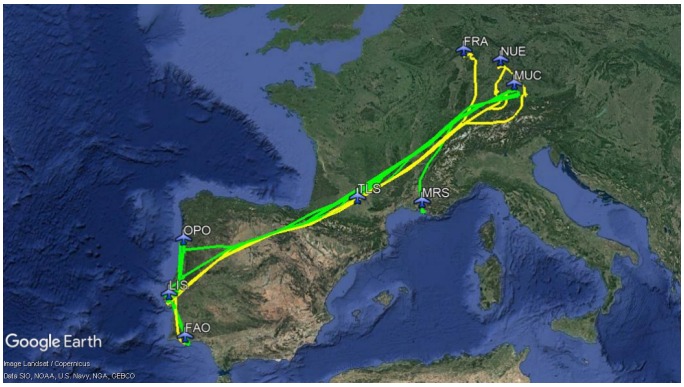
TÉCNICO LISBOA



Test scenario generation for validation of a black-box automated aircraft trajectory generator

Miguel Consiglieri Pedroso Mendes Dias

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Cláudia Alexandra Magalhães Soares
Prof. Maria da Conceição Esperança Amado

Examination Committee

Chairperson: Prof. Fernando José Parracho Lau
Supervisor: Prof. Maria da Conceição Esperança Amado
Member of the Committee: Prof. Frederico José Prata Rente Reis Afonso

November 2021

Acknowledgments

First and foremost, I wish to express my gratitude to Prof. Cláudia Soares and Prof. Maria da Conceição Amado for their precious guidance during the development of this thesis. It was a true pleasure to be oriented by them and to benefit from their knowledge and continued warmest availability.

I would also like to thank Prof. Fernando Lau for his support from the very beginning of my “Instituto Superior Técnico - ISAE-Supaero” journey, as well as for his larger efforts to create and broaden as much as possible IST’s Aeroespacial students international opportunities.

To my work colleagues, to all my IST and ISAE-Supaero colleagues, and to all my friends, I want to thank for the great work-study-personal life balance they helped me achieve, always enjoying happy moments in the three environments (professional, university and personal), which was fundamental to succeed.

Finally, I would like to thank my family for always motivating me to thrive and work to reach my full potential, as this encouragement strengthened my resolve to enrol on this Double Degree program from two of the most prestigious Universities in the air and space field.

Resumo

O progresso da Inteligência Artificial, suportado por *hardware* sem precedentes, promete revolucionar a aviação. Numa emergência, um sistema de navegação autónoma deveria propor uma trajetória segura até uma pista adequada para salvar o avião em tempo real. Sendo a segurança crítica em aviação, pois falhas podem levar à perda de vidas humanas, validar estas funções é fundamental. No entanto, a complexidade ou natureza de caixa preta dos sistemas tornam os métodos formais de validação impraticáveis.

Propõe-se uma estrutura baseada em dados para sustentar o desenvolvimento e validação de uma caixa preta que gera automaticamente procedimentos de emergência. Trajetórias suaves completas são reconstruidas a partir de observações históricas do FlightRadar24 com optimização convexa. Estas, permitem avaliar se a representação simplificada do terreno utilizada pelo sistema é suficientemente precisa para aceitar como seguras rotas habituais. Grupos de trajetórias são obtidos com *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN), sendo para cada grupo a lista de aeroportos adequados para desvio construida resolvendo um problema de cobertura de conjuntos, utilizando dados históricos e garantindo que existe sempre uma opção dentro de certa distância. Cada grupo é representado por um conjunto de células e o sistema é nestas avaliado, cobrindo o espaço relevante. Cenários representativos são construidos utilizando zonas restritas reais e obstáculos históricos de meteorologia. Um algoritmo genético guia a procura por cenários desafiantes.

A estrutura proposta foi validada aplicando-a num protótipo em desenvolvimento, permitindo identificar com sucesso eixos de melhoria e provando ser bastante útil para sustentar o processo de *design*.

Palavras-chave: Validação de segurança de caixas negras, Falsificação, Optimização, *Clustering* de trajetórias, *Big Data*

Abstract

Considerable progress in Artificial Intelligence, supported by unprecedented hardware capabilities, promises to revolutionize aviation. In an emergency, an aircraft autonomous navigation system should propose a safe trajectory until a suitable diversion runway to save the aircraft in real time. However, aviation is safety-critical, since failures might incur loss of human lives. Validating these automatic functions is therefore fundamental. Due to systems increasing complexity, or black-box nature, formal validation methods become impractical.

We propose a data-driven framework to support the development and validation of a black-box automated emergency procedure generator. Smooth complete trajectories are reconstructed from FlightRadar24 historical measurements through convex optimization. These allow to evaluate whether the simplified terrain representation used by the system is sufficiently precise to accept usually flied routes as safe. Trajectory clusters are obtained thanks to Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) and for each cluster a list of suitable diversion airports is constructed from historical data, ensuring that an option always exists within a certain distance through a set cover problem. Each cluster is represented by a group of cells and the system is evaluated on them, ensuring relevant search space coverage. Representative environments are constructed using real restricted areas and historical weather obstacles. A genetic algorithm guides the search for challenging scenarios.

The proposed framework was validated by applying it on a prototype under development, successfully identifying several axis of improvement, hence, proving itself useful to support the design process.

Keywords: Black-Box Safety Validation, Falsification, Optimization, Trajectory Clustering, Big Data

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

θ_{wx}	Critical weather obstacles orientation.
λ_1, λ_2	Regularization parameters.
μ	Cluster mean.
π	Cluster frequency.
Σ	Intracluster covariance matrix.
φ	Temporal formula specification.

Roman symbols

a	System action.
d	Ground distance between the aircraft and the diversion target.
D_2	Acceleration operator matrix.
D_3	Jerk operator matrix.
D_D	Dynamic Time Warping distance.
D_E	Euclidean distance.
D_H	Hausdorff distance.
f	Fitness function in a Genetic Algorithm.
f_i	Cost function term to evaluate the diversion target selection.
f_{SUT}	Cost function term to evaluate the optimality of the trajectory produced.
g	Cost function term allowing to determine the optimality of a trajectory length-wise.
h	Cost function term allowing to compare the trajectories from two prototypes.
K	Number of clusters.
K_{ts}	Number of piece-wise constant input segments in a time-staged approach.

l Ground length of a trajectory produced by the SUT.

N Number of points of a trajectory.

n_{restr} Restricted areas selected from a library to construct the scenario.

n_{wx} Critical weather obstacles selected from a library to construct the scenario.

o System observation.

P Reconstructed trajectory.

$\widehat{P}$ Measured trajectory.

$P_s\left(x^i\right)$ Probability of selection of individual i in a Genetic Algorithm.

Q State-action value function.

q Sampling rate.

R Reward in a Reinforcement Learning problem.

s System state.

S_i Subset of U.

t Execution time.

T_{ldg} Trajectory duration.

T_{ts} Time horizon in a time-staged approach.

U Universe in a set cover problem.

u Hybrid system input.

X Disturbance trajectory.

$\mathbf{x}$ Optimization variable in the search for challenging scenarios.

x_{AC} Aircraft state.

x_{S_i} Binary variable determining if subset S_i is selected in a set cover problem.

x_{wx} Critical weather obstacles position.

Subscripts

F Frobenius norm.

Superscripts

-1 Inverse.

H Conjugate-transpose.

T Transpose.

Glossary

ACAS	Airborne Collision Avoidance System
ADS-B	Automatic Dependent Surveillance-Broadcast
AIC	Akaike Information Criterion
AIP	Aeronautical Information Procedures
AIRAC	Aeronautical Information Regulation And Control
AST	Adaptive Stress Testing
ATC	Air Traffic Control
ATIS	Automatic Terminal Information Services
BIC	Bayesian Information Criterion
CAT	CATegory of an ILS
DA	Decision Altitude
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DH	Decision Height
DTW	Dynamic Time Warping
EDDM	ICAO code for Munich airport
EM	Expectation Maximization
ETA	Estimated Time of Arrival
FAA	Federal Aviation Administration
FL	Flight Level
FMS	Flight Management System
GA	Genetic Algorithm
GMM	Gaussian Mixture Model
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GS	GlideSlope
HDBSCAN	Hierarchical DBSCAN*
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
ILP	Integer Linear Program
ILS	Instrument Landing System

LCS	Longest Common Subsequence
LIS	IATA code for Lisbon airport
LL	Log-Likelihood
LOC	Localizer
LPPT	ICAO code for Munich airport
LVO	Low Visibility Operations
MBN	Means-Based Navigation
MCTS	Monte Carlo Tree Search
METAR	METeorological Aerodrome Report
MLE	Maximum Likelihood Estimation
MST	Minimum Spanning Tree
MUC	IATA code for Munich airport
NASA	National Aeronautics and Space Administration
NDB	Non-Directional Beacon
NOTAM	Notice to Airmen
NPA	Non-Precision Approach
OCC	Operations Control Center
OPTICS	Ordering Points To Identify the Clustering Structure
PBN	Performance-Based Navigation
PCA	Principal Components Analysis
RDD	Resilient Distributed Datasets
RL	Reinforcement Learning
RNAV	aRea NAVigation
RNP	Required Navigation Performance
RRT	Rapidly-exploring Random Tree
RVR	Runway Visual Range
SIA	Service de l'Information Aéronautique
SID	Standard Instrument Departure
SRTM	Shuttle Radar Topography Mission
STAR	Standard Terminal Arrival Route
SUT	System Under Test
TCAS	Traffic Collision Advisory System
TMA	Terminal Maneuvering Area
UCB	Upper Confidence Bound
UCT	Upper Confidence Trees
VHF	Very High Frequency
VOR	VHF Omnidirectional Range

Chapter 1

Introduction

In this chapter, the industrial and academic motivations behind the topic chosen for this master thesis are presented in Section 1.1. From these are derived the objectives and deliverables presented in Section 1.2. Finally, Section 1.3 presents the thesis outline.

1.1 Motivation

In 2018, the International Air Transport Association (IATA) has forecast the growth in air passenger numbers for the next two decades taking into account political and economic trends [1]. In a scenario in which policy framework remained the same over that period, the forecast anticipated a 3.5% compound annual growth rate, suggesting that the number of passengers could double to 8.2 billion in 2037. Two other scenarios, reverse globalization and maximum liberalization were also considered, delivering different projections. However, regardless of the growth scenario, a significant rise in demand would take place (see Figure 1.1). As a result, a shortage of qualified pilots in the future was expected. Even though the COVID-19 pandemic has greatly impacted air traffic, as IATA does not expect global passenger traffic to return to pre-COVID-19 levels until 2024 [2], the problem remains for the long term.

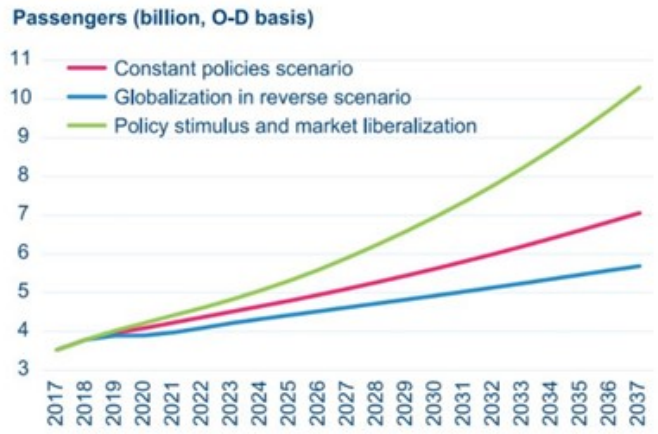


Figure 1.1: IATA's 2018 air traffic growth forecast. From [1].

Additionally, the cost associated with pilots' salaries, benefits and training is a significant part of aircraft operating costs, especially for aircraft with less seats (see Figure 1.2).

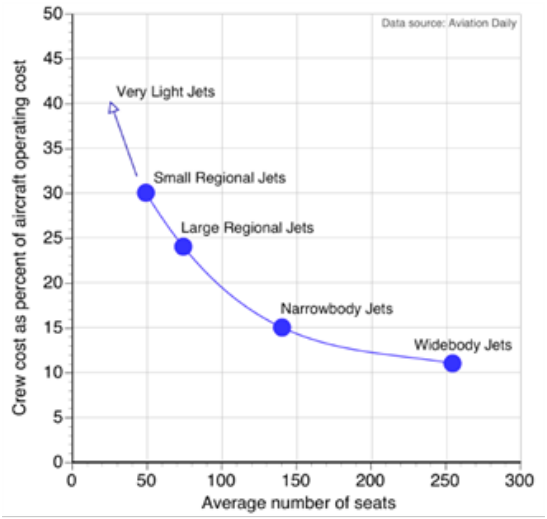


Figure 1.2: Crew cost versus number of seats. From [3].

For these reasons, working towards autonomous aircraft is important. It is equally important to develop functions that can assist pilots in stressful emergency situations where important decisions must be made quickly. In both of these contexts, a function capable of proposing in real-time a suitable diversion target and a safe and flyable trajectory until it to save the aircraft in case of emergency would be useful. Several works on this topic can be found in the literature [4] [5]. However, if such a function was to ever be validated and implemented on aircraft, being able to validate it is a far from trivial necessity. Since aviation is a safety-critical domain, where failures can incur severe consequences such as loss of life and property, any functions designed to render aircraft more autonomous or assist pilots must undergo extensive validation and testing before certification and deployment. As technological constraints are relaxed and systems become increasingly complex, often using Artificial Intelligence (AI) algorithms, formal validation methods become impractical, in favor of black-box techniques not requiring internal system knowledge.

This thesis addresses the design and implementation of an efficient strategy for the evaluation (and hopefully validation) of a black-box system that performs automatic diversion target selection and trajectory generation for emergency landing scenarios in real-time, avoiding terrain, critical weather and restricted areas. Between the aircraft and environment states, the search space is huge. Hence the interest of being able to efficiently search for falsifying inputs (inputs for which the System Under Test (SUT) is not capable of producing a solution or produces one not compliant with the requirements) while providing a good search space coverage. This allows to either find errors and identify axis of improvement (supporting the design and development processes) or be confident on the system validation.

Since big data infrastructures store huge amounts of relevant data, putting in place data-driven validation strategies that profit from this data is also desirable when pertinent. FlightRadar24 recordings show what real world commercial aircraft operations look like and therefore represent the scenarios on which the SUT would typically be used if ever implemented and deployed on commercial aircraft.

1.2 Objectives and Deliverables

The validation of functions contributing to the automation of aircraft or assistance to pilots is of huge industrial importance. For instance, autopilot (which allows to guide an aircraft without direct assistance from the pilot) and Flight Management System (which can be seen as an on-board computer for navigation, performance, and aircraft operations) functions have allowed to automate a wide variety of tasks and reduce the pilot workload while increasing safety. In light of the importance of being able to validate this type of function, the main objective of this master thesis is to design and implement an innovative and efficient framework providing an intelligent generation of test scenarios for a black-box automated diversion selection and trajectory generation function for emergency landing scenarios. The framework should allow to either find axis of improvement of the SUT (to feed the design and development processes) or provide confidence in its validation (with assurance of a good coverage of the relevant search space when no falsifying inputs are found). It should be able to construct scenarios representative of the ones on which the SUT is designed to be used at. For doing so, it should profit from historical aircraft data such as the one available on FlightRadar24, as well as from real restricted areas and representative historical weather obstacles.

The framework should also be tested on a use case. A black-box prototype being designed to perform real-time diversion selection and trajectory generation in case of emergency served as a use case. An objective is therefore to either validate the prototype or provide pertinent feedback on necessary improvements to the design team.

In developing the framework, a contribution should be made to the development of so needed new validation and certification strategies for future more intelligent aircraft functions. Given the interest of the subject, the identification of future work to continue the development of the framework is also crucial.

1.3 Thesis Outline

This section briefly explains the content of the different chapters composing this document and presents an overview of the framework proposed.

Chapter 2 presents the background required to understand the algorithms chosen to be implemented during the thesis, as well as some knowledge necessary to tune the simulation environment and validate the solutions produced by the SUT in a meaningful way. Section 2.1 presents black-box system validation methods from the literature, allowing for an informed choice of an intelligent state space exploration technique to be implemented. Section 2.2 briefly reviews aircraft navigation principles, providing operationally realistic navigation uncertainties to be considered when evaluating whether the simplified terrain representation used by the SUT is sufficiently precise to not declare common aircraft positions as unsafe. Section 2.3 presents a brief overview of approach operations. The understanding of these and associated automatic landing and low visibility capabilities enables an informed construction of a realistic list of diversion options to be inputted to the system. Section 2.4 presents a trajectory reconstruction method found in the literature and adopted for the reconstruction of complete and smooth aircraft trajec-

tories from discrete historical aircraft state recordings available on FlightRadar24. Reconstructed past trajectories feed both the terrain representation massive testing strategy and a designed data-driven validation strategy of the SUT. The latter requires the ability to cluster past trajectories, for which reason Section 2.5 reviews trajectory clustering algorithms from the literature.

In Chapter 3, the SUT is presented. Section 3.1 presents its purpose and the functionalities that it was designed for and which are under evaluation. Section 3.2 presents its interfaces. Sections 3.3 and 3.4 then support the construction of realistic restricted areas and critical weather obstacles respectively. These are two important environment definition inputs that allow generating complex and challenging environments for the SUT. Section 3.5 then specifies the criteria that must be met by a solution generated by the SUT for it to be accepted as valid. At last, Section 3.6 designs performance metrics and cost functions with the purpose of efficiently evaluating the complexity of the scenarios and the quality of the corresponding solutions generated by the SUT. Such performance metrics and cost functions will drive the search for falsifying inputs (inputs for which the SUT is not able of generating solutions or generates solutions not in accordance with the requirements defined in Section 3.5).

Chapter 4 presents the exploratory data analysis and data preprocessing carried out to support the data-driven evaluation strategies. The data source and dataset used are presented in Sections 4.1 and 4.2 respectively. Section 4.3 presents a data-driven support to the construction of at least part of the diversion list to be fed to the SUT. In fact, diversion choices that have been made by pilots or airlines Operations Control Center (OCC) historically should inspire the definition of a list of diversion options. In Section 4.4 part of the data is filtered and only complete and good quality data is kept. Finally, in Section 4.5 the implementation and tuning of the trajectory reconstruction algorithm is presented.

Chapter 5 presents a data-driven massive testing strategy designed for the SUT terrain representation validation. The requirements to be met by the terrain representation are first presented (Section 5.1). From them, the strategy is designed and applied to the SUT terrain representation of south-western europe in Section 5.2. Axis of improvement are identified and fed to the design team.

Chapter 6 proposes a strategy for the evaluation of the SUT performance on commercial routes. In Section 6.1 two clustering techniques are applied to the route trajectories clustering problem and the most suitable one is selected. Section 6.2 proposes a strategy for the construction of a list of diversion options for each cluster based on historical data and optimization to ensure that an airport is always available within one hour of flight. Section 6.3 evaluates the performance of the SUT on commercial routes. It proposes a genetic algorithm for the construction of challenging scenarios for the SUT. The algorithm successfully finds several scenarios where the SUT performance is not suitable and therefore provides important feedback to the design team. The same algorithm is used in Section 6.4 for the search of the worst scenarios for the SUT, regardless of their likelihood.

Chapter 7 reviews the most important achievements and contributions made during the thesis (Section 7.1) and lists important future work (Section 7.2). Given the interest of the framework, this is very important.

Figure 1.3 presents a high-level overview of the framework proposed.

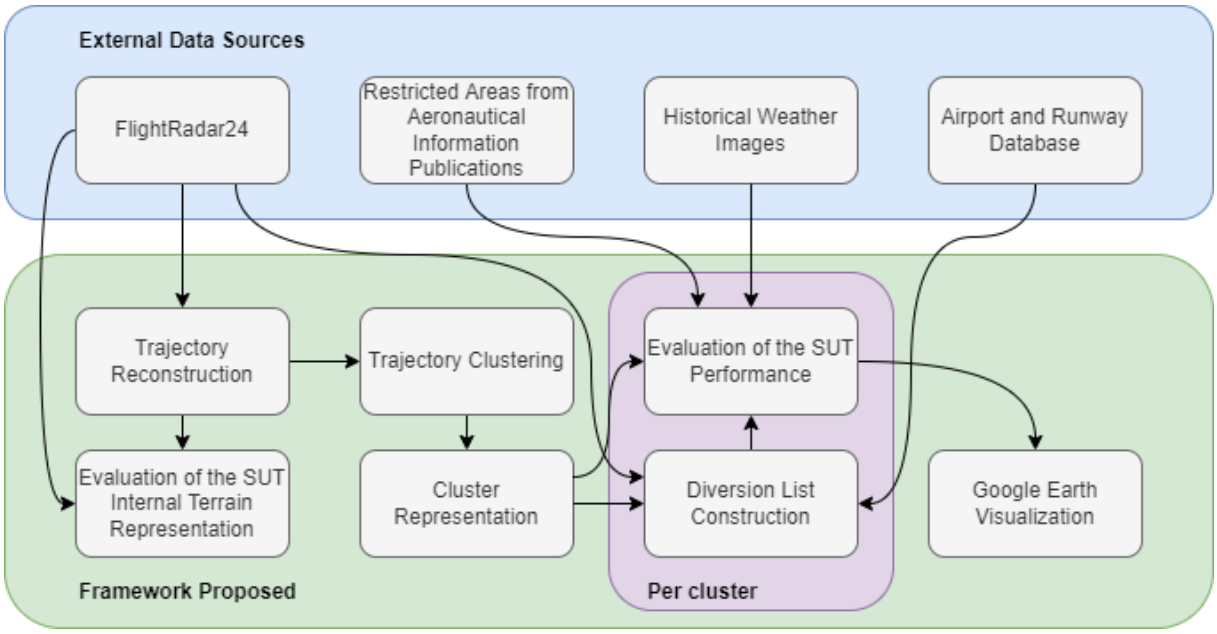


Figure 1.3: Overview of the framework proposed.

To begin with, historical FlightRadar24 recordings are used to reconstruct complete smooth trajectories. Both single recordings and reconstructed trajectories are then used to perform a data-driven evaluation of the internal terrain representation that the SUT uses in its computations. Reconstructed trajectories are also clustered and each cluster is represented by a group of cells. The cluster representation, historical aircraft diversion data from FlightRadar24 and an airport and runway database then feed the construction of a suitable list of diversion options for each cluster. Finally, the cluster representation, the cluster diversion list, real restricted areas retrieved from Aeronautical Information Publications (AIP) and historical weather images allow to feed the generation of realistic test scenarios for the SUT. The SUT is evaluated based on several cost function terms designed. The visualization and analysis of the results can be performed with the help of Google Earth.

Chapter 2

Background

This chapter presents some background required to understand the algorithms chosen to be implemented during the thesis, as well as some knowledge necessary to tune the simulation environment and validate the solutions produced by the SUT in a meaningful way.

2.1 Safety Validation of Black-Box Autonomous Systems

Considerable progress in artificial intelligence, namely in the fields of machine learning and path planning, supported by unprecedented hardware capabilities including Graphics Processing Units, has allowed autonomous systems to become increasingly capable, promising to revolutionize the automotive transportation and aviation industries, by improving convenience and efficiency while lowering cost. However, these are safety-critical domains, since failures can result in loss of life or property. Therefore, these systems must undergo extensive validation and testing prior to certification and deployment.

Safety validation is the process of ensuring the correct and safe operation of a system in an environment [6]. The desired system behavior is defined through a specification, and a failure is any violation of that specification. A SUT can be considered safe if no failure is found after adequate exploration of the input space, or if the failure probability is below an acceptable threshold. Safety validation can be incorporated at the different stages of development of an autonomous system, as shown in Figure 2.1.

During the definition phase, safety requirements such as do not make the aircraft fly through the terrain, restricted areas and critical weather obstacles are defined and mathematically formalized. The system is then designed in order to respect these constraints. When a prototype is available, safety validation includes testing, performance evaluation and interpretation of the system's failure modes. Testing may allow to discover bugs or identify requirements not met and in doing so can provide important feedback to the design team, who can improve the prototype until it meets all safety requirements. When validating a mature prototype, failures might be extremely rare and therefore challenging to find. Since it is not possible to guarantee the system safety in all situations, performance evaluation techniques are used to determine if the system can be accepted as safe. A great challenge lies on the modelling of the environment in which the aircraft flies, which is highly complex and stochastic.

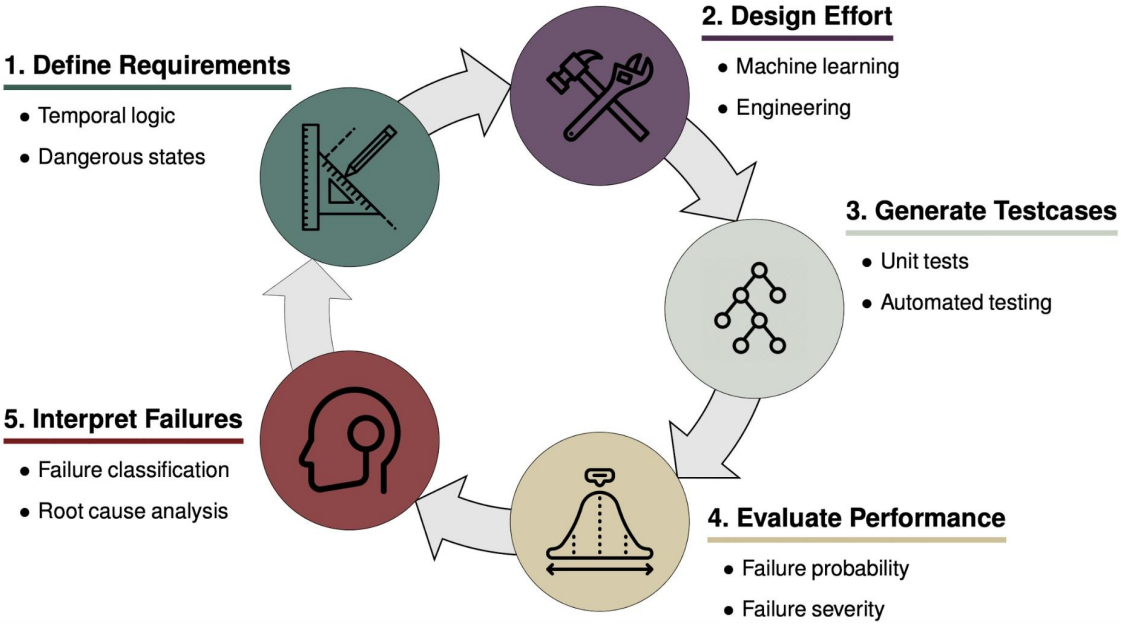


Figure 2.1: Autonomous system design cycle. From [7].

Conventional engineering analysis and testing does not scale well to the complexity of next-generation autonomous systems. Therefore, to build confidence in the systems, advanced validation techniques are required. Several approaches have been proposed in the literature for safety validation. In white-box approaches, knowledge of the internal design of the system is used for the construction of challenging scenarios and evaluation of the corresponding system's behavior. The interest of this approach is that the results are usually easily interpretable and provide a high degree of confidence in the system. However, white-box methods can suffer from scalability problems as the SUT becomes increasingly complex. Additionally, the internals of the SUT are not always available, as it is the case in this thesis. This section will therefore focus on black-box system validation algorithms, which scale well and allow to evaluate complex systems possibly containing machine learning-based components.

The work in [8] has proposed an effective method for automatic generation of test inputs for Embedded Control Systems, eliminating the time-consuming task of creating them manually while also allowing the acceleration of design cycles. A measure of coverage can be deduced from the system outputs and a coverage specification to be achieved is proposed. An optimization problem of maximizing the coverage function therefore needs to be solved. The function is constructed such that the maximum of zero is reached if and only if all the regions to be visited are reached. Since simulation models for embedded control systems are too complex, traditional gradient-based optimization algorithms cannot be used, for which reason a global search algorithm is needed. In this way, the authors propose a Genetic Algorithm (GA) where the coverage measure corresponding to an input is obtained through explicit simulation of the system and conversion of the output to the coverage measure. Selection is then performed based on it, as Figure 2.2 exemplifies.

To explain how the GA is used for test input generation, consider the initial population of chromosomes is seeded with random inputs. Then, in the selection phase, if $x^1, \ldots, x^N$ represents the popula-

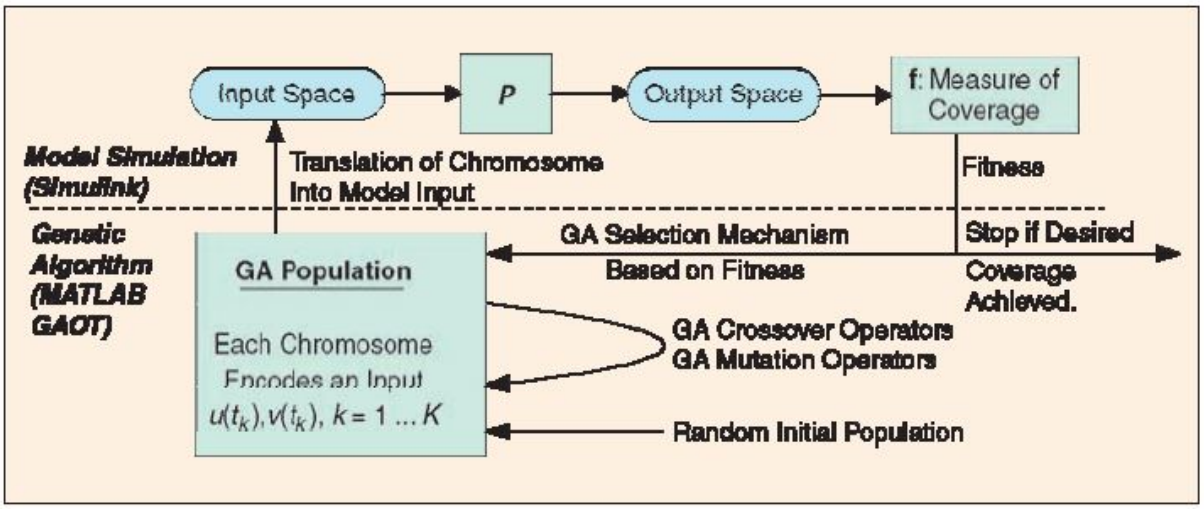


Figure 2.2: GA use for test input generation. Model simulation is used to measure fitness. From [8].

tion, a probability of selection $P_s\left(x^i\right)$ is assigned to x^i (2.1).

$$P_s\left(x^i\right) = \frac{f\left(x^i\right)}{\sum_{j=1}^{N} f\left(x^j\right)} \tag{2.1}$$

where f is the measure of coverage for each output. Crossover and mutation operators are designed to take advantage of the structure of the problem. Given two real-coded chromosomes $x^1 = \left(x_1^1, \ldots, x_n^1\right)$ and $x^2 = \left(x_1^2, \ldots, x_n^2\right)$, simple crossover produces two new chromosomes $x^{1\prime}$ and $x^{2\prime}$ (2.2).

$$\begin{aligned} x^{1\prime} &= \left(x_1^1, \ldots, x_i^1, x_{i+1}^2, \ldots, x_n^2\right) \\ x^{2\prime} &= \left(x_1^2, \ldots, x_i^2, x_{i+1}^1, \ldots, x_n^1\right), \end{aligned} \tag{2.2}$$

where $i \in \{1, \ldots, n-1\}$ is chosen randomly. Arithmetic crossover generates $x^{1\prime\prime}$ and $x^{2\prime\prime}$ (2.3).

$$\begin{aligned} x^{1\prime\prime} &= \lambda x^1 + (1-\lambda) x^2 \\ x^{2\prime\prime} &= \lambda x^2 + (1-\lambda) x^1 \end{aligned} \tag{2.3}$$

For given chromosomes x^1 and x^2, a new crossover operator proposed produces two chromosomes (2.4).

$$\begin{aligned} x_i^{1\prime} &= \min\left(\max\left(\left(x^1 + x^2\right)_i, x_i^l\right), x_i^u\right) \\ x_i^{2\prime} &= \min\left(\max\left(\left(x^1 - x^2\right)_i, x_i^l\right), x_i^u\right) \end{aligned} \tag{2.4}$$

where $i = 1, \ldots, p^\star K$, and x_i^l and x_i^u are the lower and upper bounds of variable x_i respectively.

