

IMPLEMENTATION OF DISTRIBUTED SWARM GUIDANCE ALGORITHMS ON LOW-COST UAVS

John McGuire[1], Blake McIvor[1] & Simon Crase[1]

[1]Aerial System Autonomy, Vehicle Autonomy & Dynamics, Platforms Division, Defence Science and Technology Group

Abstract

Uncrewed Aerial Vehicles (UAVs) are increasingly deployed in applications such as search-and-rescue, surveillance, and environmental monitoring, where rapid and adaptive coordination is essential to effectively achieve objectives. The use of multiple autonomous UAVs that coordinate flight paths can reduce the time taken to achieve tasks, reduce cognitive load on operators, and allow for synchronicity in the response to dynamic and uncertain environments. Decentralised guidance architectures for UAV swarms provide resilience, flexibility, and scalability for autonomous algorithms; however, they require each platform to process mission objectives and sensor perception data locally to determine control actions. The requirement for on-board processing encourages the implementation of low-compute guidance algorithms, particularly for low-cost UAVs that are resource constrained, lacking the size, weight, and power budget to support high-performance compute modules. Approaches that rely on forward trajectory prediction or complex optimisation may not be suitable for such systems, motivating the need for lightweight algorithms that can operate onboard the UAV (at the edge) without sacrificing adaptability or safety.

This paper presents the implementation of a novel vector-field-based distributed swarm guidance algorithm for UAV swarms that combines multiple vector fields to travel along a nominal trajectory while conducting collision avoidance and achieving additional objectives. The proposed method enables real-time decentralised coordination without requiring global trajectory planning or forward projection. The simplicity and adaptability of the algorithm has allowed us to build upon the existing literature aimed at tracking simple trajectories and extend this theory to achieve desired swarming behaviours. We utilise a modular implementation where configurable objective, nominal, and avoidance vector functions can be combined to achieve mission-specific trajectories. The algorithm presented in this paper demonstrates at-the-edge, distributed, autonomous swarming behaviours such as persistent observation, formation flight and area search. The autonomous behaviours inherently perform collision avoidance and can dynamically adapt to changes in the number of UAVs and their task allocation in real-time during flight.

We present validation results from multi-agent simulations utilising ArduPilot Software-In-The-Loop (SITL) simulators and custom companion computer software for generating control commands and sending these to the autopilot. Additionally, we provide results from real-world flight tests where the algorithm was flown on fixed-wing UAVs to validate performance under realistic conditions. These experimental results confirm the algorithm achieves collision-free trajectories, adhering to nominal trajectories while achieving missions-specific objectives, and importantly, operates within the computational limits of low-cost UAV hardware. The algorithm demonstrates some emergent behaviours for achieving effects that are not explicitly defined. For example, an intuitive yet unintended behaviour was observed when a platform was merging into a collaborative loiter. For a vehicle merging into the orbit where other UAVs are already there in formation, the approaching vehicle may be initially tasked to slow down below its minimum airspeed to manoeuvre to its desired relative location around the orbit. The algorithm recognises that this command would violate the UAVs performance envelope and determines an alternate velocity control that is feasible. The resultant behaviour was the vehicle traversing a path similar to a time-decayed sinusoidal, effectively reducing the tangential speed along the desired trajectory without violating airspeed constraints, efficiently settling in the desired relative location.

In conclusion, this work introduces a computationally efficient, distributed guidance algorithm that advances UAV swarm autonomy by enabling scalable, resilient, and mission-adaptive coordination. This algorithm has

been evaluated in simulation and flight test on fixed-wing platforms and has been demonstrated to achieve mission objectives of relevance to UAV swarms such as persistent observation, formation flight and area search. Future work involves integrating these swarm guidance algorithms with sensor payloads and tracking algorithms to facilitate sensor-driven control for autonomous missions.

Keywords: Autonomy, Guidance, Control, UAV, Swarm

1. Introduction

Autonomous UAV systems are commonly structured into planning, guidance, and control layers, where guidance algorithms transform mission objectives into desired motion commands such as velocity or heading that are sent to the control layer to action. One of the simplest approaches to guidance is via the use of potential or vector fields. The concept is intuitive and has been used for mobile robot guidance since the 1980s where it was used to plan a path for an omnidirectional steering ground vehicle [1]. These methods generate artificial attractive and repulsive fields that, when combined, will guide an agent toward a desired location while avoiding obstacles. Due to the simplicity, computational efficiency, and adaptability of these methods, they remain prevalent in recent autonomous systems literature. While naturally suited to platforms that can move omnidirectionally such as some ground robots and rotary-wing UAVs, these algorithms have been successfully implemented on fixed-wing UAVs for single vehicle [2, 3, 4, 5], and multi-vehicle applications [6, 7]. These extensions demonstrate that vector field methods can accommodate the non-holonomic constraints of fixed-wing platforms.

Traditional formulations of vector-field guidance typically focus on a single objective, most commonly path following, where a vehicle is guided toward and along a desired trajectory [8, 3]. While many studies consider simple geometries such as lines and circles, more complex paths have been explored [9, 10, 2]. Recent work has explored the combination of vector fields to enable additional behaviours such as collision avoidance [6], formation flight [6, 11, 12], and coordinated target tracking [13]. Other approaches modify existing vectors to achieve characteristics such as arrival position, angle or time [4].
While these methods in isolation enhance performance, combining multiple objectives introduces challenges in balancing competing influences, which can lead to oscillations or degraded performance if not carefully tuned [6]. Furthermore, many existing approaches either rely on structured coordination strategies such as leader-follower architectures [12], assume simplified operating conditions such as constant airspeed [7], or require careful tuning of weighted objectives as seen in flocking-based methods [6]. These limitations reduce the generality of existing approaches, particularly in dynamic, resource-constrained environments where robust, adaptive behaviour is required. For low-cost UAV platforms with limited onboard processing, there remains a need for lightweight, distributed guidance algorithms that can flexibly combine multiple objectives while maintaining stability, scalability, and real-time performance under practical constraint

This paper presents the development and implementation of a low compute, distributed swarm guidance algorithm for low-cost UAV platforms. The proposed approach extends classical vector field methods by utilizing a modular implementation, combining target, agent, and objective vectors to enable decentralized coordination, collision avoidance and mission-specific behaviours without requiring global planning. The algorithm can be used to achieve a range of mission objectives by changing configuration files. The algorithm is designed and demonstrated to operate at the edge, making it suitable for resource constrained systems. Algorithm performance is evaluated through multi-agent simulation using Ardupilot Software-In-The-Loop (SITL) and validated in real-world flight tests on fixed-wing UAVs. Results demonstrate the ability to achieve coordinated behaviours such as collaborative loitering and formation flight while maintaining safe operation and adaptability to dynamic environments.

2. Distributed Swarm Guidance Algorithm

This section introduces the problem formulation for the distributed swarm guidance problem and details the proposed algorithm.

2.1 Problem Formulation

A team of N autonomous platforms are tasked with traversing desired two-dimensional trajectories as part of an autonomous swarm, while avoiding collisions with other swarm members and achieving additional objectives. The platforms have a communications system for sharing vehicle state with other swarm members and to receive tasking information from a ground control station. The collection of the state of all the UAVs is refereed to as the shared swarm state. The platforms may be tasked as groups or individually, and the platforms know which other swarm members have been tasked alongside them.
The state vector for each vehicle is defined

$$\mathbf{x} = \left[x, y, z, v_x, v_y, v_z, \psi\right] \top, \tag{1}$$

where $x, y, z \in \mathbb{R}$ define a 3-D position in a north-east-down (NED) coordinate system relative to a shared home location, $v_x, v_y, v_z \in \mathbb{R}$ are the Cartesian velocities and $\psi \in [-\pi, \pi]$ is the vehicle's heading.

A guidance algorithm is required to determine desired velocities to achieve the mission objectives and a controller then converts these desired velocities into commands for the autopilot. The guidance algorithm utilizes vector fields, combining multiple fields together to determine collision free paths to achieve objectives.

