Exploring Underground Environments by Deploying and Reconfiguring a Chain of Visually Connected UAVs

Mathis Fleuriel, Elena Vanneaux, David Filliat, Olivier Simonin

Exploring Underground Environments by Deploying and Reconfiguring a Chain of Visually Connected UAVs

Mathis Fleuriel[1], Elena Vanneaux[2], David Filliat[2], Olivier Simonin[1]

***Abstract*— This article focuses on the autonomous exploration of unknown underground environments using a swarm of drones. In such settings, conventional wireless communication is often unavailable; therefore, we deploy a chain of visually connected UAVs that maintain and reorganize themselves using only local vision-based communications. The proposed algorithm is decentralized and based on two distinct roles of the agents: the 'leader" performs the exploration, and the 'followers' preserve connectivity between the Leader and an operator at the entrance. We evaluate the proposed approach across various scenarios using a realistic simulation tool that incorporates a UAV dynamics model.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly used in critical applications, such as search and rescue (SAR) operations in hazardous environments [9]. However, deploying UAVs in complex settings, particularly underground, such as in mines [13], subways, and caves [15] remains a challenge [3], [7]. Subterranean worlds present a set of properties that complicates robotic autonomy: they are often extremely long and large-scale, particularly narrow and confined, multi-branched with loops [5], self-similar, sensing degraded, and communications-deprived [14].

We propose configuring miniature autonomous drones into a chain, so that UAVs act as relays between an exploration drone (called leader) and an operator at the entrance. Flying autonomously, but keeping visual contact with its follower, each drone can estimate its relative location and communicate information to the base from drone to drone through visual signals. This approach does not need any localization system or mapping (e.g. a visual SLAM algorithm) and, as such, (1) it naturally adapts to the material and topography of the cave and (2) it works on miniature platforms with limited memory and computational power (3) it does not require establishing a wireless network which can be tricky in the underground environments [14]. The deployment of the chain is based on a leader-follower formation approach [16], while the agents use LiDAR sensors to estimate obstacles and the positions of their neighbors.

The main contributions of this work are the following:

- Extension of the idea of deploying a mini-drones chain to any indoor environment (introduced, e.g., in [12]).
- Proposal of a multi-agent algorithm ensuring complete exploration of an unknown tunnel-like environment.
- Evaluation of key functionalities in a realistic simulator.

[1]INSA Lyon, Inria, CITI lab., 69621 Villeurbanne, France. Email: mathis.fleuriel@insa-lyon.fr

[2]ENSTA Paris, U2IS Lab., 91120 Palaiseau, France.

The paper begins with a state-of-the-art review in Section II on the exploration of unknown environments by a fleet of drones. We then present the addressed problem and the considered assumptions in Section III. Section IV details our approach, first considering exploration by a single agent before describing the strategy and behaviors for deploying a chain of agents. We introduce the simulator and the initial experimental results in Section V before concluding in Section VI.

II. Related works

The exploration of cave environments using autonomous systems has recently gained significant interest from the scientific community. Accelerated by the kickoff of the DARPA Subterranean Challenge, research efforts around the world have been investigating the particular challenges of long-term underground autonomy [15], [3], [14], [11]. Many teams relied on ground robots [3], [14] to explore the caves. However, deploying these robots in complex environments—often vast, dark, slippery, and particularly narrow—poses significant challenges. In contrast, small UAVs offer a more adaptable and effective alternative [15], [7].

Unlike ground robots, miniature flying robots cannot carry a cable to stay connected with the operator. Instead, they must rely on a radio link, which functions well in open-air environments but poses significant challenges in subterranean settings. Underground, radio communication is only effective when there is a clear, unobstructed path between the emitter and the receiver. At each corridor turn, the signal is severely weakened or lost. To address this issue, the authors of [14], [15] proposed establishing a static mesh network to connect the robots with the base station. This network consists of the base station node, robot nodes, and droppable communication WiFi nodes carried by the drones [14], [15].

In [12], the authors propose deploying a chain of drones, where each drone serves as a relay between an exploration drone (referred to as the leader) and an operator at the entrance. This approach establishes a more flexible, dynamically changing network, enabling communication through visual signals instead of relying on wireless technologies. However, the proposed method is limited to a single tunnel exploration, while the structure of caves is generally more complex [5] and includes branches, intersections, dead-ends, and loops.