As far as mutation is concerned, it seeks to ensure that the search space is explored. For a given chromosome x, uniform mutation randomly selects one variable x_i and sets it equal to random number s_i uniformly distributed on the interval $\left[x_i^l, x_i^u\right]$. The result of uniform mutation is given by (2.5).

$$x' = \left(x_1, \ldots, x_{i-1}, s_i, x_{i+1}, \ldots, x_n\right) \tag{2.5}$$

Boundary mutation sets s_i to either the upper bound x_i^u or the lower bound x_i^l.

The approach proposed was first evaluated on a Matlab Simulink model of a drive train for an automobile, which receives as input a throttle value in [0,100] and outputs both speed and RPM. The model also includes a Stateflow chart for the logic of an automatic transmission. The sampling time for the throttle scheduling being 5 seconds and the test total time being 30 seconds, each chromosome consists of a six-element sequence u_1, ..., u_6, defining the continuous-time input throttle (2.6).

$$u(t) = u_k, \quad t_k \leq t < t_{k+1} \tag{2.6}$$

The fitness was designed to drive the search for output speeds and RPM values above certain thresholds while ensuring all states are reached in the switching logics of the automatic transmission system during the 30 seconds. The approach generates solutions quickly. Its effectiveness was demonstrated for large production-size models from industry as well, generating solutions much faster than random searches and with a high success rate. Genetic algorithms therefore seem suitable for the safety validation problem addressed in this thesis. Mutation ensures exploration of the search space, while crossover might successfully combine challenging restricted areas and critical weather obstacles (exploitation).

In [9], the falsification problem is defined as follows:

1) Given model $\mathcal{M}$ (takes signal $\mathbf{u}$ and outputs $\mathcal{M}(\mathbf{u})$) and a specification φ (temporal formula), and

2) find an error input, that is, an input signal $\mathbf{u}$ such that the corresponding output $\mathcal{M}(\mathbf{u})$ violates φ.

The problem of hybrid system falsification is reduced to an optimization problem thanks to the use of real-valued robust semantics of temporal logics (Signal Temporal Logics is used as a specification language), which enables the use of hill-climbing optimization methods. Instead of simply representing the specification satisfaction through a Boolean, robust semantics assigns a quantity that represents not only if the specification formula is true or false but also how robust. Inputs are generated iteratively in the direction of decreasing robustness when searching for a falsifying input, as Figure 2.3 illustrates.

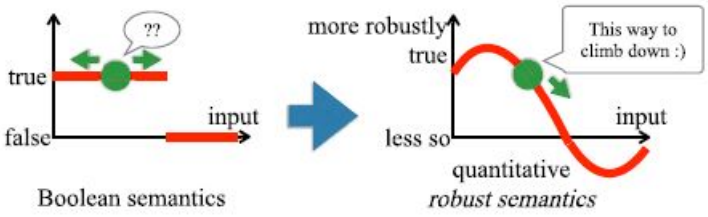


Figure 2.3: From boolean to robust semantics. From [9].

A time-staged approach to the falsification problem is followed, the full input sequence consisting of K_{ts} piecewise-constant input segments for times $[0, (T_{ts}/K_{ts})), [(T_{ts}/K_{ts}), (2T_{ts}/K_{ts})), ..., [((K_{ts} - 1)T_{ts}/K_{ts}), T_{ts}]$, T_{ts} being the time horizon. The approach can be converted to an adversarial form of Reinforcement Learning (RL).

Given the highly non-convex nature of the optimization problem, eager hill climbing could easily be trapped in local minima and fail to find falsifying inputs that exist elsewhere. It is also desirable to be confident about the system acceptance if no falsifying inputs ares found. For these reasons, it is important to

guarantee a certain coverage of the search space, which is possible if enough exploration is performed. The main contribution of this work is a two-layered optimization framework that balances exploration and exploitation in a systematic and mathematically disciplined way for hybrid system falsification.

In an upper layer, Monte Carlo Tree Search (MCTS) is used for high-level planning, picking a region from the input space. A lower layer then performs hill-climbing optimization within the region selected to sample the actual input. MCTS can therefore control the balance between exploration and exploitation. If it decides to create new children, it is forcing the lower layer to sample from a region not yet explored. Otherwise, it is forcing the lower layer to exploit a direction that seems promising. Once an input has been chosen, the system is simulated to obtain the corresponding output and the robustness is computed and fed back to the upper layer as a reward, allowing the tree search strategy to balance exploration and exploitation in subsequent iterations. This is illustrated in Figure 2.4.

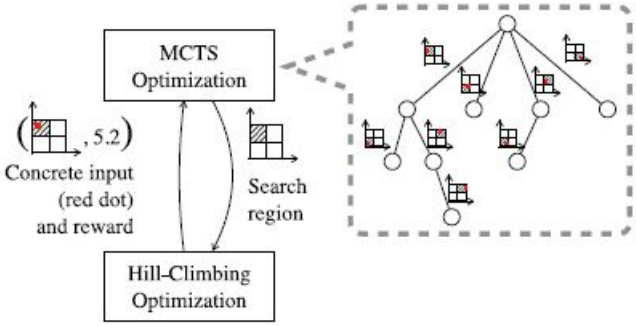


Figure 2.4: Two-layered optimisation framework. From [9].

The search tree is of depth K and an edge from depth $i-1$ to i determines the input value u_i for the interval $[((i-1)T/K), (iT/K))$. The goal is to find a sequence $u_1, ..., u_K$ that falsifies the system. Figure 2.5 illustrates the piecewise-constant inputs and corresponding system response.

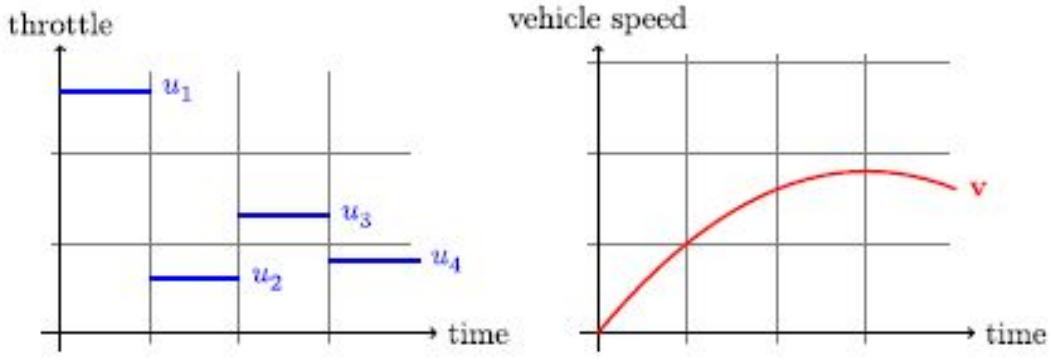


Figure 2.5: Piecewise constant input signal (1-D, throttle, left) for a simple automotive powertrain model, and the corresponding output signal (1-D, vehicle speed, right). From [9].

In the upper layer, the node choice determines the exploration/exploitation tradeoff. The Upper Confidence Trees (UCT) strategy [10], based on the Upper Confidence Bound (UCB) strategy for the multi-armed bandit problems [11], is used to solve the exploration/exploitation dilemma. For the MCTS im-

plementation, the article proposes both a basic implementation and an enhanced implementation with progressive widening. Progressive widening is a MCTS technique for dealing with a large or infinite action set, in which case expanding all children incurs a lot of computational cost or is even impossible. The principles behind MCTS and UCT are revisited shortly after in this section.

For the lower layer, three stochastic optimization algorithms are used in [9]: Simulated Annealing (SA), Globalized Nelder-Mead (GNM) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The choice of the algorithm greatly influences the results obtained.

The proposed framework is demonstrated through experiments with an automatic transmission Simulink model that receives throttle and brake continuous inputs and outputs the car's speed, engine rotation and selected gear. This model is commonly used as benchmark for falsification.

For the safety validation of ACAS (the next generation of TCAS), [12] presents Adaptive Stress Testing (AST), a framework that formulates the search for the most likely state trajectory leading to an event given only a simulator of the system as a Reinforcement Learning (RL) problem. The problem is successfully solved using MCTS and the approach shows significantly better performance than previous analysis using direct Monte Carlo search.

The system underlying the simulator is assumed to be discrete-time Markovian with stochastic transitions. However, its state being hidden, the process is non-Markovian from the search algorithm's point of view. s_t denotes the internal system state at time t and the likelihood of a trajectory is given by $\prod_{t=0}^{T} P\left(s_{t+1} \mid s_t\right)$, where $P\left(s_{t+1} \mid s_t\right)$ is the conditional probability of the system transitioning to state s_{t+1} if it is in state s_t at time t. To find a trajectory that leads to a critical event and maximize its likelihood, the problem is recast into a decision-making problem and a variation of MCTS is applied. As illustrated in Figure 2.6, the system takes as inputs a seed and basic simulation controls and outputs the likelihood of the corresponding transition and whether the current state is an event. The reward is computed and feeds the MCTS algorithm, which closes the loop choosing the next seed and control inputs based on the rewards received insofar. Since the seed is an input, a sequence of decisions uniquely determines the state of the system, rather than allowing the system to evolve naturally and stochastically.

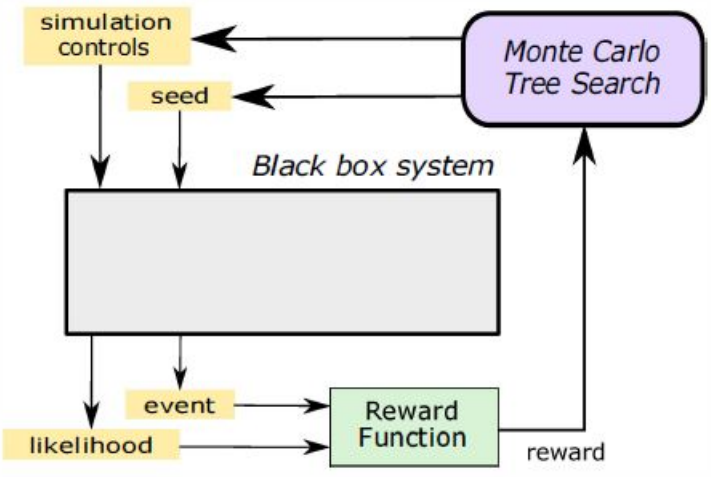


Figure 2.6: MCTS for test input generation. From [12].

RL algorithms try to maximize the reward, so the reward presented in Equation (2.7) was designed.

$$R\left(s_t, s_{t+1}\right) = \begin{cases} 0 & \text{if } s_t \in E \\ -\infty & \text{if } s_t \notin E, t \geq T \\ \log P\left(s_{t+1} \mid s_t\right) & \text{if } s_t \notin E, t < T \end{cases} \tag{2.7}$$

The reward is maximized in case an event is found and, otherwise, it favors sequences that are more likely. Using $\log P\left(s_{t+1} \mid s_t\right)$ as a reward, $\sum_{t=0}^{T} \log P\left(s_{t+1} \mid s_t\right)$ is maximized, which is equivalent to maximizing $\prod_{t=0}^{T} P\left(s_{t+1} \mid s_t\right)$. Each term in the reward function is non-positive, the total reward $\sum_{t=0}^{T} R\left(s_t, s_{t+1}\right)$ being non-positive as well. In fact, if an event E occurs, then the total reward is the log likelihood of the trajectory. Otherwise, the total reward is $-\infty$. Let us revisit the principles of MCTS.

MCTS is one of the most successful sampling-based online approaches to RL. This means that the data becomes available in a sequential order and is used to update the best predictor for future data at each step (unlike batch learning techniques, which generate the best predictor by learning on the entire training data set at once). A search tree is incrementally built using sampling and forward simulation to inform the search and focus on the most promising areas. It is based on roll-outs, in which the game is played out until the end by randomly selecting the moves and using the result of the game to weigh the nodes in the game tree and guide the random selection of the next roll-outs. A weight in the tree is the state-action value function $Q(s, a)$, which represents the expected sum of rewards resulting from choosing action a in state s. Simulations are run until a stopping criteria is met, usually a number of iterations. Every iteration includes the four following steps (illustrated in Figure 2.7):

1. **Selection:** from the root (current game state), select successive child nodes until a leaf node (node from which no roll-out has been done yet) is reached. The selection can be biased to favor child nodes that expand towards the most promising moves.

2. **Expansion:** from the leaf node chosen, choose one of the missing actions and insert it in the tree.

3. **Simulation:** compute the reward through a complete roll-out (simulation until game completion).

4. **Backpropagation:** Update information in the nodes of the tree between the root and the node from which the roll-out started, using the reward obtained from the roll-out. Return to selection.

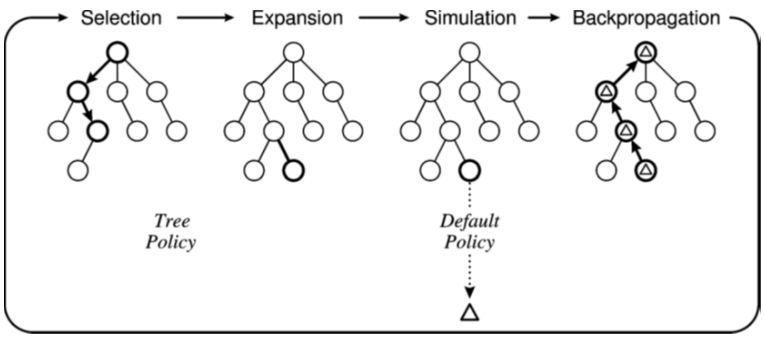


Figure 2.7: MCTS steps. From [13].

Non-Markovian systems are in principle not supported by MCTS, since MCTS requires the ability to set the system state back to any visited node of the tree, which is not possible when the system is in a stochastic environment. To solve this, the seed is an input of the simulator in [12], which means that each transition from s_t to s_{t+1} is deterministic given action a_t, and therefore it is possible to return to any visited node by remembering the sequence of actions $a_{0:t-1} = a_0, ..., a_{t-1}$ taken since s_0.

The most common selection strategy is UCT - UCB applied to Trees [10]. Derived from the UCB strategy for the multiarmed bandit problem [11], it selects action a that maximizes:

$$Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$$

where $N(s,a)$ is the total number of visits to action a in state s and c is a parameter that controls the amount of exploration in the search. The first term favors exploitation - $Q(s,a)$ is higher for nodes that have proven successful before. The second term favors exploration - it is higher for nodes that have been chosen less times before. The balance between exploration and exploitation provided by MCTS is precisely one of the reasons for its success, allowing it to both explore broadly and focus on potentially high-reward areas. There are several search strategies in the literature. For instance, [12] uses the *double progressive widening* technique to to control the branching factor of the search tree. At each visited state node, the first progressive widening chooses whether to select amongst existing actions or to generate a new one. At each visited action node, the second progressive widening criterion determines whether to follow an existing next state node, or to draw a new next state from the simulator. There are several other variations of MCTS. A detailed review of these is presented in [13].

In [6] is presented a comprehensive survey of algorithms for black-box safety validation. The problem formulation is represented in Figure 2.8. There is an environment that the system observes with its sensors and in which it takes actions based on the observations. The problem is finding a disturbance trajectory $X = [x_1, \ldots, x_N]$ that leads to failure. Disturbances might include sensor noise, the behavior of other agents in the environment, or yet environmental conditions. For the SUT in this thesis, disturbances could represent positioning system errors, the unexpected opening or closing of restricted areas or yet an unexpected temporal evolution of critical weather. The adversary represented in Figure 2.8 knows the environment state and seeks to find X that leads to failure. It has one of the following goals:

1. Falsification: Find any disturbance trajectory X that leads to a failure;
2. Most likely failure analysis: Find the most likely disturbance trajectory that leads to a failure. For this, a model of the disturbances in the environment $p(X)$ describing which sequences of disturbances are most likely, obtained either from expert knowledge or real-world data, is required. We then want to find disturbance trajectories leading to a failure that maximize $p(X)$;
3. Estimation of the probability of failure: Determine how likely any failure will occur based on $p(X)$.

This work focuses on falsification. The taxonomy in [6] categorizes three approaches for this end:

1. **Optimization.** An adaptive sampling from the space of possible disturbance trajectories is guided by a cost function. When safety requirements are expressed formally using temporal logic, the cost

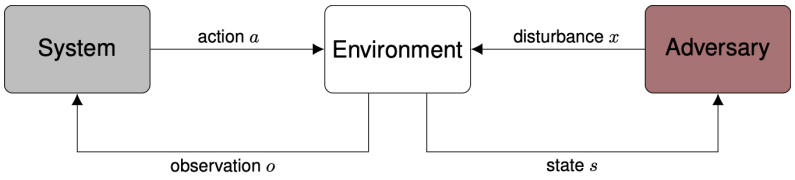


Figure 2.8: Problem formulation for the safety validation of black-box autonomous systems. From [6].

function usually uses the *robustness* [14], which measures how close a trajectory is to violating the specification [15]. Several algorithms have been successfully used for safety validation, namely Simulated Annealing (SA), Genetic Algorithms (GA), Bayesian Optimization (BO), Extended Ant-Colony Optimization (ACO), Genetic Programming (GP) and Efficient Global Optimization (EGO).

2. **Path-planning.** Algorithms such as rapidly-exploring random tree (RRT) sequentially build disturbance trajectories by choosing disturbances that bring the environment into unexplored regions of the state space. RRT has successfully been used to find failures of an adaptive cruise control system where failures involved complex motion of the lead vehicle [16]. However, the application of these methods to the safety validation of black-box systems is not direct, since they were designed for white-box systems and environments where dynamics and gradient information are available.

3. **Reinforcement Learning (RL).** When the next state of the environment is only a function of the current state and disturbance a Markov decision process (MDP) can describe its evolution. As a result, RL algorithms like MCTS and deep RL algorithms like Deep Q-Networks or Proximal Policy Optimization can be used to select disturbances based only on the current state. MCTS is designed for use with black-box systems, a great advantage being that it can be applied in information-poor environments. It has successfully been used to find failures of an aircraft collision avoidance system without access to the simulator state [12]. Deep RL takes advantage of the large representational capacity of neural networks and advanced optimization techniques to solve problems with large state spaces, complex dynamics, and large action spaces. It has successfully been used to find failures of autonomous driving policies [17].

The three approaches presented can address both falsification and most likely failure analysis. The survey [6] also presents an approach named importance sampling that addresses the estimation of the probability of failure of the system. Addressing this task might be interesting in future work.

It is important to remark that the problem of finding a disturbance trajectory X leading to failure is much more challenging than just finding a single disturbance. Such an approach can find sequences of actions leading a self-driving car from nominal operation to running over a pedestrian or crashing against a wall, for instance. This thesis only addresses a ”static” validation of the SUT, meaning its ability to select a good diversion option and generate a safe and flyable trajectory in light of the initial environment and aircraft states. To address this problem, an optimization approach seems suitable. The capacity to deal with disturbances that render the solution initially found and engaged invalid is not tested yet. For this, an optimization approach is no longer well suited. However, this search for such scenarios is fundamental in future work. Once the SUT capacity to produce suitable solutions within

a given input space is demonstrated, its robustness to dynamic disturbances must also be validated, since it will operate over time, rather than be requested from isolated scenarios only. The SUT must be capable of computing new solutions to save the aircraft in case while flying the one original computed there are disturbances such as aircraft failures or unexpected weather evolution making it invalid. A sequence of disturbances taking the aircraft to a situation from where the SUT can no longer compute an emergency trajectory would be catastrophic.

Considerable work in the literature has focused on falsification, as well as on providing runtime assurances and ensuring coverage of the search space. However, it does not usually provide information about how the autonomous system performs when executing a mission nor does it provide an insight into the environmental factors that contribute to its decision-making process. The work in [18] proposes a novel method for generating test scenarios for a black-box autonomous system that demonstrates critical transitions in its performance modes, allowing to interpret the decision-making process of the autonomous system and how environmental factors affect it. To provide guarantees of the system's decision-making capabilities without running extensive tests, [18] focuses on the regions where small changes in the scenario result in transitions between performance modes. In the SUT analysed in this thesis, a typical example would be how a small change to the position of the aircraft or of an obstacle could make the system choose a different airport or decide to go around an obstacle by the other side of it. Understanding where these transitions take place allows understanding and predicting the performance of the autonomous systems and is useful for both their design and validation.

Sampling preferentially from performance boundary regions instead of spending resources exploring regions of stable performance allows to maximize the information returned by a limited number of runs. Therefore, in the first step of the strategy proposed, adaptive sampling is used to search the parameter space. In a second step, unsupervised learning is used to determine the performance modes and the cases that make up the performance boundaries. Figure 2.9 presents an overview of this approach.

In adaptive sampling, iterative queries are submitted to the SUT and the returned scores are used for generating a meta-model. An information metric is then applied to the meta-model to generate the new set of queries, instead of attempting to optimize uniform coverage and density using pre-computed space-filling methods. To sample from performance boundaries, the metrics have been designed to look for areas with high gradients that have not been sampled yet. Two meta-models have been proposed: one using Gaussian Process Regression (GPR) and another one using a k-nearest neighbor method for density and variance estimation. To implement step 2 of the proposed strategy, Mean-Shift clustering is used to identify the different performance modes and classify the samples, which are then clustered using the DBSCAN algorithm [19]. DBSCAN is an unsupervised spacial clustering algorithm that identifies distinct regions of arbitrary shapes without requiring a priori knowledge of the number of clusters.

2.2 Aircraft Navigation

In aeronautics, Navigation is a subject that encompasses positioning, flight planning and controlling the aircraft, allowing to determine where the aircraft is, where it should go and how it can go there.

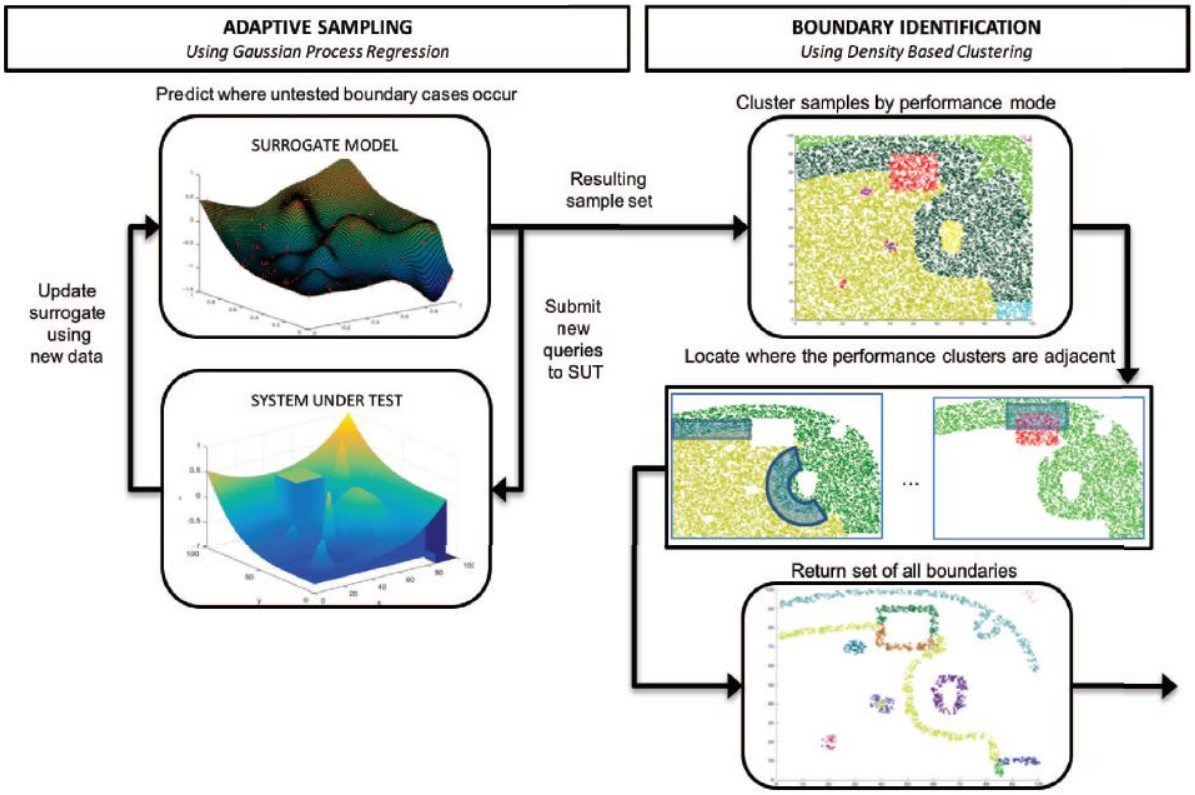


Figure 2.9: Test generation cycle overview for a SUT with continuous unlabeled outputs. From [18].

These tasks must be accomplished under several constraints, related to the following aspects:

- Weather;
- Traffic;
- Noise and Emissions;
- Fuel Consumption;
- Time;
- Safety;
- Air Traffic Control (ATC);
- Airline Operations Control Center (OCC).

Historically, Navigation was first performed by an on-board Flight Navigator (since the 1910s). Radio aids were then introduced in the 1940s and were followed by the introduction of the first Inertial Navigation System (INS) and also of waypoints in the 1970s. As a result, the Flight Navigator was decommissioned in the 1980s, in the same decade when the first FMS was introduced. In the 1990s, GNSS-based means were introduced on aircraft and, nowadays, augmented GNSS means are available on-board. GPS allowed a very significant step in accuracy and integrity.

Two types of Navigation exist: Performance Based Navigation (PBN) and Means Based Navigation (MBN). In MBN, the older one, operation is based on a particular mean that must be used. PBN operations are based on a set of performances to be met (accuracy, integrity and continuity), regardless of the means used as long as these and other linked requirements (safety, availability, operational) are met.

In traditional MBN, Navigation is constrained by the radar and the course and distance coming from ground beacons such as Very high frequency Omni-directional Range (VOR) beacons or Non-Directional Beacon (NDB) beacons. As Figure 2.11 illustrates, navigation through these is strongly constrained.

PBN describes aircraft capability to navigate using performance standards. There are families of PBN specifications, such as aRea NAVigation (RNAV) and Required Navigation Performance (RNP). These navigation specifications define the performance that the avionic system must demonstrate through the allowed Total System Error (TSE). The TSE encompasses the three different errors presented in Figure 2.10: Path Definition Error (PDE), Flight Technical Error (FTE) and Navigation System Error (NSE). PDE measures the error between the theoretical path and the path that is stored on board on the Navigation Database. It is therefore usually negligible. FTE measures the guidance error. Finally, NSE is the estimated error of the aircraft navigation, which mainly consists of the Estimated Position Uncertainty (EPU) - the estimation of the error made when computing the aircraft position.

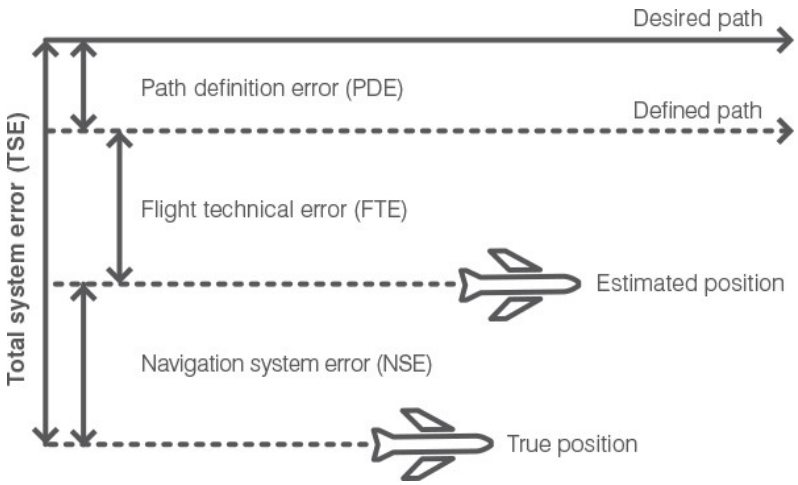


Figure 2.10: Total System Error (TSE) components. From [20].

With the FMS RNAV capability, aircraft can fly from waypoint to waypoint defined by geographic coordinates instead of through navaids [21]. Tracks are independent from ground equipment and accurate navigation and flight path tracking becomes possible (see Figure 2.11). An RNAV specification is defined as RNAV X, where X refers to the lateral navigation accuracy expected to be achieved at least 95% of the flight time within the airspace, route or procedure, in nautical miles. RNAV1 is used for Terminal flight phase and RNAV2, RNAV5 and RNAV10 for cruise.

