An intelligent leader-follower neural controller in adverse observability scenarios

Eduardo Lourenço

Faculty of Aerospace Engineering

TU Delft Delft University of Technology

An intelligent leader-follower neural controller in adverse observability scenarios

Master of Science Thesis

to obtain the degree of Master of Science in Aerospace Engineering
at the Delft University of Technology

Eduardo Lourenço

December 16, 2020

Faculty of Aerospace Engineering - Delft University of Technology

TU Delft

Acknowledgements

The document I hereby present is the final product of the work I have been developing for the past two years. It symbolizes the official end of my life as a student. When I decided to join TU Delft in 2016, I knew I was sacrificing the comfort of being close to those dearest to me for a chance of a better future. A lot has happened in these past four years. I have experienced some of the best moments of my life, but I have also experienced the worst. All those experiences, all the people I got to know and learn from, were fundamental for my personal growth and helped make this journey an unforgettable one. In the end, I was able to fulfil many of my ambitions, and I look forward to the future and to achieve new ones.

There were many people I would like to thank for helping me through this journey and making this end result possible.

To Mario and Guido de Croon, I would like to thank all the support and understanding during this long process. You are some of the sharpest minds I had the pleasure to work with, but your kindness is equally impressive.

To my parents and grandmother, I would not be where I am today without your support and sacrifices. I am forever grateful for the opportunities you allowed me to have. To them and the rest of my family, I thank you for all the love, patience and encouragement.

To my friends in Portugal, who are like brothers to me, thank you for the visits, the love and for helping me live life with a smile in my face, even if we could not see each other in person.

To the friends I brought from Portugal and who shared the Dutch adventure with me, and to the new ones I made here, especially those who I shared a home with, thank you for being my Dutch family. That is a bond I hope we can forever preserve.

To the love of my life, Madalena, there are no words to express how much I appreciate everything you do for me. Your love, support and optimism guide me through my own darkness. There is no one I would rather have by my side to face whatever the future brings me.

List of Figures

List of Tables

List of Algorithms

Contents

I

Scientific Paper

An intelligent leader-follower neural controller in adverse observability scenarios

E. M. Lourenço[1], M. Coppola[1], and G. C. H. E. de Croon[1]

[1]Micro Air Vehicle Laboratory, Delft University of Technology

Abstract – **A high-level neural controller for leader-follower flight is presented. State of the art range-based relative localization schemes that rely exclusively on onboard sensors present an additional challenge to the leader-follower control problem since they restrict the flight conditions that guarantee observability. This novel controller was developed over an evolutionary process in which the simulation environment resembled the real-life constraints a group of MAVs would encounter. During the learning stage, a group of three agents is used, where one acts as a leader and flies a random trajectory, and the other two act as followers guided by a candidate controller that dictates the desired velocity commands. In the end, when equipped with the best-evolved controller, the follower agents are able to showcase a successful following behaviour that also enhances the observability of the system, although no observability metric was included in evolution.**

I Introduction

Micro Aerial Vehicles (MAV) have been one of the most prominent subjects in aerospace. Compared to larger Unmanned Aerial Vehicles (UAV), they are smaller and lighter, therefore, more portable, less intrusive and safer to operate around humans. These properties make them suitable for a wide set of applications, such as surveillance and mapping [8, 11, 10, 30], search and rescue missions in unknown environments [28, 7], or inspection and management of potentially hazardous areas [21, 19, 2, 33].

Within this domain, there is also great interest in the study and research of multi-agent systems and swarm robotics, as their application can augment the value and potential of such vehicles [29, 1, 30]. Indeed, running a swarm of MAVs, instead of one alone, may upgrade the performance and efficiency of the aforementioned operations. By cooperating with each other, agents can not only overcome their individual limitations, but also exploit their individual assets, while reducing the execution time of their mission.

Nonetheless, deploying an autonomous team of MAVs boosts additional challenges on top of the ones a single MAV already faces. First, and foremost, in order to have a minimum viable cooperation within the group, MAVs must have knowledge about the relative localization of, at least, some of its peers. The relative localization problem is one the scientific community has tackled for a long time, with different solutions being presented in the literature [6, 15, 5, 35, 16, 34]. However, in order to have the best generalization possible, MAVs must use exclusively onboard sensors to perform the relative localization estimates, therefore, excluding any method relying on external sources like a Global Navigation Satellite System (GNSS) or a Local Positioning System (LPS). While GNSS can be reliable outdoors, it is not an option indoors due to signal distortions. Moreover, although LPS can provide accurate measurements, it fails to meet the paramount assumption of most exploratory operations: no prior knowledge of the environment. Wireless range based methods have been proposed and applied with some success both in indoor and outdoor flight [15, 5, 16, 34]. Among their benefits is, obviously, the fact that they do not require any external hardware, and that they also allow for inter-agent communication, avoiding the need to use additional sensors for that purpose, which would increase the weight and cost of the overall system.

Guo et al. [16] have implemented an Ultra Wideband (UWB) and odometry-based relative localization system for a formation task within a group of MAVs. The range measurements provided by UWB are fused with displacement information from each MAV, in order to estimate the relative positions among the different agents. Nonetheless, the proposed method requires that each MAV measures its own displacement with respect to its initial position. If MAVs obtain these displacement measurements through velocity integration, then the error will build up over time, which may affect the accuracy of the filter.

In an alternative approach, Coppola et al. [5] proposed a range-based relative localization filter that combines range measurements, provided by Bluetooth technology, with heading, velocity and height measurements, also provided by onboard sensors. Following that work, van der Helm et al. [34] proposed a similar range-based solution, facilitated by Ultra Wideband technology, where the heading dependency is removed, as it is known to be a unreliable source in many indoor environments. Comparing both solutions, results showed that the heading independent one is more likely to provide better relative localization estimates. Nonetheless, that comes with the cost of additional flight constraints, of which the most notable is not allowing for two agents to fly any kind of parallel motion.

Taking into account these consequent flight constraints that such relative localization solution implies, there is a logical question on whether its use may inherently hinder cooperation between agents. Among the set of core tasks that groups of MAVS should be capable of performing before taking part

in more complex applications is leader-follower flight.

Leader-follower flight is usually seen as a type of formation flight where, essentially, one MAV leads the way of one, or more, follower MAV(s) [4]. One way to accomplish such behaviour, is to define, beforehand, a fixed formation geometry, so that each follower MAV knows where it should be located relative to the leader [38, 3, 23]. However, this results in parallel velocity vectors, which will drive the relative localization filter to unobservable states, compromising the relative localization estimates and, therefore, sentencing the formation task to failure.

An alternative approach, which pushes leader-follower flight outside of the traditional formation flight scope and its inherent observability issues, is to have each follower MAV to follow the leader's trajectory with a certain time delay, producing a collective movement that resembles that of a snake. van der Helm et al. [34] implemented such strategy with groups of 2 and 3 MAVs, equipped with the heading independent relative localization filter, and demonstrated through flight experiments that leader-follower flight could be possible. Most of the time, MAVs were capable of avoiding unobservable conditions and, even when they could not, they were able to recover. However, this strategy has a couple downsides. First, it is difficult to scale up, since it requires a lot of past data values to be stored. Secondly, and most importantly for the scope of this research, the trajectory of the leader needs to be properly designed as to prevent possible unobservable scenarios. For example, the leader's trajectory should not have long straight line segments, otherwise, the velocity vectors of different MAVs would, again, be parallel.

In this work we study whether the use of learning methods can provide an alternative solution for the leader-follower flight control problem when observability is not guaranteed. Opposed to traditional behaviour-based control design, evolutionary robotics and reinforcement learning allow agents to develop their own control system in close interaction with the environment. By doing so, the developer's bias are taken out of the loop, while the optimal solution is built. In the end, leader-follower flight could be accomplished with a solution that exhibits micro-behaviours the developer would not be able to implement with traditional control laws, or that were never considered in the first place.

Thus, the main contribution of this paper is a novel leader-follower control architecture that is able to produce a successful following behaviour on different scenarios, even when the observability of the relative localization estimates is not guaranteed. The solution was evolved through an evolutionary process, in which the leader-following task was simulated. As previous work has proved [31], this architecture is suitable to be implemented onboard MAVs with limited resources and without relying on any external source.

The structure of the paper is organized as follows. First, section II provides a brief overview of the relevant literature on relative localization (A), leader-follower flight (B) and evolutionary robotics (C). Then, section III introduces the leader-follower task the proposed solution aims to tackle. After that, the evolutionary algorithm that was used to evolve the controller is explained in section IV. Sections V and VI present and discuss the results of the evolutionary process and the ability of the evolved controller to succeed in different following scenarios. Finally, section VII summarizes the conclusions of this work and offers remarks for future improvements.

II Related Work

A *Relative Localization*

In the case where access to positional systems - such as GNSS or LPS - is possible, it is straightforward for an agent to find its position within the reference frame of the flying environment. Assuming the different agents can communicate among themselves, or that a central computer is instead handling the data, the relative localization problem is directly solvable because global positioning information is known [24, 22]. However, in unknown indoor scenarios, these positional systems are typically not available. Therefore, in order to have versatility and flexibility to different environments, it is necessary that MAVs use relative localization methods that rely exclusively on onboard sensors. Among the solutions available in the literature, it is possible to make a distinction between vision-based and signal-based methods.

The first class of methods, makes use of the robot's camera to solve this problem [12, 36, 35]. One of its main drawbacks is the limited field of view of the cameras, which affects the availability of the relative localization estimates. Although solutions have been proposed to overcome this problem, they mostly come with the cost of increased weight or motion constraints [36, 35]. Another issue is that lighting of the environment influences the correct recognition of other agents, and despite efforts to minimize that through the use of LED markers [36, 35], it is still not sensible to ignore that effect.

Opposed to vision-based methods, signal-based ones offer an omni-directional solution, as they rely on the transmission of a wireless signal between two agents in order to obtain range measurements. Additionally, as communication is also needed, it is advantageous to build the relative localization solution on top of existent communication hardware, without any additional hardware costs. In most works, the relative localization estimates are then the result of fusing the range data with onboard ego-motion measurements [15, 5, 20, 16, 34, 25]. Among the measurements used is the orientation with respect to North, however, this is usually measured with a magnetometer, which is subject to magnetic disturbances. In order to better understand the impact of removing a common heading reference, van der Helm et al. [34] performed an observability analysis for a relative localization system with and without the heading measurements. Results

showed that removing the heading measurement comes with the cost of extended flight constraints. Especially relevant is the fact that it seems to preclude formation flight, namely traditional leader-follower flight, due to the fact that MAVs are not allowed to fly parallel motions. To solve this, the authors proposed that the leader would fly along curved paths and that the followers would follow this with a specific delay. Therefore, the observability of the relative localization filter is dependent on the leader's trajectory. For instance, if the leader flies a straight trajectory for long periods of time, the relative localization filters will diverge and the leader-follower behaviour will not be possible to reproduce.

Nguyen et al. [25] also proposed a solution to overcome the limitations of leader-follower flight with range-based relative localization methods. Typically, leader-follower flight is composed of two types of agents: the leader and the followers. They propose the addition of a third type of agent, the orbiter, which tracks the leader with a predefined periodic trajectory that aims to excite the observability of the system. In this way, the regular followers can use orbiters as a reference and obtain more accurate relative localization estimates. The downside of this approach is that one third of the group needs to facilitate the others, by flying the predefined trajectory. Hence, the feasibility of this method depends on the application and resources available.

B Leader-Follower Flight

Leader-follower flight is most commonly seen as a subclass of formation flight where one agent leads the way of one, or more, follower agents [4]. Usually, this is accomplished by establishing a fixed group formation geometry, and then having each follower maintain the same relative position to the leader over the flight [38, 3, 23]. A different approach, that overcomes the parallel velocity issues of fixed formation flight, is to have the followers fly the trajectory of the leader with a certain time delay [34].

While these strategies define following in a different way, the rationale behind the control laws that are employed to actually materialize the defined following behaviour are pretty similar. Basically, a reference trajectory is defined for the followers, and then the controller tries to minimize the positional error, so that the desired following behaviour is produced. For the first approach, the reference trajectory is simply a translation of the leader's trajectory, which only depends on the predefined relative position offsets. For the second, the reference trajectory is the same as the leader's with a certain time delayed applied. Different control methods have been proposed to accomplish such laws.

Another important aspect of leader-follower flight implementation is whether external resources are used, i.e. if the agents have any interaction with other entities out of the swarm. In decentralized methods, all the processing and decision making is done individually by the agents or through interaction between them, without communication with any external entity. It is easy to realize that a fully decentralized control strategy is more versatile and flexible than a centralized one. Nonetheless, it is also harder to implement, as the computational resources are more scarce.

C Evolutionary Robotics

Traditional behaviour-based control design methods rely on the designer to define the applicable control laws that govern the system to be controlled. This works extremely well for behaviours that can be mathematically described. However, this is not always the case, and designers end up implementing control laws based on their own expectations and beliefs, which may hinder the discovery of better fitting solutions.

An alternative approach is to use learning methods - such as evolutionary robotics (ER) and reinforcement learning (RL) - which enable agents to develop their own control system by interacting with the task environment. While ER and RL are built on different principles, they still share a similar approach: "agents obtain rewards/fitness values while behaving in their environment, and we want to find the policy/behaviour that corresponds to the maximum reward/fitness" [9]. For the purpose of this work, it was decided to employ an ER framework. The main reasons behind the decision is that ER copes better with partial observability and large continuous state-action spaces [9] when compared with RL methods that rely on the Markov property.

Evolutionary robotics is a method that employs evolutionary computation techniques to the design of autonomous multi robots' systems. In ER, robots are considered artificial organisms that evolve their own control system (and possibly body configuration) in close interaction with the environment, without direct human interference, in order to produce functional behaviours [26].

Evolutionary algorithms (EA) are the basis of any ER approach. Despite the wide variety of EAs available, they all share a common framework. Applying this framework to ER, it all starts with the initialization of a random initial population of different artificial genotypes, each encoding the robot's control system (and morphology if desired). Each of these genotypes is then decoded into a corresponding robot that acts on the environment (through simulation or real experiments) and the collective behaviour of the resulting swarm is evaluated based on one, or multiple, predefined fitness function(s). All population of artificial genotypes follows this procedure and, based on performance, a group of them will be selected to have the chance of reproducing copies of their genotype through genetic variation operations (mutation and/or recombination), generating the offspring. Next, this resulting offspring is subjected to the same evaluation process that the parents' population experienced, and a new generation will be created by selecting individuals from the parents' and offspring's population. The selection of the new generation is usually done by giving a better chance to fitter individuals to propagate their genotype. The new

generation follows this cycle, creating one generation after another, until a stopping condition is met. In the end, "good" behaviours (as judged by the fitness function) survive over different generations and "bad" ones are discarded, through an evolutionary cycle similar to the one driving natural evolution.

A significant number of research work has focused on the evolution of robot's controllers through evolutionary robotics. Frequently, artificial neural networks (ANN) are the state of the art control architecture represented by artificial genotypes [14, 32, 17, 27, 18, 31, 13]. Although the large increase of onboard computational power available to robots only occurred after the new millennium, the combination of ER and neural controllers to teach a robot to accomplish specific tasks goes far back. Floreano and Mondada [14] showed that it was possible to evolve a homing behaviour for a ground robot using a recurrent neural network to control its trajectory. The recurrent connections of the neural network were vital for the success of the solution, especially in situations where the robot could become trapped. By providing information regarding the states' history, the controller was able to yield different actions for the same sensory inputs. More recently, Scheper and de Croon [31] successfully evolved a high-level neural network controller that allowed a swarm of three MAVs to rearrange themselves in an asymmetric triangular shape. The controller focus only on the formation task, therefore, it is only responsible to output the velocity commands, while a pre-existent inner loop controller guarantees stability and translates those velocity values into angular velocity values of the rotors. Despite evolving the controller in simulation, the final solution was tested in real flight experiments in order to validate the findings. The results were similar to what simulation suggested, and it was possible to conclude that there was a reduced reality gap when passing from simulation to real flight.

III Leader-Follower Task

This paper employs an evolutionary optimization process aiming to develop a leader-follower behaviour of a homogeneous swarm of quadrotor MAVs, when the observability of the relative localization estimates is not guaranteed. It is assumed that the simulated MAVs are capable of measuring their own height, therefore, the impact of the height measurement on the filter observability is negligible [34].

The chosen leader-follower task is slightly different than the ones already referred in B. Having a team of 3 MAVs, one will act as the global leader (GL), therefore, deciding which trajectory is the group flying. Then, one MAV will follow the GL, while being followed by the third one. Thus, the leader-follower controller is only implemented in two followers MAVs, while the GL flies a pre-defined trajectory. A visual representation of the task is provided in figure 1.

Ideally, the results could be scaled up for a swarm of N MAVs, where one acts as the GL, $N-2$ act as both leader and follower, and the last one of the group only acts as a follower. In this way, there is no need to store the reference trajectory, since it is propagated through the movement of the swarm. Moreover, with this approach, one MAV only has to run one relative localization filter (for the local leader), avoiding excessive computations and processing time.

Figure 1: Visual representation of the leader-follower task with 3 MAVs. The Global Leader flies a random trajectory. The 1st Follower follows the Global Leader. The 2nd Follower follows the 1st Follower.

A *Control Scheme*

There are different levels of control involved in autonomous flight. On the one hand, there is lower level control, where the power of each propeller of the MAV is adjusted using a control law that, for instance, takes into account a desired acceleration or velocity of the vehicle and translates it into the appropriate propellers' power. On the other hand, there is higher level control, wherein a wider view over the goals of the system is assumed, in this case the leader-follower flight, in order to determine the acceleration or velocity values that promote that behaviour.

It has been showed that it can be beneficial to evolve behaviours for complex tasks using a higher-level control law with an underlying inner loop controller [31, 19]. Therefore, it was decided to follow those findings and evolve a controller that is only responsible for commanding the desired MAV's velocity setpoints, which are then achieved by the inner loop controller. For this, it was considered that each MAV has an inner loop controller that ensures stable performance of the vehicle dynamics.

In this way, the evolved controller focuses exclusively on optimizing the high level behaviour of following the virtual leader, without worrying about all the dynamics of the

MAV. Moreover, as this stable inner loop control schemes are widely available, it makes the system more robust to the reality gap between the dynamics of the simulated system and those of the real one [31]. It is worth mentioning that the evolved controller is to be used by all swarm agents except the GL, as this last one will fly a desired reference trajectory, without any regard for a leader MAV. During the simulations, the different GLs fly random trajectories.

The control architecture choice was an artificial neural network (ANN) with one recurrent hidden layer. At a given time step t, the output of the hidden layer neurons is a function of the weighted sum of the inputs at that time t, the bias node, and the output of the hidden layer at time $t-1$. The recurrence in the hidden layer was employed so that the controller can decide the velocity setpoints based not only on the current situation, but also taking into account the recent past values that led to it.

The number of input neurons of the network was varied during the experiments in order to understand which combination yielded the best results. In the end, 6 input neurons were used. Regarding the hidden layer, it was decided to compose it with 15 neurons, so that there was enough "space" for evolution to try and solve the complexities of the problem. Moreover, bias nodes were added to both the input and hidden layers. In the last layer, there are 2 output neurons, that will correspond to the desired velocity commands in the x and y body frame. Because there is a maximum velocity allowed (ν_{max}), the two outputs of the neural network controller are scaled before being fed to the inner loop controller. All neurons of the network use a $tanh$ activation function, because we want the output to be bounded between [-1,1]. Also, the network weights values were limited to the interval $[-2, 2]$, except the bias' weights which were constrained to a larger interval, $[-5, 5]$. Figure 2 illustrates this neural network architecture.

The six inputs of the neural network were chosen in order to provide valuable information to the controller and help it solve the leader-following problem. The first two inputs of the network were the desired relative position of the MAV. Ideally, the MAV should follow its leader "from behind", so that the reference trajectory can be propagated through the swarm. Therefore, $r = [r_x, r_y]^T$ was defined as:

$$r = p - v_2 \tag{1}$$

with p being the relative position between the following MAV and its local leader, and v_2 the velocity of the local leader. The latter velocity, v_2, is also an input to the network, as well as the own velocity of the following MAV, v_1. The velocities were added with the reasoning that the controller might be able to anticipate changes in the desired relative position, hence, acting more proactively rather than reactively. In this way, the error propagation of the reference trajectory through the swarm could possibly be minimized.

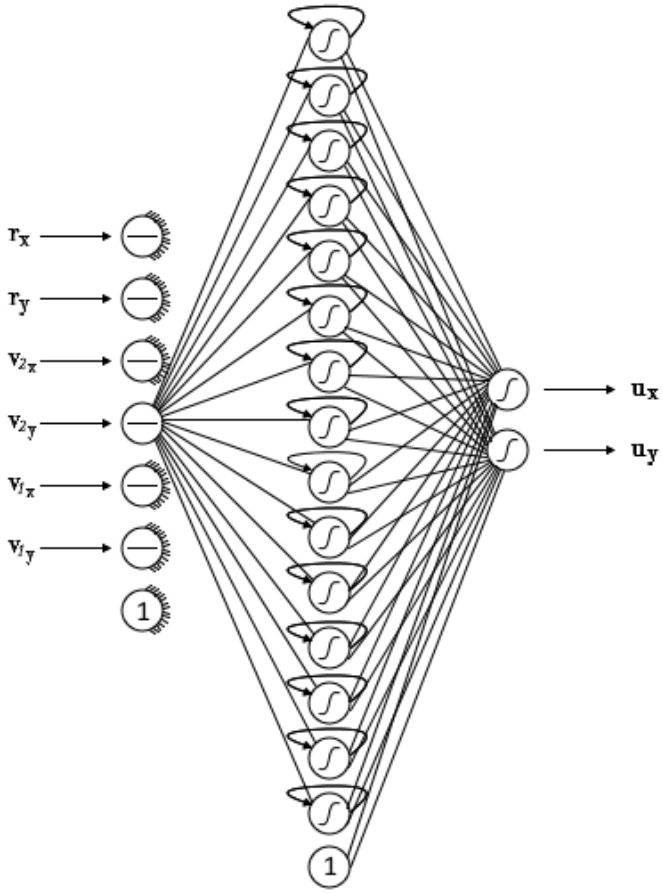


Figure 2: High-Level Velocity Controller

B *High-Level Quadrotor Kinematic Model*

Although this work only presents simulation results, it was performed with the aim of later being applied in real-world experiments. With this in mind, the Parrot ARDrone 2 quadrotor MAV kinematic model was used to simulate the MAVs' dynamics.

As previously referred, an inner loop controller is already implemented in all vehicles, and ensures a stable performance, therefore, the high-level dynamics can be modelled by a first-order system with time constant $\tau_\nu = 0.3636$ [31]:

$$\dot{v} = \begin{bmatrix} -\frac{1}{\tau_\nu + \delta_t} & 0 \\ 0 & -\frac{1}{\tau_\nu + \delta_t} \end{bmatrix} v + \begin{bmatrix} \frac{u_x}{\tau_\nu + \delta_t} \\ \frac{u_y}{\tau_\nu + \delta_t} \end{bmatrix} \nu_{max} \tag{2}$$

where $v = [v_x, v_y]^T$ represents the linear velocity vector, $u = [u_x, u_y]^T$ is the output of the neural network controller, δ_t is the discrete time step of the simulation, and $\nu_{max} = 0.6$m/s is the maximum velocity allowed for the purpose of this work. The value of ν_{max} was set by taking into

account the safety of future flight experiments. The value of τ_ν is perturbed by $\pm 5\%$ for each vehicle, at the start of each simulation, as recommended in Scheper and de Croon [31].