This paper explores a scenario where a single agent explores while others form a chain linking the leader to the base. Two key single-agent exploration algorithms exist: the "Left Wall Follower," which follows the left wall but

ensures full coverage only in simply connected mazes, and "Trémaux's Algorithm," which works in all structured mazes but requires marking already explored passages [2]. It is worth noting that some multi-agent exploration methods have adopted this principle by enabling loop detection through environment marking, as seen for example in [4]. In our approach, we benefit from the chain of UAVS to avoid marking the environment: we ensure that at every intersection between the base and the leader, there is a following UAV, which indicates that passages have already been traversed. This is similar to loop detection techniques based on robot rendezvous at the nodes of an exploration tree [10].

We, thus create a dynamically changing topological graph [17], where agents are placed in the junctions (and also on the limits of the visual connectivity range [8]). This allows us to provide a guaranteed exploration algorithm for an arbitrary cave and being able to localize the leader without mapping the environment. To deploy the chain we benefit from the leader-follower formation approaches [16], [6], while paying special attention to maintaining the connectivity on the intersections. This proposal is developed in the framework of a more general project, called VORTEX[1].

III. Assumptions about the problem and the agents

Let us specify the assumptions of the problem addressed in this paper, namely the complete exploration of unknown underground environments by a fleet of drones:

1) The environments to be explored consist of narrow corridors that are always traversable, intersections, and dead ends (see illustrations).
2) The drones are equipped with a sensor that allows them to measure their distance to the walls in 360°.
3) Communications between drones are limited to visual signals between neighbors.
4) The drones have a sensor that enables them to detect directly neighboring drones and estimate their relatives positions.
5) The drones do not perform mapping of the environment and do not have a global localization system.
6) There is no limit on the number of available drones.

The task is considered complete when the entire environment has been visited, ideally, as fast as possible. In a practical setting where the number of drones is limited (i.e., assumption (6) does not hold), the exploration depth is constrained by the maximum length of the agent chain.

IV. Exploration Based on a Chain of Drones

To accomplish the exploration task under the constraints presented in Section 1, we considered the use of a strategy involving the deployment of a chain of drones, all connected visually by at least one nearby neighbor, as radio/wifi communications are not possible. Before developing such a solution (in Section IV-B), we first consider an exploration with a single drone.

[1]ANR VORTEX 2023-28, ref. ANR-23-IAS2-0005

A. Single Agent Exploration

In this particular case of a single drone, it is evident that the drone will not maintain a permanent link with the entrance (which will be enabled by the chain in the next section).

Algorithm 1 Single Agent Exploration

```
procedure FOLLOW_LEFT_WALL
    while agent not returned to base do
        if agent meets a dead end then
            function TURN-AROUND
        if agent meets an intersection then
            function ROTATE(branch_Left)
        function MOVEFORWARD_BRANCH
    Exploration completed
```

Note : branche_Left = leftmost branch for an agent at an intersection (see figure 1).

We propose a solution inspired by the method for solving a maze without loops, known as the "left-hand method", which involves keeping one hand on a wall without lifting it. This means, in the context of our environments, that at each intersection the agent will take the branch of the intersection that is farthest to its left, as explained in Figure 1. Once the agent reaches a dead end, it will turn around and continue the exploration. The exploration ends when the agent has returned to the base. The proposal is given by Algorithm 1.

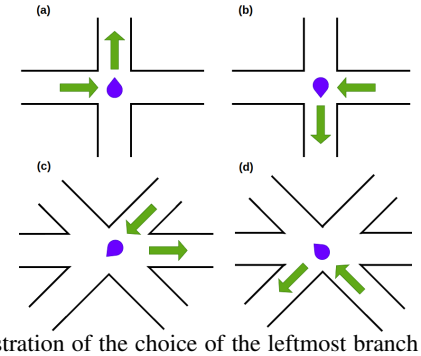


Fig. 1: Illustration of the choice of the leftmost branch for an agent on two types of intersection: (a) and (b) for a cross intersection, (c) and (d) for a 'star' intersection.

Using the exploration strategy presented in Algorithm 1, all environments without loops can be fully explored [2]. However, for environments containing loops, some branches may not be explored by the agent, as shown in Figure 2. A well-known solution for the loop problem is Trémaux's algorithm [2], which involves marking the paths taken at intersections to detect a loop, and in that case, the agent would backtrack to the previous intersection to search for a new branch to explore. In this study, our working assumptions are not to map the environment or mark it (which remains challenging to envision with drones). For this reason, we propose in the next section to deploy a chain of drones that will easily detect a loop when the head of the chain encounters one of its elements.

