An autonomous multi-UAV system for avalanche search

Guilhem Bryant
guilhemb@stud.ntnu.no
NTNU - Master thesis

July 3, 2019

Abstract

This work aims at theorizing and designing a multi-UAV system for localizing avalanche victims in an optimal way.

Using UAVs instead of human rescuers for victim search offers several advantages such as their high velocity, their agility and the fact that they do not feel pressure, which often leads to human mistakes.

Furthermore, using multiple UAVS make missions possible whatever the amplitude of the disaster thanks to the scalability of multi-agent systems. It also provides robustness through redundancy: if a UAV fails, the rest of the fleet keeps operating. Finally, it divides the time for finding victims by the number of UAVs working, and this is a property of primary importance when big avalanches happen.

The actual rescue is not considered due to obvious limitations in terms of UAV payload, hence this task is left to human rescuers.

It is also mandatory for victims to emit some kind of signal so that agents may sense it and try to localize the victim. Throughout this work, focus is given to avalanche transceiver, although alternative technologies are studied.

The final contribution yielded by this thesis is a detailed abstract model of the system, offering an overview of the wide range of challenges to overcome to make this system real. Additionally, a simulation ecosystem is deployed to test the system.

Contents

Chapter 1

Introduction about UAVs and avalanches

In the last decade unmanned aerial vehicles have significantly gained in popularity, thanks to the constant miniaturization of electronic components and the progress in aeronautics. Nowadays, drones can be used for a wide range of activities such as agriculture, package delivery, photography, search and rescue, target acquisition and surveillance. This tendency is still observable as new applications of UAVs are found every day.

Drones do not need for a pilot. They are fully autonomous or controlled remotely, which allows them to carry out dangerous missions without jeopardizing human lives. Another advantage of UAVs –especially small UAVs– is their relatively low cost.

On another hand, snow avalanche is a physical phenomenon claiming hundreds of lives every year. Despite decades of experience in grooming rescuers and designing avalanche equipment, the number of victims does not seem to decrease [6].

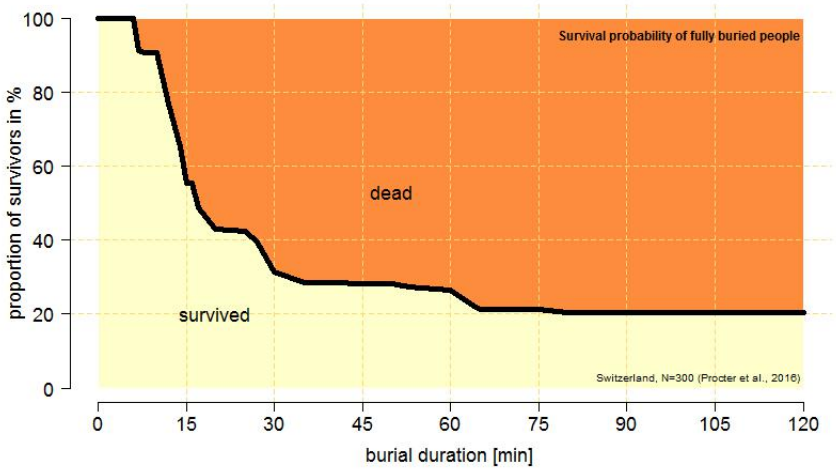


Figure 1.1: Survival probability of buried victims depending on time

This trend may even get worse with climate change, as brutal temperature change is known to destabilize snowpack. The rise in popularity of recreational activities, in particular off-piste skiing and backcountry touring, does not help reducing casualties.

As the process of locating and extracting victims is conducted by rescuers, the bottleneck in avalanche search and rescue efficiency appears to be human. Moreover it is commonly recognized that victims have the best chance of survival within the first 15 minutes; unfortunately the first rescuers usually arrive on scene when this deadline is reached.

This work aims at theorizing and designing a multi-UAV system for localizing avalanche victims in an optimal way. Using UAVs instead of human rescuers for victim search offers several advantages such as their high velocity, their agility and the fact that they do not feel pressure, which often leads to human mistakes. The actual rescue is not considered due to obvious limitations in terms of UAV payload, hence this task is left to human rescuers.

Is it even relevant to use multiple UAVs for avalanche search? It is true that avalanches are usually small, with a typical length of 150 m [5]. Such a small surface would be covered by a UAV within minutes. Furthermore, a survey showed that in 94% of avalanches, less than three persons are buried [14].

However, a multi-UAV system offers several advantages. It is scalable, i.e. it functions whatever the amplitude of the disaster. It provides robustness through redundancy: if a UAV fails, the rest of the fleet keeps operating. Finally, it divides the time for finding victims by the number of UAVs working, and this is a property of primary importance when big avalanches happen. In the 1970 Ancash earthquake, more than 20,000 people died from the avalanche triggered. On the downside, such systems are more expensive and more complex to design.

The first chapter presents some theoretic background, crucial to design the multi-UAV system. The state of the art from various topics is brought together to find answers to the stated problem. In the second chapter, an abstract model to find avalanche victims is developed. A distinction is made between the coverage process –i.e. finding victims– and the actual victim localization. Chapter 4 introduces the implementation of the abstract model. Finally, results are showcased and analysed in chapter 5.

Chapter 2

Preliminary background

2.1 General state-of-the-art

Some UAV prototypes already exist for the purpose of autonomously finding avalanche victims or similar.

Politecnico di Torino's prototype [16] applies the same coverage technique as the one used by human rescuers. It exhibits a retractable antenna to sense victims, which unfolds far from the UAV's avionics to limit electromagnetic noise.

Alcedo was developed by EPFL students in 2011 [15]. It explores various localization methods as explained in section 3.3.2.

No evidence of the existence of a multi-UAV system for avalanche search was found.

2.2 Avalanche transceivers

Avalanche transceivers (AT) is the most common technology in use. Invented in the 1960s, this is considered the most mature and reliable solution so far. An avalanche transceiver is a telecommunication device made of two separate components: transmitter and receiver. This device has to be carried and kept turned on by mountaineers and rescuers. Users can toggle between those two modes depending on the situation.

In send mode, a signal is transmitted on a short period of time (about 70 ms) every second, as illustrated in figure 2.1. This behaviour allows for energy saving.

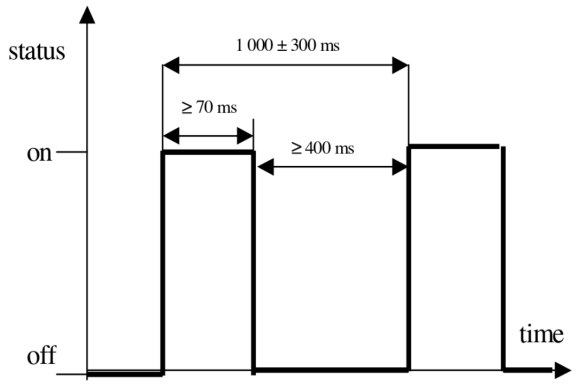


Figure 2.1: AT signal periodicity according to ETSI EN 300 718

In search mode, the AT is listening for signals emitted by victims' AT and gives information about victims' locations to the rescuer. Analogue ATs, marginally used nowadays, convert the received signal into a sound according to signal strength. To be operated adequately, this kind

of AT requires experienced rescuers. It is for example common practice to initiate the search process by finding the optimal receiving antenna's orientation for each victim.

Digital ATs, their modern counterparts, are more user-friendly. They are equipped with multiple antennas to correct wrong orientations and displays the distance to the victim and the direction to follow. They also support multiple burial cases, i.e. they are able to distinguish victims. Some digital ATs support an additional channel on another frequency to send a unique ID along the main signal, which further helps the receiver in multiple burial cases. Furthermore, accelerometers can be used to sense the victim's orientation and use a secondary antenna to maximize the transmitted signal strength, and to automatically switch to send mode when the carrier is knocked over by the avalanche. Finally, some ATs can monitor vitals to help rescuers decide which victim to save.

Although very different, both types of AT are compatible with each other.

Avalanche transmitters emit a periodic signal at a standardized frequency of 457 kHz. This is equivalent to a 656 m wavelength, benefits from a good propagation in snow, trees and is not much attenuated by any object it may go through in a mountainous environment. This wavelength gives a near-field area of about 100 m radius, in which signals are essentially magnetic [9].

As a consequence, only ferrous objects can have a negative effect on propagation. The downside is that the radiation pattern is quite complex to model in that area: fields are curved, which makes it really difficult to localize the emission source. Furthermore, the field projected on the surface may vary widely depending on the orientation of the transmitter's antenna, which brings additional complications.

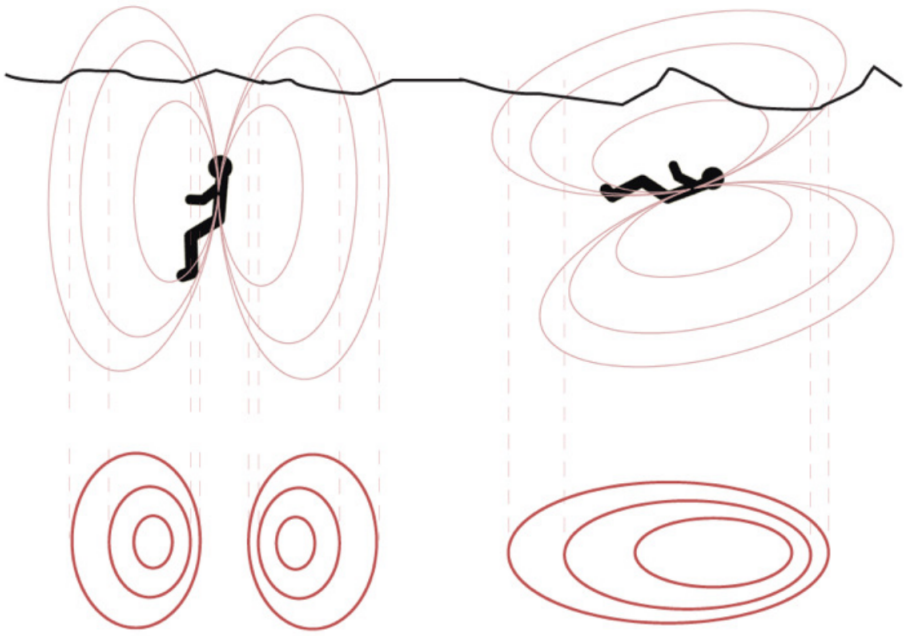


Figure 2.2: Ground projection of field lines generated by an AT, depending on the victim's orientation

As seen on figure 2.2, two fields, hence two maximums, may be perceived on the surface instead of one. This may trick the rescuer into thinking he went beyond the emission source.

Besides this issue, the received signal strength depends on the relative orientation of the receiver's and the transmitter's antenna, though the latter is constant if we suppose that the victim is not able to move once buried. As a mitigation, digital ATs use multiple antennas to improve reception but this method introduces latency due to additional steps in signal process-

ing and algorithmic overhead. This negatively impacts the mission, as it introduces a desync between sensed data and the position where this data was sensed. This error grows with the speed of agents.

Another issue lies in the very design of digital ATs. The output is tailored to humans, which makes them unsuitable for a use onboard UAVs. For example, some computations rely on human walking speed [15]. Finally, ATs are expensive, which explains why so many mountaineers do not invest in this kind of equipment.

Despite all those limits, this technology is still a good candidate for the project for the following reasons:

- ATs have proven to work well
- A system made of UAVs equipped with ATs would be backward compatible with human rescuers
- The frequency used by ATs is standardized, which reduces the risk of interference

Alternative search technologies are described in 3.3.1.

2.3 Search and rescue process

Figure 1.1 shows that about 90% of victims survive to a complete burial if they are buried for less than 10 minutes. The survival rate drops to 30% after 30 minutes. This corresponds to death by suffocation due to an insufficient air pocket. After that time, the survival rate does not decrease much and death is caused by trauma [23]. Death by hypothermia occurs a long time later.

Those statistics emphasize how quickly rescuers need to operate to save lives.

The process of saving avalanche victims is the following. Once local rescuers are informed of the disaster, they get on the scene by helicopter or snowmobile, unless of course the rescuers are the witnesses themselves (companion rescue). The leader organizes the team, delimits the search area and decides the sweep strategy. The last known position of victims helps reducing the size of the search area.

During the primary search, rescuers spread horizontally or vertically, then cover the search area from top to bottom in rectangular search strips or in zigzags, as shown in figure 2.3. The search strip width is a crucial parameter that has to be carefully chosen. A big value makes the sweeping faster but increases the probability of missing victims. Search strip width optimization is discussed in [13] according to a multitude of factors.

Once a signal coming from a victim's transmitter is detected, the objective is to progressively get closer to the victim following the indications provided by the search device (direction if using a digital AT): that is the fine-search phase. The closer to the signal source, the stronger the signal gets.

The rescuer finally finds the emission source then probes the ground with poles to precisely locate the victim. Once the position of a victim is determined, rescuers use shovels to excavate the victim: first the face is freed, then the chest to establish an airway. Finally, the victim is given first aids.

If rescuers reached the bottom of the search area and no victims have been detected yet, it is the sign that they are not equipped with an AT, in which case rescuers have to probe the whole area again using poles, which is extremely time consuming.

As we can see, for a given search surface, the time it takes to save victims essentially depends on the time to locate victims by walking on snow and the time to excavate victims. Using drones would not impact the latter, but would drastically reduce the former.

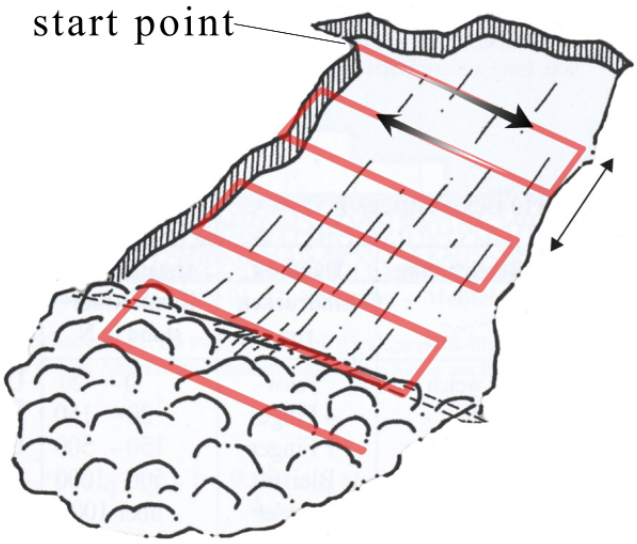


Figure 2.3: Search area swept in rectangular strips

2.4 Unmanned aerial vehicles

An unmanned aerial vehicle (UAV) is an aircraft with no human pilot onboard, remotely controlled or autonomous. Figure 2.4 introduces various aircraft classes. Rotorcrafts and fixed-wing aircrafts are two classes of aerodynes, i.e. heavier-than-air aircrafts. The choice of a class over another depends on the missions UAVs have to accomplish.