An additional advantage of using the high-level first-order system to simulate the vehicles' dynamics, is that it allows for a larger simulation time step, δ_t (at least 10 times larger). In this way, the computational effort of the simulator is significantly reduced, making it more feasible to test different alternatives and solutions.

C *Relative Localization Filter*

Like in van der Helm et al. [34], the Extended Kalman Filter (EKF) provides the necessary heading independent relative localization estimates to each simulated MAV. The system and measurement noise matrices are initialized based on the correspondent expected noise values, but with a small random perturbation of $\pm 5\%$ at the beginning of each simulation. This is done in order to reduce the reality gap between simulation and real flight experiments [31]. As for the initial state, it is initialized with the real one, as to maximize the chances of the filter converging at the start of the simulation, putting the focus on the filter performance during the leader-follower task. Consequently, the initial state covariance matrix is initialized as an identity matrix.

IV Evolutionary Algorithm

The evolutionary algorithm employed to evolve the aforementioned controller follows the framework introduced in section C. Given the fact that the controller has a fixed structure, i.e. the number of layers and respective neurons is defined beforehand, the EA is responsible for optimizing the value of the weights of the network (within the applicable range). Moreover, an additional parameter, which represents the refresh rate of the controller, is also optimized. The value of this parameter can vary between 20Hz, 10Hz, 5Hz or 1Hz, which are all multiples of the simulation time step, δ_t, equal to 0.05 seconds.

With this being said, an initial population of 40 individuals is generated. Each individual is represented by a vector of 363 elements (7×15 (connections between input plus bias and hidden neurons) $+15 \times 15$ (recurrent connections) $+16 \times 2$ (connections between hidden plus bias and output neurons) $+1$ (controller rate)), randomly initialized within the applicable ranges. Once initialized, each individual of the primal population was implemented on the two followers, which were then simulated, together with the global leader agent, using the high-level dynamics introduced in B. In order to avoid creating bias in the following behaviour, the three MAVs are always initialized in a random position, at hover, with a inter-distance between 1 and 1.5 meters. During initialization, a follower can be in front of its local leader. Additionally, all MAVs are initialized with a zero heading.

One of the main difficulties of this work was to define a proper way to evaluate good following behaviour. At the beginning, we thought that it made sense to maximize both the following performance and the observability of the relative localization states. The first one, because the task is leader-follower flight. The second, because the observability of the relative localization filter also affects the following performance. If the filter estimates' errors are too large, then it is impossible to be a good follower. However, the observability metric used (the estimation condition number) was unable to add additional value to the optimization because it provided ambiguous information. Hence, we ended up maximizing only the following performance, which was not a problem because it is expected that a good following behavior is correlated with good observability of the system.

During a trial, each "following" agent is first evaluated individually using the coming set of equations:

$$f_{\text{individual}} = f_{\text{relative position (rp)}} \times f_{\text{relative angle (ra)}} \tag{3}$$

$$f_{\text{rp}} = \frac{1}{t_{\max}/\delta_t} \sum_{t}^{t_{\max}} f_{\text{rp}_t} \tag{4}$$

$$f_{\text{rp}_t} = \begin{cases} 1 - \frac{1}{1+e^{50(p_t-0.7)}} & \text{if } p_t < 1 \\ 1 - \frac{1}{1+e^{-5(p_t-2.3)}} & \text{else} \end{cases} \tag{5}$$

$$f_{\text{ra}} = \frac{1}{t_{\max}/\delta_t} \sum_{t}^{t_{\max}} \left(1 - \frac{1}{1+e^{-10(\alpha_t - \frac{\pi}{4})}}\right) \tag{6}$$

With p_t being the relative position between a follower MAV and its local leader, and α_t being the smallest angle between the relative position and local leader's velocity vectors, at time t. Figure 3 illustrates p_t and α_t for a random motion of a leader/follower pair, and figure 4 provides an overview of the fitness function distribution for a specific leader motion.

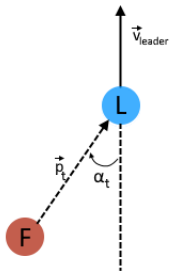


Figure 3: p and α at any given time t

Then, based on the individual performance of each follower, a final fitness function was defined:

$$f_{\text{final}} = f_{\text{follower 1}} \times f_{\text{follower 2}} \tag{7}$$

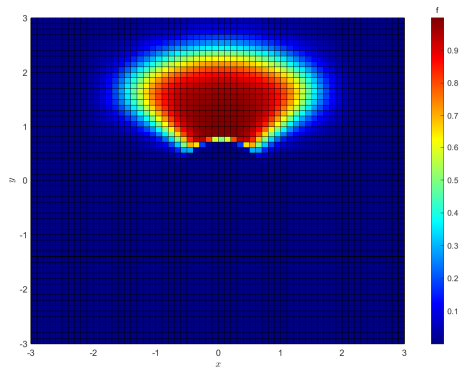


Figure 4: Individual Fitness Function Values. The leader is assumed to be at the origin ($x = 0, y = 0$), with velocity $v = (0, -1)$

As the individual fitness values are always in the interval $[0, 1]$, the product was used in order to penalize situations where followers had highly contrasting performances.

Looking at figure 4, stands out that evolution is optimizing controllers that enable a follower to keep a close distance to the local leader, but not so much that a collision occurs (equations 4 and 5); while doing it as aligned as possible with the motion of the leader (equation 6). The last component was introduced because preliminary work showed that the controller tended to lead the follower to the side of its local leader. When that happens, the reference trajectory of the global leader is no longer stored in the swarm through the motion of the followers, which can be fatal in scenarios where the global leader is the only agent capable of autonomously navigating through a cluttered room.

As for the variability component of the algorithm, following suggestions in the literature, mutation was the only variation operator used, as it seems to be more effective when dealing with real value genotypes [37]. Each parameter of the genotype had a mutation probability of 10%. For the 362 genotype parameters corresponding to weights of the neural network, mutation resulted in a perturbation of the previous value by a value drawn from a Gaussian distribution. For parameter 363, which encoded the controller's refresh rate, a mutation meant that a random rate was drawn from the available options. Every member of the initial population experiences mutation, therefore, the resulting offspring was also comprised of 40 individuals. The offspring experienced the exact same evaluation process described for the initial population.

Once the evaluation ends, the population and offspring were grouped together and ranked in descending order. Then, the next generation is chosen through a round-robin tournament, in which each individual was compared against 8 others, randomly chosen. For each time an individual had a higher fitness then its random opponent, it would get a win. In the end, the 40 individuals with more wins were selected for the next generation, and the process would reiterate.

Over the evolutionary process, although the basic setup of the evaluation remained the same, there were certain conditions that changed. Essentially, the complexity of the task gradually increased and, based on those increments, it is possible to identify three different evaluation stages.

1. In the first one, the simulation takes 90 seconds, each genotype is evaluated through 3 different trials, and only the performance of the agent that follows the GL is taken into account through equation 3.

2. After the first follower agent starts showing some ability, there is a change in the fitness function and equation 7 is used, in order to evaluate the overall performance of the group.

3. In the last stage, there are no changes to the fitness function, however, the time of the simulation is increased to 150 seconds, and the number of trials increased to 5.

This staged evolution was implemented because it was observed, in previous experiments, that focusing exclusively on the first follower at early stages would speed up the optimization process. An explanation can be offered by the fact that in early generations the controller easily saturates, which results in both followers moving in a straight line trajectory. As a consequence, the straight line motion of the second follower can erroneously be interpreted as a perfect following behaviour of the first follower's straight line trajectory, which is clearly not the case. Moreover, both the time of simulation and the number of trials contributing to the final fitness value were increased in order to get the best generalising solution possible. Finally, it should be noted that in the different trials, the GL always flies a random trajectory and, at least, one of them promotes an unobservable scenario (straight line trajectory). This random trajectory is achieved through velocity increments/decrements in both magnitude and direction at random time steps, with larger variations (which result in sharper turns) being less probable. The reasoning behind choosing random trajectories is that we wanted the followers to be able to be successful in any type of trajectory, which augments the potential applications of the solution.

V Results

Figure 5 shows the progress of the evolved solutions over different generations, namely the average fitness of the population and the best individual performance. The standard deviation of the population fitness is also included in order to make it easier to observe the variation in performance within the population. It can be seen that a following behaviour was quickly evolved during the first and beginning of the second stages. After that, a less prominent learning curve is visible,

in which the performance increases slowly but steadily. Finally, when the third stage begins, the performance stops improving and starts converging. This converging behaviour is a result of changes in the time of the simulations and the number of trials per evaluation, which reduced possible variability caused by favourable conditions (initialization or difficulty of the trajectory to follow). Ideally, there would not be the need to use different stages and the solutions would have more generations to evolve. However, because the simulation time per generation could be quite high (from, approximately, 140 to 220 seconds depending on the stage), there was the need to try to speed up the evolutionary process. The biggest reason for such high generational time is the fact that running the relative localization filters can be quite heavy (a simulated task with the relative localization filters takes an average of 3.2 seconds, while one without the filters takes an average of 0.28 seconds).

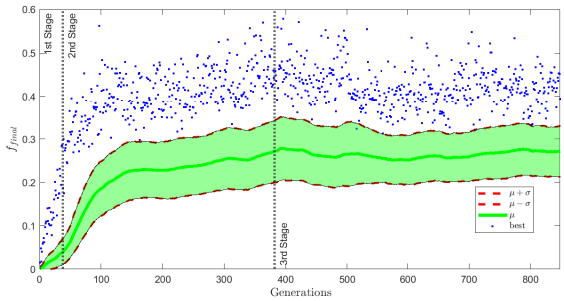
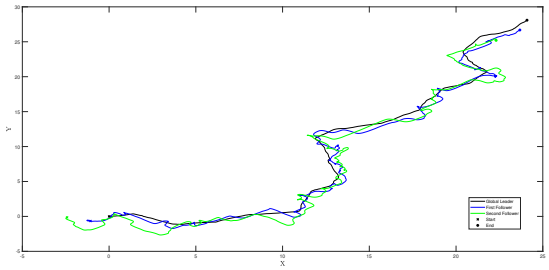


Figure 5: Fitness progress of the average performance of the population and best individual during evolution of the leader-following controller. The different stages are clearly identified.

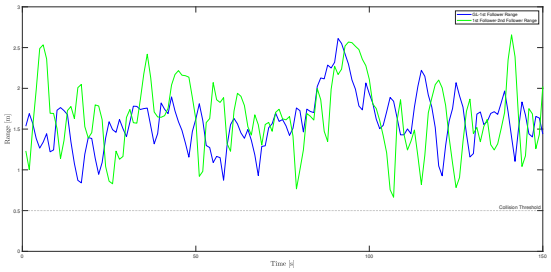
In order to validate the findings, the controller with the highest fitness was further evaluated through different tests. It is worth mentioning that the optimal controller rate of this controller, and the majority of the final population of controllers, is 10Hz. The first test tackles the task that the controller experienced during the evolutionary process. In the second test, the GL only flies straight line trajectories, so that we can try to find an answer to the hypothesis that maximizing only the following performance also has the effect of maximizing observability. Finally, the third test looks into the scalability of the controller, by showing results for a larger group of agents than the one used during evolution. Because we are primarily interested in the following behaviour of the controller, the results will not take into account collision rates. A successful trial is one in which the individual fitness value of both followers is at least higher than 0.5.

A *General Test: GL flying random trajectories*

This test consisted of 250 trials of 150 seconds, wherein the initial conditions were varied and the global leader flew different random trajectories. Figure 6 illustrates one of the successful runs. Results show that 53% of the trials were successful. When looking separately to the different follower agents, it was observed that the first follower was able to be successful in 95% of the runs, which highly contrasts with the 53% success rate of the second follower.



(a) Trajectories



(b) Range between the leader-follower pairs

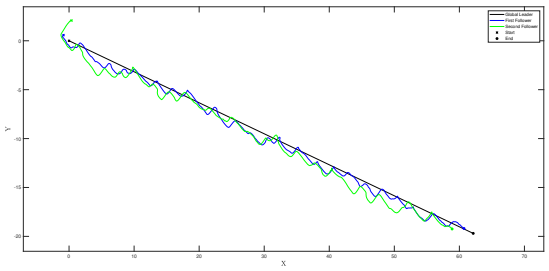
Figure 6: Ground track of one successful run of leader-follower flight when the Global Leader agent flies a random trajectory.

B *Observability Test: GL flying straight trajectories*

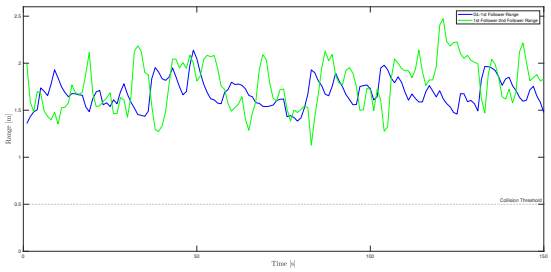
In the second test, the three agents were again simulated over 250 trials of 150 seconds, with different initial conditions. This time, the GL flies different straight line trajectories, which are known to promote unobservability. Using the same measure of success, results show that the task was completely successful in 38% of the trials. Figure 7 illustrates one of these. Looking separately, the first follower was successful in 84% of the trials, while the second follower was only successful in 39%. Again, a significant difference can be observed between both followers.

C *Scalability Test: GL flying straight trajectories and one extra follower*

To assess how the evolved solution scales to a larger group than the one used in the evolutionary environment, 250 additional simulations were run for a group of four agents, wherein the initial conditions were varied and the global leader flew different random trajectories. As expected, the two first followers show a success rate similar to the one presented in V. As for the third follower, it shows a 28% success rate, which is an additional decrease when compared with that

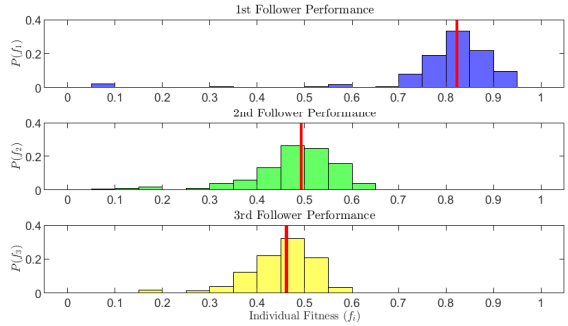


(a) Trajectories

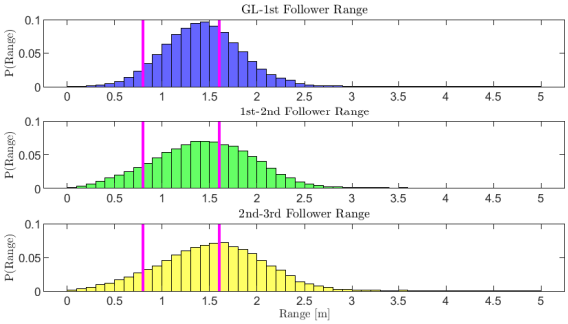


(b) Range between the leader-follower pairs

Figure 7: Ground track of one successful run of leader-follower flight when the Global Leader agent flies a pure straight line trajectory.



(a) Individual Fitness of the different follower agents



(b) Range distribution between the leader-follower pairs

Figure 8: Summary of results for 250 trials of leader-follower flight with three followers.

of the second follower. Nonetheless, looking at figure 8a, in which the distribution of the individual performances of each follower is shown, it is possible to see that its performance is actually closer to that of the second follower than one might expect if they only looked at the aforementioned success rate. Moreover, figure 8b shows the distribution of all the range values when combining all 250 trials. There, the similarity between the performance of both agents stands out even further.

Table 1 provides an overview of the success rates of the different tests.

VI Discussion

After presenting the different test results of the best evolved controller, it is time to go further in the analysis. This section starts by commenting the suggesting evidence that the proposed solution is able to evolve a controller that learns how to follow while steering itself from unobservable states. Moreover, the scalability of the controller is tackled, by showing some results for a larger group of agents than the one used during evolution. And finally, the neural controller's preferential behaviour, as well as some failure cases, will be discussed.

A *General Test: GL flying random trajectories*

The first test was done to validate the results of the evolutionary process. Moreover, it provides insights into the main limitations of the evolved solution.

While the overall performance is in line with what was expected (see final generations in figure 5), the significant difference between the performance of both followers suggests that the great success of the first follower may compromise the success of the second one. Looking into figure 6b, and taking into account that most successful trials also exhibit a similar behaviour, it seems to be that the optimal way to follow a reference leader induces such a wobbly motion that it consequently hinders others that want to follow it.

B *Observability Test: GL flying straight trajectories*

To understand whether there is an emergent behaviour that helps the evolved controller steer away from unobservable scenarios, it was decided to test it in a setup where the global leader flies, with constant speed, a pure straight line trajectory. Revisiting what has been said before, this type of motion by a leader is prohibitive for any of the previously proposed leader-follower solutions, as the observability of the relative localization states is not guaranteed for a long period.

Although the results for this test are worse than those yielded in the first one, they definitely attest the ability of the controller to steer away from unobservable states in highly challenging scenarios. In fact, only the first follower has a leader flying a pure straight line trajectory, and its results

Test/Success Rate	Overall	First Follower	Second Follower	Third Follower
A	53%	95%	53%	-
B	38%	84%	39%	-
C	20%	95%	53%	28%

Table 1: Overview of the results

are remarkable. The inferior results of the second follower seem to strengthen the hypothesis that the controller's induced wobbly motion may be detrimental for the overall group performance. This hypothesis can be further tested looking into the third test.

C *Scalability Test: GL flying straight trajectories and one-extra follower*

Test number 3 provides interesting results that help us draw several conclusions. First, the controller can be scaled up to a larger group. However, it is recommended to only scale up if the following performance expectations of the task are lower enough to allow it. For instance, if the environment does not have a narrow flying area where constant obstacle/wall avoidance is required. As seen before, the wobbling behaviour induced by the controller is most likely the cause for the drop in performance, since its effect is multiplied for the third follower.

Moreover, the difference between the performance of the first follower, and that of the other followers is more significant than the difference between the second follower and the third one. One of the main reasons for this is that the first follower is following a smooth trajectory (global leader), while the others have a slightly more complicated task. For this reason, it is more difficult to score in the angle component of the fitness function (equations 3 and 6) for the other followers than for the first one.

D *Emergent Behaviour and Failure Pattern*

It has already been established that the evolved controller shows the ability to follow its leader, as well as to steer away from unobservable scenarios. Now, it is time to go further and try to analyze the optimized behaviour.

Looking closely into the output of the neural network controller during the successful flights provided in figure 9a, it is possible to see that, most of the time, it chooses a combination of the maximum allowed (positive or negative) velocity commands. Additionally, it can also be seen that there is a continuous swing between positive and negative values, which is the main cause of the wobbling following behaviour that was earlier mentioned. Figure 9b shows that phenomenon.

Equally interesting is the fact that this swing behaviour of the velocity commands stops happening when the controller is not able to follow. In fact, what happens instead, is that the controller is so eager to correct its position, that it ends up saturating its output to the same values - see figure 10b. By doing so, it puts itself into a snowball effect, because this saturation stimulates unobservability, which in turn degrades the relative localization estimates, therefore, the ability to follow the leader. Figure 10a reinforces this idea, by showing a 33% increase in the occurrence of larger velocity command pairs. The idea of adding the follower's own velocity as an input to the controller tried to offer an exit to this loop, however, there is still room for improvement. What exactly triggers the failure situation is, unfortunately, still not clear, hence, further research should be carried, so that the solution can be augmented to tackle it.

Finally, it should be noted that the controller also shows an ability to recover from these failure situations. Figure 11 shows an example in which the first follower looses track of the global leader and, later, is able to recover.

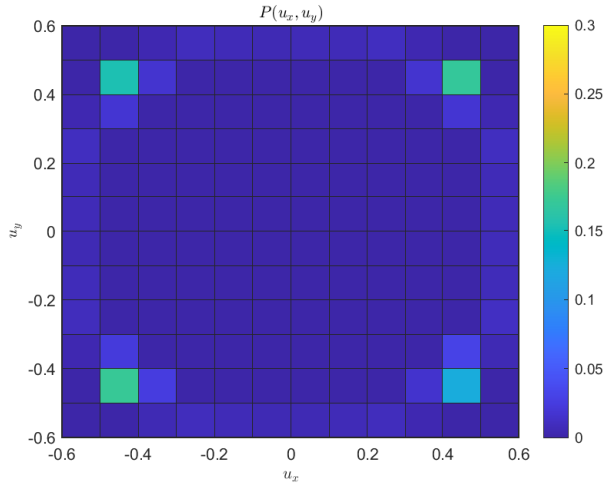
E *Collisions*

In Section V, results do not include the impact of collisions because the major aim of this work is to evaluate the following behaviour that was evolved when dealing with the observability limitations already referred. Nonetheless, collisions obviously have an impact in the real life success of a leader-follower controller solution. If we were to include those results, they would show that the controller tends to over-approximate the follower agent towards its local leader, which leads to too many potentially hazardous situations. Moreover, it seems to be a bigger problem when the GL flies random trajectories, compared with straight line ones. This may be explained by the fact that, for the latter, the direction of the trajectory is constant over time, therefore, there are less unexpected turns (in successful following flights) that may raise conflicts.

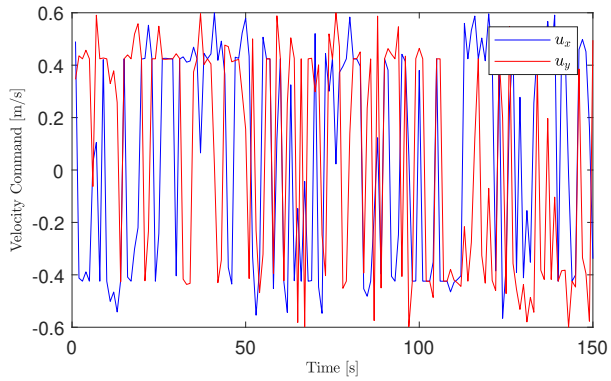
Figure 8b actually shows the tendency of the controller to over-approximate to its local leader, which in a real world scenario could lead to collisions. That is something that in the future should be tackled by, for instance, moving the fitness function a bit further from the leader (see figure 4), and/or introducing a stopping condition in evolution when a collision occurs and penalise it during the evaluation phase. For the scope of this work, where the intent was to study optimal follower behaviours, the latter was not included.

VII Conclusion

In this paper a novel control architecture for leader-follower flight is proposed. The evolved controller is able to produce a following behaviour, while showcasing the ability to enhance the observability of the system. It is shown

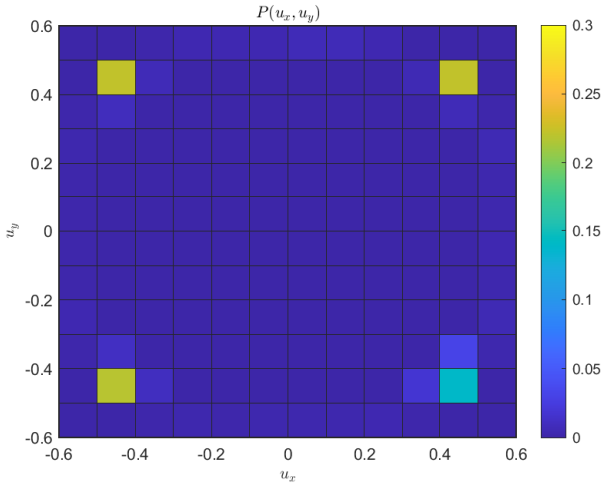


(a) Heatmap of velocity commands for 150 different successful trials

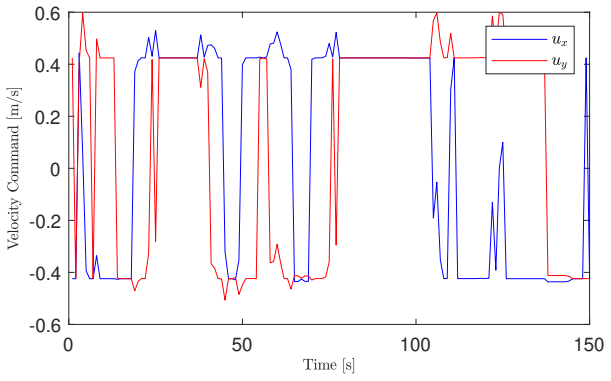


(b) Velocity Commands of one specific successful trial

Figure 9: Evolved neural network controller behaviour.