It is important to note that functions of the Algorithm 1 will be used in the chain by the drone acting as the leader,

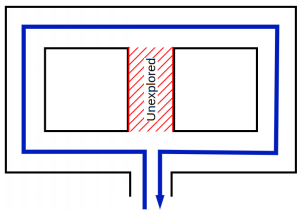


Fig. 2: Double-loop environment. The connecting branch between the two loops remains unexplored when following a wall-to-left strategy with a single agent.

as it will guide the exploration strategy.

B. Exploration based on deploying a chain of drones

We now consider a chain of agents with no limit on the number. That is, there is always an agent available to join the chain (at the entrance of the environment, called the base).

The general idea is to have a leader drone that guides the deployment of a chain of drones behind it. Before providing the details of its behavior, let's specify **the assumptions** on the behaviors of the agents forming the chain:

1) Two roles are possible: Leader or Follower. At any given time, only one agent can be the leader, and the other agents are followers (i.e., they follow the previous agent). Leader is recognized by a specific signal it broadcasts.
2) The drone chain ensures the connectivity property, which guarantees the propagation of signals along the chain, even in curved tunnels.
3) During deployment, there is always an agent present at every intersection used by the chain.
4) There is always a stationary drone at the entrance of the environment, ready to enter.

Algorithm 2 presents the exploration strategy guided by the leader. This is a centralized approach, which allows understanding the leader change when the chain encounters a dead end or a loop (i.e., itself). In Section V, we will present the agent behaviors (leader and follower) that enable a decentralized version of this strategy.

In this algorithm, the variable "AgentLeader" represents the agent in the chain that holds the leader role. By default, the leader agent advances along a branch and turns toward the leftmost branch when an intersection is encountered (following the strategy given in Algorithm 1). When the leader agent encounters a dead end or a loop, the leader agent must be changed in order to extend the chain from an intersection with an unexplored branch. This process (lines 4-14) involves assigning the "AgentLeader" role to a follower agent from an intersection upstream of the leader. To achieve this, a backtracking mechanism is used to return to previous intersections. The function IntersectionPrec(f) returns the agent located at the previous intersection of agent f. The backtracking stops if the function UnexploredBranch(f) returns "True," meaning there is at least one unexplored branch around agent f (line 11).

Note that when a reconfiguration is made, the new leader will first be followed by the agents present in the branch that has just been explored. Thus, the end of the chain temporarily forms a Y-shape, consisting of two sub-chains connected at the intersection of the new leader. Therefore, the follower role is divided into two sub-roles. The first, root followers, refers to drones that are part of the main chain connecting the leader and the base. The second, reconfiguration followers, refers to drones from the fully explored branch. Such a configuration is illustrated in Figure 6.

Let us remark that when the leader perceives a drone in front of him, it means that the chain formed the loop and the leader has arrived at the previously explored intersection. Hence, the drone on the intersection acts like a marker in Trémaux's algorithm [2]. Furthermore, the systematic deployment of this chain to the left allows agents who do not perform mapping to know whether all branches of an intersection have been explored.

Algorithm 2 Leader-guided Exploration

```
1:  procedure EXPLORATION_LEADER
2:      AgentLeader ← Agent_1
3:      while true do
4:          if AgentLeader arrives at a dead-end or a loop then
5:              f ← IntersectionPrec(AgentLeader)
6:              while no new leader do
7:                  if f at base then
8:                      AgentLeader ← f
9:                      Exploration completed
10:                 else
11:                     if UnexploredBranch(f) then
12:                         AgentLeader ← f
13:                     else
14:                         f ← IntersectionPrec(f)
15:         if AgentLeader meets an intersection then
16:             function TURN_LEFT(AgentLeader)
17:         function MOVEFORWARD_BRANCH(AgentLeader)
```

V. Behaviors of the robots

Now that the exploration strategy through the deployment of a chain of drones has been presented in its centralized version, we will detail in this section its decentralized expression, that is, in the form of agent behaviors.

In the first instance, we provide the state machine for the behaviors of the Leader and Follower roles, then we detail the low-level actions as well as the perceptions necessary for their execution.

A. Role behaviors

We now describe the operation of the Leader and Follower (root and reconfiguration) roles, which are formalized in the state machines provided in Figure 3.