RNP [22] is another family of PBN specifications. It permits the operation of aircraft along a precise flight path with a high level of accuracy and the ability to determine aircraft position with both accuracy and integrity. RNP is very similar to RNAV, the key difference being that On Board Performance Monitoring and Alerting (OBPMA) requirements apply to RNP. More specifically, the accuracy requirement defines the 95% Total System Error (TSE) acceptable for the dimensions where an accuracy requirement

is specified, in harmony with the RNAV specifications. However, additionally, the OBPMA is required to monitor the TSE and provide an alert if the accuracy requirement is not met or if the probability that the TSE exceeds two-times the accuracy value is larger than 10^{-5}. As illustrated in Figure 2.11, RNP navigation allows for very precise navigation and therefore for an optimisation of the use of airspace, as well as reproducing flight paths with great accuracy on every flight. The advantage of relying on performance rather than relying on ground beacons is clear in Figure 2.11.

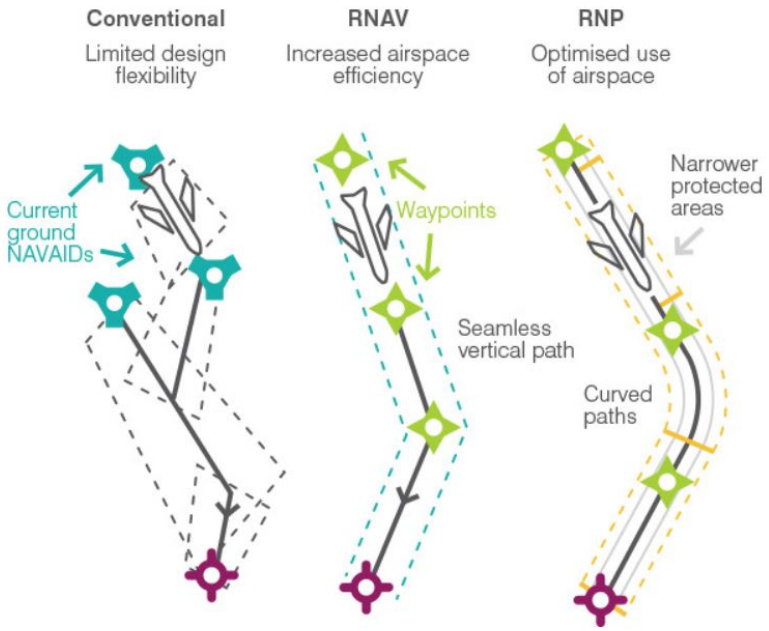


Figure 2.11: Transition from conventional navigation to RNP. From [20].

The International Civil Aviation Organization (ICAO) PBN manual [23] identifies seven RNP navigation specifications: RNP4, RNP2, RNP1, Advanced RNP, RNP APCH, RNP AR APCH and RNP 0.3. RNP 4 is for oceanic and remote continental navigation applications. RNP 2 is for en-route oceanic remote and en-route continental navigation applications. RNP 1 is for arrival and initial, intermediate and missed approach as well as departure navigation applications. Advanced RNP is for navigation in all phases of flight. RNP APCH (APproaCH) and RNP AR (authorisation required) APCH are for navigation applications during the approach phase of flight. RNP 0.3 is for the en-route continental, the arrival, the departure and the approach (excluding final approach) phases of flight and is specific to helicopter operations.

RNP specifications must be taken into account when determining whether the terrain representation of the SUT is sufficiently precise. RNAV and RNP navigation will probably ultimately replace all ground-based navigational aids, so these capabilities are assumed to be available when evaluating the SUT.

Since the trajectories output by the SUT do not consider the organization defined by air routes, airways, control areas and zones, nor typical aeronautical procedures such as STAR and SID routes, these and other navigation and ATC concepts are not reviewed here. In order to ensure a meaningful construction of the list of diversion options to be passed to the SUT, a brief review of approach operations is presented in Section 2.3.

2.3 Approach Operations

Throughout aviation history, approach operations have undergone considerable evolution (see Figure 2.12). Very early on the history of aircraft, there were only visual approaches. However, as time passed and technology evolved, several different approach operations have become available and increasingly reliable, allowing to guide the aircraft more accurately during the approach.

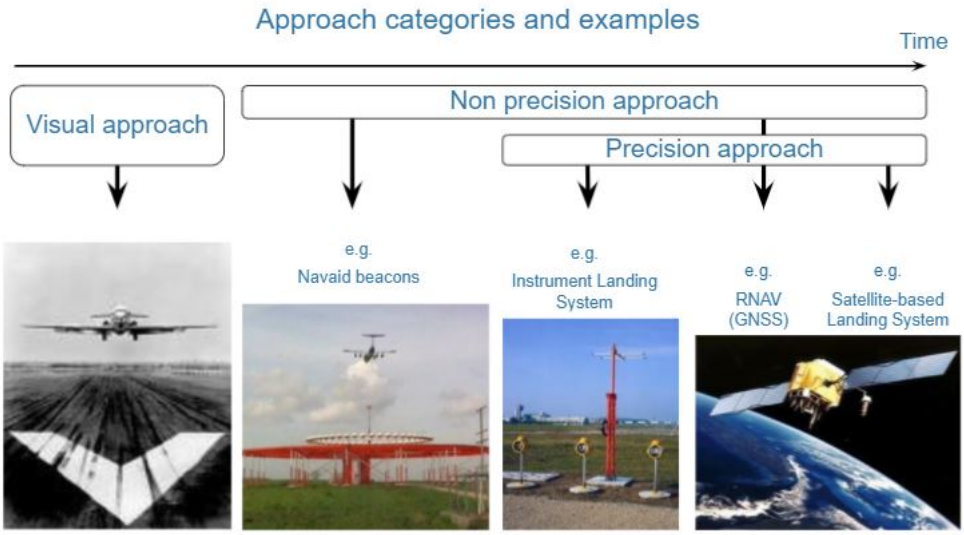


Figure 2.12: Approach operations evolution in time.

Non-precision approaches (NPA) based on Navaid beacons became available first. NPAs are approaches without on-board computed vertical guidance. These are mainly based on VOR and NDB, which are two different kinds of beacon used for navigation and approach, NDB being the less accurate one. A disadvantage of NPAs is that because of the dependence on the location of the ground station the approach course is constrained. Since NPAs do not require vertical guidance, we consider that only runways that have associated precision approaches should be chosen by the SUT as a target of emergency procedures. Therefore, when creating the list of diversion options to be fed as an input to the SUT, only runways with associated precision approaches will be considered.

Precision approaches appeared with the Instrument Landing System (ILS) [24], which is still today the pilots' preferred mean for approach. ILS is based on two directional radio signals: the localizer (LOC), which provides horizontal guidance, and the glideslope (GS), which provides vertical guidance. Figure 2.13 illustrates these signals and helps understand the ILS functioning principle. ILS LOC aerials are normally located at the end of the runway and transmit two narrow intersecting beams of different carrier frequencies, one slightly to the right and the other slightly to the left of the runway centreline. From the relative intensity of the signals it is possible to locate the aircraft with respect to the LOC axis. ILS GS aerials also transmit two narrow intersecting beams, one slightly below and the other slightly above the required vertical profile, intersecting at the GS axis. GS aerials are usually located so that the glide-slope provides a runway threshold crossing height of about 50 ft. The usual GS angle is 3º but exceptions may occur to meet particular approach constraints such as terrain or noise abatement.

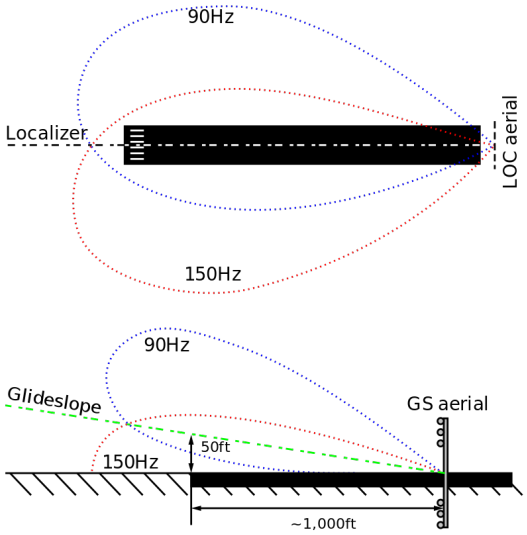


Figure 2.13: ILS principle. Localizer and glideslope radio signals.

Different categories of precision approaches are defined, depending on the value of two key parameters: the Decision Altitude/Height (DA/DH) and the Runway Visual Range (RVR). DA or DH is a specified altitude (with respect to mean sea level) or height (with respect to the threshold elevation) at which a Missed Approach must be initiated if the required visual reference to continue the approach has not been established [25], as illustrated in Figure 2.14. For category II and III approaches the minima can only be expressed as DH and not DA, since a radio altimeter is used, not a barometric altimeter.

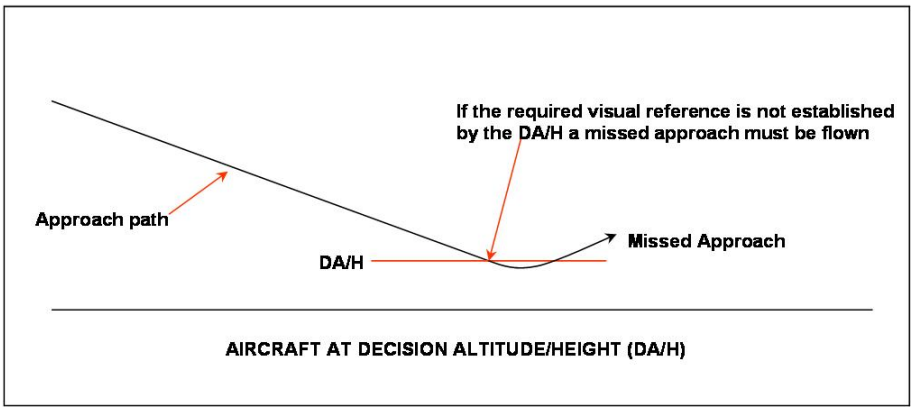


Figure 2.14: Decision Altitude/Height in precision approaches.

Runway Visual Range (RVR) is the range over which the pilot of an aircraft on the centre line of a runway can see its surface markings or the lights delineating it or identify its centre line [26]. Precision approach categories are defined based on DA/DH and RVR values as presented in Table 2.1.

The existence of these categories means that successfully flying an approach depends on the weather conditions on the airport. The higher the category the less constraining Low Visibility conditions can be. For this reason, in order to be robust to weather conditions, the lists of diversion options will be composed only of runways with associated CAT II and CAT III means, which are relatively stable for Low Visibility Operations (LVO). Approval for CAT II and CAT III operations is dependent on several

Table 2.1: Precision Approach Categories.

Category of Operation	Decision Height (DH)	RVR	Visibility
CAT I	$\geq 60m\ (200ft)$	$\geq 550m$	$\geq 800m$
CAT II	$< 60m\ (200ft) \wedge \geq 30m\ (100ft)$	$\geq 350m$	-
CAT IIIA	$< 30m\ (100ft)$	$\geq 200m$	-
CAT IIIB	$< 15m\ (50ft)$	$< 200m \wedge \geq 50m$	-
CAT IIIC	-	-	-

elements, among which is the aircraft having an automatic landing system [27]. It is considered to be the case in a highly automatic and futuristic aircraft.

With the use of satellite navigation and PBN for all flight phases including approach, other ILS look alike functions (xLS, where x varies) have been developed for precision approaches, standardizing the way of flying an approach. GLS stands for Ground-Based Augmentation System (GBAS) Landing System and it is an alternative to ILS that works thanks to differential GPS and local ground infrastructure. SLS stands for Satellite Based Augmentation System (SBAS) Landing System and it works thanks to geostationary SBAS satellites. FLS stands for FMS Landing System. For the moment, only ILS is autoland certified for all CAT approaches. In light of this, only ILS approaches are used to construct the diversion list.

As introduced in Section 2.2, there are RNAV and RNP approaches, where a bounded Total System Error (TSE) protects the approach path. For approach, a special specification, RNP AR (Authorization Required), exists for complex approach scenarios in mountainous areas with smaller margins. To fly these approaches aircraft performances have to meet higher requirements. Final approach trajectories are straight for all approach types except for RNP AR, for which they can be curved.

Many approach operation variants other than the ones presented here exist. However, the aim of this short review was not to be extensive, but rather to understand the basics of approach operations and be able to construct a meaningful list of diversion options that are compatible with autoland and with LVO.

2.4 Trajectory Reconstruction

This section presents a formulation of the problem of reconstructing a complete and smooth trajectory from a set of recorded measurements as a convex optimization problem (2.8).

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{subject to} \quad & g_i(x) \leq b_i, i = 1, \ldots, m \end{aligned} \tag{2.8}$$

where $f_0 : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the convex objective function to be minimized, $x = (x_1, \ldots, x_n)$ is the optimization variable whose value is to be determined and $g_i : \mathbb{R}^n \longrightarrow \mathbb{R}, i = 1, \ldots, m$ are the convex constraint functions that must respect the inequalities defined by $b_1, \ldots, b_m$. The success of the approach depends

on a pertinent objective function, which should ensure that not only the trajectory reconstructed passes close to the observations, but also that the flight dynamics corresponding to it is realistic. The convex optimization problem of trajectory reconstruction is presented hereafter as in [28].

Let q represent a sampling rate (in seconds) that allows to control the number of points per trajectory. Regardless of the number of observations available per trajectory, if all trajectories are uniformed to have a same duration T_{ldg}, then the number of points defining each reconstructed trajectory, N, is defined by (2.9).

$$\mathrm{N} = \frac{T_{\mathrm{ldg}}}{q} \tag{2.9}$$

Let $P = (p_1, p_2, \ldots, p_N) \in \mathbb{R}^{N\times 3}$ be the optimization variable and represent the trajectory to be reconstructed. The ith row of P, $p_i = (\ \text{lat}\ _i, \text{lon}\ _i, \text{alt}\ _i)$, is the reconstructed position at time i.

In order to ensure that the reconstructed trajectory stays truthful to the observations, a least-squares regression term $\|AP-\widehat{P}\|_F^2$, where $\|.\|_F$ defines the Frobenius norm, is inserted in the objective function. $\widehat{P} \in \mathbb{R}^{N\times 3}$ contains in the ith row the measurements of time i if these are available and zero otherwise. $A \in \mathbb{R}^{N\times N}$ is a diagonal matrix containing information on whether measurements are available at time i or not, and is defined by (2.10).

$$A_{ii} = \begin{cases} 1 & \text{if} \quad \text{an observation is available at time } i \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

Besides the fidelity to the observations, realistic flight dynamics also need to be included when reconstructing the trajectory. This is achieved through the introduction of two other terms in the objective function to encourage the reconstructed trajectory to have low acceleration and jerk on average.

In order to ensure that the acceleration is low on average, a term $\lambda_1 \|D_2 P\|_F^2$ is introduced. The scalar λ_1 is a regularization hyper-parameter to be tuned and $D_2 \in \mathbb{R}^{N-2\times N}$ is the second-order difference matrix representing the acceleration operator (2.11).

$$(D_2)_i = e_i - 2e_{i+1} + e_{i+2}, \quad i \in \{1, \ldots, N-2\} \tag{2.11}$$

where e_i denotes the i^{th} standard unit vector.

To ensure that the jerk, which represents the change of acceleration, is also low on average, the term $\lambda_2 \|D_3 P\|_F^2$ is also introduced in the objective function. Here, λ_2 is another scalar regularization hyper-parameter to be tuned and $D_3 \in \mathbb{R}^{N-4\times N}$ is the third-order difference matrix representing the jerk operator (2.12).

$$(D_3)_i = -e_{i-1} + 2e_i - 2e_{i+2} + e_{i+3}, \quad i \in \{1, \ldots, N-4\} \tag{2.12}$$

Summing the three terms that were just introduced, the objective function is defined in Equation 2.13.

$$f(P) = \|AP - \widehat{P}\|_F^2 + \lambda_1 \|D_2 P\|_F^2 + \lambda_2 \|D_3 P\|_F^2 \tag{2.13}$$

A convex unconstrained optimization problem is formulated by (2.14).

$$\underset{P}{\text{minimize}} \|AP - \widehat{P}\|_F^2 + \lambda_1 \|D_2 P\|_F^2 + \lambda_2 \|D_3 P\|_F^2 \tag{2.14}$$

The problem formulation presented in (2.14) was applied in [28] and [29], yielding good results. As far as the tuning of the regularization parameters λ_1 and λ_2 is concerned, the authors in [28] suggest that these are selected individually for each trajectory through out-of-sample validation. An out-of-sample validation can be performed by randomly holding out measurements from the trajectory, fitting trajectories with varying λ_1 and λ_2 using the measurements that were not held out, and selecting the parameters that have the lowest loss (i.e., Equation 2.13 with $\lambda_1 = \lambda_2 = 0$) on the held-out measurements. For potential formulation improvements, switching to a different loss, *e.g.*, Huber [30] or L_1, and adding constraints on the reconstructed trajectory are suggested. In [29] another tuning approach is followed, using the measurements of trajectories from a more accurate dataset to tune the regularization parameters for the reconstruction of trajectories from a dataset of lower quality.

2.5 Trajectory Clustering

In Machine Learning, Clustering is the process of dividing a set of data objects into groups (or clusters) in such a way that objects belonging to a same group are very similar to each other and very dissimilar to objects belonging to the other groups. Since this task is performed having only access to observations of the data, but not to any corresponding labels, it is considered an unsupervised learning task. It is a form of learning by observation, rather than learning by examples.

When performed successfully, clustering may allow discovering internal and previously unknown schemes inherent of the data. For this reason, it has been applied in a wide variety of contexts. In Marketing and Sales, for instance, clustering algorithms are able to perform customer segmentation, which can then be used to target customers with offers personalized to their preferences. Clustering can also be used for outlier detection. A classical application is the detection of credit card fraud. In the field of Aeronautics, clustering also has a variety of applications. In particular, trajectory clustering, the problem to be tackled in the context of this thesis, has been used to understand patterns and detect outliers in vectoring orders issued by ATC, to predict an aircraft ETA or to predict an aircraft route in the Terminal Maneuvering Area (TMA). Trajectory clustering methods aim at grouping trajectories with similar patterns together (and identifying outliers).

There are four basic clustering frameworks: partitioning, hierarchical, density-based and grid-based [31].

Given a set of n objects, a partitioning method constructs K partitions of the data (K being an input required) and each object is assigned to exactly one cluster (this requirement may be relaxed). Usually,

an initial partitioning is iteratively relocated based on the distance between objects until convergence is achieved - when objects in the same cluster are sufficiently close to each other and far from objects in other clusters for the partition to stay the same between consecutive iterations. The two most popular algorithms are K-means and K-medoids.

K-means is a centroid-based technique. Each cluster C_i is represented by its center c_i and its quality is determined by the within-cluster variation, defined by the sum of squared errors between all objects in C_i and the centroid c_i. The objective function to be minimized is defined by (2.15).

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \text{dist}\,(\boldsymbol{p}, \boldsymbol{c_i})^2 \tag{2.15}$$

Minimizing this objective function makes the clusters as compact and separate as possible. Figure 2.15 illustrates the K-means algorithm applied in a simple problem with $K = 3$.

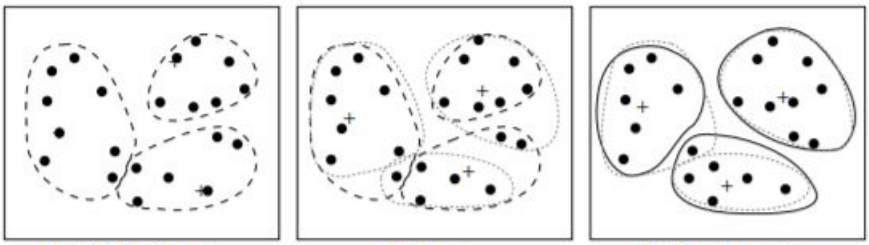


Figure 2.15: Clustering of a set of objects using the K-means method; for (b) update cluster centers and reassign objects accordingly (the mean of each cluster is marked by a +). From [31].

One of the problems of the K-means method lies in its sensitivity to outliers. This is because each object is forced to be assigned to a cluster at all times and this may significantly distort the centroid of the cluster it is assigned to. Attempting to solve this problem, in the K-medoids method each cluster C_i is represented by a representative object o_i instead, and the absolute error defined by (2.16) is minimized.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \text{dist}\,(\boldsymbol{p}, \boldsymbol{o_i}) \tag{2.16}$$

Partitioning methods are fit to cluster spherical-shaped data in small to medium-size datasets. When dealing with large-size datasets, memory use is a problem, since all data needs to be loaded to the memory. However, variants have been proposed to overcome this problem. Another drawback is the fact that the number of clusters K needs to be set in advance, since this number is usually unknown in practice. However, a heuristic called the Elbow Method is usually applied to determine the value of K and overcome this difficulty. In the Elbow Method, K-means is run for increasing values of K starting at $K = 1$ and the sum of squared errors (SSE) with respect to the cluster centres is computed for each value of K. SSE decreases as K increases and its plot against K usually resembles an arm. If such is the case, the Elbow Method selects for K the first value such that increasing the number of clusters no longer decreases SSE significantly. This point is the elbow of the arm in the plot of SSE against K. The major drawback of partitioning methods is that they do not apply well to data of irregular shape, only

to spherical-shaped data. At last, in some problems a different initialization of the algorithm on a same dataset may yield a different clustering result.

Hierarchical methods are interesting to partition the data into groups at different levels, as in a hierarchy. There are two types of approach: agglomerative (a bottom-up strategy) and divisive (a top-down strategy). In agglomerative methods, individual objects start as clusters and are iteratively merged to form larger clusters. In divisive methods all objects start belonging to a single cluster and are iteratively divided into smaller clusters. Figure 2.16 illustrates the concept of agglomerative and divisive methods. The selection of merges or splits is critical to the quality of the clusters obtained. There are several hierarchical clustering methods and distance between cluster measures used when deciding which clusters to merge or split.

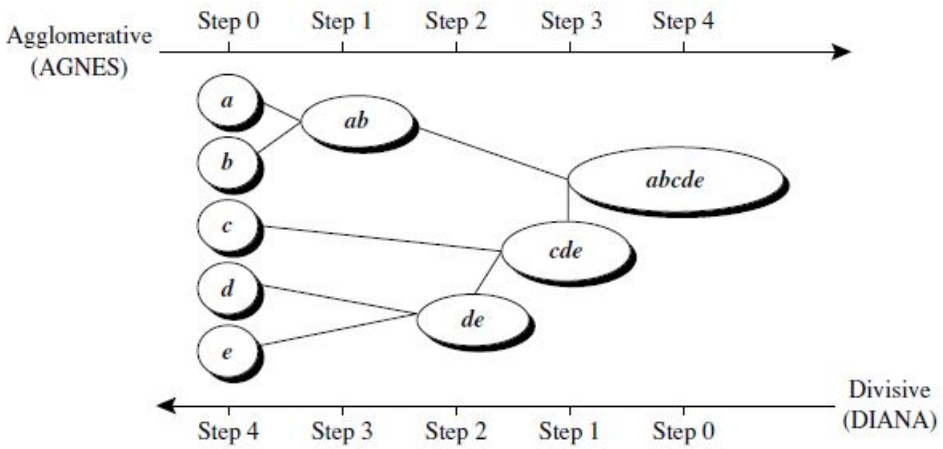


Figure 2.16: Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$. From [31].

Density-based methods are based on the notion of probability density, rather than distance between objects. As a result, these methods are able to identify clusters of data of any irregular shape, as well as identify and filter out outliers. Clusters can grow as long as the density is above a certain threshold and outliers are data objects in regions of very low density. Figure 2.17 illustrates an S-shaped cluster, an oval cluster and some outliers that would successfully be identified by a density-based method but not by partitioning methods, which are designed to identify spherical-shaped data.

Figure 2.17: Clusters of arbitrary shape. From [31].

The most popular density-based algorithm is DBSCAN [19], which stands for Density-Based Spatial

Clustering of Applications with Noise. In this algorithm, objects can be assigned different classifications. A core object is one that has an associated density above a certain threshold, the density of an object neighborhood being measured through the number of objects in its ϵ-neighborhood. Core objects are the basis of clusters, and have more than *MinObjs* objects in their ϵ-neighborhood. Directly density-reachable objects are less than an ϵ distance away from a core object, while density-reachable objects are at the end of chains of objects $o_1, ..., o_n$ such that o_{i+1} is directly density-reachable from o_i. Finally, two objects are density-connected if they are both density-reachable from a same core object. The clusters built by the DBSCAN algorithm are composed of density-connected objects. Once the clusters have been obtained, outliers are the objects that do not belong to any of them. In a probabilistic framework, some approaches allow for the estimation of the optimal K from data [32].

Grid-based clustering methods partition the space into cells, regardless of the distribution of the input objects. The clustering operations are then performed on such a grid structure. For more details on these methods please refer to [31].

The clustering techniques described were originally introduced to cluster points, not trajectories. In fact, the clustering of aircraft trajectories is a more difficult problem, because other than choosing which algorithm is most suitable for the problem, one also needs to choose or define a metric for the similarity or dissimilarity between trajectories [33].

Euclidean distance is simple and intuitive because it is parameter-free. However, the two p-dimensional trajectories L_i and L_j being compared must be composed by a same number of segments n and these must have corresponding times, which is hard to ensure, especially since the trajectories do not necessarily have the same temporal nor distance lengths. The Euclidean distance is computed by (2.17).

$$D_E\left(L_i, L_j\right) = \frac{1}{n}\sum_{k=1}^{n}\sqrt{\sum_{m=1}^{p}\left(a_k^m - b_k^m\right)^2} \tag{2.17}$$

Since the time complexity is linear, large datasets can be easily handled. However, noise may have a strong impact on the distance obtained. Variations include taking the average, maximum or minimum values. In another variation, a data dimensionality reduction is first performed through a Principal Components Analysis (PCA) and the Euclidean distance formula is then applied to the reduced trajectories. Other more interesting distances allow to consider shape or the time dimension.

The Hausdorff distance is used to measure the maximum mismatch between two trajectories, considering that two trajectories are close if every sampling point of either trajectory is close to some sampling points of the other one. It is defined by (2.18).

$$\begin{aligned} D_H\left(L_i, L_j\right) &= \max\left(h\left(L_i, L_j\right), h\left(L_j, L_i\right)\right) \\ \text{where } h\left(L_i, L_j\right) &= \max_{a \in L_i}\left(\min_{b \in L_j}(dist(a, b))\right) \end{aligned} \tag{2.18}$$

The Hausdorff distance is therefore defined by the point in L_i that is farther to L_j and the point in L_j that is farther to L_i. For this reason, it is very sensitive to noise data. It seems appropriate to measure the maximum mismatch between two trajectories, but not to give an overall idea of how similar or dissimilar

the two trajectories are. It is also computationally more expensive than the Euclidean distance, having a computation complexity of O($m * n$), m and n being the sizes of the trajectories L_i and L_j, which can be different.

Dynamic Time Warping (DTW) finds the optimal alignment between two trajectories, as opposed to using the predefined correspondences that the Euclidean distance uses, as represented in Figure 2.18.

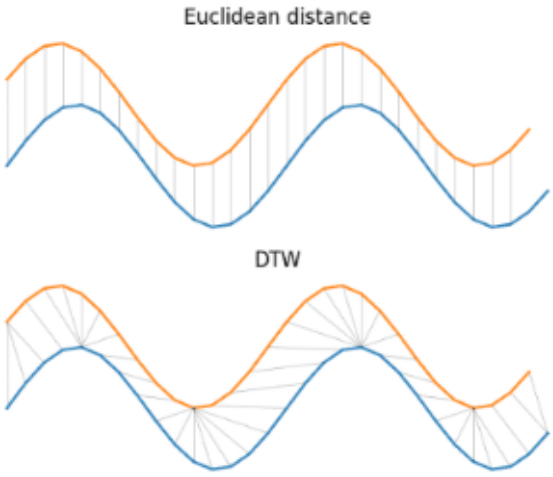


Figure 2.18: Trajectories alignment for computing the Euclidean distance and DTW. From [34].

The goal is to find a warping path $\{(p_1, q_1), (p_2, q_2), ..., (p_k, q_k)\}$ such that $\sum_{ind=1}^{k} dist\left(L_i\left(p_{ind}\right), L_j\left(q_{ind}\right)\right)$ is minimized, respecting the boundary conditions $(p_1, q_1) = (1, 1)$ and $(p_k, q_k) = (m, n)$ and guaranteeing that the mapping is monotonically non-decreasing. Additionally, each sample from each trajectory must be mapped to a sample of the other trajectory. An advantage of the metric is that the trajectories lengths m and n can be different, so different sampling rates can be handled. The forward computation of the DTW distance between L_i and L_j is performed recursively by (2.19).

$$D_D\left(L_i, L_j\right) = \text{ dist } \left(L_i(0), L_j(0)\right) + \min \begin{cases} D_D\left(\text{Rest}\left(L_u\right), \text{Rest}\left(L_j\right)\right) \\ D_D\left(\text{Rest}\left(L_i\right), L_j\right) \\ D_D\left(L_d, \text{Rest}\left(L_j\right)\right) \end{cases} \tag{2.19}$$

where $\text{Rest}\left(L_i\right)$ and $\text{Rest}\left(L_j\right)$ are the remaining trajectories L_i and L_j after the deletion of the respective first sampling points. For the computation, an mxn matrix is constructed. The computational complexity is therefore $O(mn)$. The computation could also be performed in backwards manner, starting from the distance between the last points of each trajectory instead of the first ones. The main inconvenience is the sensitivity to noise and need to have trajectory data points that are quite continuous.

On a different paradigm, there is the Longest Common Subsequence (LCS). It computes the longest common sub-sequence in two trajectories, rather than a distance. It allows for some deviation in the data, though tuning the parameters involved might be difficult. It is generally solved recursively and seems interesting to cluster trajectories built as sequences of way-points. The complexity is $O(mn)$.

Several combinations of the clustering algorithms and distance metrics presented have already been successfully applied to problems involving aircraft trajectory clustering in the literature.

