FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Data Collection and Analysis in a Distributed Simulation Platform

Tiago Rodrigues Vieira de Carvalho

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Infomática e Computação

Supervisor: Daniel Augusto Gama de Castro Silva

Co-Supervisor: Leonardo da Silva Ferreira

July 23, 2020

Data Collection and Analysis in a Distributed Simulation Platform

Tiago Rodrigues Vieira de Carvalho

Mestrado Integrado em Engenharia Infomática e Computação

Approved in oral examination by the committee:

July 23, 2020

Abstract

Nowadays vehicles have an important role in our lifestyles as they are useful in many ways, not only for transporting people over large distances, but also for other purposes, such as firefighting, surveillance, wars, and many others. On that premise, software capable of simulating these diverse uses and evaluate their performance is always welcome. With that in mind, the Artificial Intelligence and Computer Science Laboratory developed a platform with the goal of simulating realistic autonomous multi-vehicle missions with a high degree of flexibility, by allowing the user to thoroughly describe the scenario, rules and objectives of the mission. Although many similar simulators have been developed, most of them are focused either on vehicle piloting or traffic management. The simulator to be used provides support for the execution of several cooperative tasks, such as detection of an anomaly (such as a fire), following a vehicle, measuring and mapping air quality parameters, finding the source of an anomaly (for instance, a pollutant), or transporting goods from one point to another.

The goal of this project is to evaluate the mission performance and with it discover the best approach for the different missions the Platform is capable to reproduce. In order to achieve this goal, multiple metrics were created to have some indicators that can be used to calculate the missions performance. Those metrics were separated in relative and absolute metrics, depending on whether they represent performance on their own or need to be compared with other similar missions.

Various simulations were conducted, with different strategies and parameters, in order to compare the various approaches to the various missions. These experiments consisted in applying different search patterns in a search mission with distinct delimitation areas and parameters. The results obtained allow the user to easily adopt the most efficient strategy in the scope of the simulations executed. For example, the bigger the ratio between height and width of the delimitation area, the better the lawnmower pattern is compared to the others. On the other hand the closer the area is to a circle, the better the spiral search pattern is compared to the others.

Keywords: Simulation evaluation. Metrics. Aircraft missions. Search patterns.

Wisdom is
the offspring of
Suffering and Time.

Izaro

Contents

1 Introduction **1**
1.1 Context and Motivation . . . 1
1.2 Objectives . . . 2
1.3 Methodology and Expected Results . . . 2
1.4 Document Structure . . . 3

2 Contextualization **5**
2.1 Platform Description . . . 5
2.2 Missions . . . 8
2.2.1 Mission Description Language . . . 8
2.2.2 Types of Missions . . . 9
2.2.3 Mission Flow . . . 11

3 Related Work **13**
3.1 Similar Platforms . . . 13
3.1.1 AGENTFLY . . . 13
3.1.2 MOOS-IvP . . . 14
3.1.3 Other Similar Platforms . . . 17
3.2 Similar Works . . . 19
3.2.1 Performance Evaluation in Air Traffic . . . 20
3.2.2 Performance Evaluation for Similar Tasks . . . 20

4 Preliminary Work **23**
4.1 Mission Management . . . 23
4.2 Search Patterns . . . 24
4.2.1 Lawnmower . . . 24
4.2.2 Spiral In/Out . . . 26
4.2.3 Follow Shape In/Out . . . 27
4.3 Follow Mission . . . 28
4.4 Route Missions . . . 29
4.5 Patterns Shortcomings . . . 30
4.5.1 Approximate Minimum Turn Radius . . . 30
4.5.2 Mistakes in Projection . . . 30

5 Data Collection **33**
5.1 Metrics . . . 33
5.1.1 Absolute Metrics . . . 34
5.1.2 Relative Metrics . . . 34

5.1.3 Relational Values 35
5.1.4 Unfeasible Metrics 35
5.2 Logs 37
5.3 Experiments 40

6 Results **43**
6.1 Pattern Efficiency 43
6.1.1 Lawnmower Pattern 43
6.1.2 Follow Shape In/Out Pattern 45
6.1.3 Spiral In/Out Pattern 47
6.2 Pattern Comparison 48

7 Conclusion and Future Work **53**

References **55**

List of Figures

2.1 Platform General Architecture 6
2.2 Control Panel Mission Tab 7
2.3 Mission Element Definition 8
2.4 Phase Element Definition 9
2.5 Target Element Definition 10

3.1 Conflict scenario result using real UAVs 14
3.2 MOOS-IvP architecture 15
3.3 MOOS-IvP pMarineViewer 16
3.4 Path plannings considered to map the bottom of the ocean 17
3.5 Representation of the capture the flag environment in MOOS-IvP 17
3.6 Snapshot of AirSim Simulation 18
3.7 Four patterns developed in [Arbeit, 2013] 19
3.8 Experiments using two different Sensor Control Algorithms in a Creeping Line search pattern 20
3.9 Area coverage results using a Spiral Inward and Spiral Shift 21

4.1 Representation of the waypoints calculated in a lawnmower search pattern 25
4.2 Possible turns in a lawnmower search pattern 25
4.3 Lawnmower pattern in a square 26
4.4 Spiral In pattern in a square 27
4.5 Follow Shape In pattern in a square 28
4.6 Explanation for follow mission formula 29

5.1 Rectangles made in order to calculate the area covered and overlapped 40
5.2 Different Areas used in the experiments 41
5.3 Cessna Skyhawk (left) and Beechcraft Baron(right) models in FSX 42

6.1 Distance Efficiency in a Lawnmower search pattern 44
6.2 Different parameters in a Lawnmower pattern in a regular polygon shaped area . 45
6.3 Area covered and overlapped in a Lawnmower search pattern 45
6.4 Different aircrafts performing the Follow Shape In pattern in the flatten area . . . 46
6.5 Area covered and overlapped in a Follow Shape search Pattern 47
6.6 Some examples of the Follow Shape pattern (separation of 1000 meters in the left and 2000 in the right) 47
6.7 Area covered in a Spiral search pattern 48
6.8 Spiral Out experiments with less and highest area coverage 48
6.9 Time Spent on Different Patterns in a Squared Area in seconds 49
6.10 Time Spent on Different Patterns in a Pentagon Shaped Area in seconds 50

6.11 Time Spent on Different Patterns in a Flatten Area in seconds 50
6.12 Time Spent on Different Patterns in a Small Area in seconds 51
6.13 Time Spent on Different Patterns in a Circle Area in seconds 51

List of Tables

4.1 Projections 31

5.1 Absolute Metrics 34
5.2 Relative Metrics 35
5.3 Relational Values 35
5.4 Fuel Simulation Variables 36
5.5 Target and Phase Related Metrics 37
5.6 Logs Required to Calculate the Metrics 38
5.7 Aircraft Specs 42

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ATC	Air Traffic Controller
ATM	Air Traffic Management
CSV	Comma Separated Values
FEUP	Faculdade de Engenharia da Universidade do Porto
FSX	Flight Simulator X
GPC	General Polygon Clipper
LIACC	Artificial Intelligence and Computer Science Laboratory
MDL	Mission Description Language
MOOS-IvP	Mission Oriented Operating Suite - Interval Programming
RRT	Rapid exploring Random Tree
SOTA	State Of The Art
TRACON	Terminal Radar Approach CONtrol
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
USA	United States of America
XML	eXtensible Markup Language

Chapter 1

Introduction

This section represents an overview of the work developed. First, it has a brief introduction to the context and motivation of the project. Secondly, it details the goals intended to be achieved in it. Thirdly, it is described the approach and expected results of this work. Lastly, the general report structure is explained.

1.1 Context and Motivation

Recently, the number of remotely operated vehicles has been constantly increasing due to their utility in diverse missions [DoD, 2005] [SESAR, 2016]. However, the manning of said vehicles implies various problems since they need to be operated by humans. Humans are prone to mistakes, which can be very costly, as they, in some cases, can be the cause of various accidents [Williams, 2004].

In this regard, trying to use real vehicles in order to test the effectiveness of different operations and missions is unfeasible due to how expensive they are [Incorporated, 2004]. This created the need for various simulations for experiments and human training. The arrival of unmanned vehicles also came to increment the need for simulators capable of handling them since they allowed the existence of fully automated missions, as can be seen in section 3.

That being said, a platform capable of simulating diverse missions with a team of different vehicles is being developed in LIACC [Silva, 2011]. The platform allows the simulation of various missions with an enormous degree of freedom, as it enables the user to simulate whatever mission it wants inside of the possible existing types, detailed in section 2.2.2. It is useful to conclude whether a certain task is going to be successful or not before using real vehicles, as well as trying to check which approach is more successful for said task. However, the platform doesn't have a way to describe the performance of a particular mission, being the only way to check its performance the visualization, in real-time, of its execution.

Obtaining the performance of a certain task is also not the only benefit of calculating metrics. They can be useful to discover how certain parameters such as aircrafts used and search areas shape can affect the mission. With this, it's possible to not only know whether or not the task was efficient or not, but also have some insight on how to improve it by changing the possible parameters.

1.2 Objectives

This project has the main goal of discovering which strategy is overall the best for each type of mission and correlate parameters with strategies. It seeks to answer questions like which is the most efficient strategy to perform a mission with certain parameters? How does the shape of the area to search affects the performance of the mission, and what can we do to improve it? How does the aircraft speed and minimum turn radius impact efficiency? Does the best strategy for a certain mission change depending on the aircraft used?

All these questions have the intent of understanding what affects mission performance, and by doing so, understanding how to improve it. That is the objective of this work, to create metrics capable of answering these questions, and uncover the relations between all mission parameters. This work is also useful for future work in the platform, such as comparing new implementations with the previous ones, or discover if something went wrong with the simulation. It also might uncover possible flaws in the strategies used.

1.3 Methodology and Expected Results

In order to achieve the results intended in this work, the methodology of this project started by an extensive familiarization with the Platform in order to know what was or wasn't already implemented and confirm the ground work needed to be done before the creation of the metrics. Through this familiarization it was discovered that the mission logic in the platform was severely lacking in many aspects.

That being said, a research of the state of the art in mission oriented vehicle simulations and performance evaluation was done in order to identify the most common strategies and metrics used. This knowledge was the base for the decision of what missions to implement and metrics to calculate.

The implementation of the tools needed for the metrics was then done in order to simulate the same patterns studied and researched in the State Of The Art (SOTA). This implementation focused in being as embracing as possible, in order to avoid edge cases breaking the normal mission flow. It was also focused in trying to maintain a acceptable balance between accuracy and efficiency, taking also in consideration the under-actuated factor of the vehicles.

The metrics were implemented in a way to balance accuracy and computation complexity. The reason for this balance resides on the fact that although accuracy is paramount, the calculations of metrics should not impact the performance of the missions themselves by spending too much computational resources.

To achieve the goal of answering the questions proposed, there is need for the study and analyses of the metrics calculated. This analyses consists on mainly comparisons between the different experiments executed.

1.4 Document Structure

In chapter 2, a general contextualization of the problem, the platform and missions is given. It contains a brief explanation of all platform elements and a detailed one of the missions possible to perform in it, so the reader can understand the rest of the document better. In chapter 3, similar works and platforms are discussed by their approaches, problems and results, in order to compare the work to do with what was already done. Chapter 4 details the ground work made in order to calculate the metrics. It explains extensively the implementation of the mission management logic and the strategies used, as well as it's shortcomings, so the reader can understand the intricacies of the missions. Chapter 5 expounds the process of data collection by indicating the metrics calculated and their usefulness, as well as how their calculation is made, so the reader can better understand the metrics calculated. In the end, Chapter 6 analyses the data collected and announces the results achieved.

Chapter 2

Contextualization

Before an explanation of the implementation and results obtained, an explanation of the Platform, its components and capabilities is required. This chapter tries to explain the required information for the user to understand the rest of the work. That being said, it contains a detailed description of The Platform and all the components related to the work to develop. It has a brief explanation of how the Platform is organized and how it works, as well as its components. In section 2.2 the various mission parameters and types are described in more detail.

2.1 Platform Description

The work will be developed on top of a simulator being developed by the Artificial Intelligence and Computer Science Lab at Faculty of Engineering of the University of Porto [Silva, 2011]. The simulator's main objective is to simulate realistic multi-vehicle missions where vehicles are capable of self-coordination. In a broader sense, the Platform makes use of Microsoft Flight Simulator X [Microsoft Corporation, 2008a] to have realistic simulation, and controls its scenario elements such as airports and aircrafts via the SimConnect API [Microsoft Corporation, 2008b]. The user is supposed to define the scenario of the simulation, such as the airports, control towers and aircrafts, the goals and targets of the mission, and possible strategies. Having that information, it is intended for the agents to self-coordinate and complete the mission without any input from the user.

The general architecture of the simulator is depicted in Figure 2.1 and its main components are described below.

- **Control Panel:** Component responsible for configuring and starting the simulation and interact with it in real-time. It allows the user to configure the scenario, team, disturbances and missions. A snapshot of it can be seen in figure 2.2.
- **ATC Agent:** Agents corresponding to air-traffic controllers. They are responsible for monitoring their respective airspace and they can be of two different types: Terminal and En-route. Terminal are the agents responsible for monitoring the airports airspace. En-route are

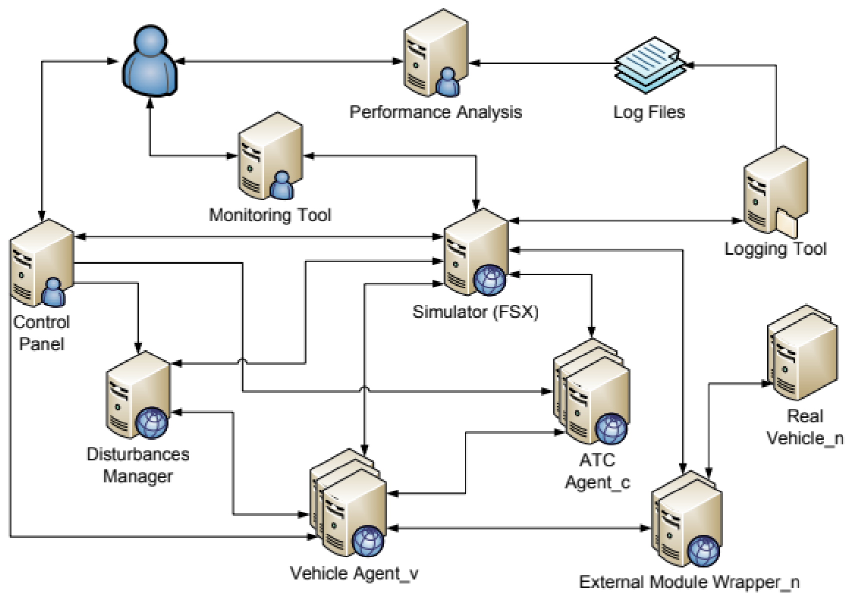
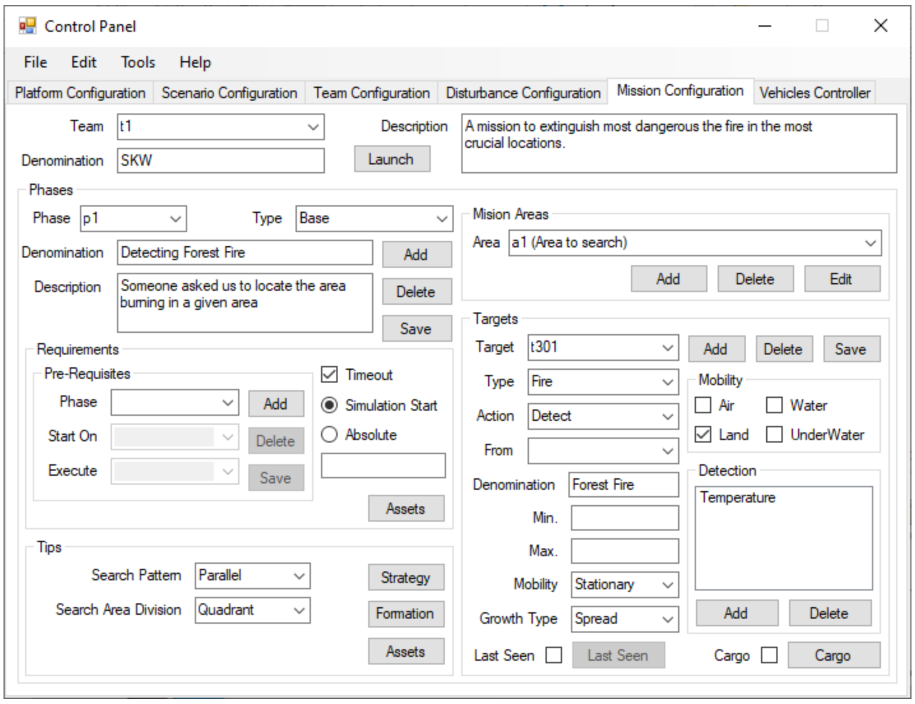


Figure 2.1: Platform General Architecture (extracted from [Silva, 2011])

the agents correspondent to TRACONs, which are responsible for the aircrafts out of the airports [Neto, 2017].