The behavior of the **leader** is divided into 4 possible situations:

1) Explore a branch: When a branch has been chosen to be explored, for example after an intersection, this behavior is performed by default. It consists of moving straight ahead until a new intersection or dead end is reached. This behavior is carried out while ensuring

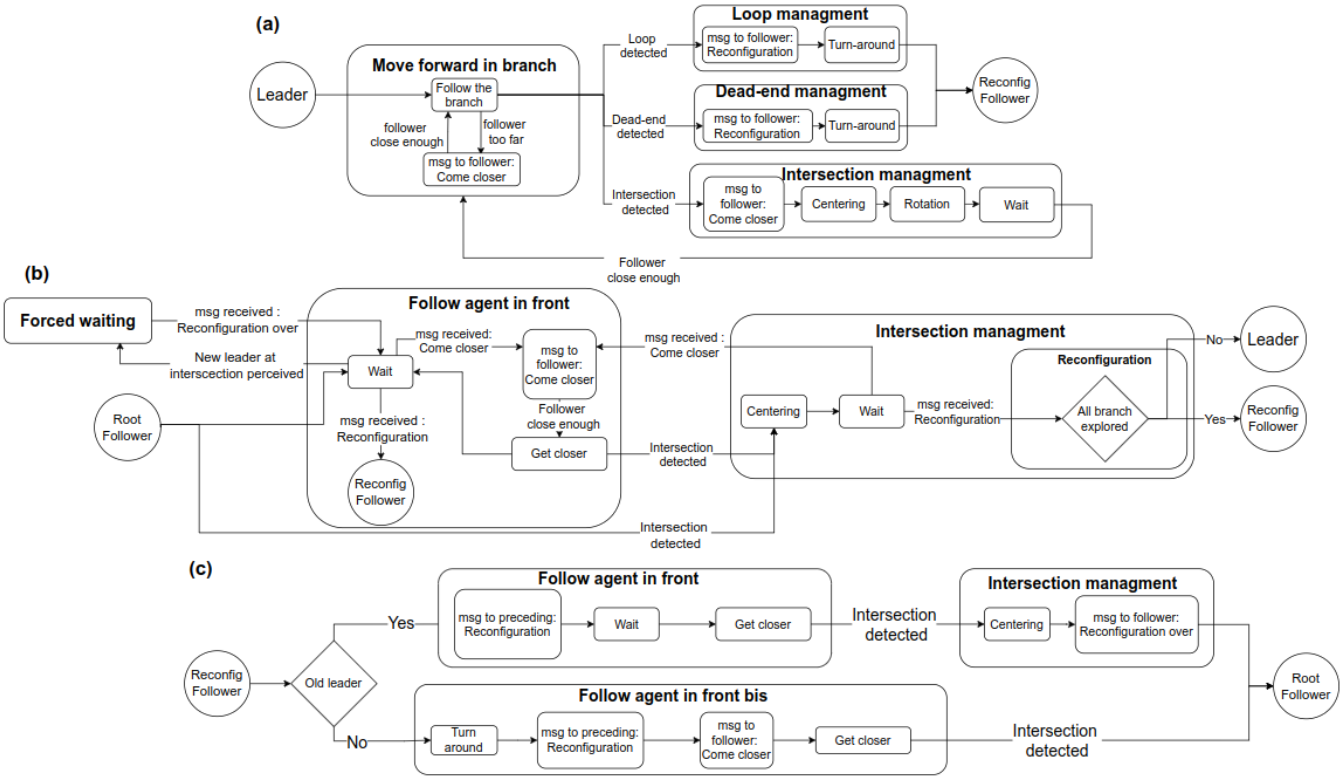


Fig. 3: (a) state machine of leader role (b) state machine of root follower role (c) state machine of reconfiguration follower role

that the distance between the leader and the drone behind it does not become too large.

2) Manage an intersection: When the leader reaches an intersection, a specific behavior is triggered. It must send a message to the follower behind it to get closer so that the exploration can continue without loss of visual connectivity.

3) Manage a dead end: When the leader reaches a dead end, the drone chain must be reconfigured by assigning a new leader, following the strategy presented in Algorithm 2. This involves the leader drone sending a reconfiguration message and then becoming a follower of the chain.

4) Manage a loop: The behavior for managing a loop is identical to that of a dead end.

