FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mixed Initiative Planning and Control of UAV Teams for Persistent Surveillance

João Filipe Fortuna Araújo

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Advisor: João Tasso Borges de Sousa (Eng.)

July 31, 2012

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | 2011/2012

A Dissertação intitulada

"Mixed Initiative Planning and Control of UAV Teams for Persistent Surveillance"

foi aprovada em provas realizadas em 27-07-2012

o júri

[illegible]

Presidente Professora Doutora Maria do Rosário Marques Fernandes Teixeira de Pinho
Professora Associada do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

[illegible]

Professor Doutor José Augusto Nunes Vicente Passos Morgado
Tenente Coronel do da Força Aérea Portuguesa

[illegible]

Mestre João Tasso de Figueiredo Borges de Sousa
Assistente Convidado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

[illegible]

Autor – João Filipe Fortuna Araújo

Faculdade de Engenharia da Universidade do Porto

Resumo

Este trabalho debruça-se sobre o planeamento e controlo de trajectos para equipas de veículos aéreos não tripulados (UAVs), ou seja, gerar planos para os veículos e seguir a sua execução, intervindo quando necessário.
Em particular, é focada a interação homem-computador que visa a recolher *inputs* de humanos e de computadores e combiná-los de modo a criar uma solução que toma partido dos pontos fortes de ambos os intervenientes, designada como iniciativa mista. A iniciativa mista é um processo que combina informações e opções de um operador com sugestões geradas automaticamente. O seu objetivo é melhorar a capacidade de interação do operador com os algoritmos e processos automáticos, guiando o operador na explicitação de informações e opções. Nesta tese, a iniciativa mista será usada para criar planos para equipas de UAVs e para controlar a sua execução.
Os planos gerados serão otimizados para um tipo de missão designado por vigilância persistente. Vigilância persistente consiste em recolher dados (fotografia aérea, por exemplo) regularmente de uma dada posição. Em termos militares, o período entre capturas pode variar entre poucos minutos e vários dias, no entanto, esta tese será focada em intervalos de tempo de minutos ou horas.
Os veículos robóticos têm vindo a evoluir imensamente durante os últimos anos, atingido um elevado nível de autonomia na execução de manobras e trajetos. Os UAVs conseguem desempenhar tarefas que permitem retirar os humanos de atividades perigosas ou entediantes.
As forças militares são os principais interessados neste tipo de tecnologias, mas a utilidade destes sistemas vai muito além das aplicações militares. Devido ao seu tamanho reduzido (normalmente, mas não exclusivamente) e por não terem tripulação, os UAVs são mais baratos de voar do que os seus equivalentes tripulados, no entanto, a logística de operação atual requer uma equipa de pilotos e operadores para cada veículo, reduzindo a vantagem do custo operacional mais baixo.
Esta tese apresenta soluções para inverter o atual rácio de um UAV para vários agentes humanos. O objetivo é permitir a um único piloto gerir vários UAVs para missões de vigilância persistente. Outro aspeto importante do sistema desenvolvido é a capacidade de integrar conhecimentos de vários agentes.
De modo a facilitar a recolha rápida de dados dos UAVs por parte do piloto, em qualquer altura, algumas técnicas presentes em jogos de simulação foram usadas.
Resultados de simulações de *software* e *hardware in the loop*, assim como testes de campo provaram a capacidade de um único piloto gerir uma frota de UAVs para efetuar vigilância persistente usando o sistema desenvolvido.

Abstract

This thesis focuses on planning and control of trajectories for Unmanned Aerial Vehicles (UAVs), that is, creating paths for the vehicles and following their execution, stepping in when intervention is necessary.
Particularly, a human-computer interaction is analyzed which aims at collecting inputs from both human and computer to combine them and generate a solution which makes use of the skills of both, this interaction is called mixed initiative. Mixed initiative is a process where the human operators interact with the planning and control loops. Its goal is to enhance the ability of the operator to interact with the algorithms and automatic processes, offering guidance to the operator. In this thesis, mixed initiative will be used to create plans for teams of UAVs and to control their execution.
The generated plans will be optimized for a type of mission which is persistent surveillance. Persistent surveillance consists of regularly capturing data from a given location, which can be aerial imagery for example. According to the military definition, the period between captures can be of a few minutes up to several days, however, this thesis will focus on time intervals of minutes to hours.
Robotic vehicles have evolved hugely over the last years, having achieved a high level of autonomy which allows them to follow complex paths and execute maneuvers. UAVs can now perform many tasks which remove humans from dangerous or tedious activities.
Currently, military forces are the main stakeholders and benefiting the most from such technologies, but the usefulness of these systems goes well beyond the scope of military applications. Due to the lack of on-board pilot and the small size of most UAVs, they are cheaper to fly than full size aircraft. However, current operation logistics require a crew of pilots and operators for each vehicle, reducing the lower cost advantage.
This thesis presents solutions to change the current paradigm of a single UAV requiring a large team, to a single operator managing a fleet of vehicles. The goal is to allow a single pilot to manage several UAVs for persistent surveillance missions. Another important aspect of the developed system is the capability to integrate knowledge from several agents.
In order to allow the rapid collection of data from the team of UAVs by the pilot, intuitively and at any time, some techniques present in strategy games were used.
Results from both software and hardware in the loop simulations as well as flight tests proved the ability of a single pilot to manage a fleet of UAVs to perform persistent surveillance using the developed system.

Acknowledgments

To my family and my girlfriend who endured my mood during the most difficult times and my absence through the whole of this semester I want to say a big thank you.
I wish to thank all my colleagues and friends at AsasF and all the LSTS people who helped with the technical difficulties I had, but also for the fun moments throughout this year. To Ricardo Martins, José Pinto, Paulo Dias and Eduardo Marques, who put up with all my questions about Dune and Neptus, I express my gratitude. I also want to acknowledge the contribution of Ricardo Bencatel to my work and thesis and also for being a skillful safety pilot during the field tests. Thank you Ricardo Bencatel, Sérgio Ferreira, Joel Cardoso, Filipe Ferreira and Joel Gomes for making me feel at home, introducing me to the project and for the challenging discussions that made me go further.
I would like to extend my gratitude to my advisor, Prof. João Sousa, for all the support given which was crucial to the development of this thesis and the research that led to it.
To the great friends I made during my 5 years at Faculdade de Engenharia da Universidade do Porto I want to thank for all the awesome moments and for helping me enjoy the best of these years.

João Fortuna

"The whole fun of living is trying to make something better"

Charles Kettering

Contents

List of Figures

List of Tables

Acronyms

AFA	"Academia da Força Aérea" – Portuguese Air Force Academy
AGL	Above Ground Level
AI	Artificial Intelligence
AUV	Autonomous Underwater Vehicle
DSS	Decision Support System
DSV	Decision Support Visualisation
FAP	"Força Aérea Portuguesa" – Portuguese Air Force
FEUP	"Faculdade de Engenharia da Universidade do Porto" – University of Porto, School of Engineering
GCS	Ground Control Station
GPS	Global Positioning System
GUI	Graphical User Interface
HiL	Hardware in the Loop
IMC	Inter-Module Communication
IMU	Inertial Measurement Unit
LSTS	"Laboratório de Sistemas e Tecnologia Subaquática" – Underwater Systems and Technology Laboratory
PITVANT	"Projeto de Investigação e Tecnologia em Veículos Aéreos Não-Tripulados" – Research and Technology in UAVs Project
ROV	Remotely Operated Vehicle
SA	Situational Awareness
SBS	System Breakdown Structure
SiL	Software in the Loop
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 UAV Operations

Unmanned Aerial Vehicles (UAVs), as the name suggests are aircraft which are remotely piloted and/or operated with varying levels of autonomy. UAVs are increasingly found in military, search and rescue operations.
Even though some of these UAVs are able to perform fully autonomous flights, human operators are needed to assign tasks to the vehicles as well as pilots for emergency situations.
By removing the human pilot from dangerous flight conditions, these systems are becoming essential for both civilians and military forces all over the world [18]. Because of their usual small size, making them hard to detect, UAVs have always been interesting for military application which throughout history was proven to be a motivation for investment.
Currently, UAV operations involve a team of pilots/operators/technicians per vehicle in a many-to-one structure, making them economically less efficient than what they could potentially be, and taking no advantage of the Situational Awareness (SA) possible with multiple aircraft. One of today's challenges for researchers in UAVs is to invert the structure to a one-to-many kind of operation, having a single pilot or operator controlling several UAVs and receiving selected/processed information from all of them in a way that allows him to easily make decisions and control the vehicles in a safe manner.

1.2 Background

The "Laboratório de Sistemas e Tecnologia Subaquática" – Underwater Systems and Technology Laboratory (LSTS) from Porto University has been designing, building and operating unmanned underwater, surface and air vehicle systems for innovative applications with strong societal impact since it was established in 1997. Currently the LSTS team has over 30 researchers, including faculty and students, with Electrical and Computer Engineering, Mechanical Engineering and Computer Science backgrounds.
The "Projeto de Investigação e Tecnologia em Veículos Aéreos Não-Tripulados" – Research and

Technology in UAVs Project (PITVANT) is a collaborative research and development project funded by the Portuguese Ministry of Defense. It is a joint operation of LSTS, at FEUP, and "Academia da Força Aérea" – Portuguese Air Force Academy (AFA) that aims at further developing UAV know-how and technology.
The work presented on this thesis is done so that it can integrate with technologies and systems currently used on PITVANT operations.

1.2.1 LSTS

LSTS pioneered the development and integration of autonomous and operator-assisted vehicles and sensor networks. The research group was created in '97 and since then, LSTS designed and developed:

- A Remotely Operated Vehicle (ROV) for the inspection of underwater structures.
- Light Autonomous Underwater Vehicle (AUV) for coastal oceanography.
- Low cost sensor modules for remote and/or networked environmental data collection.
- Acoustic navigation technology for multiple AUVs.
- Operational concepts for the coordinated operation of multiple AUVs.
- Control architecture for the coordination and control of multiple vehicles.

Sensor networks and networked vehicles can also be used together to build on each other strengths. For example, networked vehicles can be used to deploy the sensor networks themselves, especially in areas where accessibility is difficult.

1.2.2 PITVANT

Over the last years, LSTS devoted an intense effort to the development of feasible concepts for the networked operation of multiple vehicles and systems. As a consequence, LSTS started an ambitious joint program with the "Força Aérea Portuguesa" – Portuguese Air Force (FAP), PITVANT, with the following objectives:

- Design and build UAVs that will work as a sensor platform.
- Use up-to-date technology to provide useful data for military, civilian and research activities.
- Develop technologies for the operator assisted concurrent operation of multiple autonomous vehicles and sensor networks.
- Give the FAP the know-how to use and operate the UAVs.

The PITVANT began in 2008 and it is expected to end in 2015. Currently there are several master's and PhD students, researchers and engineers working full time on this project, both at "Faculdade de Engenharia da Universidade do Porto" – University of Porto, School of Engineering (FEUP) and AFA. This thesis focuses on research conducted at LSTS, using technologies developed by LSTS.

1.3 Objectives

This work will focus on developing and implementing algorithms for cooperative control of UAVs to aid in persistent surveillance operations. To guarantee that UAVs can do it efficiently, several optimization algorithms such as scheduling and path planning, need to be considered when creating and editing flight plans. However, even the best and more complex optimization algorithms do not produce the best solution at all times and in a timely manner. That happens usually because it is not possible to include all the information which influences the efficiency of the solution. Can the performance of such algorithms be increased by introducing more sensors? Can an automated system be as smart as an experienced human operator? Probably not.
However, another important question is: do we need a system that can fully replace the human in every aspect of a UAV operation? Maybe a system should know its autonomy limits and know how to ask for data from humans to fill in its knowledge gaps [19]. A human operator included in the planning and control loops introduces the concept of mixed initiative control.
Cooperative control of teams of UAVs is a complex problem which involves areas like control and communication. The usual approach is to decompose complex missions into simpler individual tasks and give those tasks to the vehicles. The execution of the tasks is supervised by a hierarchy of controllers. See Fig 1.1. These algorithms were implemented using technologies developed at LSTS.

1.4 Contributions

A system was developed that could effectively maintain persistent surveillance in a defined area, using multiple UAVs.
Mixed initiative was implemented using knowledge from the human operator to improve the performance of the autonomous system.

1.5 Structure

In chapter 2 some technical and theoretical background is given about UAVs in general, the PITVANT, its objectives, achievements and the software tool-chain used during development.
In chapter 3, the State of the Art is analyzed. Current developments from researchers of the field that motivate and act as a starting point to the work done.

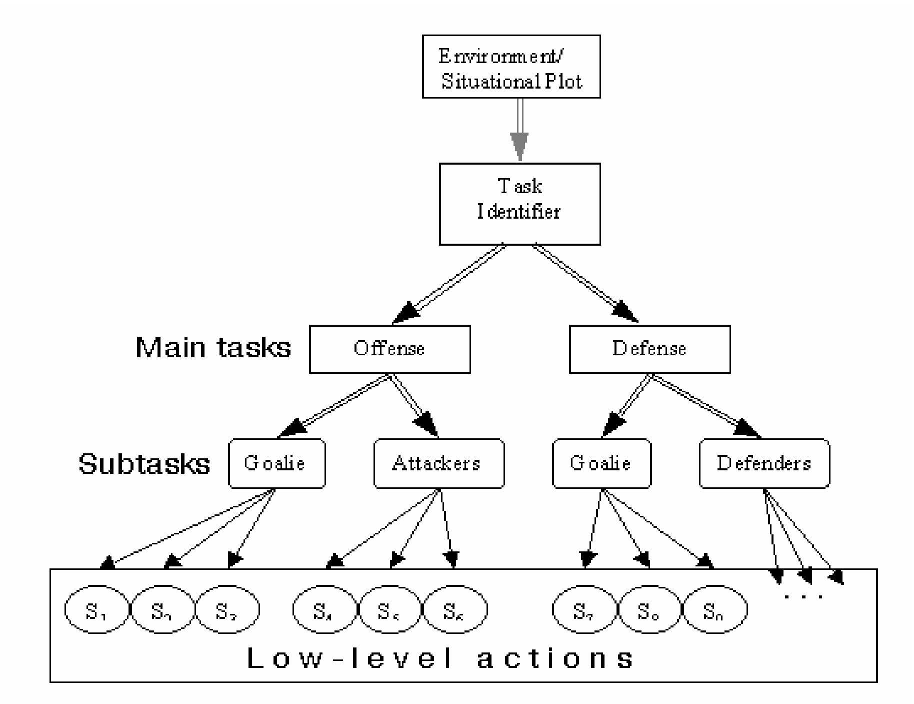


Figure 1.1: Cooperative Control Hierarchy in Robot Soccer – from [1]

In chapter 4, the problem is defined in a formal manner. This formalization allows the solution to be simplified and to measure its performance.
In chapter 5 the studied algorithms are discussed and the approach to the problem is described.
In chapter 6, the results are discussed and some projections are made to larger scenarios.
Finally, the last chapter, includes the conclusions taken from all the work done during the development of this project and resulting thesis.

Chapter 2

Background

2.1 Introduction

This chapter provides background material on UAVs, on the PITVANT project and on the LSTS software tool chain.

2.2 UAVs

An Unmanned Aerial Vehicle, sometimes known as drone, is an airborne system without a human pilot on board.

2.2.1 Brief History

The first UAVs were as simple as balloons loaded with explosives and little control over the flight (in 1849, about 20 years before the Wright brothers were even born!). Entering the 20th century, fixed-wing aircraft (commonly known as airplanes) were invented and a whole new domain for Unmanned Aircraft System (UAS) was unveiled.
Military applications have always been a major development motivation for UAVs and technology in general, so it is no surprise that during World War I many designs and prototypes for UAVs were created.
Nikola Tesla was one of the first to describe a fleet of unmanned aerial combat vehicles (in 1915) and the first attempt at a powered UAV was made in 1916 with a flying bomb.
Unsurprisingly, during World War II there were many developments in all fields of military technology, and UAVs were not excluded. Radioplane OQ-1 (and 2) were the first mass-produced UAVs (Fig. 2.1).
During the Cold War, the first jet-powered UAV was built - the Ryan Firebee (Fig. 2.2). Even though there were many produced UAVs, most were remotely piloted or autonomous with very little control over the followed trajectory, UAVs were unmanned but not autonomous.
Modern age UAVs appeared in the last two decades of the 20th century after the invention and

Figure 2.1: Radioplane Launch – from [2]

Figure 2.2: Ryan Firebee – from [3]

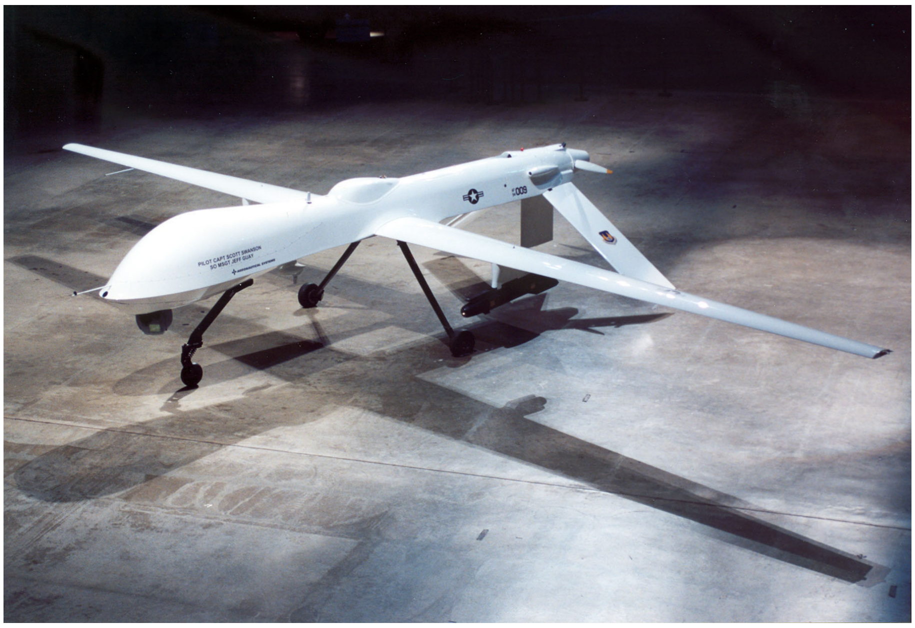


Figure 2.3: MQ-1 Predator – from [4]

activation of the Global Positioning System (GPS). The well known Predator (Fig. 2.3) was first flown in 1995, and it was piloted from the ground in a fake cockpit which made the pilots feel as if they were inside the airplane. This procedure is still used in the active Predators, but they have a higher degree of autonomy.
Fully autonomous UAVs are currently a major area of research, however due to the classified nature of the information it is unknown exactly how autonomous are the military aircraft owned by each country.

2.2.2 Typical UAV configuration

UAVs can be made in different shapes and configurations. These are roughly the same as manned aircraft, so the same terminology can be used. There are many levels of detail to which one can go to classify aircraft but the following superficial divisions are enough to clearly understand the rest of the report.

2.2.3 Lighter than air aircraft

Also known as aerostats, this kind of aircraft uses the principle of buoyancy to float in the air, much like a boat floats in the water. It usually has a large pocket of a gas less dense than the air around it (hot air, helium, hydrogen). They can stay still in midair. Examples are hot-air balloons, airships (Zeppelins), blimps and helikites.

- can hold a position in midair without using energy
- cannot fly at high speeds due to the large drag

- require a very large gas pocket to lift a reasonable weight

2.2.4 Heavier than air aircraft

Also known as aerodynes, they require motion relative to the surrounding air to generate lift. The kind of motion used can help us separate the vehicles into different classes:

2.2.4.1 Rotary-wing aircraft

The part of the aircraft that creates lift is rotating, the most common vehicle of this type is the helicopter, however there are more. A usual configuration for UAVs is the multi-copter (quadcopter, hexacopter, octacopter).

- allows both slow and fast flight, however, the rotating blades limit the maximum speed
- can hold a position in midair but needs to use energy
- fuel consumption is larger than that of airplanes to carry the same payload
- cannot glide and turn off engines/motors to save energy (low endurance)

2.2.4.2 Fixed-wing aircraft

This common type of aircraft includes most variations of airplanes (gliders, flying wings, space shuttles, etc).
Lift is caused by the motion of wings and body of the aircraft so it requires a minimum airspeed.

- slow flight might be difficult to attain (depends on the aircraft)
- fastest flight of all vehicle types
- cannot hold a position in midair
- can glide and turn off engine/motor to save energy (potentially high endurance)
- safer to handle
- inherently more stable than rotary-wing aircraft

Because of the good characteristics of this vehicle configuration, it is widely used as UAV. All the UAVs developed by PITVANT are fixed-wing aircraft.