2.2 Vector field path following problem

The basis of the vector field used in this paper is presented by Yao et al. [2]. We utilize two-dimensional trajectories for path planning due to their simplicity in implementation. The algorithms presented in this paper command and control altitude separately from the track to achieve objectives that require specific altitudes.

The following describes the theory for a vector field used to travel along a desired path.
In [10], the desired path P is described by the zero-level set of an implicit function:

$$P = \{(x, y) \in \mathbb{R}^2 : \phi(x, y) = 0\}, \tag{2}$$

where $\phi : \mathbb{R}^2 \to \mathbb{R}$ is a twice continuously differentiable function. If the path is non-self-intersected, then a valid 2-D vector field $\chi : \mathbb{R}^2 \to \mathbb{R}^2$ to solve the vector field path following navigation problem is [10]:

$$\chi(x, y) = E\nabla\phi(x, y) - k\omega(\phi(x, y))\nabla\phi(x, y), \tag{3}$$

where $E = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$, $k \in \mathbb{R}_{>0}$,and $\omega : \mathbb{R} \to \mathbb{R}$ is a strictly increasing function satisfying $\omega(0) = 0$. For simplicity, one can choose $\omega(\phi(x, y)) = \phi(x, y)$ [10] and Eq. (3) becomes

$$\chi(x, y) = E\nabla\phi(x, y) - k\phi(x, y)\nabla\phi(x, y). \tag{4}$$

The first term in Eq. (4) is tangential to the desired path and the second term is perpendicular pushing the vehicle along the path and toward the path respectively. The gain k allows for scaling between these two actions. For each possible vehicle location in $\mathbb{R}^3$, Eq. (4) returns a vector that when converted into a velocity input to the vehicle, will drive the platform towards and along the desired path.

While many applications require individual vehicles to travel along particular paths, other applications require a swarm of UAVs to collaborate to achieve a desired objective. The research presented in this paper uses vector fields as defined in Eq. (4) to guide UAVs along nominal paths, and investigates how multiple vector fields can be combined to allow a swarm of cooperative platforms to achieve additional requirements.

2.3 Algorithm Details

This section presents a generalized vector-field-based guidance algorithm to achieve the objectives introduced in Sec. 2.1. Each agent performs a sequence of vector field calculations, evaluated with its knowledge of the current global state and combining the results to determine a desired velocity vector. This paper distinguishes between three categories of vector field: *target*, *agent*, and *objective*.

- **Target Vector:** The target vector field defines the nominal trajectory of the vehicle. In the absence of the effect of any other vectors, the vehicle will travel along this trajectory at its cruise speed.
- **Agent Vector:** Agent vectors are assigned to other platforms that the vehicle is aware of. The agent vector typically is a repulsive force that enforces collision avoidance.
- **Objective Vector:** Objective vectors are used to modify the vehicle's behaviour to achieve another objective simultaneously. For example, a vehicle's target trajectory may be a circular loiter, and the objective vector enforces a desired separation between swarm members loitering around the same track.

For vehicle $i \in 1,\ldots,N$, the vector field equation is:

$$\chi_R = \chi_T(\mathbf{x_i}) + \chi_O(\mathbf{x_i}, *) + \sum_{j=1}^{N} \chi_{A,j}(\mathbf{x_j}), j \neq i, \tag{5}$$

where $\mathbf{x_i}$ is a state vector in the form of Eq. (1) for vehicle i. χ_O is the objective vector, $\chi_{A,j}$ is the agent vector for vehicle j (these may be different for different agents), χ_T is the target vector, and χ_R is the resultant vector. Specific arguments for the objective vector are omitted as they are dependant on the function selected.

χ_R needs to be mapped to a physical quantity for the control of the aircraft, it also needs to be bounded within a feasible set to ensure that the command is actionable by the UAV. Let V_{max}, V_{min}, and V_c denote the maximum, minimum, and cruise speeds of the UAV respectively. We introduce the mapping $\alpha : \mathbb{R} \to \mathbb{R}$ where α is a strictly increasing function satisfying $\omega(1) = V_c$, used to convert $||\chi_R||$ into a desired airspeed V_d. The components of the desired airspeed in the x and y direction, v_x^d and v_y^d respectively, can be calculated using the following expression:

$$\begin{bmatrix} v_x^d \\ v_y^d \end{bmatrix} = \alpha(||\chi_R||)\hat{\chi}_R, \tag{6}$$

where

$$\alpha(||\chi_R||) = (K_v(||\chi_R|| - 1) + 1)V_c. \tag{7}$$

$||\cdot||$ denotes the magnitude of a vector and $\hat{\cdot}$ denotes a normalized vector. $K_v > 0$ is a gain that modifies the gradient of the line relating the vector magnitude to the desired airspeed. The selection of a mapping α is not a requirement for the algorithm and was included to specifically shape the response after performance testing. A clamping function is use to ensure that $V_d \in [V_{\text{min}}, V_{\text{max}}]$. Figure 1 presents the output of Eq. (6) for different values of K_v to demonstrate how its selection effects the dynamic range of the applied airspeed for given airspeed limits.

To enable the combination of multiple vectors as per Eq. (5), while keeping the resultant vector actionable in terms of the allowable control inputs the following procedure is followed.

- For the target vector χ_T, a function $\phi(x,y)$ is selected and a vector is computed according to Eq. (4) to traverse the desired trajectory.
- The target vector is normalized to ensure that in the absence of the effect of the avoidance and objective vectors, the vehicle will traverse the nominal track at its cruise speed according to Eq. (6).

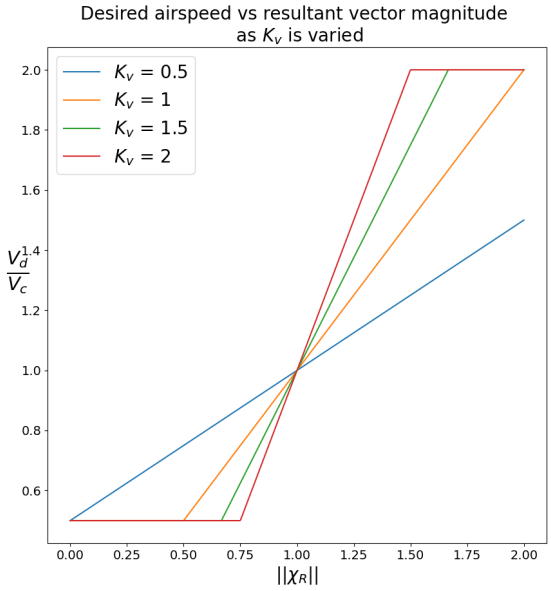


Figure 1 – Scaled and clamped desired airspeed vs resultant vector magnitude as K_v is varied. To demonstrate the airspeed clamping, V_{max} and V_{min} are arbitrarily set to 2 and 0.5 times the cruise velocity respectively.

- The agent vector χ_A and objective vector χ_O are computed.
- The resultant vector is computed as per Eq. (5).
- The resultant vector is turned into a desired velocity vector using Eq. (6).
- The desired velocity is sent to the controller which then sends desired set points to the autopilot.

2.4 Example Application

An example application is presented here to provide detail on how the combination of a target, agent, and objective vector can be used to achieve a particular mission. A team of N aircraft are tasked with traversing a circular path, the vehicles must avoid collisions with each other, and the desired swarm behaviour exhibits equiangular separation between the UAVs around the circumference of the circle. This type of behaviour is relevant to swarms conducting coordinated sensing, where optimal relative sensor geometries can improve estimator performance [14].

2.4.1 Target Vector

The following function defines a circular path:

$$\phi(x,y) = \frac{(x-x_0)^2}{r^2} + \frac{(y-y_0)^2}{r^2} - 1, \tag{8}$$

where (x_0, y_0) is the centre location of a circle with radius r. Eq. (4) is then used to determine the vector field χ_A.

