Evolutionary Synthesis of Communication-Based Aerial Swarms

THÈSE N^{O} 4900 (2010)

PRÉSENTÉE LE 6 DÉCEMBRE 2010

À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE SYSTÈMES INTELLIGENTS
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Sabine HAUERT

acceptée sur proposition du jury:

Prof. B. Rimoldi, président du jury
Prof. D. Floreano, Dr J.-C. Zufferey, directeurs de thèse
Prof. A. Ijspeert, rapporteur
Prof. G. Sukhatme, rapporteur
Prof. A. Winfield, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2010

Acknowledgements

Least known to scientific research are the human discoveries we make along the way.

This work would not have been possible without the support of Dr. Alain Jaquier and armasuisse, competence sector Science + Technology for the Swiss Federal Department of Defense, Civil Protection and Sports.

I am most grateful to Dario Floreano and Jean-Christophe Zufferey for the opportunity to work on this amazing project that pushed me into unexplored territories and helped me think out of the box. I have no doubt that their visionary and ambitious perspective on science will forever influence the way I approach research. In particular, I thank Dario for his trust and support that allowed me to follow my own ideas. He has also taught me how to best communicate my work. Finally I thank him for the amazing learning experiences that have had me podcasting, organizing conferences and workshops and writing research proposals. To Jean-Christophe I am in debt for the huge amount of time he spent discussing and reading my work. His input was always extremely valuable and veracious. Furthermore, I thank him for his contagious motivation to work with flying robots.

Central to this entire project is the work by Severin Leven who made the flying robots. Over the last four years it has been an immense pleasure working with him. His patience, engineering skills, and systematic approach at doing research have lead to the most robust aerial testbed developed to this day and no aerial experiments would have been possible without this rigor. I also thank him for his precious support and willingness to discuss ideas.

I am thankful to the referees, Auke Ijspeert, Gaurav Sukhatme and Alan Winfield and to the president of the jury, Bixio Rimoldi, for taking the time to

read this thesis and for their valuable feedback that contributed to improving this work.

The Laboratory of Intelligent Systems is one of the best places to work and I love seeing all sorts of robots rolling, jumping and flying around. Central to this environment are the amazing people working there. In particular, I would like to thank all my colleagues for the relaxing lunch breaks, coffees and evening outings. A special thanks to Peter Dürr and Maja Varga who read this thesis.

In addition to friends from the laboratory, I have been able to count on my friends from Neuchâtel and from chemistry for unforgettable memories.

Finally, this work would not have been possible without the support of my loving family. Thank you to my parents Sandra and Daniel who always encouraged me in my work and provided a nurturing environment. A special thanks to my brother Christopher for contributing his immense talent to the art that is presented in this thesis and to my sister Sylvie for being my best friend in all situations.

Last but not least, I thank Sébastien for standing by me for the past 10 years. He's my pillar and balance. Without his love and support, I would no doubt break.

Abstract

Aerial swarms have the potential to search for forest fires, chemical plumes or victims and serve as communication and sensor networks in the sky. Flying robots are interesting for such applications because they are fast, can easily overcome difficult terrain and provide line-of-sight communication or aerial perspectives. However, swarms of flying robots have so far only been demonstrated in simulation or in few examples in reality. Current simulators typically rely on unrealistic assumptions concerning robot sensing capabilities and motion. To bridge this reality gap we propose to address two key challenges.

The first challenge consists in discovering swarm controllers that do not use position information. Swarm controllers in the literature rely on global or relative position information which can be obtained using GPS, cameras or range and bearing sensors. However, position requirements typically entail robots that are complex, heavy and expensive or that rely on specific environmental conditions to function. Lifting the need for position could instead lead to swarm controllers that can be deployed in a variety of environments and using very simple robots.

The second challenge addressed consists in developing swarm controllers that can accommodate motion constraints of flying robots. In particular, we consider fixed-wing robots that can not stop or turn on the spot like ground robots or rotor crafts. Instead they must fly at relatively high speeds to avoid stalling. Therefore, making the robots advance slowly in average can only be done by having them turn.

To address these challenges, we consider robot controllers that continuously react to wireless communication with neighboring robots or people by changing their turn rate (communication-based behaviors). Using communication as a

sensor for flying robots is interesting because most robots are equipped with off-the-shelf radio modules that are low-cost, light-weight and relatively long-range.

The design of robot controllers that can lead to desired swarm behaviors is done following a systematic approach. First, artificial evolution is used to automatically discover simple and unthought-of controllers for swarms of flying robots. Evolved controllers are then reverse engineered to obtain basic behaviors that can be modeled and used in a variety of swarm applications.

In order to discover a range of basic behaviors, we consider a challenging swarm application that requires robots to direct themselves in their environment, move in groups, cover an area, and maintain a communication relay. Discovered behaviors are then extended to scenarios with wind.

Overall, this thesis presents the evolutionary synthesis of communication-based behaviors for swarms of fixed-wing flying robots.

Keywords: flying robot, MAV, UAV, swarm, communication-relay, ad-hoc network, wireless communication, artificial evolution, reverse engineering

Résumé

Des essaims aériens sont capables de détecter des incendies de forêt, des nuages toxiques, ou encore des personnes isolées, et peuvent servir de réseaux aériens de communications ou de recherche. Dans cette perspective, les robots aériens sont intéressants grâce à leur faculté à fonctionner même en cas de terrain accidenté, ainsi que par leur capacité à fournir des images d'ensemble par des prises de vues aériennes. Cependant, il n'existe que très peu de démonstrations d'essaims de robots volants fonctionnant en réalité. Jusqu'à présent, la majorité de la recherche dans ce domaine s'est cantonnée à la simulation. De plus, les simulateurs développés supposent généralement des robots possédant des capacités de mouvement et de perception irréalisables en pratique. Dans ce travail, nous poursuivons deux objectifs principaux visant à développer des essaims de robots volants.

Le premier objectif est le développement de contrôleurs d'essaim qui ne nécessitent pas d'information quant à la position des robots. Les contrôleurs d'essaim connus dans la littérature requièrent une connaissance de la position globale ou relative, qui est en général obtenue par des moyens tels que le GPS, des caméras, ou encore des capteurs de distance et d'angle. La nécessité de devoir déterminer leur position augmente la complexité, le poids et le prix des robots, et peut imposer une limitation concernant les conditions dans lesquelles ils fonctionnent. En s'affranchissant de la nécessité de déterminer la position, des contrôleurs d'essaim et des robots simplifiés et adaptables à de nombreuses situations peuvent être imaginés.

Le second objectif consiste à développer des contrôleurs d'essaim adaptés aux contraintes propres à des robots volants. Dans notre cas particulier, nous considérons des robots de type aile volante qui ne peuvent pas s'immobiliser ni

tourner sur eux-mêmes comme le feraient des robots au sol ou des hélicoptères. Au contraire, les ailes utilisées doivent voler à une vitesse suffisante pour éviter de décrocher. La direction et la vitesse relative au sol des robots ne peuvent donc être modifiées qu'en faisant tourner les robots.

Les études visant à atteindre ces objectifs sont basées sur des contrôleurs de robots qui réagissent continuellement à la communication sans fil avec des robots avoisinants ou avec des utilisateurs au sol (comportement basé sur la communication). L'utilisation de la communication comme capteur principal pour des robots volants est intéressante car elle ne nécessite qu'une instrumentation simple, légère et non coûteuse (modules de radio) qui équipe la majorité des robots produits aujourd'hui.

Une approche systématique est employée afin de définir des contrôleurs qui résultent en comportements d'essaims. Premièrement, l'évolution artificielle est utilisée afin de découvrir des contrôleurs d'essaims de robots volants. Les contrôleurs ainsi obtenus sont ensuite étudiés par ingénierie inverse afin d'isoler des comportements de base. Ceux-ci peuvent ensuite être modélisés et recombinés, ce qui résulte en une série de comportements pour l'essaim.

Afin de déterminer une série de comportements de base, une ambitieuse application est présentée, qui implique des robots devant se diriger dans leur environnement, se déplacer en groupe, explorer une superficie donnée, ainsi qu'établir et optimiser un réseau de communication. A l'étape suivante, ces comportements sont adaptés de façon à ce que l'essaim puisse voler en tenant compte du vent.

En résumé, ce travail présente le développement à partir de l'évolution artificielle de comportements pour des essaims de robots volants. Ces comportements sont basés sur la communication, et spécifiquement adaptés à des ailes volantes.

Mots clés: robot volant, communication, réseau ad-hoc, évolution artificielle, ingénierie inverse

Contents

1 Introduction

"The art of simplicity is a puzzle of complexity." Douglas Horton

Thanks to Christopher Hauert for capturing the essence of the different puzzle pieces that make this thesis.

Abstract

Collective aerial systems can be rapidly deployed in real-life applications to overcome difficult terrain and provide an aerial perspective of the world or line-of-sight communication. Most flying robots described in the literature are equipped with Global Positioning Systems (GPS) or position sensors that tend to make them expensive, heavy and unusable in certain environments. Furthermore, their motion constraints are seldom taken into account.

To address these challenges, we propose to design robot behaviors that use off-the-shelf radio modules for sensory input instead of position sensors (communication-based behaviors). Robots react to receiving messages from other robots or people by modulating their turn rate while maintaining a constant speed and altitude. This leads to motions that are suitable for fixed-wing robots that can not stop or turn on the spot like ground robots or helicopters.

However, designing swarm controllers is challenging because of the distributed nature of the system. To this end, artificial evolution is used to automatically find simple, efficient and unthought-of swarm controllers. By analyzing the best evolved solutions, basic behaviors are extracted to achieve individual robot motion, group motion, area coverage and communication relay. Each behavior is modeled and validated in theory, simulation or reality. Basic behaviors can then serve as building blocks for the design of a large variety of future aerial swarm systems.

1.1 State of the Art

Flying robots are increasingly being considered for real-world applications, including search-and-rescue, surveillance, monitoring or reconnaissance[115]. Aerial systems have the advantage of rapidly overcoming difficult terrain typical of disaster areas or cities, and providing an aerial perspective of the world. Current flying robots are often controlled by an operator on the ground that either teleoperates them or determines a sequence of GPS waypoints they have to follow. However, a single robot has a limited reach and missions would benefit from using several robots to multiply the performance of the system, share computation, communication and sensing resources or to provide multiple view points[25].

In particular, groups of flying robots have been envisioned for a wide range of applications mainly involving exploration and coverage. Examples include detecting and sensing chemical plumes[55,57,77,113], wild fires[36,66], victims[71] and other targets of interest[4,18,56,88,90,95,111]. More recently, flying robots have been considered to serve as sensor or communication networks in the sky[6,23,46,47,56,94].

Literature on collective aerial systems mostly covers work done in simulation. Initial work in the field concentrated on centralized solutions where robots relay mission information to a central planner or human operator that then takes care of computing optimal deployments for each robot[2,7,9,36,66,84]. With this approach however, each robot added to the system increases the amount of information that needs to be communicated to and treated by the centralized planner, therefore limiting the amount of robots that can be deployed. In the case of a human planner, this limit has been shown to be around 5[22]. Furthermore, the system is entirely dependent on the correct functioning of the ground operator, which in the case of a failure would compromise the mission.

To increase the amount of robots in the air and therefore their added value to a mission, roboticists are concentrating on giving decision power to the individual robots rather than to a central station on the ground. Inspiration can be taken from swarming in nature where animals with limited capabilities are able to cooperate towards a common goal. In these systems, the intelligence of the group emerges from local interactions among the agents and through the environment (stigmergy) without any centralized planning or global communication[11]. These systems have the advantage of being scalable (as many individuals as needed can be added) and robust (the failure of an individual has little

effect). As an example, bacteria and ants read and emit chemical signals in their environment that can then be used to coordinate motility, synchronize actions[98] and explore an environment[34]. Likewise, birds are able to flock by reacting to neighboring birds through attraction and repulsion mechanisms[83]. In physics, particles are able to create self-ordered motion[106]. Recently, principles underlying swarming in nature have been extracted and applied to robots[8,89,110]. Making simple robots that are able to achieve complex tasks using only local interactions is appealing for aerial systems because of limited sensor and computation payload and for practical, safety and economical reasons[115].

Aerial systems imitating bird flocking have been imagined by Reynolds[83], with extensions implemented in simulation by many researchers[6,21,23,24,53,78]. One noticeable work shows the flocking of indoor blimps in reality[108]. In artificial physics, local interactions are implemented as forces that allow robots to be attracted or repulsed, thereby creating regular grids[16,94,114]. Other aerial swarms take inspiration from ants that self-organize by depositing and sensing chemical signals (pheromones) in their environment. In these systems, virtual pheromone is deposited on a map based on the coordinates of the robots[56,90,104]. Broadly speaking, aerial robots can deposit any type of information on a map and exchange these maps with nearby robots. Using these maps, robots are then able to follow information gradients to areas of interest while avoiding danger[57,80,102]. For a complete overview of collective aerial systems, readers may refer to a recent review[115].

However, assumptions made about flying robots in simulation often undercut their transfer to reality. This can be seen by the few demonstrations performed with physical platforms[3,7,36,49,50,108]. In particular, all algorithms developed for aerial swarming assume robots are able to precisely know their global or relative position and that of their neighbors. Furthermore, proposed controllers often do not consider the actual motion constraints of flying robots and rely on simplistic omnidirectional models[53,94].

1.2 Challenges

In this thesis, we identify two key challenges that, if overcome, would significantly facilitate the deployment of large scale aerial swarms in reality.

1.2.1 Positionless Swarming

The approach to design aerial swarms has so far been to assume that robots precisely know their position and that of their neighbors[115]. This means that they either know their global position (x,y,z) or their relative position (range and bearing) to objects and robots in their environment. However, inferring position in a robust and dependable manner is one of the main challenges in aerial robotics. Current technologies depend on GPS, vision, radars, infrared or ultrasound sensors. Each approach however is subject to shortcomings[46,47]. GPS is not available in areas which do not have access to at least four satellites, such as indoors or in cluttered environments[92]. Vision is not usable at night. Range and bearing sensors based on radars, infrared and ultrasound are often only usable at short ranges (<10 m) or are too heavy for most flying robots. Finally, computationally intensive algorithms are often needed to convert the data into position information (Simultaneous Localization and Mapping, SLAM). The assumption that position is needed, the challenges that entails, and the resulting controllers have led to rather complex flying robots which are, for the most part, only demonstrated in tightly controlled environments[103] or with the help of backup pilots on the ground[3].

Instead, alleviating the need for position can allow for simple, low-cost and light-weight robots that could be used in a variety of environmental conditions. Such systems could be used in places typically deprived from GPS such as space, or on challenging platforms such as insect-sized flyers[31] that will not have the means to localize themselves. Furthermore, removing position leads to interesting research questions and original swarm controllers. One such question looks at how to cope with wind that typically causes robots to drift away in positionless systems.

The first challenge tackled in this thesis is therefore to design aerial swarm controllers that do not rely on position or complex sensors to function.

1.2.2 Motion Constraints

Fixed-wing robots are interesting platforms for outdoor aerial experiments because they are safer, lighter, cheaper and more energy efficient than rotorcrafts. They also have better controllability and wind resistance than lighter-than-air crafts. However, they have highly constrained motion dynamics[28]. Unlike

ground robots or rotorcrafts, fixed-wing robots need to maintain their flight velocity above a certain limit in order not to stall. Slowing down the global speed of the robots can therefore only be done by having the robots turn. The rate with which these robots turn is also upper-bounded (robots can not turn on the spot) depending on the dynamics of the platform. These properties lead to more complex robot trajectories than would typically be observed with holonomic vehicles as shown in Fig. 1.1.

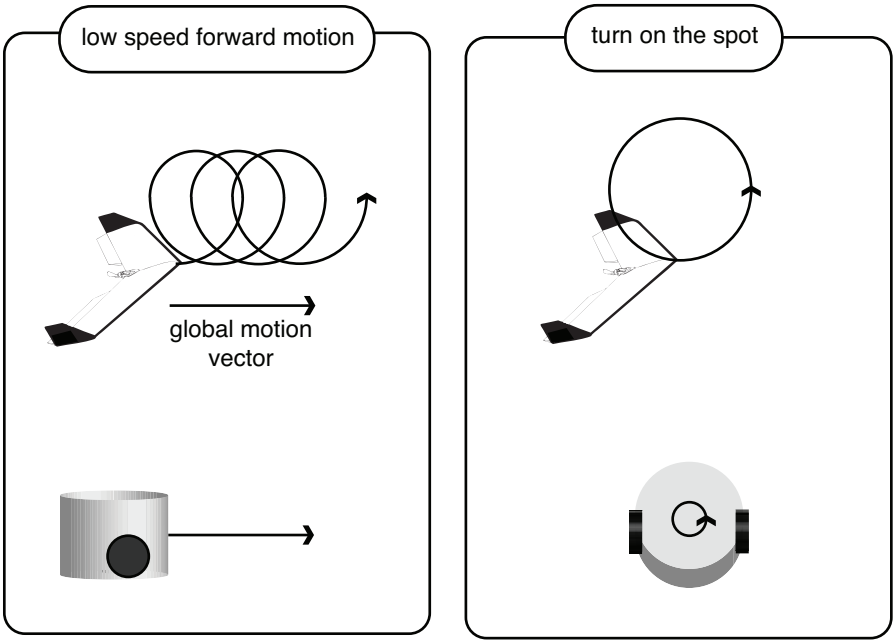


Figure 1.1: Differences in motion constraints between fixed-wing robots and wheeled robots.

Literature on controlling swarms of flying robots has often been limited to assuming robots can move in discrete steps along a grid-like structure[105] or stop and turn on the spot[23,53,94]. Instead, work considering realistic motion constraints of fixed-wing robots typically separates swarm control from robot motion[27,35,57]. In these systems, low-level controllers are responsible for having robots follow a virtual point mass by loitering around it. Swarm controllers then take care of moving this point mass. The individual motion of robots however can be disruptive to swarm behaviors, for example by causing communication links between robots to become intermittent. Separating robot motion and swarm control can therefor be unrealistic.

The second challenge in this thesis is therefore to develop controllers that can cope with the typical motion constraints of fixed-wing robots.

1.3 Method

The aim of the methodology proposed here is to develop swarm controllers for the deployment of positionless fixed-wing aerial robots.

In order to address the first challenge we focus on communication as the main sensory input to control flying robots. Fortunately, most flying robots today are equipped with off-the-shelf radio modules that are cheap, light-weight, relatively long-range and benefit from years of development in industry. For such robots, communication can serve a dual purpose. First the content of a message can be used to relay or exchange sensor data, status information and commands with humans or other robots. Second, simply receiving a message, independently of its content, can give information concerning a robot's ability to communicate with nearby nodes (situated communication[97]). In addition, the robot can rely on typical proprioceptive sensors used for flight such as a compass, pressure sensors and accelerometers.

To fulfill constraints of fixed-wing motion, we focus on controllers that vary the turn rate of the robot within the possibilities of the platform. All robots are commanded to fly at a constant flight speed (to avoid stalling) and at the same altitude. A low level autopilot is responsible for respecting speed, turn rate and altitude commands[59] as well as avoiding collisions among robots, typically by briefly changing their altitude as demonstrated in reality with five fixed-wing robots[60]. Such a controller architecture can be seen in Fig. 1.2.

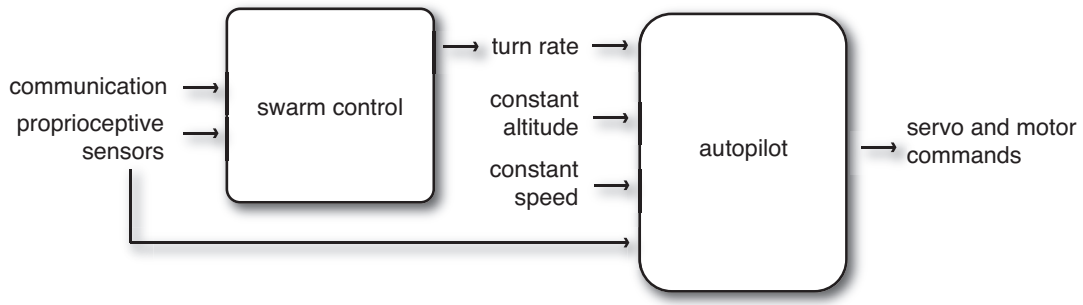


Figure 1.2: Controller architecture suitable for positionless fixed-wing aerial robots.

Controllers that react to communication input by modifying robot motion (in this case turn rate) are referred to as "communication-based behaviors" throughout this thesis. Determining what communication-based behaviors can lead to aerial swarming is challenging because there is usually no obvious relationship between the individual robot behaviors and the emergent behavior of the group[110]. This issue is exacerbated in systems with limited information (e.g. no position) and motion constraints of fixed-wing platforms because no existing algorithms from the literature can be applied.

To overcome these challenges, we propose a systematic approach to develop communication-based behaviors for swarms of fixed-wing robots. The first step consists in using artificial evolution[32] to automatically design swarm controllers. Artificial evolution has shown its potential in the past in finding simple, efficient and original solutions to complex problems[17,74]. This approach has successfully been used in the literature to evolve controllers for swarms of flying robots[38,85,88].

However, evolved controllers are often constrained to scenarios for which they were evolved[32]. Instead, robot controllers for real-world applications must often operate across different scenarios (different environments, different number of robots, different tasks, etc.). Furthermore, evolved controllers such as neural networks, electronic circuits and programs are typically difficult to adapt to different scenarios. This makes it difficult to evolve robots that are rapidly and robustly usable out-of-the-box in various unexpected situations. One solution is to evolve a different controller before each operation which usually is too time consuming. Moreover, one could imagine evolving a controller that takes as an input mission parameters. Unfortunately, it is currently challenging to find optimal controllers for multi-objective systems[101]. Finally, controllers could be evolved online, provided that robots can be given some time to fail and learn[32]. This is not necessarily obvious for applications involving flying robots where failure might result in destructive crashes.

Rather than optimizing the evolutionary process to obtain application-ready controllers for flying robots, we propose to reverse engineer basic behaviors found by evolution. Reverse engineering consists in systematically analyzing how an evolved controller works and then capturing its main functionalities in a hand-designed controller. Resulting controllers have the advantage of being easier to understand than evolved controllers and can be mathematically pa-

rameterized for a variety of scenarios. Reverse engineered controllers are then implemented in simulation or reality to validate their capacity to reproduce the desired basic behaviors found by evolution. To show the use of basic behaviors as building blocks for a wide variety of aerial applications, we extend their use to scenarios which go beyond the scope of the evolutionary scenario. Possible mechanisms to combine basic behaviors for swarming have been studied in the past by Mataric[64]. The methodology from evolution to human-usable building blocks for the design of aerial swarming is summarized in Fig. 1.3.

This methodology is applied to a test scenario aimed at deploying emergency communication networks for rescuers using flying robots (as shown in Fig. 1.4). Communication is essential in the case of a disaster to coordinate relief efforts and provide connectivity to victims in areas where the communication infrastructures have been damaged[29,52,63,76]. Flying robots have the advantage of rapidly overcoming difficult terrain and providing unobstructed wireless communication by autonomously placing themselves in the environment.

This scenario was chosen because it requires robots to perform a collection of basic behaviors that can be useful across a wide variety of aerial swarm applications. Individually, each robot must be able modulate its global direction and speed based on communication input. Swarms of robots should move coherently in groups to avoid getting lost, search an area, and maintain a communication network between themselves and people on the ground.

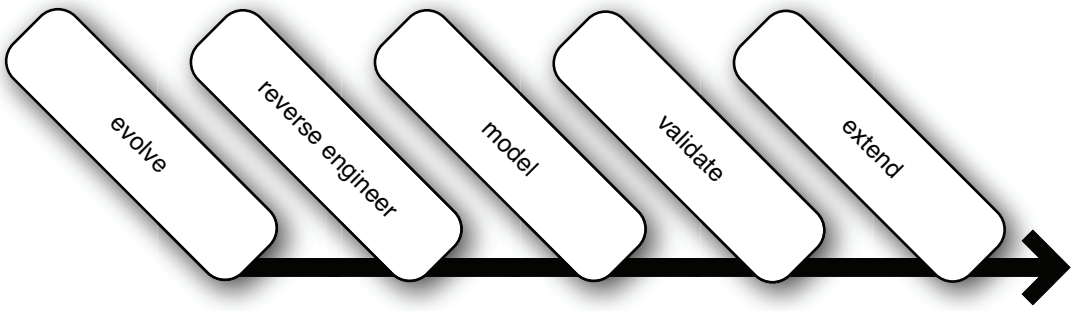


Figure 1.3: Methodology to design basic behaviors for the deployment of swarms of flying robots.

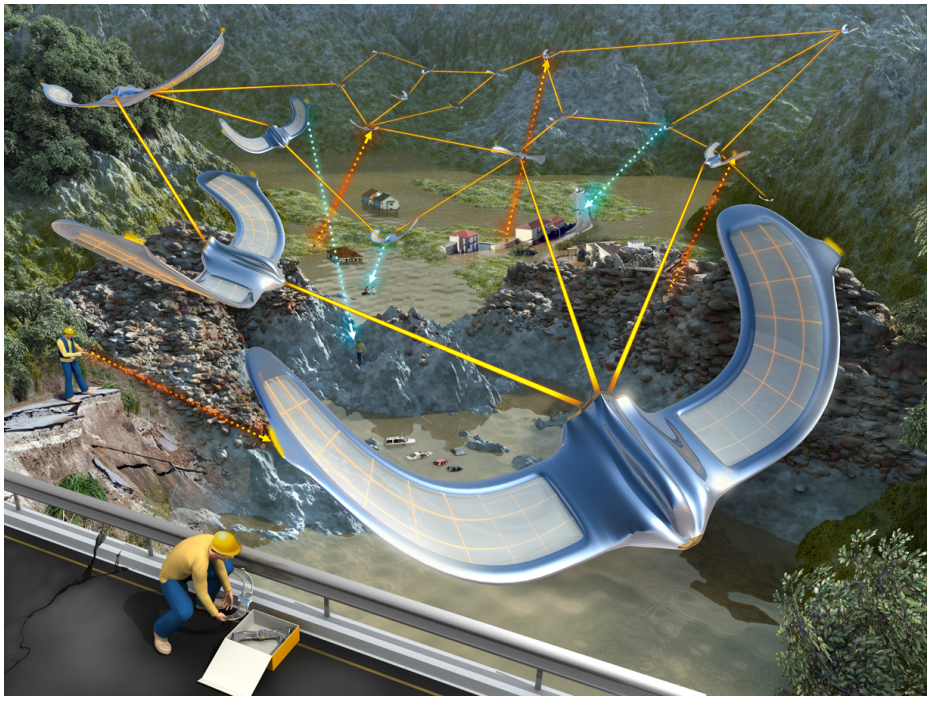


Figure 1.4: Artistic view of the use of a group of flying robots for establishing communication networks between rescuers and victims in a flood scenario.

1.4 Contributions

The main contributions of this thesis are threefold and of equal importance:

Communication-based Behaviors The first contribution is the use of communication as main input to control swarms of flying robots instead of global or relative position. Considered robots have challenging motion dynamics since they can not stop or turn on the spot. Results include basic behaviors to control the motion of individual robots or perform group motion, area coverage and communication relay. These behaviors can serve as basic building blocks to achieve a wide variety of aerial swarm behaviors. A possible extension is then identified to allow robots to cope with windy environments.

Methodology The second contribution is the systematic approach to develop swarm controllers for flying robots. Artificial evolution is used to automatically discover simple, efficient and original controllers for fixed-wing robots. Controllers are then reverse engineered, analyzed, modeled, validated and extended

to new applications.

Crossing the Reality Gap The third contribution is the development of algorithms suitable for the deployment of large aerial swarms in reality. These swarm controllers are simple, reactive and can be implemented on safe, lightweight and low-cost robots that do not require position sensors to function.

The feasibility of our approach is demonstrated using a testbed of up to 10 robots developed during the project* (Fig. 1.5). Aiming towards a specific application and target platform is essential to design systems that can be transferred to reality. To the best of our knowledge, this is the largest autonomous aerial swarm ever demonstrated outdoors.†

Figure 1.5: Aerial testbed composed of 10 fixed-wing flying robots produced during this project.

*Platforms were developed by Severin Leven. They are now commercialized by SenseFly LLC.

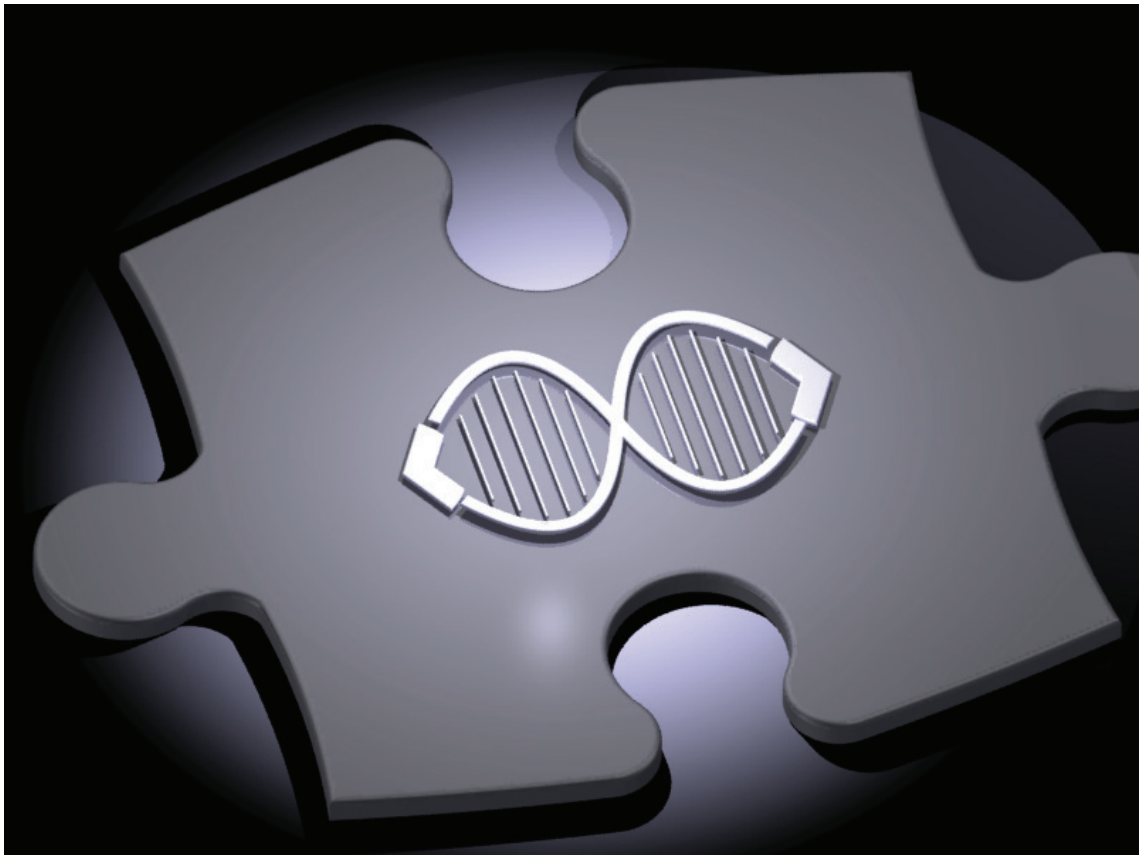
†Details of our experimental setup can be found in appendix B.2 and in videos on our project page (http://lis.epfl.ch/smavs/).

1.5 Structure

In Chapter 2, artificial evolution is used to discover a variety of swarm behaviors suitable for positionless fixed-wing robots. Chapters 3 through 6 then aim at reverse engineering the best evolved controller with each chapter focusing on one behavior. In particular, Chapter 3 copes with the problem of modulating the global motion speed and direction of fixed-wing robots using communication as an input. Chapter 4 looks at how to make robots move coherently in groups. Chapter 5 studies solutions to have a swarm cover an area in search applications. Finally, Chapter 6 focuses on mechanisms to maintain and optimize communication relays. The structure of these chapters follows the methodology developed in this thesis and described in Fig. 1.3. Lessons learnt from reverse engineering evolved controllers are then extended to scenarios where robots need to avoid being pushed away by wind in Chapter 7. Finally, in Chapter 8 we conclude and discuss current limitations and future directions of this project.

This thesis contains two appendices. The first looks at an alternative approach to design controllers for flying robots by taking inspiration from swarming in nature. More specifically, inspiration is taken from army ants to deploy communication networks in disaster areas. This approach although promising, does not fit the evolutionary framework of this thesis. The second appendix contains information on the software and hardware used during this project and the experimental setup to deploy 10 autonomous flying robots.

2 Evolved Solution



Abstract

The main contribution of this chapter is the use of artificial evolution to discover novel swarm controllers for positionless fixed-wing flying robots.

Designing swarm controllers is challenging because there is often no obvious relationship between the behaviors of the individual robots and the emergence of a desired swarm behavior. Furthermore no precedent exists in designing positionless controllers for robots with motion constraints of fixed-wing platforms. To overcome these challenges, artificial evolution is used to automatically discover simple, efficient and unthought-of swarm controllers in simulation. This method is successful in discovering basic robot behaviors for a challenging scenario aimed at deploying communication networks for rescuers in disaster areas. Basic behaviors, including individual robot motion, group motion, area coverage and communication relay, have the potential to serve as building blocks for the design of future swarm applications.

This chapter is based on:

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1) pp. 21-32.

2.1 Background

Artificial evolution is at its best when it allows scientists to discover solutions that had not yet emerged through traditional approaches. Examples include the evolution of electronic circuits[100], artificial life-forms[62], art or music[87] and experiments aimed at understanding evolved behaviors in animals[33,68,107].

However, for evolution to be successful it is necessary to have some insight into the design of evolutionary experiments. For swarm applications, this means understanding what parameters can lead to the evolution of cooperative behavior[45]. As a starting point, scientists have turned to the biological systems that inspired evolutionary robotics. Over billions of years, animals have evolved to solve a variety of collective tasks from navigation to collective hunting, which evolutionary biologists have studied extensively. By tapping into decades of research in biology, it has recently been discovered what conditions can lead to the evolution of cooperation in animals and robots. More specifically, Floreano et al.[33] and Waibel et al.[107] have found that cooperation can evolve either if robots in a group are clones (homogenous swarms) or if they share their performance scores with other members of their group. The highest performance in cooperative tasks is achieved when both these conditions are true.

Building on this insight, we aim at discovering novel communication-based controllers for swarms of flying robots that address the particularities of our system in terms of limited sensing (no position) and motion constraints.