2.2.5 Dynamics

In order to fully understand the results and research presented in this thesis it is necessary to be familiar with basic fixed-wing aircraft dynamics. Figure 2.4 shows the three axes which describe the airplane attitude (and their names) as well as the control surfaces.
The roll angle, also called bank, is controlled by the Ailerons (on the wings). The yaw angle is

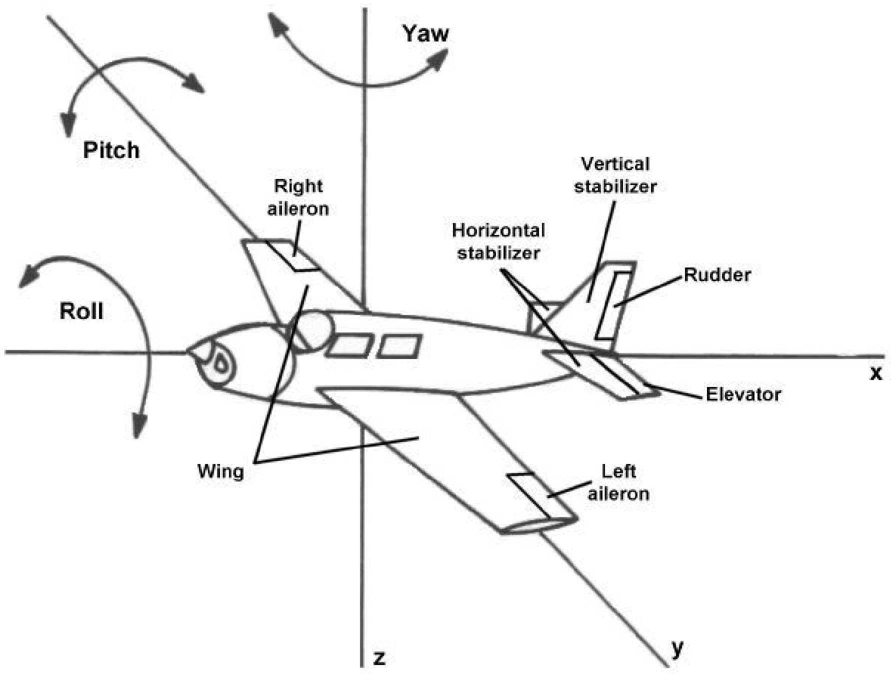


Figure 2.4: Fixed-wing flight dynamics and control surfaces – from [5]

controlled by the Rudder (on the tail). The pitch angle is controlled by the Elevators (on the tail). Unlike cars that directly change the yaw angle to turn, airplanes bank to turn when a change of heading is needed. That means that a fixed camera placed on the belly of the UAV will only point to the ground (and collect useful ground surveillance data) when the airplane is flying straight (or almost). This has some implications when developing algorithms to implement ground surveillance and it will be discussed again later on this report.

2.3 PITVANT Project

The "Projeto de Investigação e Tecnologia em Veículos Aéreos Não-Tripulados" – Research and Technology in UAVs Project is a joint venture of "Faculdade de Engenharia da Universidade do Porto" – University of Porto, School of Engineering and "Academia da Força Aérea" – Portuguese Air Force Academy, it started in 2009 and is scheduled to finish at the end of 2015. So far, approximately 30 UAVs have been or are part of PITVANT's fleet and there is a constant expansion. AFA has developed and built many of these platforms. UASs include a number of subsystems (see Fig. 2.6).
The PITVANT brings together knowledge from different countries due to the international cooperation with several European military forces and industries.
Several tools and platforms have been developed at "Laboratório de Sistemas e Tecnologia Subaquática" – Underwater Systems and Technology Laboratory :

- Neptus

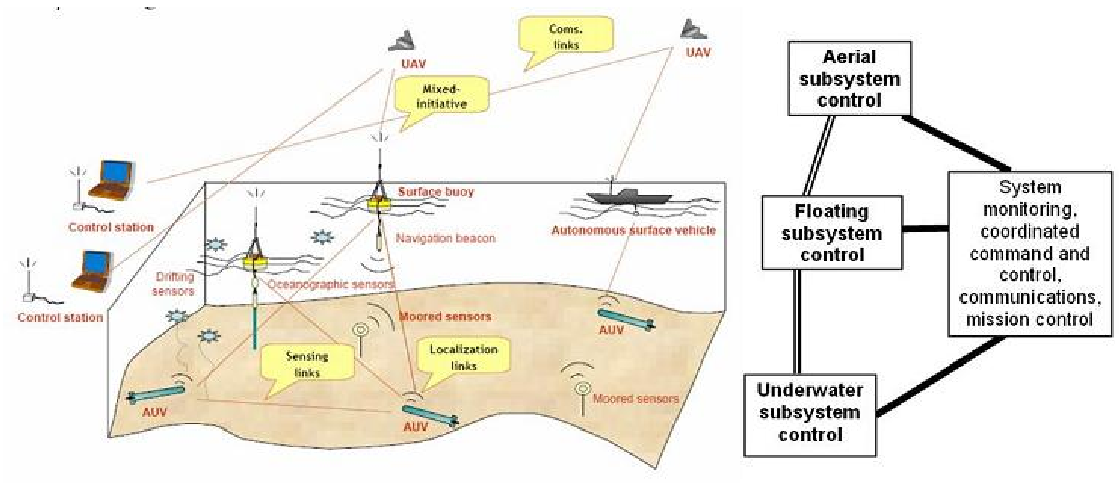


Figure 2.5: UAV and AUV joint Mission

- Dune
- IMC
- AUVs, ROVs and UAVs

The "Laboratório de Sistemas e Tecnologia Subaquática" – Underwater Systems and Technology Laboratory , as the name suggests was created for AUV research, so one of the interesting Mission Scenarios will be interaction between UAVs and AUVs, see Fig. 2.5.
Persistent surveillance and mixed initiative control are of great interest for the Air Force, and an important reason for their investment in the PITVANT.

2.3.1 Objectives

At PITVANT there is an effort to develop cooperative control for multiple UAVs with mixed initiative, data fusion and navigation systems. Some of the project's objectives are:

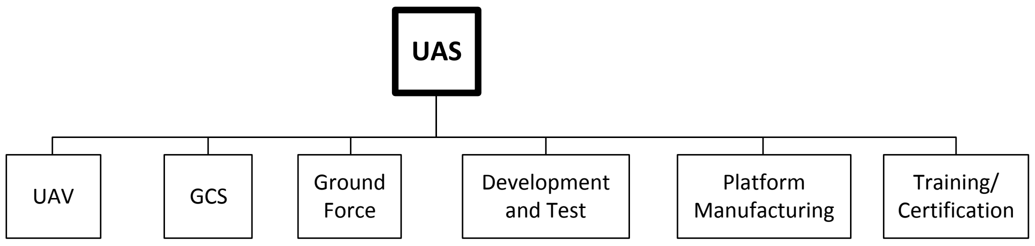


Figure 2.6: Simplified System Breakdown Structure

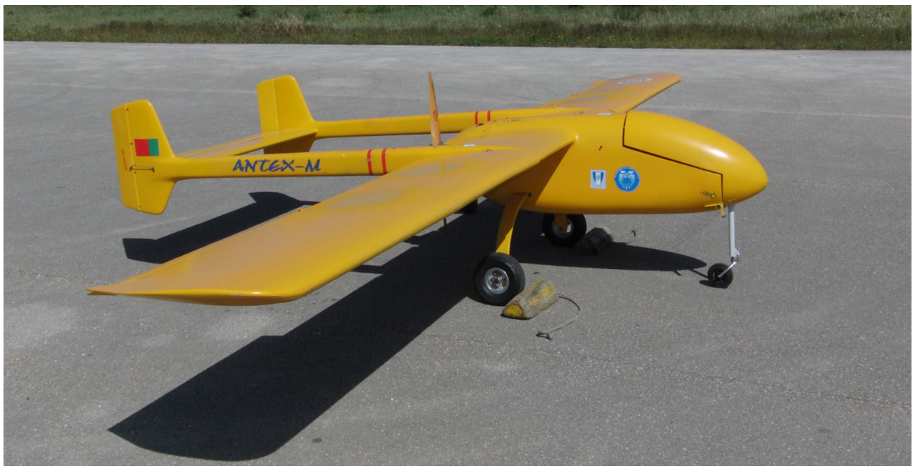


Figure 2.7: ANTEX-X03 model

- Design and build UAVs to act as sensor platforms for remote data collection.
- Develop mixed initiative control architectures to allow the usage of multiple UAVs for co-operative missions
- Train AFA and FAP personnel to pilot and extract data from UAV missions.

2.3.2 Platforms

The project has been integrating and developing three classes of UAVs defined by their size are as follows:

- Large - Ex.: ANTEX-X03 series (see Fig. 2.7), not at LSTS because of its size (see Fig. 2.11 for comparison) but a part of the PITVANT;
- Medium - Ex.: Pilatus (see Fig. 2.8) and ANTEX-X02 (a.k.a. Alfa) series (see Fig. 2.9);
- Small - Ex.: Cularis (see Fig. 2.10) series;

2.3.2.1 Cularis

Cularis is a light-weight sailplane (or glider) with an electrical motor (see Fig. 2.10). Its wingspan is roughly 2.5 m and presents an autonomy of around 30 minutes.
Because of its simplicity of deployment it is the perfect testing tool for multiple UAVs cooperative missions. It was the main platform used for field tests.

2.3.3 Accomplishments

Since the project began, there has been a constant evolution. Some of the important achievements include:

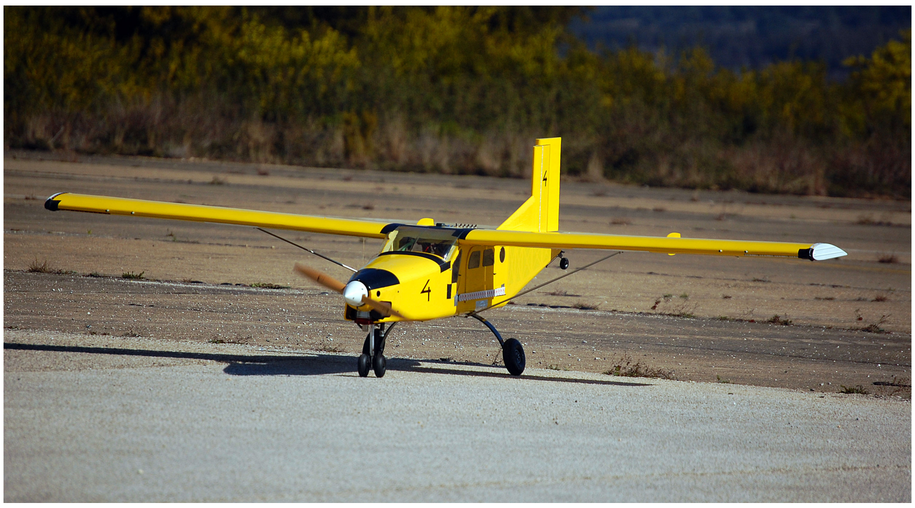


Figure 2.8: Pilatus model

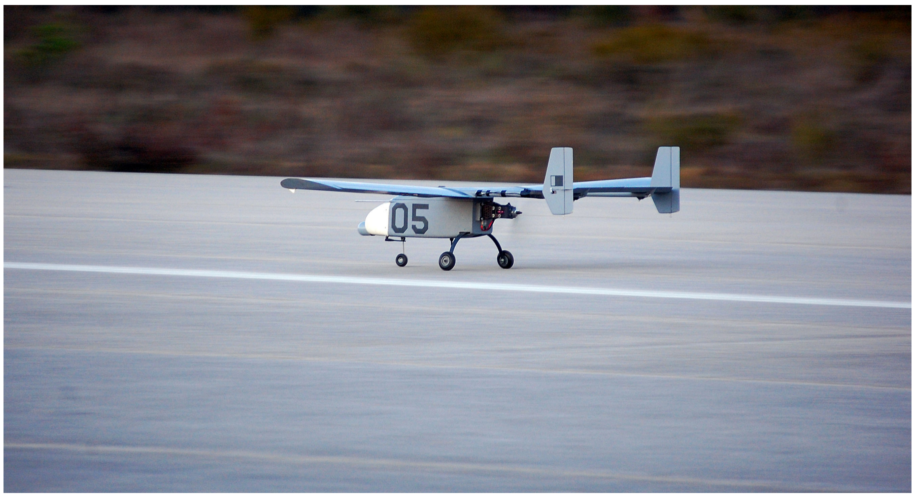


Figure 2.9: Alfa model (electric version)

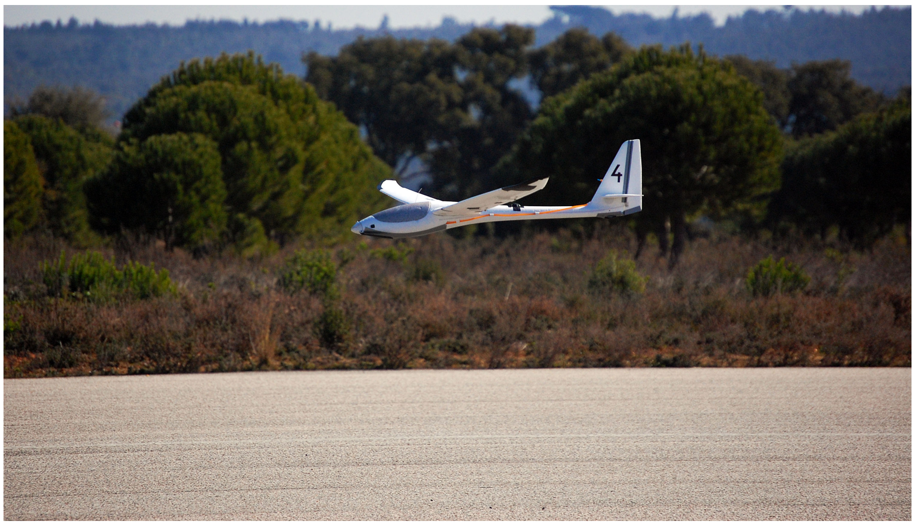


Figure 2.10: Cularis model

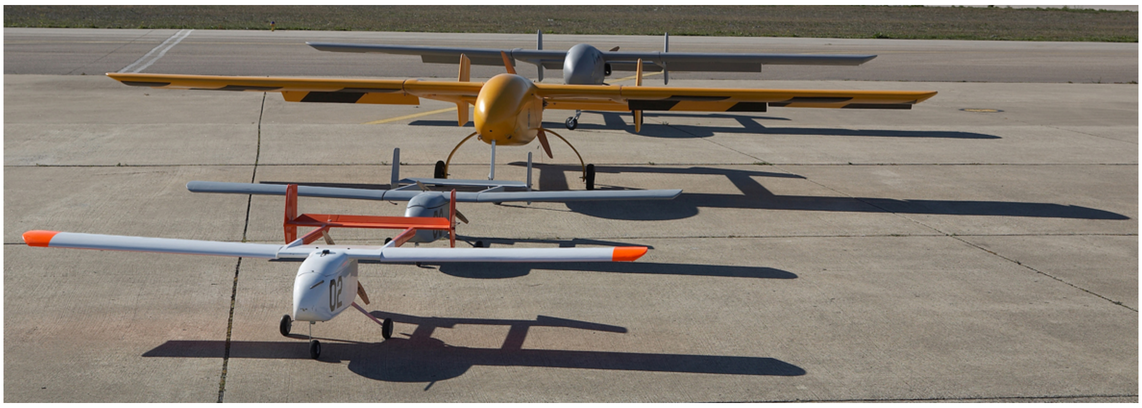


Figure 2.11: ANTEX models line-up

- PITVANT has performed over 600 UAV flights which total over 250 hours of flight time.
- Success rate is over 99% with only 3 unrecoverable crashes.
- Earlier this year, the first flight over sea was performed from Santa Cruz (Torres Vedras municipality) to the Berlengas islands (Peniche municipality) with a straight line distance of roughly 35 kilometers (see Fig. 2.12). The flight lasted about 80 minutes and the vehicle was moved by an electrical motor.

2.3.4 Autopilot

The autopilot is the system responsible for maintaining the vehicle safely in the air (see Fig. 3.2). It does so by commanding the aircraft's control surfaces (see Fig. 2.4). The autopilot software computes the required position for the control surfaces and its hardware sends signals to the actuators and reads data from sensors like GPS and an Inertial Measurement Unit (IMU) for positioning and attitude control.
At LSTS there are two different autopilot systems available: Ardupilot and Piccolo.

2.3.4.1 Ardupilot

Arduino is a cheap micro-controller with a programming language similar to C++. There is a community dedicated to developing an open source autopilot for Arduino, called the Ardupilot. Because of the low price, ease of programming and a base program to work on, Ardupilot is a good choice as a prototyping and testing tool.
The latest generation of Ardupilot is the Ardupilot-mega 2.0 (see Fig. 2.13), it comes with a GPS, an improved IMU, several I/O pins and a serial port.

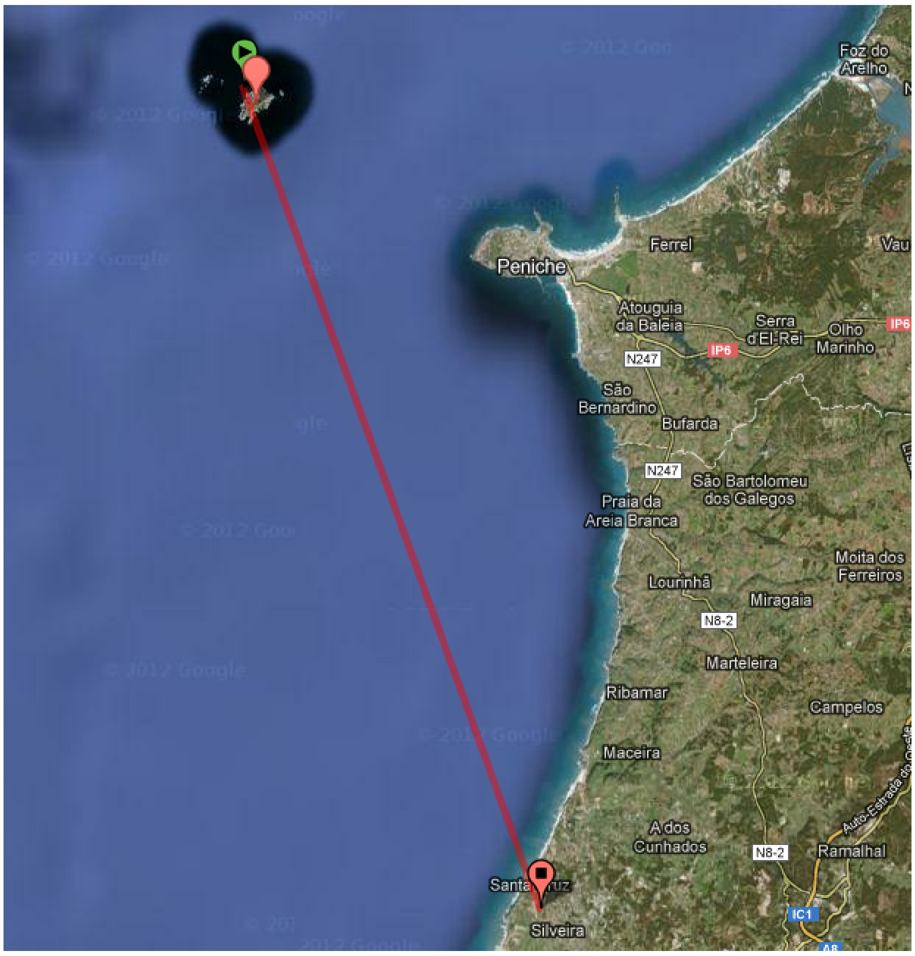


Figure 2.12: Santa Cruz to Berlengas flight

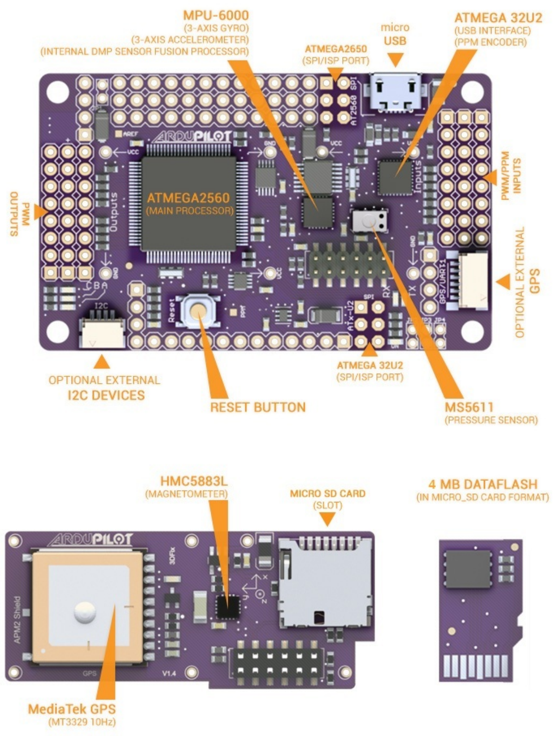


Figure 2.13: Ardupilot-mega 2.0 board overview – from [6]

2.3.4.2 Piccolo

Piccolo is a commercial autopilot developed by Cloud Cap Technology (see Fig. 2.14).
It provides a complete, off the shelf solution including the core autopilot, flight sensors, navigation, wireless communication, and payload interfaces [7].

