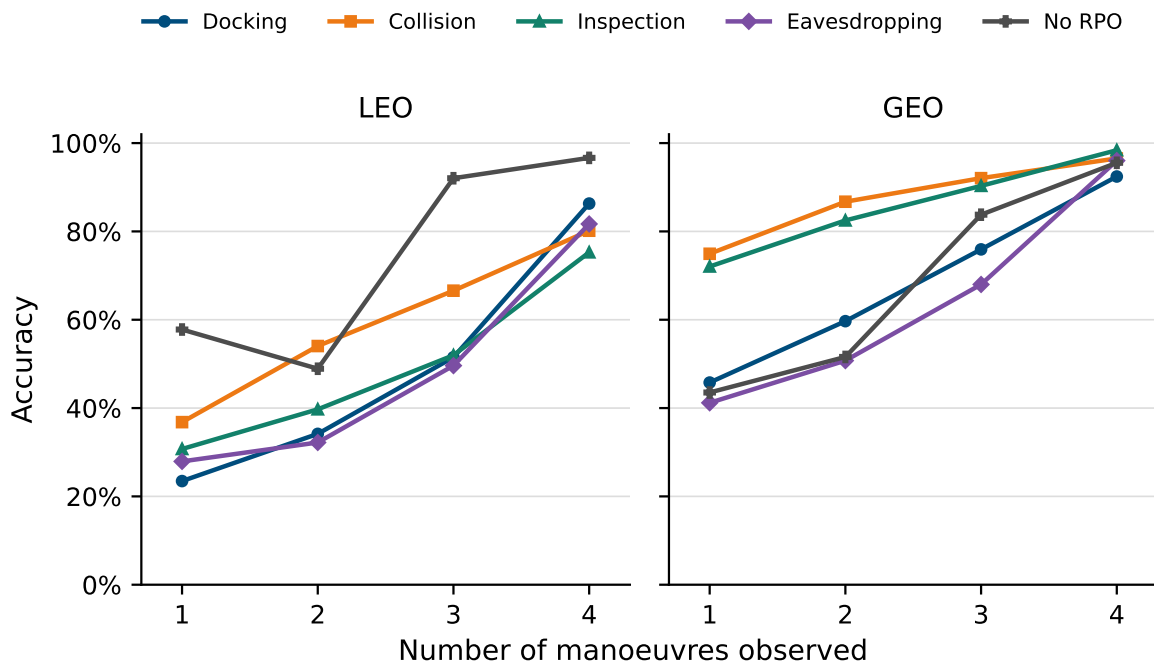


Figure 6.20: Accuracy of XGBoost's predicted-class classifications as a function of manoeuvres observed, per intent, for LEO and GEO.

For all intents, there is a strong correlation with manoeuvres observed and the accuracy of the class. Notably, not all classes perform equally well and there is a difference between LEO and GEO. In LEO, despite overall performance being lower than in GEO, the model is able to discern whether an RPO is going to happen or not better than in LEO. This is potentially due to the GEO belt where most satellites

share the same orbital plane; changes in plane can indicate a specific type of manoeuvre helping with differentiating between intents, but since everything is in the same plane there is less indication of if a satellite is performing an RPO with a specific target. *Collision* performs the best in both domains, owing to its faster approach trajectory which sets it apart from all other intents.

Translating manoeuvre count into a distance or time metric introduces its own limitations, since neither quantity carries a consistent meaning across intent classes. Distance behaves differently per intent: for *Docking*, the distance between target and chaser is expected to approach zero by the fourth manoeuvre, whereas for *Eavesdropping* and *Inspection* this is not the case, as the chaser instead settles into a stand-off distance. A time metric requires a reference epoch to count from, but no single reference point is meaningful across all intents: time-to-closest-approach is well-defined for *Docking*, but is ill-suited to *Eavesdropping*, whose stable oscillating orbit can produce multiple close approaches, and which may already be performing its intent before reaching closest approach at all.

For this reason, the accuracy-versus-distance and accuracy-versus-time figures are both split by intent and by manoeuvre count. The distance metric is the distance between target and chaser following the most recent manoeuvre, while the time metric is the time elapsed since the most recent manoeuvre, meaning it does not capture the full manoeuvre history contained within a given sample. Within each bin, the plotted accuracy is the mean accuracy across all test samples falling in that distance or time range. These relationships are shown in Figure 6.21.

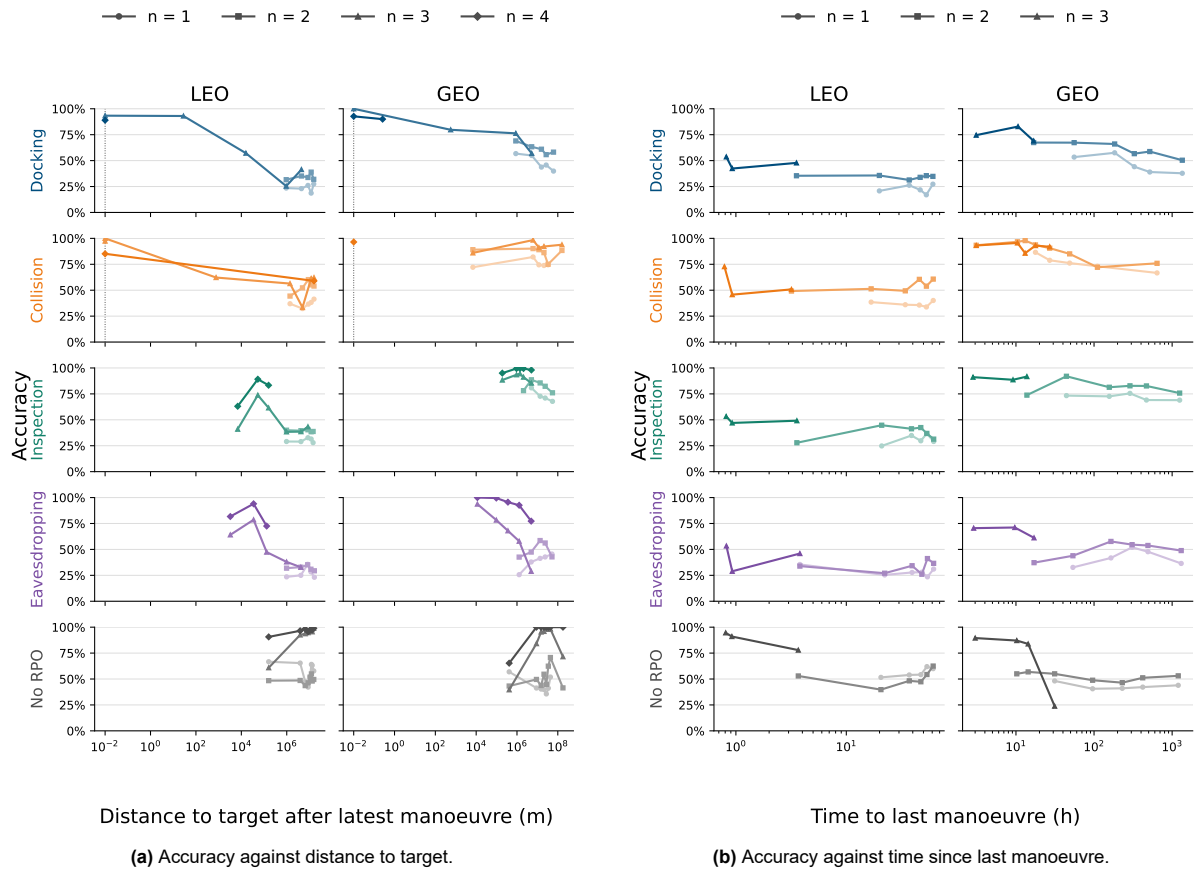


Figure 6.21: Classification accuracy per intent and manoeuvre count, binned by (a) distance to target and (b) time elapsed since the most recent manoeuvre.

Figure 6.21a shows that the relationship with distance is messy. Whilst decreasing distance in general leads to better performance, samples at the same distance but with increased manoeuvre context perform better. *No RPO* is an exception, gaining accuracy at larger distances instead, which is logical but limits the possibility of stating a single distance threshold as a clear boundary for good model performance. *Inspection* and *Eavesdropping* in LEO also gain accuracy with decreasing distance, but

only until around 1000 m, below which performance drops as the model gains confusion with *Docking* at these distances.

Time, shown in Figure 6.21b, has a more consistent relationship with accuracy overall, but performance remains limited by the lack of full-context samples. This holds especially for the *Inspection* and *Eaves-dropping* classes in LEO, which had high mutual confusion rates that would be resolved by the last manoeuvre. Comparing by order of magnitude, accuracy in GEO peaks around 10 h, whereas in LEO this occurs at under 1 h.

Early identification is thus potentially possible in GEO, whilst in LEO it could be operationally difficult.

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Turing, *Computing Machinery and Intelligence* - 1950

7

Model Robustness and Operational Realism

This chapter documents the model's performance on degraded and real-world data. First, Section 7.1 gives the background and approach to data degradation. Then, Section 7.2 presents the resulting assessment of performance on degraded data. Next, Section 7.3 presents the performance on real-world data. Finally, Section 7.4 investigates the real-world failure modes not present in the synthetic evaluation.

7.1. Degradation Background and Approach

The model up until now assumes accurate, instantaneous information of impulsive manoeuvres. This is not realistic: manoeuvres, even high-thrust ones, are not instantaneous, and neither is the orbit determination process, which is limited by factors such as the size of a sensor network and the timing of a satellite's overpasses. Data quality is similarly imperfect, instrument noise and bias, together with errors in the orbit determination model, cause discrepancies between the true and determined states. TLEs from Space-Track, for instance, carry systematic errors at the orbit determination epoch [102], and these errors are further biased in certain directions [123].

The un-degraded timeline of observations is presented in abstract state-space in Figure 7.1, where the ground truth remains constant over time as long as no manoeuvres occur. This timeline will be degraded by both noise and observation frequency.

Noise reflects the inaccuracies of the orbit determination process, which can arise from both the observations and the model used. This differs per format and method used: Orbital Ephemeris Messages from GNSS data can be quite accurate, and have an indication of their error embedded in the message through an error covariance matrix. TLEs, or Orbit Mean-elements Messages, on the other hand, can have higher errors, partly due to the SGP4 orbit determination process [124]. These data formats do not come with an indication of their accuracy embedded in the message itself.

Due to the lack of embedded error covariance for Space-Track data, a standard indicator of error is required. Flohrer, Krag, and Klinkrad [123] assessed the Space-Track database, considering 8463 objects in LEO and 886 in GEO, comparing TLE data to more precise positional reference data sources. The average errors are presented in Table 7.1.

Subdividing further into groupings beyond orbital domain, some differences can be observed. For satellites in near-circular orbits in LEO, inclination is a strong indicator of the error in the TLEs: satellites with an inclination greater than 60 degrees show an in-track error of 517 metres, whilst those below 30 degrees show only 118 metres [124], for example. These differences are not taken into account here.

Noise is applied to the synthetic data based on an error covariance matrix for both position and velocity

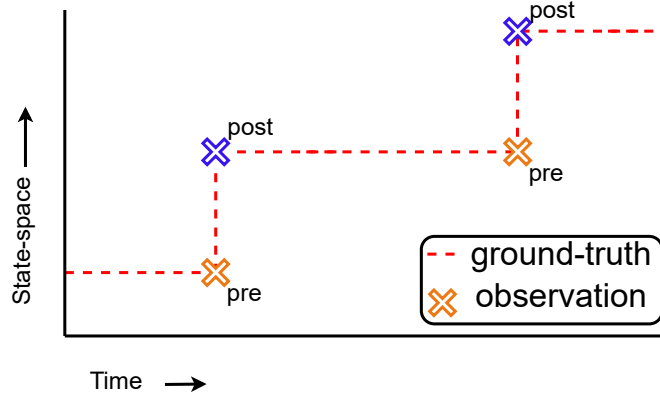


Figure 7.1: Timeline of satellite in abstract state-space and observations of clean data.

Table 7.1: Averaged standard deviations per component, per orbital domain, for TLE errors at the orbit determination epoch [123]

Domain	σ_R [m]	σ_S [m]	σ_W [m]
GEO	359	432	86
LEO	102	471	126

in the RSW frame. From the covariance matrix, a random error is sampled and added to the synthetic ground truth. The noise is fixed per seed for reproducibility. The standard deviations, and thus the representative noise levels, differ per orbital domain. The correlation matrices used to construct the covariance matrices are based on real data and are fixed per domain.

The covariance matrix $\Sigma \in \mathbb{R}^{6 \times 6}$ is constructed as

$$\Sigma = D C D, \quad (7.1)$$

where C is the fixed correlation matrix for the domain and $D = \text{diag}(\sigma_R, \sigma_T, \sigma_N, \sigma_{\dot{R}}, \sigma_{\dot{T}}, \sigma_{\dot{N}})$ contains the per-domain standard deviations for the radial, along-track, and cross-track position and velocity components. The velocity standard deviations are taken as the position standard deviations scaled by a factor of 10^{-3} , reflecting the tighter uncertainty typically associated with velocity determination relative to position.

The correlation matrices were obtained by averaging the correlation matrices derived from the covariance matrices of 30 LEO and 30 GEO satellites' Orbital Ephemerides Data (OEMs); these OEMs were sampled from Wayfinder by Privateer Space, see Appendix A for more information on these satellites and the correlation matrices used. These OEMs support the scale factor for the velocity uncertainty, where the velocity uncertainty was 700 to 1200 times smaller than the uncertainty in position.

For a given domain, a noise vector $\delta = [\delta R, \delta T, \delta N, \delta \dot{R}, \delta \dot{T}, \delta \dot{N}]^\top$ is drawn from a zero-mean multivariate normal distribution, $\delta \sim \mathcal{N}(\mathbf{0}, \Sigma)$. The ground-truth Cartesian state is first rotated into the RSW-frame using the instantaneous position and velocity vectors, after which the sampled noise is added directly to the position and velocity components in that frame. The perturbed state is then rotated back into the inertial frame, yielding the degraded observation used as input to the classifier. This conversion allows noise to be added across all simulated data in accordance with the standard deviations tabulated in Table 7.1.

In addition to measurement noise, the synthetic data can be degraded to reflect a discrete observation cadence, rather than assuming continuous state knowledge. This is modelled by propagating

the ground-truth Cartesian state at the manoeuvres from its original epoch to the nearest observation epoch, defined by a fixed observation frequency Δt_{obs} .

Given an epoch t , the offset within the current observation window is computed as $t \bmod \Delta t_{\text{obs}}$. Depending on whether the pre- or post-manoeuve state is required, the state is then propagated either forward to the next observation epoch,

$$t_{\text{obs,post}} = t + (\Delta t_{\text{obs}} - (t \bmod \Delta t_{\text{obs}})), \quad (7.2)$$

or backward to the previous observation epoch,

$$t_{\text{obs,pre}} = t - (t \bmod \Delta t_{\text{obs}}). \quad (7.3)$$

The observation-frequency degradation is combined with the noise degradation to form “fully” degraded data. For the combination of both degradations, the pre- and post-manoeuve states are first propagated to their relevant observation epochs, after which the noise representative of the orbital domain is added to the propagated state. This ordering reflects the actual process being modelled: an observation occurs at an epoch dictated by the sensor availability, and that observation is then degraded by the sensor. This combined degradation timeline is presented in Figure 7.2, as opposed to the timeline of the clean data in Figure 7.1.

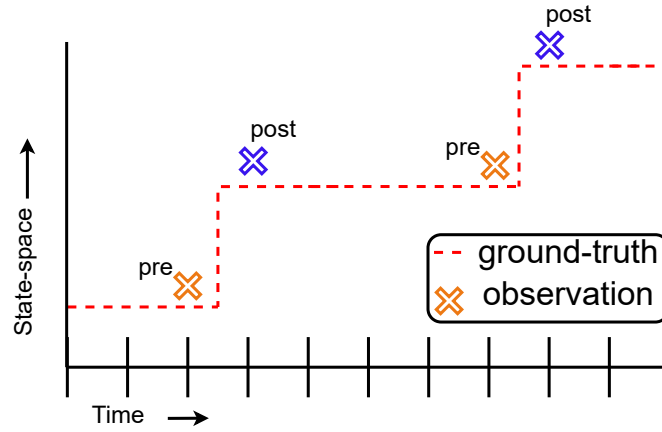


Figure 7.2: Timeline of satellite in abstract state-space and observations of noisy, reduced-cadence data.

For all degradation options, the data input is slightly altered from Chapter 6. The *distance* feature is different pre- and post-manoeuve due to the addition of noise and observation cadence, leading to the inclusion of both a pre-manoeuve and post-manoeuve distance. The new data input is represented in Figure 7.3.

7.2. Assessment on Degraded Synthetic Data

Applying the noise and reduced-cadence degradations to the synthetic data affects the classification performance of the model trained on non-degraded (clean) data, as illustrated in Figure 7.4, which shows the confusion matrices in the full four-manoeuve context.

Data Input Per Case							
Manoeuvre 1		Manoeuvre 2		Manoeuvre 3		Manoeuvre 4	
pre	post	pre	post	pre	post	pre	post
Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}	Δa_{rc}
Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}	Δe_{rc}
Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}	Δi_{rc}
$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$	$\Delta \Omega_{rc}$
$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$	$\Delta \omega_{rc}$
Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}	Δv_{rc}
d_{rcpre}	d_{rcpost}	d_{rcpre}	d_{rcpost}	d_{rcpre}	d_{rcpost}	d_{rcpre}	d_{rcpost}

Figure 7.3: Full input sequence considering all four manoeuvres for degraded observations.

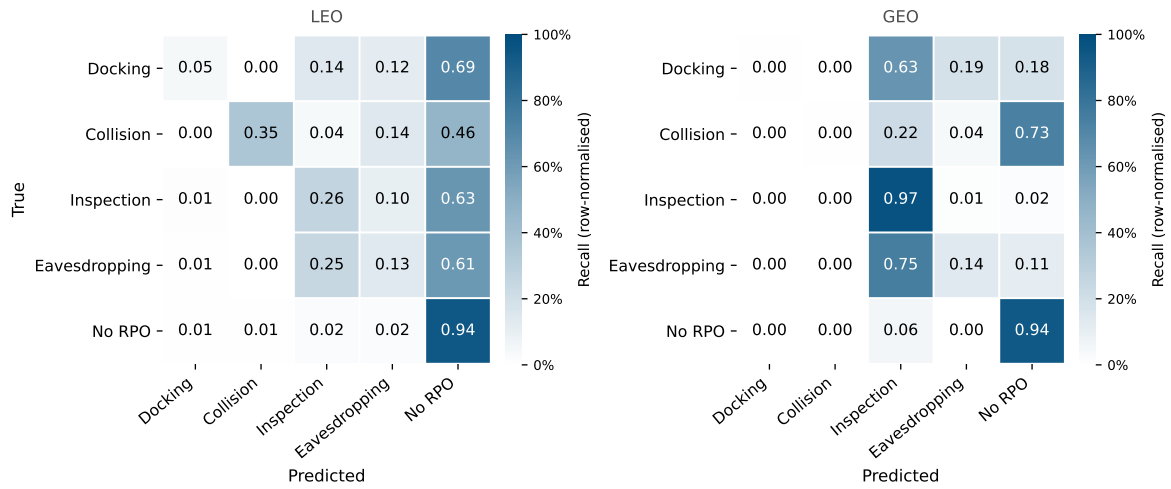


Figure 7.4: LEO and GEO XGBoost models trained on non-degraded data, assessed on synthetic, noisy, reduced-cadence data

The effects differ substantially between the two domains. In GEO, the model never predicts *Docking* or *Collision*, even where these are the true intent. In LEO, classification of these two intents is also significantly reduced, though some true positives remain. In LEO, most misclassifications shift towards *No RPO*, whereas in GEO most RPO intents shift towards *Inspection*, with the exception of *Collision*, which is most often classified as *No RPO*. The key take-away is that a model must be trained on data that reflects realistic noise and degradation; otherwise, it cannot perform intent inference reliably.

Re-training the models on, and assessing them against, the respective degradations (noise, reduced cadence, and their combination) can potentially alleviate these classification issues. Figure 7.5 shows the confusion matrices for models trained and assessed on noisy data.

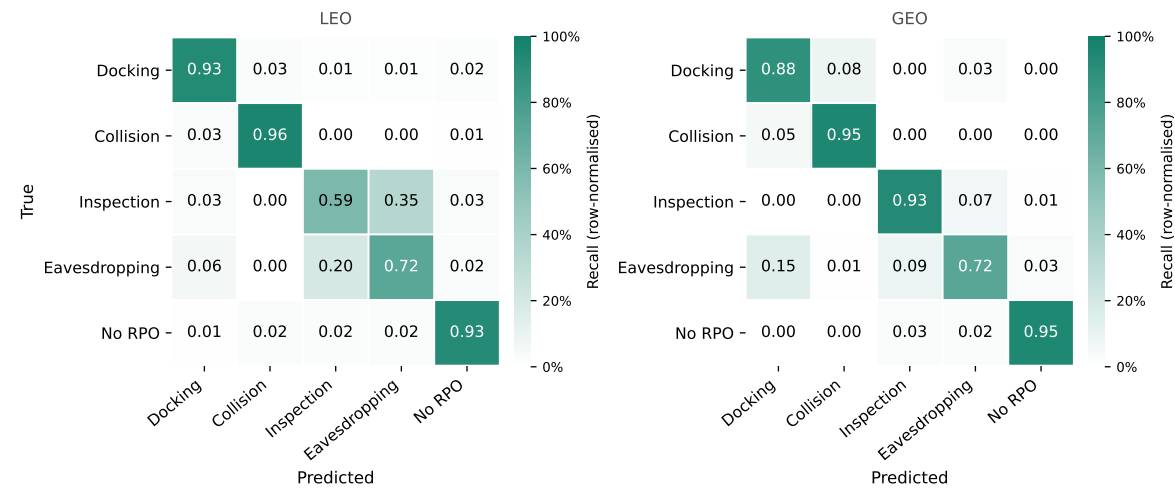


Figure 7.5: LEO and GEO XGBoost models trained and assessed on synthetic, noisy data

Adding noise lowers classification performance overall. In LEO, all RPO classes show some misclassification as *Docking*, whereas in GEO this pattern is observed only for *Eavesdropping*. GEO also shows a significant drop in performance for *Eavesdropping*, driven by misclassification as *Inspection* and, more so, as *Docking*.

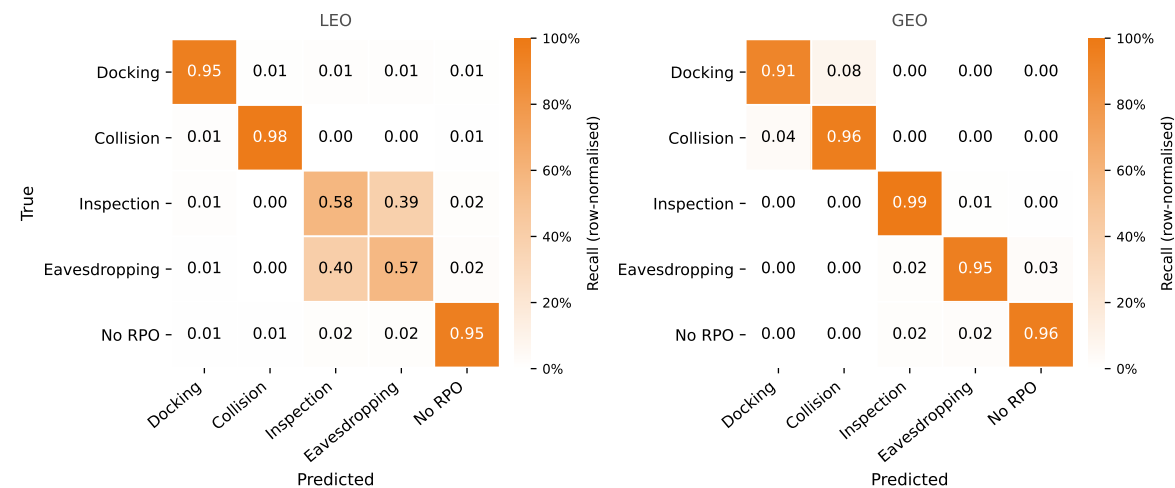


Figure 7.6: LEO and GEO XGBoost models trained and assessed on synthetic, reduced-cadence data

Figure 7.6 shows that lowering the observation cadence does not significantly affect the GEO domain. LEO, however, exhibits a discriminability problem between *Inspection* and *Eavesdropping* that is worse than that observed under added noise.

