Evolving Behaviour Trees to Control a Swarm of Flapping-Wing Micro Aerial Vehicles for Greenhouse Exploration

Lukas Uptmoor

TU Delft

Evolving Behaviour Trees to Control a Swarm of Flapping-Wing Micro Aerial Vehicles for Greenhouse Exploration

Thesis report

by

Lukas Uptmoor

to obtain the degree of Master of Science
at Delft University of Technology
to be defended publicly on September 19, 2025 at 15:00

Thesis committee:	
Chair:	Prof. Dr. Guido de Croon
Supervisors:	Dr. Marija Popović
	ir. Stein Stroobants
External examiner:	Dr. Jordan Boyle
Place:	Faculty of Aerospace Engineering, TU Delft
Project Duration:	December, 2024 - September, 2025
Student number:	5011965

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

Completing this thesis has been one of the most demanding challenges of my academic journey. The process required sustained focus and perseverance through periods of difficulty and solitude. While it was far from easy, it has strengthened my ability to work independently and think critically.

A big relief for me occurred when Stein Stroobants joined the research project, giving me a daily mentor and partner who turned weeks of solitary computer work into fun, interactive times of experimentation. Without him, this work would have been far less extensive.

I would also like to thank my supervisor, Dr Marija Popović for her detailed feedback regarding academic rigour and experiment design. Most importantly, however, I am grateful for her inspiration and preceding research about agricultural robotics, which significantly shaped my outlook on a future career.

Furthermore, I would like to thank Professor Guido de Croon for the past two years of guidance in the TU Delft Swarming Lab, culminating in this thesis work. I am very grateful for his perspectives on bio-inspired intelligence, which sparked a renewed passion in me that has led me through my postgraduate studies.

At last, I want to say thank you for all the people who accompanied me on this journey, most notably my girlfriend Julia who supported me emotionally in these challenging times, and my housemate Sara who backed and listened to my ideas when few others did.

Lukas Uptmoor
Köln, August 2025

Contents

List of Figures

List of Tables

Introduction

Europe is the continent most severely suffering from demographic change: as society gets older, the workforce shrinks [1]. The agricultural sector is hit especially hard: from 1994 to 2019, agriculture's share of the workforce has fallen from 11% to 4% within the EU [2]. To keep up living standards, production per person needs to increase. The technical solution to this problem lies in automation. Recent advancements in robotics and artificial intelligence offer new opportunities for maintaining agricultural output with a decreasing labour force [3, 4]. In what is often referred to as the fourth industrial revolution, every available bit of data is harvested to make production processes more efficient, thereby increasing output.

Applied to agriculture, this could mean that crops are fitted with sensors that in real-time feed intelligent models of the plants, often called digital twins [5]. But placing these sensors on the growing plants and removing them when they are harvested demands a lot of manual work that – paradoxically – is best carried out by humans [5, 6]. Instead, sensing could take place on board of micro aerial vehicles (MAVs) that repeatedly inspect the crops, taking photos or measurements about volatile compounds that signal stress and necessary intervention [6, 7]. Greenhouses, due to their weatherproof environments and especially their high-value crops, are identified as potential starting grounds to develop this technology and assess feasibility for greater scales.

Moreover, the advent of commercially available flapping-wing MAVs offers inherently safe drone platforms that minimise the risk of harming the plants with fast-spinning rotors [8, 9, 10]. On the contrary, the oscillatory movements of these drones limit the use of certain sensing capabilities, such as optical flow [11]. Visual navigation or time-of-flight sensors could pose a feasible alternative [12].

Another promising technology that has gained a foothold lately is swarming: instead of relying on a single MAV with limited range and payload capacity, a team of MAVs can be equipped with different sensors and cover significantly more area in a given time [7, 13]. Recent breakthroughs in reciprocal localisation methods using ultra-wideband communication [14, 15, 16] have shown that swarms of tiny drones are fit for increasingly more applications, ranging from search-and-rescue [17] to gas-source localisation [18]. Furthermore, new methods for the automatic design of swarming behaviour have been developed. The most promising ones lie in the field of evolutionary robotics [9], which optimise reactive planning representations such as finite-state-machines [19] or behaviour trees [9, 20]. These methods circumvent the burden of explicit planning, which is typically too computationally expensive for micro aerial vehicles [21].

This work aims to find a suitable planning strategy that enables a swarm of flapping-wing MAVs to enhance automatic crop monitoring in greenhouses. Potential solutions need to address the limited computational resources and sensor availability of flapping-wing drones, e.g. by building on recent advancements in reciprocal localisation and visual navigation.

Part I

Literature Review

1

Flapping-Wing Drones

Flapping wings are arguably the most natural way to generate lift. They have evolved in nature by very distinct types of animals. The first ones were flying insects almost 350 million years ago [22], which are also considered the most sophisticated fliers today. Furthermore, there used to be flying reptiles (dinosaurs) which eventually evolved into flying birds. There are even flying mammals (bats) and essentially also flying fish (manta rays) that rely on flapping wings. This omnipresence of flapping-wing flight must have also inspired minds like Leonardo Da Vinci in 1505 for his theoretical ornithopter sketch [23]. The advent of mobile combustion engines in the late 19th century gave rise to a wide range of flapping-wing machines, which, however, were all unsuccessful. Although humanity eventually achieved controlled and sustained flight, both crewed and uncrewed, it were lifting gases, fixed wings and propellers that solved the problem, not flapping wings.

1.1. Advantages of Flapping Wings

That was until recently. The need for ever smaller drones had researchers resort again to flapping wings. Fixed wing approaches like aeroplanes show poor performance in miniature vehicles due to the very low Reynolds numbers, which can be more than 1000 times smaller for centimetre-scale vehicles compared to general aviation aircraft [24]. The Reynolds number describes the ratio of inertial to viscous forces in a flow with higher Reynolds numbers resulting in higher lift-to-drag ratios and thus higher aerodynamic efficiencies [25]. Moreover, flapping wings can simultaneously produce useful aerodynamic thrust and lift, whereas rotor-based drones must counteract their own torque-induced rotation, typically by using additional rotors or complex control schemes, thus requiring part of their thrust to be spent on stability rather than propulsion. This ability furthermore allows flapping-wing MAVs to achieve a smooth transition from hovering to fast forward flight, in which the wings generate extra lift, similar to fixed-wing aircraft [26]. These factors make flapping wings among the most efficient means of generating lift on tiny MAVs. On top of that, the relatively low flapping frequencies compared to multicopters result in a more low-pitch, less aggressive noise pattern. As flapping wings are often soft as well, they are inherently safe to operate around humans or delicate crops [26].

1.2. Recent Developments

The first flapping-wing MAV was the DelFly I, developed in 2005 by Jongerius et al. with a wingspan of 50 cm [28]. In 2008 the DelFly Micro reduced the wingspan to just 10 cm [10], being the first true insect-scale flapping-wing MAV. The same year saw the first take-off of the RoboBee with a wingspan of 3.5 cm [29]. However, unlike the DelFly Micro, the RoboBee did not carry its own power supply and had to be guided with a rail to maintain stability. Nevertheless, the RoboBee is a major achievement in smart composite microstructures and piezoelectric propulsion which will be useful in the future. Instead of reducing the wingspan further, the DelFly Explorer with a wingspan of 28 cm focussed on flight autonomy, which it achieved in 2013 by means of a custom 4g depth sensor [30]. Furthermore, 2018 saw the first flight of a DelFly that uses insect-inspired wing motions for steering, instead of relying on a conventional rudder-elevator tail. With a wingspan of now 33 cm, the tailless Delfly Nimble showed strong improvements on robustness and agility [27]. An overview is given in Figure 1.1. The DelFly Nimble also functioned as a

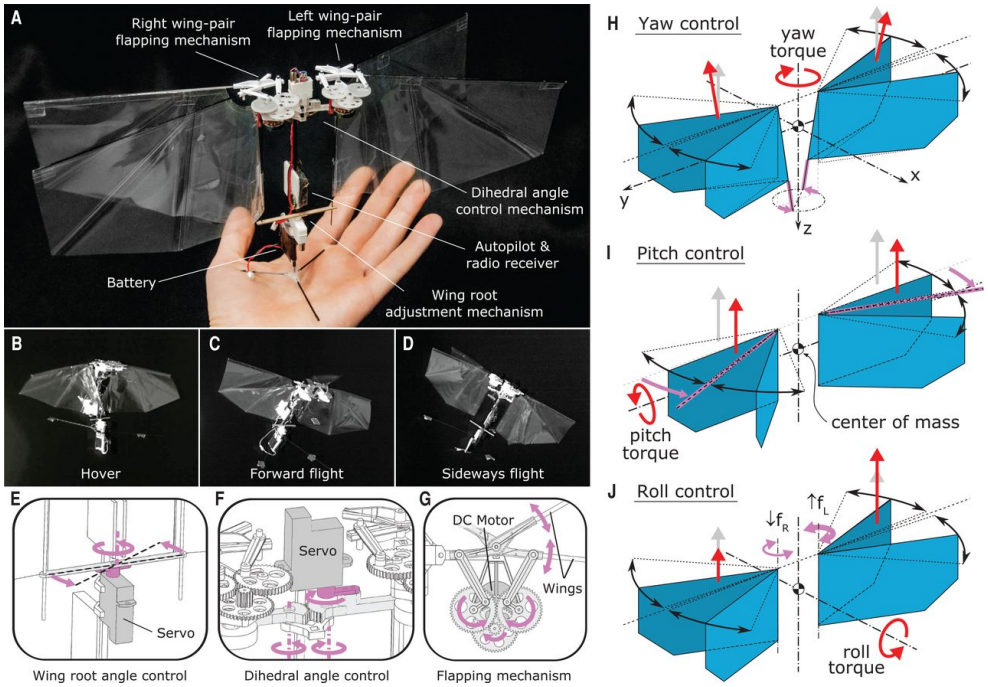


Figure 1.1: Overview and flight dynamics of the DelFly Nimble, which served as inspiration for the Flapper Nimble+, from [27].

blueprint for the commercialised version Flapper Nimble+, available since 2021 [8]. Combining all the achievements of the DelFly prototypes with efficient production and professional support, the Flapper Nimble+ is chosen as ideal candidate to study industrial implementation in higher numbers, like a swarm in a greenhouse.

The dynamics of the DelFly or Flapper Nimble has been extensively analysed [27, 31, 16]. Despite its outstanding agility, the model developed in this work will limit itself to slow hoverflight, as greenhouse monitoring is not a time-critical task and the navigation robustness is not yet strong enough for faster and thus riskier maneouvres. This is due to the Flapper's oscillatory nature, which imposes high noise onto measurements from the inertial measurement unit (IMU) and time of flight (ToF) sensors, limiting them to coarse navigation only. Moreover, it was found that optic flow based velocity estimation is difficult to use on the Flapper due to feature matching problems. Previous work has thus relied on a relationship between pitch/roll angle and the horizontal velocities that was found experimentally[16]. Albeit with reduced accuracy, it is possible to use these velocity estimates alongside ultra-wideband (UWB) ranging for reciprocal localisation among Flappers [16] or for position control with fixed UWB anchors [32]. More on this in Chapter 3.

2

Drones in Agriculture

The versatile challenges faced in agriculture offer a myriad of use cases for robots and especially drones. This is also reflected in the exponential growth of scientific articles about agricultural robots since 2015 [3, 33]. While many of these use cases just deal with automated versions of already existing machines, e.g. for tilling [34], harvesting [35] or pruning [36], aerial robots are really a novelty in the agricultural world. Characterised by their ability to simply fly over the crops, drones can help farmers by applying fertilisers and pesticides without the need for fruitless driving lanes. Another advantage is that with drones, pesticides and fertilisers can be applied exactly where they are needed, avoiding unnecessary spread of toxic chemicals on healthy plants and adversely high concentrations of minerals in the soil [33].

2.1. Aerial Information Gathering

On the contrary, fertilisers and pesticides are rather heavy and therefore require frequent refilling of the drones. Moreover, this task can be carried out with equivalent precision by ground-robots, for which weight is less critical. Most importantly, however, this site-specific treatment requires detailed information about the location and the severity of a plant's need. It is exactly this aerial information gathering (Figure 2.1) that will be the most valuable addition of using drones in agriculture. A bibliometric analysis conducted by Rejeb et al. in 2022 indicates that crop monitoring is by far the most researched topic related to drones in agriculture [33]. This is further supported by Canicattì and Vallone, which analysed drone research on vegetable farms and found that a mere 2% of articles were dedicated to drone spraying, while all 129 others were dedicated to aerial information gathering [38].

Conventional aerial information gathering is typically conducted with multispectral cameras, which are capable of measuring the three visible light bands red, green and blue, along with two infrared bands: red edge and near infrared. Using these bands, *vegetation indices* like the normalised difference vegetation index (NDVI) or the normalised difference water index (NDWI) can be calculated that inform the farmers about plant stress due to disease or drought [38]. However, these cameras tend to be rather heavy and expensive. That is why late research focussed on estimating vegetation indices like NDVI with artificial intelligence (AI) from visual camera images only [39, 40]. These methods allow the use of smaller drones, which are subject to weaker regulations due to their inherent safety.

While vegetation indices can detect plant stresses few days after their initial occurrence, Schuman and Baldwin found in 2016 that plants react to herbivore attacks by emitting *volatile bioactive compounds* that can be measured within hours after the attack began [41]. Geckeler et al. suggests using drones to deploy measuring stands in greenhouses that collect the volatile compounds over time [6]. These stands are then retrieved by drones again to be analysed in a laboratory. While this method may give very accurate information about the composition of the gases, it defies the main advantage of volatile sensing, namely the short latencies. Moreover, physical interaction is a complicated matter and carrying the measurement stands requires bigger drones. Instead, volatile compound sensors could be mounted on the drones directly. These sensors typically have a lower resolution and provide less information about the composition of the compounds, which needs to be addressed in data processing [6].

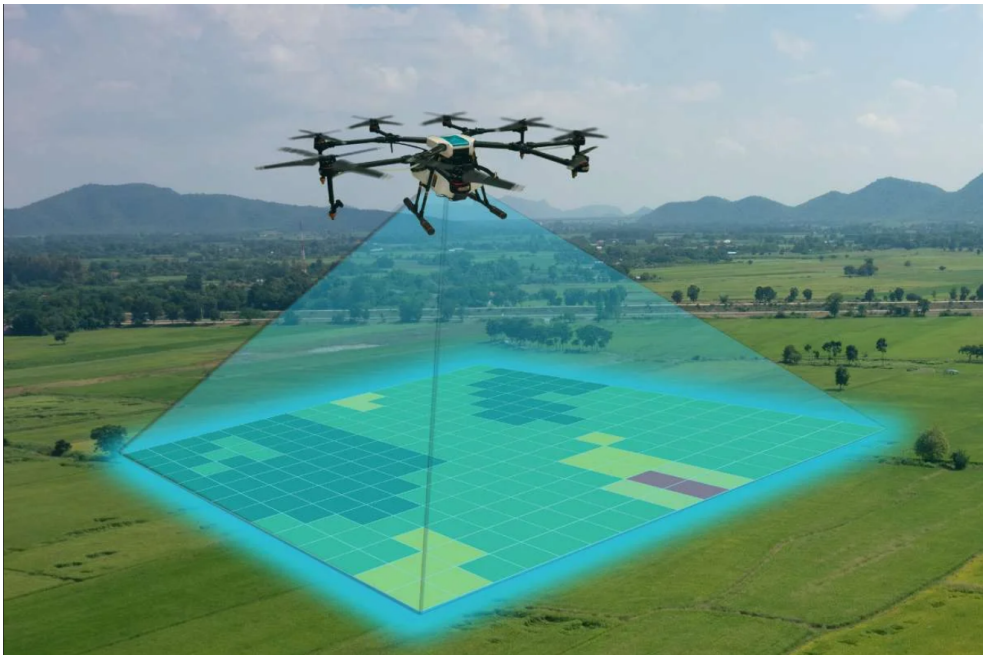


Figure 2.1: Visualisation of a drone gathering aerial information about a crop field, from [37].

A common problem faced both by outdoor and greenhouse drones is the relatively short flight duration due to limited battery capacity. A possible solution to that problem is using multiple drones simultaneously [7, 13], a practice often referred to as *swarming*. An experimental study conducted by Ju and Son in 2018 proved that using distributed control algorithms, a drone swarm can increase the field coverage per unit time without interference. However, their swarm consisted of only three drones, and problems related to the scalability of swarms have not been investigated.

2.2. Greenhouse Applications

Other applications related to greenhouses were studied by Aslan et al. in 2022. Research here focusses on providing sensor feedback to the *climate control system* by measuring temperature, as well as gas concentrations like CO2, methane and water vapor [7]. The strength of drones is their ability to reach any point in space at any time. Furthermore, studies investigate the use of drones for *pollination* [42, 43], in which drones first need to locate flowers and then mechanically pollinate them. Hiraguri et al., however, leave out information about how the drones manage to navigate in cluttered environments like tomato greenhouses [42].

Conventional crop monitoring based on vegetation indices is also possible in greenhouses, but it is not as common as outdoors [7]. This is because there is a range of challenges that drones face in indoor environments: a) lack of reliable satellite positioning, b) obstacles such as pillars and other instruments, c) constrained motion space due to relatively low ceiling and walls. Generally, greenhouses pose a more three-dimensional navigation and planning problem, involving flights between the crop rows and avoiding obstacles, while outdoor agricultural drones tend to fly in a more two-dimensional manner, multiple metres above the crops with motion taking place mostly horizontally.

The use cases surveyed in this section can be partitioned into two planning scenarios: 1) repetitive visits of certain places such as flowers, fruits or infested plant parts, and 2) area or volume coverage, e.g. to measure gas concentrations or monitor a certain area from above. Moreover, it was found that operation in greenhouses comes with a few challenges regarding the navigation, which will be addressed in the following chapter.