Behavior of a **root follower**:

1) Follow the agent in front: This behavior ensures the connectivity assumption 2 of the chain, presented at the beginning of section IV-B. Its goal is to ensure that a follower does not move forward until it has been requested to do so, nor that its own follower is not sufficiently close to it. Thus, visual connectivity between agents is maintained at all times.

2) Manage intersections: This behavior ensures the assumption 3 that there is an agent at every intersection crossed by the chain. Indeed, followers must be able to detect intersections and wait for a ”Get closer” or ”Branch reconfiguration” message. The latter message implies that the agent at the intersection must test whether it becomes the new leader or not.

3) Forced waiting: When an explored branch is being emptied of its reconfiguration drones, the root follower just before the intersection must wait for the process to complete.

Behavior of a **reconfiguration follower**:

1) Follow agent in front: same as the root follower, but with the particularity of directly coming closer if the agent is the old leader, and the ability to send the reconfiguration message to make other agents of the explored branch turn around.

2) Intersection Management: upon detecting an intersection, the agent reverts to the root follower role. However, before this role change, the last reconfiguration follower in the explored branch (the former leader) sends a message signaling the end of reconfiguration. This message, received by the root follower just before the intersection, ends the Forced Waiting behavior.

B. Details of behaviors and communications

This section aims to further clarify the operation of certain behaviors, by detailing either the necessary perceptions or the processes involved.

a) Communications between drones: The state machine (Figure 3) uses messages broadcast between neighboring

drones. The receivers interpret or ignore the messages depending on their state. However, we only use three different messages to manage the strategy (”Get closer” and ”Reconfiguration”, ”Reconfiguration is over”). This minimal number of messages allows for the possibility of implementing it with a very simple visual signal system.

b) Detection of loops: Our state machine allows the chain to handle loops in the environment, but it is necessary to be able to detect them, which is not straightforward without marking the environment or mapping. We propose that their detection be based on the ability of the agents, especially the leader, to perceive other agents. Indeed, as shown in Figure 4, if a leader perceives one of its followers in front of it, there must necessarily be a loop in the environment. The leader then enters the ”Loop Management” state (cf. Figure 3). The presence of a follower at the intersection indicates that the leftmost branch has already been explored, and therefore, we avoid re-entering this branch, which forms a loop.

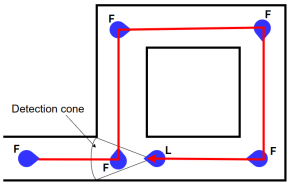


Fig. 4: Principle of the loop detection

c) Communication of reconfiguration requests: Reconfiguration messages have the specific characteristic of needing to be propagated along the chain to the future leader. All followers not positioned at intersections must forward the same message to the drone behind them. However, followers positioned at intersections must, upon receiving the message, assess whether they become the new leader. If so, they do not forward the message; otherwise, they forward it to the drone behind them. This process is illustrated in Figure 5 (a) and (b).

d) Dead-end management: Dead ends are easier to detect than loops, as the sensors inform the agent that a branch is closing. The chain must therefore reconfigure, and this is done using the same process as for loops (cf. above).

e) Conditions to become the new Leader: Here, we detail the process of self-determining the new leader, following the detection of a dead-end or a loop by the chain. When a follower at an intersection receives the ”Reconfiguration” message, it must determine whether it becomes the leader or continues propagating the message. To do so, it must assess whether all the branches it perceives have already been explored, as if they haven’t, it becomes the leader. The agent determines if there is an available branch by performing a right rotation (corresponding to the next left branch for the agent that just transmitted the message).

The figures 5 (c) and (d) illustrate the case of a new leader discovering the next free branch to explore.

f) Exit of agents from an already explored branch: When a new leader at an intersection guides the chain into a new branch, the agents present in the old branch must be reintegrated into the exploration chain. Thus, at the intersection, they will be prioritized to follow the leader. This Y-shape formation at the end of the chain is temporary, lasting only until the agents leave the old branch. The last follower in the branch then unfreezes the waiting at the intersection of the root follower, by sending the message ”Reconfiguration is over”. This process is illustrated in Figure 6.