(a) Heatmap of velocity commands for 50 different unsuccessful trials



(b) Velocity Commands of one specific unsuccessful trial

Figure 10: Controller behaviour for unsuccessful trials.

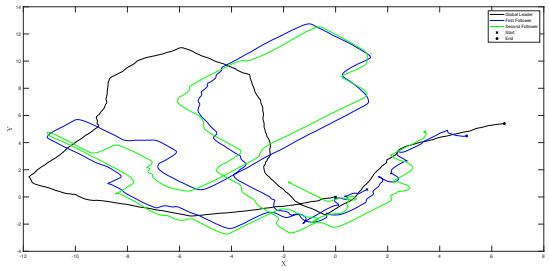
that despite the limitations of the heading independent filter, leader-follower flight is still possible, even when confronted with a leader trajectory that was previously thought to be prohibitive for such task.

Nonetheless, the wobbling behaviour that results from the controller's velocity commands is a double-edged sword. On the one hand, it seems to give the follower the ability to steer away from unobservable states, while on the other hand, it complicates the task of an additional follower. Despite this, the controller still shows a promising ability to be employed in larger groups than the one used during the learning process.

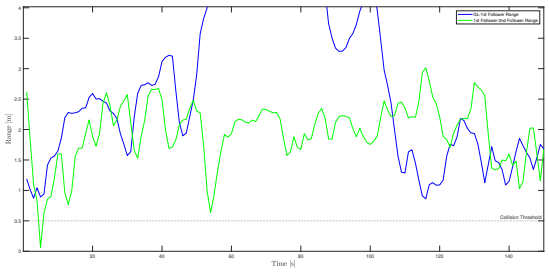
Future work should necessarily tackle some identified flaws. First, the tendency of the controller to overapproximate to its local leader needs to be fixed before experimenting with real flights. Moving the fitness range component further from the leader, i.e. penalize heavily any range smaller than, for instance, 1m instead of 0.7m, could be a solution. Otherwise, a collision stopping condition that penalizes the fitness function could be introduced in evolution. Another approach could be to implement a separate collision avoidance behaviour that overwrites the leader-follower controller when facing potentially dangerous situations. Secondly, further research should be done to try to identify the trigger of the failure situations, so that possible solutions that anticipate them can be incorporated. Until then, a simple solution could be to introduce an additional safety component to the controller that avoids the saturation of its commands by, for example, perturbing them by 20% whenever it finds itself outputting the same commands for too long.

References

[1] Markus Achtelik, Michael Achtelik, Yorick Brunet, Margarita Chli, Savvas Chatzichristofis, Jean-Dominique Decotignie, Klaus-Michael Doth, Friedrich Fraundorfer, Laurent Kneip, Daniel Gurdan, et al. Sfly: Swarm of micro flying robots. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2649–2650. IEEE, 2012.



(a) Trajectories



(b) Range between the leader-follower pairs

Figure 11: Ground track of one unsuccessful run of leader-follower flight when the Global Leader agent flies a random trajectory. Although the 1st Follower looses track of the Global Leader, it is able to recover later.

[2] Kostas Alexis, Christos Papachristos, Roland Siegwart, and Anthony Tzes. Uniform coverage structural inspection path–planning for micro aerial vehicles. In *2015 IEEE international symposium on intelligent control (ISIC)*, pages 59–64. IEEE, 2015.

[3] Jian Chen, Dong Sun, Jie Yang, and Haoyao Chen. Leader-follower formation control of multiple non-holonomic mobile robots incorporating a receding-horizon scheme. *The International Journal of Robotics Research*, 29(6):727–747, 2010.

[4] Yang Quan Chen and Zhongmin Wang. Formation control: a review and a new consideration. In *2005 IEEE/RSJ International conference on intelligent robots and systems*, pages 3181–3186. IEEE, 2005.

[5] Mario Coppola, Kimberly N McGuire, Kirk YW Scheper, and Guido CHE de Croon. On-board communication-based relative localization for collision avoidance in micro air vehicle teams. *Autonomous robots*, 42(8):1787–1805, 2018.

[6] Alejandro Cornejo and Radhika Nagpal. Distributed range-based relative localization of robot swarms. In *Algorithmic Foundations of Robotics XI*, pages 91–107. Springer, 2015.

[7] Jin Q Cui, Swee King Phang, Kevin ZY Ang, Fei Wang, Xiangxu Dong, Yijie Ke, Shupeng Lai, Kun Li, Xiang Li, Jing Lin, et al. Search and rescue using multiple drones in post-disaster situation. *Unmanned Systems*, 4 (01):83–96, 2016.

[8] Lefteris Doitsidis, Stephan Weiss, Alessandro Renzaglia, Markus W Achtelik, Elias Kosmatopoulos, Roland Siegwart, and Davide Scaramuzza. Optimal surveillance coverage for teams of micro aerial vehicles in gps-denied environments using onboard vision. *Autonomous Robots*, 33(1-2):173–188, 2012.

[9] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E Gusz Eiben. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4, 2015.

[10] David Droeschel, Matthias Nieuwenhuisen, Marius Beul, Dirk Holz, Jörg Stückler, and Sven Behnke. Multilayered mapping and navigation for autonomous micro aerial vehicles. *Journal of Field Robotics*, 33(4): 451–475, 2016.

[11] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. *Journal of Field Robotics*, 33(4):431–450, 2016.

[12] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *2013 IEEE International Conference on Robotics and Automation*, pages 993–998. IEEE, 2013.

[13] T.A. Fijen. Persistent surveillance of a greenhouse: evolved neural network controllers for a swarm of uavs. Master's thesis, TU Delft, 2018.

[14] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(3):396–407, 1996. doi: 10.1109/3477.499791.

[15] Kexin Guo, Zhirong Qiu, Wei Meng, Lihua Xie, and Rodney Teo. Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in gps denied environments. *International Journal of Micro Air Vehicles*, 9(3):169–186, 2017.

[16] Kexin Guo, Xiuxian Li, and Lihua Xie. Ultra-wideband and odometry-based cooperative relative localization with application to multi-uav formation control. *IEEE transactions on cybernetics*, 50(6):2590–2603, 2019.

[17] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32, 2009.

[18] David Howard and Farid Kendoul. Towards evolved time to contact neurocontrollers for quadcopters. In *Australasian Conference on Artificial Life and Computational Intelligence*, pages 336–347. Springer, 2016.

[19] Jaime L Junell, Erik-Jan Van Kampen, Coen C de Visser, and Q Ping Chu. Reinforcement learning applied to a quadrotor guidance law in autonomous flight. In *AIAA Guidance, Navigation, and Control Conference*, page 1990, 2015.

[20] Anton Kohlbacher, Jens Eliasson, Kevin Acres, Hoam Chung, and Jan Carlo Barca. A low cost omnidirectional relative localization sensor for swarm applications. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 694–699. IEEE, 2018.

[21] CH Kuo, CM Kuo, A Leber, and C Boller. Vector thrust multi-rotor copter and its application for building inspection. In *International micro air vehicle conference and flight competition, Toulouse, France, Sept*, pages 17–20, 2013.

[22] Bryce Mack, Christopher Noe, Trevor Rice, In Soo Ahn, and Jing Wang. Distributed localization and control of quadrotor uavs using ultra-wideband sensors. In *2019 IEEE Aerospace Conference*, pages 1–9. IEEE, 2019.

[23] György Max and Béla Lantos. Adaptive formation control of autonomous ground vehicles in leader-follower structure. In *2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 000013–000018. IEEE, 2016.

[24] Patrick P Neumann, Paul Hirschberger, Zhandos Baurzhan, Carlo Tiebe, Michael Hofmann, Dino Hüllmann, and Matthias Bartholmai. Indoor air quality monitoring using flying nanobots: Design and experimental study. In *2019 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*, pages 1–3. IEEE, 2019.

[25] Thien-Minh Nguyen, Zhirong Qiu, Thien Hoang Nguyen, Muqing Cao, and Lihua Xie. Distance-based cooperative relative localization for leader-following control of mavs. *IEEE Robotics and Automation Letters*, 4(4):3641–3648, 2019.

[26] Stefano Nolfi, Dario Floreano, and Director Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[27] Soo-Hun Oh and Jinyoung Suk. Evolutionary controller design for area search using multiple uavs with minimum altitude maneuver. *Journal of Mechanical Science and Technology*, 27(2):541–548, 2013.

[28] Piotr Rudol and Patrick Doherty. Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. In *2008 IEEE aerospace conference*, pages 1–8. Ieee, 2008.

[29] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[30] Martin Saska, Vojtěch Vonásek, Jan Chudoba, Justin Thomas, Giuseppe Loianno, and Vijay Kumar. Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent & Robotic Systems*, 84(1-4):469–492, 2016.

[31] Kirk YW Scheper and Guido CHE de Croon. Abstraction, sensory-motor coordination, and the reality gap in evolutionary robotics. *Artificial Life*, 23(2):124–141, 2017.

[32] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Evolving coordinated group behaviours through maximisation of mean mutual information. *Swarm Intelligence*, 2(2-4):73–95, 2008.

[33] Dinesh Thakur, Giuseppe Loianno, Wenxin Liu, and Vijay Kumar. Nuclear environments inspection with micro aerial vehicles: Algorithms and experiments. In *International Symposium on Experimental Robotics*, pages 191–200. Springer, 2018.

[34] Steven van der Helm, Mario Coppola, Kimberly N McGuire, and Guido CHE de Croon. On-board range-based relative localization for micro air vehicles in indoor leader–follower flight. *Autonomous Robots*, 44(3): 415–441, 2020.

[35] Viktor Walter, Martin Saska, and Antonio Franchi. Fast mutual relative localization of uavs using ultraviolet led markers. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1217–1226. IEEE, 2018.

[36] Viktor Walter, Nicolas Staub, Martin Saska, and Antonio Franchi. Mutual localization of uavs based on blinking ultraviolet markers and 3d time-position hough transform. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 298–303. IEEE, 2018.

[37] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[38] Ben Yun, Ben M Chen, Kai-Yew Lum, and Tong H Lee. A leader-follower formation flight control scheme for uav helicopters. In *2008 IEEE International Conference on Automation and Logistics*, pages 39–44. IEEE, 2008.

II

Literature Review

1

Relative Localization

Autonomous navigation is influenced not only by the goals of the flight, but particularly by the sensors available to the MAV so that it can perceive its environment in order to know what to do. Moreover, if the goal is to achieve autonomous flight of a group of MAVs instead of only one, the task becomes increasingly complex, since each MAV needs to account for additional dynamic elements within the environment. In fact, the MAVs need to plan their trajectory considering, not only, the static environment, but also, the other members of the group. Furthermore, if complex collective behaviours want to be explored, it is vital that an agent has an accurate estimate regarding the relative position of its neighbours. For this reason, relative localization is a high interest topic within the scientific community.

In the case where access to positional systems, such as GNSS or LPS, is possible, it is easier for an MAV to find its position within the flying environment, therefore, enabling an easy relative localization estimate within the global frame of reference, as long as the agents can communicate with each other. However, access to these systems is not possible when flying in unknown indoor scenarios, which are the ones MAVs should ultimately aim for. Hence, relative localization methods that only rely on on-board sensors must be explored.

In this chapter, an overview of different relative localization methods will be performed. First, in Section 1.1, a relative localization framework is introduced. Next, Section 1.2 discusses on-board based solutions that can provide the required measurements for the aforementioned relative localization framework. Finally, Section 1.3 reviews different sensor fusion approaches used in the literature, and Section 1.4 delves into depth on the state of the art work this thesis aims to follow.

1.1. State of the Art Framework

The first step to solve the relative localization problem is defining which framework to follow in order to get relative localization estimates. Depending on the framework that one decides to use, different relative pose measurements may be needed.

The general ego-centric framework inspired in the work of Howard et al. [30] describes a method where each robot can determine the pose of every other robot in a swarm, relatively to itself. Considering two MAVs, R_i and R_j, with body frames F_{B_i} and F_{B_j}, respectively, the full relative pose of R_j with respect to R_i can be defined as:

$$\vec{P}_{ji} = [x_{ji}, y_{ji}, z_{ji}, \phi_{ji}, \theta_{ji}, \psi_{ji}], \tag{1.1}$$

where x_{ji}, yji and z_{ji} are the position of R_j in F_{B_i}. Further, ϕ_{ji}, θ_{ji} and ψ_{ji} are the roll, pitch and yaw of F_{B_j} with respect to F_{B_i}.

However, it is possible to simplify the relative pose of R_j with respect to R_i by using a variation of the typical F_{B_i} body frame: $\mathcal{H}_i$. $\mathcal{H}_i$ can be defined as a horizontal body reference frame, i.e., the Z-axis remains parallel to the earth-fixed North-East-Down (NED) reference frame, $\mathcal{I}$. In this way, Equation 1.1 can be re-arranged [69] with cylindrical coordinates as:

$$\vec{P}_{ji_{cc}} = [r_{ji}, h_{ji}, \psi_{ji}], \tag{1.2}$$

where r_{ji} is the absolute range between the origins of $\mathcal{H}_i$ and $\mathcal{H}_j$, see Equation 1.3; h_{ji} is the height between R_i and R_j, see Equation 1.4; and ψ_{ji} is the orientation of R_j with respect to R_i, see Equation 1.5. This framework is illustrated in Figure 1.1.

$$r_{ji} = \sqrt{x_{ji}^2 + y_{ji}^2 + z_{ji}^2} \tag{1.3}$$

$$h_{ji} = h_j - h_i \tag{1.4}$$

$$\psi_{ji} = \psi_j - \psi_i \tag{1.5}$$

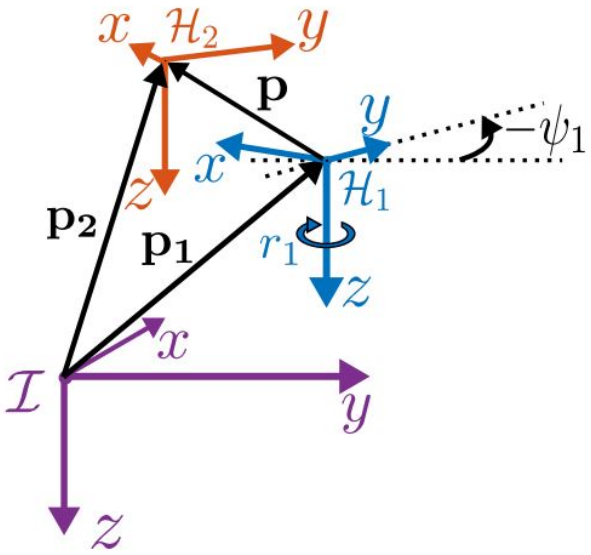


Figure 1.1: Reference frame $\mathcal{I}$ (purple) is the earth-fixed NED frame (assumed to be inertial); reference frames $\mathcal{H}_1$ (blue) and $\mathcal{H}_2$ (orange) are the horizontal body-fixed frames for drones 1 and 2, respectively [69].

Established the framework, two things stand out. First, the measurements that are required to produce relative localization estimates: range, height difference and relative orientation. Secondly, inter-MAV communication is highly recommendable to facilitate the computation of certain estimates. For instance, consider that each drone is equipped to obtain its own height and heading via on-board sensors, such as a pressure-sensor or a magnetometer, respectively. Then, by exchanging these measurements, each drone can compute the corresponding relative estimates. As for the range, drones are not equipped with specialized sensors to measure it, therefore, it is necessary to investigate which combination of on-board sensors and methods can render it. Once again, inter-drone communication may prove to be vital.

1.2. On-Board Methods for Range Measurements

An overview of the different on-board methods that have been successfully used to obtain relative range measurements will be performed in this section. As it was already referred, the solution this thesis aims to develop excludes the use of any external source of information. Therefore, the use of external systems, like Motion Capture Systems (MCSs) or Global Position System (GPS), to produce relative measurements will not be discussed.

The following subsections will focus on the state of the art methods used in the literature. Both vision-based and signal-based methods will be discussed, by providing a concise explanation on how they work, their advantages and handicaps, as well as how they were applied for research purposes.

1.2.1. Vision-based Methods

Vision-based methods make use of the video/image captured by a robots' camera. There are different vision-based methods that can provide useful measurements for the relative localization problem.

Starting with Simultaneous Localization and Mapping (SLAM) algorithms, these try to solve the problem in which a mobile robot is placed in an unknown environment without any knowledge of its location and, over time, tries to, simultaneously, incrementally create a map of its surroundings and estimate its own location within this map.

Taking this approach into account, it is possible to have different robots collaborate with each other, by communicating and sharing their cognition about the environment, and generate a shared map of the environment while determining the position of each robot in the map [7, 29]. Carlone et al. [7] presents a SLAM strategy for a multi robot system in which robots only exchange information when a rendezvous event occurs. The robots are equipped with a camera, a custom visual marker, and a wireless communication system. A similar solution is introduced in Hidayat et al. [29]. In this way, and using only on-board sensors, it is possible to approximately replicate the scenarios in which a global reference frame is available, therefore, facilitating the relative localization estimates.

In order to construct the map of the environment, the robot(s) must be equipped with sensors that allow it to perceive the environment and obtain measurements from it. When cameras are employed as the sole exteroceptive sensor, SLAM algorithms fall in the visual SLAM category. These methods are often augmented with information from proprioceptive sensors, such as accelerometers and gyroscopes. Visual SLAM methods are preferred because of their ability to obtain range information while retrieving other environment's characteristics, as colors and textures, that may be useful for other purposes. Furthermore, cameras are relatively cheap and light, which is important for MAVs applications, for instance. Nonetheless, cheap and light cameras that do not demand too much power also have problems (low camera resolution, sensitivity to lighting, lack of textures) that negatively affect the quality of the resulting estimates, leading to inconsistent maps that may endanger the exploration of collective behaviours of a group of MAVs. Moreover, SLAM algorithms are computationally expensive, which is an additional handicap for their implementation in this type of robots.

As referred, in SLAM methods cameras are used to obtain range information. Therefore, they can be an option to simply retrieve the necessary range measurements for the relative localization framework presented in Equation 1.2. A robot can derive range measurements by directly analyzing the images captured by its own camera(s). Usually this involves two steps, recognizing the target robot and, using pixels matching techniques and geometrical properties, estimate the desired range (and orientation, if one wants to avoid the use of magnetometers). Because the recognition process may be overly demanding, some authors suggest the addition of easily identifiable markers to the MAVs' in order to facilitate it [16, 74, 75].

Walter et al. [74], for instance, proposes an outdoor solution where UAVs are equipped with Ultraviolet (UV) Light-Emitting Diode (LED) markers and a suitable camera with specialized bandpass filters. The active ultraviolet markers emit light in frequencies less familiar in nature, which assures better precision and reliability against different conditions and environments, without compromising the cost, size and weight of the overall system as well as the required computational load. Later [75], their work was extended by using the Hough Transform to encode individual marker IDs in blinking patterns. This feature enables the distinction between different UAVs and sets up an easier approach to filter out other light sources and its reflections (non-blinking bright spots).

A different type of marker is proposed by Faigl et al. [16]. Instead of using LEDs, a circular marker formed by two concentric circles of contrasting colours is attached to each robot. The results showed that relative localization within a robotic swarm can be achieved with a good trade-off between detectability, precision and computational power. However, by experimenting with the radius of the circles, it became clear that smaller sizes of the pattern, despite favouring the practical deployment of the system, would affect negatively the performance of the system. Moreover, this strategy comes with an additional limitation: the marker of a target MAV needs to be in the field of view of the other's camera in order to occur detection. This means that it is not possible to guarantee continuous access to range measurements, which is vital for the relative localization framework.

A complete different way of using cameras to produce range measurements is presented in Milano et al. [45]. Robots are equipped with a light emitter and a camera, both pointing perpendicularly to the ceiling. Hence, each robot can observe, on the ceiling, a map with its own position at the azimuth and the relative position of the other robots, which allows to measure the relative pose among them. Despite the results showing the effectiveness of the method for multi-robot tasks - rendezvous and formation control - in a group

of ground robots in a controlled environment, its translation to unknown scenarios might be difficult since it is built on several assumptions that will probably not hold true in most cases: robots need to move at the same Y-plane, height of the ceiling is available (or can be estimated), and the ceiling is approximately flat.

Overall, all the aforementioned methods have limitations that raise concerns regarding their applicability within a swarm of MAVs in an unknown indoor environment. First, they are computationally expensive. Secondly, despite improvements with the use of LED markers [74, 75], they are still dependent on the illumination and vision contrast of the environment. Moreover, the limited field of view of the cameras can lead to restricted Line of Sight (LoS) operations. Finally, adding markers to facilitate the recognition stage may affect the size and weight of the overall system, which is intended to be small, as well as compromise the generalization of the method to other systems.

1.2.2. Signal-based Methods

Signal-based methods rely on the transmission of a signal between two MAVs and can be supported by different hardware systems. These methods are usually divided into two main categories:

- Received Signal Strength (RSS), in which the range between a transmitter and a receiver is estimated based on the signal strength.
- Time-based, in which the range between two antennas is estimated from measuring the signal propagation time between them;

Received Signal Strength

The power of electromagnetic waves decreases over distance, hence, it is possible to calculate the distance between a receiver and a transmitter based on the strength of the received signal. This can be achieved with different technologies, e.g. Bluetooth or Ultra Wideband (UWB), when communicating a signal between two antennas. Like any other signal, the estimated strength will have a noisy component which can be, for instance, a consequence of reflections in the environment.

There are two main methods when using RSS to estimate distances for indoor localization purposes: finger-printing and model-based. The first method requires a previous assessment of the environment, since it uses multiple beacons - static or mobile - to build a map - finger-print - of the RSS distribution in an environment. With that, position can be derived for a certain individual. It becomes clear that this method is not suitable for the goals this thesis aims to achieve, as it is not compatible with operating in unknown environments.

Regarding the second method, model-based, it entails the construction of a model of the propagation of the signal that correlates RSS with distance. In Coppola et al. [8], a Bluetooth RSS based relative localization scheme is developed in order to study collision avoidance strategies. The algorithm was tested with teams of two and three UAVs, providing satisfying results for the range measurements and consequence localization task. Despite proving to be an interesting solution, one of its drawbacks is that it is difficult to account for multi-path interference, since the environment is unknown and there can be several different sources of interference. Moreover, RSS has a limited maximum range, which affects the applicability of methods relying on it.