2.4.2 Agent Vector

The simplest avoidance function is a purely repulsive force; a unit vector is calculated from the object being avoided to the UAV and an appropriate gain is applied to scale the avoidance action. The aim of this avoidance function is to perturb the trajectory of the vehicle if a collision is possible without intervention. In most scenarios, the vehicle should still be influenced by the *target vector* while avoiding, to shape the trajectory such that both objectives are considered.

Let $\mathbf{d}_i$ denote the vector between the UAV and the i^{th} swarm member in the x-y plane i.e.,

$$\mathbf{d}_i = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \tag{9}$$

where x, y denotes the position of the UAV and x_i, y_i denotes the position of the i^{th} swarm member. The normalized position vector is defined as

$$\hat{\mathbf{d}}_i = \frac{\mathbf{d}_i}{||\mathbf{d}_i||}, \tag{10}$$

where $||\mathbf{d}_i||$ is the Euclidean norm of $\mathbf{d}_i$. This returns a vector with magnitude 1 in the direction from the i^{th} swarm member to the UAV performing the calculation.
The repulsion vector between the UAV and the i^{th} swarm member can be generated by multiplying the unit distance vector with the desired avoidance gain:

$$\chi_{A,i} = K_{a,i}\hat{\mathbf{d}}_i. \tag{11}$$

The gain for the avoidance $K_{a,i}$ is calculated using the following expression:

$$K_{a,i} = \frac{k_a}{||\mathbf{d}_i||\gamma}, \tag{12}$$

where $k_a > 0$ is a scalar for avoidance, and $\gamma > 0$ is the repulsion exponent, enabling the scaling of the avoidance to other powers of the distance between swarm members.

2.4.3 Objective Vector

This section presents a vector function that can produce the desired equiangular separation in a swarm of UAVs collaboratively traveling along a circular trajectory.
The objective vector fields are implemented differently to Eq. (4) as they are not strictly a function for planning a path. The objective function here can be considered a modifier to the path planning algorithm, not directly trying to alter the desired trajectory, but altering how the vehicle conducts the desired trajectory. Effectively, a vector magnitude is produced based on a specified heuristic, and then applied to a vector tangential to the desired trajectory, essentially slowing or speeding up the vehicle along its desired trajectory to achieve a specific outcome.

θ_i is the angular position of each platform relative to x_0, y_0. The angular separation between the vehicle and agent i is denoted ϑ_i and the desired angular separation is $\vartheta_d = \frac{2\pi}{N}$.

If the angular separation between the UAV and all other swarm members are mapped $\in [0, 2\pi]$, then along a closed list of angular separations, the UAV is between the agents with the largest angular separation ϑ_{max} and the smallest angular separation ϑ_{min}. Listing vehicles by minimum and maximum angular separation is the easiest way to sort which agents the UAV is trying to align itself between, but for calculation it is easier to consider them as the closest counter-clockwise and closest clockwise agent. The closest vehicle becomes the closest clockwise vehicle and the furthest vehicle is the closest counter-clockwise vehicle with their angular differences denotes ϑ_{CW} and ϑ_{CCW} respectively. $\vartheta_{\text{CCW}} = 2\pi - \vartheta_{\text{max}}$.

The goal of the objective function is to alter the speed of the UAV along the desired trajectory such that ϑ_{CW} and ϑ_{CCW} track towards ϑ_d. This can be achieved in a distributed fashion, if all agents aim to achieve equiangular separation between the vehicle directly before and aft of it in the loiter. At any given time, the vehicle is trying to drive $\vartheta_{\text{CW}} - \vartheta_{\text{CCW}}$ to 0. The following equation defines $\beta \in [-1, 1]$, which represents the relative strength and directionality of the objective force vector. Note that for clockwise and counter clockwise force vectors, β must be greater or less than zero, respectively.

$$\beta = \frac{\vartheta_{\text{CW}} - \vartheta_{\text{CCW}}}{2\pi}. \tag{13}$$

A shaping function can be used to implement a non-linear relationship between angular error and objective vector magnitude. Let $\zeta \in [0,1] = ||\chi_O||$ denote the objective vector magnitude:

$$\zeta = \frac{a^{||\beta||} - 1}{a - 1}, \tag{14}$$

where $a \in (0,1)$. The mapping in Eq. (14) causes the objective vector field to have more of an effect at further distances, increasing the convergence speed to the desired angular separation.

The direction of the objective vector is that of the tangential component of the target vector. This enables an objective function to only effect the vehicles speed along the desired trajectory, without impacting its ability to travel towards the desired trajectory.

Let χ_T' denote the tangential component of the target vector χ_T. The objective vector magnitude is applied to the tangential component of the target vector

$$\chi_O = \frac{\chi_T'}{||\chi_T||}\zeta. \tag{15}$$

2.5 Transit Details

Another example application of the proposed algorithm is to enable a swarm of UAVs to perform a transit in formation flight. The transit application uses the same underlying approach to achieve swarm guidance, but the implementation has additional steps.

The command to the UAV provides a series of waypoints that make up the nominal transit path. The algorithm achieves a transit by utilizing a dynamic χ_T, that is defined based on the current waypoint m. The target vector takes the form of a line segment intersecting the previous waypoint (x_{m-1}, y_{m-1}) and the next waypoint (x_m, y_m). Let

$$A = y_{m-1} - y_m, \qquad B = -(x_{m-1} - x_m), \qquad C = x_{m-1}y_m - x_m y_{m-1}, \tag{16}$$

the explicit equation that can be used to generate a vector field as per Eq (4) is

$$\phi(x,y) \doteq \frac{Ax + By + C}{\sqrt{A^2 + B^2}}. \tag{17}$$

The above function gives a vector field for an infinitely long line intersecting the two waypoints. For tracking towards the first waypoint on initialization, an initial transit line directly towards the first waypoint from the current vehicle location is used. The algorithm determines when the vehicle has arrived at a waypoint and generates a new target vector pointing towards the next waypoint. This allows arbitrary transit paths to be generated and followed, making it suitable for a number of applications including those requiring transiting through locations with multiple exclusion zones.

The objective vector field for the transits can be arbitrarily selected based on the mission objectives. One example is that the transit path may be looped and the objective vector could enforce a desired separation in meters or equally spread out along a polygon to perform a search. Another option is to specify close-up formations for formation flight. Some commmon formations are line astern; where platforms are sequenced in a line from nose to tail with a defined distance between each agent, and line abreast; where platforms are aligned from wingtip to wingtip with a defined distance between agents.

For formation flight, the same agent vector from Sec. 2.4.2 can be used for transit participants, however the gain should be reduced to accommodate for the fact that the platforms are meant to be in a close formation. To maintain an approximate formation, the objective vector can take the form of an attractive force to the other agents that is only active when the platforms are sufficiently far apart, i.e., does not impact any avoidance behaviours. This attractive force modifies the commanded velocity

of the agents performing the transit such that after a sufficient amount of time the vehicles are close together. This simple function allows for an approximation of formation flight while maintaining the low-cost and low-compute desires of this algorithm. True formation flight algorithms would require faster state updates, either through high-bandwidth communication channels or sensors for relative localisation between platforms.

The difference between the transit line astern and the line abreast is how the target vector is calculated. All UAVs share the same target vector for a line astern formation, but for a line abreast formation, each vehicle has a slightly different target vector. A parameter C is defined as the corridor width, which defines the width allowable for the transits along the transit line that the vehicles can spread themselves out over. Vehicles independently assign themselves to one of N parallel trajectories equally spaced within the specified corridor width based on their relative cross-track error. This assignment is recomputed to adapt to changes in vehicle state or the swarm composition.

3. Implementation details

The algorithm presented in 2. returns a desired two-dimensional velocity vector and an altitude. This formulation lends itself naturally to systems with holonomic constraints such as rotary-wing platforms. For fixed-wing UAVs, an approximation must be used as the Cartesian velocities cannot be commanded directly in the ArduPlane software.