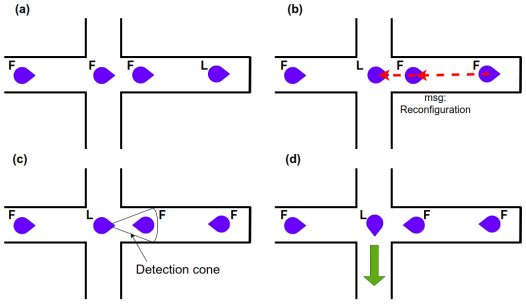


Fig. 5: Principle for Determining the New Leader and Choosing the Next Branch to Explore

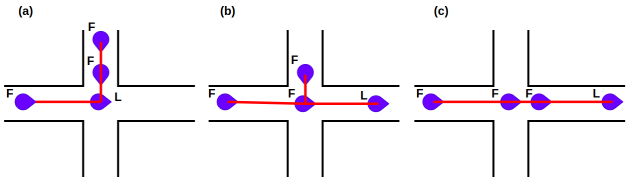


Fig. 6: Y-shaped chain formation

VI. Experiments in Simulation

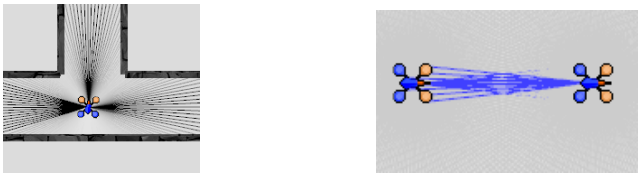
The objective of this section is to evaluate, through simulation, the exploration strategy based on the deployment of a chain of agents, as defined by the state machine in Figure 3. We aim to conduct this evaluation in environments composed of narrow corridors, intersections, loops, and dead ends (see Section III).

A. Simulation tool

In this article, we focus on the exploration of galleries with planar topology, where drones fly at constant altitude, which can be represented in 2D. We used the swarm-rescue simulator [1] to implement this strategy. It is a 2D multi-agent simulator written in Python that allows easy simulation of drone teams with autonomous behaviors to perform search & rescue tasks in custom-created complex environments. Figure 8 shows an example of an environment created to test our algorithm.

The drones are simulated using a dynamic physical model. They are set in motion by the forces applied to them, and they have weight, inertia, and a body that prevents them from being reduced to points in space. They are equipped with two sensors: a laser rangefinder that provides the distance to the walls over 360° and a sensor that allows them to detect other drones within a 360° range (cf. figure 7). Both sensors operate within a limited range. Each of these sensors returns a noisy data vector. The drones can

also send messages to each other (emulating, in our case, communication via signals, e.g., light signals).



(a) Simulation of the Laser rangefinder

(b) Localization of a neighboring drone

Fig. 7: Drone Sensors

The simulator works as follows: upon launch, a state machine instance is created for each drone. Then, at each time step, all drones calculate a control function that returns three force values (forward, lateral, rotation) to enable their movement. The sensor data and message exchanges modify the states of the state machine as the exploration progresses.

B. Deployment of the drone chain

At the beginning of the simulation, the drones are positioned in a "storage" space as shown in Figure 8. From this space, they will be sent into the environment to join the chain when necessary.

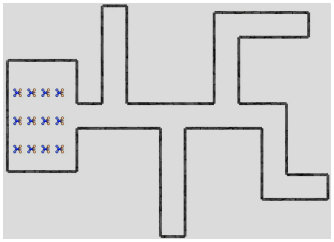


Fig. 8: Start of the simulation.

Figure 9 shows the deployment of the chain in the simulated environment until the first dead-end is encountered and before the reconfiguration of the chain. The video [2] corresponds to this simulation. We describe below the main stages of this first phase of exploration, i.e. until the chain reaches its first dead end.

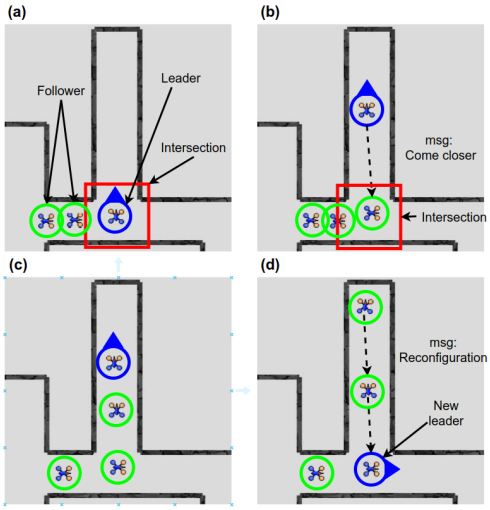


Fig. 9: Deployment of the chain up to the first encountered dead end

In Figure 9(a), the leader has detected an intersection, so it is in the "Manage Intersection" state (cf. Figure 3). The blue tip indicates the direction the leader is heading, which is the leftmost branch, in accordance with our exploration strategy. Since this intersection is at the beginning of the environment, two followers are sufficient in the chain to maintain a connection to the entry.