2.2 Method

Artificial evolution requires us to select an experimental scenario, a controller architecture, a fitness measure and evolutionary parameters.

2.2.1 Experimental Setup

A test scenario in 2D simulation is considered, in which a group of robots must deploy and maintain a wireless communication network between two rescuers in a disaster area[47]. Twenty robots are launched by one rescuer at a rate of 1 every 15 s ($\pm$ 7.5) and the group must then cooperate to find a second rescuer that is positioned within a $\pm$ 30° angle of a predefined search direction and a

distance of 500 ± 50 m (Fig. 2.1). Once the communication link between the two rescuers is established, it must be maintained until the end of the mission, which lasts a maximum of 30 minutes. The robots are simulated using a physics engine in which we implement a first-order dynamics model of a fixed-wing robot that flies at a speed of 14 m/s and turns with a minimum radius of 18 m. The communication range of robots and rescuers is of maximum 100 m with added noise between 90 m and 100 m. Additional information concerning the simulator can be found in Appendix B.1.1.

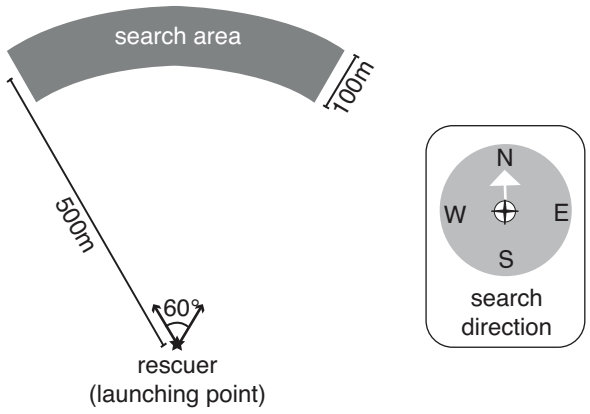


Figure 2.1: Scenario environment as defined in our application. The swarm should be capable of finding any rescuer placed within a $\pm 30^\circ$ angle of the desired search direction and within 500 ± 50 m of the launching rescuer.

2.2.2 Neural Controller

Each robot is controlled using a feed-forward neural controller shown in Fig. 2.2 consisting of three inputs, four hidden neurons and one output controlling the turn rate of the robot (speed and altitude are constant)*. The first input to the network is the heading of the robot given by a magnetic compass. The second and third inputs are the number of network hops separating the robot from the two rescuers (high values indicate that the robot is disconnected), where network hops can be seen as the number of times a message sent from a rescuer needs to be forwarded from one robot to another before it reaches the desired robot.

*We sequentially tested neural architectures with zero to four hidden neurons. Architectures with four hidden neurons were found to yield swarm controllers with the highest fitness.

Hop values are calculated in a decentralized fashion as explained in Appendix B.1.1. These controller inputs were shown to yield best results based on extensive studies of possible variants in previous work by Hauert et al.[44]. The genome of each robot consists of the 21 synaptic weights of the neural network, each represented by 8 bits, making a total genome size of 168 bits.

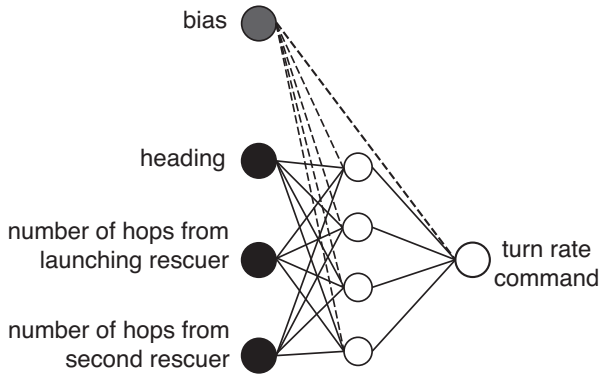


Figure 2.2: Artificial neural network architecture used to control fixed-wing robots.

2.2.3 Genetic Algorithm

To favor the evolution of cooperation following the findings of Floreano et al.[33] and Waibel et al.[107], we use homogeneous swarms and apply group-level selection. Evolutionary parameters are taken from work by Floreano et al.[33]. A population of 100 genomes is used, which are cloned 20 times to construct 100 groups of 20 robots each. The performance of each genome is computed based on the performance of the group of 20 robots. Group performance here represents the connectivity of the relay between both rescuers, or the minimum number of robots that need to fail for the communication between the rescuers to break, averaged over 30 minutes and 10 missions. In order to put additional pressure against the loss of robots, trials during which robots are disconnected from the launching rescuer for more than 30 s are assigned a fitness of 0. This performance measure favors the rapid creation of communication pathways between rescuers and the robustness of the network over time. After ranking the genomes according to the measured performance of the group, the 10 best genomes in

the population are copied to the new population (elitism) and cloned to make groups of 20 robots each. The remaining population is generated by repeatedly selecting two random individuals from the best 30% of the genomes, applying one-point crossover to the pair with a probability 0.2 and then mutating the newly created individual with a probability of 0.01 per bit, and cloning it 20 times.

2.3 Results

The evolutionary run during which the controller with the highest fitness was found can be seen in Fig. 2.3. Out of 15 evolutionary runs, 2 were able to find controllers displaying similar strategies as the best individual while the remaining 12 evolutionary runs converged to a local optimum. This shows that the evolutionary process is not straightforward. However, the main focus of this thesis is on transferring good controllers found through evolution to real-world applications and not on optimizing the evolutionary process. To this end, the performance of the best controller is quantitatively and qualitatively analyzed.

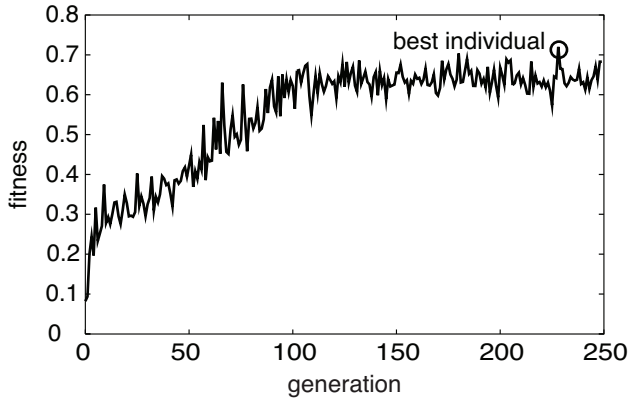


Figure 2.3: Maximum individual fitnesses for each generation of the best evolutionary run. The individual with the highest fitness was found at generation 228.

2.3.1 Performance

After evolution, the best controller was tested 1000 times on randomly positioned rescuers within the area described in Fig. 2.1. In the end, 97.5% of the

rescuers were found and only 2.24% of the 20'000 deployed robots (1000 runs with 20 robots) were permanently disconnected from the swarm. The connectivity measures of the networks over the 1000 trials are shown in Fig. 2.4 (left). The connectivity measures equal 0 during the first couple of minutes of a deployment because few robots have been launched, and the swarm has not traveled far enough to find the second rescuer. However, once the connection between the two rescuers is established, it is maintained in a robust manner, which can be seen by the fact that the connectivity measures remain stable to the end of the trials. After 30 min, 22.4% of the 1000 trials had a connectivity of zero, 74% had a connectivity of one and 3.6% had connectivity of two. Over the 1000 trials, only those where the second rescuer was not found (2.5%) displayed a constant connectivity of zero. The remaining trials maintained at worst intermittent connections, displaying varying connectivities between zero, one and two. A perfect (uninterrupted) connection between the rescuers is not required as long as the swarm remains coherent. Fig. 2.4 (right) shows statistics on the mean connectivity over 30 min trials as described in Sec. 2.2.3. The median fitness over 1000 trials is of 0.68 which is coherent with the performance of the best individual found through evolution, as shown in Fig. 2.3.

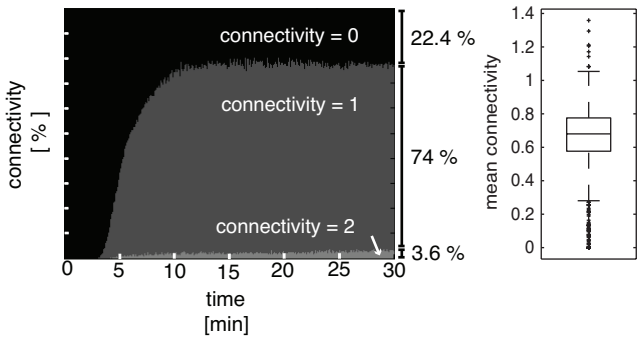


Figure 2.4: Left: Connectivity of the best evolved controller when tested one thousand times on randomly positioned rescuers within the area described in Fig. 2.1. At each time-step, the proportion of networks (out of the 1000 trials) having connectivities of 0, 1 and 2 is shown. Connectivity values above 2 were not encountered. Right: Statistics on the mean connectivity over 1000 trials of 30 min. Each box has lines at the lower quartile, median, and upper quartile values. The whiskers extend to the farthest data points that are within 1.5 times the interquartile range. Outliers are shown with a + sign.

2.3.2 Behavior

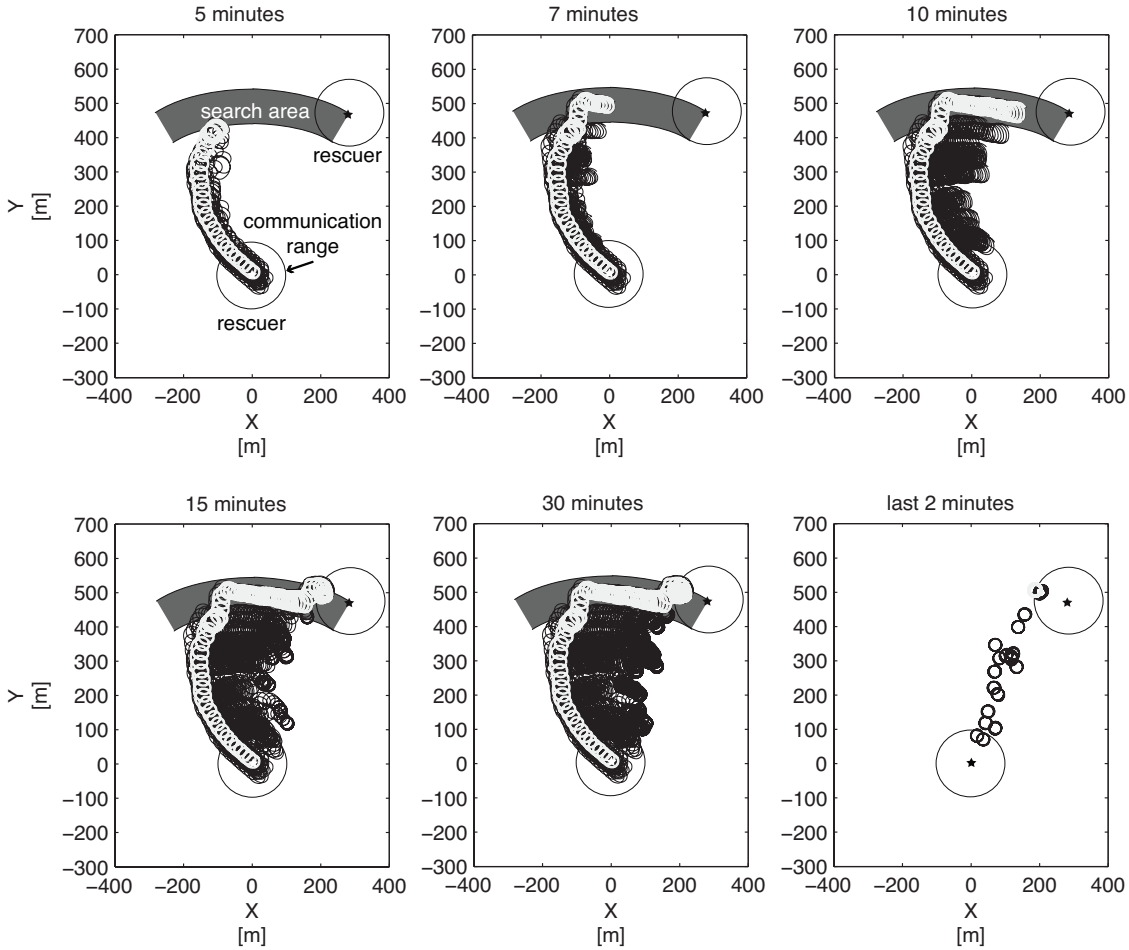


Figure 2.5: Trajectories of 20 robots in simulation with the best evolved controller (over all populations in all generations) during a 30 min mission. In this mission, the second rescuer is to the North-East of the launching rescuer to show the full extent of the chain translation displayed by the group. The trajectory of the robot launched first is shown in light grey.

An example showing the behavior of the best evolved controller can be seen in Fig. 2.5 or in a video on our website*. The strategy adopted by the swarm consists in forming a tight chain of robots which grows as long as additional robots are launched. Robots forming the chain are able to slowly move in a common direction. Once all robots have been launched, they synchronize their heading

*http://lis.epfl.ch/smavs

and the entire chain then shifts along the communication border of the launching rescuer, covering the search area from West to East until the second rescuer is found. The communication link between the two rescuers is maintained by having all robots turn on the spot with the smallest possible radius given the dynamics of the platform.

As expected, the strategy found through evolution is novel, efficient and unthought-of given the limited amount of control inputs that were provided to the system.

2.4 Limitations

Robot controllers for real-world applications must often adapt across different scenarios depending on the needs of a given operation (different environments, different number of robots). For this purpose we characterize the limitations of the obtained swarm controller with respect to variations in swarm size, communication range, robot launch interval and robot speed.

Fig. 2.6 shows the effect of each parameter variation on the mean connectivity over 30 min trials. Statistics are produced over 1000 trials with rescuers randomly positioned within the area described in Fig. 2.1. For each set of experiments, only one parameter was changed, the remaining parameters being set to the original values described in the experimental setup (Sec. 2.2).

Results show a decrease of performance by at most 10% with respect to the original median performance of 0.68 (Fig. 2.4, right) for any of the following variations in setup parameters:

- variations in swarm size from 17 to 30 (30 being the maximum tested)
- variations in communication range from 100 m to 200 m (200 m being the maximum tested)
- variations in launch interval from 10 s to 20 s
- variations in robot speed from 10 m/s to 16 m/s

This shows that the evolved controller is robust to bounded modifications in scenario. However, certain scenarios result in a large decrease in performance or even complete failure of the mission. A decrease in performance is to be

expected for scenarios that largely differ from the ones used during evolution. Such new scenarios would typically require re-evolving a new controller, which is time consuming and unpractical for systems that must be usable out-of-the-box in real-world situations.

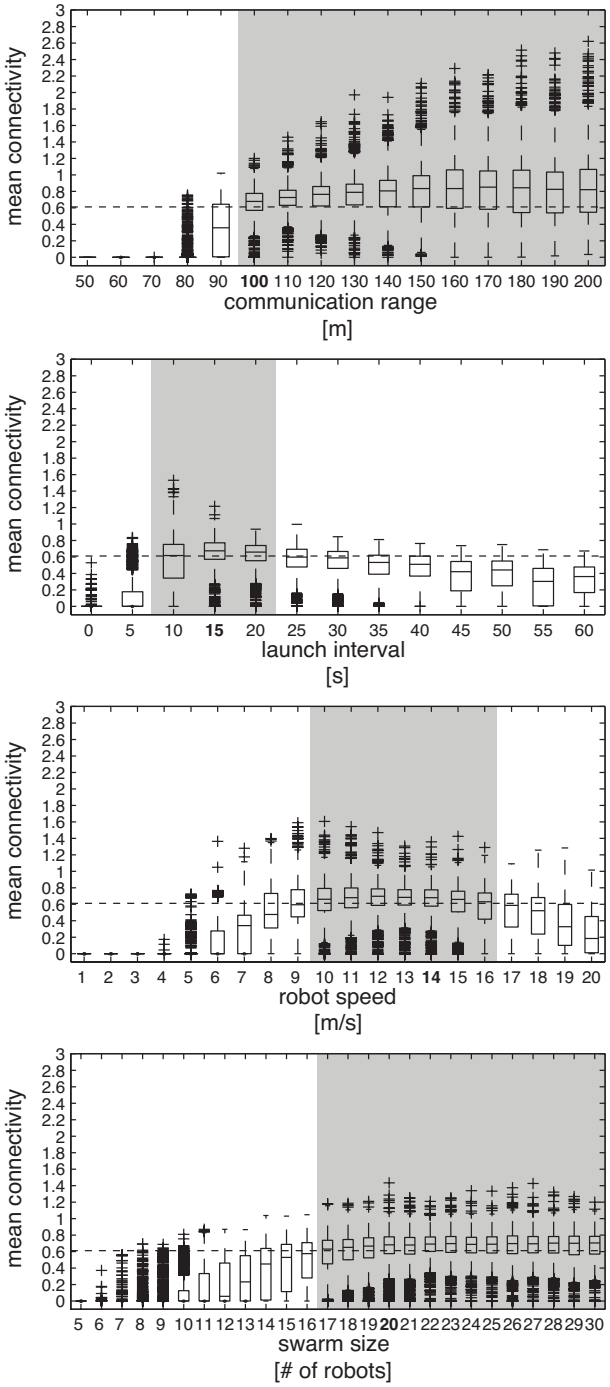


Figure 2.6: Effect of varying a parameter of the experimental setup on the mean connectivity over 1000 trials of 30 min. The dashed line represents a 10% decrease in performance with respect to the original parameters shown in bold. Sectors in grey include parameter values which perform above this limit.

2.5 Conclusion

Artificial evolution is successful in automatically discovering a variety of behaviours for simulated swarms of flying robots in complex real-world applications. Controllers do not use position and are able to harness motion constraints of fixed-wing robots.

The explored scenario consists in creating a communication relay between two rescuers in a disaster area. In the best evolved strategy, robots can modulate the speed and direction of their motions based on heading and communication input. This allows robots launched by a rescuer to form chains in a specific direction. Robots within these chains are able to synchronize their headings and move coherently as a single group. Synchronized chains then translate along the communication border of the launching rescuer, thereby covering a deterministic area in search for a second rescuer. Once a connection is established between both rescuers it is maintained and optimized by the robots. Individual robot motion, group motion, area coverage and communication relay are basic building blocks that could serve a variety of swarm applications.

However, evolved controllers are not able to perform well when tested on scenarios for which they were not intended. For this reason, the following chapters focus on reverse engineering the best evolved controller so as to capture its essence in hand-designed controllers that can serve as building blocks for the design of swarm controllers in the future.

3 Individual Motion

Abstract

The main contribution of this chapter is the identification of basic principles governing the motion of individual robots.

Swarm behaviors emerge from the motions performed individually by each robot. To avoid the need for position information and to accommodate motion constraints of fixed-wing robots, controllers modulate the turn rate of a robot based on its heading and communication. Designing such motions is done following the systematic approach proposed in this thesis. Starting from the evolved controller described in Chapter 2 we focus on analyzing the motion of individual robots. This analysis leads to reverse engineered controllers that can be used to change the global direction and advancement speed of robots or have them translate along the communication border of a ground user in a predictable manner. We then validate the controllers in theory and reality for a large variety of controller parameters. Finally we show how discovered principles can be extended to explore an area around a user or patrol along its communication border.

This chapter is based on:

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1) pp. 21-32.

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Reverse-engineering of Artificially Evolved Controllers for Swarms of Robots. *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 55-61.

3.1 Background

Motions of flying robots in the literature typically rely on position information. More specifically, robots that have access to GPS use waypoint navigation[43,72] whereas controllers based on relative positioning often rely on attraction or repulsion interactions[6,21,24,53,78]. For the latter, motion constraints of fixed-wing robots are seldom taken into account.

The most complete controller that accommodates the motion constraints of fixed-wing robots can be found in work by Lawrence et al.[58]. In this work robot motions can be modulated to meet any requirement in terms of global speed and direction. The controller outputs the heading that steers a robot asymptotically towards the desired robot motion using vector fields. The heading error determined with a GPS or compass sensor is then used to command the turn rate of the robot. This approach however requires GPS.

Controllers for flying robots that react to communication information can be found in work by Basu et al.[6] who studied the use of a flock of robots as an ad-hoc network in the sky that could adapt to its users on the ground. Robot motions included attraction and repulsion to neighboring robots as well as attraction to good positions above users on the ground and a "random walk" rule to repair breaches in the network. Other work by Lawrence et al.[57] generates information energy potentials on a map that capture the quality of measurements made by other robots, their energy loss due to motion and the quality of the ad-hoc network they form. The resulting gradients are then navigated by the robots to map toxic plumes while maintaining a coherent communication network to a central processing unit on the ground. Both studies were conducted in simulation and position information was the dominant sensor used.

Instead, communication-centered motions for flying robots have been investigated by Dixon et al.[27] who used the signal-to-noise ratio (SNR) of incoming radio signals to determine if a robot is in an optimal location to communicate with a user on the ground or with other robots. However, experiments on real flying robots relied on position information to function[35]. Furthermore, SNR indications are not always made available by radio modules. Indeed, SNR information is treated at the driver level and whether or not this information is passed up to the robot controller depends on which off-the-shelf radio module

is used. In addition, even if the SNR is given, the source of the signal might not be specified. This is problematic when multiple emitters are present.

Work by Daniel et al.[23] considers simulated swarms of flying robots that must cover an area while ensuring connectivity among the robots. To achieve this, robots react to the RSSI from neighboring robots whose positions are known. In addition to challenges in using signal strength measurements described above, considered models do not consider motion constraints of physical flying robots.

Overall, examples of positionless controllers that react to communication information are scarce. The most noticeable example was designed by Nembrini et al. to maintain coherent groups of ground robots that could move towards a beacon[73]. In this work, robots go straight as long as they were connected to the swarm. When disconnected, robots turn around by 180° and then continue straight to reconnect to the swarm. Such behaviors however do not meet the motion constraints of fixed-wing flying robots that are not able to stop and turn on the spot.

By reverse engineering evolved controllers we aim towards novel motion control for flying robots that addresses challenges in using robots with limited sensing (no position) and constrained motion.

3.2 Evolved Behavior

In the best evolved strategy from Chapter 2, robots do not move straight. Instead, they adopt different turn rates, depending on their heading and whether they are connected (even indirectly) to the rescuers.

The first observation is that robots perform trajectories where they are continuously turning while advancing at a constant speed. The overall trajectory of the robot can be summarized by a vector which gives its global direction and speed. Fig. 3.1 shows examples of such trajectories using the best evolved controller from Chapter 2. Fig. 3.2 shows how such trajectories are generated by modulating the turn rate of the robot based on its heading.

The second observation is that communication impacts the parameters of the directed trajectories in terms of speed and direction. The effect is that robot trajectories are steered and speed regulated by communication. This is shown in Fig. 3.1 where robots that are connected to the launching rescuer (low hop

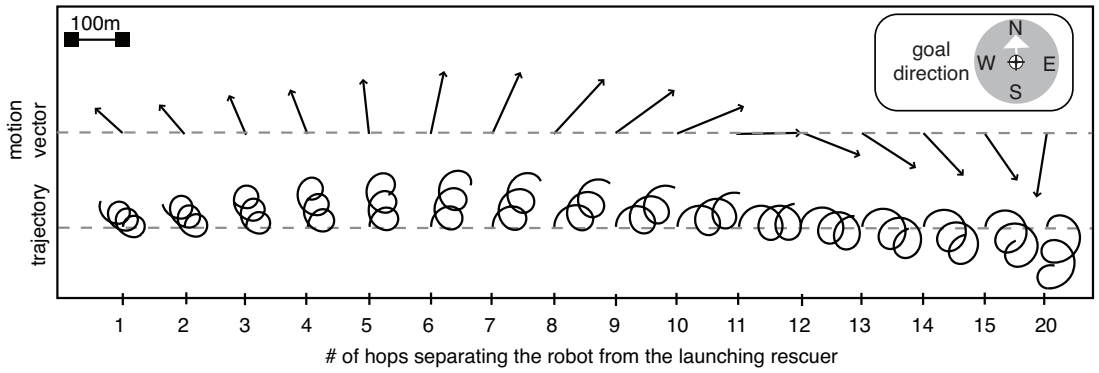


Figure 3.1: Effect of the number of hops which separate the launching rescuer from a robot on its trajectory. Here, we plot the trajectories of the best evolved controller from Chapter 2 over 30 s. Robots were never connected to the second rescuer during these experiments.

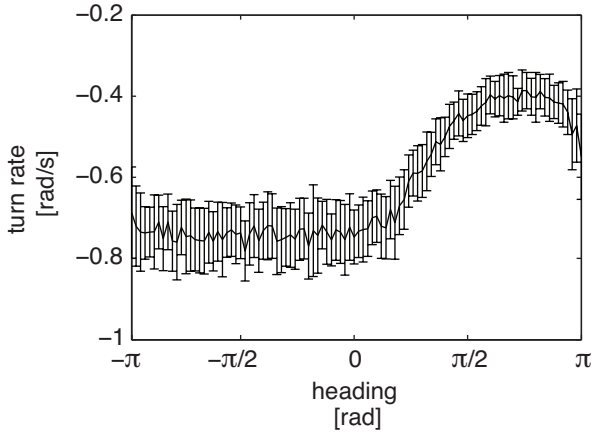


Figure 3.2: Effect of the robot's heading on its turn rate when the other inputs to the neural controller are maintained constant. Experiments were conducted in simulation using the best evolved controller from Chapter 2.

counts) are directed to the North whereas robots disconnected (hop count of N=20) have trajectories directed to the South.

In addition, robots perform trajectories with different average turn rates depending on communication. In Fig. 3.1, robots that are connected to the launching rescuer display high turn rates while the disconnected robots perform low turn rates. The effect of this is that robots can translate along communication borders as shown in Fig. 3.3. A communication-border, in this thesis, is assumed

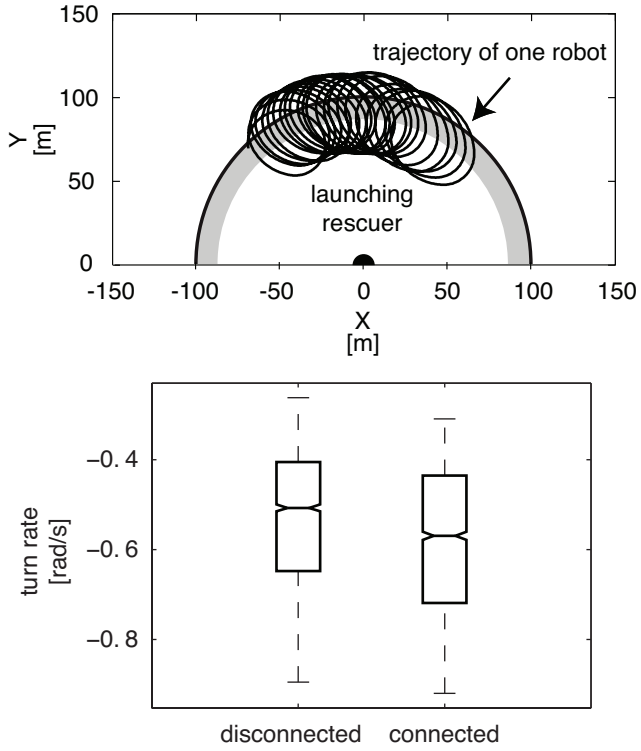


Figure 3.3: The top figure shows the trajectory of the best evolved robot in simulation moving along the communication border of the launching rescuer (the grey area shows the noisy communication zone in simulation). This translation behavior, along the communication border, is induced by a change in turn rate when connected and disconnected from the launching rescuer. A box plot in the bottom figure shows that the turn rates when connected and disconnected are significantly different ($p<0.01$).

to be a continuous edge that separates an area where robots and users are within communication range or not. Notice that this edge can be the result of a filter.

3.3 Reverse Engineered Controller

There is often a tradeoff between designing reverse-engineered controllers that exactly copy behaviors found through evolution and making controllers that are easy to implement and understand by a human designer. High-fidelity reverse-engineering consists in designing controllers that produce near-identical output for a set of sensory input as the evolved controller.

As an example, to produce nearly identical trajectories as those shown in Fig. 3.1, one can use a controller based on prolate cycloids which are trajectories described by a fixed point at a radius $b > a$, where a is the radius of a rolling circle. The entire prolate cycloid can be rotated by an angle α to suit any directional needs. To reach high-fidelity reverse-engineering, evolved trajectories are analyzed to determine parameters a, b and α as a function of the number of hops separating the robot from the launching rescuer. Results of suitable functions are given in Fig. 3.4.

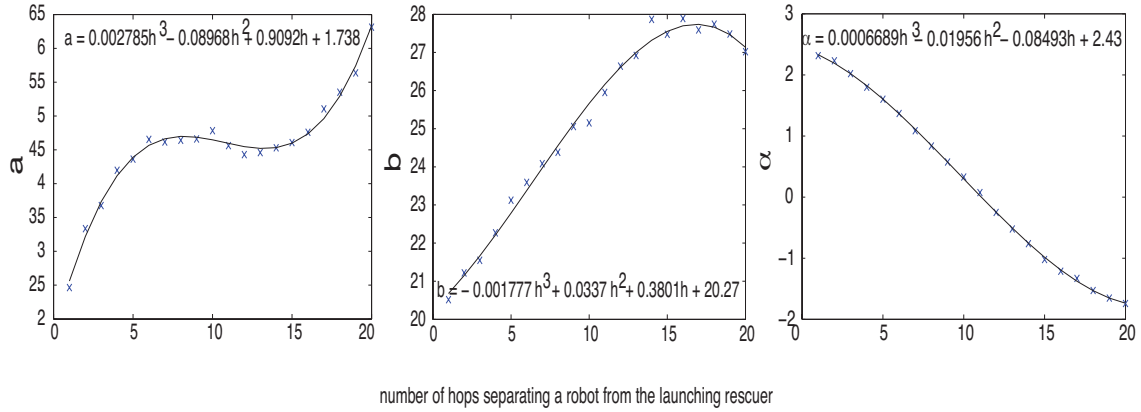


Figure 3.4: Controller parameters a, b and α as a function of the number of hops separating the robot from the launching rescuer h.

However, high-fidelity controllers might be as difficult to understand and use by a human-engineer as the initial neural controller.

On the other side of the spectrum, low-fidelity reverse-engineering takes inspiration from the general behavior of the robot to design new controllers. One example would be to extract the general behavior "robots can translate along communication borders" and design a different type of controller to achieve this. The disadvantage of such an approach is that the challenge of understanding how each robot should move to achieve a desired emergent behavior remains while good solutions found through evolution are lost.

Instead, throughout this thesis we aim at balancing the need for reverse-engineered controllers that are usable by human engineers while still capturing the ingenuity of evolved solutions.

In our mid-fidelity controllers, robot motion is generated by modulating overtime the turn rate ω of a robot with respect to its heading. For this purpose we approximate the continuous function shown in 3.2 by the step function shown

in Fig. 3.5. Following this function, robots perform a fixed turn rate ω_1 when the angle $\widehat{h_{lim}h}$ between a predefined heading limit h_{lim} and their current heading h is positive or a fixed turn rate ω_2 when this angle is negative (angles are measured between $-\pi$ and π) as described in the following controller:

$$\omega_{\omega_1,\omega_2,h_{lim}}(h) = \begin{cases} \omega_1 & \text{if } \widehat{h_{lim}h} > 0 \\ \omega_2 & \text{otherwise} \end{cases} \tag{3.1}$$

This controller is more intuitive to understand and analyze than the evolved neural network. Indeed modifying ω_1 or ω_2 directly translates into a predictable change in robot motion, whereas changing a weight in a neural network might have many unpredictable effects.

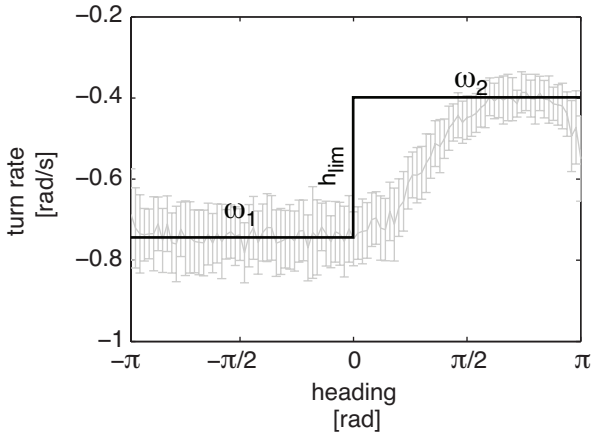


Figure 3.5: Effect of the controller in Eq. 3.1 represented by a step function on the turn rate of a robot. The step function is an approximation of the evolved solution shown in grey.

Furthermore, as seen in Fig. 3.1, communication affects the global speed, direction and turn rate of trajectories described by Eq. 3.1. For the sake of simplicity, the large range of hop count values can be summarized by two communication states, namely "connected" (hops between the launching rescuer and the robot smaller than N=20) or "disconnected" (hop values of N=20). This produces controllers that are easier to analyze because of the fewer and constant number of states when compared to states based directly on hops. In this manner, when a robot is connected to the launching rescuer, it performs a controller of the form $\omega_{\omega_1,\omega_2,h_1}(h)$ given by Eq. 3.1. When disconnected, the robot performs a different trajectory described by the controller $\omega_{\omega_3,\omega_4,h_2}(h)$. Therefore,

for each communication condition c the turn rate ω of the robot can be described by $\omega(h, c)$ given below:

$$\omega(h,c) = \begin{cases} \omega_{\omega_1,\omega_2,h_1}(h) & \text{if } c = 1 \text{ (connected)} \\ \omega_{\omega_3,\omega_4,h_2}(h) & \text{if } c = 0 \text{ (disconnected)} \end{cases} \tag{3.2}$$

3.4 Model

A precise understanding of the effect of controller parameters (ω_1, ω_2, h_1, ω_3, ω_4, h_2) on the motion of robots is needed to adapt reverse engineered controllers to different real-world scenarios. To this end, we focus on characterizing robot motions in terms of global direction $\overline{d}$ of the robot trajectories and their global speed $\overline{v}$.

In particular, the controller described in Eq. 3.1 produces trajectories that advance perpendicular to h_{lim} and in the direction $\overline{d}$ given by the vector between points A and B which describe the start and end of the trajectory shown in Fig. 3.6. The trajectory corresponds to a full revolution of the robot. The global speed $\overline{v}$ of the robot motion performed by a robot advancing at speed v is then given by the distance between points A and B ($|AB|$) over the time it takes to go from one point to the other ($t(AB)$).