- **Vehicle Agent:** Responsible for the intelligence of each aircraft. It handles are the information received from the exterior and commands the aircraft in the FSX via the SimConnect API.
- **Simulator(FSX):** Component responsible for simulating the aircrafts and other vehicles in an realistic environment. In reality, it is an execution of Flight Simulator X and we use its API to connect it with the remaining platform. The connection between FSX and the other components comes from the SimConnect API, which lets the system retrieve and send information.
- **Real Vehicle and External Module Wrapper:** Component responsible for the construction and integration of real vehicle in the simulation platform. Ferreira built and integrated a UAV into the platform to study and test the interaction between real and virtual vehicles and environment [Ferreira, 2018].
- **Disturbances Manager:** Component responsible for handling the disturbances on the simulator [Almeida, 2017]. It is responsible for the creation and management of realistic forest fires and wind. It is being upgraded in parallel with this work by [Damasceno, 2020].
- **Monitoring Tool:** Component capable of showing the global state of the simulation in real-time, as well as detailed vehicle information.
- **Logging Tool:** Component responsible for generating the log files during a simulation. Since the creation of logs is something that is constantly happening, to avoid the creation of a large number of messages between components, it is distributed across the Platform.
- **Performance analysis:** Component responsible for extracting information from the log

files and showing it to the user in a understandable way. After this work, it is supposed to read the log files and calculate metrics from it. It allows the user to choose what metrics to obtain.


Control Panel

File Edit Tools Help

Platform Configuration | Scenario Configuration | Team Configuration | Disturbance Configuration | Mission Configuration | Vehicles Controller

Team: t1
Denomination: SKW
Description: A mission to extinguish most dangerous the fire in the most crucial locations.
Launch

Phases
Phase: p1 Type: Base
Denomination: Detecting Forest Fire
Description: Someone asked us to locate the area burning in a given area
Add Delete Save

Requirements
Pre-Requisites
Phase: Start On: Execute:
Add Delete Save
[✓] Timeout
(●) Simulation Start
( ) Absolute
Assets

Tips
Search Pattern: Parallel
Search Area Division: Quadrant
Strategy Formation Assets

Mision Areas
Area: a1 (Area to search)
Add Delete Edit

Targets
Target: t301
Type: Fire
Action: Detect
From:
Denomination: Forest Fire
Min.:
Max.:
Mobility: Stationary
Growth Type: Spread
Last Seen [ ] Last Seen
Add Delete Save

Mobility
[ ] Air [ ] Water
[✓] Land [ ] UnderWater

Detection
Temperature
Add Delete

Cargo [ ] Cargo

Figure 2.2: Control Panel Mission Tab

The objective of this work is to create a valid Performance Analysis component. That means the most essential modules for it are the Simulator and Logging Tool, so we can store the information needed to create evaluation metrics after the simulation, and the Performance Analysis component, which will to be made.

In a simulation, there are four main elements needed for it to be described correctly. They are defined by their own XML language and each can be created by an XML file, or directly on the Control Panel. These elements are:

- **Scenario:** Describes the airports, TRACONs, no-fly areas and the airspace used in the simulation. The airspace is defined by several waypoints representing positions in the real world, each one having an identifier, a longitude and a latitude. This airspace allows us to indicate paths or strategies in the Missions file in a more approachable way.
- **Team:** Indicates which vehicles are going to participate in the mission and some of the possible team conditions, such as additional no-fly areas or unusable bases.
- **Disturbances:** Describes possible disturbances in the simulation, as well as their behaviour at a superficial level.
- **Missions:** Describes the objectives of the simulation as well as how to clear said objectives. It can also detail when to start each mission and when it is completed. Additionally, it is possible to mix various different missions at the same time.

2.2 Missions

Missions are a main feature in the Platform. They are responsible for describing the goals and objectives of the simulation, as well as detail the process to achieve them.

A mission is a process in which a team has a certain objective that can be achieved by executing several steps correctly. It can range from simply transport people or goods to put out the most dangerous zones of a forest fire.

2.2.1 Mission Description Language

In order to standardise the concept of mission, there is the need to make a language capable of detailing all its steps, leading to the creation of the Mission Description Language (MDL) [Silva et al., 2014].

The mission, which is attributed to a given team, is composed by several phases. These phases correspond to steps of the mission which are not necessarily sequential, being able to happen simultaneously. The mission element of MDL is described in Fig. 2.3.

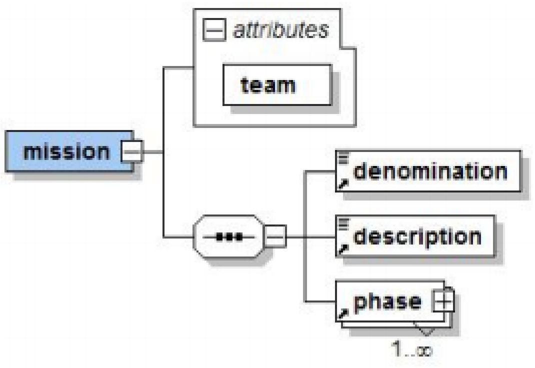


Figure 2.3: Mission Element Definition (extracted from [Silva, 2011])

Each phase is constituted by a type, denomination, description, areas, requirements, tips and targets, which will be explained in detail in Fig. 2.4.

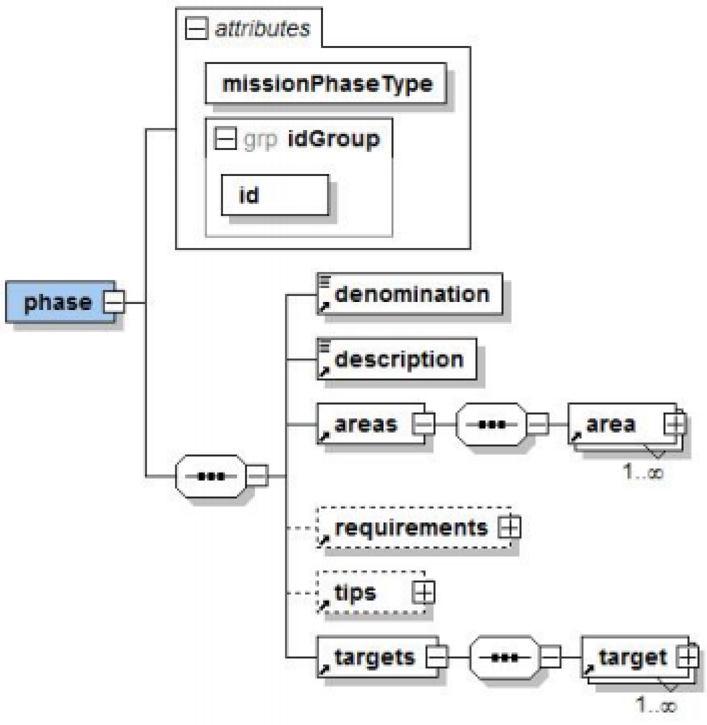


Figure 2.4: Phase Element Definition (extracted from [Silva, 2011])

- **Type:** The phase type decides the order in which phases in a mission are started by defining their priority.
- **Tips:** Responsible for the actual strategy for the task, can have information about the assets desired for the phase, possible formations, the search pattern to use, how to divide the area to search and lastly the various strategies to follow.
- **Targets:** Goals of the mission phase. It depends on the task we are handling.

The targets are the phase objective, containing the parameters described in Fig. 2.5. It contains a denomination, the vehicles that interact with it, the maximum number of occurrences, its mobility, the sensors and cargo that can interact with it and the last position it was seen.

2.2.2 Types of Missions

The platform is supposed to be able to handle various types of missions, which can all be described by MDL. The tasks are based on real-life problems worth simulating to find flaws or simply measuring its efficiency. In this subsection, some examples of possible missions are explained along with their usefulness in real world applications.

Detect

- Forest fire: Detect the burning area, using one of a variety of detection techniques, having as initial information the area where to search. The timely detection of forest fires is of the

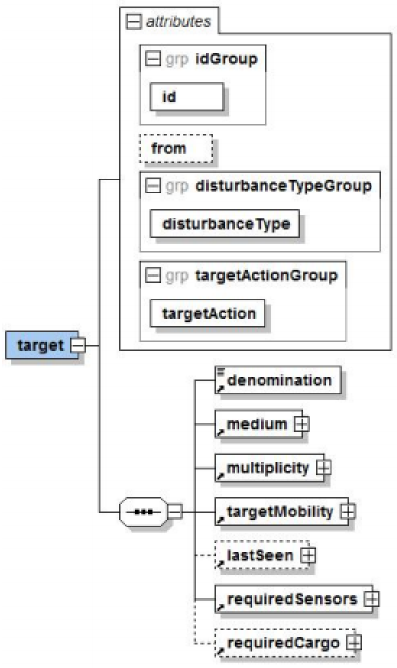


Figure 2.5: Target Element Definition (extracted from [Silva, 2011])

utmost importance to the protection of forests and sometimes habitations. The study of this kind of jobs would allow a big degree of automation. This, combined with satellite detection, which is not precise enough for now, would allow a greater response to this hazard.

- Pollution: Similar process as a forest fire, but using a threshold to decide contaminated area from non-polluted area. Nowadays, the amount of pollution and chemicals in the atmosphere is larger than ever. This raised a lot of attention to the issue of air quality. A tool capable of measuring the amount of chemicals in the atmosphere and mapping the locations where the living conditions are affected by this hazard has immense value in the protection of numerous peoples living standards.

Detect Origin

- Pollution: Detect the origin of pollution, using one of a variety of location techniques with feedback, having as initial info either a general location of the pollution or multiple locations of it. As said previously, the detection of pollution is a major issue nowadays. Not only the detection of polluted areas but also the detection of the pollution source are crucial to stop it from contaminating the area even more. In this regard, the study of autonomous missions to detect the source of such hazard has an immense value to the world.
- Hydrothermal Vent: Although not as important as detecting a source of pollution, the detection of a thermal vent has some benefits.

Drop On

- Drop item: Drop any item, from large amounts of water to bombs, in a given location or target. This mission complements the two above, making it possible to actually be useful beyond generating information. Dropping large quantities of water in certain points of a forest fire promptly can decrease by a lot the number of resources needed to put out the fire, for example.

Follow

- Person: Following people using drones. Possibility of following more targets than the number of resources. The detection and follow of a person can be really valuable for law enforcement. The capability of following someone without the need for human interaction decreases the amount of risk of letting a criminal or lawbreaker flee.
- Vehicles: Follow a vehicle using the appropriate resource (something of similar speed and maneuverability). Also very useful for law enforcement. The capability of following fast vehicles like cars allows the use of fewer resources to catch criminals.

Transport

- People/Cargo: Simple transport task of people and objects. By far, the most useful and most executed. Perfecting the way the airlines function would save millions of dollars. It is already possible to create an errand like this in the present simulator.

Measure

- Pollution: Measurement of pollution in certain areas, and mapping it in both space and time. It allows us to map the evolution of the pollution.
- Temperature: Measure the temperature in certain areas, creating a heat map. Eventually can be useful to check how a forest fire is propagating for example.

2.2.3 Mission Flow

The platform allows the user to decide how the missions are going to be performed with a high degree of freedom, by using the MDL. It lets him chain various phases, change the behaviour of the aircrafts if a certain event occurs, have various requirements needed to start a certain phase, stop if necessary, etc. Although the flow of a mission can be extracted from the MDL, it is still important to describe to better understand the degree of freedom the Platform missions offer.

A mission consists of multiple phases, which start when the requirements are fulfilled. If two or more phases with overlapping requirements needed to start, the one with highest priority starts first (priority is given by the missionPhaseType element). The requirements can be a time stamp, the completion of other phases (or one of various phases, being able to execute only once or one time for each one of the phases required) or the availability of certain elements, such as vehicles with a certain sensor, cargo, a specific agent or a type of vehicle. The goal of each phase is to interact in some way with its targets.

The targets can contain information about their multiplicity, if they move or grow, their last information gathered, which sensors can interact with it, if they need any cargo (for example water to extinguish a forest fire) and the medium were they can be (land, water, underwater and air). It must contain information about the type of disturbance it is (fire, pollution, etc) and how to deal with it, i.e., if it is supposed to follow it, detect it, detect its origin, etc. The phase is considered concluded if the target is dealt with, and if it has an area delimitation of where the mission is to occur, then the target must be inside said area.

The user is allowed to give various clues to the agents on how to perform each phase. It starts by defining what agents, cargo, sensors or vehicle types are needed to perform the phase. It also defines the default formation for the aircrafts to perform, the search pattern used if needed, the way to divide the area delimitation, the altitude preferred for the mission and, in the case of mission where there is the need to detect or measure the target, the separation between measures.

Besides the default tips of how to perform the mission, the user can also create some strategies that can change the normal flow of the phase when a condition occurs. The condition is a Boolean expression of multiple possible conditions, which can be:

- **Area Enter:** If the vehicle enter the predetermined area of the phase.
- **Area Exit:** If the vehicle exits the predetermined area of the phase.
- **Distance Target:** If the distance to a target is bigger, smaller or equal to a certain value.
- **Distance Base:** If the distance to an airport or base of operations is bigger, smaller or equal to a certain value.
- **Altitude:** If the altitude of the vehicles is bigger, smaller or equal to a certain value.
- **Time:** If a time stamp was passed.
- **Environment Wind:** If a vehicle detected a wind speed bigger, smaller or equal to a certain value.
- **Environment Rain:** If it is raining or not.
- **Environment Terrain:** If the vehicles passed by a certain surface type.
- **Team Vehicle Delete:** If a team member was removed from the mission.
- **Goal Achieved:** If the goal was achieved (useful to make the vehicles return to the base for example).

The strategies can change the normal flow of a mission by changing the elements of the mission, including the route, the search pattern used, the division of the area, the formation, or changing a certain element of the team (removing or adding a new member).

It is important to notice, however, that this flow is currently not implemented in the platform. In this work, a simple version of it was implemented in order to run simple missions, with only one aircraft per phase, so the simulation of the search patterns could be done.

Chapter 3

Related Work

3.1 Similar Platforms

Because aircraft missions can be quite expensive and important, there are various works regarding them. However, most of them are mainly focused on the most valuable and crucial problems, such as traffic management, airport management and pilot and drivers training. These simulations do not really have the possibility of simulating tasks like forest fire detection or measuring something in a predetermined area. In this section we describe some simulators similar in capability to the platform, which are usually mission-oriented.

3.1.1 AGENTFLY

AgentFly Technologies is a team focused on researching air traffic simulations, having already created some simulations whose purposes match with some of ours. They have four main areas of focus, which are:

- ATM (Air Traffic Management) Modeling and Simulation
- UAS (Unmanned Aircraft System) Integration into Shared Airspace
- Tactical Operations - Military and Security
- Aerial Work Automation

Their main simulation is ATM AGENTFLY, which is an air-traffic management simulator. It is a multi-agent air-traffic system that focuses on precise emulation of human behavior and the human-system interaction [Šišlák et al., 2012]. Although quite different from our platform, it is still interesting to analyse the metrics used to validate their platform, which are:

- Number of aircraft in each state at each time (handoff, landing, free flight, departure).
- Workload of a controller.
- Number of unaccepted handoffs over 15 minute intervals.
- Number of unresolved collisions over 15 minute intervals.

Although measuring the workload of a controller is not useful to help evaluate the performance of missions in our platform, the others are useful for different types of tasks. Although they are all useful for an air-traffic mission, some also are for other missions, such as the first, which is interesting for a follow mission, or the last ones, which can report problems in the remaining ones.

They also have a study on the impact the integration of an Unmanned Aircraft System (UAS) into a shared airspace has in the ATC workload [Rollo et al., 2017]. Like ATM AGENTFLY, there is a large emphasis on human behaviour simulation, simulating various human actors, such as pilots, pilots-in-command, traffic managers, etc.

In this project however, the metrics used to evaluate the ATC performance are not very explicit, although it points to the quality of the communication between components.

Its only on their works on Tactical Operations and Aerial Work Automation that AgentFly Technologies starts to overlap features with our platform.

In [Selecký et al., 2017] they detail the integration of a real component into the simulation, creating a mixed-reality simulator. Our platform also has a similar component , by being able to simulate the movements of a real-life vehicle in the simulator [Ferreira, 2018]. However, our physical agent is remote controlled by the user, whereas they made it an integration on both sides.

In [Selecký et al., 2013] there is some work made with physical UAVs in order to increase the autonomy of individual ones with the ultimate goal of allowing an operator to control more UAVs.

They implemented conflict resolution by using an asynchronous decentralized prioritized planning scheme and trajectory planning by using accelerated A* or RRT-based trajectory planner.