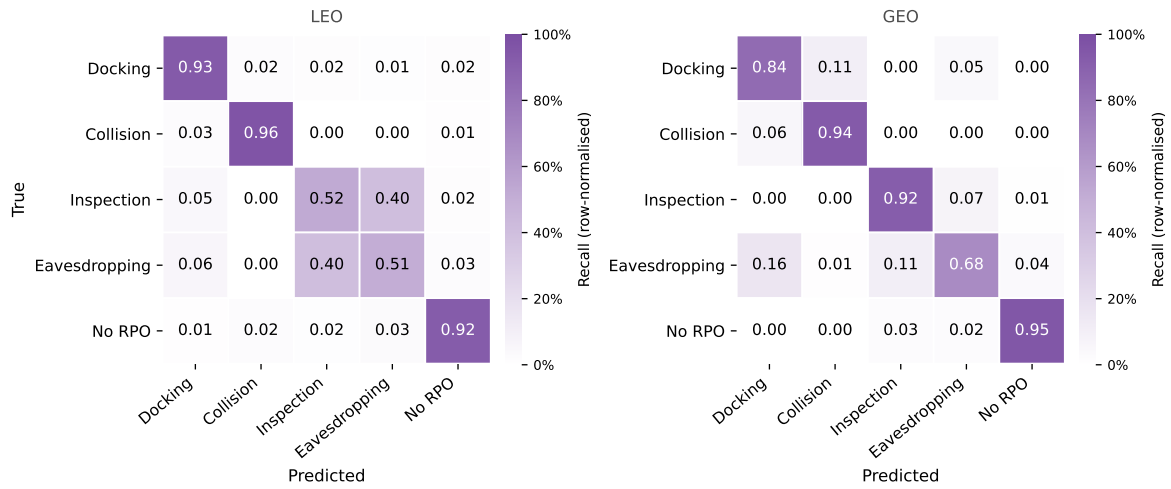


Figure 7.7: LEO and GEO XGBoost models trained and assessed on synthetic, noisy, reduced-cadence data

As shown in Figure 7.7, in both LEO and GEO, *Collision* and *No RPO* are the most robust intents: models trained and assessed on fully degraded data discern them with recall comparable to the clean-data case. *Docking* behaves differently between the two domains; in LEO it remains robust, whereas in GEO it is confused with both *Collision* and *Eavesdropping*. A further difference lies in the confusion between *Inspection* and *Eavesdropping*: in LEO, classification between the two is close to a coin-flip, whereas in GEO the confusion, while present, is less severe, with *Inspection* occasionally misclassified as *Eavesdropping* and vice versa.

Overall, retraining on degraded data alleviates many of the classification issues introduced by noise and reduced cadence, but does not fully recover clean-data performance in either domain. In LEO, the persistent weakness is discriminating *Inspection* from *Eavesdropping*; in GEO, overall performance drops only slightly, with *Eavesdropping* being the main source of confusion.

Furthermore, the implication for a real sensor network is that improvements should be domain-specific to aid intent inference capabilities: GEO surveillance would benefit most from reducing measurement noise, while LEO surveillance would benefit most from a higher observation cadence but also from reduced noise.

7.3. Real World Cases Assessment

Real-world RPO cases are generated using the methods described in Section 5.5. For LEO and GEO, not all intents have examples of them happening, and even less so in a non-cooperative manner. An overview of the cases in LEO is presented in Table 7.2, where if a source has more than one count, it is either because it performed the said intent multiple times with one target, or because it visited multiple targets.

7.3.1. LEO - Cases and Assessment

Table 7.2: Validation dataset composition (LEO).

Source	Intent	Count	Note
ASTRO	Docking	1	Cooperative; Orbital Express demonstrator, docking with the purpose-built NEXTSat target.
Crew Dragon Endurance	Docking	1	Cooperative; crewed vehicle docking with the ISS.
Progress MS-27	Docking	1	Cooperative; uncrewed resupply vehicle docking with the ISS.
Shenzhou-8	Docking	1	Cooperative; uncrewed, first Chinese docking, with Tiangong-1.
Shenzhou-9	Docking	1	Cooperative; crewed, with Tiangong-1.
Shenzhou-10	Docking	1	Cooperative; crewed, with Tiangong-1.
Shenzhou-11	Docking	1	Cooperative; crewed, with Tiangong-2.
Soyuz MS-23	Docking	1	Cooperative; uncrewed, replacing the damaged Soyuz MS-22 at the ISS.
Cosmos 2504	Inspection	6	Domestic; approaches to its own spent Briz-KM upper stage.
Cosmos 2521	Inspection	21	Domestic; subsatellite inspecting with its parent, Cosmos 2519.
Victus Haze	Inspection	5	Non-cooperative; mutual inspection between Jackal-0004 and Puma.
Cosmos 2504	No RPO	6	From the Cosmos 2504 inspection cases; target replaced.
Cosmos 2521	No RPO	168	From the Cosmos 2521 inspection cases; target replaced.
Victus Haze	No RPO	45	From the Victus Haze inspection cases; target replaced.

Notably for LEO, *Eavesdropping* is missing as an intent, as observed in Table 7.2, alongside *Collision*, which is also absent. For the other intents, most were performed in a cooperative manner. Even Victus Haze's inspection of Jackal-0004 is not fully non-cooperative, since it was a fabricated training scenario.

The confusion matrices for LEO, for the models trained on clean and on degraded data, are shown in Figure 7.8.

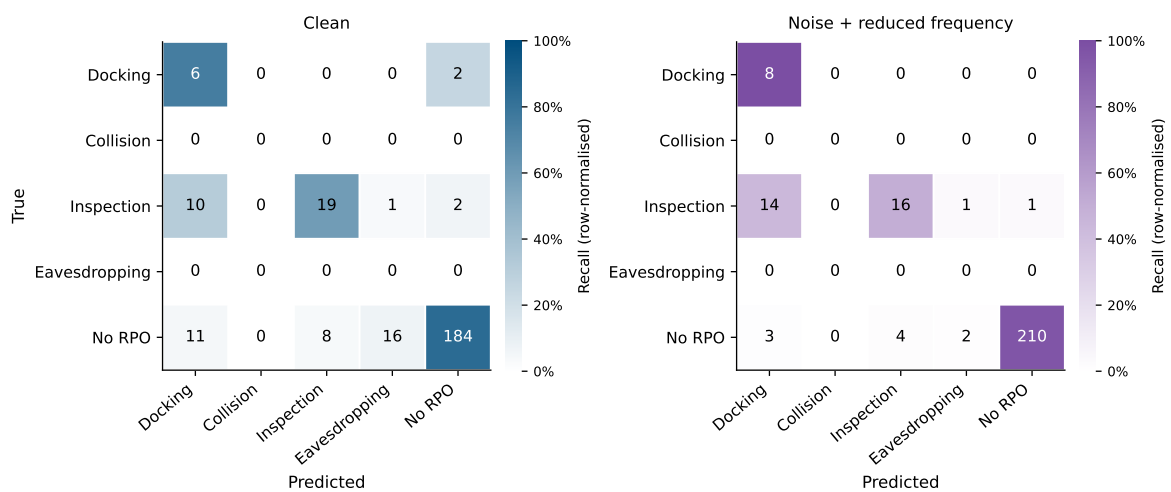


Figure 7.8: Confusion matrices on real LEO data (four manoeuvres) for the XGBoost model trained on clean data (left) versus the model trained on the noisy, reduced-frequency variant (right).

The shift from the clean model to the degraded model shows two improvements for LEO: the *Docking* scenarios are correctly classified in all cases, and more of the *No RPO* scenarios are correctly classified, as observed in Figure 7.8. However, *Inspection* performs slightly worse overall, with increased misclassification as *Docking*. In general, for both models the performance on *Inspection* is worse than when assessed on synthetic data, and the failure mode from *Inspection* to *Docking* is not observed when evaluated on synthetic data. This means there is a fundamental difference between some of the simulated *Inspection* cases and the real ones. This was already noted in Section 5.6, where the real *Inspection* cases clustered closer to *Docking*.

The other failure modes from synthetic data (mutual *Docking–Collision* and *Eavesdropping–Inspection* confusion) are also not present. In part this is due to the lack of examples, with *Collision* and *Eavesdropping* not even being present as true intent labels. However, of the eight *Docking* samples, none are classified as *Collision*, and with a recall of 93% on fully degraded data, the expected number of misclassifications is less than one. For *Inspection* the performance is unexpected; on synthetic degraded data there is significant confusion with *Eavesdropping*, but this is not at all present here. With 32 supporting samples, at least some misclassification as *Eavesdropping* would be expected if synthetic and real data were aligned. Instead, significant confusion with *Docking* is noted, suggesting a difference between simulation and real life: *Inspection* takes place with smaller distances between the Target and Chaser in real data than in simulated data.

7.3.2. GEO - Cases and Assessment

GEO has more intents than LEO, with only *Collision* lacking a real-world example, as observed in Table 7.3. Also, the intents themselves are non-cooperative more often, with the GSSAP and LUCH satellites performing their actions in this manner. For *Docking*, only cooperative examples are present.

Table 7.3: Validation dataset composition (GEO).

Source	Intent	Count	Note
MEV-1	Docking	1	Cooperative; life-extension docking with Intel-sat 901.
MEV-2	Docking	1	Cooperative; life-extension docking with Intel-sat 10-02.
SJ-21	Docking	2	Compass G2 grapple and SJ-25 docking.
SJ-17	Inspection	6	Cooperative(?), might have been <i>Eavesdropping</i> .
USA 270 (GSSAP-3)	Inspection	8	Non-cooperative; close approaches along the GEO belt.
USA 271 (GSSAP-4)	Inspection	6	Non-cooperative; close approaches along the GEO belt.
USA 324 (GSSAP-5)	Inspection	2	Non-cooperative; close approaches along the GEO belt.
LUCH / Olymp-K 1	Eavesdropping	29	Non-cooperative; loitering near commercial communications satellites.
LUCH / Olymp-K 2	Eavesdropping	8	Non-cooperative; loitering near commercial communications satellites.
LUCH	No RPO	27	From the LUCH eavesdropping cases; target replaced.
MEV-1	No RPO	2	From the MEV-1 docking case; target replaced.
MEV-2	No RPO	18	From the MEV-2 docking case; target replaced.
USA 270 (GSSAP-3)	No RPO	2	From the GSSAP-3 inspection cases; target replaced.
USA 271 (GSSAP-4)	No RPO	6	From the GSSAP-4 inspection cases; target replaced.
USA 324 (GSSAP-5)	No RPO	3	From the GSSAP-5 inspection cases; target replaced.

The confusion matrices for GEO, for the models trained on clean and on degraded data, are shown in Figure 7.9.

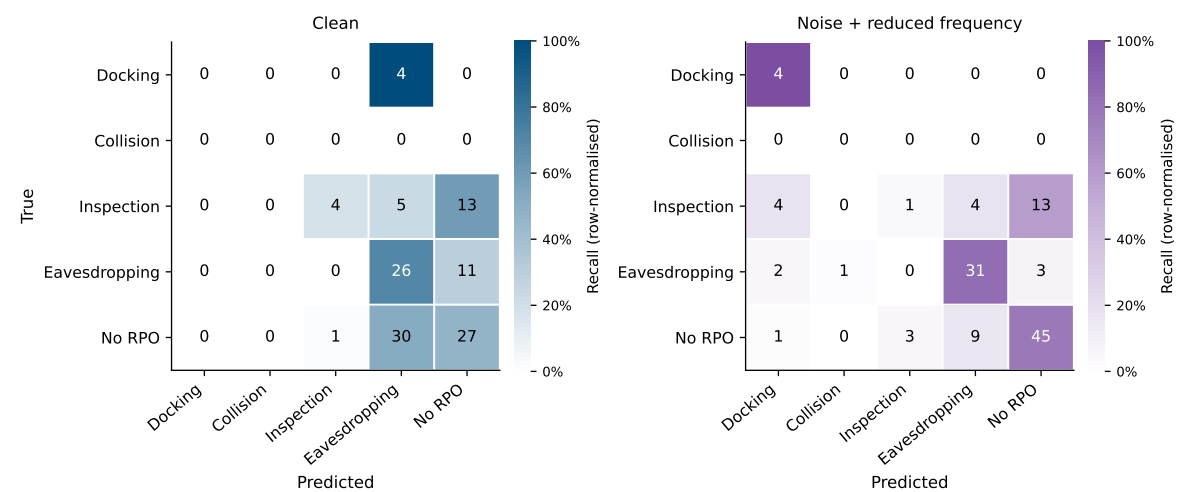


Figure 7.9: Confusion matrices on real GEO data (four manoeuvres) for the XGBoost model trained on clean data (left) versus the model trained on the noisy, reduced-frequency variant (right).

The move from the clean-data model to the degraded-data model provides significant improvements in GEO, as observed in Figure 7.9. As expected from synthetic degraded data, the clean model never predicts *Docking* or *Collision*, and switching to the degraded model leads to correct classifications of *Docking*. What is not expected for the clean model is the general misclassification as *Eavesdropping*, which seems to be the misclassification sink in GEO on real data. On synthetic data, *Inspection* fulfilled this role. The performance for both *Eavesdropping* and *No RPO* improves as well; it is notable, however, that half of the *No RPO* scenarios were classified as *Eavesdropping* by the clean model, whereas this is not observed in synthetic data. Most likely, the fake targets placed in these real-data scenarios are closer to the real target than in the data the model was trained on, but training on noisy data aids separability. The worst-performing class is *Inspection*, where both the clean model and the degraded model struggle to identify the real intent. Whilst there is some confusion with *Eavesdropping*, as expected from the degraded synthetic data performance and the uncertainty in SJ-17's intent, there is also significant misclassification as *No RPO* and as *Docking*, which is not observed on synthetic data. Similarly to Figure 6.20, the accuracy per manoeuvres observed is presented in Figure 7.10 for real data.

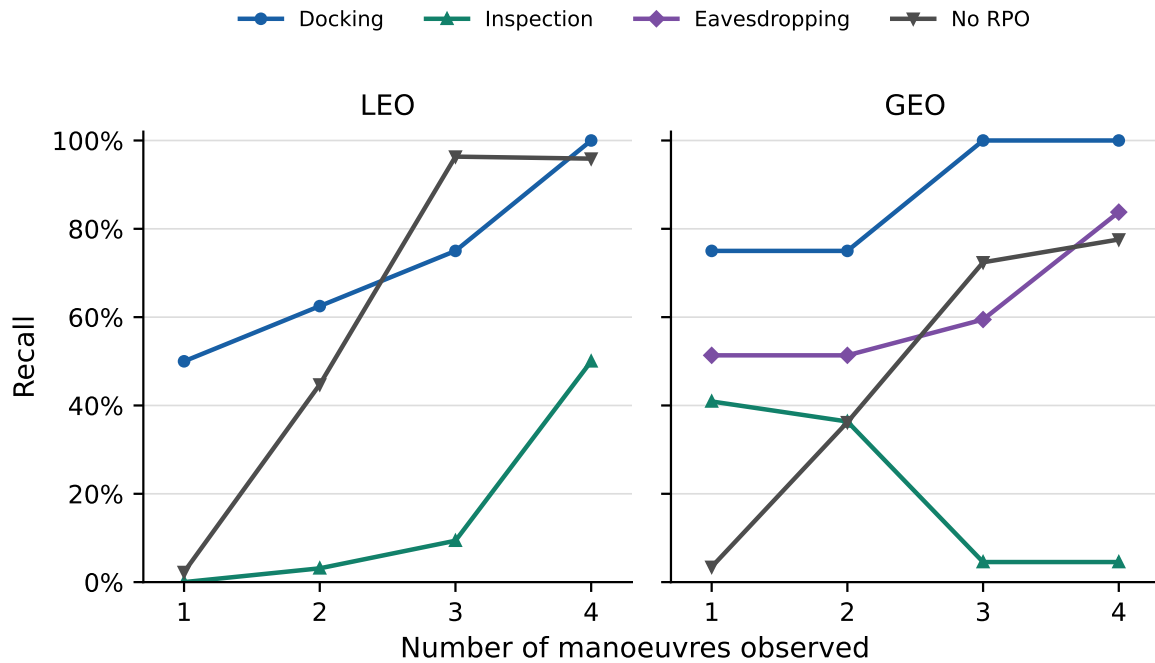


Figure 7.10: Accuracy of XGBoost's predicted-class classifications as a function of manoeuvres observed, per intent, for LEO and GEO, real data.

Conclusions are harder to draw from Figure 7.10 due to the limitations of the real data. Not only the quality but also quantity makes concrete conclusions impossible. Compared to synthetic data, *Docking* seems to be better identified than on real data across all manoeuvres and regimes. *No RPO* is impossible to predict at early manoeuvres, but later manoeuvres do give an indication as to whether this occurs or not, with LEO being slightly better with real data than synthetic, whilst GEO is slightly worse. *Eavesdropping* in GEO starts with slightly better performance on synthetic data, but is worse at later manoeuvres. A notable performance drop is *Inspection* in both LEO and GEO, where with full context the model is a coin flip in LEO, and barely classifies any cases correctly in GEO. The reasons for these differences is further explored in Section 7.4.

7.3.3. Real World Cases Classification Behaviour

The clean model's performance in LEO is better than the synthetic-data results would suggest (see Figure 7.4). On fully degraded synthetic data, *Docking* is a class that is never predicted, whereas on real data it can be classified correctly, albeit alongside some misclassifications from other intents. In GEO, the pattern is more similar to that observed on synthetic data: no (mis)classifications as *Docking* or *Collision*. However, on synthetic data the clean model tended to classify most RPOs as *Inspection*, whereas on real data the sink is instead *Eavesdropping*.

This points to a structural difference between real data and the synthetic degraded data, with the issue lying in both the data generation process and the degradation process. Training on degraded data does improve model performance on real data, but since the clean model's misclassification patterns on synthetic data are not reproduced in LEO, and only partially reproduced in GEO, this improvement should not be read as validating the synthetic degradation process itself. The model trained on degraded data thus performs best overall, but this does not mean the underlying data generation and degradation processes are free of issues.

Furthermore, the lack of truly non-cooperative examples for some of the intents is a notable gap in the real-world dataset. When a non-cooperative RPO is being performed, other operational considerations might be in play; for example, the timeline might be shorter, or the burns might be of greater magnitude, than in a cooperative version of the same intent. This in turn can lead to different inputs for the models and thus potentially different classification results.

A related gap concerns *Docking*, which is considered to also encompass grabbing behaviour. Only SJ-21 may have exhibited this behaviour in real data, and no clear distinction can be drawn as to whether it should be treated differently from standard docking.

7.4. Simulation-to-Real Failure Mode Shift

Three failure modes on real world data are found that are not common for the model when assessed on synthetic data; in LEO a significant portion of *Inspection* scenarios are classified as *Docking*, in GEO two were found; *No RPO* as *Eavesdropping* and *Inspection* as *No RPO*. Furthermore, there are some singular examples of failures not necessarily in a grouping in GEO. A handful of RPOs like *Inspection*, *Eavesdropping*, and even *No RPO* classified as *Docking*, which is not an expected failure mode from the synthetic data analysis.

For these failure modes, the drivers for wrongful classifications are investigated using SHapley Additive exPlanations (SHAPs), which for an individual evaluation can determine the contribution of inputs to correct or incorrect classification [125]. Only the input parameters contributing to wrongful classifications are used and averaged. The contribution is measured as log-odds, which is the natural-log ratio of the the probability of a class occurring and not occurring and the standard output of XGBoost before.

7.4.1. LEO: Inspection-to-Docking

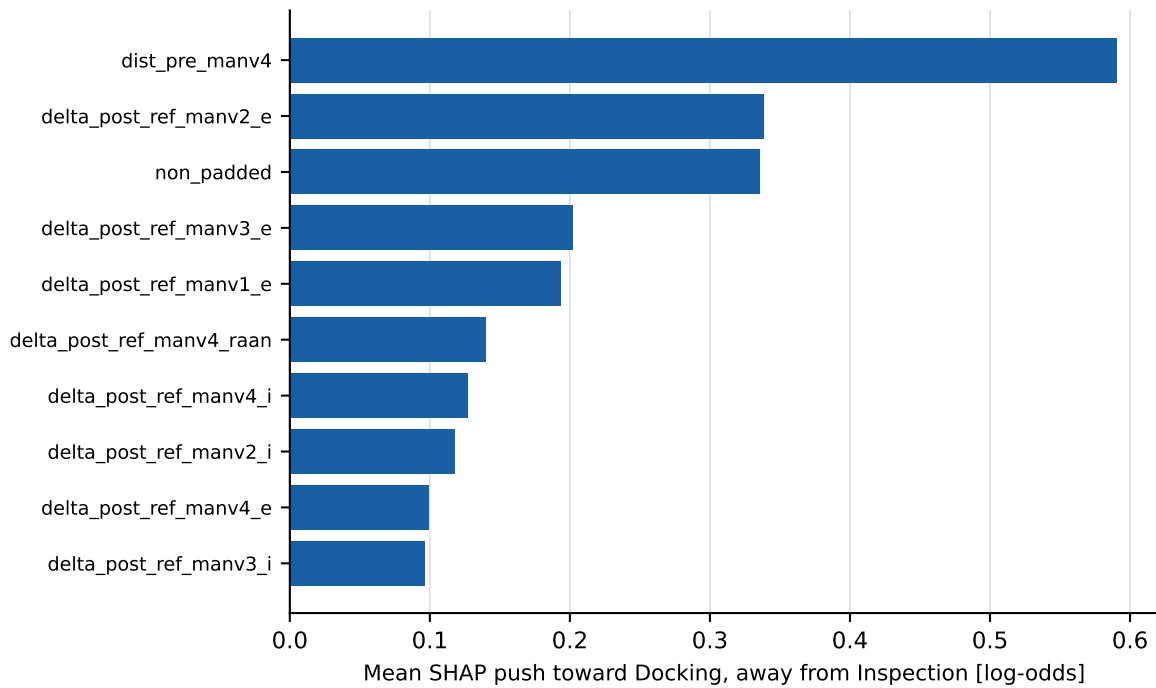


Figure 7.11: SHAP feature contributions for misclassified LEO *Inspection*-to-*Docking* episodes.

Figure 7.11 shows that misclassifications of LEO *Inspection* as *Docking* are driven mainly by the distance before the last manoeuvre. Distance itself is likely a strong distinguisher between *Inspection* and *Docking* in the training data, with smaller distances indicating a higher likelihood of docking. The second-strongest indicator is the eccentricity after the second manoeuvre, together with smaller contributions from the eccentricity at the first and third manoeuvres. *Docking* scenarios are less eccentric than *Inspection* scenarios in the training set. The final major contributor is the sequence number of the manoeuvres (as indicated by *non_padded*): observing the four manoeuvres in the present order pushes the classification towards *Docking*, but the exact mechanism behind this is unclear.

Since the last manoeuvre distance is the most important driver, the distribution of these distances for the true labels of *Inspection* and *Docking* is presented in Figure 7.12. Here, the distance at the last manoeuvre is plotted for both the real and synthetic data sets, for correct and incorrect classifications

as *Inspection* and *Docking*.

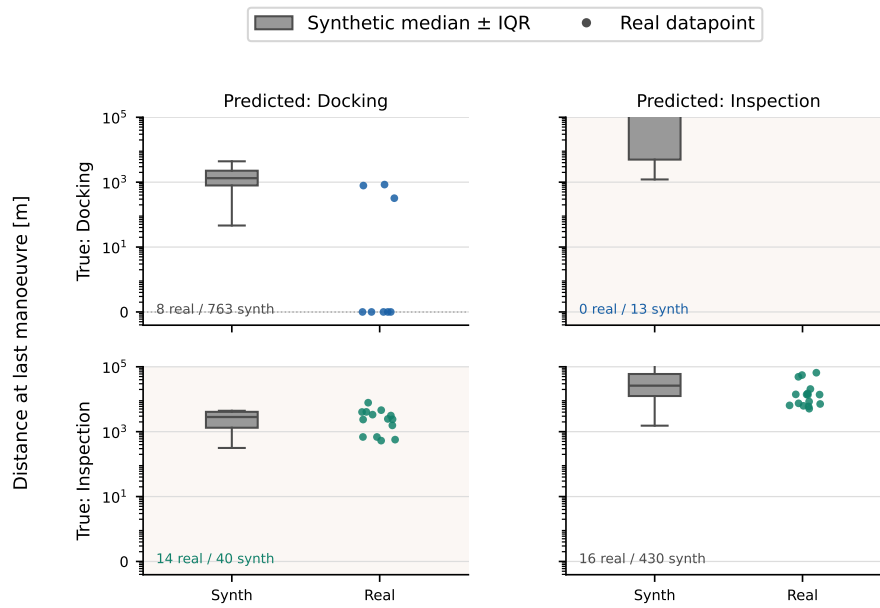


Figure 7.12: Distance distributions at last manoeuvres for *Inspection* and *Docking* (mis)classifications in LEO, real and synthetic data.