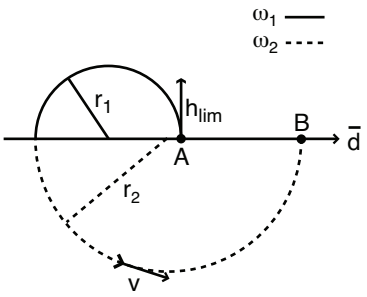


Figure 3.6: Theoretical trajectory performed by a robot implementing the controller in Eq. 3.1. Symbols are used to analyze the global speed and direction of the robot trajectory using Eq. 3.3 and 3.4.

This is summarized in the following equations:

$$\overline{v} = \frac{|AB|}{t(AB)} \tag{3.3}$$

$$\overline{d} = h_{lim} + sgn(AB)\frac{\pi}{2} \tag{3.4}$$

where

$$AB = 2(r_1 - r_2) \tag{3.5}$$

$$t(AB) = \frac{\pi(r_1 + r_2)}{v} \tag{3.6}$$

and

$$r_i = |\frac{v}{\omega_i}| \tag{3.7}$$

Communication can impact robot trajectories in two ways. In the first, robots are connected or disconnected for long periods of time, during which they advance only based on heading, as described in Eq. 3.1. In these cases, analysis from Eq. 3.3 and 3.4 apply. In the second, robots oscillate between being connected and disconnected on a short timescale. These frequent changes typically make robots adopt different average turn rates when connected or disconnected, thereby producing motions that translate along the border of communication of the rescuer (Fig. 3.3). For the sake of simplicity, we consider a scenario where the communication-based controller pushes the robots towards the communication border. That is, robots that are connected to the rescuer move away from it while robots that are disconnected move towards it. Assuming a border pointing in direction d_{com} (with the connected area to the left of the vector) we have the following communication-based controller:

$$\omega(h, c) = \begin{cases} \omega_{\omega_1,\omega_2,d_{com}}(h) \text{ if } c = 1 \text{ (connected)} \\ \omega_{\omega_3,\omega_4,d_{com}+\pi}(h) \text{ if } c = 0 \text{ (disconnected)} \end{cases} \tag{3.8}$$

Robots performing this controller converge to stable trajectories regardless of their starting heading as will be explained in the next chapter. Stable trajectories are such that robots return to their initial heading after every connection and disconnection. Such a trajectory, whose start and end are labeled by symbols A and B, is shown in Fig. 3.7 (left).

The global speed $\overline{v}_{com}$ of the robot motion advancing in direction $\overline{d}_{com}$ is then given by the distance between points A and B ($|AB|$) over the time it takes to

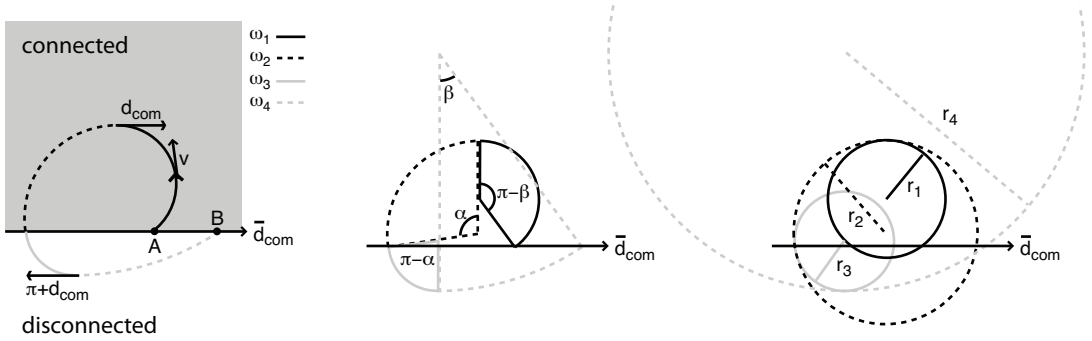


Figure 3.7: Theoretical trajectory performed by a robot implementing the controller described in Eq. 3.8. Symbols are used to analyze the global speed and direction of the robot trajectory in Eq. 3.9 through 3.10

go from one point to the other ($t(AB)$). $\overline{v}_{com}$ and $\overline{d}_{com}$ can be modeled using the equations described below and symbols shown in Fig. 3.7 (center and right).

$$\overline{v}_{com} = \frac{|AB|}{t(AB)} \tag{3.9}$$

$$\overline{d}_{com} = h_{com} + \frac{\pi}{2} - sgn(AB)\frac{\pi}{2} \tag{3.10}$$

Knowing that stable trajectories are such that the heading at point A must be identical to the heading at point B, we have that:

$$AB = -r_1 sin(\pi - \beta) - r_2 sin(\alpha) + r_3 sin(\pi - \alpha) + r_4 sin(\beta) \tag{3.11}$$

$$t(AB) = |\frac{r_1(\pi - \beta) + r_2\alpha + r_3(\pi - \alpha) + r_4\beta}{v}| \tag{3.12}$$

where

$$r_1 + r_1 cos(\beta) = r_2 - r_2 cos(\alpha) \tag{3.13}$$

$$r_3 + r_3 cos(\alpha) = r_4 - r_4 cos(\beta) \tag{3.14}$$

leading to

$$\alpha = cos^{-1}(\frac{2r_1r_4 - r_1r_3 - r_2r_4}{r_1r_3 - r_2r_4}) \tag{3.15}$$

$$\beta = cos^{-1}(\frac{2r_2r_3 - r_1r_3 - r_2r_4}{r_1r_3 - r_2r_4}) \tag{3.16}$$

and

$$r_i = \frac{v}{\omega_i} \tag{3.17}$$

	ω_1	ω_2	h_{lim}
a	-0.7	-0.1	$-\pi/2$
b	-0.6	-0.2	$-\pi/2$
c	-0.5	-0.3	$-\pi/2$
d	-0.3	-0.5	$-\pi/2$
e	-0.5	-0.3	$-\pi/2$
f	0.3	0.5	$-\pi/2$
g	0.5	0.3	$-\pi/2$
h	-0.5	-0.3	$-\pi/2$
i	-0.5	-0.3	π
j	-0.5	-0.3	$\pi/2$
k	-0.5	-0.3	0

Table 3.1: Parameters of the controller described by Eq. 3.1 used in Fig. 3.8 and Fig. 3.9

3.5 Validation

We aim at demonstrating the transfer of reverse engineered controllers to reality and the adequacy of the developed model. Fig. 3.8 and 3.9 show robot motions generated using controllers in Eq. 3.1 in theory (trajectories follow exactly that of the mathematical description) and reality for different sets of parameters described in Table 3.1 using a single flying robot. Parameters were chosen to demonstrate the capability of the developed controller in changing the global speed and direction of robot trajectories. Trajectories in reality lasted 60 s and were corrected for wind based on wind measurements taken on the platform (details on wind measurements can be found in Appendix B.2.1). Results from theory and reality are qualitatively and quantitatively comparable in that changing controller parameters results in similar changes in motion. More specifically, predictions are compared to measured results in reality in Table. 3.2.

Finally, motions resulting from communication-based controllers described in Eq. 3.8 are shown in theory (Fig. 3.10) and reality (Fig. 3.11) using a single flying robot. As before, trajectories were post-processed to remove the effect of wind. Communication with a straight border was simulated by artificially cutting communication based on the heading of the robot. Robot motions in

	$\overline{v}$		$\overline{d}$	
	theory	reality	theory	reality
a	5.7296	7.8905	π	2.7668
b	3.8197	3.8329	π	2.7326
c	1.9099	2.1651	π	2.7560
d	1.9099	1.8071	0	-0.5923
e	1.9099	2.2743	π	2.8090
f	1.9099	1.5636	0	0.5412
g	1.9099	1.9050	π	-2.8409
h	1.9099	2.1862	π	2.7992
i	1.9099	1.5384	$\pi/2$	1.1501
j	1.9099	1.9603	0	-0.4248
k	1.9099	1.8820	$-\pi/2$	-1.9326

Table 3.2: Comparison of global speed $\overline{v}$ and direction $\overline{d}$ of robot trajectories shown in theory (Fig. 3.8) and reality (Fig. 3.9).

theory and reality are qualitatively comparable although robot dynamics prevent the robot from rapidly changing from one turn rate to the other, leading to differences in the overall advancement speed (2.88 m/s predicted versus 4.22 measured) and direction of the trajectory (0 rad predicted versus 0.36 measured).

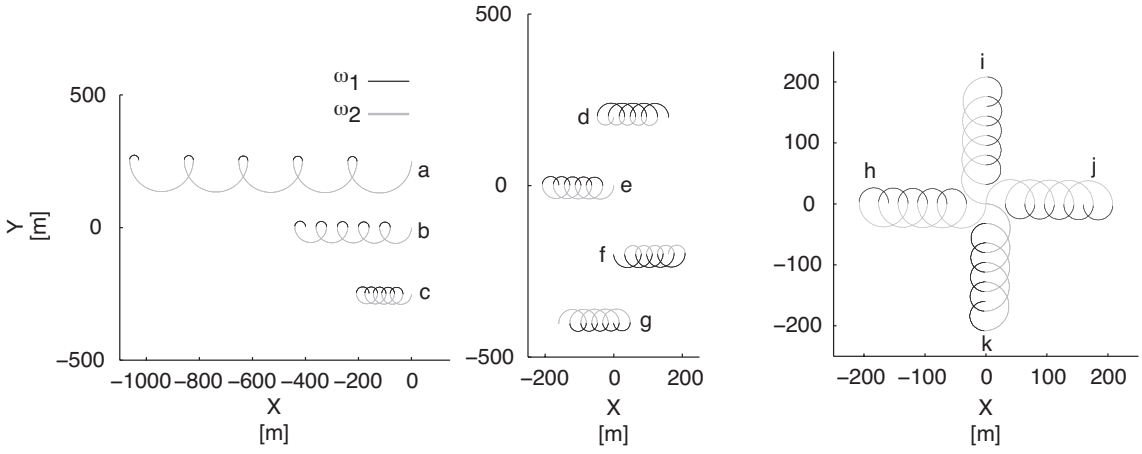


Figure 3.8: Theoretical trajectories resulting from the controller in Eq. 3.1 with parameters described in Table 3.1. The robot is launched in x=0.

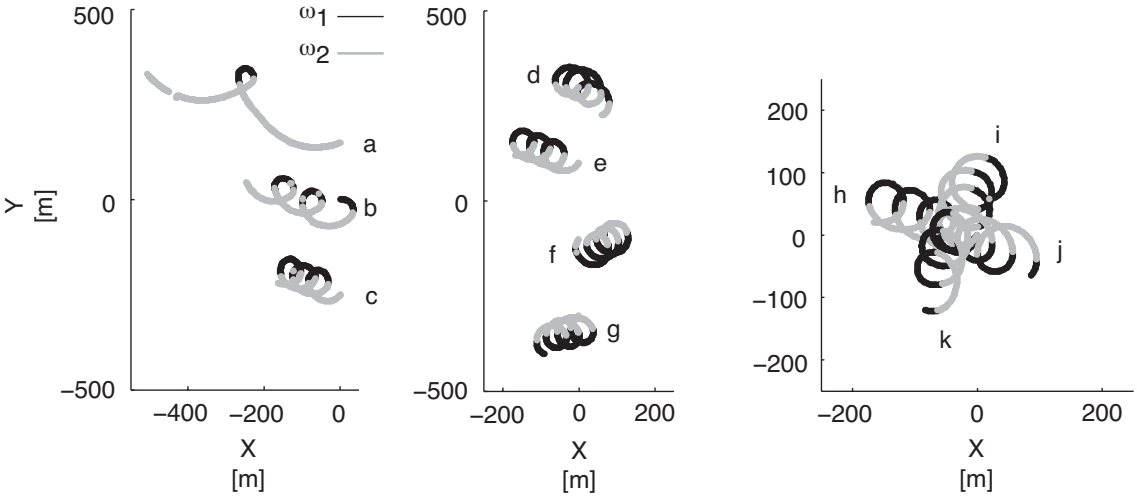


Figure 3.9: Trajectories performed in reality by a single flying robot implementing the controller in Eq. 3.1 with parameters described in Table 3.1. The robot is launched in x=0.

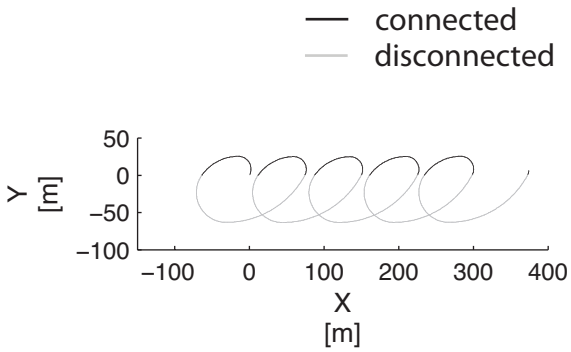


Figure 3.10: Theoretical trajectories resulting from the controller in Eq. 3.8 with ω_1=0.7, ω_2=0.2, ω_3=0.3, ω_4=0.1 and d_{com}=0. The robot is launched in (0,0).

3.6 Extensions

Artificial evolution provides a basis for the design of controllers for fixed-wing flying robots. These behaviors can be used as an inspiration to create new controllers that go beyond those found through evolution. In particular, we consider a simplified communication-based controller of the form:

$$\omega(h,c) = \begin{cases} \omega_{\omega_1,\omega_1,h_{lim}}(h) = \omega_1 \text{ if } c = 1 \text{ (connected)} \\ \omega_{\omega_2,\omega_2,h_{lim}}(h) = \omega_2 \text{ if } c = 0 \text{ (disconnected)} \end{cases} \tag{3.18}$$

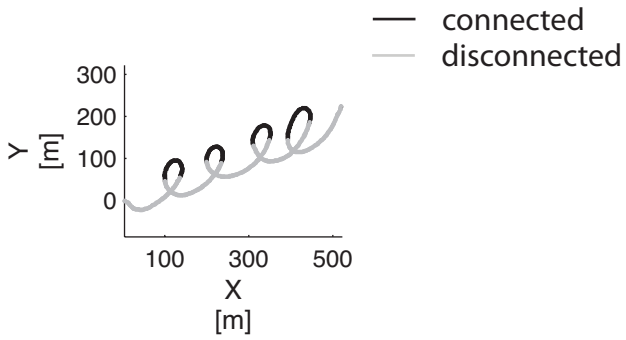


Figure 3.11: Trajectories performed in reality by a single flying robot implementing the controller in Eq. 3.8 with ω_1=0.7, ω_2=0.2, ω_3=0.3, ω_4=0.1 and d_{com}=0. The robot is launched in (0,0).

Using this controller, robots perform turn rate ω_1 when connected to a user on the ground and turn rate ω_2 when disconnected. Fig. 3.12 show results in theory using a single flying robot. Such behavior could be used alone to achieve simple behaviors such as perimeter patrolling (left and center) or exploration (right).

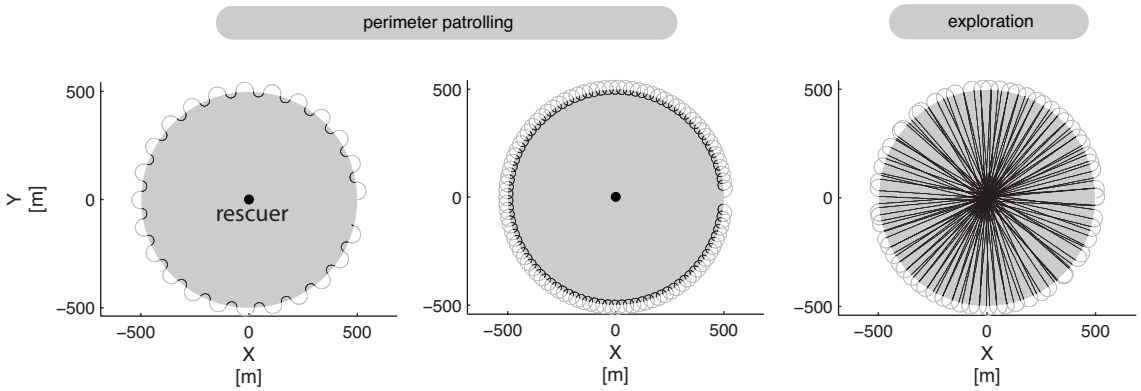


Figure 3.12: Theoretical trajectories resulting from the controller in Eq. 3.18 with ω_1=-0.5 and ω_2=0.3 for the left figure, ω_1=0.5, ω_2=0.3 for the center figure and ω_1=0, ω_2=0.3 for the right figure. The robot is launched on the border of communication with range r in (0,500) with a heading of 0. Notice that these behaviors could be used for perimeter patrolling and exploration.

3.7 Conclusion

By reverse engineering the best evolved controller from Chapter 2, we were able to identify basic principles governing the motion of individual robots. Discovered controllers do not use position information and are suitable for fixed-wing robots.

More precisely, robot motion is governed by communication and heading input. Communication inputs reflect the connection or disconnection of robots to users on the ground. They are responsible for changing the average speed, direction and turn rate of robot trajectories. Heading input then modulates the turn rate of robots so that they follow a desired trajectory.

Overall, developed controllers are able to steer and speed regulate robots and have them translate along communication borders. Controllers are then characterized in terms of motion speed and direction and validated in theory and reality using a single flying robot. Finally, to demonstrate the use of basic principles described here in a variety of applications, we apply them to a scenario where robots must explore the area around a user on the ground.

4 Group Motion

Abstract

The main contribution of this chapter is the identification of basic principles allowing for the coherent motion of groups of robots.

To avoid getting lost, robots without position information need to move as a group. In the best evolved controller discovered in Chapter 2, this is done by having the robots synchronize their headings. Synchrony can allow robots to move as if they were a single entity, potentially facilitating the sharing of sensing and communication capabilities. However, synchronizing fixed-wing robots typically requires position information, and entails frequent sensing and heavy communication.

Instead, the solution found through evolution relies on heading information and on on-off beats sent from a robot or a user using wireless communication. Robots use these beats to synchronize and modulate their speed and direction in a predictable manner. Group motion is validated in theory and using two to five physical flying robots. Principles are then extended to an application where robots in the group can become leaders that steer the swarm.

This chapter is based on:

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1) pp. 21-32.

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2010) Beat-based synchronization and steering for groups of fixed-wing flying robots. *Proceedings of the 10th International Symposium on Distributed Autonomous Robotics Systems*, in press.

4.1 Background

Synchronizing the heading of robots within a swarm can help them avoid collisions, maintain relative distance among robots for sensor fusion and favor communication[25].

Synchronizing loitering trajectories in real-time across robots while respecting commands in terms of global direction and speed is challenging. Work on formation path following for unicycle-type vehicles, that have similar motion constraints as fixed-wing robots, has so far concentrated on mathematical models and simulations built upon the assumption that robots know the precise relative position of neighbors (range and bearing) and sometimes their heading and speed[39,40,61,69]. Using this knowledge, robots continuously align their position to that of their neighbors and to the trajectory they need to follow. However, so far no results have been demonstrated with real flying robots. In addition, simulations only depict scenarios without sensor noise or with low forward speeds and limited turn rates unrealistic for fixed-wing flying robots.

Another approach for synchronization is inspired from birds flocking, and the resulting controllers described by Reynolds[6,21,53]. However, the trajectories of such flocks are typically guided by the need to synchronize. Additional behaviors, sometimes conflictual, are needed to steer and speed regulate the swarm. Furthermore, robots are required to continuously exchange their position or sense their neighbors, which is not feasible with positionless robots.

Instead, inspired from the best evolved controller presented in Chapter 2, we present a positionless strategy to synchronize and steer swarms of real flying robots that does not require memory, computation or high-bandwidth communication.

4.2 Evolved Behavior

In the best evolved solution from Chapter 2, robots synchronize their heading over time while translating along the border of the launching rescuer's communication range (Fig. 3.3). This allows for the coherent motion of groups of robots. Synchronization starts after all the robots have been launched and the chain navigates to the edge of the communication range of the launching rescuer. The swarm then alternates between being connected to and disconnected

from the launching rescuer, which generates a beat on which robots gradually align (Fig. 4.1).

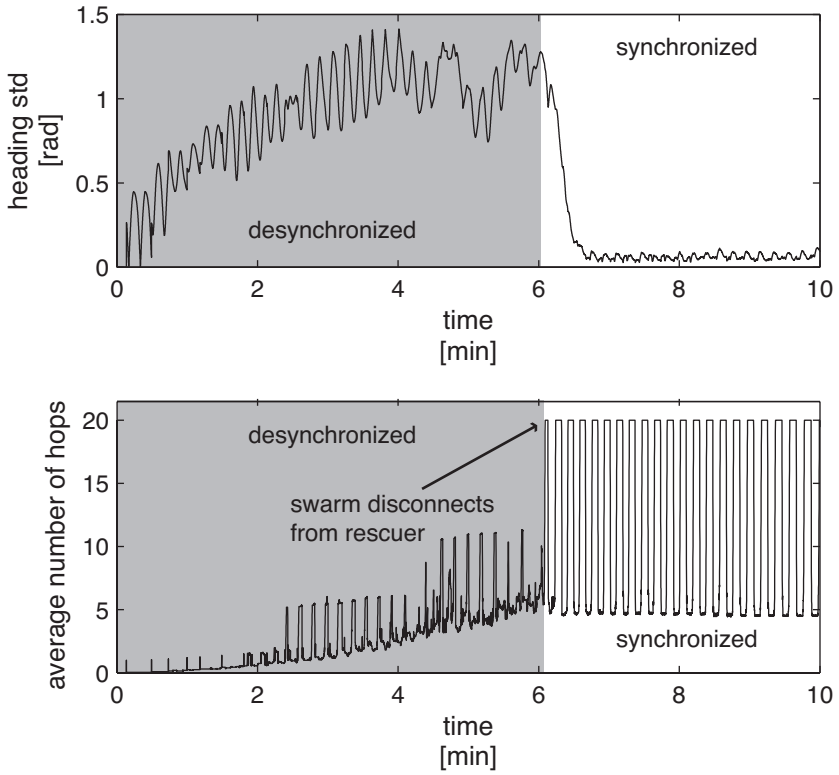


Figure 4.1: Top: standard deviation on the heading of all robots during a single run of the best evolved controller in simulation. After the swarm disconnects from the launching rescuer, the headings of the robots synchronize. Bottom: Average number of hops separating robots in the swarm from the launching rescuer. Notice robots oscillate between being connected to and disconnected from the launching rescuer which generates a rhythmic beat.

4.3 Reverse Engineered Controller

As seen in Chapter 3, communication-based controllers can generate motions where robots oscillate between being connected to and disconnected from an emitter while translating along its communication border. More generally, the notion of "connections" and "disconnections" can be replaced with any beat signal composed of an "on" phase of duration t_1 and "off" phase of duration t_2. These beats can actively be emitted using any actuator and sensor modality. For

the purpose of this thesis, beats are emitted using wireless communication from a user on the ground or one of the robots in the swarm.

Starting from the communication-based controllers described in the previous chapter, controllers of the following form are considered (Fig 4.2):

$$\omega(h,c) = \begin{cases} \omega_{\omega_1,\omega_1,h_{lim}}(h) & \text{if } c = 1 \text{ (connected)} \\ \omega_{\omega_2,\omega_1,h_{lim}}(h) & \text{if } c = 0 \text{ (disconnected)} \end{cases} \tag{4.1}$$

which can be rewritten as:

$$\omega(h,c) = \begin{cases} \omega_1, & \text{if c=1} \\ \omega_2, & \text{if c=0 } \widehat{h_{lim}h} > 0 \\ \omega_1, & \text{if c=0 } \widehat{h_{lim}h} < 0 \end{cases} \tag{4.2}$$

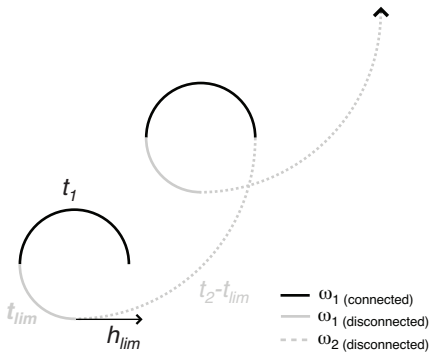


Figure 4.2: Theoretical trajectory of a robot using the controller described in equation 4.2. Here a robot receives a beat composed of an "on" phase of duration t_1 (black) and an "off" phase of duration t_2 (grey). Using this, the robot controller sets the turn rate to ω_1 or ω_2 depending on the beat and the heading of the robot with respect to a predefined heading h_{lim} (see dashed lines).

This controller has the property of having robots converge to identical headings at the beginning of each beat as shown in Fig 4.3. This can be explained by the fact that the amount of time t_{lim} spent between the moment the beat is turned off and the robot reaches the heading limit h_{lim} depends on the initial heading of the robot. If the robot starts at a heading as shown in Fig. 4.4 (left), it

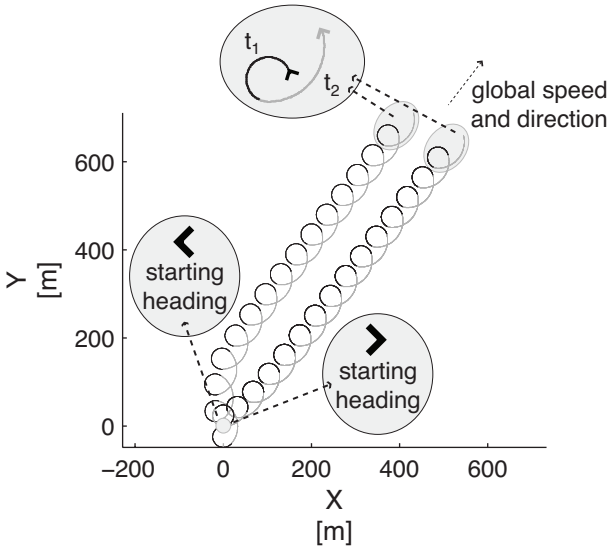


Figure 4.3: Example of synchronized steering in theory. Robots launched from (0,0) in opposite directions receive a beat composed of an "on" phase of duration t_1 (black) and an "off" phase of duration t_2 (grey). Over time, the robot headings synchronize. This can be seen by the fact that at each start of a beat, the headings of the two robots are identical. Furthermore, the trajectories converge to a fixed global velocity (speed and direction).

will perform more than 2π within one beat ($t_1 + t_2$), thereby changing its starting heading for the next beat. However, if the robot starts at the heading shown in Fig. 4.4 (right), it will perform less than 2π within one beat. Instead, once synchronized, the robot will perform 2π during one beat, meaning that it will start the next beat with the same heading.

The overall effect is that robots listening to identical beats and using the same controller parameters will synchronize over time. In the particular case shown in Fig. 4.5, robots synchronize after 2 beats.

Suitable parameters (t_1, t_2, ω_1 and ω_2) that lead to trajectories that perform 2π during one beat must be such that the minimum value for t_{lim} named t_{lim_min} produces trajectories that perform less than 2π during one beat while the maximum value t_{lim_max} produces trajectories that perform more than a full revolution during one beat. These conditions can be mathematically described as:

$$|\omega_1| \cdot t_1 + |\omega_1| \cdot t_{lim\_min} + |\omega_2| \cdot (t_2 - t_{lim\_min}) < 2\pi \tag{4.3}$$

$$|\omega_1| \cdot t_1 + |\omega_1| \cdot t_{lim\_max} + |\omega_2| \cdot (t_2 - t_{lim\_max}) > 2\pi \tag{4.4}$$

where

$$t_{lim\_min} = t_2 - min(t_2, \frac{\pi}{|\omega_2|})$$

$$t_{lim\_max} = min(t_2, \frac{\pi}{|\omega_1|})$$

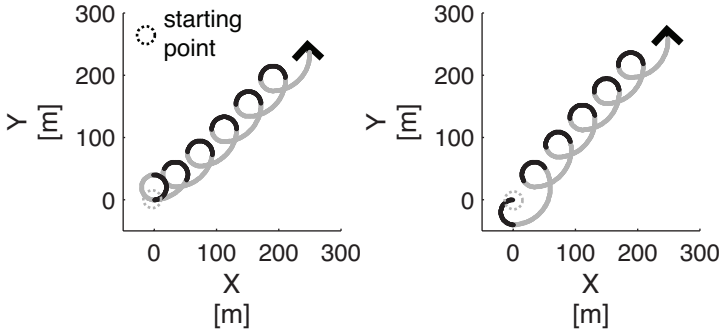


Figure 4.4: Theoretical robot trajectories synchronize over time by converging to a state where at the beginning of each beat of duration t_1 (black) + t_2 (grey), the robot returns to the same heading, thereby performing a full revolution. Synchronization is achieved independently of the initial heading of the robot. In particular, we show two examples with opposite initial headings. In the left figure, the robot performs more than one revolution during the first beat at the start of the trajectory. In the right figure, the robot performs less than one revolution during the first beat at the start of the trajectory. Notice that at the end of both trajectories, robot headings are identical.

As an advantage, this controller is able to compensate for perturbations and resynchronize. This is shown in Fig. 4.6 where we introduce 8 large perturbations to the system by increasing or decreasing the turn rate ω_1 and ω_2 by 0.05 rad/s and 0.1 rad/s during an entire beat.

Furthermore, two robots implementing identical turn rate commands and speed commands will generally not perform identical trajectories due to sensor noise and hardware differences. The effect of turn rate bias on the synchronization and steering of the robots can be seen in Fig. 4.7 where we show the

simulated trajectories resulting from turn rates of value $\omega_1 = [0.6, 0.65, 0.7]$ rad/s and $\omega_2 = 0.1$ for $t_1 = 4.488$ s and $t_2 = 17.9520$ s. However, robots that display different turn rates will still perform one revolution during one beat if they meet requirements described in Eq. 4.3 and 4.4. Therefore, the shift in heading among the robots is stable over time.

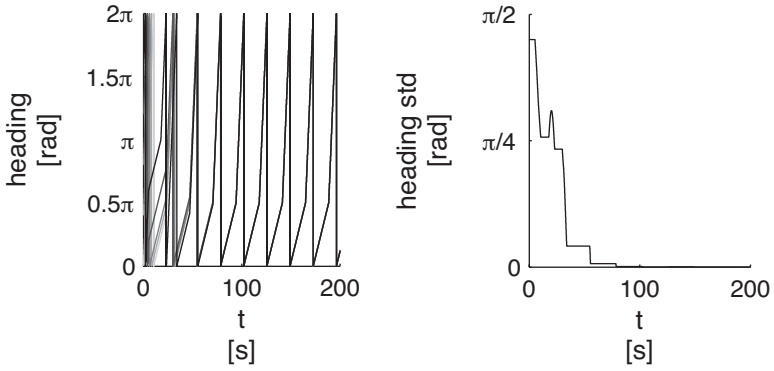


Figure 4.5: Headings (left) of 5 simulated robots initialized at headings 0, $\frac{2\pi}{5}$, $\frac{4\pi}{5}$, $\frac{6\pi}{5}$, $\frac{8\pi}{5}$. Notice that over time, the standard deviation (right) across robot headings goes down to 0, meaning the robots are synchronized.

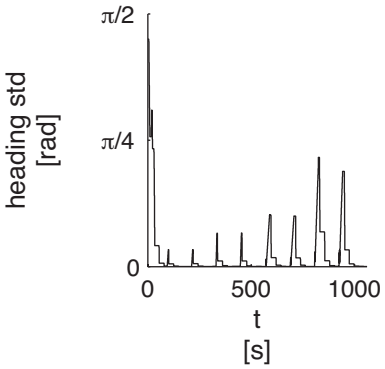


Figure 4.6: Capacity of the robot controller to synchronize after 8 large perturbations to its turn rate.

4.4 Model

Once synchronized, swarms of robots can be modeled as a single entity. This is an advantage compared to systems where the individual motions of robots need to be considered. In particular, we aim at analyzing the global advancement

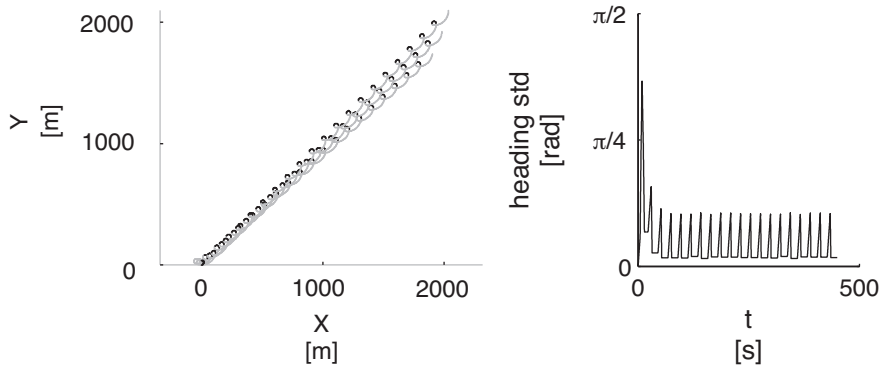


Figure 4.7: Effect of turn rate bias in simulation on the robot trajectories (left) and heading (right) of three robots with turn rates equal to $\omega_1 = [0.6, 0.65, 0.7]$ rad/s and $\omega_2 = 0.1$ for $t_1 = 4.488$ s and $t_2 = 17.9520$ s. Notice that while the robots implement different controllers, their headings still synchronize with a constant shift. The direction and advancement speed of the swarm is also slightly modified across robots.

speed $\overline{v}_{sync}$ and direction $\overline{d}_{sync}$ of the swarm motion. This knowledge is then used to identify parameters that can be modified to easily steer swarms of robots while keeping them synchronized. In particular, using symbols in Fig. 4.8 and knowing that robots with forward speed v will perform 2π during one beat, we can calculate

$$\overline{v}_{sync} = \frac{\sqrt{a^2 + b^2}}{t_1 + t_2} \tag{4.5}$$

and

$$\overline{d}_{sync} = h_{lim} + tan^{-1}\frac{a}{b} - \beta + \frac{3\pi}{2} \tag{4.6}$$

where

$$a = \frac{v}{\omega_1} sin(\beta) + \frac{v}{\omega_2} sin(\gamma) \tag{4.7}$$

$$b = \frac{v}{\omega_2} - \frac{v}{\omega_2} cos(\gamma) - (\frac{v}{\omega_1} - \frac{v}{\omega_1} cos(\beta)) \tag{4.8}$$

$$\beta = \omega_1(t_1 + t_{lim}) \tag{4.9}$$

$$\gamma = \omega_2(t_2 - t_{lim}) \tag{4.10}$$

$$t_{lim} = \frac{2\pi - |\omega_1| \cdot t_1 - |\omega_2| \cdot t_2}{|\omega_1| - |\omega_2|} \tag{4.11}$$

While these equations allow us to predict in what direction and at what speed the swarm will move, it is challenging to set the parameters in order to

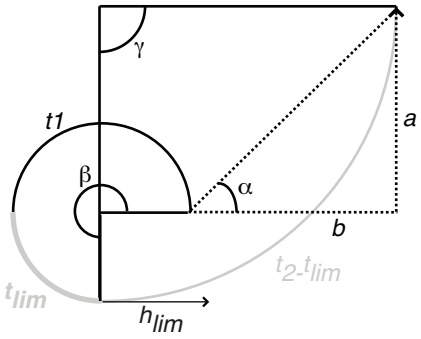


Figure 4.8: Symbols used to determine the global advancement speed and direction of the robot trajectories using Eq. 4.5 and 4.6.

achieve a desired command because the parameters (ω_1, ω_2, t_1 and t_2) can not be isolated analytically. The problem can however be simplified by only considering trajectories where

$$\beta = \frac{3}{2}\pi$$
$$\gamma = \frac{\pi}{2}$$

which can be achieved if

$$t_1 = \frac{\pi}{|\omega_1|} \tag{4.12}$$
$$t_2 = \frac{\pi}{2 \cdot |\omega_1|} + \frac{\pi}{2 \cdot |\omega_2|} \tag{4.13}$$

leading to trajectories where

$$\overline{v}_{sync} = \frac{\sqrt{2}|\frac{v}{\omega_2} - \frac{v}{\omega_1}|}{t_1 + t_2} \overline{d}_{sync} = \qquad h_{lim} + \frac{\pi}{4} \tag{4.14}$$
$$\tag{4.15}$$

Thanks to Eq. 4.14 and 4.15, the parameters can easily be modified to modulate the global motion direction and speed of the swarm. In particular, the direction can be changed by modifying h_{lim}. Furthermore, increasing or decreasing the global speed of each robot can be done by increasing or decreasing the difference between ω_2 and ω_1 respectively within the boundaries set by Eq. 4.3 and Eq. 4.4. Notice that the same approach can be used with β and γ negative.