Time-based Methods

The velocity of electromagnetic signals or sound waves in the air are well established values. With that information, it is possible to estimate the distance between two devices by measuring the propagation time of a signal transmitted between them. There are several time-based methods that may be employed for this purpose:

- Time of Arrival (ToA), which measures the time it takes for a signal to travel from a transmitter to a receiver. Then, the receiver can extract the range. Usually, these methods include another signal being transmitted that acts as a trigger.
- Round-trip Time of Arrival (RToA), where a transmitter measures the time a signal takes to travel to a receiver and back to it again. If the transmitter knows the processing time of the other device, it can compute the range between them.

- Time Difference of Arrival (TDoA), where a transmitter emits a signal that is received by at least three receivers. If the position of the three receivers is known, the difference between their arrival times can be used to perform trilateration and extract the localization of the transmitter.

The aforementioned methods are implemented in the literature using different technologies and hardware. State of the art systems rely on radio-based technology, like UWB or Bluetooth, or sound-based systems.

ToA methods alone do not have great value for this research, since if one MAV wants to measure the range with respect to another, the one being targeted needs to send back information about the range, because only the receiver can measure it. Thus, if communication in both ways is necessary, you might as well use RToA methods, since they use this two-way communication scheme to compute better range estimates.

In Lin et al. [39], acoustic signals are used in order to estimate the relative pose of a team of mobile robots through a RToA method. The robots emit custom sound waveforms that enable both robot identification and range estimation. Moreover, in Guo et al. [24] and later in Guo et al. [25], a RToA method is implemented in order to collect range measurements for a UWB-based localization filter for UAVs navigation. Furthermore, in van der Helm et al. [70], a range-only relative localization strategy is developed, based on RToA UWB range measurement. Results showed that the distribution of the ranging error throughout flights of two MAVs is close to a Gaussian distribution around a mean ranging error of about -6.4cm.

Despite the good results, similarly to RSS range measurements, time-based methods are also sensitive to multi-path interference that may cause ambiguity. With this in mind, there are several extensions of the basic RToA method that account for this possibility and decrease the worst case scenario error [37, 46]. Obviously, this extensions come with a computational cost, thus, it is a decision of the designer to determine if there is indeed a need to implement them, or if the accuracy of the regular range measurements is already good enough.

With respect to TDoA methods, they mostly require the use of fixed beacons with a well-known position, in order to obtain range measurements. Therefore, their value for this research is scarce. Nonetheless, Kohlbacher et al. [36] proposes a relative localization sensor system for swarm applications, using UWB to obtain ranging measurements with a TDoA method. Each agent is equipped with a tetrahedral antenna array with UWB sensors that enables the application of TDoA to measure the relative range (and orientation) of other agents. Although the addition of an antenna array is not ideal, because it increases the size and weight of the system, depending on the applications it might represent a great solution, since it only weights 56g.

It is important to add, regarding the specific use of acoustic systems, that they are negatively affected by other sounds that may occur in the environment, which can be problematic when tackling most scenarios. On the other hand, radio-frequency based systems, have a great advantage in versatility. They can be used, not only, to obtain range measurements, but also, to communicate between agents [70]. This is a powerful asset, as mentioned in Section 1.1, since it erases the need of having a different system on-board to handle communication tasks, reducing the weight of the overall system.

Within radio technology, UWB is becoming the preferred option within localization systems, since it has several advantages when compared with others, such as Bluetooth, Wifi or ZigBee. First, it can operate in a wide range of frequencies, which helps overcoming multi-path issues and dealing with interference from other signals. Moreover, it has a wide communication range, as long as communication speed can be sacrificed, providing flexibility for different swarm applications. Finally, indoor environments are often populated with obstacles for communication between agents, however, with UWB this is not a big problem, since it has favourable material penetrating properties, when compared with other wireless technologies.

Having examined the different methods and technologies that are used on-board robots in order to obtain range measurements, it is time to discuss how can one use range, height difference and relative orientation measurements to build a relative localization system for a group of mobile robots.

1.3. Sensor Fusion

Sensor Fusion is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually [15]. In a real scenario, measurements and states are stochastic, i.e., they are subject to unexpected noise that leads to biased estimates, reducing the quality of the derived dynamics models. By combining different sensory data from different sensors, it is possible to mitigate this noise and improve the accuracy of state

estimates. Actually, it has already been stated that sensor fusion between relative ranging and relative motion measurements leads to better results for the relative localization estimates. In this case, the strengths of inertial sensors - high sample rates and short term accuracy - are combined with those of, for instance, UWB - stable accuracy over time; in order to overcome the weaknesses of both: bias drift for inertial sensors and lower short term accuracy for UWB.

Great part of the researches in the literature that use UWB to produce range measurements does not fit the interests of this thesis, since fixed beacons with well-known locations are used to obtain a localization estimate through trilateration. Nonetheless, they are worth mentioning in this section as they are an example of how sensor fusion of UWB ranging measurements with other sensory data can improve the final localization estimate.

In fact, in González et al. [23], the problem of robot localization using UWB range measurements and trilateration is addressed by using a Particle Filter to combine different measurements from the UWB beacons and robot odometry. Moreover, a probabilistic model for the UWB ranges within different environments is derived, in order to correct for the possible errors due to multi-path effects. The proposed solution was validated through different scenarios, proving its feasibility. Using trilateration in a similar way [48], an Extended Kalman Filter (EKF) was implemented to fuse UWB range measurements and IMU ones in order to improve localization estimates. After some test flights, the Root Mean Square Error (RMSE) of the localization estimate in the x-y plane was 0.123m and in altitude was 0.345m.

The trilateration algorithm is again used in Guo et al. [24]. An EKF is implemented to achieve highly accurate position and velocity estimates from UWB range measurements. Later [25], the work was extended to tackle the UWB based Relative Localization problem for a team of UAVs. Using, again, an EKF to fuse UWB range and self-displacement measurements, relative localization estimates were achieved for a group of three flying UAVs.

Continuing with the relative localization problem for a team of MAVs, in Coppola et al. [8], an EKF is used to combine Bluetooth RSSI range measurements, ultrasonic sensor height measurements, magnetometer heading measurements, and Lucas-Kanade based optical flow velocity measurements to perform relative localization. In a test flight with two drones, the RMSE of the relative range estimates was 1.18m, whereas for relative heading estimates the RMSE was 0.77rad. Following this work, van der Helm et al. [70] replaced Bluetooth RSSI with UWB in order to obtain the range measurements. Moreover, because magnetometer measurements are usually noisy, the dependency on heading measurements was removed, and the impact of this removal on the relative localization estimates was investigated. In the end, an EKF solution was implemented as to combine range, velocity and height measurements. When flying two drones in a leader-follower strategy, with a maximum distance of 5.2m between themselves, the relative localization mean error was 22.6cm, whereas the maximum error was 75.8cm, confirming that dropping the magnetometer measurements would indeed not spoil the relative localization estimates. Since this solution [70] does not require extra hardware, and hence is promising even for MAVs, it will be the one this thesis aims to follow and, if possible, improve. With this in mind, the following section will delve deeper on this study and provide further details regarding its development and implementation.

1.4. State of the Art Work

In order to find a control solution for the problem of high risk of collisions between teams of MAVs, Coppola et al. [8] first introduced a Bluetooth-based relative localization scheme. The proposed localization strategy combines relative range measurements, provided by Bluetooth technology, with three on-board states: velocity, height and orientation. Following this work, van der Helm et al. [70] proposed a similar solution with only a couple changes. First, the range measurements would be performed by UWB technology, for the reasons already explored in the end of section 1.2.2. And secondly, the orientation measurements would not be available, because magnetometers' readings are easily affected by interference, which might spoil the relative localization estimates, especially in unknown indoor scenarios. The new scheme was tested with an EKF and the results corroborated the hypothesis that a range-only relative localization filter provides a better performance than a filter that also has access to orientation measurements, even for small magnetic perturbations.

The following sections review the nonlinear system that was proposed [70] to describe the relative localization problem, as well as the EKF implemented to combine the different sensor measurements and perform the state estimation task, in order to execute simulations and real flight experiments.

1.4.1. Nonlinear System Description

Recalling the framework and reference frames introduced in Section 1.1, and considering MAV 1 wants to estimate the relative position of a different MAV 2, then, this relative position can be denoted by vector $p = p_2 - p_1$, as seen in Figure 1.1. Additionally, the difference in heading between two MAVs can be represented by $\psi_{21} = \psi_2 - \psi_1$, and the yaw rates by r_i. Moreover, the linear velocities and accelerations in frame $\mathcal{H}_i$ with respect to frame $\mathcal{I}$, expressed in frame $\mathcal{H}_i, i = 1,2$, can be defined as v_i and a_i. Lastly, since $\mathcal{H}_i$ is a horizontal reference frame and MAVs have direct access to height measurements, height will not be included in the system description, so as to simplify the problem. This simplification leads to all linear variables being 2D vectors.

Having introduced the variables of interest, the complete state, x, and the inputs, u, of the system can be defined by:

$$x^T = \left[p^T, \psi_{21}, v_1^T, v_2^T\right]^T \tag{1.6}$$

$$u^T = \left[a_1^T, a_2^T, r_1, r_2\right]^T \tag{1.7}$$

Then, the continuous time state can be defined by:

$$\dot{x} = f(x,u) = \begin{bmatrix} -v_1 + Rv_2 - S_1 p \\ r_2 - r_1 \\ a_1 - S_1 v_1 \\ a_2 - S_2 v_2 \end{bmatrix} \tag{1.8}$$

$$y = h(x) = \begin{bmatrix} ||p|| \\ v_1 \\ v_2 \end{bmatrix} \tag{1.9}$$

With R being the 2D rotation matrix from frame $\mathcal{H}_2$ to $\mathcal{H}_1$:

$$R = R(\psi_{21}) = \begin{bmatrix} cos(\psi_{21}) & -sin(\psi_{21}) \\ sin(\psi_{21}) & cos(\psi_{21}) \end{bmatrix} \tag{1.10}$$

And matrices S_1 and S_2 being the skew-symmetric matrix equivalent of the cross product for the 2D scenario. Matrix S_i can be defined as:

$$S_i = S_i(r_i) = \begin{bmatrix} 0 & -r_i \\ r_i & 0) \end{bmatrix}, i = 1,2 \tag{1.11}$$

It is worth mentioning that variables a_i and r_i, for $i = 1,2$, are inputs of the system, to which MAV 1 has access to by means of its own accelerometer and gyroscope data, as well as MAV 2 data transmitted by an UWB signal.

1.4.2. Filter Design

The most famous type of filter used in signal processing applications, from radio systems to robotics, is the Kalman filter (KF). The main idea behind it is the calculation of a weighted average between the measured and the predicted state, where the weight, called Kalman gain, depends on the uncertainty in the measurement. One of the limitations of the KF is that it only works for linear systems. However, in the real world, the system and measurement equations are mostly nonlinear. Under this scenario, the Extended Kalman Filter (EKF) can be used to estimate the state vector.

The presented relative localization framework is employed through an EKF. The reason to do so is because the way the state-space system was described 1.4.1 fits directly into the way the EKF is structured, since the EKF also utilizes a state differential model and an observation model. Therefore, one can just apply Equations 1.8 and 1.9, respectively. Nonetheless, the EKF has additional parameters that must be tuned: the system and measurement noise matrices, the initial state and initial state covariance matrix. Regarding the first two matrices, the system and noise ones, they can be tuned accordingly to the expected noises that

result from experiments. In these experiments, the perceived states and the on-board measurements are collected, together with their true values, so that later they can be analysed, in order to figure out what the noise values should be. Regarding the initial state conditions, the EKF is initialized with values similar to the real ones, otherwise the filter can be difficult to converge, due to the uncertainty of certain states' regions. Consequently, the initial state covariance matrix should be initialized with small values (≤ 0.1).

1.4.3. Observability

Despite yielding good results, this relative localization filter comes with the cost of extended flight conditions constraints that must be met in order to guarantee the observability of the system and, as a consequence, the relative localization estimates. If the state becomes unobservable, then the filter will most likely start diverging, compromising the safety of the MAVs.

With this in mind, Chapter 2 will dive further into the topic of observability in range based relative localization systems.

2

Observability in Range Based Relative Localization Systems

The main goal of a relative localization scheme is that a robot R_i is able to track the relative position of another robot R_j. When a filter is used to fuse different sensory data, variables that are not measured directly may become observable. This is the case in case detailed within Section (1.4), where the filter does not have any sensor that provides orientation measurements. Nonetheless, as a consequence, the observable subspace of the system is also reduced, which means that in practice MAVs must comply to certain flight conditions in order to guarantee the convergence of the filter. This constraint may or may not restrict the individual movement of the MAVs, compromising the exploitation of particular collective behaviours that are essential to all collaborative missions. Thus, it becomes important to understand how the direct measurements of range-only relative localization schemes impact the observability of the system states.

This chapter starts by introducing the theoretical concept of local weak observability - Section 2.1, which is the state of the art tool used to evaluate the observability of nonlinear systems. Then, Section 2.2 performs an overview of observability in range-based relative localization frameworks, as well as its practical implications. Finally, Section 2.3 discusses the possibility of quantitatively assessing the observability of a system over time, which may be helpful to better understand the real impact of uncertainty in relative state estimation.

2.1. Local Weak Observability

Before introducing the concept of local weak observability [28], one should recall what observability means. First, let's introduce the notion of indistinguishable states. Two states x_0, x_1 are indistinguishable if, at every time $t \geq t_0$ and for each admissible input function $u(t)$, $y(t, t_0, x_0, u(t)) = y(t, t_0, x_1, u(t)$. Then, one can say that state x_0 is observable if the set of indistinguishable states from x_0 is empty. Finally, a system is considered observable if every state x of the state space is observable [28]. However, this take on the observability of a system does not necessarily indicate that the initial conditions of a state are distinguishable for every input. In fact, this is only possible to imply for linear systems, where the output can be written as a function of the initial state and the input.

In nonlinear systems, one can evaluate observability through an analytical tool called local weak observability. This property is the combination of two others: local observability and weak observability. The first one states that two states are locally observable if their evolution allow to immediately distinguish between them. On the other hand, if it is possible to distinguish a state x_0 from the states belonging to its neighbourhood, the system is said to be weakly observable. Therefore, one can argue that a system is locally weakly observable if $y(t, x_0(t), u(t))$ and $y(t, x_1(t), u(t))$ are different for two different states in the same neighbourhood $x_1(t)$, $x_2(t)$; $t > t_0$ and any admissible input function $u(t)$. The latter property can be inferred easily by performing some algebraic tests, which is the reason to its popularity among researchers.

Considering a generic non-linear state-space system $\sum$ defined by:

$$\dot{x} = f(x,u) \tag{2.1}$$
$$y = h(x) \tag{2.2}$$

With the state vector $x \in \mathbb{R}^n$, an input vector $u \in \mathbb{R}^l$, and an output vector $y \in \mathbb{R}^m$.
And defining the Lie derivatives of the output function as:

$$\mathcal{L}_f^0 h = h \tag{2.3}$$
$$\mathcal{L}_f^1 h = \nabla \otimes \mathcal{L}_f^0 h \cdot f \tag{2.4}$$
$$\vdots$$
$$\mathcal{L}_f^i h = \nabla \otimes \mathcal{L}_f^{i-1} h \cdot f \tag{2.5}$$

Where ∇ is the differential operator defined as $\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_n}\right]$ and $\otimes$ is the Kronecker product.

Conclusions regarding the local weak observability of the system can be drawn by applying the observability rank condition. The latter states that $\sum$ is locally weakly observable if matrix $\mathcal{O}$ is full rank [28]. This is known as sufficient condition for observability. $\mathcal{O}$ is defined as:

$$\mathcal{O} = \begin{bmatrix} \nabla \otimes \mathcal{L}_f^0 h \\ \nabla \otimes \mathcal{L}_f^1 h \\ \vdots \\ \nabla \otimes \mathcal{L}_f^i h \end{bmatrix}, i \in \mathbb{N} \tag{2.6}$$

2.2. Observability in Range Based Relative Localization Systems

All type of systems may have limited access to global localization measurements. It can happen not only with MAVs and ground robots, but also with Autonomous Underwater Vehicles (AUV). For this reason, the observability issues related to range-based relative localization solutions have been widely investigated.

Antonelli et al. [1] investigates how the motion of AUVs can be driven in order to avoid unobservable states. First, when linearizing the model and analysing its observability matrix, they realized that a null angular velocity from the vehicle performing localization results in a rank 2 matrix. Moreover, if both vehicles are aligned along the same vertical axis and the vehicle being localized has null roll and pitch angles, the matrix has rank 1. This has influence on a typical underwater movement: diving. Additionally, for this specific maneuver, not even local weak observability could be guaranteed. With this information, they implemented a new diving maneuver, where the vehicle being localized descends with an helix path, as to maintain observability. Furthermore, they realized that another typical cooperative movement, when the vehicles navigate with same speed and direction, would lead the system into unobservable scenarios. Consequently, they implemented a new translational movement where the vehicle being localized makes use of a sinusoidal velocity. These unobservability scenarios are also confirmed in Arrichiello et al. [2].

Additionally, Cornejo and Nagpal [9] discusses the existence of two degenerate motions featured in range-only systems: flip ambiguity and rotation ambiguity. The first emerges when two robots follow a linear trajectory, not necessarily parallel. In this case it is not possible, with only range measurements, to establish on which side a robot is with respect to the other. To solve this issue, relative orientation and velocity measurements must be used. The second motion is similar to the translational maneuver already discussed in the AUVs case. If one robot replicates the motion of another robot, relative bearing becomes unobservable, unless relative orientation measurements are available.

van der Helm et al. [69] goes a step further and performs a study over the differences in observability of a relative localization filter with and without a common heading reference. First, when heading measurements were available, the results mentioned in the AUVs case were also confirmed. Moreover, it was concluded that

at least one of the vehicles must be moving to render the system observable, which has direct influence in MAVs hovering, for instance. For the case where only range measurements were available, the removal of the relative heading measurements results in a system that requires at least one extra Lie derivative in the range observation to make the system locally weakly observable. Therefore, the condition that guarantees full relative observability of the system are more complex. Nonetheless, some intuitive conditions could be identified. Again, the vehicles can not move along the same vertical axis. Moreover, both MAVs need to be moving - either through a non-zero velocity or non-zero acceleration - contrary to the system with heading measurements, where only one needed to be moving. Additionally, they cannot fly in parallel regardless of their speed, unless one of them is simultaneously accelerating; as opposite to the case with heading readings, where they can move in parallel if they do it with different velocities. Despite being difficult to evaluate the full observability condition for the case with no heading measurements, a numerical analysis for a particular set of velocities and accelerations, as well as an analysis of data from a leader-follower flight with two MAVs were performed. The latter showed that only a small percentage of the data - 4.75% for the flight with only on-board measurements - is unobservable. Moreover, it was concluded that the system could easily recover from short periods of unobservability, and that these did not have a great influence on the relative localization error.

In order to deal with the constraints of relative localization schemes based on range measurements, Nguyen et al. [49] presents an innovative solution. Their goal is to implement a leader-following control strategy on a team of MAVs, using a relative localization filter that fuses range measurements and odometry sensors. To overcome the challenges previously mentioned in this section, they propose a cooperative estimation-control framework for a leader-follower task. In addition to the leader and the followers, there are other agents called orbiters, which mission is to continuously maintain a flying trajectory that guarantees the convergence of the filter for itself and the other agents. The method is validated through simulation results for a group of six MAVs (one leader, 3 followers and two orbiters), and then successfully implemented in real flying experiments qith three MAVs (one of each). Despite the efficacy of the method, its efficiency can be questioned, since one third of the system is basically just repeatably flying the same trajectory. Moreover, the scalability of the method might be challenging, since in order to grow the "usable" system (leader and folowers) by two, one needs to always add an additional orbiter, increasing the airspace density, therefore, the probability of collisions. This can be critical in narrow indoor spaces.

Despite the aforementioned limitations, it is important to state that the observability rank condition only provides binary information regarding the observability of the system. However, it would be interesting to have a quantitative indication about how observable a particular system is, as it would allow a better understanding into what influences and drives uncertainty in practice. Moreover, this would possibly allow the exploration of control solutions that take advantage of observability information to model the behaviour of robots. The next section introduces observability measures that can be used to overcome this issue.

2.3. Observability Measures

At this point, certain conditions under which a range-based relative localization filter is (un)observable have been discussed. However, not only are these conditions insufficient to represent all the possible scenarios, but also this type of analysis does not provide information on the degree of observability of the system.

Imagine that a group of robots equipped with a range-based relative localization filter could also measure, in real time, the degree of observability of the system. Then, they could possibly leverage this information and steer themselves into situations where observability is maximized.

For nonlinear systems, the condition that yields local weak observability is a full rank observability matrix as the one from Equation 2.6. It is well known that a matrix is full rank if its determinant is different than zero, which is why in the previous section those conditions were easily drawn. Looking at the combination of variables that result in a null determinant can be pretty straightforward, depending on the complexity of the system. Therefore, it would be intuitive to look at the determinant of the matrix as a measure of how observable the system is: if the determinant is close to zero, then the system is close to being unobservable. However, it would be wrong. The determinant of a matrix does not provide a good measure of how close a matrix is to being singular [2]. In fact, looking at its singular values would be a much more valuable solution [22]. Following this line of thought, it is possible to quantify the unobservability of the system using either the unobservability index, ξ, or the estimation condition number, λ:

$$\xi = \frac{1}{\sigma_{min}} \tag{2.7}$$

$$\lambda = \frac{\sigma_{max}}{\sigma_{min}} \tag{2.8}$$

The first metric expresses the least observable mode of the system and provides a worst-case observability measure. The second, describes the range of observability in the system. Large values of ξ or λ are correlated with higher unobservability. Ideally, when dealing with the condition number λ, we are interested in values close to 1.

These two measures can be obtained and used in real time. In fact, they have been previously used to develop observability-based controllers [13, 41, 62] or adaptive filters, capable of recovering from unobservable situations [32, 35, 42]. Nonetheless, because of the limited computational power available within MAVs, it is necessary to investigate whether computing these metrics in real time is actually feasible. If not, one could explore the possibility of using other metrics, based on the intuitive cases that can be extracted from the non-null determinant condition (Section 5.7). Usually, the variables necessary for this calculations can be obtained almost directly from the on-board sensors, then it depends on how many calculations must be performed to get the metrics. As always, a thorough investigation is the key to understand where additional value can be obtained.

Having finished the discussion around relative localization frameworks, which are the basis for any attempt of successfully achieving collaboration between MAVs, it is time to discuss how to design and implement this collaboration. The next chapter introduces the concept of swarm robotics and discusses the different design methods that can make it reality.

3

Swarm Robotics

Swarm robotics is an approach to the coordination of a large number of robots, inspired by the emergent collective behaviours found in nature. The great marvel of nature swarms, and the reason why they serve as inspiration for swarm robotics researches, is their ability to emerge robust, scalable and flexible group behaviours. Despite the lack of individual abilities or special talents shown by single individuals, nature swarms exhibit highly advanced behaviours when working together. For instance, if one thinks about termites colonies, it is hard to realise how such small and limited creatures can successfully build huge complex mounds. A single termite would never be able to accomplish such thing, yet there are impressive mounds spread worldwide that show the colonies ability to work together towards a greater goal.

By taking inspiration from biological swarms, swarm robotics is interested in applying its principles to solve problems with groups of simple low cost robots with limited resources. By having each individual follow simple local rules, without any communication with a centralized entity, a desired collective and complex behaviour should emerge. As a consequence of the methods' decentralization, solutions are scalable to different group sizes, robust to single robot failures and flexible to different tasks and environments.