The distance distributions differ across the groupings. For the synthetic failures, *Docking* is only misclassified as *Inspection* when the last distance metric is more than 1 km, whilst all wrongful classifications of *Inspection* as *Docking* occur at less than 10 km; the model struggles to distinguish these two classes within the 1 km to 10 km range. For the failure mode on real data, confusion occurs in this same zone: *Inspection* is always classified as *Docking* below 1 km, there is confusion between 1 km and 10 km, and above 10 km the classification is consistently correct.

Figure 7.12 does show that this is a genuine issue for discernibility by the model, as there is a non-negligible group of synthetic *Inspection* scenarios that are misclassified as *Docking*, which have a distance distribution similar to the real data from the failure mode. Training the model on more *Inspection* scenarios that perform their inspections at a closer distance might improve performance, but could also cause more *Docking* scenarios to be misclassified as *Inspection*.

7.4.2. GEO: Inspection-to-No RPO

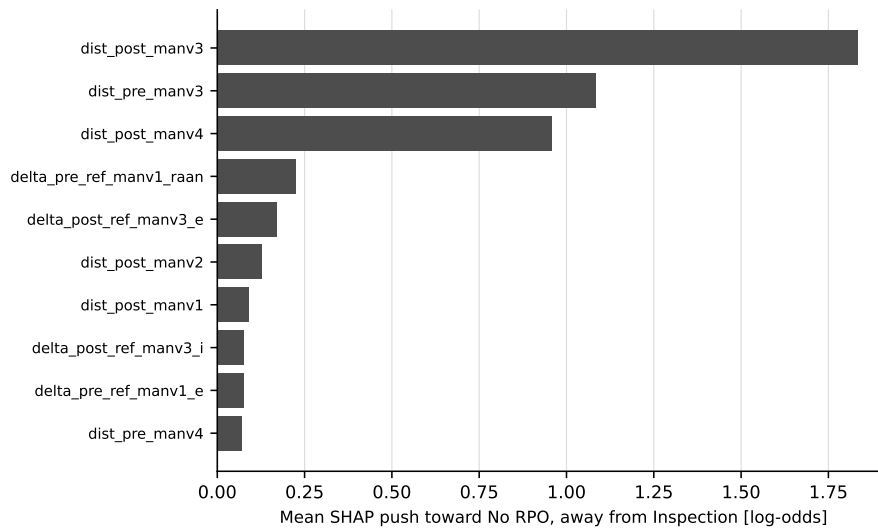


Figure 7.13: SHAP feature contributions for misclassified GEO *Inspection-to-No RPO* episodes.

Figure 7.13 shows that misclassifications of *Inspection* as *No RPO* are driven almost entirely by the distances at which the manoeuvres occur. Distance is a good distinguisher between an RPO taking place and not taking place, and these results suggest that such episodes have distance features considerably greater than those observed in the training set. The distance evolution after a manoeuvre is thus shown in Figure 7.14.

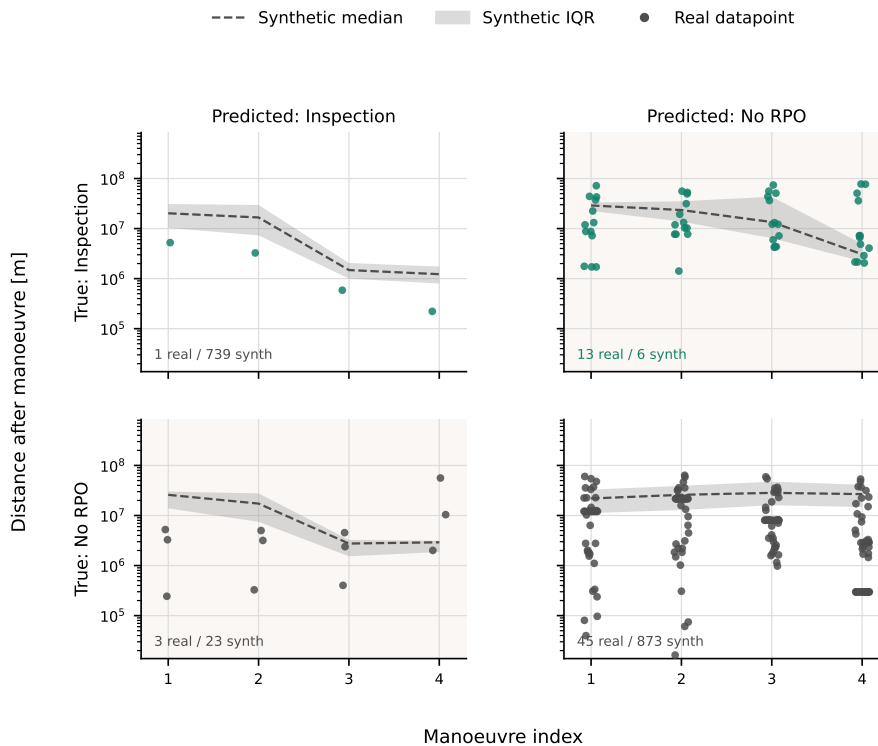


Figure 7.14: Distance evolution after a manoeuvre for *Inspection* and *No RPO* (mis)classifications in GEO, real and synthetic data.

Comparing the true *Inspection* labels reveals a clear distinction between the real *Inspection* episodes

correctly classified as such and those misclassified as *No RPO*: the correctly classified samples are below 1000 km at the last manoeuvre, whereas all misclassified ones are more than 2000 km away, with some between 10 000 km and 100 000 km. Comparing with the synthetic data reveals a further major difference: at the third manoeuvre, the distance metric for the correctly classified episodes is about 2000 km, but for the synthetic cases misclassified as *No RPO* the median is 10 000 km.

For the misclassified real *Inspection* episodes, there are two groupings: those that fall within the range of the (also misclassified) simulated *Inspection* episodes, and those that do not approach the target at all. For this latter class, the distances involved raise the question of whether these are genuine inspections, or whether the reported inspections did not actually take place. A potential explanation lies in the source data, which come from the GSSAP satellites (see Chapter 4). Due to their classified nature, data on them can be sparse, and no TLEs are available during the close approach, which leads to these great distances. For the former grouping, the episodes are borderline in their suitability as *Inspection* RPOs. In terms of distance, they do align with some of the simulated entries; however, they are significantly further away than the majority of the correctly identified samples. It is thus ambiguous whether this failure mode reflects a model failure or is better classified as an input-data issue.

7.4.3. GEO: No RPO-to-Eavesdropping

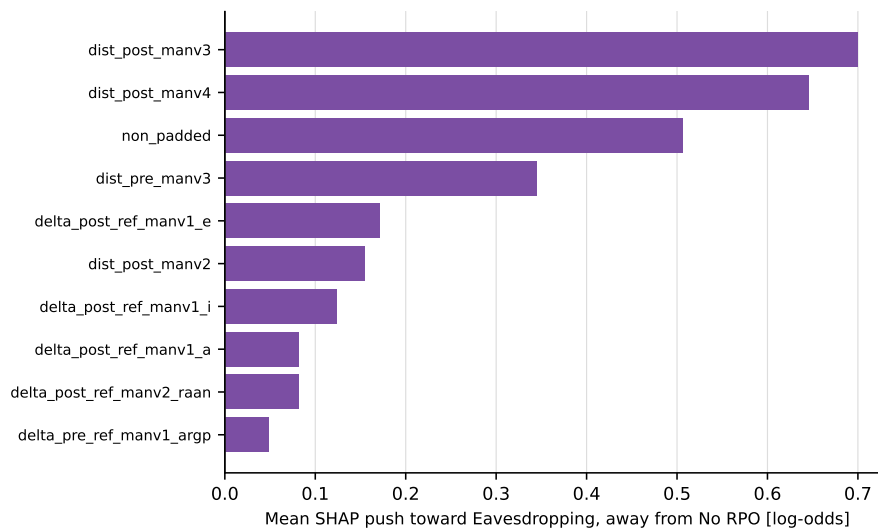


Figure 7.15: SHAP feature contributions for misclassified GEO *No RPO-to-Eavesdropping* episodes.

Figure 7.15 shows the reverse of the *Inspection* misclassification pattern: distance is again the main driver, but here it pushes classifications from *No RPO* towards *Eavesdropping*. In these cases, the *No RPO* examples occur closer to the target than in the simulated data. These distances are further investigated in Figure 7.16.

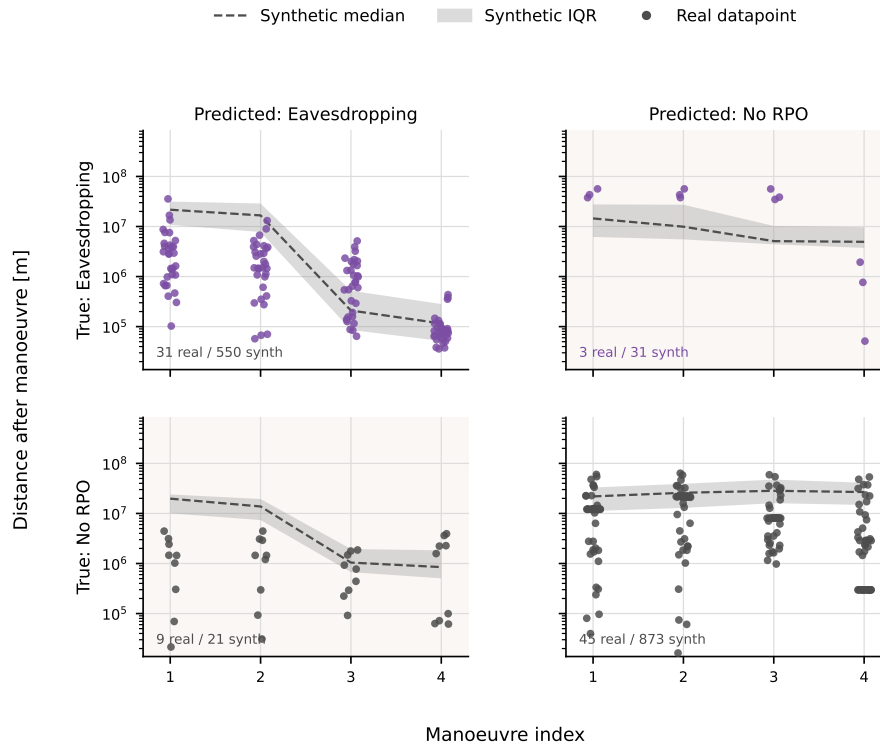


Figure 7.16: Distance at manoeuvres for *No RPO* and *Eavesdropping* (mis)classifications in GEO, real and synthetic data.

The *No RPO* real data points that are classified as *Eavesdropping* share some characteristics with the true *Eavesdropping* predictions: they start off far away, move a bit closer at manoeuvre 2, and are significantly closer at manoeuvres 3 and 4, as the chaser closes the distance to the target. What differs is the final distance: for true *Eavesdropping* episodes this is around 100 km, whereas for the false classifications it depends on the grouping: one is also close to 100 km, while another is more than 1000 km. This latter grouping can be explained in part by the synthetic *Eavesdropping* episodes that are misclassified as *No RPO*, which start to occur at 8000 km; the *Eavesdropping*-to-*No RPO* transition occurs somewhere between 1000 km and 10 000 km.

The issue is a genuine separability problem for the model in some cases: if the chaser closes the distance and falls within a certain distance bound from the target, it is hard for the model to distinguish what the actual target of an *Eavesdropping* episode was. For the others, a possible improvement would be to simulate the synthetic *No RPO* episodes so that they come closer to the false target.

“All models are wrong, but some are useful.”

George E. P. Box, *Robustness in the Strategy of Scientific Model Building* - 1979



Discussion

This chapter discusses the main findings of the report. First, the answers to the sub-research questions are presented in Section 8.1. The key implications from these are discussed in Section 8.2. Finally, the limitations are covered in Section 8.3.

8.1. Answer to Research Questions

The main research question was investigated by answering five sub research questions. The answers to them are as follows.

Sub Research Question 1

RQ 1: What observable characteristics define non-cooperative rendezvous and proximity operations in Earth orbit?

Non-cooperative RPOs are shrouded in uncertainty regarding what exactly has taken place. The intents that have historically occurred, according to open-source knowledge, are *Docking*, *Collision*, *Eavesdropping*, and *Inspection*.

Docking and *Collision* have not occurred between non-cooperative targets as defined in this thesis, only as technology demonstrators, and *Collision* only in the previous century, which limits the available orbital data. Both close the relative distance to zero; *Docking* additionally aims for smooth contact and shapes its approach profile accordingly, whereas *Collision* disregards relative velocity. *Inspection* has occurred in both LEO and GEO, in all observed cases as a fly-by, typically circling the target and sometimes offset in the radial direction. In LEO, repeated close approaches are common. *Eavesdropping* has occurred only in GEO, characterised by a stable relative trajectory offset from the target, from which the chaser can intercept communications sent from the ground.

Sub Research Question 2

RQ 2: How can synthetic datasets of non-cooperative RPO trajectories be generated and represented to enable intent inference?

Synthetic datasets can be generated by splitting the problem into the orbit or end condition characteristic of the intent, and the approach used to reach it. *Docking* and *Collision* reduce to an end condition on the chaser’s relative state, both requiring zero relative distance and *Docking* additionally zero relative velocity. *Eavesdropping* and *Inspection* are instead defined by an orbit that executes the intent, obtained by optimising for ground-signal intercept and stability, and for coverage and minimum distance

respectively.

All approach trajectories use a four-impulse transfer with a Lambert solver to reach the final condition, optimised with a hybrid genetic algorithm combining SADE and SGA, with early stopping and Nelder–Mead refinement. This does not guarantee a global optimum. Comparison with real-world data shows that the later manoeuvres match observed behaviour closely for the intents available, whilst the earlier manoeuvres are more variable.

Furthermore, representing these trajectories by the difference in Keplerian elements between target and chaser, before and after manoeuvres, is most suitable for intent inference. Adding a distance metric to this input significantly improves real-world performance.

Sub Research Question 3

RQ 3: Which machine learning model architectures are suitable for inferring intent in non-cooperative RPO scenarios?

XGBoost performed best among the models considered. This is worth setting against the wider literature, where intent inference is addressed almost exclusively with deep learning architectures (like LSTM, GRU, CNN, and Transformer variants) motivated by the need to process a full observation time series, for which simpler architectures such as MLPs are known to perform poorly. This thesis instead reduces each approach to a fixed set of manoeuvres resulting in a sparse, structured representation rather than a raw time series. That a comparatively lightweight classical model performs strongly here suggests the deep sequential architectures common in the literature are compensating for a dense input representation, rather than being intrinsically necessary for intent inference itself. Accuracy increases with each manoeuvre of the approach, reaching 84% in LEO and 96% in GEO at full context. Intent becomes clear progressively with manoeuvres, which has a correlation with time and distance but there is not clear performance metric for all intents at a given distance or time.

The residual confusion is concentrated on *Eavesdropping* in both domains, but is wider in LEO, where it is mutually confused with *Inspection*. In GEO the remaining intents are well separated and *Eavesdropping* alone is ambiguous, being read as either *Inspection* or *Docking*. *No RPO* is the most distinguishable class in both domains.

Manoeuvre count is the strongest predictor of confidence: accuracy rises consistently with each additional observed manoeuvre, and reliable predictions are available from the final or penultimate manoeuvre onward in both regimes. Time before execution correlates with confidence too, but less consistently, which is why the resulting lead time differs sharply between regimes, approximately 10 h in GEO against less than 1 h in LEO, despite both being anchored to the same manoeuvre-based milestone. Distance is the weakest and least generalisable predictor: there is no single distance at which the model is consistently confident, since the intents become distinguishable at different ranges depending on which intent is present.

Sub Research Question 4

RQ 4: How does the performance of the intent prediction model degrade under non-ideal data conditions?

A model trained without exposure to degraded data does not degrade gracefully when evaluated on noisy or sparsely observed encounters; it collapses onto a small number of classes, with *Docking* and *Collision* rarely predicted in either domain.

The two degradations act through different mechanisms. Reduced observation cadence obscures the manoeuvres themselves and pushes all intents towards *No RPO*, which dominates in LEO. Noise obscures the precision of the burn and erodes the distinction between the intents that depend on it, affecting *Docking* and *Collision* most in both domains.

Training on degraded data offsets most of this loss, which shows that both noisy data and observation cadence are required for real-world performance. It does not recover everything: in LEO the *Inspection–Eavesdropping* confusion worsens, and in GEO a new failure mode emerges in which *Eavesdropping*

is classified as *Docking*.

Sub Research Question 5

RQ 5: How does the model perform on real-world (non-cooperative) rendezvous and proximity operations?

On real data the model can classify correctly in both domains, and the errors that remain are systematic. In LEO the main failure mode concerns *Inspection*: episodes closer than any in the training set are classified as *Docking*. This accounts for approximately half of the real *Inspection* episodes. It is not evidence of a fundamental separability limit between the two intents: episodes at this range may still be distinguishable, provided the training set includes *Inspection* trajectories at comparably close distances. In GEO the errors are driven by the *No RPO* decision boundary, with *Inspection* episodes misclassified as *No RPO* and some *No RPO* cases as *Eavesdropping*.

Performance follows the match between training and real-world data. *Eavesdropping* in GEO is both the best-matched intent and the best-classified one, which indicates that the limits encountered are limits of the training distribution rather than of the inference approach.

For *Inspection* episodes misclassified as *No RPO* the limits are mainly an indication of the data quality of these real-world cases. Model performance for this class in GEO is thus not indicative of the true potential if better data was available.

8.2. Implications

A recurring theme across RQs 1–4 is that LEO and GEO behave as genuinely distinct regimes for intent inference, rather than as interchangeable data conditions: manoeuvre-based accuracy, robustness to degradation, and the residual confusion between *Eavesdropping* and *Inspection* all differ between the two. Practically, this means these domains need different considerations for both model training data and potential improvements.

The intent categories used in this thesis sit within the literature's Mission Class definition of intent, which the review found the most informative of the three definitional groups but also the least justified in terms of linking these classes to real-world behaviour. Defining the intent categories directly from behaviour observed in Earth orbit yields this direct link to potential operational objectives.

This thesis also differs from the literature in what data is classified and how. The reviewed machine learning approaches almost universally classify the executed mission trajectory itself, whether directly through its shape (Motion Classes) and represent it as a dense, evenly sampled time series. This thesis instead classifies the approach phase leading up to that trajectory, represented sparsely as a fixed set of manoeuvres. As discussed in Chapter 2 and Chapter 6, this sparser representation is plausibly why a lightweight classical model performs as well as it does here, without the recurrent or attention-based architectures the literature relies on.

The classification accuracies obtained here (84% in LEO, 96% in GEO at full context) compare favourably with those reported for long-term or Mission Class intent elsewhere in the literature. Among literature that investigated long-term intent, the classifier used falls below 60% accuracy from approximately 14 km outward in GEO, with performance improving at closer range. At this distance in GEO, the developed model reaches 75% to 90% accuracy for RPOs in GEO. The comparison is not one-to-one, however: that work reports performance purely as a function of distance, whereas RQ3 shows that no single distance threshold is consistently informative across the five intents considered in this thesis, so a time- or manoeuvre-based framing is arguably the more meaningful metric to report against.

The finding that training on degraded data recovers most of the accuracy lost under noisy or sparsely observed conditions (RQ4) extends the retraining result of Sun et al. [86], who demonstrated a comparable recovery for just noise degradation in LEO. This thesis shows the same principle holds across two orbital regimes and five intent classes, reinforcing that operational realism needs to be represented in the training distribution, not only introduced at evaluation.

The validation against real-world data (RQ5) addresses a gap that is explicit in the literature review:

every study surveyed there relies exclusively on synthetic data, and only one considers LEO at all. That the errors observed on real data are systematic rather than arbitrary, and traceable to a specific mismatch between the training distribution and observed behaviour, indicates that the limiting factor is training data coverage rather than a weakness in the inference approach itself.

8.3. Limitations

The model is trained and evaluated using a fixed input of four manoeuvres leading up to the intent. How to handle episodes with fewer or more than four, which observed manoeuvres to select, and how many are required for a reliable prediction, is not investigated here. One plausible approach for episodes with fewer than four manoeuvres would be padding at the start of the sequence (e.g. treating three observed manoeuvres as manoeuvres two, three, and four), but this would introduce a distribution mismatch relative to training and is expected to degrade performance rather than resolve the limitation.

The classification approach assumes that manoeuvres have already been detected and segmented as an upstream step; this thesis does not investigate or evaluate the manoeuvre detection algorithm in much depth. Model performance on real-world data is contingent on the quality of manoeuvre detection, but the effect a has not been quantified.

Training assumes impulsive manoeuvres under point-mass central-body gravity, with no orbital perturbations modelled; real manoeuvres are not perfectly impulsive, and real dynamics include perturbations absent from training. In principle, both simplifications could explain the misclassification observed on real data. In practice, neither appears to be the driver. The LEO *Inspection–Docking* confusion is a training-coverage issue, not a dynamical one: real *Inspection* misclassified episodes occur closer than most example in training, as described above. In GEO, *Inspection* episodes are misclassified as *No RPO* because of GSSAP’s tracking data: its TLEs can predate the close approach by days to weeks, making the last observed manoeuvres appear much further from the target than in training and pushing the prediction towards *No RPO*. Separately, *No RPO* episodes are misclassified as *Eavesdropping* in 9 of 58 GEO cases (16%), against 2 of 217 in LEO (1%); the GEO case is a genuine limit of the model rather than a data or coverage issue, whereas the LEO occurrence is not systematic. In the GEO cases, the *No RPO* episodes are themselves RPOs directed at a different, real target, and when the chaser genuinely closes to a comparable distance from the labelled target in the process, the resulting trajectory is not distinguishable from *Eavesdropping* using the features available. That none of the observed misclassification traces back to the impulsive-manoeuve or no-perturbation assumptions suggests that the current level of simulation fidelity is not the limiting factor for real-world performance.

The dataset does not include exogenous variables such as those explored by Bala [88] (chaser ownership, NORAD ID, or onboard payload), which that work found useful for distinguishing Mission Class intent, particularly through payload information. This thesis does not evaluate whether including such variables would improve classification, nor, as Bala [88] also notes, how this information would be acquired in an operational setting.

Real-world validation is constrained by the availability of labelled data. Open-source knowledge of non-cooperative RPO episodes is itself uncertain, which limits how confidently ground-truth intent labels can be assigned; and the volume of confirmed real-world examples per intent class is small, which limits how robustly model performance on real data can be assessed. Both the knowledge of and the quantity of these real-world non-cooperative RPOs will most likely expand with time: several of the examples used here occurred during the course of the thesis itself, such as Victus Haze, and the work by Samson and Brett [4] and Chang and Bingen [96] also being released during this period. Further case studies will thus arise after this thesis’s deadline.

Furthermore, current real-world RPOs might themselves be limited by contemporary technology. In the future, as dynamic space operations potentially become more common, greater changes in velocity may be possible. This could lead to chaser satellites manoeuvring more aggressively, potentially increasing (false) classification as *Collision*, and could allow target spacecraft to evade an incoming chaser. The model shows some robustness to this latter aspect: the Victus Haze *Inspection* labels in LEO were the correctly classified inspections, where Jackal and Puma were performing chaser-evasion games for at least some of the inspections.

"All truths are easy to understand once they are discovered; the point is to discover them."

Galileo Galilei, *Attributed in "Angels in the Workplace: Stories and Inspirations for Creating a New World of Work"* - 1999

9

Conclusion

Non-cooperative RPOs threaten space infrastructure, which has become a vital part of the global economy and of military operations. Determining what a satellite is going to do during such an operation is necessary to decide on mitigating action. The aim of this thesis was therefore to answer the research question:

Main Research Question

MQ: How can the intent of a non-cooperative satellite's rendezvous and proximity operation be inferred early?