Firstly, the velocity vector undergoes a Cartesian to polar transform, decomposing the desired input into a desired speed and a desired velocity heading. This formulation allows for a decoupling of control inputs, a desired airspeed can be sent to the autopilot and the desired velocity heading can be achieved (not instantaneously) by commanding the desired roll angle of the platform. By commanding the roll, the platform will turn and the course-over-ground can be influenced, thereby redirecting the velocity heading such that the desired ground-track is achieved. Despite the limitation of the algorithm being dependent on the autopilot's inner control loops for lateral motion, we show in simulation and flight test that this limitation does not have significant impact on the platforms ability to traverse a trajectory, at least for the size and speed of the platforms under test in this research.

The UAVs share state information between themselves at a frequency of 1 Hz. The guidance algorithms onboard run at a frequency of 10 Hz.

4. Simulation Results

Ardupilot Software-In-The-Loop (SITL) is a simulation environment provided as part of the open-source ArduPilot firmware[1] that simulates virtual UAVs with inbuilt flight dynamics models. The use of this open-source simulation infrastructure allows for the rapid prototyping of algorithms and a way of validating performance before flight-test. Because the simulation communicates using MAVLink messaging protocol like the physical autopilot, the same code that is used in the simulation can be deployed to the UAV. The algorithms presented in this paper are developed in Python utilizing the PyMavlink[2] Python library for communicating with the autopilot.

We use simulations to validate the algorithm performance prior to flight testing. The algorithms are validated using four simulated aircraft designated *sim-1*, *sim-2*, *sim-3*, and *sim-4*. The algorithm is not specific to a particular aircraft model and the default Rascal110 ArduPlane dynamics model is used for the simulated agents.
The following simulation scenarios are presented:

- Sim Scenario 1: Figure 2 presents the experimental results of four aircraft traversing a circular path, aiming to separate equidistantly around the circumference of the circle.

[1] https://github.com/ardupilot/ardupilot
[2] https://github.com/ardupilot/pymavlink

- Sim Scenario 2: Figure 3 presents the experimental results of four aircraft traveling along a transit line attempting a line astern formation.
- Sim Scenario 3: Figure 4 presents the experimental results of four aircraft traveling along a transit path attempting a line abreast formation.

4.1 Sim Scenario 1

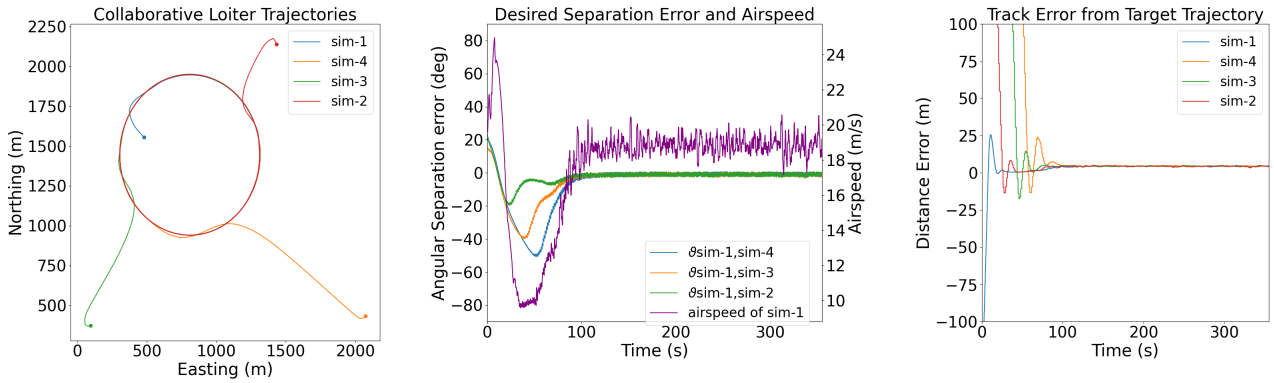
Figure 2 presents the experimental results for four simulated aircraft (labelled *sim-1*,*sim-2*,*sim-3*, and *sim-4*) using the target, agent, and objective vectors defined in the Sec. 2.4. The aircraft begin at arbitrary locations and are commanded to cooperatively traverse a circle of 500 m radius, while simultaneously maintaining equiangular separation from other agents around the circle. Figure 2a shows the trajectories of the aircraft as they attempt to satisfy the mission objectives and show the aircraft smoothly transitioning to the desired track.

Figure 2b plots the error in the desired separation of the agents relative to *sim-1* and the airspeed of *sim-1*. *sim-1* was the first UAV to reach the desired track. The commanded airspeed varies as expected to regulate angular separation, increasing when the vehicle lags and decreasing when approaching other agents. For example, consider how *sim-1* is traveling along the track in the beginning of the experiment, *sim-2* is still traveling towards the track and its angular position around the circle isn't changing. This is accounted for by *sim-1* by commanding a reduced airspeed as can be seen in Fig. 2b. After approximately 100 seconds of flight, the aircraft are equiangularly separated as they traverse the circular trajectory. The average error in angular separation between the platforms is approximately -1 degree as can be seen in Table 1. This implies that all vehicles were on average 1 deg too close to $sim-1$. This observation is consistent with the recorded airspeed of the UAV, which has a mean of 17.68 m/s which is less than the cruise airspeed of 20 m/s as expected based on the objective vector for equiangular separation introduced in Sec. 2.4.3. The airspeed standard deviation was 2.98 m/s indicating that there was a lot of movement of the aircraft around the desired set points. This is likely an artifact of the shared swarm state that is only updated between the UAVs at 1 Hz. Because the state sharing is performed less frequently than the guidance, the guidance is often making calculations several samples in a row with the same state information. This results in the perception that the UAVs are moving closer together, producing the response to slow the aircraft down. To mitigate the effect of this, the sharing of swarm member state could be increased between platforms, or each platform could perform interpolation of swarm member state.

After settling onto the desired tracks, the vehicles maintained a close tracking to the desired path. *sim-1* achieved a mean cross track error of 4.48 m with other vehicles achieving similar mean distances of between 4.42 and 4.67 m. The non-zero steady state error is expected as the reactive guidance algorithm does not project forward and therefore requires constant correction to track a the circumference of a circle using heading and speed commands. The low standard deviation of the track error is a result of the fast update rate of the algorithm, it is expected that less frequent inputs would result in a larger track error. Table 1 summarises the numerical results from the first simulation scenario.

Metric	Mean	Standard Deviation
airspeed	17.68 m/s	2.98 m/s
track error	4.48 m	0.18 m
ϑ_2^e	-0.86 deg	0.76 deg
ϑ_3^e	-0.97 deg	0.76 deg
ϑ_4^e	-1.03 deg	0.79 deg

Table 1 – Metrics from platform *sim-1* in Flight Scenario 1 from the last 200 seconds of flight. ϑ_i^e denotes the error in desired angular separation between *sim-1* and *sim-i*.



(a) Initial loiter paths, initial vehicle locations are indicated by dots.

(b) Airspeed and angular error in the desired separation

(c) Cross track error for each platform

Figure 2 – Results of Sim Scenario 1, a 4-ship circular trajectory with equiangular separation.