In (b), the leader drone is now in the "Move along a branch" state, and the drone following it has entered the "Manage Intersection" state. However, the leader is too far from the rest of the chain. To continue its exploration, it sends a "Come closer" message, which is perceived by the follower at the intersection, who in turn sends the message to the next drone. This ensures that each agent moves only if another agent is nearby behind it, guaranteeing connectivity.

In (c), we observe that the leader drone perceives a follower sufficiently close to it to continue exploring the branch, thus returning to the "Move forward in branch" state (cf. Fig. 3). The situation in (d) presents the first encounter of the chain with a dead-end. The drone that detected the dead-end has propagated a "Reconfiguration" message and left its role as leader. The drone positioned at the intersection was able to self-determine as the new leader because the intersection has unexplored branches.

[2] https://mathismbl.github.io/swarm-rescue/

C. Chain reconfiguration

In this last section, we present the continuation of the exploration process, that is the reconfigurion of the chain when a deadlock is detected.

After detecting a dead-end, the chain must proceed with its reconfiguration. A new leader is designated, and it is now this leader who will guide the chain. Figure 10 (a) shows the situation where all the agents in the chain have moved closer to the new leader so it can continue the exploration. In Figure 10 (b), the chain is in a transitional phase where its end forms a "Y", while the agents present in the previously explored branch reinsert themselves properly into the chain.

Figure 10 shows the end of the exploration. When the last branch of the environment has been explored, the reconfiguration message, sent by the leader who encountered the last dead-end, is propagated back to the agent located at the entrance of the environment. This agent then becomes the new leader, and the exploration is completed in accordance with the termination condition defined in Algorithm 2. In the end of exploration, the leader retracts the chain out of the environment.

Thus, the exploration path (followed by the leader agents) is represented in Figure 11. The red arrows represent the physically traveled path of the drone holding the leader role. The green arrows represent the path of reconfiguration messages that transfer the leader role to another agent.

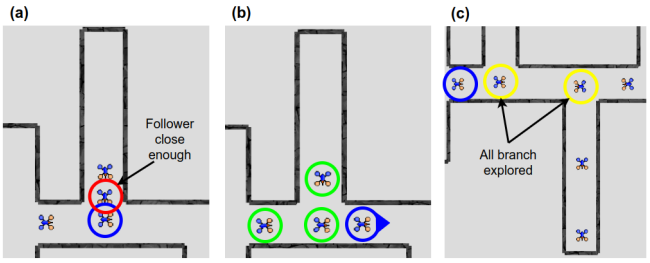


Fig. 10: Chain evolution at the new leader's intersection (in blue): a) the agents in the previously explored branch have moved closer to the new leader, b) the new leader progresses, and the end of the chain temporarily takes a 'Y' shape while these agents follow, c) end of exploration, the last explored branch results in the agent at the entrance becoming the new leader.

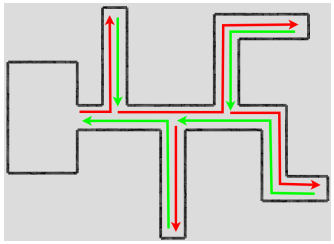


Fig. 11: Path followed by the leader: in red, the progression phases through the corridors; in green, the reconfiguration request messages for the assignment of a new leader.

VII. Conclusion

In this paper, we have developed an algorithm ensuring the complete exploration of any underground environment composed of arbitrarily connected tunnels. The exploration is carried out by a fleet of mini-drones forming a chain, consisting of follower drones between the leader (guiding the exploration) and the base. The deployment of the chain of drones allows loop detections, and hence makes "wall-following" exploration strategy complete in any cave environment.

Since all the agents in the chain are visually connected, one can also transmit information from the leader to the base using only visual signals (no wireless communication network is required) and provides a topological representation of the environment without mapping it. Signals also enable dynamically determining a new leader for fast and efficient reconfiguration of the chain when the former leader meets a dead end or a loop.

As a decentralized solution, we proposed the state machine of the different roles. We presented the first experimental results, using a realistic quadrotor mini-drone simulator, showing the ability of the chain of drones to explore an unknown environment and to reconfigure when meeting dead-ends or loops.