The intent of a non-cooperative RPO can be inferred by considering the operational objectives behind each intent and simulating trajectories that fulfil them, which provides the training set for a machine learning classifier. The classifier discerns between the simulated intents with up to 84% accuracy in LEO and 96% in GEO. Intent can be inferred early at the ultimate or penultimate manoeuvre of the approach, which corresponds to a lead time of 10 h in GEO and less than 1 h in LEO. On real-world data the method performs well where the encounter resembles the training set, and its failures elsewhere are traceable to that mismatch.

This thesis makes three main contributions to the literature. First, a concrete definition of intent based on operational objectives, with the intent categories drawn from groupings observed in Earth orbit. This formulation states what a non-cooperative satellite is going to do rather than how it might do it, and it leads to a different way of modelling intent: through the objectives that fulfil the mission, not through pre-defined motions. The classification that occurs with this definition has a clear, direct link to what is going to occur operationally in orbit.

Second, it infers intent from the approach phase of the chaser rather than from the mission trajectory that executes the intent. The mission trajectory is still simulated by optimising objective functions related to the intent, but it is the approach towards that trajectory which is classified. This is a fundamental shift from the literature, which focuses on the mission trajectory itself. In addition, the method uses only the manoeuvres during the approach and the change in relative orbital geometry they produce, rather than the full time series. This input representation is far sparser and allows lighter models to perform intent inference.

Third, an extensive assessment of the method on real-world behaviour in both LEO and GEO. The assessment exposes the sim-to-real shift that the literature leaves untested: good performance on synthetic data does not imply correct classification of real encounters. It also shows that the training

data must resemble real-world behaviour, whilst demonstrating that some extrapolation beyond the training set is possible.

9.1. Recommendations for Future Work

Three recommendations follow from these findings.

First, the data generation should be brought closer to observed behaviour. The clearest evidence for this is *Eavesdropping* in GEO, which is simultaneously the intent whose simulated trajectories match real encounters most closely and the intent classified most reliably. The LEO failure mode points to the same conclusion from the opposite direction, since the misclassified *Inspection* episodes are those that fall outside the range of approach distances the training set spans. Practically, this means widening the sampled ranges to cover the closer approaches seen in LEO, changing the target replacement for the *No RPO* class to be closer to the real-targets, and investigating early-phase manoeuvres to understand why these manoeuvres differ more from synthetic data sets than the later ones.

Second, the manoeuvre detection deserves attention in its own right. *Eavesdropping* used a specific extraction algorithm for the intent, so part of its advantage may stem from the extraction rather than from the data generation alone. Investigation the impact of manoeuvre detection to generate real-world input data on the model classification performance can inform where part of the sim-to-real shift comes from; either from the manoeuvre extraction itself, or the data generation of synthetic data. This in turn can then inform where more effort needs to be put for model improvement; better manoeuvre detection algorithms or better data generation.

Third, the real-world assessment should be extended, both in the number of encounters considered and in what is asked of the model. The present evaluation establishes that the method transfers under distribution match and identifies where it does not, but the number of documented non-cooperative encounters is small, particularly in LEO. A larger dataset would allow the model's performance to be quantified, rather than assessed qualitatively as is done here, and could also point to avenues for further synthetic data generation. Building such a dataset will take time, but new non-cooperative RPOs are most likely to occur after the deadline for this thesis allowing for more samples.

References

- [1] World Economic Forum and McKinsey & Company. *Space: The \$1.8 Trillion Opportunity for Global Economic Growth*. Report. World Economic Forum, Apr. 2024. URL: <https://www.weforum.org/publications/space-the-1-8-trillion-opportunity-for-global-economic-growth/> (visited on 2026-07-22).
- [2] T. Harrison and K. Johnson. *Defense Against the Dark Arts in Space: Protecting Space Systems from Counterspace Weapons*. Report. Center for Strategic and International Studies (CSIS), 2021. URL: <https://www.csis.org/analysis/defense-against-dark-arts-space-protecting-space-systems-counterspace-weapons>.
- [3] C. Swope et al. *Space Threat Assessment 2025*. Report. Center for Strategic and International Studies (CSIS), 2025. URL: <https://www.csis.org/analysis/space-threat-assessment-2025> (visited on 2026-02-10).
- [4] V. Samson and K. Brett. *2026 Global Counterspace Capabilities Report*. Tech. rep. Secure World Foundation, 2026. URL: <https://www.swfound.org/publications-and-reports/2026-global-counterspace-capabilities-report> (visited on 2026-06-30).
- [5] K. Pruetz, N. McNaughton, and M. Schneider. “Closely-Spaced Object Classification Using MuyGPyS”. In: *arXiv preprint arXiv:2311.10904* (2023). arXiv: 2311.10904.
- [6] X. C. Yee et al. “Machine Learning Classification GEOs Using Spectral Data”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2023.
- [7] C. A. Mello et al. “Advancing Geosynchronous Satellite Classification Utilizing Spectral Data via Fine-Tuned Pretrained Deep Learning Models”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2024.
- [8] H. She et al. “Multi-Layered Machine Learning for Rapid LEO Object Characterization Leveraging Global Radar Data”. In: *Proceedings of the 22nd IAA Symposium on Space Debris*. 2024, pp. 870–884.
- [9] G. Lavezzi et al. “Early Classification of Space Objects based on Astrometric Time Series Data”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2024. URL: <https://amostech.com/TechnicalPapers/2024/AstroDynamics/Lavezzi.pdf>.
- [10] A. Agathangelou et al. “Machine Learning for Satellite Characterisation”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2022. URL: <https://amostech.com/TechnicalPapers/2022/MLSSA/Agathangelou.pdf>.
- [11] D. Shen et al. “General-sum Game Modeling of Generative Adversarial Networks for Satellite Maneuver Detection”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2022.
- [12] S. Shivshankar and D. Ghose. “Time-to-Event Data (Survival Analysis) based Modelling of Maneuver Occurrence of Non-Cooperative Satellites”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. Maui, Hawaii, 2023. URL: <https://amostech.com/TechnicalPapers/2023/Poster/Shivshankar.pdf>.
- [13] M. Squires et al. “Detection and Characterization of Maneuvers Using a Global Radar Network”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2023.

- [14] N. Perovich, Z. Folcik, and R. Jaimes. “Applications of Artificial Intelligence Methods for Satellite Maneuver Detection and Maneuver Time Estimation”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2022.
- [15] E. Gerber et al. “RPO Maneuver Detection from Pixel Space using Deep Learning”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2023.
- [16] I. Connerney et al. “In-Space Inspection Maneuver Analysis Using Trajectory Optimization”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2021, pp. 133–140. URL: <https://amostech.com/TechnicalPapers/2021/Conjunction-RPO/Connerney.pdf>.
- [17] M. Guimarães, C. Soares, and C. Manfletti. “Data-Driven Identification of Main Behavioural Classes and Characteristics of Resident Space Objects in Low Earth Orbit through Unsupervised Learning”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2024.
- [18] T. G. Roberts et al. “End-to-End Behavioral Mode Clustering for Geosynchronous Satellites”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2023. DOI: 10.64861/PYCE7526.
- [19] D. Higgins et al. “SOM-erizing Cislunar Orbits: Classification of Cislunar Orbits Using Self-Organizing Maps (SOMs)”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2024. DOI: 10.64861/PKVP5982.
- [20] P. M. Siew et al. “Satellite Pattern-of-Life Identification Challenge: Competition Design and Results”. In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2024. DOI: 10.64861/KLYS1084.
- [21] S. G. Jo et al. “Spacecraft rendezvous and proximity operation control in the cislunar region”. In: *2024 AAS/AIAA Astrodynamics Specialist Conference*. Vol. 8. 2024.
- [22] J. O. Romero and F. Khoury. “Cislunar Rendezvous and Proximity Operations in the Bi-Circular Restricted Four-Body Problem”. In: *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2023. URL: <https://amostech.space/year/2023/cislunar-rendezvous-and-proximity-operations-in-the-bi-circular-restricted-four-body-problem/>.
- [23] S. Sharma and S. D’Amico. “Neural Network-Based Pose Estimation for Noncooperative Spacecraft Rendezvous”. In: *IEEE Transactions on Aerospace and Electronic Systems* 56.6 (2020), pp. 4638–4658. DOI: 10.1109/TAES.2020.2999148.
- [24] H. Gao, Z. Li, N. Wang, et al. “SU-Net: pose estimation network for non-cooperative spacecraft on-orbit”. In: *Scientific Reports* 13 (2023), p. 11780. DOI: 10.1038/s41598-023-38974-1.
- [25] J. Posso, G. Bois, and Y. Savaria. “Mobile-URSONet: an Embeddable Neural Network for On-board Spacecraft Pose Estimation”. In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2022, pp. 794–798. DOI: 10.1109/iscas48785.2022.9937721.
- [26] D. Rondao, N. Aouf, and M. A. Richardson. *ChiNet: Deep Recurrent Convolutional Learning for Multimodal Spacecraft Pose Estimation*. 2021. arXiv: 2108.10282.
- [27] Y. Zhen et al. *Cross-Modal RGB-D Fusion Transformer for 6D Pose Estimation of Non-Cooperative Spacecraft with Stereo-Derived Depth*. 2026. arXiv: 2605.08592 [cs.CV]. URL: <https://arxiv.org/abs/2605.08592>.
- [28] M. Pontani and B. A. Conway. “Numerical Solution of the Three-Dimensional Orbital Pursuit-Evasion Game”. In: *Journal of Guidance, Control, and Dynamics* 32.2 (2009), pp. 474–487. DOI: 10.2514/1.37962.
- [29] Y. LI, X. LIANG, and Z. DANG. “Nash-equilibrium strategies of orbital Target-Attacker-Defender game with a non-maneuvering target”. In: *Chinese Journal of Aeronautics* 37.10 (2024), pp. 365–379. ISSN: 1000-9361. DOI: <https://doi.org/10.1016/j.cja.2024.06.002>.
- [30] W. Yu, C. Liu, and X. Yue. “Reinforcement learning-based decision-making for spacecraft pursuit-evasion game in elliptical orbits”. In: *Control Engineering Practice* 153 (2024), p. 106072. ISSN: 0967-0661. DOI: <https://doi.org/10.1016/j.conengprac.2024.106072>.

- [31] G. He et al. *Detecting Satellite Maneuver Intent Using Game Theory*. Tech. rep. 2025.
- [32] O. B. Iskender. "Intelligent Game Countermeasures for Non-Cooperative Spacecraft Proximity Operations on Circular and Elliptical Orbits". In: *SSRN Electronic Journal* (2026). DOI: 10.2139/ssrn.6952642.
- [33] J. J. Kruger et al. "Adaptive End-to-End Architecture for Autonomous Spacecraft Navigation and Control During Rendezvous and Proximity Operations". In: *AIAA SCITECH 2024 Forum*. 2024. DOI: 10.2514/6.2024-0430. eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.2024-0430>.
- [34] M. C. Wijayatunga. "Trajectory Design and Guidance for Space Missions with Rendezvous and Proximity Operations". PhD thesis. University of Auckland, 2025. URL: <https://researchspace.auckland.ac.nz/items/986caf1c-1a74-4462-8393-8bc46700d6b2>.
- [35] E. G. Chang and K. Schroeder. "Reinforcement Learning for Rendezvous and Proximity Operations". In: *AIAA SCITECH 2026 Forum*. 2026. DOI: 10.2514/6.2026-1161.
- [36] Z. Zhang et al. "Convex Optimization for Rendezvous and Proximity Operation via Birkhoff Pseudospectral Method". In: *Aerospace* 9.9 (2022). ISSN: 2226-4310. DOI: 10.3390/aerospace9090505.
- [37] T. Lebeke and A. Fiorentino. "A Modular Approach for Rendezvous and Proximity Operations Missions: From Simulations to Operations". In: *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. Sept. 2022. URL: <https://amostech.space/year/2022/a-modular-approach-for-rendezvous-and-proximity-operations-missions-from-simulations-to-operations/>.
- [38] J. C. Sanchez, F. Gavilan, and R. Vazquez. "Chance-constrained Model Predictive Control for Near Rectilinear Halo Orbit spacecraft rendezvous". In: *Aerospace Science and Technology* 100 (2020), p. 105827. ISSN: 1270-9638. DOI: <https://doi.org/10.1016/j.ast.2020.105827>.
- [39] J. A. Rebollo et al. *MPC for Tracking applied to rendezvous with non-cooperative tumbling targets ensuring stability and feasibility*. 2024. arXiv: 2403.10986.
- [40] A. Garcia et al. "Electric Propulsion Intelligent Control (EPIC) Toolbox for Proximity Operations in Low-Earth Orbit (LEO)". In: *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. 2021, pp. 95–110. URL: <https://amostech.space/year/2021/electric-propulsion-intelligent-control-epic-toolbox-for-proximity-operations-in-low-earth-orbit-leo/>.
- [41] A. Shirazi, J. Ceberio, and J. A. Lozano. "Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions". In: *Progress in Aerospace Sciences* 102 (Oct. 2018), pp. 76–98. ISSN: 03760421. DOI: 10.1016/j.paerosci.2018.07.007.
- [42] C. Blum and A. Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". In: *ACM Comput. Surv.* 35.3 (Sept. 2003), pp. 268–308. ISSN: 0360-0300. DOI: 10.1145/937503.937505.
- [43] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. 2nd. Hoboken, New Jersey: John Wiley & Sons, 2004. ISBN: 0-471-45565-2. DOI: 10.1002/0471671746.
- [44] M. Rosa Sentinella and L. Casalino. "Cooperative evolutionary algorithm for space trajectory optimization". In: *Celestial Mechanics and Dynamical Astronomy* 105.1 (Nov. 2009), pp. 211–227. ISSN: 1572-9478. DOI: 10.1007/s10569-009-9223-4.
- [45] M. Pontani, P. Ghosh, and B. A. Conway. "Particle Swarm Optimization of Multiple-Burn Rendezvous Trajectories". In: *Journal of Guidance, Control, and Dynamics* 35.4 (2012), pp. 1192–1207. DOI: 10.2514/1.55592.
- [46] J. Shrager and T. Finin. "An Expert System That Volunteers Advice." In: Jan. 1982, pp. 339–340.
- [47] M. Ben-Bassat and A. Freedy. "Knowledge Requirements and Management in Expert Decision Support Systems for (Military) Situation Assessment". In: *IEEE Transactions on Systems, Man, and Cybernetics* 12.4 (1982), pp. 479–490.
- [48] W. R. Swartout. "Rule-based expert systems: The mycin experiments of the stanford heuristic programming project: B.G". In: *Artificial Intelligence* 26.3 (1985), pp. 364–366. ISSN: 0004-3702. DOI: [https://doi.org/10.1016/0004-3702\(85\)90067-0](https://doi.org/10.1016/0004-3702(85)90067-0).

- [49] S. Carberry. "Techniques for Plan Recognition". In: *User Model. User-Adapt. Interact.* 11 (), pp. 31–48. DOI: 10.1023/A:1011118925938.
- [50] Q. R. Niu et al. "Research on Intention Recognition of Air Target Based on Machine Learning". In: *ACM International Conference Proceeding Series*. Association for Computing Machinery, June 2025, pp. 386–395. ISBN: 9798400717178. DOI: 10.1145/3703935.3704116.
- [51] J. Azarewicz et al. "Template-Based Multi-Agent Plan Recognition for Tactical Situation Assessment". In: *[1989] Proceedings. The Fifth Conference on Artificial Intelligence Applications*. 1989. DOI: 10.1109/CAIA.1989.49160.
- [52] E. Charniak and R. P. Goldman. "A Bayesian Model of Plan Recognition". In: *Artificial Intelligence* 64 (1993), pp. 53–79. ISSN: 0004-3702. DOI: [https://doi.org/10.1016/0004-3702\(93\)90060-0](https://doi.org/10.1016/0004-3702(93)90060-0).
- [53] J. Inglis and G. Shafer. "A Mathematical Theory of Evidence". In: *Technometrics* 20 (1 Feb. 1978), p. 106. ISSN: 00401706. DOI: 10.2307/1268172.
- [54] M. Bauer. "A Dempster-Shafer Approach to Modeling Agent Preferences for Plan Recognition". In: *User Modeling and User-Adapted Interaction* 5.3 (1995), pp. 317–348. ISSN: 1573-1391. DOI: 10.1007/BF01126114.
- [55] D. Ju-Long. "Control problems of grey systems". In: *Systems & Control Letters* 1.5 (1982), pp. 288–294. ISSN: 0167-6911. DOI: [https://doi.org/10.1016/S0167-6911\(82\)80025-X](https://doi.org/10.1016/S0167-6911(82)80025-X).
- [56] J. Jiang et al. "A Bayesian network approach for dynamic behavior analysis: Real-time intention recognition". In: *Information Fusion* 118 (2025), p. 102873. ISSN: 1566-2535. DOI: <https://doi.org/10.1016/j.inffus.2024.102873>.
- [57] S. Zhang et al. "A Method of Inference Intention with Uncertain Aerial Information". In: *Journal of Air Force Engineering University (Natural Science Edition)* 9.3 (2008). In Chinese, pp. 50–53. DOI: 10.3969/j.issn.1009-3516.2008.03.012.
- [58] G. L. Dai et al. "Method of Target Tactical Intention Recognition Based on Interval Grey Relational Degree". In: *Mathematics in Practice and Theory* 44.20 (2014). In Chinese, pp. 198–207.
- [59] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.
- [60] C. Cortes and V. Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018.
- [61] T. Chen and C. Guestrin. *XGBoost: A Scalable Tree Boosting System*. 2016. arXiv: 1603.02754. URL: <http://arxiv.org/abs/1603.02754>.
- [62] S. P. Lloyd. "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.
- [63] J. H. Ward. "Hierarchical Grouping to Optimize an Objective Function". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244. DOI: 10.1080/01621459.1963.10500845.
- [64] M. Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 1996, pp. 226–231.
- [65] L. Grinsztajn, E. Oyallon, and G. Varoquaux. "Why do tree-based models still outperform deep learning on tabular data?" In: *arXiv preprint arXiv:2207.08815* (2022). DOI: 10.48550/arXiv.2207.08815.
- [66] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: <https://doi.org/10.1038/323533a0>.
- [67] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.
- [68] K. Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

- [69] J. Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *arXiv preprint arXiv:1412.3555* (2014). arXiv: 1412.3555.
- [70] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.
- [71] A. Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 2017, pp. 5998–6008.
- [72] H. Zhang et al. "An intention inference method for the space non-cooperative target based on BiGRU-Self Attention". In: *Advances in Space Research* 72 (5 Sept. 2023), pp. 1815–1828. ISSN: 18791948. DOI: 10.1016/j.asr.2023.04.032.
- [73] Y. Chen et al. "Orbital Behavior Intention Recognition for Space Non-Cooperative Targets Under Multiple Constraints". In: *Aerospace* 12 (6 June 2025). ISSN: 22264310. DOI: 10.3390/aerospace12060520.
- [74] Q. Sun, L. Zhao, and Z. Dang. "BiGAT: A Model for Recognizing Motion Intentions of Space Noncooperative Targets". In: *IEEE Transactions on Aerospace and Electronic Systems* 61 (2 Apr. 2025), pp. 2586–2600. ISSN: 0018-9251. DOI: 10.1109/TAES.2024.3477459.
- [75] W. Yuan et al. "An intelligent hierarchical recognition method for long-term orbital maneuvering intention of non-cooperative satellites". In: *Advances in Space Research* 75 (6 Mar. 2025), pp. 5037–5050. ISSN: 18791948. DOI: 10.1016/j.asr.2025.01.002.
- [76] A. Forootani and M. Khosravi. "Synthetic Time Series Forecasting with Transformer Architectures: Extensive Simulation Benchmarks". In: (May 2025). arXiv: 2505.20048.
- [77] C. Tallec and Y. Ollivier. *Can recurrent neural networks warp time?* 2018. arXiv: 1804.11188 [cs.LG].
- [78] J.-B. Cordonnier, A. Loukas, and M. Jaggi. *On the Relationship between Self-Attention and Convolutional Layers*. 2020. arXiv: 1911.03584 [cs.LG].
- [79] R. Balo et al. "Automatic Intent Assignment through Estimation of Proximity Maneuvers". In: *AIAA SCITECH 2026 Forum*. Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2026. ISBN: 978-1-62410-765-8. DOI: 10.2514/6.2026-2053.
- [80] H. Jing et al. "Intention Recognition of Space Noncooperative Targets Using Large Language Models". In: *Space: Science and Technology (United States)* 5 (2025). ISSN: 26927659. DOI: 10.34133/space.0271.
- [81] Y. Yu et al. "Intention recognition for space non-cooperative targets using Kolmogorov-Arnold transformer". In: *Aerospace Science and Technology* 170 (Mar. 2026). ISSN: 12709638. DOI: 10.1016/j.ast.2025.111497.
- [82] H. Zhang et al. "Intention inference for spacecraft proximity maneuver using cascaded neural network". In: *Aerospace Science and Technology* 168 (Jan. 2026). ISSN: 12709638. DOI: 10.1016/j.ast.2025.111128.
- [83] R. Balo, M. Kumar, and A. Soderlund. "Evidential Intent Assignment using Belief States in the Hill Frame". In: *2025 American Control Conference (ACC)*. IEEE, 2025, pp. 978–983. ISBN: 9798331569372. DOI: 10.23919/ACC63710.2025.11107649.
- [84] M. R. Goulet et al. "Inferring Spacecraft Maneuver Intention via Inverse Optimal Control". In: *AIAA SCITECH 2025 Forum*. 2025. DOI: 10.2514/6.2025-0982. eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.2025-0982>. URL: <https://arc.aiaa.org/doi/abs/10.2514/6.2025-0982>.
- [85] J. Li, Z. Yang, and Y. Luo. "Intention inference for space targets using deep convolutional neural network". In: *Advances in Space Research* 75 (2 Jan. 2025), pp. 2184–2200. ISSN: 18791948. DOI: 10.1016/j.asr.2024.10.006.
- [86] Q. Sun et al. "Orbital motion intention recognition for space non-cooperative targets based on incomplete time series data". In: *Aerospace Science and Technology* 158 (Mar. 2025). ISSN: 12709638. DOI: 10.1016/j.ast.2024.109912.

- [87] R. Zhao. "Intention recognition method for spatial non-cooperative target based on improved random forest". In: *Advances in Space Research* (2025). ISSN: 18791948. DOI: 10.1016/j.asr.2025.09.063.
- [88] A. G. Bala. "Developments for the Characterization of Spacecraft Proximity Operations for Improved Space Situational Awareness". PhD thesis. Virginia Polytechnic Institute and State University, Aug. 2024. URL: <https://vtechworks.lib.vt.edu/items/906a8a1c-ee3b-472a-bf1c-d8e23d502667>.
- [89] J. Yu et al. "Spatial Non-cooperative Target Behavior Intent Recognition Based on Data Generation and Deep Neural Networks". In: *Chinese Journal of Space Science* 44.6 (2024). In Chinese, pp. 1134–1146. DOI: 10.11728/cjss2024.06.2023-0151.
- [90] L. Yihan et al. "Research on Intent Recognition Method for Non-Cooperative Satellite Based on LSTM Network". In: *Proceedings - 2023 China Automation Congress, CAC 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 3854–3859. ISBN: 9798350303759. DOI: 10.1109/CAC59555.2023.10451770.
- [91] Q. Sun and Z. Dang. "Deep neural network for non-cooperative space target intention recognition". In: *Aerospace Science and Technology* 142 (Nov. 2023). ISSN: 12709638. DOI: 10.1016/j.ast.2023.108681.
- [92] T. Kelecý et al. "Predicted Intent Inferred from Real-time Rendezvous and Proximity Behavior". In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. The Stratagem Group. 2022. URL: <https://amostech.com/TechnicalPapers/2022/Conjunction-RPO/Kelecý.pdf>.
- [93] L.-j. Wu and H. Zhang. "Threat Analysis Methods of Actively Approaching Spacecraft". In: *Modern Defense Technology* 50.6 (2022), pp. 11–18. DOI: 10.3969/j.issn.1009-086x.2022.06.002.
- [94] W. H. CLOHESSY and R. S. WILTSHIRE. "Terminal Guidance System for Satellite Rendezvous". In: *Journal of the Aerospace Sciences* 27 (9 Sept. 1960), pp. 653–658. DOI: 10.2514/8.8704.
- [95] H. Djojodihardjo and A. S. M. Harithuddin. "Spacecraft Formation Flying for Tropical Resources and Environmental Monitoring: A Parametric Study". In: *Advances in the Astronautical Sciences*. Vol. 138. AAS 10-441. 2010, pp. 433–450.
- [96] M. Chang and K. A. Bingen. *New Rhythms in GEO: A Quantitative Analysis of Unusual Behavior in Geosynchronous Orbit by Chinese Satellites, 2016–2025*. Report. Center for Strategic and International Studies (CSIS), Apr. 2026. URL: <https://www.csis.org/analysis/new-rhythms-geo-quantitative-analysis-unusual-behavior-geosynchronous-orbit-chinese> (visited on 2026-08-12).
- [97] S. Erwin. *Rocket Lab launches satellite for U.S. Space Force Victus Haze responsive space exercise*. SpaceNews. June 2026. URL: <https://spacenews.com/rocket-lab-launches-satellite-for-u-s-space-force-victus-haze-responsive-space-exercise/> (visited on 2026-09-16).
- [98] *True Anomaly Details 'Space Pursuit' in Victus Haze Mission*. Via Satellite. July 2026. URL: <https://www.satellitetoday.com/government-military/2026/07/29/true-anomaly-details-space-pursuit-in-victus-haze-mission/> (visited on 2026-09-16).
- [99] *Space tag: Jackal, Puma spacecraft to chase each other in Victus Haze demo*. Breaking Defense. July 2026. URL: <https://breakingdefense.com/2026/07/space-tag-jackal-puma-spacecraft-to-chase-each-other-victus-haze-demo/> (visited on 2026-09-16).
- [100] W. Fehse. "The drivers for the approach strategy". In: *Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press, Dec. 2009, pp. 112–170. DOI: 10.1017/cbo9780511543388.006.
- [101] B. Rhodes. *Python SGP4*. URL: <https://github.com/brandon-rhodes/python-sgp4> (visited on 2026-08-27).
- [102] D. A. Vallado et al. "Revisiting Spacetrack Report #3". In: *AIAA/AAS Astrodynamics Specialist Conference*. Keystone, CO, 2006. DOI: 10.2514/6.2006-6753.