To validate their results, they first tested conflict resolution in a scenario where the two agent were in conflict and then tested a scenario with combined missions of surveillance and tracking a non-static target, all while avoiding collisions and conflict. They were successful as the physical UAVs were able to complete their task of following a movable target without colliding with each other. Their results are shown in Fig. 3.1.

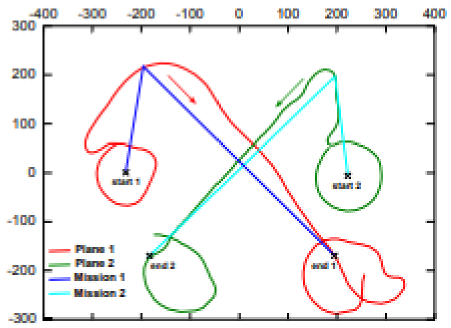


Figure 3.1: Conflict scenario result using real UAVs (extracted from [Selecký et al., 2013])

3.1.2 MOOS-IvP

MOOS-IvP is a simulator focused on operating tasks with marine vehicles. It is composed by two open-source projects, the first one being the Mission Oriented Operating Suite (MOOS), developed by the Mobile Robotics Group at the University of Oxford, and the second one the IvP Helm. MOOS provides middle-ware capacities in a publish-subscribe architecture and IvP Helm is the decision-making module. The IvP stands to interval programming and refers to the multi-objective optimisation method used by IvP Helm in a behaviour based architecture. In figure 3.2 we have the general architecture of MOOS-IvP, where we can confirm MOOS is responsible for the simulation and IvP Helm is responsible for the decision making. The IvP Helm works by choosing between a set of predetermined behaviours.

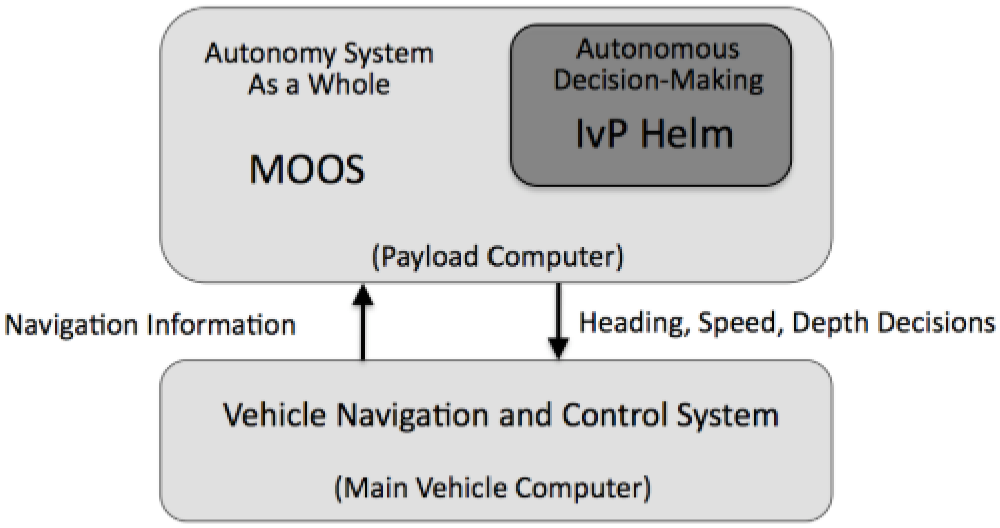


Figure 3.2: MOOS-IvP architecture (extracted from [Benjamin et al., 2019])

Our platform and MOOS-IvP are quite similar in the way they control the vehicles. They both use waypoints as means to indicate the path to the vehicle, like it is depicted in Fig. 3.3, and in both simulations, the vehicles always show a minimum turning radius, i.e., the vehicles cannot turn at will and need some distance to do it. We can also find similarities in the missions both platforms can perform, as they both can map an area, transport something, detect a shape and measure the concentration of a certain element or the temperature.

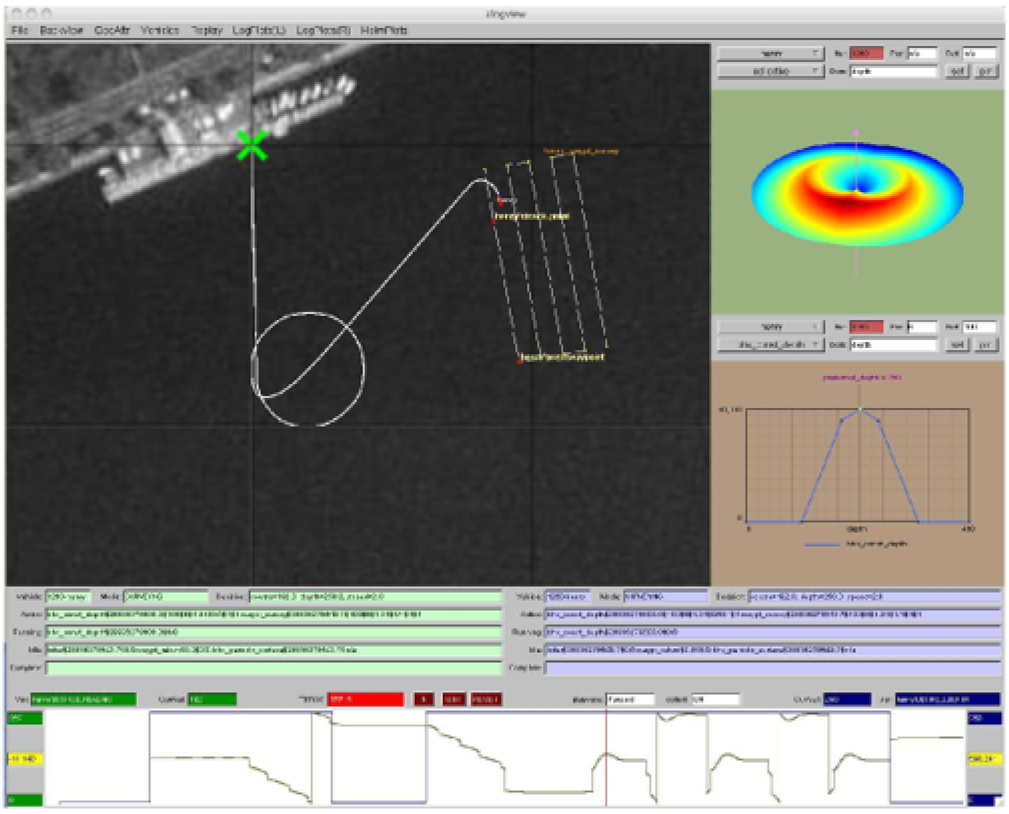


Figure 3.3: MOOS-IvP pMarineViewer (extracted from [Benjamin et al., 2019])

Although they are quite similar, there is an important factor that distinguishes them, as MOOS-IvP is an only aquatic vehicle simulator, and our platform focuses on aircrafts, although allows all types of vehicles. This is a difference mainly because of the behaviour of these two different sets of vehicles, which cannot be simulated with precision by the other platform. The disturbances are also quite different and create different types of errors, since underwater disturbances tend to be more continuous and create errors over time if not corrected and air disturbances have a more unpredictable behaviour and can even be catastrophic in rare cases. There are also some smaller differences, such as the landing and take-off of aircrafts, and the parking management of an airport, which MOOS-IvP is not able to simulate. That being said, it is still interesting to check how they handle some tasks.

In [Goh and Fan, 2019] we get some insight about which searching strategies they consider to solve their problem, which is mapping the bottom of the ocean. Although they considered four searching strategies initially, they rapidly discarded three of them because they would end up with distorted images and considered only the boustrophedon (lawnmower-like searching strategy) with an overshoot (leaving the borders), since it would not make distorted images. They also only considered a square area, which can not be the case in our simulation. That being said, their results are not transferred to our platform, since it is not mandatory to have a perfect grid of measures because images are not involved and we can search in different types of area. It is still interesting to consider all the initial searching strategies and adapt them to different shapes, which are depicted in Fig. 3.4.

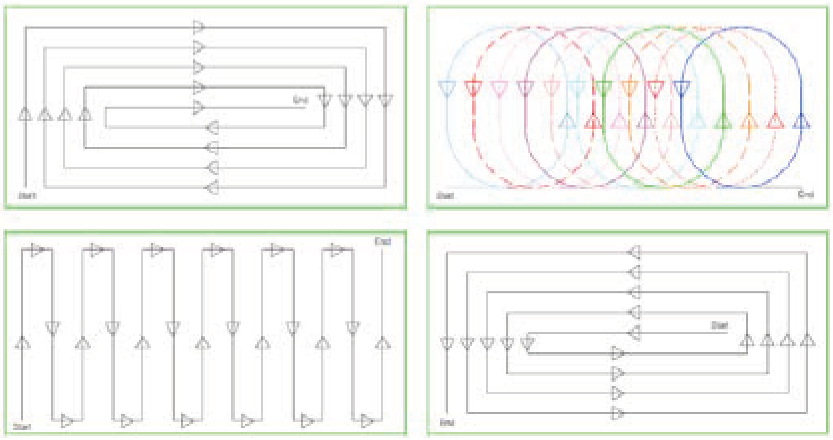


Figure 3.4: Path plannings considered to map the bottom of the ocean (extracted from [Goh and Fan, 2019])

In [Gupta et al., 2018] they tried to use Deep Q Learning to teach an autonomous aquatic vehicle to play capture the flag, and achieving good results. Although their results and approaches are not useful for the work to be developed, it is interesting to consider the value of the metrics developed as a reward function for future works on Deep Learning in the platform. An example of a game is depicted in Fig. 3.5

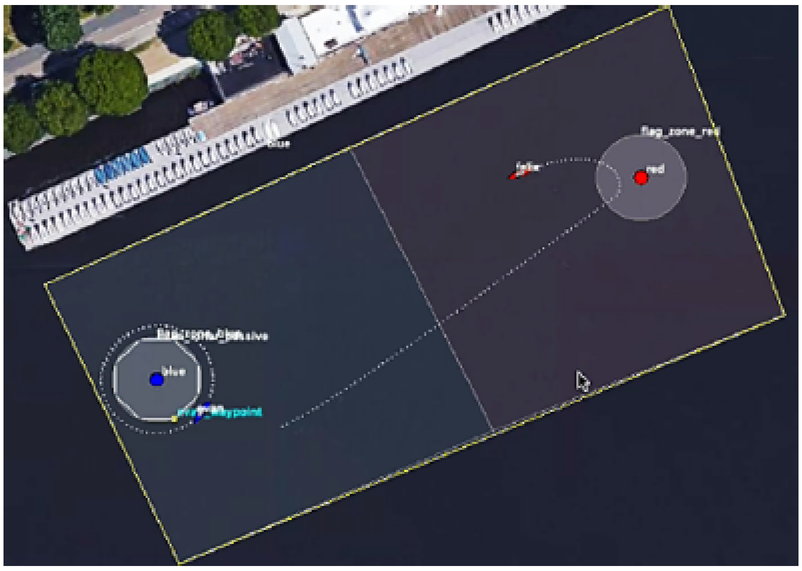


Figure 3.5: Representation of the capture the flag environment in MOOS-IvP(extracted from [Gupta et al., 2018])

3.1.3 Other Similar Platforms

Another similar platform is AirSim ([Shah et al., 2017]), made by Microsoft. This platform allows the piloting of different vehicles in a physically and visually realistic environment, as shown in

Fig. 3.6, being somewhat similar to Flight Simulator X in that regard. The main focus of their simulation is on UAVs, such as drones, and cars and the intended use is to use the platform to train agents by using machine learning.

AirSim generates a vast amount of data from its simulation so an agent can learn from it. It also generates the feed of view depth, object segmentation and vehicle camera. Like our platform, it also allows hardware-in-the-loop if needed.

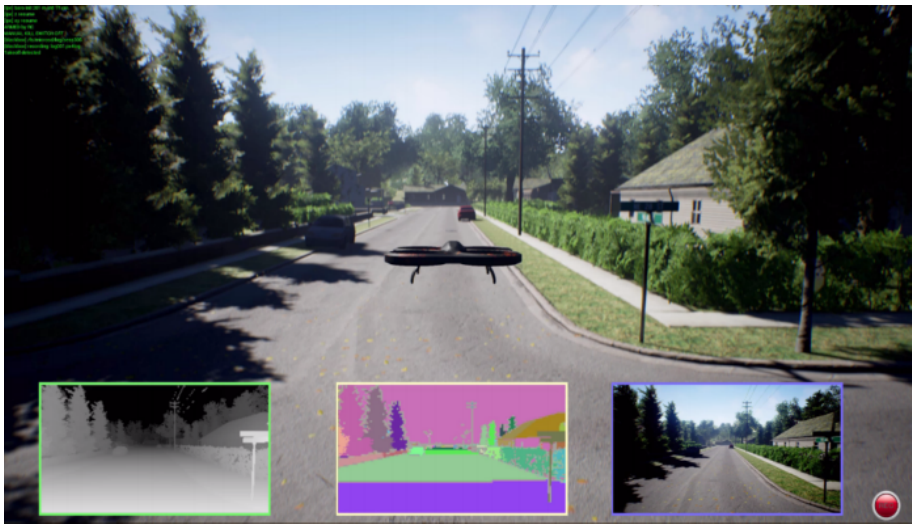
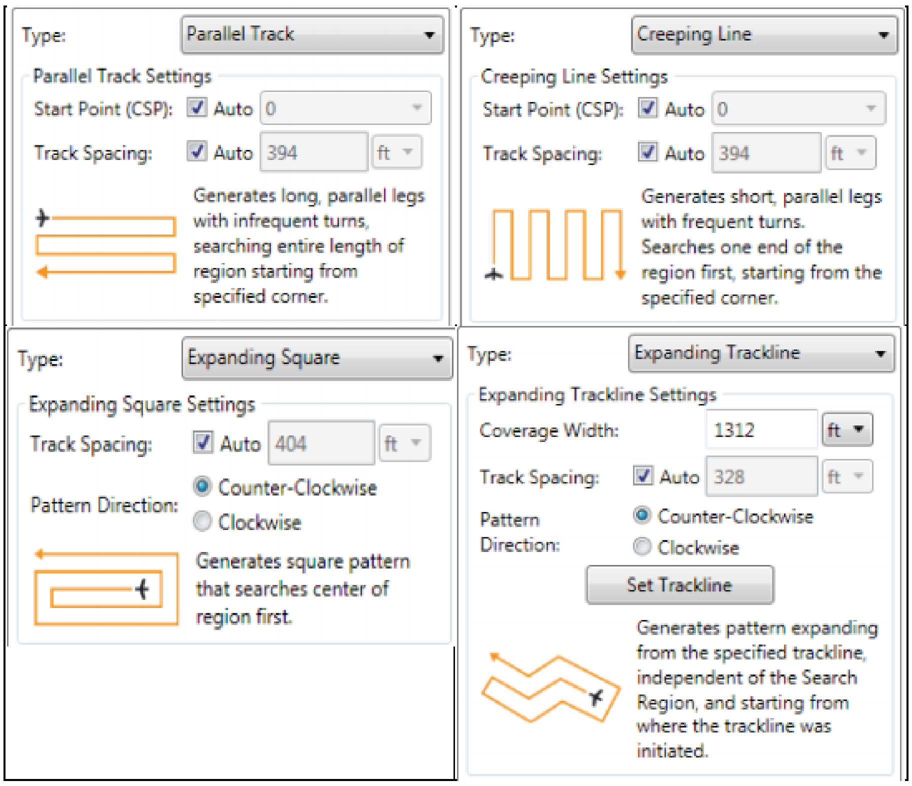


Figure 3.6: Snapshot of AirSim Simulation (extracted from [Shah et al., 2017])

In [Elston and Frew, 2008] it is described the implementation of tracking and follow missions using collaboration between aircrafts. Their simulator allowed hardware-in-the-loop, which they used to test their results. Their aircrafts were able to successfully follow a moving target in a real scenario. They also made a bigger simulation in their simulator, with more aircrafts and targets, and got small position errors (in relation to the target) in all aircrafts, except when they were launched.

In [Besada et al., 2018] a software capable of inspecting 3D elements using AUVs was implemented. This inspection is done by taking various photos from different angles of the target. The software allows the user to describe missions, indicating the starting position, ending position and objects to inspect. The panel of mission definition also has a 2D map of the area as well as a 3D one using google maps. The waypoints are then defined depending on the targets to inspect, by creating either a cylinder, a cone or a plane. In the end, to test their results, they compared the errors in position between the lines connecting waypoints and the actual position. Their errors were almost insignificant, and the images taken had quality.

In [Arbeit, 2013], a tool was made to simulate search and rescue missions using UAS (Unmanned Aircraft Systems), with the intent of automating them to increase their efficiency and effectiveness. In this tool, four search patterns were implemented: Parallel Track, Creeping Line, Expanding Square Pattern and the Trackline Pattern, which are depicted in Fig. 3.7.