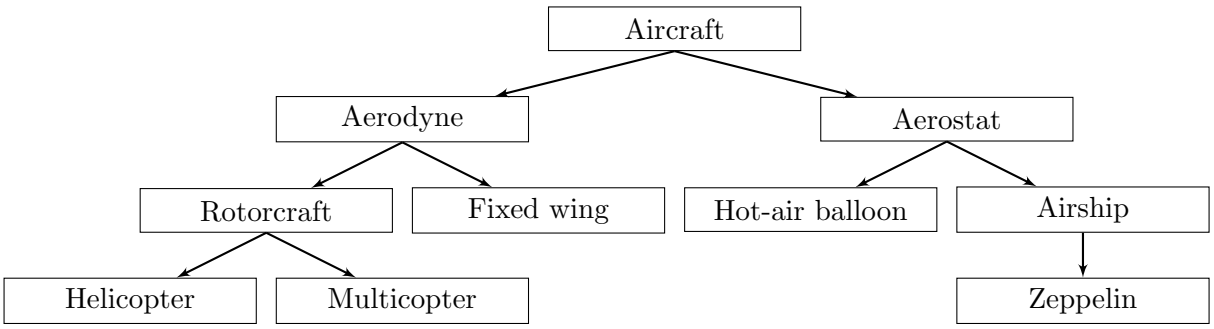


Figure 2.4: Taxonomy of aircrafts

Generally, fixed-wing aircrafts are used when there is a need to cover long distances, for example to map areas. In contrast, rotorcrafts suffer from a smaller operational duration (around 20 minutes) because they are much more energy-consuming to maintain in the air. This limit is ignored all along this report because 1) UAVs may takeoff from a near station, 2) battery technologies evolve fast, 3) alternative ways of powering UAVs exist (e.g. laser

On another hand, the design of rotorcraft makes them highly maneuverable and able to hover. This type of aircraft is also easier to build and cheaper. These are the reasons why rotorcrafts, in particular multicopters, are so popular in civil UAV applications such as agriculture and photography.

For the aforementioned reasons and for the sake of simplicity, it is decided that UAVs are multicopters from that point on.

2.4.1 Physics

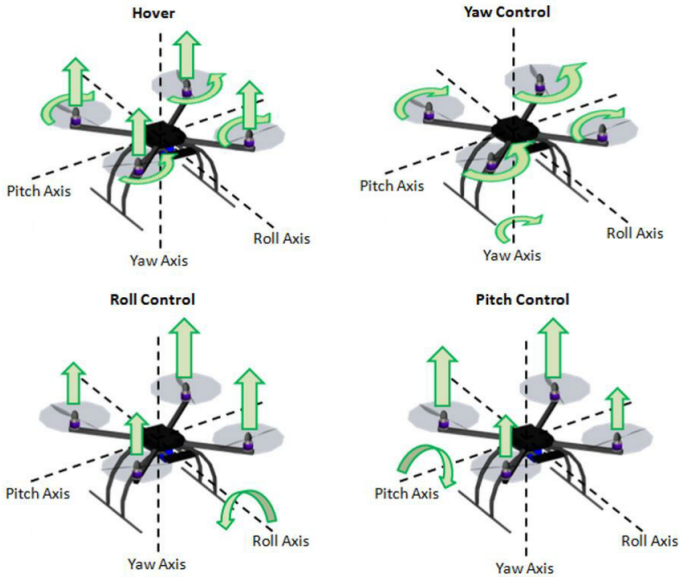


Figure 2.5: Attitude control of a multicopter

Multicopters are made of rotors (usually 4, 6 or 8). They provide both propulsion and lift, the two mandatory conditions for an aircraft to fly actively. Rotors are assembled in pairs to spin the same way. Under the condition that speed rotation is balanced, this cancels the resulting momentum on every axis, consequently the UAV moves on the yaw axis, i.e. it is whether ascending, descending or maintaining height. Otherwise, the orientation of the aircraft is controlled by adjusting the rotor speed in pairs: respectively increasing and decreasing the speed of rotors 1-2 and rotors 3-4 induces a change on the yaw axis. Similarly, adjusting the rotor speeds of the other sets of pairs controls the aircraft on the roll and pitch axes. The action of each rotor combination is illustrated in figure 2.5.

The flight controller is in charge of choosing appropriate rotor speed values to steer the UAV as intended. The inertial measurement unit (IMU) includes accelerometers, gyroscopes and magnetometers. Its purpose is to measure the orientation of the aircraft on each axis in order to stabilize the aircraft.

The location and orientation of an object in space can be expressed in various geographical coordinate systems. The following are commonly used in aviation:

- Local tangent plane: based on a vertical direction from a point on Earth. Three coordinates define the system: (east, north, up) or (north, east, down). This coordinate system is very relevant in a simulation environment because it corresponds to its coordinate system, give or take a sign
- Global coordinate system: defined by latitude, longitude and altitude. Altitude is measured according to a reference:
  - above sea level
  - above home (takeoff location by default)
  - above the terrain, measured by an on-board sonar or rangefinder, or from topographic terrain data. This is the most interesting option in regard of the project, as UAVs should follow the terrain at a certain altitude: close enough to acquire the most accurate victim measures, but far enough to avoid disturbing rescuers on ground. 2 m is a good compromise

2.4.2 System components

The following list enumerates the components of UAVs, in particular UAVs qualified to complete the mission of looking for avalanche victims:

- Frame: element that brings all components together. It has to be flexible to sustain vehicle crashes without breaking. This should not be confused with reference frames a.k.a. coordinate systems
- Electronic speed controller: as the name suggests, it controls the speed of motors. It receives input from the flight controller via PWM signals
- Motors: can be electrical or gas-fueled
- Propellers: mounted on motors. The air that goes through them is accelerated to provide thrust. They are characterised by the number of blades, their length and pitch. A trade-off between efficiency and thrust is to be found by adjusting those parameters
- Battery: powers all electric components on-board. LiPo batteries are the most popular due to their high energy density
- Communication module: in respect of the project requirements, communicating with other UAVs is essential, as well as indicating victim positions to rescuers. In addition, a human pilot should be able to take over at any time. Although regulations do not clearly stipulate this, it is either way a reasonable safety measure
- GNSS (GPS, Galileo or GLONASS): mandatory for some navigation modes, as dead reckoning is not a full substitute to absolute positioning. In the context of this application, it is required to move relatively to a position on a map.
- Flight controller: composed by the IMU and the autopilot, a real-time critical piece of software in charge of controlling the low-level behaviour of the aircraft (stabilization, navigation modes, waypoint-based path following, etc.). It runs on a dedicated board
- Companion computer: non-critical counter part of the flight controller unit. It is used to carry out tasks unrelated to flight, such as automatic path planning, networking or computer vision. Due to its softer constraints, it is usually running on a separate CPU or system on chip

A UAV along with a ground station makes an unmanned aerial system. A human can control UAVs through a software interface. It is for example possible to change the flight mode, add waypoints, monitor flight parameters (current speed, orientation, altitude, etc.).

2.5 Multi-agent systems

2.5.1 Definitions

Although there is no consensus, an agent is commonly defined as an entity capable of autonomous actions and interactions with other agents, through cooperation or negotiation [17]. Agents evolve in a certain environment, from which subjective knowledge, beliefs, is drawn. Even in the same environment, agents may have different beliefs depending on the way they sense it. Multiple agents united by some goal or mission constitute a multi-agent system (MAS).

Multi-agent systems share a lot of similarities with the research area of distributed systems. A MAS is distributed by essence, that is there is no agent designated as the decision maker. Such a system is therefore more robust through redundancy (i.e. there is no single point of failure), but this makes it more complex, due to the need for efficient communication among other things. The major difference with distributed systems is the human-like behaviour agents share, in that they have some kind of intelligence that enables them to dynamically adjust to the

environment. Moreover, they are usually self-interested and independent in terms of goals. On the contrary, the elements of a typical distributed system implicitly share the common goal of making the overall system function correctly. They rely on hardwired coordination mechanisms, hence there is no need for sophisticated communication mechanisms such as negotiation. It is worth mentioning that agents can be benevolent as well, i.e. they can strive towards a common goal as this is the case for distributed systems. This is a kind of behaviour that is especially suitable for search and rescue.

Because agents behave like humans, MAS has deep roots in the field of artificial intelligence. Depending on mission requirements, agents may have to run complex algorithms to move in the environment (e.g. reinforcement learning, collision avoidance, etc.) or make any other kind of heavy computation. The difference is that artificial intelligence is about the components of intelligence and has no interest about social intelligence exhibited by agents. MAS focuses on the entities integrating those components, e.g. the way AI can be used by agents to solve problems of interest so that they can fulfill their mission.

A MAS can be made of an heterogeneous agent set. In that case, agents have various capabilities, making them experts for some specific tasks and incompetent for others. For example, an agent may be able to see whereas another may be able to hear. This is an aspect that may emerge in real-life applications out of budget or physical limitations (e.g. limited amount of sensors mounted on an agent). Accidents during a mission can also cause agents to focus on a more limited set of capabilities. For example, an agent whose main sensor stops working might want to switch to a secondary sensor.

2.5.2 Task sharing

Task sharing is a crucial feature of MAS which allows the system to reach synergy by combining the potential of multiple agents. It is important to find a compromise in task granularity: small tasks imply communication overhead and reduce the benefit of task sharing (communication is slower than computation), whereas large tasks entail a bigger negative impact on performance if for some reason no response is received by the manager. In the contract net protocol [21], a manager proposes a task to a set of agents (contractors). The announcement is sent to contractors and comprises the task description and some additional constraints (e.g. deadlines). In the next step, each contractor interested in the task makes an offer, depend on their expertise. Finally, the manager assesses the bids, select the best candidate (or candidates) and makes a contract with him.

2.5.3 Contextualization

The MAS designed through this work consists of benevolent agents, i.e. helping each other as much as possible, which seems to be the best approach for search and rescue. A full collaboration between agents devoting themselves to the cause of finding missing persons is indeed more productive than self-interest. For instance, a greedy agent would not reveal its beliefs, i.e. the areas it already visited, in order to achieve more success than the others, which would lead to some areas being repeatedly visited by different agents.

It should be noted that there are some advantages in self-interest. In case a rogue agent somehow infiltrated the system, distrust would provide an additional security layer, consisting in e.g. verifying that data sent by other agents make sense. Additionally, self-interest may boost competitiveness: reward is an incentive for agents to reach their full potential.

Goal divergence in benevolence

Despite agents apparently sharing the same overall goal, they may have been designed and implemented by different individuals, e.g. some agents could be part of the local rescue patrol while others belong to a from a close foreign city's patrol – which is especially likely in the Alps, as this massif overlaps several neighbouring countries. In that case, the differences in ethics or

any design-related matter may induce slight variations in goals, which could lead to different choices e.g. when it comes to choosing a victim to save over another.

Ideally, the design of such agents should be standardized and rely on centralized organizations publishing for instance avalanche statistics, which is used to optimize search algorithms. Alternatively, this divergence can be alleviated by negotiation mechanisms. Hopefully, two conflicting agents will find a common agreement to work together.

Task sharing

In this context, agents perform distributed computation to maximize processing speed. This is particularly useful for accurately localizing a victim once a victim signal is found. The main idea is to divide a task into subtasks and assign them to other UAVs. Any computation task should therefore be divisible. A contract net protocol as described above can be used along with an acknowledgement mechanism. If a contractor does not send back results within a certain time, then the manager can do the task himself or look for other contractors. Distributed computation can be made less prone to errors by asking multiple agents to do the same computation than the one done locally, then averaging the results.

If agent A asks agent B for computing assistance, A may finish its computational task before B finishes its own, in which case A has to wait for B to finish. Stall is not desirable. Solutions to this issue exist:

- Adjusting task granularity
- Letting the manager select contractors with similar computational power so that tasks get completed at the same time
- Intermediate results can be produced by delayed contractors. The manager could preempt contractors, forcing them to send a result

Hypotheses

The environment in which agents evolve is mountainous, thus subject to changes (e.g. wind, blizzard). For the sake of simplicity, it is henceforth considered to be static. In this environment, agents can make the following actions:

- Move in all space dimensions
- Position themselves in the environment
- Sense the environment, in particular the ground and potential victims
- Communicate with other agents or the rescue team

The expertise of agents is from this point forward limited to a single expertise, i.e. a single kind of victim sensor.

2.6 Networking in flying ad hoc networks

Inter-agent communications play a key role in collaboration, thus efficiency highly depends on network performance.

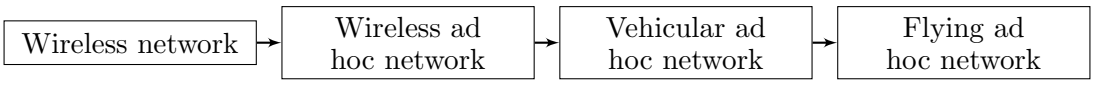


Figure 2.6: Taxonomy of flying ad hoc network, from top to bottom

Figure 2.6 details the hierarchical structure of flying ad hoc networks. A wireless network does not require wires to communicate.

A wireless ad hoc network is a mesh network in that it is decentralized. Each node participates in routing by forwarding data for other nodes. This contrasts with infrastructure-based networks and allows for more flexibility.

A vehicular ad hoc networks concerns the communication between vehicles. It is a step further in terms of mobility as topology, i.e. the arrangement of nodes in the network, becomes highly dynamic. This has severe implications on routing.

Flying ad hoc networks (FANETs) is the last iteration invented so far. Nodes can move anywhere in the three dimensions of space. This offers the ability to quickly deploy a temporary backup network e.g. in case of disasters. The advantage is that signals can efficiently propagate from one node to the next as there are no obstacles to interfere with.