- [103] D. Dirkx et al. "The open-source astrodynamics Tudatpy software – overview for planetary mission design and science analysis". In: *Europlanet Science Congress 2022*. 2022, EPSC2022–253. DOI: 10.5194/epsc2022-253.
- [104] D. A. Vallado. *Fundamentals of Astrodynamics and Applications*. 5th ed. Microcosm Press, 2022. ISBN: 978-1881883210. URL: <https://microcosmpress.com/vallado/>.
- [105] M. I. Skolnik. *Introduction to Radar Systems*. 3rd ed. Boston: McGraw-Hill, 2001. ISBN: 9780072881387. URL: <https://archive.org/details/introductiontora0000merr>.
- [106] National Aeronautics and Space Administration. *11.0 Ground Data Systems and Mission Operations*. State-of-the-Art of Small Spacecraft Technology. May 2026. URL: <https://www.nasa.gov/smallsat-institute/sst-soa/ground-data-systems-and-mission-operations/> (visited on 2026-09-01).
- [107] Sodern. *HICAM – 1 MP Space Detection & Navigation Camera*. Sodern. URL: <https://www.satnow.com/products/satellite-cameras/sodern/113-1193-hicam> (visited on 2026-07-02).
- [108] AAC Clyde Space. *IM200: 4 MP Optical Imager for Nano Satellites – Datasheet*. AAC Clyde Space. URL: <https://www.aac-clyde.space/wp-content/uploads/2021/11/IM200.pdf> (visited on 2026-07-02).
- [109] AAC Clyde Space. *ST200-1: 4 MP Optical Imager for Nano Satellites – Datasheet*. AAC Clyde Space. URL: <https://www.aac-clyde.space/wp-content/uploads/2021/11/ST200-1.pdf> (visited on 2026-07-02).
- [110] Sodern. *AuriCam Line-Up – D35, D80, D8 – Datasheet*. Sodern. URL: https://api.sodern.com/uploads/AURICAM_8_P_5784b5f538.pdf (visited on 2026-07-02).
- [111] M. R. Ackermann et al. "A Systematic Examination of Ground-Based and Space-Based Approaches to Optical Detection and Tracking of Artificial Satellites". In: *31st Space Symposium, Technical Track*. Sandia National Laboratories report. Colorado Springs, CO, USA, 2015. URL: <https://www.osti.gov/servlets/purl/1253293> (visited on 2026-07-02).
- [112] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.
- [113] D. Izzo. "Revisiting Lambert's Problem". In: *Celestial Mechanics and Dynamical Astronomy* 121.1 (2015), pp. 1–15. DOI: 10.1007/s10569-014-9587-y.
- [114] R. H. Gooding. "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem". In: *Celestial Mechanics and Dynamical Astronomy* 48.2 (1990), pp. 145–165. DOI: 10.1007/BF00049511.
- [115] F. Biscani and D. Izzo. "A parallel global multiobjective framework for optimization: pagmo". In: *Journal of Open Source Software* 5.53 (2020), p. 2338. DOI: 10.21105/joss.02338. URL: <https://doi.org/10.21105/joss.02338>.
- [116] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Berlin: Springer, 2005. ISBN: 978-3-540-20950-8. DOI: 10.1007/3-540-31306-0.
- [117] M. Clerc. *Particle Swarm Optimization*. London: ISTE Ltd and John Wiley & Sons, 2006, p. 244. ISBN: 978-1-905-20904-0. DOI: 10.1002/9780470612163. URL: <https://onlinelibrary.wiley.com/doi/book/10.1002/9780470612163>.
- [118] N. Hansen. "The CMA Evolution Strategy: A Comparing Review". In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Ed. by J. A. Lozano et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. ISBN: 978-3-540-32494-2. DOI: 10.1007/3-540-32494-1_4.
- [119] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.
- [120] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

- [121] R. P. Goldman, C. W. Geib, and C. A. Miller. "A New Model of Plan Recognition". In: *Proceedings of the International Conference on Autonomous Agents*. 1999.
- [122] M. Sokolova and G. Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing & Management* 45.4 (July 2009), pp. 427–437. doi: 10.1016/j.ipm.2009.03.002.
- [123] T. Flohrer, H. Krag, and H. Klinkrad. "Assessment and Categorization of TLE Orbit Errors for the US SSN Catalogue". In: *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. Maui, HI, USA: The Maui Economic Development Board, 2008.
- [124] D. Ly, R. Lucken, and D. Giolito. "Correcting TLEs at epoch: Application to the GPS constellation". In: *Journal of Space Safety Engineering* 7.3 (2020). Space Debris: The State of Art, pp. 302–306. ISSN: 2468-8967. DOI: <https://doi.org/10.1016/j.jsse.2020.07.032>.
- [125] S. M. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4765–4774. URL: <https://mlanthology.org/neurips/2017/lundberg2017neurips-unified/>.

*“Earth is the cradle of humanity, but
one cannot live in a cradle forever.”*

Konstantin Eduardovich Tsiolkovsky,
*The Exploration of Cosmic Space
by Means of Reaction Devices* —
1911/1912



Additional Information

This appendix collects supporting material referred to from the main body but not essential to the line of argument there. Section A.1 describes the metaheuristic optimisers used for synthetic trajectory generation, giving for each the underlying update rules and the corresponding `pygmo` configuration as it was run. Section A.2 presents further lead-up optimisation results for a single LEO seed across all four RPO intent classes, complementing the examples shown in the main text. Section A.3 documents the covariance correlation analysis used to justify the noise model, based on operational ephemeris data for objects in both orbital domains.

A.1. Optimisers

The optimisers below are presented in the order in which they appear in the main text. Each subsection gives the `pygmo` configuration in pseudocode, and some also additional context.

A.1.1. Non-dominated Sorting Genetic Algorithm II (NSGA-II)

The full functioning of the NSGA-II algorithm is described in Chapter 5.

Algorithm 1 NSGA-II — pygmo configuration

Require: Population size N , generations per chunk g , crossover rate c_r , crossover distribution index η_c , mutation rate m , mutation distribution index η_m

```

1: Initialize  $N$  individuals and evaluate all objective vectors
2: Perform fast non-dominated sorting and assign crowding distance
3: for  $k = 1, 2, \dots, g$  do
4:                                     ▷ Selection — tournament
5:   Select parents via binary tournament using non-domination rank as the primary criterion and
   crowding distance as the secondary criterion
6:                                     ▷ Crossover — simulated binary crossover (SBX)
7:   for each consecutive parent pair do
8:     With probability  $c_r$ , apply SBX with distribution index  $\eta_c$  to generate offspring
9:   end for
10:                                     ▷ Mutation — polynomial mutation
11:   for each offspring do
12:     With probability  $m$ , apply polynomial mutation with distribution index  $\eta_m$ 
13:   end for
14:   Evaluate objective vectors of all offspring
15:                                     ▷ Replacement — elitist environmental selection
16:   Combine parent and offspring populations
17:   Perform fast non-dominated sorting
18:   Fill the next generation front-by-front until the population size is reached
19:   If the last accepted front exceeds the remaining capacity, rank its individuals by descending
   crowding distance and retain the most widely spaced solutions
20: end for
21: return Final non-dominated front(s)

```

A.1.2. Simple Genetic Algorithm (SGA)

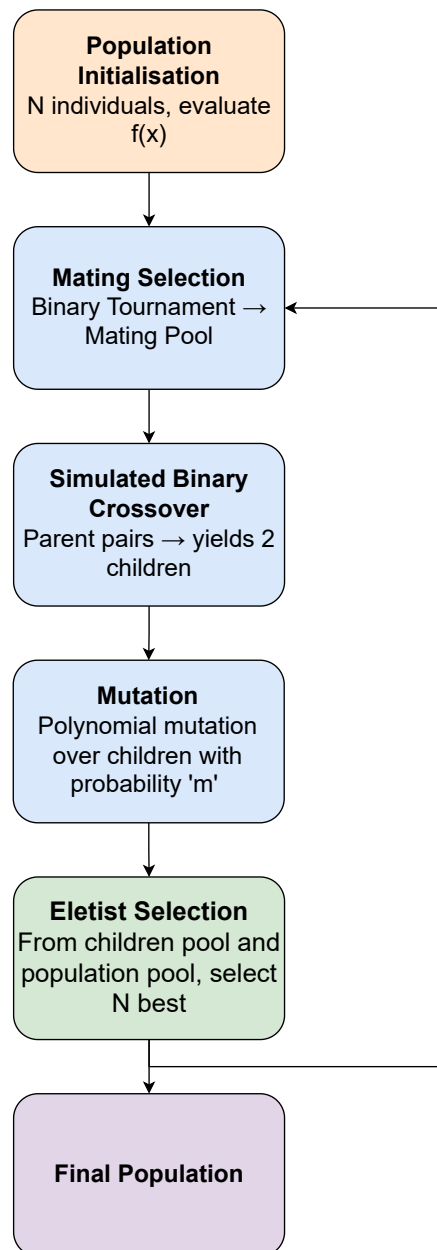


Figure A.1: SGA flowchart

The SGA algorithm is one of the first evolutionary algorithms developed and, as the name implies, one of the simplest considered [43]. The implementation follows a generational cycle: an initial population undergoes binary tournament selection to form a mating pool, after which simulated binary crossover (SBX) and polynomial mutation are applied sequentially to generate offspring before population replace-

ment.

In binary tournament selection, two individuals are drawn at random from the population, and the fitter of the two is copied into the mating pool; this is repeated until the mating pool matches the population size. Since only relative fitness within each pair matters, the selection pressure is largely insensitive to the absolute scale or distribution of the fitness values.

Offspring are generated from the mating pool using SBX, which mimics the behaviour of single-point crossover in binary-encoded GAs but operates directly on continuous decision variables. For two parent values x_1, x_2 , a spread factor β is sampled from a distribution controlled by a user-defined distribution index η_c , and the two offspring are generated as

$$x'_1 = \frac{1}{2}[(1 + \beta)x_1 + (1 - \beta)x_2], \quad x'_2 = \frac{1}{2}[(1 - \beta)x_1 + (1 + \beta)x_2]. \quad (\text{A.1})$$

A higher η_c concentrates offspring closer to the parents, while a lower value permits more exploratory offspring further from the parent pair.

Following crossover, polynomial mutation perturbs each decision variable independently with a small probability, using a distribution shaped by a mutation distribution index η_m in the same manner as η_c shapes crossover; this maintains genetic diversity in the population and helps prevent premature convergence to a local optimum.

Population replacement follows a pure elitist strategy: the parent and offspring populations are pooled together and ranked by fitness, with only the best individuals retained to form the next generation. This guarantees that the best solution found so far is never lost between generations, at the cost of a higher risk of premature convergence compared to non-elitist replacement schemes, since fitter individuals can dominate the mating pool across many consecutive generations.

Algorithm 2 Simple Genetic Algorithm (SGA) — pygmo configuration

Require: Population size N , generations per chunk g , crossover rate $c_r = 0.9$, mutation rate $m = 0.15$, mutation distribution index $\eta_m = 20.0$

```

1: Initialize  $N$  individuals; evaluate fitness  $f(x_i)$  for  $i = 1, \dots, N$ 
2: for  $k = 1, 2, \dots, g$  do
3:                                     ▷ Selection — tournament
4:   Select  $N$  individuals via binary tournament selection based on fitness to form the mating pool
5:                                     ▷ Crossover — simulated binary crossover (SBX)
6:   for each consecutive pair  $(x_i, x_{i+1})$  in mating pool do
7:     With probability  $c_r = 0.9$ , apply SBX to produce two offspring
8:   end for
9:                                     ▷ Mutation — polynomial
10:  for each offspring  $x_i$  do
11:    With probability  $m$ , apply polynomial mutation with distribution parameter  $\eta_m$ 
12:  end for
13:                                     ▷ Replacement
14:  Select new population pool based upon elitist selection of parents and offspring
15:  Evaluate fitness  $f(x_i)$  for all new individuals
16: end for

```

A.1.3. Self-Adaptive Differential Evolution (SADE)

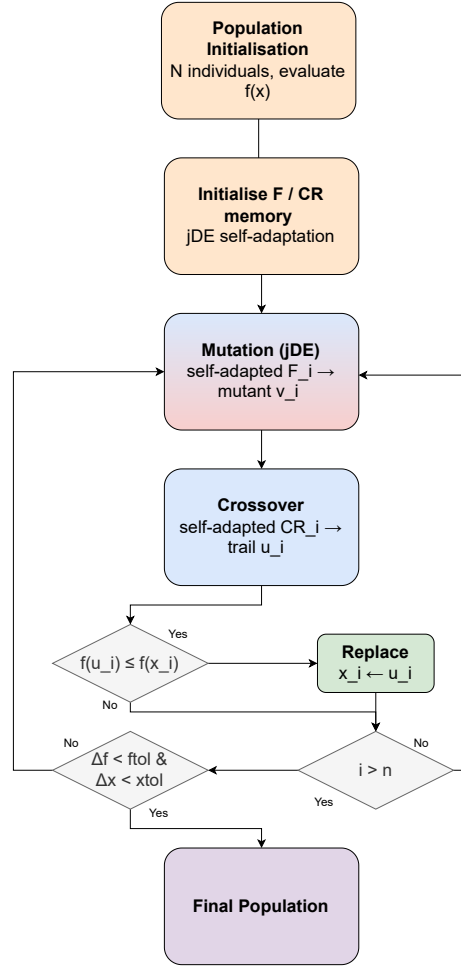


Figure A.2: SADE flowchart

SADE extends classic differential evolution by self-adapting its control parameters, allowing the mutation scale factor F and crossover rate CR to evolve alongside the population rather than being fixed, eliminating the need for manual tuning during execution. For each individual, a mutant vector is generated from randomly selected population members, combined with the current individual via exponential crossover to form a trial vector, and accepted only if it improves on the parent's fitness through a greedy one-to-one replacement.

The mutant vector for the i -th individual is constructed as

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}), \quad (\text{A.2})$$

where r_1, r_2, r_3 are indices of three distinct individuals drawn at random from the population, all different from i . Rather than fixing F and CR for the whole run, each individual carries its own pair of values, which are themselves subject to a small probability of random reinitialisation at each generation; individuals whose trial vectors are successfully accepted pass their F and CR values on to the next generation, while unsuccessful values are more likely to be replaced. Over successive generations, this biases the population towards F and CR values that have empirically produced improving trial vectors, without requiring these parameters to be tuned in advance for the specific objective landscape.

Exponential crossover then constructs the trial vector by copying a contiguous run of components from the mutant vector into the parent, starting from a randomly chosen index and continuing for a length controlled probabilistically by CR : a higher CR increases the expected length of the copied segment, producing trial vectors that draw more heavily on the mutant. This differs from the component-wise (binomial) crossover used in the SGA's SBX operator, in that the inherited components form a connected block along the decision vector rather than being selected independently at each position.

The algorithm terminates early once the population's spread in both fitness and decision variables falls below the specified tolerances, indicating that the population has effectively converged and further generations are unlikely to yield meaningful improvement.

Algorithm 3 Self-Adaptive Differential Evolution (SADE) — pygmo configuration

Require: Population size N , generations per chunk g , self-adaptive variant parameters (variant_adptv = 1), $f_{\text{tol}} = 10^{-6}$, $x_{\text{tol}} = 10^{-6}$

- 1: Initialize N individuals; evaluate fitness $f(x_i)$ for $i = 1, \dots, N$
- 2: Initialize learning memories for F and CR adaptation strategies
- 3: **for** $k = 1, 2, \dots, g$ **do**
- 4: **for** each individual x_i , $i = 1, \dots, N$ **do**
- 5: $\triangleright$ Mutation — jDE Variant (variant 2)
- 6: Select distinct random base vectors from the population based on chosen strategy variant
- 7: Generate mutant vector v_i using self-adapted scale factor F_i
- 8: $\triangleright$ Crossover — exponential
- 9: Apply exponential crossover to (x_i, v_i) using crossover rate CR_i to form trial vector u_i
- 10: $\triangleright$ Selection
- 11: **if** $f(u_i) \leq f(x_i)$ **then**
- 12: $x_i \leftarrow u_i$ $\triangleright$ Greedy one-to-one replacement
- 13: **end if**
- 14: **end for**
- 15: $\triangleright$ Convergence check
- 16: **if** $\max_i |f(x_i) - f(x_{\text{best}})| < f_{\text{tol}}$ **and** $\max_{i,j} |x_{i,j} - x_{\text{best},j}| < x_{\text{tol}}$ **then**
- 17: **break**
- 18: **end if**
- 19: **end for**

A.1.4. Particle Swarm Optimisation (PSO)

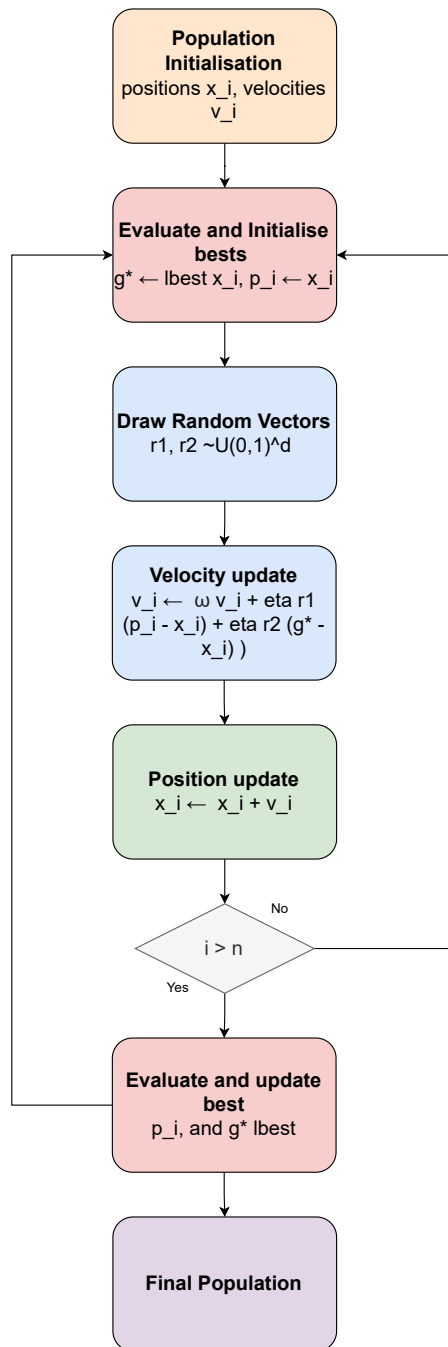


Figure A.3: PSO flowchart

PSO models the population as a swarm of particles moving through the search space, each carrying a position and velocity rather than undergoing genetic operators. At every generation, each particle's velocity is updated using an inertia term, a cognitive term pulling it towards its own best-known position,

and a social term pulling it towards the best position found across the swarm (or its neighbourhood), before the velocity is clamped to a maximum magnitude and the particle's position is updated accordingly.

For particle i at generation k , the velocity update is

$$\mathbf{v}_i^{(k)} = w \mathbf{v}_i^{(k-1)} + c_1 r_1 (\mathbf{p}_i^{(k-1)} - \mathbf{x}_i^{(k-1)}) + c_2 r_2 (\mathbf{g}^{(k-1)} - \mathbf{x}_i^{(k-1)}), \quad (\text{A.3})$$

where $\mathbf{x}_i$ is the particle's current position, $\mathbf{p}_i$ its own best-known position (personal best), and $\mathbf{g}$ the best position found by the swarm, or by its neighbourhood if a non-global topology is used. The inertia weight w controls how much of the particle's previous velocity is retained, and governs the trade-off between exploration and exploitation: a higher w favours continued movement in the current direction, while a lower w allows the cognitive and social terms to dominate. The cognitive and social coefficients c_1 and c_2 weight the pull towards the particle's own best position and the swarm's best position respectively, and $r_1, r_2 \sim \mathcal{U}(0, 1)$ are independent random numbers redrawn at each generation, introducing stochasticity into an otherwise deterministic update.

The updated velocity is then clamped component-wise to a maximum magnitude $v_{\max}$,

$$v_{i,j} \leftarrow \max(-v_{\max,j}, \min(v_{i,j}, v_{\max,j})), \quad (\text{A.4})$$

which prevents particles from overshooting the search space in a single step, a risk that grows when the social and cognitive terms reinforce each other in the same direction. The particle's position is then updated by simple discrete update,

$$\mathbf{x}_i^{(k)} = \mathbf{x}_i^{(k-1)} + \mathbf{v}_i^{(k)}. \quad (\text{A.5})$$

Personal and global bests are refreshed only once all particles in the swarm have moved for the current generation, rather than incrementally as each particle updates; this keeps $\mathbf{g}$ fixed across a full generation and avoids the swarm's collective target shifting mid-update, at the cost of the improved best position only becoming visible to other particles in the following generation. The cycle of velocity update, clamping, position update, and best-refresh then repeats for the specified number of generations.

