IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 May 2022, accepted 25 May 2022, date of publication 8 June 2022, date of current version 15 March 2023.

*Digital Object Identifier 10.1109/ACCESS.2022.3181244*

# Energy-Efficient Post-Failure Reconfiguration of Swarms of Unmanned Aerial Vehicles

**ANAM TAHIR[1], HASHEM HAGHBAYAN[1], (Member, IEEE), JARI M. BÖLING[2], AND JUHA PLOSILA[1], (Member, IEEE)**
[1]Autonomous Systems Laboratory, Department of Computing, Faculty of Technology, University of Turku, 20014 Turku, Finland
[2]Laboratory of Process and Systems Engineering, Åbo Akademi University, 20500 Turku, Finland

Corresponding author: Anam Tahir (anam.tahir@utu.fi)

This work was supported in part by the University of Turku; and in part by the Finnish Culture Foundation, Finland.

**ABSTRACT** In this paper, the reconfiguration of swarms of unmanned aerial vehicles after simultaneous failures of multiple nodes is considered. The objectives of the post-failure reconfiguration are to provide collision avoidance and smooth energy-efficient movement. To incorporate such a mechanism, three different failure recovery algorithms are proposed namely thin plate spline, distance- and time-optimal algorithms. These methods are tested on six swarms, with two variations on failing nodes for each swarm. Simulation results of reconfiguration show that the execution of such algorithms maintains the desired formations with respect to avoiding collisions at run-time. Also, the results show the effectiveness concerning the distance travelled, kinetic energy, and energy efficiency. As expected, the distance-optimal algorithm gives the shortest movements, and the time-optimal algorithm gives the most energy-efficient movements. The thin plate spline is also found to be energy-efficient and has less computational cost than the other two proposed methods. Despite the suggested heuristics, these are combinatorial in nature and might be hard to use in practice. Furthermore, the use of the regularization parameter $\lambda$ in thin plate spline is also investigated, and it is found that too large values on $\lambda$ can lead to incorrect locations, including multiple nodes on the same location. In fact, it is found that using $\lambda = 0$ worked well in all cases.

**INDEX TERMS** Unmanned aerial vehicles, formation maintenance, collision avoidance, failure recovery system, swarm intelligence, multi-drone systems.

## I. INTRODUCTION

In swarm systems, reliability and safety are crucial properties that can be improved by fault diagnosis technologies [1]. When focusing on navigation in a swarm of unmanned aerial vehicles (UAVs), the main challenges revolve around formation control and collision avoidance [2]–[5]. In any system, three main types of problems can occur at run-time, namely faults, errors, and failures [6]. These problems are interrelated in such a way that faults are the primary cause where a problem originates, an error is the consequence of a fault or multiple faults, and a failure is the ultimate manifestation of a problem in operation. As a solution, a system needs to be made resilient by design, providing a high degree of reliability and availability at run-time. The construction of such fail-safe mechanisms is application dependent. The operational phases include fault detection and diagnosis, evidence generation, assessment, and recovery [7]. Furthermore, in a swarm formation, the faults can be classified into component faults and topology faults depending upon their influence on the overall integrated system [8]–[11]. In the case of a topology fault, the system topology changes due to, for example, a faulty sensor link, a faulty communication link between nodes, and/or intrusions. Those faults in any module of a node, such as a local controller, sensor, and/or actuator, that do not alter the system topology, are called component faults. In this paper, any type of failure, such as engine failure or collision, that makes the UAVs disappear from the swarm is considered, and the target is to restore the formation as good as possible. During this type of partial failure within a swarm, the proposed failure recovery methods enable the system to continue functioning, preventing losses due to unexpected service unavailability or operation failures. This sequential process of deviation and recovery must be robust and reliable.

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Tang.

One of the main problems in distributed formation control of UAVs is to enable dynamic management of creation, maintenance, and termination of swarms. The control system should be robust and energy-efficient, being constantly aware of the formation of the swarm at mission time and able to perform reconfiguration in the case of failure of one or multiple nodes. In the considered scenario, when some UAVs in the swarm experience failures, the controller discards them from the swarm, changing the formation. The newly created formation is not necessarily efficient for the mission at hand, and, therefore, subsequent reconfiguration into a more efficient formation may be needed, taking into account the mission requirements. In this paper, a pre-specified initial formation for a swarm determines the disturbed formation immediately after a failure occurs on a number of UAVs. Three fail-safe reconfiguration schemes are proposed to map the disturbed formation due to the pre-specified formation that is inspired by the thin plate spline (TPS), distance- and time-optimal (DOA, TOA) algorithms, based on which UAVs have been reconfigured. The movement to the positions is handled by the setpoint tracking controller, implemented using a linear quadratic regulator (LQR) with integral action. This architecture is capable of managing transmission/actuation failures in swarms of UAVs. Along with this reliable mechanism, the minimization of energy consumption of the whole swarm is considered in the sequential process of deviation and reconfiguration. Furthermore, it is assumed that the swarm can (if necessary) be moved away from the obstacles prior to the reconfigurations therefore considering static and dynamic obstacles are out of the scope of this work. The main contributions of this paper are summarized as follows:

- Proposing a novel idea to reconfigure the formation of swarms of UAVs when simultaneous failures of multiple nodes have been diagnosed within a cluster, having the aim of formation control with collision avoidance.
- Three different algorithms for reconfiguration of swarms, TPS, DOA, and TOA are studied, which can provide reliability and availability at run-time. All the considered algorithms take into account the movement of all UAVs at the same time. Standard tree search algorithms such as A* do the path planning for one node at a time [12].
- The proposed algorithms are tested on six different swarms having two different failure cases each, in a total of twelve cases.
- Lastly, the performance of reconfigurations is compared on distance travelled, kinetic energy, and energy efficiency.