In [35], a framework for predicting aircraft arrival times is presented, focusing on vectoring orders issued by ATC. Major trajectory patterns were successfully identified through agglomerative hierarchical clustering, applied using the Ward minimum variance criterion that aims at minimizing the total within-cluster variance after merging operations. The metric chosen to compute the distance between trajectories was DTW, in view of the optimal trajectories alignment it provides.

Two methods for identifying aircraft trajectory clusters are presented in [36]. The first method is way-point-based. Firstly, it identifies turning points, where there is a change in heading. Secondly, it clusters these points using the DBSCAN algorithm to obtain a finite number of main way-points and represent the trajectories through sequences of these. DBSCAN was chosen thanks to its ability to filter noise when the distribution of turning points is large, as opposed to K-means that would require that every point is assigned to a cluster. Lastly, LCS is used to cluster the trajectories. The method shows good empirical results. However, it only keeps trajectories going over way-points. A parallel trajectory might be seen as an outlier, while a trajectory containing a large rerouting period will belong to a cluster if it contains its way-points. In the second method presented in [36], trajectories are resampled to obtain time series of equal length. The data is then augmented. Adding the distance to the airport, the distance to a corner point, an angular position (to break symmetries) and the heading to the aircraft position allows to improve the results of the clustering. A PCA is then run to reduce data dimensionality, and finally the clusters are obtained using DBSCAN, chosen thanks to its ability to find irregular shaped clusters and identify outliers.

In [28], a method for learning a probabilistic generative model of aircraft motion in terminal airspace from position measurements is presented. Trajectories are first time-aligned through a shifting, so that the first measurement in each trajectory is always at t = 0. They are then interpolated, smoothed, and extrapolated by solving the least-squares optimization problem presented in Section 2.4. Clusters are obtained using Euclidean distance and the K-means++ algorithm. Guiding these decisions by the authors was the fact that DTW and DBSCAN would not scale well to the large dataset of the problem, as well as the fact that this choice suited well the Gaussian Mixture Model (GMM) eventually fit. As just said, at last, a GMM is constructed based on the intra-cluster covariance matrices of each cluster, allowing accurate inference and realistic generation of trajectories. Although the inference of trajectories is not an objective in this thesis, the ability to easily generate plausible 3D trajectory samples could be particularly useful in the context of the generation of test trajectories for the SUT. GMMs are very easy to sample from thanks to the fact that conditioning a Gaussian distribution on measurements gives another Gaussian distribution. Additionally, the probabilistic generative model may be used for anomaly detection tasks as well. In fact, Bayesian inference is applied to obtain the probability distribution over clusters and time index in that cluster's distribution, rather than a rigid cluster attribution. The model presented is compact, trains quickly, and is efficient, performing well on real, noisy data. It has shown good performance applied on data limited to the position measurements that are less than 5 NM in the East or North dimensions and less than 3000 feet in the Up dimension from the airport.

The approach followed in [28] is successfully extended in [29] and makes trajectory predictions and computes landing times with good accuracy when aircraft are still 45 minutes from the airport. Moreover,

it introduces two new features indicated by domain knowledge, which improve the accuracy of the models obtained. The first feature introduced concerns the density of aircraft. It makes sense that density measures might have an important effect, since two aircraft in a same position might follow very different trajectories depending on the congestion and disposition of the airspace. The second feature introduced concerns wind conditions at the airport, which is fundamental for the prediction of the runway in use.

Chapter 3

System Under Test

In this chapter, the SUT is presented. Section 3.1 presents the system's purpose, the needs it was designed to answer. Section 3.2 presents the black box system's interfaces. In Sections 3.3 and 3.4 respectively the construction of realistic restricted areas (such as military zones) and critical weather obstacles to feed the system is presented. Section 3.5 presents all the requirements that must be met by the solutions output by the system. Finally, Section 3.6 presents the performance metrics and cost functions designed to evaluate the ability of the system to produce solutions in real-time, as well as their quality. These guide the search for falsifying, challenging or dimensioning inputs.

3.1 System Purpose

The SUT computes automatically and in real-time an emergency diversion solution, that could either serve the purpose of assisting pilots or be automatically engaged by an autonomous aircraft, depending on the use case. For providing such a solution, the system is composed of two modules. The first one selects the most suitable diversion target from a list of diversion options. The second one computes a safe trajectory starting at the aircraft and ending at the runway chosen by the first module. A safe trajectory avoids the terrain, restricted areas and critical weather obstacles susceptible of damaging the aircraft.

In order to ensure terrain avoidance, the SUT uses a simplified terrain representation obtained from a raster with elevation data such as NASA's Shuttle Radar Topography Mission (SRTM) database [37]. Such a representation must fully enclose the terrain, so that any trajectory judged safe by the SUT is indeed safe from the real terrain point of view. However, the representation cannot be too simplistic either, so that common aircraft positions are not considered as unsafe and the SUT can launch computations starting from them. Although the inner terrain representation structure is unknown, an interface allows to communicate with the software that determines the safety (or not) of a point or part of a trajectory. Therefore, a data-driven approach using FlightRadar24 data is designed and implemented in Section 5 to evaluate whether the system's terrain representation meets the requirements and provide important feedback.

3.2 System Interfaces

As inputs, the SUT requires the aircraft position (defined by its latitude, longitude and altitude), speed (horizontal and vertical) and heading, the restricted areas and critical weather obstacles to be avoided (defined by shapes and their corresponding positions and orientations) and a list of diversion options. The system outputs the selected diversion option and the trajectory produced to reach it. Figure 3.1 presents a simplified diagram of the interfaces that were just presented.

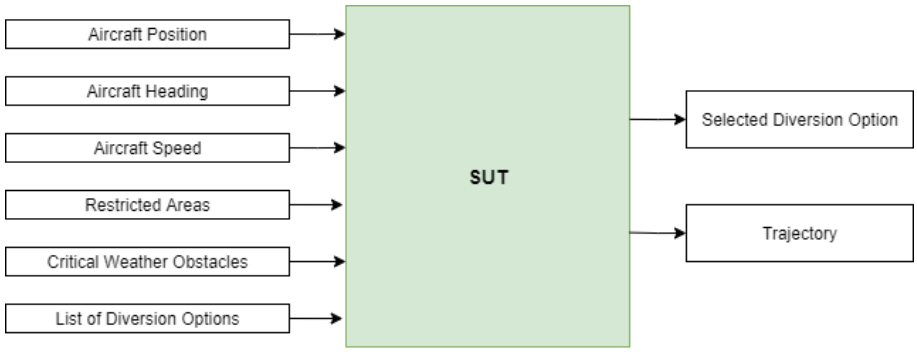


Figure 3.1: Simplified diagram representing the SUT interfaces.

3.3 Construction of Restricted Areas

In order to ensure test scenarios as realistic as possible, real restricted areas were recovered from Aeronautical Information Procedures (AIPs). An AIP [38] contains information (such as procedures or regulations) that is relevant to the operation of aircraft in a particular country permanently or for a long period of time. It is published by a State or with its authority. Documents are usually divided into three parts - GEN (general), ENR (en route) and AD (aerodromes). In the context of this thesis, en-route charts are interesting, more specifically cruise charts, which are divided between lower airspace and upper airspace. Figure 3.2 presents part of an en-route lower airspace chart for France updated on the 22nd of April 2021. This chart was published by the SIA [39].

There are three different types of restricted areas: Prohibited areas, Danger areas and Restricted areas. As the name indicates, prohibited areas are areas within which the flight of aircraft is prohibited. Danger areas are areas within which activities dangerous to the flight of aircraft may exist at specified times and they are usually operated by military authorities. They enclose areas where hazardous operations such as military exercises involving live firing, parachute dropping or others may take place. Although the authorities monitor the zone and cease operations if an unauthorised penetration occurs, the aircraft should avoid danger zones. Restricted areas are areas where the flight of aircraft is restricted in accordance with specific conditions. The SUT should produce trajectories that avoid all of these types of zones. To test the SUT on realistic scenarios, real declared zones where recovered from AIPs. Figure 3.3 presents them. Although AIPs are usually revised every AIRAC cycle of 28 days, a single version is enough to be representative during the first validation stages. One boolean variable associated to each zone allows to change its status between open and closed.

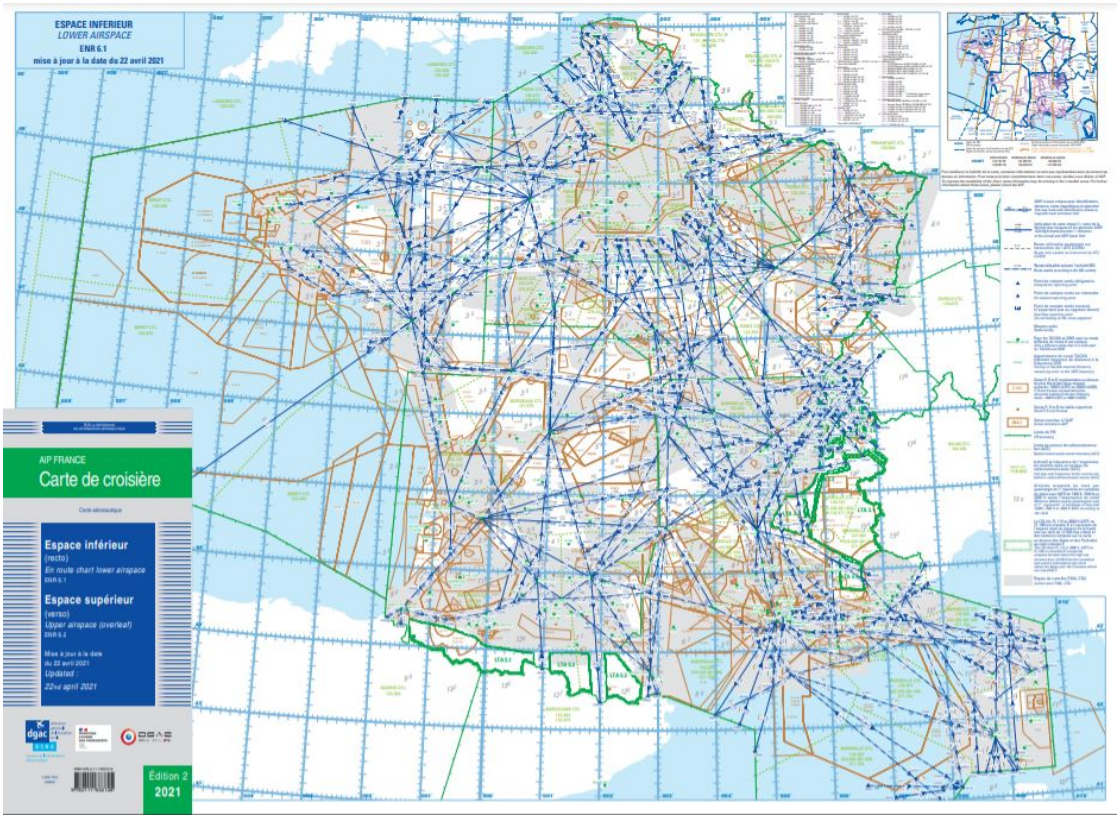


Figure 3.2: En-route lower airspace chart example from France. Version from the 22nd of April 2021.

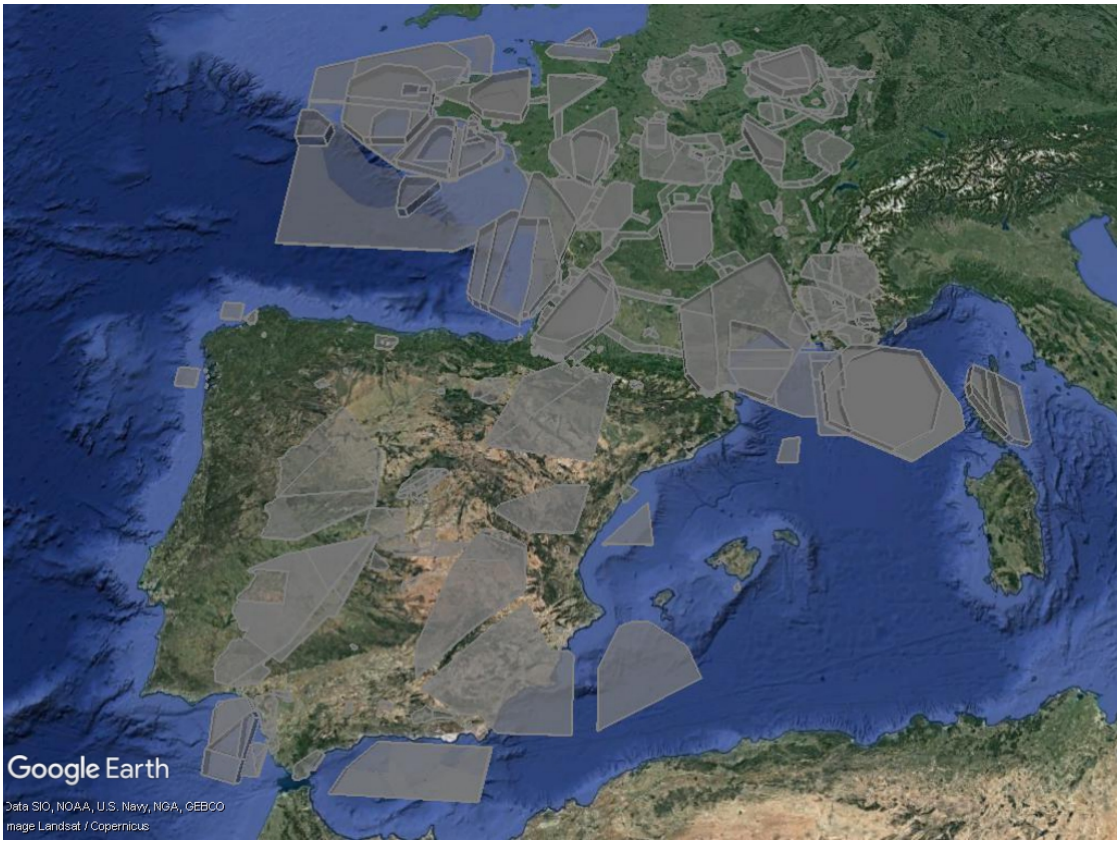


Figure 3.3: Restricted areas retrieved and formatted to feed the SUT.

3.4 Construction of Weather Obstacles

The ability to avoid hazards associated with critical weather such as Cumulonimbus clouds is fundamental to ensure a safe autonomous diversion. A Cumulonimbus is a heavy and dense cloud of considerable vertical extent in the form of a mountain or huge tower, often associated with heavy precipitation, lightning and thunder [40]. Aircraft flying close to or into a Cumulonimbus can reasonably expect to encounter some or all of the following effects which can cause severe damage to aircraft: severe turbulence, icing, electrical disturbances (such as lightning) or heavy precipitation (including hail).

Generating realistic weather obstacles is part of the creation of representative environments for the evaluation of the SUT. An easy way to reproduce realistic weather obstacle shapes is to retrieve historical data. During flight planning, the crew can plan a route to avoid hazardous weather thanks to the weather previsions available, among others, through:

- TAF - Terminal Aerodrome Forecast. It consists of a concise statement of the expected meteorological conditions at an aerodrome for a specified period.
- Significant Weather Chart (SIGWX) - Presents the most important meteorological phenomena.
- WINTEM - Chart with wind forecast.
- SIGMET - Significant Meteorological Information. SIGMET charts are issued by a meteorological watch office and describe in plain language the expected occurrence of specified en-route weather phenomena which may affect the safety of aircraft operations.

Once airborne, the weather radar can be used for in-flight avoidance. As Figure 3.4 illustrates, weather detection by the radar is based on the reflectivity of water droplets and the weather echo appears on the Navigation Display (ND) with a color scale from red (high reflectivity) to green (low reflectivity) [41]. For radars with a turbulence display mode, areas of high turbulence are in magenta.

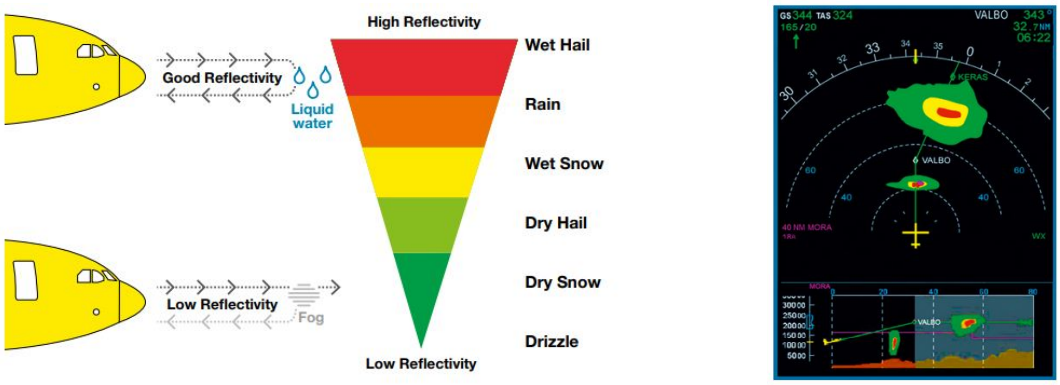


Figure 3.4: Weather radar principle and information display on the ND and VD. From [41].

In order to create representative weather obstacles, one can simply define shapes that enclose the critical zones in the radar image faithfully. Typically, in cruise, the Pilot Monitoring (PM) adjusts the range to 160 NM to plan the long-term weather avoidance strategy, while the Pilot Flying (PF) adjusts the

range to 80 NM to monitor the severity of adverse weather and decide on avoidance tactics [41]. From a dataset of radar images, a set of representative weather obstacles can be created. Then, the SUT can be fed by selecting some shapes from this set and playing with their positioning and orientation.

In case the aircraft can be informed of all the relevant weather obstacles around it, and not only those within its radar reach, the process of creation of representative shapes from weather images covering large areas can be followed. These images can be obtained through satellite for instance. Figure 3.5 illustrates a weather image and the corresponding representative enclosing shapes constructed. Scenarios can be generated through a selection, re-positioning and reorientation of these shapes.

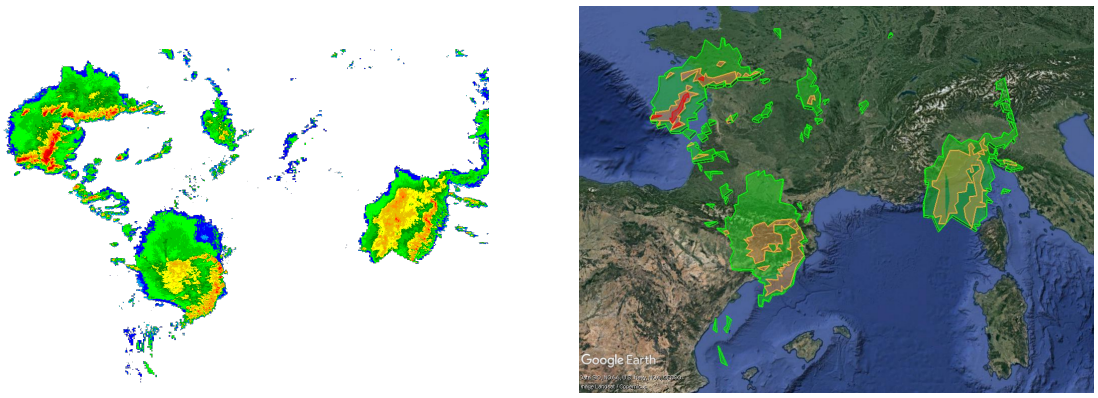


Figure 3.5: Example of weather image and corresponding representative enclosing shapes.

3.5 Validation of Solutions

In this section, the criteria to be evaluated when determining if a trajectory produced by the SUT is valid are presented. If one of them is not met, the trajectory is not valid and the inputs are falsifying.

The trajectory produced must be safe in light of the terrain. To ensure that such is the case, the terrain profile under the lateral trajectory is retrieved from the SRTM database. Such a terrain profile is represented in red in Figure 3.6 for one of the trajectories produced by the SUT. The trajectory is considered safe with respect to the terrain, since its vertical profile is always above the terrain elevation profile. Likewise, all the restricted areas and weather obstacles fed to the SUT that are placed along the trajectory produced are retrieved and the trajectory is valid because its vertical profile avoids these.

The validation of the trajectory with respect to terrain and obstacles avoidance is performed until the Final Approach Fix (FAF) only. Then, for the final approach, information regarding the runway chosen and the ILS approach procedure is retrieved and it is only checked that the glide slope is coherent with the ILS procedure and that the trajectory finishes on the runway threshold.

3.6 Performance Metrics and Cost Functions

The previous section addressed the validity check of solutions. This one presents some key indicators and cost functions allowing to evaluate the pertinence of the solutions.

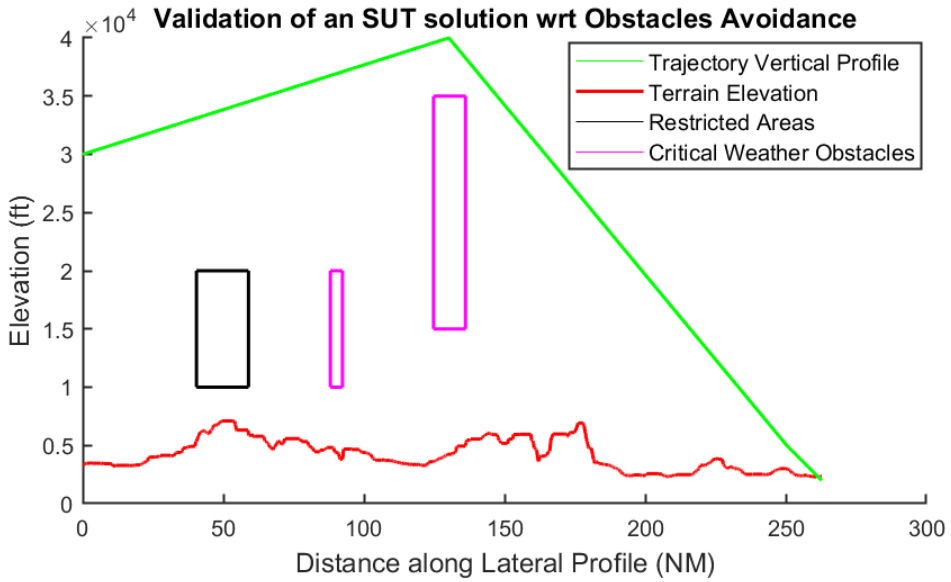


Figure 3.6: Validation of the safety of a trajectory produced by the SUT.

A diversion option selected is considered to be good if the trajectory generated by the SUT to reach it is shorter than the trajectory that the SUT would generate to reach any of the other diversion options in the list. Let us consider a list containing n diversion options. Let us also consider that the $n-1$ options not selected by the SUT for a given set of inputs are sorted by increasing order of ground distance between the aircraft and their runway threshold (so in index 1 is the closest option among those not selected, and so on). Let also l_{SUT} represent the ground length of the trajectory produced by the SUT to reach the diversion option selected by it and l_i the ground length of the trajectory produced by the SUT to reach diversion option i (it can be easily obtained by feeding the SUT with a list containing only option i). Cost function f_i is therefore defined by (3.1) to evaluate the pertinence of the diversion selection.

$$f_i = \frac{l_{SUT}}{l_i} \tag{3.1}$$

The SUT diversion selection is not suitable in case $f_i > 1$ for any $i \in 1, ..., n-1$. This would mean that choosing diversion option i would provide a shorter trajectory. In order not to waste computational resources calling the SUT for each diversion target, only f_1 is computed, since option 1 is the alternative closest to the aircraft and therefore the one with more probability of being better than the one selected by the SUT. The SUT does not need to generate a trajectory to reach option 1 neither when the direct distance between the aircraft and this option is already greater than l_{SUT}.

Let us now turn towards the optimality of the trajectory produced. A search for sub-optimal trajectories could be guided by a maximization of l_{SUT}. However, this would favor scenarios where the aircraft is far from the airport and would not identify sub-optimal trajectories when the aircraft is closer to it. Let d_{SUT} be the ground distance between the aircraft and the runway selected by the SUT. The search for sub-optimal trajectories could be guided by f_{SUT} defined by (3.2).

$$f_{SUT} = \frac{l_{SUT}}{d_{SUT}} \tag{3.2}$$

Guiding a search to maximize f_{SUT} seems a good first strategy to find scenarios where a shorter trajectory than the one produced by the SUT could exist. However, f_{SUT} might be large for optimal trajectories if obstacles need to be avoided or the aircraft is close to the runway at high altitude and needs to dissipate substantial energy. In future work, it could be interesting to design a cost function to guide the search based on energetic considerations.

Ideally, if another algorithm could compute optimal solutions (not necessarily in real-time) with ground length l_{opt}, cost function g defined by (3.3) could drive the search for non-optimal trajectories.

$$g = \frac{l_{SUT}}{l_{opt}} \tag{3.3}$$

In case different alternative prototypes were available, applying the same principle would be interesting to drive a search to find scenarios for which a prototype a is better than a prototype b (or the opposite). Let l_a and l_b represent the ground lengths of the trajectories produced by prototypes a and b respectively. The cost function, h, just proposed could be defined by (3.4).

$$h = \frac{l_a}{l_b} \tag{3.4}$$

A strategy capable of efficiently searching for inputs maximizing or minimizing cost function h would be a very interesting support to the design and development process. In fact, in case b is an improved version of a, function h can be used to understand in which scenarios the improvements are more clear, while the inverse of h can be used to find scenarios where there is a regression in the SUT performance.

Since the SUT is to be used in real-time, the last term of the cost function that should drive the validation scenarios choice is proportional to the execution time, t. Its worse value is dimensioning. It determines the maximum rate at which the SUT can be called. The principle of cost function h is interesting to support the design and development processes. Between successive prototype versions a and b with execution times t_a and t_b respectively, cost function t_h can drive the search for scenarios where b is faster than a if defined by (3.5).

$$t_h = \frac{t_a}{t_b} \tag{3.5}$$

Likewise, the inverse of cost function t_h can be used to guide the search for scenarios where a is faster than b. It is important to remark that unlike the outputs produced by the SUT, which are deterministic and remain constant for a given set of inputs, the execution time can vary between two calls to the SUT using the same inputs. There are several reasons why this happens, such as the fact that different processes might be running in the background or caching of memory might have changed between runs. For this reason, the system should ideally be called several times to obtain a confidence average of the execution time. The number of times should be determined to ensure a low variance of the average estimator. However, in reality the dimensioning value is the maximum execution time. The SUT can be called only once per scenario at first. Then, for the dimensioning scenarios it is re-run to ensure good confidence in the values obtained.

Chapter 4

Exploratory Data Analysis and Data Preprocessing

This chapter introduces and explores the dataset used to support the data-driven evaluation strategies later proposed in this document. Apart from the dataset itself, the environment allowing to access and transform it, as well as some necessary preprocessings are presented. The Lisbon-Munich (LIS-MUC) route was used as a case study to understand the available data, identify the required data preprocessings, understand the value that can be taken out of the data in the context of this thesis and evaluate the interest of the strategies proposed in this document.

4.1 Data Source - Skywise

Skywise is the leading data platform for the aviation industry [42]. It is powered by Palantir Foundry, a platform that allows receiving valuable data from every player in the aviation value chain, storing, transforming, and accessing it, regardless of size or format. Hence allowing the creation of operational and business value across sector-wide challenges (see Figure 4.1). Built on leading commercial cloud services that can handle long term increases in data size and user count, Skywise can scale both infrastructure and capabilities.

The Foundry ecosystem offers a wide variety of data-focused applications to build pipelines (data ingestion, transformation and connection), analyse data (discovering insights from it), operationalize data (showcasing insights and creating operational workflows) and more. In this project, Code Workbook, an application that allows to apply transforms to datasets in order to generate new datasets, was used. Each Workbook is associated with a configurable environment that consists of a set of packages installed on an Apache Spark [43] module. Spark is a unified analytics engine for large-scale data processing across a distributed network of servers (see Figure 4.2). Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object, the driver program. An overview of how Spark runs on clusters is given in [44].

In Code Workbook, Spark applications can be written in Python and SQL. Python was chosen due

Figure 4.1: Skywise allows to create value from data across sector-wide challenges. From [42].

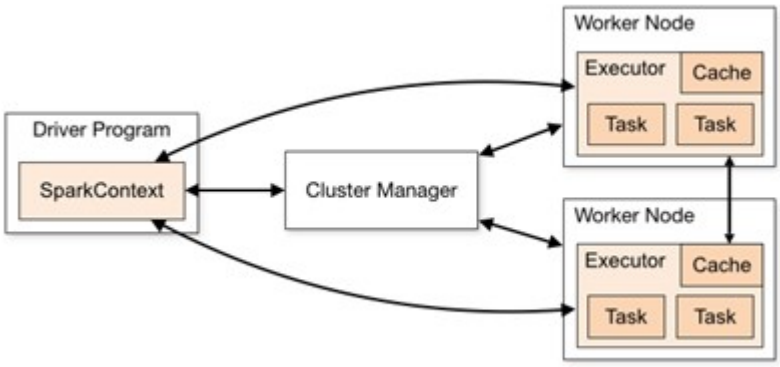


Figure 4.2: Spark cluster architecture. From [44].

to more experience using it. PySpark is the wrapper language that allows to interface with the Apache Spark backend to quickly process data. Even if all happens under the hood, it is important to know what is happening behind the scenes when writing PySpark to understand performances and what can be done. PySpark DataFrames are immutable (they cannot be changed, only transformed, resulting in two DataFrames) and lazily evaluated (a series of transformation tasks are evaluated as a single action, performed when a build is triggered). The underlying data structure of a DataFrame is Resilient Distributed Datasets (RDD): by partitioning the DataFrame into multiple non-intersecting subsets, transformations can be evaluated in parallel on multiple computers (nodes) in a cluster (network) of computers.