Type: Parallel Track

Parallel Track Settings

Start Point (CSP): ☑ Auto 0

Track Spacing: ☑ Auto 394 ft

Generates long, parallel legs with infrequent turns, searching entire length of region starting from specified corner.

Type: Creeping Line

Creeping Line Settings

Start Point (CSP): ☑ Auto 0

Track Spacing: ☑ Auto 394 ft

Generates short, parallel legs with frequent turns. Searches one end of the region first, starting from the specified corner.

Type: Expanding Square

Expanding Square Settings

Track Spacing: ☑ Auto 404 ft

Pattern Direction: [✓ Counter-Clockwise] Clockwise

Generates square pattern that searches center of region first.

Type: Expanding Trackline

Expanding Trackline Settings

Coverage Width: 1312 ft

Track Spacing: ☑ Auto 328 ft

Pattern Direction: [✓ Counter-Clockwise] Clockwise

Generates pattern expanding from the specified trackline, independent of the Search Region, and starting from where the trackline was initiated.

Figure 3.7: Four patterns developed in [Arbeit, 2013] (extracted from [Arbeit, 2013])

Their sensor consists on a directional camera with a certain radius and field of view, which can be rotated to any intended direction. In this case, two ways of sensor control were made: Pattern-based Sensor Control and Occupancy Map-based Sensor Control. The first one consists on following the pattern direction, with a little adjustments when turning. The second one consists in facing the zone with less area searched. The differences between both this strategies can be seen in Fig. 3.8.

The goal of this work was to compare the different Sensor Control algorithms using the various search patterns. The results obtained were the following:

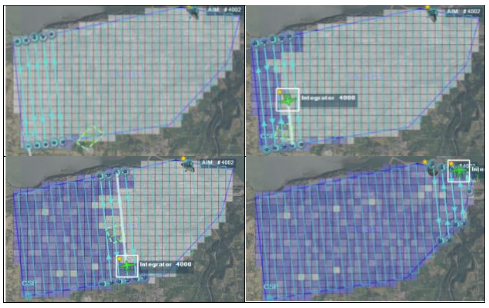
- For the Expanding Square search pattern, both algorithms reached the same area coverage in the end, but the Occupancy Map-based Sensor Control reached it slightly faster.

- For the Creeping Line search pattern, the Occupancy Map-based Sensor Control was faster covering area, but ended up with less area coverage in the end.

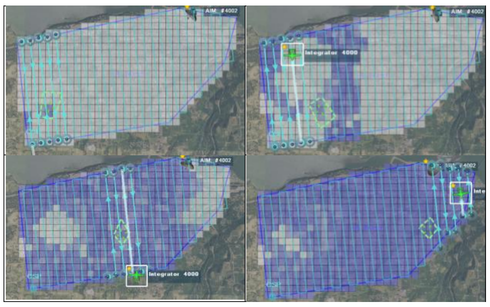
This work didn't study the effects of the different search patterns in the mission and used drones, which are not under-actuated vehicles.

3.2 Similar Works

Although there are not many studies focused on performance evaluation of a simulation platform, it is present most of the times in the result validation or data gathering for analysis.



(a) Pattern-based Sensor Control in a Creeping Line search pattern



(b) Occupancy Map-based Sensor Control in a Creeping Line search pattern

Figure 3.8: Experiments using two different Sensor Control Algorithms in a Creeping Line search pattern (extracted from [Arbeit, 2013])

3.2.1 Performance Evaluation in Air Traffic

In [Vitor et al., 2019], it is discussed how having a helper when executing missions by piloting drones can help its efficiency. Their motivation was the fact that pilots are sometimes heavy loaded with tasks other than piloting, decreasing the performance of the said mission. To test the increase in efficiency, a test was made where a pilot and an analyst would pilot a course where a certain description of the course would need to be done. The analyst would handle all the information gathering and would give instructions to the pilot.

The evaluation of performance in this work was measured by two factors: correct depictions of the scenario and a subjective evaluation via form. Both the pilot and the analyst would evaluate factors like difficulty in piloting, communication, tiredness, sickness, immersion, etc.

In [Labib et al., 2019], it is implemented an abstract approach to low altitude UAV traffic management using multi-layer network of nodes and airways. They evaluate their performance by measuring the impact the quantity of UAVs circulating has on traffic time, traffic energy consumption, number of paths changed and layers changed. Although their approach is quite different from ours, their way of measuring the performance is still interesting to analyse.

3.2.2 Performance Evaluation for Similar Tasks

It is of the most importance for this work to check platforms doing similar missions and discover, if they have some form of performance evaluation, which metrics they consider most important.

In [Bosse et al., 2007] it is discussed a new coverage strategy for an under-actuated car-like vehicle by combining the strategies Spiral Inward and Spiral Shift. This work is very similar to our Detect, Detect Origin and Measure missions since they both work with area coverage and both consider under-actuated vehicles. They tested their work using a robotic mower in a 30m by 40m smooth area, obtaining the results depicted in Fig. 3.9, which shows the amount of overlaps. In the end, they considered as valuable metrics the following:

- Percentage of covered area.
- Time taken to cover the area.

- Percentage of redundant coverage.

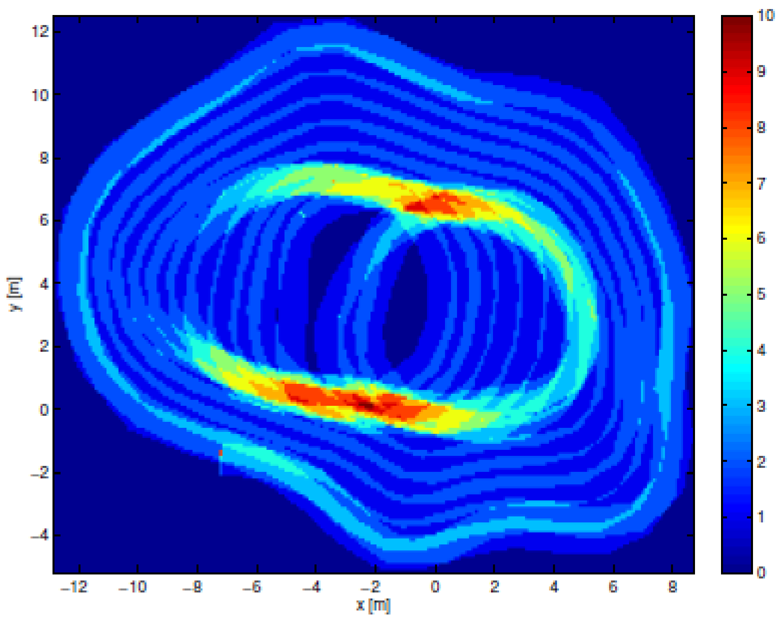


Figure 3.9: Area coverage results using a Spiral Inward and Spiral Shift (extracted from [Bosse et al., 2007])

As said above, [Elston and Frew, 2008] describes the implementation of a mission similar to the follow one. In their case, they used the distance between their aircrafts and the target over time as a way to evaluate the performance of the mission.

In the work [Arbeit, 2013] described above, four metrics were calculated:

- Coverage Factor: A metric that estimates the effectiveness of a given search mission. It is similar to the sensor radius divided by the division between the tracks, but with some differences due to using a directional sensor.
- Search Duration: Duration, in hours and minutes, of the mission.
- Percent Searched: Area coverage during the mission. This metric is calculated using the number of cells searched.
- Post-Search Fuel Time Remaining: Duration, in hours and minutes, left until the aircraft runs out of fuel.

Chapter 4

Preliminary Work

This chapter details some of the groundwork needed for running the different types of missions the platform allows us to make. Since there was a severe lack of development in this area of the Platform, much of the time of the dissertation was invested into it.

4.1 Mission Management

One of the first problems founded was the lack of any functional code after retrieving the information from the mission configuration file. Essentially, there wasn't anything in The Platform that handled the mission's data and made the agents move accordingly. Because of this, a mission manager logic was added to The Platform to indicate to the agents what mission to accomplish and how to do so.

For each team, a team leader is elected (the first member of the team in the configuration file), making it responsible for managing the missions corresponding to the team by running a mission management logic component.

Such component has information related to the status of each phase (unresolved, resolved and completed) as well as the status of each team member. The phase is considered resolved when an agent receive it, and completed when the agent completes it. Each phase of a mission configuration file is considered a mission to be performed by one member (a simplification explained in 2.2.3) and it is either completed by the first free member of the team or by a specific agent if the requirements ask for such. For a mission to begin, the mission management component checks if all requirements for the mission are met. Such requirements can be:

- **Agent:** Requires a specific agent to perform the mission phase.
- **Phase:** Check if certain phases were already completed. This allows the chaining of various phases and also allows one agent to perform more than one.
- **Time Out:** Confirms that a particular period time or timestamp has passed.
- **Sensor:** Checks if the required aircrafts have a specific type of sensors. It isn't implemented since it doesn't affect this project.

- **Cargo:** Checks if the required aircrafts have a certain cargo. It isn't implemented since it doesn't affect this work.

If the requirements are met, then the mission management component sends a message to the team member with the information needed for him to execute the phase. The team member has the responsibility of performing it by creating and following a route of waypoints. When the aircraft arrives at the last waypoint of the mission, it informs the team leader that it is completed and he is free.

4.2 Search Patterns

Search patterns are paths the vehicles follows in order to cover a predetermined area, in order to measure a certain element or detect one or multiple targets. Since it can do so in multiple ways, different search patterns were implemented to compare their efficiency. The patterns implemented are well-known patterns that were seen numerous times in section 3. These patterns were implemented in a way to balance efficiency and accuracy, i.e., trying to go through the shortest path and avoid zones where there is no measures (not covered). This implementations are, however, not perfect, having some shortcomings.

4.2.1 Lawnmower

This pattern consists of zigzagging through the predetermined area in parallel lines. This search pattern is implemented in a way to minimize the number of turns the aircraft needs to make. This minimization is important because the turns made by the vehicle are the main source of inefficiency in the pattern.

In order to be able to create the pattern, the Sinusoidal projection is applied to the area, transforming it's geographical coordinates into orthogonal ones. Using the new area, the program starts to discover which orientation the lines must have to minimize the number of turns. It does so by simulating a bounding box of the original area for each edge of the polygon, and then find the one with the shortest side. In the case of the circle, the orientation of the box will depend on the aircraft's position.

After discovering the best orientation, the area is rotated in order to have the lines parallel to the Y-axis to simplify the calculations. Then, using the distance between measures retrieved from the mission XML file, the X values of the lines are calculated (by creating a separation between them of distance between measures), like it is depicted in Figure 4.1.

After that, the same is done for each individual line, by using the length of each one and the distance between measures, calculating the waypoints that determine the route of the aircraft (marked in blue in Figure 4.1). The length of each individual line is calculated by intersecting the line with all edges of the shape, obtaining the points and calculating the distance between them.

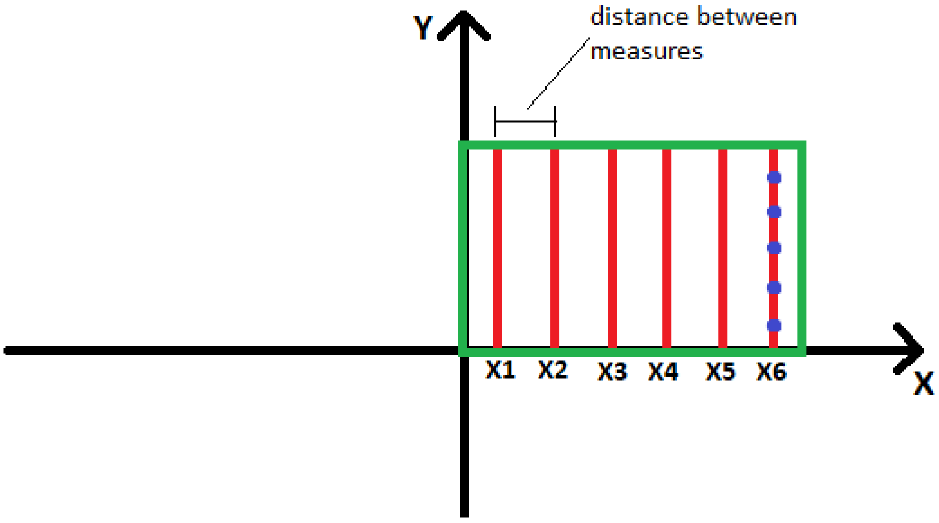


Figure 4.1: Representation of the waypoints calculated in a lawnmower search pattern

After getting the waypoints across all lines, the starting point of the pattern is obtained using the rotated position of the aircraft. After that, it applies an inversion of half of the lines (depending on which the starting point is), so the intended shape can exist.

So the aircraft can turn correctly between each line, additional waypoints are created, to avoid letting the aircraft miss waypoints. This is important because if the agent doesn't go through one of the waypoints, it turns around to catch it, creating a loop to do so. These waypoints can take three different forms depending on the distance between lines and the turn radius of the aircraft at cruise speed.

- If the turn radius is equal, the aircraft will make a semi-circle.
- If the turn radius is bigger, the aircraft will turn at a right angle, go in a straight line and then turn another right angle to catch the next line.
- If the turn radius is smaller, the aircraft will form a horseshoe lookalike shape, where it turns outward until it has enough space to perform a semi-circle rotation and get to the next line.

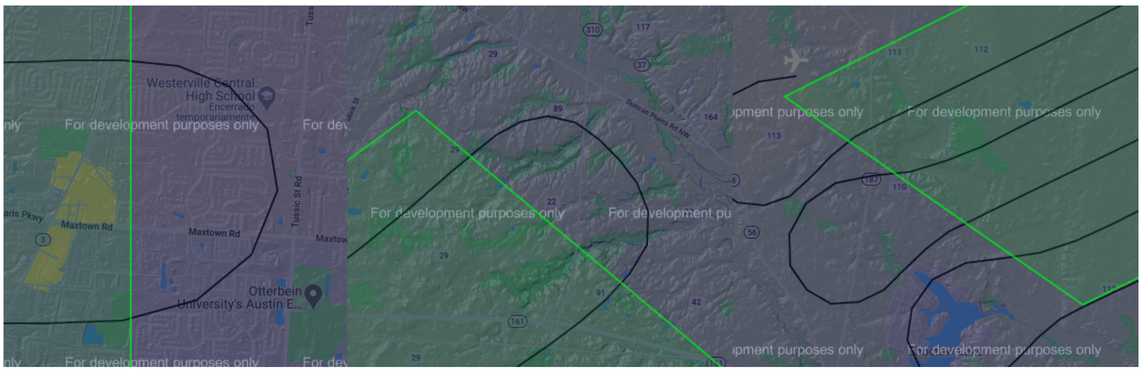


Figure 4.2: Possible turns in a lawnmower search pattern

These turns could be, however, improved since we are equalizing the position of the start and endpoints of the curve. This means, when we are turning on an inclined line, some distance is added to the "shortest" line to perform the turn entirely outside of the area, which can be easily confirmed in the third image of the Figure 4.2. This could be improved by using that distance to turn and decrease its length. Such improvement wasn't made due to increased complexity. In the end, the rotation is reversed to have the real waypoints positions.

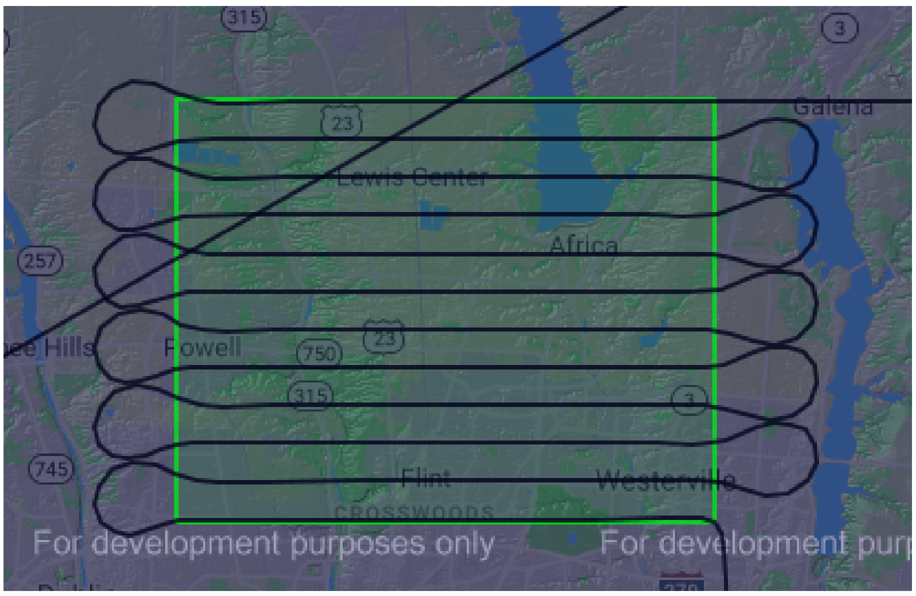


Figure 4.3: Lawnmower pattern in a square

4.2.2 Spiral In/Out

As the name implies, this pattern consists of covering an area by following a spiral from the furthest point to the center or vice-versa, like it's shown in figure 4.4.