The rest of the paper is organized as follows. Section II covers the related work. In Section III, the development of the proposed failure recovery system is described. In Section IV, the techniques for energy-efficient post-failure reconfiguration of a swarm of UAVs are presented. In Sections V and VI, simulation set-up and results are elaborated respectively. Lastly, concluding remarks are presented in Section VII.

## II. RELATED WORK

The fault diagnosis and fault-tolerant controls are important to ensure the reliability and safety of any swarm system. Qin *et al.* in [8] present an extensive survey of the faults and the fault diagnosis algorithms and also classifies them as per their specifications. Similarly, Yang *et al.* in [13] build upon the trends and methodologies of fault-tolerant cooperative control of multi-drone systems which, despite being more challenging, are more flexible than single-robot systems. Complementing this, in their attempts to solve a constrained optimization problem for fault-tolerant control of a quadrotor, Chamseddine *et al.* have implemented the differential flatness in [14]. A fault-tolerant algorithm using the A* pathfinding method for multi-robot coverage path planning is proposed by Sun *et al.* in [12]. Furthermore, Liu *et al.* in [15] consider a linear multi-drone system comprising of multiple leaders, which has the ability to tolerate actuator faults and deal with input saturation. For this purpose, the control algorithms have been designed, with a backstepping approach, using local information of neighbouring nodes within a time-varying formation. A method of fixed-wing UAV swarm formation control using waypoints based on distributed ad hoc networks is elaborated by Suo *et al.* in [16]. A hypothesis related to a distributed fault-tolerant mechanism using a policy-based election algorithm for an uncertain environment is addressed by Wang *et al.* in [17]. Another approach to fault-tolerant control has been discussed by Wang *et al.* in [18] in which a cooperative fault-tolerant control has been illustrated for linear leader-follower networks with switching directed graph as the interaction typology network applicable to all nodes. This network structure functioned efficiently in the presence of multiple heterogeneous actuator faults that included not only the actuator bias fault but also the partial loss of effectiveness fault. A fault-tolerant formation control approach for UAVs of leader-follower type is discussed by Yu *et al.* in [19] against the actuator faults of only followers based on finite-time adaptive techniques. Raja *et al.* in [20] present a fault-tolerance module for a leader-follower mechanism to handle the actuator fault of a leader. In this algorithm, a new leader is chosen by using the locations of the follower and failed leader. The distance between the failed leader and followers is calculated using Euclidean distance, and the follower with minimum distance from the faulty leader is selected as a new leader for a swarm. A similar failure recovery strategy based on the shortest path planning/replanning problem is proposed by Tahir *et al.* in [21] for rerouting UAVs, which is the prequel work of this study.

## III. FAILURE RECOVERY SYSTEM

In most cases, single or multiple failure occurrences in a swarm introduce gaps, and it is desirable to reconfigure the swarm and fill the gaps, as shown in Fig. 1. This process raises a formation construction problem [22] that is widely covered in the literature. The main novelty of this work is to propose the architecture for reconfiguration of the swarms in case of failure of multiple nodes simultaneously. Also, the objective is to keep a safe distance while manoeuvring. Furthermore, all the UAVs are considered to be identical. Hence, it is not required to re-establish initial neighbouring states among the swarm. In the reconfiguration process, two main questions need to be addressed:

1) What is the best shape of the reduced swarm? This procedure is defined as the mapping problem.
2) What is the optimal movement of each node from the initial to final formation?

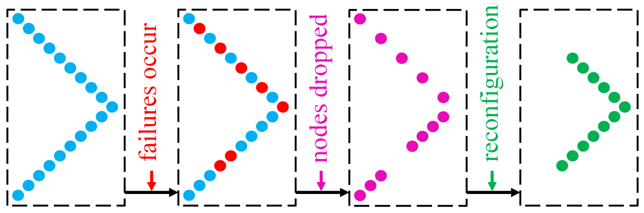


FIGURE 1. 2D representation of proposed failure recovery procedure.

Consider the reconfiguration of a swarm of UAVs from the initial to the final formation, as presented in Fig. 1. This process is divided into four stages. The predefined formation is given in the first stage. Failures occur by breaking the initial formation in the second stage, which is followed by the vacant positions in the third stage. Eventually, the remaining nodes reconfigure themselves in the original formation while minimizing the overall nodes' deviations.

In order to solve the proposed problem, the whole system is divided into two subsystems dealing with (a) reconfiguration of UAVs, and (b) tracking control, as shown in Fig. 2. In a multi-drone swarm, each active (non-failed) node $n \in \{1, .., N\}$ receives a reference command $r_n = r + \Delta_n$ where $r$ is the reference for the whole swarm and $\Delta_n$ is the offset for node $n$, which is needed to keep all the nodes apart. When failures occur, the reconfiguration algorithms update the constant offsets $\Delta = \left[\Delta_1 \ \Delta_2 \ \ldots \ \Delta_N\right]^{\mathrm{T}}$. Hence, the formation is re-established as good as possible with the current number of active nodes.