4.2 Dataset - FlightRadar24

Automatic Dependent Surveillance-Broadcast (ADS-B) is a surveillance technology in which an aircraft periodically broadcasts information about itself, such as its identification, position, altitude and velocity, through an on-board transmitter. The information provided by ADS-B to ATC is often more ac-

curate than the information available with current radar-based systems. For this and other reasons, the FAA is actually transitioning away from radar and towards ADS-B technology. This provides us a good degree of confidence in the FlightRadar24 datasets available. FlightRadar24 is a data source that compiles ADS-B data from the majority of aircraft worldwide, including nearly all commercial flights. Its data is available through the Skywise platform. The data of interest (flight recordings) comes in two tables. The first table contains flight metadata generated by FlightRadar24, including the following parameters that were considered to be interesting in the context of this document:

- Flight_id - internal ID number generated by FlightRadar24 to identify specific flights.
- Equipment - aircraft model used for the flight.
- Departure_airport_code - IATA code of the departure airport.
- Scheduled_airport_code - IATA code of the scheduled arrival airport
- Arrival_airport_code - IATA code of the arrival airport.
- Adsb_start_flight_phase - flight phase during which ADS-B transmission began. Phase can be "stationary" ($speed = 0$, $altitude = 0$), "ground_slow" ($speed < 15kt$, $altitude = 0$), "ground_fast" ($speed > 15kt$, $altitude = 0$), "ascent" ($0 < altitude < 25000$) or "cruise" ($altitude >= 25000$).
- Adsb_end_flight_phase - flight phase during which ADS-B transmission ended. Phase can be "stationary" ($speed = 0$, $altitude = 0$), "ground_slow" ($speed < 15kt$, $altitude = 0$), "ground_fast" ($speed > 15kt$, $altitude = 0$), "descent" ($0 < altitude < 25000$) or "cruise" ($altitude >= 25000$).

The second table contains the actual ADS-B time-series information for each flight, including the following parameters considered to be interesting in the context of this document:

- Flight_id - internal ID number generated by FlightRadar24 to identify specific flights.
- Snapshot_id - the timestamp of the ADS-B reading, as a Unix Timestamp (the number of seconds elapsed since January 1, 1970).
- Altitude - calibrated altitude reported from the aircraft in ft. It is a pressure-derived value.
- Heading - the track angle of the aircraft relative to true North, in degrees.
- Latitude - the GPS latitude coordinate of the aircraft in decimal notation.
- Longitude - the GPS longitude coordinate of the aircraft in decimal notation.
- Speed - the ground speed of the aircraft in kt.
- Vertical speed - the vertical speed of the aircraft in ft/s. Computed for each datapoint by differentiating the altitude with respect to time.

The datasets used contain information from late February 2014 to the end of December 2018. During this period, over 200 million flights are available in the datasets.

4.3 Diversions in the Dataset

Since the IATA codes of the departure, scheduled and arrival airports are available, we can easily identify flights where a diversion occurred. These are simply flights where the scheduled airport is different from the actual arrival airport. The diversions that occurred during LIS-MUC or MUC-LIS flights were obtained and plotted on Google Earth for analysis. They are presented in Figure 4.3.

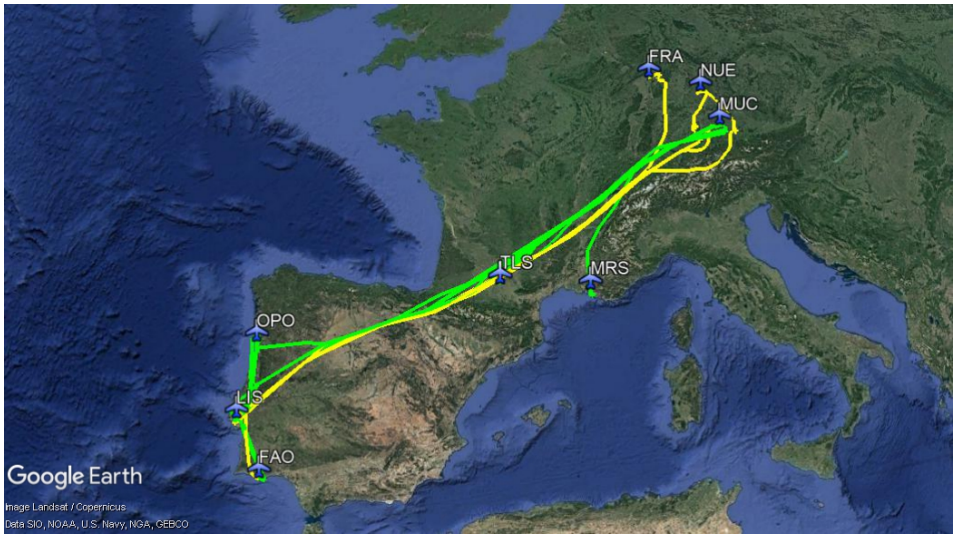


Figure 4.3: Diversions that occurred during LIS-MUC (in yellow) or MUC-LIS (in green) flights.

There were clearly three different moments when flights diverted. The first one was right after take-off. Even though it is not easy to see in Figure 4.3 due to the zoom out, there is a case where the aircraft took-off from Lisbon with Munich as a destination but immediately landed back at Lisbon. A probable cause is the aircraft having problems during take-off. There is also a diversion to Faro after take-off. The cause could be the same, plus a difficulty in finding a slot to land back at Lisbon. Diversions also occurred two times while in cruise during MUC-LIS flights: a diversion to Toulouse and another one to Marseille. In this case, both a problem with the aircraft or a passenger feeling sick are plausible causes. Finally, there were diversions performed when the aircraft was already close to the scheduled airport: to Frankfurt and Nuremberg when going to Munich and to Porto and Faro when going to Lisbon. Severe meteorological conditions at the scheduled arrival airport could be the cause. Figure 4.4 shows two cases where the aircraft performed holding patterns while waiting to determine if landing at Lisbon was possible (this is typical when waiting for a slot or because the weather is critical) before diverting to Porto, as well as a case where the aircraft even passed above Lisbon before diverting to Faro.

Recovering information from ATIS, METAR, NOTAM, news or airlines information connected to the diversions under analysis to understand their root causes would be a very interesting exercise. However, for the moment, it is enough to acknowledge the important fact that the dataset tells us which airports were chosen by the pilots and airlines OCC when a diversion was needed. Therefore, these diversion options should be introduced in the list of diversion options to be fed to the SUT.

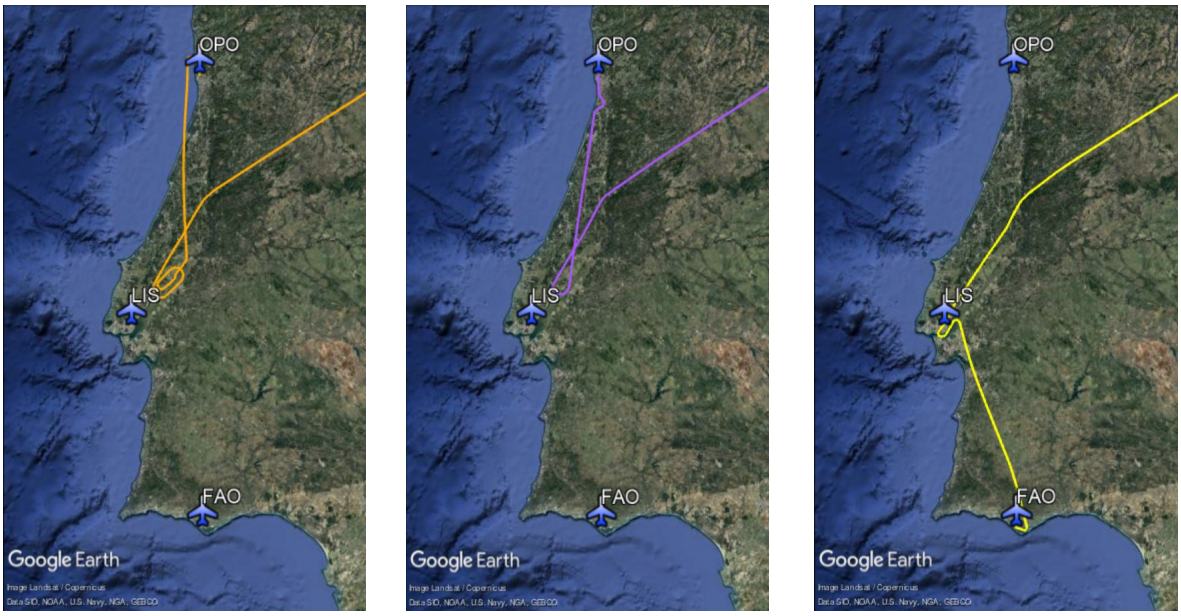


Figure 4.4: Zoom on flights where the aircraft came close to the scheduled airport and eventually diverted, probably due to critical weather around it, runway closure or lack of slot.

4.4 Data of Interest

Both for clustering purposes and to allow testing the SUT throughout whole flights, the trajectories used should start before or during take-off (so Adsb_start_flight_phase as ”stationary”, ”ground_slow” or ”ground_fast”) and finish during or after landing (so Adsb_end_flight_phase as ”ground_fast”, ”ground_slow” or ”stationary”). The distribution of Adsb_start_flight_phase and Adsb_end_flight_phase for the LIS-MUC and MUC-LIS flights available is presented in Figure 4.5.

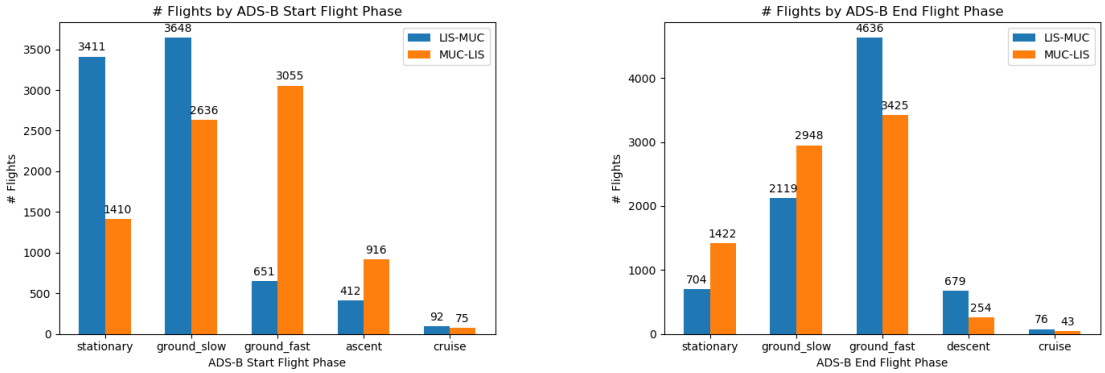


Figure 4.5: Distribution of start and end flight phases for LIS-MUC and MUC-LIS flights available.

Since the dataset is large and the majority of flights are complete, flights with an Adsb_start_flight_phase of ”ascent” or ”cruise” or an Adsb_end_flight_phase of ”cruise” or ”descent” were filtered out. After this, more than 7000 flights are still available, which is more than enough to understand route patterns through a trajectory clustering algorithm. Some trajectories do not start or finish exactly on a runway, as illustrated in Figure 5.2 in Section 5.2. However, they are a minority and they were filtered out.

4.5 Trajectory Reconstruction

As presented in Section 2.4, the problem of trajectory reconstruction from historical recordings can be formulated as an unconstrained optimization problem defined by (4.1).

$$\underset{P}{\text{minimize}} \|AP - \widehat{P}\|_F^2 + \lambda_1 \|D_2 P\|_F^2 + \lambda_2 \|D_3 P\|_F^2 \tag{4.1}$$

The cost function can be written as $f(P) = f_1(P) + f_2(P) + f_3(P)$ with the definitions in (4.2).

$$f_1(P) = \|AP - \widehat{P}\|_F^2 \ , \ f_2(P) = \|D_2 P\|_F^2 \ , \ f_3(P) = \|D_3 P\|_F^2 \, . \tag{4.2}$$

In order to obtain P that minimizes f, we derivate f with respect to P and equal that to zero. The derivative is obtained by summing the derivatives of the three components.

We can obtain ∂f_1 and ∂f_2 using the fact that $\|\mathrm{A}\|_F = \sqrt{\mathrm{Tr}\,(\mathrm{A}\mathrm{A}^{\mathrm{H}})}$, with A^{H} being the conjugate-transpose and Tr the trace of a matrix. Since A is a diagonal matrix and therefore $\mathrm{A}^{\mathrm{H}} = \mathrm{A}^{\mathrm{T}}$, and $\partial(\mathrm{Tr}(\mathbf{X})) = \mathrm{Tr}(\partial\mathbf{X})$, $\partial(\mathbf{X}\mathbf{Y}) = (\partial\mathbf{X})\mathbf{Y} + \mathbf{X}(\partial\mathbf{Y})$ and $\mathrm{Tr}(\mathbf{X}) = \mathrm{Tr}(\mathbf{X}^{\mathrm{T}})$, then

$$\partial f_1(P) = \partial(\|AP - \widehat{P}\|_F^2) = \partial\left(\sqrt{\mathrm{Tr}\left((AP - \widehat{P})(AP - \widehat{P})^{\mathrm{H}}\right)}^2\right) = \mathrm{Tr}\left(\partial\left((AP - \widehat{P})^{\mathrm{T}}(AP - \widehat{P})\right)\right)$$

$$= \mathrm{Tr}\left(\partial\left((AP - \widehat{P})^{\mathrm{T}}\right)(AP - \widehat{P})) + (AP - \widehat{P})^{\mathrm{T}}\partial\left(AP - \widehat{P})\right)\right) = \mathrm{Tr}\left(\partial(P^{\mathrm{T}})A^{\mathrm{T}}(AP - \widehat{P}) + \left(AP - \widehat{P}\right)^{\mathrm{T}} A\partial(P)\right)$$

$$= 2\,\mathrm{Tr}\left((AP - \widehat{P})^{\mathrm{T}} A\partial(P)\right), \text{ and}$$

$$\partial f_2(P) = \partial(\|D_2 P\|_F^2) = \mathrm{Tr}\left(\partial\left((D_2 P)^{\mathrm{T}}(D_2 P)\right)\right) = \mathrm{Tr}\left(\partial(P^{\mathrm{T}})D_2^{\mathrm{T}} D_2 P + P^{\mathrm{T}} D_2^{\mathrm{T}} D_2 \partial(P)\right) = 2\,\mathrm{Tr}\left(P^{\mathrm{T}} D_2^{\mathrm{T}} D_2 \partial(P)\right).$$

f_3 is derivated like f_2. The derivate is equalled to 0 to find the solution of the problem.

$$D(f(P)) = D(f_1(P)) + D(f_2(P)) + D(f_3(P)) = 0 \Leftrightarrow A^{\mathrm{T}}(AP - \widehat{P}) + \lambda_2 D_2^{\mathrm{T}} D_2 \partial(P) + \lambda_3 D_3^{\mathrm{T}} D_3 \partial(P) = 0 \Leftrightarrow$$

$$P = \left[A^T A + \lambda_1 D_2^T D_2 + \lambda_2 D_3^T D_3\right]^{-1} A^T \widehat{P} \tag{4.3}$$

Equation (4.3) allows to obtain the full reconstructed trajectory P from available measurements $\widehat{P}$ as a function of scalar regularization parameters λ_1 and λ_2, which are greater than 0 (otherwise no smoothing) to be tuned. To understand their effect on the reconstructed trajectories, a LIS-MUC trajectory was reconstructed using different values of λ_1 and λ_2. Figure 4.6 presents the reconstructed trajectories,

zooming on the departure from Lisbon. Only the lateral trajectory is displayed for the sake of clarity. However, the vertical profile is much easier to interpolate, as Figure 4.7 later illustrates.

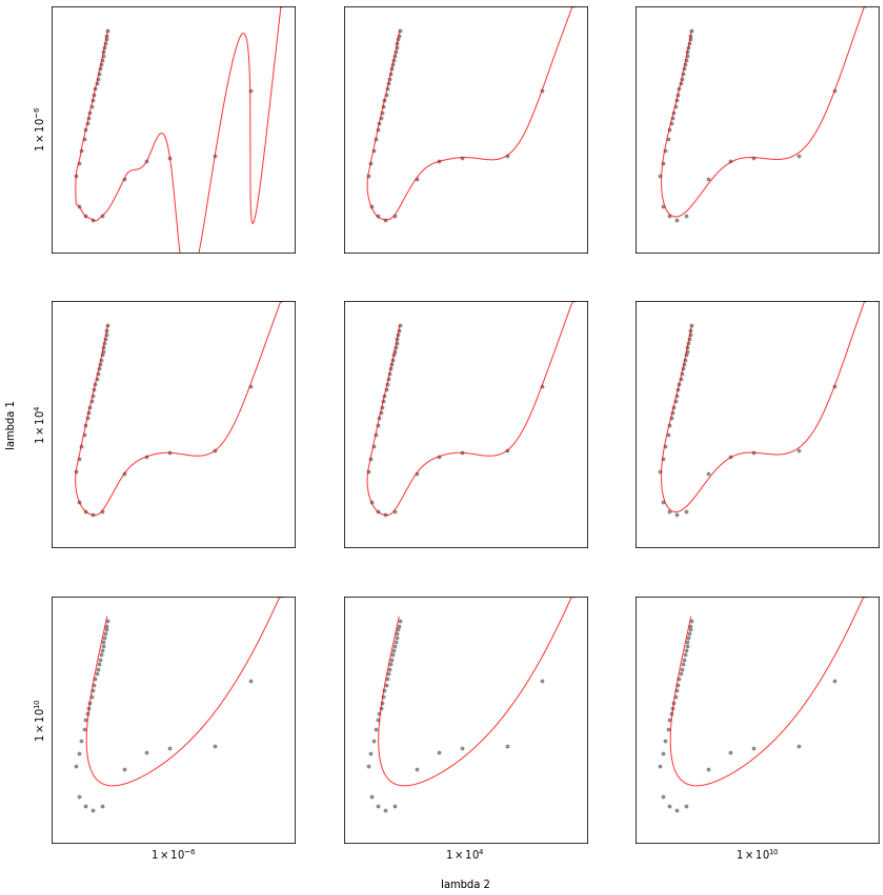


Figure 4.6: Effect of λ_1 and λ_2 on the reconstruction of smooth trajectories from historical measurements.

As Figure 4.6 shows, if the regularization parameters are too small, then rapid and aggressive maneuvers that are unrealistic from an aircraft dynamics point of view can exist to ensure a position error almost nonexistent. On the contrary, if they are too high, aircraft performances are too limited, not allowing the proper reconstruction of the actual aircraft trajectory neither. In this case, the positioning error becomes high while ensuring feasible and very comfortable performances. A good balance therefore needs to be found. An out-of-sample-validation strategy is adopted for this. The principle of this strategy is to randomly hold out measurements from the trajectory, fitting it with varying λ_1 and λ_2 using the measurements not held out, and selecting the parameters that have the lowest loss on held-out measurements. This loss is easily obtained by (4.4).

$$\mathcal{L}(P, \widehat{P}) = \|A_{hom}P - A_{hom}\widehat{P}\|_F^2 \tag{4.4}$$

where A_{hom} is a diagonal matrix where $a_{ii} = 1$ if measurement i was available and held-out and $a_{ii} = 0$ otherwise. For the trajectory presented in Figure 4.6, Table 4.1 presents the loss obtained for varying λ_1 and λ_2 when performing out-of-sample validation with a 60% fraction of the measurements being used for the fitting and the remaining 40% for computing the out-of-sample-validation loss.

Table 4.1: Out-of-sample-validation loss for varying λ_1 and λ_2 values. Minimum losses in bold.

$\lambda_1 \parallel \lambda_2$	$\mathbf{10^{-6}}$	$\mathbf{10^{-2}}$	$\mathbf{10^{2}}$	$\mathbf{10^{5}}$	$\mathbf{10^{8}}$
$\mathbf{10^{-6}}$	295.322	11.860	21.889	19.889	11.296
$\mathbf{10^{-2}}$	**10.667**	**10.667**	11.707	17.348	11.295
$\mathbf{10^{2}}$	10.668	10.668	10.692	11.234	11.364
$\mathbf{10^{5}}$	10.691	10.691	10.691	10.691	11.223
$\mathbf{10^{8}}$	13.706	13.706	13.706	13.706	13.727

The loss values presented in Table 4.1 confirm what had already been exposed: for values of λ_1 and λ_2 too small, the loss is high. There is a range of values for which the loss is at its lowest and it then starts increasing with λ_1 and λ_2. For the range of values with a low loss associated, the out-of-sample-validation process was done again using a finer grid and the best λ_1 and λ_2 were used to reconstruct the trajectory using all measurements available. Figure 4.7 presents the reconstructed trajectory. It is visible that the strategy provides a good reconstruction from both lateral and vertical points of view.

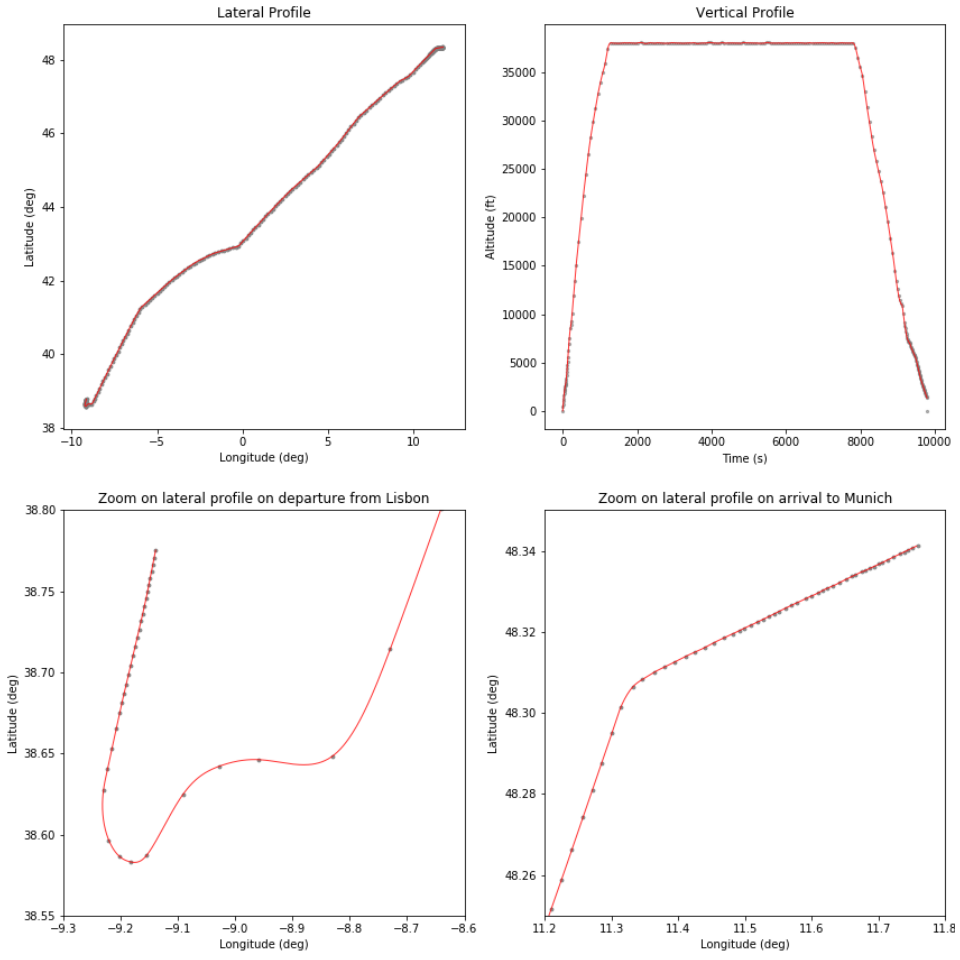


Figure 4.7: Trajectory reconstructed using λ_1 and λ_2 from out-of-sample validation.

Chapter 5

Data-Driven Terrain Representation Validation

This chapter presents the data-driven strategy designed and implemented to evaluate the SUT's inner terrain representation, as well as the key required improvements identified when applying such a strategy to evaluate the SUT's representation of Southwestern Europe.

5.1 Terrain Representation Requirements

As presented in Section 3.1, the SUT has its own inner simplified terrain representation, which needs to be validated. The representation must fully enclose the terrain to ensure that any trajectory produced by the SUT is safe in light of the real terrain raster. To ensure that this requirement is met, it was verified that each triplet of (latitude, longitude, elevation) values from the terrain raster was considered to be inside or on the surface of the SUT inner terrain representation. Since this was the case, the simplified terrain representation indeed fully encloses the real terrain.

The representation cannot be too simplistic and increase the terrain size by too much neither. Otherwise, some positions usually flown by aircraft would be declared inside the terrain representation and the SUT would obviously not be able to find a safe emergency procedure starting from them. The representation must therefore be precise enough for common aircraft operations to be accepted and judged as safe. To ensure that such is the case, a first complex option would be to recover all existing aeronautical procedures (such as airways, STARs, SIDs or others) for a given geographical area and verify if each of them was accepted by the SUT terrain representation. However, not only would this option demand a lot of work, but also it would disregard some common aircraft operations that are not explained in the procedures, such as common vectoring orders from ATC.

FlightRadar24 recordings contain real trajectories flown by aircraft in the past. Therefore, a data-driven approach using FlightRadar24 recordings to determine if the terrain representation is sufficiently precise to accept common procedures and vectoring orders is proposed. In this evaluation, aircraft navigation performances are taken into account through the RNP values.

5.2 Data-Driven Validation Strategy and Improvements Required

In the context of the data-driven validation of the SUT terrain representation, we are interested in the recordings available by geographical area. Given the huge amount of data available in the dataset (over 200 million flights over 4 years of recordings), at first only one week of data is used for the evaluation, for the purpose of efficiency. There are some seasonal routes that only exist during some periods of the year, as well as interesting outliers that only occur sporadically. However, one week of data provides a good vision of the different flights that occur and is enough to perform the first evaluations of the terrain representation and provide important feedback to iteratively improve it. Only once it becomes more mature and finding falsifying recordings becomes difficult will using more data become necessary.

The SUT was designed for commercial aircraft, for which reasons only recordings corresponding to commercial flights inside the geographical area under test were kept. However, the dataset available contains many other flight categories. There are over 500 different types of aircraft with associated recordings in the one-week dataset, including military aircraft, private jets, airliners or yet training or pleasure aircraft. Figure 5.1 presents the top-5 aircraft with the most recordings associated.

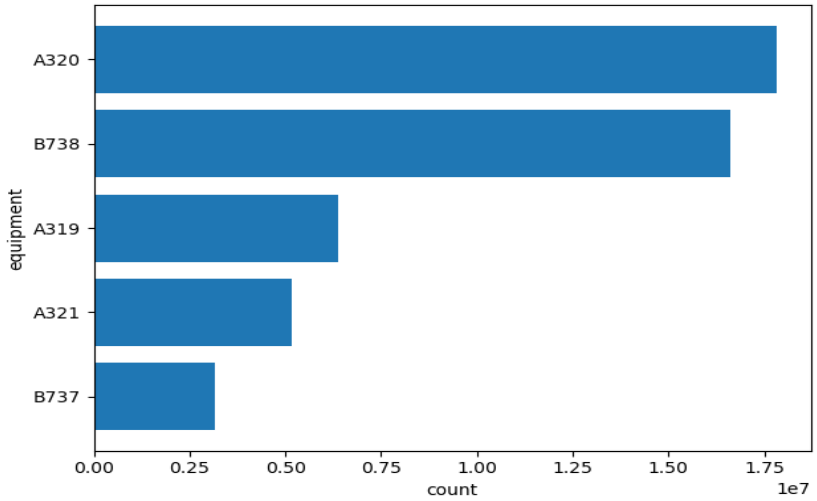


Figure 5.1: Aircraft types and respective recordings count in the one-week dataset from FlightRadar24.

It is possible to see from Figure 5.1 that most of the recordings in the dataset actually belong to airliners. From a geographical point of view, the terrain representation was evaluated on South Western Europe (between 35ºN and 50ºN and between 10ºW and 21ºE), which contains the Lisbon-Munich route to be used as a use case for the SUT evaluation later. All points are then tested for collisions with the terrain raster and collisions are plotted, to understand if in some cases even the precision of the raster is not enough or if the recording is erroneous and indeed places the aircraft inside the terrain. Some recordings considered to be inside the raster are plotted on Figure 5.2. Some are right on the terrain during take-off or landing and so they are actually correct. The raster and the flight radar altitude measurements are just not sufficiently precise for such a closeness to the terrain. However, this is not a problem. Firstly, the SUT is not supposed to be launched from a runway. Secondly, when validating the safety of a trajectory produced by the SUT with respect to the terrain raster, the trajectory only

needs to be monitored until the Final Approach Point (FAP). Figure 5.2 also shows that other recordings are clearly erroneous, with the trajectory not starting or finishing at the runway. A distance threshold $d_{threshold}$ is defined and when trajectories do not start or end within $d_{threshold}$ of the corresponding runway, they cannot be used to evaluate the SUT during the departure or arrival procedure respectively, due to the positioning errors being too great in such a situation.

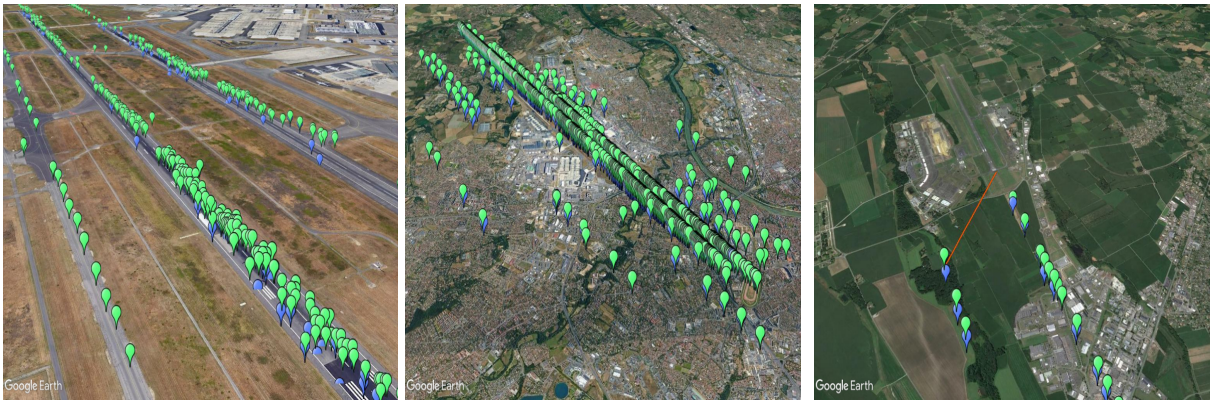


Figure 5.2: Recordings from FlightRadar24 (in blue) inside the SRTM terrain raster (in green).