4.2 Sim Scenario 2

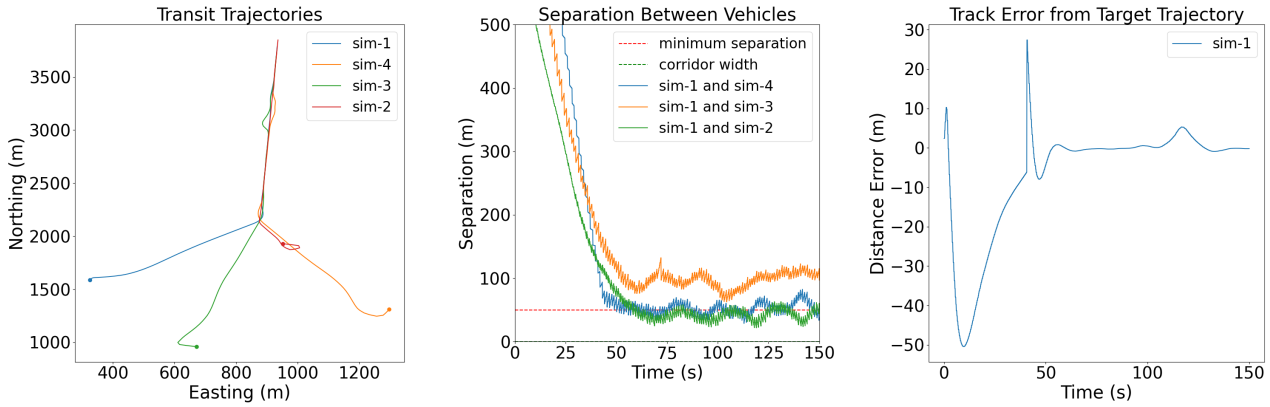
Figure 3 presents the simulation outcomes of four platforms performing a transit in a line astern formation. Fig. 3a shows the transit paths for the four platforms. The UAVs fly directly towards the first waypoint and then along the transit path. The attractive objective vector influences their desired velocity to pull together and the avoidance of the agent vector ensures that there are no collisions. After approximately 1000 m of travel, some of the UAVs are perturbed off the transit line. This behaviour is evidence that the repulsion force of the avoidance is too high, and highlights the dependency of this particular method on parameter tuning. Because the vehicle is operating at the edge of an avoidance and an attractive field, the behaviour is naturally oscillatory. This effect is evident in Fig. 3b, showing the distance between the agents oscillating. Even if the avoidance function tuning is highly optimized for a particular separation, in most formations, the vehicles are not all equidistant from each other. In this example the minimum separation is set to 50 m, but in a line astern with four vehicles, the distance between the first and the last platform is 150 m, resulting in a stronger attractive force between them, compressing the formation tighter than the requested 50 m.

The track error for *sim-1* is shown in Fig. 3c. There are some divergences of the UAVs from the desired path, initially from the vehicles attempteing to get into formation, and at the 120 s mark where the UAVs are significantly perturbed from there path. During periods of stable formation, the track error is very low with a mean of -0.02288 m with a standard deviation of 0.284 m, this tracking is significantly better than in Flight Scenario 1 where the circular trajectory is followed. From 50 s to 150 s the mean and standard deviation of the track error is 0.3707 and 1.474 respectively.

4.3 Sim Scenario 3

Figure 4 shows the simulation results from a 4-ship line abreast transit. The four platforms were commanded to travel along a rectangular transit path, and the corridor width was set to 100 m. Fig. 4a presents the transit path. Let *leg 1* be the transit leg from north to south, *leg 2* from west to east, and *leg-3* from south to north. The platforms are constantly evaluating which lane they are attempting to fly along each transit leg, and can be seen that there is significant lane switching in the transit from leg 2 to leg 3. It took approximately 20 seconds to transition between waypoints, but this can extend when there is some avoidance due to lane switching.

Table 2 details the steady-state cross-track error mean and standard deviation for each UAV along each transit legs, omitting the time navigating between transit lines. The standard deviation for the cross-track error remains small for all of the platforms, however, the mean varies, hinting at a systematic error. These errors are due to the inter-agent attractive component of the objective function. The platforms have an attraction to other members of the swarm as well as to the transit line, and the vehicle trajectory is located at the minima of these two interactive fields. Platforms on either side of



(a) Transit paths, initial vehicle locations are indicated by dots.

(b) Separation between vehicles.

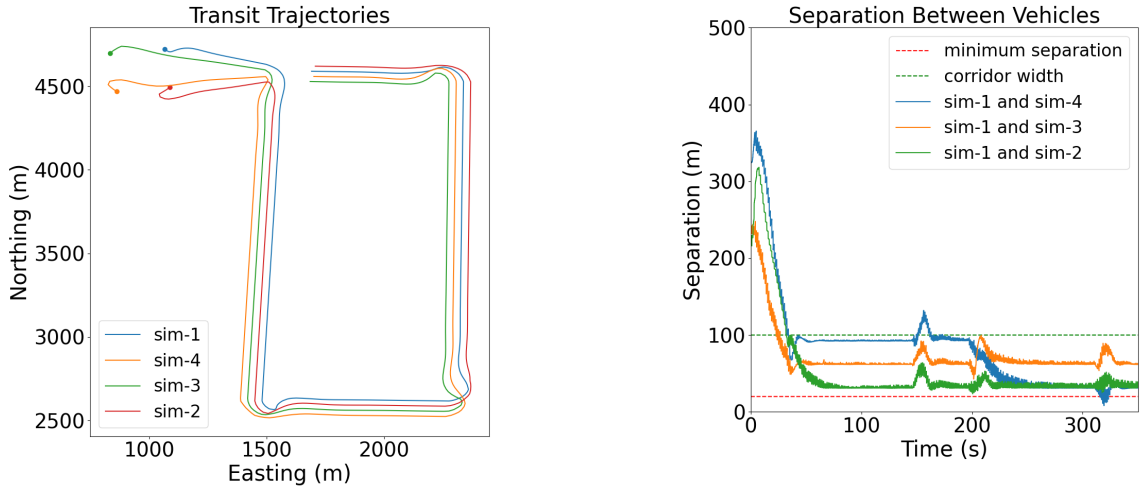
(c) Cross track error for each platform

Figure 3 – Results of Sim Scenario 2, a 4-ship line astern transit.

the corridor width are furthest away from each other and therefore have the largest attractive force between them, causing their tracking to have a larger error than UAVs in the inner lanes. If a smaller corridor width was selected, or more agents were selected this formation, oscillatory behaviour would likely be observed as the vehicles approach the avoidance distance.

	sim-1	sim-2	sim-3	sim-4
leg 1	(-4.50, 0.0543)	(-1.99, 0.0592)	(0.466, 0.0480)	(3.24, 0.0448)
leg 2	(-4.523, 0.0592)	(-1.920, 0.142)	(0.6046, 0.176)	(3.162, 0.081)
leg 3	(0.692, 0.0372)	(3.00, 0.0427)	(-4.237, 0.0463)	(-1.91, 0.0553)

Table 2 – Track error mean and standard deviation for each UAV in Sim Scenario 3.



(a) Transit paths, initial vehicle locations are indicated by dots.

(b) Separation between vehicles.

Figure 4 – Results of Sim Scenario 3, a 4-ship line abreast transit.

4.4 Computational Complexity

The algorithms presented in this paper are myopic, i.e., there is no receding horizon path planning implemented. This design decision ensures that the computational cost of running the algorithm is

low, at the expense of running a purely reactive guidance system. In general, the computational complexity for the algorithm increases proportionally to the number of platforms n in the swarm $\mathcal{O}(n)$. Due to the distributed nature of the algorithm, each vehicle only has to compute its own commands. The calculations for the agent vectors and some objective vectors require computations for each possible swarm member.

On representative hardware (Raspberry Pi Compute Module 4), the algorithm was running at 10Hz frequency and utilizing approximately 20% of a single core. The limiting factor of the implementation of the algorithm is not the algorithm computation, but the time waiting for responses from remote function call requests to get the state of the swarm members and to send desired control commands to the autopilot interface.

While experiments were conducted on small teams, the distributed architecture and linear computational complexity suggest that the approach is scalable to larger swarms, subject to communication bandwidth constraints.

5. Flight Test

Flight tests were conducted to validate the performance of the algorithms in a practical setting. The guidance algorithms presented as part of this research are a fundamental part of an autonomous software stack designed to facilitate the test and evaluation of autonomy on UAVs. It is important that these algorithms are validated in flight to ascertain if they are fit for purpose. Flight tests were conducted on two separate occasions, firstly with two platforms designated *str-001* and *x8-16* a heterogeneous swarm comprised of two different fixed-wing aircraft due to platform availability. The second was conducted with two platforms designated *str-001* and *str-002*, a homogeneous swarm.