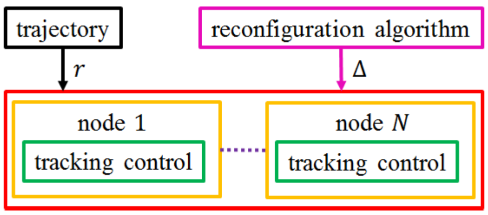


FIGURE 2. Proposed architecture of post-failure swarm reconfiguration.

The post-failure reconfiguration for the UAVs is designed using three different algorithms TPS, DOA and TOA. In all cases, the reference commands $r_n$ are updated by updating the $\Delta_n$s. Besides, the trajectory of each node is controlled by the LQR technique with integral action. Each method is described in the following sections.

## IV. RECONFIGURATION OF UAVs

### A. THIN PLATE SPLINE

Consider a point set registration [23]–[25] for robust non-rigid 2D transformation using TPS, which is commonly used to solve data interpolation and smoothing problems [26]. In [27], TPS is applied for reshaping the trajectories of the UAVs in a swarm formation due to obstacles. In this paper, TPS is used for reconfiguration of the UAVs in a swarm formation after simultaneous failures of multiple nodes.

A spline is a function defined by polynomials in a piece-wise manner. Spline curves are used for the approximation of complicated shapes via curve fitting due to their ease of use and non-complicated construction [26]. The algorithm is analyzed in 2D to make it simpler; consequently, two sets of correspondence data points, $X = x_i$ and $V = v_i$ where $i = 1, 2, 3, \ldots, n$ are considered. Here, the locations of a point in the scene (desired formation) and the model (initial formation) are given by $x_i$ and $v_i$ respectively. A mapping function $f(v_i)$ can be acquired while keeping the shape of the disturbed formation/function under consideration, by minimising the energy function

$$E_{TPS}(f) = \sum_{i=1}^{n} ||x_i - f(v_i)||^2 + \lambda \iint [(\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})]dxdy. \tag{1}$$

The amount of formation disturbance is evaluated by the energy function $E_{TPS}$. By minimizing the first error measurement term, data points of $V$ are mapped as closely as possible to the data points of $X$. The second regularization term is a penalty on the smoothness of $f$, and it is in a general case needed to make the mapping unique. In this case, one can put $\lambda = 0$, and still gets unique solutions, which also reduces the likelihood of a biased mapping $f$, see Fig. 8. Once the desired mapping is obtained, the constant offsets $\Delta$ are updated accordingly. For example, if node $n$ is mapped to the location of node $m$ then $\Delta_n = \Delta_m$. Each UAV in the model starts following the shortest path to reach its next position in the scene.

### B. DISTANCE- AND TIME-OPTIMAL ALGORITHMS

The full failure of an engine of a node in the swarm of UAVs using leader-follower tightly coupled formation is considered in [21]. As a solution, a bottom-up reconfiguration is imposed to bypass the failed node in order to keep the formation intact. For this study, the proposed method in [21] is extended for the reconfiguration of a multi-drone system after simultaneous failures of multiple nodes. The following two alternative algorithms are used for reconfiguration of the swarms, which are based on a combinatorial search with some application-dependent pruning heuristics.

The first algorithm minimises the total travelling distance

$$D_T = \sum_q \sqrt{(x_{q,r} - x_{q,i})^2 + (y_{q,r} - y_{q,i})^2} \tag{2}$$

**Algorithm 1** Pseudocode for Distance-Optimal Algorithm

```
1: Replace all the missing nodes $m$, starting with the node that has the lowest index, with an active node $n$, by testing all alternative nodes with index $n > m$, which has a direct line of sight to node $m$.
2: For each reconfiguration obtained in step 1, calculate the costs according to $D_T$ (Eq. 2).
3: Select the reconfiguration with the minimum cost.
4: If multiple minimum costs are obtained, then among these, choose the one smallest cost according to $D_M$ (Eq. 3). If this does not resolve it, then arbitrarily choose among the optimal ones the movement involving the node with the smallest index.
5: Update the constant offsets $\Delta$ accordingly.
```

**Algorithm 2** Pseudocode for Time-Optimal Algorithm

```
1: Replace all the missing nodes $m$, starting with the node that has the lowest index, with an active node $n$, by testing all alternative nodes with index $n > m$, which has a direct line of sight to node $m$.
2: For each reconfiguration obtained in step 1, calculate the costs according to $D_M$ (Eq. 3).
3: Select the reconfiguration with the minimum cost.
4: If multiple minimum costs are obtained, then among these, choose the one smallest cost according to $D_T$ (Eq. 2). If this does not resolve it, then arbitrarily choose among the optimal ones the movement involving the node with the smallest index.
5: Update the constant offsets $\Delta$ accordingly.
```

from initial $(x_{q,i}, y_{q,i})$ to reconfigured $(x_{q,r}, y_{q,r})$ positions where $q$ is the index of a reconfigured node. The second algorithm minimises the maximal individual travelling distance

$$D_M = \max_q \sqrt{(x_{q,r} - x_{q,i})^2 + (y_{q,r} - y_{q,i})^2}. \tag{3}$$

Algorithm 1 is distance-optimal, as it directly chooses the movement with the least sum of the distances. Algorithm 2 is time-optimal under the assumptions that all the travelling take place in parallel, and a longer distance takes a longer time to travel than a shorter one. This means that the longest distance travelled is the limiting factor. Thus, choosing the movement with the smallest maximal individual distance is time-optimal. Both algorithms fill all the vacant places, starting with the lowest index, by moving UAVs with higher to lower indices. The algorithms fill the missing nodes one at a time, and never reconsider prior movements, therefore they will eventually converge, filling the first $N$ vacant places of the swarm formation.