Algorithm 4 Particle Swarm Optimisation (PSO) — pygmo configuration

Require: Swarm size N , generations per chunk g , inertia weight $\omega = 0.7$, cognitive component $\eta_1 = 2.0$, social component $\eta_2 = 2.0$, maximum velocity limit $v_{\max} = 0.2$

- 1: Initialize particle positions x_i and velocities v_i within bounds for $i = 1, \dots, N$
 - 2: Evaluate fitness $f(x_i)$; set personal bests $p_i \leftarrow x_i$ and global best $g^* \leftarrow \arg \min_i f(p_i)$
 - 3: **for** $k = 1, 2, \dots, g$ **do**
 - 4: **for** each particle i **do**
 - 5: Draw random vectors $r_1, r_2 \sim \mathcal{U}(0, 1)^d$
 - 6: $\triangleright$ Velocity update with max velocity capping
 - 7: $v_i \leftarrow \omega v_i + \eta_1 r_1 \odot (p_i - x_i) + \eta_2 r_2 \odot (g^* - x_i)$
 - 8: Apply constraint check: $v_i \leftarrow \max(-v_{\max}, \min(v_{\max}, v_i))$
 - 9: $\triangleright$ Position update
 - 10: $x_i \leftarrow x_i + v_i$
 - 11: **end for**
 - 12: Evaluate $f(x_i)$ for all particles
 - 13: Update personal bests p_i and global best g^*
 - 14: **end for**
-

A.1.5. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

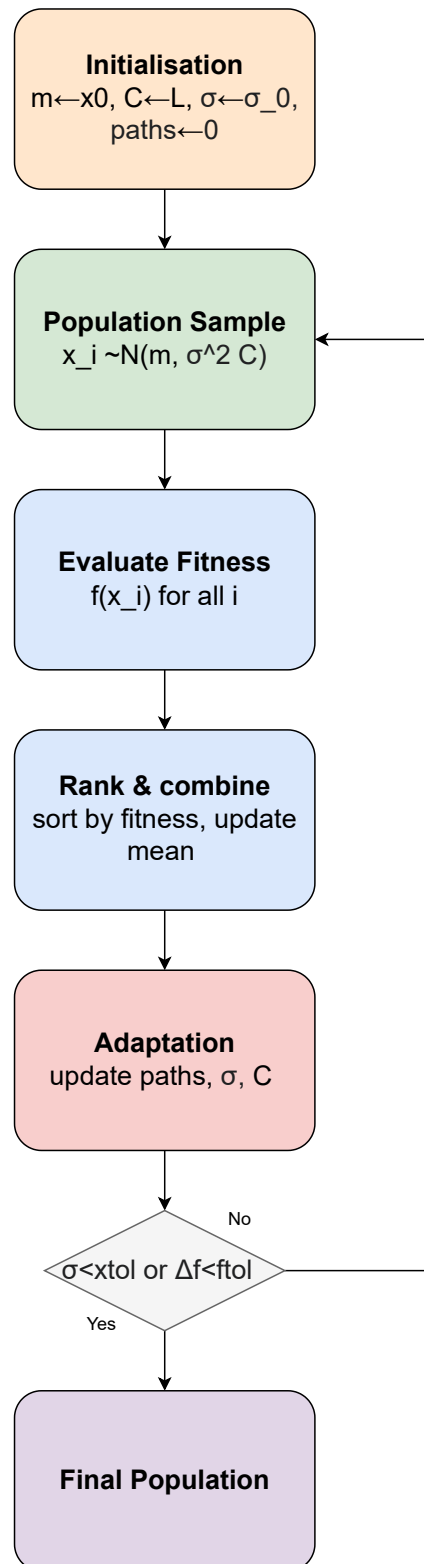


Figure A.4: CMA-ES flowchart

CMA-ES departs from the other algorithms in that it maintains no population between generations; instead, it samples an entire offspring population directly from a multivariate Gaussian distribution parameterised by a mean and covariance matrix. The offspring are ranked by fitness and used to update the distribution's mean via weighted recombination, after which the evolution paths, step size, and covariance matrix are adapted to reshape the sampling distribution for the next generation.

At generation g , λ offspring are sampled as

$$\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)} \mathbf{y}_k, \quad \mathbf{y}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}), \quad k = 1, \dots, \lambda, \quad (\text{A.6})$$

where $\mathbf{m}^{(g)}$ is the current distribution mean, $\sigma^{(g)}$ the step size, and $\mathbf{C}^{(g)}$ the covariance matrix, which encodes both the scale and the orientation of the search distribution across the decision variables. After ranking the λ offspring by fitness, only the best μ are retained for recombination; the new mean is a weighted average of these,

$$\mathbf{m}^{(g+1)} = \sum_{k=1}^{\mu} w_k \mathbf{x}_{k:\lambda}, \quad (\text{A.7})$$

where $\mathbf{x}_{k:\lambda}$ denotes the k -th best of the λ offspring and the weights w_k decrease with rank, so fitter offspring contribute more strongly to the new mean.

Two evolution paths are then updated: a step-size path $\mathbf{p}_\sigma$, which accumulates the direction of successive mean shifts and is used to adapt σ (increasing it when consecutive steps point in similar directions, indicating the distribution is moving too cautiously, and decreasing it when steps cancel out, indicating overshoot), and a covariance path $\mathbf{p}_c$, which accumulates correlated movement of the mean over several generations and is used to update $\mathbf{C}$ so that the sampling distribution stretches along directions that have recently proven productive. This adaptation is what allows CMA-ES to implicitly learn the local curvature and correlation structure of the objective landscape, rather than sampling with a fixed or isotropic step, which is particularly relevant given the non-smooth, discontinuous penalty structure of the objective in this problem.

The algorithm terminates early once the step size σ or the change in objective value falls below the specified tolerances, indicating that the sampling distribution has effectively collapsed onto a local optimum and further generations are unlikely to move it meaningfully.

Algorithm 5 Covariance Matrix Adaptation Evolution Strategy (CMA-ES) — pygmo configuration

Require: Initial mean solution x_0 , initial step size $\sigma_0 = 0.2$, generations per chunk g , stopping tolerances $f_{\text{tol}} = 10^{-6}$, $x_{\text{tol}} = 10^{-6}$

- 1: Initialize distribution mean $m \leftarrow x_0$, covariance matrix $C \leftarrow I$, and evolution paths
 - 2: **for** $k = 1, 2, \dots, g$ **do**
 - 3: ▷ Sample offspring
 - 4: Draw candidate solutions from mutation distribution: $x_i \sim \mathcal{N}(m, \sigma^2 C)$
 - 5: Evaluate fitness $f(x_i)$ for all generated individuals
 - 6: ▷ Rank and Recombine
 - 7: Sort population by fitness and update weighted mean distribution m
 - 8: ▷ Adaptation
 - 9: Update evolution paths, step-size σ , and covariance matrix C
 - 10: ▷ Convergence check
 - 11: **if** $\sigma < x_{\text{tol}}$ **or** change in objective function value $< f_{\text{tol}}$ **then**
 - 12: **break**
 - 13: **end if**
 - 14: **end for**
-

A.2. Lead-up Optimisation Results

The following figures show example lead-up optimisation results for a single LEO seed across all four RPO intent classes, supplementing the examples given in the main text.

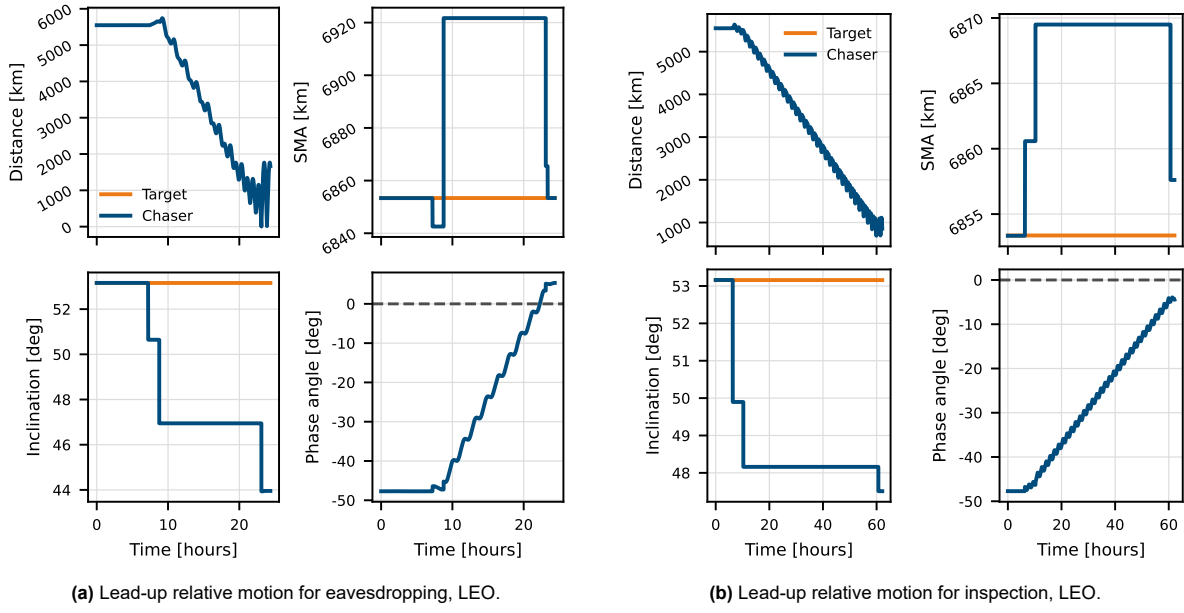


Figure A.5: Example lead-up optimisation results for eavesdropping and inspection, LEO, seed 28.

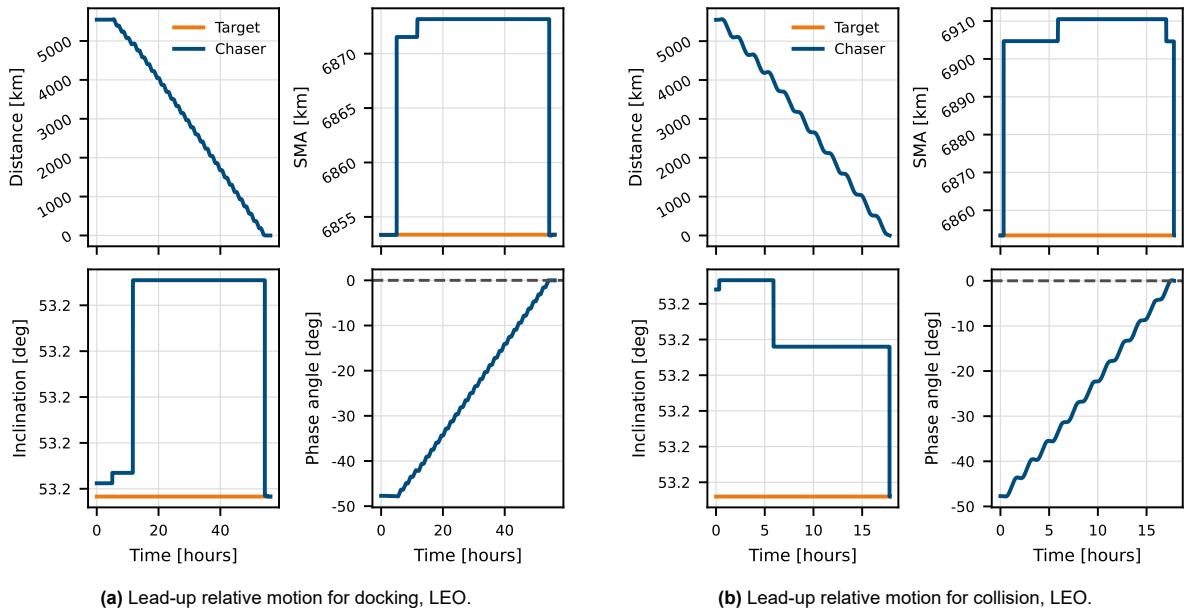


Figure A.6: Example lead-up optimisation results for docking and collision, LEO, seed 28.

A.3. Noise Correlation

Wayfinder Privateer Space samples are used, sourced from <https://wayfinder.privateer.com/>, retrieved 31-07-2026. The correlation matrix of the covariance is assessed for a set of satellites in both LEO and GEO, to determine whether the components of the state vector display any degree of correlation.

Table A.1: NORAD IDs of objects used per orbital domain

Domain	NORAD IDs
GEO	24936, 25153, 26388, 27380, 27691, 28446, 29162, 32299, 32763, 34713, 38978, 41471, 42709, 43271, 43867, 45246, 51850, 53961, 58204, 61503, 63924
LEO	49027, 51883, 52867, 53170, 57629, 58605, 58864, 59199, 61064, 61867, 62047, 63680, 65020, 66418, 66546, 66623, 67838, 68035, 68167, 68897, 69292, 69418

For each satellite listed in Table A.1, an Orbital Ephemeris Message (OEM) was obtained. Each OEM provides the satellite's state vector at a series of epochs, together with the associated 6×6 covariance matrix at those epochs. As the covariance matrix is symmetric, only its lower triangular part is stored in the file; the full matrix is reconstructed from this before further processing.

For each epoch at which a covariance matrix is available, the corresponding state vector is used to rotate the covariance matrix from the inertial frame into the local RSW/LVLH frame. The correlation matrix is then obtained by normalising this rotated covariance matrix with the standard deviations along its diagonal. Correlation matrices are computed for every epoch of every satellite, and then averaged across all epochs and all satellites within a given orbital domain to yield a single representative correlation matrix per domain.

$$\mathbf{CORR}_{\text{LEO}} = \begin{bmatrix} 1.00 & -0.07 & 0.00 & 0.65 & -0.16 & 0.00 \\ -0.07 & 1.00 & 0.00 & -0.38 & 0.20 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.11 \\ 0.65 & -0.38 & 0.00 & 1.00 & 0.41 & 0.00 \\ -0.16 & 0.20 & 0.00 & 0.41 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.11 & 0.00 & 0.00 & 1.00 \end{bmatrix} \quad (\text{A.8})$$

$$\mathbf{CORR}_{\text{GEO}} = \begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.26 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.24 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.23 \\ 0.26 & 0.00 & 0.00 & 1.00 & 0.06 & 0.00 \\ 0.00 & 0.24 & 0.00 & 0.06 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.23 & 0.00 & 0.00 & 1.00 \end{bmatrix} \quad (\text{A.9})$$

B

Verification

This appendix contains unit and integration tests of the data generation code. Section B.1 contains the unit test on the orbital propagation models used and the infrastructure around them for the approach trajectory. Section B.2 contains unit test related to the approach trajectory objectives. Finally, Section B.3 contains unit tests related to the mission phase optimisation.

B.1. Simulation Framework Unit Tests

B.1.1. Initial State Construction

`test_returns_array_with_correct_shape` Verifies that `build_initial_state` returns a state vector of length 12 for two bodies. Inputs: Keplerian elements at 400 km and 800 km altitude. Expected output: state vector of shape (12,).

`test_single_body` Verifies that `build_initial_state` returns a state vector of length 6 for a single body. Input: Keplerian elements at 400 km altitude. Expected output: state vector of shape (6,).

`test_position_magnitude_matches_semi_major_axis` Checks that the constructed position vector's magnitude equals the semi-major axis for a circular orbit, to within 1 m. Input: Keplerian elements at 500 km altitude. Expected output: $|\mathbf{r}| = R_E + 500\,000$ m, to within 1 m.

`test_velocity_magnitude_circular_orbit` Checks that the constructed velocity vector's magnitude matches the analytical circular-orbit speed, to within 0.1 m/s. Input: Keplerian elements at 400 km altitude. Expected output: $|\mathbf{v}| = \sqrt{\mu_E / (R_E + 400\,000)}$, to within 0.1 m/s.

`test_multiple_bodies_have_distinct_positions` Confirms that two bodies at different altitudes are assigned distinct position vectors. Inputs: Keplerian elements at 400 km and 800 km altitude. Expected output: the two position sub-vectors are not equal.

B.1.2. Keplerian Propagation

`test_zero_time_returns_initial_state` Confirms that propagating over a zero-length time interval returns the initial state unchanged. Input: circular state at 400 km altitude. Expected output: propagated state equal to the initial state, to within 10^{-6} .

`test_energy_conserved` Verifies that specific orbital energy is conserved after propagating a quarter period. Input: circular state at 400 km altitude. Expected output: $E_0 = E_1$, to within a relative tolerance of 10^{-6} .

`test_radius_constant_for_circular_orbit` Checks that the orbital radius remains constant (to within 1 m) at 10%, 25%, 50%, and 75% of the orbital period. Input: circular state at 400 km altitude.

Expected output: radius equal to $R_E + 400\,000$ m at each of the four fractions, to within 1 m.

test_one_full_period_returns_to_start Confirms that propagating over one full orbital period returns the state to its initial value, to within 1 m in position and 1 mm/s in velocity. Input: circular state at 400 km altitude. Expected output: final position within 1 m and final velocity within 1 mm/s of the initial state.

test_output_shape Verifies that the propagated state has shape (6,). Input: circular state at 400 km altitude, propagated over 3600 s. Expected output: state vector of shape (6,).

test_non_zero_start_time_equivalent_to_time_shift Confirms that propagation is time-shift invariant, i.e. propagating from $t = 0$ to $t = \Delta t$ gives the same result as propagating from $t = T_{\text{offset}}$ to $t = T_{\text{offset}} + \Delta t$. Inputs: circular state at 400 km altitude, $\Delta t = 600$ s, $T_{\text{offset}} = 1800$ s. Expected output: the two propagated states are equal, to within 10^{-6} .

B.1.3. Inertial--LVLH Frame Transformation

test_zero_relative_state_maps_to_zero Confirms that a zero relative position and velocity map to zero in the LVLH frame. Input: chief state at 400 km altitude. Expected output: $\delta \mathbf{r}_{\text{LVLH}} = 0$, $\delta \mathbf{v}_{\text{LVLH}} = 0$, to within 10^{-9} .

test_roundtrip_inertial_lvlh_inertial Verifies that transforming a relative state from the inertial frame to LVLH and back recovers the original state. Inputs: chief state at 400 km altitude; relative position [100, 0, 0] m and velocity [0, 0.1, 0] m/s. Expected output: recovered relative position and velocity equal to the original inputs, to within 10^{-9} .

test_roundtrip_lvlh_inertial_lvlh Verifies that transforming a relative state from LVLH to the inertial frame and back recovers the original state. Inputs: chief state at 400 km altitude; LVLH relative position [50, -30, 10] m and velocity [0.05, 0.02, -0.01] m/s. Expected output: recovered LVLH relative position and velocity equal to the original inputs, to within 10^{-9} .

test_radial_offset_points_along_r Checks that a purely radial LVLH offset maps to an inertial vector parallel to $\hat{r}$. Inputs: chief state at 400 km altitude; radial LVLH offset [100, 0, 0] m. Expected output: $\delta \mathbf{r}_{\text{inertial}} \times \hat{r} = 0$, to within 10^{-6} .

test_norm_preserved Confirms that the Euclidean norm of the relative position and velocity vectors is preserved under the inertial-to-LVLH transformation. Inputs: chief state at 400 km altitude; relative position [200, -100, 50] m and velocity [0.2, -0.1, 0.05] m/s. Expected output: $\|\delta \mathbf{r}_{\text{LVLH}}\| = \|\delta \mathbf{r}_{\text{inertial}}\|$ and $\|\delta \mathbf{v}_{\text{LVLH}}\| = \|\delta \mathbf{v}_{\text{inertial}}\|$, to 6 decimal places.

B.1.4. Impulsive Manoeuvres

test_zero_delta_v_unchanged Confirms that applying a zero Δv leaves the state unchanged. Input: circular state at 400 km altitude. Expected output: state unchanged (exact equality).

test_position_unchanged Verifies that applying a Δv does not alter the position components of the state. Inputs: circular state at 400 km altitude; $\Delta v = [100, 0, 0]$ m/s. Expected output: position components unchanged (exact equality).

test_velocity_incremented_correctly Checks that the velocity components are incremented by exactly Δv . Inputs: circular state at 400 km altitude; $\Delta v = [10, -5, 3]$ m/s. Expected output: new velocity equals original velocity plus Δv , to within 10^{-12} .

test_output_shape_preserved Confirms that the returned state has the same shape as the input state. Inputs: circular state at 400 km altitude; $\Delta v = [1, 2, 3]$ m/s. Expected output: output shape equal to input shape.

test_prograde_burn_increases_speed Verifies that a prograde Δv of 100 m/s increases the orbital speed. Input: circular state at 400 km altitude. Expected output: $\|\mathbf{v}_{\text{new}}\| > \|\mathbf{v}\|$.

test_retrograde_burn_decreases_speed Verifies that a retrograde Δv of 100 m/s decreases the orbital speed. Input: circular state at 400 km altitude. Expected output: $\|\mathbf{v}_{\text{new}}\| < \|\mathbf{v}\|$.

B.1.5. Lambert-Targeted Transfers

`test_returns_two_vectors` Confirms that the Lambert solver returns two three-component Δv vectors. Inputs: departure/arrival states at 400 km and 800 km altitude; transfer time $T = T_{\text{Hohmann}}/1.1$. Expected output: Δv_1 and Δv_2 each of shape (3,).

`test_zero_transfer_time_raises` Verifies that a zero transfer time raises an exception (`ValueError`, `ZeroDivisionError`, or `RuntimeError`). Input: circular state at 400 km altitude. Expected output: one of the three listed exceptions is raised.

`test_hohmann_dv_magnitude` Checks that the solved Δv magnitudes match the analytical Hohmann-transfer values to within 1 m/s. Inputs: departure/arrival states at 400 km and 800 km altitude; Hohmann transfer time. Expected output: $\|\Delta v_1\|$ and $\|\Delta v_2\|$ match the analytical Hohmann values, to within 1 m/s.

`test_non_hohmann_dv_magnitude` Confirms that a non-Hohmann (faster) transfer requires strictly greater Δv than the Hohmann case at both ends. Inputs: departure/arrival states at 400 km and 800 km altitude; transfer time $T_{\text{Hohmann}}/1.1$. Expected output: $\|\Delta v_1\|$ and $\|\Delta v_2\|$ strictly exceed the corresponding Hohmann values.

B.1.6. End-to-End Rendezvous Consistency

`test_non_hohmann_trajectory` Verifies end-to-end consistency of a non-Hohmann Lambert rendezvous: applying the departure Δv , propagating over the transfer time, and applying the arrival Δv reproduces the target arrival state. Inputs: departure/arrival states at 400 km and 800 km altitude; transfer time $T_{\text{Hohmann}}/1.1$. Expected output: the propagated arrival state equals the target arrival state (default `numpy.allclose` tolerance).

B.2. Optimisation Objective Unit Tests

B.2.1. Epoch Ordering Penalty

`test_all_ordered_no_penalty` Confirms that strictly increasing epochs incur no penalty. Input: epochs (1, 2, 3, 4). Expected output: 0.0.

`test_equal_epochs_no_penalty` Confirms that equal consecutive epochs are not penalised, since ordering only requires non-strict monotonicity. Input: epochs (1, 1, 1, 1). Expected output: 0.0.

`test_single_inversion_at_first_pair` Checks that an inversion between the first pair of epochs incurs a penalty of 10^{10} . Input: epochs (3, 2, 4, 5). Expected output: 10^{10} .

`test_single_inversion_at_second_pair` Checks that an inversion between the second pair of epochs incurs a penalty of 10^{10} . Input: epochs (1, 3, 2, 4). Expected output: 10^{10} .

`test_single_inversion_at_third_pair` Checks that an inversion between the third pair of epochs incurs a penalty of 10^{10} . Input: epochs (1, 2, 4, 3). Expected output: 10^{10} .

`test_two_inversions` Confirms that two simultaneous inversions incur a penalty of 2×10^{10} . Input: epochs (3, 2, 4, 3). Expected output: 2×10^{10} .

`test_all_three_inversions` Confirms that a fully reversed epoch sequence incurs a penalty of 3×10^{10} . Input: epochs (4, 3, 2, 1). Expected output: 3×10^{10} .

`test_returns_numeric` Verifies that the penalty function returns a non-negative numeric value for both an ordered and a fully disordered epoch sequence. Inputs: epochs (1, 2, 3, 4) and (4, 3, 2, 1). Expected output: both results are numeric (`int` or `float`) and ≥ 0 .

B.2.2. Perigee Penalty

`test_none_states_returns_zero` Confirms that a `None` state list yields no penalty. Expected output: 0.0.

`test_empty_list_returns_zero` Confirms that an empty state list yields no penalty. Expected output: 0.0.

- test_all_states_above_earth** Confirms that a state with a positive-altitude perigee incurs no penalty. Input: circular state at 100 km altitude. Expected output: 0.0.
- test_one_state_below_earth** Checks that a single sub-surface state incurs a penalty of 10^{10} . Input: circular state at -100 km altitude. Expected output: 10^{10} .
- test_two_states_below_earth** Checks that two sub-surface states incur a penalty of 2×10^{10} . Inputs: circular states at -100 km and -200 km altitude. Expected output: 2×10^{10} .
- test_mixed_states** Confirms that only the sub-surface state among a mixed set contributes to the penalty (10^{10}). Inputs: circular states at 100 km, -100 km, and 400 km altitude. Expected output: 10^{10} .
- test_exactly_at_surface_is_penalised** Confirms that a perigee exactly at the Earth's surface is not penalised, since the comparison is strict. Input: circular state at 0 km altitude. Expected output: 0.0.