Remaining recordings are then tested for collision with the SUT terrain representation. An interface allows to communicate with the SUT software component that determines the safety (or not) of a point, line segment or arc of circle. Point recordings that were safe with respect to the raster should also be safe with respect to the SUT terrain representation. Points inside the latter were plotted on Google Earth (see Figure 5.3) and analysed to understand if the representation needs to be more precise.

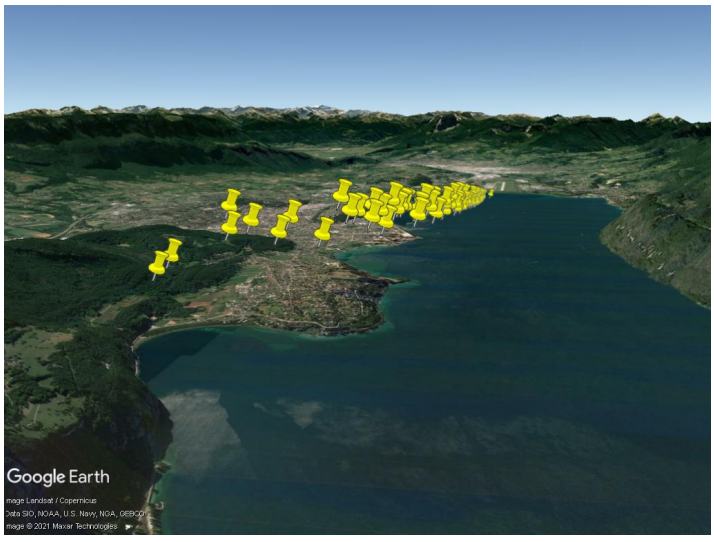


Figure 5.3: FlightRadar24 recordings considered to be inside the SUT inner terrain representation.

It is clear that the representation needs to be more precise close to airports, since good part of departure and arrival procedures are declared as colliding with the terrain. This means that the SUT cannot produce emergency trajectories when flying these procedures, even though it should. Since emergencies can occur during these phases, accepting them as safe is an essential improvement required. The

design team was informed of it. However, the SUT cannot be evaluated on departure and arrival phases yet. To determine the minimum height, $h_{threshold}$, above the runway after departure or before landing at which the SUT terrain representation is acceptable and from where the SUT can therefore be evaluated, the height of declared collisions with respect to the runway was computed. Figure 5.4 shows its distribution.

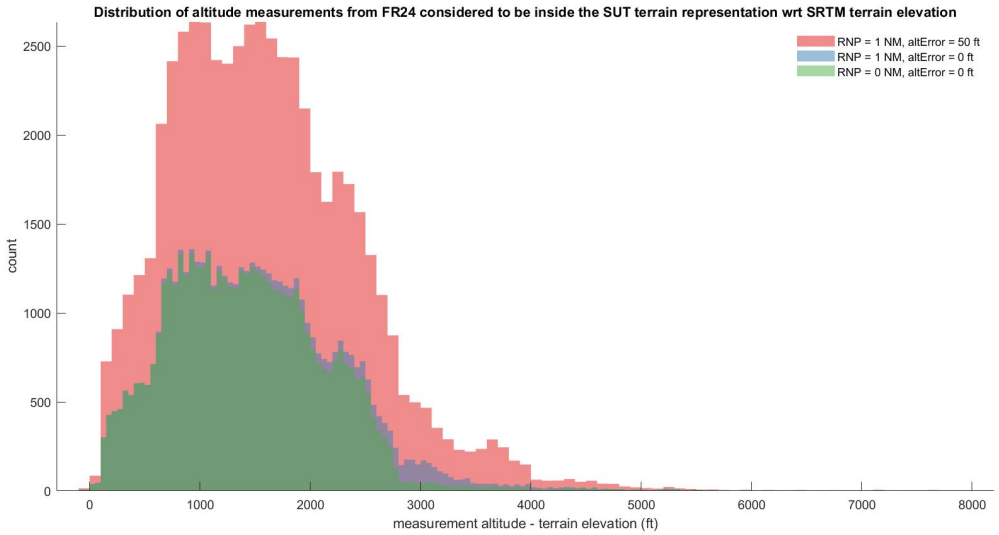


Figure 5.4: Histogram of heights with respect to runway for recordings inside the SUT inner terrain representation.

When a RNP is considered in the test for collisions, it means that the system tests the circle of radius RNP centred on the point instead of just the point itself. For this first analysis of the system, RNP was set to 1 for all collision tests because collisions are found mainly during departure and arrival procedures, for which a typical RNP value is 1. However, for future work, as the system becomes more mature and collision tests need to be more precise, using the correct values for RNP AR APCH, RNP 0.3 or RNP 0.1 procedures becomes important. The distance from a point to the terrain raster could help put in place an automatic determination of which RNP value should be used in the test for collisions within a RNP radius. Incorporating realistic altitude errors in the evaluation is also important in future work.

Even without considering the RNP value the SUT terrain representation is not sufficiently precise to test the SUT close to take-off and landing. In light of Figure 5.4, $h_{threshold}$ is set to 5000 ft, to only test the SUT on regions where the inner terrain representation is known to be sufficiently precise. This covers not only cruise but also a good part of climb and descent phases.

In future work, once the SUT terrain representation is sufficiently precise for the number of collisions found when taking the RNP value and vertical errors into account to become acceptable, reconstructed trajectories should be used to feed the last stage of the terrain representation validation instead of points. In fact, two consecutive points may be considered safe without the trajectory connecting them being so.

The data-driven strategy proposed allows to quickly give important feedback to the design team until a final suitable SUT terrain representation is obtained.

Chapter 6

Evaluation of the SUT

This chapter proposes a strategy for the evaluation of the SUT, as well as presents the main conclusions regarding its performance and important feedback to design future improvements, thanks to the implementation and application of the strategy proposed. In order to evaluate the performance that the SUT would present on the environments it was designed for, a data-driven strategy taking advantage of historical data from FlightRadar24 is proposed. It relies on the ability to efficiently cluster trajectories from a given route, so Section 6.1 reviews two promising clustering algorithms (HDBSCAN and GMM) and evaluates their performance on the LIS-MUC route to select the most suitable one for the route trajectories clustering problem. Section 6.2 then presents a strategy to construct a realistic and as optimal as possible list of diversion options for any given route cluster. Having clustered route trajectories and constructed the diversion list for each cluster, a data-driven evaluation of the performance that the SUT would present if used when flying on a trajectory from a cluster is performed in Section 6.3. At last, Section 6.4 focuses on the search for scenarios that are challenging for the SUT, attributing less importance to how likely they are. This is still interesting, since it allows to understand the kind of scenario in which the SUT would face more difficulties and helps identify improvements required, regardless of the scenarios likelihood.

6.1 Commercial Route Trajectories Clustering

6.1.1 HDBSCAN Clustering

The Hierarchical DBSCAN* (HDBSCAN) algorithm [45] is a density-based, hierarchical clustering method that provides a clustering hierarchy. This hierarchy contains all possible DBSCAN*-like solutions for an infinite range of density thresholds and a simplified hierarchy composed only of the most significant clusters can be easily extracted from it. [45] also proposes a novel cluster stability measure, formalizes the problem of maximizing the overall stability of selected clusters and formulates an algorithm that computes an optimal solution to this problem, allowing to obtain a flat partition consisting only of the most significant clusters, extracted from optimal local cuts through the cluster tree.

The proposed approach presents a significant advantage when compared to other density-based

methods. DBSCAN [19] can only provide a flat labeling of the data objects based on a global density threshold. However, using a single density threshold cannot properly characterize datasets with clusters of different densities and/or nested clusters. OPTICS [46] has the same problem of using a global density threshold. The proposed approach does not have this problem, since cuts corresponding to different density thresholds can be made when obtaining the flat clusters from the clustering hierarchy.

Since HDBSCAN can be seen as an improvement of DBSCAN* (a variation of the DBSCAN algorithm), we start by revisiting the DBSCAN* algorithm as done in [45]. Let $\mathrm{X} = \{\mathrm{x}_1, \cdots, \mathrm{x}_n\}$ be a dataset of n objects, and let $d(\mathrm{x}_p, \mathrm{x}_q)$ be the distance between $\mathrm{x}_p, \mathrm{x}_q \in \mathrm{X}$. DBSCAN* defines density-based clusters based on *core objects*, accordingly to the definitions that follow.

Definition 1. (Core Object): An object x_p is called a core object w.r.t. ε and m_{pts} if its ε-neighborhood contains at least m_{pts} many objects, i.e., if $|\mathbf{N}_\varepsilon(\mathbf{x}_p)| \geq m_{pts}$, where $\mathbf{N}_\varepsilon(\mathbf{x}_p) = \{\mathrm{x} \in \mathbf{X} \mid d(\mathrm{x}, \mathbf{x}_p) \leq \varepsilon\}$ and $|\cdot|$ denotes cardinality. An object is called noise if it is not a core object.

Definition 2. (ε-Reachable) : Two core objects x_p and x_q are ε-reachable w.r.t. ε and m_{pts} if $\mathbf{x}_p \in \mathbf{N}_\varepsilon(\mathbf{x}_q)$ and $\mathbf{x}_q \in \mathbf{N}_\varepsilon(\mathbf{x}_p)$.

Definition 3. (Density-Connected): Two core objects x_p and x_q are density-connected w.r.t. ε and m_{pts} if they are directly or transitively ε-reachable.

Definition 4. (Cluster): A cluster $\mathbf{C}$ w.r.t. ε and m_{pts} is a non-empty maximal subset of X such that every pair of objects in C is density-connected.

DBSCAN* therefore conceptually finds clusters as the connected components of a graph in which the objects of X are vertices and every pair of vertices is adjacent if and only if the corresponding objects are ε-reachable w.r.t. user-defined parameters ε and m_{pts}. Non-core objects are labeled as noise. In HDBSCAN, m_{pts} is the single input required and different density levels in the resulting density-based cluster hierarchy w.r.t. m_{pts} correspond to different values of the radius ε. HDBSCAN is based on the following definitions introduced in [45]:

Definition 5. (Core Distance): The core distance of an object $\mathbf{x}_p \in \mathbf{X}$ w.r.t. m_{pts}, $d_{\text{core}}(\mathbf{x}_p)$, is the distance from $\mathbf{x}_p$ to its m_{pts}-nearest neighbor (including $\mathbf{x}_p$).

Definition 6. (ε-Core Object) : An object $\mathrm{x}_p \in \mathbf{X}$ is called an ε-core object for every value of ε that is greater than or equal to the core distance of $\mathbf{x}_p$ w.r.t. m_{pts}, i.e., if $d_{\text{core}}(\mathbf{x}_p) \leq \varepsilon$.

Definition 7. (Mutual Reachability Distance): The mutual reachability distance between two objects x_p and x_q in $\mathbf{X}$ w.r.t. m_{pts} is defined as $d_{\text{mreach}}(\mathbf{x}_p, \mathbf{x}_q) = \max\{d_{\text{core}}(\mathbf{x}_p), d_{\text{core}}(\mathbf{x}_q), d(\mathbf{x}_p, \mathbf{x}_q)\}$.

Definition 8. (Mutual Reachability Graph): It is a complete graph, $G_{m_{pts}}$, in which the objects of $\mathbf{X}$ are vertices and the weight of each edge is the mutual reachability distance (w.r.t. m_{pts}) between the respective pair of objects.

If $G_{m_{pts},\varepsilon} \subseteq G_{m_{pts}}$ is the graph that results from removing all edges from $G_{m_{pts}}$ having weights greater than ε, then the connected components of ε-core objects in $G_{m_{pts},\varepsilon}$ correspond to DBSCAN* clusters w.r.t. m_{pts} and ε. This means that all DBSCAN* partitions for $\varepsilon \in [0, \infty[$ could be produced in a nested, hierarchical way by removing edges in decreasing order of weight from $G_{m_{pts}}$. The DBSCAN* partition would therefore be obtained by first running Single-linkage over the mutual reachability distances and then cutting the dendrogram at level ε. Connected components would form clusters and singletons with

$d_{\text{core}}\ (\mathbf{x}_p) > \varepsilon$ would be noise. In [45] a more efficient alternative is proposed.

Since an object is considered to be noise if ε is less than its core distance, an extension of a Minimum Spanning Tree (MST) of the Mutual Reachability Graph $G_{m_{pts}}$ with edges connecting each vertex to itself was proposed to construct the hierarchy. This extended MST not only contains all possible DBSCAN* partitions but also information about when an object becomes noise (when the weight threshold decreases below the object's core distance stored in the self edge). Algorithm 1 presents the pseudo-code for the HDBSCAN hierarchization algorithm just presented. For more details refer to [45].

Algorithm 1 HDBSCAN main steps (from [45])

1. Compute the core distance w.r.t. m_{pts} for all data objects in X.
2. Compute an MST of $G_{m_{pts}}$, the Mutual Reachability Graph.
3. Extend the MST to obtain MST_{ext}, by adding for each vertex a "self edge" with the core distance of the corresponding object as weight.
4. Extract the HDBSCAN hierarchy as a dendrogram from MST_{ext} :
   - 4.1 For the root of the tree assign all objects the same label (single "cluster").
   - 4.2 Iteratively remove all edges from MST_{ext} in decreasing order of weights (in case of ties, edges must be removed simultaneously):
     - 4.2.1 Before each removal, set the dendrogram scale value of the current hierarchical level as the weight of the edge(s) to be removed.
     - 4.2.2 After each removal, assign labels to the connected component(s) that contain(s) the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level: assign a new cluster label to a component if it still has at least one edge, else assign it a null label ("noise").

In case of large datasets, the processing and visualization of the resulting dendrograms would not be easy. Since in many cases reducing the weight threshold simply removes an object from a cluster and converts it into noise but keeps the clusters' general structure unchanged, there is a need to keep only thresholds corresponding to significant clusters' changes, rather than all possible cluster changes. The simplification of the HDBSCAN hierarchy proposed in [45] is based on the observation that when reducing the threshold and converting to noise an object that belongs to a certain cluster, there are only three possibilities: the cluster shrinks but remains connected, the cluster is divided into smaller clusters or the cluster disappears. Interesting thresholds are those at which clusters either split into smaller clusters or disappear. When removing noise from a cluster, the cluster should keep the same label.

HDBSCAN can be adapted to this by replacing step 4.2.2 in Algorithm 1 by step 4.2.2 in Algorithm 2, where an optional parameter $m_{clSize} \geq 1$ allows to consider connected components with less than m_{clSize} objects as spurious and disregard removals of such components from bigger components. m_{clSize} is set to m_{pts}, making m_{pts} both a smoothing factor and a threshold for cluster size.

Algorithm 2 HDBSCAN step 4.2.2 with (optional) parameter $\text{m}_{\text{clsize}} \geq 1 (from [45])$

4.2.2. After each removal (to obtain the next hierarchical level), process one at a time each cluster that contained the edge(s) just removed, by relabeling its resulting connected subcomponent(s):

Label spurious subcomponents as noise by assigning them the null label. If all subcomponents of a cluster are spurious, then the **cluster has disappeared**.
Else, if a single subcomponent of a cluster is not spurious, keep its original cluster label (**cluster has just shrunk**).
Else, if two or more subcomponents of a cluster are not spurious, assign new cluster labels to each of them (**true cluster split**).

Once the HDBSCAN hierarchy has been obtained, the aim is to obtain a flat partition of the data, with the clusters possibly having different local densities and not being detectable by a global density threshold. If a density function $f(x)$ is defined as $f(x) = 1/d_{core}(x)$ and a density threshold λ is defined as $\lambda = 1/\epsilon$, then the density-contour DBSCAN* clusters are the maximal connected subsets of the level set defined as $\{x \mid f(x) \geq \lambda\}$. As density increases, clusters get smaller until they eventually disappear or split into sub-clusters. To measure the stability of a cluster, the density function is evaluated in the range of densities at which the cluster under evaluation exists. Without loss of generality, and to simplify the understanding and visualization of the concept, let us consider x a continuous variable. The *excess of mass* of a cluster $\mathbf{C}_i$ is defined in Equation 6.1 and illustrated in Figure 6.1, with the darker shaded areas representing the excesses of mass of three clusters $\mathbf{C}_3$, $\mathbf{C}_4$ and $\mathbf{C}_5$.

$$E\left(\mathbf{C}_i\right) = \int_{x \in \mathbf{C}_i} \left(f(x) - \lambda_{\min}\left(\mathbf{C}_i\right)\right) dx \tag{6.1}$$

where $\lambda_{\min}\left(\mathbf{C}_i\right)$ is the minimum density level at which $\mathbf{C}_i$ exists.

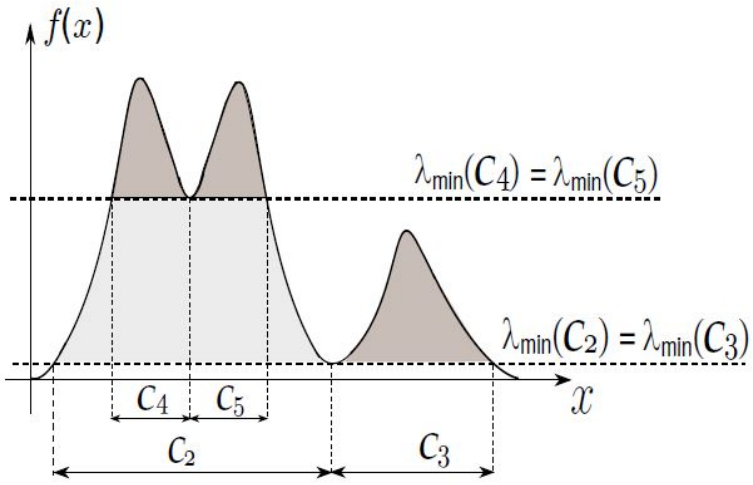


Figure 6.1: Illustration of a density function, clusters and excesses of mass. From [45].

Since the excess of mass presents a monotic behaviour along any branch in the hierarchy tree, the *relative excess of mass* defined in Equation 6.2 is introduced to compare the stability of nested clusters.

$$E_R\left(\mathbf{C}_i\right) = \int_{x \in \mathbf{C}_i} \left(\lambda_{\max}\left(x, \mathbf{C}_i\right) - \lambda_{\min}\left(\mathbf{C}_i\right)\right) dx \tag{6.2}$$

with $\lambda_{\max}\left(x, \mathbf{C}_i\right)$ defined as $\lambda_{\max}\left(x, \mathbf{C}_i\right) = \min\left\{f(x), \lambda_{\max}\left(\mathbf{C}_i\right)\right\}$ and $\lambda_{\max}\left(\mathbf{C}_i\right)$ the density level at which cluster C_i is split or disappears. For instance, for cluster $\mathbf{C}_2$ in Figure 6.1, $\lambda_{\max}\left(\mathbf{C}_2\right) = \lambda_{\min}\left(\mathbf{C}_4\right) = \lambda_{\min}\left(\mathbf{C}_5\right)$ and the corresponding relative excess of mass is represented by the lighter shaded area.

For a finite dataset, the cluster stability is adapted and defined in Equation 6.3.

$$S\left(\mathbf{C}_i\right) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \left(\lambda_{\max}\left(\mathbf{x}_j, \mathbf{C}_i\right) - \lambda_{\min}\left(\mathbf{C}_i\right)\right) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \left(\frac{1}{\varepsilon_{\min}\left(\mathbf{x}_j, \mathbf{C}_i\right)} - \frac{1}{\varepsilon_{\max}\left(\mathbf{C}_i\right)}\right) \tag{6.3}$$

with $\varepsilon_{\max}\left(\mathbf{C}_i\right)$ and $\varepsilon_{\min}\left(\mathrm{x}_j, \mathrm{C}_i\right)$ the density values corresponding to $\lambda_{\min}\left(\mathbf{C}_i\right)$ and $\lambda_{\max}\left(\mathrm{x}_j, \mathrm{C}_i\right)$.

Finally, obtaining the best flat, non-overlapping partition is formulated as an optimization problem defined in Equation 6.4.

$$\begin{aligned} \max_{\delta_2,\ldots,\delta_\kappa} \quad & J = \sum_{i=2}^{\kappa} \delta_i S(\mathbf{C}_i) \\ \text{subject to} \quad & \begin{cases} \delta_i \in \{0,1\}, \quad i = 2, \cdots, \kappa \\ \sum_{j \in \mathbf{I}_h} \delta_j = 1, \forall h \in \mathbf{L} \end{cases} \end{aligned} \tag{6.4}$$

where $\{\mathrm{C}_2, \cdots, \mathrm{C}_\kappa\}$ is the collection of all clusters in the simplified cluster hierarchy (except the root C_1 that contains the whole dataset), $\delta_i (i = 2, \cdots, \kappa)$ indicates whether cluster C_i is included in the flat solution ($\delta_i = 1$) or not ($\delta_i = 0$), $\mathrm{L} = \{h \mid \mathrm{C}_h$ is a leaf cluster $\}$ is the set of indexes of leaf clusters, and $\mathbf{I}_h = \{j \mid j \neq 1$ and $\mathbf{C}_j$ is ascendant of $\mathbf{C}_h (h$ included $)\}$ is the set of indexes of all clusters on the path from C_h to the root (excluded). The constraints prevent nested clusters on the same path to be selected.

[45] also presents a recursive algorithm to solve the optimization problem just presented. This algorithm is not presented here since for the implementation of the HDBSCAN algorithm, as well as for obtaining a flat partition of our dataset, the hdbscan library [47] [48] was used. Figure 6.2 illustrates the optimal selection of clusters from an example cluster tree.

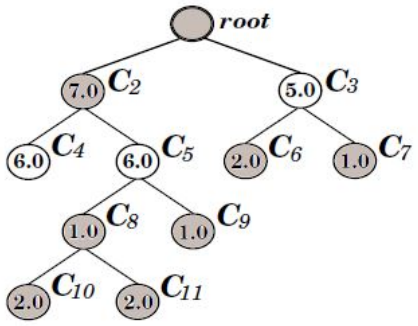


Figure 6.2: Illustration of the optimal selection of clusters from a given cluster tree. From [45].

6.1.2 Gaussian Mixture Model

A Gaussian Mixture is a function that contains K (the number of clusters) Gaussians, each defined by a mean μ_k, a covariance matrix Σ_k defining the width and a mixing probability π_k defining the cluster frequency (the overall probability of observing a sample that belongs to cluster k) such that (6.5) holds.

$$\sum_{k=1}^{K} \pi_k = 1 \tag{6.5}$$

The problem is determining the optimal values for the parameters π_k, μ_k and Σ_k. They correspond to the Maximum Likelihood Estimates of the Gaussian density function differentiated with respect to the mean and covariance and equalled to zero. However, obtaining an analytical solution is

very hard. Therefore, an iterative method is used to estimate the parameters instead: the Expectation—Maximization (EM) algorithm, which is often used for optimization problems when the objective function is too complex. The algorithm can initialize the parameters from the results obtained from a previous K-means run as a good starting point. There are then two steps:

1. Expectation step

$$\tau_{ik} = \frac{\pi_k \mathcal{N}\left(x_i|\mu_k, \Sigma_k\right)}{\sum_{l=1}^{K} \pi_l \mathcal{N}\left(x_i|\mu_l, \Sigma_l\right)}$$

2. Maximization step

$$\pi_k = \frac{1}{K}\sum_{i=1}^{K} \tau_{ik}, \quad \mu_k = \frac{\sum_{i=1}^{K} \tau_{ik} x_i}{\sum_{i=1}^{K} \tau_{ik}}$$
$$\Sigma_k = \frac{\sum_{i=1}^{K} \tau_{ik} \left(x_i - \mu_k\right)\left(x_i - \mu_k\right)^T}{\sum_{i=1}^{K} \tau_{ik}}$$

where x_i is a trajectory, Σ_k is the covariance matrix of cluster k and $\mathcal{N}$ is the Normal distribution. The values revised in step 2 are then used in step 1 of the next EM iteration, and so on until convergence is reached.

The algorithm just described is based on probabilistic distribution models and performs *soft clustering*, which means each trajectory is assigned to each cluster with a certain probability. This is fundamentally different from K-means' *hard clustering*, where each trajectory is rigidly assigned to a single class. Gaussian Mixtures therefore allow the identification of outliers, unlike K-means. Additionally, there is the possibility to create a generative model from a Gaussian Mixture. In fact, given that a gaussian distribution conditioned on past distributions gives another gaussian distribution, from samples of part of a trajectory it would be possible to generate the posterior trajectory conditioned on the initial measurements. This strategy has been successfully used for trajectory prediction in the TMA [29]. This thesis does not treat the subject of trajectory prediction. However, such a generative model could be interesting for sampling test trajectories accordingly to the underlying distributions of clusters. This was not done in this thesis but could be an interesting axis of future work.

To determine the best value for the number of clusters K, one can use model selection, which is the process of fitting several models on a dataset and choosing the best one. Two probabilistic model selection options are considered: Akaike Information Criterion (AIC) [49] and Bayesian Information Criterion (BIC) [50]. Models are scored based both on their performance and complexity. The model complexity is evaluated simply through the number of clusters K, whereas the model performance may be evaluated using a probabilistic framework, such as log-likelihood under the framework of Maximum Likelihood Estimation (MLE). MLE wishes to maximize $P(\mathbf{X}|\theta)$, $\mathbf{X}$ being the data containing N observations that follows the probability distribution and θ its parameters. Since $\mathbf{X}$ contains many observations and the joint probability distribution obtained when multiplying many small conditional probabilities for observing each sample can be unstable, the log-probability or log-likelihood (LL) function is usually used instead.

AIC is derived from frequencist probability and is defined by (6.6).

$$AIC = -\frac{2 * LL}{N} + \frac{2 * K}{N} \tag{6.6}$$

BIC is derived from Bayesian probability and is defined by (6.7).

$$BIC = -2 * LL + log(N) * K \tag{6.7}$$

Although AIC and BIC are different, they can be shown to be proportional to each other, with BIC penalizing the model more for its complexity and AIC penalizing the model more based on its performance.

The sklearn implementation of Gaussian Mixture [51] was used to train the Gaussian Mixtures and to obtain the AIC and BIC values of the different models trained.

6.1.3 Choice of the Clustering Algorithm

The LIS-MUC case study was used to evaluate the quality of the clusters obtained when applying the HDBSCAN and GMM algorithms and choose the most suitable one for future SUT evaluations on different routes. Figure 6.3 presents the partition of the HDBSCAN hierarchy cluster tree that maximizes the sum of stabilities of the extracted clusters presented in Equation 6.4.

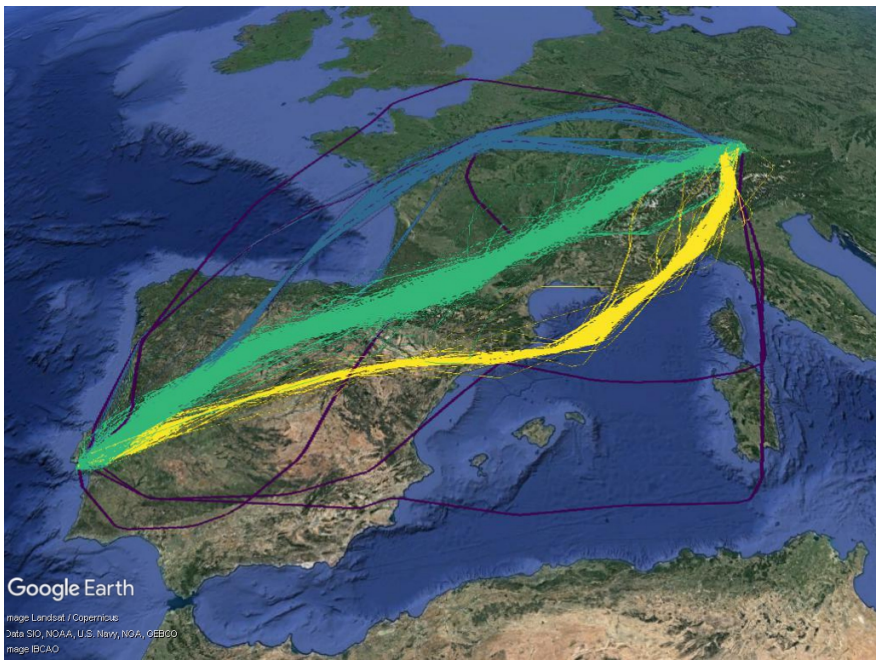


Figure 6.3: Clusters (yellow, green and blue) and noise (purple) found by HDBSCAN on LIS-MUC route.

Three clear distinct clusters are identified. The one in yellow contains approximately 300 trajectories that pass south of the Pyrenees. The one in green contains approximately 6500 trajectories that pass over or just north of the Pyrenees. The one in blue contains approximately 300 trajectories that start by heading considerably more to the north and only then head east. These clusters are probably associated to Flight Plans that use different airways. There are also a few trajectories (8 in total) in purple, which were correctly identified as noise. These are either indeed very exotic or simply they have only some parts following the cluster or even they switch from cluster during the flight. In the latter case, it could mean that the airway that the aircraft was following was closed. Figure 6.4 presents the condensed tree produced by the HDBSCAN algorithm and from where the clusters in Figure 6.3 have been obtained.