Collisions are avoided due to the following two features of the algorithms:

a) All the movements of nodes are done with a direct line of sight, meaning that the moving nodes are not colliding with the non-moving ones.

b) If two separate movements would have crossing paths, they would neither be distance- nor time-optimal. Thus, distance- or time-optimally moved nodes are not colliding with each other.

## V. SIMULATION SET-UP

The dynamics of each node in the swarm formation are based on the model of a quadcopter, i.e. a UAV that has four propellers with fixed pitch mechanically movable blades, as shown in Fig. 3, where $\theta$, $\phi$, and $\psi$ are defined as pitch, roll, and yaw respectively.

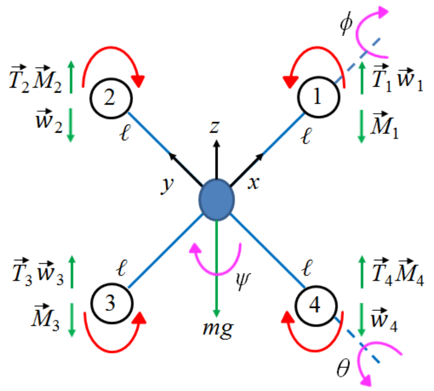


**FIGURE 3.** Kinemetics of the quadcopter.

The major forces acting on the quadcopter are the gravity $g$ and the thrust $T_i$, $i \in \{1, 2, 3, 4\}$, of the propellers. In this model, the inertial reference is the earth shown as $(x, y, z)$ that is the origin of the reference frame. The UAV is assumed to be a rigid body that has the constant mass symmetrically distributed with respect to the planes $(x, y)$, $(y, z)$, and $(x, z)$. The orientation of a quadcopter reference frame $(x, y, z)$ with respect to an inertial frame $(x, y, z)_0$ can be expressed mathematically in a state variable form [28]–[30], where translational and angular accelerations are given by

$$\begin{aligned} \dot{v}_x &= -v_z w_y + v_y w_z - g\sin\theta \\ \dot{v}_y &= -v_x w_z + v_z w_x + g\cos\theta\sin\phi \\ \dot{v}_z &= -v_y w_x + v_x w_y + g\cos\theta\cos\phi - \frac{T}{m} \end{aligned} \tag{4}$$

and

$$\begin{aligned} \dot{w}_x &= \frac{1}{J_x}(-w_y w_z(J_z - J_y) + M_x - \frac{k_{wT}}{k_{MT}} J_{mp} M_z w_y) \\ \dot{w}_y &= \frac{1}{J_y}(-w_x w_z(J_x - J_z) + M_y - \frac{k_{wT}}{k_{MT}} J_{mp} M_z w_x) \\ \dot{w}_z &= \frac{M_z}{J_z} \end{aligned} \tag{5}$$

respectively. The thrust produced by each propeller $T_i$ is translated into a total thrust $T$ and the reactive torques $M_i$, $i \in \{x, y, z\}$, which are affecting the rotations about the corresponding axis. The $J_i$, $i \in \{x, y, z\}$, is known as the moment of inertia along the corresponding axis, and $J_{mp}$ is the moment of inertia of a motor with propeller. The angular velocities of propellers are assumed to be proportional to thrusts of propellers, i.e. $w_i = k_{wT} T_i$, $i \in \{x, y, z\}$. Similarly, the reactive moments of propellers are assumed to be proportional to the thrust of propellers, i.e. $M_i = k_{MT} T_i, i \in \{x, y, z\}$. The velocities corresponding to Equations (4)-(5) are

$$\begin{aligned}\dot{x} &= v_x \cos\psi \cos\theta + v_y(-\sin\psi \cos\phi + \cos\psi \sin\theta \\ &\quad \sin\phi) + v_z(\sin\psi \sin\phi + \cos\psi \sin\theta \cos\phi) \\ \dot{y} &= v_x \sin\psi \cos\theta + v_y(\cos\psi \cos\phi + \sin\psi \sin\theta \sin\phi) \\ &\quad + v_z(-\cos\psi \sin\phi + \sin\psi \sin\theta \cos\phi) \\ \dot{z} &= v_x \sin\theta - v_y \cos\theta \sin\phi - v_z \cos\theta \cos\phi\end{aligned} \tag{6}$$

and

$$\begin{aligned}\dot{\theta} &= w_y \cos\phi - w_z \sin\phi \\ \dot{\phi} &= w_x + w_y \sin\phi \tan\theta + w_z \cos\phi \tan\theta \\ \dot{\psi} &= w_y \frac{\sin\phi}{\cos\theta} + w_z \frac{\cos\phi}{\cos\theta}\end{aligned} \tag{7}$$

respectively. The Equations (4)-(7) represent the complete nonlinear model of a quadcopter, composed of twelve states, four inputs, and twelve outputs. These equations are furthermore linearized, resulting in

$$\begin{aligned}\dot{\mathbf{x}} &= \left[-g\theta \; g\phi \; -\tfrac{T}{m} \; \tfrac{M_x}{J_x} \; \tfrac{M_y}{J_y} \; \tfrac{M_z}{J_z} \; w_y \; w_x \; w_z \; v_x \; v_y \; -v_z\right]^{\mathrm{T}} \\ \mathbf{y} &= \mathbf{x}\end{aligned} \tag{8}$$

which are used for controller designing. The system parameters are taken from [30] and illustrated in Table 1.