5.1 Aircraft Hardware

Most of the flight testing was conducted using a Striver mini, a conventional twin-engine fixed-wing unmanned aircraft with a total take-off mass of 5.2 kg. Some flight tests also involved an X8 Skywalker, a similarly sized flying-wing UAV. The core avionics suite comprised a CubePilot Cube Orange flight controller and a Raspberry Pi Compute Module 4 (CM4) acting as a companion computer. The total mass of the core avionics is approximately 120 g, representing 2.3% of the aircraft's total mass. In addition to the core avionics, a 2.4 GHz mesh radio is utilised to provide inter-agent and ground-control communication. This mesh network enables decentralised information sharing between vehicles, as well as command and monitoring capabilities for the ground-based operator.

The mesh link is used to exchange essential vehicle state information between agents to support distributed coordination. However, such communication channels are inherently bandwidth-limited, particularly as the number of participating vehicles increases. To address this, the proposed guidance algorithm is designed to operate using minimal shared information at relatively low update rates. By avoiding reliance on high-frequency state exchange or large data payloads, the approach is intended to reduce network congestion and improve scalability within the swarm.

The efficiency of the guidance algorithm directly translates to reduced size, weight, and power (SWAP) requirements for the avionics system. This reduction enables two key advantages: firstly, the approach can be scaled to smaller UAS platforms with strict SWAP constraints; and secondly, the reduced computational and communication burden allows greater allocation of onboard resources and bandwidth to mission-specific payloads and higher-level autonomy functions. As such, the presented architecture demonstrates how algorithmic simplicity supports practical, distributed autonomy on low-cost UAV platforms while preserving flexibility for expanded mission capability and larger swarm sizes.

5.2 Flight Test Results

The versatility and modularity of the proposed approach means that many different combinations of vectors can be used and tested to achieve a range of different objectives. A fleet of three UAVs were

used to conduct the flight tests Strivers *str-001* and *str-002*, and an X8 *x8-16*. The following scenarios are presented here:

- Flight Scenario 1: Figure 5 presents experimental results of two aircraft transitioning from seperate loiters to a collaborative loiter with equiangular separation.
- Flight Scenario 2: Figure 6 presents experimental results of two aircraft performing a line astern formation transit along a circular path.
- Flight Scenario 3: Figure 7 presents experimental results of two aircraft performing a line astern formation transit along a rectangular path.
- Flight Scenario 4: Figure 8 presents experimental results of two aircraft performing a line abreast formation transit along a rectangular path.

5.2.1 Flight Scenario 1

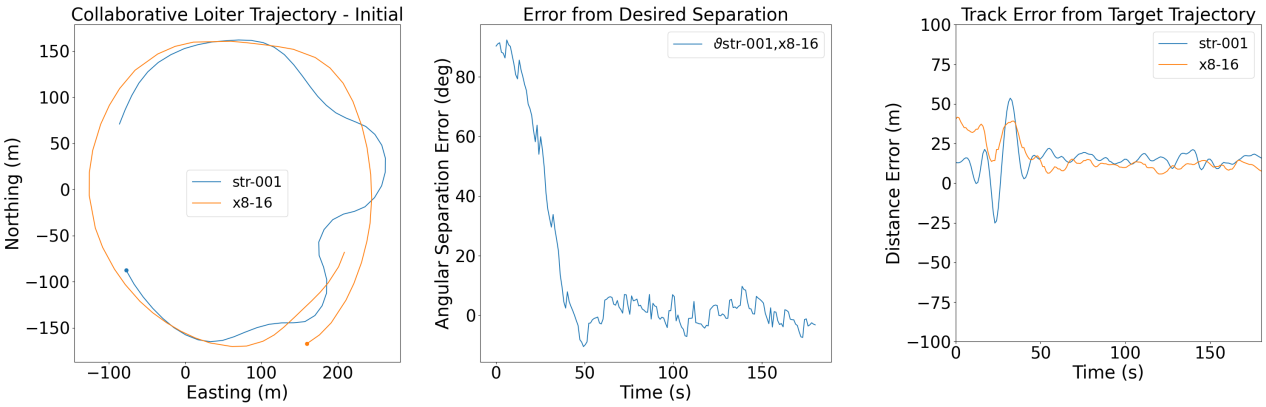
Figure 5 presents results from the flights of two aircraft tasked with transitioning from nearby individual loiters to a collaborative loiter with equiangular separation. Figure 5a shows the flight path of the initial 60 seconds of flight where the aircraft are tasked to travel counter clockwise along a 300 m radius circle. The combination of the two objectives of path tracking, and platform separation result in an interesting behaviour that can be observed in the path of *str-001*. *str-001* begins the collaborative behaviour in front of *x8-16* and due to the objective vector action presented in Sec. 2.4.3, both platforms adjust their airspeed to equalize the angular distance between them on the circumference of the circle. *str-001* decreases its velocity component tangential to the desired trajectory, causing a relative increase in the perpendicular component of the velocity. This causes the aircraft to perform a sinusoidal perturbation along the desired trajectory. This maneuver was not explicitly commanded, and emerges as a result of the objective vector selected and its implementation. The maneuver drastically reduces the vehicles ground speed tangential to the desired track without violating the minimum airspeed constraint. The trade off between the two objectives can be seen when comparing the error in the desired separation in Fig. 5b with the cross-track error of the platforms present in Fig. 5c. The error in desired separation is drastically reduced in the initial parts of the loiter, resulting in a trade-off with tracking performance which can be seen as some large initial errors in the track error. The mean track error for *str-001* is 15.07 m with a standard deviation of 2.77 m and $x8-16$ achieves a track error of 11.18 m with a standard deviation of 2.163. At the conclusion of this maneuver the platforms are approximately equidistant around the circle loiter as can be seen in Fig. 5b. The angular separation error has a mean of 0.984 deg and a standard deviation of 3.75 after convergence. The variability can be attributed to the rate limitations of the aircraft, environmental factors such as wind, and communications delays in the sharing of swarm state between swarm members.

metric	mean	standard deviation
airspeed	16.82 m/s	1.66 m/s
track error	15.07 m	2.77 m
ϑ str-001,x8-16	0.984 deg	3.75 deg

Table 3 – Metrics from platform *str-001* in Flight Scenario 1 from the last 230 seconds of flight.

5.2.2 Flight Scenario 2

Figure 6 presents the results of two aircraft performing a line astern formation transit along a circular path. In this instance, the target vector is the same as that used in the previous scenario, but the agent and objective vectors are different. Specifically, the agent vector repulsion is decreased and the objective vector used is that of the attraction function for formation flight introduced in Sec. 2.5. This demonstrates how different objective vectors can be used to alter the outcome of the behaviours without significantly changing the algorithm. Fig. 6b shows the separation between the two vehicles slowly reducing, and Fig. 6c shows the cross-track error of both platforms is low, *str-001* maintains a cross track error mean of 6.94 m and a variance of 3.33 m and *str-002* maintains a cross track error of 6.80 m and a variance of 2.72 m.

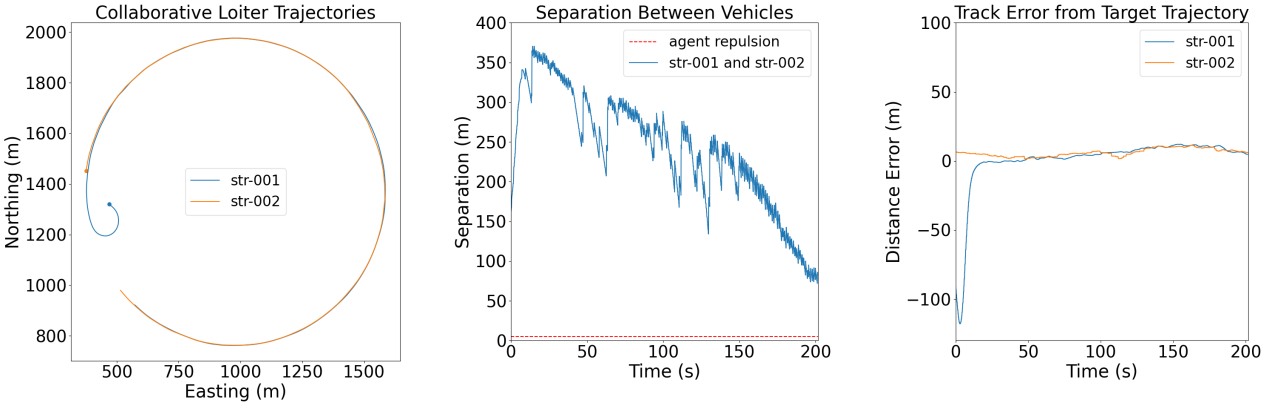


(a) Initial loiter paths, initial vehicle locations are indicated by dots.