The implementation of this pattern consists of creating various semi-circumferences with increasing or decreasing radius at a certain interval each step until it reaches the furthest point in the area or the aircraft minimum turning radius, depending if the spiral goes outwards or inwards respectively.

Each waypoint was calculated using the heading relative to the area center and its distance to it. That being said, the points were calculated directly in the orthogonal coordinate system, which were then converted to a geographical coordinate system by assuming they were projected in a Sinusoidal projection.

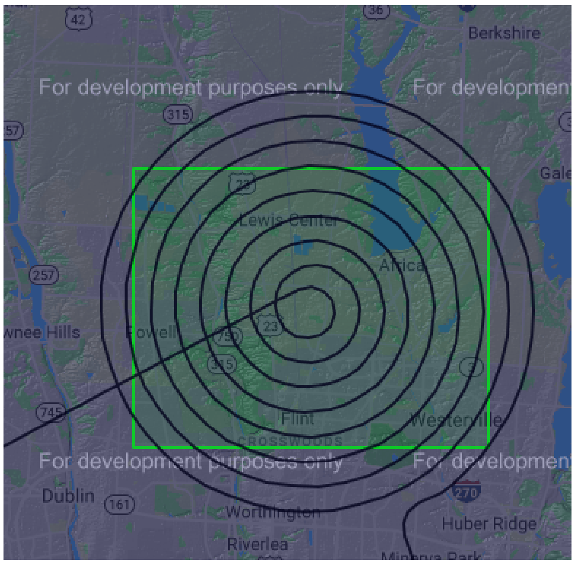


Figure 4.4: Spiral In pattern in a square

4.2.3 Follow Shape In/Out

This pattern consists of replicating the area shape and replicate it until it becomes a circle which radius is the minimum turning radius, like it is seen in Figure 4.5.

The implementation pattern starts by calculating all vertices angles of the area by using the formula described in [Roberts, 2009]. By using the distance between the three consecutive vertices we can calculate the angle of the middle one by using the formula described in Equation 4.1, where P_{12} means the distance between the first and second vertex, being the second vertex the middle one.

$$angle = \arccos(\frac{P_{12}{}^2 + P_{23}{}^2 - P_{13}{}^2}{2 \times P_{12} \times P_{23}}) \tag{4.1}$$

These angles are useful to change the waypoints after their calculation on order to allow turning of the aircraft. After this, the starting point of the mission is calculated based on the position of the aircraft, prioritizing distance. This starting point corresponds to the closest vertex of the polygon or closest point in case of a circumference. This point is calculated whether the pattern goes inward or outward.

Starting from the first point, multiple vectors are calculated, going from the boundary vertices to the corresponding point in a circumference of radius minimum turn radius centered on the center of the area. These vectors are then inverted if the pattern starts in the center and goes outward.

The lengthiest vector is then divided into multiple points separated by a certain distance, defined by the user. The other vectors are also segmented uniformly based on the number of divisions the lengthiest vector suffered. From this segments, multiple new shapes are formed similar to the original one, going from the surface to it's center. After this, each line segment is transformed into three waypoints: a center one and two close to the edge. The distance from the edges of each line segment is needed to give the aircraft space to turn correctly and is calculated using the formula shown in Equation 4.2.

$$leisureDistance = \frac{minimumTurnRadius}{\tan \frac{angle}{2}} \tag{4.2}$$

If the area is a circle, however, there is no need to change waypoints in order to let the aircraft turn. That being said, the waypoints are created in circumferences of different radius which are then connected.

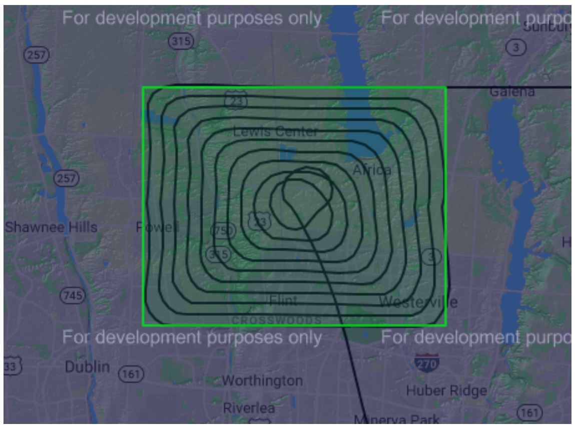


Figure 4.5: Follow Shape In pattern in a square

4.3 Follow Mission

The follow mission consists of making an aircraft follow another vehicle by predicting the closest point they can intercept and updating that point each second. This point is calculated through the current direction of the target, its position, speed, and the position and cruise speed of the vehicle performing the mission.

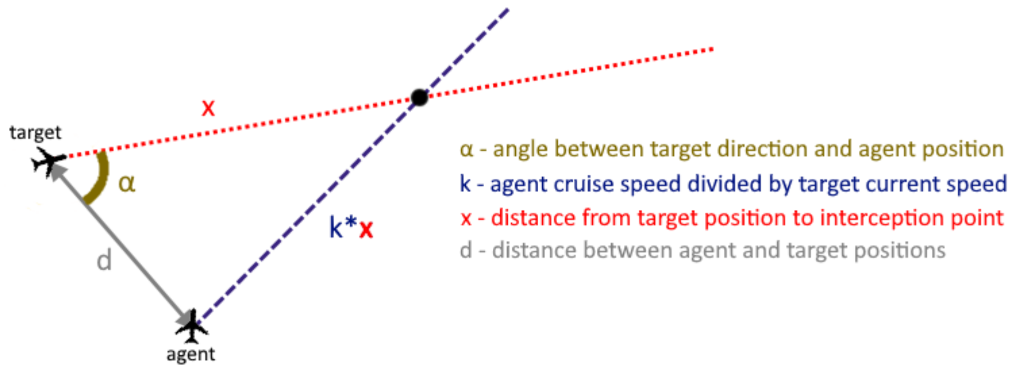

target
x
α
d
k*x
agent
α - angle between target direction and agent position
k - agent cruise speed divided by target current speed
x - distance from target position to interception point
d - distance between agent and target positions

Figure 4.6: Explanation for follow mission formula

The idea is to calculate the distance between the target position and the intersection point in order to use the target heading and position to calculate the intersection point as we see in Figure 4.6.

This distance can be calculated by using the law of cosines, using the Equation 4.3:

$$(k \times x)^2 = x^2 + d^2 - 2 \times d \times x \times \cos\alpha \tag{4.3}$$

This equation can be simplified by solving for x, ending up with the formula described in Equation4.4:

$$x = \frac{d \times \left(-\cos\alpha \pm \sqrt{\cos^2\alpha + (k^2 - 1)}\right)}{k^2 - 1}, k \neq 1 \wedge k > 0 \tag{4.4}$$

However, the formula doesn't work in some edge cases. If the value inside the square root is negative (discriminant binomial), it means our agent will never be able to catch the target.

If both aircrafts have the same speed (k = 1), then the formula used will be different (otherwise, it would result in a division by zero). The Equation 4.5 results from the Equation 4.3 as well, but when simplifying, the x^2 factor disappears.

$$x = \frac{d}{2 \times \cos\alpha} \tag{4.5}$$

4.4 Route Missions

Routes correspond to missions where a certain aircraft has to follow a predetermined course of waypoints. These waypoints are stored in the airspace element of the scenario configuration file and referred to in the mission configuration file in the correct order.

They are useful to represent transport and drop on missions as they need to follow a particular path. With this, we can easily simulate real-life traffic for example. The waypoints are given in geographical coordinates.

This missions were already used to simulate the air traffic of the USA (United States of America) for an entire day by [Neto, 2017].

4.5 Patterns Shortcomings

The implementation of any pattern in the earth is complicated due to its curvature. That being said, the implementations described in Section 4 have some known problems.

4.5.1 Approximate Minimum Turn Radius

The minimum turn radius corresponds to the radius of the smallest curve the aircraft can make. This radius depends on various factors such as wind, max bank, speed and some other elements, making it not trivial to calculate. In theory, this radius could be calculated by the formula 4.6

$$radius = \frac{velocity^2}{g \times \tan(bank)} \tag{4.6}$$

However, the value of that radius would always be significantly bigger than what the FSX aircrafts could make. Since such differences would impact the mission's performance significantly, a workaround was made, to measure the radius of a turn made by the aircraft at cruise speed. Such radius would be different for each vehicle. The workaround consisted in making an aircraft complete a circumference at cruise speed, and measure the radius of that circle.

This workaround limits the maneuvers to be always at cruise speed and brings slight mistakes and imperfections to the patterns.

4.5.2 Mistakes in Projection

For most of the patterns presented, there was a transformation from longitude and latitude coordinated to an orthogonal system using distances (in meters). Because the earth is an oblate spheroid, the transformation from one system to another always brings errors since the sphere surface cannot become a plane.

This problem is extensively studied and there are no easy solutions. Table 4.1 shows some projections which were pondered to be used and why they weren't.

Table 4.1: Projections

Projection	**Pros**	**Cons**
Mercator	Easy to implement and same as google maps projection	Giant errors in longitude on high and low latitudes.
Gall-Peters	Equal area projection	Same errors in longitude as Mercator. Equal area results from deformations in latitude values.
Robinson	Less errors in longitude values than Mercator	Weird to implement as the projection depends on a table of values for each latitude. Mistakes in latitude are not different from Mercator.
AuthaGraph	Considered the most correct projection	The projection consists on an irregular shape, and it's not trivial to implement.
Mollweide	Small errors in longitude and latitude	Implementation is not trivial
Sinusoidal	Areas are represented accurately, distortion is reduced when used for a single land mass rather than the entire globe. Almost no impact on performance	Severe distortion near outer meridians at high latitudes.

In the end, since we were only working with small areas, to use the Sinusoidal projection, using the center of the area as the origin of the system. The formula to transform the geographic coordinate system to a orthogonal one is described in Equation 4.7 and Equation 4.8:

$$x = (altitude + earthRadius) \times \cos lat \times lon \tag{4.7}$$

$$y = (altitude + earthRadius) \times lat \tag{4.8}$$

Chapter 5

Data Collection

This chapter expounds the entire process of data collection in the platform, such as which metrics were chosen, how they are calculated, and which experiments were ran and why. This process is crucial to the main goal of this project since the metrics obtained affect the results obtained directly. The values obtained by this process were validated by looking at them with a critical eye and also comparing them with the path the agent took to complete the mission . Albeit this is not the best approach to validate the results, it was enough to detect some errors that appeared.

5.1 Metrics

As the name suggests, metrics are values that can represent the efficiency of a certain mission. However, because the priorities of a certain mission depend not only on its type, but also on the user, multiple metrics needed to be calculated so that the vast amount of priorities can be met. For example, in an area coverage mission, where we can use search patterns, some users might prefer more area covered, in percentage, while others prefer a more time-efficient search pattern or even something in between. That being said, multiple metrics were chosen, which can be separated into three categories:

- **Absolute Metrics:** Metrics that represent something absolute, such as the area covered, time spent, or distance traveled. These values are useful when comparing with other absolute metrics from other experiments.
- **Relative Metrics:** Metrics that represent a percentage, and are a measure of efficiency themselves. Such metrics can include area coverage divided by total area, minimum possible time divided by time spent, etc.
- **Relational Values:** This values represent the parameters chosen in the mission, which are a simplification of the mission configuration. They help not only identifying the mission performed, but most importantly, relate the different tasks in order to compare them. For example, when comparing the distinct search patterns in the same area, it is useful to have the indication of the pattern and area used.

5.1.1 Absolute Metrics

These metrics are useful to compare results between similar experiments, such as having the same area or same aircraft. Table 5.1 depicts all the absolute metrics calculated as well as a description of what they represent and the type of mission they can be used in.

Table 5.1: Absolute Metrics

Name	Description	Missions
Time Spent	Time spent in the mission. Only starts when entering the mission area in the case of Detect and Measure missions.	Detect, Measure, Transport, Drop On
Area Covered	Area covered by the aircraft.	Detect, Measure
Area Overlapped	Area which was covered twice or more.	Detect, Measure
Distance Covered	Distance the aircraft went through. Only starts when entering the mission area in the case of area detection missions.	Detect, Measure, Transport, Drop On, Follow
Time Between Measures	Frequency of measures over time.	Detect, Measure
Average Speed	Average speed of the aircraft.	Detect, Measure, Transport, Drop On, Follow

5.1.2 Relative Metrics

These metrics are useful to calculate the efficiency of an experiment without needing to compare to other experiments. Table 5.2 shows all the relative metrics calculated as well as a description of what they represent and which mission type they make sense in. These metrics are percentage values that can also be used to calculate a final score for the mission performance.

Table 5.2: Relative Metrics

Name	Description	Missions
Time Efficiency	Minimum time required to perform the mission divided by the time spent in the mission. The minimum time required is calculated by dividing the minimum possible distance by the cruise speed of the aircraft.	Detect, Measure, Transport, Drop On
Area Covered	Area covered by the aircraft divided by the total area to search.	Detect, Measure
Area Overlapped	Area which was covered twice or more by the aircraft divided by the total area to search.	Detect, Measure
Distance Efficiency	Distance the aircraft went through divided by the minimum possible distance. The minimum possible distance corresponds to the distance traveled by the aircraft if it could turn instantly at any angle (like a drone).	Detect, Measure, Transport, Drop On.
Speed Efficiency	Average speed divided by the aircraft cruise speed.	Detect, Measure, Follow and Transport.

5.1.3 Relational Values

These values are needed to link the different experiments to compare metrics between them. They can almost be considered the same as ids in a relational database. Table 5.3 expounds all the links between experiments necessary.

Table 5.3: Relational Values

Name	Description	Missions
Area id	Area id so we can compare experiments using the same area.	Detect, Measure
Vehicle Type	The aircraft used.	Detect, Measure
Search Pattern	The search pattern used to complete the mission.	Detect, Measure
Separation	The distance between the measures in a search pattern.	Detect, Measure

5.1.4 Unfeasible Metrics

This subsection details metrics which were intended to be measured, but weren't for diverse reasons, such as incompatibility with FSX information or lack of implementations in the platform.

Fuel Metrics

Many metrics related to the fuel spent during a mission were thought since it was the most similar metric to the monetary cost of the mission. However, such metrics are impossible to calculate because the FSX doesn't handle fuel values for AI traffic aircrafts. Since the mission is performed by this kind of aircrafts instead of user ones, calculating fuel expenditures is impossible.

Table 5.4: Fuel Simulation Variables

Simulation Variable	User value	AI value
Fuel Total Quantity	Yes	No
Fuel Total Capacity	Yes	Yes
Estimated Fuel Flow	Yes	Yes
Fuel Tank Center Capacity	Yes	No
Fuel Tank Center Level	Yes	No
Fuel Right Quantity	Yes	No
Fuel Left Quantity	Yes	No
Fuel Selected Quantity(all)	Yes	No
Fuel Total Quantity Weight	Yes	No
Fuel Weight Per Gallon	Yes	Yes
Fuel Selected Quantity Percent(all)	Yes	No

As we can see, only static values about fuel are stored in the AI traffic, having no information about dynamic ones, problem that also appeared in a previous work with other variables [Garrido, 2019]. Having the estimated fuel flow is also not very interesting since it is an estimate of the fuel flow at cruise speed. Since the aircraft is not always at cruise speed, it becomes a wrong estimate to the fuel spent on the mission. These values correspond to the aircraft Cessna Skyhawk 172SP.

Target Metrics

The platform is supposed to be able to simulate missions like searching and detecting fires, pollution, aircrafts, and many others. Unfortunately, the disturbances and targets were not fully implemented in the platform, making it impossible to calculate any metric related to them. That being said, this version of the mission implementation was simplified. That makes it so each aircraft performs a phase, which can only correspond to an action. Various phases can be then chained, just as explained in Section 4.1. There were also some meta-metrics related to successful and unsuccessful phases which were abandoned because, without targets, a phase cannot be categorized as successful or unsuccessful. Table 5.5 contains the list of the abandoned metrics related to this factor.

Table 5.5: Target and Phase Related Metrics

Metric name	Description
Number of successful phases	Meta-metric containing the successful phases in a mission.
Number of unsuccessful phases	Meta-metric containing the unsuccessful phases in a mission.
Percentage of successful phases	Number of successful phases by Number of phases of a mission.
Percentage of following time	Following time in a follow mission. Not considered because the present version is always following the target(has perfect information on it).
Closing to origin	Percentage of time in a mission of Detect Origin where the aircraft is going towards the origin. Since in order to have an origin a target has to exist, the implementation of Detect Origin missions was postponed.
Targets found	Number of targets found in a phase
Percentage of targets found	Targets found by targets in the simulation.
Successful flag	Whether or not the phase was successful

5.2 Logs

In order to calculate the metrics described in Section 5.1, multiple logs were need to be calculated or measured. Albeit some metrics were instantly written in the log file, others had notable calculating efforts. This section contains the information of all logs used as well as the calculation of the more complex ones.