**TABLE 1. System parameters [30].**

| Symbol | Quantity | Value |
|---|---|---|
| $g$ | gravitational force | $9.81\ \mathrm{m/s^2}$ |
| $\ell$ | length of the fixed pitch to mechanically movable blades | $0.2$ m |
| $m$ | mass of quadcopter | $0.8$ kg |
| $J_{mp}$ | moment of inertia of motor with propeller | $\approx 0$ |
| $J_x, J_y$ | moment of inertia w.r.t. axis x, y | $1.8 \times 10^{-3}\ \mathrm{kgm^2}$ |
| $J_z$ | moment of inertia w.r.t. axis z | $1.5 \times 10^{-3}\ \mathrm{kgm^2}$ |
| $k_{MT}$ | ratio of the reactive moment and thrust | $0.1$ m |

### A. TRACKING CONTROL OF UAVs

The motion of each drone is controlled using a standard LQR with integral action [21], [31]–[34]. LQR is a method for the design of an optimal state feedback law based on a linear model, in this case Equation (8). Fig. 4 shows the control system of each node, which is based on the feedback law that consists of a sum of a proportional and integral terms. Both contribute to generating the thrusts $T$ and torques $M_i$, $i \in \{x, y, z\}$.

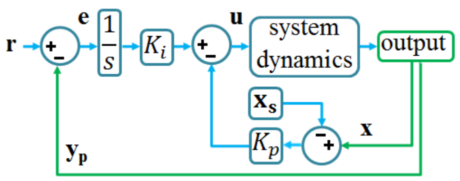


**FIGURE 4. Block diagram of LQR with integral action.**

The states that are relevant to control without offset from setpoints are collected in a vector $\mathbf{y_p} = [x, y, z]^T$, and the final feedback law is

$$\mathbf{u} = \frac{K_i}{s}(\mathbf{r} - \mathbf{y_p}) + K_p(\mathbf{x_s} - \mathbf{x}) \tag{9}$$

where $K_i$ is the integral gain, $\mathbf{r}$ is the setpoints for $\mathbf{y_p}$, $K_p$ is the state feedback gain, and $\mathbf{x_s}$ is the setpoints of the states $\mathbf{x}$. The weight matrices $Q$ and $R$ are given in the Appendix. More details are provided in our earlier work [21], [31], [32].

## VI. RESULTS

Autonomous UAVs act in pursuit of their agenda. The local assignments are evaluated even when serving the requests of other nodes. The requests can be refused in case of a weaker notion of autonomy [35]. In this paper, such type of system is protected against performance degradation beyond a point where a node effectively becomes useless. The proposed architecture in Fig. 2 consists of a base case in which reconfiguration requests are forwarded to the nodes of a swarm in order to restore the formation shape. Secondly, the nodes acknowledge the requests using local control units for smooth navigation. Hence, in this approach, each post-failure reconfiguration algorithm, i.e. TPS with $\lambda = 0$, DOA, and TOA as discussed in Sections III and IV, decide the reconfiguration of the UAVs for each vacant pose within a swarm, and an associated LQR with integral action is subjected to track it. Each node is simulated using the nonlinear model equations of a quadcopter, described in Section V.

Six different swarms of UAVs are considered. The number of UAVs in each swarm and the two different failure cases are listed in Table 2. The system is simulated in Simulink® with default parameters under MATLAB 2018b, and the results are shown in Figs. 5 and 6. The red, blue, and green markers represent the fully failed, active, and reconfigured nodes respectively. To restore the initial formation as good as possible, exact coordinates of the active nodes are obtained and all the possible combinations for reconfiguration are examined, and the decision is made accordingly.

**TABLE 2. Swarm sizes and failure cases.**

| Swarm # | # of UAVs | Index of failed nodes | |
|---|---|---|---|
| | | Failure case 1 | Failure case 2 |
| 1 | 5 | $\{1, 5\}$ | $\{2\}$ |
| 2 | 6 | $\{1\}$ | $\{2, 3, 4\}$ |
| 3 | 7 | $\{1, 3, 6\}$ | $\{4\}$ |
| 4 | 8 | $\{1, 4\}$ | $\{3, 5, 6\}$ |
| 5 | 9 | $\{3, 4, 6, 7\}$ | $\{1, 2, 3, 7, 8\}$ |
| 6 | 12 | $\{3, 5, 7, 11\}$ | $\{2, 5, 6, 9, 10, 12\}$ |

For all the reformed movements, the Euclidean distance

$$D_C(p, q) = \min_t \sqrt{(x_p(t) - x_q(t))^2 + (y_p(t) - y_q(t))^2} \tag{10}$$

is calculated and presented in Tables 3–5 where $t$ is time, and $D_C(p, q)$ is the minimum distance that should be greater than