2.3.4.3 Remarks

Both systems are available at LSTS, and compatible with the developed control and management software tool-chain. The Piccolo autopilot is more expensive than the Ardupilot (roughly 50 times) however, it offers a much higher level of quality and reliability. Although, because Ardupilot is open-source there is a possibility to implement new capabilities for specific needs.
A Ground Control Station (GCS) software is needed for either autopilot, for Ardupilot there are

Figure 2.14: Piccolo autopilot products – from [7]

some programs like Ardupilot Mission Planner or QGroundControl. For Piccolo, Cloud Cap Technology has developed Piccolo Command Center (PCC). Neptus can replace any of these GCSs and that is one of the goals for the Neptus development team.

2.3.5 Operation Setup

UAV missions require a set of systems, some of which are completely autonomous. However, there are systems who have to understand human decisions.
A mission setup includes:

- GCS - and certified Pilot
- Certified Safety Pilot
- UAV - and autopilot
- Communication system
- Possible on-board data collection and processing system (payload)

2.4 LSTS Software tool chain

2.4.1 Neptus

Neptus is a software developed at LSTS ([20]). It is a C3I (Command, Control, Communication and Information) infrastructure for the coordination and control of teams of multiple autonomous and semi-autonomous vehicles. It was initially designed for AUV mission planning, control and analysis, but gradually grew to offer that functionality for UAVs as well (see Fig. 2.15).
It is a Java based tool and it is constantly evolving to deliver more advanced features. Neptus allows interaction with multiple vehicles and it has been used to plan and supervise UAV missions. This software's architecture allows the creation of "consoles" with the desired plugins, it is then possible to build a console tuned for a chosen application. Plugins are developed to offer a new functionality and can be reused on different consoles allowing the creation of numerous consoles ideal for each type of mission or user.
With reference to figure 3.2, Neptus is in the outer Mission Management loop.

2.4.2 Dune

Dune is another software developed at LSTS. It runs on a small computer (PC-104) with a Linux distribution on-board of the UAV. This computer has limited processing power and resources, hence Dune must be as much light-weight as possible.
Dune is connected to the autopilot through a serial connection and to Neptus through a Wi-Fi link. Depending on which autopilot is used, Dune uses a specific protocol to exchange information with it (waypoints, telemetry, system status, etc). Communication to the ground is made using the IMC

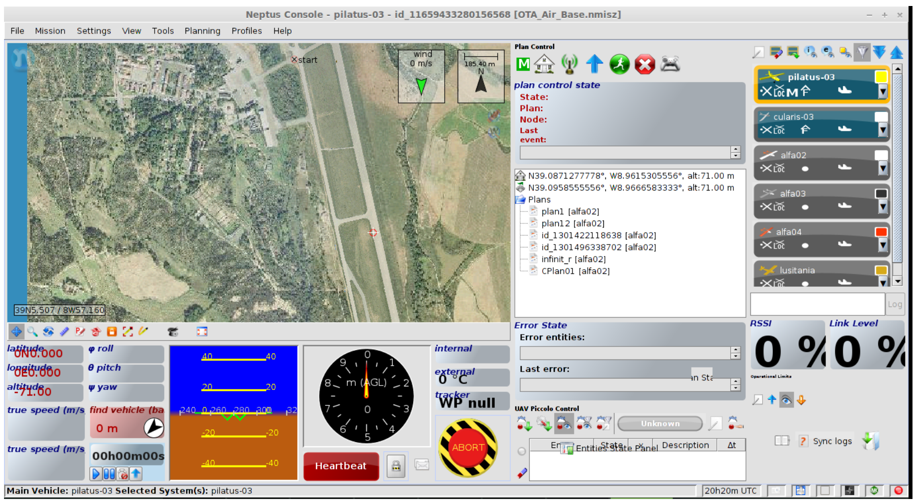


Figure 2.15: Neptus UAV console

protocol.
Besides being able to control navigation and report back the telemetry and vehicle status Dune can also manage payload, a vast selection of sensors like cameras (and gimbals), radar, laser altimeter is supported by Dune.
With reference to figure 3.2, Dune is in the middle Navigation loop but it also has some capabilities of the outer loop (Payload Management).

2.4.3 IMC

Inter-Module Communication (IMC) is a message-oriented protocol developed at LSTS [21]. It was designed for communication between heterogeneous vehicles (ground, sea and aerial), sensors and mission management interfaces (through which humans can interact with the system).
IMC's goal is to make communication transparent to all the nodes in the network.
Typical IMC messages include:

- Mission control
- Vehicle control
- Maneuver
- Guidance
- Navigation
- Sensor
- Actuator

Because of the modular architecture of the whole software tool-chain, it is possible to develop vehicles with very different capabilities and configurations which will all interact seamlessly.

Chapter 3

State of the Art

In this chapter the concepts in which the developed system was built upon will be presented. Concepts such as UAV teams and their configuration, mixed initiative interactions, persistent surveillance, control architecture, decision support systems and area decomposition methods.

3.1 UAV teams

For missions like surveillance, environmental and topological data collection, UAV teams offer interesting capabilities like efficiency and reliability [14]. Teams of multiple UAVs, can accomplish different and more complex objectives than those of a single vehicle. The possibilities offered by cooperative teams of UAVs are very appealing for search, localization (see [22]) and persistent surveillance applications.
These teams can have multiple configurations.

3.1.1 Homogeneous

In this type of team, which is the most common, all of the UAVs have the same capabilities and tools [23].
Homogeneous teams simplify planning and control because the same rules can be applied to all the vehicles, and replacing a vehicle (damaged, out of fuel, destroyed) is very easy in terms of readjusting the team members' individual objectives (if a new UAV is launched, all the others keep their initial tasks).

3.1.2 Heterogeneous

A more challenging situation occurs when the team has different types of UAVs ([24] and [12]) and they have different sizes, flight dynamics, operational limits and sensors.
Although these teams can potentially produce better results because each UAV can be equipped with dedicated sensors and is given a specific role, the planning system is required to take into

account the capabilities of each vehicle and use the best UAV suited for each mission or role on a mission. Determining which UAV or UAV team is the best for a given, might be a difficult task.

3.2 Mixed Initiative

Currently, UAV operations require about 5 humans per vehicle. However, there has been an effort to invert that ratio so that one operator can control several UAVs [25]. UAVs are complex systems, and it is difficult for an autonomous system to fully understand all the environmental variables that affect their performance. The concept of mixed initiative is to allow experienced human Operators/Pilots to interact with planning algorithms, by introducing restrictions which are not visible or obvious for the algorithm. These restrictions which may seem simple for a human, sometimes would require vast sensor networks and time costly processing for an automated system to deduce. Mixed initiative allows us to take advantage of what each agent (human and autonomous) can do best. By having the autonomous system negotiate with the human while planning and executing we can combine the skills, capacities and knowledge of each one to achieve better results any of them could by itself [26]. In [27], some limitations of purely Artificial Intelligence (AI) based systems in comparison with Mixed Initiative ones are discussed.
In [25] the importance of having a task specification standard is discussed. For mixed initiative systems where missions are assigned through complex and sophisticated interactions, it is important that the tasks are well defined. For that, a Task Specification Tree is used. The tasks are recursively assigned to multiple agents in order to satisfy all the constraints.
In [28] the Hydra interface is described in which automated agents and humans share the control over mission planning for the UAVs and try to find a target in a limited area. Every agent sends a request for each vehicle and each agent has a certain authority $\alpha_i \in [0,1]$, the system will calculate the most probable combination of requests for the team of UAVs that will lead to a higher probability of finding the target taking into account the authority of each agent when doing so.
[29] presents a scenario where mixed initiative is used to monitor an oil spillage from a tank ship. The human operators take part in the planning and control of the UAVs' mission and analyze data pre-processed and sent by the vehicles.

3.3 Persistent Surveillance

Persistent Surveillance - *A collection strategy that emphasizes the ability of some collection systems to linger on demand in an area to detect, locate, characterize, identify, track, target, and possibly provide battle damage assessment and retargeting in near or real-time. Persistent surveillance facilitates the prediction of an adversary's behaviour and the formulation and execution of preemptive activities to deter or forestall anticipated adversary courses of action.* in Joint Publication 2-0, 2007 [30]

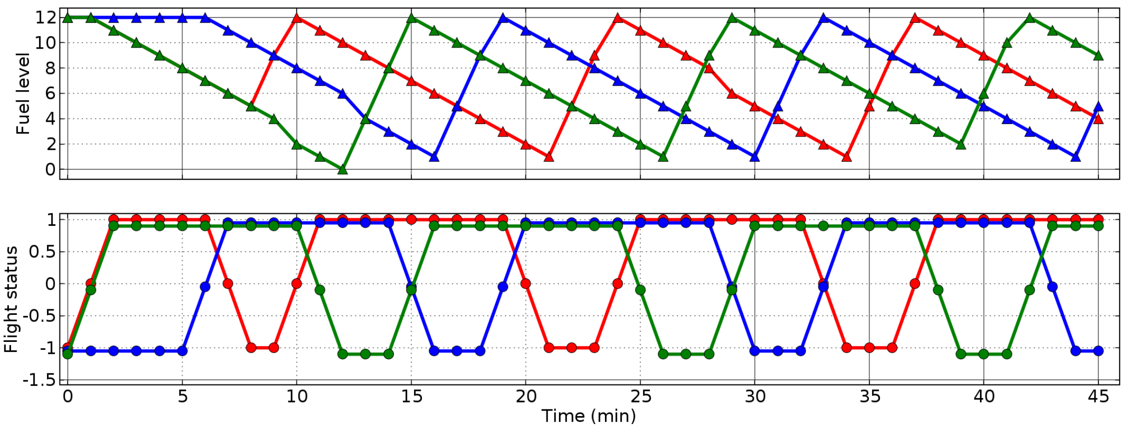


Figure 3.1: Persistent surveillance with health management ($r = 2, n = 3$) – from [8]

Persistent surveillance operations are inherently long endurance and require multiple UAVs to coordinate effectively to complete the mission.
UAVs can carry many sensors to monitor the system performance, like fuel consumption, battery level, motor/engine temperature. These health-aware vehicles can achieve a higher level of global performance for the mission. In [8] a group health management framework is described to aid in persistent surveillance missions.
For health management to be useful the multi-agent system should be:

- Proactive. That means it should be able to look into the future, predict events that are likely to occur and knowing that, it should take action to get the best possible state. In contrast to a reactive system which merely acts upon the knowledge it has of the present (can respond to a problem but cannot avoid it).
- Able to manage the health of the whole group and not only at individual level. Failure of a single UAV might require the reassignment of all the others.

The problem presented in the same paper is of persistent surveillance over an area which is a certain distance from the base. To fulfill the mission it is required that at all times there are r of the n total UAVs flying over the area ($r < n$). By using dynamic programming it is proved that the system can effectively manage a team of UAVs so that the fuel level of each vehicle never reaches zero (and the vehicle is lost). See Fig. 3.1 – flight status -1 is at base (refueling/wanting), 1 is surveying the area.

3.4 Control Architecture

In [9] we learn that there are many human factors that cannot be quantified like overconfidence, tiredness, attention allocation, which are difficult to take into consideration when building a mixed initiative interface. Results analyzed in the same article show that for an operator to control multiple UAVs, the Motion Control loop (see Fig. 3.2) has to be fully automated and reliable. In that

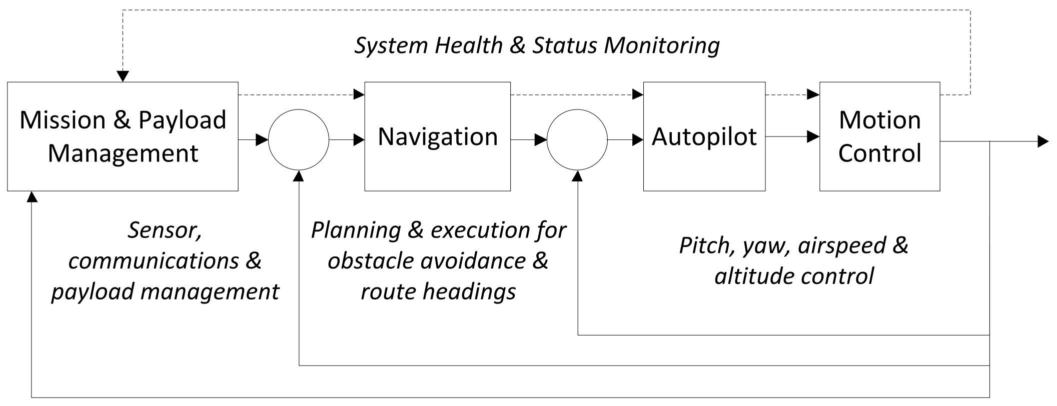

System Health & Status Monitoring
Mission & Payload Management
Navigation
Autopilot
Motion Control
Sensor, communications & payload management
Planning & execution for obstacle avoidance & route headings
Pitch, yaw, airspeed & altitude control

Figure 3.2: Hierarchical Control Loops for a Single UAV – adapted from [9]

situation, with a Navigation Control loop which requires human validation, the optimal number of UAVs per operator is 2-4. If the Navigation Control loop becomes fully automated, we can see a jump to a plateau of 8-12 UAVs per operator, who interacts only with the outer control loop, the Mission Management. These optimal values were obtained by extensive testing and observation by experts, cost of failed missions and cost in terms of mission delays introduced by inefficient human interactions.
In [31] it is possible to see that once again, the Motion Control has to be fully automated. The Navigation Control loop is defined as three controller processes: Waypoint Controller, Orbit Controller and Vision Controller. The Mission Management loop assigns tasks to UAVs, composed of chains of Navigation processes, and tries to fulfill the objectives defined by the human operator, asking for interaction when the available resources are not enough to complete the mission. [31] focuses in search and patrol, while [9] often mentions attacking the target, systems like these have a much higher need for operator confirmation before the UAVs engage in offensive maneuvers.
We can estimate that a higher number of observer UAVs can be under a single operator's responsibility when compared to offensive ones.

3.5 Decision Support Systems

A Decision Support System (DSS) is a system designed to help a human make a choice. It can do so by manipulating large amounts of data, models and other sources of information and present it in a way that is easily understood by humans. These systems can be found mainly in business and industrial sets as well as in critical applications like defense and emergency services. A Decision Support Visualisation (DSV) is an important part of a DSS because it can make the interaction much easier for the human operator. In [32] we can read about different types of DSVs and how well they perform. Some results were unexpected, and are explained by the human behavior factor which was hard to predict. Although the DSVs described in this article present the data to the

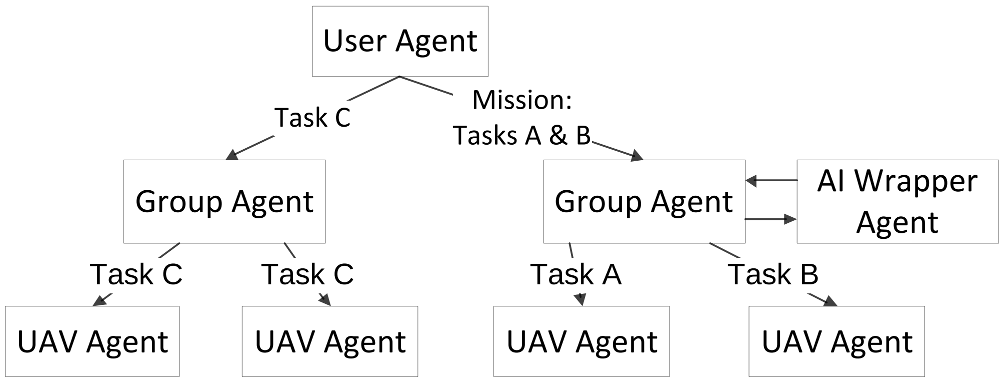


Figure 3.3: Agent Organization to complete a set of Tasks – adapted from [10]

operator in a way that it is easier to understand and that increases the operator's SA, the systems do not provide any recommendation about the course of action to take. That functionality would probably increase the performance greatly.
The systems described so far were implemented in a static operator architecture, but in [10] a different design is introduced. The UAV's operator is a pilot flying on his own aircraft. This requires the pilot to split his attention between managing the UAVs and flying a fast jet simultaneously. The UAVs are controlled by a set of collaborating agents, and the different AI planning techniques are wrapped inside specialist planning agents. Each UAV's movement is controlled by a UAV Agent that is coordinated by a Group Agent who in turn gets his tasks from the User Agent, the pilot (see Fig. 3.3). Another important contribution of [10] is the conclusion that sometimes a sub-optimal, but simpler system is a better choice for implementation because it allows for a easier interface with the pilot and faster algorithms.
Monterey Bay Aquarium Research Institute (MBARI) has developed a DSS, the ODSS (see [33]). It is used to help increase the SA and make mission planning easier for multiple vehicles and systems.

3.6 Area decomposition

When using teams of robots to perform tasks, it is usual to divide the total area into segments and assign them to the robots. By ensuring each robot stays inside its own area, inter-vehicle collision problems are avoided. Decomposing an area into smaller parts is a problem which can be solved by a number of different algorithms.
In [12] the area is divided according to the initial position of the UAVs and their relative capabilities. Relative capabilities are used as a measure of how big a fraction of the total area each UAV should get. Areas are defined by convex polygons without obstacles inside.

Chapter 4

The Problem

UAVs are suited for persistent surveillance applications, which require several UAVs and possibly other systems to coordinate. There is a need for a system/tool that allows a single Pilot to efficiently plan and control the execution of a mission for multiple UAVs. Such system should consider inputs from AI algorithms and human Pilots. The tools developed LSTS which were previously discussed can integrate such mixed initiative system.
Examples of operation scenarios for multiple UAVs systems are:

- Fire detection and tracking
- Coastline surveillance, of particular interest to Portugal considering the geography
- Target search
- Area patrolling

4.1 User Requirements

A Planning, Execution and Monitoring tool is to be developed that will enable a single Pilot to coordinate multiple UAVs. Having Fig. 3.2 in mind, we need to abstract from the inner control loops, which have already been fully automated, by Dune and the autopilot.
Mission Management, the outer loop, can include a first stage of planning before the UAVs are deployed.
With the current setup an operator is required to manually input all the waypoints for each UAV individually, which is not practical at all and does not guarantee that the resource allocation is optimal.
In order to solve this issue, we need a system that automatically generates plans for multiple UAVs simultaneously.
This system should have as inputs:

- Mission Type (Persistent Surveillance),

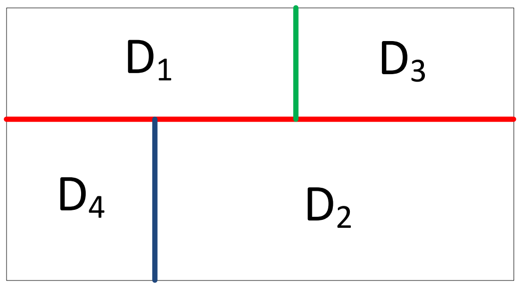


Figure 4.1: Area division

- Mission Parameters (Area, Altitude, Priorities),
- Available UAVs,
- UAV Status,
- UAV Type and Tools,

and generate a plan for each UAV.
Uncertainty is a measurement of the knowledge of each point in the environment, the higher the uncertainty is, the less we know about that location. Uncertainty grows with time, which means it is directly linked to the age of data captured from each point. Its value is also affected by the priority in that place.
The plan will be created in order to minimize the uncertainty of information captured (images for example) at each location, dividing the total area into n smaller areas, using $n-1$ divisions, where n in the number of available UAVs [14]. See Fig. 4.1.

When defining areas to be patrolled, some sections of those areas may be of more interest, the uncertainty level rises faster. See Fig. 4.2. Another area might be of high risk so it is preferable to send a smaller and harder to detect UAV. The system should be able to understand these restrictions given by the Pilot and propose a better plan.
Initially the system should divide the area in as many smaller areas as the number of available

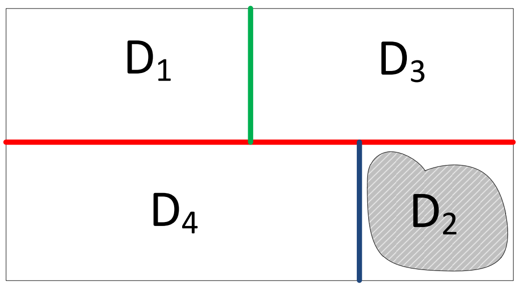


Figure 4.2: Area division with higher uncertainty section

UAVs. This division would take into consideration the previously mentioned restrictions.
Then, the path each UAV will follow inside its assigned area should be such that it will ensure effective data capture and full area coverage.