B.2.3. Delta-V Cost

- test_collision_sums_stride3** Confirms that, for the collision problem type, only every third element of the Δv vector is summed. Input: $\Delta v = [1, 1, 1, 1]$. Expected output: total = 3.0, penalty = 0.0.
- test_collision_large_dv_penalty** Checks that a large stride-summed Δv for the collision type incurs a quadratic penalty equal to the square of the total. Input: $\Delta v = [1000, 1000, 1000, 100]$. Expected output: total = 3000.0, penalty = 3000.0^2 .
- test_docking_sums_all** Confirms that, for the docking problem type, all elements of the Δv vector are summed. Input: $\Delta v = [1, 2, 3, 4]$. Expected output: total = 10.0, penalty = 0.0.
- test_inspection_sums_all** Confirms that, for the inspection problem type, all elements of the Δv vector are summed. Input: $\Delta v = [100, 200, 300, 400]$. Expected output: total = 1000.0.
- test_penalty_threshold_boundary_below** Confirms that a total Δv exactly at the 1000 m/s threshold does not trigger the quadratic penalty. Input: $\Delta v = [1000]$. Expected output: penalty = 0.0.
- test_penalty_threshold_boundary_above** Confirms that a total Δv marginally above the 1000 m/s threshold triggers a penalty equal to the square of the total. Input: $\Delta v = [1000.001]$. Expected output: penalty = total².
- test_zero_dv** Confirms that a zero Δv vector yields zero cost and zero penalty. Input: $\Delta v = \mathbf{0}_6$. Expected output: total = 0.0, penalty = 0.0.
- test_returns_tuple_of_two** Verifies that the function returns a two-element tuple. Input: $\Delta v = \mathbf{1}_6$. Expected output: a tuple of length 2.

B.2.4. Epoch Decoding

- test_returns_four_values** Confirms that `decode_epochs` returns four values. Input: decision-vector time fractions $[0.1, 0.2, 0.3, 0.4]$. Expected output: a tuple of length 4.
- test_epochs_are_non_decreasing_typical** Checks that decoded epochs are non-decreasing for a representative set of time fractions. Input: time fractions $[0.4, 0.5, 0.3, 0.8]$. Expected output: $t_1 \leq t_2 \leq t_3 \leq t_4$.
- test_epochs_non_decreasing_for_all_zeros** Confirms that all-zero time fractions collapse every epoch to the start epoch, which trivially satisfies non-decreasing order. Input: time fractions $[0, 0, 0, 0]$. Expected output: $t_1 \leq t_2 \leq t_3 \leq t_4$ (all equal to `start_epoch`).
- test_epochs_non_decreasing_for_all_ones** Confirms non-decreasing order when the first fraction consumes the entire window. Input: time fractions $[1, 1, 1, 1]$. Expected output: $t_1 \leq t_2 \leq t_3 \leq t_4$.
- test_epochs_non_decreasing_random_inputs** Verifies non-decreasing ordering across 200 randomly sampled time-fraction vectors. Input: time fractions drawn uniformly from $[0, 1]$, seed 42. Expected output: $t_1 \leq t_2 \leq t_3 \leq t_4$ for every one of the 200 samples.

- test_all_zeros_gives_start_epoch** Confirms that all-zero time fractions decode every epoch to the mission start epoch. Input: time fractions $[0, 0, 0, 0]$, `start_epoch = 0`, `duration = 3600` s. Expected output: $t_1 = t_2 = t_3 = t_4 = \text{start_epoch}$.
- test_first_frac_one_rest_zero** Confirms that a first fraction of 1 consumes the full window, so all four epochs equal `start_epoch + duration`. Input: time fractions $[1, 0, 0, 0]$. Expected output: $t_1 = t_2 = t_3 = t_4 = 3600.0$.
- test_equal_splits** Verifies the exact decoded epoch values for equal 0.5 splits at each stage of the shrinking-window encoding. Input: time fractions $[0.5, 0.5, 0.5, 0.5]$, `duration = 3600` s. Expected output: $t_1 = 1800.0$, $t_2 = 2700.0$, $t_3 = 3150.0$, $t_4 = 3375.0$.
- test_termination_never_exceeds_start_plus_duration** Confirms that the final decoded epoch never exceeds `start_epoch + duration`, across 200 random time-fraction vectors. Input: time fractions drawn uniformly from $[0, 1]$, seed 7. Expected output: $t_4 \leq \text{start_epoch} + \text{duration} + 10^{-9}$ for every sample.
- test_nonzero_start_epoch_offset** Confirms correct offsetting when the start epoch is non-zero. Input: `start_epoch = 1000` s, `duration = 2000` s, time fractions $[0, 0, 0, 0]$. Expected output: $t_1 = 1000.0$.
- test_nonzero_start_epoch_with_full_first_gap** Confirms correct offsetting when the start epoch is non-zero and the first fraction consumes the full window. Input: `start_epoch = 500` s, `duration = 1000` s, time fractions $[1, 0, 0, 0]$. Expected output: $t_1 = 1500.0$, $t_4 = 1500.0$.
- test_remaining_window_shrinks_monotonically** Confirms that successive epoch gaps remain non-negative and the total span does not exceed the mission duration. Input: time fractions $[0.3, 0.5, 0.2, 0.9]$. Expected output: each gap ≥ 0.0 ; total span $\leq \text{duration} + 10^{-9}$.

B.2.5. Decision-Vector Decoding

- test_midpoint_gives_zero_dv** Confirms that a raw decision value of 0.5 decodes to zero Δv in both manoeuvre components. Input: decision vector with Δv components at 0.5. Expected output: $\Delta v_1 = [0, 0, 0]$, $\Delta v_2 = [0, 0, 0]$.
- test_zero_raw_gives_negative_max_dv** Confirms that a raw decision value of 0 decodes to $-\text{max_dv}$ in each component. Input: decision vector with first Δv component at 0, `max_dv = 500` m/s. Expected output: $\Delta v_1 = [-500, -500, -500]$.
- test_one_raw_gives_positive_max_dv** Confirms that a raw decision value of 1 decodes to $+\text{max_dv}$ in each component. Input: decision vector with first Δv component at 1, `max_dv = 500` m/s. Expected output: $\Delta v_1 = [500, 500, 500]$.
- test_epochs_scaled_via_decode_epochs** Confirms that the decoded epochs from `_decode_inputs` are non-decreasing. Input: time fractions $[0.1, 0.3, 0.5, 0.9]$. Expected output: $t_1 \leq t_2 \leq t_3 \leq t_4$.
- test_all_zero_fracs_give_start_epoch** Confirms that zero time fractions decode all epochs to the start epoch. Input: time fractions $[0, 0, 0, 0]$. Expected output: $t_1 = t_2 = t_3 = t_4 = \text{start_epoch}$.
- test_returns_six_values** Confirms that `_decode_inputs` returns six values (two Δv vectors and four epochs). Expected output: a tuple of length 6.
- test_dv_components_independent** Confirms that the two Δv vectors decode independently of one another. Input: first Δv raw $[1, 0.5, 0]$, second Δv raw $[0, 0.5, 1]$, `max_dv = 500` m/s. Expected output: $\Delta v_1 = [500, 0, -500]$, $\Delta v_2 = [-500, 0, 500]$.

B.2.6. End-Condition Loading

- test_collision_returns_zeros** Confirms that the collision problem type returns zero relative position and velocity without file access. Expected output: `pos = [0, 0, 0]`, `vel = [0, 0, 0]`.
- test_docking_returns_zeros** Confirms that the docking problem type returns zero relative position and velocity without file access. Expected output: `pos = [0, 0, 0]`, `vel = [0, 0, 0]`.

- test_inspection_loads_file** Confirms that the inspection problem type loads relative position and velocity from a (mocked) JSON file. Input: seed 42, domain train. Expected output: pos = [1, 2, 3], vel = [0.1, 0.2, 0.3].
- test_eavesdropping_loads_file** Confirms that the eavesdropping problem type loads relative position and velocity from a (mocked) JSON file. Input: seed 7, domain test. Expected output: pos = [4, 5, 6], vel = [0.4, 0.5, 0.6].
- test_invalid_problem_type_raises** Confirms that an unrecognised problem type raises a `ValueError`. Input: problem type "AlienInvasion". Expected output: `ValueError` raised.
- test_inspection_uses_correct_filepath** Confirms that the inspection problem type constructs the expected file path from seed and domain. Input: seed 3, domain val. Expected output: open called once with path data_mission/observation/seed_3_domain_val.json.
- test_eavesdropping_uses_correct_filepath** Confirms that the eavesdropping problem type constructs the expected file path from seed and domain. Input: seed 5, domain test. Expected output: open called once with path data_mission/eavesdropping/seed_5_domain_test.json.

B.2.7. Simulation Execution

- test_returns_result_on_success** Confirms that a successful (mocked) simulation call returns its result unchanged. Input: zero Δv vectors, epochs (0, 1, 2, 3). Expected output: the (mocked) simulation result, returned unmodified.
- test_returns_none_on_exception** Confirms that an exception raised by the (mocked) simulation backend is caught and `None` is returned. Input: zero Δv vectors, epochs (0, 1, 2, 3). Expected output: `None`.
- test_end_conditions_forwarded_for_collision** Confirms that zero relative position and velocity are correctly forwarded to the simulation backend for the collision problem type. Input: zero Δv vectors, epochs (0, 1, 2, 3). Expected output: forwarded rel_pos = [0, 0, 0], rel_vel = [0, 0, 0].
- test_epochs_unpacked_correctly** Confirms that the four manoeuvre/termination epochs are unpacked and forwarded correctly to the simulation backend. Input: epochs (10, 20, 30, 40). Expected output: forwarded maneuver_epoch1 = 10, maneuver_epoch2 = 20, maneuver_epoch3 = 30, termination_epoch = 40.

B.2.8. Keplerian-End Objective Function

- test_returns_1e10_on_simulation_failure** Confirms that a simulation exception causes the objective to return the penalty value 10^{10} . Expected output: 10^{10} .
- test_feasible_solution_below_1e10** Confirms that a feasible docking solution with Δv components [10, 10, 10] (total 30) returns an objective value of 30, with no penalties applied. Expected output: 30.0.
- test_epoch_penalty_not_triggered_by_valid_fracs** Confirms that the epoch-ordering penalty is never triggered for valid decision-vector time fractions, since `decode_epochs` guarantees ordering. Input: 20 randomly sampled time-fraction vectors, seed 0. Expected output: result $< 10^{10}$ for every sample.
- test_perigee_penalty_added_when_state_below_surface** Confirms that a sub-surface trajectory state adds the perigee penalty (10^{10}) to the objective. Input: circular state at -100 km altitude. Expected output: 10^{10} .
- test_perigee_penalty_absent_for_valid_orbit** Confirms that a valid-altitude trajectory state does not add a perigee penalty. Input: circular state at 400 km altitude. Expected output: 0.0.
- test_collision_adds_tf_contribution** Confirms that, for the collision problem type, the final time contributes to the objective as $t_f/10^3$. Input: $t_f = 3000$ s. Expected output: 3.0.
- test_non_collision_no_tf_contribution** Confirms that the final-time term is absent from the objective for the inspection problem type. Input: $t_f = 3000$ s. Expected output: 0.0.

- `test_eavesdropping_no_tf_contribution` Confirms that the final-time term is absent from the objective for the eavesdropping problem type. Input: $t_f = 5000$ s. Expected output: 0.0.
- `test_large_dv_triggers_quadratic_penalty` Confirms that a total Δv of 1500 m/s (above the 1000 m/s threshold) adds a quadratic penalty. Expected output: $1500 + 1500^2$.
- `test_dv_just_at_threshold_no_penalty` Confirms that a total Δv exactly at the 1000 m/s threshold does not trigger the quadratic penalty. Expected output: 1000.0.
- `test_collision_dv_uses_only_first_three_components` Confirms that, for the collision problem type, only the first three Δv components contribute to the total. Input: $\Delta v = [10, 10, 10, 500, 500]$. Expected output: 30.0.
- `test_returns_scalar_float` Confirms that the objective function returns a finite scalar float (not NaN). Expected output: a finite float value.

B.3. Mission Optimisation Unit Tests

B.3.1. Design-Space Bounds

- `test_shape_and_symmetry` Confirms that, for every domain/problem-type combination, the lower and upper bound vectors have shape (6,), are symmetric (lower = -upper), and satisfy lower $\leq$ upper. Expected output: all conditions hold for every combination.
- `test_sma_bound_comes_from_the_table` Confirms that the semi-major-axis bound returned by `_default_bounds` matches the corresponding entry in `_SMA_BOUND_BY_TYPE` for every domain/problem-type combination. Expected output: `upper[0]` equals the table entry in every case.
- `test_ecc_and_angle_bounds_are_shared` Confirms that the eccentricity and angle bound components are the shared module-level constants `_ECC_BOUND` and `_ANGLE_BOUND`. Input: problem type observation, domain GEO. Expected output: `upper[1] = _ECC_BOUND`, `upper[2:] = _ANGLE_BOUND`.
- `test_eavesdropping_geo_is_tighter_than_observation` Confirms that the GEO semi-major-axis bound for the eavesdropping problem type is smaller than for observation, reflecting the tighter required formation. Expected output: $SMA_{GEO,eavesdropping} < SMA_{GEO,observation}$.
- `test_unknown_domain_raises` Confirms that an unrecognised domain raises a `KeyError`. Input: domain MEO. Expected output: `KeyError` raised.
- `test_unknown_problem_type_raises` Confirms that an unrecognised problem type raises a `KeyError`. Input: problem type rendezvous. Expected output: `KeyError` raised.

B.3.2. Mission Configuration

- `test_observation_config_defaults` Confirms the default field values of `observation_config`. Expected output: `problem_type = "observation"`, `propagation = "analytical"`, `n_output_steps = 1000`, `duration  $\approx 10 \times JULIAN\_DAY$` , bound arrays of shape (6,).
- `test_eavesdropping_config_defaults` Confirms that `eavesdropping_config` defaults to problem type eavesdropping with analytical propagation. Expected output: `problem_type = "eavesdropping"`, `propagation = "analytical"`.
- `test_overrides_are_applied` Confirms that explicit keyword overrides (seed, generations, propagation backend) are applied while the problem type remains fixed. Input: seed = 999, generations = 5, propagation = numerical. Expected output: seed = 999, generations = 5, propagation = "numerical", `problem_type = "observation"` (unchanged).
- `test_bounds_resolve_from_problem_type_and_domain` Confirms that `MissionConfig.__post_init__` selects the correct bound row from `_SMA_BOUND_BY_TYPE` for every domain/problem-type combination. Expected output: `upper_bounds[0]` matches the table entry and `lower_bounds = -upper_bounds` in every case.
- `test_explicit_bounds_are_not_overwritten` Confirms that explicitly supplied lower/upper bound arrays are preserved rather than recomputed. Input: lower = -1_6 , upper = 1_6 . Expected output:

`lower_bounds = -16, upper_bounds = 16` (unchanged).

`test_early_stopping_defaults` Confirms the default early-stopping configuration. Expected output: `early_stopping = True, plateau_window = 10, improvement_threshold = 10-3`.

`test_removed_api_is_gone` Regression check confirming that the deprecated `rendezvous_config` function and `proximity_termination_distance` attribute no longer exist. Expected output: `hasattr(mo, "rendezvous_config")` is False; `hasattr(MissionConfig(), "proximity_termination_distance")` is False.

B.3.3. Dependent-Variable Schema

`test_columns_fit_inside_the_assembled_array` Confirms that every column index or slice in `_dep_var_columns` lies within the expected assembled-array width. Expected output: highest column index < 12 for observation, < 2 for eavesdropping.

`test_observation_column_positions` Confirms the exact column layout for observation dependent variables. Expected output: `rel_pos = slice(1,4)`, `distance = 4`, `sun_wrt_target = slice(5,8)`, `sun_wrt_inspector = slice(8,11)`, `eclipse = 11`.

`test_eavesdropping_only_saves_distance` Confirms that the eavesdropping dependent-variable schema contains only a distance column. Expected output: `{"distance": 1}`.

`test_unknown_problem_type_raises` Confirms that requesting dependent variables for an unrecognised problem type raises a `ValueError`. Input: problem type `rendezvous`. Expected output: `ValueError` raised.

`test_schema_keys_match_supported_problem_types` Confirms that `_DEP_VAR_SCHEMA` contains exactly the keys observation and eavesdropping. Expected output: `{"observation", "eavesdropping"}`.

B.3.4. Numerical Helpers

`test_shape_and_endpoints` Confirms that `interpolate_variables` regrids to the requested number of steps and preserves the first and last time values. Input: 11-sample array over $t \in [0, 100]$, $n_{\text{steps}} = 250$. Expected output: output shape (250, 4); `out[0, 0] = 0.0`; `out[-1, 0] = 100.0`.

`test_linear_data_is_reproduced_exactly` Confirms that cubic-Hermite interpolation exactly reproduces linear input data on the new grid. Input: 11-sample linear data over $t \in [0, 10]$, $n_{\text{steps}} = 21$. Expected output: `out[:, 1] = 3 out[:, 0]` and `out[:, 2] = -2 out[:, 0] + 5`, to within 10^{-9} .

`test_radial_offset_lands_on_index_zero` Confirms that a purely radial inspector offset appears entirely in the first (radial) LVLH component. Input: target on the $+X$ axis; radial offset of 10 m. Expected output: `rel_pos = [10, 0, 0]`, to within 10^{-9} .

`test_in_track_and_cross_track_axes` Confirms that in-track and cross-track offsets map to the second and third LVLH components respectively. Input: offset [0, 20, 30] m. Expected output: `rel_pos = [0, 20, 30]`, to within 10^{-9} .

`test_relative_velocity_projection` Confirms that a relative velocity offset projects correctly onto the LVLH radial axis. Input: velocity offset [10, 0, 0] m/s. Expected output: `rel_vel = [10, 0, 0]`, to within 10^{-9} .

`test_pos_only_matches_compute_lvlh` Confirms that `EavesdroppingProblem._compute_lvlh_pos_only` agrees with the position output of the general `compute_lvlh` function. Input: offset [1, 2, 3] m. Expected output: outputs agree, to within 10^{-9} .

`test_vel_only_matches_compute_lvlh` Confirms that `EavesdroppingProblem._compute_lvlh_vel_only` agrees with the velocity output of the general `compute_lvlh` function. Input: velocity offset [1, 2, 3] m/s. Expected output: outputs agree, to within 10^{-9} .

`test_zero_vector_does_not_produce_nan` Confirms that normalising a zero vector returns a zero vector rather than NaN, while a non-zero vector normalises correctly. Input: [3, 4, 0] and [0, 0, 0]. Expected output: `result[0] = [0.6, 0.8, 0.0]`, `result[1] = [0, 0, 0]`, no NaN values.

- `test_both_problem_classes_share_the_behaviour` Confirms that `ObservationProblem._safe_normalise` and `EavesdroppingProblem._safe_normalise` give identical results. Input: zero vector. Expected output: identical outputs from both classes.
- `test_zero_srp_area_passes` Confirms that the two-body force-model guard passes silently when the solar-radiation-pressure reference area is zero. Expected output: `None` (no exception raised).
- `test_nonzero_srp_area_raises` Confirms that a non-zero solar-radiation-pressure reference area causes the guard to raise a `RuntimeError`. Input: `_SRP_REFERENCE_AREA = 1.0`. Expected output: `RuntimeError` raised.
- `test_second_identical_call_is_served_from_cache` Confirms that repeating a Sun-position query on the same time grid does not increase the underlying SPICE call count relative to the first call. Input: 5-point time grid over `[0, 1000]` s. Expected output: `first.shape = (5, 3)`; `first = second`; `mock_spice.call_count = 5`.
- `test_different_grid_is_a_cache_miss` Confirms that querying two different time grids results in separate SPICE calls for each. Input: 3-point grids over `[0, 1000]` s and `[0, 2000]` s. Expected output: `mock_spice.call_count = 6`.
- `test_blas_threads_are_pinned_and_kernels_loaded` Confirms that `_worker_init` pins all relevant BLAS thread-count environment variables and triggers SPICE kernel loading. Expected output: each of the five thread-count environment variables set to "1"; kernel-loading mock called exactly once.

B.3.5. Analytical Propagation Backend

- `test_time_grid_spans_the_mission_window` Confirms that the internal time grid has the configured number of steps and spans exactly from the mission initial to final time. Input: eavesdropping problem, 40 output steps, 3600 s duration. Expected output: `len(problem._times) = 40`; first entry equals `mission_initial_time`; last entry equals `mission_final_time`.
- `test_state_history_is_13_columns` Confirms that the propagated state history has shape $(N_{\text{steps}}, 13)$ and that the first row matches the target/inspector initial states. Input: eavesdropping problem with inspector initial state equal to the target. Expected output: shape `(40, 13)`; column 0 equals `problem._times`; row 0, columns 1–7 and 7–13, both equal the target state.
- `test_reached_final_time_is_true_by_construction` Confirms that the analytical backend always reports having reached the final mission time. Expected output: `True`.
- `test_eavesdropping_dep_vars_are_time_and_distance` Confirms that eavesdropping dependent variables consist of time and a constant separation distance. Input: inspector offset by `[0, 5000, 0]` m in-track. Expected output: shape `(40, 2)`; column 0 equals `problem._times`; distance column equal to 5000.0 for every row.
- `test_observation_dep_vars_column_layout` Confirms the full observation dependent-variable column layout against a known 3–4 km offset. Input: inspector offset `[0, 3000, 4000, 0, 0, 0]` m. Expected output: shape `(40, 12)`; `rel_pos[0] = [0, 3000, 4000]`; `distance = 5000.0` for every row; `sun_wrt_target[0]` and `sun_wrt_inspector[0]` equal to the Sun position minus the target/inspector position at row 0; `eclipse = 1.0` for every row.
- `test_assemble_rejects_unknown_problem_type` Confirms that assembling dependent variables for an unrecognised problem type raises a `ValueError`. Expected output: `ValueError` raised.
- `test_propagate_records_last_run` Confirms that `_propagate` stores its dependent variables, state history, and offsets for later retrieval. Input: offsets `[1, 0, 0, 0, 0, 0]`. Expected output: `last_dep_vars()` equals the returned dependent variables; `last_state_history()` equals the returned state history; `problem._last_offsets` equals the input offsets.
- `test_accessors_raise_before_any_propagation` Confirms that calling `last_dep_vars` or `last_state_history` before any propagation raises a `RuntimeError`. Expected output: `RuntimeError` raised by both accessors.

- `test_dynamics_simulator_unavailable_on_analytical_path` Confirms that `get_last_run_dynamics_simulator` raises `NotImplementedError` on the analytical backend. Expected output: `NotImplementedError` raised.
- `test_last_state_history_refines_on_request` Confirms that requesting a finer grid via `last_state_history(n_steps)` returns a re-evaluated 7-row state history. Expected output: shape (7, 13); final row's time equals `mission_final_time`.

B.3.6. Numerical Propagation Backend

- `test_initial_state_is_12_elements` Confirms that the numerical backend assembles a 12-element combined initial-state vector from the target and inspector states. Expected output: `len(settings.initial_states) = 12`, equal to `arange(12)`.
- `test_output_is_regridded_to_n_output_steps` Confirms that raw integrator output is regridded to the configured number of output steps for both dependent variables and state history. Input: 50 output steps. Expected output: `dep_vars.shape = (50, 2)`; `state_history.shape = (50, 13)`.
- `test_reached_final_time_true_when_run_completes` Confirms that reaching the configured mission duration is reported correctly. Input: run duration equal to the configured 86 400 s. Expected output: `True`.
- `test_reached_final_time_false_on_early_termination` Confirms that a run stopping before the configured duration is reported as not having reached the final time. Input: run duration 43 200 s against a configured 86 400 s. Expected output: `False`.
- `test_no_seed_cache_is_built` Confirms that the numerical backend does not construct an analytical seed cache. Expected output: `problem._analytical = False`; `no_seed_cache` attribute present.

B.3.7. Base Problem Interface

- `test_get_bounds_returns_two_lists_of_six` Confirms that `get_bounds` returns lower/upper lists of length 6 with $\text{lower}_i \leq \text{upper}_i$ for all i . Expected output: `len(lower) = 6`, `len(upper) = 6`, $\text{lower}_i \leq \text{upper}_i$ for all i .
- `test_base_class_does_not_implement_the_udp` Confirms that the base problem class raises `NotImplementedError` for `get_nobj` and `fitness`. Expected output: `NotImplementedError` raised by both calls.