2.6.1 Network properties

FANETs can exhibit the following properties:

- Adaptability [22]: adds a dynamic dimension to missions, notably it allows clients to join and leave the network. It also enables UAVs to change to a new mission on-the-fly. This is not exactly an interesting feature for the project as UAVs are only meant for a single mission: finding avalanche victims.
- Scalability: the given network functions properly whatever the value of key parameters (number of nodes, size of search area, number of victims, etc.), without any adjustments. In particular, congestion should not grow when scaling up. This property depends on the underlying communication technology and the routing protocol
- Delay-tolerant networking: suitable when nodes are intermittently connected. The store-and-forward approach consists in keeping received data and sending them at a later time to the destination. This is useful when nodes repeatedly get out of range of one another

Various metrics are used to assess network performance:

- End-to-end latency: delay between sending and final reception (destination)
- Packet delivery ratio: packet loss
- Data rate: amount of data processed per unit of time
- Jitter: measures unexpected variations in signal characteristics

2.6.2 Interference mitigation

Interference happens when two electromagnetic signals with the same or nearly the same frequency collide. Depending on the phase difference between them, the resulting signal is distorted to some extent (constructive interference) or utterly destroyed (destructive interference). As a result, data carried by the initial signals is respectively altered and lost. This is why frequency bands are carefully regulated: spectrum is a precious resource.

Interference is what limits network performance, in particular packet delivery ratio and throughput. This subject is particularly relevant in this application because interference increases with the number of nodes involving in networking activities.

Various techniques, implemented as part of a routing protocol or designed at the physical layer, exist to mitigate this phenomenon.

Channel access methods

The idea behind channel access methods is to share a transmission medium between multiple nodes. It can be seen as a link-layer protocol.

- Frequency-division multiple access (FDMA) allocates a frequency band to each node. This is not scalable, unless additional effort is made to dynamically reallocate frequency bands according to surrounding nodes
- Time-division multiple access (TDMA) allocates time slots to nodes in a cyclical way. This has the benefit of saving bandwidth but it makes impossible to exceed the time slot in case high-priority data has to be sent. TDMA is used in GSM and in Zigbee's MAC layer
- Spread-spectrum multiple access (SSMA): the transmitted signal is spread on a wider bandwidth than needed, making it so short that it is impossible to discern from noise, thereby reducing interference. Direct sequence spread spectrum and frequency-hopping are instances of SSMA. This method provides security (confidentiality and steganography) through the use of so-called spreading codes, which are pseudo-random. SSMA is implemented in 802.15.4 (Zigbee) and 802.11 (WiFi)
- Space-division multiple access (SDMA): uses smart antennas, i.e. antennas capable of changing their orientation to optimize transmission and reception. This requires to keep track of node positions. This technique can be used in this project as each UAV has to periodically send status information to other UAVs, as seen in chapter 4.

Cognitive radio

A cognitive radio is aware of its electromagnetic environment and dynamically adjusts its parameters to minimize interference with other devices in the vicinity. Parameters include power, frequency, waveform and protocol. This technique relates to link adaptation.

Antenna diversity

Antenna diversity is about using multiple antennas to transmit a signal over multiple propagation paths. The receiver needs multiple antennas as well, otherwise an interference occurs on that side.

Dirty paper coding

Dirty paper coding consists in adjusting the data to send so that the resulting signal after interference corresponds to the original signal, i.e. the unaltered data. This techniques requires the sender to know precisely the interference that is going to be applied, which is not realistic in a constantly changing and complex network such as FANETs.

Realistically, no interference mitigation technique is scalable for a spatially limited network. In other words, the only viable option to deal with a huge number of nodes is to spread the network. This is what hierarchical routing protocols are based on: nodes are grouped into hierarchical clusters geographically apart from each other. This scenario is however not likely to happen as it would take a huge amount of nodes and/or traffic.

2.6.3 Routing protocols

Routing is the process of transmitting data from one node to another using intermediary nodes. The path between two nodes chosen for the transmission is called a route. This concept is the base of the Internet. Routing is an important matter for this work because it is not guaranteed

that each UAV is within range of each other. This depends on the unpredictable size of the search area.

As most avalanches are relatively small –about 7200 m^2 according to [7]–, it may be contemplated to ignore routing considerations and use broadcasting, at the expense of losing scalability. This track is not explored any further as it is not interesting nor adaptable to big avalanches. The goal here is to highlight the various options available and discuss about the pros and cons of each one.

Taxonomy

A taxonomy of routing protocols for FANETs based on their dynamism is proposed in [18].

Static routing Routing tables are not updated during the mission. This makes it a very simple protocol with no overhead, but it is not suitable for networks where nodes constantly move relatively to one another. Depending on the collaborative search strategy explained in chapter 3, this protocol could be used. For instance, allocating search zones to UAVs would make the topology more deterministic. However, those allocations and the whole topology can be subject to changes for a lot of reasons (differences in UAV velocities, unpredictable battery discharge, UAV getting disconnected, new UAVs joining the network, one UAV switching to another area once it is done with its own, etc.): static routing is not fault tolerant

Proactive routing Routing tables are kept up-to-date at all times. This way, a route is always available and near optimum with no additional delay. The downside is its increased complexity and overhead (messages are sent just for the purpose of optimizing routing), hence power consumption increases too. Sequence numbers can be used to avoid routing loops and favour newer routes, as it is the case in destination-sequenced distance vector routing (DSDV). In that protocol, routing information gets incrementally updated to limit overhead. Optimized link state routing (OLSR) discovers state information about nodes and efficiently disseminates it throughout the network. Multipoints relays, in charge of local routing, are elected by the other nodes. This measure significantly decreases overhead

Reactive routing Routes are created on demand, i.e. when the need arises. Overhead is smaller than proactive protocols, so is memory usage, at the expense of increasing the delay to find a route. In ad hoc on-demand distance vector (AODV) routing, nodes requesting a route broadcast a route request, which is relayed until a node that knows how to reach the destination or the destination itself receives the message and replies

Hybrid routing combines the advantages of reactive and proactive protocols. This can be achieved by spatially clustering nodes into zones so that proactive routing is used within clusters and reactive routing is used for inter-cluster communicate. By doing so, routing messages get spatially bounded, therefore the network is prone to less interference. Nonetheless, nodes have to be clustered again every time the network topology changes

Geographic routing Nodes are identified by their geographic location instead of a traditional network address. The destination node corresponds to the closest node to the location of the destination. By definition, this protocol fits the network topology, so there is no need to keep routing tables up-to-date. The source however has to be aware of the location of the destination. This is realistic in the context of this work, as UAVs will receive periodic status information about other UAVs.

Flooding routing

Flooding routing, also known as epidemic, or blind routing, is another worth-mentioning routing protocol that does not fall into the above taxonomy, because it overlaps with some other

protocols. In flooding routing, every incoming packet is sent through every outgoing link except the one it arrived on. It is in that respect a kind of broadcasting but goes beyond that. Flooding stands out from traditional routing in that it is stateless, i.e. routes are not kept in memory. Actually neither the source nor the destination, nor any other node, knows the route taken by the message.

This method is very simple, does not require any network management whatsoever, and offers low latency and robustness: messages arrive from multiple paths, including the shortest one. On the downside, it uses more power, wastes bandwidth and is susceptible to cause so-called "broadcast storms" that congest the whole network, due to the exponential increase of packets with the number of nodes. Therefore, flooding is not scalable as it is.

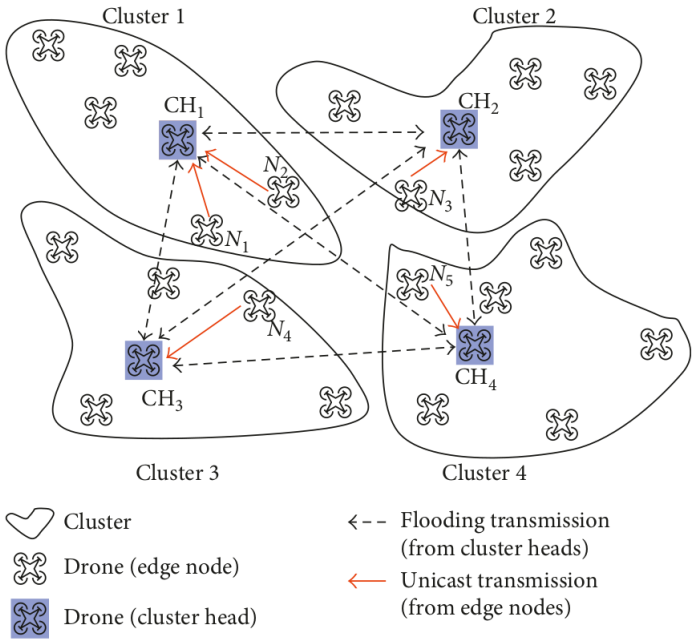


Figure 2.7: FANET based on hierarchical flooding

The context described in [24] is very similar to this one. The routing algorithm, illustrated in figure 2.7, is based on flooding methods combined with hierarchical node clustering. Each cluster contains two kinds of nodes: edge nodes and a single cluster head, appointed by other nodes depending on their capabilities (e.g. range). Edge nodes are limited to unicast transmissions to the head only, though they may indirectly communicate with nodes from other clusters. The head queues packets from edge nodes and dispatches them using flooding. Head transmissions are limited to specific time slots (cf. TDMA) to reduce collisions. It is proven in [24] that the worst-case end-to-end delay of this protocol is bounded, thanks to the time-division method.

The issue in this solution is that the more clusters there are, the shorter time slots get. At some point, it should be envisaged to assign the same time slots to several clusters and use interference mitigation techniques to maintain scalability. Adaptability is not provided out of the box either. That is another challenge that can be overcome through dynamic cluster reconfiguration.

2.6.4 Contextualization

This part discusses the design of a suitable network for this very context.

Network properties

The network should exhibit the following properties:

Real-time The goal is to localize victims as fast as possible. The end-to-end latency should thus be low and deterministic.

Adaptability It should be easy to insert or remove UAVs from the fleet. UAVs should by default collaborate (i.e. be online), but they should also be able to work offline in case the network is shut down or not accessible.

Decentralized The system should never entirely rely on a specific agent because anything could go wrong, which does not meet the principle of maximum effort. FANETs are by definition decentralized.

Packet delivery ratio Ideally this metric should be maximised.

Self-configurable and fast to deploy The SAR mission should start as soon as possible with no intervention. FANETs share those features.

Unicast An agent should be able to communicate with specific agents e.g. to negotiate.

Broadcast Agent beliefs should be sent to every agent in the vicinity along with telemetry data (e.g. battery level, status information). This way agents are aware of who is active or inactive. This makes the network adaptable. As nodes do not know the network topology beforehand, the egg and chicken problem faced during mission initialization is solved by broadcasting.

Bandwidth

Bandwidth should be carefully chosen to avoid interference with other networks, mainly mobile networks and any other network used by rescuers. Using a licensed bandwidth would reduce interference but it would require to modify some aspects of the technology used (cf. below).

Routing

Any routing protocol described above is relevant for this project. It should be chosen according to the way coverage is proceeded. For more information, see section 3.2.

The routing algorithm could be dynamically determined based on message priority. For example, high-priority messages (e.g. "victim found" message) could be sent by flooding, whereas messages with lower priority would be dispatched by a reactive protocol.

Choice of technology

Various communication technologies are described in [22] based on UAV applications. For avalanche search, Bluetooth, Zigbee, Wi-Fi, WAVE, WiMAX are potential candidates. Mobile networks are scrapped because they require a base station.

Implementation details (e.g. messages being sent) are given in chapter 4.

2.6.5 Security matters

Finding avalanche victims is a matter of life and death. It is by definition critical, hence the risk impact is big. As for the incentive, the main reason to attack the system is fun. The attacker probably do not even know who is buried under the avalanche. On another hand, controlling a few UAVs with a very limited set of sensors is not profitable to an attacker. Nevertheless, unintentional attacks may happen (e.g. indirect jamming or interference).

The following attacks can be carried out against the system:

- Eavesdropping: the attacker may intercept the victim's position, vital signs or any sensitive information
- Packet injection: the network can be taken down (denial of service) or UAVs can be mislead
- Communication jamming: UAVs cannot collaborate anymore but can still complete their task autonomously. They cannot communicate with the rescue team to pinpoint victim locations either
- GNSS jamming: UAVs are not able to fly correctly anymore, as they strongly rely on GNSS

Chapter 3

Abstract model

The process of finding avalanche victims is split into two independent tasks: coverage (sweeping the area looking for a victim signal) and victim localization (accurately finding the victim's position). Most of the work that has been done about finding avalanche victims with UAVs concerns the latter task. This split makes this work entirely compatible with the state of the art.

An effort is made to introduce concepts in a modular way. For example, some concepts are valid at the UAV level, while others are applied at the system level. This way, engineers designing the system may combine features as they see fit.

3.1 Hypotheses

For the sake of simplicity, the model elaborated in this chapter is made under several hypotheses:

- Victims are immobile. This enables UAVs to incrementally cover the area
- Collision avoidance is ruled out for the sake of simplicity
- The search area (a.k.a. the debris area) is already provided, e.g. by a satellite, and may take into account the location where victims were seen for the last time
- No assumption is made about the size of the search area
- A focus is given to ATs and similar technologies as the main victim sensor, although alternative sensors are not entirely put aside
- The terrain is supposed to be homogeneous and static, i.e. it does not change during the mission. It may be sloped
- Homogeneity of sensed data: in the same conditions –regarding agent position, victim position, etc.–, agents sense the same data, which implies they use the same kind of victim sensor. This makes it all simpler as sensed data would fluctuate depending on those parameters
- Heterogeneous fleet: UAVs do not necessarily have the same characteristics (e.g. velocity, weight, battery capacity). This allows the system to be backward and forward compatible with other multi-UAV systems, in particular with systems from other regions or countries. This is especially relevant when UAVs operate close to borders, e.g. in the Alps

3.2 Coverage problem

The essential challenge in finding avalanche victims is to efficiently cover the search area. Using multiple UAVs allows to combine the capabilities of all agents through collaboration. Coverage

is the equivalent of the basic search performed by human rescuers until the moment a victim signal is sensed. The idea behind collaborative covering is to reduce overlap –i.e. covering an area that has already been covered– through inter-agent cooperation.

This section focuses on coverage, i.e. basic search. Valuable concepts are first introduced, then a collaborative coverage scheme is proposed.

3.2.1 Optimization considerations

The system has only one true objective: saving as many lives as possible. This depends on a multitude of parameters. A non-exhaustive list of those parameters with their dependencies is proposed below:

- Survival probability as a function of time
- Time for UAVs to arrive: location of UAV station, UAV velocity
- Time for locating victims
  - Time for covering the search area: number of UAVs, number of victims, size of search area, UAV velocity, battery level
  - Time for locating one victim: algorithm efficiency, latency, battery level, etc.
- Time for rescue team to arrive (if the rescue team is far, there is no rush which could be not productive): location of rescue team station, transport used
- Time for rescue team to excavate victims: number of rescuers, number of victims, burial depth, excavation time, walking distance to victims, walking speed, search time, etc.