After a first plan is presented to the Pilot, some changes might be required:

- Change the proposed areas.
- Change the trajectory inside the area.
- Change the UAVs.

These changes can be made before the UAVs are deployed, or during the mission. These interactions of the Pilot are classified as mixed initiative.
When making changes during the mission execution, limits should be imposed to those changes. If Pilot were to change plans every few seconds it would lead to system instability.

4.2 Notation

4.2.1 Vehicles

Each UAV is defined by the following properties:

- v - speed (consider ground speed the same as air speed - no wind)
- aut - autonomy (time)
- FoV - sensor field of view, or angle of view (perpendicular to movement direction)
- R - sensor resolution (in the same direction as previous item)

4.2.2 Mission

A mission is defined by a set of restrictions:

- A - area to be surveyed, defined by a set of coordinates (vertexes of a polygon)
- P(x,y) - priority of each point in A
- Rt - Resolution on ground (length/pixel)

4.2.3 Terminology

Definition 1 (Reference Frame)**.** The frame of reference is an orthogonal set of axes. XX is aligned with the North-South direction (varying Latitude) pointing to North, YY is aligned with the East-West direction (varying Longitude) pointing to East. Taking into account the “right-hand rule”, the ZZ axis is pointing down, to the earth’s center.

Remark. For most of the process, only two dimensions are considered, XX and YY.

Areas can be defined as:

$$A_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{in} \\ y_{i1} & y_{i2} & \cdots & y_{in} \end{bmatrix}, \; i \in [0, N] \tag{4.1}$$

where

$$\begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix} \tag{4.2}$$

is the jth vertex of the polygon which defines the area, with coordinates x_{ij} and y_{ij}.

Definition 2 (Survey area)**.** A_0 is the area to be patrolled, and is assumed to be convex.

Definition 3 (Priority areas)**.** The other areas A_i, $i \in [1, N]$, are used to define zones with higher or lower priority. They are not required to be convex. Every area has an associated priority p_i:

$$\overline{A_i} = \{A_i, p_i\}, \; p_i \in \mathbb{R}_0^+ \tag{4.3}$$

$$p_0 = 1 \tag{4.4}$$

The default priority is 1, and the higher the priority value is, the more frequent the visits will be to that location. It is also assumed that:

$$\forall_{i,j}, \; i, j \in [1, N], \; i \neq j, \; AP_i \cap AP_j = \emptyset \tag{4.5}$$

which means that priority areas do not intersect with each other.

Definition 4 (Sweep Direction)**.** Let $y = mx + b$ be a straight line in $\mathbb{R}^2$ then

$$\theta = \arctan(m) \tag{4.6}$$

is the direction of the sweep when the vehicle flies that line over the area. See Figure 4.3.

Definition 5 (Lane)**.** A lane is a segment of the path of a vehicle. It consists of a start and end points and a width. It is defined as:

$$L_i = \{X_{li1}, X_{li2}, wd_i\} \tag{4.7}$$

$$X_{lij} = (x, y) \tag{4.8}$$

Lanes are numbered starting from the side of A_0 which is parallel to the sweeping direction.

Definition 6 (World)**.** The world is represented as a grid, with discrete coordinates (x, y).

Lemma 1. *Let $(x, y) \in \mathbb{R}^2$ then $(x, y) \in A_i$ if a ray (or half-line) starting at (x, y) crosses the polygon boundaries an odd number of times.*

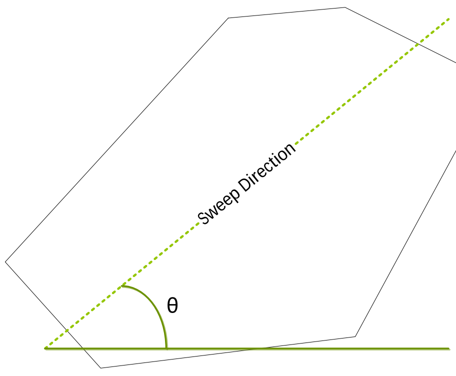


Figure 4.3: Sweep Direction – θ

Proof. This is proven by the Jordan Curve Theorem, see [34]. □

Definition 7 (Priority@(x,y))**.** The priority function gives the priority of a point in space. Let

$$P(x,y) : \mathbb{R}^2 \to \mathbb{R}_0^+ \tag{4.9}$$

Then P is the priority at $(x,y) \in A_0$:

$$P(x,y) = p_i \; if \; (x,y) \in A_i \tag{4.10}$$

Definition 8 (Natural Uncertainty)**.** Describes the evolution of the Uncertainty in the area, if no vehicle is observing. Let

$$U(x,y,t) : \mathbb{R}^2 \times \mathbb{N}_0^+ \to \mathbb{R}_0^+ \tag{4.11}$$

Then U describes the uncertainty at $(x,y) \in A_0$:

$$U(x,y,t) = U(x,y,t-1) + P(x,y)k \tag{4.12}$$

$$U(x,y,0) = 0, \; \forall_{x,y} \tag{4.13}$$

where k is a positive parameter that represents how fast uncertainty should grow.

There are M available UAVs:

$$UAVs = \{u_1, u_2, \dots, u_M\} \tag{4.14}$$

Definition 9 (Sensing Width)**.** Depending on the sensor's field-of-view and distance to the ground (altitude Above Ground Level (AGL)), the visible width from the vehicle is variable. The sensing width is a measure of that distance. Sensing Width is given by:

$$\begin{cases} sw_i = 2 \times AGL_i \times \tan(\frac{FoV_i}{2}) \\ sw_i = R_t \times R_i \end{cases} \tag{4.15}$$

We know from section 4.2 that both R_i and FoV_i are properties of each vehicle. By forcing either AGL_i or R_t, we get the value of the other and also sw_i.

Definition 10 (Visible Area)**.** This is the area a vehicle is observing at a given instant in time. It depends on the position of the UAV and its sensing width. We consider the visible area to be a circle with diameter sw_i, centered in the position of u_i. The visible area is then:

$$(x,y) \in V_i(t)\ if\ (x - x_{ui}(t))^2 + (y - y_{ui}(t))^2 \leq \left(\frac{sw_i}{2}\right)^2 \tag{4.16}$$

where $(x_{ui}(t), y_{ui}(t))$ is the position of vehicle u_i at instant t.

Definition 11 (Uncertainty under UAV observation)**.** We are now able to define the evolution of Uncertainty under UAV observation.

$$U(x,y,t) = \begin{cases} U(x,y,t-1) + P(x,y)k,\ if(x,y) \notin V_i(t),\ i = 1...M \\ 0,\ if(x,y) \in V_i(t),\ i = 1...M \end{cases} \tag{4.17}$$

Definition 12 (Lane Uncertainty)**.** To define the uncertainty of a lane, we first need to specify the points which are in a lane:

$$(x,y) \in L_i\ if\ distance\ from\ (x,y)\ to\ \overline{X_{li1}X_{li2}}\ is\ \leq \frac{wd_i}{2} \tag{4.18}$$

so lane uncertainty U_l is given by:

$$U_{li}(t) = \max_{(x,y)\in L_i} \{U(x,y,t)\} \tag{4.19}$$

Definition 13 (Operator Parameters)**.** The operator is required to set one of these parameters:

1. AGL_i - altitude Above Ground Level
2. R_t - resolution on ground

This allows the system to know sw_i.

4.3 Design Constraints

The system to be developed has to fit in with the existing setup, in Fig. 4.4 it is possible to see a simple System Breakdown Structure (SBS), only with the top level.
This system can be represented as in Fig. 4.5, and it can fit with the rest of the setup like in Fig. 4.6

4.4 Use Case

Ria Formosa, located in Algarve, is a system of barrier islands between the sea and the mainland (see Fig. 4.7). Due to its complex water canal system and many small islands it is an area often

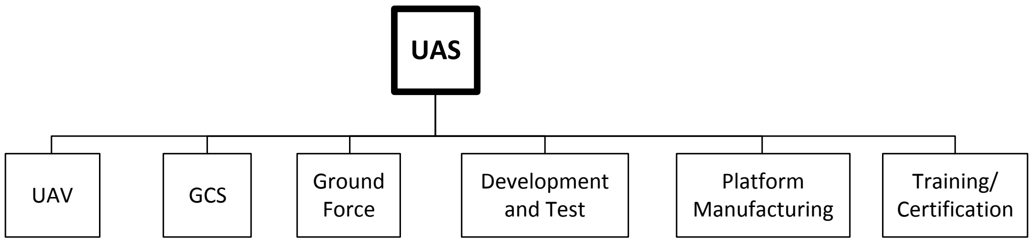


Figure 4.4: Simplified System Breakdown Structure

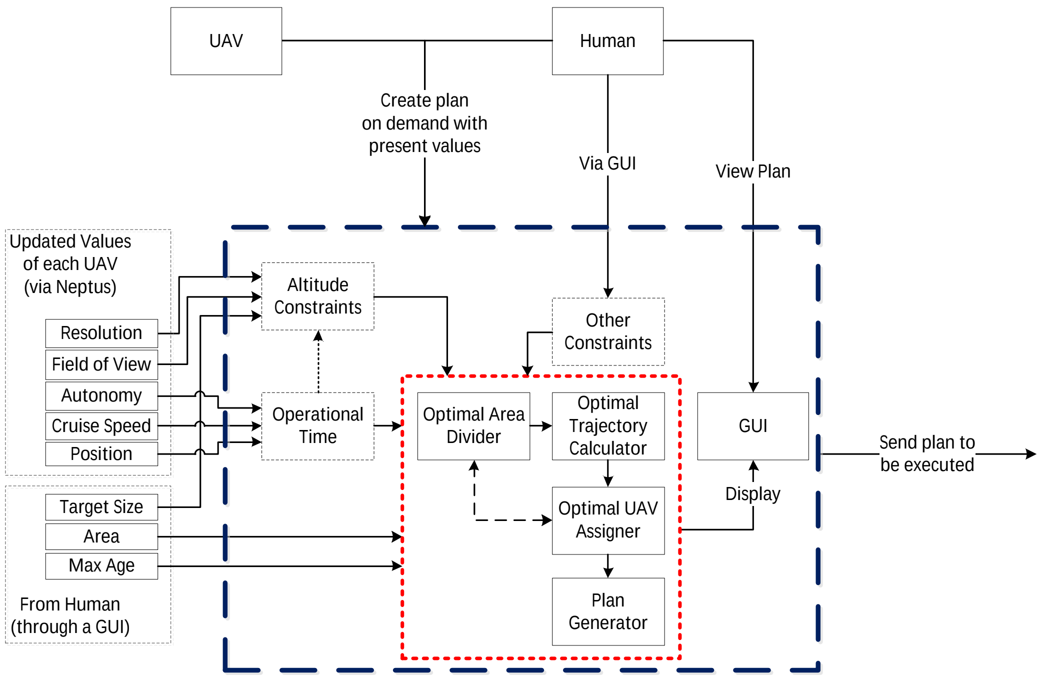


Figure 4.5: System Functional Decomposition

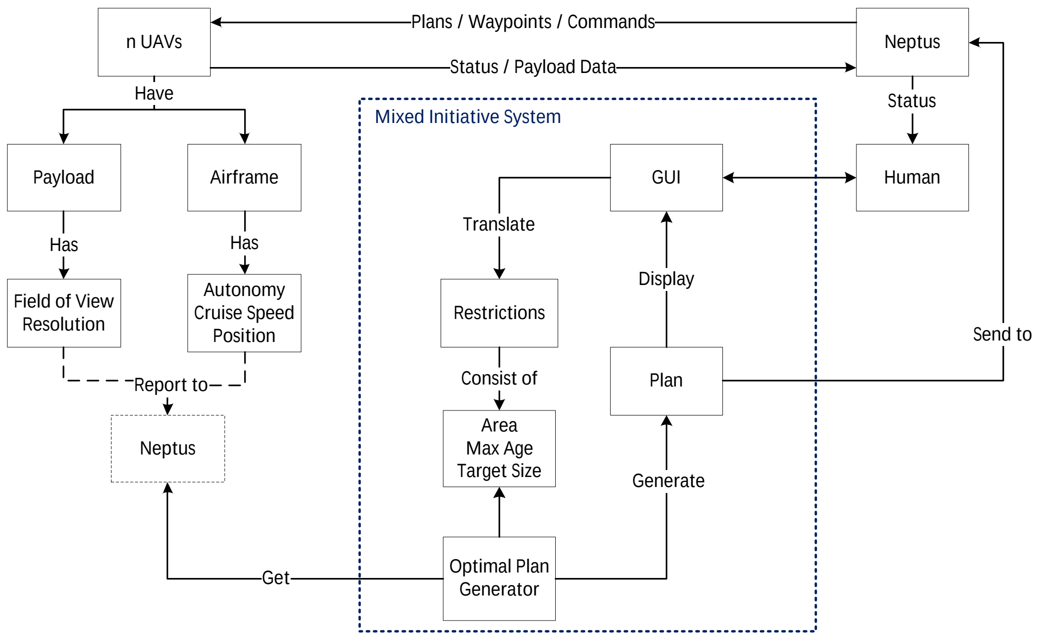


Figure 4.6: System Architecture

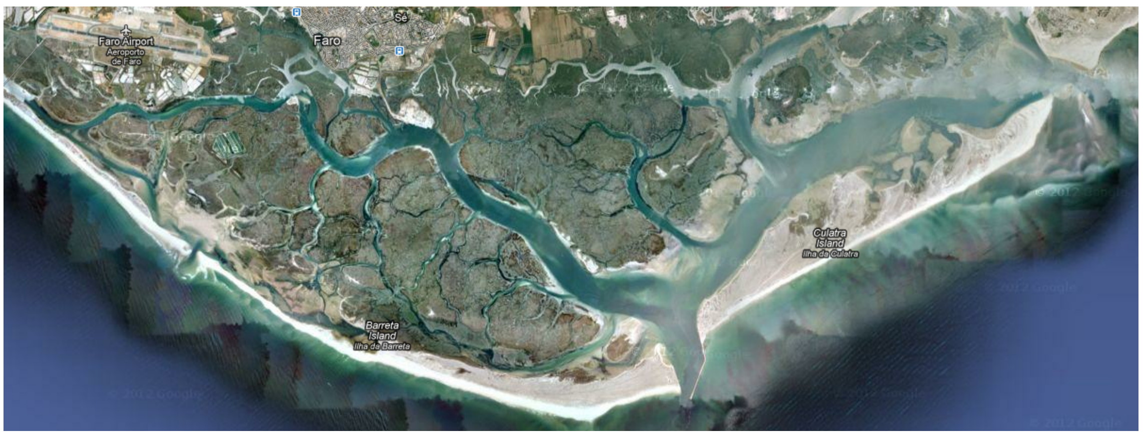


Figure 4.7: Ria Formosa

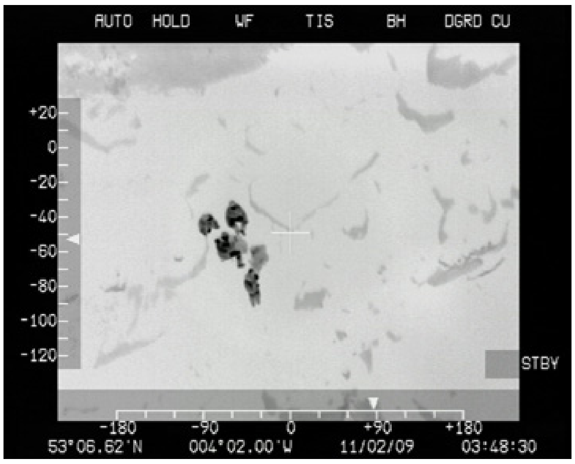

AUTO HOLD WF TIS BH DGRD CU
+20
0
-20
-40
-60
-80
-100
-120
STBY
-180 -90 0 +90 +180
53°06.62'N 004°02.00'W 11/02/09 03:48:30

Figure 4.8: Aerial Night Vision Image – from [11]

used for drug trafficking [35]. Over the last years many operations were performed to successfully catch the criminals, however the activities have not stopped.
Consider a scenario in which the Maritime Police is suspicious of a major deal happening during the night. Because of the numerous islands and canals it is impossible to place a boat in every one of them. Also, if the criminals suspect that the police is present they will not show up and the trail will be lost.
The Maritime Police contacts the "Força Aérea Portuguesa" – Portuguese Air Force and asks for help with the operation, the UAVs team is assigned to assist them.
Four Alfas (see Fig. 2.9) equipped with night vision cameras are deployed from the nearby Faro Airport. Due to their low noise and small size they are undetectable by the suspects.
The pilots set the interest area with an easy to use Graphical User Interface (GUI) and select high probability locations of where the deal is happening. Then, the UAVs start patrolling the area, taking particular care with the high interest sections, doing more frequent passages above those areas.
Suddenly, a few hours into the night one of the UAVs locates some people on a small island (see Fig. 4.8 – picture is unrelated to actual drug deal), the alert is given and the Maritime Police goes to the location using one of its fast boats and apprehends the suspects and merchandise.
The UAVs safely return to the airport after the successful mission.
Mission parameters were:

- Area: 100 Km^2
- UAVs: 4
- Speed: 22 m/s
- Resolution on ground: 0.5 m/pixel
- Sensor Footprint Width: 360 m

4.5 Formulation

Problem 1. *Area*
Given:

- $A_i,\ i = 0, ..., N$
- $u_i,\ i = 0, ..., M$
- *Operator parameters (mixed initiative)*

Find: *divisions of the area and allocate vehicles to the partitions.*

Problem 2. *Lanes*
Given:

- *Partitions*
- $u_i,\ i = 0, ..., M$
- *Operator parameters (mixed initiative)*

Find: *lanes for each vehicle*

Problem 3. *Path planner*
Given:

- *Lanes*
- $U(x, y, t)$
- *Operator parameters (mixed initiative)*

Find: *next lane to follow for each vehicle*

Chapter 5

Approach

In the previous chapter, the problem was defined and three main sub-problems emerged. We devised algorithms to address these problems:

1. dividing the area,
2. generating a path inside the divided areas,
3. how to follow that path.

The diameter function, which is presented in 5.2.2 is useful for both area division and trajectory planning.
After a reasonable solution was found to the area division problem, we needed to study different search patterns and compare their usefulness to the problem. Choosing a search pattern is intricately linked to the possible control over path following.