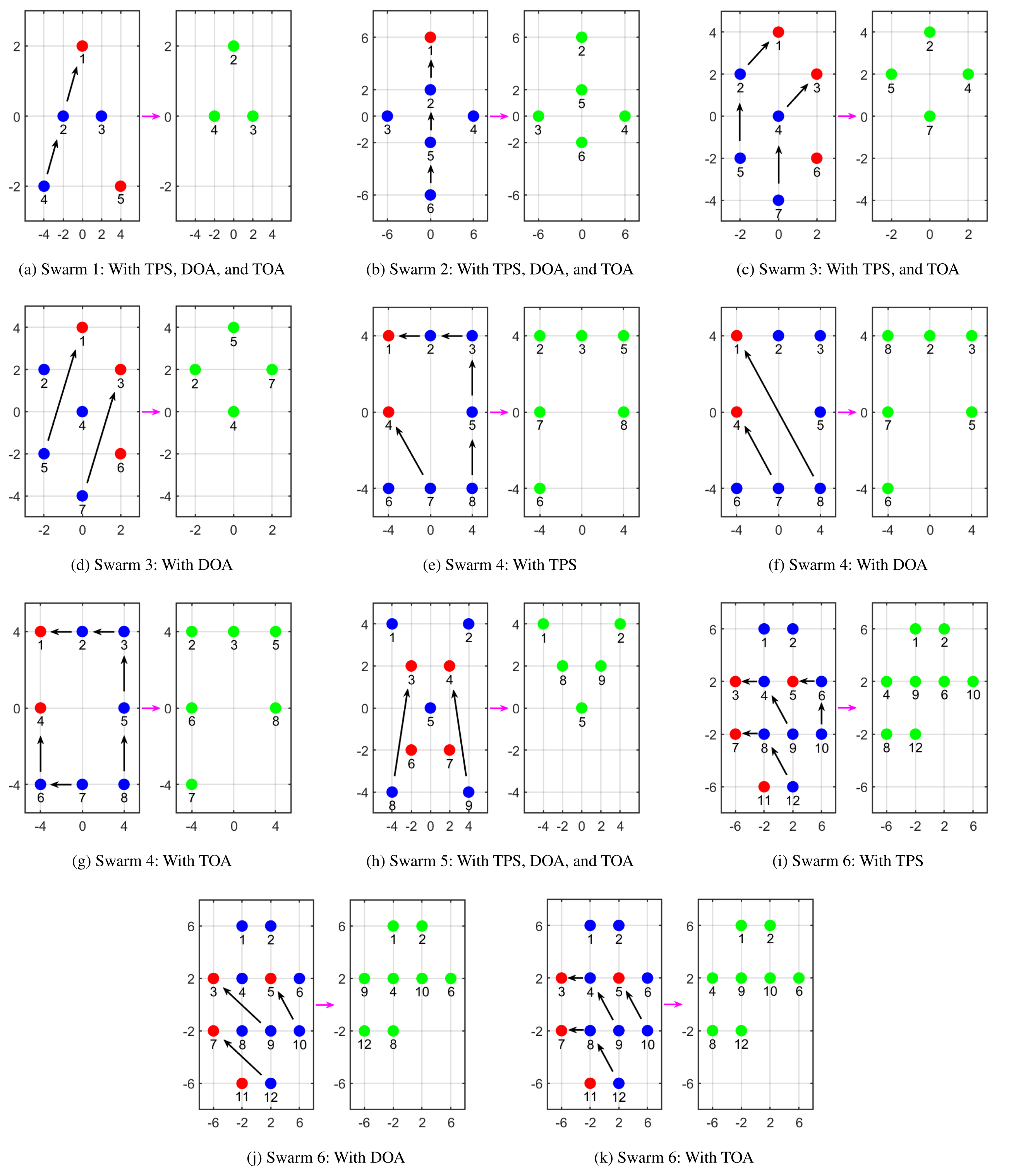
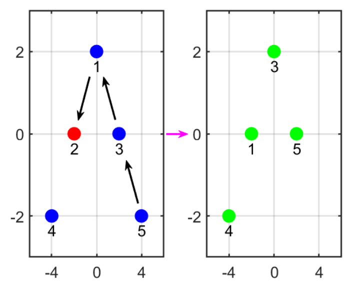
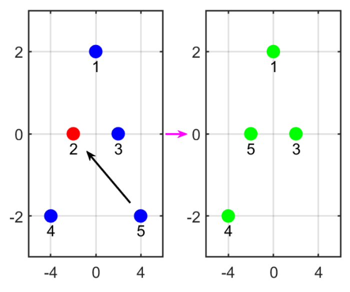
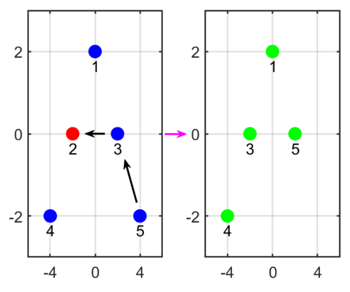
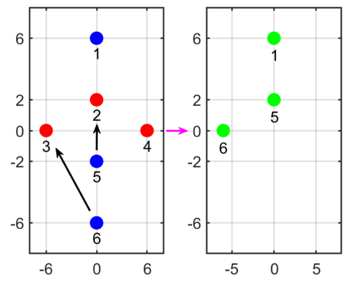


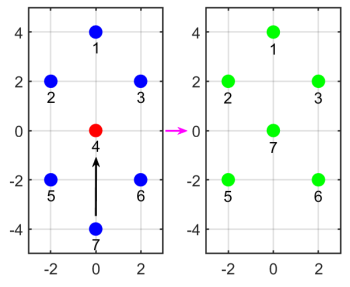
**FIGURE 5.** Failure case 1 — Reconfiguration of the swarms in 2D where ● is a failed node, ● is an active node, and ● is the position after reconfiguration. The omitted axis labels are x (m) horizontally, and y (m) vertically.