These logs are calculated during the mission and stored in a specific log file. This log file is named by adding all the different parameters of a mission plus the time of simulation start. That way, when searching for the mission log, we can easily separate and identify them by their parameters. The parameters are the following:

- **Mission Type:** The type of mission, which can be Detect, Measure, Follow, Detect Origin, Drop On and Transport.
- **Area Denomination:** A description of the area. When deciding to create a set of experiments, the user can create various areas and give them a specific description for later comparison. The user can then replicate this area making so its denomination represents the shape of the area.

- **Aircraft Type:** The type of aircraft used. This is useful since the aircraft used can change the value of the metrics by a lot since it's minimum turn radius and cruise speed also change.
- **Search Pattern:** The search pattern used to complete the mission. It is important to compare only some specific search patterns or compare the effect other parameters have in a certain search pattern.
- **Separation:** The distance between the measures in a search pattern. It's also a value that greatly affects the performance of a mission and might be important to not mix missions with different separations when comparing data.

It's important to specify that these parameters are in the name only for identification purposes since they are also stored in the logs as relational metrics.

Table 5.6: Logs Required to Calculate the Metrics

Log name	Description	Units
Area Size	Size of the area where the mission is performed	Squared meters
Perfect Minimal Distance	The shortest path the aircraft can make in order to complete it's mission	Meters
Start Time	FSX time when the aircraft enters the area for the first time. This time corresponds to the number of seconds from the beginning of the Gregorian calendar.	Seconds
Radar Info	Log containing the position of the aircraft, its altitude, speed, time and the temperature in that position. The position of the aircraft is logged both in geographic coordinates and in orthogonal distances to the area. The temperature is not useful for metrics but can be useful to create temperature maps in the future. This log is measured each second of simulation time.	Various
End Time	FSX time when the aircraft terminates the mission. This time corresponds to the number of seconds from the beginning of the Gregorian calendar.	Seconds
Aircraft Cruise Speed	The usual aircraft velocity when cruising. Important to calculate some of the relative metrics.	Knots

Metrics Calculations

The logs represented in Table 5.6 are the logs required to calculate all the absolute and relative metrics. These logs are made while the simulation is running and are used to calculate the metrics at the end of it. It's important to clarify that all the metrics are calculated and stored in the log file for later treatment. This is useful because we might want to store the metrics derived from the logs in more than one place, like a CSV file for example. This way, we avoid redundancy of calculations.

The area size of the area to search was implemented in two different ways, whether it was a circle or a polygon. If the area consisted in a circle, then the circle area formula was used. If it was a polygon, however, a more elaborate approach was needed. In order to calculate the area of a polygon, the transformation from geographic coordinates to Sinusoidal projection was mandatory since the method used only works with orthogonal systems. This method consisted in the following steps:

- Considering the center of the area as the origin of the coordinate system.
- For each pair of consecutive vertices create a trapeze formed by the vertices and two other points in the x-axis with the same x value as the vertices.
- Calculate the area of the trapezes.
- Add all the trapezes areas depending on the direction the vertices are going in relation to the x-axis, i.e. add if the first vertex has an x value inferior to the second one, and subtract if not.

The Perfect Minimal Distance was calculated by calculating the minimum distance between the multiple waypoints given to the aircraft. This calculation ignores points given to the aircraft to smooth its turns.

All the other logs are information directly received from the FSX simconnect.

Absolute Metrics

- **Time Spent:** Calculated by subtracting the start time to the end time.
- **Average Speed:** Calculated using the various speed values in Radar Info logs.
- **Distance Covered:** Calculated by multiplying the average speed and the time spent.

The calculation of the area covered and overlapped was, however, more challenging. It was calculated using the General Polygon Clipper library [Murta, 2009] to join and intersect rectangles and then calculate the final result in the same way the original area was calculated. This is done by first creating different line segments uniting the various measure points in the sinusoidal projection. These line segments were then transformed into rectangles by creating other four points at a distance equal to half the separation parameter. This successfully creates rectangles in which two edges are parallel to the initial line segment and the other two are perpendicular to it. After creating the polygons, by the usage of a wrapper, we made the union of all the rectangles and calculated the area of the final result. The overlapped area was calculated by intersecting the rectangles with the union area before this area contained the rectangle. A simple representation of the rectangles is seen in Figure 5.1.

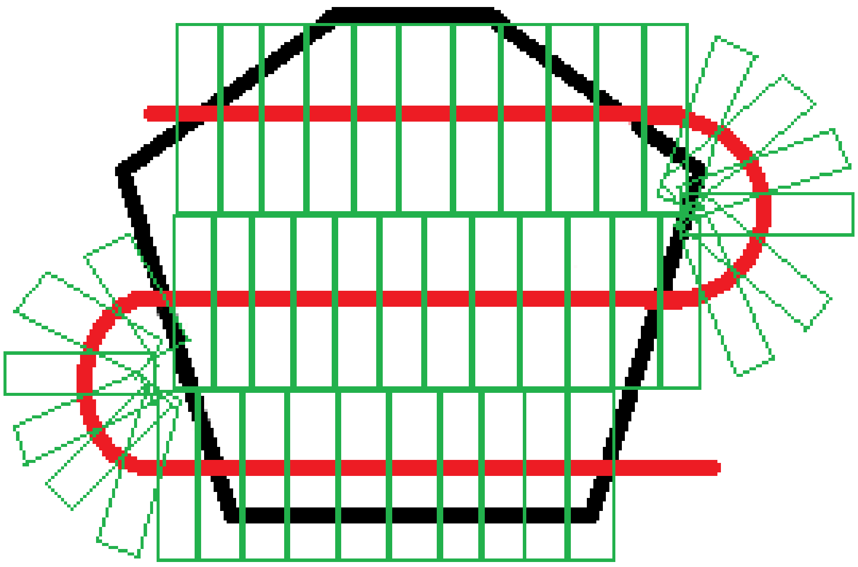


Figure 5.1: Rectangles made in order to calculate the area covered and overlapped

Relative Metrics

- **Time Efficiency:** Calculated by first dividing the perfect minimal distance by the cruise speed obtaining the minimum time needed to complete the mission. Obtained by dividing the minimum time by the time spent.
- **Area Covered:** Calculated by dividing the area covered by the total area.
- **Area Overlapped:** Calculated by dividing the subtraction of the total area by the overlapped area by the total area.
- **Distance Efficiency:** Calculated by dividing the perfect minimum distance with the distance covered.
- **Speed Efficiency:** Calculated by dividing the average speed by the aircraft cruise speed.

5.3 Experiments

To obtain various relations between parameters and results, multiple experiments were planned out by making simulations combining different parameters. Since both the follow missions and transport ones are not useful in terms of data collection, the tests were restricted to measure missions. This section details how the parameters were changed and why they were changed that way as well as what was expected to obtain by changes. These parameters consist of the area shape, the aircraft used, the search pattern used and the separation between the measures.

Area Shape

The area shape can greatly affect the efficiency of the mission. For example, a flatter shape is better for the lawnmower search pattern and worst for the spiral and follow shape ones, which happens because while the first makes fewer turns, therefore being more efficient, the latter ones waste lots of turns by trying to keep the separation equal in the furthest points of the area. That being said, it is expected that some search patterns behave better or worst depending on the area shape. The shapes decided were a circle, a rectangle, a pentagon, a small irregular polygon, and flatten irregular polygon, which can be seen in Figure 5.2.

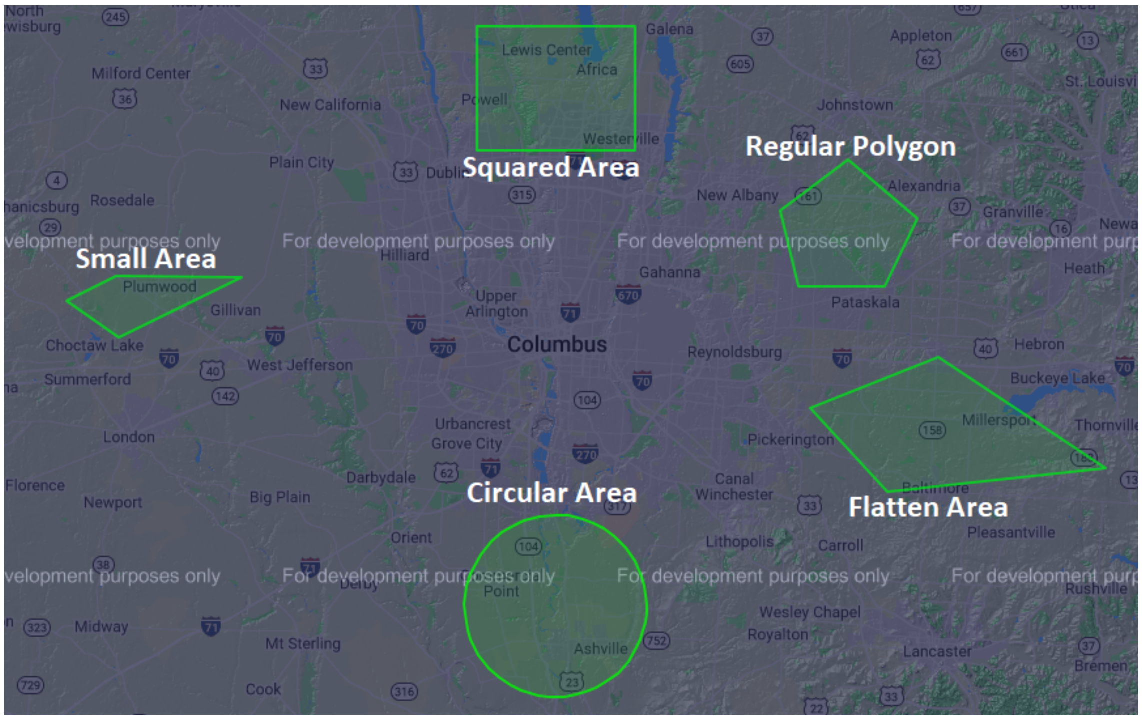


Figure 5.2: Different Areas used in the experiments

Aircraft used

The aircraft used makes the cruise speed and minimum turn radius to change drastically. These values have a big impact on the performance, especially if we are comparing results between aircrafts. A large turn radius impacts the mission performance negatively, by making the aircraft follow longer routes to turn, problem which affects the lawnmower more than the other search patterns. A faster aircraft, however, completes the mission in less time than a slower one. That being said, the aircraft chosen are represented in the Table 5.7 and their model in FSX can be seen in Figure 5.3.

Table 5.7: Aircraft Specs

Aircraft Name	**Minimum Turn Radius**	**Cruise Speed**
Cessna Skyhawk 172SP	984 meters	110 knots
Beechcraft Baron 58	2534 meters	176 knots

Figure 5.3: Cessna Skyhawk (left) and Beechcraft Baron(right) models in FSX

The experiments performed will have in the end, the following parameters:

- Search Pattern used: The five different search patterns.
- Area Shape: The five areas seen in Figure 5.2.
- Separation: The separation between the lines in a search pattern. The values used on the experiments will be 1000 meters and 2000 meters.
- Aircraft used: The Cessna Skyhawk and the Beechcraft Baron, which can be seen in Figure 5.3.

Chapter 6

Results

This chapter presents a study and analysis of the metrics obtained by running the experiments defined in section 5.3, in order to better understand the relation between the different parameters and their effect on the performance of the missions.

6.1 Pattern Efficiency

In order to calculate the efficiency of a certain pattern, a study of the relative metrics obtained is required. Since these metrics can be seen as a comparison between the experiments and their best possible result, having higher scores means the experiment was more efficient in the domain of the parameters used. This means that some tests might be worse than others in terms of absolute metrics but obtain better relative metrics because of the parameters used. It is also relevant to notice that some metrics make more sense in some patterns than others. For example, when using the lawnmower search pattern, relative time and distance tell us more about the experiment's efficiency than the relative area covered and overlapped. Such behaviour happens because this pattern has almost 100% of area coverage and 0% of area overlapped, while the other metrics suffer from the turns the aircraft has to make.

6.1.1 Lawnmower Pattern

As seen in Figure 4.3, the searching strategy consists of following equidistant parallel lines, performing the necessary turns outside of the mission predetermined area. Because of that, it can cover almost all the designated area while avoiding overlapping measures. However, this pattern loses efficiency in terms of distance travelled and time spent due to the turns it has to make. That being said, the Figure 6.1 contains the distance efficiency for all the area shapes, separation values and aircrafts used.

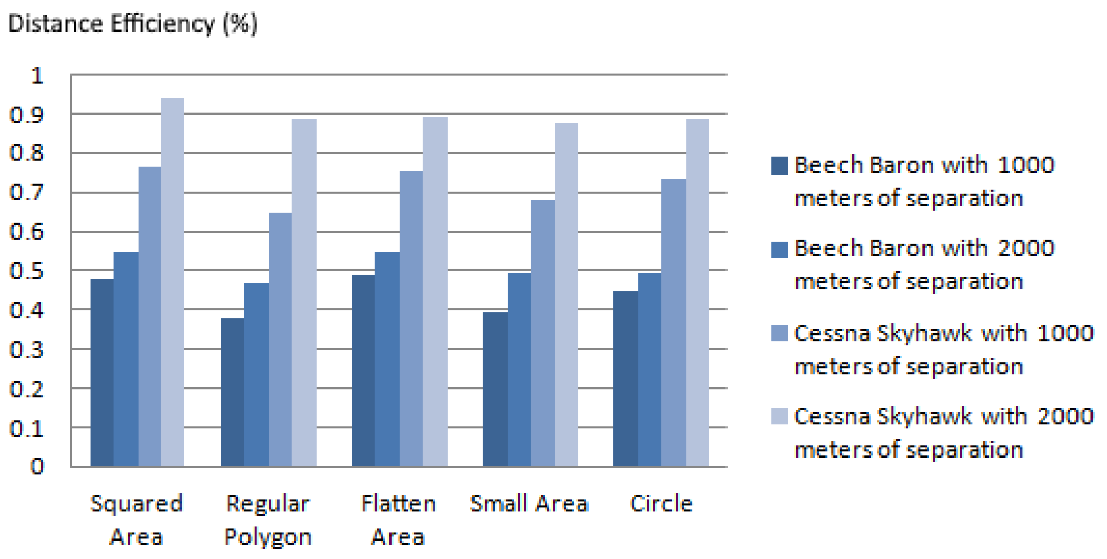


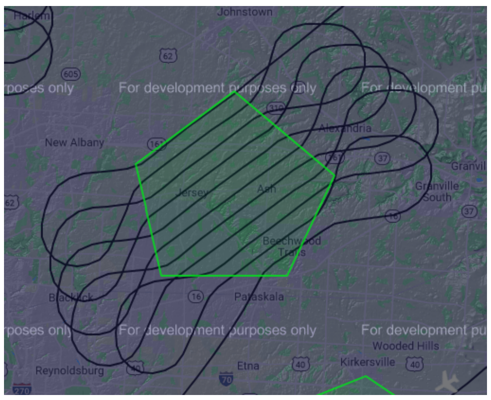
Figure 6.1: Distance Efficiency in a Lawnmower search pattern

As we can see, this pattern presents minimal variation between the different areas in terms of redundant movement. That happens because the number of turns needed is always given by the number of lines crossed minus one, which means that the height of the original bounding box has almost no impact on the metric. The slight variations are probably the result of a shortcoming when turning on an inclined line explained in subsection 4.2.1.

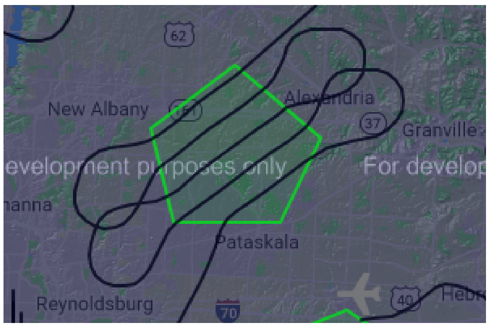
However, it shows great variations depending on the aircraft used due to the difference in the minimum turn radius between aircrafts. A lower turn radius greatly decreases the length of the turn, improving the metric displayed. Since Cessna Skyhawk has a minimum turn radius of 984 meters at cruise speed while Beechcraft Baron has 2534 meters, it is expected for the distance efficiency to have better values for the former one, such as it is demonstrated.

The effects of the separation of measures are more tricky to detect. It seems to have a negligible impact when using the Beechcraft Baron and a substantial impact when using the Cessna Skyhawk. Increasing the separation has the effect of flattening the curvature of the turn, making its length smaller compared to the distance of the lines it connects. However, this does not explain the difference that appears when using the aircraft with smaller turn radius. Such effects are probably the result of changing the shape of the curvature because the separation became more significant than the turn diameter.