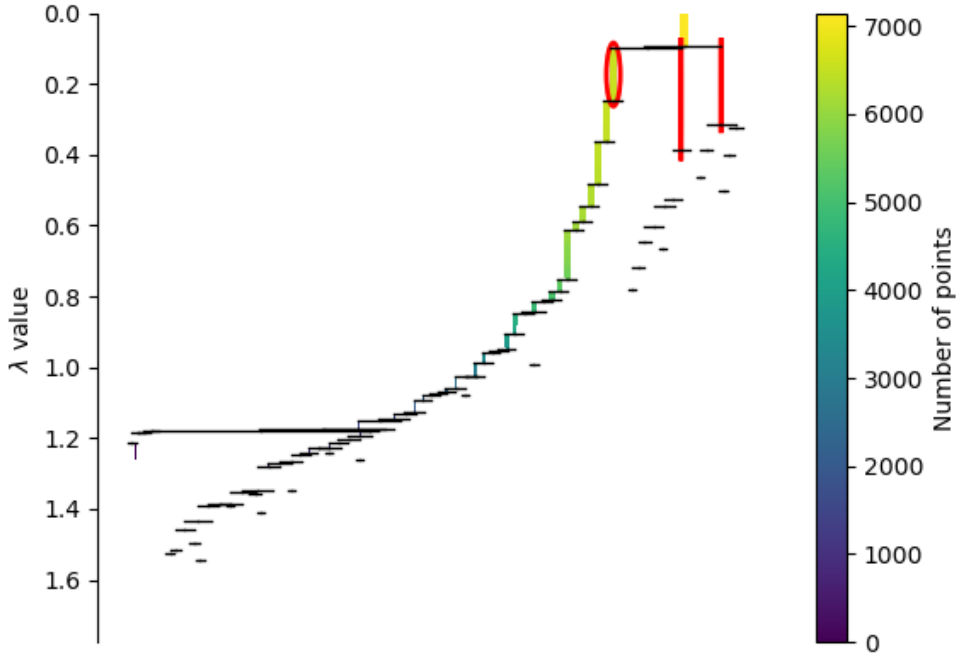


Figure 6.4: Condensed tree produced by the HDBSCAN algorithm applied to the LIS-MUC route.

The results are very satisfactory. Even though the algorithm could have split the data into more clusters, the three clusters identified by HDBSCAN have the perfect size to construct a diversion list by cluster for the evaluation of the SUT on the LIS-MUC route.

GMM was evaluated on the same dataset. Figure 6.5 presents the AIC and BIC evolution with K.

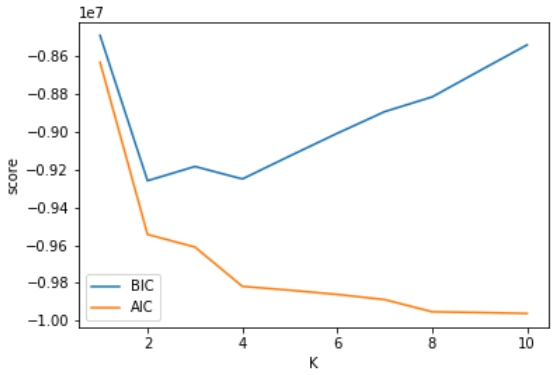
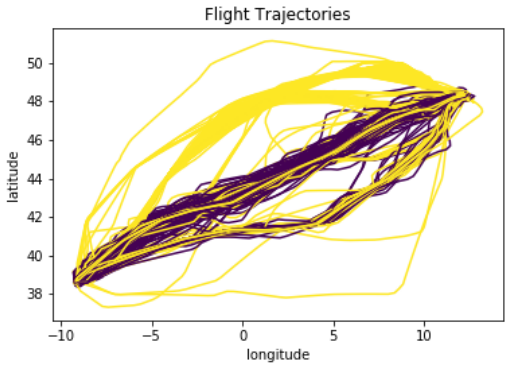
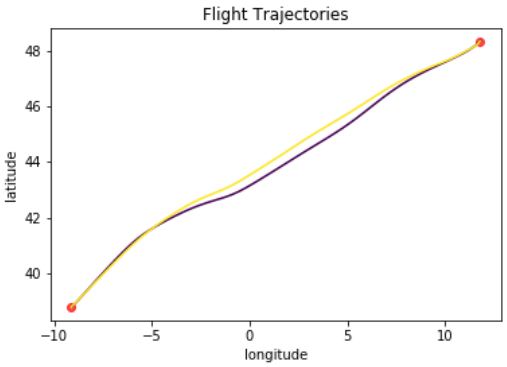


Figure 6.5: AIC and BIC vs number of clusters K when training GMM on the LIS-MUC route.

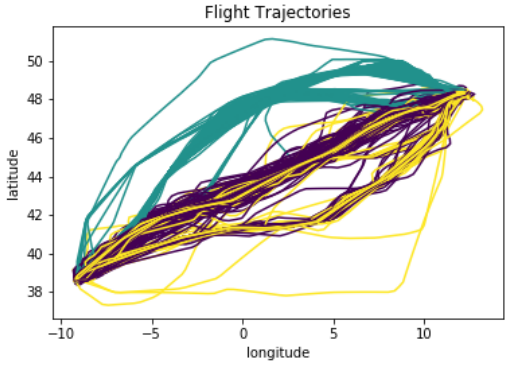
It is clear that BIC penalizes more the model complexity while AIC penalizes more the model performance, as explained in Section 6.1.2. The BIC score curve indicates 2, 3 and 4 as interesting values for K, which is coherent with $K = 3$ found by HDBSCAN. The AIC score curve suggests 8 clusters. The clusters obtained after training a GMM with K equal to 2, 3, 4 and 8 are presented in Figure 6.6.



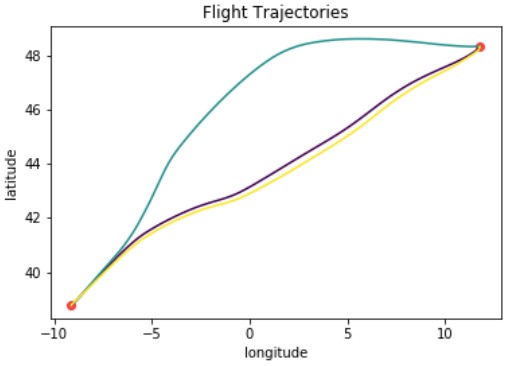
(a) Clusters for K=2



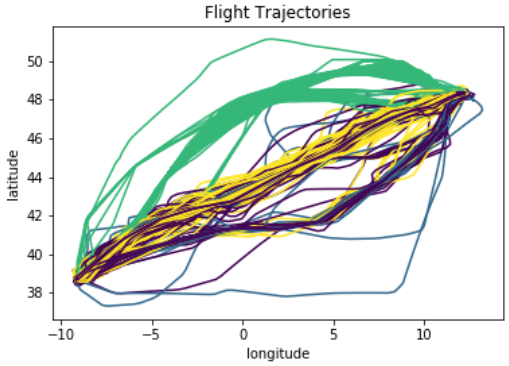
(b) Cluster centres for K=2



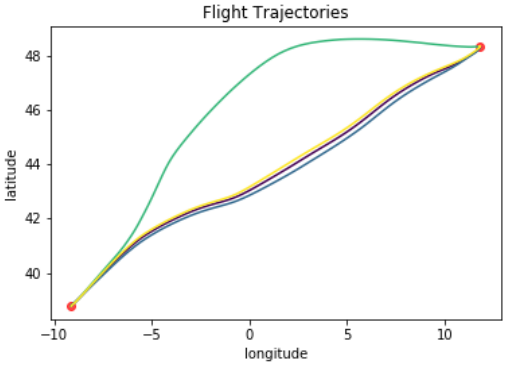
(c) Clusters for K=3



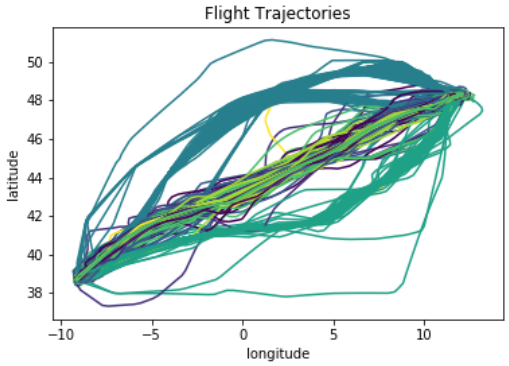
(d) Cluster centres for K=3



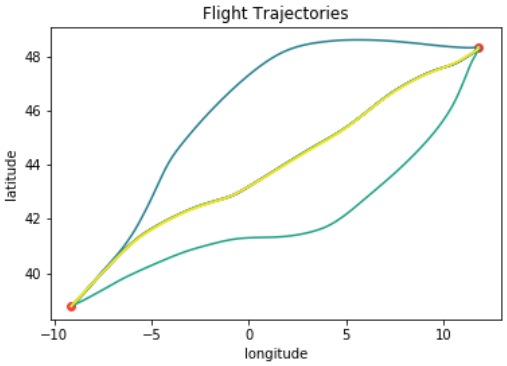
(e) Clusters for K=4



(f) Cluster centres for K=4



(g) Clusters for K=8



(h) Cluster centres for K=8

Figure 6.6: GMM clusters from the LIS-MUC route.

In Figure 6.6 it is possible to see that the clustering produced by GMM models is significantly worse than the one produced by HDBSCAN. For $K = 2$ the trajectories more to the North mostly belong to the same cluster, which is positive. However, that same cluster also contains several trajectories more to the South and this mix of trajectories ends up resulting in the two cluster centres being almost coincident with the big central cluster that had already been identified by HDBSCAN previously. For $K = 3$ a cluster clearly contains the North trajectories, which is an improvement with respect to $K = 2$. As far as the two remaining clusters are concerned, we would expect one cluster to contain the central trajectories and the other one to contain the trajectories that are more to the South, precisely as it has been with HDBSCAN. However, even though this is more or less the case, the fact that once again there are trajectories from the central cluster attributed to the South cluster and vice-versa results in the two cluster centres being almost coincident. The GMM obtained with $K = 4$ suffers from the same problem. When $K = 8$ it is finally possible to find the 3 clear clusters that had been found by HDBSCAN. However, there are once again several cluster centres almost coincident with each other. From the cluster centres obtained with $K = 8$ it seems clear that 3 is the ideal number of clusters, as it had been found by HDBSCAN. However, GMM does not find the 3 clusters that it should, performing quite worse than HDBSCAN.

In light of the great performance exhibited by HDBSCAN and the significantly worse performance exhibited by GMM, HDBSCAN is chosen to be used henceforth for clustering route trajectories.

6.2 Diversion List Construction

As presented in Section2.3, there are different categories of precision approach. Table 2.1 showed that the higher the category, the lower the Decision Height and Runway Visual Reference values can be. For this reason, in order to ensure that the approach of the emergency procedure computed by the SUT is safe and accepted even under low visibility conditions, a choice was made to only consider runways with CAT II or III means when constructing the list of diversion options.

A dataset containing runways equipped with ILS CAT II or III means was used for the construction of the list of diversion options. It is proposed that the area covered by each cluster is represented by a set of points from a grid containing a point every 0.5 degrees of latitude or longitude. For each cluster, each recording available is assigned to the grid point it is closest to. Then, only the grid points that have at least one recording assigned to them are kept. They represent the cluster, as Figure 6.7 illustrates.

Throughout a flight there should always be at least one suitable diversion airport within one hour of flight. Considering a very simplified energy management descent profile, let us assume that aircraft can descend at a rate of 3000 ft per minute until reaching FL 100 (flight level 10000 ft) and from then on 1500 ft per minute. Assuming cruise at FL 340, if we consider a ground speed of 300 kts during the descent until FL 100 and 250 kts from there on (assuming a runway at sea-level), then descent requires approximately 15 minutes and a distance of approximately 70 Nautical Miles (NM). To ensure that a diversion is always possible in less than 1 hour, we could still admit 45 minutes of cruise flight before starting descent. If a ground speed of 400 kts was considered, this would mean a distance of 300 NM before the descent. Since the model is extremely simplified, not taking winds into account when

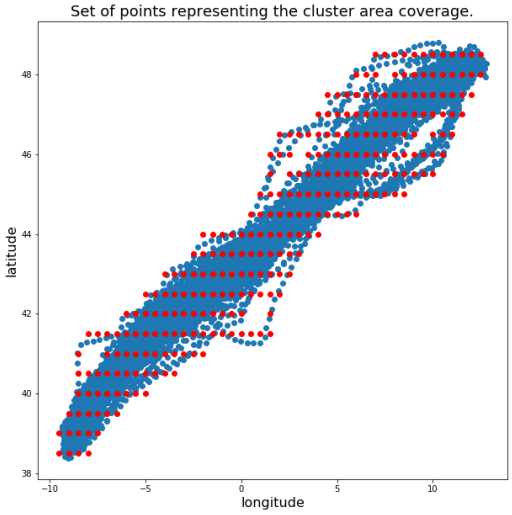


Figure 6.7: The area covered by a cluster that contains approximately 6500 flights, each of them with several associated recordings displayed in blue, can be represented by the few hundred red points.

assuming ground speed values, and to be conservative and ensure diversion is always indeed possible within 1 hour of flight, it is considered that a suitable list must ensure the permanent existence of a diversion airport within a range of 250 NM from the aircraft. The list of diversion airports should also always contain diversion airports:

- close to the departure airport (which can either be the departure airport itself, a close one or even both), to ensure a quick diversion in case of problems during take-off;
- close to the destination, to ensure a close diversion if landing at the destination is not possible;
- the destination, in case an automated emergency landing is necessary when already close to it.

In the diversion list construction, a data-driven approach should be used to take value out of available historical data. As it was remarked in Section 4.3, the airports to which aircraft diverted in the past have been selected by pilots or airlines OCC at the time. For this reason, they are assumed to be preferred options. In light of the historical diversions presented in Figure 4.3, LPPT (ICAO code for Lisbon), LPPR (Porto), LFBO (Toulouse), LFML (Marseille), EDDN (Nuremberg) and EDDM (Munich) airports are inserted in the diversion list for a LIS-MUC use case cluster. It is then checked whether there are cluster representative points that are more than 250 NM away from these airports. For the cluster use case there is a single point in that situation, as Figure 6.8 shows. All airports with ILS CAT II or III capabilities that are less than 250 NM from the point yet to be covered are also displayed. LEMD (Madrid) is easily selected in this case to ensure the coverage of the uncovered cluster representative point.

In general, the minimal number of airports allowing to ensure that all uncovered cluster representative points are less than 250 NM away from at least one airport needs to be determined. This can be formulated as a set cover problem. A set cover problem is defined in [52] as follows:

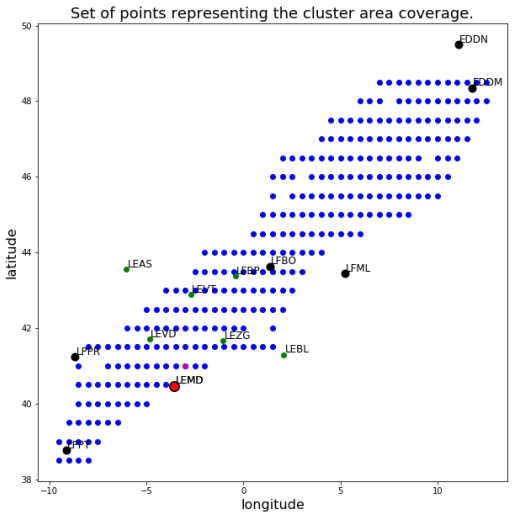


Figure 6.8: Cluster diversion list construction. Airports to where aircraft diverted in the past in black. Cluster representative points less than 250 NM from at least one of these in blue. Uncovered cluster representative points in magenta. Airports with ILS CAT II or III capabilities less than 250 NM from at least one magenta point in green. LEMD (in red) is selected to ensure coverage of the magenta points.

Set Cover Problem. Given a universe U of n elements, a collection of subsets of U, $S = \{S_1, ..., S_k\}$, and a cost function $c : S \longrightarrow \mathbf{Q}^+$, find a minimum cost subcollection of S that covers all elements of U.

The universe U consists of the uncovered cluster representative points. Each set S_i contains all points from U within 250 NM of airport i. Airports that are farther than 250 NM from any of the points from U are disregarded. To solve the set cover problem there are some different approaches in the literature. Among these, defining the problem as an Integer Linear Program (ILP) was chosen. To formulate it, a variable x_{S_i} is assigned for each set $S_i \in S$. This variable is allowed 0 or 1 values: 1 if set S_i is picked and 0 otherwise. The constraint of the problem is that for each element $e \in U$ at least one of the sets containing it must be picked. The problem is therefore formulated by (6.8).

$$\begin{aligned} \min_{x_{S_i}} \quad & \sum_{S_i \in S} x_{S_i} \\ \text{subject to} \quad & \begin{cases} \sum_{S_i : e \in S_i} x_{S_i} \geq 1, & e \in U \\ x_{S_i} \in \{0, 1\}, & S_i \in S \end{cases} \end{aligned} \tag{6.8}$$

To solve this problem, the required code was written using the PuLP library [53], which is an LP modeler written in Python. PuLP allows to define the problem variables x_{S_i} as binary variables.

For the cluster previously presented LEMD was a direct choice. For the cluster of southern trajectories, the same initial list of airports was used and LEMD was once again chosen to ensure the coverage, as presented on the right side of Figure 6.9. For the cluster containing the more northern trajectories, LFBO and LFML no longer make sense as they are quite far from the cluster. Therefore, they were eliminated from the initial list of airports. The method described in this section selected LFBD and LFPG

to complete the diversion list, which is presented on the left side of Figure 6.9.

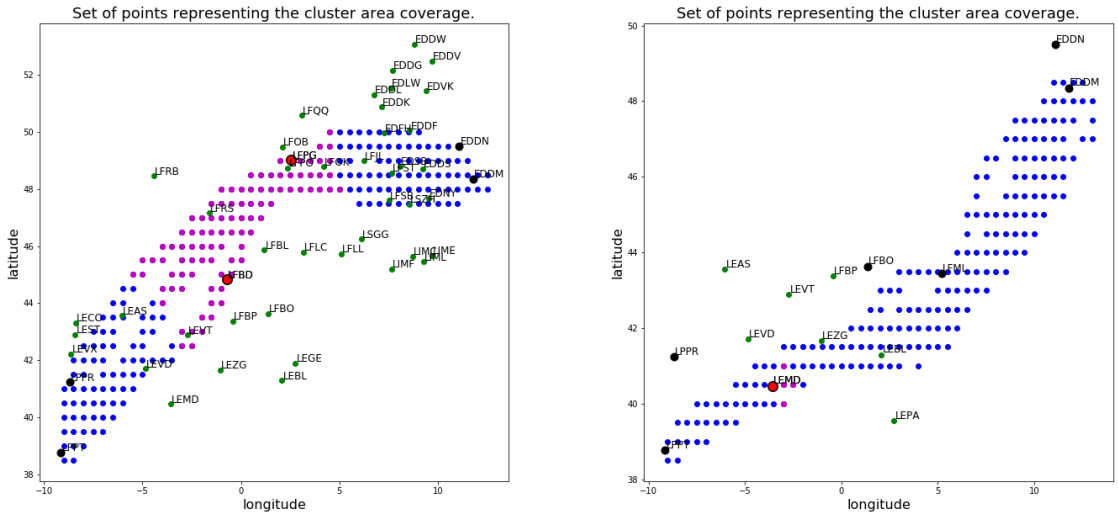


Figure 6.9: Final list of diversion airports for the northern and southern clusters of the LIS-MUC route.

It is important to remark that although a list of airports is first constructed, the list fed to the SUT then consists of the runways with associated ILS CAT II or III capabilities from these. When evaluating the SUT, the list corresponding to the cluster to which the flight under analysis belongs is fed to the SUT.

6.3 Evaluation of the SUT on Commercial Routes

This section presents a strategy to evaluate the performance the SUT would have if embarked on flights from a route under evaluation. It should find the worst scenario, but also provide a view of its mean performance and ensure a good scenario coverage. An obstacle-free environment (only the terrain is to be avoided) is first considered (Section 6.3.1) and then the construction of challenging environments is included in the search strategy (Section 6.3.2). LIS-MUC is once again the test use case.

6.3.1 Nominal Environment

To ensure good scenario coverage, it is proposed that the range of possible values for the aircraft state is discretized into a grid of cells. Points composing the trajectories reconstructed from FlightRadar24 measurements are then assigned to the cell containing them and the group of cells with at least one point assigned to them represents the portion of the search space covered by the route or cluster on which the SUT is to be evaluated. Cells should not be too large, in order to avoid enlarging the area to be explored by too much. However, they should not be too precise either, in order to avoid representing the relevant search space by too many small cells and then wasting resources exploring them all. If a cell holds promise of containing the worst scenario, a local more in-depth search can be performed. The discretization was performed using the following deltas:

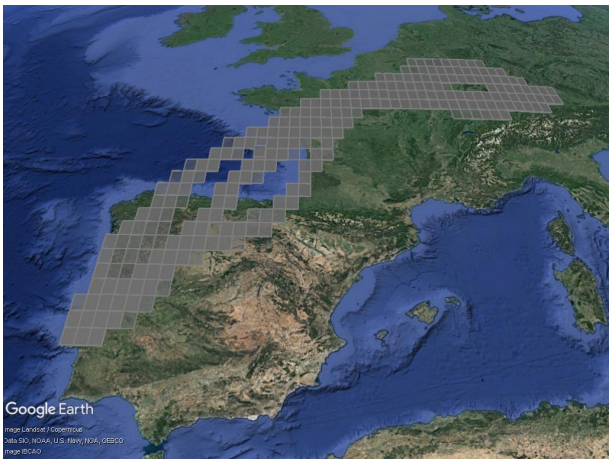


Figure 6.10: Cells representing the LIS-MUC route.

- Δ Latitude = 0.25 deg
- Δ Longitude = 0.25 deg
- Δ Altitude = 10000 ft
- Δ Heading = 10 deg
- Δ TASSpeed = 15 kt
- Δ VerticalSpeed = 5 kt

Figure 6.10 shows the cells representing the LIS-MUC route obtained with these deltas. It is important to remark that Figure 6.10 presents only the latitude-longitude discretization. However, each cell represented in it actually contains sub-cells for the different values of altitude, heading and speeds.

The cells representing each cluster and the corresponding diversion list are selected. To ensure good coverage of the relevant search space, n random samples are taken from each cell and fed to the SUT. The value of the cost functions proposed in Section 3.6 is then computed. Table 6.1 presents the maximum and mean value of some cost functions from the evaluation on the LIS-MUC route. The number of random samples, n, was set only to 3 to ensure a quick evaluation, since the cells are not very large. It is reminded that the SUT cannot be evaluated during departure nor arrival procedures yet (Chapter 5). Therefore, as decided before, it was evaluated only between reaching 5000 ft above the departure runway and until reaching 5000 ft above the destination runway. During the evaluation, all inputs and corresponding cost function values are stored. Therefore, scenarios can then be sorted by decreasing cost function values and the ones with the highest costs can be plotted and further analysed.

Table 6.1: Coverage-oriented evaluation of the SUT on the Lisbon to Munich route using n=3.

$mean(t)$	$max(t)$	$mean(f_1)$	$max(f_1)$	$mean(f_{SUT})$	$max(f_{SUT})$
0.0054 s	0.0783 s	1.4487	0.4252	35.6757	1.5262

From Table 6.1 it is possible to conclude that on an obstacle-free environment (only the terrain is to be avoided) the SUT performance is very good from an execution time point of view. When sorting the

trajectories produced by descending order of f_{SUT} value and analysing them it is clear that f_{SUT} is a poor cost function to search for sub-optimal trajectories produced by the SUT. In fact, when the aircraft is close to the target at high altitude, it needs to dissipate energy. In these cases, f_{SUT} has a high value even if the SUT produces optimal or near-optimal solutions. Analysing several solutions produced by the SUT, they all seem optimal from a length point of view, which is natural since no obstacles have been introduced. Should an alternative prototype be available, cost function g would be more suitable to guide a search for sub-optimal trajectories produced by the SUT. When sorting the trajectories by descending order of f_1 value and analysing both them and the trajectories that would be produced in case diversion option 1 was chosen by the SUT, it is possible to find some scenarios where the diversion selection performed by the SUT is not optimal. The problem seems to be that the SUT does not take into account the aircraft course when performing the diversion option selection, as Figure 6.11 illustrates.

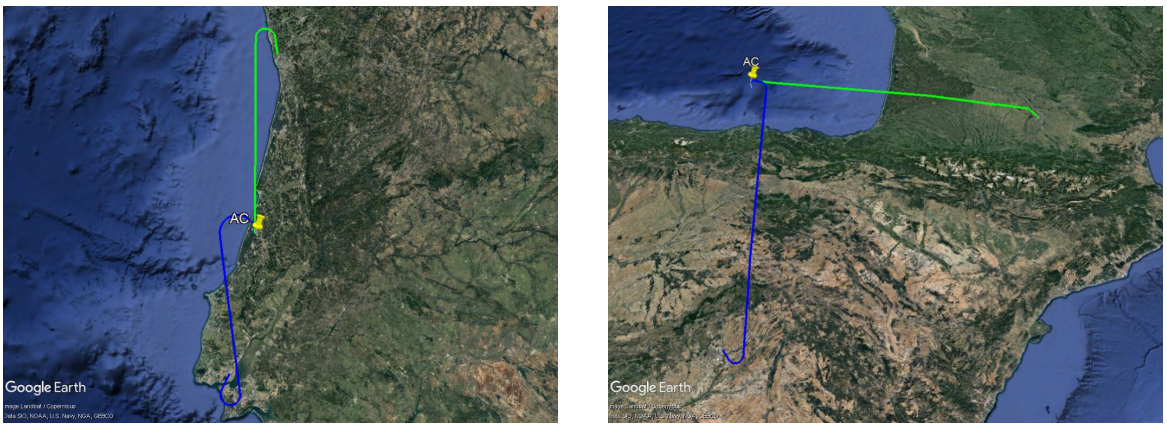


Figure 6.11: Scenarios where the diversion selection by the SUT is sub-optimal. The trajectory produced to reach the selected option (in blue) is longer than the one to reach diversion option 1 (in green).

Figure 6.11 shows a scenario where even though the diversion option chosen by the SUT is slightly closer to the aircraft than the alternative, the aircraft heading makes it so that the trajectory to reach the alternative is shorter. Furthermore, it is more intuitive to keep the course rather than turn back in an emergency procedure. This feedback was given to the design team.

It is worth remarking that a coverage-oriented strategy was proposed in this section. However, a temporal-oriented strategy could also easily be applied. It would suffice to sample from each cell with a probability proportional to the time aircraft historically spend on it. However, in this thesis the focus is not on the temporal mean performance but rather on ensuring ensuring that the SUT works as intended in all the space in which it will potentially be called. A coverage-oriented strategy is better-suited for this.

In case the performance was not as uniform and optimal everywhere, an interesting way to understand and be able to explain how the SUT works would be to identify cells with an associated high variance of the cost function values. These are cells where the behavior of the SUT changes. They can for instance contain the separation at which the SUT decides to go left of a mountain instead of going right of it. In the test performed they contain the separation between selecting an airport or another one.

Until this point, the SUT was evaluated only on nominal environment conditions. Section 6.3.2 introduces restricted areas and critical weather obstacles in the search for a challenging environment.

6.3.2 Challenging Environment Search

In this section, an algorithm is proposed to efficiently construct the most challenging environment for the SUT, for an aircraft state inside a given cell. The environment is built from the libraries of existing restricted areas and realistic critical weather obstacles presented in Sections 3.3 and 3.4 respectively.

In light of the bibliographic study presented in Section 2.1, an optimization problem seems suitable to tackle this problem. To define the aircraft and environment states, the optimization variable, $\mathbf{x}$, is defined by (6.9).

$$\mathbf{x} = \begin{bmatrix} x_{AC} & n_{restr} & n_{wx} & x_{wx} & \theta_{wx} \end{bmatrix}^\top \tag{6.9a}$$

$$x_{AC} = \begin{bmatrix} lat_{AC} & lon_{AC} & alt_{AC} & hdg_{AC} & spd_{AC} & vSpd_{AC} \end{bmatrix}^\top \tag{6.9b}$$

$$n_{restr} = \begin{bmatrix} nr_1 & nr_2 & nr_3 & nr_4 \end{bmatrix}^\top \tag{6.9c}$$

$$n_{wx} = \begin{bmatrix} nwx_1 & nwx_2 & nwx_3 & nwx_4 \end{bmatrix}^\top \tag{6.9d}$$

$$x_{wx} = \begin{bmatrix} lat_1 & lon_1 & lat_2 & lon_2 & lat_3 & lon_3 & lat_4 & lon_4 \end{bmatrix}^\top \tag{6.9e}$$

$$\theta_{wx} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^\top \tag{6.9f}$$

where x_{AC} defines the aircraft state (latitude, longitude, altitude, heading, speed, vertical speed), n_{restr} is an array of integers indicating which restricted areas are closed and therefore need to be avoided, n_{wx} is an array of integers indicating which critical weather obstacles from the weather obstacle library are selected to construct the weather environment, x_{wx} defines the location of these weather obstacles and θ_{wx} their orientation.

From these definitions it is clear that the environment can be composed by a maximum of 4 restricted areas and 4 critical weather obstacles. Filling the environment with many obstacles would obviously be more challenging for the SUT. However, it would be less realistic. Therefore, for the first tests and to validate the interest of the genetic algorithm, a maximum of 4 obstacles of each type seemed reasonable.

Each parameter of n_{restr} and n_{wx} is an integer between 0 and the number of restricted areas or critical weather obstacles respectively. A parameter set to 0 corresponds to not considering an obstacle. Since cells are hyper-cubes, ensuring that the aircraft state remains inside the cell being considered can easily be done using inequality constraints. As far as the angles in θ_{wx} are concerned, they are simply bounded between 0 and 360 degrees. At last comes the definition of the location of the selected weather obstacles, through x_{wx}. In order to avoid wasting computational budget placing weather obstacles completely out of the way between the aircraft and the diversion option chosen, constraints are defined to impose that the centres of the weather obstacles are placed inside a large geographical bounding box that surrounds both the aircraft and the diversion option selected by at least 3 degrees of latitude and longitude.

To tackle the optimization problem, a Genetic Algorithm (GA) approach was chosen due to its suc-

cess in the past for several applications. Inspired by Darwin's theory of evolution and natural selection, a GA is a method for solving both constrained and unconstrained optimization problems. It starts the generation of an initial population, either randomly or heuristically. A population is a group of individuals, also called chromosomes. The latter are solutions to the problem and they are composed by a string of genes, each gene being a parameter or variable of the problem.

An iterative process then makes the initial population evolve using several evolution-theory-inspired operators, favoring the survival of the fittest. On each iteration, each individual of the population is evaluated through the computation of its fitness function. The fitness function can be seen as the score of an individual, and the probability that it is selected for reproduction is based on it. In the selection phase, individuals are selected based on their fitness scores, with the probability of an individual being chosen being higher the fitter the individual. The new population is then generated from the selected individuals, thanks to crossover and mutation operations. Crossover is a very important re-combination operator, taking two parent individuals and swapping some of their genes to produce two children. Crossover therefore favors exploitation, biasing the search towards promising regions of the search space. Mutation favors more exploration, by randomly sampling new points from the search space. Hence preventing premature convergence to local optima. The algorithm terminates either when the computational budget has been reached or when the population has converged - when the children produced do not significantly differ from the previous few generations. The algorithm provides a set of solutions. The pseudo-code of a typical genetic algorithm just described is presented in Algorithm 3.