As can be seen, the total mission time is the most important parameter on which the system has an effect. It should be noted that some objectives diverge. For example, increasing UAV velocity may decrease the total mission time but this leads to a higher power consumption, which is not something desirable. Thus a middle ground should be found through multi-objective optimization.

3.2.2 Collaboration levels

Agents can exhibit several collaboration levels:

- None: This happens when agents get disconnect from the network and fall back to a one-man mission
- Partial collaboration: Agents communicate their beliefs but do not agree on anything (first arrived on a zone, first served). This eliminates the need for negotiation and area allocations
- Full collaboration: Agents communicate their beliefs and work together. It seems like the most efficient solution providing it is correctly implemented, however the inter-agent dependency should be limited to avoid deadlocks

3.2.3 Reality of coverage

Victims are sensed using a victim sensor. As discussed in 3.3.1, it may be an AT, a Wi-Fi access point, etc. Naturally, one would choose the value of the search strip width based on the sensing range of the victim sensor. However there is not such a thing as true sensing range: only assumptions can be made about it. This stems from an egg and chicken problem whereby the emission characteristics of a victim's data (power, antenna orientation, gain, temperature, etc.) are unknown until that victim is found. Consequently, the actual range is unknown beforehand and has to be speculated. A way of doing so is by a trial-and-error process: the search area

is covered through multiple iterations (passes), the sensing range value being reduced for each new pass. There is a trade-off: a pass with a big sensing range is fast but it makes it less likely to find victims. This is why the sensing range value has to be carefully chosen on each iteration to minimize the overall cost of the mission. The worst-case scenario, in which there are victims to be found, has to be considered by default. That being said, the mission should be stopped once a certain threshold (number of passes or strip width) is reached.

3.2.4 Mapping

A belief map represents the subjective knowledge of an agent about its environment. It is kept up-to-date by each agent locally. The goal of the belief map is to keep track of areas that have been covered, shared among agents. This is used in further steps by agents to identify uncovered areas (UA) and keep track of the location of other agents. The map is updated according to the agent's current position and the sensed data. The position can be obtained with the GNSS or via dead reckoning (although the latter option is not elaborated in this work).

There are several reasons to use a belief map:

- It makes the multi-pass technique possible. Without a map, agents would have to start all over again for each pass, or complex algorithms would be required to make a new pass without overlapping with what has already been covered. The map allows to use past contributions from previous passes
- In most real-life systems, there is a discrepancy between the input command and the output result. In particular, one cannot assume that a waypoint has been reached just because an agent has been asked to do so; feedback has to be checked otherwise the system operates blindly. This phenomenon can be caused by a lot of reasons: windwhirls, obstacles, faulty actuators, any kind of noise, path planning strategy (e.g. the maximum distance with a waypoint to consider it to be reached), etc. As a possible mitigation, intermediary waypoints could be inserted to enforce the path is followed, but this would slow down the agent –trying hard to reach an irrelevant waypoint–, hence the whole operation
- Some spots are missed even if the path is followed as expected. It happens when trying to cover a rectangular area with a circle. Results from simulation indicate that this is the main source of residues (cf. 5)
- Agents make some special moves (e.g. when an agent takes a shortcut to catch up with the others (shortcut), or navigates to a certain point to provide triangulation assistance to another agent). Those moves should be counted as part of the coverage process, though the gain may be minimal

Footprint

When covering the search area, UAVs are supposed to fly at a constant height above the ground (about 2 m) to let rescuers operate. The search area is therefore reduced to two dimensions, hence the problem is simpler to solve. Depending on the victim sensor, the sensing range projected on the ground (footprint) can be represented by:

- a circle when the victim sensor is an AT, a WiFi receiver or any other receiver able to sense signals from any direction. This comes from the fact that the reception is approximately isotropic (digital ATs are usually equipped with three antennas)
- a rectangle for ground penetrating radars, cameras, etc.

These shapes are only an approximation of the real footprint. Depending on terrain slope and burial depth, the projection of a sphere on the ground can be a circle or an ellipse, as shown in figure 3.1. Moreover, depth detection varies with the slope.

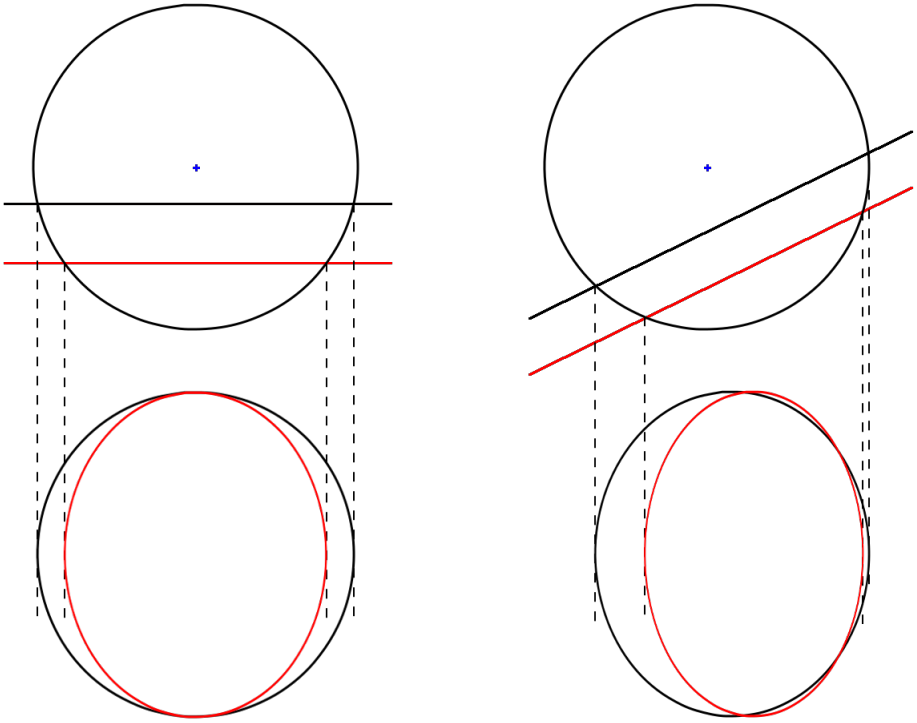


Figure 3.1: Footprint of an agent (blue cross) depending on ground level. The maximum sensing range (ground level) is represented in black. An alternative sensing range is represented in red. Top: side view of sensing range. Bottom: top view of sensing range, under the hypothesis that the terrain is flat along the contour line
. Left: flat ground. Right: sloped ground

Rangefinders or offline topographic data could be used to improve the footprint model.

Another challenge stems from two facts: 1) the map is represented on a flat surface, whereas the Earth is spherical; 2) agents position themselves using a global coordinate system. One could think that the search area is so small that it can be approximated by a flat surface. That is a fair hypothesis, though the length of a longitude degree varies considerably depending on latitude. As a consequence, a footprint would not have the same shape at different latitudes. This approximation can be kept if the footprint shape is changed accordingly. Alternatively, a map projection should be made to transform the real curvy belief map into a flat map, and vice versa.

Density function

A density function, or several, can be used as a mask on top of the map to prioritize the covering of certain areas over others. The following reasons make it particularly relevant:

- An area is never absolutely covered for various reasons. This is especially true when ATs are used, because avalanche transmitters do not emit all the time (duty cycle inferior to 0.1), hence an agent cannot be sure to detect a victim even if it is in range. The density function could in that case represent the number of times or the time spent above an area (covering weight). This brings an additional challenge as each position sent to the swarm

has to be timestamped and updated frequently. Density function could also be used to take the unexpected sensing range into consideration, as discussed above

- Knowledge about the last seen point (position where victims were seen for the last time) is useful to reduce the search space. This information can be obtained from satellite observations
- A further optimization consists in using the knowledge of the place (buildings, streets, ski resort) to determine areas with a high burial probability

Map update

Several schemes can be contemplated to keep the map up-to-date and share it with other agents. It has to be decided whether to send:

- The navigation path: It lets agents render the map themselves. This allows to adjust the map to various sensing ranges and use vectorization techniques to compress data, however storage requirements and computation increase with flight time.
- The map already rendered, i.e. a picture representing the coverage work done by agents

To reduce the amount of data to send, only what has been done since the last time agents communicated should be sent.

Synchronization between position and sensed data

Data acquisition requires time and space synchronization with positioning. If not, sensed data would be inaccurately positioned on the map. In particular, there may be a desynchronization between the data produced by the GPS and the avalanche receiver, due to the fact that those sensors produce data at a certain rate: in general, GPS calculates the position once per second, while avalanche receivers do not receive more than one victim signal per second, which gives a maximum delay of 0.5 s between those two measures. Depending on the agent velocity, this gives an error of a few meters. Fortunately, the flight controller fuses the GPS data with the measures produced by the IMU to get intermediate positions.

3.2.5 Work desynchronization

A lot of things can jeopardize the functional homogeneity of the multi-UAV system: an agent quits, goes AWOL or switches to fine search (victim localization); the planned flight gets affected by obstacles or wind whirls; UAVs have different velocities; etc. As a consequence, some agents may be done with their tasks before the others, and stalling is not desirable. The following solutions can be explored:

- Agents finishing earlier should make themselves useful. For example, they could sweep uncovered residual areas, switch to the next pass, provide full assistance (computation, triangulation) to other agents, etc. Arranging this is not trivial though.
- Adjust the workload to make agents finish at the same time. This can be done through search area splitting (3.2.7).

3.2.6 Coverage techniques (agent level)

In this report, a coverage technique refers to a way of covering an area with a single agent. The goal is to optimally cover an area, i.e. with the minimum cost, as discussed in 3.2.1. Search areas come in various shapes, generally rectangles or triangles. Coverage techniques should thus work for any shape of search area.

Formally, coverage consists in finding a set of points so that the resulting surface, made of combined footprints, covers as much of the UA as possible, under optimization constraints (cost). In that respect, the coverage technique depends greatly on the underlying sensing technology, but also on the unique specifications of each agent. For instance, agents with different weights have different inertia, hence they do not turn the same way, which should be taken into consideration. Two main techniques are explored in this work. The performance of each one is assessed in chapter 5.

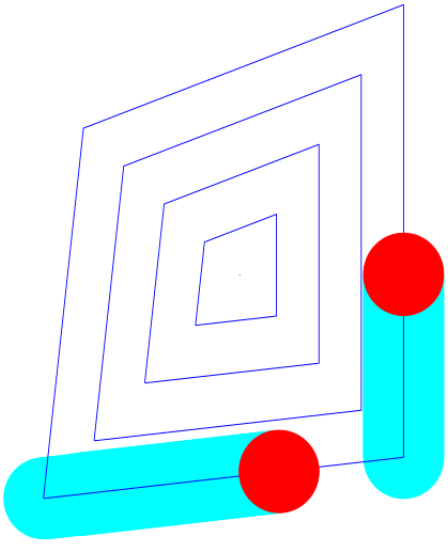


Figure 3.2: Concentric spiral coverage applied to two agents

Spiral pattern This technique has several variations. Focus is given to the concentric spiral technique described in [19] and illustrated in figure 3.2. The zone to cover can be any convex polygon. This works for any shape of search area as it can be over approximated by a convex polygon. The zone is subsequently divided into inner levels, based on the center of mass of the spiral. The footprint chosen is circular. Interestingly, this coverage technique is designed to be applied by multiple agents simultaneously.

The algorithm does the following: agents navigate counter-clockwise from one edge to the next on the same level until reaching an edge that was already visited by some agents, in which case they transition to the next inner level. The advantage of this solution is the possibility for agents to arrive on scene from anywhere. Furthermore, the completion time is proven to be deterministic, which allows to make predictions. On the downside, it requires many turns to reach the spiral center, which is a major factor impacting coverage speed.

Boustrophedon This is the same technique as the one used by human rescuers: an area is covered by means of rectangular strips. This solution is straightforward which allows to represent strips by lines, easier to encode. On the downside, strips do not fit tightly in areas that are not rectangular. A further optimization consists in determining the polygon sweep direction for which the number of turns is minimal [8].

These techniques could be supplemented by a digital twin to accurately match the specificities of each agent. Further details are discussed in 6.1.

3.2.7 Coverage schemes (multi-agent level)

The techniques mentioned above can be used as part of a wider scheme, at the system level. Two schemes are elaborated below.

Area splitting The first method consists in splitting the search area into subareas, each of one allocated to an agent. Then each agent uses basic coverage techniques to cover its subarea.

The polygon decomposition problem concerns the division of a polygon into a set of smaller polygons. The Voronoi diagram is a solution to this problem. The resulting partition is determined based on a set of initial points (sites). In [8], subareas are assigned to UAVs relatively to their individual capabilities. The elaborated algorithm supports reconfiguration, i.e. it adjusts area partitioning in case agents join or leave the system. This is however a heavy process as the map feature, described in 3.2.4, should be used to report the area that has been covered to avoid overlap.

This method has several pros. It makes the need for a map, not trivial to use and maintain up-to-date, optional –if the reconfiguration feature is dropped. Likewise, the need for collaboration, at least after the initialization during which subareas are allocated, is reduced. Furthermore, this decreases the risk of physical collisions, as UAVs stay in their allocated area.

On the downside, balancing the amount of work among UAVs during area allocation is tricky

Dynamic zone allocation The second method consists in assigning areas to agents on the fly. The multi-UAV system is arranged in a certain layout at the beginning with the UAVs available so far, then the map-based algorithm, described below, is used when desync arises. Any coverage technique discussed above can be used.

This scheme allows for agent reconfiguration as it is dynamic by definition. Additionally, desync is greatly alleviated. However, those features come at a price: it requires agents to make use of the map-based algorithm detailed in the next section. Moreover, the relative position of agents quickly gets unpredictable, which among other things makes it difficult to reason about real-time guarantees and to implement a position-based routing protocol.

3.2.8 Map-based algorithm

The goal of this algorithm is to cover, mainly collaboratively, UAs based on the information provided by the belief map. It is specifically designed to be applied to the dynamic coverage scheme, though it can be used with the area splitting scheme as well, in which case it is applied to a single UAV instead of the whole fleet. This is done in two steps: first UAs are determined, then they get assigned to agents.

Determining UAs

Depending on the way the map is implemented, finding UAs can be done based on color shades. An overview of what a map may look like is given in figure 5.3.

Some UAs are less interesting to cover than others. For example, it is less likely that small UAs contain a victim compared to bigger ones. Skipping those uninteresting UAs is a heuristic that can be applied to simplify the problem. Other heuristics can be applied through the density function (3.2.4).