3

Resource-constrained Indoor Navigation

According to Kayton, navigation is the determination of the position, attitude and velocity of a moving vehicle [44]. Modern roboticists like Siegwart, Nourbakhsh, and Scaramuzza also include the perception of the environment and sometimes even path planning in their definitions [45]. While outdoors typically navigation satellites like GPS or Galileo are used along with inertial navigation systems (INS) to determine position and velocity, indoors this service is often not available with sufficient reliability. That is why indoor navigation, and GNSS-denied navigation generally, is a vibrant research topic.

3.1. Navigation Requirements

Based on the challenges that drones face in greenhouses, the following requirements for an indoor navigation system are identified:

1. The drones must be able to localise obstacles in their environment
2. The drones must be able to localise their peers in the swarm
3. The drones must be able to localise themselves in an absolute coordinate system
4. The navigation method must be viable with the limited power and compute on the Flapper Nimble+
5. The navigation method must be viable without external infrastructure
6. The navigation method must be compatible with the oscillatory nature of the Flapper Nimble+

The need for an ability to sense the absolute position in the greenhouse is motivated twofold: firstly, the drones need to assign observations about fruit ripeness or disease to a certain plant, such that in the next step, actions can be taken. This is possible without an explicit map, but not without a sense of absolute position. Secondly, modern drones still have quite a limited battery capacity and thus flight time. So before their battery reaches a critical level, the drones need to move to a safe space to land, where their batteries can be changed and operation can be resumed. This should ideally be a fixed space in the absolute coordinate system.

3.2. Simultaneous Localisation and Mapping

Rejeb et al. suggests applying simultaneous localisation and mapping (SLAM) to overcome the navigation bottleneck and enable the flight of drones in greenhouses. In SLAM, the environment is scanned while the drone moves, typically with a rotating LiDAR module or depth camera. These scans are then assembled together with what is called pose graph optimisation (PGO). Knowing the distance to the mapped surroundings, the drone can position itself with respect to its environment [46].

However, traditional SLAM requires the processing of gigabytes of data and a high computational power [47]. Moreover, LiDAR modules are heavy, which along with the high energy needs limits these methods to larger drones. But there is a lot of movement in the SLAM research area, with many modern approaches relying on visual data instead of LiDAR scans, known as visual SLAM (V-SLAM) [46]. Commonly used for

its robustness, ORB-SLAM2 [48] was developed by Mur-Artal and Tardós in 2017 as a continuation of ORB-SLAM [49] or in a wider sense Mono-SLAM [50]. While it works best with stereo cameras, monocular cameras are also supported, which can be very affordable and lightweight nowadays. ORB-SLAM2 works by extracting features from the environment and tracking their motion. In contrast, Large-Scale Direct SLAM (LSD-SLAM) works with raw images directly without the need for feature extraction. This makes it even more computationally efficient and more applicable to large environments with less distinct features [51].

Other research focusses on distributed SLAM, in which a team of drones collaborate to map an environment [52]. In fact, 2024 has seen the first successful application of a collaborative SLAM algorithm on a swarm of Crazyflies, miniature quadrotor drones, working on a mere 1.5 MB of RAM [53]. This achievement was made possible by the use of Time-of-Flight (ToF) sensors, like the VL53L5CX which generates a 8x8 pixel depth map of the direction it is facing [54]. These ToF-sensors are in principle similar to LiDAR scanners, but have significantly lower ranging distance and do not rotate. ToF-based SLAM, however, requires an extra computing module and an optic flow sensor for motion priors. Moreover, the oscillatory nature of flapping-wing MAVs makes the ToF-based measurements very noisy, which complicates the loop closure. Nevertheless, ToF-sensors offer a lightweight option to sense the environment and avoid collisions. More on that will be elaborated in Chapter 4.

3.3. Visual-Inertial Odometry

Reducing the computational demand even further can be attained by leaving out the mapping of the surroundings and simply reacting to obstacles when necessary. The most common mapless navigation method is *visual-inertial odometry* (VIO), which fuses measurements from the IMU about linear and angular accelerations with visual feature tracking to determine velocities and rotational rates [55]. These methods work best with depth cameras, like OKVIS, published in 2015 [56], which relies on the non-linear optimisation of motion estimates between keyframes. Depth cameras, however, are heavier than single cameras, and their benefits are hardly usable on miniature drones due to the limited possible distance between the cameras, which decreases depth resolution.

A low-compute monocular VIO approach is MSCKF [57]. MSCKF uses an Extended Kalman Filter (EKF) to fuse information from extracted visual features with IMU measurements. Another monocular algorithm, Robust VIO (ROVIO) [58], also uses an EKF to fuse visual and inertial information, but works with pixel intensities directly, instead of extracted features. In fact, the difference between IMU-predicted image patches and actual camera measurements can directly be used as innovation in the Kalman filter update step. Working on pixel-intensities directly allows ROVIO to be used in environments with little texture, which increases robustness. While both ROVIO and MSCKF are already highly computationally efficient, in 2022, Bahnam et al. managed to decrease the average computational time necessary to process a frame by 40 % [59]. This improvement makes ROVIO suitable for miniature drones like the Crazyflie or Flapper Nimble+.

More rudimentary algorithms based on the fusion of visual and inertial measurements build on the concept of *optic flow*, the motion of distinct points in the image plane. Optic flow can be used alongside depth information to draw conclusions on the ego-motion of the drone [11]. While this typically does not yield accurate 3D position estimates, it can help in stabilising the horizontal motion on the drone with minimal effort. Depth information is not always necessary. For instance, if optic flow is used for landing with a fixed-gain proportional controller, de Croon has shown that the onset of instability in the vertical motion can be used to estimate the distance to the ground with a monocular camera [60]. In fact, it was proven that optic flow is also leveraged by flying insects, such as honeybees, to navigate in complex environments [61]. Although most flying insects possess two eyes and could hence in principle infer depth information from parallax, their small size does not allow for a sufficient distance between the eyes to obtain useful depth cues. This problem translates accurately to miniature drones. For optic flow as well as all VIO algorithms holds that they can estimate an absolute position, but the missing reference to a static environment and the integration of measurement errors lead to the accumulation of positioning errors, known as *drift*.

3.4. Indoor Positioning Systems

So while VIO is a very lightweight localisation approach, it needs correction over longer periods of time. This correction can be provided by indoor positioning systems, which aim to observe the drones' position directly, without integrating their prior motion. In fact, this resembles very closely what satellite positioning systems do for the outdoor use case. By measuring the distance to anchors with known positions, the drones can calculate their position in a local coordinate frame with multilateration methods [62]. The drawback of these systems is that they tend to be less accurate. However, fusing these positioning estimates with the more accurate odometry estimates allows for precise localisation with limited drift.

Most commonly, these systems are installed in the indoor workspace of the drones. What distinguishes different approaches is the way they estimate the distances from the drone to the anchors. The most accurate methods are *motion capture systems*, which use cameras to track the positions of reflective infrared markers installed on the drone. This way, they have a direct estimate of the drone's position and even their attitude. A disadvantage is that the pose is estimated off-board and needs to be communicated to the drone. This issue is mitigated by equipping the drone with sensors to receive the infrared rays emitted by the anchors. These sensors can measure the angles to the respective anchors and hence compute their position with multiangulation methods. An example for this technique is the *Lighthouse* positioning system developed by Bitcraze that uses SteamVR beacons as infrared-emitting anchors [63].

Since both these approaches rely on infrared light, they require a direct line of sight between the drone and the anchors. Especially in greenhouses, when drones fly low between the crop rows, they might not maintain this line of sight to a sufficient number of anchors. Moreover, the greenhouse glass walls sharply reflect the infrared rays which leads to artifacts that result in very confident but false position estimates. For this reason, approaches based on radio-frequency waves have been developed. As radio-waves can penetrate most materials and warp around them, a direct line of sight is not required. Next to that, they also do not get reflected off glass panes, which makes them fit for greenhouse applications. Initial research started in the late 2010s with measuring the *received signal strength* (RSS) of WiFi and Bluetooth Low Energy (BLE) signals as a proxy for distance [14, 62]. Accuracies for these approaches lie in the order of decimetres to metres and are hence not suitable for precise navigation between crop rows.

A more promising technique uses very sharp, high-bandwidth pulses for ranging, known as *Ultra-Wideband* (UWB). The time needed for these pulses to be propagated back and forth from the anchors to the drone is used to calculate the distance in what is called Two-Way-Ranging (TWR). Alternatively, the time difference of arrival (TDOA) of signals emitted by multiple anchors can be used by a drone to calculate its position [64]. With these UWB methods, drones can estimate their position in the order of a few decimetres, and combined with optic flow or VIO even to one decimetre. Hence, UWB positioning is considered the most promising indoor positioning technique these days [64].

Looking at swarms of drones, current research has come close to avoiding external infrastructure for localisation altogether. In the field of *reciprocal localisation*, drones do not estimate their positions with respect to static anchors but with respect to other moving drones. This was first achieved in 2021 for the 2D case by Li et al. [65], and for the 3D case later in 2022 [16], both for tiny quadrotor drones and flapping-wing drones. The current challenge is that in contrast to the static anchors, the initial positions of the drones are unknown, so the underlying EKF needs about 20-30s to converge. In this initial uncertain time, the drones need a free space to move in until their relative positions are known [16]. Moreover, while this approach can be used for collision avoidance between drones, there is no absolute position information, so even if a map of the environment is available, it cannot be used.

3.5. Homing

An absolute sense is in principle possible with a VIO or optic flow approach, but it drifts over time. Long-term absolute positioning is only possible with an external positioning system, such as UWB anchors. However, it might be sufficient to cancel the drift every time the drones land to change their batteries. In the swarming case, battery swaps can be scheduled subsequently with a constant period to have more frequent drift

resets. However, as the drift will be maximal just before landing, it might not be fair to assume that this landing succeeds. For a guaranteed landing, a homing algorithm might be necessary. [47]

A great example of a minimal effort homing strategy was demonstrated by McGuire et al. in 2019 [17]. When a certain battery threshold is reached, the drones simply follow the gradient of the received signal strength of a radio beacon. Since drones need to be monitored via radio anyway, this solution comes at no extra weight or hardware cost. A more complex, yet entirely infrastructure-less solution was published by van Dijk et al. in 2024 [12]. In this approach, a 360 degree camera was placed on top of a Crazyflie Brushless to capture periodic snapshots of the environment. When the drone was commanded to return to home, it simply followed its optic-flow-based absolute path back to the locations where snapshots have been taken. The drone then checks with its camera if the snapshot location has been reached by comparing the low-pass filtered horizontal features. This way, the drift could be cancelled regularly over longer distances.

3.6. Conclusion

The performance of the different navigation methods discussed in this chapter with respect to the requirements for greenhouse monitoring is summarised in Table 3.1.

Table 3.1: Qualitative comparison of selected indoor navigation methods with respect to the requirements for greenhouse monitoring.

Method	Pos. of Obstacles	Pos. of Other Drones	Absolute Position	Required Compute	External Infrastructure	Flapper Suitability
V-SLAM [46, 48, 51]	Yes	No	Yes	High	No	Low
ToF-SLAM [53]	Yes	No	Yes	Mid	No	Low
Direct ToF [54]	Yes	No	No	Low	No	High
ROVIO [58, 59]	No	No	Yes	Mid	No	Mid
Optic Flow [60, 61]	No	No	Yes	Low	No	Mid
Anchored UWB [66]	No	Yes	Yes	Low	Yes	Mid
Reciprocal UWB [16]	No	Yes	No	Low	No	Mid

Fields coloured in red indicate that a requirement is violated, orange indicates compliance but with problems. Since no single method can fulfil all the requirements, a combination must be chosen. An ideal combination would be to use **ToF-sensors** for obstacle detection and **optic flow** to estimate an absolute position. However, the absolute position estimates from optic flow are significantly less accurate than those from **ROVIO**, which should alternatively be used. The trade-off here is that ROVIO also requires significantly more compute and is heavier. Lastly, the positions of the other drones in the swarm need to be estimated with **reciprocal ultra-Wideband** ranging. The subsequent thesis research focusses on peer localisation with UWB and ToF sensors to avoid collisions.

4

Path Planning

Having established a suitable means of navigation for flapping-wing MAVs in greenhouses, the drones know where they are relative to their environment and their peers, ready to carry out their tasks. This chapter aims to answer the question of how exactly drones can move in a smart but efficient way to do their jobs.

4.1. Explicit Path Planning

Most commonly, drones and robots plan their paths explicitly, i.e. with an explicit list of coordinates in the local coordinate frame that needs to be followed. Being a very intuitive approach, paths can be easily visualised for verification. Moreover, explicit planning comes with the opportunity of optimality, i.e. methods exist to plan paths optimally with respect to a certain utility function.

One of these optimal methods that is very applicable to crop monitoring is *Informative Path Planning* (IPP). IPP, adapted for UAV-based terrain monitoring by Popović et al. in 2017, aims to maximise the information gain relative to a budget that typically represents the flight time [67]. This way, drones can focus on more information-rich regions first while constructing a stochastic map belief. For that purpose, the flight path is updated online based on new discoveries. The stochastic map belief is modelled as a Gaussian process (GP) which comprises two components distributed over a 2D or 3D domain: the first is the expectation of the monitored variable, e.g. NDVI, and the second makes up the covariance of that monitored variable and hence gives insight about the local uncertainty. The paths are planned adaptively with the CMA-ES metaheuristic solver [68]. This requires significant compute however, with real-world experiments performed on a 3.2 GHz / 16 GB RAM system.

To reduce planning time, the method was refined in 2022 by Cao et al. by discretising the planning space into a graph and encoding the map beliefs into node-features for context [69]. This approach is called CAtNIPP: Context-Aware Attention-based Netwrork for IPP. Using *deep reinforcement learning* (DRL), a reactive policy for IPP was learned that can be run on a Raspberry Pi 4 with 216 MHz / 2 GB RAM. Furthermore, this single-drone approach can be paired with sequential greedy assignment (SGA) [70]. SGA is a fast but suboptimal algorithm to let a team of drones sequentially but collaboratively plan their flight paths. Combined with CAtNIPP, individual drones can share their map beliefs and intended flight paths with the swarm. This way, each drone can sequentially plan its path to maximise information gain and avoid collisions with the others [71]. Deep Reinforcement Learning is the state-of-the-art in decentralised multi-drone adaptive IPP, with other approaches like Collaborative IPP [21] (Figure 4.1) based on counterfactual multi-agent policy gradients (COMA) [72] or information gain [73].

However, most approaches have so far only been evaluated in simulation with no information on computational demand provided. Furthermore, the IPP approaches above deal with conditions of ideal perception: flying over almost horizontal fields without occlusion and uniform lighting from above. In greenhouses however, drones have to deal with occlusions, e.g. due to leaves, and potentially backlighting from the lamps. These so-called *perceptual factors* should be incorporated next to past observations and peer drones' intent in a holistic IPP framework [74]. In fact, perceptual factors also show repercussions

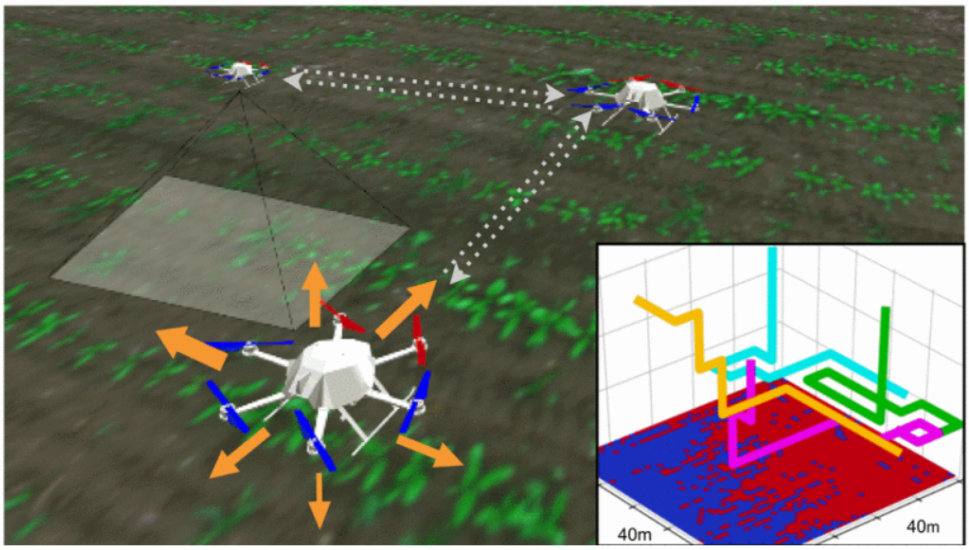


Figure 4.1: Three simulated drones performing collaborative IPP to map a crop field, from [21].

in navigation. Navigation and planning are often treated separately, but modern approaches tend to acknowledge their subtle coupling [45]. In nature, this so-called *sensory-motor coordination* SMC) [75] is for example exploited by locusts, which shake their head to create motion parallax and therefore obtain depth information before jumping [76]. A distinct example from swarm robotics is the inclusion of random movements in a drone swarm to increase the observability of each others position in UWB ranging [47]. More on this in Chapter 5.

Moving back to IPP, other approaches focus on *gas source localisation* (GSL). This could be applicable to volatile compound sensing in greenhouses. Adaptive Cascaded Local Optimal Planning (ACLOP) [77] for instance employs convolutional kernels on local gas concentration belief maps to plan from the small scale to the large scale. While the authors claim ACLOP to be lightweight and executable on small embedded systems, there is no quantitative metric given. Moreover, experiments have only been performed in simulation and not on real devices. These simulations also reduced the planning problem to two dimensions, despite dealing with a clearly three-dimensional phenomenon. A more promising approach is Sniffy Bug [18], which tackles the GSL problem in a reactive approach that has been demonstrated on the Crazyflie, a tiny drone that works on the same processing unit as the Flapper Nimble+.