5.1 Overview

Figures 5.1 and 5.2 show the functional architecture of the developed system, the composing blocks will now be explained in detail. The main system components are inside the red box in figure 5.1, they solve the problems that were just mentioned:

1. dividing the area,
2. generating a path inside the divided areas,
3. how to follow that path.

and they are implemented as:

- Area definer and partitioner
- Path planner
- Uncertainty tracker

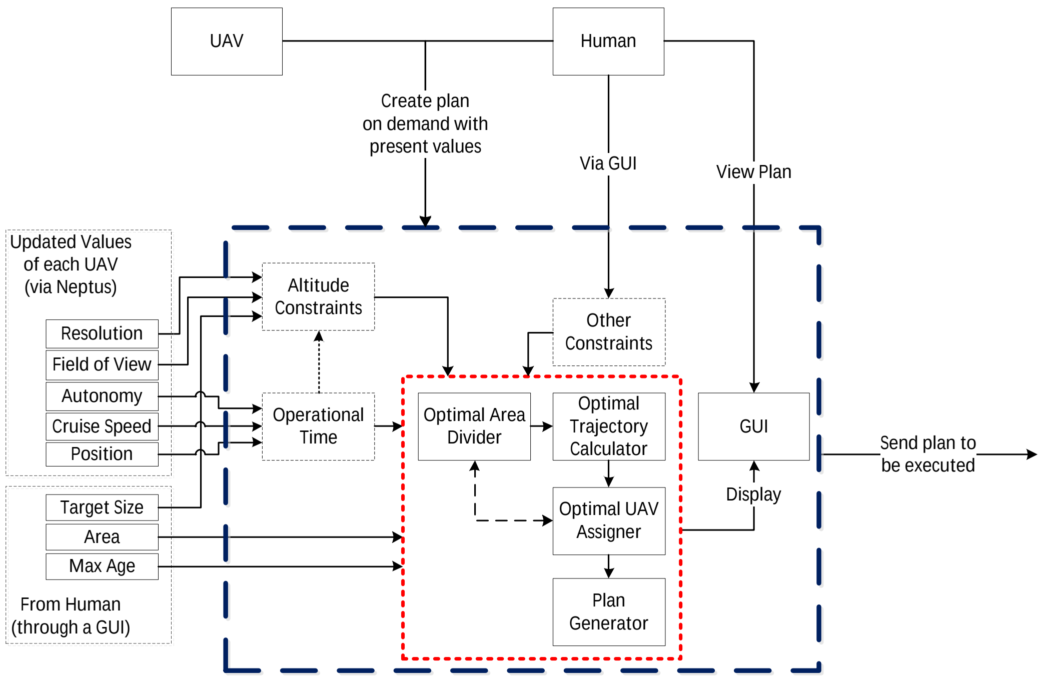


Figure 5.1: System Functional Decomposition

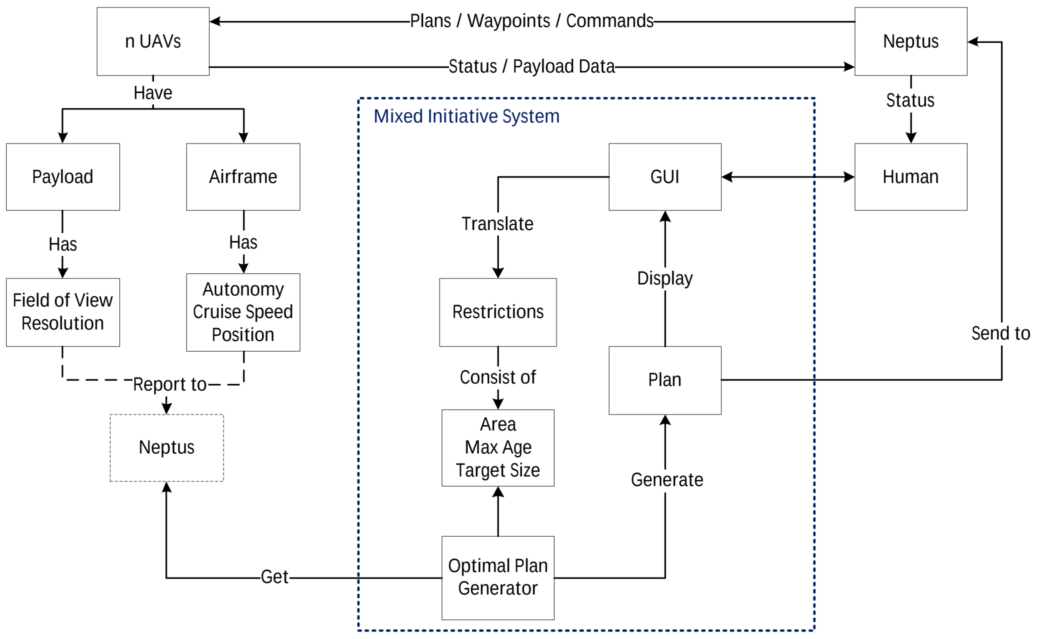


Figure 5.2: System Architecture

- Path controller

- User interface

5.2 Background

5.2.1 Terminology

Definition 14 (Rotation Matrix)**.** It is of use to define the rotation matrix:

$$rot(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{5.1}$$

In the following equation,

$$A_{0r}(\alpha) = rot(\alpha)A_0 \tag{5.2}$$

$A_{0r}(\alpha)$ is a polygon with the same dimensions as A_0 but it is rotated by an angle α about the point $(0,0)$.

5.2.2 Diameter Function

The diameter function $d(\theta)$ describes the altitude of the polygon perpendicular to the sweep direction [12].
To simplify notation we can say that:

$$A_{0r}(\alpha) = \begin{bmatrix} A_{0rx}(\alpha) \\ A_{0ry}(\alpha) \end{bmatrix} \tag{5.3}$$

so $A_{0ry}(\alpha)$ contains all the y coordinates of $A_{0r}(\alpha)$. The y axis is perpendicular to the angle $\beta = 0$, so if the sweeping direction is $\beta = 0$ then the altitude of the polygon is the amplitude of the values in y. Sweeping A along a direction θ is the same as surveying $A_r(-\theta)$ with a direction $\beta = 0$ (i.e. we either rotate the polygon or the sweep direction).
For a given angle θ, the diameter of the polygon is determined by rotating it by an angle of $-\theta$, then the height difference is measured between the highest and lowest points. That means

$$d(\theta) = \max_{i\in[1,n]}(A_{0ryi}(-\theta)) - \min_{j\in[1,n]}(A_{0ryj}(-\theta)) \tag{5.4}$$

See Figure 5.3. The shape of the diameter function can be understood by imagining the polygon rolling on a surface. As we can see in Figure 5.4 the highest point is always a vertex, so for a n-sided polygon, the diameter function can be defined as in Equation 5.5.

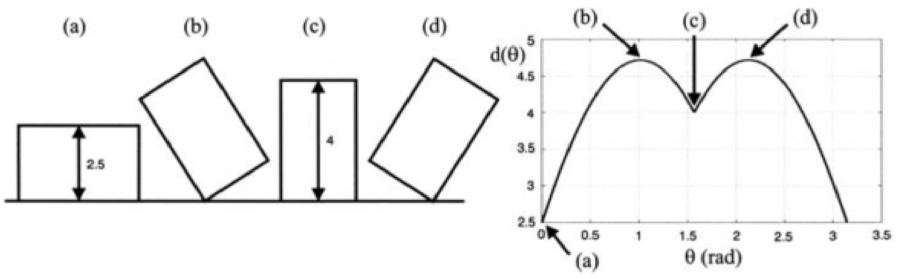


Figure 5.3: Example diameter function – from [12]

$$d(\theta) = \begin{cases} k_1 sin(\theta + \phi_1) & \theta \in [0, \theta_1[\\ k_2 sin(\theta + \phi_2) & \theta \in [\theta_1, \theta_2[\\ \cdots & \\ k_n sin(\theta + \phi_n) & \theta \in [\theta_{n-1}, \pi[\end{cases} \tag{5.5}$$

where k_n is the length of the chord from the pivot vertex (the one touching the flat surface) to another vertex, and ϕ_n is the orientation of chord n before the polygon is rotated. The function is piece-wise sinusoidal, the breakpoints happen when one of the polygon's sides is horizontal. In these cases, the long rows of the lawnmower pattern are parallel to the edge. The diameter function draws from the sine curve only in the interval $\theta \in (0, \pi)$, so $d''(\theta)$ is always negative except for the breakpoints. To minimize $d(\theta)$ means minimizing the number of rows, which in turn means minimizing the number of turns.

$$\theta_{opt} = \underset{\alpha \in [0,\pi[}{\arg\min} \{d(\alpha)\} \tag{5.6}$$

By minimizing the number of turns we reduce the amount of time in which data capture is useless (discussed in section 5.3.5). Using $d''(\theta)$ it is also possible to find the orientation of all the sides of the area polygon.

5.3 Algorithms

5.3.1 Sweep direction

Using the diameter function on the Area (A) polygon, we can get the optimal sweep direction, which is .

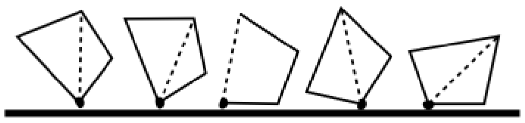


Figure 5.4: Creating the diameter function by rolling the polygon – from [13]

Theorem 2. *Optimal sweep direction holds less lanes*

Proof. Lanes are defined by a direction an a width and the lane direction is the same as the sweep direction. So if the width of the lane is perpendicular to its direction, like the diameter is the altitude of the polygon perpendicular to the sweep direction:

$$L_i, i \in [1, l] \tag{5.7}$$

$$l = \lceil \frac{d(\theta)}{wd_i} \rceil \tag{5.8}$$

$$l_{min} = \min_{\theta \in [0,\pi[} \left\{ \lceil \frac{d(\theta)}{wd_i} \rceil \right\} \tag{5.9}$$

$$\theta_{opt} = \arg\min_{\alpha \in [0,\pi[} \{d(\alpha)\} \tag{5.10}$$

$$\theta_{opt} \Rightarrow l_{min} \tag{5.11}$$

□

Remark. Less lanes do not minimize path length. However, it minimizes the number of turns, hence minimizing the amount of time during which the camera is capturing useless data.

Definition 15 (Optimal sweep direction)**.** The direction that requires the less lanes to cover the entire area

However, the human Pilot is given a chance to use another direction if he thinks it is best. This might be useful in situations where wind is blowing in a known direction and it might be advantageous to avoid a certain flying direction.

5.3.2 Area partitioning

In case the team has more than 1 UAV, it is necessary to divide the area, first we calculate how much of the total area should be assigned to each UAV:

$$u_i, i = 1, 2, ..., M \tag{5.12}$$

$$A_{ui} = A_0 \frac{C_i}{\sum_{j=1}^{n} C_j} \tag{5.13}$$

Maximum area of each UAV is calculated from its relative capabilities (C).

Definition 16 (Relative Capabilities)**.** Relative capabilities tell us how do the vehicles compare in terms of potentially visited area per time interval:

$$C_i = sw_i * v_i \tag{5.14}$$

Then, the total area is divided along the sweep direction so that the area of each part is the area calculated before. This can be done by sweeping a line along the polygon with the direction

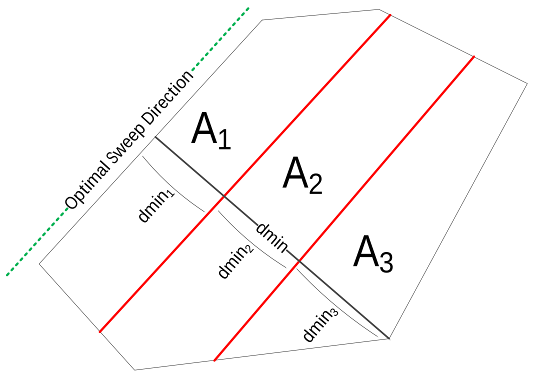


Figure 5.5: Area Partitioning

defined by the diameter function, or another one chosen by the pilot. This line slices the polygon each time the area on one of the sides of the line is the same as any A_i that has not been cut yet. See figure 5.5.
The slice direction is the same as the sweep direction for the following reasons:

1. If the vehicles needs to take a turn some distance outside of their assigned area, with this condition the vehicle will not enter the other UAVs' areas, avoiding the collision risk.

2. If the chosen direction is the one that yields less turns, then the sum of the minimum diameters of the area sections will be minimum (and equal to the minimum diameter of the total area)
$$\sum_{i=1}^{n} dmin_i = dmin \tag{5.15}$$

3. If one of the vehicles has a performance level different from expected it is easy to grow or shrink its area by simply moving the dividing line to one of the sides.

5.3.3 UAV path planning

5.3.3.1 Distance between sweeps

Ideally this should be the same as the sensing width, however due to possible (and mostly certain) errors in positioning and slight deviations in trajectory following (because of wind for example), we should set an overlap of captured data. If the collected data are images which can be used for surveillance or mapping, a small overlap is necessary to correctly stitch and blend them to form a seamless mosaic.

5.3.3.2 Trajectory generation

Using the sweep direction and distance between sweeps, generate a search pattern (zigzag for example – see figure 5.6). Several search patterns were explored and results are discussed in section 5.3.5.

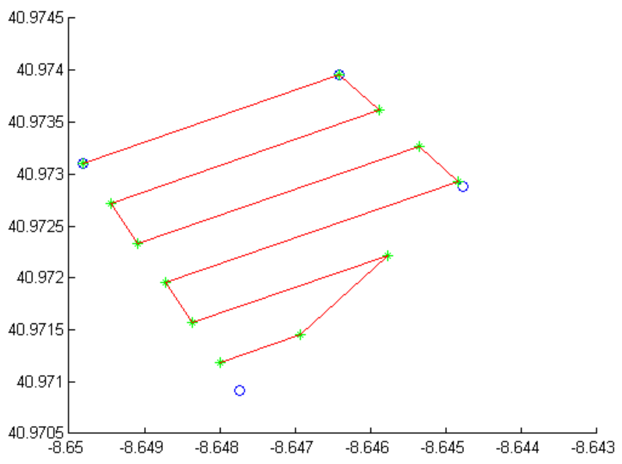


Figure 5.6: Zigzag trajectory along optimal sweep direction

5.3.4 Limitations

This method only works for convex areas, without obstacles inside the defined area.

5.3.5 Search Patterns

The problem has been defined as persistent surveillance with multiple UAVs over an area which might have different priorities (refer to Chapter 4). Different priorities means that for each sweep of the entire area, some places will be visited more than once. It has also been mentioned that the team of UAVs can be heterogeneous, which means we can have Geo-stabilized gimbal cameras (best case scenario) but also fixed cameras (most common scenario in small/micro UAV), this constrains the search pattern to be used. At this point of the system execution, the area for each UAV has already been simplified to a convex polygon.
To sum up, a good search pattern should:

- Allow different number of visits for different locations in the area.
- Have a minimum amount of turns, which are more costly for the aircraft to perform than straight lines and also affect greatly the quality of data gathered by fixed cameras.

Having in mind the previous requirements we can now analyze some common search patterns.

5.3.5.1 Spiral

This pattern is very common in single sweep searches when the whole area needs to be completely covered in the minimum amount of time. The UAV holds a small bank rate, allowing the camera to not deviate too much from the vertical. It is not suitable for repeated sweeps (persistent surveillance) and not at all flexible to allow areas with different priorities.

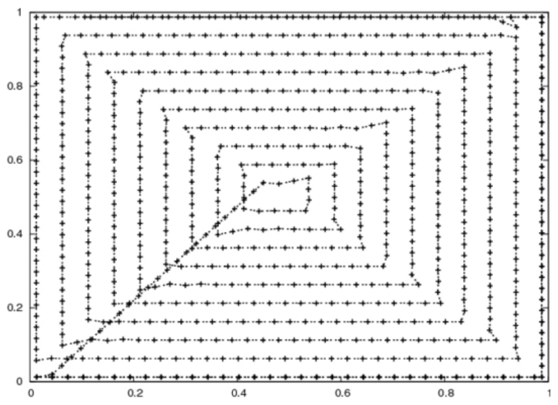


Figure 5.7: Spiral-like pattern – from [14]

5.3.5.2 Spiral-like

As visible in Fig. 5.7, this pattern resembles a spiral, but has straight sides and right angles (not flyable as is). The flyable path is a combination of leveled flight and maximum bank turns. During these turns, fixed cameras will not be able to capture any data of interest, so to fully cover the area, the UAV will need to fly the trajectory marked in green in Fig. 5.8, after it has flown the spiral-like path. Regarding the possibility of different priorities, the flight path is also very restricting, as the previous one.

5.3.5.3 Lawnmower

As the name hints, this pattern is similar to that of lawnmowers: long lanes with 180° turns at the ends (see Fig. 5.9).
This pattern's efficiency greatly depends on the amount of turns it has to do, so it is important to find the optimal sweep direction. See Fig. 5.10.
To determine the optimal sweep direction (with less turns) we use the diameter function. This pattern is suitable for persistent surveillance, however, in its simplest form it requires the UAV to

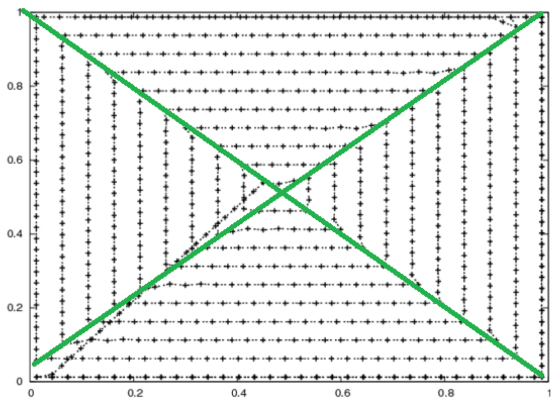


Figure 5.8: Additional path to fully cover the area

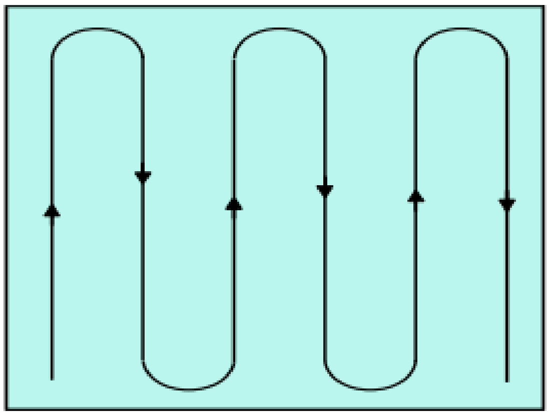


Figure 5.9: Lawnmower pattern – from [15]

turn outside the area of interest in order to fully cover the area, and it does not allow for multiple visits to specific regions.

5.3.5.4 Zamboni

The Zamboni pattern gets its name from the machines used to resurface the ice in hockey arenas, they have a large turning radius and hockey arenas are barriers at their edges, so the traditional lawnmower pattern was not suitable to fully cover the arena. This pattern is shown in Fig. 5.11. The machine starts in the black path, then changes to the blue, then red and finally the green one. This small change allows the area where the machine turns to be covered when it passes over it again, this time not turning, and removes the need for difficult maneuvers to make the sharp turn. In the case of a Zamboni, that would mean reversing and turning, but a UAV cannot fly backwards, so it would have to make a larger trajectory, firstly defined by Dubins for forward-only cars. This pattern improves the area coverage over the previous one, but still does not improve the different priorities aspect.

5.3.5.5 Dubins Path

Given two oriented points (x_1, y_1, θ_1) and (x_2, y_2, θ_2), the Dubins curve is the shortest path between them when the motion is constrained by a minimum radius [36]. The focus of this report is not to explain in detail the Dubins curve, it is enough to present some results. Fig. 5.12a shows the

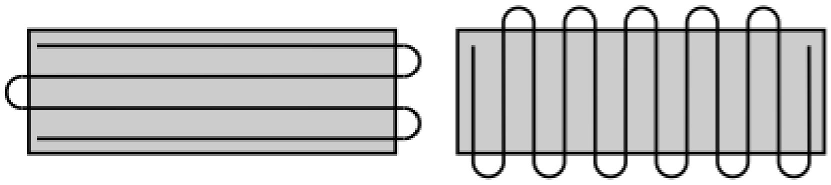


Figure 5.10: The area is the same, but the number of turns varies with the sweep direction – from [13]

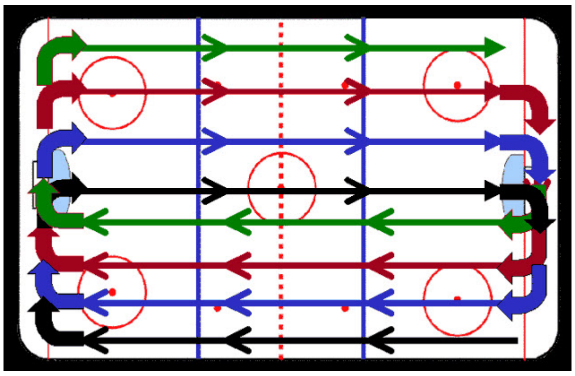


Figure 5.11: Zamboni pattern – from [16]

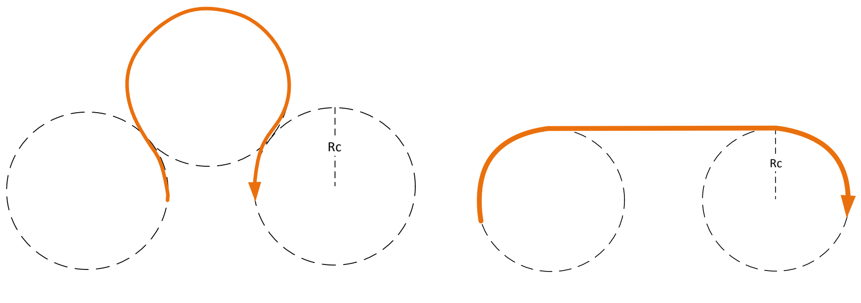
minimum path when the distance between consecutive rows is smaller than the minimum turn diameter. This path is all made of curves with maximum bank angle, which means nothing is captured by the camera. If instead of consecutive rows the UAV leaves at least one row between them, it can fly a path like in Fig. 5.12b. This trajectory has a straight segment which overlaps on the location of previous/future turns, so the entire area is covered.
Minimum turn diameter is given by the equation:

$$d_{turn} = 2 * \frac{v^2}{g * \tan(\theta_{maxbank})} \tag{5.16}$$

As an example the minimum turn diameters for some aircraft are presented in Table 5.1

5.3.5.6 Modified Lawnmower/Zamboni

All of the previously discussed patterns are unfit to deal with zones of different priorities. However, with a slight modification, the Lawnmower/Zamboni patterns can overcome that limitation. Usually, after the UAV has finished a row, it proceeds to the next row that has not been visited, but we can tell it to revisit a previous row, which has a higher priority.