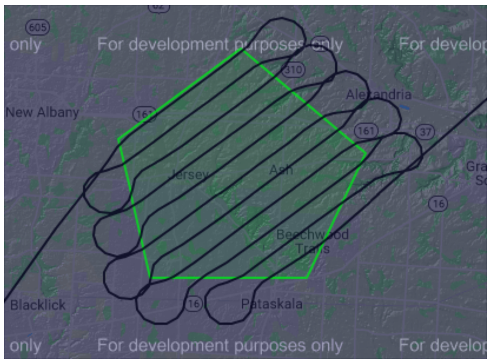
Figure 6.2 contains the path followed by the aircrafts when executing the lawnmower search pattern on the regular polygon-shaped area, with all the different parameters. We can easily confirm the results obtained in the metrics by comparing them with the images presented. By analysing Figure 6.2a, we can confirm that the turns section is larger than the parallel lines one. On the other hand, by comparing Figure 6.2c and Figure 6.2d, we can validate the origin of such a large gap in the metrics obtained.



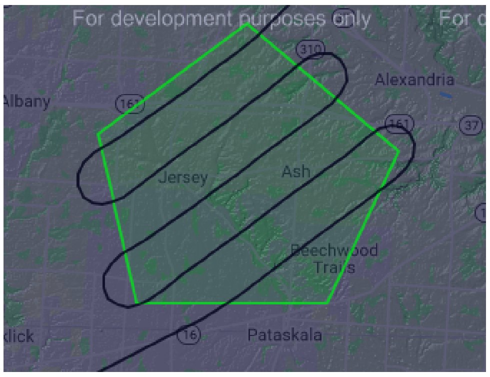
(a) Beech Baron with a separation of 1000 meters



(b) Beech Baron with a separation of 2000 meters

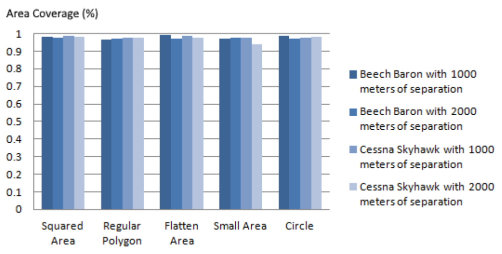


(c) Skyhawk with a separation of 1000 meters

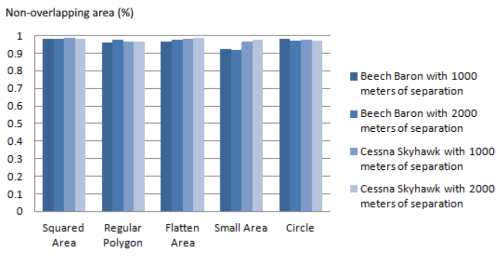


(d) Skyhawk with a separation of 2000 meters

Figure 6.2: Different parameters in a Lawnmower pattern in a regular polygon shaped area



(a) Area covered relative to the area size



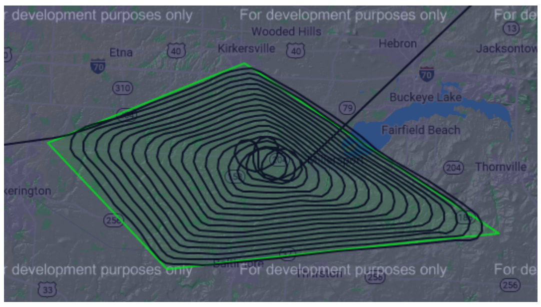
(b) Area not overlapping relative to the area size

Figure 6.3: Area covered and overlapped in a Lawnmower search pattern

The other relative metrics are not worth discussing since the average speed is equal to the cruise speed. This makes the time efficiency equal to the distance efficiency and speed efficiency equals to 1. The pattern also always cover almost 100% of the area while almost not overlapping it, as it is shown in the Figure 6.3.

6.1.2 Follow Shape In/Out Pattern

As described in Section 4.2.3, this pattern consists of a spiral that starts with the shape of the area to search and ends with a circle of radius equal to the minimum turn radius of the aircraft.


(a) Cessna Skyhawk

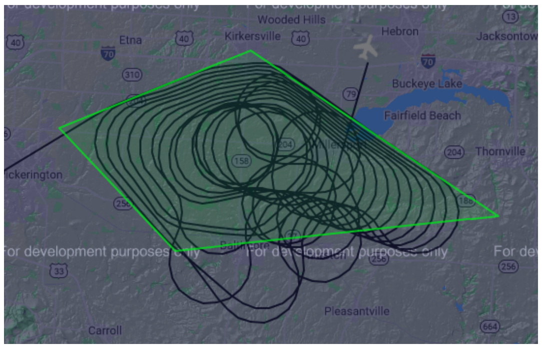
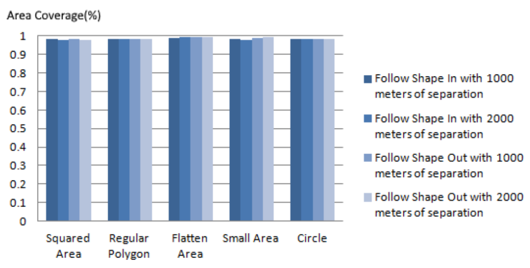

(b) Beech Baron

Figure 6.4: Different aircrafts performing the Follow Shape In pattern in the flatten area

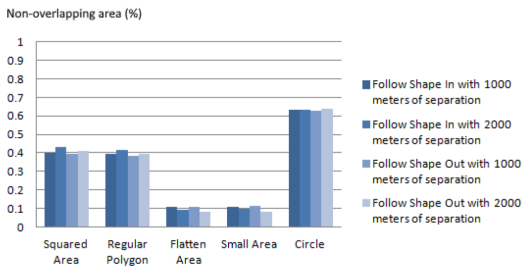
The separation in this pattern is used to calculate the distance between lines when considering the furthest point from the center of the area. This means places where the distance from the center to the edge is inferior to that of the furthest point will have the lines closer to each other, creating overlapping in measures. This was decided to maximize area coverage by losing on other metrics. Additionally, the way the pattern updates the vertices waypoints to turn more efficiently reduces not only area coverage (by not measuring the areas close to the vertices) but also the distance covered. This makes the path the aircraft goes through shorter than the minimum distance calculated (therefore obtaining values bigger than 1 in distance efficiency and time efficiency).

This pattern seems not to work correctly when handling aircraft with a big turn radius like the Beechcraft Baron like it can be seen in Figures 6.4. The reason behind this is the heading on which the aircraft enters some waypoints, not giving him leeway to reach correctly to the next waypoint, and making it go in multiple turns. This error makes it studying the metrics of this pattern using the Beechcraft Baron not worth it as it does not represent what it is intended to be.

That being said, the Figures 6.5 represent the covered and overlapped area of the search pattern. As we can see, the area covered is similar to the parallel search pattern however, such values are obtained at the cost of overlapping measures. As we can see, the amount of overlapping seems to depend on from the area shape, not being disturbed by the amount of separation, nor the direction the pattern is followed by. By analyzing the values, we can confirm that areas where the boundary consists of points of similar distance from the center are bound to have less overlapping area, like the circle. Areas shaped like the flatten one, where the furthest boundary point differs a lot from the other ones, present, however, a large amount of overlapping area, which can be confirmed in figure 6.4a. It was expected for the circle-shaped area to have a similar result in the overlapped area as the lawnmower search pattern. The origin of such amount of intersecting points is probably the way the metric is calculated. Figures 6.6 represent some examples of the pattern being executed.

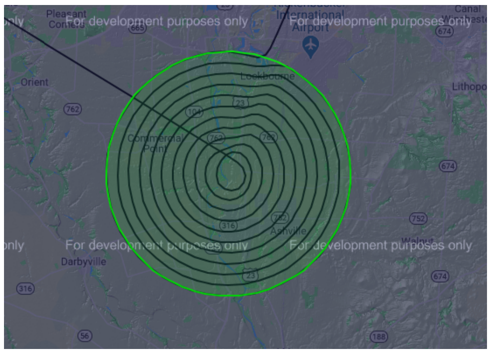


(a) Area covered relative to the area size

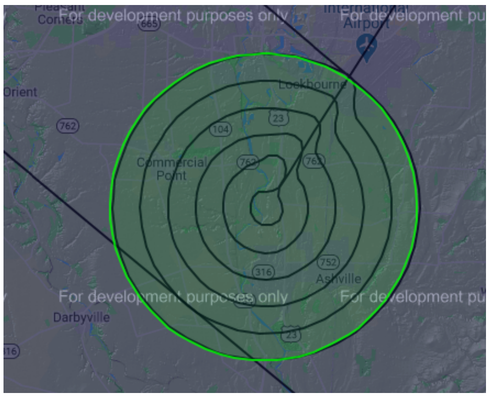


(b) Area not overlapping relative to the area size

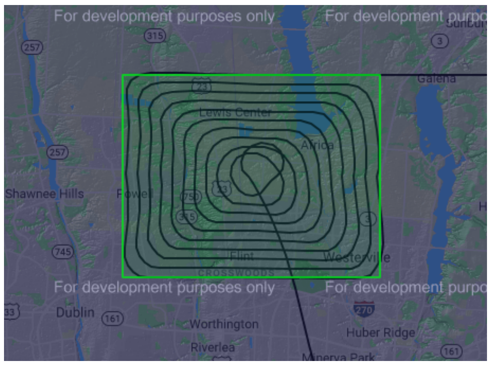
Figure 6.5: Area covered and overlapped in a Follow Shape search Pattern



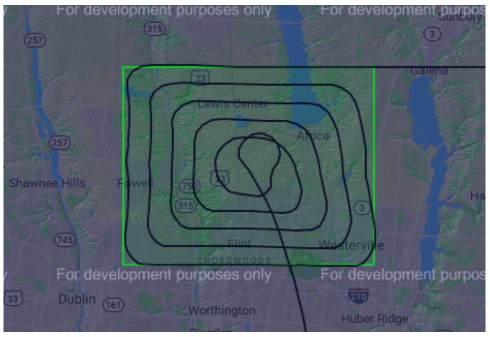
(a) Follow Shape In in a circle shaped area



(b) Follow Shape Out in a circle shaped area



(c) Follow Shape In in a squared shaped area

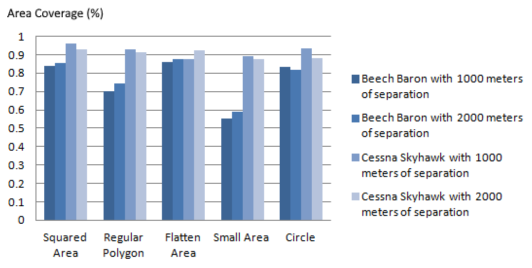


(d) Follow Shape Out in a squared shaped area

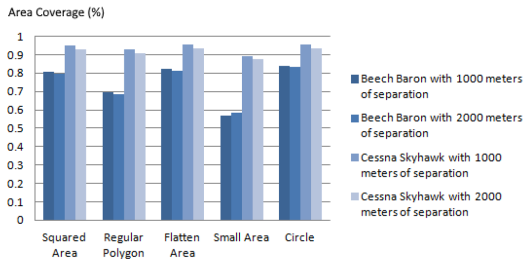
Figure 6.6: Some examples of the Follow Shape pattern (separation of 1000 meters in the left and 2000 in the right)

6.1.3 Spiral In/Out Pattern

As explained in section 4.2.2, this pattern consists in forming a spiral with equidistant lines, that starts in the furthest point and travels to the area center or vice-versa. That being said, depending on the area shape, much of the distance covered by the aircraft is usually spent outside of the area. However, such fact does not affect the distance efficiency metric since the minimal distance also considers the path outside of the area. With this, the average value of the metric is 98.8%, without any outlier. The same happens with the overlapped area metric, which has an average of 94.6%,

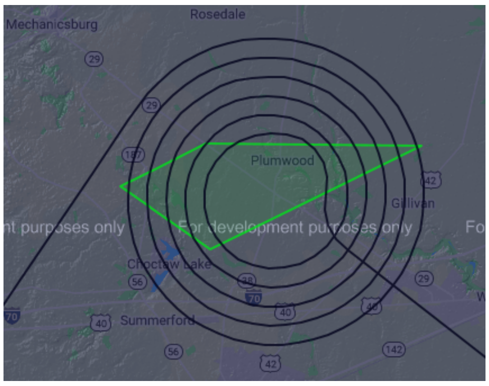


(a) Area covered in a Spiral In search pattern

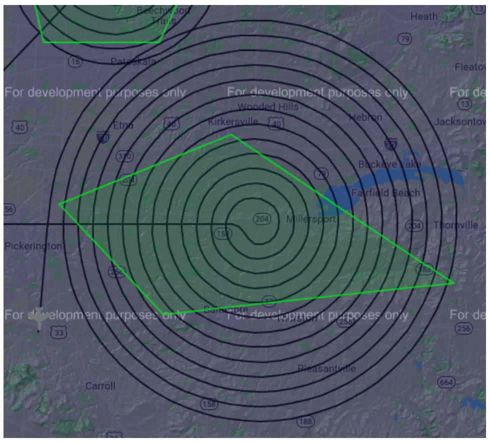


(b) Area covered in a Spiral Out search pattern

Figure 6.7: Area covered in a Spiral search pattern



(a) Spiral Out performed in the small area



(b) Spiral Out performed in the flatten area

Figure 6.8: Spiral Out experiments with less and highest area coverage

without little variation (standard deviation of 1.4%) between experiments. This value is probably not closer to 100% due to the way the metric is calculated, which doesn't work perfectly with constant curvatures. The area covered does, however, suffer variations.

As we can see in the Figures 6.7, the area covered depends mainly on the type of aircraft and area shape. In fact, this pattern does not cover the entire area due to the circle it leaves in the center, as we can see in Figures 6.8. This means that what really affects the area coverage is the minimum turn radius of the aircraft and the size of the mission area. The experiment shown in Figure 6.8a was run by the Beechcraft Baron, with a separation of 1000 meters in the small area. Figure 6.8b was run by the Cessna Skyhawk with 1000 meters of separation.

6.2 Pattern Comparison

Although studying the efficiency of the patterns is important to understand how the different parameters affect them, it does not provide a comparison between them. In order to understand which pattern has the best result for each area, we need to compare the absolute metrics such as the time spent and the distance covered. The average speed is always equal to the cruise speed of the aircraft used, so it is not worth comparing. On the other hand, the distance covered is equal to the time spent times the average velocity, which means using it to compare patterns is the same as simply using the time spent. The area covered, although it is a relative metric, is also a good way to compare the various patterns. However, from Section 6.1, it was already concluded that all the search patterns have a really good area coverage except the Spiral In and Out. This leaves us with the time spent as main metric to compare the different search patterns.

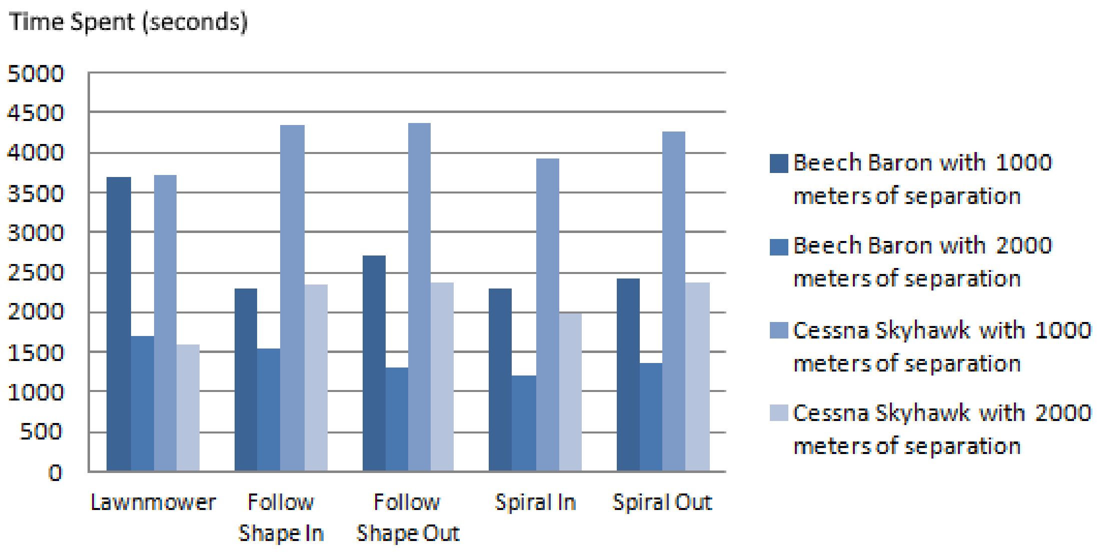


Figure 6.9: Time Spent on Different Patterns in a Squared Area in seconds

The chart 6.9 seems to indicate that the best pattern for a squared shape area depends on the aircraft used. In fact, it was already confirmed in figure 6.2a that a large turn radius affects the Lawnmower pattern more than the other ones. It is also possible to discover two interesting results:

- The speed of the Beechcraft Baron seems to compensate the increased path distance in the same rate, making the time spent equal as when using the Cessna Skyhawk.