Assigning UAs to agents

Techniques have been proposed to cover areas with a single agent, then schemes have been introduced to assign areas to agents first. Now, a set of stochastic UAs has to be assigned to a set of agents whose position is not known beforehand. The idea is to estimate the coverage cost of each UAs for each agent, then assign those UAs to agents so that the overall coverage cost is minimized. Coverage techniques can still be used but they are part of something bigger now:

there are many ways of covering the same UA, and many combinations of UAs. Combinatorial optimization is meant to solve this kind of problems.

The vehicle routing problem (VRP), generalization of the multiple traveler salesmen problem, can be formulated as follows: given a set of customers and multiple vehicles, the goal is to find a set of routes that passes through each customer and minimizes the global transportation cost.

As we can see, it is very similar to the coverage problem faced here. In this case, each agent is a vehicle starting from its current position, stochastic (depends on where they started covering, their speed, whether they found victims, etc.). They are not required to return to the starting point (depot): this is the open variant of VRP. Instead of customers, UAs are visited. More specifically, each UA is resolved into a set of dots, each dot corresponding to a customer. The criteria to optimize is the total cost to visit UAs, in respect to the variables that have been chosen to be optimized for the overall mission.

The VRP is widely covered by the literature. A common solution to this problem is achieved in two steps:

- Problem reduction: dots are clustered (e.g. using the K-means algorithm) to make subsequent computations converge faster. Each agent gets assigned a cluster
- Combinatorial optimization: various methods are available, such as simulated annealing, particle swarm optimization, genetic algorithms or ant colony optimization

Limits

Because the VRP is NP-hard, solving it is computationally expensive. Furthermore, it has to be run continuously to take map updates into account. Estimating the coverage cost is incredibly expensive as well, as it would require each agent to simulate the coverage process for each UA. All this increase need of computing power jeopardizes the real-time response of the system. Finally, nothing proves that the map-based algorithm would significantly increase mission efficiency. According to the results from simulation (5), residual surface may be very small (inferior to 5%).

It comes without saying that even additional challenges have to be overcome to make this algorithm work well. For example, sharing an UA among multiple agents may be envisaged to further optimize the result. It should also be thought about the way agents should agree on their beliefs –they may be different– or even on the sensing range to use.

3.3 Victim localization

Most of the related work that has been done on drone-based avalanche search focuses on the actual victim localization, the coverage technique being ignored or kept very simple (mere zig-zags, no map). This chapter is essentially a survey of victim localization techniques. The contribution here is the fact that search technologies (victim sensor) is kept separate from search algorithms.

3.3.1 Search technology

This section briefly details the state of the art of technical solutions for detecting avalanche victims. ATs are ignored as this matter has been covered in section 2.2.

RECCO©

RECCO©devices are very similar to ATs and compatible with them. The difference is that mountaineers do not need to carry any transmitter. Instead, they wear a reflector which passively reflects back signals sent by rescuers. RECCO® devices have roughly the operational range than ATs. They are more and more used as an addition to ATs.

Mobile phone

Localizing victims' mobile phones is an interesting solution because it does not require any investment from mountaineers, as almost anyone owns one. The localization can be achieved by measuring e.g. radio signal strength indication, time of arrival, time differential of arrival and angle of arrival [10]. The actual signal to triangulate depends on the underlying technology used by the mobile phone.

Cellular network There is a priori no certainty that a user equipment will spontaneously send data, especially in remote locations where connectivity is low. A solution consists in mounting a base station onboard UAVs to force cell phones to send messages, e.g. by deauthentication. This could also enable to uniquely identify each device, hence each victim, which solves the multiple burial problem.

An IMSI-catcher is a device meant to eavesdrop mobile phone traffic by a man-in-the-middle attack: it behaves as a base station and forces nearby phones to emit their unique identity (IMSI). IMSI-catchers rely on the lack of mutual authentication between users and base station, ergo they only work for GSM networks, though downgrade attacks exist to force users to use non-LTE mechanisms.

It is worth mentioning that the use of such equipment is strictly illegal, can be misused and involves technical difficulties (e.g. antenna size).

WiFi, Bluetooth Alternatively, a smartphone may host a WiFi (802.11) hotspot with a unique identifier. To save power, a background application should wait for the moment the owner gets buried to switch to send mode. Bluetooth can be used in a similar way.

Ground Penetrating Radar

A ground penetrating radar (GPR) works by emitting electromagnetic waves that get partially reflected by encountered layers [3]. The resulting signal strength depends on the variations among those layers in terms of electrical properties. There is a trade-off in the choice of the emission frequency: high frequencies yield sharper images, while low frequencies allow for a deeper ground penetration. Because it is cumbersome, GPR is usually mounted on a sledge [1].

Infrared imaging

The current state-of-the art in infrared imaging does not allow to detect victims buried beneath a substantial snow layer.

GNSS

Accessing GNSS data allows to directly get the victim's position. The challenge here is to transmit this information to rescuers. Furthermore GNSS signals are severely attenuated by snow. This solution could be used in combination with the other techniques detailed above.

3.3.2 Search algorithms

Agents may employ various search algorithms to process victim signals and determine the position of victims.

Traditional fine search (CROSS)

In [16], Politecnico di Torino researchers apply the standard avalanche search method used by human rescuers to UAVs. During GRID phase (primary search), the UAV sweeps the area using the boustrophedon technique. When a victim is sensed, it switches to CROSS phase (secondary search), consisting in finding the maximum emission point on an axis, then proceeding again

on the perpendicular axis. This technique seems to be suboptimal due to the need to get progressively closer to the victim. However, UAVs will have to get very close to the victim anyway to pinpoint the location, e.g. using paint. Moreover, the technique is reliable and compatible with ATs as well as other technologies.

Magnetic field modeling

In [15], the model of the magnetic field generated by an AT is found to be too complex to be analysed as is. An approximation function is derived using MATLAB. However, authors stress the difficulty of making an accurate model. Results yielded by that technique are inconsistent and inaccurate.

Intersection of spheres in 2D and 3D

[15] suggests another solution based on modeling sensing capabilities with a sphere whose radius corresponds to signal strength, and intersecting them. Taking many measures seems to reduce noise and provide satisfying results.

K-means

This technique is explored in [20]. Each measure is represented on a map by a dot. Dots under a certain threshold are discarded. Finally, a K-means algorithm computes the spatial average of the remaining dots. Although this algorithm is efficient, signal strength is not taken into consideration. Also, multiple burial cases bring an additional challenge, as it requires to analyse the compactness of clusters to determine the number of victims buried.

Gaussian mixture model

Each measure is expressed as a Gaussian function. The sum of each contribution gives an approximation of the signal strength sensed by agents. Once this sum is obtained, finding the victim's position boils down to finding the maximum of the resulting function. Apparently the variance value is chosen empirically [20]. In [11], the same technique is used as part of an extended Kalman filter.

This algorithm is more complex. In average, results are more accurate than the K-means technique. It may be difficult to determine victims without knowing the number of victims.

Self-organizing maps

A self-organizing map is an artificial neural network that applies competitive learning to approximate a distribution of training data: for each iteration, the nearest neuron to the training datum is selected. Consequently, it is stretched towards the training datum, as are its neighbors to a lesser extent. After a certain number of iterations, the neuron map closely approximates the data distribution. In the case of finding a victim, this technique is used to interpolate intermediate dots, i.e. intermediate measures, which fastens the process. The associated algorithm is found to be the most expensive to run, but also the one providing the most accurate results [20].

A combination of several algorithms mentioned above could be used.

3.3.3 Multiple burial case

In case of a multiple burial scenario, a choice has to be made regarding the victims to save in priority, as it takes a significant amount of time to excavate a victim. Although out of scope, this has to be thought through as UAVs may greatly help rescuers. For example, providing the the burial depth to the rescue team is fundamental to help them make a decision. Triage methods are discussed in [12].

3.3.4 Fine search algorithm

The following algorithm can be considered to perform the fine search:

- Start the search algorithm
- Stop the search algorithm when the position estimation is decent (a few meters)
- Bring one UAV (among the ones performing the fine search) closer to the estimated position to improve it further and determine the burial depth
- Send the victim location to the rescue team (high-priority message with acknowledgement) and the other agents
- Pinpoint the area (e.g. using paint) to make it easier for the rescue team to localize the position

3.3.5 Contextualization

Inter-agent collaboration in fine search is achieved by helping the agent with triangulation. The algorithms discussed above can be run in parallel by several agents. Task sharing makes victim localization faster but it is also a way to balance power usage among agents. Details about the implementation of networking are discussed in chapter 4.

Chapter 4

Implementation

The goal of making a real proof of concept of the multi-UAV system was soon abandoned in favour of simulation. There are several reasons for this choice:

- Current regulations do not allow fully autonomous UAVs
- Drones are expensive
- Delivery delays
- Sensor integration takes time
- ATs with the right specifications are not easy to get, as discussed in 4.7.3
- Simulation saves a lot of time in the development process. It allows to automatically run lots of experiments, useful to optimize many aspects of the mission

4.1 Amphibious model

Though the whole proof of concept revolves around simulation, attention is payed to the real-world system as well. Ideally, one should be able to easily switch from the simulation to the real world and vice versa. An abstract model is drawn accordingly, illustrated in figure 4.1.

The components worth focusing on are the following:

- Sensors (victim sensor, distance sensor, etc.). Flight-specific sensors (IMU) are embedded in the flight controller
- Flight controller: adjusts the UAV attitude based on sensed data and control signals from the companion computer and the human pilot
- Companion computer: adjusts the high-level behaviour of the agent (e.g. path planning, collaboration with other agents, algorithm to find victims, etc.). In simulation, it can be fully virtualized or represented by one or several processes
- Network interface: enables agents to communicate with other agents and the rescuers

4.2 Simulation ecosystem

For a multitude of reasons cited above, the simulation world was favoured over the real world. The ecosystem used throughout this project is detailed in figure 4.2.

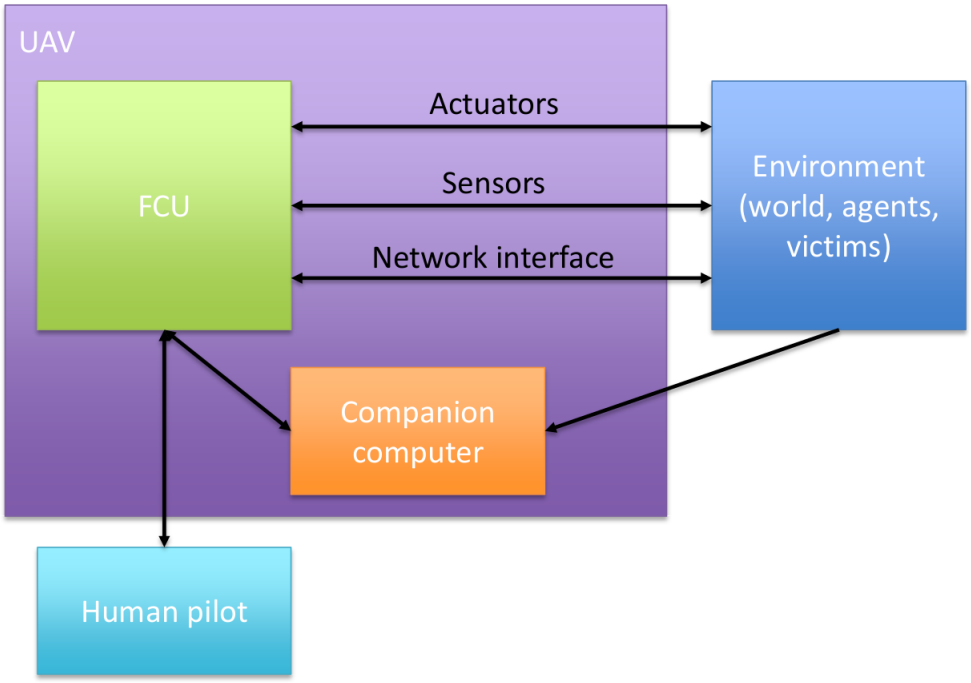


Figure 4.1: Amphibious model of the system

4.2.1 Gazebo

Gazebo is an open-source 3D simulation environment for robotics applications. It is massively used in research and in the industry. In this context, it serves as the flight dynamics model (FDM).

Gazebo supports several physics engine including the Open Dynamics Engine (ODE), Bullet, DART and Simbody. ODE, enabled by default, is a popular choice for robotic applications. ODE is limited to the simulation of rigid body dynamics, which is acceptable here. Gazebo plugins introduce additional forces such as lift and drag. For these reasons, simulation in Gazebo is considered to be realistic enough to be used as a substitute for a real-world application.

Essentially, Gazebo handles models. A model is an object with a physical reality. UAVs and the ground plane are examples of models. A model may contain submodels, as it is the case of the UAV used in this work, adapted to allow for victim sensing. Details are given in 4.4.1. Plugins or sensors may be attached to models to extend their capabilities or change their behaviour. All the models to be used in a simulation are described in an SDF file. SDF format is based on XML.

4.2.2 ArduCopter

ArduPilot is a general autopilot that can be used to control various vehicles including rotorcrafts, fixed-wing aircrafts, rovers, boats and submarines. ArduCopter is the multicopter version of ArduPilot. Although other popular solutions such as PX4 and Paparazzi exist, ArduCopter is considered to be the most advanced open-source autopilot with the most vibrant community, which makes it suitable for research. Also, it is cross-platform and compatible with most flight controllers on the market. Like other autopilots, ArduCopter offers a wide variety of flight modes including altitude hold, landing, stabilize, guided. The latter allows to navigate through predetermined waypoints. ArduCopter also offers interesting features such as terrain following –based on a distance sensor or topographic data– and collision avoidance.