(a) Tight Dubins Curve (b) Long Dubins Curve

Figure 5.12: Dubins Curves

Vehicle	airspeed (m/s)	$\theta_{maxbank}$ (deg)	Turn diameter (m)
Cularis	11	25	53
Pilatus	18	25	142
Alfa	18	30	115

Table 5.1: Minimum turn diameter

To implement this search pattern we will need to use the previously defined functions and variables:

1. Grid map of the area
2. Uncertainty: $U(x,y,t)$
3. Uncertainty of lane: $U_{li}(t)$
4. To determine which row to visit next, a weight function is necessary: $W_i(t)$
5. Then, to decide which row to visit next:

$$target_{row}(t) = \arg\max_{j\in[1,l]}\{W_j(t)\} \tag{5.17}$$

The modified Zamboni/Lawnmower pattern was implemented in a Matlab simulation.
Four different weight functions were tested:

$$W_i(t) = U_{li}(t) \tag{5.18a}$$

$$W_i(t) = U_{li}(t) - \frac{1}{v_j}d_{ij} \tag{5.18b}$$

$$W_i(t) = \begin{cases} U_{li}(t) - \infty & d_{ij} < d_{min} \\ U_{li}(t) - \frac{1}{v_j}d_{ij} & otherwise \end{cases} \tag{5.18c}$$

$$W_i(t) = U_{li}(t) - \frac{1}{v_j}\max\{d_{ij}, a + \frac{b-a}{b}d_{ij}\} \tag{5.18d}$$

where d_{ij} is the distance between the UAV u_j position and the closest end, X_{li1} or X_{li2}, of lane L_i, d_{min} is a minimum distance between lanes to visit, a and b are variables which define a function like in Fig. 5.13. The simulation parameters are as follows:

1. Area: 15 rows of 40 cells length (1 cell wide) each
2. The UAV can visit 2 cells per time step
3. Starting position is random for each run
4. 50 tries for each weight function

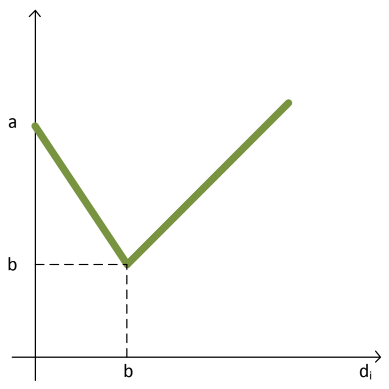


Figure 5.13: d_{ji} function (penalizes the choice of lanes that are too close)

5. UAV can view the cell even if it is turning (to simplify the simulation, but it will be changed on a next one)

6. $d_{min} = 2$, $a = 500$, $b = 2$

The following plots show the results of the simulations (maximum age over 50 tries, each with 1000 time steps), the weight functions in each are: blue (5.18a), red (5.18b), green (5.18c) and black (5.18d):

5.3.5.7 Remarks

It is possible to see that for the more interesting scenarios (Figs. 5.15 and 5.16), (5.18b) and (5.18d) perform better, so these were the weight functions considered for the final implementation, namely (5.18d).
The main difference between the two weight functions is that 5.18d penalizes the rows that are too close, this led to a lower performance in this basic simulations, however they did not take into account the vehicle dynamics which make it hard to perform two very close lanes sequentially.

5.3.5.8 Creating the lanes

We have discussed how to follow a path composed by lanes, but not where the lanes are located.
To generate lanes the Algorithm 1 is used. Because lanes are created inside each area and do not take into consideration the other areas, the first and last lane in each area might overlap with the neighboring area. This overlap is in its maximum, equal to half the width of the lane to each side. So we can have a maximum loss of performance which corresponds to surveying 1 lane per area which is useless.
This could be corrected by combining the area division and lane creating algorithms into a single algorithm which would take into account both steps.

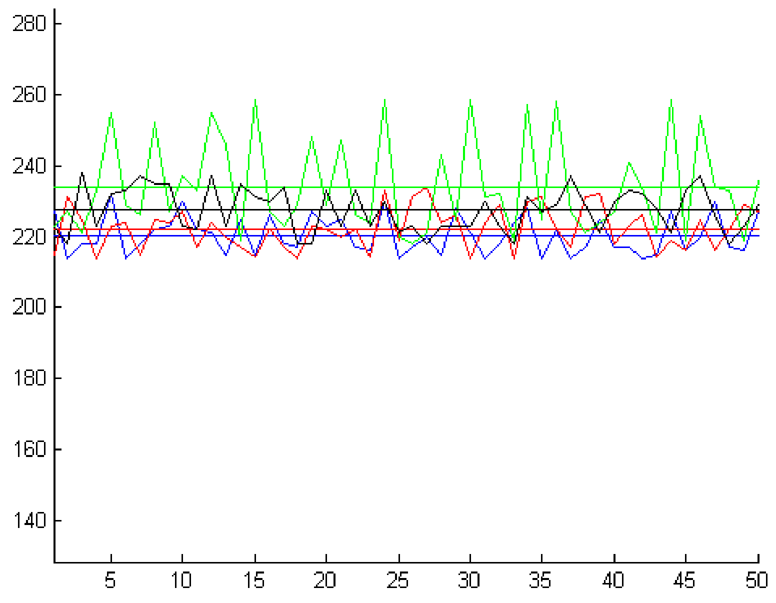


Figure 5.14: Whole area has same low priority (1)

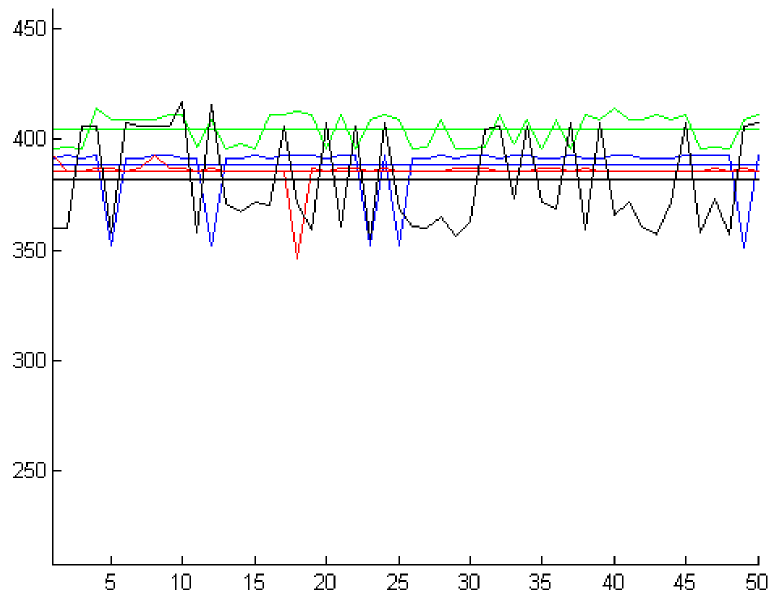


Figure 5.15: 2 cells have higher priority (2 and 3)

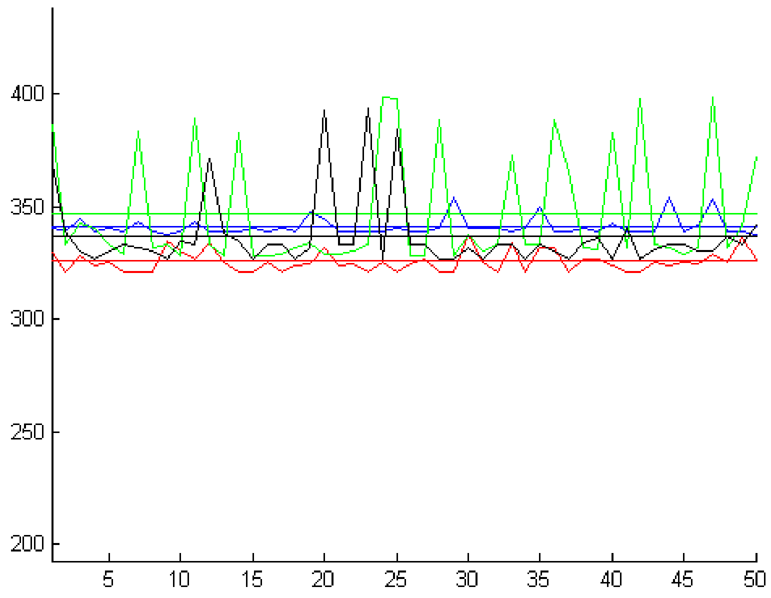


Figure 5.16: Priority is random in each cell, but constant (1 to 1.5)

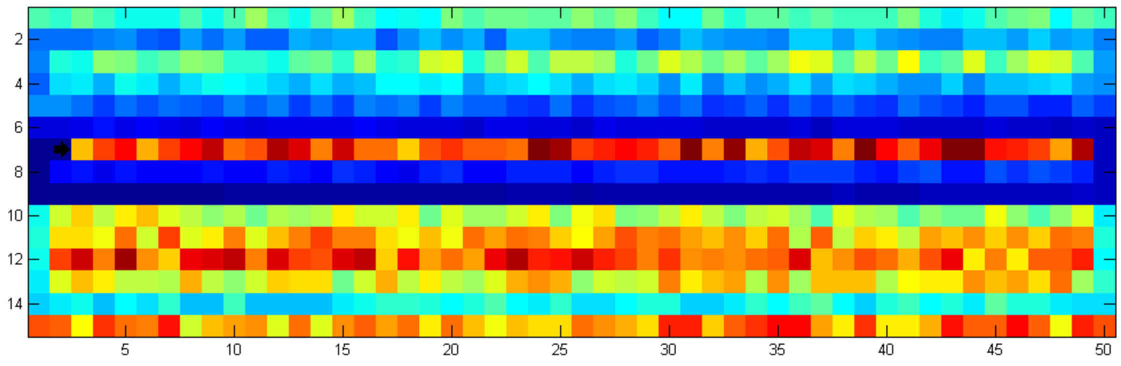


Figure 5.17: Area uncertainty at a certain point in time (darker red means higher uncertainty, darker blue means lower)

```
Algorithm 1 Create lanes
Require:
        $u_i, i = 1,2,...,M$        (5.19)
        $A_{ui}, i = 1,2,...,M$     (5.20)
        $sw_i, i = 1,2,...,M$       (5.21)
        $\theta_{opt}$              (5.22)
Ensure: $L_j, j = 1,...,l$
  for all $i$ such that $1 \leq i \leq M$ do
    rotate $A_i$ by $-\theta_{opt}$:
        $A_{ir} = rot(-\theta_{opt})A_i$        (5.23)
    get the diameter of $A_i$ in $\theta_{opt}$:
        $d_i(\theta_{opt})$        (5.24)
    determine how many lanes should be created for $A_i$:
        $nl_i = \left\lceil \frac{d_i(\theta_{opt})}{sw_i} \right\rceil$        (5.25)
    calculate offset of first lane relative to the $A_i$ polygon side:
        $os_i = \frac{d_i(\theta_{opt}) - nl_i * sw_i}{2}$        (5.26)
    because $A_{ir}$ is rotated according to the optimal angle, the lanes are horizontal line segments
    which intersect the $A_i$ polygon, they are vertically spaced by $sw_i$. The first and last lanes are
    at a distance $os_i$ from the boundary of $A_i$
    the width of the lanes is $sw_i$
    the lanes of vehicle $u_i$ will be:
        $L_{uik},\ k = 1,...,nl_i$        (5.27)
  end for
  rotate all the lanes to compensate the initial rotation of $A_i$:
        $L_j = rot(\theta_{opt}) * L_{uik}$        (5.28)
  where rotating a lane is rotating its start and end points
```

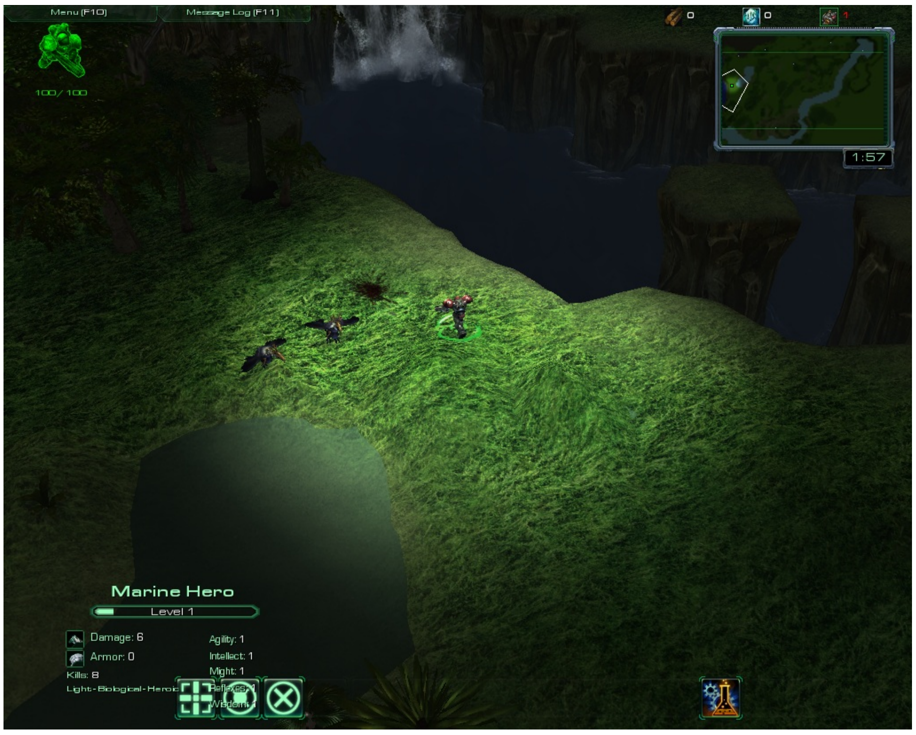


Figure 5.18: Fog-of-War graphical effect in a computer game – from [17]

5.4 User Interface

To make mixed initiative efficient it is necessary to have a User Interface which allows the Pilot to:

- quickly understand the current situation
- intuitively input the mission constraints
- easily input his own knowledge to help the algorithm

5.4.1 Fog-of-War

Fog-of-War is a term used in military which means the level of uncertainty in SA of an area.
It is also a graphical effect used by many strategy games which translates the military concept [37]. This effect is visible in Fig. 5.18, the darker areas have a higher uncertainty. This effect is extremely straightforward to understand and does not overload the interface with extra plots to let the Pilot know the current uncertainty.
Fig. 5.19 shows the effect implemented in Matlab for the multi-UAV console. The higher the uncertainty is of a point in the map (last visited a long time ago) the darker it gets. The UAV brightens the map area around itself when it flies over it.

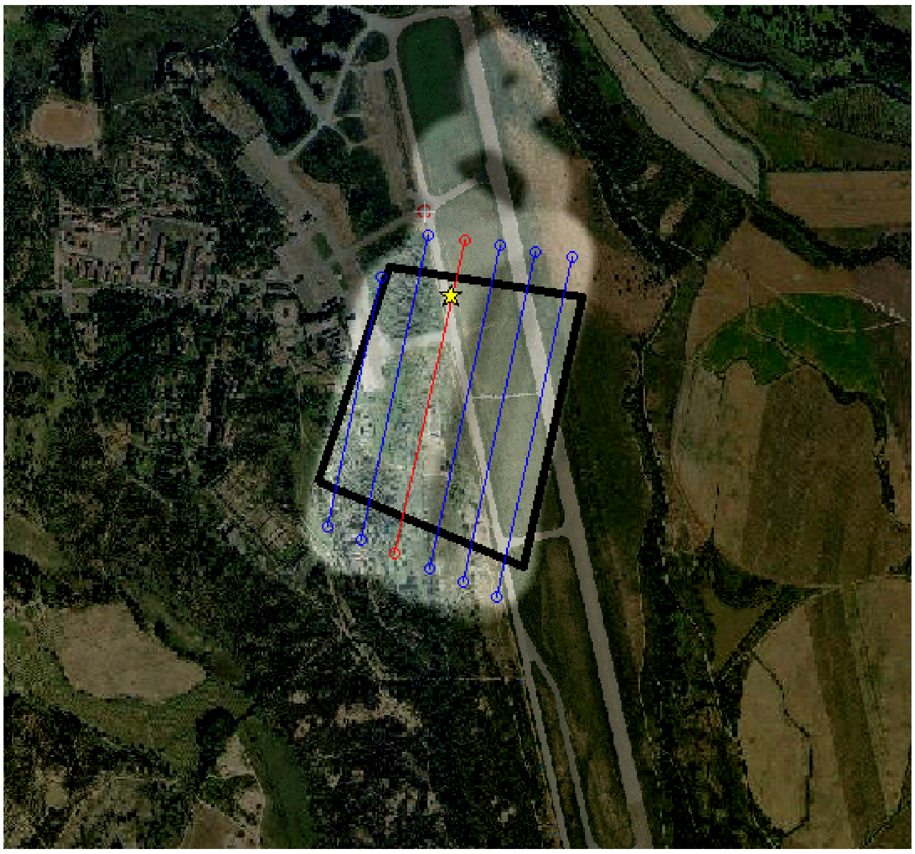


Figure 5.19: Fog-of-War graphical effect developed in Matlab for this project

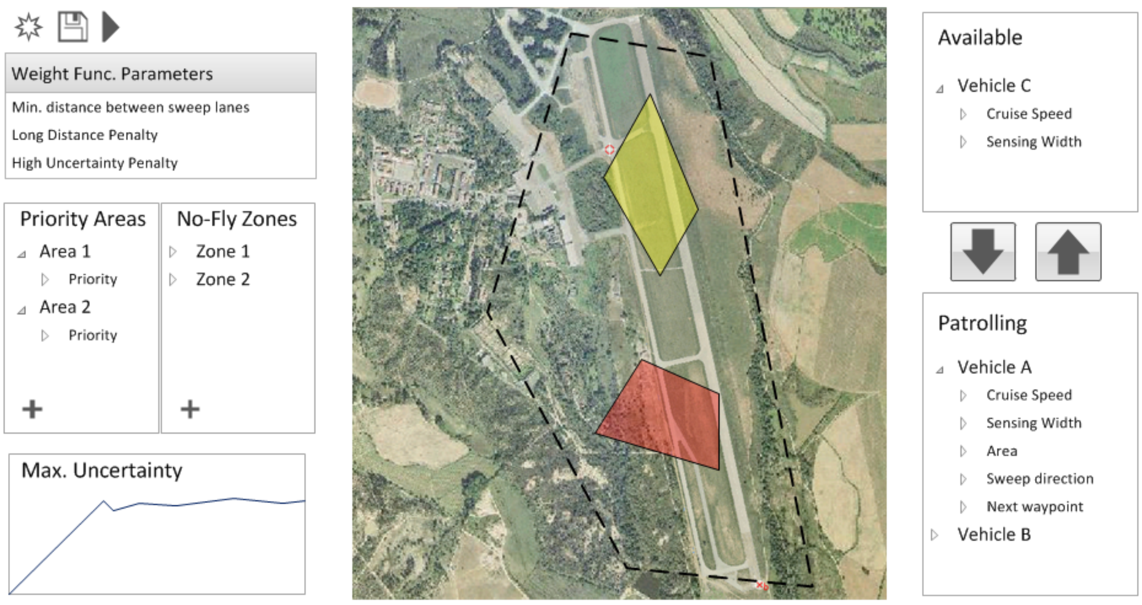


Figure 5.20: Mock-up of the GUI

5.4.2 Design

In order to better build a structured GUI, firstly a mock-up was drawn so that it was possible to gather opinions from Neptus users and developers. Fig. 5.20 shows the first draft of the GUI.

5.4.3 Final GUI

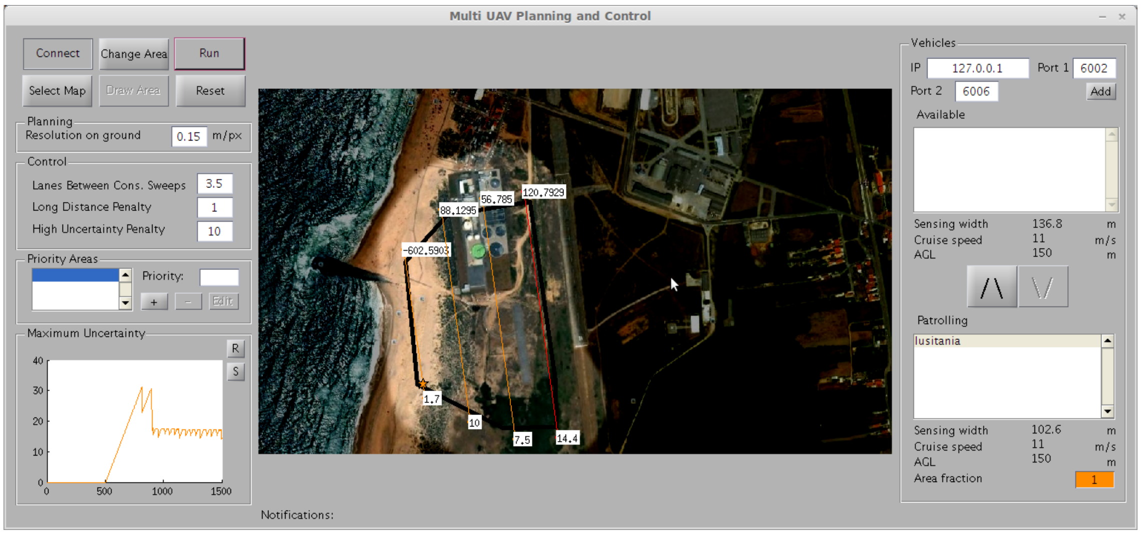
After analyzing all the criticisms and opinions, the GUI in Fig. 5.21 was designed and built. Some features were removed due to limitations of the algorithms.