4.2. Reactive Planning

In contrast to explicit planning, reactive planning does not compute a flight path as an explicit list of waypoints, but instead performs small local actions at each time step, based on most recent sensor information [78]. This allows for a drastic reduction in the computational cost of path planning, but also results in suboptimal paths. While purely reactive approaches do not have knowledge about any past or future states, they can be adapted to memorise certain bits of information, which allows them to solve more complex problems without a strong increase in computational cost [78].

In the case of Sniffy Bug [18], the reactive behaviour is implemented in form of a *finite state machine* (FSM) [79]. FSMs put the drones in certain behavioural states, such as random exploration if no obstacles are in sight or wall-following if the ToF-sensors detect walls in range. Each state contains a limited set of actions which are executed periodically. Broadly speaking, these actions are computed by combining a constant set of parameters with the sensory inputs available at this execution. Next to executing its current action, the robot also checks if its sensory inputs require it to transition to another state. For example, in the state *moving_forward*, the robot might transition to the state *turning_right* if a ToF sensor has detected an obstacle in front. The exact condition of when to transition is also determined by the constant parameters.

The wall-following in Sniffy Bug is based on the so-called *bug algorithm* [80] (Figure 4.2) which first came up in 1986 and is based on the observed behaviour of insects. Bug algorithms were also used for wall-following in swarm-based exploration of indoor environments [17].

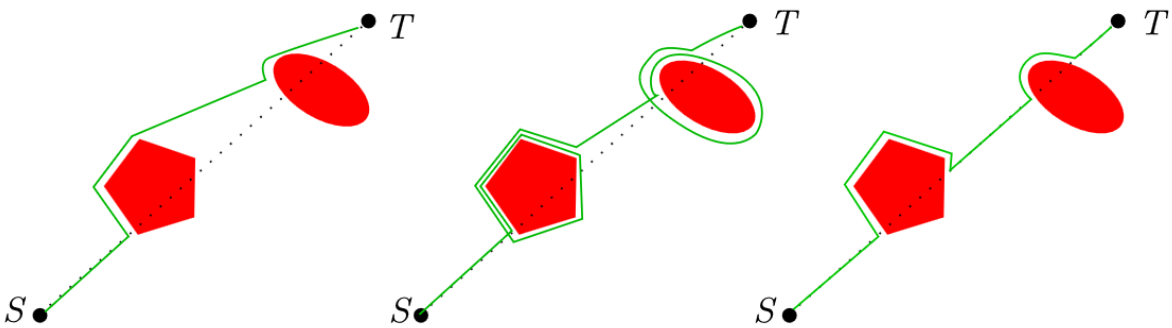


Figure 4.2: Visualisation of different versions of bug algorithms, from [81].

For reactive collision avoidance between drones, *artificial potential fields* (APF) [82] offer an intuitive approach. When a drone enters another drone's repulsion radius, the on-board controller can simulate a spring force on the drone and set an according velocity command to fly away from the other drone. This lightweight approach for example allows drones to fly together in a flock with decentralised control and sensing [83]. APF can also combine repulsive forces from walls and other drones with attractive forces to target points such as plants, which need to be monitored next, as demonstrated by McGuire et al. [17].

The state of the art in inter-drone collision avoidance is set by Alonso-Mora, Beardsley, and Siegwart in 2018, which have adapted the *Optimal Reciprocal Collision Avoidance* (ORCA) [85] method for non-holonomic systems such as quadrotor or flapping-wing drones. ORCA is based on the concept of velocity obstacles, i.e. the representation of obstacles in the velocity domain. For that purpose, the drone assumes a constant velocity of the obstacle for a small time frame. Knowing the relative velocity and positions of the two drones, a set of relative velocity vectors can be found for which the two drones will not collide by the end of the time frame. The drones can then pick the velocity vector that is closest to their desired velocity vector, e.g. by sampling the velocity domain. This will typically mean that a drone needs to make a concession from its desired path, but in a collaborative swarming scenario, this burden can be split in half for each pair of drones, equally distributing the path correction and thereby keeping the swarm in tact as a whole. The adaptation to non-holonomic systems consists of increasing the drones' collision radius by a small amount to account for the position error due to the assumption of holonomic motion [84].

4.3. Conclusion

While explicit planning methods offer immense potential for crop monitoring in general, they are not quite suitable for the limited power and compute available on the Flapper Nimble+. Thus, we must resort to the toolbox of reactive methods, combining them in an efficient behavioural framework to overcome the challenges of multi-drone greenhouse monitoring.

5

Evolutionary Robotics

The limited power and compute of the Flapper Nimble+ and the complexity of navigating through greenhouse environments with multiple drones at once demand utmost efficiency for the planning strategy. In Chapter 4, a range of reactive planning methods were reviewed. This chapter focusses on how to combine these methods with efficiencies that outperform intuitive, human-made strategies. A key approach here lies in Evolutionary Robotics.

5.1. Evolutionary Robotics and Reinforcement Learning

Evolutionary Robotics (ER) is considered a major bio-inspired behavioural approach to robotics, next to Reinforcement Learning (RL). While RL mimics the natural process of learning based on rewards that can be seen in humans day to day, ER mimics the interplay of natural variation and selection that have defined the inherited characteristics of humans and other animals long before they were born [86]. In nature, evolution is responsible for very slow but profound behavioural changes, while learning allows for fast adaptation to new circumstances. However, ER and RL are highly abstracted versions that take place in computer simulations and thus are both able to find solutions for the behaviour of robots in a relatively short time. Both RL and ER can find find efficient reactive behaviours for robots, which ideally succeed at exploiting features of the simulation environment that are un- or even counter-intuitive to humans and would hence be overseen in a manual user-defined solution [86].

While RL is an excellent choice for *Markov Decision Processes* (MDPs), their suitability for only partially observable MDPs or non-markovian tasks is limited. This is because RL aims to estimate a value-function for policies, i.e. the mapping from states to actions. However, if certain states are ambiguous or the value function is discontinuous, this becomes a problem [87]. Swarming is such a non-markovian task as from the perspective of the individual robot, the global state of the swarm as a whole is not observable [47]. ER mitigates these issues by searching in the policy-space directly, bypassing the complex relationship between value and action. Moreover, it is difficult to define suitable rewards for the actions of the individual that stem from the success of the swarm. This is known as the credit assignment problem [88].

5.2. Practical Considerations

Taking a closer look at ER, it relies on meta-heuristic optimisation in the form of evolutionary algorithms, like the highly bio-inspired genetic programming [89], or evolutionary strategies like the more abstract but highly efficient covariance matrix adaptation (CMA-ES) [68]. All optimisation algorithms rely on a scalar *objective function*, which rates the performance of obtained solutions with respect to the user-defined goals. The difficulty herein lies in the precise choice of an accurate objective function that is robust against trivial solutions, which distract the optimiser from finding meaningful solutions [86, 47]. Moreover, like all meta-heuristic optimisation approaches, ER is subject to non-optimality as solutions will almost always converge to local optima instead of finding the global optimum. That is why ER and other meta-heuristics are typically only used for highly complex optimisation problems, for which no closed-form solution exists [86].

Lastly, it must be acknowledged that ER optimises a behaviour for the given simulation environment, which is not guaranteed to be useful in the real world. This so-called *reality gap* can be reduced by adding sensor noise and varying the simulation parameters randomly in order to prevent overfitting, often called Monte Carlo simulation [90]. The reality gap has also been seen in the context of the bias-variance trade-off, concluding that a more abstracted simulation environment, i.e. a more biased environment, will lead to reduced variance in the real world, and thus in a reduced reality gap [91]. According to Scheper and De Croon, this abstraction does not necessarily hinder the evolution from exploiting sensory-motor-coordination (SMC) or the exhibition of emergent behaviours. Moreover, abstraction helps to overcome the bootstrapping problem, i.e. the problem that initial low-level behaviours struggle to cater to the high-level goals of the objective function, and abstraction reduces the computational time of the simulations [92].

5.3. Application to Swarm Robotics and Behaviour Trees

One of the first approaches to ER applied to swarm robotics was F-Race in 2002 by Birattari et al. [93]. F-Race, short for focussed race, generated random sets of behavioural parameters for a swarm of homogeneous ground robots. These solutions were then evaluated with respect to a certain task and after every round of evaluations, the worst performing solutions were discarded [93]. Thus, the more promising solutions receive more thorough evaluation until a final solution remains. This focussed approach was computationally efficient and robust at the same time, which was an important requirement in 2002. F-Race introduced the concepts of evaluation and selection, however the true power of ER lies in recombination and mutation of the most effective solutions to obtain potentially better solutions in future generations. This breakthrough took place in 2007 with Iterated F-Race [94], which improved performance significantly. In 2014, the same research group published AutoMoDe-Vanilla [91], introducing Finite State Machines (FSMs) as behavioural framework for the robots. FSMs are explained in Chapter 4. Together with Iterated F-Race, they allowed ER to outperform a manually designed swarming behaviour for the first time in 2015 [19].

Another behavioural framework especially suitable for ER is the *behaviour tree* (BT). Behaviour trees are a generalised form of FSMs. The main difference is that BTs allow a behaviour to be defined with very high precision by branching the tree over and over again and adding new nodes [95]. In FSMs, this is only possible with hierarchical states. The problem is that adding a new state to refine the behaviour requires connecting it to all other states within a super state. This leads to combinatorial overload, i.e. a rapidly rising number of links, too large to be maintained and understood by humans. In contrast, BTs only require the understanding of nearby nodes to understand a sub-behaviour. The shift from FSMs to BTs is akin to the shift from GOTO-based programming to the nowadays ubiquitous function-based programming [95].

BTs were implemented alongside a tournament selection based genetic algorithm in 2016 to evolve a behaviour for the DelFly Explorer to fly through a window [9]. Again, the evolved solution outperformed a manually designed solution in the real world, underlining the potential of ER. Scheper et al. emphasize the benefits of BTs: the high interpretability of evolved BTs allow the user to identify and remove inactive components in the solution which can reduce the size and thus computational demand of the behaviour enormously without impairing its performance. Furthermore, this interpretability enables the user to make intuitive adaptations that can bridge the gap from simulation to reality, which is the decisive advantage of BTs over FSMs or other representations such as neural networks [9, 96, 47].

The success of behaviour trees was echoed by the authors of the AutoMoDe series, who in 2021 published a tool for creating and evaluating BTs [97] for swarms, that in 2022 was used for AutoMoDe-Cedrata, a BT and genetic programming based swarming framework that supports peer-to-peer communication [20]. While Cedrata proved to be effective, it did not manage to outperform a manual user-designed BT for various swarming tasks. This can potentially be due to the limited complexity of BTs in Cedrata. With a maximum of 13 nodes, the evolutionary algortihm might not be able to explore sufficiently many designs before converging on more compact solutions. For comparison, the approach in [9] temporarily generated BTs with up to 7000 nodes before settling at 32 nodes for the final solution. Hence, it must be ensured that the BTs have sufficient exploratory freedom to maximise their utility.

6

Conclusion

This literature review has given an overview of the current state of flapping-wing MAVs and the use of drones in agriculture. Their robust and non-destructive properties make flappers ideal candidates for real-time greenhouse monitoring. However, unlike their larger quadrotor counterparts flying outdoors, indoor flappers cannot rely on GNSS satellites for positioning and do not have the computational capacity for map-based optimal path planning. While different approaches to indoor positioning exist, there is currently no single method providing bounded position estimates, obstacle detection and peer localisation simultaneously.

Deploying a swarm of flappers enhances the robustness of the system to individual failures. Moreover, it creates room for complex collective behaviour, outperforming that of single agents. But for swarming to work, there must be an efficient coordination strategy that can be implemented on the Flapper's microprocessor. Examples for effective coordination are given by finite state machines or behaviour trees. Moreover, it was found that strategies obtained by evolutionary robotics can outperform manually designed ones and in for the multi-agent case, also those obtained by reinforcement learning.

6.1 Research Objective

Hence, the objective of this research is to develop a coordination strategy for a swarm of flapping-wing MAVs for autonomous greenhouse exploration by means of reactive planning methods, time-of-flight sensors and ultra-wideband based peer localisation.

6.2 Research Question

Central to the pursuit of said objective is the following research question:

How can a swarm of flapping-wing MAVs be coordinated to perform decentralised autonomous greenhouse exploration by means of reactive planning methods?

This can be further split in the following sub-questions:

1. Which methods of evolutionary robotics can be applied to find the best possible coordination strategy?
2. Which reactive planning methods implemented as useful behaviour modules for multi-drone greenhouse monitoring?
3. How can a coordination strategy found in simulation be effectively implemented on the real drone platform?

References

[1] Roland Rau et al. “Europe, the Oldest-Old Continent”. In: *The Demography of Europe*. Ed. by Gerda Neyer et al. Dordrecht: Springer Netherlands, 2013, pp. 119–137. DOI: 10.1007/978-90-481-8978-6_6. URL: https://doi.org/10.1007/978-90-481-8978-6_6.

[2] *Share of the labor force employed in agriculture*. en. https://ourworldindata.org/grapher/share-of-the-labor-force-employed-in-agriculture?tab=chart&time=1994..latest&country=~European+Union. Accessed: 2025-1-16.

[3] Eldert J. Van Henten et al. “Agricultural Robotics and Automation [TC Spotlight]”. In: *IEEE Robotics and Automation Magazine* 29 (4 Dec. 2022), pp. 145–147. DOI: 10.1109/MRA.2022.3213136.

[4] Eldert Van Henten et al. *Embracing Robotics and Intelligent Machine Systems for Smart Agricultural Applications [From the Guest Editors]*. Dec. 2023. DOI: 10.1109/MRA.2023.3322917.

[5] Peter G. Steeneken et al. *Sensors in agriculture: towards an Internet of Plants*. Dec. 2023. DOI: 10.1038/s43586-023-00250-x.

[6] Christian Geckeler et al. “Robotic Volatile Sampling for Early Detection of Plant Stress: Precision Agriculture Beyond Visual Remote Sensing”. In: *IEEE Robotics and Automation Magazine* 30 (4 Dec. 2023), pp. 41–51. DOI: 10.1109/MRA.2023.3315932.

[7] Muhammet Fatih Aslan et al. *A Comprehensive Survey of the Recent Studies with UAV for Precision Agriculture in Open Fields and Greenhouses*. Feb. 2022. DOI: 10.3390/app12031047.

[8] *Flapper Drones - Bioinspired flyig robots*. en. https://flapper-drones.com/wp/. Accessed: 2025-1-16.

[9] Kirk YW Scheper et al. “Behavior trees for evolutionary robotics”. In: *Artificial life* 22.1 (2016), pp. 23–48.

[10] Guido CHE De Croon et al. “Design, aerodynamics and autonomy of the DelFly”. In: *Bioinspiration & biomimetics* 7.2 (2012), p. 025003.

[11] Hann Woei Ho et al. “Optical flow-based control for micro air vehicles: an efficient data-driven incremental nonlinear dynamic inversion approach”. In: *Autonomous Robots* 48.8 (2024), p. 22.

[12] Tom van Dijk et al. “Visual route following for tiny autonomous robots”. In: *Science Robotics* 9.92 (2024), eadk0310.

[13] Chanyoung Ju et al. “Multiple UAV systems for agricultural applications: Control, implementation, and evaluation”. In: *Electronics* 7.9 (2018), p. 162.

[14] Mario Coppola et al. *A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints*. Feb. 2020. DOI: 10.3389/frobt.2020.00018.

[15] Feng Shan et al. “Ultra-wideband swarm ranging protocol for dynamic and dense networks”. In: *IEEE/ACM Transactions on Networking* 30.6 (2022), pp. 2834–2848.

[16] Sven Pfeiffer et al. “Three-dimensional relative localization and synchronized movement with wireless ranging”. In: *Swarm Intelligence* 17.1 (2023), pp. 147–172.

[17] KN McGuire et al. “Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment”. In: *Science Robotics* 4.35 (2019), eaaw9710.

[18] Bardienus P Duisterhof et al. “Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 9099–9106.

[19] G Francesca et al. “AutoMoDe-Chocolate: a Method for the Automatic Design of Robot Swarms that Outperforms Humans”. In: (2014).

[20] Jonas Kuckling et al. “AutoMoDe-Cedrata: Automatic Design of Behavior Trees for Controlling a Swarm of Robots with Communication Capabilities”. In: *SN Computer Science* 3 (2 Mar. 2022). DOI: 10.1007/s42979-021-00988-9.

[21] Jonas Westheider et al. “Multi-uav adaptive path planning using deep reinforcement learning”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 649–656.

[22] Robin J Wootton. “Palaeozoic insects”. In: *Annual review of entomology* 26.1 (1981), pp. 319–344.

[23] Leonardo da Vinci. *Codex on the Flight of Birds*. Manuscript, 18 folios. 1505.

[24] Charles P Ellington. “The novel aerodynamics of insect flight: applications to micro-air vehicles”. In: *Journal of Experimental Biology* 202.23 (1999), pp. 3439–3448.

[25] John David Anderson et al. *Introduction to flight*. Vol. 582. McGraw-Hill Higher Education New York, NY, USA, 2005.

[26] Guido de Croon. “Flapping wing drones show off their skills”. In: *Science Robotics* 5.44 (2020), eabd0233.

[27] Matěj Karásek et al. “A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns”. In: *Science* 361.6407 (2018), pp. 1089–1094.

[28] SR Jongerius et al. “Design of a flapping wing vision-based Micro-UAV”. In: *Design synthesis exercise, Faculty of Aerospace Engineering, TU Delft* (2005).