(b) Angular error in the desired separation.

(c) Cross track error for each platform

Figure 5 – Results of Flight Scenario 1, a 2-ship circular trajectory with equiangular separation.



(a) Initial loiter paths, initial vehicle locations are indicated by dots.

(b) Angular error in the desired separation.

(c) Cross track error for each platform.

Figure 6 – Results of Flight Scenario 2, a 2-ship circular trajectory with line astern formation.

5.2.3 *Flight Scenario 3*

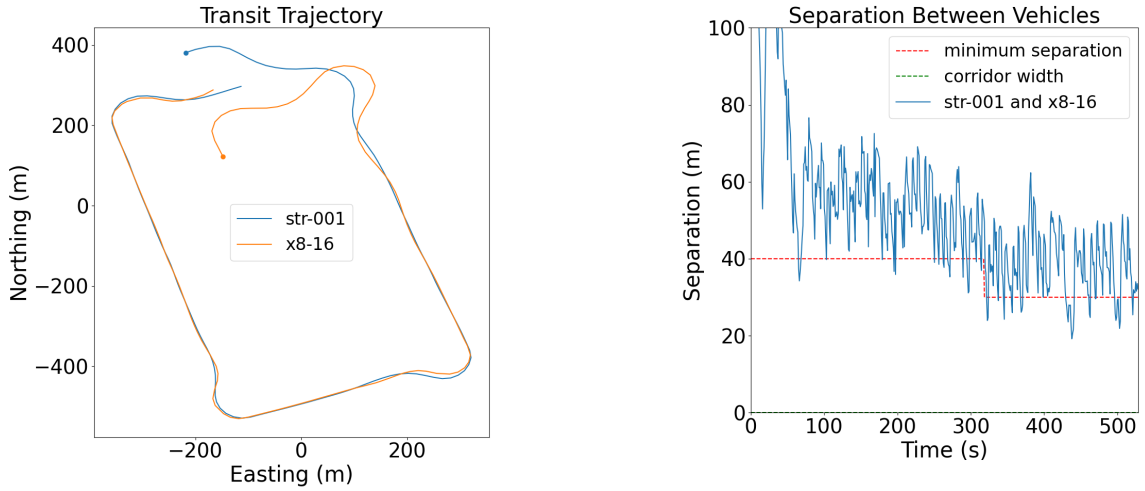
Figure 7 presents two aircraft performing a line astrern formation flight along a rectangular path. Fig. 7a shows the vehicles successfully converging on the desired transit path. Let *leg 1* be the transit leg from north to south, *leg 2* from east to west, and *leg-3* from south to north. Fig. 7b presents the desired separation between the platforms. The red dotted line is the desired separation, practically achieved by the transition from the attractive force of the formation with the repulsive force of the avoidance. The blue line is the achieved separation between the vehicles. The reported distance between the agents differed as much as 20 m between samples, even in favourable circumstances, due to the delays in received swarm member position updates. However, the separation does trend towards the desired separation of 40 meters with a mean separation error of 13.78 m and standard deviation of 8.79 m. In flight the desired separation distance is reduced from 40 m to 30 m and the average vehicle separation error reduces to 8.45 m, with a standard deviation of 8.58 m. Table 4 displays the mean and standard deviation of the cross-track error for both platforms for one run of the transit path. The results show good tracking, with mean error of between approximately 0.5 and 2.4 m.

5.2.4 *Flight Scenario 4*

Figure 8 presents two aircraft performing a line abreast formation transit along a rectangular path. Let *leg 1* be the transit leg from north to south, *leg 2* from east to west and *leg 3* is south to north. Lane switching is evident in the transit trajectories in Fig. 8a. An artifact of both platforms having

	str-001	x8-16
leg 1	(2.39, 1.79)	(0.642, 5.45))
leg 2	(1.80, 2.40)	(0.496, 1.38)
leg 3	(1.27, 1.06)	(0.728, 2.005)

Table 4 – Track error mean and standard deviation in meters for each UAV on one run of Flight Scenario 3.



(a) Transit path of 2-ship line astern transit, initial vehicle locations are indicated by dots.

(b) Desired separation between transit agents.

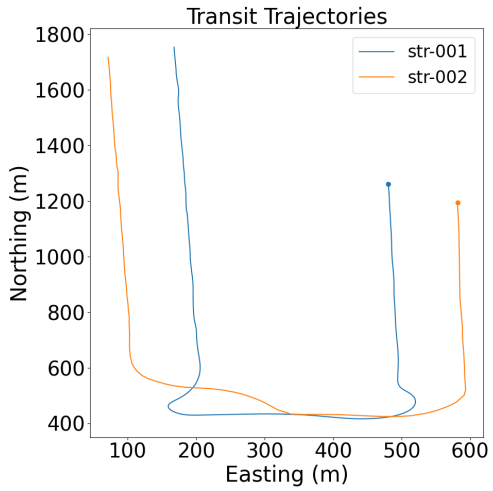
Figure 7 – Results of Flight Scenario 3, a 2-ship line astern transit.

slightly incorrect estimations of each other's position can also be observed, during the very short transit along leg 2, both platforms assigned themselves to the southern-most lane, thinking they were the southern-most platform. This was quickly resolved and *str-002* selected the top lane in time to turn onto the next transit line.

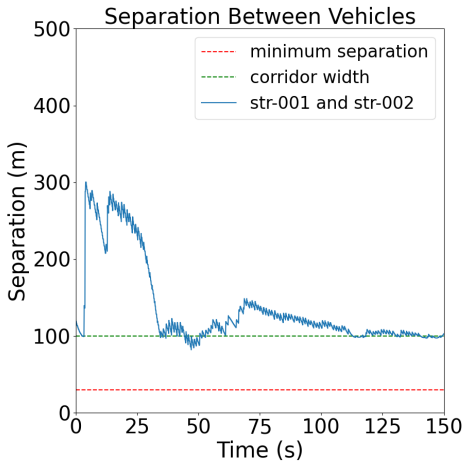
Fig. 8 shows the desired separation between transit agents. The line abreast formation flight had less oscillation during a long transit than the line astern formation. During the third leg of the transit, the mean separation is 112.64 m with 13.00 m standard deviation. This is due to the fact that in this instance, the corridor width is 100 m and the individual lines of travel are much further away than the minimum separation of the platforms. For most of the transit no repulsive forces are active as each platform is traveling along its desired transit line that is further away than the avoidance distance, and the objective vector is slowly bringing them abreast by influencing the desired airspeed. Table 5 shows the mean and standard deviations of the cross-track error for both platforms during Flight Scenario 4. Other than in the instance where *str-002* performed some lane switching and the platforms evaded each other temporarily, the mean track error was between approximately 0.7 and 2 m.

	str-001	str-002
leg 1	(1.15, 0.373)	(-1.97, 1.98)
leg 2	(2.82, 7.51)	(43.76, 36.74)
leg 3	(1.89, 0.67)	(-0.69, 0.45)

Table 5 – Track error mean and standard deviation in meters for each UAVs on one run performing a line abreast formation in Flight Scenario 4.



(a) Transit path of 2-ship line abreast transit, initial vehicle locations are indicated by dots.



(b) Desired separation between transit agents.