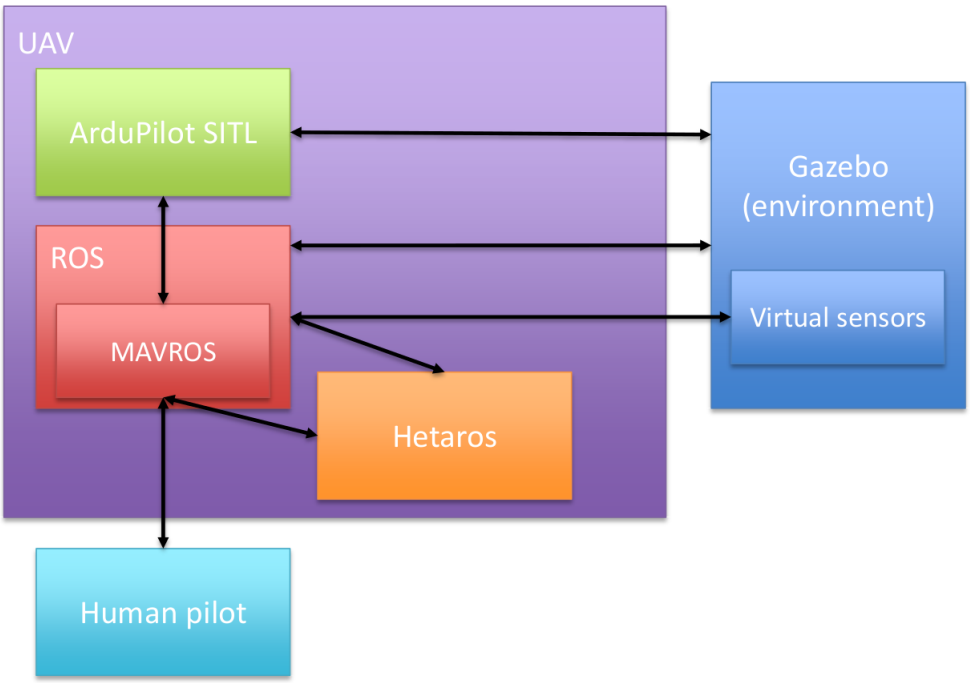


Figure 4.2: Architecture of the simulation ecosystem

4.2.3 Virtual sensors

Gazebo provides many sensors out of the box. In particular wireless transceivers can be used to generate the victim signal (transmitter) and be mounted on UAVs to sense it (receiver).

A rangefinder is a sensor that measures the distance to a target. It can be used to measure the distance with the ground and adjust the UAV's height accordingly to follow the terrain. It can be considered to use terrain data for the same purpose, however it is less accurate, especially for regions whose height constantly changes depending on snow fall, and takes up a lot of storage space, which is not a good thing in constrained embedded systems.

ArduCopter software-in-the-loop (SITL) also offers its share of virtual modules (GPS, IMU, battery).

4.2.4 Hetaros

Hetaros, developed throughout this project, is the software running on the companion computer. This is where all the magic happens to fulfill the mission of finding victims. Further details are given in 4.3.

4.2.5 ROS

ROS (Robot Operating System) is an open-source robotic framework. Despite not being a real operating system, it provides similar features like hardware abstraction, device drivers, inter-process communication and package management. It offers many sensors out of the box and allows to simulate robots in a realistic environment. In that respect, it could have been used standalone instead of Gazebo; however it does not provide all the sensors that are required for this project, hence the use of Gazebo for simulation.

ROS manages the whole simulation ecosystem. It spawns agents and runs the components of interest (cf. figure 4.3). Thanks to this hierarchical execution, terminating a simulation is made easier as it alleviates the risk of getting orphan processes.

A ROS system is made of several independent nodes that communicate using a publish/subscribe messaging model (topics). For one-shot commands or for critical communication, services can

be used. They include an acknowledgement mechanism to make sure the message has been received. In that regard, it can be seen as the ecosystem's *Yggdrasil*. Using ROS makes it easy to switch between the real world and the simulated one.

4.2.6 Network interface

Several solutions exist to simulate networking:

- Simulated sensors. Gazebo does provide wireless transceivers but it would require quite a lot of modifications to actually use them –implementing a protocol stack basically. Moreover, the propagation model is very basic and does not take natural phenomenons into considerations (interference, diffraction)
- Inter-process communication, e.g. using ROS topics and services. This is not realistic at all as there are no considerations about routing or signal propagation whatsoever: reception is immediate.
- MAVLink can be used to communicate internally (e.g. with the companion computer) as well as externally. It allows to spare a network interface
- A true network simulator would. NS-3 is used alongside Gazebo in [4]

Further details are given in section 4.3.4.

4.2.7 Interaction between components

The ecosystem's components interact with each other via miscellaneous channels (figure 4.2.

MAVLink (Micro Air Vehicle Link) is a protocol for communicating with UAVs, especially between a ground control station and a UAV, or between the subsystems of the vehicle. It is compatible with many autopilots including ArduCopter. Several implementations of this protocol exists. The one chosen for this project is MAVROS. It offers Python bindings and can be extended by plugins. In a nutshell, MAVROS exposes the autopilot through ROS topics and services, to allow any node to fetch information from the autopilot or send control commands.

The link between ROS and Gazebo is arranged by *gazebo_ros*, a ROS package translating Gazebo topics into ROS topics.

ArduCopter interacts with the other components through the following channels:

- Simulation channel: ArduCopter sends actuator commands in one way while Gazebo sends their results on the simulation environment (FDM data) in the other way. This is made possible by Ardupilot Gazebo plugin. Default ports: 9002 and 9003
- Base (telemetry, parameters and modules loaded initialization, battery status, altitude, etc.). Default port: 5760
- RC in: radio control. It concerns changes in thrust, pose, modes, camera, etc. Default port: 5501

The two last channels are part of MAVLink communications.

4.3 Hetaros

Hetaros was chosen to be written in Python to develop faster thanks to the plentiful availability of many libraries, namely OpenCV and mathematical libraries. It is also actively supported by ROS. Hetaros is an alpha version at the moment, hence a limited amount of features of the system described in chapter 3 are supported.

4.3.1 Tasks

Hetaros is in charge of many tasks that are crucial to carry out the SAR mission:

- Coverage problem solving: runs algorithms to generate a list of waypoints the UAV has to go through to cover the area, determine UAs and so on (section 3.2)
- Victim localization (section 3.3)
- Communication (with the rescue team and inter-agent)
- Logging (in a broad sense): receives external events and quickly triggers callbacks accordingly; keeps track of the agent's state for debugging purpose
- Distributed computation
- Digital twin (speculative concept so far)

4.3.2 Asynchronous programming

The program is based on the asynchronous paradigm. This makes a lot of sense in this application as agents are stimulated by external events (e.g. victim detection, messages from other agents). Moreover, several independent tasks have to be executed in parallel (e.g. communication, coverage and collaboration). It is also more appropriate for real-time applications as callbacks are triggered at once, rather than being triggered after a certain time spent in a loop as it is the case in synchronous programming. On the downside, this introduces concurrency issues that have to be carefully taken care of. In particular, ROS runs callbacks in a new thread upon message reception, which may lead to a value being written while it is being processed by another thread. Concurrency control mechanisms (e.g. mutual exclusion) can be used as mitigation.

4.3.3 Multithreading

Here is a list of threads running in Hetaros. Most of them are created by ROS when a message is received by the subscriber.

- Mission control: toggle between basic and fine search, detect the end of mission
- Update the map based on information from other agents
- Update the map based on self information
- Distributed computation: receive or send help request and results
- Send updates to a specific target (if using incremental update scheme) or everyone
- Solve the coverage problem and update the mission (waypoints) according to the coverage solution NB: Navigation is not handled in a thread. Waypoint are passed to the FCU, in charge of path planning.

4.3.4 Networking

Data is sent to agents via ROS topics and services. This solution was chosen for its ease of use, though it is not a realistic way of communicating. The following messages are defined:

- Location of victims found (high priority)
- Path history or map, depending on strategy (cf. 3.2.4)
- Secondary state updates: battery level, status (mode: basic search, fine search, out-of-order, etc.), average velocity (useful to rank UAVs in terms of velocity)

4.3.5 Using OpenCV for coverage

OpenCV is a computer vision library. It is the basis for rendering the map and getting information from it. OpenCV is meant for real-time applications.

In theory, representing the result of coverage on the map consists in drawing a series of consecutive footprints. OpenCV provides the *line()* function to draw a line between two points. The line thickness can be specified to render a contour around the line in an optimized way: instead of rendering a huge amount of circles –one at each intermediate point –, one circle is drawn at each extremity, then a rectangle is drawn between them. This algorithm should be slightly adjusted depending on the footprint shape.

4.4 Setting up and using the ecosystem

The simulation ecosystem was deployed on Debian 10 (Buster). Because all components were not packaged for this version (namely ROS), they were all compiled from source. This brings additional benefits as it allows to modify core features (cf. below). ROS was by far the most tedious component to build from source, as it includes almost 200 packages, each of one with dependencies that are not clearly specified.

At system level, simulation can be:

- Distributed, one processor (or machine) is assigned to each agent
- Centralized: a limited amount of resources (bunch of processes, ports, etc.) on a single machine is assigned to each agent. This was the solution chosen as it is easier to implement

4.4.1 Tailoring the ecosystem

The whole ecosystem does not meet the project's expectations as is, therefore some modifications and some development have been made to tailor it to the project:

- Gazebo's wireless transmitters have been adjusted to broadcast an ESSID that depends on the victim's identifier. For example, the AT belonging to *victim0* broadcasts "*victim0*". To avoid forking Gazebo, plugins can be developed to bring modifications to some aspects of the simulation, however plugins are limited to some basic hooks (e.g. sensor initialization)
- Ardupilot Gazebo plugin was patched to support multiple UAVs by changing listening ports according to agent identifiers. It was also modified to send rangefinder data
- Gazebo was modified to allow for "ghost collisions": UAVs do not have to bother about collisions with other UAVs because they are now able to go through each other with no collisions (noclip mode)
- A bridge has been made to expose additional Gazebo topics through ROS
- Models for ATs have been created based on wireless transceivers. An AT has been mounted on each UAV (nested model)
- A rangefinder has been mounted on UAVs. They allow UAVs to follow the terrain at a constant height. The number of vertical and horizontal rays as well as aperture should be chosen so that the ground is sensed whatever the position of the UAV. This sensor is showed used by a UAV in figure 5.2
- A realistic terrain has been generated from scratch through the following steps:
  - Draw a gray-scale gradient. Each shade corresponds to a certain height

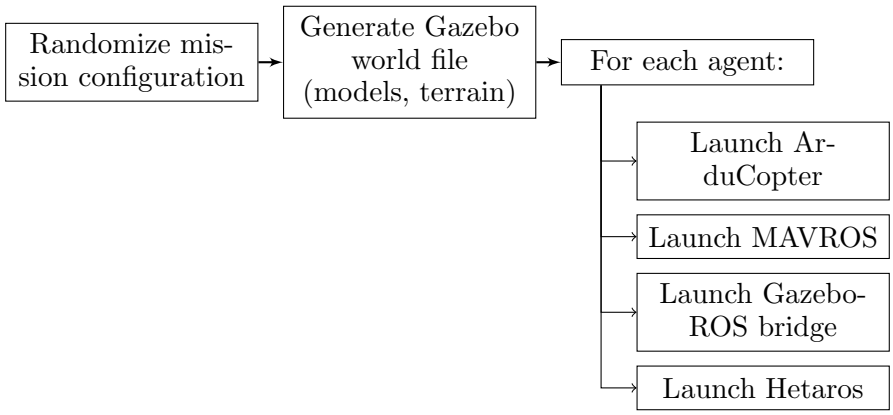
- Change the coordinate reference system to ESPG4258/ETRS89 using *qgis* tool
- Change the map dimensions using *gdalwarp*

The terrain generated exhibits a slope, bumps and ground textures, as shown in figure 5.1

4.4.2 Exploitation scripts

One of the main interests in simulation is the ability to automatically run experiments. This is what the exploitation scripts are meant to do. The goal here is to automatically generate a random mission configuration and launch the ecosystem components accordingly using ROS. Figure 4.3 shows the execution tree.

Figure 4.3: Simulation execution tree



The following parameters can be set by the user or randomized:

- number of agents
- number of victims
- position of agents
- position of victims
- size of search area

For the sake of convenience, each agent is spawned in a *screen* instance. This allows to monitor logs and potentially interact with the system. Hetaros is informed of the number of buried victims right from the start to avoid wasting simulation time. When all victims are localized or when the "deadline" is exceeded, Hetaros kills the experiment and the results (logs, mission time, battery, distance etc.) are displayed.

4.5 Instrumentation

Several tricks can be used to gain productivity in the simulation environment:

- Running Gazebo headless (server): besides debugging, there is no reason to render the simulation
- Making UAV faster between waypoints. This is done by adjusting some MAVLink parameters related to waypoint navigation

- Changing the simulation speed. Beware: if the simulation is too fast, the companion computer may not react fast enough to changes, which may yields skewed results. Solutions should be investigated to make sure the companion computer and the simulator stay in sync. However, it was shown empirically that simulations are far from being executed fast enough for this issue to arise. When simulating 10 UAVs on a high-end machine (an i7-7700 CPU with 32GB RAM), a significant amount of MAVLink messages are timed out and the real-time factor (simulation speed versus real time) is inferior to 0.5
- Pausing/resuming the simulation. Some topics exist for this very purpose
- Running simultaneous simulations. Gazebo makes use of a single thread to compute physics, which becomes a bottleneck when the number of agents grows. To run multiple simulations in parallel, one should avoid collisions between simulations e.g. by adjusting ports and ROS namespaces.

4.6 Issues

Various issues and challenges arose during the development phase.

By default ArduCopter does not allow to change waypoints while all current waypoints have not been reached. This is an issue as they should be updated whenever the companion computer sees fit. A workaround consists in clearing waypoints, sending a new list of waypoints then switching to a mode (loiter or stabilize) then back to auto mode. This mode reset forces the autopilot to restart the waypoint mission. However, this is a dangerous procedure that makes crashes more likely. Properer solutions exist:

- Patching ArduCopter to natively support waypoint injection
- Using setpoints instead of waypoints, in which case all the benefits of path planning (waypoint stacking, radius to assess whether a waypoint has been reached or not, etc.) implemented in the waypoint feature are lost

As stated previously, one of the few project requirements is to allow a human pilot to take over the vehicle, ideally seamlessly, without interfering with the mission. This is achieved by isolating human control from the companion computer's control. Several solutions exist:

- Waypoint-based: the companion computer controls the UAV through waypoints. To take over, the human pilot can switch to any mode, although he has to switch back to guided mode when he is done
- Mode-based: create a new mode to be exclusively used by the companion computer, e.g. "search victims" or "AI" mode. If this mode is enabled, computer controls commands can be executed; otherwise the human leads. This is not straightforward to use as the autopilot has to be forked. Also, command authors have to be identified, e.g. using component IDs or a flag

An issue was identified when spawning UAVs in Gazebo using ROS. They are somehow not initialized properly, leading to crash within seconds after takeoff. This was worked around by dynamically generating a Gazebo world file for each simulation instance.

4.7 Discussion on the real-world system

Altough a full simulation-based solution has been chosen, some investigation work has been done on the real-world system. Due to the possibility of designing this "real" multi-UAV system later on, it is useful to present this information here.