[29] Robert J Wood. “The first takeoff of a biologically inspired at-scale robotic insect”. In: *IEEE transactions on robotics* 24.2 (2008), pp. 341–347.

[30] Christophe De Wagter et al. “Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 4982–4987.

[31] Jorgen Nijboer et al. “Longitudinal grey-box model identification of a tailless flapping-wing mav based on free-flight data”. In: *AIAA scitech 2020 forum*. 2020, p. 1964.

[32] G Gonzalez Archundia et al. “Position controller for a flapping wing drone using uwb”. In: *12th International Micro Air Vehicle Conference*. 2021, pp. 85–92.

[33] Abderahman Rejeb et al. *Drones in agriculture: A review and bibliometric analysis*. July 2022. DOI: 10.1016/j.compag.2022.107017.

[34] B Kh Akhalaya et al. “Automated multifunctional tillage machine”. In: *Russian Agricultural Sciences* 44.1 (2018), pp. 105–107.

[35] Sandro Augusto Magalhães et al. “Active perception fruit harvesting robots—A systematic review”. In: *Journal of Intelligent & Robotic Systems* 105.1 (2022), p. 14.

[36] Alexander You et al. “Semiautonomous precision pruning of upright fruiting offshoot orchard systems: An integrated approach”. In: *IEEE Robotics & Automation Magazine* (2023).

[37] The Enterprise World. *Drone mapping technology: Industries gaining precision and efficiency*. en. https://theenterpriseworld.com/drone-mapping-technology/. Accessed: 2025-1-31. Dec. 2024.

[38] Marco Canicattì et al. “Drones in vegetable crops: a systematic literature review”. In: *Smart Agricultural Technology* (2024), p. 100396.

[39] Corey Davidson et al. “NDVI/NDRE prediction from standard RGB aerial imagery using deep learning”. In: *Computers and Electronics in Agriculture* 203 (2022), p. 107396.

[40] Jianliang Wang et al. “NDVI Estimation Throughout the Whole Growth Period of Multi-Crops Using RGB Images and Deep Learning”. In: *Agronomy* 15.1 (2024), p. 63.

[41] Meredith C Schuman et al. “The layers of plant responses to insect herbivores”. In: *Annual review of entomology* 61.1 (2016), pp. 373–394.

[42] Takefumi Hiraguri et al. “Autonomous drone-based pollination system using AI classifier to replace bees for greenhouse tomato cultivation”. In: *IEEE Access* (2023).

[43] Qiang Shi et al. “Study on assistant pollination of facility tomato by UAV”. In: *2019 ASABE Annual International Meeting*. American Society of Agricultural and Biological Engineers. 2019, p. 1.

[44] Myron Kayton. *Avionics Navigation Systems*. Vol. 2. John Wiley and Sons, Inc. google schola, 1997.

[45] Roland Siegwart et al. *Introduction to autonomous mobile robots*. MIT press, 2011.

[46] Iman Abaspur Kazerouni et al. *A survey of state-of-the-art on visual SLAM*. Nov. 2022. DOI: 10.1016/j.eswa.2022.117734.

[47] Mario Coppola et al. “A survey on swarming with micro air vehicles: Fundamental challenges and constraints”. In: *Frontiers in Robotics and AI* 7 (2020), p. 18.

[48] Raul Mur-Artal et al. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262.

[49] Raul Mur-Artal et al. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[50] Andrew J Davison et al. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067.

[51] Jakob Engel et al. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[52] Danping Zou et al. “Collaborative visual SLAM for multiple agents: A brief survey”. In: *Virtual Reality & Intelligent Hardware* 1.5 (2019), pp. 461–482.

[53] Vlad Niculescu et al. “Ultra-Lightweight Collaborative Mapping for Robot Swarms”. In: *arXiv preprint arXiv:2407.03136* (2024).

[54] Vlad Niculescu et al. “Towards a Multi-Pixel Time-of-Flight Indoor Navigation System for Nano-Drone Applications”. In: *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2022, pp. 1–6. DOI: 10.1109/I2MTC48687.2022.9806701.

[55] Guoquan Huang. “Visual-inertial navigation: A concise review”. In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 9572–9582.

[56] Stefan Leutenegger et al. “Keyframe-based visual–inertial odometry using nonlinear optimization”. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[57] Anastasios I Mourikis et al. *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation*.

[58] Michael Bloesch et al. “Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback”. In: *International Journal of Robotics Research* 36 (10 Sept. 2017), pp. 1053–1072. DOI: 10.1177/0278364917728574.

[59] SA Bahnam et al. “Improving the computational efficiency of ROVIO”. In: *Unmanned Systems* 12.03 (2024), pp. 589–598.

[60] Guido C.H.E de Croon. “Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy”. In: *Bioinspiration & biomimetics* 11.1 (2016), p. 016004.

[61] Mandyam V Srinivasan et al. “How honeybees make grazing landings on flat surfaces”. In: *Biological cybernetics* 83 (2000), pp. 171–183.

[62] Naser El-Sheimy et al. “Indoor navigation: State of the art and future trends”. In: *Satellite Navigation* 2.1 (2021), p. 7.

[63] Arnaud Taffanel et al. “Lighthouse positioning system: dataset, accuracy, and precision for UAV research”. In: *arXiv preprint arXiv:2104.11523* (2021).

[64] Siyuan Chen et al. “A survey of robot swarms’ relative localization method”. In: *Sensors* 22.12 (2022), p. 4424.

[65] Shushuai Li et al. “An autonomous swarm of micro flying robots with range-based relative localization”. In: *arXiv preprint arXiv:2003.05853* (2020).

[66] Wenda Zhao et al. “Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3639–3646.

[67] Marija Popović et al. “An informative path planning framework for UAV-based terrain monitoring”. In: *Autonomous Robots* 44.6 (2020), pp. 889–911.

[68] Nikolaus Hansen. “The CMA Evolution Strategy: A Tutorial”. In: (Apr. 2016). URL: http://arxiv.org/abs/1604.00772.

[69] Yuhong Cao et al. “CAtNIPP: Context-aware attention-based network for informative path planning”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 1928–1937.

[70] Micah Corah et al. “Efficient Online Multi-robot Exploration via Distributed Sequential Greedy Assignment.” In: *Robotics: Science and Systems*. Vol. 13. Cambridge, MA. 2017.

[71] Tianze Yang et al. “Intent-based Deep Reinforcement Learning for Multi-agent Informative Path Planning”. In: *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE. 2023, pp. 71–77.

[72] Jakob Foerster et al. “Counterfactual multi-agent policy gradients”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[73] Carlos Carbone et al. “Monitoring and mapping of crop fields with UAV swarms based on information gain”. In: *Distributed Autonomous Robotic Systems: 15th International Symposium*. Springer. 2022, pp. 306–319.

[74] David Morilla-Cabello et al. “Perceptual factors for environmental modeling in robotic active perception”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 4605–4611.

[75] John Dewey. “The reflex arc concept in psychology.” In: *Psychological review* 3.4 (1896), p. 357.

[76] Erik C Sobel. “The locust’s use of motion parallax to measure distance”. In: *J. Comp. Physiol. A* 167.5 (1990), pp. 579–588.

[77] Omar Velasco et al. “An adaptive informative path planning algorithm for real-time air quality monitoring using UAVs”. In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2020, pp. 1121–1130.

[78] Stefano Nolfi. “Power and the limits of reactive agents”. In: *Neurocomputing* 42.1-4 (2002), pp. 119–145.

[79] David Lee et al. “Principles and methods of testing finite state machines-a survey”. In: *Proceedings of the IEEE* 84.8 (1996), pp. 1090–1123.

[80] V. Lumelsky et al. “Dynamic path planning for a mobile automaton with limited information on the environment”. In: *IEEE Transactions on Automatic Control* 31.11 (1986), pp. 1058–1063. DOI: 10.1109/TAC.1986.1104175.

[81] K.N. McGuire et al. “A comparative study of bug algorithms for robot navigation”. In: *Robotics and Autonomous Systems* 121 (2019), p. 103261. DOI: https://doi.org/10.1016/j.robot.2019.103261. URL: https://www.sciencedirect.com/science/article/pii/S0921889018306687.

[82] Oussama Khatib. “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots”. In: *The International Journal of Robotics Research* 5.1 (1986), pp. 90–98. DOI: 10.1177/027836498600500106. URL: https://doi.org/10.1177/027836498600500106.

[83] Gábor Vásárhelyi et al. “Optimized flocking of autonomous drones in confined environments”. In: *Science Robotics* 3.20 (2018), eaat3536.

[84] Javier Alonso-Mora et al. “Cooperative collision avoidance for nonholonomic robots”. In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 404–420.

[85] J Van et al. “Reciprocal n-body collision avoidance”. In: *Robotics research*. Springer, 2011, pp. 3–19.

[86] Stefano Nolfi et al. *Evolutionary robotics*. Springer, 2016.

[87] GCHE de Croon et al. “Evolutionary learning outperforms reinforcement learning on non-Markovian tasks”. In: *Workshop on Memory and Learning Mechanisms in Autonomous Robots, 8th European Conference on Artificial Life, Canterbury, Kent, UK*. 2005.

[88] Manuele Brambilla et al. “Swarm robotics: a review from the swarm engineering perspective”. In: *Swarm Intelligence* 7 (2013), pp. 1–41.

[89] John H Holland. “Genetic algorithms”. In: *Scientific american* 267.1 (1992), pp. 66–73.

[90] Nick Jakobi et al. “Noise and the reality gap: The use of simulation in evolutionary robotics”. In: *Advances in Artificial Life: Third European Conference on Artificial Life Granada, Spain, June 4–6, 1995 Proceedings 3*. Springer. 1995, pp. 704–720.

[91] Gianpiero Francesca et al. “AutoMoDe: A novel approach to the automatic design of control software for robot swarms”. In: *Swarm Intelligence* 8 (2014), pp. 89–112.

[92] Kirk YW Scheper et al. “Abstraction, sensory-motor coordination, and the reality gap in evolutionary robotics”. In: *Artificial life* 23.2 (2017), pp. 124–141.

[93] Mauro Birattari et al. “A Racing Algorithm for Configuring Metaheuristics.” In: *Gecco*. Vol. 2. 2002. Citeseer. 2002.

[94] Prasanna Balaprakash et al. “Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement”. In: *Hybrid Metaheuristics: 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007. Proceedings 4*. Springer. 2007, pp. 108–122.

[95] Michele Colledanchise et al. “Behavior Trees in Robotics and AI: An Introduction”. In: (Aug. 2017). DOI: 10.1201/9780429489105. URL: http://arxiv.org/abs/1709.00084%20http://dx.doi.org/10.1201/9780429489105.

[96] Simon Jones et al. “Evolving behaviour trees for swarm robotics”. In: *Distributed Autonomous Robotic Systems: The 13th International Symposium*. Springer. 2018, pp. 487–501.

[97] Jonas Kuckling et al. *AutoMoDe Editor: a visualization tool for AutoMoDe*. Tech. rep. Tech. Rep. TR/IRIDIA/2021-009 (IRIDIA, Université libre de Bruxelles …, 2021.

Part II

Scientific Article

Evolving Behaviour Trees to Control a Swarm of Flapping-Wing Micro Aerial Vehicles for Greenhouse Exploration

L.H. Uptmoor, S. Stroobants, M. Popović, G.C.H.E. de Croon
Delft University of Technology, Delft, The Netherlands

Abstract

Micro aerial vehicles have shown promising use to further automate food production in greenhouses recently. Compared to conventional multirotor drones, flapping-wing drones offer safe and robust operation around plants due to their soft, slowly-moving wings. Their limited sensing and computational capabilities, however, prohibit the use of map-based navigation methods. To compensate for individual shortcomings, swarming ensures scalability and redundancy. This work proposes a hardware setup combining time-of-flight (ToF) and ultra-wideband (UWB) sensing and explores the artificial evolution of behaviour trees as a reactive planning strategy. Genetic programming, paired with CMAES fine-tuning was able to improve a human-designed exploration strategy by 50%. Neuroevolution has been investigated to encourage emergent swarming behaviours, but requires further experimentation in combination with behaviour trees. The solution obtained in simulation can be readily ported to hardware, but a reality gap in performance persists. These findings contribute to the development of lightweight, scalable aerial systems for autonomous greenhouse monitoring.

1 Introduction

The way our food is produced is drastically changing: from 1994 to 2019, the fraction of Europeans working in agriculture has fallen from 11% to just 4%[1]. This is only possible because of automation and intensification [1]. An outstanding example for that are Dutch greenhouses, which have evolved from simple glass containers to food factories with meticulously controlled climates [2]. Since the early 2020s, micro aerial vehicles (MAVs), have joined static sensors in providing fine-grained data to the growers, increasing productivity even further [3, 4, 5, 6]. The advent of commercially available flapping-wing drones, like the Flapper Nimble+, looks especially promising: their soft and comparatively slowly moving wings are inherently safe to use around delicate crops and even humans. While quadrotor drones would cut leaves and crash in the event of a direct contact, flapping-wing drones have demonstrated their robustness and ability to maintain secure flight around plants [7].

But challenges remain: the Flappers' limited on-board processing capabilities have ruled out simultaneous localisation and mapping (SLAM) [8, 9] and map-based explicit planning methods like informative path planning (IPP) [10, 11]. What's more, their oscillatory nature has so far hampered the use of visual-inertial navigation methods. As a result, we have chosen reactive planning approaches which make decisions based on recent sensory inputs [12].

To unfold the true potential of these computationally constrained Flappers, we resort to the bio-inspired concept of swarming [13, 14, 15]. Swarming has three key advantages: (1) it allows flexibly scaling a team of agents to environments of any size; (2) when individual reliability is limited by hardware, decentralised swarms offer redundancy and thus robustness to individual failures; (3) even with limited individual cognition, a swarm of agents can exhibit emergent behaviours that allow solving far more complex tasks than a single agent could ever achieve [16]. How to design for these emergent behaviours remains subject to research. Mimicking natural evolution seems a promising approach.

The contributions of this article are twofold:

1. We present a reproducible hardware setup that allows simultaneous height estimation, obstacle detection and decentralised peer-to-peer localisation for the Flapper Nimble+ drone platform, and provide real-world autonomous peer localisation data.

2. We explore different levels of evolving a low-compute, decentralised reactive planning algorithm for a swarm of Flappers, using hybrid approaches of neuroevolution, genetic programming of behaviour trees, and parameter tuning with CMAES.

The remainder of this article is structured as follows: section 2 gives a short account of related work in resource-constrained indoor navigation and swarm intelligence with behaviour trees. Next, section 3 presents the methods by

[1] https://ourworldindata.org/grapher/share-of-the-labor-force-employed-in-agriculture?tab=chart&time=1994..latest&country=~European+Union

giving an overview of the Flapper Nimble+ and its modified sensing capabilities, as well as the different evolution frameworks for its behaviour tree. The experimental setup is outlined in section 4. Thereafter, section 5 provides the evolution results, followed by the limitations and discussion of potential shortcomings in section 6. At last, section 7 concludes this work and presents an outlook on future work.

2 Related Work

A cornerstone in the field of swarming is the ability of agents to locate other members of the swarm. This allows for collision avoidance or coordinating collective movements such as formation flight [17]. Pfeiffer et al. achieved a major breakthrough on this matter by the dual-use of a DWM1000 ultra-wideband (UWB) antenna: the antenna can be used to share on-board velocity estimates with the swarm, but also to estimate the distance between drones. Correcting relative velocities with UWB distance estimates by means of an Extended Kalman Filter (EKF) eliminates drift and thus yields bounded peer-to-peer positions in 3D [18, 19]. While their approach paved the way for other UWB-localisation methods, it lacks scalability in its communication protocol and requires long convergence times if initial positions are unknown. The scalability problem has been mitigated by more scalable broadcast-based communication protocols [20], and the convergence problem we mitigate by launching the drones from pre-defined starting locations.

As swarms are typically constituted by small or even expendable agents, they tend to lack the computational power for mapping or explicit path planning. McGuire et al. have addressed this by implementing reactive behaviour in the form of a finite-state machine (FSM) to explore an unknown indoor environment [15]. They relied on radio signal strength indicator (RSSI) for inter-drone collision avoidance. The success was echoed by Duisterhof et al. in 2021, who adapted the FSM approach to include UWB-based peer localisation with the goal of gas source localisation [21]. However, both approaches rely on optic flow sensing that is unsuitable for flapping-wing platforms due to feature matching failures [22]. We propose a reactive control scheme that builds on more oscillation-resistant sensing modalities.

Small FSMs have proven their utility in reactive planning. But once a more complex behaviour is necessary, they tend to suffer from combinatorial overload, i.e. the transitions between a large number of states become hard to overview for a human designer. Behaviour trees (BTs), on the other hand, allow for the same expressivity as FSMs, but organise their actions more locally, which significantly improves their scalability [23]. A major advantage over black-box methods like neural networks is the human readability of BTs, which facilitates manual adaptation and bridging the reality gap [14].

Behaviour Trees for swarm robotics are often used along with evolutionary algorithms. Evolution's edge over Reinforcement Learning (RL) for instance, is its ability to reward communal efforts directly, while RL needs a state-action pair of an individual agent to assign reward, which scales poorly with number of agents [24, 25]. Evolved BTs have been successfully implemented on a flapping-wing drone by Scheper et al. in 2016 with the goal of flying through a window [26]. It was then not long until their application to swarm robotics. Jones et al. used genetic programming to evolve a BT for a ground-robot foraging task [13]. Although agents were given the ability to communicate and store information, the evolved solution did not make use of that, so the agents essentially work for themselves. More recently, Kuckling et al. also pursued the goal of evolving swarm communication [27], with limited success. One reason for that could be a too restrictive maximum tree size constraint. Thus, this work will permit larger trees to allow more exploration.