4.5 Validation

To validate the synchronization and steering of swarms of robots we perform in-flight experiments with two to five physical fixed-wing robots described in section B.2.1. For these experiments, one of the robots was sending beats by emitting heartbeat messages at an interval of 5 ms during the "on" phase and no messages during the "off" phase. This was done to increase the robustness of beat signals to communication failure. In a more economical mode, only on and off edges would need to be sent.

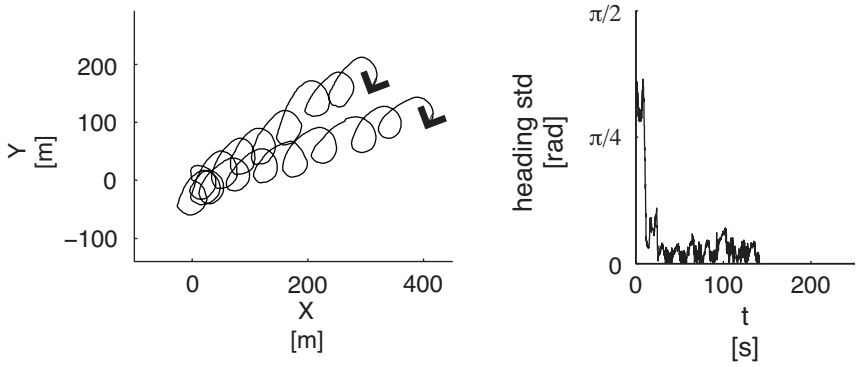


Figure 4.9: Demonstration of synchronization on board two real flying robots in an outdoor experiment. Left: trajectories of the robots. Right: standard deviation on the robot headings.

The first experiment shown in Fig. 4.9 is aimed at demonstrating that robots that start with different initial headings will synchronize over time. Parameters for this experiment are based on Eq. 4.2 with $h_{lim} = 5.4$ rad $\omega_1 = -0.7$ rad/s and $\omega_2 = -0.1$ rad/s with t_1 and t_2 set following Eq. 4.12 and 4.13 respectively. Notice how the standard deviation on robot headings rapidly goes down to nearly zero, thereby indicating synchronization.

Beyond synchronization, we aim at showing that the robot group can be steered and speed regulated. In particular, we propose three mission goals. In the first, robots are directed to go towards the North, h_{lim} is then changed, thereafter directing the group to the South (phase II). In the third phase the turn rate ω_2 is changed to slow down the global progression speed of the group. As a result, Fig. 4.10 shows how the speed and direction of the robots can be changed while remaining synchronized. Parameters for this experiment are given in Ta-

	h_{lim} [rad]	ω_1 [rad/s]	ω_2 [rad/s]
phase I	5.8	-0.7	-0.1
phase II	2.7	-0.7	-0.1
phase III	2.7	-0.7	-0.3

Table 4.1: Controller parameters used to achieve trajectories shown in Fig. 4.10.

ble 4.1. Notice that because of wind to the South of around 3.5m/s, the desired speed and direction of the group in not exact with respect to theoretical calculations. Good synchronization and group steering is however achieved.

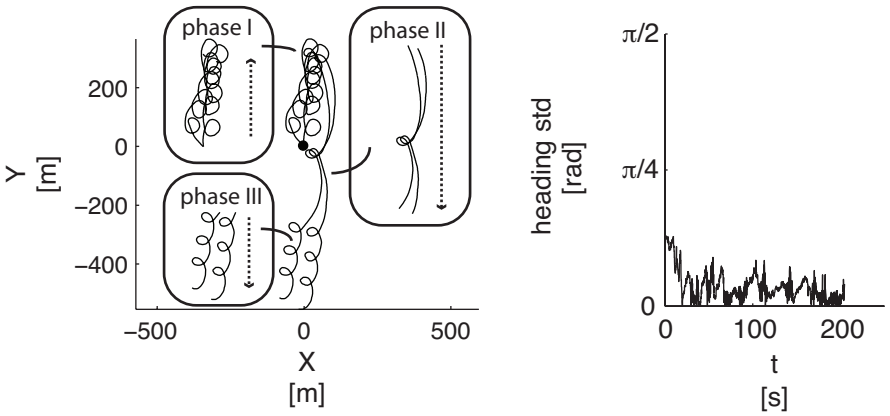


Figure 4.10: Demonstration of synchronization and steering of two real flying robots in an outdoor experiment. Left: trajectories of the robots. Right: standard deviation on the robot headings. Three phases are shown here, in phase I, the robot group is directed to the North against the wind. In phase II, robots are directed to turn around and proceed South. Phase III then shows how the robots can be slowed down. Notice that the robots remain synchronized throughout the experiment.

Finally, in Fig. 4.11 we show that this method scales to five flying robots. For this experiment, we propose two mission goals. In the first, robots are directed to go towards the West, h_{lim} is then changed, thereafter directing the swarm to the East (phase II). Fig. 4.11 (right) shows the standard deviation on the heading of the robots which rapidly decreases over time as robots synchronize. The five trajectories are summarized by their mean which highly resembles the individual trajectories because all robots are synchronized. In this experiment

wind between 1 m/s and 2 m/s to the North-East was present. Overall, robots are able to achieve good synchronization and steering. Because of wind and the dynamics of the robots, which prevent them from rapidly changing their turn rate, the actual direction performed by the swarm is slightly shifted with respect to the initial goal. A demonstration of five synchronized robots can be seen in a video on our website*.

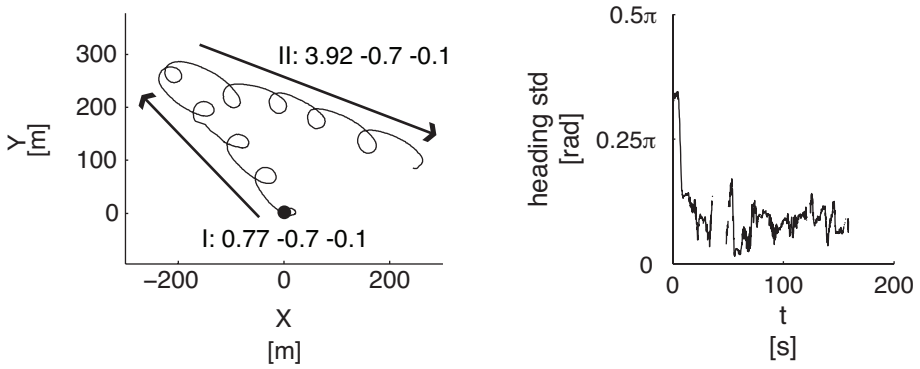


Figure 4.11: Demonstration of synchronization and steering of five real flying robots in an outdoor experiment. Left: mean trajectory of the robots. Right: standard deviation on the robot headings. Two phases are shown here, in phase I, the robot swarm is directed to the West. In phase II, robots are directed to turn around and proceed East. Parameters represent h_{lim}, ω_1 and ω_2. Notice that the robots remain synchronized throughout the experiment which is why it is possible to plot such a mean trajectory.

4.6 Extensions

The synchronized steering of swarms of robots can serve as an essential building block towards deploying multiple flying robots in real-world applications. One option to extend this behavior consists in implementing an outer-loop responsible for issuing commands for the steering and speed regulation of the robots. This outer-loop can reactively increase or decrease the speed of the swarm and make it turn more or less based on sensory input from the robots. As an example, we consider a scenario where the swarm must remain connected to a user on the ground. Each time a robot loses its connection to the user, it records

*http://lis.epfl.ch/smavs

its heading and broadcasts a new set of controller parameters to all robots that make them pursue a global direction opposite from its disconnection heading. In that manner it becomes the "leader" of the swarm. Results in Fig. 4.12 show that robots starting from different headings are able to synchronize and move in groups while remaining connected to the user. Notice that rather than relying on leaders, swarms could also collectively decide on new controller parameters.

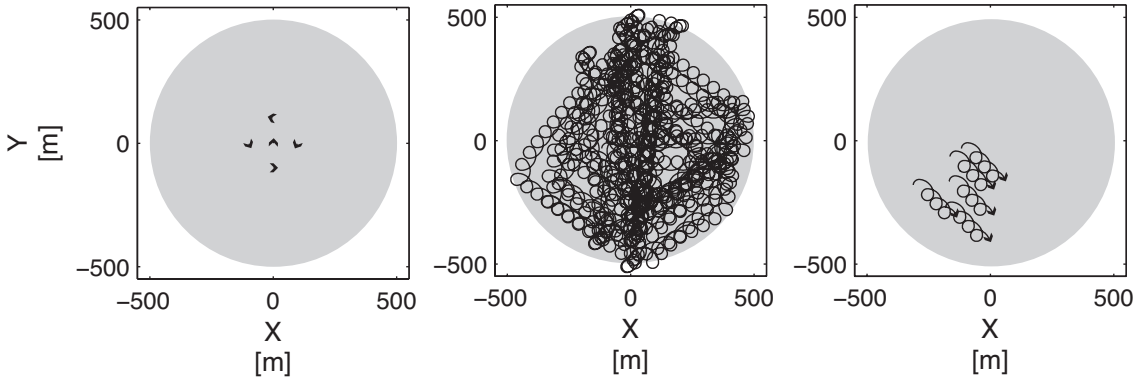


Figure 4.12: Simulated swarms synchronize and move in groups while remaining connected to a user on the ground. The grey area represents the communication range of the user in (0,0). The figure to the left shows the starting position and heading of the robots, the center image shows all trajectories over a 30 min trial and the figure to the right shows the end of the trial and the synchronized heading of the robots.

4.7 Conclusion

By reverse engineering the best evolved controller from Chap. 2, we were able to identify basic principles governing the coherent motion of groups of robots.

Group motion of fixed-wing platforms can be achieved by synchronizing the heading of the robots. In this manner, the group can be considered as a single entity whose motion can be controlled. Rather than relying on complex controllers based on position, frequent sensing, or memory, we propose basic principles to modulate the turn rate of robots based on their current heading and a beat signal received from a user on the ground or a robot. Beats are generated by sending heartbeat messages using communication. The designed controller

has the property of synchronizing the robot headings to the beats, regardless of their initial headings. Furthermore, it can easily be parameterized to steer a swarm of robots and change its global speed in a predictable manner. Results are validated theoretically and using two to five fully autonomous physical flying robots. Finally, discovered principles are used in an application where robots must move in groups while remaining connected to a user on the ground.

In the future, efforts should be made to describe our controller in terms of synchronized oscillators[67,96]. Such a venue would allow for stability proofs, more formal mathematical models and a large range of extensions based on different forms of synchronization states found in the literature.

5 Area Coverage

Abstract

The main contribution of this chapter is the identification of basic principles for the deployment of robot chains that perform area coverage.

Area coverage is one of the most commonly used behaviors in aerial systems to search an environment for areas of interest. Current strategies typically rely on position information to do this. Instead the solution found through evolution in Chapter 2 focuses on covering the area by forming a chain of synchronized robots that can move from side to side. Following the systematic approach proposed in this thesis, we reverse engineer a controller that can easily be parameterized to obtain a desired area coverage. Controllers are validated in simulation and in preliminary experiments with 9 flying robots in reality. Principles are then extended to scenarios where chains perfom more complex behaviors.

This chapter is based on:

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1) pp. 21-32.

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Reverse-engineering of Artificially Evolved Controllers for Swarms of Robots. *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 55-61.

5.1 Background

Groups of flying robots are often envisaged to rapidly explore large environments. Several strategies have been devised to search an environment for areas of interest with applications in the detection of chemical plumes[55,57,77,113], wild fires[36,66], victims[71] and other targets of interest[4,18,56,88,90,95,111].

The straightforward way to coordinate an exploration task using multiple flying robots is to divide the environment into regions that will each be assigned to a separate robot[36,55,66]. An example involving five fixed-wing robots was demonstrated outdoors[3]. As an extension, centralized controllers are used to plan the trajectories of the robots so that the correct robot visits the correct search areas at the correct time while avoiding collisions. The problem is described using mixed-integer linear programming (MILP) that can be solved optimally[2,9,84]. One noticeable example of optimal path planning in reality was conducted outdoor with three fixed wing platforms by Beard et al.[7].

In another approach that is instead distributed, robots use maps to keep a history of events (position, sensory data, etc.) concerning their neighbors and themselves. In work by Yang et al.[111], flying robots search for targets in an unknown and uncertain environment based on the sensor measurements and position of all the robots in the swarm (evidence). Using this data, robots build evidential maps based on probabilistic methods and then use these maps to plan their trajectories so as to maximize their chances of finding the target. In another method loosely inspired from ants, Parunak et al.[90,104] have robots search for targets by coordinating their actions through a map containing virtual chemicals (pheromone). Attractive and repulsive pheromones can be deposited on a map or withdrawn from it by a robot and its neighbors. Furthermore, pheromone evaporates over time meaning that it gradually disappears from the map. It can also diffuse to nearby areas thereby creating a pheromone gradient. Robots follow attractive pheromone gradients to areas of interest in the environment while depositing repulsive pheromone to prevent robots from monitoring overlapping areas[56].

Quite similarly to pheromone maps, Peng et al.[80] have been using flying robots to search an environment by using hormone maps where the attractive and repulsive pheromones are replaced by activator and inhibitor hormones. Finally, Lawrence et al.[57] create information energy potentials that capture the

quality of the measurements made by other robots, their energy loss due to motion and the quality of the ad-hoc network they form. The resulting gradients are then navigated by the robots to map toxic plumes while maintaining a coherent communication network to a central processing unit on the ground.

Beyond maps, researchers have been looking at flocking algorithms and artificial physics to allow robots to create regular grids above an environment by spacing out with equal distance using attraction and repulsion rules[6,21,24,53,78].

However, the question of how to solve this problem without GPS remains open. As a solution, previous work performed during this thesis and presented in Appendix A looked at how robots themselves can replace maps by serving as a substrate on which information can be deposited and read from using local communication. In particular we took inspiration from ants foraging for food to create controllers for flying robots that must search an area for users and create an ad-hoc network between them. Like in nature, robots decide on where to go based on the amount of pheromone present in their environment. To achieve this, robots are separated into two categories, namely "nodes" that form a physical grid and serve as a substrate for virtual pheromone and "ants" that can navigate through this grid by reading the pheromone information on the nodes while depositing pheromone on them (Fig. A.7). Using this pheromone-based system, robots create dynamic grid structures that reach out from a launching rescuer in search for a user without the need for global or relative positioning.

However, ant-based controllers were not as simple as those discovered through evolution and parameters could not be analytically determined. We therefore focus on reverse engineering evolved controllers for the deployment of chains of flying robots that can cover a large area by moving from side to side.

5.2 Evolved Behavior

An example showing the behavior of the best evolved controller from Chapter 2 can be seen in Fig. 5.1. The strategy adopted by the swarm consists in forming a tight chain of flying robots which grows as long as additional robots are launched. Once all robots have been launched, the robot chain shifts along the communication border of the launching rescuer, sweeping the area from West to East in search for the second rescuer.

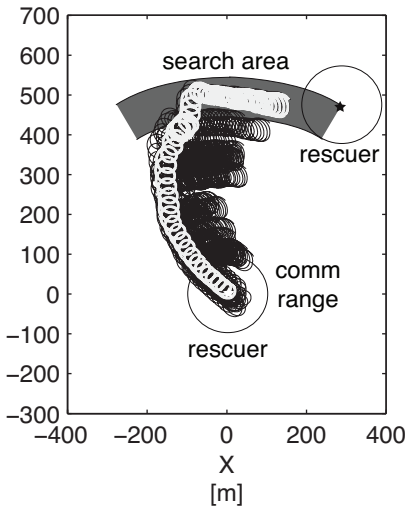


Figure 5.1: Trajectories of 20 simulated robots with the best evolved controller (over all populations in all generations) performing chain formation and translation. The trajectory of the first launched robot is shown by a light grey line.

Through a systematic analysis of the effect of each input of the best evolved neural controller on the turn rate of the robots, we identify two simple behaviors performed by the individual robots:

- Robots that are connected to the launching rescuer, even indirectly, move away from it (Fig. 5.2, low hop values) while loitering.
- Robots that are disconnected from the launching rescuer, move towards it with a different average turn rate than when connected (Fig. 5.2, high hop values).

The effect of these communication-based behaviors on the entire group can be hypothesized as follows. Loitering trajectories allow robots to move slowly, so as to adapt to the pace at which robots are launched. As a result, each newly launched robot becomes a new link in a growing chain of robots that ensures that the swarm remains connected to the launching rescuer (at least indirectly). The swarm therefore advances in a common direction away from the launching rescuer. Once all robots have been launched, the chain continues to advance until it disconnects from the launching rescuer. To reconnect, the robots change direction and move towards the launching rescuer. Notice that this "reconnection" behavior is also useful for individual robots that have been

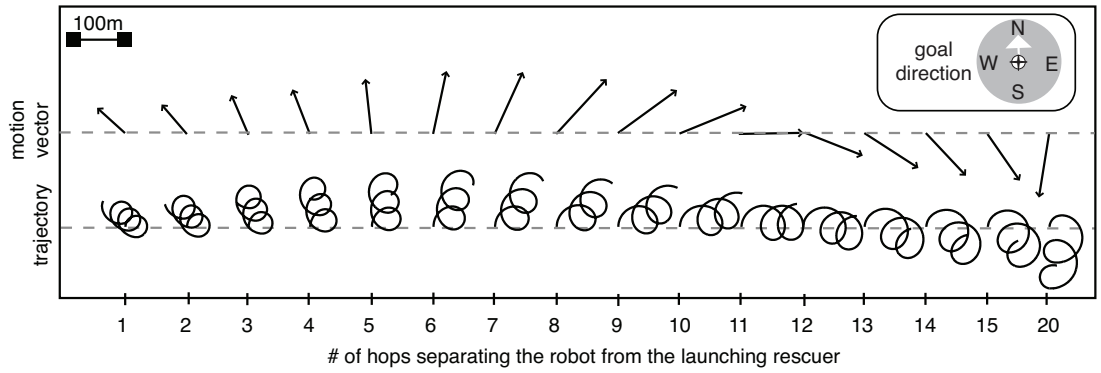


Figure 5.2: Effect of the number of hops which separate the launching rescuer from a robot on its trajectory. Here, we plot the trajectories of the best evolved controller from Chapter 2 over 30 s. Robots were never connected to the second rescuer during these experiments.

separated from the swarm. The oscillation of the swarm between connections and disconnections allows the robots to synchronize and translate along the communication range of the launching rescuer. As seen in the previous chapters, translation is due to the different turn rates performed by the robots depending on heading and communication input.

5.3 Reverse Engineered Controller

We translate behaviors observed in the best evolved solution into communication-based controllers for flying robots. As previously, we simplify the strategy found through evolution by only considering if a robot is receiving messages from the rescuers (i.e., whether it is connected to the rescuers rather the number of hops separating it from the rescuers).

Based on the individual robot motions presented in the previous chapter we consider controllers of the following form:

$$\omega(h,c) = \begin{cases} \omega_{\omega_1,\omega_2,h_{away}}(h) \text{ if } c = 1 \text{ (connected)} \\ \omega_{\omega_3,\omega_4,h_{towards}}(h) \text{ if } c = 0 \text{ (disconnected)} \end{cases} \tag{5.1}$$

Where h_{away} is the heading limit that will make the robots navigate away from the launching rescuer. Likewise, $h_{towards}$ is the heading limit that will produce trajectories where robots go towards the launching rescuer.

5.4 Model

We aim at quantifying the area coverage achieved by a swarm for a given set of controller parameters. Models of area coverage can then be used to rapidly optimize parameters for a desired scenario which is important for real-world applications.

Our model considers the maximum length l of the robot chain that extends from the launching rescuer and the width w of the area covered due to the chain translating along the communication border of the launching rescuer. Missions have a duration $t_{mission}$ and N robots are launched at a rate of 1 every Δt_{launch} seconds. The area coverage is then given by the rectangular area of size l x w.

Chain extension During chain formation, N robots are launched at a rate of 1 every t_{launch} seconds. Each robot performs the same controller $\omega_{\omega_1,\omega_2,h_{away}}(h)$ when connected to the launching rescuer. This controller generates trajectories with global advancement speed $\overline{v}$ analyzed in Eq. 3.3 in the chapter on individual robot motions (Sec. 3.4). Therefore, the length l of the chain can be approximated as the distance covered by a newly launched robot with global advancement speed $\overline{v}$ before a new robot is launched, times the number of robots. This leads to:

$$l = \overline{v} \cdot \Delta t_{launch} \cdot N \tag{5.2}$$

If the robots advance too fast, they will disconnect from the launching rescuer before a new robot has been launched. Because of their controller, these robots then turn around to reconnect. The maximum distance that separates two robots can then be approximated by the communication range r_{com} of the launching rescuer such that:

$$l = min(N \cdot \overline{v} \cdot \Delta t_{launch}, N \cdot r_{com}) \tag{5.3}$$

Finally, considering that the chain will navigate to the edge of the communication range of the rescuer, the maximum reach of the chain can be given as:

$$l = min(N \cdot \overline{v} \cdot \Delta t_{launch}, N \cdot r_{com}) + r_{com} \tag{5.4}$$

Chain translation Once the chain has navigated to the communication border of the launching rescuer, it will oscillate between being connected to and disconnected from it. This beat will synchronize the robots as shown in Chapter 4. We can therefore model the synchronized chain as a single robot. The translation speed $\overline{v}_{com}$ of a robot along the communication border of a user was analyzed in Eq. 3.9 in the chapter on individual robot motions (Sec. 3.4). Therefore, the width w that the chain translates along the communication range of the launching rescuer for a mission of duration $t_{mission}$ is given by the global translation speed of a robot times the duration of the mission once the chain has deployed. This leads to:

$$w = \overline{v}_{com} \cdot (t_{mission} - t_{chain}) \tag{5.5}$$

where

$$t_{chain} = N \cdot \Delta t_{launch} + \frac{r_{com}}{\overline{v}} \tag{5.6}$$

5.5 Validation

Experiments are run in a realistic event-based simulator which implements 802.11-b communication models, physics-based wave propagation and a first order model of a robot platform which flies at 10 m/s, has a minimum turn radius of 10 m and is affected by sensor and actuator noise as described in Appendix B.1.2.

In the evolved scenario, area coverage is limited by the range of the launching rescuer. To fully explore the possibilities of area coverage we consider a scenario where rescuers arriving by road, deposit several wireless beacons along the way at a small enough interval for the beacons to be directly or indirectly interconnected. Here, beacons with a communication range of 100 m are dropped from a rescue vehicle, every 50 m, along a straight road which extends from West to

East (Fig. 5.3). A rescuer will then sequentially launch 20 robots every 15±7.5 s by throwing them into the air from the West-most beacon. Notice that robots connected to beacons are also connected to the launching rescuer (indirectly). The swarm must then cover an area l x w to the North of the launching rescuer during a mission of 30 min.

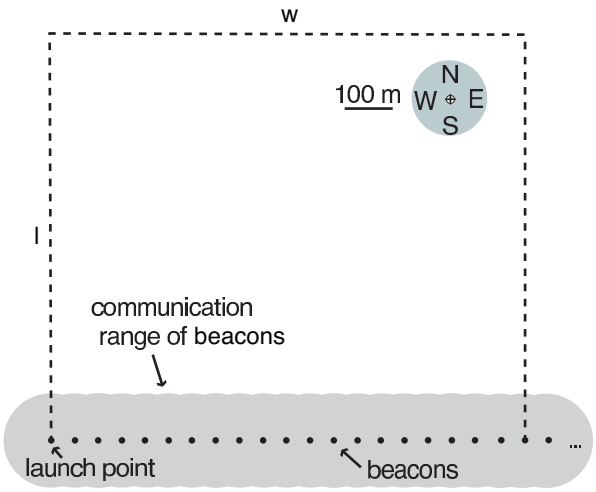


Figure 5.3: The swarm composed of 20 robots must cover an area of l x w to the North of the launching rescuer. The beacons that extend the range of the launching rescuer have a communication range of 100 m and are positioned every 50 m along a straight road. Robots are launched from the West-most beacon.

As a first step, Figure 5.4 shows that the communication-based controller described in Eq. 5.1 effectively leads to chain extension and translation.

The area coverage model is then used to quickly adapt swarm controllers to new scenarios. In particular, we determine optimal parameters ω_1, ω_2, ω_3 and ω_4 so as to achieve desired area coverages l x w. The remainder of the parameters are set by the scenario specifications (here $h_{away} = \pi$ and $h_{towards} = 0$). To optimize the parameters, we compute the summed square error $e = (l - l_{pred})^2 + (w - w_{pred})^2$ for each combination of r_1, r_2, r_3, r_4 in the range of natural numbers from r_{min} to r_{max} with the constraint that $r_1 < r_2$ and $r_3 < r_4$. Here r_i corresponds to the turn radius performed by a robot when applying turn rate ω_i with $\omega_i = v/r_i$ and v is the speed of the robot. The value of r_{min} reflects the smallest possible radius performed by the robot, in this case 10 m. To ensure that the turn radius of the robot remains small with respect to

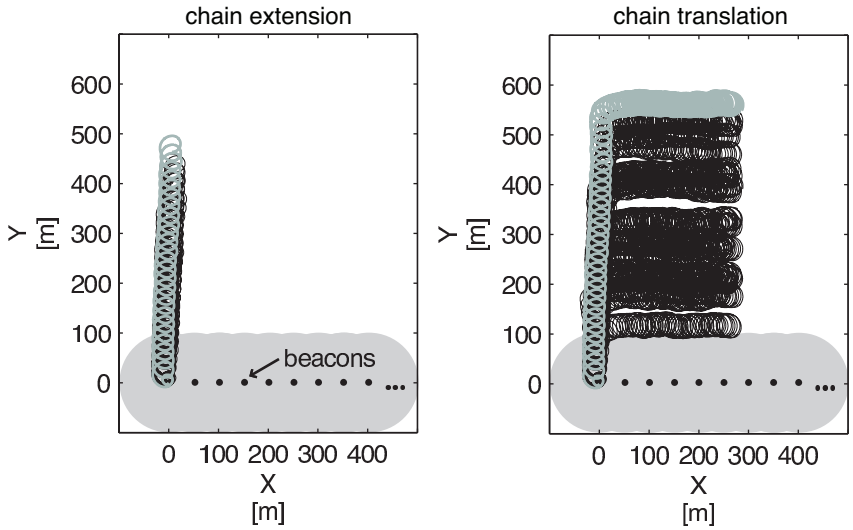


Figure 5.4: Trajectories of all the robots performing chain extension and translation in simulation. The trajectory of the first launched robot is shown by a light grey line.

the communication range, we define the maximum turn radius as equal to 25 m. The combination of parameters with the smallest error e is selected as the optimized parameter set. We test our approach on five different coverages with the corresponding parameters listed in Table 5.1. Fig. 5.5 shows the distances l_{sim} and w_{sim} reached in simulation for each desired area coverage. As can be seen, the optimized robot controllers are successful since the simulated swarms are able to achieve the desired area coverages. Small shifts between the desired and obtained coverages are due to the fact that the time needed for the robots to synchronize and stabilize their trajectories is not taken into account in the model.

l x w $[m$ x $m]$	r_1 $[m]$	r_2 $[m]$	r_3 $[m]$	r_4 $[m]$
250 x 500	11	13	12	15
500 x 750	14	22	23	24
500 x 500	16	25	23	25
750 x 500	10	21	10	25
500 x 250	14	22	16	22

Table 5.1: Optimized parameters for varying area coverages l x w

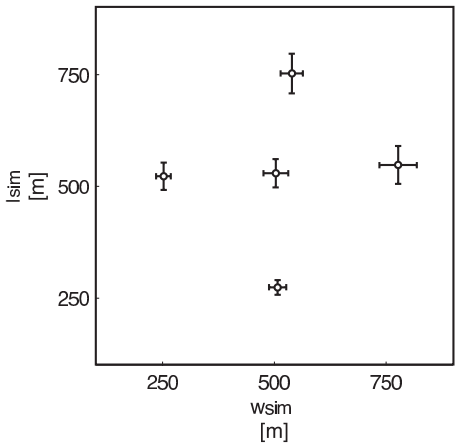


Figure 5.5: Chain length l_{sim} and sweep translation w_{sim} along the communication range of the launching rescuer (Fig. 5.3) reached in simulation by the swarm over 100 trials for desired area coverages of 250 m x 500 m, 500 m x 500 m, 750 m x 500 m, 500 m x 250 m and 500 m x 750 m. For each of the five desired area coverages, we plot the mean coverage obtained in simulation with a point and the standard deviations as bars extending from this point.

Finally, Fig. 5.6 (left) shows preliminary results with 9 real flying robots launched every 5 s around the position (0,0). Robots run controllers described in Eq. 5.1 with controller parameters $\omega_1 = 0.7$, $\omega_2 = 0.4$, $\omega_3 = 0.25$, $\omega_4 = 0.1$, $h_{away} = 2.0$, $h_{towards} = 2.0 + \pi$. Notice that the robots start by forming a chain to the north until all robots disconnect from the launching rescuer. At which point the robots start translating to the left.

5.6 Extensions

Behaviors related to area coverage can be extended to a variety of scenarios that could be useful in real-world applications. In particular, controller parameters could easily be changed during a mission to reverse the translation direction of the chain or redeploy the chain to the opposite side of the launching rescuer. One such mission is shown in Fig. 5.7.

Likewise, parameters h_{away} and $h_{towards}$ could be adapted during a mission to better follow the curvature of the communication border of the launching rescuer. Indeed robots sensing connections and disconnections while loitering

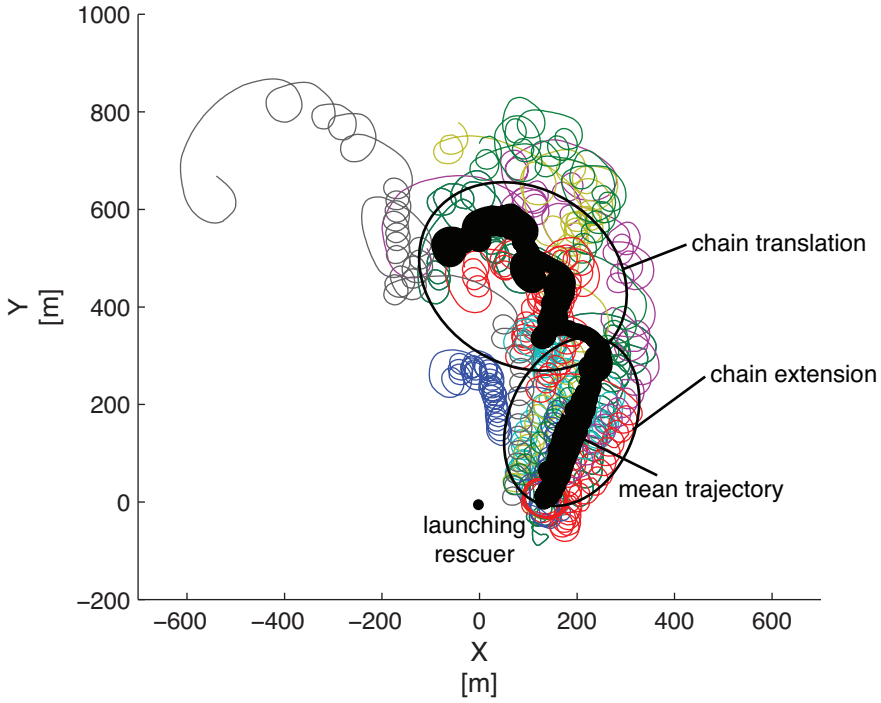


Figure 5.6: Preliminary results with 9 real flying robots showing chain extension and translation. The line in bold shows the mean trajectory of the robots.

could produce statistics on the predicted direction of the launching rescuer and use that to change parameters accordingly.

5.7 Conclusion

By reverse engineering the best evolved controller from Chapter 2, we were able to identify basic principles to cover an area using a group of robots. Discovered controllers do not use position information and are suitable for fixed-wing robots.