Section 3.1 discusses the different design methods that can be followed to develop such collective behaviours within a group of robots. In the end, Section 3.2 briefly elaborates on the leader-following problem.

3.1. Design Methods

Design is the stage during which the control system is developed. Unlike other scientific and engineering methods, there is no rule of thumb to the development of decentralized collective complex behaviours, since it is not a trivial problem. Nonetheless, these design methods can be separated into two categories [6]: behaviour-based design and automatic design.

Behaviour-based design is the traditional approach to develop swarm robotics systems. First, the designer tries to decompose the desired global behaviour into simpler individual behaviours and local interactions within the overall system. Then, having determined what local control rules originate the global behaviour, the designer tries to implement them in each robot and, through an iterative process, adjusts them until the desired final collective behaviour or task is accomplished. Depending on the complexity of the problem, and the objectives the designer wants to achieve, this design technique may be preferential since there are mathematical models for a large variety of specific simple behaviours. Nonetheless, the complex nonlinear connections between most local control rules and the global behaviour, makes this design technique a difficult job and heavily dependent on the designer's beliefs and ideas, since the implemented control rules directly reflect them.

An alternative to this design method is to focus on the robotic system as a whole, instead of trying to decompose it in smaller rules and trying to implement them. Automatic design methods trust robots to learn what local rules and behaviours lead to the desired collective behaviour or task. Thereby, exploiting these methods reduces the influence of the designer in the final solution. This assures a better exploration of the solution space and promotes the emergence of more complex behaviours that the designer could never think

of or could never implement given its nonlinearity nature. Automatic design methods can be further divided into two categories: evolutionary robotics (ER) and multi-robot reinforcement learning (RL).

A thorough review of behaviour-based and automatic design methods, as well as examples of main contributions present in the literature, will follow.

3.1.1. Behaviour-based design methods

Behaviour-based design is the method where the designer manually develops the individual behaviour of the single robots, which then produces the collective behaviour of the swarm. Usually, it is a trial and error process, in which the individual control rules are adjusted and improved until the desired collective behaviour is achieved. In this section different state-of-the-art behaviour-based strategies for groups of robots will be reviewed.

One of the most known behaviour-based methods in swarm robotics is the Boid model [56]. It is a distributed behavioural model that draws inspiration from biological collective behaviours as the ones from a flock of birds, a herd of land animals, or a school of fish. In its basic framework, each individual, or boid, in the group follows three simple rules: collision avoidance, velocity matching and flock centering. The first one is straightforward, a boid should avoid collisions with neighbour flockmates. Additionally, it should try to match the velocity of its closest flockmates and attempt to stay close to them. Each simple rule is implemented locally and generates an acceleration input, with a certain magnitude and direction. Therefore, there are always three independent suggestions about the motion of the boid, since each rule generates an acceleration request. It was shown through experiments, that simply weight-averaging these requests would yield a final acceleration that resulted in a reasonable flight performance. Nonetheless, when critical situations appear, such as imminent collisions, conflicts must be solved. In these cases, each simple behaviour might provide opposite acceleration requests which would cancel each other out, and led the boid to continue in the same direction. To overcome this issue, a prioritized acceleration allocation strategy was implemented, so that when the accumulated magnitude of the accelerations surpasses a threshold value, only the prioritized acceleration is activated, since it is the one with more pressing demands. This model can also be augmented with other rules, for instance, for obstacle avoidance. Although it works well for simple simulated models, it is hard to guarantee that its implementation in real life agents will yield such results, not only because their dynamics are significantly more complex, but also because the way the final acceleration vector is calculated does not seem to be robust. Nonetheless, the Boid model still serves as inspiration as can be further improved using evolutionary robotics, as it will be discussed in the next section.

Coppola's work [8] develops an on-board collision avoidance behaviour for a group of MAVs, based on the Collision Cone (CC) framework [19, 76]. This framework uses the concept of collision cone as the set of all velocities of an agent that are predicted to lead to a collision with an obstacle somewhere in the future. In a team of MAVs, each single agent builds a collision cone for all the other agents and, if at a certain time an agent's velocity belongs to the set of velocities of any collision cone, a clockwise search is started in order to establish the desired escape velocity. The escape strategy is rather simple, but it poses the advantage of having a predictable escape route that prevents possible conflicts when two agents are trying to avoid each other, since they are both looking to continue their motion to their own right side. For instance, in CC based avoidance methods that choose the desired escape velocity by minimizing the change in velocity, two agents facing each other may end up continuing their motion towards each other by selecting the same escape route. The method was later validated through simulations and implemented in real flight experiments in order to evaluate the efficiency of the avoidance behaviour. Depending on the application, this kind of strategy may be the best to follow. If the available airspace is wide enough for the group of MAVs and we simply want a straightforward mechanism to avoid collisions, then it is probably more efficient to use this type of control strategy as a complement to the main controller that guides the MAVs. However, if there are airspace constraints or the goals of the mission require the MAVs to fly in proximity, it may be better to develop a more extensive controller that inherently avoids collision situations, which may be easier to do with automatic methods.

In [70], van der Helm implements an on-board leader-follower behaviour control strategy, for a system with several flight constrains that affect its observability, as already discussed in Section 2.2. Since a lot of research over leader-follower flight relies on the development of fixed geometrical formations [57, 68], that constraint could be an obstacle to the development of a leader-follower behaviour control strategy. To overcome this problem, van der Helm came up with a strategy where the followers fly a delayed version of the leader's trajectory, instead of maintaining a relative fixed position. By doing so, there will always be relative motion between the leader and each follower, unless a straight line trajectory is adopted by the

leader, guaranteeing the observability of the system. The leader-follower control law was designed on top of the stable inner loop control already running on board of the MAVs. In this way, the leader-follower behaviour is accomplished by having a control law that outputs velocity commands to the inner loop control, taking into account the follower's positional error regarding the delayed position of the leader. Flight experiments validated the control strategy, and the results show that leader-follower behaviour could be achieved for teams of two and three MAVs without largely compromising the observability of the system. Despite the good results, this strategy presents a major obstacle to scaling up the system: it requires too much memory to store the trajectory of the leader.

Thinking about this last example, probably there are other ways in which the followers can replicate the trajectory of the leader without the need to explicitly program them to do it in a pre-specified way. Moreover, by letting the MAVs figure out their environment and how it relates to their goal, they may be able to understand that the success of their performance is directly related to that of the filter, therefore, coming up with a following behaviour that inherently guarantees its observability. Possibly one that addresses situations that are not comprised in the ones already discussed. The next section introduces automatic design methods, namely evolutionary robotics and multi-robot reinforcement learning, that are known for promoting intelligence in robotic systems.

3.1.2. Automatic design methods

As said before, the use of automatic design methods enables the development of behaviours without the direct influence of the developer. Within this class, there are two sub-classes: evolutionary robotics and multi-robot reinforcement learning. In this section, both will be discussed, with particular focus on ER.

Evolutionary Robotics: Evolutionary robotics is a method that employs evolutionary computation techniques [14] to the design of autonomous multi robots' systems. In ER, robots are considered autonomous artificial organisms that evolve their own control system (and body configuration) in close interaction with the environment, without human intervention [50], in order to produce functional behaviours.

Evolutionary algorithms (EA) are the basis of any ER approach. Despite the existence of different variants of EA, they all share a regular framework, which will be further detailed.

Given a population of individuals within some environment that has limited resources, competition for those resources forces natural selection [14], which, consequently, causes an increase of the fitness of the population. Establishing a fitness function to be maximised/minimized, one can randomly create a set of candidate solutions, with different artificial genotypes, that can be evaluated. The best candidates are chosen from these fitness values, and will be used to seed the next generation, through recombination and/or mutation, producing a set of new candidates, called the offspring. The individuals from the offspring will be evaluated and, then, will compete for a place in the next generation. This process is iterated until a stopping condition is met - a good enough candidate is found, the population fitness increase steadies, or a computational limit is reached. The global scheme of an EA follows in Algorithm 1.

Algorithm 1 Evolutionary Algorithm Pseudocode

```
1: INITIALIZE population with random candidate solutions;
2: EVALUATE each candidate;
3: while TERMINATION CONDITION is not satisfied do
4:     1 SELECT parents;
5:     2 RECOMBINE pairs of parents;
6:     3 MUTATE the resulting offspring;
7:     4 EVALUATE new candidates;
8:     5 SELECT individuals for the next generation;
9: end
```

Translating this framework to ER, it all starts with the initialization of a random initial population of different artificial genotypes, each encoding the robot's control system (and morphology). Each of these genotypes is then decoded into a corresponding robot that acts on the environment (through simulation or real experiments) and the collective behaviour of the resulting swarm is evaluated based on a pre-defined fitness function. All population of artificial genotypes follows this procedure and, based on performance, a group of them will be selected to have the chance to generate copies of their genotype through genetic

variation operations (mutation and/or recombination), generating the offspring. Next, this offspring follows the same evaluation process that the parents' population did, and individuals from the parents' population and the offspring will be selected for a new generation. The new generation follows this cycle, creating one generation after another, until a stopping condition is met. In the end, "good" behaviours survive over different generations and "bad" ones are discarded, through an evolutionary cycle similar to the one driving natural evolution. A visual representation of an ER framework for a leader-follower task with a group of MAVs is given in Figure 3.1.

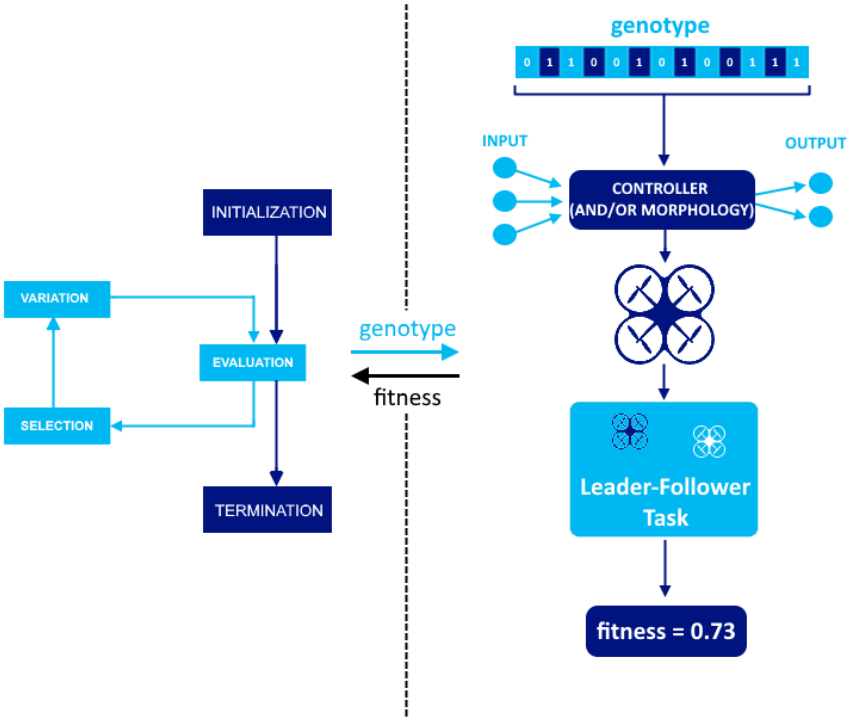


Figure 3.1: Evolutionary Robotics framework example. Inspired by an example provided in Eiben et al. [14]

There are two major drivers of evolutionary systems: variation operators (mutation and recombination) that assure diversity within the population and enhance a better exploration of the solution space; and selection (of parents and next generation) which boosts the rise of the population fitness by giving a better chance to fitter individuals genotype to be propagated. Moreover, the stochastic nature of these operators augments the search for global optimal solutions, rather than local. For instance, when recombination or mutation occurs, the choice of which pieces of the genotype will be recombined or mutated is random. Likewise, the selection process is not deterministic (unless the developer wants it to be), i.e., all individuals have an opportunity to become a parent or survive (fittest individuals will have better odds though). This stochastic essence is of the utmost importance, in order to guarantee diversity and a broad exploration of the solution space. At the same time, the fact that fittest individuals have a better chance of being propagated to next generations, assures the exploitation of the most promising solutions.

There is freedom in deciding which controller architecture to evolve. One can decide to use a fix structure and just optimize its parameters, or let the evolutionary process evolve both the architecture and the parameters. This choice will influence how the genotypes are defined. With the increase of on-board computation power available, feed-forward artificial neural networks (ANN) have become one of the state of the art control architectures represented by artificional genotypes [33, 59]. This can be explained by ANN's inherent ability to smoothly map highly nonlinear functions, making it suitable not only for low-level controllers, but also for high-level ones.

For instance, Izzo et al. [33] proposes a decentralized control architecture for satellite formation flying. The controllers are homogeneous across the group of satellites and do not rely on any individual characteristic of the satellites, which allows for the exchange of satellites at any time. The main goal is to achieve a triangular formation and, using an evolutionary robotics approach, two neural network controllers are evolved: one to tackle the translational kinematics, i.e., it outputs velocity commands; and

the other to deal with the attitude kinematics, outputting the desired angular velocity commands. After implementation in a simulated swarm of three satellites, the analysis of the results showed that the evolved ANN controllers converge to an asymmetrical triangular formation that ends up in one of two locations in space, using equilibrium points of the coupled system. Leveraging these equilibrium points to accomplish the desired task is not intuitive, therefore, a tradition behaviour-based approach would probably not render such results.

A similar task is evolved in Scheper and de Croon [59]. Moreover, a comparison study between evolved low-level and high-level controllers is presented. The goal is for a swarm of MAVs to arrange itself in an asymmetric triangular shape. In both cases a feed-forward ANN was employed and is homogeneous across all MAVs. In the low-level controller case, there is a need to not only achieve the desired formation, but also guarantee stable flight, therefore, the controller is responsible to directly command the angular velocity of the rotors of each MAV, by receiving information from both its own flight stability and the formation task. In opposition, the high-level controller works on top of an inner loop control architecture that already guarantees the stability of the system, therefore, only focusing on the formation task. In this case, the ANN is responsible to output velocity commands and only takes into account the distance to other swarm members. The low-level behaviour, despite good results in simulation, was not capable of showing such efficacy in real flight, with the MAVs quickly loosing stability. This disparity between simulation and real experiments is known as the reality gap, and can be one of the main challenges in ER approaches. However, the successful results of the evolved high-level controller, showing an emerging behaviour where the formation always converges to the same orientation in both simulation and real flight experiments, validate the hypothesis that abstraction of the inputs and outputs of the controller can be a valuable tool to bridge this gap and evolve robust behaviours.

A different approach that revisits the topic of self-organizing flocks is presented in Vásárhelyi et al. [72]. Instead of manually defining the flocking model parameters, they use evolutionary robotics to optimize the model. The goal is to ensure that large flocks of autonomous MAVs smoothly navigate in confined spaces without conflicts. The model combines several rules: local repulsion of nearby agents, velocity alignment of nearby agents and interaction with walls and obstacles. All this rules result in velocity vectors that are summed up, together with a self-propelling velocity term, into a final desired velocity that the flock agent will follow. The optimization of the 11 parameters that govern these rules represents the mission of the evolutionary process. In order to accomplish its goals, the evolutionary process uses a single-valued fitness function that is the product of five different sub-functions (valued from 0 to 1), each evaluating a specific previously defined coherence or avoidance metric. The evolved swarm behaviour was numerically tested and proved to remain stable under realistic conditions for significant flock sizes. Additionally, the coherence of the collective behaviour seemed undisturbed even near obstacles. Moreover, and contrary to most existing flocking models in the literature, they were able to validate the model on-board real MAVs, by performing an experiment with a swarm of 30. It would have been impossible to achieve the same results if the developer had decided to choose the parameters through trial and error. This conclusion adds evidence to the advantages of using evolutionary methods to develop significant collective behaviours.

Multi-Robot Reinforcement Learning: Reinforcement Learning is a goal oriented computational approach to learning from interaction, inspired by human learning. The usual RL problem is based on a simple interaction between two entities: an agent and the environment (see Figure 3.2).

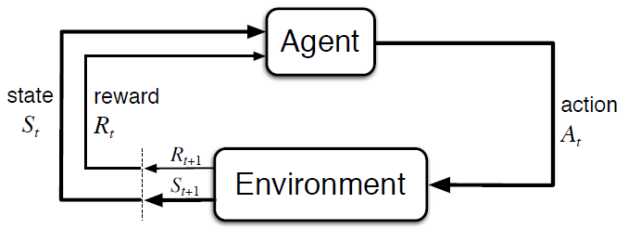


Figure 3.2: Reinforcement Learning framework [66]

Using the classical control nomenclature, the environment can be defined as the system to be controlled. However, in RL, the environment provides the agent with not only information about its current state, but

also a reward signal, indicating the positive, or negative, impact that a certain action A_t produced. On the other hand, the agent is defined as the controller. By receiving feedback from the environment, it can learn how to properly control the overall system, through the development of an optimal policy that maps states to actions.

In a more formal way, at each time step t, the agent and environment interact and the agent obtains the environment's state, $S_t \in S$ (with S being the set of all possible states). Then, taking S_t into account, it selects an action, $A_t \in A(S_t)$ (with $A(S_t)$ being the set of all actions available in state S_t). In the next time step, as a consequence of A_t, the agent gets into a new state, S_{t+1}, and receives a numerical reward, $R_{t+1} \in \mathbb{R}$.

The main difference between single and multi-robot systems in RL is present in the environment. When RL is applied to multi-robot systems, there are other robots that belong to the environment and are also adapting their behaviour, which makes it harder to rely on the traditional RL frameworks.

One of the main challenges for many real world applications is the fact that the state and action states are continuous and have high dimensionality, therefore, it is not possible to store the policy for each state.

Matarić [43] addresses the first issue in its work. To deal with the high dimensionality problem, the action space is reduced to single behaviours (introduced a priori into the system), and the state space is reduced by defining a set of conditions that are necessary and sufficient to trigger those behaviours. Then, robots learn how to appropriately map those conditions to the available behaviours implemented. The obvious question mark around this approach is that the available local behaviours that the robots can adopt are manually designed, which yields the same disadvantages already discussed in section 3.1.1.

Another solution to deal with that problem is to use function approximators that are able to generalize over the continuous state and action spaces. Shang and Sun [61] presents a distributed policy gradient reinforcement learning method using a feed-forward artificial neural network as the function approximator. The method is applied to a group of three ground robots, in a leader-follower task, for both discrete and continuous action spaces. In the discrete case, the output of the ANN is simply one of three actions, while in the continuous case it is the mean value and standard deviation of the translational and rotational velocities. The discrete case, where both linear and circular trajectories are tracked, shows good results, nonetheless, the controllers of the different agents are heterogeneous which hampers the scalability and flexibility of the method for larger groups. Moreover, the continuous space case, where the agents follow linear trajectories, does not provide good results, discouraging the application of the method for even more complex systems.

Furthermore, RL assumes the environment exhibits a Markov property, i.e., the environment's response at a time step $t+1$ can be predicted from the state and action representations at t [66]. Nonetheless, this assumption is far from being true in the leader-following task with only on-board measurements. There are solutions proposed in the literature that deal with this problem, but they will not be reviewed in this report.

Conclusions: Evolutionary robotics seems like the best option to apply in order to accomplish the goals of this thesis. Most work around high-level collaborative behaviours without manual design lies under this method, and some of its results are encouraging. Unlike RL, the higher dimension of the state space does not represent a problem, and its fitness based nature provides a great tool, if properly defined, to evaluate the overall performance of the system and explore multi-objective tasks.

3.2. Leader-Following Task

From the literature, it seems that there are two main strategies to accomplish a leader-following behaviour within a team of robots. The first, basically defines a fixed formation, or a fixed distance offset, which the followers try to keep over time, while the leader moves. The second, has the followers fly a different delayed version of the leader's trajectory. One can argue that the first strategy is more related to formation flight than to leader-following, however, it is less expensive than flying a delayed trajectory since it does not require the on-board storage of an array of positions. There are two immediate solutions that come to mind to overcome the latter concern. One could be to only store the position at each 5 seconds, for instance, and then the follower could perform polynomial regression of the trajectory and directly apply it on the controller. The other, could be to have the leader trajectory distributed over the followers, i.e., if each follower flies with an accumulated 5 seconds delay depending on its position, the leader would store its trajectory over the last 5 seconds, the first follower would store the leader's trajectory over the last 5 to 10 seconds, and so on.

Another possibility, that is somewhere between the two already mentioned, would be to have the followers follow the leader within a certain combination of radial ranges, i.e., there are no fixed relative positions, as in formation flight, but the followers need to be at a distance between, for example, 0.8 m and 1.6 m from the

leader, regardless of the direction. Depending on the applications it may be a good solution. Nonetheless, for an hypothetical case where the leader is the only agent able to perform obstacle avoidance, it would be ideal to keep the trajectories as similar as possible, as in the flight delay scenario.

Future work will necessarily come back to the discussion of this section.

4

Evolutionary Robotics

Inspired by the Darwinian principle of biological evolution, evolutionary robotics can reduce the designer's influence by trusting that the evolutionary process can drive the generation of better robotics systems. One of the problems when designing autonomous robots is that the desired robot behaviour is a dynamical process resulting from interactions between robot's body and mind, and its environment. Manual design only addresses part of this process, and is highly influenced by the designer's beliefs, which prevents the individuals from exploring all the solution space and derive possible more suitable behaviours the designer did not thought of. Moreover, this type of problems are commonly subject to a set of complex constraints and objectives, that is difficult for a designer to work with. ER can better cope with these challenges, first, by taking the designer out of the loop while including interactions between robot(s) and its environment and, second, by evaluating the performance with high level fitness functions that can explore good solutions in complex spaces. Undoubtedly, one of the great achievements of ER is demonstrating that sophisticated evolved robot control structures (e.g. neural networks) can produce functional behaviours in autonomous robots Nelson et al. [47] and replace the traditional manual design.

After the brief introduction in Section 3.1.2, this chapter frames the problem at hands into the ER framework, and further discusses its technical implementation details.

4.1. How does ER fit the problem statement?

As already mentioned, this thesis follows the state of the art work in Coppola et al. [8], van der Helm et al. [69]. It aims to develop an autonomous leader-follower controller architecture that maximizes observability, and is suitable to be implemented on-board small MAVs with limited resources and without relying on any external source. Therefore, the literature review of ER will focus mainly on the evolution of autonomous controller's solutions for MAVs and other ground robots.

There is freedom in deciding which controller architecture to explore. Nonetheless, most of the work combining ER with autonomous vehicles adopts feed-forward artificial neural networks (ANN) [17, 27, 31, 53, 59, 63]. Additionally, there is research that explores the use of behaviour-based architectures, such as behaviour trees [60, 67].

ANNs are inspired by the biological neural networks that drive humans' brain. There are two types of ANN: feed-forward and recurrent, depending on if the flow of data is only in one direction or is allowed to loop back, respectively. Feed-forward ANNs consist in a set of interconnected neurons where, in its simplest form, each neuron i can be mathematically described as:

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij}x_j - \theta_i\right) \tag{4.1}$$

being y_i the output of neuron i, f_i the activation function of the neuron, w_{ij} the connection weight between neuron j and i, x_j the j_{th} input to the neuron, and θ_i the bias of the neuron. A graphic representation is presented in figure 4.1. By connecting different neurons, within different layers, it is possible to generate an ANN of any complexity (e.g. figure 4.2).