4.7.1 Drone choice

A drone for research purposes should be as open as possible: it should be easy to replace components, in particular the communication module; the autopilot should be open source to allow for source code modifications; every aspect of the overall system should be accessible for instrumentation (e.g. sensors data). Regarding hardware features, a WiFi access point should be mounted on the vehicle to make debugging easier. The UAV should also be equipped with a GNSS and potentially some computer vision sensors for indoor navigation or collision avoidance. In that respect, Intel Aero RTF was chosen for this project.

4.7.2 Networking

WLAN networks based on IEEE 802.11 have two types of basic services sets (BSS): independent and infrastructure BSS. The former does not need for a central access point, which makes it suitable for a wireless ad hoc network. Moreover, 802.11 includes counter-interference mechanisms as discussed in 2.6.2.

4.7.3 Avalanche receiver

An avalanche receiver has to be mounted and integrated in UAVs to enable them to sense victims. To achieve that, it should output raw data (signal strength of each antenna) through any kind of hardware interface. Some companies may exceptionally offer such products:

- Mammut: Barryvox VS 2000 PRO EXT [16], PULSE Barryvox [15]
- Pieps [20]

Most of the literature on the subject seems to agree that analogue receivers should be preferred over digital receivers. According to [16], analogue receivers offer a better sensitivity with no signal processing delay, and getting access to the processing algorithms of a digital receiver is not guaranteed. In [15], it is found that the design of digital ATs on the market is based on the assumption that it is to be used by humans, which makes them unsuitable for a use onboard UAVs.

As an alternative, it should be envisaged to design avalanche receivers from scratch. Schematics of a homemade avalanche receiver are designed in [11]. A prototype is developed in [2].

Chapter 5

Results

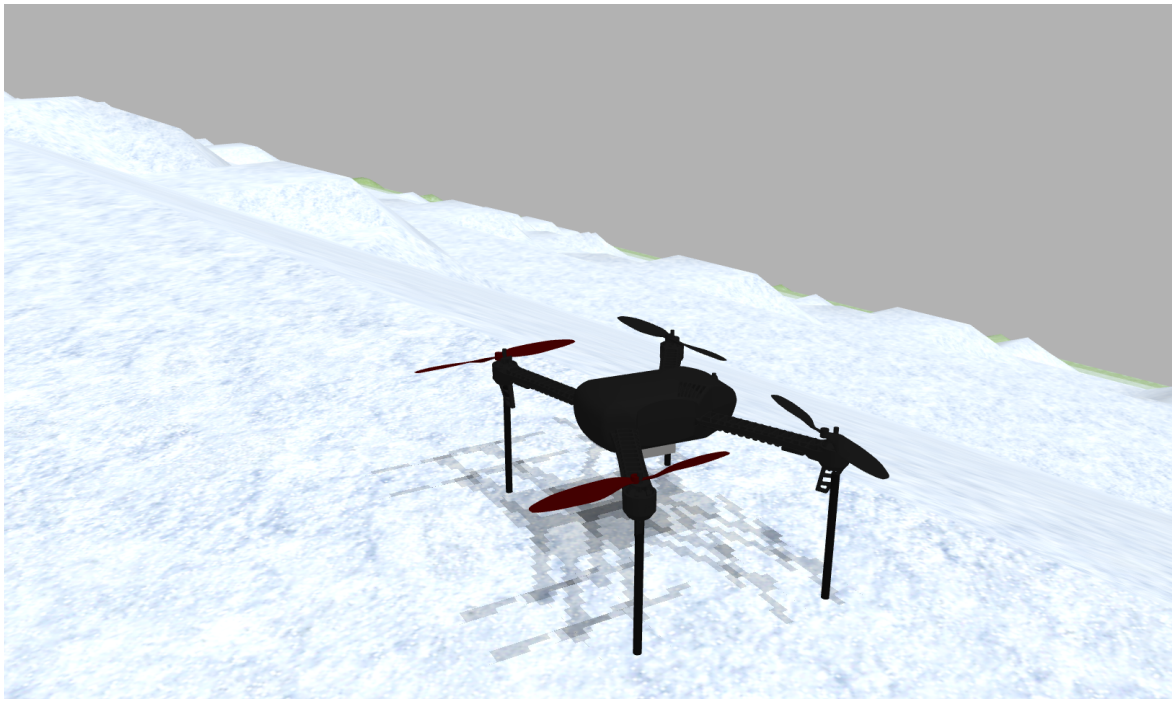


Figure 5.1: UAV made throughout the project, lying on a sloped terrain generated from scratch

This work answers the question: How to make a multi-UAV system for avalanche search and what challenges rise and have to be overcome? In that regard, the results are the actual abstract system, the simulation ecosystem and all the underlying concepts introduced above.

Illustrations of the simulation are given below. Figure 5.1 shows the UAV tailored for this project. The model is based on an Iris copter. The use of the rangefinder is illustrated in figure 5.2. The sensing range of each ray is represented in blue.

However, the relevance of the concepts introduced throughout this master thesis have to be assessed. Two coverage techniques (agent level) are explored: concentric spiral and boustrophedon. Due to the short amount of time allocated to this project, the collaborative AI (Hetaros) was partially implemented. As a consequence, some concepts (namely the map mechanism) were not put to the test.

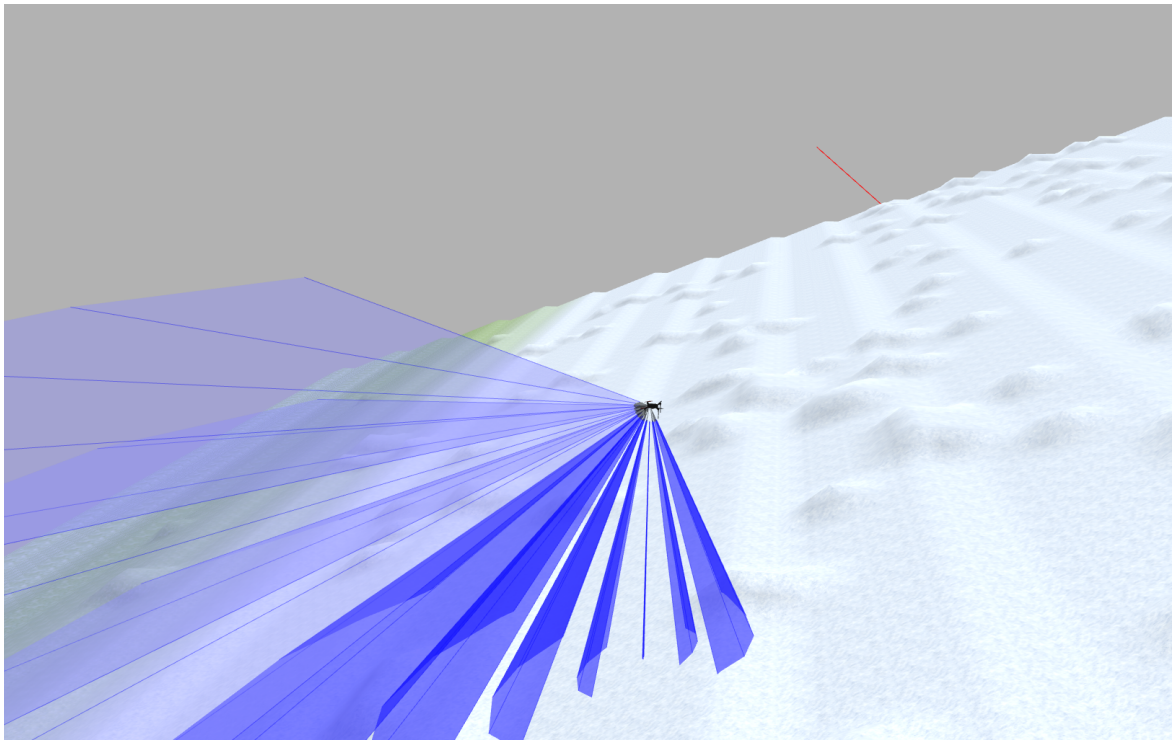


Figure 5.2: UAV navigating using a rangefinder

5.1 Simulation setup

Simulations are run on a high-end machine (i7-7700 CPU, 32GB RAM) running Debian 10 (Buster). To make things easier, the 0.002°x 0x002°(about 220x220 meters) search area is a square, though the algorithm for area splitting adjusts to any shape. A single UAV with the following specifications is simulated:

- Mass: 2.1kg
- Navigation speed (*WPNAV_SPEED* MAVLink parameter): 10 m/s
- Navigation acceleration (*WPNAV_ACCEL*): 2 m/s^2

The UAV starts at the center of the map (0°longitude, 0°latitude). Simulations are run with different sensing range values. It should be noted that this is not equivalent to shrinking or expanding the search area: with a bigger search area, UAVs would be faster, as it takes time to reach maximum velocity.

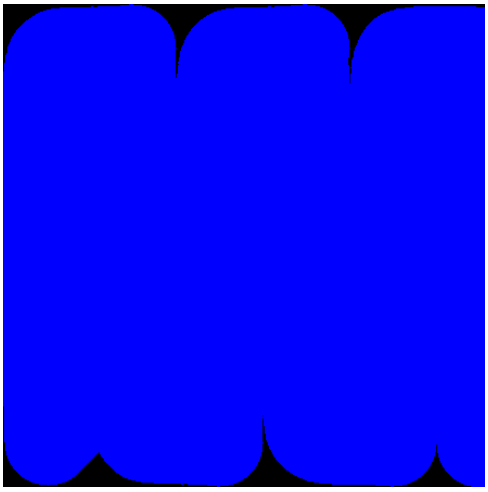
What is measured is the mission time, starting when the first waypoint is reached and stopping when the last one is reached.

5.2 Boustrophedon

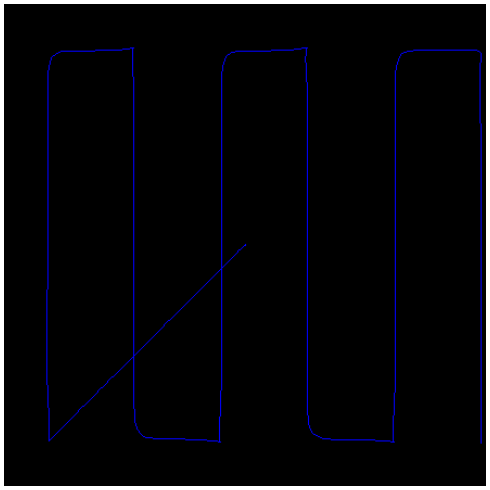
5.3 Concentric spiral

5.4 Comparison

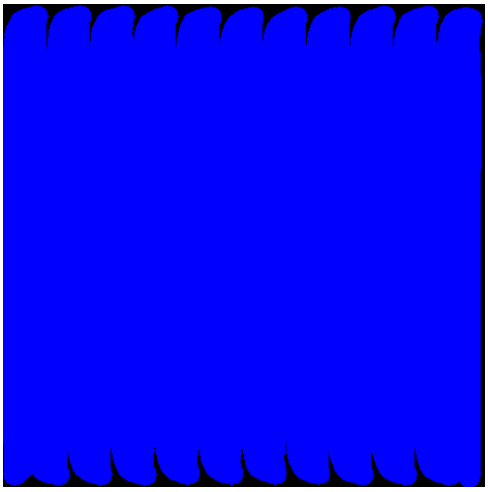
The two following tables compare both techniques, for different sensing range values:



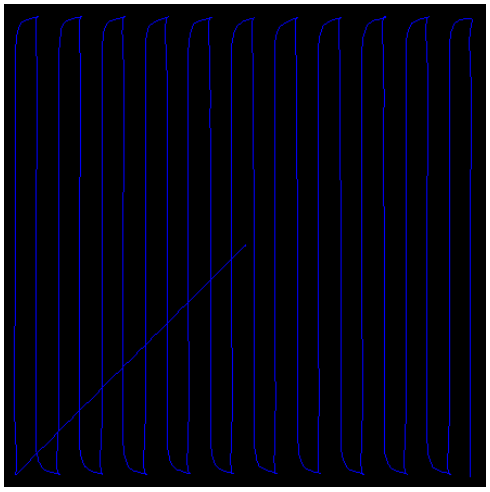
(a) Coverage, sensing range = 20 m



(b) Navigation path, sensing range = 20 m



(c) Coverage, sensing range = 5 m



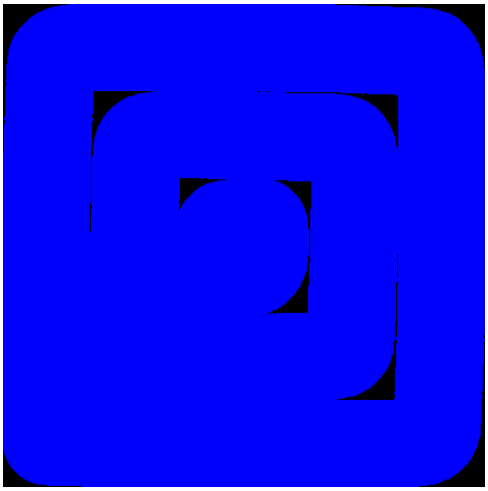
(d) Navigation path, sensing range = 5 m

Figure 5.3: Boustrophedon coverage of a square area

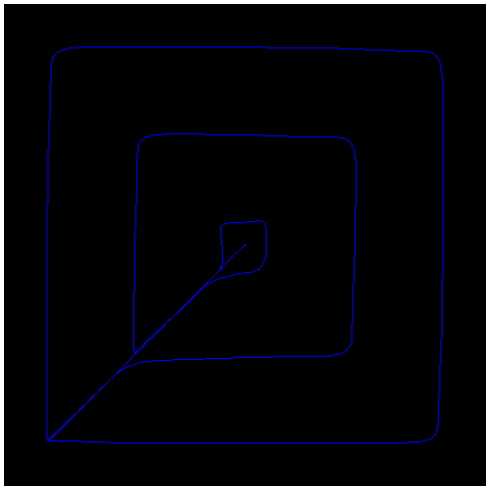
Sensing range = 5		
Technique	Coverage ratio	Duration
Boustrophedon	95.78%	624.04 s
Concentric spiral	97.86%	656.37 s
Sensing range = 20		
Technique	Coverage ratio	Duration
Boustrophedon	96.05%	169.67 s
Concentric spiral	96.20%	173.89 s

5.5 Analysis

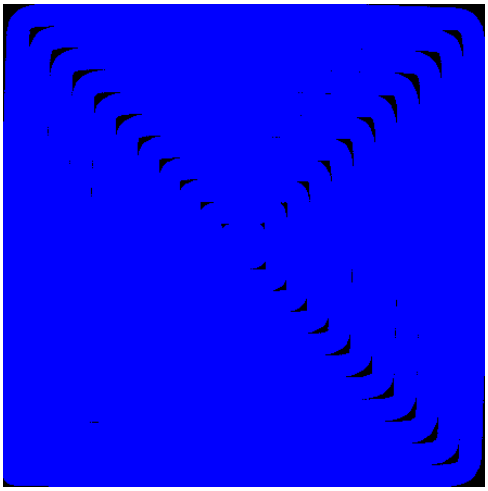
Above tables show that both techniques have a similar efficiency. Boustrophedon is slightly faster but covers a smaller amount of terrain. Concentric spiral, though forcing UAVs to repeatedly cover the same diagonal, is therefore a fit solution, not only in scenarios where multiple UAVs share the same area, but also when a single one performs. One should nonetheless be critical toward these results, as the square shape of the search area is highly unrealistic and favours the boustrophedon technique.