More precisely, controllers build on individual robot motions discovered in Chapter 3 and group motions discovered in Chapter 4. When launched sequen-

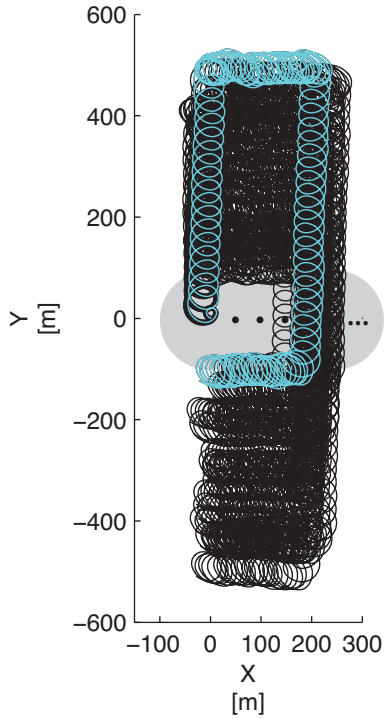


Figure 5.7: Trajectories of all the robots performing chain extension and translation in simulation. The trajectory of the first launched robot is shown by a light grey line. The entire trial lasts 30 min. After 15 min, the controller parameters are changed to make the robots translate along the opposite side of the launching rescuer.

tially from a user on the ground, robots deploy in a common direction to form a chain. Chains eventually disconnect from the launcher and start oscillating between being connected to and disconnected from it. This generates a beat on which robots can synchronize their heading to move in coherent groups. Synchronized chains then translate along the communication border of the launcher, thereby deterministically covering an area of predictable size. Controllers and area coverage predictions are validated for a variety of scenarios demonstrated in simulation and with preliminary results using 9 physical robots. Discovered controllers are then extended to scenarios where chains can change their deployment direction during a mission.

6 Communication Relay

Abstract

The main contribution of this chapter is the identification of principles to maintain and improve communication relays using flying robots.

Swarms of aerial robots can be used to relay wireless messages between users who are not within direct communication range. The placement of these robots in the environment affects the amount of messages they relay. By reacting to received messages, robots can close the loop between motion and communication. Following the systematic approach proposed in this thesis, we reverse engineer a controller that allows robots that are in a good position for wireless relay to circle while other robots continue to move until a better location is found. Discovered behaviors are then validated in simulation and in preliminary experiments outdoors with 10 flying robots. Behaviors are finally extended to scenarios where robots are able to reposition themselves so as to largely improve the throughput and lower the latency of a simulated wireless relay between two users.

This chapter is based on:

- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1) pp. 21-32. d

6.1 Background

In communication relay applications, people equipped with wireless devices are unable to directly transmit packets to one another because they are not within transmission range. Instead they must communicate via a network of wireless relay nodes capable of forwarding a packet to its final destination.

Robotic relay nodes are interesting because they can rapidly be positioned in novel environments in an autonomous manner[99]. Furthermore, robots have the possibility to continuously move so as to improve the performance of the communication relay[19,41] for example by changing the topology of the network[81].

In current research, aerial robotic relay nodes are typically positioned at equal distance from one another[53,94]. However, deployments based on the geometrical position of robots usually do not translate to a good performance of the communication relay since the quality of wireless transmissions is highly dependent on the environment (e.g. weather, terrain).

Interesting work by Dixon et al.[27] has been looking to have robots position themselves in an environment based on the signal-to-noise ratio (SNR) of transmissions from neighboring robots and ground users rather the geometrical position. Using this information, they were able to optimize a relay between two ground users in reality by having a robot navigate SNR gradients to optimal positions[35]. SNR information however suffers shortcomings described in the Sec. 3.1.

Instead, we look to improve communication by having robots move following communication-based behaviors that directly reflect the quality of the relay which they serve.

6.2 Evolved Behavior

In the best evolved strategy from Chapter 2, robots that are connected to both the launching rescuer and a second rescuer in the environment turn with the highest possible turn rate so as to maintain the communication link active as shown in Fig. 6.1.

However, robots are typically not immediately in positions that allow them to continuously relay packets between the two rescuers. Instead when disconnected from the second rescuer, robots continue performing area coverage de-

scribed in Chapter 5. This oscillation between performing area coverage and communication relay persists until the robots are stably connected to the two rescuers.

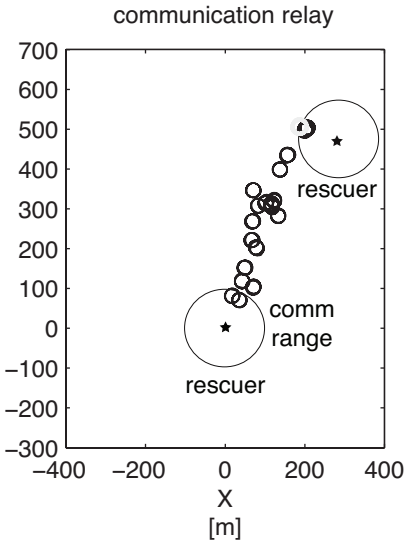


Figure 6.1: 20 robots in simulation with the best evolved controller performing a communication relay between two rescuers by turning with the highest possible turn rate.

6.3 Reverse Engineered Controller

Communication-based controllers capturing the behavior found through evolution simply display their maximum turn rate ω_{max} when connected to both the launching rescuer ($c_{lr} = 1$) and the second rescuer ($c_{sr} = 1$). When disconnected from either rescuers, the robot continues to perform area coverage as given in the controller below:

$$\omega(h, c_{lr}, c_{sr}) = \begin{cases} \omega_{\omega_1,\omega_2,h_{away}}(h) \text{ if } c_{lr} \bigwedge \neg c_{sr} \\ \omega_{\omega_3,\omega_4,h_{towards}}(h) \text{ if } \neg c_{lr} \\ \omega_{max} \text{ if } c_{lr} \bigwedge c_{sr} \text{otherwise} \end{cases} \tag{6.1}$$

6.4 Model

The behavioral analysis presented here stems from the fact that robots stabilize their trajectories only when fully connected to both rescuers, at which point they will turn on the spot. Any other connection configuration will lead to robots performing alternative behaviors, which will push the robot to change position until a stable connection is found. This amounts to a local search behavior.

One simple example of this phenomenon is shown in Fig. 6.2. Assuming there is an area where a robot is connected to both rescuers (grey zone) we can see that any trajectory starting from outside this area will converge to a stable circling behavior within the connection area given the controller described in Eq. 6.1.

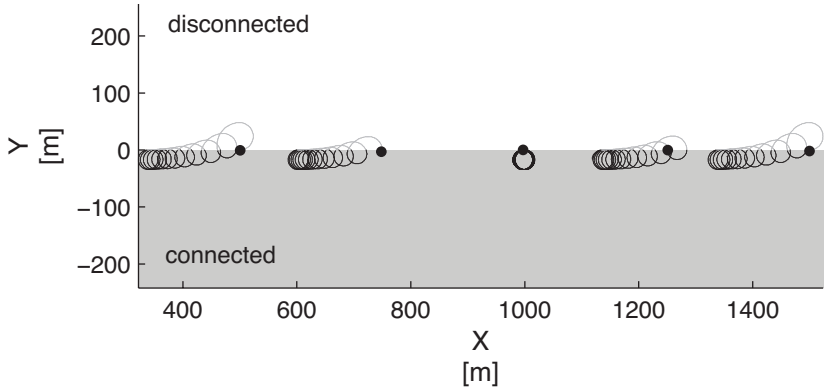


Figure 6.2: Simulated robots with controller parameters $\omega_1 = 0.5$, $\omega_2 = 0.3$, $\omega_{max} = 0.7$, $h_{away} = \pi$ and $h_{towards} = 0$ are launched from the different points marked by a full black circle. Over time, robot trajectories converge to the connection area marked in grey.

6.5 Validation

As in the previous chapter, experiments are run in a realistic event-based simulator that is described in Appendix B.1.2.

Fig. 6.3 shows the behavior of 20 robots converging to a stable position for communication relay in simulation.

Moreover, we aim at quantifying the quality of the communication between the two rescuers. To do so, we consider a swarm designed to cover an area of

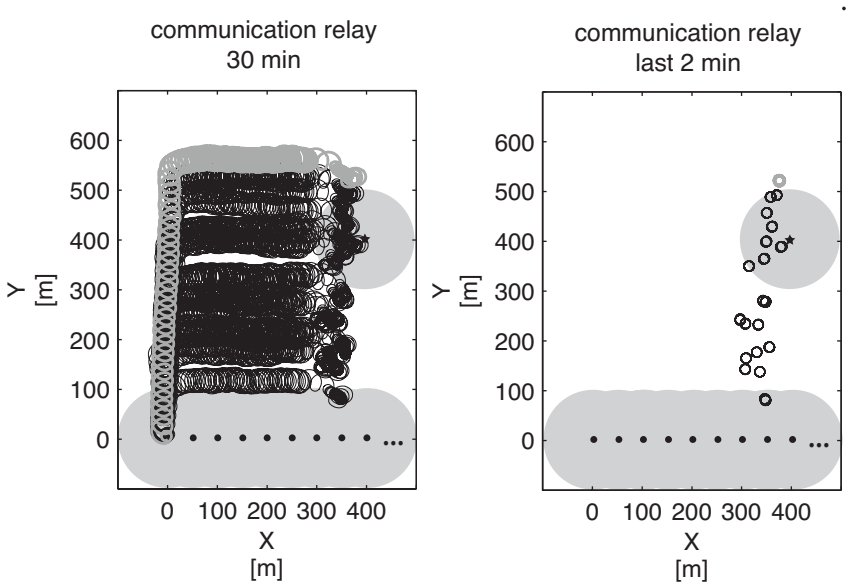


Figure 6.3: Simulated trajectories of all the flying robots during a 30 min trial. The user is located in (400, 400). The trajectory of the first launched flying robot is shown by a light grey line. Notice the behavior of robots once the connection has been established between the two rescuers.

500 m x 500 m in which the second rescuer is randomly positioned over a 30 min trial. Once a connection between the two rescuers is established, the probability of the second rescuer receiving a data packet sent every second from the launching rescuer is given in Fig. 6.4. Results show that the median probability is of 81% which is sufficient to achieve usable communication networks. The probability is not maximal because it takes time for robots to create a stable connection. Finally, although sometimes intermittent, the communication links are maintained in 100% of the cases to the end of the trial durations.

Finally, Fig. 6.5 shows preliminary results with 10 real flying robots. Robots run controllers described in Eq. 6.1 with controller parameter $\omega_1 = 0.7$, $\omega_2 = 0.25$, $\omega_3 = 0.3$, $\omega_4 = 0.2$, $\omega_{max} = 0.7$, $h_{away} = 2.8$, $h_{towards} = 2.8 + \pi$. As can be seen, the robots are able to receive and relay messages from both rescuers.

6.6 Extensions

The basic principle behind the communication relay developed so far is that robots that are useful for communication should try to remain at their current

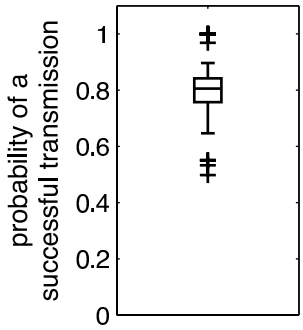


Figure 6.4: Probability of transmitting a data packet between the two rescuers, when tested in simulation over 100 trials with users randomly positioned in a 500 m x 500 m area as shown in Fig. A.1. Data packets are only sent after the swarm has created a first connection from the two rescuers. The box has lines at the lower quartile, median, and upper quartile values. The whiskers extend to the farthest data points that are within 1.5 times the interquartile range. + symbols denote outliers.

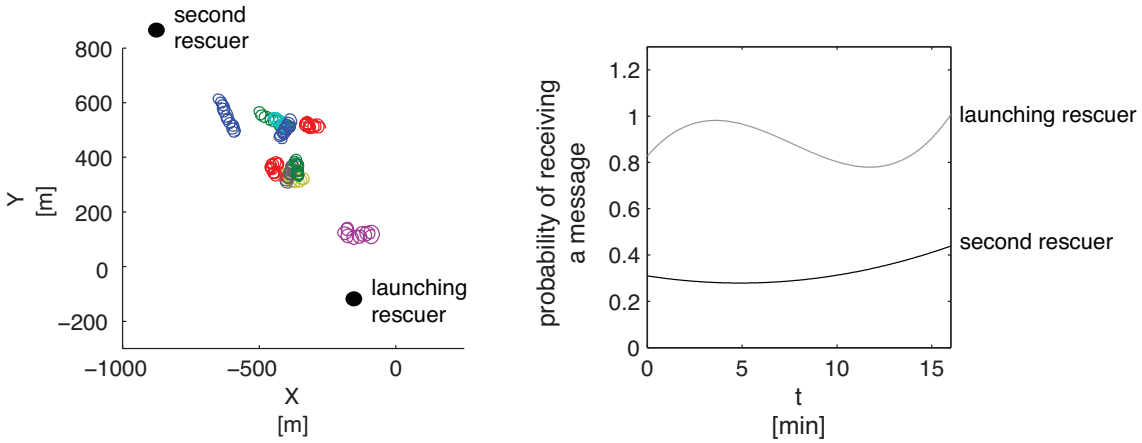


Figure 6.5: Preliminary results with 10 real flying robots showing communication relay. Left: trajectory of the robots circling during the last minute of the trial. Right: probability of receiving a message from the launching rescuer and from the second rescuer averaged over all the robots.

position while other robots should search for a better position. This idea can be extended to a variety of communication relay scenarios by either changing the definition of "usefulness" or changing the manner in which robots search for a better position.

As an example we focus on a simulated scenario in which a swarm of 20 flying robots cooperatively improve the throughput and latency of the wireless network they form between two rescuers positioned 500 m apart at positions (0,0) and (0, 500). Initially, the robots are randomly distributed in the environment within a rectangle of 500 m x 200 m delimited by coordinates (-100, 0), (100, 0), (100, 500), (-100, 500). Rescuers and robots have a communication range of approximately 100 m. Only robot placements which allow for the relay (at least intermittently) of messages between the two rescuers are considered. From the start of the scenario, rescuers talk to each other using voice over IP (VOIP) for a duration of 15 minutes which is the length of the trial.

More specifically, VOIP uses the G.711 codec to encode the audio as 160 byte messages to be sent every 20 ms by each user. An additional 40 byte payload is added for the header and 5 bytes that include additional information are piggybacked to the message. As a result, messages of 205 bytes are sent by each rescuer. Successful wireless relay leads to 80 kbps received by each rescuer or a total of 160 kbps over the entire network. If only the audio data is considered to compute the throughput, then 125 kbps are relayed over the network.

Robots are able to detect if they are "useful" to the wireless network by measuring the number of messages they relay over a predefined time window of duration Δt_{win} while positioned on the shortest path between users. This measure of "usefulness" directly reflects the throughput (relay rate) and latency (hop count) capabilities of the network. Robots then compare their "usefulness" u to the average "usefulness" of neighboring robots $\overline{u}$. Robots for which $u < c \cdot \overline{u}$ reposition themselves in the environment. Here c is a predetermined constant which determines when a robot is not deemed useful for the network with respect to its neighbors.

By default, robots turn with the highest possible turn rate. When they detect that they are not useful for the wireless relay, they navigate throughout the network by performing straight lines until disconnected from the swarm and then spiraling to reconnect. Using this strategy, robots zig-zag throughout the network and reposition themselves so as to maximize the communication relay as shown in Fig. 6.6.

Furthermore, the throughput and latency of the network over a trial of 15 minutes and averaged over 100 trials can be seen in Fig. 6.7. The throughput here is defined by the added amount of audio data received by the two users over

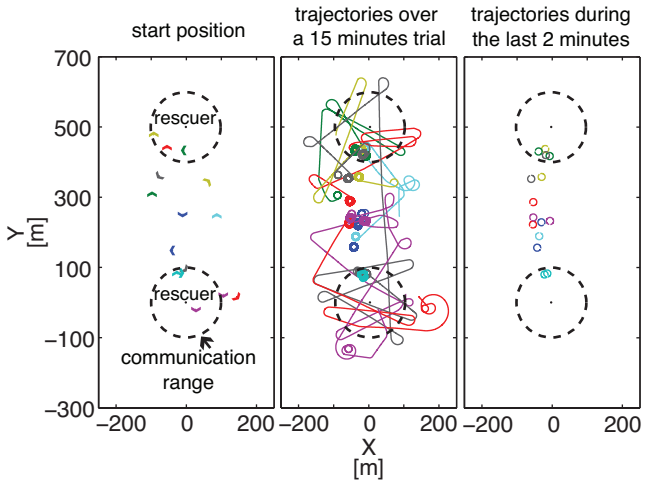


Figure 6.6: Trajectories of the 20 simulated flying robots over an entire trial duration of 15 min.

1 s. Notice that initially, the robots are randomly distributed in the environment and provide poor relay performance. As a baseline we show the performance of the system when the robots never change their position ($c = 0$). On the other hand, with $c = 0.7$ and $\Delta t_{win} = 2\pi$ (the time it takes a robot to perform a full revolution when considering a turn rate of 1 rad/s) the throughput is improved up to nearly the maximum value of 125 kbps and the latency is reduced by nearly one hop.

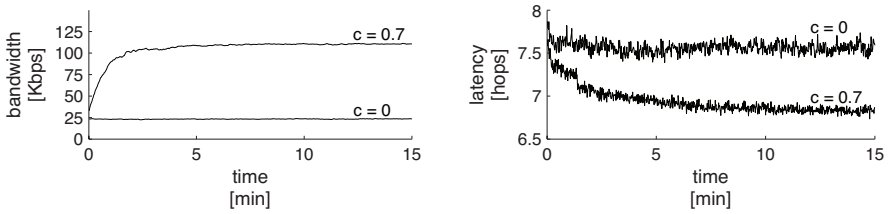


Figure 6.7: Mean network throughput and latency over 100 trials of 15 min in simulation.

6.7 Conclusion

By reverse engineering the best evolved controller from Chapter 2, we were able to identify basic principles to maintain and improve communication relays using flying robots.

In the evolved solution, robots are used to relay wireless messages between two users who are not directly within communication range. As an advantage, robots can autonomously move in their environment and in doing so, improve the performance of the wireless relay which they serve. Discovered principles are based on the fact that robots should remain on the spot when they are receiving messages from both users since they are in a useful position to relay messages. To remain on the spot, robots adopt the highest possible turn rate. However, before being stably connected to both users, robots oscillate between turning on the spot and searching the environment. As seen in Chapter 5, search behaviors can be performed by deterministically covering an area. Overtime, robots converge to a position in the network that allows for stable communication.

Controllers for communication relay are validated in simulation and with preliminary results using 10 physical robots. Discovered controllers are then extended to a simulated scenario in which 20 robots must maintain a communication network between two users in the environment conversing using voice over IP. Robots detect if they are useful for the communication relay and search for a better position otherwise. Results show that robots are able to optimize the throughput and latency of communication relay.

7 Coping with Wind

Abstract

The main contribution of this chapter is the identification of principles to mitigate the effect of wind on flying robots.

Controllers discovered in Chapter 2 were evolved in a simplified simulator without wind. Models developed throughout this thesis can be expanded to predict the effect of wind on the swarm when its direction and speed is known. However, unknown wind sources can lead robots to be pushed away from users on the ground which would cause them to get lost in a positionless system. To mitigate this effect we propose to extend individual robot motions discovered in Chapter 3 to allow robots to compensate for displacements due to wind. Rather than applying fixed turn rates, robots perform spiral trajectories that can expand in all directions at constant speed. The motion is modeled to determine the maximum amount of wind it can withstand in an application aimed at maintaining a flying robot within the communication range of a user (leashing). Experiments in theory and reality with a single flying robot show successful leashing in the case of wind or other sources of displacement such as mobile users.

This chapter is based on:

- Hauert, S., Leven, S., Zufferey, J.-C. and Floreano, D. (2010) Communication-based Leashing of Real Flying Robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 15-20.

7.1 Background

To demonstrate challenges related to noisy environments on the deployment of communication-based controllers for flying robots, we focus on a simple scenario where robots are required to remain leashed to a user on the ground in situations with wind, unreliable communication or mobile users. Ensuring wireless communication between a flying robot and a user is an intrinsic requirement to receive commands, transmit sensor data or relay information[6,27,35,53,56]. Leashing can also be used as a safety mechanism to ensure that a robot does not get lost by flying too far away from the user. This is especially interesting for testing new research algorithms on a robot or for learning how to pilot.

Previous research aimed at maintaining a flying robot within communication range of a user by setting a predetermined limit to the distance from the user at which the robot can fly[6,94]. However, distance has been shown to correlate poorly with the quality of successful radio transmissions because the propagation of radio signals depends on the environment (weather, reflections, interferences)[54].

Instead, we aim at developing a generic controller that allows fixed-wing robots to mitigate the effect of wind without the need for any position information.

7.2 Method

Following behaviors presented in Section 3.6, we propose a simple communication-based controller whereby a robot can move freely as long as it is connected to a user on the ground, and is pulled back towards the user when the extent of the leash has been reached (disconnected) as shown in Fig. 7.1. Similar controllers have been proposed in the past for ground robots[73].

The manner in which the leashing is performed highly depends on the dynamics of the platform and the noise present in the environment. Real-life conditions are such that there are often disturbing relative displacements between the user and robot due to wind, noisy communication or mobility. The most obvious reconnection behavior would be to have the robot turn-around by performing a circular trajectory with a constant turn rate as proposed in certain

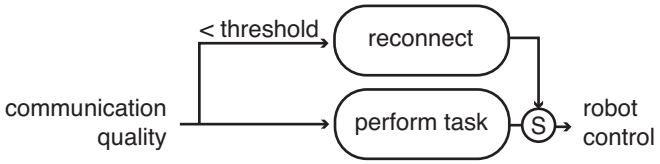


Figure 7.1: Communication-based leashing controller for a robot using the subsumption architecture[12]. Here, the reconnection behavior takes priority over the robot's task when the communication quality is too low.

previous communication-based behaviors (Sec. 3.6). However, in the case of wind, the robot might be continuously pushed away from the user (Fig. 7.2).

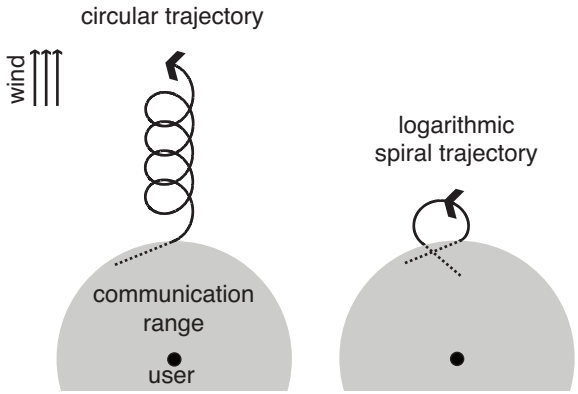


Figure 7.2: The effect of wind on the theoretical trajectory of a robot performing a circular trajectory and a logarithmic spiral. Notice that the robot performing the spiral is able to reconnect to the user while the one performing a circular trajectory drifts away.

To counter this effect, we propose to use logarithmic spirals as reconnection trajectories. Indeed, by expanding these spirals can compensate for displacements between the user and robot. As an example, Fig. 7.3 shows how a robot following a spiral trajectory will behave with strong wind in different directions. Rather than being pushed away from the spiral starting point, the robot remains anchored. Other spirals such as Archimedean curves have been studied in the literature as random search patterns for robots[48]. However, these spirals are not able to compensate a constant amount of wind in all directions and are thus unsuited as a reconnection trajectory.

We propose to theoretically determine the parameters of the spiral to be performed by the flying robot and the amount of wind that it can tolerate. The

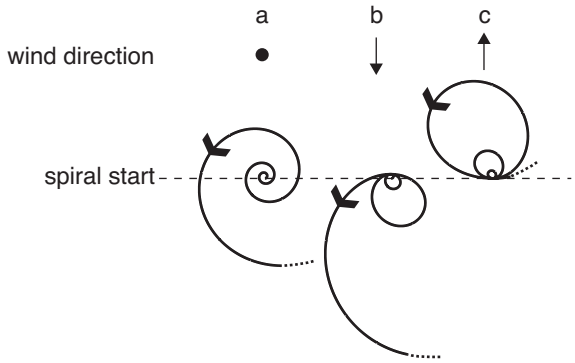


Figure 7.3: The theoretical trajectory of a robot performing a logarithmic spiral when no wind is present (a) or under the influence of wind to the South (b) or North (c). Notice that the spirals remain anchored to one position rather than drift away.

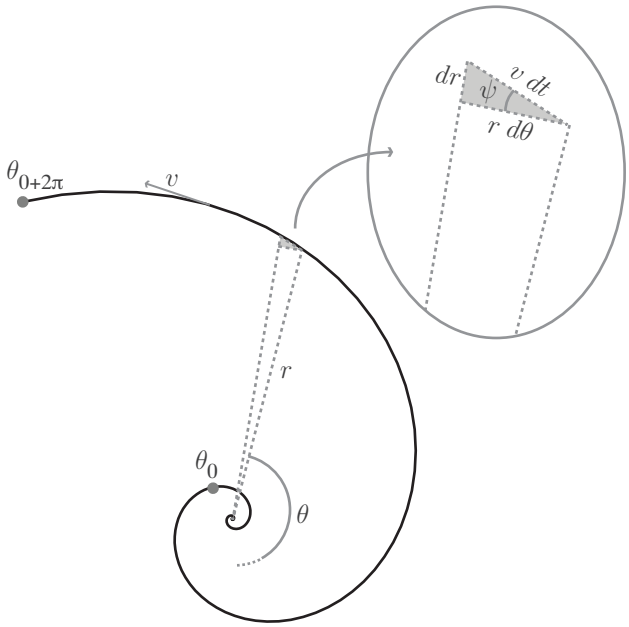


Figure 7.4: Theoretical logarithmic spiral trajectory where θ is the angle and r is the distance traveled from its origin. θ_0 is the angle at which the robot starts its trajectory and v is the speed of the robot.

theoretical wind tolerance v_{wind} is measured by the spiral expansion, which is defined as the distance between two points on the spiral which are spaced by $\Delta\theta = 2\pi$ divided by the time $t(\theta)$ needed for traveling between these points (Fig. 7.4). Starting from the polar equation of a logarithmic spiral with coefficients a and b and symbols defined in Fig. 7.4,

$$r(\theta) = a \cdot e^{b\theta} \,, \tag{7.1}$$

we can define the wind tolerance of the spiral as being

$$v_{\text{wind}} = \frac{r(\theta_0 + 2\pi) - r(\theta_0)}{t(\theta_0 + 2\pi) - t(\theta_0)} \,. \tag{7.2}$$

To solve this equation, we need to determine $t(\theta)$. Considering the local relationship of spiral variables (Fig. 7.4), we derive with Eq. 7.1

$$\tan\psi = \frac{dr}{r \cdot d\theta} = b \tag{7.3}$$

$$r \cdot d\theta = v \cdot \cos\psi \cdot dt \,. \tag{7.4}$$

For the general initial condition of a logarithmic spiral, $t = 0\,\text{s} \Rightarrow \theta = -\infty$, we obtain by integration of Eq. 7.4

$$\frac{a}{b} \cdot e^{b\theta} = v \cdot \cos\psi \cdot t \,, \tag{7.5}$$

leading to

$$t(\theta) = \frac{a}{b} \cdot \frac{e^{b\theta}}{v \cdot \cos\psi} \,, \tag{7.6}$$

with Eq. 7.3 we obtain

$$\cos\psi = \frac{1}{\sqrt{1 + \tan^2\psi}} = \frac{1}{\sqrt{1 + b^2}} \,, \tag{7.7}$$

and finally, based on Eq. 7.1, 7.2, 7.6 and 7.7 we find

$$v_{\text{wind}} = \frac{b \cdot v}{\sqrt{1 + b^2}} \,. \tag{7.8}$$

Theoretically, one could parameterize the spiral with a large b, resulting in a tolerance to wind speeds near the speed of the robot. However, large values of b lead to spirals that take too long to perform and could be too big to allow for a reconnection to the user (Fig. 7.5). For this reason, we impose that the distance between the initial point at which the robot starts the spiral at θ_0 and the point at $\theta_0 + 2\pi$ be equal to the communication range of the user r_{com}. This condition ensures that the robot is not more than $2 \cdot r_{\text{com}}$ away from its disconnection

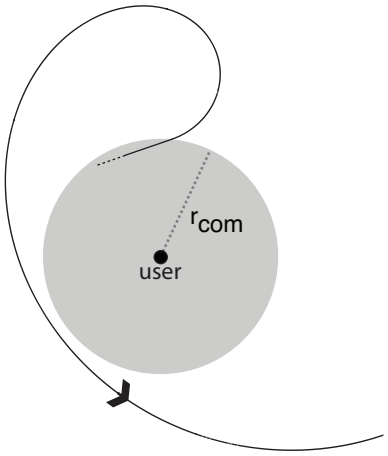


Figure 7.5: The size of the spiral is constrained by the communication range of the user r_{com} to avoid that the reconnection behavior fails, as shown here.

point, even if wind is displacing the robot by r_{com} between $t(\theta_0)$ and $t(\theta_0 + 2\pi)$. Therefore, the parameter b must satisfy the condition

$$r_{\text{com}} = r(\theta_0 + 2\pi) - r(\theta_0) \tag{7.9}$$

$$= a \cdot e^{b\theta_0} \cdot (e^{2b\pi} - 1) \, . \tag{7.10}$$

To solve this equation, we first need to obtain the initial condition θ_0, which is given by the maximum turn rate ω_0 of the considered platform. Based on Eq. 7.6 we derive

$$\theta(t) = \frac{1}{b} \cdot \ln\left(\frac{b}{a} \cdot v \cdot \cos\psi \cdot t\right) \tag{7.11}$$

such that

$$\frac{d\theta}{dt} = \omega = \frac{1}{b \cdot t} \Rightarrow \omega_0 = \frac{1}{b \cdot t_0} \, . \tag{7.12}$$

Then, using Eq. 7.7, 7.11 and 7.12 we obtain

$$\theta_0 = \frac{1}{b} \cdot \ln\left(\frac{v}{a \cdot \omega_0 \cdot \sqrt{1 + b^2}}\right) \, . \tag{7.13}$$

This yields with Eq. 7.10

$$r_{\text{com}} = \frac{v \cdot (e^{2b\pi} - 1)}{w_0 \cdot \sqrt{1 + b^2}} \, , \tag{7.14}$$

which must be solved numerically for b.

Given the time t_s that elapsed since the robot started the spiral at t_0, and the maximum turn rate w_0 of the robot, we can define a turn rate controller for the robot as:

$$\omega(t_s) = \frac{1}{b \cdot (t_0 + t_s)} \tag{7.15}$$

where, using Eq. 7.12,

$$t_0 = \frac{1}{\omega_0 \cdot b} \, . \tag{7.16}$$

Using these theoretical developments, each robot controller is designed for a regime that describes the conditions for which the behavior will function. In particular, the regime here is described as the minimum communication range of the user and maximum amount of wind in the environment. Note that, the faster the robot and the larger the communication range of the user, the higher the wind tolerance. While the focus here has been on wind, other sources of displacements such as a moving user or noisy communication can directly be addressed in the same manner.

7.3 Results

We hereby show the parameterization of robot controllers and their use to leash robots to a user in reality. We progress by first identifying the minimal communication range of the user. We then use it to parameterize the reconnection behavior (by setting b of the spiral in Eq. 7.1) and identify the amount of wind the behavior can tolerate. Within this regime, we then demonstrate that the robot is able to remain leashed to the user over long periods of time and even in different locations than where the communication was characterized. This is demonstrated in scenarios with noisy communication, wind and a mobile user.

There are many different approaches to estimating the minimum communication range of the user. This can be done by making a conservative guess based on knowledge of the radio module, or by relying directly on the value given in the data-sheet. Because of changes made to our radio module driver, we chose to characterize the radio environment experimentally. This was done using a flying robot that followed the predefined pattern shown in Fig. 7.6 (top) so as to cover a sufficiently large area around the user. Every 50 ms, the robot recorded

the number of messages it received from the user and its position. The area was then segmented into bins of 10 m x 10 m and the average number of messages received per bin over 3 separate flights was recorded. Only bins over which the robot navigated were considered for statistics. We then plot the median, minimum and maximum number of messages recorded by the bins as a function of distance. As shown in Fig. 7.6 (bottom), the minimum communication range for our experiments equals 135 m and corresponds to the first bin with a message rate of zero. This is thought to be a very conservative estimate since most messages are still transmitted for greater distances.

Using Eq. 7.14, we calculated $b = 0.3732$. The resulting controller can thus resist relative displacements up to a speed of 4.1957 m/s (Eq. 7.8). These constraints define the regime in which we use our controller for the remaining experiments.

Fig. 7.7 shows the trajectory of our aircraft performing autonomous communication-based leashing. During an entire 20 min flight, the robot was successful at spiraling towards the user when disconnected*. All of the 137 disconnections were successfully countered even though wind between 0.5 m/s and 2.4 m/s was present. Interestingly, there is an unusual point in (-10 m, 150 m) where the robot does not receive messages. This point was situated exactly above the only house within that experiment environment. Abrupt changes in the communication landscape have the same effect as a relative displacement of the robot with respect to the user and therefore need to be mitigated by the logarithmic spiral trajectories. Overall, disconnections from the user have a median value of 0.8 seconds (Fig. 7.8). Long disconnections of up to 26 seconds were recorded on rare occasions, followed by successful reconnections.

Moreover, it is interesting to see the large range of disconnection distances which is between 120 m and 258 m, further backing the case that one can not rely on distance to predict communication quality. In addition, the median communication radius is around 180 m which is, as expected, larger than our conservative measurement of 135 m.

Finally, to show that logarithmic spirals can be used to mitigate the effect of different sources of environmental noise, we present a scenario where an addi-

*Robots are considered disconnected as soon as they stop receiving messages and are considered reconnected when the number of messages received during 50 ms, low-pass filtered with a time constant of 2 s, surpasses 5 (10 being the maximum).