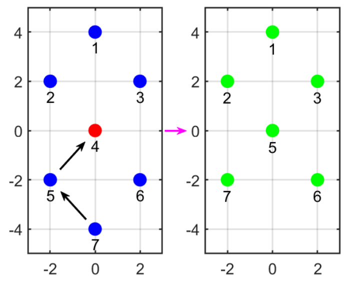

(a) Swarm 1: With TPS

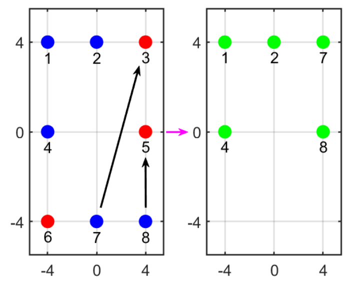

(b) Swarm 1: With DOA

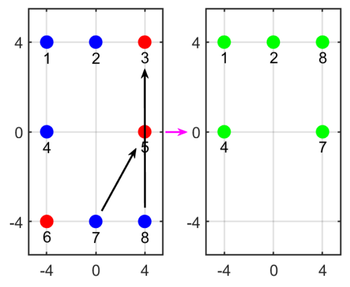

(c) Swarm 1: With TOA

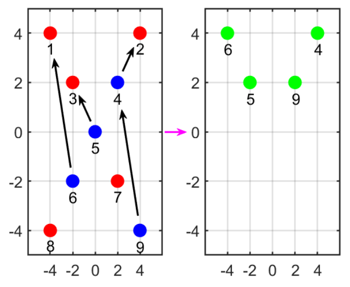

(d) Swarm 2: With TPS, DOA, and TOA

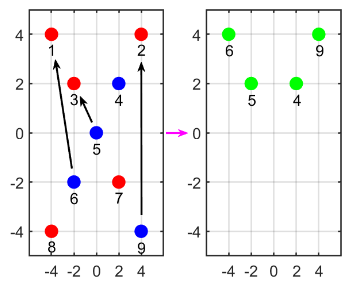

(e) Swarm 3: With TPS, and DOA

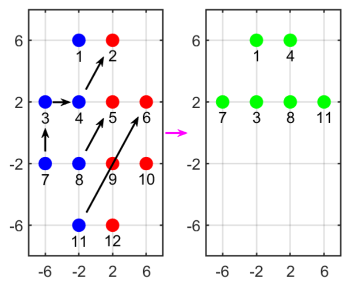

(f) Swarm 3: With TOA

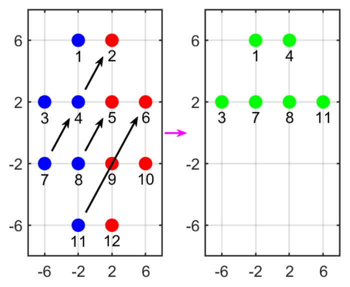

(g) Swarm 4: With TPS, and DOA


(h) Swarm 4: With TOA


(i) Swarm 5: With TPS, and TOA


(j) Swarm 5: With DOA


(k) Swarm 6: With TPS


(l) Swarm 6: With DOA

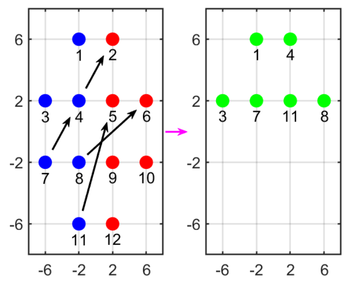

(m) Swarm 6: With TOA

**FIGURE 6. Failure case 2 – Reconfiguration of the swarms in 2D where ● is a failed node, ● is an active node, and ● is the position after reconfiguration. The omitted axis labels are x (m) horizontally, and y (m) vertically.**

$2r_d$ between nodes $p$ and $q$ to avoid collisions, as $r_d$ is the radius of a UAV. Moreover, the total kinetic energy

$$KE = \max(0.5m(\sum_{q=1}^{n} \mathbf{v}^2)) \tag{11}$$

describes how much work is conserved in the process of the swarm movement. The performance of the swarm to redirect its manoeuvres in terms of total energy consumption

$$E = \sum_{q=1}^{n} \int_{t_i}^{t_f} (P_{total} - P_{hover})dt \tag{12}$$

is calculated where $t_i$ is the initial time and $t_f$ is the time where reconfiguration ends. This power is needed to generate thrust and the force of the thrust can be related in a nonlinear way. Generally, $P^2 \propto T^3$ where $P$ and $T$ are defined as power and thrust respectively. Furthermore, all the obtained performance results are elaborated in Table 6.