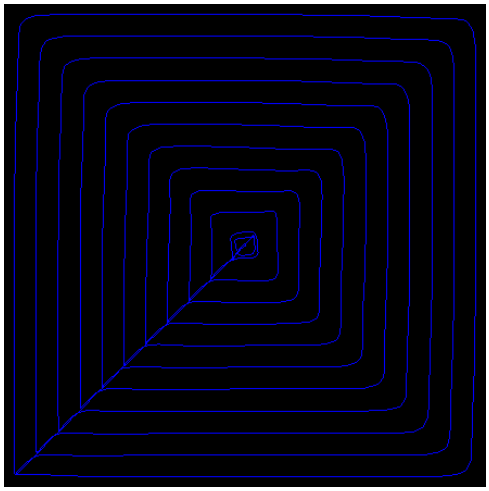
(a) Coverage, sensing range = 20 m



(b) Navigation path, sensing range = 20 m



(c) Coverage, sensing range = 5 m



(d) Navigation path, sensing range = 5 m

Figure 5.4: Concentric spiral coverage of a square area

In general, UAV navigation is found to be relatively accurate. Waypoints are reached with high accuracy though inertia makes drones slightly drift away from their objectives. It should be kept in mind that it is made possible by the simulator which communicates location data as accurate as it can get.

The biggest residual UAs therefore do not stem from disturbance on the feedback loop, but from the fact of sweeping a rectangular area with a circle-shaped footprint. Heuristics can be made out of this observation. In the case of boustrophedon, residues are located near each turn on the border of the area (figure 5.3). In concentric spiral, residues lie in the corners of each inner level (figure 5.4). In both cases, residues are aligned, making them easy to collect. Assuming residues can be covered in a single pass, the minimum distance to cover them is determined below:

- Boustrophedon: $min\_distance = 3 \times side$
- Concentric spiral: $min\_distance = \frac{3}{2} \times diagonal = \frac{3}{2} \times \sqrt{side^2 + side^2} \approx 3.1 \times side$

5.5.1 Density function

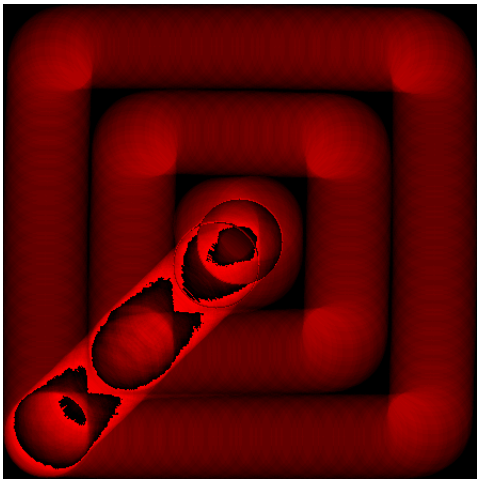


Figure 5.5: Representation of concentric spiral coverage with a density function. The peculiar motif along the diagonal is a glitch

The amount of time spent over an area can be depicted with a density function, explained in 3.2.4. It allows to select UAs with priority in mind, but it is also a tool to measure performance. As shown in figure 5.5, the locations where UAVs slow down are highlighted. As expected, slowdowns happen when UAVs turn. Density function can hence be used to determine the most profitable phases of coverage.

5.5.2 Coverage ratio

The coverage ratio metric was used to compare both techniques. to compare the performance of boustrophedon and concentric spiral covering techniques. Although it gives a good overview of the coverage performance when coupled with mission duration, it is a mere average and does not tell how well UAVs perform at a specific moment of the mission. To do so, coverage ratio has to be computed whenever a footprint is rendered.

5.5.3 Expectations

Coverage schemes, that is coverage at the MAS level, were not explored in this work. It can be expected that the mission performance saturates when the number of agents grows for the same size of search area. Subareas being smaller, turns would impact UAV speed more significantly. Moreover, this would lead to more overlap and more potential collisions. Finally, having a big number of nodes communicating in a confined space leads to interference and congestion.

Chapter 6

Conclusion

The contribution of this work is materialized by a multi-UAV system simulated in an exhaustive simulation ecosystem. Specifications on agents are flexible to allow for many possible applications: UAVs can be of any type, can join or leave at any time, etc. Moreover, it has been thought to allow to switch from simulation to the real world seamlessly.

Should the system described in this report be built, it could be used for SAR operations (e.g. earthquake), but more generally to find any object or being as long as it emits some kind of signal and does not move. Examples of applications include archaeology, finding a wounded animal or flight recorders after a crash.

This work arises ethical side effects of my research. In multiple burial cases, it may encourage rescuers to triage victims, i.e. comparing their vitals and deciding which one to save first. Moreover, it may diminish the fear mountaineers feel for avalanches, which may stimulate their desire to take part in even more dangerous mountain activities than before.

6.1 Future work

Due to the highly interdisciplinary nature of this topic, there is a tremendous amount of further tracks to investigate.

Hetaros The implementation of Hetaros was the bottleneck of this project. It is important to get it done to estimate the relevance of mechanisms described in chapter 3.

Real-world system A study on how the system can be brought to real life has to be done. As discussed earlier, a homemade antenna can be made to improve sensing capabilities. To improve mission performance, it should be possible to access detailed statistics about avalanche victims, e.g. the average burial depth, the number of victims, excavation time, etc. This way, agents would have more visibility to make choices (e.g. keep covering versus helping another agent with triangulation). On another note, it should be thought about deploying UAV stations spread over the mountains to make UAVs arrive sooner on scene. As for the right to fly drone swarm, one will have to wait for the law to get more specific.

Simulation Simulation can be significantly enhanced to make it more realistic. A network simulator can be combined with the physics simulator to model signal propagation, depending on antennas, snow depth, temperature and so on. The terrain can be enriched by obstacles (trees, rocks). Any kind of noise can be applied to sensors. Wind can be introduced as environment disturbance. Wind and temperature can be modeled to introduce noise.

Optimization Many aspects of the mission can be refined. Some coverage technique have not been investigated (e.g. zig-zagging). Depending on the specification of each UAV, some techniques may fit well certain UAVs, while being unsuitable for others. For example, a heavy

UAV has more inertia, which changes its dynamics and e.g. reduces its ability to perform narrow turns. Consequently, it could perform large turns then go back in a second phase to cover the surface left because of those larger turns. In that respect, a digital twin can be used to optimize path planning and missions in general. Each agent would create a virtual model of itself based on its history, constantly fed with new knowledge along the missions. Many aspects could be monitored, such as the average velocity, the performance in turns, etc. This knowledge should be combined with the knowledge that is acquired by the agent during the mission, so that its specific conditions (e.g. weather) are taken into account. Digital twins could therefore adapt the mission of any UAV in a dynamic way (no hardwired models). They could adjust to any context and event. For instance, should a rotor get broken during the mission, digital twins could react by adjusting their prediction of the cost to cover an area, in terms of time or power. This technology would obviously use magnitudes more processing time and storage.

The coverage problem was reduced to two dimensions, which makes it easier but less accurate, due to the fact that UAVs have a hard time keeping the relative height with the ground steady. Solving the coverage problem in three dimensions should be investigated.

Secondary optimizations Many optimization questions arise:

- How many agents should be allocated to a SAR mission? It should be considered that it will take a certain amount of time for the rescue team to arrive on scene; ideally, rescuers should never wait for UAVs to spot victims. The answer depends on many parameters including the size of the search area, burial depth, the search strip width, the distance between the starting point of UAVs and the avalanche, etc. A more general problem consists in determining the optimal size of a MAS if there were multiple avalanches throughout the country and not enough UAVs to cover all of them optimally. Avalanches could be triaged depending on the potential amount of victims, the distance with the closest UAVs, the battery level of agents, etc.

- How fast should UAVs move? This should depend on the power strategy and the inherent limitations of victim sensors (e.g. ATs emit periodically), as going too fast may make agents miss victim signals. If there is still ample time for agents to find victims before rescuers arrive, agents should prioritize power efficiency over speed. Otherwise, they should adopt a best-effort approach and locate the victims as quickly as possible whatever the power consumption

- Is it better to spread drone teams all over the mountains or single drones?

- Should agents wander in the mountain or wait at their station? These questions can be answered using numerical optimization methods over many simulations. Monte-Carlo experiments can be considered

In this work, focus has been given to agents exhibiting the same type of victim sensor. It would be interesting to add support for heterogeneous UAVs. Each agent would have to be aware of other agents' capabilities to solve the collaborative coverage problem. Agents with different sensors could collaborate simultaneously thanks to sensor fusion, which would also make sensed data more homogeneous. Heterogeneity in sensed data comes from the fact that an agent may get different data at the same location, depending on its orientation or its vertical oscillation; also, two agents may end up with different data at the same location, due to different sensor calibrations. Combining this data through sensor fusion would improve consistency. On another hand, an altitude controller can be implemented to keep the distance with the ground as steady as possible.

One of the hypotheses under which this work has been done is the fact that the search area is provided from the start. Alternatively, it could be detected by agents using various techniques involving computer vision, optical flow and surface mapping. In the same vein, GNSS should be supplemented with dead reckoning techniques, as agents should be able to fly

with no GNSS to some extent. This is an important aspect as GNSS accuracy decreases in mountain environments due to multipath.

Collision avoidance is a topic by itself and was therefore scrapped from this project from the beginning. This challenge was circumvented in the simulation thanks to the "ghost collision" trick. It is major that agents in the real-world application exhibit this feature. Collision avoidance commonly relies on two components: collision detection using sensors or algorithms, and a protocol to decide what the agents involved should do to prevent it. Collision avoidance in this case should aim at disturbing as little as possible the mission.

Fault tolerance and security Mountain environment can be harsh for agents (freezing cold, wind whirls, blizzard, darkness, etc.). UAVs should be designed to be resilient in this environment. For example, anti-ice mechanisms could be integrated to them. Effort should be invested in UAV recovery from crash. Because snow is soft enough to make a crash harmless, UAVs should try hard to get back in the air after a crash. There are either way no benefits in keeping a UAV on the ground. On another hand, some issues related to cybersecurity threats have been pointed out. Further work could investigate ways for agents to apply critical thinking to data from other agents.

Finally, avalanche is a mysterious phenomenon lacking scientific explanations. It is complex to model due to the dual behaviour of snow: liquid and solid. Realistic simulations could allow to determine zones where avalanches are the most likely to get triggered, or zones in the debris area where it is the most likely to find victims.

Chapter 7

Appendices

List of Figures

Bibliography

[1] Andreas Albrigtsen. The application of unmanned aerial vehicles for snow avalanche search and rescue, 2016.

[2] Benjamin D. Dickensheets. AvaDrone: An Autonomous Drone for Avalanche Victim Recovery, 2015.

[3] Christian Jaedicke. Avalanche victim search by ground penetrating radar.

[4] Fabio D'Urso, Corrado Santoro, and Federico Fausto Santoro. Integrating Heterogeneous Tools for Physical Simulation of multi-Unmanned Aerial Vehicles.

[5] Swiss Federal Institute for Snow and Avalanche Research (SLF). Avalanche sizes. `https://www.slf.ch/en/avalanche-bulletin-and-snow-situation/about-the-avalanche-bulletin/avalanche-sizes.html`.

[6] Swiss Federal Institute for Snow and Avalanche Research (SLF). Long-term statistics. `https://www.slf.ch/en/avalanches/destructive-avalanches-and-avalanche-accidents/long-term-statistics.html`.

[7] Ingrid Reiweger, Manuel Genswein, Peter Paal, and Jürg Schweizer. A concept for optimizing avalanche rescue strategies using a Monte Carlo simulation approach, 2016.

[8] Ivan Maza and Anibal Ollero. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms.

[9] John Hereford and Bruce Edgerly. 457 KHz electromagenetism and the future of avalanche transceivers.

[10] Jonas Ekskog and Jacob Sundqvist. Victim Localization using RF-signals and Multiple Agents in Search & Rescue, 2015.

[11] Lars Kambe Fjæra. Optimal Avalanche Search using Drones, 2018.

[12] Lee B.Bogle, Jeff J. Boyd, and Kyle A. McLaughlin. Triaging Multiple Victims in an Avalanche Setting: The Avalanche Survival Optimizing Rescue Triage Algorithmic Approach, 2010.

[13] Manuel Genswein and Jürg Schweizer. Numerical simulation of the survival chance optimized search width, 2008.

[14] Manuel Genswein and Stephan Harvey. Statistical analyses on multiple burial situations and search strategies for multiple burials, 2002.

[15] Manuel Grauwiler and Luc Oth. Fully Autonomous Search for Avalanche Victims Using an MAV, 2010.

[16] Mario Silvagni and Andrea Tonoli and Enrico Zenerino and Marcello Chiaberge. Multipurpose UAV for search and rescue operations in mountain avalanche events, 2017.

[17] Michael Wooldridge. An Introduction to Multi-agent Systems, 2002.

[18] Muhammad Asghar Khan, Alamgir Safi, Ijaz Mansoor Qureshi, and Inam Ullah Khan. Flying Ad-Hoc Networks (FANETs): A Review of Communication architectures, and Routing protocols, 2017.

[19] Peter F. Hokayem, Dušan Stipanović, and Mark W. Spong. Dynamic Coverage Control with Limited Communication, 2007.

[20] Philipp Brandtner. Embedded Neuromorphic Quadrotor Position Control for Rescue Missions, 2016.

[21] Reid G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, 1980.

[22] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint, 2016.

[23] Scott E. McIntosh, MD; Colin K. Grissom, MD; Christopher R. Olivares, MD; Han S. Kim, PhD, MSPH; Bruce Tremper, MS. Cause of Death in Avalanche Fatalities, 2007.

[24] Taemin Ahn, Jihoon Seok, Inbok Lee, and Junghee Han. Reliable Flying IoT Networks for UAV Disaster Rescue Operations, 2017.