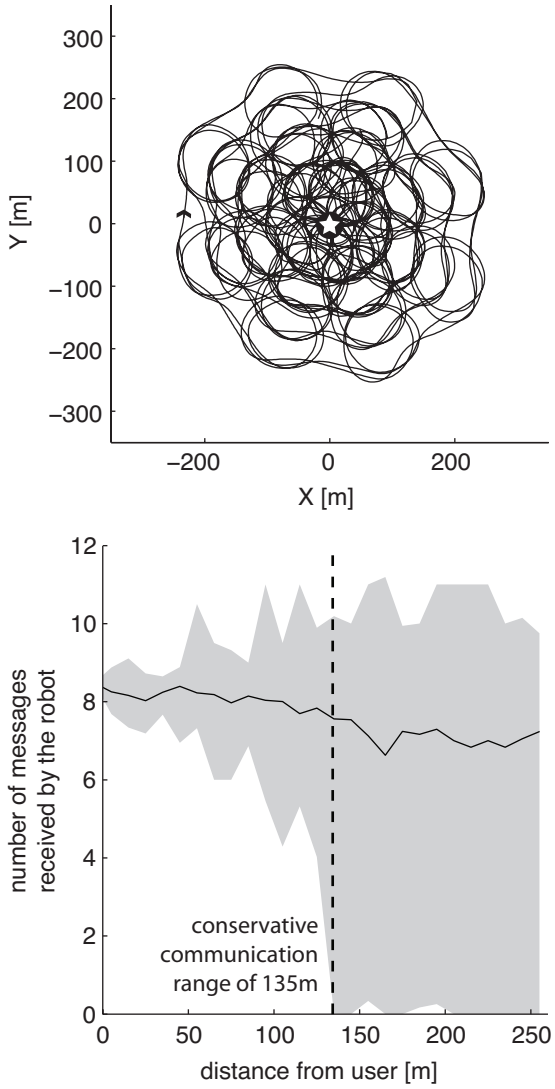


Figure 7.6: Characterization of the communication between the user (star) and a real flying robot. The top graph shows the trajectory of the robot performing the characterization. The bottom graph shows the minimum, median and maximum number of messages received by the robot over 50 ms as a function of distance from the user.

tional displacement is caused by a mobile user. Fig. 7.9 presents the trajectory of the robot when the user moves for 9 min along a road at a speed of 1 m/s with front wind between 1.1 and 1.8 m/s. As before, the robot was able to mitigate disconnections and therefore follow the user.

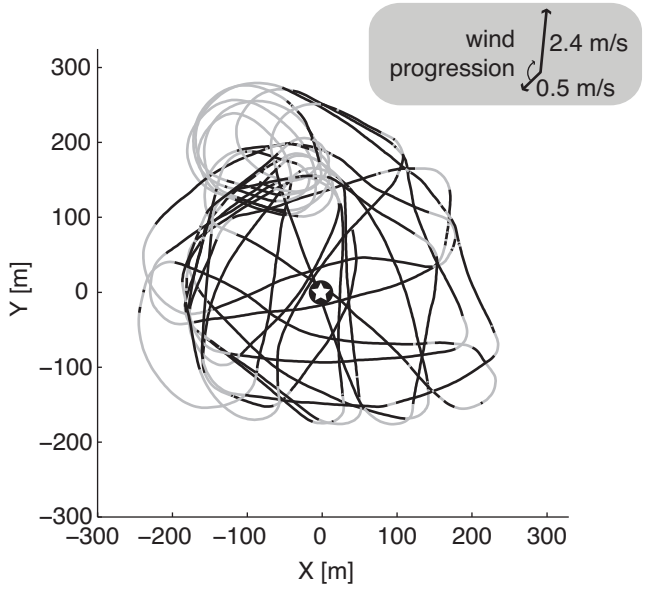


Figure 7.7: 20 min trajectory of a real flying robot leashed to a user (star) in an outdoor experiment. Here the robot flies at constant speed and 70 m altitude. Light grey lines indicate sections of the trajectory where the robot is reconnecting using the spiraling behavior while black lines indicate that the robot is performing its assigned task, in this case, straight flight.

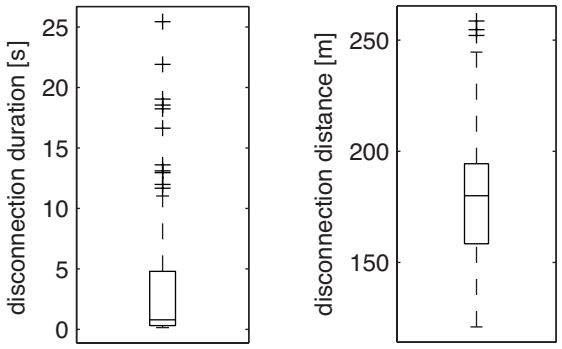


Figure 7.8: Boxplot showing the duration of disconnections between a real flying robot and a static user and the 2D distance from the user at which they occur over 137 samples.

7.4 Conclusion

By extending individual robot motions discovered in Chapter 3, we are able to mitigate unwanted robot displacements due to wind.

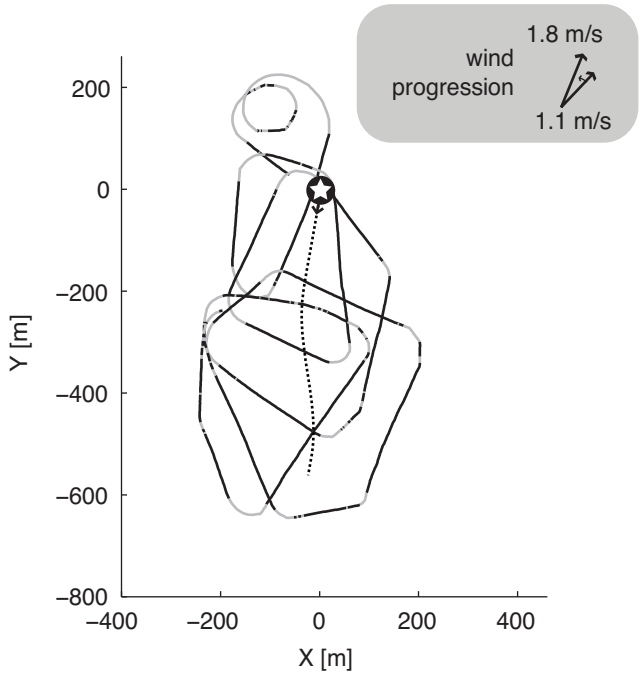


Figure 7.9: 9 min trajectory of a real flying robot leashed to a user (star) moving to the South at 1 m/s along a road in an outdoor experiment with wind pushing the robot to the North-East at a speed between 1.1 m/s and 1.8 m/s. Here the robot flies at constant speed and 70 m altitude.

More precisely, we identify logarithmic spiral trajectories as a suitable motion for fixed-wing robots because it is able to expand in all directions at a constant speed. The speed with which the spiral expands determines its wind resistance. Because of this property, spirals can easily be parameterized for a variety of situations.

Developed controllers are validated in reality using a single flying robot in a scenario aimed at maintaining a robot within the communication range of a user on the ground. Experiments are conducted in the case of moderate wind and mobile users.

In the future, spiral motions can be added as a safety mechanism to allow lost robots to reconnect to the swarm. Furthermore, it is worth exploring whether spiral trajectories can systematically be envisaged as a replacement for fixed turn rates in behaviors developed throughout this thesis.

8 Conclusion

8.1 Achievements

Swarms of flying robots have the potential to serve tomorrow's real-world applications. Their main advantage resides in being above obstacles, which makes it easier to rapidly overcome difficult terrain, provides an aerial perspective of the world and enables line-of-sight communication. However, aerial swarms have to this day only been demonstrated in simulation or in limited numbers in reality.

To bridge this reality gap, we identify two key challenges that need to be addressed. The first considers how to alleviate the need for global or relative position information for robots. Most swarm controllers rely on unreasonable assumptions concerning the capacity of robots to precisely determine their position. Furthermore, position sensors such as GPS, cameras and other range and bearing sensors result in robots becoming complicated, expensive, and unusable in certain environments that are for example deprived from GPS or from light. Instead, positionless robot controllers could be used in a large variety of environments on minimal platforms such as those developed during this project.

The second challenge considers how to harness the motion constraints of fixed-wing robots. Indeed, most literature on aerial swarming considers robots as omnidirectional vehicles that can stop and turn on the spot like ground robots or rotorcrafts. Instead, fixed-wing platforms are required to fly at relatively high speeds to avoid stalling. Slowing down their trajectory can therefore only be done by continuously turning. The minimum radius with which they can turn is bounded, meaning they can not turn on the spot.

To address these challenges, we propose to use communication rather than position as the main sensory input. Conveniently, flying robots are usually

equipped with off-the-shelf radio modules that are low-cost, light-weight, and relatively long-range. Communication hardware can serve a dual function. Their usual function is to transmit data to other robots or users. A second function exploited in this thesis looks at how flying robots can react to the reception of a message, rather than its content, by changing their behavior (communication-based behavior). Such behaviors must accommodate the challenging motion constraints of fixed-wing robots. We therefore propose to develop controllers that modulate the turn rate of flying robots while maintaining their speed and altitude constant.

No swarm controllers addressing both challenges exist in the literature. To this end, we propose a systematic approach aimed at discovering swarm controllers for flying robots. The approach uses artificial evolution for its potential to automatically discover simple and unthought-of solutions. Evolved controllers are then reverse engineered, modeled, validated and extended to a variety of swarm scenarios.

This methodology was applied to a scenario that had high potential in resulting in a variety of basic behaviors that could be useful for other swarm applications. In particular, we considered an application aimed at creating communication networks for rescuers in disaster areas. Discovered behaviors include controlling the motion of individual robots by modulating their global speed and direction based on communication and heading input. Swarms of robots can then move together in coherent groups. This is achieved by synchronizing the headings of robots over time. Synchronization is guided by messages that are emitted at regular intervals, thereby producing a beat on which robots can align. Robots are then able to form chains of synchronized robots that can translate along the communication borders of users on the ground, thereby deterministically covering a specific area. Finally, robots can maintain communication relays by circling. Individual robot motion, coherent group motion, area coverage and communication relay are the basic behaviors discovered using our methodology. Independently, each behavior can be used in a variety of scenarios that go beyond the initial evolutionary scenario presented this thesis. Examples include behaviors to allow robots or groups of robots to remain connected to a user on the ground or patrol along its communication border. Lessons learnt from the evolved controllers were extended to controllers that are able to compensate for robot displacements due to wind.

By developing positionless controllers that respect motion constraints of fixed-wing robots we aimed towards bridging the gap between simulation and reality. In order to validate this approach and the basic behaviors developed throughout this thesis, we perform experiments in reality using up to 10 safe, lightweight and low-cost platforms.

8.2 Future Work

Improvements should be made before we see aerial swarming used in real-world applications. Current shortcomings and bottlenecks to achieve this include scalability, robustness and usability issues.

Ad-hoc networks in general suffer from scalability issues since it has been shown that it is challenging to relay data over many hops[1]. It is therefore unrealistic to assume this approach can be extended to systems where large numbers of robots are needed to relay information between users. To increase the scalability of the system, robots should be able to only consider users that are closest to them to deploy local communication networks without having to relay data through a large scale network.

To validate our work, demonstrations were conducted on actual physical platforms which allows for proof-of-concept of the developed approaches. However, robustness of the system could further be ensured by verifying that swarms fulfill safeness and liveness conditions. This means checking that the swarm does what it is supposed to (liveness) while avoiding catastrophic failures (safeness)[110]. Swarm engineering principles proposed by Winfield et al. provide insight into how this can be done. Furthermore, models developed throughout this thesis are useful tools to move in this direction since they allow users to understand how the swarm should function. Additional work is also needed to control swarms of aerial robots in an intuitive manner[5,37], including using hand-held devices[82].

Generally, to increase robustness of the proposed behaviors, more work is needed to consider challenging environments with wind and noisy communication. Challenges due to wind can be addressed at three levels. First, sensing displacements due to wind could allow for robots to correct their behaviors. Second, individual behaviors such as spiral-based controllers developed in Chapter 7 can provide an essential building block to mitigate the effect of wind. Finally,

the swarm as a whole can strive to compensate for displacements due to wind by continuously self-organizing.

In order to study the effect of noisy communication, more realistic simulators could be considered that take into account terrain irregularities, robot motion, antenna orientation, weather and obstacles. A large body of work on modeling communication environments of robots has been done by Mostofi et al.[70], although for ground robots. In the shorter term, robustness could be increased by combining communication with other simple omnidirectional sensors such as sound or cameras[51] to know if robots are "within range" of each other.

Beyond the communication quality, work on routing in highly dynamic ad-hoc networks could be considered. One solution would be to merge robot simulators with simulators commonly used in the wireless community (NS-2, NS-3, QualNet or OMNeT++)[112].

Along these lines, additional efforts can be made to reduce the difference between the performance of swarm systems in reality and predictions made by theoretical models. This can be done by refining communication and motion assumptions or by improving the sensory information used by the robots.

The usability of the swarm in real-world applications will depend on the discovery of behaviors for more complex scenarios with many mobile rescuers and controllers to retract the swarm to a landing position. These behaviors can be extended from basic behaviors discovered in this thesis, or can be evolved. However, engineering evolutionary experiments is not necessarily straightforward. Rather than having to decide on a controller architecture, algorithms such as AGE might be suitable to do this automatically[65]. Complexifying the scenario might make finding any interesting controller challenging. Possible solutions can be found with incremental evolution to slowly complexify the scenario along the evolutionary generations[42].

Furthermore, reverse engineering complex controllers might be intractable by hand. One direction worth exploring consists in requiring evolution to build on easy to understand basic behaviors that have been predesigned. This approach however might limit the potential of the evolutionary approach in finding unthought-of solutions.

Finally, in addition to applications that us flying robots, one could imagine using artificial evolution to discover new swarm controllers for challenging platforms such as nanosystems, space robots or underwater robots.

A Ant-based Swarming

Abstract

The main contribution of this appendix is the development of a positionless controller inspired from army-ant foraging for the deployment of communication networks using swarms of fixed-wing robots.

As an alternative to artificial evolution we turn to biology for inspiration in the design of swarm controllers for fixed-wing flying robots aimed at creating emergency communication networks in disaster areas. More specifically, inspiration is taken from army ants which are capable of laying and maintaining pheromone paths leading from their nest to food sources in nature. This is analogous to the deployment of communication pathways between multiple rescuers. However, instead of being physically deposited in the air or on a map, pheromone is virtually deposited on the robots using local communication. This approach is investigated in 3D simulation in a simplified scenario with two rescuers.

Material for this chapter was taken from the following paper:

- Hauert, S., Winkler, L., Zufferey, J.-C. and Floreano, D. (2008) Ant-based Swarming with Positionless Micro Air Vehicles for Communication Relay. *Swarm Intelligence*, 2(2-4) pp. 167-188.

A.1 Introduction

Determining the local interactions responsible for the emergence of any swarm behavior is challenging and no methodology currently exists to deterministically solve this task. The approach presented here consists in taking inspiration from studies conducted on biological swarms to design the robot controllers. Specifically, inspiration is taken from army ants, which are capable of deploying to search for and maintain paths leading to food sources in nature by depositing and sensing pheromone in their environment[13]. Similarly, we show in simulation that we can deploy and maintain communication pathways between rescuers and then retract the swarm to its initial launching point. Finally, the developed algorithm is scalable in swarm size and reasonably robust to robot failures.

A.2 Experimental Setup

A.2.1 Scenario

The scenario consists in having a swarm of 15 robots search for a second rescuer positioned on the ground while maintaining radio connection to the rescuer from which the robots are launched. Robots must directly or indirectly (by means of other robots) remain connected to the launching rescuer in order to ensure that they do not get lost and that the swarm as a whole remains coherent. At the beginning of each deployment, a common search direction is broadcasted to each robot in the swarm which must find a randomly positioned rescuer in the area described in Fig. A.1. This area was designed to be clearly out of the communication range of the launching rescuer and is located in the North sector with respect to the launching rescuer. Robots are launched every 15 $\pm$ 7.5 seconds within a 5 m radius from the launching rescuer to model the fact that they will be launched by hand by a single human operator. The swarm is given 30 minutes to establish and maintain a communication link between the two rescuers. Communication is then interrupted and the robots must retract to the launching rescuer as rapidly as possible and land.

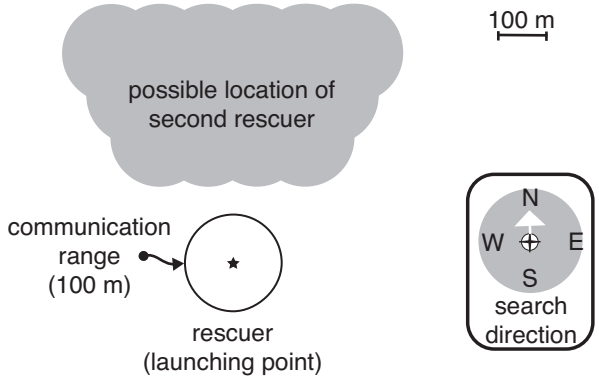


Figure A.1: The swarm composed of 15 robots must be able to find any rescuer positioned in the area in grey. Here, the search area is located to the north of the rescuer from which the robots are launched.

A.3 Control Strategy

There currently exists no methodology to deterministically design the local interactions responsible for the emergence of a desired swarm behavior. The approach presented here consists in taking inspiration from natural systems to design efficient swarm controllers for robots. Similar approaches have investigated foraging tasks with ground robots based on the trophallactic behavior performed by honey-bees[14,20,91] or the formation of robot chains between objects in the environment inspired by the observation of ant colonies foraging for food[75]. Foraging in nature has also inspired work by Campo et al.[15] for the design of robot controllers. We take inspiration from army ant colonies which are able to lay and maintain pheromone paths leading to food sources in nature, analogous to the deployment and maintenance of communication networks.

A.3.1 Army Ant Raid Patterns in Nature

Army ant colonies display complex foraging raid patterns involving thousands of individuals communicating through chemical trails (pheromone). These structures are thought to reflect an optimized mechanism to explore and exploit food resources in nature[93]. Different army ant species display different raid patterns (Fig. A.2), allowing them to adapt to different distributions of food. For example, the *E. Hamatum* hunt for sparse and large sources of food while the *E. Burchelli* can rely on uniform distributions of small food sources[34].

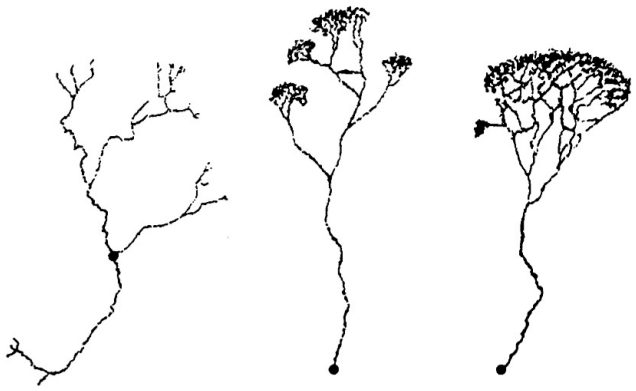


Figure A.2: Foraging pattern of three army ants *Eciton Harnatum*, *E. Rapax*, and *E. Burchelli* (from left to right) each covering some 50 m x 20 m[26].

In work by Deneubourg et al.[26], a model capable of capturing the self-organizing mechanism used for the formation of army ant raid patterns is presented. In this model, ants leave the nest at a constant rate and navigate through a binary grid of Y-junctions while depositing pheromone. At each bifurcation, the amount of pheromone on each branch influences the ant's choice to turn left or right. The speed at which the ants advance increases sigmoidally as a function of the strength of the trail's pheromone. Ants which have found food carry it back to the nest while depositing larger amounts of pheromone to reinforce successful paths. The deposited pheromone then evaporates over time.

A.3.2 Adaptation to Robots

By taking inspiration from the foraging mechanism found in army ants, we want to create and maintain communication pathways between rescuers. However, in application-oriented swarms, it is often undesirable to modify the environment in which robots deploy (by physically depositing chemicals or objects) and the deploying substrate is often unstable (e.g., air, water and quickly modifiable environments). Also, depositing virtual pheromone on a map[56,105,90] is not possible when no global positioning is available. To solve this issue in our system, pheromone is virtually deposited on the robots (pheromone robotics[79]). The approach proposed here consists of separating the robots into two types, namely nodes and ants. Nodes constitute the environment on which pheromone can be virtually deposited and read from. Ants are capable of navigating through a grid

of nodes while depositing virtual pheromone through the use of local wireless communication.

Ideally, nodes should position themselves following the Y-junction grid shown in Fig. A.3, where the length of each branch is approximately equal to the mean communication range of robots and rescuers ($\approx$ 100 m) and the junction angle is approximately of 60°[1]. Coordinates (i, j) are assigned to each node in the grid where i and j are the number of left and right branches followed to reach a desired position relative to the launching rescuer. This directional positioning is possible because robots have a magnetic compass and a fixed launching point at the root of the grid.

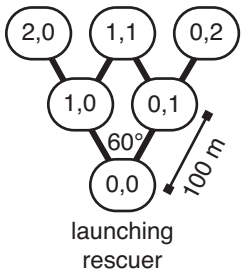


Figure A.3: Ideal positioning of the launching rescuer and nodes in the Y-junction pheromone grid. Coordinates are relative to the launching rescuer and correspond to the number of left and right branches needed to reach a position.

The ant-based swarm algorithm described here and presented in Fig. A.4 results in the deployment, maintenance and retraction of communication networks. Robots can either explore the environment as an ant or direct ants when in node state (using pheromone information) while ensuring that all robots remain connected to both rescuers.

In detail, at launch a robot is of type ant and its initial reference node is the launching rescuer positioned at the root of the pheromone grid. The refer-

[1]To the benefit of our ad-hoc network, 60° Y-junctions have the advantage of generating redundant communication pathways while maximizing the area coverage of the grid. More precisely, the chosen angle presents the advantage of generating a grid where all the nodes are at equal distance from one another. Because this distance corresponds to the mean communication range, two robots at the extremity of a Y-junction can directly communicate (unlike deployments with larger angles). Smaller angles would also have this property, however, the coverage of the deployed grid would be reduced.

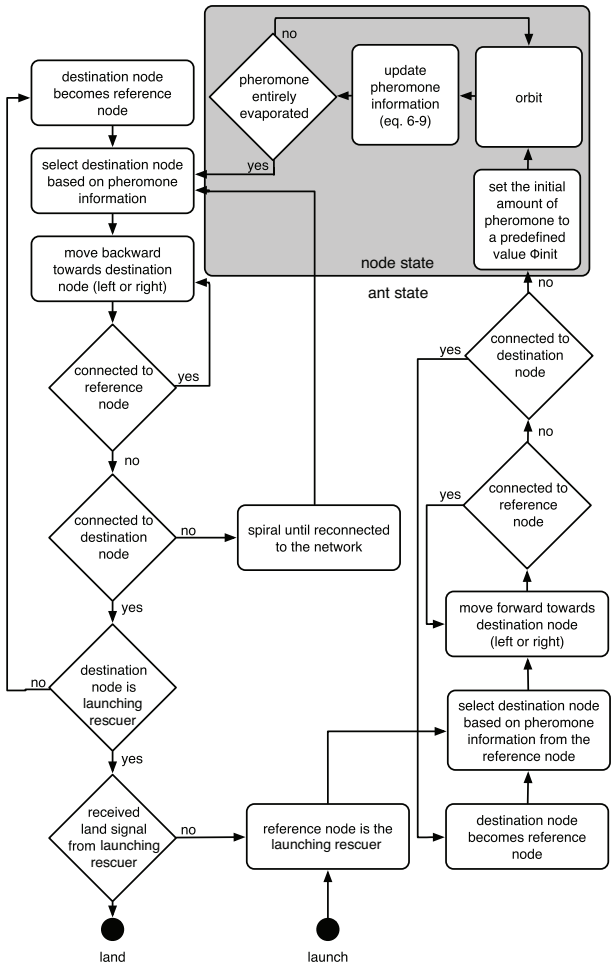


Figure A.4: Schematic representation of the behavior of a robot in our homogenous swarm. Robots, can either be in node or ant state. Initially, ants are launched from a rescuer which communicates pheromone information concerning its left and right branches. Based on this information, ants will choose a "destination node". The node which is communicating pheromone information to an ant is its "reference node". Ants navigate from "reference nodes" to "destination nodes" until the "reference node" is out of range, at which point the "destination node" becomes the ant's new "reference node" and a further "destination-node" is chosen. If the "destination node" is not within communication range, ants change state to become nodes. Finally, when a node does not store any more pheromone (evaporation) it returns to the ant state and navigates back to the launching rescuer using a similar "reference node" to "destination node" navigation, it then lands or redeploys.

ence node broadcasts the pheromone strengths of its left $\phi_{i+1,j}$ and right $\phi_{i,j+1}$ branches.[2] On reception, this information allows the ant to probabilistically choose a given branch following equations (A.1) to (A.5) where π_L and π_R are the probabilities to choose the left or right branches respectively. These equations were determined on the basis of the original equations describing the probability p_L to choose a left or respectively right p_R branch in the natural model developed by Deneubourg et al.[26]. As a result, ants favor branches with higher amounts of pheromone. The parameter μ represents the attractivity of unexplored directions and affects the amount of exploration versus path following displayed by the robots (e.g., with $\mu = 0$ ants have a probability of 0 of favoring an unexplored path over one with pheromone, whereas $\mu = \infty$ yields nearly equal probabilities of choosing a given branch over another). However, in the natural model, areas in the center of the grid had a higher probability of being explored than areas on the sides (Fig. A.5, left). For example, a robot performing a sequence of two turns had a 50% chance of reaching the position (1,1) by performing a left then right turn or a right then left turn. Reaching the positions (0,2) or (2,0) however is not as likely (25% chance of reaching any of these positions). To ensure a uniform search of the environment (Fig. A.5, right) a correction factor c is applied to adjust the probabilities described by p_L and p_R.

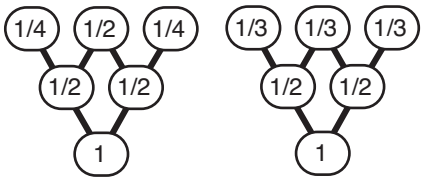


Figure A.5: Original (left) and corrected (right) probability to reach a given node in the pheromone grid given an equal amount of pheromone on each branch.

$$\pi_L(i,j) = \frac{p_L(i,j) \cdot c_L(i,j)}{p_L(i,j) \cdot c_L(i,j) + (1 - p_L(i,j)) \cdot (1 - c_L(i,j))} \tag{A.1}$$

$$\pi_R(i,j) = 1 - \pi_L(i,j) \tag{A.2}$$

[2]Initially, the pheromone strengths sent by the launching rescuer are equal to zero.

$$c_L(i,j) = \frac{i+1}{i+j+2} \tag{A.3}$$

$$p_L(i,j) = \frac{[\mu + \phi_{i+1,j}]^2}{[\mu + \phi_{i+1,j}]^2 + [\mu + \phi_{i,j+1}]^2} \tag{A.4}$$

$$p_R(i,j) = 1 - p_L(i,j) \tag{A.5}$$

Once a left or right branch is chosen, the coordinates of the node to which the ant should navigate (destination node) are stored and the ant will advance in the corresponding direction. While advancing, the ant virtually deposits $\Delta\phi_{ant}$ amounts of pheromone on its reference node.

Eventually, the ant will break the communication link with its reference node and briefly wait for a message from its destination node. If no message is received, the destination node is assumed nonexistent and the robot changes its type from ant to node with coordinates corresponding to the aimed destination and an initial amount of pheromone ϕ_{init}. However, if the destination node exists, it becomes the reference node for the ant and a new destination is chosen based on received pheromone information. Subsequently, the ant navigates through the grid until it reaches a position which is not yet occupied by a node.

Nodes maintain information concerning their own pheromone strength $\phi_{i,j}$ and the pheromone strength of adjacent nodes ($\phi_{i+1,j}$, $\phi_{i,j+1}$, $\phi_{i-1,j}$, $\phi_{i,j-1}$). This allows them to continuously update their knowledge concerning the amount of pheromone on their left and right branches in the forward and backward directions. Once the grid has formed a pathway between the rescuers, the pathway is used to relay data packets between the two. Nodes can detect if they are on a communication pathway requiring the smallest number of network hops (least-hop routes) to go from the launching rescuer to the second rescuer (see appendix B.1.1). Analogously to the mechanism whereby ants reinforce successful paths by depositing additional pheromone when carrying prey to the nest, nodes will receive an increase in pheromone ($\Delta\phi_{conn}$) when positioned along the least-hop routes.

In addition, pheromone saturates at a maximum value ϕ_{max}. To model evaporation, nodes decrease their emitted pheromone strength every time-step, following a subtractive decay rate $\Delta\phi_{decrement}$.

Once the pheromone is entirely evaporated, nodes become ants and return to the launching rescuer. No exploration is done during this retraction phase whose sole purpose is to bring the ants to the launching rescuer in a rapid and dependable manner so that it can redeploy as if newly launched. Therefore, during retraction ants always navigate along paths with highest amounts of pheromone and they never change to node state. As before, the choice of left or right branching is made based on pheromone information broadcasted by nodes (i.e., the amount of pheromone on left $\phi_{i-1,j}$ and right $\phi_{i,j-1}$ branches in direction of the launching rescuer). To avoid splitting the network, only nodes whose added coordinates are higher or equal to the added coordinates of neighboring nodes should become ants[3]. To enforce this, nodes which are required to maintain a communication link between the launching rescuer and further nodes in the grid receive $\Delta\phi_{internal}$ amounts of pheromone at each time-step. Maintaining a direct or indirect connection between all nodes and the launching rescuer is essential in a system without positioning. Equations A.6 through A.9 summarize the evolution of pheromone present on nodes over time.

$$\Delta\phi_a = \begin{cases} \Delta\phi_{internal} & \text{if node}_{i,j} \text{ is connected to node}_{k,l} \\ & \text{with coordinates such that } i+j < k+l \\ 0 & \text{otherwise} \end{cases} \tag{A.6}$$

$$\Delta\phi_b = \begin{cases} \Delta\phi_{conn} & \text{if node}_{i,j} \text{ is on the least-hop route between rescuers} \\ 0 & \text{otherwise} \end{cases} \tag{A.7}$$

$$\Delta\phi_c = n \cdot \Delta\phi_{ant} \quad \begin{array}{l} \text{where n the number of ants within communication} \\ \text{range having as a reference-node node}_{i,j} \end{array} \tag{A.8}$$

$$\phi_{i,j}(t) = max[\phi_{max}, \phi_{i,j}(t-1) - \Delta\phi_{decrement} + \Delta\phi_a + \Delta\phi_b + \Delta\phi_c] \tag{A.9}$$

Triggering the retraction of the robots is done by ending the communication between the rescuers. Because the nodes no longer receive an increase in pheromone for being on a least-hop route between the rescuers, their pheromone decreases until entirely evaporated. They then become ants and retract to the

[3] As an example, a node in (1,1) should retract after the one in (1,2) because it might be serving as communication relay between the first node and the remainder of the swarm.

launching rescuer. Once arrived, they are signaled to land through a message broadcasted by the launching rescuer. This mechanism allows for the retraction and landing of the swarm.

Finally, in case an ant loses connection with the grid for a duration greater than Δt_{lost}, it performs a spiral trajectory and eventually reconnects and proceeds to the nest.

Parameters of the ant-based algorithm used in our scenario are listed in Table A.1.

Table A.1: Parameters of the ant-based controller

Parameter	Value
ϕ_{init}	0.7 [units]
ϕ_{max}	1 [units]
μ	0.75 [units]
$\Delta\phi_{ant}$	0.002 [units per communication]
$\Delta\phi_{conn}$	0.01 [units per time-step]
$\Delta\phi_{internal}$	0.001 [units per time-step]
$\Delta\phi_{decrement}$	0.001 [units per time-step]
Δt_{lost}	2 [s]

A.3.3 Motion Primitives

The following motion primitives describe the actual behaviors of the robots in terms of physical movement of the platforms with respect to a given high-level command such as launch, land, orbit, turn left and right or avoid robots.

Launching and landing Robots perform a helicoidal trajectory until they reach a relative altitude of 20 m or the ground respectively.

Heading maintenance Allows the ant to turn following the smallest turn radius until the desired heading is met and continue straight following this heading. This is used to follow left and right Y-junction paths.

Orbiting Unlike hovering aircrafts or ground robots, fixed-wing robots must constantly remain in motion to produce lift. The motion primitive of nodes

consists in performing the smallest possible circular trajectory. In our system, this corresponds to a circle of approximately 10 m radius.

Robot avoidance Robot avoidance is done through altitude differentiation to avoid changing the turning behaviors of the robots. Nodes fly at an altitude of 20, 25 or 30 m depending on their coordinate in the grid, as can be seen in Fig. A.6. This ensures that neighboring nodes do not fly at the same altitude. Ants on the other hand must verify the target altitude of neighboring ants received through local communication and decide either to maintain their current altitude in the case where no conflicts are detected, or to adopt the lowest possible non-conflicting target altitude starting from 35 m and going up with steps of 5 m. While this constraint does not exclude all collisions (robots can collide while trying to reach their target altitude), it largely reduces them to a suitable level for a swarm system robust to failures of single nodes.

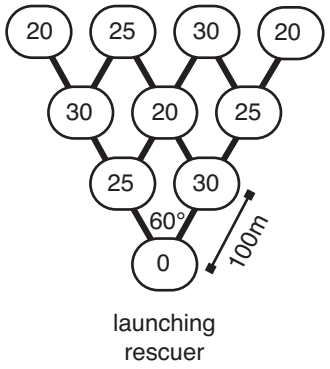


Figure A.6: Altitude assignment for node based on grid coordinates. This distribution ensures that neighboring node are at different altitudes.

A.4 Results

Here we present the qualitative behavior of the swarm, its performance in terms of search, communication, and retraction capacities and the robustness of the network to robot failures or varying swarm sizes. Experiments we run using the simulator described in Appendix B.1.2.

A.4.1 Swarm Behavior

An example of the behavior of the swarm in simulation can be seen in Fig. A.7 and in a video on our website*. Observed behaviors include the formation of grids composed of several short branches deployed in multiple directions or longer chain-like grids capable of searching in a single direction for distant rescuers. The overall network changes between different configurations until a rescuer is found. The paths requiring the least amount of hops between the rescuers are then maintained and nodes which are not positioned along these paths eventually retract and redeploy, often in more suitable positions for the communication network. Retractions are done in a wave-like manner, starting from the nodes furthest away from the launching rescuer. The rate at which the network retracts depends on the time needed to evaporate the pheromone present on the furthest nodes of the network. Retraction times could thus greatly be reduced by lowering the maximum amount of pheromone present at each node (ϕ_{max}).

Obviously, the grid formed by the nodes does not precisely follow the 60° branches of constant length envisioned in the ideal grid presented in Fig. A.3. The discrepancy between the positions of robots with respect to the envisioned pheromone grid was expected considering the noise present in the sensors and actuators of the simulated robots, the noisy communication links which control the behavior of the robots and finally, the fact that robots are not able to navigate precisely above emitting nodes but rather "loosely" within their communication range. Despite this, the swarm system is able to deploy, maintain and retract the swarm. This is due to the fact that highly precise trajectories are unnecessary when the range at which a piece of information is sensed is largely superior to the uncertainty in robot motion. Therefore instead of having robots navigate following a precise trajectory, it is more suitable to ensure that they remain within the communication range of the nodes forming the grid. In the case where a robot leaves the communication range of the grid, additional behaviors such as spiraling allow them to reconnect. Of interest is the case of the lost robot which can be seen disconnected from the swarm in Fig. A.7 and subsequently reconnected to the swarm in the following images.