In future work, we plan to address the current limitations of the model, particularly its robustness to agent loss. We also intend to compare our approach with other strategies, such as those relying on mapping, in order to highlight differences in exploration time and memory consumption. Furthermore, we aim to improve the leader-follower behavior to ensure a smoother progression of the chain and extend the proposed approach to multi-leader exploration scenarios (forming a dynamic exploration tree from the entrance). Moreover, it is envisioned to allow drones to temporarily and selectively detach from the rest of the chain in specific situations, in order to reduce the number of drones required for the exploration. We have also started building a demonstrator with real drones, in preparation for real missions as part of the ANR VORTEX project[3].

References

[1] E. Battesti. Swarm rescue simulator. `https://github.com/emmanuel-battesti/swarm-rescue`.

[2] E. Bienias, K. Szczepański, and P. Duch. Maze Exploration Algorithm for Small Mobile Platforms. *Image Processing & Communications*, 21(3):15–26, Sept. 2016.

[3] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 37(8):1363–1388, Dec. 2020.

[4] S. Das, P. Flocchini, A. Nayak, and N. Santoro. Distributed exploration of an unknown graph. In *Structural Information and Communication Complexity*, pages 99–114. Springer Berlin Heidelberg, 2005.

[5] M. Dharmadhikari, H. Nguyen, F. Mascarich, N. Khedekar, and K. Alexis. Autonomous Cave Exploration using Aerial Robots. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 942–949, June 2021. ISSN: 2575-7296.

[6] N. Evangeliou, D. Chaikalis, A. Tsoukalas, and A. Tzes. Visual collaboration leader-follower uav-formation for indoor exploration. *Frontiers in Robotics and AI*, 8:777535, 2022.

[7] A. Farooq, A. Anastasiou, N. Souli, C. Laoudias, P. S. Kolios, and T. Theocharides. UAV Autonomous Indoor Exploration and Mapping for SAR Missions: Reflections from the ICUAS 2022 Competition. In *19th Int. Conf. on Ubiquitous Robots (UR)*, pages 621–626, 2022.

[8] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transactions on Mechatronics*, 14(2):163–175, 2009.

[9] S. A. Ghauri, M. Sarfraz, R. A. Qamar, M. F. Sohail, and S. A. Khan. A review of multi-uav task allocation algorithms for a search and rescue scenario. *Journal of Sensor and Actuator Networks*, 13(5), 2024.

[10] C. Gong, S. Tully, G. Kantor, and H. Choset. Multi-agent deterministic graph mapping via robot rendezvous. In *2012 IEEE International Conference on Robotics and Automation*, pages 1278–1283, 2012.

[11] N. Kottege and al. Heterogeneous robot teams with unified perception and autonomy: How team csiro data61 tied for the top score at the darpa subterranean challenge. *Field Robotics*, 4:313–359, 2024.

[12] P. Laclau, V. Tempez, F. Ruffier, E. Natalizio, and J.-B. Mouret. Signal-Based Self-Organization of a Chain of UAVs for Subterranean Exploration. *Frontiers in Robotics and AI*, 8:614206, Apr. 2021.

[13] R. Lösch, S. Grehl, M. Donner, C. Buhl, and B. Jung. Design of an Autonomous Robot for Mapping, Navigation, and Manipulation in Underground Mines. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1407–1412, 2018. ISSN: 2153-0866.

[14] K. Otsu, S. Tepsuporn, R. Thakker, T. S. Vaquero, J. A. Edlund, W. Walsh, G. Miles, T. Heywood, M. T. Wolf, and A.-A. Agha-Mohammadi. Supervised Autonomy for Communication-degraded Subterranean Exploration by a Robot Team. In *2020 IEEE Aerospace Conference*, pages 1–9, Mar. 2020. ISSN: 1095-323X.

[15] P. Petráček, V. Krátký, M. Petrlík, T. Báča, R. Kratochvíl, and M. Saska. Large-Scale Exploration of Cave Environments by Unmanned Aerial Vehicles. *IEEE Robotics and Automation Letters*, 6(4):7596–7603, 2021.

[16] R. Rafifandi, D. Asri, and E. Ekawati. Leader–follower formation control of two quadrotor uavs. *SN Applied Sciences*, 1:539, 2019.

[17] D. Silver, D. Ferguson, A. Morris, and S. Thayer. Topological exploration of subterranean environments. *Journal of Field Robotics*, 23(6-7):395–415, 2006.

[3] ANR VORTEX 2023-28, ref. ANR-23-IAS2-0005