3 Methods

3.1 Hardware Modifications

The chosen drone platform for this work is the Flapper Nimble+[2], a commercially available flapping-wing MAV, rooted in the academic research around the Delfly [28, 29, 30]. With two wing pairs, a wingspan of $49\,\mathrm{cm}$ and a maximum takeoff weight of $127\,\mathrm{g}$, it is capable of stable and controllable flight for up to 5 minutes. It uses differential flapping frequencies on its two wingpairs to control its roll angle. Pitch is controlled by setting the wing dihedral angle with a servo motor. And a yawing moment is generated by feathering its wing pairs in opposite directions with another servo. It can be steered remotely via a 2.4 GHz antenna, interfacing with its STM32F405 microcontroller unit on the Crazyflie Bolt 1.1[3] flight controller board.

To enable autonomous navigation, obstacle avoidance and peer-to-peer communication, the following changes were made: the landing gear is amended with two VL53L8CX[4] time-of-flight (ToF) sensors [31] on breakout boards, mounted with 3D-printed parts that ensure one sensor faces forward, and one sensor faces downward, see Figure 1 and Figure 2, respectively. The sensors provide a 8×8 depth array within a square field of view of 48.4^o. While the manufacturers claim a 4.00 m depth range, practice has shown that reliable estimates are only obtainable until about 1.50 m. The front sensor is intended for obstacle recognition, the bottom one for height estimation.

The data streams from both sensors are joined on a I2C bus, which is read by a Teensy 4.1 microprocessor board[5] that is mounted on the back of the Flapper with two 3D-printed

[2] https://flapper-drones.eu/nimbleplus/

[3] https://www.bitcraze.io/products/crazyflie-bolt-1-1/

[4] https://www.st.com/en/imaging-and-photonics-solutions/vl53l8cx.html

[5] https://www.pjrc.com/store/teensy41.html

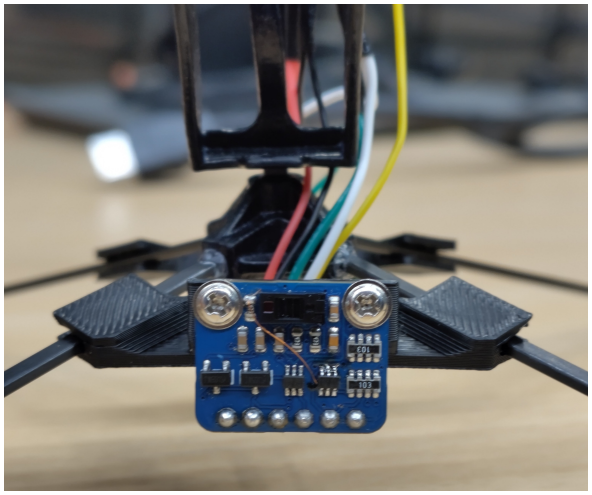


Figure 1: Front view of the Flapper's landing gear including the forward ToF breakout board in blue.

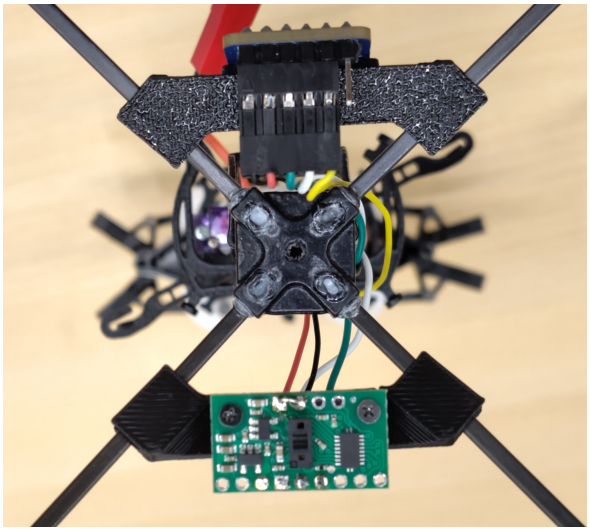


Figure 2: Bottom view of the Flapper's landing gear. The downward ToF breakout board can be seen in green.

brackets. The Teensy takes over pre-processing of the raw ToF sensor data, like filtering, height estimation and calculating obstacle densities in the horizontal field of view. It is soldered to a Crazyflie battery holder to facilitate the plug-and-play pin connection with the Crazyflie Bolt. The Teensy and Bolt communicate via UART2. The setup can be seen in Figure 3.

At last, a Crazyflie Locodeck[6] with a DWM1000 UWB module is placed on the front of the Crazyflie Bolt, also via the pin connectors. On the back of the locodeck, the traces GS1 and GS2 had to be cut, and the pads GS3 and GS4 soldered together to use the alternative UART pin set. This is because the default UART pins are occupied by the Teensy now. The use of these external decks requires the LED ring to be deactivated in the Flapper configuration file. The front view in Figure 4 shows the setup. The locodeck allows peer-to-peer communication, distance estimation, and thus also relative localisation [18].

With all modifications, the weight of the Flapper including battery totals 127g, which is in line with the maximum, although without margin. Future weight optimisation is likely possible. The firmware necessary to fly with the hardware expansions can be found on GitHub [7], just like the firmware for the Teensy 4.1[8].

3.2 Behaviour Tree Implementation

To bestow the agents with a modular and lightweight exploration capability, behaviour trees (BTs) have been chosen as reactive planning strategy. In this work, the BT framework as explained by Colledanchise and Ögren is used [23]. It can be summarised as follows: BTs consist of leaf nodes, which can execute an action or assess a condition, and composite nodes, which control the logic in which the leaf nodes are triggered. Composite nodes can have other nodes as their children, which creates depth of the tree, while leaf nodes always mark the end of a branch, hence the name.

Action nodes need to be defined by the user and can be either low-level actions like "move forward" or more complex like "fly towards closest peer". They can return either "Running" when an action is currently carried out, "Success", if it has been successfully completed, or "Failure" if not. One may readily implement action nodes which always return "Running" without a stopping criterion. However, if all actions only return "Running", the BT is functionally equivalent to a simple if/else decision tree, which drastically reduces its expressivity.

Condition nodes read variables from a so-called blackboard, and compare them between each other or with respect to pre-set constants. If a condition is true, the node returns "Success", if not it returns "Failure". What information is contained in the blackboard is also to be decided by the user. Actions and conditions are constrained by what the robot can physically sense and carry out, but also by the capabilities of the simulation environment. Both condition and action nodes are defined by a qualitative description, like "Turn" or "Path clear?" that answer the question "What is to be done?", but they can also have one or more quantitative parameters like "turn_rate: 0.2" that define "How is it to be done?".

Composite nodes on the other hand only take shape of two logic operators: firstly, sequence nodes execute all their child nodes in a sequence until they return "Success" or "Failure".

[6] https://www.bitcraze.io/products/loco-positioning-deck/

[7] https://github.com/luptmoor/flapper-firmware/tree/teensydeck_behaviour_tree_uwb

[8] https://github.com/sstroobants/VL53L8CX_Teensy

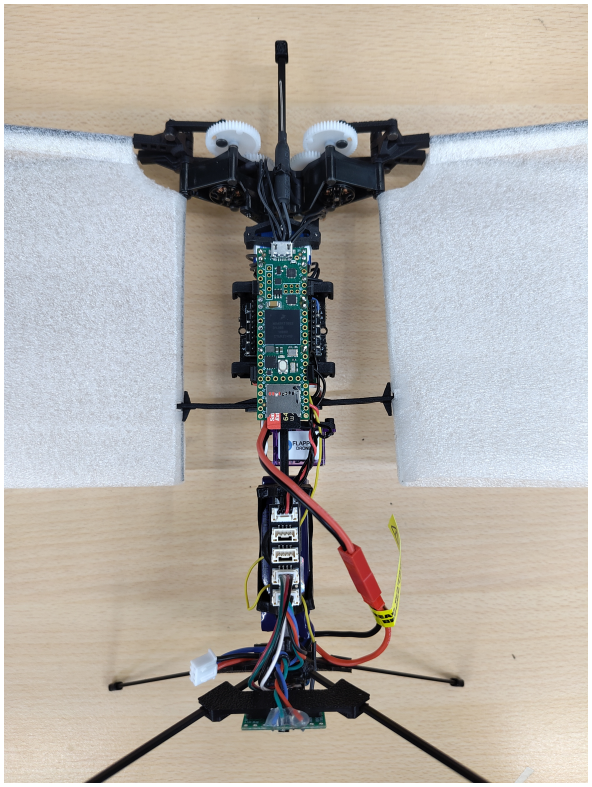


Figure 3: Dorsal view of Flapper. The Teensy 4.1 can be seen in the centre between the wings in green. At the bottom, the outputs from the bottom and forward ToF sensors are joined on a I2C bus.

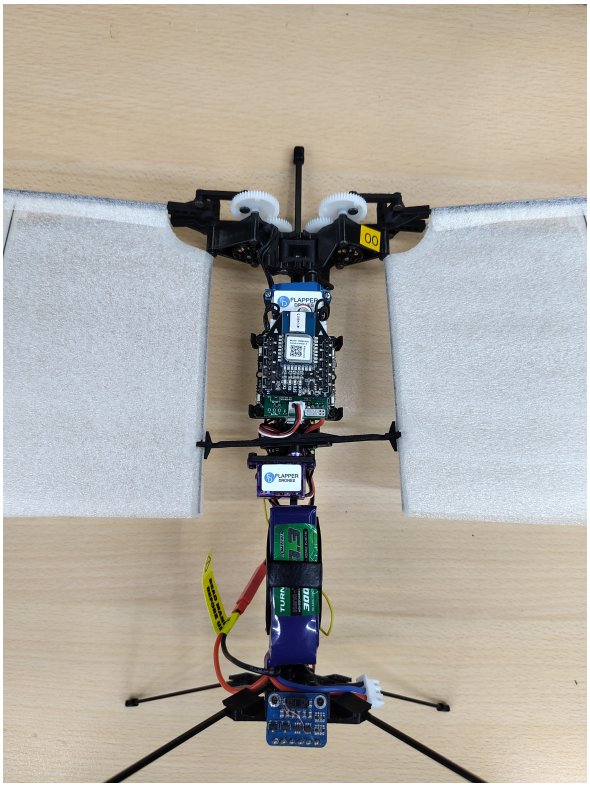


Figure 4: Ventral view of Flapper. The Loco deck can be seen in the centre between the wings in black/blue.

If the child returns "Success", it moves on. If it returns "Failure", the sequence is stopped and the sequence node returns "Failure", too. A sequence node only returns "Success" when *all* their child nodes return "Success". Sequence nodes are visualised with a right-pointing arrow in a rectangular node.

Selector nodes are the opposite of sequence nodes: they return "Success" if *any* of their children return "Success" and move on when their currently active child returns "Failure". Selector nodes are visualised by a question mark in a circular node. An example behaviour tree for the greenhouse exploration task can be seen in Figure 5.

As the main goal of the BTs in this work is motion planning, most their actions contain changing the setpoints for forward speed $V_{x_{\text{cmd}}}$, vertical speed $V_{z_{\text{cmd}}}$ and yaw rate r_{cmd}. Flying backwards and sidewards is not permitted, since the drone only holds a ToF distance sensor on the front, and thus can only avoid obstacles in the direction it is facing.

3.3 Genetic Programming of Behaviour Trees

Evolutionary algorithms like Genetic Programming (GP) can outperform human-made intuitive designs, especially in complex situations like multi-agent coordination in 3D. Hence, GP is used to find a BT that effectively guides the Flappers through the greenhouse. In the first step of GP, BTs need to be randomly generated. The maximum tree depth was set to 4, and the maximum number of child nodes that a composite node can have is set to 4, too. Thus, in total $4^4 = 256$ nodes can be used. For every potential child node of a composite node, there is a 50% chance of being a leaf node and a 25% chance of being another composite node. Hence, there is also a 25% chance that no node is added. In the case of leaf nodes, action and condition nodes have equal chances, just like sequence or selector for composite nodes. The first node in the tree, the root, is always a composite node. Leaf nodes are chosen randomly from the set of user-defined leaf nodes, which are explained in the next section. Their parameters are randomly chosen from a user-defined range that is deemed useful for that particular parameter.

A generation consists of 30 BTs. Each BT is evaluated by simulating a swarm of 5 Flappers that follow the BT's mapping from sensory input to action. Upon collision with the walls, obstacles or other drones, a drone is removed from the simulation. The simulation is terminated when there is only 1 Flapper left or 300 simulated seconds have passed. At the end, a score is calculated based on how much of the environment has been collectively discovered in 3D. To that end, the

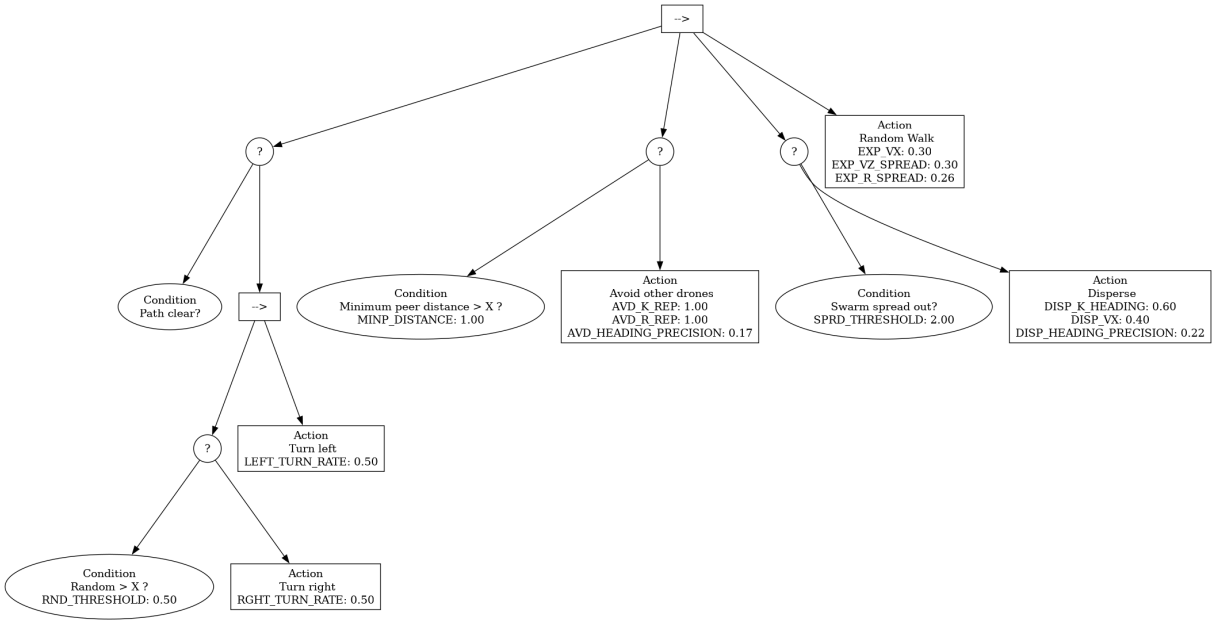


Figure 5: Manually designed behaviour tree with composite and leaf nodes including their parameters. The tree has a depth of 4, as defined by the longest chain. Node parameters are listed in capitals. Read from top to bottom, left to right.

domain is discretised in cubes of size $0.50\,\mathrm{m}$. The score is then equal to the fraction of cubes visited, which lie below 2.00 m. This is to prevent the agents from learning to exploit the trivial solution of simply flying in the obstacle-free space above the crops. Every BT is simulated in 3 different randomly generated greenhouse environments. Their final score is the average score of all 3 simulation runs.

Having simulated all BTs of a generation, the fittest 2 BTs are directly admitted to the next generation (elitism). The remainder is chosen by tournament selection with tournament size 5 [32]. 80 % of the new generation are then subject to crossover: a random node in the tree is selected and the branch that is formed by its child nodes is replaced by a random branch of another member of the new generation. At last, 20 % of the new generation is subject to mutation: in 80 % of the cases, a random leaf node is re-instantiated as a new leaf node of random type and with random parameters (micromutation). In 20 % of the cases, a random composite node is re-instantiated with new child nodes, up to the maximum tree depth (macromutation). The new generation obtained after all these steps is then evaluated again by simulation, and the cycle is repeated 160 times.

3.4 Manually Designed Behaviour Modules

While composite nodes provide an efficient and scalable switching logic, it is the leaf nodes that actually define what sensor inputs to consider and what actions to take. With the goal of fast, coordinated and collision-free greenhouse exploration in mind, the following actions and conditions were implemented. An overview of the numerical parameters for each node is given in Table 1.

Random Walk [27] The drone performs a random walk with constant $V_{x_{\text{exp}}}$, a vertical velocity that randomly varies between $\pm\Delta V_{z_{\text{exp}}}$, and a yaw rate that varies between $\pm\Delta r_{\text{exp}}$. Always returns "Running".

Disperse [27] The drone moves away from the geometric centre of the swarm. It calculates a target heading ψ_{cmd} and then sets a yaw rate command to $r_{\text{cmd}} = K_{\text{disp}}\,(\psi_{\text{disp}} - \psi)$. Once the heading error is within the precision range $\pm\epsilon_{\text{disp}}$, the forward velocity is set to V_{disp}. Always returns "Running", unless drone is the last one active, then "Success".

Turn The drone halts and turns with a yaw rate of r_{turn}, which can be either left or right, depending on the parameter. It returns "Running" until a full 360° turn since the time this node was initiated has been completed. Then it returns "Success".