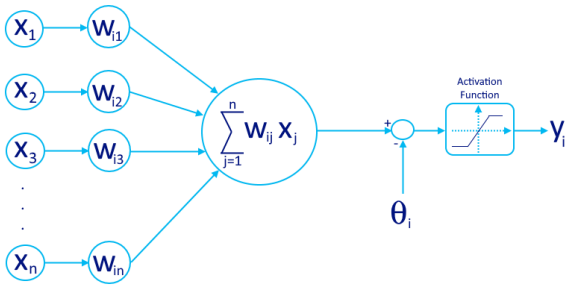


Figure 4.1: Single neuron connection

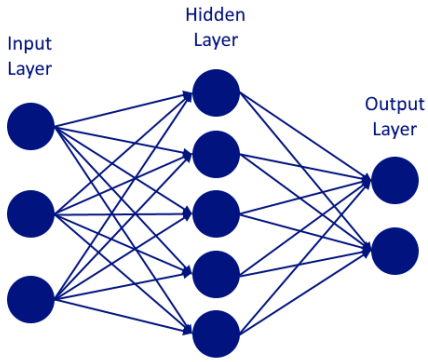


Figure 4.2: Feed-forward ANN with 3 input neurons, one hidden layer with 5 neurons, and 2 output neurons

With the increase of on-board computational power available, ANNs have become one of the state of the art control tools used by researchers working in the field of evolutionary robotics. This can be explained by its inherent ability to smoothly map highly nonlinear functions, making it suitable not only for low-level controllers, but also for high-level ones (where certain tasks or behaviours are evolved). For instance, the work from Scheper and de Croon [59], already discussed in Section 3.1.2, shows how evolutionary robotics coupled with neural controllers can be employed successfully to evolve higher-level behaviours within a team of MAVs.

Another good example is presented in Howard and Kendoul [31], in which a neural network is employed in a quadcopter landing task. Using optical flow theory, the authors designed a neural controller that is responsible for controlling the MAV's thrust, by having as inputs the desired and actual time-to-contact to the landing target (as well as their derivatives).

Furthermore, ANN's design flexibility enables the possibility of evolving both the connection weights and the network architecture, which provides evolution with better chances to exploit emerging behaviours and dynamics that researchers may have not envisioned. In fact, this inspired the development of a method called NeuroEvolution of Augmenting Topologies (NEAT) [65]. NEAT shows how it is possible to both optimize and complexify neural network solutions through evolution, which ultimately reduces the designer influence and enhances the resemblance with biological evolution. With the purpose of developing a proof-of-concept study for intelligent vehicles that autonomously navigate along motorways van Willigen et al. [71], NEAT was employed to evolve controllers for such task. In this particular case, the neural network controller was not responsible for developing any low-level behaviour, instead, its output was simply the action the vehicle should perform, which would activate a different controller. Despite being only a proof-of-concept study, the simulation results were encouraging, since the evolved controller outperformed the human drivers model used in traffic flow modelling.

A different control architecture that also appears in the literature coupled with evolutionary robotics is one that relies on the construction of Behaviour Trees (BT). BTs became renowned in the computer game

industry as a tool to model the behaviour of non-player characters. Later, due to their ability to create complex behaviours with simple sub-behaviours in a modular fashion, they started to be applied as a control framework for MAVs and other robots [60, 67].

BTs are rooted tree structures, composed of hierarchical nodes (root, control flow and execution nodes) that direct the flow of decision making. For each pair of connected nodes, the one from where the connection starts is called parent, and the other is called child. Every BT starts with a root node, which does not have any parent, only children. Control flow nodes, usually composite nodes, have one parent and at least one child. Finally, execution nodes, typically condition and action nodes, have one parent and no child. Figure 4.3 illustrates the structure of a BT, highlighting node types [58].

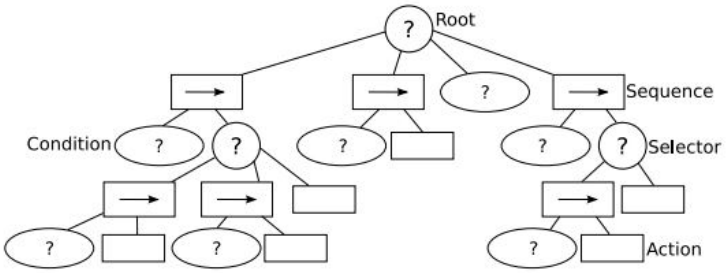


Figure 4.3: Graphical representation of a Behaviour Tree [58]

The flow of the decision making within BTs starts from the root, which sends ticks to its children (this tick acts like a trigger to the nodes, enabling their execution). When the execution of one node is allowed, it returns a status to its parent: running if its execution is still going, success if it achieved its goal, or failure if it failed to achieve its goal. Ticks are sent from the left to the right, i.e. not all children are triggered at the same time. This framework implies that not all nodes are evaluated in every tick, yielding a prioritized execution flow.

The basic control flow nodes are selectors or sequences. Considering ticks are sent from the left to the right, they run each of their children in turn, and when the task of its first child is finished, depending on the status returned, the control flow node decides whether to execute the next child or not. Selector nodes, basically, execute the first child that does not fail. This means that if one if its child returns success, it will also return success, or if all children return failure, it will return failure. As for sequence nodes, they will return failure when one of its children fails, or success if all of them return success. It is important that the execution nodes are organized in the right way (for selectors' children, the most important to the left, and for sequences', the correct sequence should be applied).

Because BTs don't have a fix depth, the more control flow nodes are created, the more sub-behaviours can be explored. A good example of this capability is presented in Scheper [58]. The authors used BTs and evolutionary learning to enable a 20g flapping wing MAV - the DelFly Explorer [10] - to perform a fly-through-window task using only its limited on-board resources. The results were promising, and the final genetically optimised BT used to control the MAV in this task is depicted in figure 4.4.

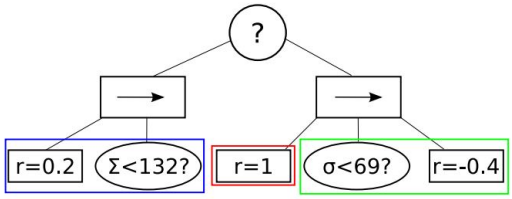


Figure 4.4: Graphical representation of a BT genetically optimised for the fly-through-window task. The different colours highlight the different stages of the flight, i.e. different sub-behaviours [58]

Additionally, BTs' modular structure facilitates an iterative development where new sub-trees can be added without affecting the already existent ones, which makes it possible to continuously improve a certain behaviour or ability to perform tasks. In fact, in Ogren [52], it is proposed to implement a BT framework as

the decision making system for a combat UAV. It is stated that BTs modularity and reusability are especially suitable for the design of UAV guidance and control systems, not only because the tree structure facilitates the transition between UAV flight states and the task of (re-)designing the system, but also because they increase the interpretability of the overall algorithm.

Despite the aforementioned advantages, which should not in any way be undervalued, it seems that BTs are a better option as an overall decision making entity, instead of acting directly as a controller. For instance, in a group of MAVs, each MAV could be equipped with a BT based entity that activates specific controllers depending on the task it intends to perform (collision avoidance, following a target, hovering, etc). On the other hand, because of their ability to map highly nonlinear functions, ANN seem like a better option to employ as the controller for tasks such as leader-follower.

4.1.1. Inputs and Outputs of the Controller

Since this thesis focus on the development of a higher-level task, leader-following, it makes sense that the controller is exclusively dedicated to this mission. Therefore, it will run on top of a inner-loop controller that guarantees stable flight, just as the example from [59]. The output of the high-level controller is taken as an input by the inner-loop one, and there are different possibilities to choose from. The most intuitive option is to output the velocity setpoints which are then given to the inner-loop controller as velocity commands.

As for the inputs, in order to evolve some following behaviour, the controller needs to have information regarding the trajectory to be followed. This can be done in different ways: using the relative position of the leader; using the relative position of the desired trajectory; or using an error measure regarding the desired relative position. Moreover, if observability is to be maximized, one can also incorporate observability measures, as discussed in Section 2.3, into the controller's input. This could possibly lead the controller to reactively steer towards more favourable observability situations, or proactively avoid unobservable scenarios. Additionally, variables like relative velocity could also be added to the input, as to provide information regarding the relative motion of the vehicles, which could help the controller anticipate changes in the relative positioning, and become more proactive towards changes of direction in the leader's trajectory. If the only spacial measure for the input is the relative position, than it will always be waiting for its changes to react, leading to more aggressive maneuvers.

4.2. Representation, Mutation and Recombination

The main observation from genetics is that each individual is a dual entity: its phenotypic characteristics are represented at a genotypic degree Eiben et al. [14]. Essentially, this means that all individual's characteristics are encoded in a set of genes. For this reason, the definition of the genetic representation is of the utmost importance, because it needs to be meaningful and interpretable in order to produce a valid solution.

There are different options to choose from and the decision depends, ultimately, of the application domain and the resources available. It is different to represent an ANN or a BT; additionally, it is different to evolve connection weights of an ANN or both connection weights and architecture design.

Furthermore, the genetic representation choice also affects the way recombination and mutation will be performed and its influence in the evolutionary process, therefore, one should take it into account when deciding which variation operators to use and how to apply them.

Despite the existence of numerous alternatives, in this review only Binary Representation and Real-Valued Representation will be explored, since these are the most used in the literature, especially when dealing with the controller architectures already introduced.

4.2.1. Binary Representation

One of the earliest and most frequently used representations is the binary one, where the genotype consists of a string of binary digits. This form of representation is more often employed with discrete distributions of values, rather than continuous.

Mutation

The most common mutation operator for binary representations, considers each bit separately and allows it to flip value with a a probability (usually small) p_m. This guarantees the stochastic nature of this operation, since the number of changes depends on the random numbers that are drawn. Figure 4.5 illustrates an example of bitwise mutation.

Figure 4.5: Bitwise mutation for binary representations ([14])

Recombination

The main methods of recombination for binary representations start from two parents that originate two children. The three most used methods (one-point, n-point, uniform crossover) are built on the same logic. Since one-point and uniform crossover can be derived from n-point crossover, it is sufficient to explain the later. Essentially, the chromosome is divided into $n+1$ segments of adjacent genes (using n crossover points), and the offspring is created by taking alternative segments from the parents. An example of n-point crossover with $n = 2$ is showed in figure 4.5.

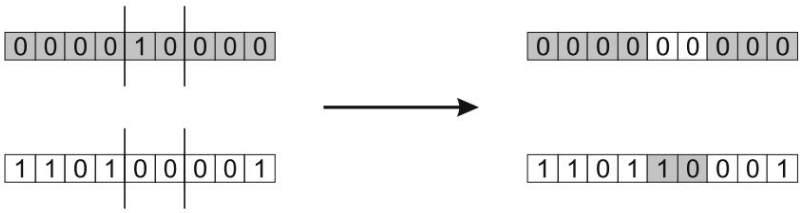


Figure 4.6: n-point crossover with $n = 2$ ([14])

4.2.2. Real-valued Representation

On the other hand, there are problems where one needs to represent genes as values from a continuous distribution. This is usually the case when morphological physical quantities like the different sizes of a certain component of the design. Additionally, one of the most used cases in the literature is when neural networks are employed as controllers and ER evolves the weights of the neural connections. It is important to underline that, although this representation is called real-valued, in fact it should be referred to as floating-point representation, since the precision of real values is limited in computational applications. In summary, an individual will be represented by a vector $< x_1, ..., x_k >$ with $x_i \in \mathbb{R}$.

Mutation

For real-valued representations, the most common mutation methods use the same position-wise mutation probability logic that binary representations use. Then, instead of flipping values, they are changed randomly within a limited domain. Depending on whether the mutation operator samples a random new value, or a random value to add to the current one, floating-point mutation methods can be divided into two categories: uniform or nonuniform mutations.

For uniform mutation methods, the values of the new vector are simply drawn uniformly randomly from the defined limited range. Regarding nonuniform mutation, these methods are arranged in order to, usually, induce smaller changes. Instead of simply sampling the new values from the chosen domain, the new value is obtained by adding the current value and a value drawn randomly from a Gaussian distribution with zero mean and a pre-defined standard deviation (constraining this resulting value to the limited domain). In this case, the value of the standard deviation regulates the extent of the perturbation, e.g. larger values will enhance larger changes. Another way to explore this type of mutation is to apply mutation to all genes (p_m = 1) and draw the standard deviation according to some distribution.

Additionally, there are nonuniform mutation methods, called self-adaptive, where the standard deviations which control the extent of the changes are also included in the genotype. In this way, standard deviations are not set by the designer, but rather undergo variation and selection, co-evolving with the solutions. This results in individuals being evaluated, not only based on their performance on the given problem, but also on their ability to create good offspring.

An alternative to the Gaussian distribution could be the Cauchy distribution, which has an higher probability of generating larger changes.

Recombination

There are three main options to condute recombination with real-valued representations. First, there is discrete recombination, similar to n-point crossover in binary representation recombination, where the parents' vectors are segmented and the two children will be the combination of both parents. Then, there is arithmetic recombination, which can have different implementations, but basically the value of each gene in the offspring vectors will lie between those of the parents ($z_i = \alpha x_i + (1-\alpha) y_i$ with $alpha \in [0,1]$. Finally, it is possible to use blend recombination, where the values of each gene of the children will be close to that of one of the parents, but may be larger or smaller.

4.3. Population Management

In the latest section, the variation operators that affect the population and drive the evolutionary process were addressed. These operators act on the population to create a offspring population that, typically, combines new properties with others inherited from the first one, producing novel candidate solutions to be evaluated. As a result, in the end of each iteration, there are two populations: the initial one and its offspring. However, replicating what happens in nature, not all individuals can survive and be part of the next generation. Hence, it is important to define how the population is managed during the evolutionary cycle, since it will affect when and how selection takes place and, therefore, the outcome of the evolutionary process.

There are two population management models: the generational and the steady-state. In the generational model, an initial population of μ individuals is created, from which parents are selected based on relative fitness. Then, λ offspring are created, by applying the previous introduced variation operators, and evaluated. In the end of each cycle, the complete population is replaced by μ individuals of the offspring (again, based on relative fitness), which will be the next generation. In this model, the number of created offspring, λ can be the same or larger than the size of the population μ. In the steady-state model, the first stage is similar to the generational model, but then the population is not completely changed at once. Instead, $\eta (< \mu)$ old individuals are replaced by η new ones, from the offspring.

The population management models discussed above rely on fitness based selection to decide which solutions may generate copies of their genotype, and which solutions will survive to the next generation. This was already expected, since it follows the general evolutionary framework earlier presented in algorithm 1. The fitness based selection concept can be explained in two different steps: fitness evaluation of every candidate solution and selection of solutions considering this evaluation. The first step will be discussed in the following section, and the second one will be approached in the next after.

4.4. Fitness Functions and Multi-Objective Optimization

Through this chapter, the need to evaluate the different genotypes of the population, in order to compare their suitability for the problem, has been mentioned. This evaluation is performed with the help of fitness functions. A fitness function is, basically, a tool to evaluate the ability of the different artificial genotypes, when decoded into a simulated or real entity, to perform a certain task, or behaviour, one desires to produce. Essentially, it represents the requirements to which the population should adapt to, by being the source of selection. If one thinks about the engineering perspective, it is a representation of the task to be solved, however, in a more technical point of view, it is a function that assigns a quality measure to genotypes [14].

Furthermore, it was highlighted that when designing control systems with great complexity, traditional methods have trouble dealing with the myriad of constraints and unpredictable dynamics between robot and environment. Whilst the choice of a proper fitness function poses a huge advantage for ER techniques, it is not a straightforward decision. In fact, the ability to successfully evolve intelligent robots is fundamentally influenced by the choice of a suitable fitness function that is capable of promoting good behaviours, without the need of specifying the low-level implementation details.

Additionally, this complexity means ER deals mostly with multi-objective problems, which translates into the existence of different fitness functions that need to be taken into account when comparing the fitness of each candidate solution. The difficulty here is that this type of problems, typically, have a set of optimal solutions Deb et al. [12], rather than a single one. Therefore, it is also imperative to make a good choice about the method used to compare the relative fitness of the population.

Because of the undeniable importance of fitness in the success of ER strategies, and the fact that designing the right fitness functions, as well as finding the most optimal solution, are not trivial problems, these are

highly investigated topics within ER that will be further investigated.

4.4.1. Fitness Functions Classes

Taking into consideration the level of influence the designer has in the robot's behaviour, by incorporating pre-conceived ideas or a priori knowledge related with the interaction robot-environment, it is possible to define classes for fitness functions [47] - table 4.1.

Fitness Function Class	A priori knowledge included
Training data fitness functions (for use with training data sets)	Major
Behavioural fitness functions	High
Functional incremental fitness functions	Moderate-high
Tailored fitness functions	Moderate
Environmental incremental fitness functions	Moderate
Competitive and co-competitive selection	Low
Aggregate fitness functions	Minor

Table 4.1: Fitness Function Classes [47]

These different types of fitness functions influence the result of the evolutionary cycle in different ways, since they have different evaluation criteria. An explanation concerning the potential different fitness functions that can be used, as well as a review of examples of their application in the literature will follow, in order to provide a better understanding on how different fitness functions may guide evolution towards different results.

Behavioural and Functional Incremental Fitness Functions

Behavioural fitness functions are task-specific functions that measure the way the robot acts, without accounting for the end result, i.e. the goal(s) of the task. These functions are, usually, a combination of different sub-functions that measure simple behavioural features which the robot should be able to evolve.

For ground wheeled robots, these kind of fitness functions have been widely used in order to develop controllers capable of, for instance, obstacle avoidance. Banzhaf et al. [4], Floreano and Mondada [20], Lund and Miglino [40], Matellán et al. [44], Nordin et al. [51]. For this particular task, authors, usually, opt to base the fitness of the robots on local motor behaviours (e.g. wheels velocity) and sensors' feedback (e.g. activation of sensors), instead of measuring the task completion itself. Banzhaf et al. [4] evolved three other behaviours, by combining different sub-functions, each accounting for a different behaviour.

In Jakobi [34], obstacle avoidance in legged robots was evolved. The behavioural fitness function subdivides the task in four smaller ones, which are mutual exclusive between them. When one smaller task is on-going, a corresponding sub-function will evaluate the fitness of the robot at that given task and, in the end, the overall function will be a combination of these four sub-functions. Once gain, local motor behaviours and sensors' feedback was the basis for fitness assessment.

One of the main difficulties in ER is that, if a task it too complex, it may happen that all controller's candidate solutions, right from the initialization moment, are restrained to a sub-minimal optimal level. Thus, the fitness function will fail to drive the selective pressure towards a promising direction. Functional incremental fitness functions can overcome this problem by augmenting the fitness function through the evolutionary process, until a final desired controller is obtained. First, it starts by evolving a simple competence and, once it is evolved, the fitness function will change or be augmented to account for a new and more complex behaviour. Functional incremental functions are, usually, comprised of behavioural ones.

There are several applications of this type of function in the literature [5, 18, 26, 38, 54], which demonstrate the viability of such approach. In Harvey et al. [26], ER was used to develop a robot's neural network controller, equipped with a camera, capable of moving towards a triangular target placed on a white wall while avoiding a rectangular one. Their experiment used a three stage functional incremental fitness function. The first sub-function applied consisted on the sum of the distance between the robot and the wall opposite to the target. When fitness converged, a new sub-function was used and evolution continued with a new population derived from the fittest individual of the population evolved with the first sub-function. The second sub-function consisted on the sum of the distance to the target over the course of the overall try. Similarly, a third fitness function was applied, measuring the distance to both the triangular and rectangular target.

The main characteristic, and often the most criticized, of the behavioural functions previously described is that they are built to select behaviours the designer assumes will provide good solutions. Going back to the work of Floreano and Mondada [20], where a neural network controller was evolved for a robot to navigate while avoiding obstacles, it is possible to understand what behavioural features the designer believed would provide good avoidance: higher velocities ($mean(v_l, v_r)$), engaging in forward/backward motion ($1 - \sqrt{|v_l - v_r|}$) and turning when facing forward obstructions ($1 - s_{ir}$). In this case, selection will drive the population towards those behaviours, beliefs the designer is convinced can produce obstacle avoidance ability, instead of evolving the robots to actually avoid obstacles intrinsically, which should be the goal. Moreover, though functional incremental fitness functions may overcome some limitations of purely behavioural ones, they do it at the expense of increasing the designer's influence since he/she decides what the search path on the robot's controller solution space will be.

Tailored fitness functions

One way to reduce the designer's influence is by combining behaviour terms with aggregate ones, using tailored fitness functions. Aggregate terms evaluate if a certain task was accomplished, regardless of the particular behaviours employed to perform it. Take the example of a drone capable of charging its own battery with solar energy, i.e. in the case of low battery it should be able to fly towards an illuminated area (assume the drone is equipped with the necessary sensors to measure light exposure). A potential fitness function could contain an aggregate term that rewards the controller if the drone arrives to a light source, independently of the sensor-actuator mapping behaviours that were used to perform the task. Additionally, the function could contain a second behavioural term minimizing the velocity of the drone during the task. This second term, unlike the first, represents an assumption about a behaviour that could benefit the drone to accomplish its final goal, which is not necessarily the case.

Another option in tailored fitness functions is to use a term that measures partial completion of the task, instead of only measuring the total completion, like aggregate terms do. In the latter example, imagine that instead of the reward for reaching the light source, a new term would assign a scaled value depending on how close the drone flew to the light source. Contrary to the aggregate term, the new one involves some a priori knowledge about the environment, because it implies that being closer to the final goal is inherently better, which might not be true in a environment with constraining conditions, like obstacles or wind.

Aggregate Fitness Functions

Finally, there is a type of fitness functions that can significantly reduce the human bias transmitted to the robot: aggregate fitness functions. These, only look at the high level success or failure of a robot trying to complete a task, without judging how it was done.

In Scheper and de Croon [59], when trying to evolve a controller so that MAVs are able to reproduce a desired formation regardless of the initial conditions, the authors opted to use aggregate functions in order to evaluate the population. One of the functions evaluated if the correct formation was completed, by analysing the distances between MAVs, and other evaluated if the flight was stable by checking the time of the simulation (if a collision occurred, the simulation would be interrupted).

One of the problems that might occur when using this type of functions can be the lack of competence shown by all the solutions in the initial population. In this case, when the initialization process only produces poor candidate solutions, the chances of any of them completing the task are indubitably small, which prevents the success of the evolutionary process.

Nonetheless, if one wants to generate solutions for more complex challenges, aggregate fitness functions seem to be the best option, since they enable the discovery and evolution of new complex behaviours. One might argue that behavioural functions are, basically, an optimization of human designed robots. On the contrary, aggregate fitness functions enable the evolution or learning of truly intelligent behaviour.

Because of the high expectations around the potential of aggregate fitness functions, and in order to tackle its possible handicap, it was proposed to combine them with tailored ones. In the beginning of the evolutionary cycle, a tailored fitness function could be used to evolve robots until the point where they can actually accomplish the proposed task. Then, an aggregate fitness function is applied in order to let intra-population competition do its job and, possibly, evolve solutions not envisioned by human designers, that could actually be more successful.

4.4.2. Multi-Objective Optimization

The examples discussed above, regardless of the type of fitness functions they use, have one thing in common: the problem requires more than one goal - fitness function - to optimize. This poses a challenge because instead of having only one optimal solution, the problem will have a set of optimal solutions, also known as Pareto-optimal solutions.

One may think that the simplest way to deal with it would be to combine all fitness functions into a single one. However, this obliges the designer to define a relative importance between the different fitness functions - do they have different weights, how much different? Moreover, different fitness functions may have different ranges, one may range from 0 to 1, and another from 0 to 100, which makes the first meaningless if both are combined equally. These two simple examples show how practically unfeasible it is to find the right balance in transforming a multi-objective problem into a single-objective one.