B.3.8. Observation Problem

- `test_nobj_is_two` Confirms that the observation problem has two objectives. Expected output: 2.
- `test_phase_angle_window` Confirms the phase-angle acceptance window. Expected output: `theta_min = 1`; $115 \leq \text{theta_max} \leq 155$.
- `test_objective1_selection_is_deterministic_per_seed` Confirms that the first-objective function selected for a given seed is reproducible across separate instantiations. Input: seed 7. Expected output: identical selected objective-function name across both instantiations.
- `test_objective1_selection_is_domain_weighted` Confirms that the flyby objective is selected far more often for GEO than for LEO. Input: 200 seeds per domain. Expected output: GEO flyby count > 150 ; LEO flyby count < 50 (out of 200).
- `test_fitness_early_termination_returns_large` Confirms that a propagation which does not reach the final time returns the large penalty for both objectives. Expected output: $[10^9, 10^9]$.
- `test_fitness_returns_get_nobj_values` Confirms that `fitness` returns a finite value for each declared objective under nominal dependent-variable data. Expected output: result length equal to `get_nobj() = 2`; all values finite.
- `test_objective_distance_is_the_mean_of_the_distance_column` Confirms that the distance objective equals the mean of the distance dependent-variable column. Input: distances $[1000, 2000, 3000]$ m. Expected output: 2000.0.

- `test_objective_flyby_is_zero_in_the_safe_corridor` Confirms that the flyby objective is zero when all sampled distances lie within the safe corridor. Input: distances [1000, 20 000, 45 000] m. Expected output: 0.0.
- `test_objective_flyby_penalises_close_approach` Confirms the quadratic penalty applied for distances below the safe minimum. Input: distances [100, 1000] m. Expected output: $(500 - 100)^2 \times 10^6$.
- `test_objective_flyby_penalises_flying_away` Confirms the quadratic penalty applied for distances above the safe maximum. Input: distances [1000, 60 000] m. Expected output: $(60\,000 - 50\,000)^2$.
- `test_coverage_penalty_when_sun_is_behind_the_target` Confirms that a 0° phase angle (below `theta_min`) incurs the fixed coverage penalty. Expected output: 10^6 .
- `test_coverage_penalty_when_fully_eclipsed` Confirms that a zero shadow (eclipse) function value invalidates an otherwise favourable geometry. Expected output: 10^6 .
- `test_coverage_returns_negative_bin_fraction` Confirms the exact negative coverage score for two opposite look directions at 90° phase angle, each occupying a distinct bin. Expected output: $-2/(48 \times 24)$.
- `test_coverage_counts_unique_bins_not_samples` Confirms that repeated samples of the same look direction do not improve the coverage score beyond that of a single occupied bin. Input: 10 identical samples. Expected output: $-1/(48 \times 24)$.

B.3.9. Eavesdropping Problem

- `test_nobj_is_two` Confirms that the eavesdropping problem has two objectives. Expected output: 2.
- `test_parameter_draw_is_in_range` Confirms that the drawn half-angle and nominal standoff distance fall within their domain-specific bounds. Input: 20 seeds, domains GEO and LEO. Expected output: `half_angle` and `d_nominal` within their respective domain bounds; `sep_min` equal to the domain table value, for every seed.
- `test_parameter_draw_is_deterministic` Confirms that the drawn half-angle and nominal distance are reproducible for a fixed seed. Input: seed 42. Expected output: identical `half_angle` and `d_nominal` across two instantiations.
- `test_half_angle_is_drawn_before_nominal` Regression test confirming that the half-angle is the first value drawn from the seeded random generator. Input: seed 42, domain GEO. Expected output: `half_angle` equal to `rng.uniform(*HALF_ANGLE_BY_DOMAIN["GEO"])` as the first draw.
- `test_invalid_domain_raises` Confirms that an unrecognised domain raises a `KeyError`. Input: domain MEO. Expected output: `KeyError` raised.
- `test_fitness_early_termination_returns_large` Confirms that a propagation which does not reach the final time returns the large penalty for both objectives. Expected output: $[10^9, 10^9]$.
- `test_sma_from_circular_state_is_the_radius` Confirms that the semi-major axis recovered from a circular-orbit state equals its radius. Input: GEO-radius circular state. Expected output: ratio ≈ 1.0 , to 9 decimal places.
- `test_stability_is_zero_at_the_assigned_standoff` Confirms that the stability objective is zero when the inspector sits exactly at the assigned standoff with no drift or crossover. Expected output: 0.0, to 6 decimal places.
- `test_stability_penalises_standoff_error` Confirms the tracking-error penalty when the actual standoff differs from the nominal value by a known fraction. Expected output: $0.5 \times W_{\text{TRACK}}$.
- `test_stability_penalises_floor_breach` Confirms that a separation below the configured minimum floor incurs a penalty greater than half of W_{FLOOR} . Input: 200 m separation, `sep_min` = 5000 m. Expected output: `score` $> 0.5 \times W_{\text{FLOOR}}$.

- `test_stability_penalises_crossover` Confirms that a trajectory crossing from one side of the target to the other over the horizon saturates the crossover penalty. Expected output: $W_{\text{CROSSOVER}}$, to 5 decimal places.
- `test_stability_penalises_secular_drift` Confirms that a semi-major-axis mismatch (500 m) increases the stability penalty relative to an on-station trajectory. Expected output: drifting-trajectory score strictly greater than on-station score.
- `test_eavesdrop_is_zero_on_the_beam_axis` Confirms that a purely radial inspector offset lying exactly on the uplink beam axis gives zero eavesdrop penalty. Expected output: 0.0, to 9 decimal places.
- `test_eavesdrop_saturates_far_outside_the_cone` Confirms that a 5° transverse offset — well outside a 0.5° half-angle cone — produces a large penalty. Expected output: score > 1.0 and $\leq 1 + W_{\text{CENTRE}}$.
- `test_eavesdrop_prefers_a_tighter_offset` Confirms that the eavesdrop penalty increases monotonically with off-axis angle. Input: 1° and 20° offsets against a 0.2° half-angle. Expected output: near-offset score $<$ far-offset score.
- `test_eavesdrop_transmitter_sits_on_the_earth_surface` Confirms that the beam axis is anchored to a ground station at Earth's radius. Expected output: zero penalty for zero separation; $R_{\oplus} < a_{\text{GEO}}$.

B.3.10. Problem Factory

- `test_analytical_path_builds_no_propagator_settings` Confirms that building an eavesdropping problem with analytical propagation yields no propagator settings. Expected output: instance of `EavesdroppingProblem`; `propagator_settings_function() = None`; `problem._analytical = True`.
- `test_analytical_path_still_registers_eclipse_settings` Confirms that eclipse settings are still registered when building an observation problem via the analytical path. Expected output: `eclipse_settings` called exactly once.
- `test_numerical_path_delegates_to_the_settings_builder` Confirms that the numerical propagation path delegates construction of propagator settings. Expected output: `_build_numerical_propagator_settings` called exactly once; `problem.propagator_settings_function()` is the mocked sentinel.
- `test_dispatches_on_problem_type` Confirms that `build_problem` returns an `ObservationProblem` instance for the observation problem type. Expected output: instance of `ObservationProblem`.
- `test_unknown_problem_type_raises` Confirms that an unrecognised problem type raises a `ValueError` during problem construction. Expected output: `ValueError` raised.

B.3.11. Optimisation Runner

- `test_penalty_rows_are_excluded` Confirms that fitness rows at the penalty threshold do not alter the computed Pareto hypervolume relative to a clean front. Expected output: `hypervolume(clean) = hypervolume(polluted)`.
- `test_all_penalised_front_scores_zero` Confirms that a front consisting entirely of penalised points has zero hypervolume. Expected output: 0.0.
- `test_points_beyond_the_reference_do_not_crash` Confirms that fitness points lying beyond the reference point do not cause the hypervolume computation to fail. Expected output: a non-negative hypervolume value.
- `test_full_run_without_early_stopping` Confirms that, with early stopping disabled, the optimiser evolves for the full configured number of generations. Input: 5 generations, population size 10. Expected output: algorithm constructed once; population constructed once with size = 10, seed = `config.seed`; `evolve.call_count = 5`; `len(fitness_list) = len(population_list) = 5`.
- `test_early_stopping_on_a_hypervolume_plateau` Confirms that an unchanging Pareto front triggers early stopping after the configured plateau window. Input: 100 generations, plateau window 3.

Expected output: $\text{len}(\text{fitness_list}) = \text{plateau_window} + 1 = 4$, strictly less than the configured 100 generations.

`test_all_penalised_generations_do_not_crash` Confirms that a run in which every individual is penalised still completes. Expected output: $\text{len}(\text{fitness_list}) \geq 1$.

`test_picks_the_point_closest_to_the_normalised_ideal` Confirms that `select_best_solution` picks the decision vector whose normalised objective distance to the ideal point is smallest. Expected output: `best = [10.0, 10.0, 0.0, 0.0, 0.0, 0.0]`; `pareto_optimums` called once with the second-generation fitness array.

`test_flat_front_does_not_divide_by_zero` Confirms that a degenerate (flat, zero-range) Pareto front does not cause a division-by-zero error. Expected output: $\text{len}(\text{best}) = 6$, returned without error.

B.3.12. Result Output

`test_path_layout` Confirms the constructed result file path for a given seed and domain. Input: seed 112987, domain GEO. Expected output: `data_mission/observation/seed_112987_domain_GEO.json`.

`test_problem_type_selects_the_folder` Confirms that the eavesdropping problem type resolves to the corresponding output folder. Input: seed 1, domain LEO. Expected output: `data_mission/eavesdropping/seed_1_domain_LEO.json`.

`test_make_dir_flag` Confirms directory-creation behaviour under the `make_dir` flag. Expected output: `makedirs` not called when `make_dir = False`; called once with `("data_mission/observation", exist_ok = True)` when `make_dir = True`.

`test_payload_and_path` Confirms the full saved-results payload. Expected output: returned path equal to `data_mission/observation/seed_112987_domain_GEO.json`; `fitness` called once with the best solution; `last_state_history` called once with `n_steps = TCA_SEARCH_STEPS`; `open` called once for writing; `makedirs` called once; payload keys = `{"rel_pos", "rel_vel", "tca_time", "tca_distance"}`; seed = 112987, domain = "GEO"; $\text{len}(\text{rel_pos}) = \text{len}(\text{rel_vel}) = 3$; `tca_time = 200.0`; `tca_distance ≈ 5000.0`.

`test_payload_is_json_serialisable` Confirms that the saved-results payload contains no stray NumPy scalar types. Expected output: `json.dumps(payload)` succeeds without raising `TypeError`.

`test_skip_false_returns_everything` Confirms that, with seed-skipping disabled, all input seeds are returned unfiltered. Input: seeds 0, 1, 2. Expected output: `[0, 1, 2]`.

`test_existing_files_are_dropped` Confirms that seeds whose result file already exists are excluded from the pending list when skipping is enabled. Input: seeds 0, 1, 2, with seed 1's file present. Expected output: `[0, 2]`.

B.3.13. Live-Environment Setup (requires SPICE kernels)

`test_create_simulation_bodies` Confirms that the simulation body set includes Earth, Sun, Target, and Inspector. Expected output: `bodies.does_body_exist(name)` is `True` for each of the four names.

`test_get_acceleration_models` Confirms that acceleration models are built for both Target and Inspector bodies. Expected output: models dictionary contains "Target" and "Inspector" keys, each including an "Earth" entry.

`test_termination_is_time_based_only` Confirms that termination settings are purely time-based. Input: $t_0 = 0$, $t_f = 86\,400\text{ s}$. Expected output: instance of `PropagationTerminationSettings`; passing an additional proximity-distance argument raises `TypeError`.

`test_target_initial_state_is_finite` Confirms that the target's initial state is finite and consistent with the specified semi-major axis. Expected output: shape `(6,)`; no NaN values; $\|\mathbf{r}\|/a \approx 1.0$, to 3 decimal places.

`test_inspector_offsets_shift_the_state` Confirms that a non-zero LVLH offset produces an inspector state distinct from the target state. Input: offset `[10\,000, 0, 0, 0, 0, 0]` m. Expected output:

inspector state of shape (6,), not equal to the target state.

`test_zero_offsets_reproduce_the_target_state` Confirms that a zero offset reproduces the target's initial state exactly. Expected output: inspector state equal to the target state, to $\text{rtol} = 10^{-12}$, $\text{atol} = 10^{-6}$.

`test_angles_are_treated_as_radians` Regression test confirming that orbital angles are interpreted as radians (not degrees). Input: 45° inclination. Expected output: out-of-plane displacement fraction > 0.1 .

`test_create_propagator_settings` Confirms that propagator settings are constructed correctly. Input: zero 12-element initial state. Expected output: instance of `PropagatorSettings`; `settings.initial_states = 012`.

B.3.14. Live-Environment Dependent Variables (requires SPICE kernels)

`test_observation_saves_five_variables` Confirms that five dependent variables are configured for the observation problem type. Expected output: 5.

`test_eavesdropping_saves_only_relative_distance` Confirms that only one dependent variable is configured for the eavesdropping problem type. Expected output: 1.

B.3.15. Target Keplerian Elements

`test_geo_applies_the_lan_filter` Confirms that, for the GEO domain, the underlying conditions generator is called with a longitude-of-ascending-node filter. Input: seed 1234. Expected output: `result["semi_major_axis"] = 42 164 000.0`; generator called once with `seed = 1234`, `domain = "GEO"`, `dv_max = 20.0`, `filters = {"LAN" : 90.0}`.

`test_non_geo_passes_no_filter` Confirms that, for a non-GEO domain, the underlying conditions generator is called without any filter argument. Input: seed 5678, domain LEO. Expected output: generator called once with `seed = 5678`, `domain = "LEO"`, `dv_max = 20.0` (no `filters` keyword).



Thesis Planning

This chapter contains the thesis planning. Figure C.1 shows the Gantt chart from the literature review and thesis plan. The two phases following the literature phase accounted for the majority of the thesis work. The Midterm phase and Green Light phase both underwent changes compared to the original planning, presented in Figure C.2 and Figure C.3 respectively.

For the Midterm phase, the creation of the RPO model took longer than planned, mostly due to a fundamental shift away from the literature in how the data was generated. As a result, the data processing work was moved accordingly. However, the creation of the machine learning model for the generated data did stay within the initial timeframe, mostly due to parallel work with the data generation model and the use of less complex models than those found in the literature.

For the Green Light phase, a major departure from the original plan was made. The main driver for this was the better-than-expected availability of data on both cooperative and non-cooperative RPOs, which allowed for an assessment using real-world data. Furthermore, the initial plan was to further develop the model along the lines of the literature, by adding convolutional layers, self-attention, and other improvements. This was not undertaken; instead, a further investigation into the causes of misclassification was carried out. Lastly, an additional literature review was conducted due to the topicality of the subject, as relevant (non-)cooperative RPOs were being conducted during the thesis period itself, alongside ongoing research.

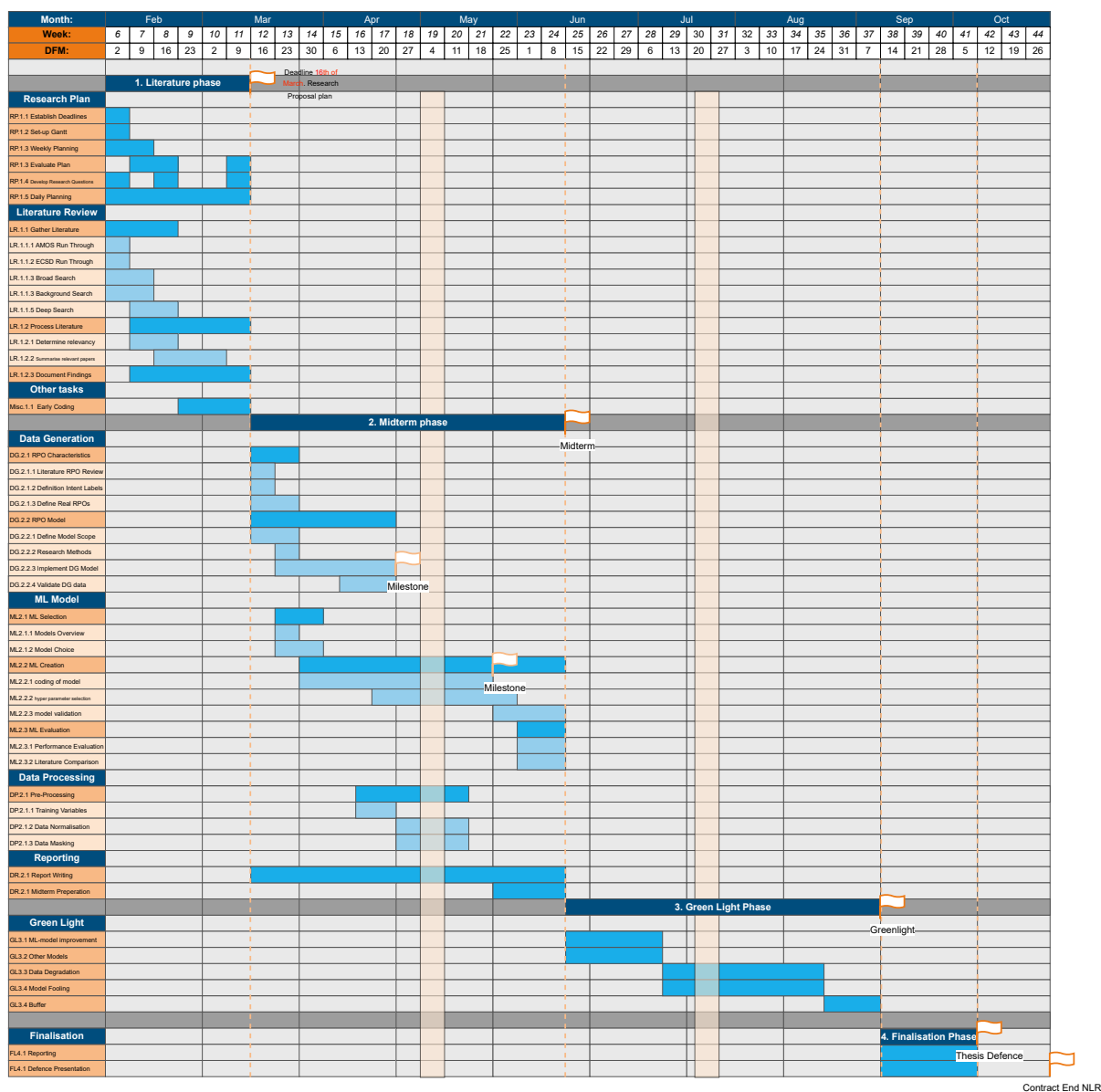


Figure C.1: Gantt chart from the literature review and planning

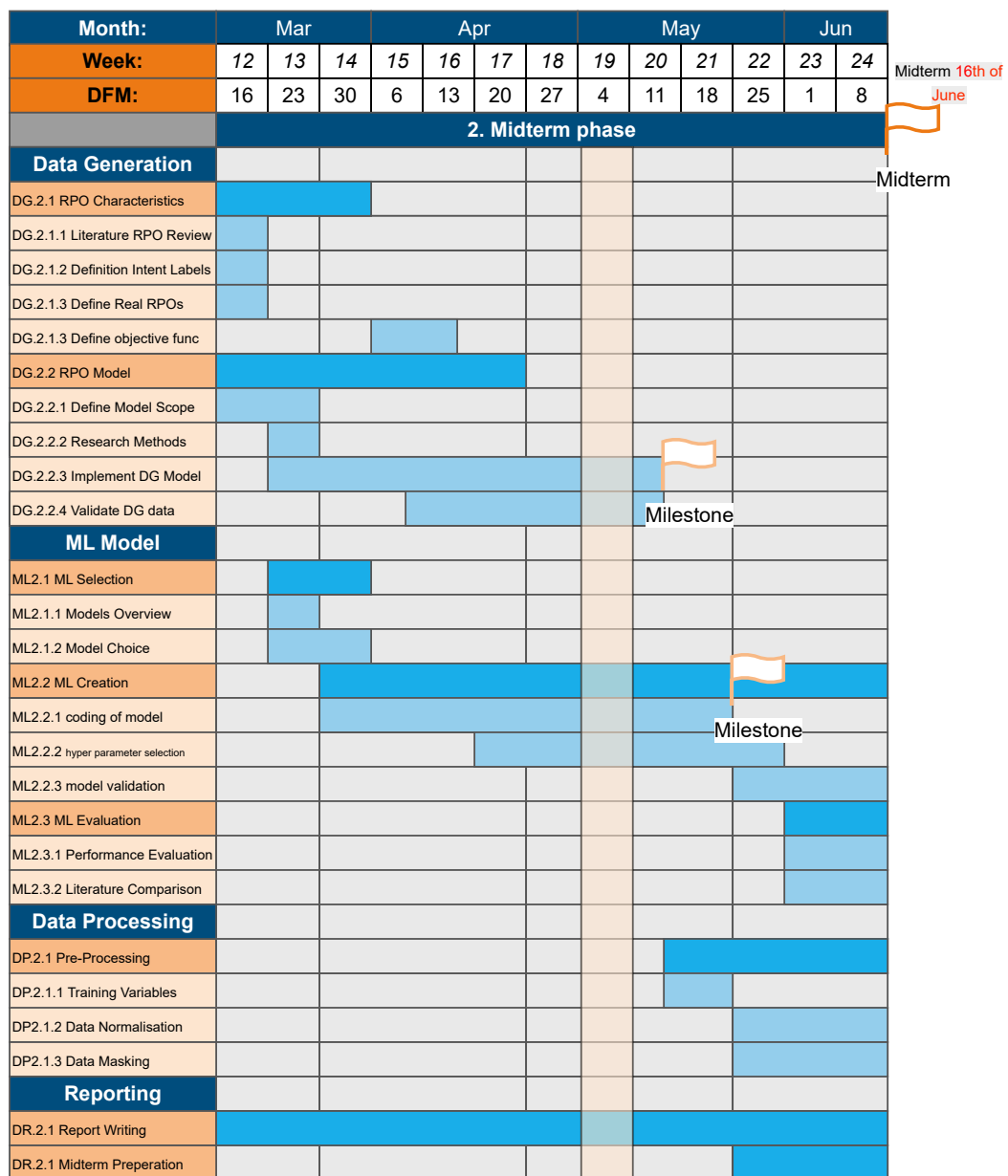


Figure C.2: Midterm phase Gantt chart after iterations

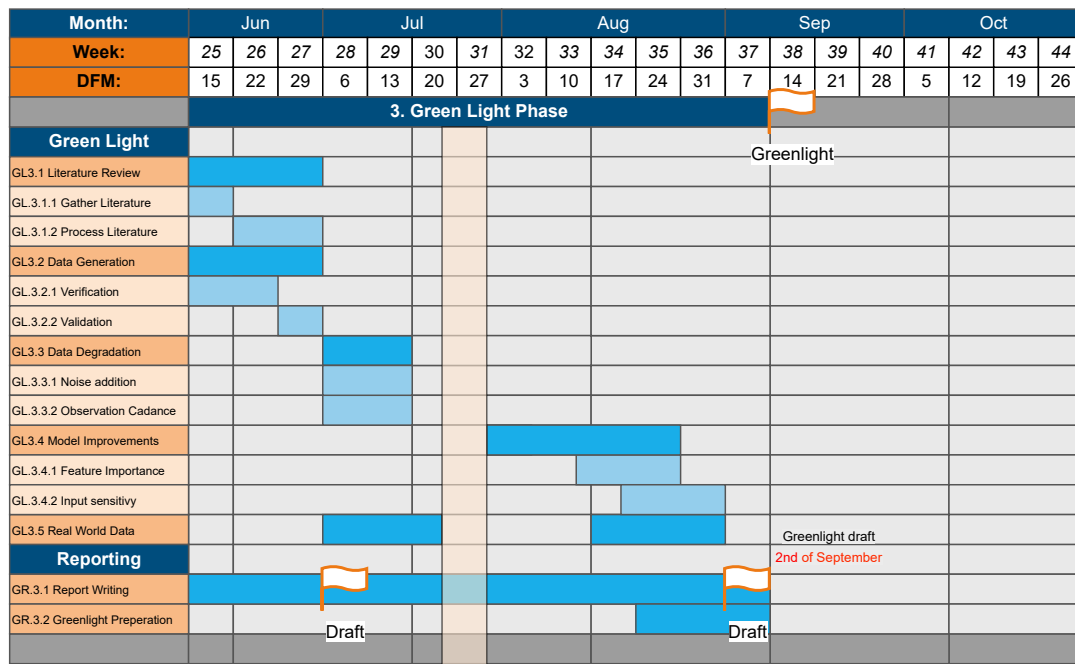


Figure C.3: Green Light phase Gantt chart after iterations