Avoid The drone avoids nearby drones with the method of an artificial potential field (APF) [33]. When peer drones are within a certain radius r_{APF}, they generate a specific force component in all three principal axes. For instance, in the x-direction $\Delta f_x = K_{\text{APF}}\frac{\Delta x}{r^2}$, where r is the direct distance between the peers. The direction of the resulting force is then used as reference heading ψ_{APF}. A yaw rate command

$r_{\text{cmd}} = K_{\text{APF}}(\psi_{APF} - \psi)$ Once the heading error is within the precision range $\pm\epsilon_{\text{APF}}$, the forward velocity is set to V_{APF}. Always returns "Running", unless drone is the last one active, then "Success".

Vertical The drone halts and moves vertically with $V_{z_{\text{vert}}}$, which can be either up or down, depending on the parameter. It returns "Running" until it has reached either the ceiling or the floor. Then it returns "Success".

Brake The drone halts completely. Returns "Success" when velocity has reached zero. Until then it returns "Running".

Send Message [27] The drone sets its binary message flag from 0 to 1. Returns "Success" immediately.

Next to these actions, the following condition nodes are implemented. Many conditions were conceptualised to go along with certain actions, known as *implicit success conditions* [23].

Path clear? Returns "Success" if central 50% of ToF-sensor readings contain no obstacle closer than 1.00 m. Otherwise it returns "Failure".

Peer distance $< r_{\text{peer}}$? Returns "Success" if at least one peer drone is closer than r_{peer}. Otherwise it returns "Failure".

Swarm spread? Returns "Success" if the closest third (rounded down) of active agents are beyond a threshold d_{spread} away from the agent. Otherwise it returns "Failure".

Message received? [27] Returns "Success" if at least one peer has set their binary message flag to 1. Otherwise it returns "Failure".

Random condition Returns "Success" if a random number from a uniform distribution between 0 and 1 is larger than a threshold β. Otherwise it returns "Failure".

Timer condition Returns "Success" if the recorded time since takeoff is greater than T_{timer}. When this node returns "Success", the recorded time is reset to zero. Otherwise it returns "Failure".

Table 1: Overview of parameters for manually designed action and condition nodes.

Module	**Parameter**	**Range**
Random Walk	$V_{x_{\text{exp}}}$	$0.1 - 1.0\ \text{m/s}$
	$\Delta V_{z_{\text{exp}}}$	$0.1 - 0.5\ \text{m/s}$
	Δr_{exp}	$0.1 - \frac{\pi}{2}\ \text{rad/s}$
Disperse	K_{disp}	$0.1 - 5.0$
	ϵ_{disp}	$0.0 - 45.0^o$
	V_{disp}	$0.1 - 1.0$
Turn	r_{turn}	$-\frac{\pi}{2} - \frac{\pi}{2}\ \text{rad/s}$
Avoid	K_{APF}	$0.1 - 10.0$
	ϵ_{APF}	$0.0 - 45.0^o$
	d_{APF}	$0.5 - 10.0\ \text{m}$
Vertical	$V_{z_{\text{vert}}}$	$-0.5 - 0.5$
Brake	–	–
Send Message	–	–
Path clear?	–	–
Peer distance $< d_{\text{peer}}$?	d_{peer}	$0.0 - 8.0\ \text{m}$
Swarm spread?	d_{spread}	$0.0 - 8.0\ \text{m}$
Message received?	–	–
Random condition	β	$0.0 - 1.0$
Timer condition	T_{timer}	50–1500 ticks

3.5 Neuroevolution of Behaviour Modules

Besides the manually designed behaviours, more high-level behaviours controlled by neural networks were considered. These neural behaviours were thought to compliment the manually designed nodes in a behaviour tree and expected to be especially useful for swarm coordination tasks based on relative positions and peer-to-peer communication. Neural networks' ability to approximate very complex functions seem a promising means to enable useful emergent behaviours of the swarm, i.e. the ability to perform more complex tasks as a group than the sum of individuals could ever achieve.

To that end, two neural modules were conceptualised: one for exploration, in which the agents could learn to spread the domain between each other rather than each blindly discovering it on their own, and one for exploitation which lets the agents repetitively visit the stored fruit locations from the discovery phase. The output of these neural network are setpoints for forward velocity $V_{x_{\text{cmd}}}$, vertical velocity $V_{z_{\text{cmd}}}$ and yaw rate r_{cmd}, as well as a communication message to the peers, which are all limited to a acceptable ranges with a sigmoid function.

The neural exploration mode works with the peer position estimates in the agent's local frame and also processes the scalar message from every peer. The message is limited between -1 and 1 and could be used by the drones to indicate attraction or repulsion. In reality, individual agents can fail or not respond to the UWB-based positioning queries. This must be accounted for in the network architecture.

The neural exploitation mode works with stored 3D positions of discovered fruit. Similar to the exploration mode, there is a fourth information which weighs the respective position, namely the time since last visit of that fruit. Both visit times and positions are broadcasted across the swarm. While these messages may not reach all drones at all times, the implementation shall allow the drones to still use the neural networks to their most up to date knowledge.

Both networks need to be invariant to the number of input data points. Hence, for both networks we propose the neural architecture of the Weighted DeepSet, inspired by the work of [34] [34]: all relative 3D positions are initially encoded in a 16-dimensional latent space. The encodings are then weighted by the correspondent message for the exploration case, or time since last visit for the exploitation case. At last, the weighted encodings are summed and mapped to the four-dimensional output, which consists of the three motion commands and the message to the peers. An overview is given in Figure 6.

The weight-sharing for every data point also drastically reduces the parameter count and execution time, which is essential for implementation on board of the microcontroller and simulation. It must furthermore be acknowledged that the proposed architecture is partially recursive, i.e. the transmitted peer message determined by the network is amplified by the received peer messages. This feedback loop can lead to decay, blow-up or oscillations of the neural activations, which likely harms the performance of the swarm. To hedge against this, instead of weighting the inputs, the message could be used as input just like the position, or the message could be left out of the neural network altogether.

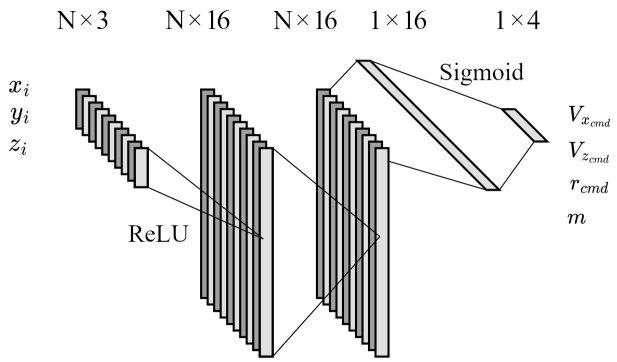


Figure 6: Weighted DeepSet architecture chosen for neuroevolution. The input is a set of 3D points in the agent's body frame, which is then encoded to 16 dimensions and lastly aggregated by weighing the encodings and summing them. The output is a 4D vector obtained by parsing the weighted sum through a dense layer and applying a sigmoid. In total this architecture has 404 trainable parameters.

As the networks shall be given room for counter-intuitive emergent behaviours, there is no behavioural reference given as training data. Instead, parameters shall be set by neuroevolution, a type of unsupervised learning that optimises utility with respect to a preset objective function. For the exploration network, this objective function is equal to the fraction of discovered fruit in the environment, for the exploitation network, the objective is equal to the negative average time a fruit had to wait for an inspection. Neuroevolution is implemented via covariance matrix adaptation evolutionary strategy (CMAES) [35], a very efficient abstract algorithm that works well for low-dimensional parameter spaces, like the 404 parameters of this architecture.

3.6 Parameter Fine-Tuning

Finally, next to tuning the parameters of the neural networks, CMAES also serves to fine-tune the numerical parameters of the BTs obtained through genetic programming and a manual benchmark. CMAES was chosen due to its high efficiency and suitability for optimisation of low-dimensional, continuous variables. To that end, all parameters that can be seen in Figure 5 are treated as independent parameters with initial 3-σ environments set to the ranges seen in Table 1.

4 Experimental Setup

4.1 Simulation Environment

All evolutionary approaches need a simulation environment to evaluate the performance of a solution with respect to the goal of greenhouse exploration. The greenhouse is simulated in a 15.00 m × 8.00 m environment with a ceiling height of 3.00 m. To represent crop rows, 3-5 cuboids with a breadth of 0.60 m, length of 12.90 m and a height of 1.80 m are placed along the length of the greenhouse, leaving 10% of the length free for the drones to be placed. The drones themselves are represented by spheres with radius 0.25 m.

For simplification, full position, attitude and velocity knowledge of the drones is assumed. State estimation is explicitly left out. Moreover, the ToF-sensor array is assumed to be perfectly reliable in providing distances within the field of view, while in reality, the chosen sensor model requires significant filtering and postprocessing.

The simulation is accelerated with *numba*, a library that provides decorators allowing just-in-time (JIT) compilation of Python code [36]. JIT compilation adds a little overhead processing time at the start of the simulation, however significantly reduces processing time during repeated function calls in the simulation loop. However, this is only possible when adhering to certain requirements, i.e. a significant range of the Python repertoire like classes and dynamically typed structures become unavailable.

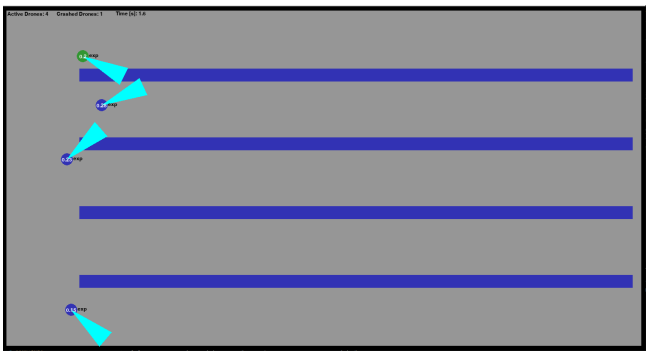


Figure 7: Screenshot of simulated environment shortly after a simulation run has started. The drones are shown in dark blue on the left, with their central field of view visualised in cyan. The crop rows are also shown in dark blue.

For verification, the environment can be visualised using the *PyGame* library. A screenshot of the visuals is shown in Figure 7. To increase evolution speed, visuals are typically turned off. Moreover, the current state of the active behaviour tree, which nodes are running, have succeeded or failed, can be visualised for one drone using *graphviz*. This drastically increases simulation time though.

The Python code to run the simulation and evolution is available on GitHub[9].

[9] https://github.com/luptmoor/flapper_greenhouse_swarm

4.2 Flapper Dynamic Model

Despite multiple efforts of system identification for the Flapper, there currently exists no comprehensive dynamic model. As the drones will navigate with the three control inputs $V_{x_{\text{cmd}}}$, $V_{z_{\text{cmd}}}$ and r_{cmd}, thus limiting sideways movement as much as possible, the longitudinal dynamics are most important. The most recent longitudinal model is from 2019 [37]. This nonlinear model has been replicated in numba-Python and fitted with a Runge-Kutta solver.

The yaw dynamics were assumed to follow a simple first-order lag with time constant τ_r = 0.05 s. A step response is shown in Figure 8.

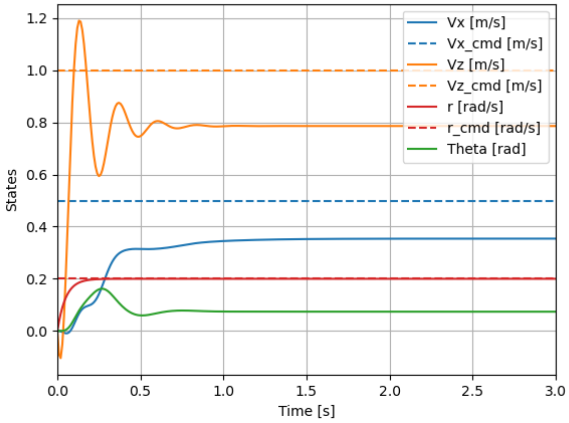


Figure 8: Step response of simulated longitudinal and yawrate model with $V_{x_{\text{cmd}}} = 0.5, V_{z_{\text{cmd}}} = 1.0$ and $r_{\text{cmd}} = 0.2$.

It can be seen that there is a significant steady-state error for the velocity commands. This is in line with observations of the real Flapper. Furthermore, as this is a closed-loop response, performance depends on the controller gains implemented on the drone. While the PD gains for the pitch angle θ were taken from the model of Kajak et al., the velocity gains were manually tuned to $K_x = 0.2$ and $K_z = 80$. Moreover, it can be observed that vertical and forward velocity are coupled, which explains why the pitch oscillations are propagated to the vertical velocity.

4.3 Real World Tests

To validate performance of the obtained solution in the real world, the best evolved solution is implemented with a custom BT framework for the Flapper firmware, using the app layer. The created app allows flying every Flapper in the swarm manually with a remote transmitter and then switching

Figure 9: Photo of the initial test setup in the CyberZoo. The Flappers are spaced 1.50 m apart at first and facing towards to aisles that are supposed to mimic crop rows in a greenhouse.

to autonomous flight mode. While flying autonomously, the implemented behaviour tree commands the Flappers motion and peer-to-peer communication, but motion commands can be overridden with the manual remote for safety.

Two Flappers are deployed in the CyberZoo test facility at TU Delft's Aerospace Engineering faculty. Using mobile walls, a simple maze is constructed that mimics the row-like structure of the simulation environment. The drones are spaced 1.5 m apart, which is the pre-defined relative position for the relative localisation EKF. An overview of the initial scenario can be seen in Figure 9.

The relative position estimates are logged on-board on a micro SD card. Absolute positions and attitude are logged using the Optitrack motion capture system. During post-processing, the relative position beliefs are reconstructed in the absolute frame using affine transformations, as seen in Equation 1.

$$\begin{aligned} \hat{x}_A &= x_B + \hat{x}_{AB}\cos\psi_B - \hat{y}_{AB}\sin\psi_B \\ \hat{y}_A &= y_B + \hat{x}_{AB}\sin\psi_B + \hat{y}_{AB}\cos\psi_B \end{aligned} \tag{1}$$

5 Results

5.1 Neuroevolution Results

To understand how well the neuroevolution modules can fulfil the goal of obstacle-free volume exploration, the average and maximum fitnesses are plotted versus generation count, as shown in Figure 10. Overall, the best solution reaches a fitness of around 60%. That means, 60% of the voxelised 3D domain is visited by any of the agents. For the neuroevolution runs, collisions with crops and walls were disabled, as the exploration nodes were supposed to focus on domain exploration and peer-to-peer collision avoidance. After around 80 generations, the population converges to a local optimum, after which no significant improvement could be observed. It was found that using messages in the interval $-1 \leq m < 1$ as weights leads to numerical instability, since basically all encodings are multiplied by a number with magnitude smaller than 1, and thus decay over time. For this reason, the allowed message range was shifted to $2 \leq m < 3$. However, this lead to binary messages switching between 2 and 3, again due to numerical instability, this time due to exponential growth.

The evolved behaviour learned this way can be described as follows: the drones engage in rapid but nearly constant turning and use their communication ability to tell their peers when to move forward. They move forward when receiving a "High" message of 3, and stop when receiving a "Low" message of 2, with a period roughly synchronised to their turn rate. This way the agents perform a translating circular sweep of the domain with a slight beneficial noise due to imperfect synchronisation. Inter-drone collision avoidance has not been proven to work, as hoped for.

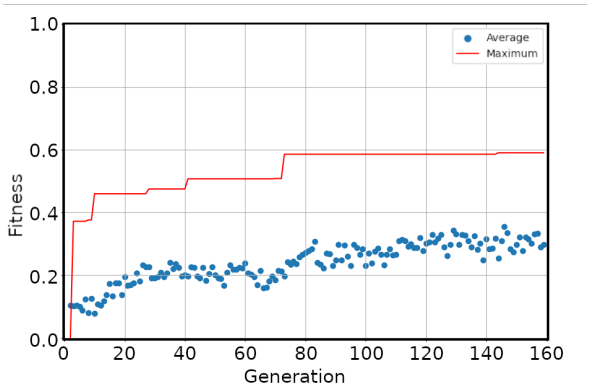


Figure 10: Improvement of fitness score versus increasing number of generations for the message-weighted DeepSet of peer positions. The mean fitness of a generation is shown by a blue dot, the total best-performing solution up until a generation is shown as the red solid line.

To prevent mentioned numerical instability, two follow-up experiment were conducted: one using the peer message as fourth dimension of data next to the three relative position coordinates, and one without communication at all. The weighting of encodings is thus left out in both cases. Results are shown in Figure 11 and Figure 12, respectively.

With communication, the algorithm quickly converges to a local optimum of similar fitness, around 60%, interestingly though with a smaller average score. Qualitatively, the apprehended behaviour is very similar to the one described for the previous experiment.

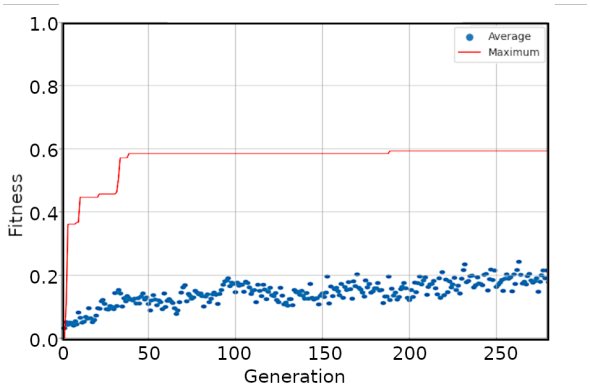


Figure 11: Improvement of fitness score versus increasing number of generations for the unweighted DeepSet of peer positions and peer message. The mean fitness of a generation is shown by a blue dot, the total best-performing solution up until a generation is shown as the red solid line.

Although in previous experiments communication-free solutions could be occasionally observed among the high-performing individuals, completely restricting communication leads to a reduced maximum fitness overall, namely 50% compared to 60% with communication. Intuitively, this is to be expected.