To overcome this, several multi-objective evolutionary algorithms have been proposed Deb et al. [12], Fonseca et al. [21], rey Horn et al. [55], Srinivas and Deb [64], Zitzler and Thiele [77]. One of the most used algorithms in the literature is called Fast Nondominated Sorting Genetic Algorithm (NSGA-II) Deb et al. [12]. Basically, it uses two different stages in order to produce the final fitness ranking of all possible solutions: fast nondominated sorting and crowding distance sorting.

The first one, nondominated sorting, divides the population into different nondomination levels. To achieve this, all solutions are compared against each other in order to determine if one dominates the other or not. Assuming there are M fitness functions, a solution x_1 is said to dominate a solution x_2 ($x_1 \preceq x_2$) if the following two conditions are met simultaneously Deb [11]:

- $f_j(x_1) \not\lhd f_j(x_2)$ for all $j = 1, 2, ..., M$.
- $f_{\bar{j}}(x_1) \lhd f_{\bar{j}}(x_2)$ for at least one $\bar{j} \in \{1, 2, ..., M\}$.

The operator $\lhd$, as in $f_1(x_1) \lhd f_1(x_2)$, means that x_1 is better than x_2 on the fitness function f_1 (works with both minimization or maximization functions). Using these dominance relations, it is possible to build different nondominated solution sets. Within a population P, the nondominated set of solutions P' is that whose solutions are not dominated by any other of the set P Deb [11]. With this in mind, and considering all the solution space S, fast nondominated sorting separates S into different non-dominated levels $F_1, F_2, ..., F_l$, with F_1 being the first nondominated set, also called Pareto-optimal front, therefore, the one with best fitness level.

After that, within every nondominated set, crowding distance sorting is applied in order to rank solutions based on their proximity with others. This sorting is done by computing, for every possible solution, the density of other solutions in its neighborhood. This means that solutions which have less performance similarities with others will have a better ranking, enhancing diversity elitism during evolution.

The overall NSGA-II scheme is illustrated in figure 4.7.The solution space S_t includes both the initial population P_t and its offspring O_t, at time t.

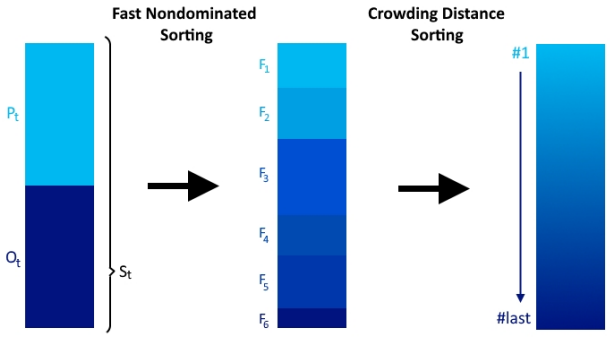


Figure 4.7: NSGA-II Deb et al. [12]

This works aims to solve both the leader-following task and the observability limitations of the relative localization filter, therefore, multi-objective optimization will have a major role in evolving solutions that are

successful for this scenario. In order for this to happen, appropriate fitness functions must be defined for promoting successful behaviours and discourage weak ones.

For leader-following, we should recall that we are interested in a collective behaviour. Therefore, if we are evolving an homogeneous controller for the entire group of MAVs, its fitness will be a combination of the overall performance of the group. For instance, if we are interested in formation following, the fitness of each MAV can be evaluated by simply computing in real time the error between the actual position of the MAV and its desired one (according to the formation), and then use that evaluation to get a final grade for each MAV. Then, one can average their performance and get the final fitness or, more interesting, get the product of the performances (forcing each individual performance to be between 0 and 1) as the final fitness. In this way we avoid the situation where some agents perform really well, and others perform terrible, since the worst case scenario will be penalizing the best one. On the other hand, if we are interested in the delayed scenario, the same logic can be used, only changing the real time reference to which the MAVs' positions are compared. Additionally, we should probably have a function that penalizes eventual collisions within MAVs, since they endanger any type of prospective collaboration.

Regarding observability, again, the observability measures introduced in Section 2.3 can be assessed in real-time for each filter. Then, the final fitness can be obtained by looking, for example, into the proportion of time that a filter has a estimation condition number λ lower/higher than a certain threshold. Or, instead of using that measure, one can opt to simply look at the more intuitive degenerative motions and see how often they occur. There are a lot of possibilities and some trial and error will probably take place.

4.5. Selection

As referred in Section 3.1.2, selection is one of the main drivers of evolution. It affects the overall diversity and fitness of the population by deciding, based exclusively on individual fitness, which individuals can, on the one hand, generate copies of their genotype and, on the other hand, survive to the next generation.

At this point, a review on the design of fitness functions already took place, as well as their optimization in order to get an appropriate ranking for all the population based on relative fitness. Thus, it is time to analyse how selection can use this, or other, fitness based ranking to push evolution in the direction of finding the best optimal solution. Literature is vast on different ways to undertake selection, therefore, a broad overview of the most relevant methods will take place.

4.5.1. Parent Selection

Parent selection decides which individuals of the population are allowed to become parents. These parents undergo variation (4.2) and create offspring. Usually, parent selection is a probabilistic process, since better ranked solutions - higher fitness - have better chances of becoming parents than those with lower quality. This probabilistic approach enhances improvements on the overall quality of the population over different generations, while assuring the search for an optimal solution is not too greedy and aims for a global optimum, rather than a local.

Ranking Probability and Roulette Wheel Selection

Ranking selection uses a fitness based ranking of the population to assign selection probabilities to every individual. The probability distribution can be linearly or exponentially decreasing, depending on how much selection pressure is desired. In a population of size μ, rankings are usually built from number 1 to μ. To help with formality, lets instead number the ranks so that it indicates how many worse solutions there are in the population (best one has rank $\mu-1$ and the worst has rank 0). Then, one can use the following equations to design a linear (equation 4.2) or exponential (equation 4.3) distribution:

$$P_{sel_lin}(i) = \frac{2-s}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}, 1 < s \leq 2, \tag{4.2}$$

where s is a parametrisation variable that can adjust the selection pressure;

$$P_{sel_exp}(i) = \frac{1-e^{-i}}{c}, \tag{4.3}$$

with c being a properly chosen normalisation factor that forces the sum of probabilities to be the unity.

After setting the likelihood, $P_{sel}(i)$, of each solution i, from the population, being selected for variation, selection needs to actually take place. Ideally, the number of any solution in the parents population would be

given by the multiplication of its selection probability with the size of the desired population. However, this is not feasible in practice because the parents population is finite and these numbers would be non-integers. Therefore, it is necessary to sample from the probability distribution.

To do so, the roulette wheel algorithm as been widely applied. It uses a cumulative probability distribution, $[a_1, a_2, ..., a_\mu]$ such that $a_i = \sum_1^i P_{sel}(i)$ and $a_\mu = 1$. Then, if the parents population has size λ, λ random values are drawn from $[0,1]$ and depending on which ith cumulative probability position they belong to, the corresponding individuals are selected.

Though it seems conceptually simple and effective, it has been noticed that roulette wheel does not provide an accurate sample of the distribution. Usually, because of the computational power available, populations are not significant large, therefore, it is more difficult to have an accurate representation of the probability distribution.

To overcome this misrepresentation, the Stochastic Universal Sampling (SUS) algorithm Baker [3] can be used. It uses the same cumulative probability distribution, however, instead of drawing λ random values, it draws one random value in the range $[0, \frac{1}{\lambda}]$. Then, the cumulative probability position of the drawn value will correspond to the first selected individual. After, the drawn value is increased by $\frac{1}{\lambda}$, resulting in a value from which its cumulative probability distribution position will select the second parent. This logic of increasing by an amount $\frac{1}{\lambda}$ and selecting the parent corresponding to that cumulative probability distribution position continues until λ selections are made. In this way, it is guaranteed that the number of parents corresponding to a certain individual i from the population is, at least, the integer part of $\lambda \times P_{sel}(i)$.

Tournament Selection

Tournament Selection is a simpler operator that does not require any calculations to be performed. Instead, it only needs an ordering relation, like a ranking, to compare any two individuals. For this reason, it is fast to implement. Basically, if one wishes to select λ parents, λ tournaments will occur. For each tournament, k individuals are chosen randomly and compared against each other, with the best being selected.

With this algorithm, the probability of a solution being selected depends on its rank in the population, the size of the tournament (the larger it is, the higher it is the probability that the tournament integrates better ranked individuals, increasing the selection pressure) and whether chosen individuals are replaced or not (if there is no replacement, the probability to select lower ranked individuals also decreases).

Additionally, instead of having a deterministic tournament where the best individual in the tournament is always chosen, one can also introduce parameter p ($p < 1$) that represents the probability of choosing the best ranked individual. Since the decrease of p will increase the chances of worst candidate to be chosen, it consequently decreases the selection pressure.

4.5.2. Survivor Selection

Survivor selection is similar to parent selection, since it also distinguishes individuals based on their quality. However, like showed in algorithm 1, it is applied later in the evolutionary cycle, as it is only used after the generation of the offspring population. It reflects the population management model of the evolutionary strategy in place, and because the size of the population usually does not change, it is necessary to decide which individuals from the current and offspring populations are allowed to belong to the next generation.

The algorithms presented for parent selection can still be used for survivor selection, nonetheless, other methods have been suggested and are widely present in the literature.

There are survivor selection algorithms that rely on the age of the individuals and do not take into account their fitness. If a generational population model is used, this approach makes sense. For instance, in the simple genetic algorithm [73] all individuals are selected as parents and undergo mutation (no recombination occurs), creating one offspring each ($\lambda = \mu$). In the end, all parents are replaced by their offspring, so that each individual only exists during one generation.

This kind of strategy can also be used with steady-state models, especially because they prevent convergence to local optimum and suit problems where the fitness functions become more complex during evolution. Having a population of size μ, λ offspring are produced to replace the oldest λ individuals of μ in the next generation, assuming $\lambda < \mu$. In this case it is important to have a good parent selection pressure, so that possible optimal solutions are not lost. Another way to avoid optimal solutions to get lost is through elitism. It works similar to the latter described algorithm, with an extra detail. A trace of the best solution in the population is kept and, in the case where it is one of the λ oldest solutions and no offspring has better quality, it is kept in the next generation and one of the offspring solutions is discarded. These examples rely on having an offspring smaller than the population ($\lambda < \mu$), but there can be the case where this does not

happen. In (μ, λ) Selection, $\lambda > \mu$ offspring are generated from a population of μ parents. For the next generation, all parents are discarded and the top μ children will form the next generation. Most ER problems are multimodal, therefore, this latest method is a promising option to search for a global optimum.

Round-Robin Tournament Selection

Another tournament-based strategy for selection is round-robin. Essentially, the population and the offspring are grouped together, and every individual is compared against q, randomly chosen, others. For every q tournament, an individual gets a "win" if it is better than its opponent. In the end, the μ individuals with more wins are selected for the next generation. This algorithm can also be used for parent selection, and likewise the tournaments presented in that subsection, the parameter q influences the chances of having less-fit solutions in the next generation.

III

Preliminary Experiments

5

Preliminary Results

In this chapter, the implementation of a preliminary evolutionary robotics approach to the leader-follower problem within a group of MAVs will take place. Moreover, its results will be presented and analysed.

To demonstrate the capabilities of this design method, two high-level controllers were evolved in simulation. First, Section 5.1 discusses the optimized task that was tackled. Then, Section 5.2 describes the kinematic model of the MAVs implemented in the evolutionary process as well as other simulation's parameters of interest. Section 5.3 introduces the control scheme used in the experiment, and is followed by Section 5.4 with implementation details over the evolutionary process. Finally, Section 5.5 presents the results of the optimization.

5.1. Leader-Follower Task

The proposed evolutionary robotics approach is tasked with developing an autonomous leader-follower behaviour for a group of three MAVs. As a starting point, it is desired that the group movement resembles that of a snake, i.e., there is one MAV leader following a random trajectory, a second MAV following the leader, and a third one following the second. Basically, all MAVs, except the last one, have both leading and following roles.

MAVs are able to acknowledge the relative position of other members of the group by making use of a range-based relative localization scheme implemented through an EKF. The relative localization estimates are heading independent, and the parameters of the filter were chosen in accordance to previous experiments held in [70], in order to approximate the real life scenario. The filter runs at 50 Hz.

Additionally, each MAV is considered to have a inner loop controller that ensures stable performance of the vehicle dynamics, therefore, the evolved controllers are only responsible for commanding the velocity setpoints of the MAVs. Then, this velocity is achieved by the inner loop controller. There is one main reasons that motivated this approach: instead of focusing in all the dynamics of a MAV, the controller is only responsible for optimizing the high level behaviour, which is the aimed goal. Also, this helps making the system more robust to the reality gap between the dynamics of the simulated system and those of the real one.

5.2. High Level Quadrotor Kinematic Model

The Parrot ARDrone 2 quadrotor MAV is the most popular choice within MAVLab for real-world experiments, therefore, the simulated dynamics were implemented based on those of this quadrotor. Since it is considered that each drone is already equipped with a stable inner loop controller, the model can be described as a second-order system with $\omega = 14.52$ and $\xi = 58.65$ [17]:

$$\dot{x} = \begin{bmatrix} -2\xi\omega & 0 & -\omega^2 & 0 & 0 & 0 \\ 0 & -2\xi\omega & 0 & -\omega^2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} \omega^2 u_x & 0 \\ 0 & \omega^2 u_y \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} v_{max} \tag{5.1}$$

With $x = [\ddot{x}, \ddot{y}, \dot{x}, \dot{y}, x, y]^T$; v_{max} set to 0.5 m/s, and u_x and u_y the outputs of the high-level controller. The model runs in simulation with a timestep of 0.001 s and integration is performed with Runge-Kutta integration methods.

5.3. Controller

A feed-forward artificial neural network (FFANN) with one hidden layer was selected to perform the desired control task. All neurons are activated by a tanh activation function, and the network weights were constrained to the range $[-2,2]$. Furthermore, bias nodes were added to the input and hidden layers, with the respecting weights constrained to the range $[-5,5]$.

The FFANN controller has 4 input neurons (plus the bias node), $r_x, r_y, \Delta v_x$ and Δv_y. The first two represent the relative distance in the x and y axis of the horizontal body frame, and are directly obtained from the relative localization filter - See Equation 1.6. The other two, Δv_x and Δv_y, represent the difference in velocity in those same axis, and can be computed from the velocity states of the filter, since each MAV has knowledge regarding its own velocity and that of another MAV from the group - See Equation 1.6.

As for the hidden layer, it was decided to have 15 neurons plus one bias node. There was no scientific argument behind this choice, it was purely instinctive. Lastly, it was already mentioned that the output of the evolved controller are the velocity commands for the inner loop structure. Therefore, the output layer consists of 2 output neurons, that yield u_x and u_y. These velocity setpoints are within the range $[-1,1]$ and are then translated into velocity commands by multiplying its value with v_max. Figure 5.1 illustrates the architecture of the FFANN controller.

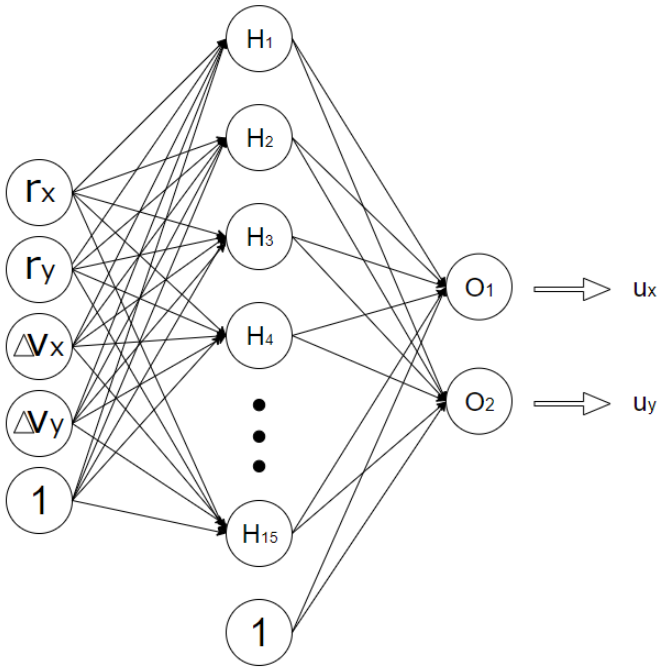


Figure 5.1: High-Level Velocity Controller Architecture used in this Preliminary Experiment

5.4. Evolutionary Algorithm

It was already mentioned that this research aims to evolve a high-level controller that promotes a leader-follower behaviour while enhancing observability. Additionally, the FFANN architecture of the controller was previously introduced. Therefore, the evolutionary process is responsible for optimizing the interconnecting neural weights of the controller. With this in mind, a population of 100 individuals was randomly (within the applicable range) generated. Each individual is represented by an artificial genotype that consists of a numerical array (real-valued representation). Two controllers are evolved together, one for the first MAV of the group, from here on referred to as the leader, and one for the other two MAVs, from here on referred to as the followers. Therefore, each individual's genotype encodes two controllers through a numerical array that concatenates the values of the interconnecting neural weights of both networks.

Once initialization takes place, all individual genotypes were simulated using the kinematic model introduced in Section 5.2 - it was used a time step of 0.001 s. The three MAVs are initialized hovering at the same height with a random x and y position within a 3.5 diameter circle, separated by a minimum distance between vehicles of 1 m. All trials take 60 seconds, and each genotype was evaluated over three trials, as to reduce any possible randomness effect. The performance was evaluated by applying a set of two fitness functions to be maximized:

$$f_1 = \frac{\sum_i^N \Phi_{MAV_i}}{N} \tag{5.2}$$

$$f_2 = \prod_i^N \Phi_{MAV_i} \tag{5.3}$$

With

$$\Phi_{MAV_i} = \frac{\int_{t=0}^{t_{max}} \phi_{range_i}(t)dt}{t_{max}} \tag{5.4}$$

And ϕ_{range} being defined at every time step according to the following rules:

$$\phi_{range_i}(t) = \begin{cases} 1 & if \quad 0.8 < range\_to\_leader < 1.6 \\ 0 & if \quad range\_to\_leader < 0.8 \\ \frac{1}{range\_to\_leader^2} & if \quad 1.6 < range\_to\_leader \end{cases} \tag{5.5}$$

Basically, ϕ_{range_i} evaluates each ith leader-follower distance (in this case there are three: the leader (l) following the random trajectory (rt), the first follower (1f) following the leader, and the second follower (2f) following the first one) with a value from 0 to 1. 0 means the drones are too close, therefore at risk of colliding, and 1 means the follower is at the perfect distance from its leader. In the end of each trial, those values are averaged over the full trial, and an objective metric over each following behaviour is obtained: $\Phi_{MAV_{l,rt}}, \Phi_{MAV_{1f,l}} and \Phi_{MAV_{2f,1f}}$. Then, the two fitness scores are simply the average and the product of these metrics, and are bounded to the range $[0,1]$. After the three trials, each genotype final fitness values are the averages over those trials.

The two fitness functions were chosen this way so that, on the one hand, evolution accounts for the average following behaviour of the three MAVs ($f1$); and, on the other hand, it forces all MAVs to demonstrate good following behaviour, since the worts case scenario will spoil the positive impact of the other two (f_2).

Once evaluated, the initial population produces an offspring. Mutation was the only evolutionary operator used, since it seems to be the best option when dealing with real-value genotype representations [59]. Each genotype had a 10% probability of being mutated. Nonuniform mutation was applied, by simply inducing a random perturbation (Gaussian) on the original value, while constraining the values to the applicable ranges already mentioned. Then, the offspring is evaluated in the same way its parenting population was. Once both the initial population and its offspring are evaluated, they are combined together and ranked through the NSGA-II (see 4.4.2). With the ranking established, selection is done by having a Round Robin Tournament of eight randomly selected individuals (see 4.5.2, where the most successful ones survive to the next generation. After that, the cycle reiterates by having the population deliver an offspring, which will be evaluated, and so on.

Figure 5.2 shows the population average performance over the different generations of the evolutionary process. It shows that fitness gradually increases until it approximately stabilizes after the $700th$ generation. The average fitness values of the top 10 solutions is 0.92 and 0.77.

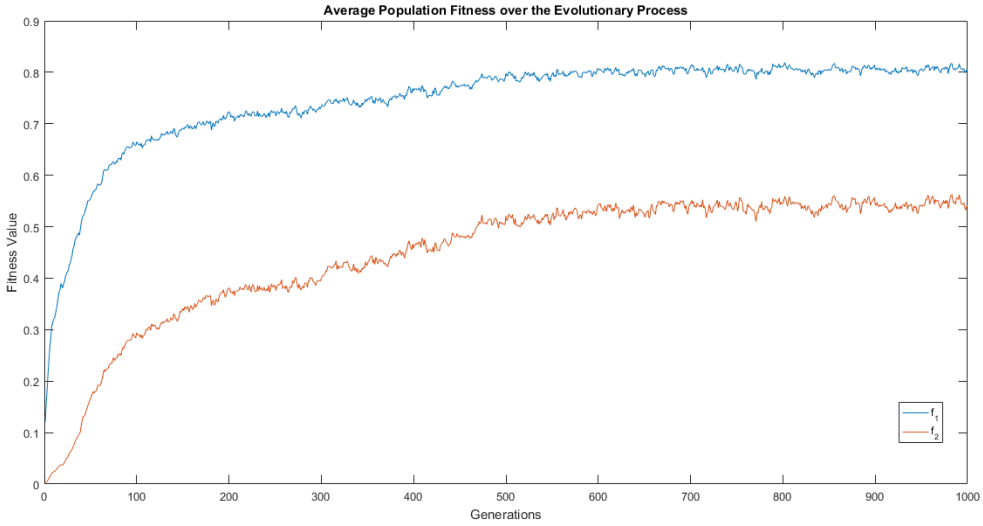


Figure 5.2: Average Population Fitness over the Evolutionary Process

5.5. Preliminary Results

In the end of the evolutionary cycle, the eight best ranked controllers were further validated during 100 trials of 90 seconds, with varying initial conditions. Then, the one that performed best was additionally simulated over 250 trials of 90 seconds. Some results that are obtained from the latter 250 trials will be presented next.

First, before analysing if the following behaviour was indeed achieved, let's take a look at the performance of the relative localization filters throughout the simulated trials, since it is what allows the MAVs to actually sense its leaders. Each MAV has its own filter which estimates the relative localization of its local leader. The probability distribution of the localization estimates errors are shown in Figure 5.3. In red, the median of the distributions is shown, and it can be seen that the relative localization estimates are quite reliable, with 50% of the time showing an error lower than approximately 0.25 for all MAVs.

In the simulations, the filter is initialized close to the true initial conditions and the measurements were randomly perturbed by sampling values from a normal distribution with mean= 0 and standard deviation= 0.2. Therefore, the main source of error for the localization estimates should be caused by the unobservability issues due to the lack of a heading reference.

Having verified that the relative localization estimates are trustworthy, let's move on to assessing if the evolved controllers accomplish the main goal of the experience: achieve leader-follower behaviour for a group of three simulated MAVs. Figure 5.4 shows the range values of each MAV with regard to its local leader. The MAV that acts as an leader shows a great ability to follow the random trajectory, as 87% 0f the time it is within a distance between 0.8 and 1.6. The first follower also shows a good ability to follow its leader, spending 71% of the time within the previously mentioned range. Finally, the second follower shows the worst performance, only being within the aforementioned range 51% of the time. Despite the difference in performance between the first and second follower, we can still say that the leader-follower behaviour was indeed evolved through the controllers. Nonetheless, we can also observe a gradual drift in performance that propagates backwards, similar to the effect that happens in traffic jams where the start and stop of a vehicle can generate long queues by the propagation of an increasing start and stop time over the queue.