Figure 8 – Results of Flight Scenario 4, a 2-ship line abreast transit.

6. Discussion

The results demonstrate a strong alignment of simulation results and flight test results, in both cases the qualitative behaviours are consistent such as convergence to desired formations and stable trajectory tracking. However, flight test results demonstrate increased tracking error and variability when compared to simulation. For example, cross-track error for circular trajectories increases from approximately 4.5 m in simulation to $7-15$ m during flight test. Equiangular separation was achieved to approximately 1 degree in both simulation and flight test, however the standard deviation increased from ≈ 0.8 deg to 3.7 deg. This degradation is primarily attributed to real-world effects such as wind disturbances, actuator limitations, and communication delays.

A key limitation of the current implementation is the reliance on the shared swarm state of other UAVs, transmitted at only 1 Hz. While this rate is sufficient in simulation, during flight test communication degradation had a significant effect on performance. Despite this the algorithm remains stable under these conditions but delayed state information introduces variability in separation control and command generation, particularly when vehicles are near equilibrium. This effect is most pronounced in tightly coupled formations, where outdated positional information can lead to alternating attraction and avoidance responses, resulting in oscillatory behaviour. One way to mitigate the effect of degraded communications would be to extrapolate other swarm member positions based on last known state and known mission parameters. The performance of the algorithm is sensitive to the selection of tuning parameters such as avoidance gain and desired separation. Higher avoidance gains when preforming formation flight can also lead to oscillatory behaviour.

The trade-off between computational efficiency and optimality is evident in the observed performance. The reactive, myopic nature of the algorithm enables low computational cost and scalability, with complexity growing linearly with swarm size. However, this comes at the expense of increased steady-state tracking error and the potential for oscillatory behaviour in scenarios involving competing objectives, such as formation control.

Overall, the results indicate that the proposed algorithm is well-suited to low-cost UAV platforms operating under constrained communication and computational conditions. A key outcome is that the demonstrated behaviours were achieved without modification to the core algorithm, requiring only changes to vector definitions and parameters to achieve a range of mission objectives. This modularity allows for a general purpose guidance algorithm implementation for UAV swarming.

7. Conclusions

This paper presents a computationally efficient, distributed swarm guidance algorithm designed for implementation on low-cost, resource-constrained UAVs. The core contribution is a modular vector-field-based framework that enables decentralized coordination without requiring global trajectory planning or computationally expensive prediction methods. Each UAV independently computes its desired motion by combining three types of vector fields: a *target vector* for nominal trajectory tracking, *agent vectors* for collision avoidance (and optionally cohesion), and *objective vectors* to achieve higher-level swarm behaviours.

The algorithm extends classical vector field methods by allowing multiple vector components to be combined, enabling flexible mission design. Importantly, it maps resultant vector magnitudes to feasible airspeed commands while enforcing platform constraints, ensuring effective operation on fixed-wing UAVs. The approach is reactive, avoiding receding-horizon optimization, which keeps computational complexity low, scaling linearly with swarm size, and is suitable for onboard execution.

Simulation results using ArduPilot SITL demonstrate the algorithm's effectiveness across several scenarios: collaborative circular loitering with equiangular spacing, line astern and line abreast formation flight with waypoint-based transits. In loitering tasks, UAVs achieved near-uniform angular separation (≈ 1 deg mean error) and maintained low cross-track errors (≈ 4.5 m), while dynamically adjusting speed to coordinate spacing. Formation flight simulations showed accurate path tracking but revealed oscillatory behaviours caused by competing attractive and repulsive vector components, highlighting tuning trade-offs.

Flight tests with fixed-wing UAVs validated real-world performance, confirming the emergence of stable, collision-free behaviours such as formation maintenance and cooperative loitering. Notably, the system exhibited beneficial emergent dynamics, such as sinusoidal approach trajectories to enable efficient formation merging that respect airspeed constraints. This emergent behaviour arise form the interaction of vector components and highlights the ability of the framework to generate feasible trajectories under the system constraints. The approach proved robust to limited communication bandwidth, though performance variability increased under communication delays. During flight test the vehicles travelling in circular trajectories achieved cross-track errors between ≈ 7 m and ≈ 15 m, whereas for straight line trajectories cross-track error was between 0.5 m and 2.4 m during convergence.

Future work will focus on improving robustness to communication delays, for example through state extrapolation or filtering techniques, and refining objective vector formulations to reduce oscillatory behaviour in tightly coupled formations. To optimize the performance of the algorithm, platform-specific models will be used in SITL to reduce the sim-to-real gap to enable the deployment of algorithms and gains directly from a simulated scenario to flight test.

Overall, the results demonstrate that complex swarm behaviours such as persistent observation, formation flight, and adaptive coordination can be achieved using lightweight, decentralized algorithms suitable for low-cost UAV platforms, with strong potential for scalable, real-world deployment.

8. Contact Author Email Address

john.mcguire2@defence.gov.au

9. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Johann Borenstein, Yoram Koren, et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[2] Weijia Yao, Héctor Garcia de Marina, Bohuan Lin, and Ming Cao. Singularity-free guiding vector field for robot navigation. *IEEE Transactions on Robotics*, 37(4):1206–1221, 2021.

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

[4] Wooyoung Jung, Seunghan Lim, Dongjin Lee, and Hyochoong Bang. Unmanned aircraft vector field path following with arrival angle control. *Journal of Intelligent & Robotic Systems*, 84(1):311–325, 2016.

[5] Hector Garcia De Marina, Yuri A Kapitanyuk, Murat Bronz, Gautier Hattenberger, and Ming Cao. Guidance algorithm for smooth trajectory tracking of a fixed wing uav flying in wind flows. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 5740–5745. IEEE, 2017.

[6] Yeongho Song, Seunghan Lim, Hyunsam Myung, Hokeun Lee, Junho Jeong, Heungsik Lim, and Hyondong Oh. Distributed swarm system with hybrid-flocking control for small fixed-wing uavs: Algorithms and flight experiments. *Expert Systems with Applications*, 229:120457, 2023.

[7] Hector Garcia De Marina, Zhiyong Sun, Murat Bronz, and Gautier Hattenberger. Circular formation control of fixed-wing uavs with constant speeds. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5298–5303. IEEE, 2017.

[8] Vinicius M Goncalves, Luciano CA Pimenta, Carlos A Maia, Bruno CO Dutra, and Guilherme AS Pereira. Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4):647–659, 2010.

[9] Yajing Wang, Xiangke Wang, Shulong Zhao, and Lincheng Shen. Vector field based sliding mode control of curved path following for miniature unmanned aerial vehicles in winds. *Journal of Systems Science and Complexity*, 31(1):302–324, 2018.

[10] Yuri A Kapitanyuk, Anton V Proskurnikov, and Ming Cao. A guiding vector-field algorithm for path-following control of nonholonomic mobile robots. *IEEE Transactions on Control Systems Technology*, 26(4):1372–1385, 2017.

[11] Thiem V. Pham, Dung V. Vu, Dong T. Nguyen, and Nhat N. Dong. Loitering formation of fixed-wing uav swarm under communication delay and switching topology. *IFAC-PapersOnLine*, 56(2):8512–8517, 2023. 22nd IFAC World Congress.

[12] Ximan Wang, Simone Baldi, Xuewei Feng, Changwei Wu, Hongwei Xie, and Bart De Schutter. A fixed-wing uav formation algorithm based on vector field guidance. *IEEE Transactions on Automation Science and Engineering*, 20(1):179–192, 2023.

[13] Eric W. Frew, Dale A. Lawrence, and Steve Morris. Coordinated standoff tracking of moving targets using lyapunov guidance vector fields. *Journal of Guidance, Control, and Dynamics*, 31(2):290–306, 2008.

[14] Sheng Xu and Kutluyıl Doğançay. Optimal sensor placement for 3-d angle-of-arrival target localization. *IEEE Transactions on Aerospace and Electronic Systems*, 53(3):1196–1211, 2017.