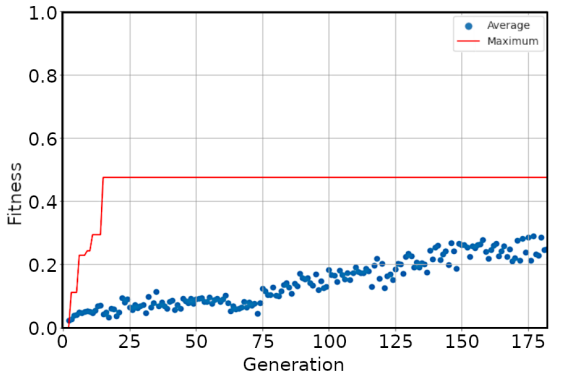


Figure 12: Improvement of fitness score versus increasing number of generations for the unweighted DeepSet of peer positions only. The mean fitness of a generation is shown by a blue dot, the total best-performing solution up until a generation is shown as the red solid line.

5.2 Genetic Programming Results

Next, the genetic programming results are studied to see how well the programme can assemble BTs for volume exploration, this time including obstacles and wall collisions. Figure 14 shows the fitness over generation count. The best solution manages to explore 8% of the domain, which in practice means that most of the time the agents do not manage to travel between crop rows. Compared to the the neural exploration mode, the fitness is significantly less (8% versus 60%), as collisions with walls and plants prevent drones from discovering more parts of the domain. Even if a successful collision avoidance strategy is learned, exploration in cluttered environments within a given time is significantly harder than without obstacles.

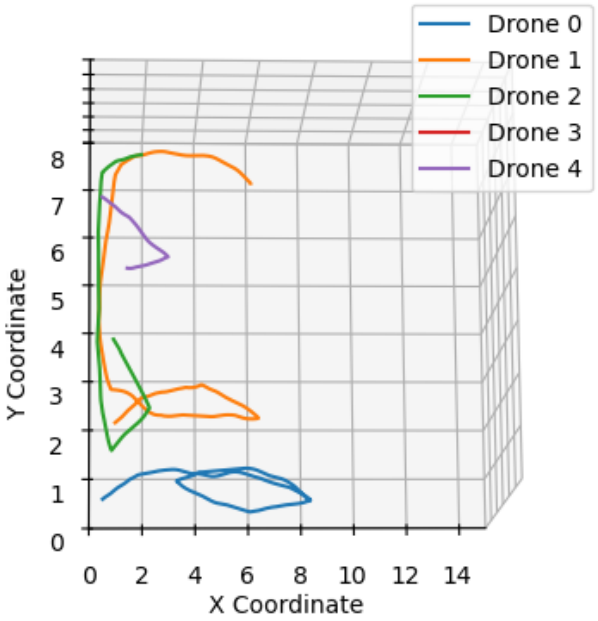
Due to the random manner of BT generation during genetic programming, not all nodes can be reached during execution, and some may be redundant. In both cases, the fitness is not negatively affected by this, so these evolution artefacts typically persist. Due to BTs' human readability, these artefacts can be removed manually after evolution, without adding human bias to the solution. The pruned, absolute best BT obtained from GP is shown in Figure 15.

Despite its differing structure, the evolved BT exhibits similar traits to the manually designed one in Figure 5. Most important - and thus leftmost in the tree - is the "Path clear?" condition which decides whether or not moving forward is safe. If that is the case, both trees go on to explore the domain. If the path is not clear, then both trees resort to turning. While for the manual tree it was decided to turn randomly left or right, the evolved tree only turns right. There is no clear benefit in either solution. The biggest difference between evolved and designed tree is that the evolved tree utilises neither peer-to-peer communication, nor localisation, which makes the ultra-wideband module obsolete.

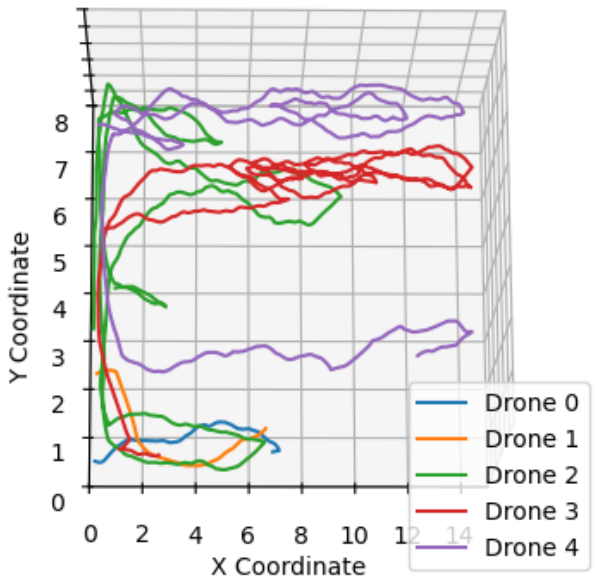
5.3 Parameter Fine-Tuning Results

To improve exploration performance further, CMAES was applied to fine-tune the continuous parameters of the evolved tree from Figure 15, while keeping its structure unchanged. This has increased fitness from 8% to 36%, due to significantly more travel between crop rows. The progress for that is plotted in Figure 16 and the fine-tuned evolved solution is shown in Figure 17. The tuning can be described as follows: The random condition is essentially bypassed, as it always returns "Failure" and is thus skipped by the selector. This makes sense, as there is no benefit if a drone just waits when encountering an obstacle. Moreover, the exploration velocity was slightly increased with more vertical variation and less horizontal variation. The latter likely leads to longer distances travelled without encountering obstacles, while the former allows the agents to occasionally fly over the rows, improving exploration. Also, the avoidance turn rate has been increased to save time. Interestingly, the direction was switched from right to left turns. This is likely a coincidence without a clear benefit.

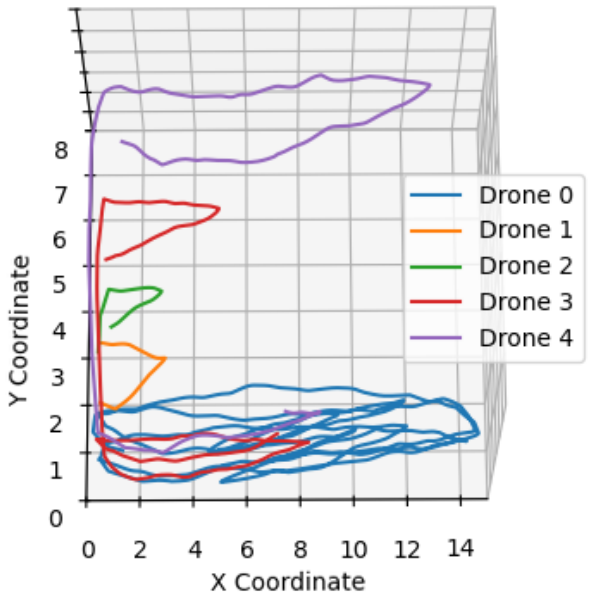
In a second run, CMAES was also applied to fine-tune the manually designed tree from Figure 5, improving its fitness from 25% to 37%. The progress is shown in Figure 18 and



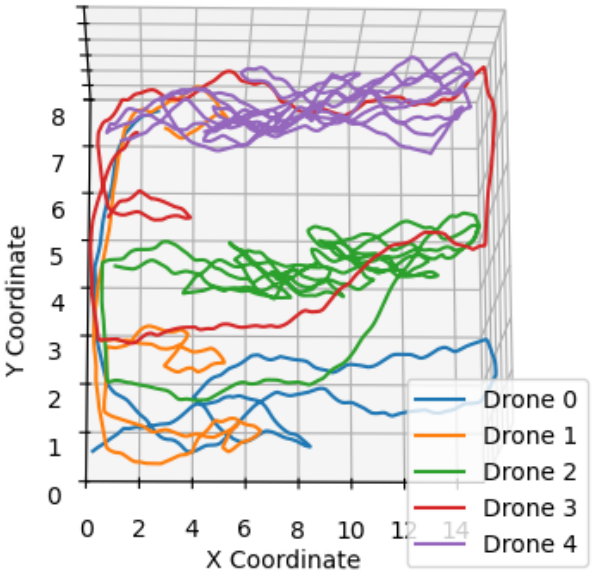
(a) Trajectory flown by BT obtained through genetic programming.



(b) Trajectory flown by manually designed BT.



(c) Trajectory flown by CMAES-tuned BT from genetic programming.



(d) Trajectory flown by CMAES-tuned manually designed tree.

Figure 13: Comparison of the trajectories of the four obtained solutions. The top row (a) and (b) shows the result from genetic programming compared with the manually designed solution. The bottom row (c) and (d) shows the trajectories of both solutions after fine-tuning with CMAES. All spatial dimensions are given in metres.

the fine-tuned manual tree is visualised in Figure 19. Compared to the manually selected parameters, the main difference is that drone avoidance is at the core of the behaviour now. This can be seen by the very large distance of 9.55 m that is now required between the drones. Interestingly, the threshold for the related but distinct "Swarm spread?" condition is below the 9.55 m, which renders this condition and the following "Disperse" node unreachable. For the rare cases of "Random Walk", the forward velocity and vertical deviation are increases, like for the BT obtained by GP. The "Turn" ac-

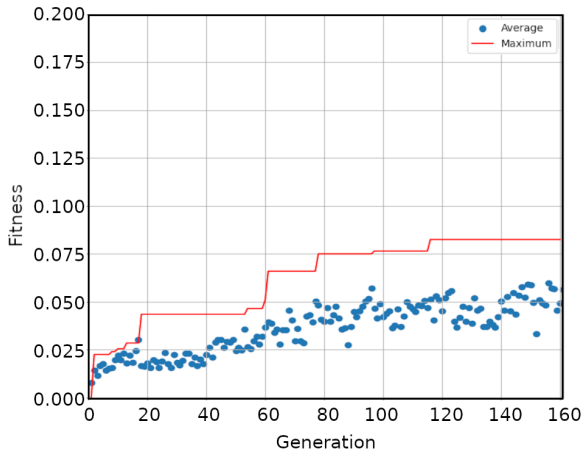


Figure 14: Improvement of fitness score versus increasing number of generations for genetic programming. The mean fitness of a generation is shown by a blue dot, the total best-performing solution up until a generation is shown as the red solid line.

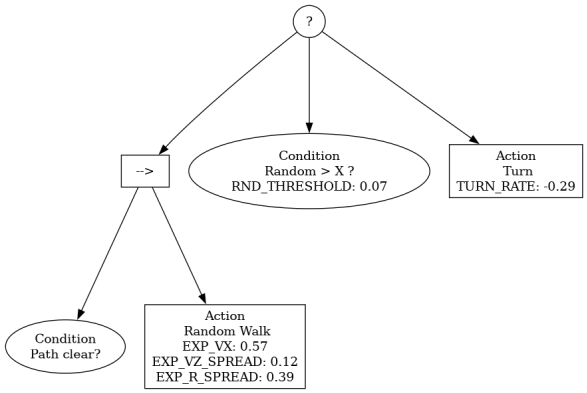


Figure 15: Pruned evolved behaviour tree from genetic programming. Read from top to bottom, left to right.

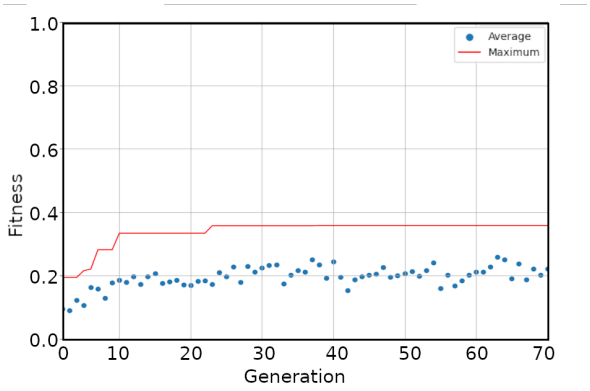


Figure 16: Improvement of fitness score versus increasing number of generations for CMAES fine-tuning of the evolved BT structure. The mean fitness of a generation is shown by a blue dot, the total best-performing solution up until a generation is shown as the red solid line. A maximum fitness of 36% is reached.

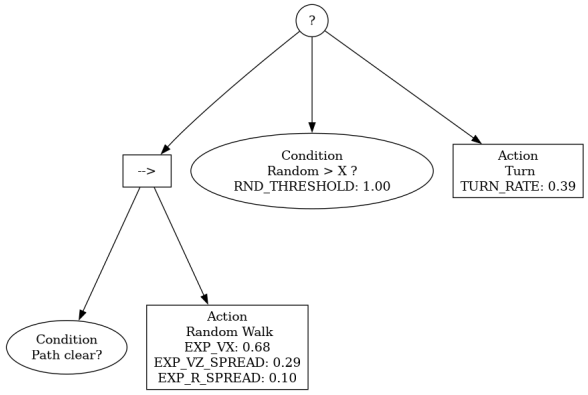


Figure 17: Fine-tuned evolved behaviour tree from genetic programming. Read from top to bottom, left to right.

tions in response to a blocked path is also made more rapid, just like for the GP tree, with a slight bias for left turns, as caused by the increased random threshold. This is in opposition to the bias for right turns in the GP tree, which indicates that this decision is arbitrary.

An overview of the simulated trajectories of all four solutions, manually and evolved, with and without fine-tuning is given in Figure 13.

5.4 Real World Test Results

The evolved, fine-tuned BT was implemented on board of two Flappers and tested in the Cyberzoo test cage to investigate how well a simulated solution performs in reality. Trajectories can be seen in Figure 20. Unfortunately but as expected, the evolved solution did not transfer to the real world as-is. While both Flappers managed to navigate through their corridors for a few seconds, when they reached an orthogonal wall, they did not manage to successfully turn and both crashed.

This is likely because of the low-quality on-board velocity estimation that was not simulated. While the agents correctly recognise the wall, command a halt and initiate a turn, their real velocities probably did not entirely reach zero, so they kept on drifting and crashed into the wall. This is more significant when encountering orthogonal obstacles than simply adjusting their course to keep away from side walls.

Another major result of the real-world test is the recording of relative localisation data on a Flapper during autonomous flight of both agents. Due to erroneous yaw data, the trajectory as seen from the peer drone could only be reconstructed for one drone. Looking at the relative trajectory, we see that although the initial position was precisely known, the estimate drifted away and needed some time to converge again.

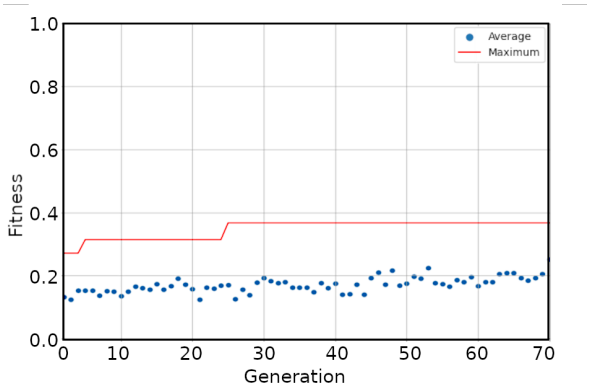


Figure 18: Improvement of fitness score versus increasing number of generations for CMAES fine-tuning of the manually designed BT structure. The mean fitness of a generation is shown by a blue dot, the total best-performing solution up until a generation is shown as the red solid line. A maximum fitness of 37% is reached.

This is because for this setup, the drones remained on the ground for a few seconds with the relative EKF already running. Because both drones during this time had a slightly non-zero pitch and roll angle, the on-board velocity estimator predicted a non-zero relative velocity between the drones [18], which drifted away the relative position estimate from ground truth.

Once taken off, the on-board velocity estimate is correct again and the relative positions get closer to reality. In the current firmware version, this issue has been addressed by only using the attitude-based velocity estimation if the drones are actually flying. On ground, velocities are reset to zero. Moreover, the bottom ToF height estimates still need to be used in the relative EKF, as it currently only relies on barometer estimates. This will likely reduce localisation error further.

The drones thus demonstrate a coarse understanding of each other's position during autonomous flight - notably while being separated by an opaque wall and without external infrastructure - which is available to the reactive planning strategy in real-time.

6 Discussion

Genetic Programming allowed for the automatic creation of a behaviour tree that enables a swarm of Flappers to explore around 8% of a simulated greenhouse environment, nearly collision-free (Figure 14). Fine-tuning the continuous parameters of the evolved tree increases the exploration to around 36% (Figure 16). However, a manually designed tree achieves to explore 25% of the domain, 37% with fine-tuning (Figure 18).

This gives us two insights: firstly, the evolved tree structure has a similar performance potential as the human-made tree (36% and 37% respectively), but the chosen GP approach fails to also find a suitable parameter set (only 8%). Secondly, human intuition is able to find acceptable parameter sets, as seen by the 25% un-tuned fitness score, but is nevertheless outperformed by the CMAES fine-tuning (37%). Thus, evolution trumps the manual solution by around 50%.

Behaviour Trees have convinced their users with their scalability, modularity and human-readability. Compared to Finite State Machines, this allows human-designed BTs to govern far more complex behaviours by handling the problem of combinatorial overload more efficiently. However, in the context of genetic programming, combinatorial overload is still a problem, as only specific combinations of nodes result in desirable behaviour. The more fine-grained a behaviour tree can be, the harder it is for genetic programming to find these combinations. This is exacerbated by the reliance on so-called *explicit success conditions* [23], as has been the case in this research. Explicit success conditions are very useful when paired with actions that can eventually fulfil these conditions. One example is "Path clear?" and "Turn". The "Turn" action is likely to eventually make the "Path clear?" condition true. This has been used in the manually designed tree in Figure 5. What is obvious to a human designer, however takes many generations of guided search in genetic programming.