This deterioration can also be seen in figure 5.5, which shows the distribution of the metric Φ_{MAV_i} for the different MAVs. Again we can notice that performance drifts backwards.

Figure 5.6 shows the distribution of the fitness scores over the different trials, and we can see that the results are not in accordance with the steady fitness values of the evolutionary cycle. This was already expected from the results presented in the two figures above. In fact, we can conclude that the solution does

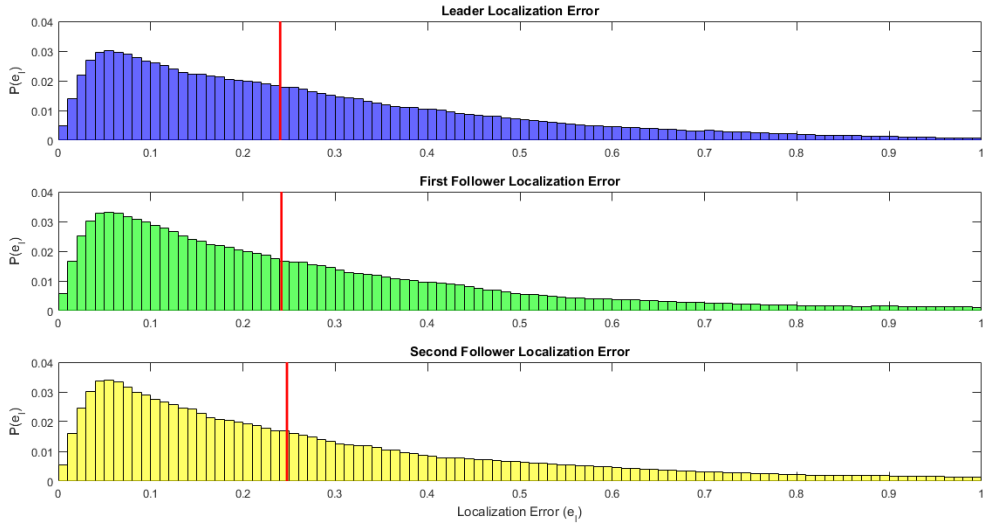


Figure 5.3: Localization Estimate Error Distribution for the different MAVs. Median in red.

not generalize well. One of the causes for this can be easily identified: during the evolutionary process all solutions are only evaluated in three trials, which evidently is not enough. Future work should constantly evaluate the surviving population and not only its offspring, in order to assure that the evolved solutions are not a product of beneficial initial conditions.

After evaluating the ability of the system to adopt a leader-follower behaviour, it is time to assess the observability of the system while doing so. As mentioned before, because of the lack of heading measurements, the states of the filter are only fully observable if the MAVs' movement respects certain constraints. This directly affects the relative localization estimates, which is the basic tool that enhances any group behaviour, therefore, it is desired that the MAVs are capable of avoiding unobservable situations. Using the concept of estimation condition number λ, introduced in Section 2.3, Figure 5.7 shows its distribution over the experiment. The smaller the value of λ, the more observable the system, however, without having a comparison study that serves as benchmark for the observability of the system, it is hard to draw conclusions on whether the evolved behaviour takes observability into account. Moreover, since this experiment did not include observability measures neither in the controller's input, or in the fitness functions, it is hard to expect that any observability-based behaviour was evolved. Future work should take this into account.

As a last note, it is worth mentioning that the biggest obstacle of this experiment was its computational load. For future work, the kinematic model should be replaced by a first order system, as in the work of Scheper and de Croon [59], and the EKF should be run at a lower frequency. By doing so, the use of a larger time step in simulation is possible since the time constant of the system is large. Moreover, this type of abstraction has provided great results in overcoming the gap from simulation to real flight, which is ultimately what we would like to achieve.

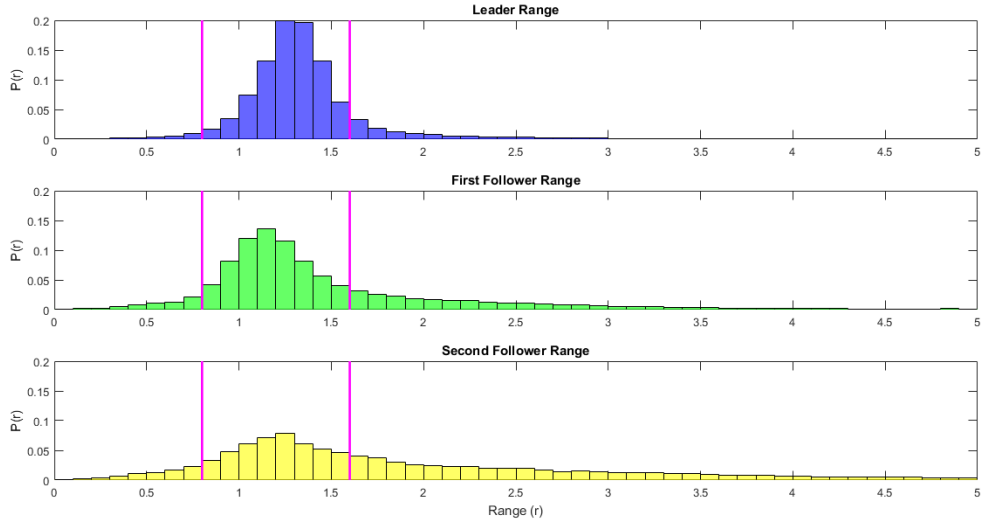


Figure 5.4: Range Distribution for the different MAVs. Vertical lines signal the ranges of 0.8 and 1.6.

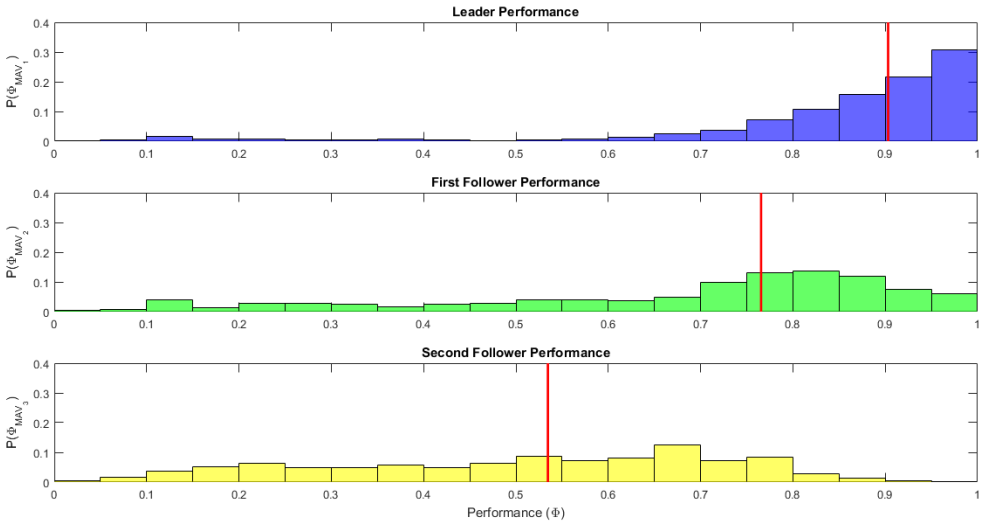


Figure 5.5: Performance Distribution for the different MAVs. Median in red.

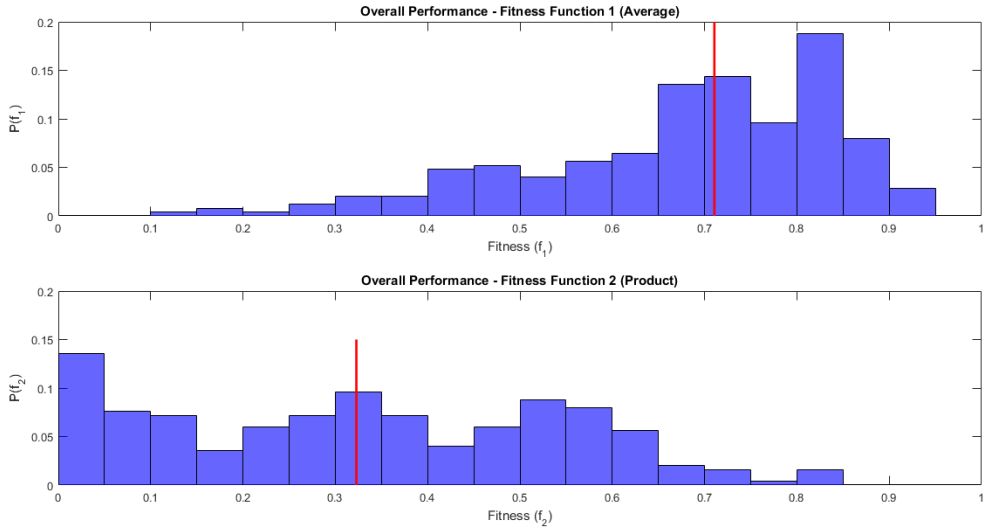


Figure 5.6: Fitness Distribution for the overall system. Median in red.

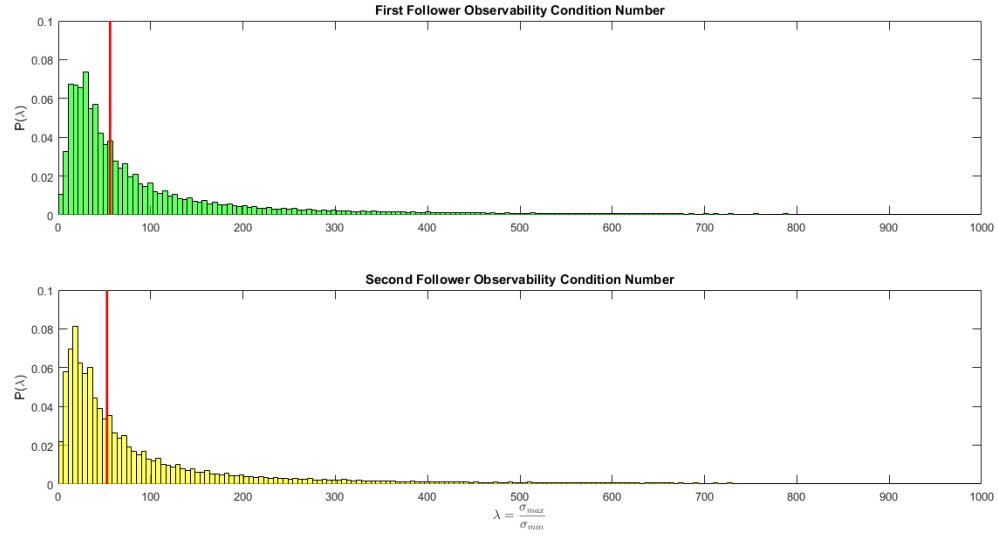


Figure 5.7: Condition Number Distribution. Median in red.

Bibliography

[1] Gianluca Antonelli, Andrea Caiti, Vincenzo Calabrò, and Stefano Chiaverini. Designing behaviors to improve observability for relative localization of auvs. In *2010 IEEE International Conference on Robotics and Automation*, pages 4270–4275. IEEE, 2010.

[2] Filippo Arrichiello, Gianluca Antonelli, Antonio Pedro Aguiar, and Antonio Pascoal. An observability metric for underwater vehicle localization using range measurements. *Sensors*, 13(12):16191–16215, 2013.

[3] James E Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms*, volume 206, pages 14–21, 1987.

[4] Wolfgang Banzhaf, Peter Nordin, and Markus Olmer. Generating adaptive behavior using function regression within genetic programming and a real robot. In *2nd International Conference on Genetic Programming*, pages 35–43, 1997.

[5] Gregory J Barlow, Leonardo S Mattos, Edward Grant, and Choong K Oh. Transference of evolved unmanned aerial vehicle controllers to a wheeled mobile robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2087–2092. IEEE, 2005.

[6] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[7] Luca Carlone, Miguel Kaouk Ng, Jingjing Du, Basilio Bona, and Marina Indri. Simultaneous localization and mapping using rao-blackwellized particle filters in multi robot systems. *Journal of Intelligent & Robotic Systems*, 63(2):283–307, 2011.

[8] Mario Coppola, Kimberly N McGuire, Kirk YW Scheper, and Guido CHE de Croon. On-board communication-based relative localization for collision avoidance in micro air vehicle teams. *Autonomous Robots*, pages 1–19, 2018.

[9] Alejandro Cornejo and Radhika Nagpal. Distributed range-based relative localization of robot swarms. In *Algorithmic Foundations of Robotics XI*, pages 91–107. Springer, 2015.

[10] GCHE De Croon, M Perçin, BDW Remes, R Ruijsink, and C De Wagter. *The DelFly*. Springer, 2016.

[11] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.

[12] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[13] Levi DeVries and Derek A Paley. Wake sensing and estimation for control of autonomous aircraft in formation flight. *Journal of Guidance, Control, and Dynamics*, 39(1):32–41, 2015.

[14] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[15] Wilfried Elmenreich. An introduction to sensor fusion. *Vienna University of Technology, Austria*, 502, 2002.

[16] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *2013 IEEE International Conference on Robotics and Automation*, pages 993–998. IEEE, 2013.

[17] T.A. Fijen. Persistent surveillance of a greenhouse: evolved neural network controllers for a swarm of uavs. Master's thesis, TU Delft, 2018.

[18] David Filliat, Jérôme Kodjabachian, Jean-Arcady Meyer, et al. Incremental evolution of neural controllers for navigation in a 6-legged robot. In *Proceedings of the Fourth International Symposium on Artificial Life and Robots*, pages 753–760. Oita University Press Oita, Japan, 1999.

[19] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[20] Dario Floreano and Francesco Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(3):396–407, 1996.

[21] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.

[22] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2012.

[23] Javier González, Jose-Luis Blanco, Cipriano Galindo, A Ortiz-de Galisteo, Juan-Antonio Fernández-Madrigal, Francisco Angel Moreno, and Jorge L Martínez. Mobile robot localization based on ultra-wide-band ranging: A particle filter approach. *Robotics and autonomous systems*, 57(5):496–507, 2009.

[24] Kexin Guo, Zhirong Qiu, Cunxiao Miao, Abdul Hanif Zaini, Chun-Lin Chen, Wei Meng, and Lihua Xie. Ultra-wideband-based localization for quadcopter navigation. *Unmanned Systems*, 4(01):23–34, 2016.

[25] Kexin Guo, Zhirong Qiu, Wei Meng, Lihua Xie, and Rodney Teo. Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in gps denied environments. *International Journal of Micro Air Vehicles*, 9(3):169–186, 2017.

[26] Inman Harvey, Philip Husbands, and David Cliff. *Seeing the light: Artificial evolution, real vision*. School of Cognitive and Computing Sciences, University of Sussex Falmer, 1994.

[27] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32, 2009.

[28] Robert Hermann and Arthur Krener. Nonlinear controllability and observability. *IEEE Transactions on automatic control*, 22(5):728–740, 1977.

[29] F Hidayat, BR Trilaksono, and H Hindersah. Distributed multi robot simultaneous localization and mapping with consensus particle filtering. In *Journal of Physics: Conference Series*, volume 801, page 012003. IOP Publishing, 2017.

[30] Andrew Howard, Maja J Mataric, and Gaurav S Sukhatme. Cooperative relative localization for mobile robot teams: An ego-centric approach. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES DEPT OF COMPUTER SCIENCE, 2003.

[31] David Howard and Farid Kendoul. Towards evolved time to contact neurocontrollers for quadcopters. In *Australasian Conference on Artificial Life and Computational Intelligence*, pages 336–347. Springer, 2016.

[32] Guoquan P Huang, Nikolas Trawny, Anastasios I Mourikis, and Stergios I Roumeliotis. Observability-based consistent ekf estimators for multi-robot cooperative localization. *Autonomous Robots*, 30(1): 99–122, 2011.

[33] Dario Izzo, Luís F Simões, and Guido CHE de Croon. An evolutionary robotics approach for the distributed control of satellite formations. *Evolutionary Intelligence*, 7(2):107–118, 2014.

[34] Nick Jakobi. Running across the reality gap: Octopod locomotion evolved in a minimal simulation. In *European Workshop on Evolutionary Robotics*, pages 39–58. Springer, 1998.

[35] Alexander Katriniok and Dirk Abel. Adaptive ekf-based vehicle state estimation with online assessment of local observability. *IEEE Trans. Contr. Sys. Techn.*, 24(4):1368–1381, 2016.

[36] Anton Kohlbacher, Jens Eliasson, Kevin Acres, Hoam Chung, and Jan Carlo Barca. A low cost omnidirectional relative localization sensor for swarm applications. In *Wf-IoT 2018*, 2018.

[37] Myungkyun Kwak and Jongwha Chong. A new double two-way ranging algorithm for ranging system. In *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on*, pages 470–473. IEEE, 2010.

[38] Wei-Po Lee, John Hallam, and Henrik H Lund. Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 501–506. IEEE, 1997.

[39] Yuanqing Lin, Paul Vernaza, Jihun Ham, and Daniel D Lee. Cooperative relative robot localization with audible acoustic sensing. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3764–3769. IEEE, 2005.

[40] Henrik Hautop Lund and Orazio Miglino. From simulated to real robots. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 362–365. IEEE, 1996.

[41] Filip Mandić, Nikola Mišković, and Zoran Vukić. Range–only navigation–maximizing system observability by using extremum seeking. *IFAC-PapersOnLine*, 48(16):101–106, 2015.

[42] Vicent Rodrigo Marco, Jens Kalkkuhl, and Jörg Raisch. Ekf for simultaneous vehicle motion estimation and imu bias calibration with observability-based adaptation. In *2018 Annual American Control Conference (ACC)*, pages 6309–6316. IEEE, 2018.

[43] Maja J Matarić. Reinforcement learning in the multi-robot domain. In *Robot colonies*, pages 73–83. Springer, 1997.

[44] Vicente Matellán, Camino Fernández, and JoséM Molina. Genetic learning of fuzzy reactive controllers. *Robotics and Autonomous Systems*, 25(1-2):33–41, 1998.

[45] Alessandro Milano, Attilio Priolo, Andrea Gasparri, Maurizio Di Rocco, and Giovanni Ulivi. An experimental validation of a low-cost indoor relative position localizing system for mobile robotic networks. In *Control & Automation (MED), 2011 19th Mediterranean Conference on*, pages 169–174. IEEE, 2011.

[46] Dries Neirynck, Eric Luk, and Michael McLaughlin. An alternative double-sided two-way ranging method. In *Positioning, Navigation and Communications (WPNC), 2016 13th Workshop on*, pages 1–4. IEEE, 2016.

[47] Andrew L Nelson, Gregory J Barlow, and Lefteris Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370, 2009.

[48] Thien Minh Nguyen, Abdul Hanif Zaini, Kexin Guo, and Lihua Xie. An ultra-wideband-based multi-uav localization system in gps-denied environments. In *Proc. Int. Micro Air Vehicle Conf. Competition*, pages 1–6, 2016.

[49] Thien-Minh Nguyen, Zhirong Qiu, Thien Hoang Nguyen, Muqing Cao, and Lihua Xie. Distance-based cooperative relative localization for leader-following control of mavs. *IEEE Robotics and Automation Letters*, 4(4):3641–3648, 2019.

[50] Stefano Nolfi, Dario Floreano, and Phil Husbands. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[51] Peter Nordin, Wolfgang Banzhaf, and Markus Brameier. Evolution of a world model for a miniature robot using genetic programming. *Robotics and Autonomous Systems*, 25(1-2):105–116, 1998.

[52] Petter Ogren. Increasing modularity of uav control systems using computer game behavior trees. In *Aiaa guidance, navigation, and control conference*, page 4458, 2012.

[53] Soo-Hun Oh and Jinyoung Suk. Evolutionary controller design for area search using multiple uavs with minimum altitude maneuver. *Journal of Mechanical Science and Technology*, 27(2):541–548, 2013.

[54] Frank Pasemann, Uli Steinmetz, Martin Hülse, and Bruno Lara. Evolving brain structures for robot control. In *International Work-Conference on Artificial Neural Networks*, pages 410–417. Springer, 2001.

[55] Je rey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*, volume 1, pages 82–87. Citeseer, 1994.

[56] Craig W Reynolds. *Flocks, herds and schools: A distributed behavioral model*, volume 21. ACM, 1987.

[57] Martin Saska, Jan Vakula, and Libor Přeućil. Swarms of micro aerial vehicles stabilized under a visual relative localization. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3570–3575. IEEE, 2014.

[58] Kirk YW Scheper. Behaviour trees for evolutionary robotics: Reducing the reality gap. Master's thesis, TU Delft, 2014.

[59] Kirk YW Scheper and Guido CHE de Croon. Abstraction, sensory-motor coordination, and the reality gap in evolutionary robotics. *Artificial life*, 23(2):124–141, 2017.

[60] Kirk YW Scheper, Sjoerd Tijmons, Cornelis C de Visser, and Guido CHE de Croon. Behavior trees for evolutionary robotics. *Artificial life*, 22(1):23–48, 2016.

[61] Wen Shang and Dong Sun. Distributed neural network-based policy gradient reinforcement learning for multi-robot formations. In *2008 International Conference on Information and Automation*, pages 113–118. IEEE, 2008.

[62] Rajnikant Sharma. Observability based control for cooperative localization. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 134–139. IEEE, 2014.

[63] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Evolving coordinated group behaviours through maximisation of mean mutual information. *Swarm Intelligence*, 2(2-4):73–95, 2008.

[64] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[65] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[66] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[67] T Szabo. Autonomous collision avoidance for swarms of mavs based solely on rssi measurements. Master's thesis, TU Delft, 2015.

[68] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, 33(1-2):143–156, 2012.

[69] Steven van der Helm, Kimberly N McGuire, Mario Coppola, and Guido CHE de Croon. On-board range-based relative localization for micro aerial vehicles in indoor leader-follower flight. *arXiv preprint arXiv:1805.07171*, 2018.

[70] Steven van der Helm, Mario Coppola, Kimberly N McGuire, and Guido CHE de Croon. On-board range-based relative localization for micro air vehicles in indoor leader–follower flight. *Autonomous Robots*, pages 1–27, 2019.

[71] Willem H van Willigen, Evert Haasdijk, and Leon JHM Kester. Evolving intelligent vehicle control using multi-objective neat. In *2013 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, pages 9–15. IEEE, 2013.

[72] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai, Tamás Nepusz, Agoston E Eiben, and Tamás Vicsek. Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20):eaat3536, 2018.

[73] Michael D Vose. *The simple genetic algorithm: foundations and theory*, volume 12. MIT press, 1999.

[74] Viktor Walter, Martin Saska, and Antonio Franchi. Fast mutual relative localization of uavs using ultraviolet led markers. In *2018 International Conference on Unmanned Aircraft Systems*, 2018.

[75] Viktor Walter, Nicolas Staub, Martin Saska, and Antonio Franchi. Mutual localization of uavs based on blinking ultraviolet markers and 3d time-position hough transform. In *14th IEEE International Conference on Automation Science and Engineering*, 2018.

[76] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. Generalized velocity obstacles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5573–5578. IEEE, 2009.

[77] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.