5.5 System components

Due to limited time and the good prototyping abilities of Matlab, this was the software used to develop the console for multi UAV planning and control. Figures from the developed GUI will be of a Matlab generated GUI.

5.5.1 Area definition and partitioning

The area to be surveyed is input via a click and drag interface which is very intuitive to use.
When the user's input is confirmed, the system will have several ways of completing the area division, so it will ask the pilot for help. It will present the possible sweep/division line angles and the corresponding diameters achievable with each direction. Then it will wait for the user to select one of the directions.
After having defined the sweep direction, it looks at the selected UAVs and calculates their relative capabilities. With that information it estimates an optimal area division. That estimation is again



Resolution on ground 0.15 m/px
Lanes Between Cons. Sweeps 3.5
Long Distance Penalty 1
High Uncertainty Penalty 10
Priority:

IP 127.0.0.1 Port 1 6002
Port 2 6006

Sensing width 136.8 m
Cruise speed 11 m/s
AGL 150 m

lusitania

Sensing width 102.6 m
Cruise speed 11 m/s
AGL 150 m
Area fraction 1

Figure 5.21: Final GUI, in Matlab

presented to the user, who will have the choice to keep it or input its own.
Finally, each UAV will be assigned an area fraction, using the algorithm discussed before.

5.5.2 Path planner

After each UAV has its own area, the path planner will generate the waypoints necessary for the vehicles to perform the lanes previously mentioned in the modified lawnmower algorithm section. The procedure to find those waypoints is to intersect a set of line segments with the sweep direction that was previously set with the area polygon assigned to the vehicle. The line segments are all parallel to each other and are separated by a distance related to the sensor footprint (discussed on a previous section).

5.5.3 Uncertainty tracker

To use the algorithm discussed before (modified Lawnmower/Zamboni) it is necessary to keep a record of the uncertainty over the whole area. This data is also necessary for the Fog-of-War effect.
Uncertainty is recorded centrally, which means neither UAV knows the uncertainty, not even of their own area. Data is stored on a matrix (can be called an image) with the same dimensions as the map image in use. Each pixel in this gray-scale image contains the uncertainty on the location shown on the corresponding pixel in the map. This uncertainty image is used as the Fog-of-War mask to use on the map image.
As mentioned before, the control algorithm needs to know the maximum uncertainty in each lane. To get that information a mask is generated around each lane, which allows to ignore all the locations which are not visible from that lane, then it is simple to retrieve the maximum value among the visible locations.

5.5.4 Path controller

Every time one of the UAVs finishes a lane, the weight function for each of its assigned lanes is calculated. The weight function used is a tweaked version of the one in Eq. 5.18d:

$$W_i(t) = K_1 * U_{li}(t) - K_2 * \frac{1}{v_j} \max\{d_{ji}, a + \frac{b-a}{b} d_{ji}\} \tag{5.29a}$$

$$a = K_3 * b \tag{5.29b}$$

The user can change K_1, K_2 and b during the mission execution to find values that yield better results. K_3 is defined in the source code and cannot be changed during run-time.
The lane with the higher weight will be the next one to visit, so a IMC message is sent to the Dune process controlling the UAV (real or simulated). Communication between Matlab and Dune is possible thanks to the Java IMC library.
The lane sent to the vehicle appears in red on the GUI so it is easy to follow the algorithm's requests and what is happening in the "backstage".

Chapter 6

Results

After the implementation methods have been described, it is possible to present some of the results and test conditions.
Simulation tests were performed at LSTS, field tests were carried out in Espinho, at "Aeroclube da Costa Verde".
Details regarding the procedure to run the tests are in Appendix A.

6.1 Simulation Tests

To test any algorithm which affects the real world it is always a good policy to start by doing simulations. In this particular case, hardware and vehicles can add up to several tens of thousands of euros, so experiments in the real world before thorough simulation tests are completely out of question.

6.1.1 Software in the Loop

The first logical step in development are Software in the Loop (SiL) simulations, they are fast, practical and easy ways to test the correct behavior of the system since the very beginning. The setup for these simulations is show in Fig. 6.1. Initially, Matlab development was done under Windows environment, however, Linux proved more practical by allowing to run Dune on the same machine. The Windows environment represented in the figure is a server at LSTS that is permanently running simulations.
Simulations were performed with single and multiple UAV teams. Results form one of the simulations were the following:

Mission parameters:

- Area: 500000 m^2
- UAVs: 2
- Speed: 18 m/s

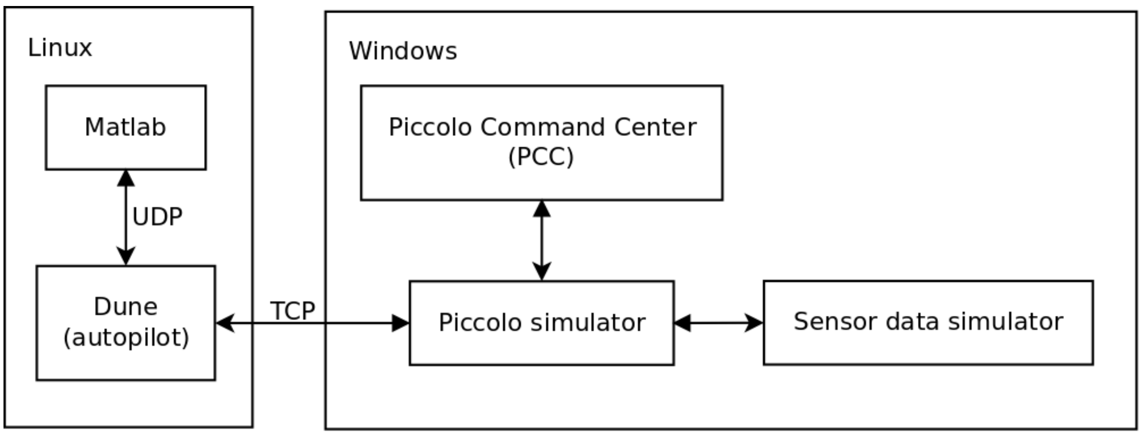


Figure 6.1: Software in the Loop system architecture

- Resolution on ground: 0.2 m/pixel
- Sensor Footprint Width: 140 m

By examining the performance results on this test mission, it is possible to predict the performance on a mission like the one described in the User Story (4.4).

Area:

$$\frac{500000m^2}{100Km^2} = 0.005 \tag{6.1}$$

UAVs:

$$\frac{2}{4} = 0.5 \tag{6.2}$$

Speed:

$$\frac{18}{22} = 0.82 \tag{6.3}$$

Sensor Footprint Width:

$$\frac{140}{360} = 0.38 \tag{6.4}$$

So the average time between visits to a location will be:

$$\frac{0.5 * 0.82 * 0.38}{0.005} = 31.2 \tag{6.5}$$

roughly 30 times larger than the one registered for these trials.
Uncertainty during the trial evolved as seen in Fig. 6.2. Each line represents the maximum uncertainty on the area assigned to each vehicle.
It is possible to see that the maximum uncertainty stabilized around 180 seconds (3 minutes). Taking into account these results, the performance on the User Story scenario would be of 1 hour 30 minutes between visits to each location.
The peaks in the orange line happened due to a communication issue, which made the UAV go to a safety point and stop patrolling. After the problem was solved, the vehicle returned to duty and

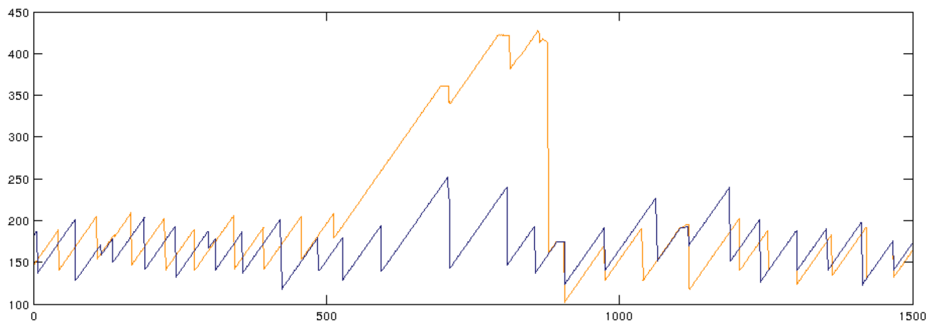


Figure 6.2: Uncertainty (in seconds since last visit) over time (seconds)

uncertainty lowered again.

6.1.2 Hardware in the Loop

Results with this simulation were similar to the previous (SiL), the autopilot software was the same and the sensor inputs as well, so that was expected. Nevertheless, Hardware in the Loop (HiL) is a very important step to flight tests, as the system visible by the developed program will be exactly the same as when the UAV is actually flying (see Fig. 6.3).
Only after the HiL tests were successful, a configuration was found that it would work for real world tests. That was an important lesson learned.

6.2 Field Tests

Flight tests were performed in Espinho, at the airfield. The platform used was the Cularis glider (remember 2.3.2.1) with a Piccolo SL autopilot on-board. Communication with the vehicle was done via the Piccolo link (2.4GHz). An Eee PC was next to the Piccolo Ground Station, translating the IMC messages sent by Matlab and sending back the telemetry from the UAV (see Fig. 6.4). With this configuration, there was no Dune running on-board of the vehicle.

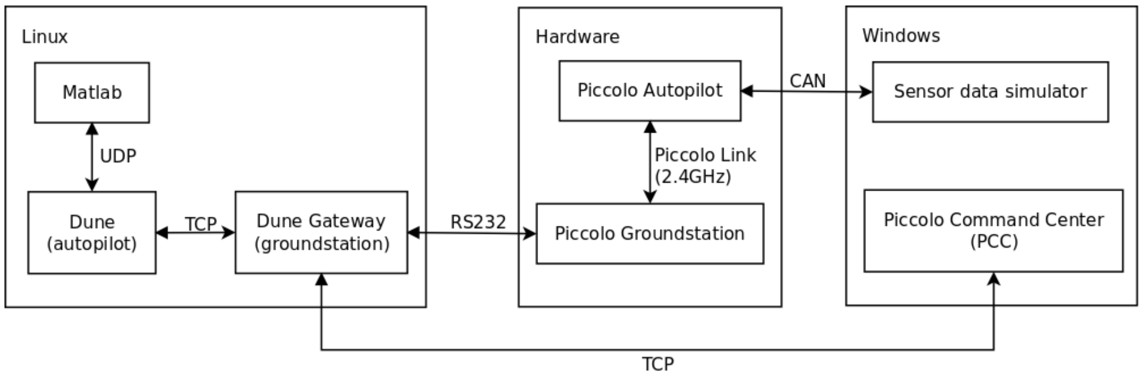


Figure 6.3: Hardware in the Loop system architecture

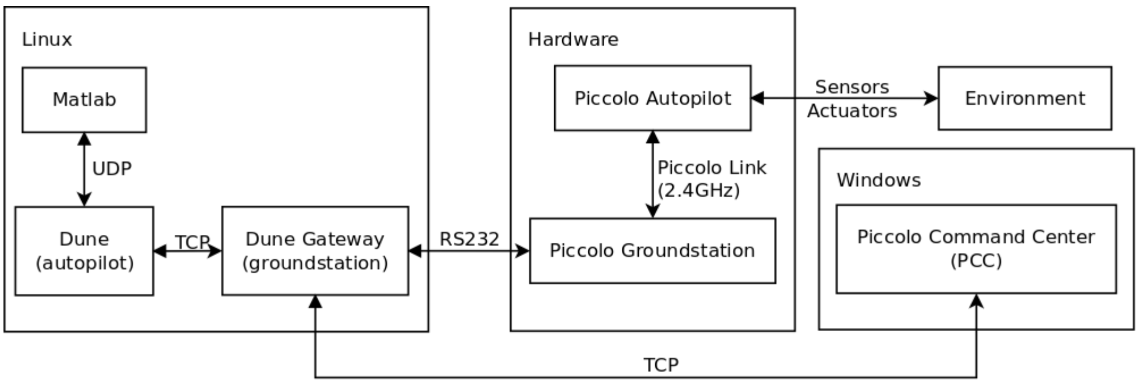


Figure 6.4: Field tests system architecture

Mission parameters:

- Area: 125000 m^2
- UAVs: 1
- Speed: 11 m/s
- Resolution on ground: 0.1 m/pixel
- Sensor Footprint Width: 72 m

By again examining the performance results on a test mission, it is possible to predict the performance on a mission like the one described in the User Story (4.4).

Area:

$$\frac{125000m^2}{100Km^2} = 0.00125 \tag{6.6}$$

UAVs:

$$\frac{1}{4} = 0.25 \tag{6.7}$$

Speed:

$$\frac{11}{22} = 0.5 \tag{6.8}$$

Sensor Footprint Width:

$$\frac{72}{360} = 0.2 \tag{6.9}$$

So the average time between visits to a location will be:

$$\frac{0.25 * 0.5 * 0.2}{0.00125} = 20 \tag{6.10}$$

20 times larger than the one registered for these trials. During the summer, the Portuguese coast is very windy throughout the day. Strong winds from North are felt near the sea all along the western coast. During the tests there was a strong wind (10-12 m/s) which was close to the indicated airspeed of the Cularis, so flying against the wind resulted in almost static position holding which

Figure 6.5: Piloting and testing the algorithms in Espinho

made the tests difficult.

Space constraints (near civilian airfield and houses) added up to the wind resulted in a chosen sweep direction far from optimal (actually perpendicular to optimal, which is the worse direction) with many lanes and turns.

The uncertainty over the area progressed as seen in Fig. 6.6.

It is possible to see that maximum uncertainty is stabilized around the 800 seconds (about 13 minutes), the initial peaks are explained by the initial position of the vehicle not being very close to the patrolled area.

Together with the previous calculations, it is reasonable to suppose that uncertainty in the User Story scenario would stabilize around 260 minutes (4 hours and 20 minutes) however, as it was mentioned, performance was very much affected by the wind and restrictions to flight space. Lower uncertainty can be expected in better conditions, especially with larger vehicles that are not so much affected by wind of this magnitude.

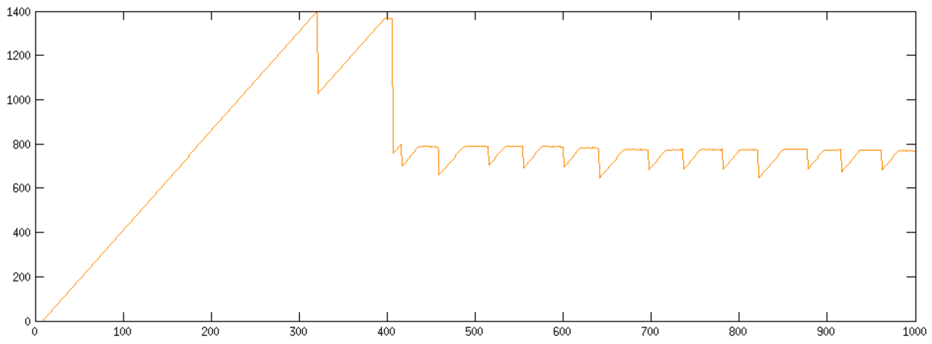


Figure 6.6: Uncertainty (in seconds since last visit) over time (seconds)

Chapter 7

Conclusions and Future Work

In this final chapter the results of this project will be summarized and some topics for future work will be discussed.

7.1 Summary

This thesis was started with the objective of implementing mixed initiative for planning and control of multiple UAVs in persistent surveillance missions. In order to accomplish this proposed objective, the following steps were taken:

- Initial study of the background of UAV technology, in order to understand capabilities and limitations of the vehicles.
- Familiarization with LSTS's software tool-chain so that the developed system would integrate with the existing architecture.
- Analysis of UAV operation procedures necessary to maintain the safety of humans and platforms.
- State of the Art review, which by allowing a better comprehension of current systems gives a good starting point for development and also results to compare with the ones achieved.
- Mixed initiative systems were studied that allow the supervision of multiple UAVs by a single person.
- Techniques for area division and search patterns were reviewed and later implemented and tested.
- Before the GUI was implemented, a mock-up was designed to gather opinions and requirements from users and developers.
- After that, the GUI was developed to allow the planning and control of persistent surveillance missions, which interacted with the existing systems. To make the interaction easy

and intuitive some graphical representations commonly found in strategy games were used on the interface.

- Finally the system was tested in simulation environment to ensure a correct behavior before flight tests were performed. Data was collected, analyzed and projected for larger areas and vehicles.

7.2 Achieved Goals

The proposed initial objective was successfully achieved, a system was developed that could effectively maintain persistent surveillance in a defined area, using multiple UAVs. Maximum uncertainty in the area was maintained at a stable level, and some predictions were made for larger scale applications.
Mixed initiative was implemented using knowledge from the human operator to improve the performance of the autonomous system.

7.3 Future Work

Even though the system can perform the proposed task, there is always room for improvement and further developments, here are some of them:

1. The Association for Unmanned Vehicle Systems International (AUVSI) holds an annual contest for students, the tasks involved include target location and area search [38]. Some of the results from this project maybe possible to use to participate in the next AUVSI Student UAS Competition.

2. An important development to build on the resulting system is an online system which can re-allocate UAVs as their autonomy decreases, other UAVs become available, and the search area or conditions change. For that to happen, vehicles need to be health-aware and a group health management system implemented.

3. The system was developed in Matlab due to its fast prototyping capabilities, however for regular mission use this should be integrated with Neptus and Dune.

4. The system was tested in controlled environment with carefully selected inputs, to make this console usable in the field it is necessary to test it intensively with Pilots that are not familiar with it and other UAVs so that the controller parameters can be adjusted to a satisfying level.

5. Mixed initiative is still a field to be further explored. There are many interfaces in Neptus that could take benefit from this type of interaction.

Appendix A

Test Setup

In order to test the system, there are a few steps to follow. Depending on the type of trial, SiL, HiL or field test, the initial procedure varies, however, the developed system will be unaware of the type of vehicles (simulated or real) that it is managing.

A.1 Vehicle

A.1.1 Start the Software in the Loop Simulation

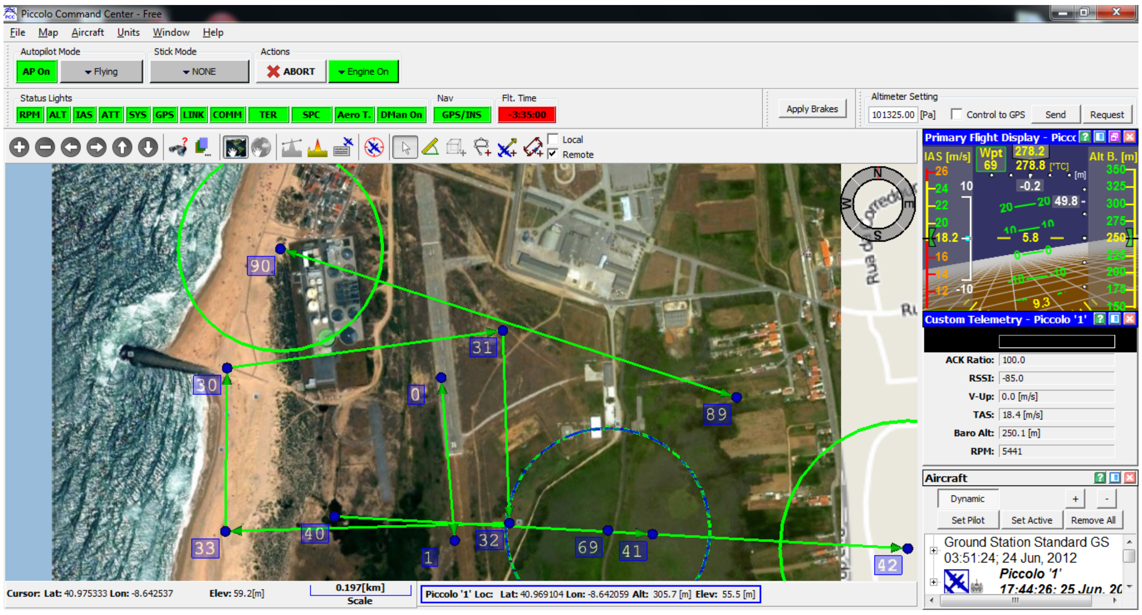
To run the simulation, Piccolo Command Center was used. It offers an interface like the real Piccolo to the remaining system so results obtained with this setup are expected to be similar to the ones obtained in real world tests. The program interface is show in Fig. A.1. System architecture when running this simulation is represented in Fig. 6.1.
This simulation is useful to verify the correct behavior of the planning and control logic.