*http://lis.epfl.ch/smavs

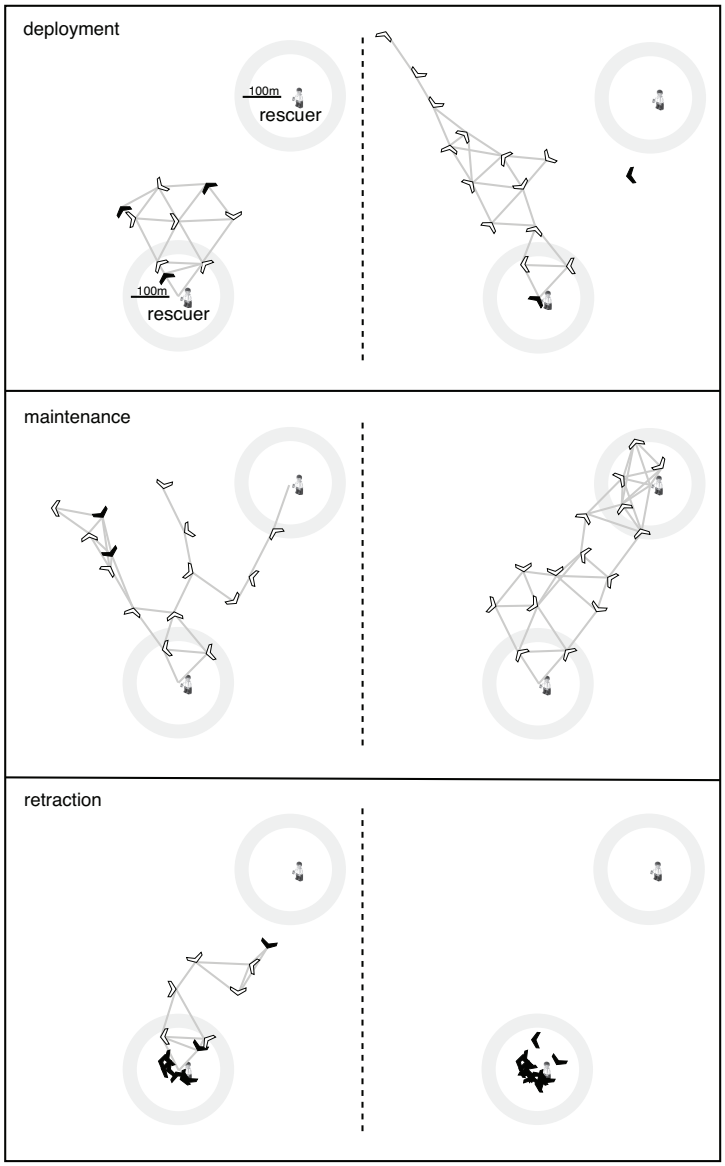


Figure A.7: Simulator screenshots showing a successful deployment, maintenance and retraction. Screenshots are temporally sequenced from left to right and from top to bottom. Towards the beginning of the trial, robots randomly favor the left of the area. The swarm is then successfully able to re-organize to search the right of the search area, find the second rescuer, maintain the connection and then finally retract and land. Nodes are white with black borders, ants in solid black and lines represent local communication links. The circle around the rescuers represents their communication range with noise.

Finally, the dynamics of the platforms and the altitude differentiation mechanism can be seen in Fig. A.8 which shows the altitude behavior of the 15 robots from the moment the first robot is launched to the landing of the last robot. As can be seen there is a clear separation in altitude between nodes (orbiting at lower altitudes of 20, 25 and 30 m) and ants (navigating above the grid).

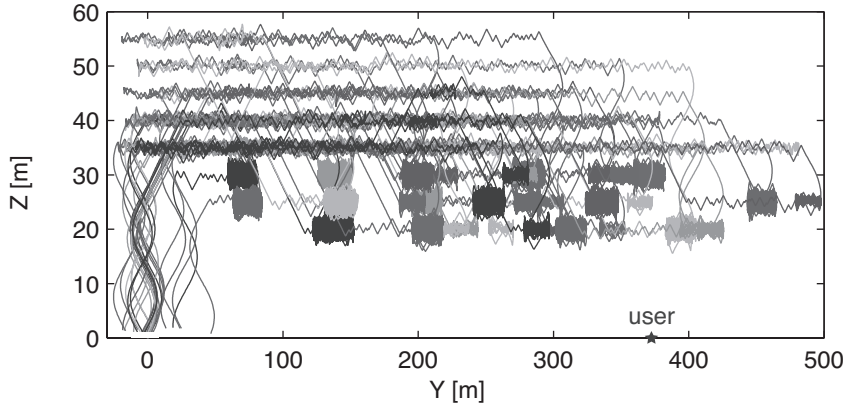


Figure A.8: Side view of the simulated trajectories of each robot in the swarm over the deployment, maintenance and retraction trial shown in Fig. A.7. The robots, are launched and land around Y=0. Nodes navigate at altitudes of 20, 25, and 30 m which can be seen by the compact orbiting trajectories at these altitudes, the ants on the other hand fly above the grid.

A.4.2 Performance

When tested on 500 trials with rescuers randomly positioned within the search area (Fig. A.1), the swarm was capable of finding more than 91% of the rescuers as seen in Fig. A.9. The mean probability of successfully delivering a packet sent from the launching rescuer to the second rescuer is measured each second and averaged over the 500 trials (Fig. A.10). In the first couple of minutes of a trial, there are typically no connections established between the rescuers because swarms have not had time to sufficiently deploy. Over time, an increasing number of trials are able to establish a connection between the rescuers. The increasing performance of the swarms indicates that established connections are maintained and improved to the end of the 30 minute trials Fig. A.10 (left). Fig. A.10 (right) presents statistics on the mean successful packet delivery probability

over 30 minutes for 500 trials. Trials where no rescuers are found have a probability 0 of delivering a packet to the rescuer, while trials in which the swarm is able to rapidly create and further maintain connections between rescuers have a probability close to 1. Finally, a median packet delivery probability of around 0.7 is largely sufficient to achieve simple communication between the rescuers.

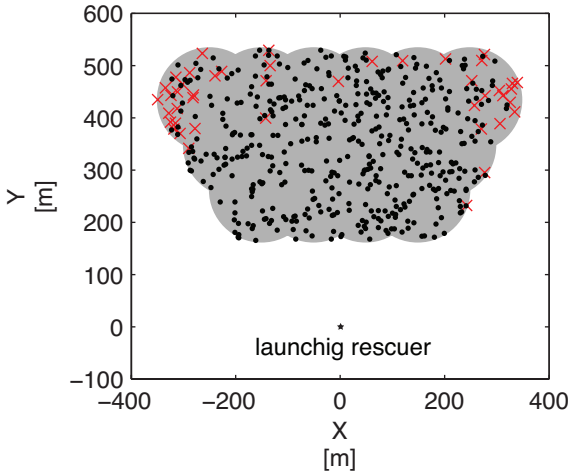


Figure A.9: Rescuers successfully connected to the launching rescuer in simulation. The search area is grey, the found rescuers are represented by points and the unfound rescuers by crosses.

Over the 500 trials, 2.6% of the 7500 robots launched were implicated in a collision with another robots and subsequently destroyed. During the retraction phase, more than 98.5% of the deployed robots (not taking into account those destroyed in collisions) were able to return to the launching rescuer. The mean retraction time is 342 s (std. dev. 81 s). Less than six minutes is reasonable for the retraction of fifteen robots as it represents 1 robot landing every 22 s.

A.4.3 Robustness

Experiments were conducted to test the robustness of the network to robot failures and to emphasize the scalability of the proposed algorithm. The scalability of the swarm is tested by deploying swarms composed of 5 to 20 robots. As seen in Fig. A.11, deploying swarms larger than 15 robots increases the perfor-

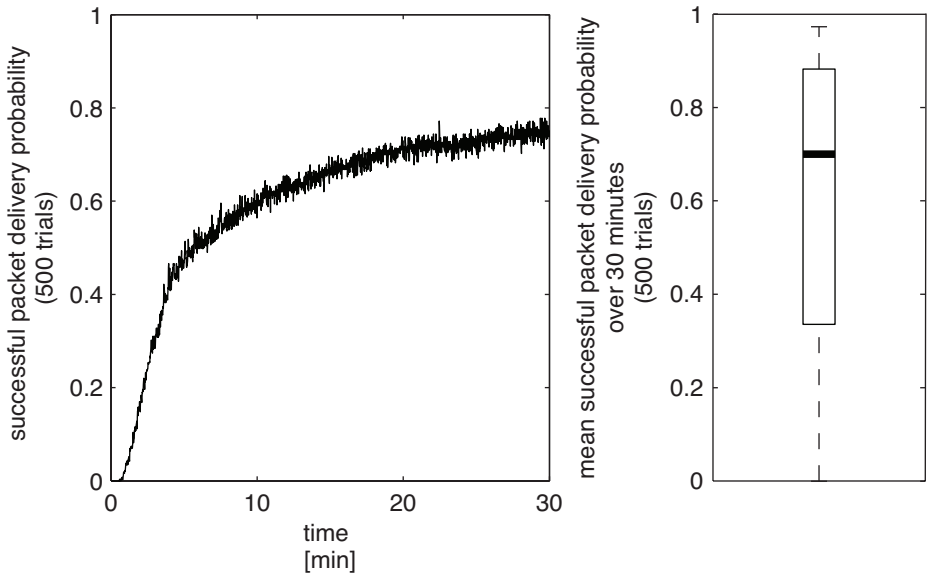


Figure A.10: Left: Mean probability of successfully delivering a packet sent from the launching rescuer to the second rescuer, measured each second and averaged over the 500 trials. Right: Mean successful packet delivery probability over 30 minutes over 500 trials.

mance of the swarm while lowering the number decreases it. This is due to the fact that small swarms have difficulties reaching rescuers positioned far from the launching rescuer and the limited amount of explorers means it would take more time to explore the same area than with larger swarms. Furthermore, it is more challenging with small swarms to create redundant communication pathways between the rescuers, thus often decreasing the quality of the connections. However, even swarms with only 5 robots are able to find nearly 50% of the rescuers over 500 trials and in some cases, maintain excellent connections with up to 98% packet delivery success. This also shows that our algorithm doesn't rely on a specific swarm size to function.

To test the robustness of the swarm to node failure, we randomly removed 0 to 10 robots from an initial swarm composed of 15 robots at sequentially random moments in the 30 minute trials. In Fig. A.12, we qualitatively show that swarms with 1 or 2 failures perform comparably to systems with no failures while the performance with more failures degrades gracefully, with examples of swarms which can even withstand 10 robot failures. Notice that the probability of finding

a rescuer decreases with the number of failures, which is similar to the effect seen when deploying small swarms, as mentioned previously.

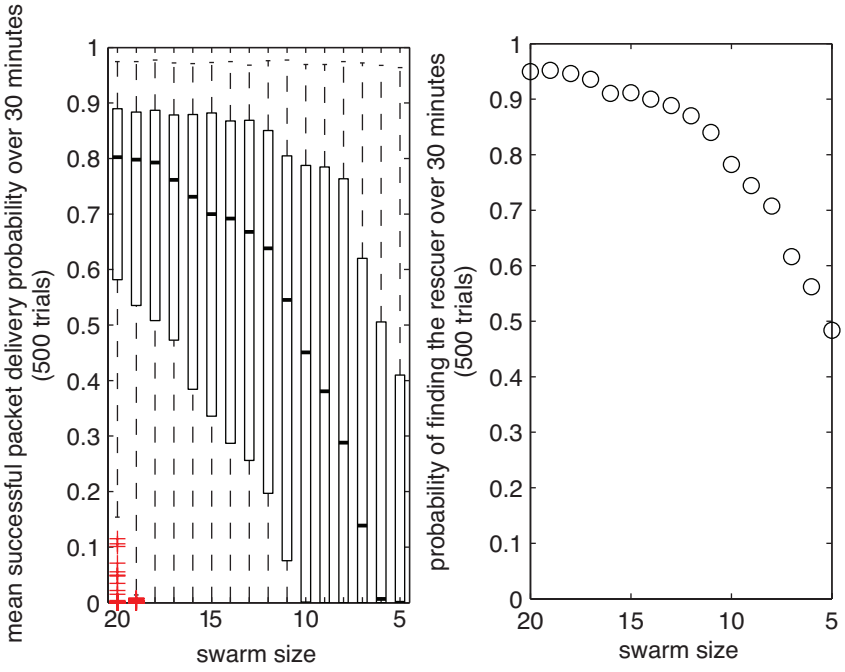


Figure A.11: Left: Mean successful packet delivery probability over 30 minutes when swarms of 5 to 20 robots are deployed. Each box represents data from 500 trials. Right: Probability of finding the rescuer over 500 trials of 30 minutes when swarms of 5 to 20 robots are deployed.

A.5 Discussion

While our swarm algorithm works robustly in cases of robot failures or varying swarm size deployments, additional measures must be taken to mitigate the problem of windy environments. Here we discuss challenges due to wind and propose three directions in which solutions can be found without requiring profound changes to the basic control strategy. Challenges arise from the fact that wind translates the individual robots following a random direction and strength. For example, one can imagine constant global wind pushing the robots away from the launching rescuer which is fixed. Ultimately, the swarm would disconnect from the launching rescuer and get lost. To counteract the effects of wind the following approaches could be investigated:

- **Mitigation at the autopilot level**: The low-level autopilot is responsible for controlling a robot based on commands sent from the motion primitives described in section A.3.3. Typical commands include setting the desired turn rate, pitch rate and speed of the robot. To reliably execute these commands, autopilots could reactively adapt the attitude of a robot with respect to measurements provided by a wind sensor (e.g., based on optic flow,[86]) . Because the effect of wind is compensated at the lower-levels of control, no modifications are needed at the level of swarm control.

- **Mitigation at the individual level**: Each robot attempts to counteract the effect of wind by sensing that it is not at its intended "position" in the pheromone grid. This detection can be derived from information concerning the robots within its neighborhood. For example, orbiting robots can sense if they are fully connected, mostly connected to or disconnected from a neighbor or the launching rescuer and comparisons can be made on its intended position with respect to the grid-coordinates of neighbors over time. Once a displacement has been sensed robots should perform local search patterns, possibly spirals, allowing them to improve their position with respect to neighbors.

- **Mitigation at the swarm level**: Instead of mitigating the effect of wind on the individual level, the swarm as a whole can strive to fix robot displacements. Unpublished initial results show for example that frequently replacing the node in the network by newly-deployed ones largely reduces the drift of the swarm. This is due to the fact that newly-deployed robots have drifted for a shorter amount of time than node already present in the network. Because the swarm can withstand imprecisions in the position of its robots (section A.4.1) small displacements due to wind over a short time are assimilated by the swarm. In our algorithm this "refresh" mechanism can be implemented by systematically swapping roles between newly launched ants and nodes. As a result, ants involved in a swap would change state to node and receive the coordinates and pheromone levels displayed by the swapping node which would continue as if it were a passing ant.

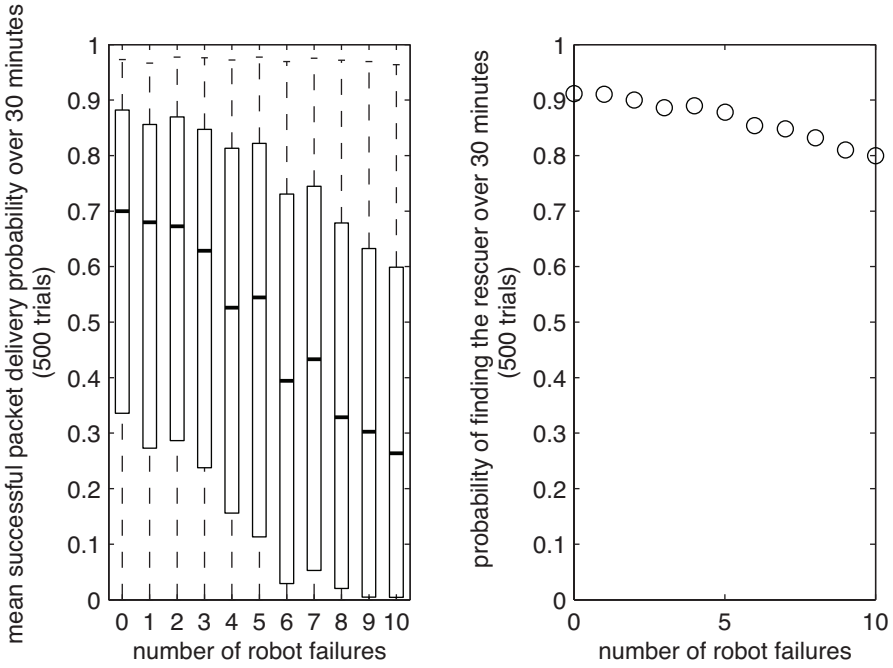


Figure A.12: Left: Mean successful packet delivery probability over 30 minutes when 0 to 10 robots are removed sequentially at random times from the initial swarm composed of 15 robots. Each box represents data from 500 trials. Right: Probability of finding the rescuer over 500 trials of 30 minutes when 0 to 10 robots are removed sequentially at random times.

A.6 Conclusion

This work provides insight into the design of positionless aerial swarms based on army-ant foraging.

The deployment, maintenance and retraction of a swarm of robots for the creation of wireless communication networks in disaster areas is demonstrated in 3D simulation with trajectories realistic for fixed-wing robots. Because the development of local interactions responsible for swarming is an unsolved problem, inspiration is taken from the biological models of the deployment, maintenance and retraction of pheromone paths deposited by army ants between their nest and varying distributions of food sources in nature. When adapted to a swarm of 15 robots, the system is capable of deploying an efficient communication network between two rescuers and subsequently retracting robots to their initial launching point. In addition we show that the swarm is scalable and robust to robot failures. Because robots do not rely on sensors which are dependent on

the environment or expensive in terms of weight, energy and monetary cost, this work paves the way towards minimalist aerial swarms applicable in most environments in a rapid, inexpensive, scalable and simple manner.

B Materials

In this appendix we present the software and hardware used throughout this thesis including both robot simulators and the aerial testbed.

B.1 Simulation

Two simulators implemented in c++ were sequentially developed during this thesis. The first simulator in 2D was used for evolutionary experiments. The simulator was then extended to make it more realistic for all other experiments in simulation.

B.1.1 2D Simulator

Flight model

The fixed-wing robots follow a first-order flight model based on experiments run on the actual robot platforms. Equations B.1 through B.6 modify the position (x, y) of the platforms after each time-step of duration dt based on a constant speed v and turn rate ω .

$$x(t) = x(t - dt) + v \cdot cos(\omega \cdot dt) \cdot dt \tag{B.1}$$

$$y(t) = y(t - dt) + v \cdot sin(\omega \cdot dt) \cdot dt \tag{B.2}$$

Robots fly at a speed of 14 m/s affected by uniform noise in the range [-5,5] m/s and are unable to hover or make sharp turns, their minimum turn radius being of 18 m. Uniform noise in the range [-5,5] $^\circ$/s is added to their turn rate. A smoothing function ensures that the turn rate can not be modified abruptly (the maximum change in turn rate is of 100 $^\circ$/s). Such physical constraints enforce a more complex controller with respect to ground robots or hovering platforms.

The only internal sensor used for swarming was a heading measurement sensor affected by Gaussian noise with a standard deviation of 5°.

Communication

Robots can communicate with other robots and users. The communication model assumes that communication between two robots is perfect if the individuals are less than 90 m apart, noisy from 90 m to 100 m and inexistent when separated by 100 m or more. The probability of entirely dropping a message increases linearly between 90 m and 100 m from 0 to 1. Similar disc models have been used in robotic swarm simulators[73] based on assumptions introduced in work by Winfield et al.[109].

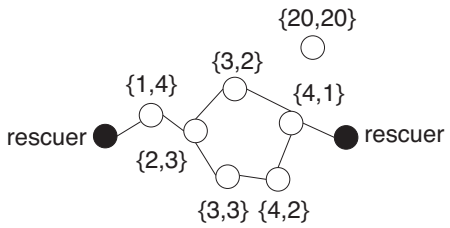


Figure B.1: Topology of the swarm determined using local communication. The rescuers are represented by black circles, the robots by white ones and the local communication links by the lines connecting them. Tags above each robot represent the number of hops from the launching rescuer and the second rescuer respectively. Robots that are isolated from the swarm receive the default values $N = 20$.

Robots and users can send two types of messages, control messages and data messages. Control messages are only used for the coordination of the swarm and are broadcasted by each robot every 50 ms. Data messages are related to the application of the swarm (e.g. video relay, voice relay, etc.), and are sent between the rescuers every 50 ms (the direction of the message flow is non-relevant). A data message is assumed to have reached its destination if there exists, at that given time-step, at least one communication pathway between the rescuers.

Information about the topology of the swarm or the connection status of a robot to the rescuers can be acquired using local wireless communication. More specifically, robots can determine the minimum number of network hops needed for a message to go from a rescuer to themselves using only local communication (Fig. B.1). Furthermore, robots can approximate their hop count (h) to any rescuer using equation B.3.

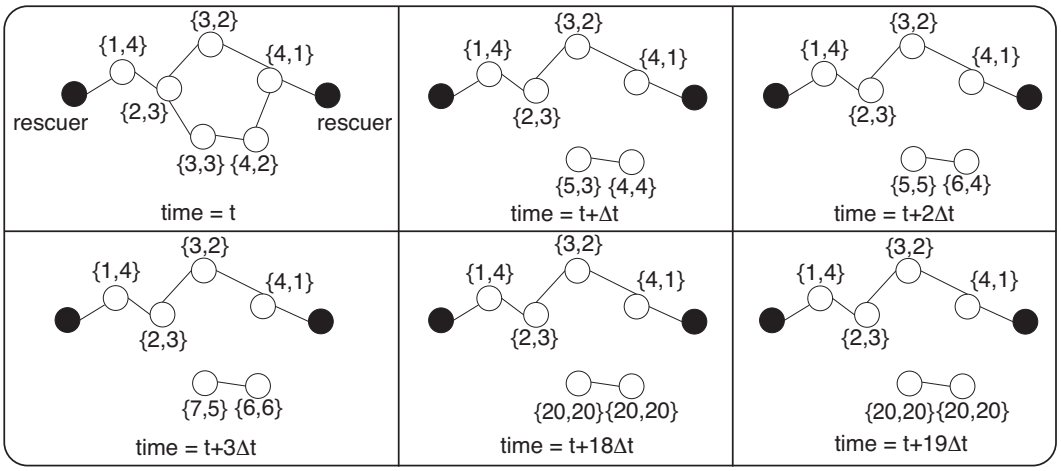


Figure B.2: Effect of the disconnection of a group of robots from the swarm. The rescuers are represented by black circles, the robots by white ones and the local communication links by the lines connecting them. Tags above each robot and the launching rescuer represent the number of hops from the launching rescuer and the second rescuer respectively. Messages are sent every time-step (Δt=50 ms).

$$h(s,i,t) = \begin{cases} N & t = 0 \\ N & n(i) = \oslash \\ 1 & s \in g(i) \\ min(\underset{n_i \in n(i)}{\text{argmin}}(h(s,n_i,t-1)) + 1, N) & otherwise \end{cases}$$

$$\begin{array}{lcl} s & = & \text{rescuer index} \\ i & = & \text{robot index} \\ t & = & \text{timestep of 50 ms duration} \\ n(i) & = & \text{robots in the neighborhood of i} \\ g(i) & = & \text{rescuers in the neighborhood of i} \\ N & = & \text{maximum number of hops between a} \\ & & \text{robot and the rescuer} \\ & & \text{(N=20 in a scenario with 20 robots)} \end{array} \quad \text{(B.3)}$$

While this approach tends towards the correct topology of the network when the swarm is stable, it can obviously be momentarily locally inexact because of

the dynamics of the network and because hop information needs to propagate throughout the network at a speed of 1 hop per 50 m/s. Another possibility would be to have rescuers broadcast a notification message which would then flood the network and almost instantaneously update the hop count of all robots with respect to the initiating rescuer. We believe however that our solution has several advantages over this solution. First of all, it is scalable in the number of rescuers (additional ground stations increase slightly the size of messages sent between robots but do not increase the number of messages which need to be sent). Also, our approach is truly decentralized and dynamic in that it does not rely on the rescuers to initiate a flooding mechanism but only on local communication between robots.

In addition, robots can detect if they are connected to the rescuers either directly or indirectly thanks to the mechanism described in Fig. B.2. This mechanism is derived from equation B.3, which pushes the hop information of robots disconnected from the rescuers to increment until the cutoff value N. Robots that have reached the cutoff value are assumed disconnected.

Finally, rescuers can also compute their hop count to other rescuers (path length), this information can be propagated throughout the network. Robots can then determine if they are positioned along the shortest communication pathway between two rescuers by comparing this path length to the addition of their hop count to both rescuers. If there is an equality, then the robot is on the shortest path between the two rescuers.

B.1.2 3D Simulator

The 2D simulator was extended to model robot trajectories and communication in a more realistic manner. Unlike the previous simulator, this one is event-based in order to model the fact that each robot has its own internal clock and that communication is in general asynchronous. Only modifications from the 2D simulator are given here.

Flight model

The flight model of the robot from the 2D simulator is extended to the third dimension as shown in equations B.4 through B.5 which modify the position (x, y, z) of the platforms every dt seconds based on a desired speed v, turn rate

ω and altitude change rate $\dot{h}$.

$$x(t) = x(t - dt) + v \cdot cos(\omega \cdot dt) \cdot dt \tag{B.4}$$

$$y(t) = y(t - dt) + v \cdot sin(\omega \cdot dt) \cdot dt \tag{B.5}$$

$$z(t) = z(t - dt) + \dot{h} \cdot dt \tag{B.6}$$

In this model, robots fly at a speed of 10 m/s affected by uniform noise in the range [-1,1] m/s and are unable to hover or make sharp turns, their minimum turn radius being around 10 m. Uniform noise in the range [-5,5] °/s is added to the turn rate of the robot. A smoothing function ensures that the turn rate can not be modified abruptly (the maximum change in turn rate is of 90 °/s). The altitude change rate is of maximum 5 °/s. This rate is affected by uniform noise in the range [-1,1] °/s and its maximum change is of 5 °/s.

Communication

This simulator implements lower layers of the open systems interconnection (OSI) model, namely the network layer, data-link layer and physical layer for 802.11b wireless communications.

Network Layer The network layer is responsible for implementing the routing protocols for relaying data messages to users. Routing consists in flooding data messages throughout the network. To do so, each router re-broadcast received packets only the first time they are received.

Data-link Layer In the data-link layer we implement the medium access control (MAC) data communication protocol which takes care of coordinating access to the physical medium. More specifically, we implemented the carrier sense multiple access with collision avoidance (CSMA/CA) protocol described in IEEE specifications for 802.11b[1]. Based on the CSMA/CA protocol, wireless modules sense the physical medium before transmitting a packet. If the medium is busy, the module chooses a random back-off time after which a retransmission will be attempted. While being able to

[1]http://standards.ieee.org/

avoid collisions between fully connected neighbors, it can not avoid hidden node terminals which might be a source of collisions in our network.

Physical Layer The shadowing propagation model[30] was used to probabilistically determine the range of inter-robot transmissions and transmissions with users on the ground following equation B.7. Packets sent a distance d are assumed received if the $P_r(d)$ is greater than the receiver's sensitivity threshold S_r. When a node receives multiple packets simultaneously, it calculates the signal-to-noise ratio of the strongest received signal to the sum of other received signal strengths and the ambient noise n. If this ratio is larger than SNR_{thresh}, the packet is correctly received. Otherwise, all packets collide and are discarded.

$$P_r(d)[dBm] = P_t[dBm] - 10 \cdot \beta \log\left(\frac{d}{d_0}\right) + X[dB] \tag{B.7}$$

where X is a Gaussian random variable with zero mean and standard deviation σ_{dB}.

Parameters of the model were set to achieve a communication range of approximately 100 m. A summary can be found in Table B.1. The received signal strength as a function of the distance between transmitter and receiver can be seen in Fig. B.3.

Parameter	Value
transmit output power P_t	16 dBm
receive sensitivity S_r	-82 dBm
path loss exponent β	4.9
shadowing deviation σ_{dB}	2 dB
signal-to-noise threshold SNR_{thresh}	10 dB
ambient noise n	-102 dBm

Table B.1: Parameters of the communication model.

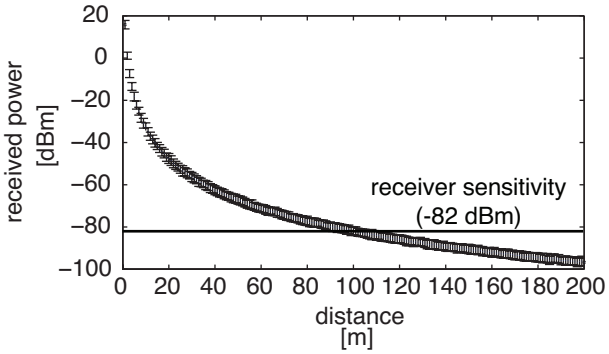


Figure B.3: Wireless signal propagation with log-normal shadowing.

B.2 Flying Testbed

All the necessary software and hardware to perform experiments with 10 flying robots was developed in the scope of this project. To the best of our knowledge, this setup is the one with the most outdoor aerial robots to this day.

Deploying large aerial swarms requires significant technological developments which are presented here.

B.2.1 Platform

Swarm systems are generally composed of large numbers of simple individuals that are eventually disposable. Furthermore, swarm algorithms often do not ensure the correct functioning of each individual in the swarm. For these reasons, flying robots aimed towards real-world applications must be low-cost, easy to transport and deploy, easy to build and maintain, and safe.

To fulfill all these conditions, a flying testbed composed of 10 fixed-wing aerial robots was developed specifically for this project by Severin Leven. The robots runs on a LiPo battery and have an autonomy of 30 min.

In particular, the flying platform shown in Fig. B.4 is light weight (420 g) and has an 80 cm wingspan. It is built out of Expanded Polypropylene (EPP) with an electric motor mounted at the back and two control surfaces serving as elevons (combined ailerons and elevator). The robot is equipped with an autopilot for the control of altitude, airspeed and turn rate that provides an interface for receiving commands from a navigation controller[59]. Embedded in the autopilot is a micro-controller that runs a minimalist control strategy based

on input from only 3 sensors: one gyroscope and two pressure sensors. The robot is further equipped with a compass.

The swarm algorithms presented in this thesis were implemented on a Toradex Colibri PXA270 CPU board running Linux, connected to an off-the-shelf USB WiFi dongle (Fig. B.5). The output of this high-level computer, namely a desired turn rate, is sent as control command to the autopilot. Altitude and airspeed commands during experiments remained constant at a value between 50 m and 150 m and 12 m/s, respectively. In order to log flight trajectories, the robot was further equipped with a u-blox[2] LEA-5H GPS module and an XBee PRO transmitter. GPS is also used to estimate wind for reporting in aerial experiments.

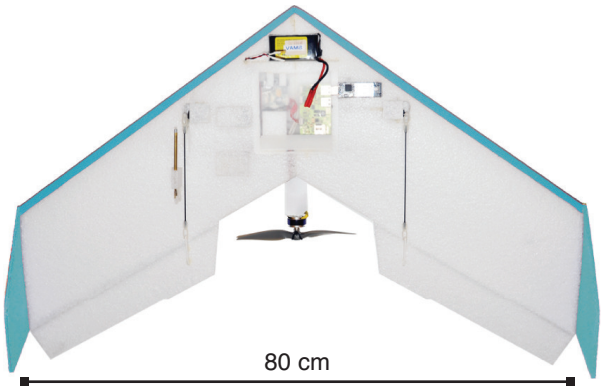


Figure B.4: Safe flying wing for outdoor experiments made out of soft material and with a back-mounted propeller. The robot is equipped with an autopilot, embedded Linux, WiFi dongle, magnetic compass and GPS+XBee (only for logging purposes).

For the WiFi communication, Netgear[3] WNDA3100 dongles were used that implement the 802.11n standard and transmit in the 5 GHz band. This is interesting with respect to transmissions in the 2.4 GHz band because it allows for less interference with the considerable number of devices currently used in this band. Dongles are configured for ad-hoc mode and have a communication range of nearly 500 m line-of-sight. For the purpose of certain experiments, the range of communication could be reduced by modifying the driver of the WiFi module. Drivers were further simplified to prevent modules from scanning the

[2]http://www.u-blox.com

[3]http://netgear.com

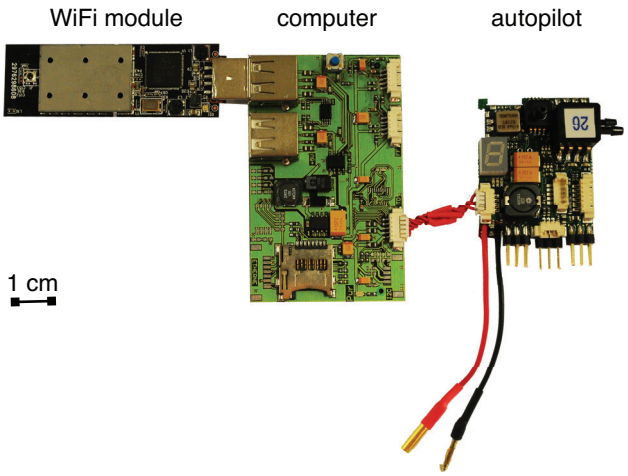


Figure B.5: Control electronics including the autopilot and the adapter-card interfacing to the Colibri Linux board and the USB WiFi dongle.

network for alternative connections. Instead, the network cell for the swarm was hardcoded in every robot as were the IPs.

Finally, rescuers on the ground are connected to the ad-hoc network using identical wireless hardware as the robots.

B.2.2 Base Station

During experiments, the swarm is operated and monitored from a computer on the ground using an intuitive graphical interface developed by Beyeler et al.[10] and extended during this thesis to accommodate swarm experiments (see Fig. B.6 for a screenshot). The base station is connected to 3 XBee receivers to accommodate data from 10 robots.