- All the other four patterns seem to have little difference in the time spent.

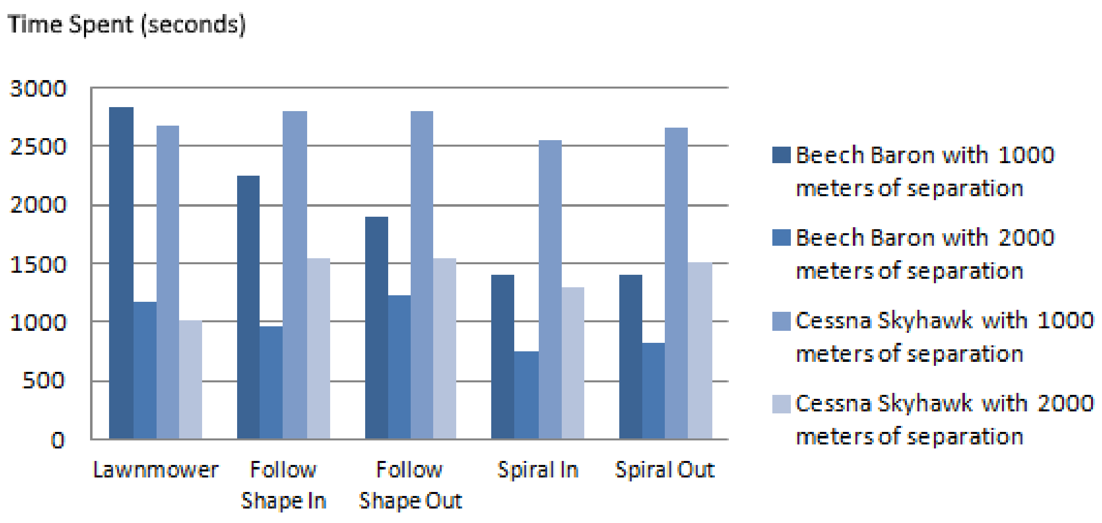


Figure 6.10: Time Spent on Different Patterns in a Pentagon Shaped Area in seconds

From Figure 6.10 it is possible to infer that the best pattern also depends on the aircraft used. However, even when using the Cessna Skyhawk, the Lawnmower pattern seems to be really close to the other ones. It is also possible to infer that the Spiral In/Out seem to be more efficient than the Follow Shape In/Out.

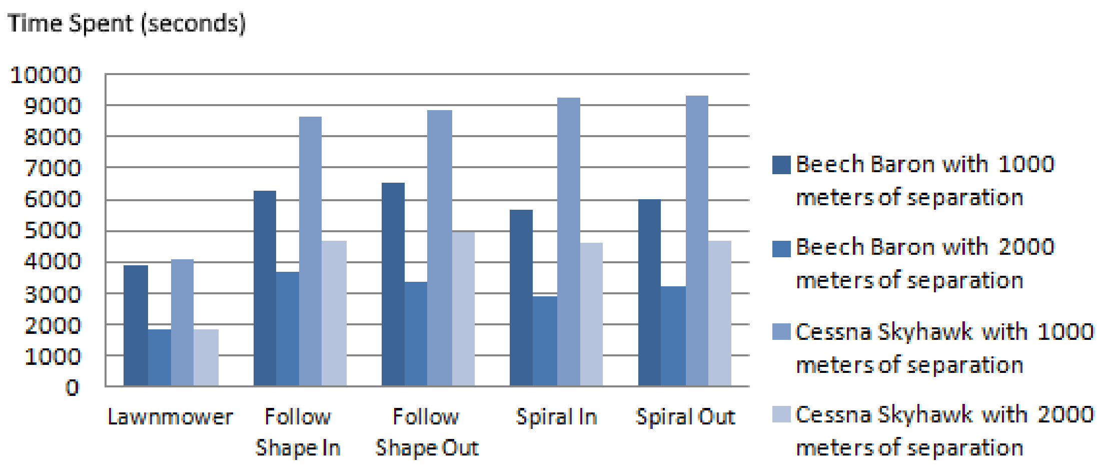


Figure 6.11: Time Spent on Different Patterns in a Flatten Area in seconds

Chart 6.11 confirms the idea that the Lawnmower pattern is the best one to handle areas in which the boundary points differ greatly in distances to the center. All the other patterns seem to also be really similar in terms of time spent on covering the area. Once again, the speed of the Beechcraft Baron compensated the increased distance in the same proportion.

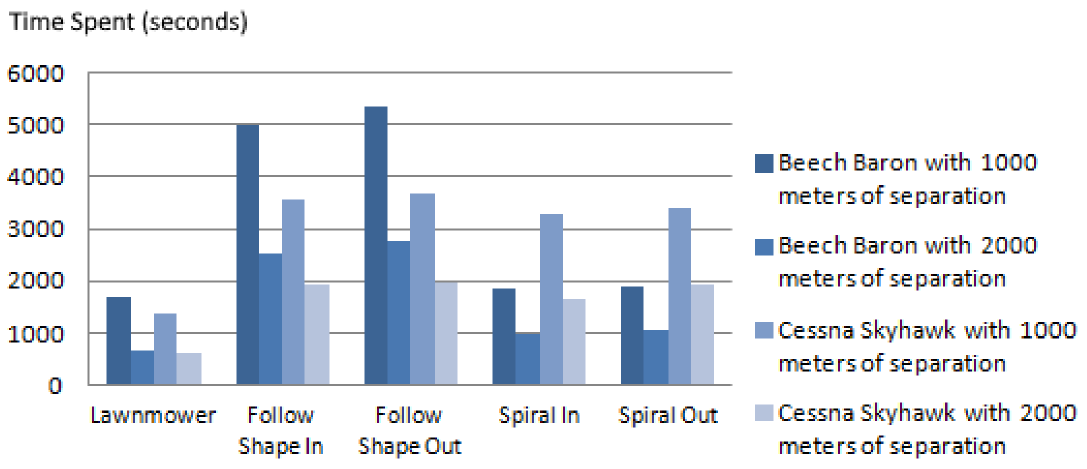


Figure 6.12: Time Spent on Different Patterns in a Small Area in seconds

After analysing the Figure 6.12, the Lawnmower pattern seems to be the best one again. However, it is hard to accredit that to the size of the area because the small area is also a flatten area. It is also hard to assume that the Spiral In/Out patterns are better than the Follow Shapes ones since the values of the Beechcraft Baron on them are outliers (resulted from missing waypoints and making numerous unnecessary turns).

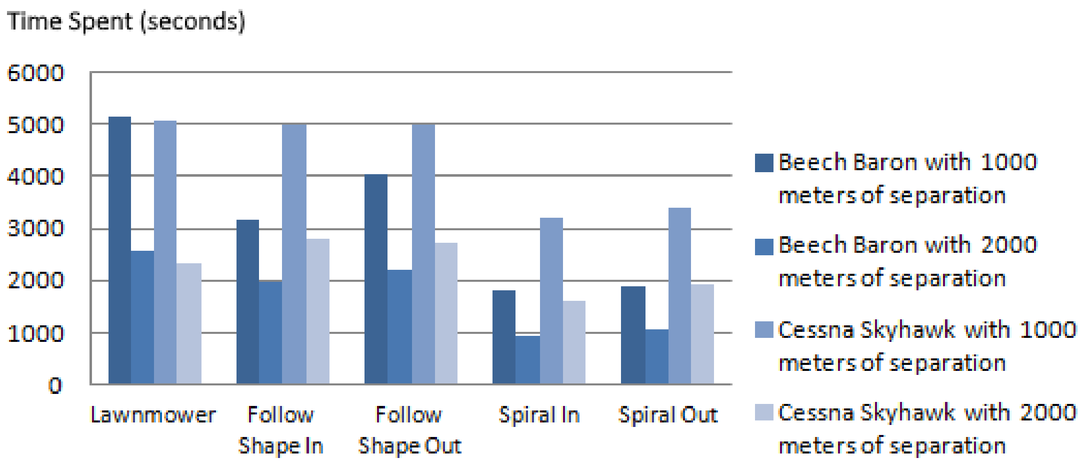


Figure 6.13: Time Spent on Different Patterns in a Circle Area in seconds

Figure 6.13 confirms that when performing a measure mission in a circle shaped area, the Spiral search pattern is the best no matter the parameters used.

By looking at the all the charts at the same time it is possible to infer some interesting connections:

- Separation is the parameter that affects the time spent the most, unsurprisingly.
- Spiral In/Out seems to be better than Follow Shape In/out the closer the area shape is from a circle. It would be interesting to confirm this hypothesis by running test in a triangle shaped area, and confirm that the Follow Shape patterns are better in said area.
- The different aircrafts seem to end up spending the same time performing a Lawnmower search pattern. It would be interesting to test other aircraft and check if it is a coincidence or not.
- The aircraft Beechcraft Baron is faster than the aircraft Cessna Skyhawk in all search patterns except the Lawnmower one. Important to say that it doesn't mean it is always better to use it when executing a mission, since other metrics can be important as well, such as fuel for example.
- The more "flat" the area is, the better is the Lawnmower search pattern compared to the others.

It is also important to notice that the implementation of these patterns should change in order to be more open to the intent of the user. For example, the Follow Shape search pattern tries to maximize area coverage, while spending more time in the process. The user could prefer less area covered in order to minimize the time spent. The same could be said for the Spiral search pattern in terms of maximizing area coverage by passing through the center.

Chapter 7

Conclusion and Future Work

The implementation of the various missions fell short to what was initially planned. Not only there were major set backs due to the lack of some key features in the platform, the amount of bugs and errors detected in edge cases delayed the entire process. Even in the end there are some undesired results from the implementation, such as the Follow Shape search pattern not working correctly with large minimum turn radius. On the other hand, the calculation of the metrics also fell short to the initial expectations. Most of the planned metrics were discarded, and even the ones who weren't have some problem with it's calculations, such as the area covered and overlapped, which don't work correctly on turns. After the study of the metrics obtained, it was discovered that there wasn't a perfect pattern that would be better in every single mission. It was concluded the various parameters used in the mission affect the performance in a way that some patterns can be better than others. However, it was possible to gather some insights that allow us to suspect of the best pattern to choose, given certain parameters:

- Spiral search patterns are better the closer the area boundary is to a circumference.
- Faster aircrafts with a bigger turn radius are better for Spiral and Follow Shape search patterns and worse for Lawnmower ones. However, a big turn radius might affect the possibility of even executing those patterns.
- Lawnmower search pattern is better than the others when covering flatten areas.

These conclusions depend, however, on the implementation done in this project. Some improvements can be made that can change how the different patterns can interact with different parameters. There are also more metrics that can be calculated that could change the way the efficiency was calculated. For example, if in the future the fuel is actually obtained (either from upgrading to FSX 2020 or outside information), this could make the use of faster aircrafts seem worse than they look now. In the end, a lot of work needs to be done, mainly in the implementation of the mission, since this version was a simplified one in order to obtain the metrics necessary to study the search patterns efficiency. The mission definition language should also be updated in order to allow more customization of the patterns executed. Albeit the metrics could also be upgraded, such improvement wouldn't be that important in the Platform, at least compared to the improvements needed in mission management and execution. It could be also interesting to study the impact that no-fly zones and air traffic can have in mission performance.

References

[Almeida, 2017] Almeida, J. F. S. (2017). Simulation and Management of Environmental Disturbances in Flight Simulator X. Master's thesis, Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal.

[Arbeit, 2013] Arbeit, A. C. (2013). Adaptation and automation of search and rescue patterns with implementation for an operational unmanned aircraft system. Master's thesis, Department of Aeronautics and Astronautics, University of Washington, Washington, United States of America.

[Benjamin et al., 2019] Benjamin, M., Schmidt, H., and Newman, P. (2019). Moos-ivp documentation. https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php.

[Besada et al., 2018] Besada, J. A., Bergesio, L., Campaña, I., Vaquero-Melchor, D., López-Araquistain, J., Bernardos, A. M., and Casar, J. R. (2018). Drone Mission Definition and Implementation for Automated Infrastructure Inspection Using Airborne Sensors. *Sensors*, 18:27–44. DOI: 10.3390/s18041170.

[Bosse et al., 2007] Bosse, M., Nourani-Vatani, N., and Roberts, J. (2007). Coverage Algorithms for an Under-actuated Car-Like Vehicle in an Uncertain Environment. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 698–703.

[Damasceno, 2020] Damasceno, R. (2020). *Co-Simulation Architecture for Environmental Disturbances*. Master's thesis, University of Porto.

[DoD, 2005] DoD (2005). Unmanned aircraft systems roadmap, 2005-2030. Technical report, Office of the Secretary of Defense.

[Elston and Frew, 2008] Elston, J. and Frew, E. W. (2008). Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks. In *2008 IEEE International Conference on Robotics and Automation*, pages 170–175.

[Ferreira, 2018] Ferreira, L. (2018). Montagem e Integração de um Quadcopter numa Plataforma. Master's thesis, Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal.

[Garrido, 2019] Garrido, D. L. G. (2019). Fault injection, detection and handling in autonomous vehicles. Master's thesis, Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal.

[Goh and Fan, 2019] Goh, Y. and Fan, S. (2019). Path Planning for AUV Area Coverage Mission based on MOOS-IvP. In *2019 IEEE Underwater Technology (UT)*, pages 1–10.

[Gupta et al., 2018] Gupta, A., Novitzky, M., and Benjamin, M. (2018). Learning Autonomous Marine Behaviors in MOOS-IvP. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–10.

[Incorporated, 2004] Incorporated, M. (2004). Cost & Business Model Analysis for Civilian UAV Missions. Technical report, Moiré Inc.

[Labib et al., 2019] Labib, N. S., Danoy, G., Musial, J., Brust, M. R., and Bouvry, P. (2019). Internet of unmanned aerial vehicles—a multilayer low-altitude airspace model for distributed uav traffic management. *Sensors*, 19(21)(4779).

[Microsoft Corporation, 2008a] Microsoft Corporation (2008a). ESP SDK Overview. Microsoft Docs. Available online at `https://docs.microsoft.com/en-us/previous-versions/microsoft-esp/cc526948(v=msdn.10)`. Last seen in 2020-07-11.

[Microsoft Corporation, 2008b] Microsoft Corporation (2008b). SimConnect SDK Reference. Microsoft Docs. Available online at `https://docs.microsoft.com/en-us/previous-versions/microsoft-esp/cc526983(v=msdn.10)`. Last seen in 2020-07-11.

[Murta, 2009] Murta, A. (2009). Gpc – general polygon clipper library. http://www.cs.man.ac.uk/ toby/alan/software/.

[Neto, 2017] Neto, T. L. P. (2017). Flexible air traffic control. Master's thesis, Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal.

[Roberts, 2009] Roberts, L. (2009). How to calculate an angle from three points? https://stackoverflow.com/questions/1211212/how-to-calculate-an-angle-from-three-points.

[Rollo et al., 2017] Rollo, M., Selecký, M., and Volf, P. (2017). Simulation of uas integration into shared airspace for validation of impact on atm systems. In *2017 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 6D2–1–6D2–7.

[Selecký et al., 2013] Selecký, M., Štolba, M., Meiser, T., undefinedáp, M., Komenda, A., Rollo, M., Vokříne, J., and Pěchouček, M. (2013). Deployment of multi-agent algorithms for tactical operations on uav hardware. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, page 1407–1408, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Selecký et al., 2017] Selecký, M., Faigl, J., and Rollo, M. (2017). Mixed reality simulation for incremental development of multi-uav systems. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1530–1538.

[SESAR, 2016] SESAR (2016). European drones outlook study. Technical report, SESAR Joint Undertaking.

[Shah et al., 2017] Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*.

[Silva, 2011] Silva, D. C. (2011). *Cooperative Multi-Robot Missions: Development of a Platform and a Specification Language*. PhD thesis, Faculty of Engineering, University of Porto, Porto, Portugal.

[Silva et al., 2014] Silva, D. C., Abreu, P. H., Reis, L. P., and Oliveira, E. (2014). Development of a Flexible Language for Mission Description for Multi-Robot Missions. *Information Sciences*, 288:27–44.

[Vitor et al., 2019] Vitor, R., Keller, B., D'Angelo, T., Azpurua, H., Bianchi, A. G. C., and Delabrida, S. (2019). Collaborative teleoperation evaluation for drones. In *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, IHC '19, New York, NY, USA. Association for Computing Machinery.

[Williams, 2004] Williams, K. W. (2004). A Summary of Unmanned Aircraft Accident/Incident Data: Human Factors Implications. Technical report, Office of Aerospace Medicine, FAA. Report No: DOT/FAA/AM-04/24.

[Šišlák et al., 2012] Šišlák, D., Volf, P., Pechoucek, M., Cannon, C., Nguyen, D., and Regli, W. (2012). Multi-agent simulation of en-route human air-traffic controller. *Proceedings of the National Conference on Artificial Intelligence*, 3:2323–2328.