Algorithm 3 Genetic Algorithm

```
1:  procedure GENETIC ALGORITHM
2:      Generate initial population
3:      while computational budget available AND no population convergence do
4:          Fitness computation
5:          Selection
6:          Crossover
7:          Mutation
8:      end while
9:      return population
10: end procedure
```

Genetic algorithms have several advantages that make it suitable for the problem treated in this thesis. To begin with, they allow performing global optimization of black-box functions (a requirement in this thesis), since the search does not require local gradient information, but only the ability to compute fitness values for given inputs. The function dealt with does not need to be continuous as long as it can be computed. Another great advantage lies on the search being performed from a population, rather than just one point. The parallelism prevents the algorithm from becoming trapped on locally optimal solutions (especially if a measure of diversity is incorporated into the algorithm, so that if one individual is trapped on a local maxima the others will search different regions), as it could happen with hill-climbing techniques. The parallelism can also be used to speed up the search. Additionally, crossover seems to be a very interesting way of exploring and combining similarities among high-performance candidates. In the use case of this thesis, for instance, it seems intuitive that bringing together a challenging disposition

of critical weather and a challenging disposition of restricted areas results in a globally more challenging environment than the parent ones. At last, it is worth mentioning the ability of the algorithm to easily deal with various complex optimization problems (stationary or non-stationary, continuous or discontinuous, linear or non-linear objective functions, and with or without random noise).

Genetic Algorithms have already been successfully used in the past for several different applications, from software testing to controllers design, and from discrete systems to continuous ones. Hence its choice to tackle the optimization problem of finding the maxima of the cost functions of interest. However, it is important to remark that there are also some disadvantages. The main risk is not choosing a good population size or some important parameters such as the types and rates of mutation and crossover, as well as the selection criteria of the new population, which could make it difficult for the algorithm to converge or produce meaningless results. Despite these drawbacks, genetic algorithms remain one of the most widely used optimization algorithms in nonlinear optimization.

Concerning the implementation, Matlab's Global Optimization Toolbox [54] was used. Since it minimizes the fitness, the negative of the cost functions to be maximized were used. The Matlab implementation of GA is prepared to deal with both mixed integer and constrained optimization.

A cell from a cluster from the LIS-MUC route was used as a use case to validate the interest of the approach proposed. Regardless of the cost function that is to be maximized, finding a case where there is a system error and nothing is produced is the best one can hope for. Therefore, the fitness is set to $-\infty$ when a system error is produced and the search is terminated. Some falsifying inputs have been found, one of which is represented in Figure 6.12.

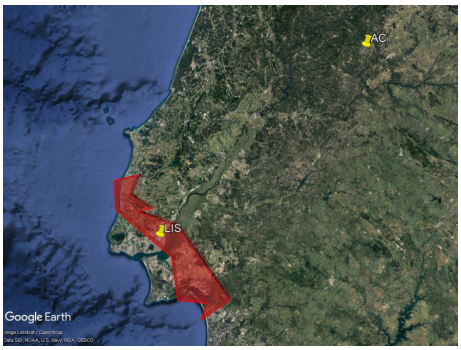


Figure 6.12: Falsifying inputs. The SUT does not check whether or not the diversion options are safe with respect to critical weather obstacles and selects a diversion option that is not possible to reach.

Figure 6.12 illustrates a scenario where the weather obstacle is placed above the airport. The system can therefore not generate a safe trajectory to it, for which reason it should have selected a different airport. The design team was informed of this problem. To prevent the system from always finding this type of falsifying scenario, the cost function was set to 0 whenever either the aircraft or the diversion option were not safe with regards to the obstacles. Running once again the search, challenging scenarios are quickly and successfully found. Figure 6.13 presents the scenarios found when searching for the maximum execution time and maximum f_{SUT} cost function values from a given aircraft cell.

As expected, the cost functions are maximized when the selected restricted areas and weather

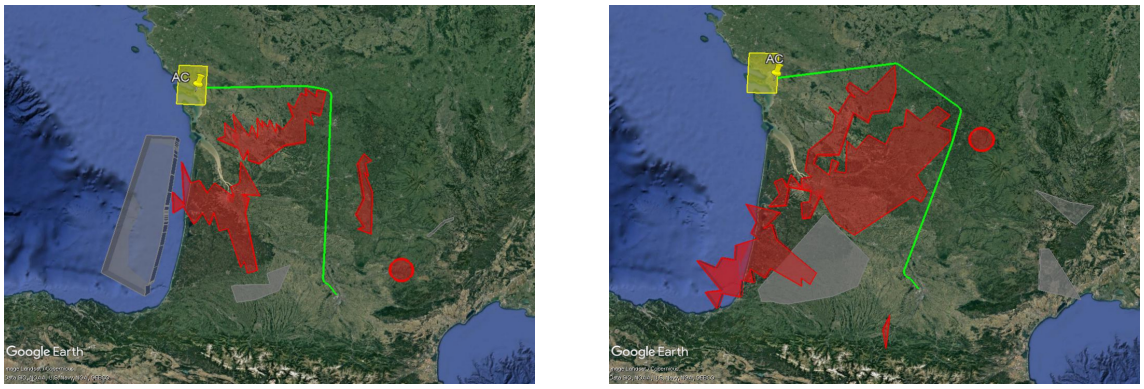


Figure 6.13: Most challenging scenarios found when searching for the maximum execution time (left image) and maximum f_{SUT} cost function value (right image) from a given aircraft cell (in yellow). Weather obstacles are represented in red and restricted areas in grey.

obstacles are placed between the aircraft and the diversion option, obliging the trajectory to go around these. Additionally, the orientation of the obstacle is such that most concavities are facing the aircraft. The design team was informed of this to support future optimizations of the algorithm.

6.4 Worst Scenario Search

This section focuses on the search for the worst scenarios for the SUT, as far as both the diversion selection and the trajectory generation algorithms are concerned.

The same genetic algorithm that was presented in Section 6.3.2 as suitable to search for scenarios maximizing the cost functions of interest when the aircraft is restricted to a cell can also be used to find the worst scenario given a fixed diversion option. In this case, the aircraft can be placed anywhere within a distance threshold $d_{threshold}$ of the airport, which is used to set the constraints to the aircraft position. Figure 6.14 presents a scenario successfully constructed to maximize the execution time when the diversion option is LFBO.

Figure 6.14 shows that dealing with a restricted area defined by many vertices directly facing the aircraft is very challenging for the SUT. The design team was informed of this difficulty.

The algorithm is used as well to find the aircraft state and environment configuration such that the diversion option selection is the worst possible given a list of diversion options. Using a list containing LPPT (Lisbon), LPPR (Porto), LEMD (Madrid), LFBO (Toulouse), LFML (Marseille) and EDDM (Munich), challenging scenarios are quickly and successfully constructed. Figure 6.15 shows one of them.

From Figure 6.15 it seems like the environment between the aircraft and the airport is not taken into account by the SUT when selecting the most suitable diversion option. As a result, it ends up choosing a diversion option that forces a much longer emergency procedure than needed. The design team was informed of this problem with the selection of the diversion target.

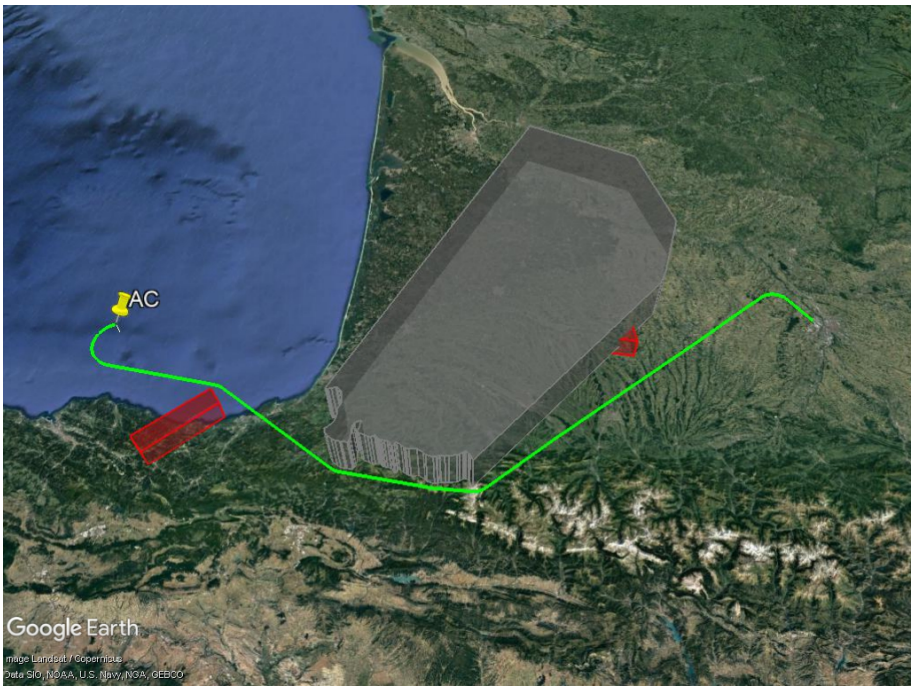


Figure 6.14: Scenario found by a GA when searching for the most challenging initial conditions assuming the diversion is performed to LFBO. It took the SUT 80.56 seconds to compute the trajectory.

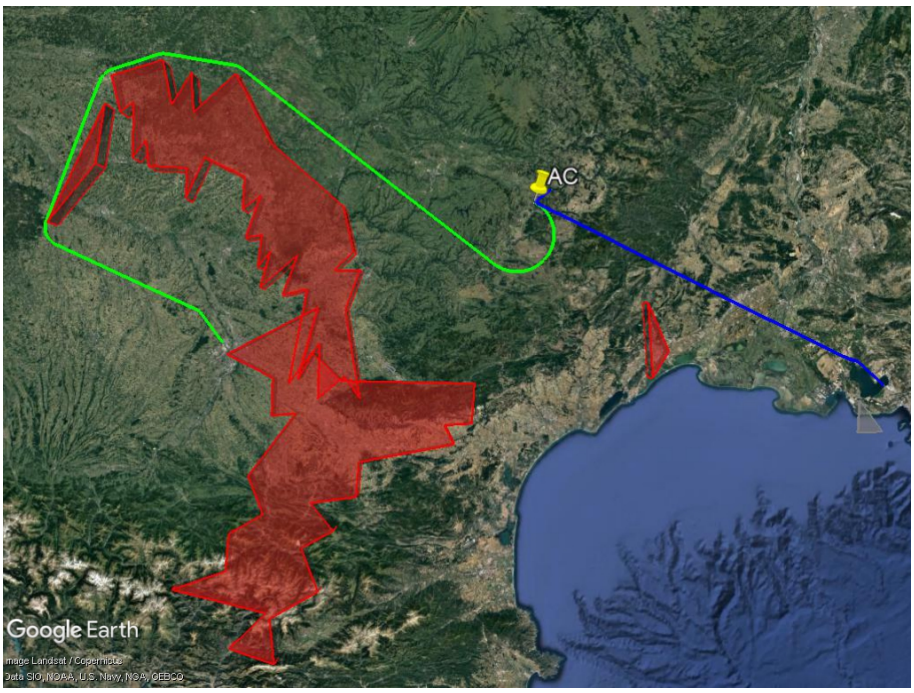


Figure 6.15: Scenario found by a genetic algorithm where the diversion selection performed is not suitable. The trajectory produced by the SUT (in green) to reach LFBO (the SUT choice) is 2.53 times longer than the one produced by the SUT to reach LFML (in blue).

Chapter 7

Conclusions and Future Work

In this chapter, Section 7.1 reviews the main work developed throughout the thesis, outlining the most important contributions and results obtained. Section 7.2 presents interesting future work.

7.1 Achievements

This thesis proposes a complete framework to support both the development and validation cycles of black-box systems whose mission is to automatically and in real-time select from a list of diversion options the most suitable one and generate a safe and flyable trajectory until it to save aircraft in case of emergency.

The thesis therefore provides important contributions from an industrial point of view. In fact, there is a strong interest in the development and validation of functions contributing to the automation of aircraft or assistance to pilots. Since aviation is a safety critical domain and most of these functions use artificial intelligence algorithms that are too difficult to validate using traditional formal methods, the development of approaches to evaluate black-box systems is fundamental. Simultaneously, such a task is very interesting from an academic point of view.

A prototype under development served as a use case to test the performance of the framework proposed. The framework has successfully found several axis of improvement for the SUT, revealing itself very useful to support the design and development processes and worth further improving in future work.

Data-driven strategies were suggested whenever pertinent to support the generation of representative test scenarios to evaluate the SUT. After filtering out erroneous or irrelevant data, historical recordings from FlightRadar24 were used to reconstruct complete and smooth aircraft trajectories. Trajectory reconstruction was formulated as a convex optimization problem, where the use of penalty terms thanks to the acceleration and jerk operators allows to penalize not only the position errors but also unrealistic aircraft dynamics. The parameters were tuned using an out-of-sample validation strategy, ensuring that an over fitting is not performed. The strategy was implemented and showed very good performance.

A data-driven strategy was proposed to evaluate the SUT's internal representation of the terrain.

This representation should be as simple and light as possible but while ensuring that all areas where aircraft may fly are judged as safe. Instead of retrieving information concerning all different existing procedures and typical ATC vectoring orders, the proposal is to simply verify whether or not all the relevant measurements and trajectories reconstructed from FlightRadar24 are judged as safe by the terrain representation under test. The strategy was implemented and allowed to conclude that the terrain representation clearly needs to be improved close to airports, since there are many positions belonging to departure and arrival procedures judged as not safe. The SUT could therefore be tested in the context of this thesis only once the aircraft surpasses 5000 ft above the departure runway and until it reaches 5000 ft above the arrival runway. Thanks to the parallelization of the tests, which is possible thanks to the PySpark implementation put in place, the framework allows to obtain feedback very quickly, for which reason it will be an important asset to the development team to evaluate terrain representation evolutions until it becomes precise enough to be validated.

The SUT's performance as far as the diversion option selection and trajectory generation are concerned was also evaluated thanks to a data-driven strategy. The strategy designed relies on the existence of an algorithm capable of identifying groups of similar trajectories. For doing this, following a bibliographic study of trajectory clustering methods, two algorithms were implemented: HDBSCAN and Gaussian Mixture Models. Using the Lisbon to Munich route as a use case, HDBSCAN showed great performance, unlike GMM. Additionally, HDBSCAN has the advantage of discovering clusters of arbitrary shape and not requiring *a priori* knowledge of the number of clusters. This is important to make the pipeline of the framework proposed in this document as automatic as possible. HDBSCAN was therefore selected to perform the clustering of trajectories from a given route.

For each cluster identified, the list of diversion options to be fed to the SUT is constructed in two phases. At first, all historical diversion that took place while flying the route under analysis in the past are retrieved from FlightRadar24 and plotted for analysis. The airports selected to perform diversion in the past have been chosen by the pilots, airlines OCC or ATC. Whichever the reason, since they were chosen they have characteristics that make them suitable for diversion. Therefore, they are introduced in the diversion list of a cluster whenever appropriate. In a second phase, it must be ensured that from anywhere inside the cluster there is always at least one airport from the diversion list that is at a maximum flying distance of 1 hour in case of emergency. To ensure this while constructing a list containing the least number of airports possible, an ILP problem was formulated to select the airports from a list and solved thanks to the PuLp library. The list of airports was obtained by keeping from an airport database all the options with associated ILS CAT II or III capabilities, to ensure robustness to low-visibility conditions.

In order to guide and quantify the evaluation of the SUT, several performance metrics and cost functions were designed with the aim of finding the worst execution time, bad diversion option selections or yet non-optimal trajectories generated by the system. Some cost functions are also proposed to compare different algorithms or different versions of the system as far as these criteria are concerned, which is interesting to support the development process and ensure there are no regressions between versions. To evaluate the performance of the SUT on a cluster under analysis, the cluster is represented

by a group of cells and scenarios are sampled from each cell to ensure cluster coverage. For cruise in the Lisbon to Munich route under nominal environment conditions the system is clearly validated. Not only does it produce trajectories very quickly but also these are optimal or near optimal. However, an axis of improvement was found. In some cases where the aircraft is approximately between two airports the diversion selection is not optimal due to the aircraft heading favoring the solution not chosen by the system. The design team was informed of this problem.

A genetic algorithm was then proposed as an efficient strategy to search for the most challenging scenarios for the SUT, including the construction of a challenging environment. Always searching for scenarios as representative as possible, the shapes of real restricted areas from Spain and France were retrieved. The genetic algorithm then selects which areas are to be considered by the system. Concerning critical weather, a library of critical weather obstacle shapes was constructed. To generate scenarios, the genetic algorithm selects shapes from this library and then places them and defines their orientation. The genetic algorithm implemented has allowed to successfully identify some system bugs (of which the design team was informed), as well as the scenarios that are most challenging and dimension the SUT use.

Since the framework has already provided valuable feedback and shown itself to be a valuable asset to support both the development and validation processes, it will likely continue being used in the future after the conclusion of this thesis. In light of the academic and industrial interest of further improving the framework that was put in place in this master thesis, the last contribution of this document consists of relevant future work suggestions that are presented in Section 7.2.

7.2 Future Work

Although the framework designed and implemented has already proven itself useful for evaluating the SUT and identifying axis of improvement, some work remains to be done to improve the framework as much as possible.

Once the SUT terrain representation becomes mature enough to accept typical departure and arrival procedures as safe, it will be necessary to verify if these procedures are also accepted when taking into account the corresponding Required Navigation Performances and altitude errors. It will be necessary to be able to identify the RNP value corresponding to the procedure being analysed, which is not trivial.

Improving the automatic construction of the diversion list is yet another stream of future work. Although the ILP problem formulation proposed in this document already ensures that the minimum number of options is selected to ensure that an airport is always available within 1 hour of flight, there are often several possible solutions. In such cases, it would be suitable to choose the best solution accordingly to other relevant criteria such as the airport capacity for instance.

As far as the cost functions and performance metrics driving the search for interesting scenarios are concerned, it is interesting to both improve the ones suggested in this document and design new pertinent ones in the future. Likewise, concerning the search for the most challenging scenarios, the standard genetic algorithm from Matlab's toolbox was used until now. Improving it by designing problem-

specific crossover and mutation operators, as well as tuning values of parameters such as the population size, might significantly improve its performance. It is also very important to validate the framework's capacity to find interesting scenarios in a variety of new routes other than the Lisbon to Munich use case. Additionally, alternative black-box validation algorithms could be implemented and their performance compared with that of the genetic algorithm.

Although the strategy suggested for the environment construction allows for the construction of very representative restricted areas and weather obstacles, it would be interesting to introduce parameters in the search space allowing to define the obstacles shape in the search for the most challenging environments for the SUT. Regardless of their likelihood, finding scenarios that the SUT has trouble dealing with would allow to identify its weaknesses and provide important feedback for its improvement.

An important characteristic of the tests performed until now is that only the initial conditions were considered. For example, a configuration of critical weather obstacles was created and it was verified whether or not the system could find a safe trajectory avoiding them. This validates the system's capacity to find a solution satisfying the initial conditions of the problem. However, in reality, there are disturbances and so the dynamic environment not always evolves as predicted. For instance, there may be sensor measurement errors, or the weather may evolve unexpectedly. Therefore, not only does the system need to be able to find a solution for given initial conditions but also it must be able to react to a dynamic environment. Once the system is mature and validated for static conditions, it is necessary to perform a search for a sequence of disturbances that leads to a situation where the trajectory initially computed can no longer be accepted while it is being flied and neither can the system compute a safe alternative trajectory. For the search of such a sequence of disturbances, implementing RL algorithms like MCTS for an AST strategy would be very interesting.

This thesis mainly focused on the search for falsifying or challenging inputs for the system. In future work, as the system becomes more mature and closer to its final version, it will be important to understand which of these scenarios are more likely to take place. It will also be important to quantify it by computing the probability of a system error occurring. As the system becomes more mature and falsifying inputs become increasingly difficult to find, choosing a relevant coverage metric and computing it is also fundamental to be confident about the system's capabilities.

At last, it is also important to better understand how the system works. For this, an approach that seems especially interesting consists of focusing the search on finding the boundaries between different performance modes instead of wasting resources exploring regions of stable performance. Analysing why on each side of these boundaries the choices made by system are different helps understand how the system works and build confidence in its decision process. Another interesting approach would be to cluster the trajectories produced by system and then manually analysing only a few trajectories representative of each cluster, as well as outliers. Implementing anomaly detection methods such as neural network auto-encoders to choose which trajectories produced by the system should be analysed is also suggested.

Bibliography

[1] International Air Transport Association. IATA Forecast Predicts 8.2 billion Air Travelers in 2037. *Press Releases*, 62, October 2018. https://www.iata.org/en/pressroom/pr/2018-10-24-02. [Last Accessed: October 31, 2021].

[2] International Air Transport Association. Recovery Delayed as International Travel Remains Locked Down. *Press Releases*, 63, July 2020. https://www.iata.org/en/pressroom/pr/2020-07-28-02/. [Last Accessed: October 31, 2021].

[3] Karl D. Bilimoria, Walter W. Johnson, Paul C. Schutte. Conceptual Framework for Single Pilot Operations. *International Conference on Human-Computer Interaction in Aerospace*, 4:1–8, July 2014. doi: 10.1145/2669592.2669647.

[4] Arno Fallast, Bernd Messnarz. Automated trajectory generation and airport selection for an emergency landing procedure of a CS23 aircraft. *CEAS Aeronautical Journal*, 8:481–492, June 2017. https://doi.org/10.1007/s13272-017-0252-5. [Last Accessed: October 31, 2021].

[5] Nicolas Meuleau, Christian Plaunt, David E. Smith, Tristan Smith. An Emergency Landing Planner for Damaged Aircraft. *Twenty-First Conference on Innovative Applications of Artificial Intelligence*, July 2009.

[6] Anthony Corso, Robert J. Moss, Mark Koren, Ritchie Lee, Mykel J. Kochenderfer. A Survey of Algorithms for Black-Box Safety Validation. *Cornell University, Computer Science, Machine Learning*. https://arxiv.org/abs/2005.02979v2. [Last Accessed: October 31, 2021].

[7] Anthony L. Corso. Safety Validation of Black-Box Autonomous Systems. August 2020. http://ai.stanford.edu/blog/black-box-safety-validation/. [Last Accessed: October 31, 2021].

[8] Qianchuan Zhao, Bruce H. Krogh, Paul Hubbard. Generating test inputs for embedded control systems. *IEEE Control Systems Magazine*, 23(4):49–57, August 2003. doi: 10.1109/MCS.2003.1213603.

[9] Zhenya Zhang, Gidon Ernst, Sean Sedwards, Paolo Arcaini, Ichiro Hasuo. Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search. *Cornell University, Computer Science, Systems and Control*, August 2018. https://arxiv.org/abs/1803.06276. [Last Accessed: October 31, 2021].

[10] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. *Machine Learning: ECML 2006, 17th European Conference on Machine Learning*, pages 282–293, September 2006. https://doi.org/10.1007/11871842_29. [Last Accessed: October 31, 2021].

[11] Peter Auer, Nicolò Cesa-Bianchi, Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, May 2002. https://doi.org/10.1023/A:1013689704352. [Last Accessed: October 31, 2021].

[12] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat and M. P. Owen. Adaptive stress testing of airborne collision avoidance systems. *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 6C2–1–6C2–13, 2015. doi: 10.1109/DASC.2015.7311450.

[13] C. B. Browne et al. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012. doi: 10.1109/TCIAIG.2012.2186810.

[14] Georgios E.Fainekos, George J.Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, September 2009. https://doi.org/10.1016/j.tcs.2009.06.021. [Last Accessed: October 31, 2021].

[15] L. Mathesen, S. Yaghoubi, G. Pedrielli and G. Fainekos. Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart. *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 991–997, 2019. doi: 10.1109/COASE.2019.8843005.

[16] M. Koschi, C. Pek, S. Maierhofer and M. Althoff. Computationally efficient safety falsification of adaptive cruise control systems. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2879–2886, 2019. doi: 10.1109/ITSC.2019.8917287.

[17] Mark Koren, Saud Alsaif, Ritchie Lee, Mykel J. Kochenderfer. Adaptive stress testing for autonomous vehicles. *Cornell University, Computer Science, Robotics*, February 2019. https://arxiv.org/abs/1902.01909. [Last Accessed: October 31, 2021].

[18] G. E. Mullins, P. G. Stankiewicz and S. K. Gupta. Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1443–1450, 2017. doi: 10.1109/ICRA.2017.7989173.

[19] J. Sander M. Ester, H.-P. Kriegel and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, August 1996.

[20] Australian Government - Civil Aviation Safety Authority. Performance-based navigation. https://www.casa.gov.au/book-page/chapter-6-performance-based-navigation. [Last Accessed: October 31, 2021].

[21] SKYbrary. Area Navigation Systems. https://www.skybrary.aero/index.php/Area_Navigation_Systems. [Last Accessed: October 31, 2021].

[22] SKYbrary. Required Navigation Performance (RNP). https://www.skybrary.aero/index.php/Required_Navigation_Performance_(RNP). [Last Accessed: October 31, 2021].

[23] International Civil Aviation Organization. Performance-based Navigation Manual. Advance 4th ed., 2012.

[24] SKYbrary. Instrument Landing System (ILS). https://www.skybrary.aero/index.php/Instrument_Landing_System_(ILS). [Last Accessed: October 31, 2021].

[25] SKYbrary. Decision Altitude/Height (DA/DH). https://www.skybrary.aero/index.php/Decision_Altitude/Height_(DA/DH). [Last Accessed: October 31, 2021].

[26] SKYbrary. Runway Visual Range (RVR). https://www.skybrary.aero/index.php/Runway_Visual_Range_(RVR). [Last Accessed: October 31, 2021].

[27] Airbus. Getting to grips with CAT II / CAT III operations. *Flight Operations Support Line Assistance*, October 2001.

[28] S. T. Barratt, M. J. Kochenderfer and S. P. Boyd. Learning Probabilistic Trajectory Models of Aircraft in Terminal Airspace From Position Data. *IEEE Transactions on Intelligent Transportation Systems*, 20(9):3536–3545, sep 2019. doi: 10.1109/TITS.2018.2877572.

[29] António Fallah. Aircraft trajectory prediction in crowded terminal areas. Master's thesis, Instituto Superior Técnico, October 2019.

[30] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics, Ann. Math. Statist.*, 35(1):73–101, mar 1964. doi: 10.1214/aoms/1177703732.

[31] Jiawei Han, Jian Pei, Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann Publishers, San Francisco, 3rd edition, 2011.

[32] Christopher M. Bishop. Pattern Recognition and Machine Learning. *Sprinter-Verlag*, 2006.

[33] Yuan, G., Sun, P., Zhao, J. et al. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47:123–144, January 2017. doi: 10.1007/s10462-016-9477-7.

[34] LumenAi. Time series aggregation. https://www.lumenai.fr/laboratoire/travaux-en-cours/time-series-aggregation/. [Last Accessed: October 31, 2021].

[35] S. Hong and K. Lee. Trajectory prediction for vectored area navigation arrivals. *Journal of Aerospace Information Systems*, March 2015. https://doi.org/10.2514/1.I010245. [Last Accessed: October 31, 2021].

[36] M. Gariel, A. N. Srivastava and E. Feron. Trajectory Clustering and an Application to Airspace Monitoring. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1511–1524, December 2011. doi: 10.1109/TITS.2011.2160628.

[37] NASA. Shuttle Radar Topography Mission. https://www2.jpl.nasa.gov/srtm/. [Last Accessed: October 31, 2021].

[38] SKYbrary. Aeronautical Information Publications (AIPs). https://www.skybrary.aero/index.php/Aeronautical_Information_Publications_(AIPs). [Last Accessed: October 31, 2021].

[39] SIA. Charts. https://www.sia.aviation-civile.gouv.fr/. [Last Accessed: October 31, 2021].

[40] SKYbrary. Cumulonimbus (Cb). https://www.skybrary.aero/index.php/Cumulonimbus_(Cb). [Last Accessed: October 31, 2021].

[41] Christian Norden David Marconnet and Laurent Vidal. *Optimum use of weather radar*. Airbus - Safety First, 1/23 edition, 2016.

[42] Airbus. Skywise. https://skywise.airbus.com/. [Last Accessed: October 31, 2021].

[43] Apache Spark. https://spark.apache.org/. [Last Accessed: October 31, 2021].

[44] Apache Spark. Cluster Mode Overview. https://spark.apache.org/docs/latest/cluster-overview.html. [Last Accessed: October 31, 2021].

[45] Campello R.J.G.B., Moulavi D., Sander J. Density-based clustering based on hierarchical density estimates. *PAKDD 2013: Advances in Knowledge Discovery and Data Mining*.

[46] Kriegel HP Sander J Ankerst M, Breunig MM. OPTICS: Ordering Points to Identify the Clustering Structure. *Conference: SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data*, 28(2):49–60, June 1999. doi: 10.1145/304182.304187.

[47] 2021]. PyPi. hdbscan. https://pypi.org/project/hdbscan/. [Last Accessed: October 31.

[48] Leland McInnes, John Healy and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11). doi: 10.21105/joss.00205.

[49] Hirotugu Akaike. Information theory and an extension of the maximum likelihood principle. *Second International Symposium on Information Theory*, pages 267–281, 1973.

[50] Gideon Schwarz. Estimating the Dimension of a Model. *Ann. Statist.*, 6(2):461 – 464, mar 1978. doi: 10.1214/aos/1176344136.

[51] sklearn. Gaussian Mixture. https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html. [Last Accessed: October 31, 2021].

[52] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.

[53] PyPi. PuLP. https://pypi.org/project/PuLP/. [Last Accessed: October 31, 2021].

[54] Matlab. Global Optimization Toolbox - Genetic Algorithm. https://www.mathworks.com/discovery/genetic-algorithm.html. [Last Accessed: October 31, 2021].