A.1.2 Start the Hardware in the Loop Simulation

This simulation is very similar to the previous one, but now a real Piccolo is used instead of the simulated autopilot.
To use Piccolo, a simulation cable has to be connected to the hardware and back to the computer running PCC (see Fig. 6.3).
The Piccolo autopilot communicates with Dune via the Cloud Cap Ground Station, using its own radio link at 2.4GHz.
The main objective of this simulation is to test the communication between all the system components as the links used are the same as in the field tests.

A.1.3 Using a real Vehicle

For the field tests, the system architecture is represented in Fig. 6.4.
During flight tests there is always a pilot ready to take over the control of the aircraft, in case something does not work as expected.



ACK Ratio: 100.0
RSSI: -85.0
V-Up: 0.0 [m/s]
TAS: 18.4 [m/s]
Baro Alt: 250.1 [m]
RPM: 5441

Ground Station Standard GS
03:51:24; 24 Jun, 2012
Piccolo '1'
17:44:26; 25 Jun, 20

Cursor: Lat: 40.975333 Lon: -8.642537 Elev: 59.2[m]

Piccolo '1' Loc: Lat: 40.969104 Lon: -8.642059 Alt: 305.7 [m] Elev: 55.5 [m]

Figure A.1: Piccolo Command Center (PCC)

A.2 Start Dune

To start Dune a .ini file was created which contained all the settings needed to test the system (testconf.ini). Then the procedure is as simple as running the following command line in a terminal window:

```
./dune -c testconf -p Simulation
```

The option **-p Simulation** tells Dune to use the **Simulation** profile.

A.3 Start Matlab program

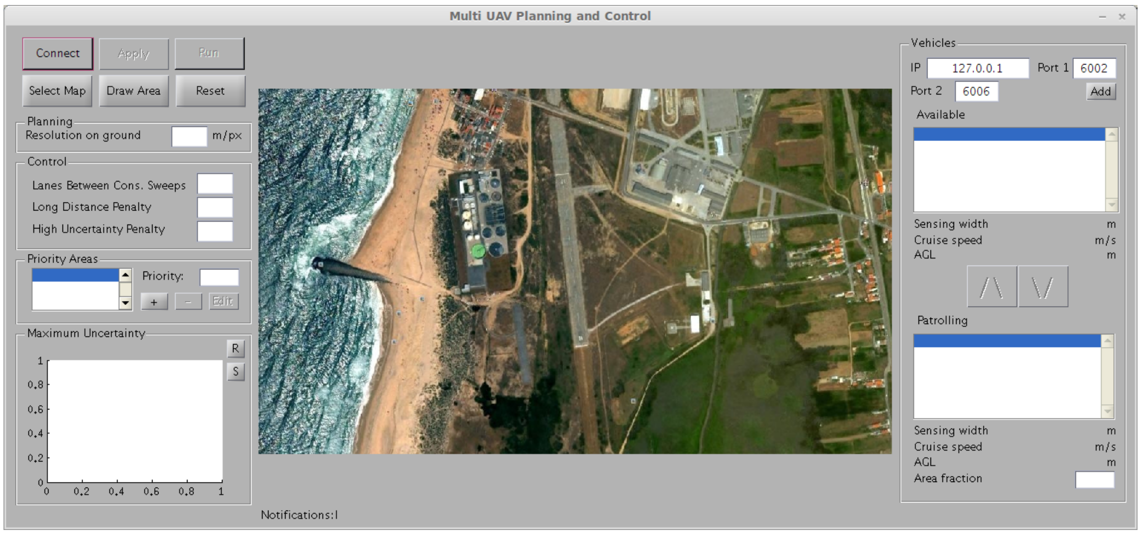
After starting the developed Matlab program, the window in Fig. A.2 shows up.

A.3.1 Choose Map

If the already loaded map is not the desired one, then the correct one should be selected by using a file navigation window which appears after clicking the **Select Map** button.

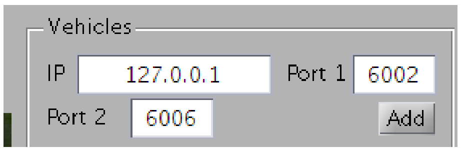
A.3.2 Add Vehicles

To add vehicles we need to know the IP address and ports defined in the .ini file – see Fig. A.3. If the connection is correctly started the name of the vehicle appears on the available vehicles list (Fig. A.4). After the vehicle has been correctly added, it is possible to get some details about it by clicking on its name (Fig. A.4).



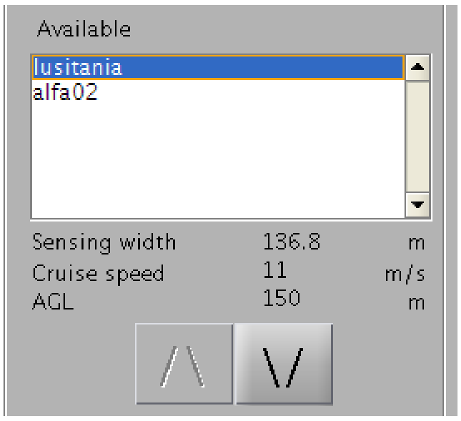
IP 127.0.0.1 Port 1 6002
Port 2 6006

Figure A.2: Multi UAV Planning and Control GUI



IP 127.0.0.1 Port 1 6002
Port 2 6006

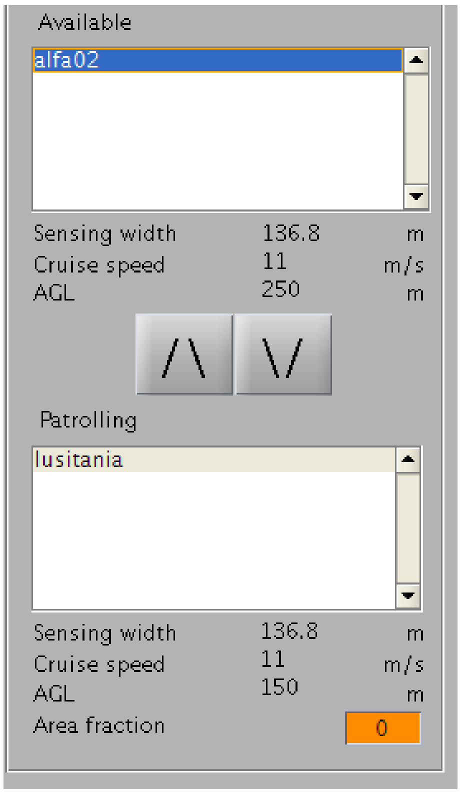
Figure A.3: Add a Vehicle by choosing its Dune IP and ports



lusitania
alfa02

Sensing width	136.8	m
Cruise speed	11	m/s
AGL	150	m

Figure A.4: After correctly adding a vehicle, it appears on the available list



Sensing width 136.8 m
Cruise speed 11 m/s
AGL 250 m

Sensing width 136.8 m
Cruise speed 11 m/s
AGL 150 m
Area fraction 0

Figure A.5: Vehicle Management panel

A.3.3 Assign Vehicles

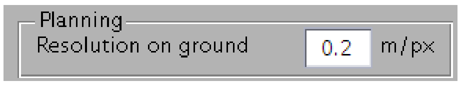
When there are available vehicles it is possible to assign them to the patrolling task. This is simply done by selecting the desired vehicle and clicking the downward arrow, which brings the vehicle to the Patrolling list (Fig. A.5). It is also possible to do the reverse operation (retire UAVs from duty). This is done by selecting it and pressing the upward arrow.

A.3.4 Set Resolution on Ground

One of the mission parameters is the required resolution on ground, this is where that is input (see Fig. A.6).

A.3.5 Draw Area

To define the area to be patrolled first click on the **Draw Area** button, then the cursor switches to a cross-hair and it is possible to click and drag points on the map until a closed polygon is drawn



Resolution on ground 0.2 m/px

Figure A.6: Resolution on ground

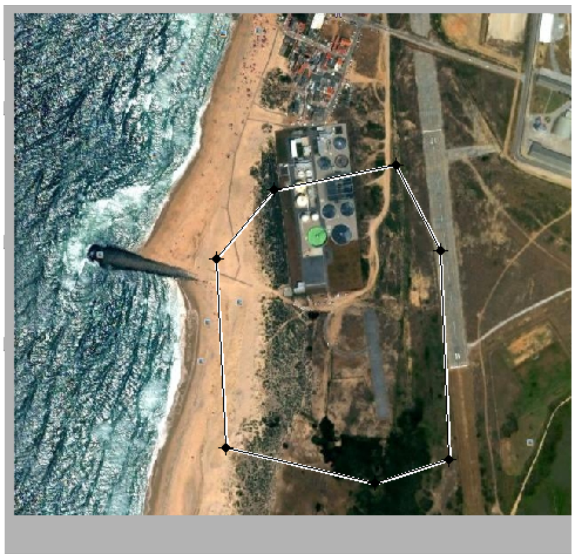


Figure A.7: Set the area to be surveyed by clicking and dragging

(see Fig. A.7). If the polygon is not convex, a warning will be given and the polygon is converted to the closest convex polygon (convex hull).

A.3.6 Sweep Direction

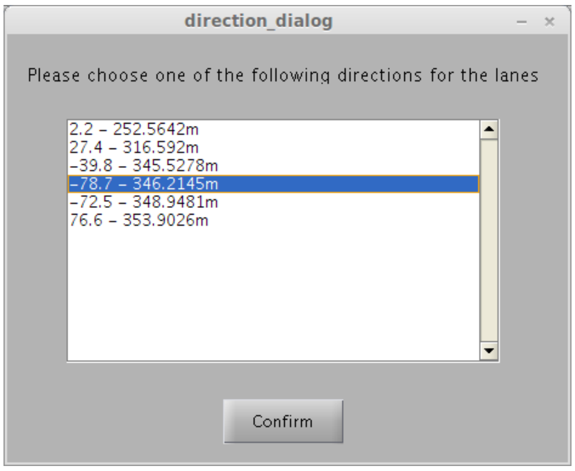
The possible sweep directions, which are parallel to the edges of the defined area, are presented in a dialog box, ordered in ascending order of the resulting diameter. In perfect weather conditions, the first solution is usually the one that yields the best results, however that might not be true with strong winds. See Fig. A.8.

A.3.7 Area Partitioning

Taking into consideration the relative capabilities of the vehicles, each of them gets a fraction of the area. However, an option is given to change the proposed area division. See Fig. A.9.

A.3.8 Run

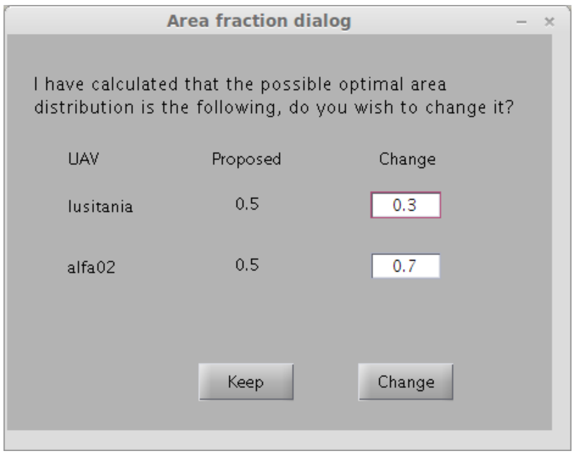
After the previous steps have been completed, the defined area will be marked with a black line, then it is possible to start the mission execution. After pressing the Run button, the program will take control of the vehicles as soon as they are sent to one of the handover waypoints (control is given to this system). The lanes will be marked with a color corresponding to the vehicle that is assigned to them. The ones drawn in red are the ones being executed at the moment. See Fig.A.10.



Please choose one of the following directions for the lanes

```
2.2 – 252.5642m
27.4 – 316.592m
-39.8 – 345.5278m
-78.7 – 346.2145m
-72.5 – 348.9481m
76.6 – 353.9026m
```

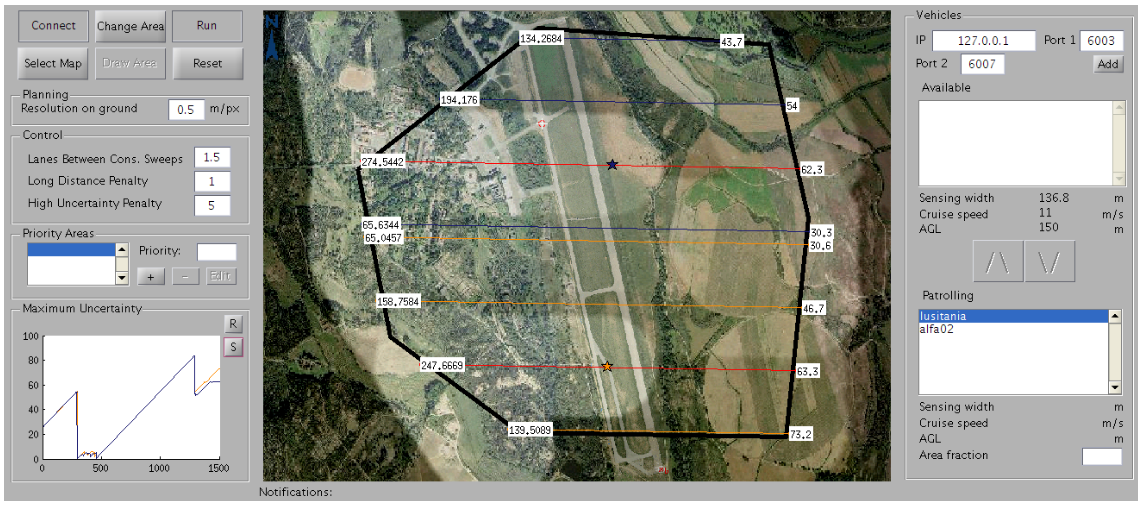
Figure A.8: Dialog showing the possible sweep directions (parallel to each side of the area polygon)



I have calculated that the possible optimal area distribution is the following, do you wish to change it?

UAV	Proposed	Change
lusitania	0.5	0.3
alfa02	0.5	0.7

Figure A.9: Set the fraction of the total area that each UAV will be assigned



Planning
Resolution on ground 0.5 m/px

Control
Lanes Between Cons. Sweeps 1.5
Long Distance Penalty 1
High Uncertainty Penalty 5

Priority Areas
Priority:

Vehicles
IP 127.0.0.1 Port 1 6003
Port 2 6007

Available

Sensing width	136.8	m
Cruise speed	11	m/s
AGL	150	m

Patrolling

lusitania
alfa02

Sensing width	m
Cruise speed	m/s
AGL	m
Area fraction	

Figure A.10: Program running with two simulated UAVs flying over Ota airbase

A.3.9 Adding Priority Areas

Sections with higher priority can be defined in a similar way as the area to be patrolled, by clicking on the map to form a polygon. Then, we need to set the priority of that area. After that we only need to save it.
Note: Only the sections inside the surveying area will be considered.

References

[1] Elmer A. Maravillas and Elmer P. Dadios. *FIRA Mirosot Robot Soccer System Using Fuzzy Logic Algorithms*. InTech, 2010.

[2] RPAV - The Radioplane Target Drone. http://www.ctie.monash.edu.au/hargrave/rpav_radioplane3.html, accessed in June, 2012.

[3] The Ryan Aeronautical Photo Collection. http://www.flickr.com/photos/sdasmarchives/sets/72157626514780879/with/5684502263/, accessed in June, 2012.

[4] Wikipedia - General Atomics MQ-1 Predator. http://en.wikipedia.org/wiki/MQ-1_Predator, accessed in June, 2012.

[5] aerospaceweb - parts of an aircraft. http://www.aerospaceweb.org/question/design/q0101.shtml, accessed in June, 2012.

[6] Official arduplane repository. http://code.google.com/p/ardupilot-mega/wiki/APM2board, accessed in June, 2012.

[7] Cloud Cap Technology. http://www.cloudcaptech.com/, accessed in June, 2012.

[8] Brett Bethke, Jonathan P. How, and John Vian. Group health management of UAV teams with applications to persistent surveillance. *2008 American Control Conference*, pages 3145–3150, June 2008.

[9] ML Cummings, S Bruni, S Mercier, and P J Mitchell. Automation architecture for single operator, multiple UAV command and control. *The International C2 Journal*, 1(2), 2007.

[10] Jeremy W. Baxter, Graham S. Horn, and Daniel P. Leivers. Fly-by-agent: Controlling a pool of UAVs via a multi-agent system. *Knowledge-Based Systems*, 21(3):232–237, April 2008.

[11] Royal Air Force. http://www.raf.mod.uk/newsweather/index.cfm?storyid=B2179573-1143-EC82-2E5B2BAAA882D0F5, accessed in June, 2012.

[12] Ivan Maza and Anibal Ollero. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. *Distributed Autonomous Robotic Systems 6*, pages 221–230, 2007.

[13] Wesley H Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *2001 IEEE International Conference on Robotics and Automation*, 2001.

[14] Nikhil Nigam and Ilan Kroo. Persistent surveillance using multiple unmanned air vehicles. *Aerospace Conference, 2008 IEEE*, pages 1–15, 2008.

[15] Jarurat Ousingsawat. UAV Path Planning for Maximum Coverage Surveillance of Area with Different Priorities. *The 20th Conference of Mechanical Engineering*, (October), 2006.

[16] Optimal Golf Solutions. http://optimalgolfsolutions.com/faq.html, accessed in June, 2012.

[17] SC**II**Mapster. http://www.sc2mapster.com/forums/development/data/22026-darker-fog-of-war/, accessed in June, 2012.

[18] Van Cleave and Dale W. *Trends and Technologies for Unmanned Aerial Vehicles*, pages 9–26. John Wiley & Sons, Inc., 2005.

[19] Stephanie Rosenthal, Joydeep Biswas, and Manuela Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. *Conference on Autonomous Agents*, 2010.

[20] Paulo Sousa Dias, Rui M F Gomes, Fernando Lobo Pereira, José Pinto, João Borges Sousa, and Gil M Gonçalves. Mission planning and specification in the Neptus framework. In *ICRA 2006*, number May, pages 3220–3225, 2006.

[21] Ricardo Martins, Eduardo R B Marques, Fernando L Pereira, João Borges Sousa, Paulo Sousa Dias, and José Pinto. IMC : A Communication Protocol for Networked Vehicles and Sensors. *Online*.

[22] John Tisdale, Allison Ryan, David Tornqvist, and J. Karl Hedrick. A multiple UAV system for vision-based search and localization. *2008 American Control Conference*, pages 1985–1990, June 2008.

[23] António Sérgio Ferreira. *Interface de Operação para Veículos não Tripulados*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2011.

[24] J. Sousa, T. Simsek, and P. Varaiya. Task planning and execution for UAV teams. *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, (0):3804–3810 Vol.4, 2004.

[25] S Thrun, G Gordon, M Burstein, and D Diller. Mixed-Initiative Control of Autonomous Unmanned Units Under Uncertainty. *Information Systems*, (February), 2006.

[26] David Landén, Fredrik Heintz, and Patrick Doherty. Complex task allocation in mixed-initiative delegation: A UAV case study (early innovation). *Proc. PRIMA*, 2010.

[27] Ora Lassila, Stephen F Smith, and Marcel Becker. Configurable, mixed-initiative systems for planning and scheduling. In *ARPI 1996*, 1996.

[28] Ephrat Bitton and Ken Goldberg. Hydra: A framework and algorithms for mixed-initiative UAV-assisted search and rescue. *Automation Science and Engineering,*, pages 2–7, 2008.

[29] E Pereira, R Bencatel, J Correia, L Félix, G Gonçalves, J Morgado, and J Sousa. Unmanned Air Vehicles for Coastal and Environmental Research. *Journal of Coastal Research*, 2009(56):1557–1561, 2009.

[30] Joint Publication. Joint Publication 2-0 Joint Intelligence. (June), 2007.

[31] Allison Ryan, Xiao Xiao, Sivakumar Rathinam, John Tisdale, Marco Zennaro, Derek Caveney, Raja Sengupta, and J Karl Hedrick. Collaborative flight demonstration with three unmanned aerial vehicles. 2006.

[32] ML Cummings and AS Brzezinski. Global vs. local decision support for multiple independent UAV schedule management. *Journal of Applied Decision Sciences*, 3(3):188–205, 2010.

[33] MBARI Oceanographic Decision Support System (ODSS). https://code.google.com/p/mbari-dss/, accessed in June, 2012.

[34] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer-Verlag, 1985.

[35] RTP. http://www.rtp.pt/noticias/index.php?article=521047&tm=8&layout=123&visual=61, accessed in June, 2012.

[36] Antonios Tsourdos, Brian White, and Madhavan Shanmugavel. *Cooperative path planning of unmanned aerial vehicles*. John Wiley & Sons, 2011.

[37] Wikipedia - Fog of war. http://en.wikipedia.org/wiki/Fog_of_war, accessed in June, 2012.

[38] AUVSI Seafarer Chapter. http://www.auvsi-seafarer.org/, accessed in June, 2012.