The interface allows a user to connect to the robots using XBee PRO modules, monitor the behavior of all robots on a large map of the environment and select individual robots to monitor their parameters and sensor values individually. Using this interface, the user can interact with individual robots or the entire swarm. To help the user cope with the large number of robots, each robot is painted in a different color. This color is used to draw the robot on the map or select the robot in the interface. Finally a single click is sufficient to log all the data being received from the robots in separate time-stamped files that are named after the colors of the robots.

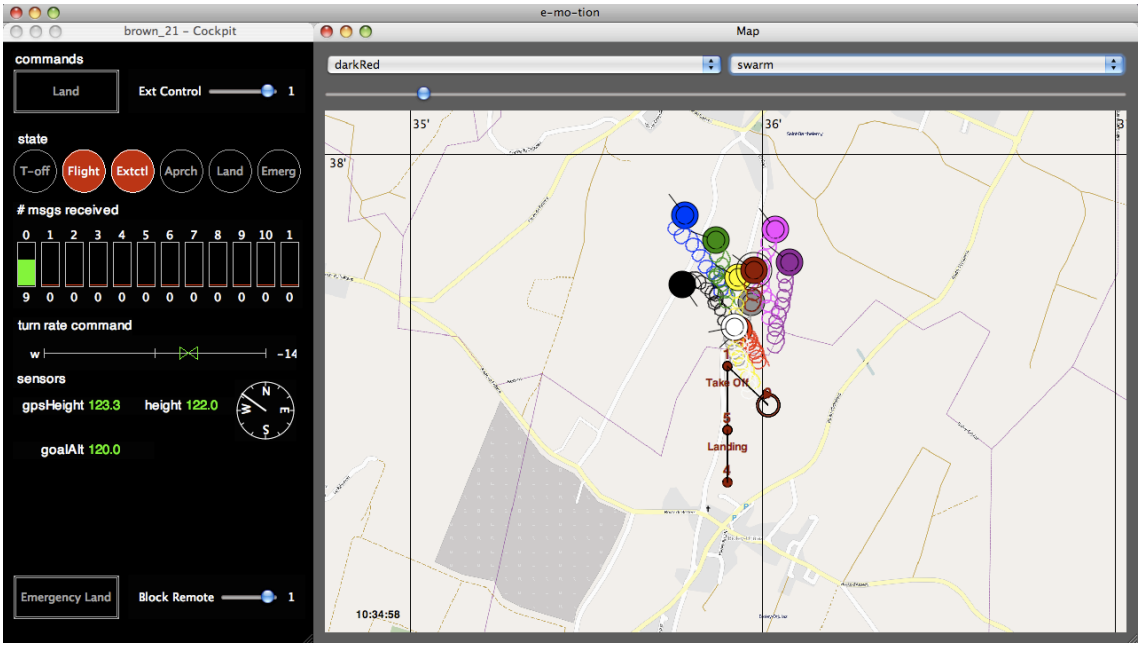


Figure B.6: User interface for the control of swarms of flying robots.

B.2.3 Experimental Setup

To perform successful swarm experiments in a safe and efficient manner, an experimental protocol needs to be designed. The protocol presented here follows years of experience with nearly 200 flights and can be adapted to most swarm systems. The result is a setup where a single operator can perform swarm experiments, although those performed during this thesis were usually done with the help of Severin Leven. This simple setup allows for experiments of around 30 min with at least 10 robots and minimal interactions from the operator. A video of the entire setup developed within this project can be seen at http://lis.epfl.ch/smavs/.

Location

The location for aerial experiments is important since it should be relatively uninhabited and should allow good visibility.

For these reasons, experiments were conducted over mostly flat farm land in Bioley-Orjulaz in Switzerland (see Fig. B.7) which is a 20 min drive from the

EPFL. Authorization was given by the Swiss Federal Office for Civil Aviation[4] to perform swarm experiments below 150 m in this area.

Figure B.7: Experimental area. The launch location is marked by "SMAVNET office".

Transporting material to and from the experiment site is burdensome and can cause damage to all the hardware involved. Therefore, the office shown in Fig. B.8 was installed in the experimental terrain with authorization from the village. The office used as shelter during difficult weather conditions was also used to store all 10 robots, a power generator, tools, a table and chairs.

Planting an orange office in farmland and flying ten robots above the area is considered odd and sometimes disturbing by the inhabitants. It is therefore necessary to make information visible by the experiment cite. In our case, this was done with posters on the office explaining the project and encouraging inhabitants to come ask questions. As a result, the project was very often presented and demonstrated to the people from the surrounding villages. Public relations is no doubt key to the success of these types of experiments.

[4]http://www.bazl.admin.ch/index.html?lang=en

Figure B.8: Office used for shelter and to store material.

Experimental Protocol

First, the base station is setup with one computer connected to three XBee receivers (for 10 robots) and wifi hardware to join the robots' ad-hoc network. The computer runs the swarm interface for the operation and monitoring of the swarm.

Robots are set on the ground at the position from which they will be launched. The robots are then equipped with batteries and propellers. Once ready, all the robots are booted on the floor. It typically takes one minute for the robots to boot and around two minutes for the robots to have a GPS fix. During this time, the operator connects to the robots one after the other, always checking that the connection was successful. The robots are then launched one after the other. This is done by tilting the nose of the robot to the sky. Robots react to this action by taking off. This is much easier than having to click a button on the interface

Figure B.9: Experimental setup for operation and monitoring of robots.

for every robot and allows the operator to move away from the computer. After take-off, robots navigate to a "stand-by" way-point which they continuously circle. Robots are spaced out from 50 m to 150 m altitude with steps of 10 m to avoid any collisions. Notice that this requires launching the robots in the correct order, with the highest altitude first. Alternative methods for inter-robot collision avoidance have been studied within this project in simulation[46] and in reality[60].

Swarm experiments are launched by sending a swarm command called "swarm" from the interface. On reception of this message, robots adopt turn rates sent from swarm algorithms implemented on the linux board to the autopilot. Experiments can be stopped at any time by sending the swarm command "stand-by" at which point the robots return to the previous way-point. This allows for quick swarm experiments that can be stopped at any time. The interface can also be used to change controller parameters on-line, updates can be made at the level of the swarm or the individual. When experiments are finished, the operator can initiate landing by sending the swarm command "land". The robots then need to be retrieved around the landing site.

Safety

Many things can go wrong during aerial swarm experiment, robots can break, fly away or crash. One of the main challenges in performing swarm experiments is reaching the level of trust necessary in the system to increase the number of robots in the air. This is acquired through safety mechanisms implemented for every possible failure imaginable. Safety mechanisms presented here were useful in ensuring robust operation of flying robots during nearly 200 flights.

One of the main challenges for which almost nothing can be done relates to hardware breakdowns. Possible solutions include checking the material frequently and having extremely robust and simple platforms such as ours. To make sure that robots don't lose their propellers, two rubber-bands are used.

Otherwise, GPS is used for most safety operations. The first safety mechanisms therefore is to ground the robot if GPS is lost. Such landings are called emergency landings and the robot simply turns off its motor and glides in circles until it reaches the ground, thereby landing softly. Emergency landings are the most inconvenient because they require the operator to retrieve the robot wherever it has landed.

Furthermore, to prevent robots from flying away, a virtual safety box is set up around the experimental area. When robots leave the box, their state is set to "stand-by" and they return to the corresponding waypoint. Finally, normal landing is initiated automatically when the robot's battery is low.

Notice that thanks to these safety mechanisms, experiments could be conducted entirely without an operator since the robots take-off, go into stand-by and land on their own when needed. Furthermore, even though GPS is used for the experimental protocol, it is never used during swarming and can be entirely removed once the system has been thoroughly tested using controllers proposed in this thesis.

Bibliography

[1] Abdelzaher, T., Prabh, S., and Kiran, R. (2004). On real-time capacity limits of multihop wireless sensor networks. pages 359–370, Piscataway. IEEE Press.

[2] Alidaee, B., Wang, H., and Landram, F. (2009). A note on integer programming formulations of the real-time optimal scheduling and flight path selection of UAVs. *IEEE Transactions on Control Systems Technology*, 17(4):839–843.

[3] Allred, J., Hasan, A. B., Panichsakul, S., Pisano, W., Gray, P., Huang, J., Han, R., Lawrence, D., and Mohseni, K. (2007). SensorFlock: an airborne wireless sensor network of micro-air vehicles. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 117–129, New York. ACM Press.

[4] Altshuler, Y., Yanovsky, V., Wagner, I., and Bruckstein, A. (2008). Efficient cooperative search of smart targets using UAV swarms. *Robotica*, 26(4):551—-557.

[5] Bashyal, S. and Venayagamoorthy, G. (2008). Human swarm interaction for radiation source search and localization. In *IEEE Swarm Intelligence Symposium*, pages 1–8.

[6] Basu, P., Redi, J., and Shurbanov, V. (2004). Coordinated flocking of UAVs for improved connectivity of mobile ground nodes. In *Proceedings of the IEEE Military Communications Conference*, volume 3, pages 1628–1634, Piscataway. IEEE Press.

[7] Beard, R. W., Mclain, T. W., Nelson, D. B., Kingston, D., and Johanson, D. (2006). Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE*, 94(7):1306–1324.

[8] Beni, G. (2004). From swarm intelligence to swarm robotics. In *8th International Conference on Simulation of Adaptive Behavior*, pages 1–9.

[9] Bertuccelli, L., Alighanbari, M., and How, J. (2004). Robust planning for coupled cooperative UAV missions. In *IEEE Conference on Decision and Control*, volume 17, pages 2917–2922, Piscataway. IEEE Press.

[10] Beyeler, A., Magnenat, S., and Habersaat, A. (2008). Ishtar: a flexible and lightweight software for remote data access. In *Proceedings of the European Micro Air Vehicle Conference*.

[11] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

[12] Brooks, R. A. and Connell, J. (1986). Asynchronous distributed control system for a mobile robot. In *Proceedings of SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering*, pages 77–84.

[13] Burton, J. L. and Franks, N. R. (1985). The foraging ecology of the army ant *Eciton rapax*: an ergonomic enigma? *Ecological Entomology*, 10(2):131–141.

[14] Camazine, S., Crailsheim, K., Hrassnigg, N., Robinson, G. E., Leonhard, B., and Kropiunigg, H. (1998). Protein trophallaxis and the regulation of pollen foraging by honey bees (*Apis mellifera L.*). *Apidologie*, 29(1):113–126.

[15] Campo, A. and Dorigo, M. (2007). Efficient multi-foraging in swarm robotics. In *Advances in Artificial Life*, volume 4648 of *Lecture Notes in Artificial Intelligence*, pages 696–705. Springer, Berlin.

[16] Chang, D., Shadden, S., Marsden, J., and Olfati-Saber, R. (2003). Collision avoidance for multiple agent systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 539–543, Piscataway. IEEE Press.

[17] Cliff, D., Husbands, P., and Harvey, I. (1993). Explorations in evolutionary robotics. *Adaptive Behavior*, 2:73–110.

[18] Cole, D., Goktogan, A., and Sukkarieh, S. (2006). The demonstration of a cooperative control architecture for UAV teams. *Experimental Robotics*, 39:501–510.

[19] Cortés, J., Martínez, S., Karatas, T., and Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on robotics and automation*, 20(2):243–255.

[20] Crailsheim, K. (1998). Trophallactic interactions in the adult honeybee (*Apis mellifera L.*). *Apidologie*, 29(1):97–112.

[21] Crowther, B. (2004). Rule-based guidance for flight vehicle flocking. *Proceedings of the Institute of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 218(2):111–124.

[22] Cummings, M., Nehme, C., Crandall, J., and Mitchell, P. (2007). Predicting operator capacity for supervisory control of multiple UAVs. *Innovations in Intelligent UAVs: Theory and Applications*, 70:11–37.

[23] Daniel, K., Rohde, S., Goddemeier, N., and Wietfeld, C. (2010). A communication aware steering strategy avoiding self-separation of flying robot swarms. In *5th IEEE International Conference on Intelligent Systems*, pages 254–259.

[24] De Nardi, R. (2004). *Flocking of UAVs Software model and limited vision simulations*. PhD thesis, University of Essex.

[25] De Nardi, R., Holland, O., Woods, J., and Clark, A. (2006). SwarMAV: A swarm of miniature aerial vehicles. In *Proceedings of the 21st International UAV Systems Conference*.

[26] Deneubourg, J. L., Goss, S., Franks, N. R., and Pasteels, J. M. (1989). The blind leading the blind: modeling chemically mediated army ant raid patterns. *Journal of Insect Behavior*, 2(5):719–725.

[27] Dixon, C. and Frew, E. W. (2009). Maintaining optimal communication chains in robotic sensor networks using mobility control. *Mobile Networks and Applications Journal*, 14(3):281–291.

[28] Etkin, B. and Reid, L. D. (1996). *Dynamics of flight - Stability and control*. John Wiley & Sons, New York.

[29] Fantacci, R., Marabissi, D., and Tarchi, D. (2009). A novel communication infrastructure for emergency management: the In.Sy.Eme. vision. *Wireless Communications and Mobile Computing*.

[30] Fenton, L. (1960). The sum of log-normal probability distributions in scatter transmission systems. *Proceedings of the IEEE Transactions on Communications Systems*, 8(1):57–67.

[31] Finio, B. M. and Wood, R. J. (2010). Distributed power and control actuation in the thoracic mechanics of a robotic insect. pages 2755–2762. in press.

[32] Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. MIT Press, Cambridge.

[33] Floreano, D., Mitri, S., Magnenat, S., and Keller, L. (2007). Evolutionary conditions for the emergence of communication in robots. *Current Biology*, 17:514–519.

[34] Franks, N. R., Gomez, N., Goss, S., and Deneubourg, J. L. (2001). The blind leading the blind in army ant raid patterns: Testing a model of self-organization (*Hymenoptera: Formicidae*). *Journal of Insect Behavior*, 4(5):583–607.

[35] Frew, E. W., Dixon, C., Elston, J., and Stachura, M. (2009). Active sensing by unmanned aircraft systems in realistic communication environments. In *IFAC Workshop on Networked Robotics*.

[36] Gancet, J., Hattenberger, G., Alami, R., and Lacroix, S. (2005). Task planning and control for a multi-UAV system: architecture and algorithms. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1017–1022, Piscataway. IEEE Press.

[37] Gancet, J., Motard, E., Naghsh, A., Roast, C., Arancon, M., and Marques, L. (2010). User interfaces for human robot interactions with a swarm of robots in support to firefighters. In *IEEE International Conference on Robotics and Automation*, pages 2846–2851.

[38] Gaudiano, P., Bonabeau, E., and Shargel, B. (2005). Evolving behaviors for a swarm of unmanned air vehicles. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 317–324, Piscataway. IEEE Press.

[39] Ghabcheloo, R., Pascoal, A., Silvestre, C., and Kaminer, I. (2007). Non-linear co-ordinated path following control of multiple wheeled robots with bidirectional communication constraints. *International Journal of Adaptive Control and Signal Processing*, 21(2-3):133–157.

[40] Ghommam, J., Saad, M., and Mnif, F. (2008). Formation path following control of unicycle-type mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 1966–1972, Piscataway. IEEE Press.

[41] Goldenberg, D. K., Lin, J., Morse, A. S., Rosen, B. E., and Yang, Y. R. (2004). Towards mobility as a network control primitive. In *Proceedings of the 5th international symposium on mobile ad hoc networking and computing*, pages 163–174, New York. ACM Press.

[42] Gomez, F. and Mikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342.

[43] Guglieri, G., Quagliotti, F. B., and Speciale, G. (2008). Optimal trajectory tracking for and autonomous uav. *Automatic Control in Aerospace (online journal)*, 1:1–9.

[44] Hauert, S. (2006). Simulation of swarming mavs for communication relay. Master's thesis, École Polytechnique Fédérale de Lausanne.

[45] Hauert, S., Mitri, S., Keller, L., and Floreano, D. (2010). Evolving cooperation: From biology to engineering. In *The Horizons of Evolutionary Robotics*. MIT Press, Cambridge, USA.

[46] Hauert, S., Winkler, L., Zufferey, J.-C., and Floreano, D. (2008). Ant-based swarming with positionless micro air vehicles for communication relay. *Swarm Intelligence*, 2(2–4):167–188.

[47] Hauert, S., Zufferey, J., and Floreano, D. (2009). Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.

[48] Hayes, A. T., Martinoli, A., and Goodman, R. M. (2003). Swarm robotic odor localization: Off-line optimization and validation with real robots. *Robotica*, 21(4):427–441.

[49] Hoffmann, G., Rajnarayan, D., Waslander, S., Dostal, D., Jang, J., and Tomlin, C. (2004). The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). In *Proceedings of the 23rd Digital Avionics Systems Conference*, volume 2, pages 1–10, Piscataway. IEEE Press.

[50] How, J., King, E., and Kuwata, Y. (2004). Flight demonstrations of cooperative control for uav teams. In *AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit*, Reston, VA. AIAA Press. AIAA paper AIAA-2004-6490.

[51] Hrabar, S. and Sukhatme, G. S. (2009). Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431–452.

[52] Jang, H., Lien, Y., and Tsai, T. (2009). Rescue information system for earthquake disasters based on MANET emergency communication platform. In *Proceedings of the International Conference on Wireless Communications and Mobile Computing Connecting the World Wirelessly*, pages 623—-627, New York. ACM Press.

[53] Kadrovach, B. A. and Lamont, G. B. (2001). Design and analysis of swarm-based sensor systems. In *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, volume 1, pages 487–490, Piscataway. IEEE Press.

[54] Kim, K.-H. and Shin, K. G. (2006). On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 38–49, New York. ACM Press.

[55] Kovacina, M., Palmer, D., Yang, G., and Vaidyanathan, R. (2002). Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, volume 3, pages 2782–2788, Piscataway. IEEE Press.

[56] Kuiper, E. and Nadjm-Tehrani, S. (2006). Mobility models for UAV group reconnaissance applications. In *Proceedings of the IEEE International Conference on Wireless and Mobile Communications*, page 33, Piscataway. IEEE Press.

[57] Lawrence, D., Donahue, R., Mohseni, K., and Han, R. (2004). Information energy for sensor-reactive UAV flock control. In *Proceedings of the AIAA "Un-*

manned Unlimited" Technical Conference, Reston, VA. AIAA Press. AIAA paper AIAA-2004-6530.

[58] Lawrence, D. A., Frew, E. W., and Pisano, W. J. (2008). Lyapunov vector fields for autonomous unmanned aircraft flight control. *Journal of Guidance, Control and Dynamics*, 31(5):1220–1229.

[59] Leven, S., Zufferey, J.-C., and Floreano, D. (2009). A minimalist control strategy for small UAVs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2873–2878, Piscataway. IEEE Press.

[60] Leven, S., Zufferey, J.-C., and Floreano, D. (2010). Mid-air collision avoidance in dense collective aeiral systems. *Journal of Field Robotics*. in press.

[61] Li, Q. and Jiang, Z. P. (2008). Formation tracking control of unicycle teams with collision avoidance. In *Conference on Decision and Control*, pages 496–501, Piscataway. IEEE Press.

[62] Lipson, H. and Pollack, J. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–8.

[63] Manoj, B. and Baker, A. H. (2007). Communication challenges in emergency response. *Communications of the ACM*, 50(3):51–53.

[64] Mataric, M. J. (1994). *Interaction and Intelligent Behavior*. PhD thesis, MIT.

[65] Mattiussi, C. and Floreano, D. (2007). Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on Evolutionary Computation*, 11(5):596–607.

[66] Merino, L., Caballero, F., Martínez-de Dios, J. R., Ferruz, J., and Ollero, A. (2006). A cooperative perception system for multiple UAVs: application to automatic detection of forest fires. *Journal of Field Robotics*, 23:165–184.

[67] Mirollo, R. E. and Strogatz, S. H. (1990). Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50:1645–1662.

[68] Mitri, S. (2009). *The evolution of communication in robot societies*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne.

[69] Moshtagh, N., Jadbabaie, A., and Daniilidis, K. (2005). Vision-based distributed coordination and flocking of multi-agent systems. In *Proceedings of Robotics: Science and Systems*.

[70] Mostofi, Y., Gonzalez-Ruiz, A., Gaffarkhah, A., and Li, D. (2009). Characterization and modeling of wireless channels for networked robotic and control systems–a comprehensive overview. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 4849–4854. IEEE.

[71] Murphy, R. R., Pratt, K. S., and Burke, J. L. (2008). Crew roles and operational protocols for rotary-wing micro-UAVs in close urban environments. In *Proceedings of the 3rd International Conference on Human Robot Interaction*, pages 73–80, New York. ACM Press.

[72] Nelson, D., Barber, D., McLain, T., and Beard, R. (2007). Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529.

[73] Nembrini, J., Winfield, A., and Melhuish, C. (2002). Minimalist coherent swarming of wireless networked autonomous mobile robots. In *From Animals to Animats 7, Proceedings of the 7th International Conference on Simulation of Adaptive Behavior*, pages 273–382. MIT Press, Cambridge.

[74] Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines.* MIT Press, Cambridge.

[75] Nouyan, S., Campo, A., and Dorigo, M. (2008). Path formation in a robot swarm. *Swarm Intelligence*, 2:1–23.

[76] Oh, E. S. (2003). Information and communication technology in the service of disaster mitigation and humanitarian relief. In *Proceedings of the IEEE 9th Asia-Pacific Conference on Communications*, volume 2, pages 730–734, Piscataway. IEEE Press.

[77] Oyekan, J. and Huosheng, H. (2009). Toward bacterial swarm for environmental monitoring. In *Proceedings of the IEEE International Conference on Automation and Logistics*, number August, pages 399–404, Piscataway. IEEE Press.

[78] Park, C. S., Tahk, M. J., and Bang, H. C. (2003). Multiple aerial vehicle formation using swarm intelligence. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Reston, VA. AIAA Press. AIAA paper AIAA-2003-5729.

[79] Payton, D., Daily, M., Estowski, R., Howard, M., and Lee, C. (2001). Pheromone robotics. *Autonomous Robots*, 11:319–324.

[80] Peng, H., Li, Y., Wang, L., and Shen, L. (2008). Hormone-inspired cooperative control for multiple uavs wide area search. In *Proceedings of the 4th international conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications-with Aspects of Theoretical and Methodological Issues*, volume 5226 of *Lecture Notes in Computer Science*, pages 808–816. Springer, Berlin.

[81] Poduri, S., Pattem, S., Krishnamachari, B., and Sukhatme, G. S. (2009). Using local geometry for tunable topology control in sensor networks. *IEEE Transactions on Mobile Computing*, 8:218–230.

[82] Quigley, M., Goodrich, M., and Beard, R. (2004). Semi-autonomous human-UAV interfaces for fixed-wing mini-UAVs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2457–2462.

[83] Reynolds, C. W. (1987). Flocks, herds and schools: a distributed behavioral model. In *SIGGRAPH Computer Graphics*, volume 21, pages 25–34, New York. ACM Press.

[84] Richards, A., Bellingham, J., Tillerson, M., and How, J. (2002). Co-ordination and control of multiple UAVs. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Reston, VA. AIAA Press. AIAA Paper AIAA-2002-4588.

[85] Richards, M. D., Whitley, D., and Beveridge, J. R. (2005). Evolving cooperative strategies for UAV teams. In *Proceedings of the Genetic And Evolutionary Computation Conference*, volume 2, pages 1721–1728, New York. ACM Press.

[86] Rodriguez, A., Andersen, E., Bradley, J., and Taylor, C. (2007). Wind estimation using an optical flow sensor on a miniature air vehicle. In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, Reston, VA. AIAA Press. AIAA paper AIAA-2007-6614.

[87] Romero, J. and Machado, P., editors (2007). *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Natural Computing Series. Springer Berlin.

[88] Ruini, F., Cangelosi, A., and Zetule, F. (2009). Extending the evolutionary robotics approach to flying machines: An application to mav teams. *Neural Networks*, 22(5-6):812–821.

[89] Şahin, E. (2005). Swarm robotics: from sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20, Berlin. Springer.

[90] Sauter, J. A., Matthews, R., Parunak, H. V. D., and Brueckner, S. A. (2005). Performance of digital pheromones for swarming vehicle control. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 903–910, New York. ACM Press.

[91] Schmickl, T. and Crailsheim, K. (2007). Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm. *Autonomous Robots*, 25(1-2):171–188.

[92] Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to autonomous mobile robots*. Bradford Book, MIT Press, Cambridge.

[93] Solé, R., Bonabeau, E., Fernàndez, P., and Marìn, J. (2000). Pattern formation and optimization in army ant raids. *Artificial Life*, 6(3):219–226.

[94] Spears, W. M., Spears, D. F., Heil, R., Kerr, W., and Hettiarachchi, S. (2005). An overview of physicomimetics. In *Simulation of Adaptive Behaviour, Workshop on Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 84–97, Berlin. Springer.

[95] Stirling, T., Wischmann, S., and Floreano, D. (2010). Energy-efficient indoor search by swarms of simulated flying robots without global information. *Swarm Intelligence*, 4(2):117–143.

[96] Strogatz, S. H. and Stewart, I. (1993). Coupled oscillators and biological synchronization. *Scientific American*, 269(6):102–109.

[97] Støy, K. (2001). Using situated communication in distributed autonomous mobile robotics. In *Proceedings of the 7th Scandinavian Conference on Artifcial Intelligence*.

[98] Surette, M. G., Miller, M. B., and Bassler, B. L. (1999). Quorum sensing in escherichia coli, salmonella typhimurium, and vibrio harveyi: A new family of genes responsible for autoinducer production. *Proceedings of the National Academy of Sciences*, 96(4):1639–1644.

[99] Tekdas, O., Yang, W., and Isler, V. (2010). Robotic routers: Algorithms and implementation. *International Journal of Robotics Research*, 29(1).

[100] Thompson, A., Layzell, P., and Zebulum, R. S. (1999). Explorations in design space: unconventional electronics design through artificial evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196.

[101] Urzelai, J. and Floreano, D. (2001). Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolutionary Computation*, 9(4):495–524.

[102] Uzol, O. and Yavrucuk, I. (2008). Collaborative target tracking for swarming MAVs using potential fields and panel methods. In *Proceedings of the 2008 AIAA Guidance, Navigation and Control Conference*, Reston, VA. AIAA Press. AIAA paper AIAA-2008-7167.

[103] Valenti, M., Bethke, B., Dale, D., Frank, A., McGrew, J., Ahrens, S., How, J. P., and Vian, J. (2007). The MIT indoor multi-vehicle flight testbed. In *Proceedings of the International Conference on Robotics and Automation*, pages 2758–2759, Piscataway. IEEE Press.

[104] Van Dyke Parunak, H., Brueckner, S., and Sauter, J. (2002). Digital pheromone mechanisms for coordination of unmanned vehicles. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 449–450, New York. ACM Press.

[105] Van Dyke Parunak, H., Brueckner, S. A., and Sauter, J. (2005). Digital pheromones for coordination of unmanned vehicles. In *Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, pages 246–263. Springer, Berlin.

[106] Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., and Sochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229.

[107] Waibel, M., Keller, L., and Floreano, D. (2009). Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computation*, 13:648–660.

[108] Welsby, J. and Melhuish, C. (2001). Autonomous minimalist following in three dimensions: A study with small-scale dirigibles. *Proceedings of Towards Intelligent Mobile Robots. Technical Report Series, Manchester University, Department of Computer Science*.

[109] Winfield, A. F. T. (2000). Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In *Proceedings of Distributed Autonomous Systems 4*, pages 273–282, Berlin. Springer.

[110] Winfield, A. F. T., Harper, C. J., and Nembrini, J. (2005). Towards dependable swarms and a new discipline of swarm engineering. In *Proceedings of the SAB Swarm Robotics Workshop*, volume 3342 of *Lecture Notes in Computer Science*, pages 126–142. Springer, Berlin.

[111] Yang, Y., Minai, A. A., and Polycarpou, M. M. (2005). Evidential map-building approaches for multi-UAV cooperative search. In *Proceedings of the IEEE American Control Conference*, pages 116–121, Piscataway. IEEE Press.

[112] Ye, W., Vaughyan, R. T., Sukhatme, G. S., Heidemann, J., Estrin, D., and Matarić, M. M. (2001). Evaluating control strategies for wireless-networked robots using an integrated robot and network simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2941–2947, Piscataway. IEEE Press.

[113] Zarzhitsky, D. and Spears, D. (2005). Swarm approach to chemical source localization. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1435—1440, Piscataway. IEEE Press.

[114] Zou, Y., Pagilla, P., and Ratliff, R. (2009). Distributed formation flight control using constraint forces. *Journal of Guidance, Control, and Dynamics*, 32(1):112–120.

[115] Zufferey, J.-C., Hauert, S., Stirling, T., Leven, S., Roberts, J., and Floreano, D. (2010). *Handbook of Collective Robotics*, chapter Aerial collective systems. Pan Stanford Publishing.

Curriculum Vitae

I was born on the 16th of July 1983 in the USA from Swiss parents. After living in Rochester NY for 10 years, I moved to Switzerland, where I completed all my studies. After choosing a scientific curriculum in middle school and high school, I entered the Ecole Polytechnique Fédérale de Lausanne in 2001 to study Computer Science. As a student, I won 2nd place at the Swiss Programming Contest "Logiquest" for the development of applications for mobile phones. I also worked as a student researcher at the Laboratory of Software Engineering on making Aspect Oriented programs under the supervision of Prof. Strohmeier.

Thanks to my excellent grades during my first 3 years at EPFL, I was awarded a scholarship in 2004 to study abroad at Carnegie Mellon University (CMU) in the USA. At CMU I received the highest grades, owing me the honors of entering that year's Dean's list. During my stay at CMU, I discovered a passion for robotics through a course on making autonomous AIBO robots (small robot dogs made by Sony). I then joined the MultiRobot Lab under the supervision of Prof. Veloso to work on AIBO odometry towards the robocup soccer competitions. In 2005, our team won the first place at the Robocup US Open.

Upon my return to Switzerland in 2005 I immediately started an Internship, then Master project, on the "Simulation of Swarming MAVs for Communication Relay" at the Laboratory of Intelligent Systems at the EPFL under the supervision of Prof. Floreano and Dr. Zufferey. This work initiated the PhD thesis presented here. During my time at the Laboratory of Intelligent Systems, I published 2 journal papers, 2 book chapters, 3 conference papers, and 4 peer-reviewed abstracts and posters. Two additional papers are in preparation as listed in Chapter B.2.3. These publications led me to give a presentation at several high-profile international venues including two invited full talks.

In parallel to my research, I was on the organization committee of the International Flying Insects and Robots Symposium, which attracted 150 participants. I also co-organized the workshop on Evolutionary Robotics at the IEEE International Conference on Evolutionary Computation. Finally, I was also heavily involved in writing a research proposal which was accepted and will allow two PhD students to continue work done during my thesis.

Finally, I gave great importance to disseminating knowledge about robotics to students and the general public. In particular, I have supervised 14 student projects and presented the LIS to innumerable prospecting students. More visibly, I have been active with the two best known podcasts on robotics, Talking Robots* and ROBOTS † which I preside. Over the years, these podcast have accounted for nearly 750'000 downloads. Through these podcasts, I have interviewed nearly 100 high-profile researchers in robotics. This work has given me huge insight into today's and tomorrow's research in robotics. I am now also Media Editor for one of the best known journals in robotics named "Autonomous Robots" and published by Springer US. As such, I am responsible for presenting the latest research in robotics in a fresh and interactive way on the Autonomous Robots Blog‡. Finally, I am also interested in producing quality video support to explain science. Along this line I have produced a video on "Bio-inspired Flying Robots" that won the best video award at the AI Video competition in 2008. Our latest video on the flight of 10 autonomous robots has also received wide media attention.

In the future, I hope to apply my knowledge on making simple robots work together towards the development of cooperative nanoparticles for medical applications.

*http://lis.epfl.ch/podcast

†http://www.robotspodcast.com/

‡http://www.autonomousrobotsblog.com/

Publications

Journal Papers

- Hauert, S., Leven, S. Zufferey, J.-C. and Floreano, D. (2011) Evolutionary Synthesis of Communication-based Aerial Swarms. in preparation.
- Leven, S., Hauert, S., Zufferey, J.-C. and Floreano, D. (2011) Bringing Swarms of Flying Robots into Reality. in preparation.
- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Evolved Swarming without Positioning Information: an Application in Aerial Communication Relay. Autonomous Robots, 26(1) pp. 21-32.
- Hauert, S., Winkler, L., Zufferey, J.-C. and Floreano, D. (2008) Ant-based Swarming with Positionless Micro Air Vehicles for Communication Relay. Swarm Intelligence, 2(2-4) pp. 167-188.

Conference Papers

- Hauert, S., Leven, S., Zufferey, J.-C. and Floreano, D. (2010) Beat-based Synchronization and Steering for Groups of Fixed-wing Flying Robots. Proceedings of International Symposium on Distributed Autonomous Robotics Systems, in press.
- Hauert, S., Leven, S., Zufferey, J.-C. and Floreano, D. (2010) Communication-based Leashing of Real Flying Robots. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 15-20.
- Hauert, S., Zufferey, J.-C. and Floreano, D. (2009) Reverse-engineering of Artificially Evolved Controllers for Swarms of Robots. Proceedings of the IEEE Congress on Evolutionary Computation, pp. 55-61.

Book Chapters

- Hauert, S., Mitri, S., Keller, L. and Floreano, D. (2010) Evolving Cooperation: From Biology to Engineering. in The Horizons of Evolutionary Robotics, MIT Press, in press.
- Zufferey, J.-C., Hauert, S., Stirling, T. Leven, S., Roberts, J. and Floreano, D. (2010) Aerial Collective Systems. in Handbook of Collective Robotic, Pan Stanford Publishing, in press.

Peer-reviewed Abstracts & Posters

- Hauert, S., Leven, S., Zufferey, J.-C. and Floreano, D. (2010) Communication-based Swarming for Flying Robots. Proceedings of the Workshop on Network Science and Systems Issues in Multi-Robot Autonomy, IEEE International Conference on Robotics and Automation.
- Hauert, S., Leven, S., Zufferey, J.-C. and Floreano, D. (2010) Communication-based Swarming for Flying Robots. Proceedings of the International Workshop on Self-Organized Systems.
- Hauert, S., Winkler, L., Zufferey, J.-C. and Floreano, D. (2007) Pheromone-based Swarming for Position-less MAVs. Proceedings of the International Symposium on Flying Insects and Robots.
- Floreano, D., Hauert, S., Leven, S. and Zufferey, J.-C. (2007) Evolutionary Swarms of Flying Robots. Proceedings of the International Symposium on Flying Insects and Robots.