So in order for behaviour trees to develop useful complex behaviours, a lot of patience and compute is needed. For smaller behaviour trees, a human design can be implemented within minutes, so genetic programming is not strictly needed. And if a simple behaviour is designed by a human user, FSMs are a lot more intuitive, so they should be the first choice, as seen in prior work [15, 21].

This is especially true, because in this research, most leaf nodes were designed by a human user. In the design process, the user typically imagines a potential use case for a node. If the same user is then asked to assemble a manual BT from these nodes, it is likely that this assembled tree is very close to the global optimum of all possible combinations. As genetic programming has a tendency of converging to local optima, it is unlikely that GP can find a better combinations with human-made leaf nodes than the human that made these leaf nodes initially. This is also supported by the GP results (36% GP versus 37%).

To shield solutions further from human bias, neural behaviour nodes were explored. These neural behaviour nodes nevertheless were subject to a human-designed objective function and human designed architecture. Although the neural behaviour nodes were not selected by the human designer or the GP algorithm, neural nodes are believed to be an important enabler for emergent behaviours in the swarm, as they offer sufficient complexity for exhibiting useful properties that a human designer would likely not be able to create intuitively. Thus, in future research, a combination of different neural nodes with different architectures could be used to en-

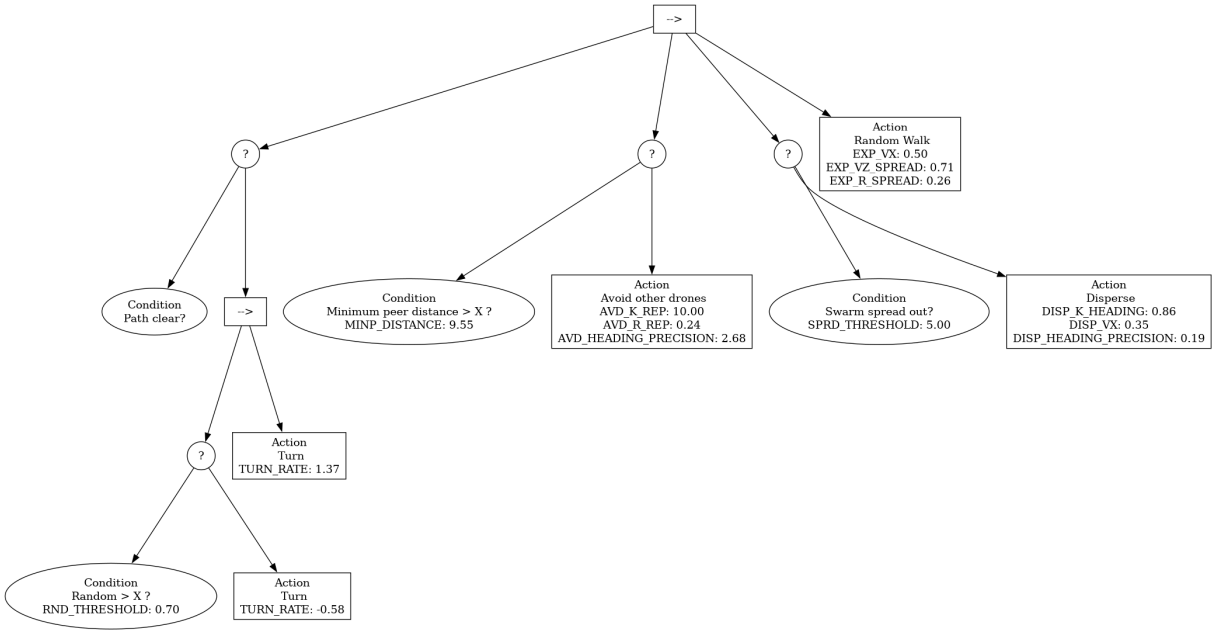


Figure 19: Fine-tuned manually designed behaviour tree. Read from top to bottom, left to right.

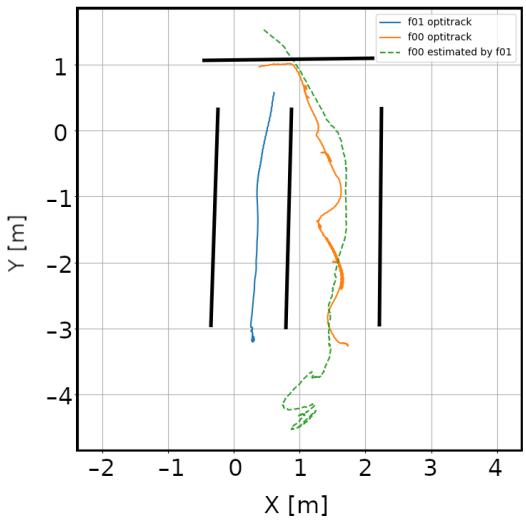


Figure 20: Trajectories of two Flappers recorded by the Optitrack motion capture system. Setup walls are shown in thick black lines. Due to faulty data recordings, only one trajectory estimate could be reconstructed in post-processing.

able more complex behaviour. The difficulty lies in letting the behaviour tree define objectives for these neural nodes without introducing human bias. This step could be comparable in magnitude to the shift from human-designed filters to convolutional neural networks in image classification tasks [38]. Just as CNNs displaced hand-crafted feature pipelines, evolved neural nodes may surpass manually engineered behavioural primitives.

Furthermore, it is believed that the volume exploration task was not complex enough for significant emergent behaviours to be realised, which is a potential reason why communication and relative localisation did not appear in the evolved tree, and only showed marginal benefits when selected by human designer. In fact, the disperse function has even been removed again by the fine-tuning algorithm. This is in line with the findings of Jones et al. [13] and Kuckling et al. [27].

7 Conclusion

In this work, a simulated greenhouse environment was created to explore different artificial evolution methods with the goal of finding a reactive planning strategy based on the sensor availability described above. Genetic programming was employed to find a suitable tree configuration to maximise collective volume exploration. The evolved behaviour tree was fine-tuned with covariance matrix adaption evolutionary strategy (CMAES) and compared to a manually designed tree that was also fine-tuned with CMAES. Fine-tuning significantly improved performance for both the manually designed and the evolved behaviour tree.

Moreover, hardware modifications for flapping-wing drones were presented with the objective of scalable peer-to-peer communication, relative localisation and obstacle detection. This was made possible, respectively, through the use of an ultra-wideband communication module, as well as time-of-flight (ToF) sensors on the front and bottom sides. The

setup was tested in a real-world environment and has proven to work reliably on the sensing side, but imperfect on the navigation side. A reality gap persists with respect to performance in simulation, and a deployment for real greenhouse monitoring is still far from feasible.

Future work could investigate the optimised placement of sensors, or the addition of further sensors. Moreover, the addition of a camera for a lightweight visual inertial odometry (VIO) suite would be interesting to improve the on-board velocity estimation, which is currently unsatisfactory. Furthermore, the shortcomings in estimation and ToF perception should be modelled in the simulation environment, along with a more accurate dynamic model of the Flapper. The evolution trials can be resumed with more computational resources and a different selection of leaf nodes for the behaviour tree. A larger step could be made by the automatic and bias-free generation of neuroevolution nodes for more fine-grained and complex behaviours. This way, the swarm is more likely to realise complex collective behaviours.

BIBLIOGRAPHY

[1] L. Prause. “Digital agriculture and labor: A few challenges for social sustainability”. In: *Sustainability* 13.11 (2021), p. 5980.

[2] M. Soussi, M. T. Chaibi, M. Buchholz, and Z. Saghrouni. “Comprehensive review on climate control and cooling systems in greenhouses under hot and arid conditions”. In: *Agronomy* 12.3 (2022), p. 626.

[3] A. Rejeb, A. Abdollahi, K. Rejeb, and H. Treiblmaier. *Drones in agriculture: A review and bibliometric analysis*. July 2022.

[4] E. Van Henten, C. Montenegro, M. Popovic, S. Vougioukas, A. Daniel, and G. Han. *Embracing Robotics and Intelligent Machine Systems for Smart Agricultural Applications [From the Guest Editors]*. Dec. 2023.

[5] M. Canicattı and M. Vallone. “Drones in vegetable crops: a systematic literature review”. In: *Smart Agricultural Technology* (2024).

[6] C. Geckeler, S. Ramos, M. Schuman, and S. Mintchev. “Robotic Volatile Sampling for Early Detection of Plant Stress: Precision Agriculture Beyond Visual Remote Sensing”. In: *IEEE Robotics and Automation Magazine* 30 (4 Dec. 2023), pp. 41–51.

[7] G. de Croon. “Flapping wing drones show off their skills”. In: *Science Robotics* 5.44 (2020).

[8] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[9] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067.

[10] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto. “An informative path planning framework for UAV-based terrain monitoring”. In: *Autonomous Robots* 44.6 (2020), pp. 889–911.

[11] J. Westheider, J. Rückin, and M. Popović. “Multi-uav adaptive path planning using deep reinforcement learning”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 649–656.

[12] S. Nolfi. “Power and the limits of reactive agents”. In: *Neurocomputing* 42.1-4 (2002), pp. 119–145.

[13] S. Jones, M. Studley, S. Hauert, and A. Winfield. “Evolving behaviour trees for swarm robotics”. In: *Distributed Autonomous Robotic Systems: The 13th International Symposium*. Springer. 2018, pp. 487–501.

[14] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, F. Mascia, V. Trianni, and M. Birattari. “AutoMoDe-Chocolate: a Method for the Automatic Design of Robot Swarms that Outperforms Humans”. In: *Swarm Intelligence* (2015).

[15] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. de Croon. “Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment”. In: *Science Robotics* 4.35 (2019).

[16] E. Şahin, S. Girgin, L. Bayindir, and A. E. Turgut. *Swarm robotics*. Springer, 2008.

[17] M. Van Der Marel and R. T. Rajan. “Distributed Kalman filters for relative formation control of multi-agent systems”. In: *2022 30th European Signal Processing Conference (EUSIPCO)*. IEEE. 2022, pp. 1422–1426.

[18] S. Pfeiffer, V. Munaro, S. Li, A. Rizzo, and G. de Croon. “Three-dimensional relative localization and synchronized movement with wireless ranging”. In: *Swarm Intelligence* 17.1 (2023), pp. 147–172.

[19] S. Li, M. Coppola, C. De Wagter, and G. C. de Croon. “An autonomous swarm of micro flying robots with range-based relative localization”. In: *arXiv preprint arXiv:2003.05853* (2020).

[20] S. Li, F. Shan, J. Liu, M. Coppola, C. de Wagter, and G. C. de Croon. “Onboard Ranging-based Relative Localization and Stability for Lightweight Aerial Swarms”. In: *arXiv preprint arXiv:2003.05853* (2020).

[21] B. Duisterhof, S. Li, J. Burgués, V. Reddi, and G. de Croon. "Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 9099–9106.

[22] Y. Zhuang, X. Jiang, Y. Gao, Z. Fang, and H. Fujita. "Unsupervised monocular visual odometry for fast-moving scenes based on optical flow network with feature point matching constraint". In: *Sensors* 22.24 (2022), p. 9647.

[23] M. Colledanchise and P. Ögren. *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

[24] A. K. Agogino and K. Tumer. "Unifying temporal and structural credit assignment problems". In: *Autonomous Agents and Multi-agent Systems Conference*. 2004.

[25] G. de Croon, M. Van Dartel, and E. Postma. "Evolutionary learning outperforms reinforcement learning on non-Markovian tasks". In: *Workshop on Memory and Learning Mechanisms in Autonomous Robots, 8th European Conference on Artificial Life, Canterbury, Kent, UK*. 2005.

[26] K. Scheper, S. Tijmons, C. de Visser, and G. de Croon. "Behavior trees for evolutionary robotics". In: *Artificial life* 22.1 (2016), pp. 23–48.

[27] J. Kuckling, V. van Pelt, and M. Birattari. "AutoMoDe-Cedrata: Automatic Design of Behavior Trees for Controlling a Swarm of Robots with Communication Capabilities". In: *SN Computer Science* 3 (2 Mar. 2022). ISSN: 2662-995X.

[28] S. Jongerius, M. Straathof, G. van der Veen, W. Roos, P. Moelans, R. Lagarde, A. Kacgor, C. Heynze, A. Ashok, K. de Clercq, et al. "Design of a flapping wing vision-based Micro-UAV". In: *Design synthesis exercise, Faculty of Aerospace Engineering, TU Delft* (2005).

[29] G. de Croon, M. Groen, C. De Wagter, B. Remes, R. Ruijsink, and B. van Oudheusden. "Design, aerodynamics and autonomy of the DelFly". In: *Bioinspiration & biomimetics* 7.2 (2012), p. 025003.

[30] M. Karásek, F. T. Muijres, C. De Wagter, B. D. Remes, and G. C. De Croon. "A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns". In: *Science* 361.6407 (2018), pp. 1089–1094.

[31] V. Niculescu, H. Müller, I. Ostovar, T. Polonelli, M. Magno, and L. Benini. "Towards a Multi-Pixel Time-of-Flight Indoor Navigation System for Nano-Drone Applications". In: *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2022, pp. 1–6.

[32] B. L. Miller, D. E. Goldberg, et al. "Genetic algorithms, tournament selection, and the effects of noise". In: *Complex systems* 9.3 (1995), pp. 193–212.

[33] O. Khatib. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots". In: *The International Journal of Robotics Research* 5.1 (1986), pp. 90–98.

[34] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. "Deep Sets". In: *Advances in neural information processing systems* 30 (2017).

[35] N. Hansen, S. D. Müller, and P. Koumoutsakos. "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)". In: *Evolutionary computation* 11.1 (2003), pp. 1–18.

[36] S. K. Lam, A. Pitrou, and S. Seibert. "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015, pp. 1–6.

[37] K. M. Kajak, M. Karásek, Q. P. Chu, and G. de Croon. "A minimal longitudinal dynamic model of a tailless flapping wing robot for control design". In: *Bioinspiration & biomimetics* 14.4 (2019), p. 046008.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (2002), pp. 2278–2324.

Part III

Closure

Conclusion

This thesis explored the artificial evolution of behaviour trees to find a coordination strategy for a swarm of flapping-wing drones. The goal was to maximise the collectively visited volume in a simulated greenhouse. Problems arose regarding the simulator: it was found that Aerial Gym does not allow simulating multiple robots in the same physical environment. Consequently, an ad-hoc Python simulator was created based on PyGame and numba.

The artificial evolution was carried out in three phases: firstly, neuroevolution of a weighted deepset was investigated to find high-level swarm coordination behaviours such as exploration or exploitation of known fruit locations in the greenhouse. However, the obtained neural networks did not prove useful with regards to their computational cost. Thus, the neuroevolution behaviours were discarded.

In a second evolution run, genetic programming was applied to assemble a behaviour tree from simple motion commands and more complex reactive planning methods such as artificial potential fields (APF). The behaviour tree obtained this way managed to let the swarm explore 8% of the volume of the simulated greenhouse.

In the last evolution run, covariance matrix adaptation evolutionary strategy (CMAES) was applied to fine-tune the continuous numerical parameters of the behaviour tree obtained in the second run. This fine-tuning lifted exploration performance from 8% to 36%, underscoring the relevance of correct numerical tuning. CMAES also lifted the performance of a manually-designed behaviour tree from 25% to 37%, which shows that genetic programming of behaviour trees does currently not outperform a human design.

Finally, the best purely evolutionarily designed behaviour tree was implemented on board of two Flapper Nimble+ drones, modified to include ultra-wideband-based peer localisation and a forward time-of-flight (ToF) sensor array for obstacle detection. While the implementation was relatively straight-forward, the behaviour exhibited a significant reality gap, most notably in its collision avoidance performance. Both drones did not manage to turn away from a traverse wall on time.

The reality gap is likely worsened by the low fidelity of the simulator: velocities are simulated to be perfectly known at all times. Moreover, the ToF-based obstacle recognition is also modelled perfectly. The simulator also lacks a comprehensive dynamic model of the Flapper Nimble+, which can bypass possible delays and dynamic couplings.

Looking back at the research question “How can a swarm of flapping-wing MAVs be coordinated by means of reactive planning methods?”, a range of answers were found, albeit not satisfactory. The research thus needs to be continued, especially finding suitable behaviour modules. Hence, the current implementation is not yet ready for deployment or even tests in a real greenhouse. The next chapter will propose future work to increase robustness and effectiveness of the Flapper swarm.

Future Work

A key problem in the current setup is that behaviour trees of the chosen complexity cause combinatorial overload, i.e. a lot of computational power is necessary to find useful solutions. It is thus recommended to accelerate the artificial evolution by moving to a more low-level simulation environment like the Swarmulator.

An improved simulator should also include the imperfect sensing and velocity estimation that is characteristic for the Flapper. This way, potential drifting velocities can be actively countered by the evolutionary algorithm. Moreover, a comprehensive dynamic model of the Flapper needs to be created that captures its nonlinearities.

Less important but central to a real deployment is also the creation of a more detailed greenhouse environment. The current environment only includes cuboidal obstacles protruding from the ground, while real greenhouses also feature suspension wires from the ceiling. Moreover, as modern greenhouses tend to be very dense, the dynamics of a Flapper touching the leaves would be interesting to incorporate.

Furthermore, it is recommended to supplement the purely model-based on-board velocity estimation with measurements from optic flow. Although optic flow estimation is difficult on oscillatory platforms, early evidence suggests a potential benefit.

This is especially true, as a mature version of the greenhouse Flapper needs to hold a frontal camera for crop inspection anyway. However, the integration of such a camera requires weight reduction by optimising the cable layout, as the current version is already at its maximum takeoff weight.

Regarding the reactive planning, the investigation of different neural architectures for the behaviour modules is still interesting. A challenge lies in preventing human bias to influence the neuroevolution. With all these ideas in mind, the development of a greenhouse Flapper looks promising and can probably be achieved in a few years.