**TABLE 3.** $D_C(p, q)$ **in (m) of the UAVs in the swarm after reconfiguration. To avoid collisions,** $D_C(p, q) > 2r_d$**.**

| Nodes | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|
| $p, q$ | TPS | DOA | TOA | TPS | DOA | TOA |
| 1, 3 | – | – | – | 2.0000 | 2.8284 | 2.0000 |
| 1, 4 | – | – | – | 2.8284 | 5.6569 | 5.6569 |
| 1, 5 | – | – | – | 4.0000 | 2.5181 | 2.8284 |
| 2, 3 | 2.8284 | 2.8284 | 2.8284 | – | – | – |
| 2, 4 | 2.8284 | 2.8284 | 2.8284 | – | – | – |
| 3, 4 | 4.0000 | 4.0000 | 4.0000 | 5.6569 | 6.3246 | 2.8284 |
| 3, 5 | – | – | – | 2.8284 | 1.2289 | 2.8225 |
| 4, 5 | – | – | – | 6.3246 | 2.8284 | 6.3246 |

(a) Reconfiguration 1

| Nodes | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|
| $p, q$ | TPS | DOA | TOA | TPS | DOA | TOA |
| 1, 5 | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| 1, 6 | – | – | – | 8.4853 | 8.4853 | 8.4853 |
| 2, 3 | 6.3246 | 6.3246 | 6.3246 | – | – | – |
| 2, 4 | 6.3246 | 6.3246 | 6.3246 | – | – | – |
| 2, 5 | 4.0000 | 4.0000 | 4.0000 | – | – | – |
| 2, 6 | 8.0000 | 8.0000 | 8.0000 | – | – | – |
| 3, 4 | 12.0000 | 12.0000 | 12.0000 | – | – | – |
| 3, 5 | 6.0000 | 6.0000 | 6.0000 | – | – | – |
| 3, 6 | 6.3246 | 6.3246 | 6.3246 | – | – | – |
| 4, 5 | 6.0000 | 6.0000 | 6.0000 | – | – | – |
| 4, 6 | 6.3246 | 6.3246 | 6.3246 | – | – | – |
| 5, 6 | 4.0000 | 4.0000 | 4.0000 | 3.7556 | 3.7556 | 3.7556 |

(b) Reconfiguration 2

| Nodes | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|
| $p, q$ | TPS | DOA | TOA | TPS | DOA | TOA |
| 1, 2 | – | – | – | 2.8284 | 2.8284 | 2.8284 |
| 1, 3 | – | – | – | 2.8284 | 2.8284 | 2.8284 |
| 1, 5 | – | – | – | 6.3246 | 6.3246 | 4.0000 |
| 1, 6 | – | – | – | 6.3246 | 6.3246 | 6.3246 |
| 1, 7 | – | – | – | 4.0000 | 4.0000 | 6.3246 |
| 2, 3 | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| 2, 4 | 2.8284 | 2.8284 | 2.8284 | – | – | – |
| 2, 5 | 2.8283 | 1.2389 | 2.8283 | 4.0000 | 4.0000 | 2.8281 |
| 2, 6 | – | – | – | 5.6569 | 5.6569 | 5.6569 |
| 2, 7 | 4.0000 | 3.7829 | 4.0000 | 2.8284 | 2.8284 | 4.0000 |
| 3, 5 | – | – | – | 5.6569 | 5.6569 | 2.8284 |
| 3, 6 | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| 3, 7 | – | – | – | 2.8284 | 2.8284 | 5.6562 |
| 4, 5 | 2.8237 | 1.3016 | 2.8237 | – | – | – |
| 4, 7 | 2.8283 | 1.2389 | 2.8283 | – | – | – |
| 5, 6 | – | – | – | 4.0000 | 4.0000 | 2.8284 |
| 5, 7 | 2.8284 | 2.8284 | 2.8284 | 2.0000 | 2.0000 | 2.0000 |
| 6, 7 | – | – | – | 2.0000 | 2.0000 | 2.8230 |

(c) Reconfiguration 3

**TABLE 4.** $D_C(p, q)$ **in (m) of the UAVs in the swarm after reconfiguration. To avoid collisions,** $D_C(p, q) > 2r_d$**.**

| Nodes | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|
| $p, q$ | TPS | DOA | TOA | TPS | DOA | TOA |
| 1, 2 | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| 1, 4 | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| 1, 7 | – | – | – | 7.1010 | 7.1010 | 8.4374 |
| 1, 8 | – | – | – | 8.9443 | 8.9443 | 8.0000 |
| 2, 3 | 4.0000 | 4.0000 | 4.0000 | – | – | – |
| 2, 4 | – | – | – | 5.6569 | 5.6569 | 5.6569 |
| 2, 5 | 5.6462 | 5.6569 | 5.6462 | – | – | – |
| 2, 6 | 8.0000 | 8.9443 | 4.0000 | – | – | – |
| 2, 7 | 4.0000 | 5.6567 | 8.0000 | 3.5565 | 3.5565 | 5.6567 |
| 2, 8 | 8.4416 | 2.7825 | 8.4416 | 5.6569 | 5.6569 | 4.0000 |
| 3, 5 | 2.7852 | 4.0000 | 2.7852 | – | – | – |
| 3, 6 | 8.9443 | 11.3137 | 5.6569 | – | – | – |
| 3, 7 | 5.6569 | 8.4374 | 8.9443 | – | – | – |
| 3, 8 | 5.6560 | 5.5304 | 5.6560 | – | – | – |
| 4, 7 | – | – | – | 5.3036 | 5.3036 | 5.6462 |
| 4, 8 | – | – | – | 8.0000 | 8.0000 | 8.0000 |
| 5, 6 | 8.9443 | 8.9443 | 8.9443 | – | – | – |
| 5, 7 | 5.6569 | 5.6462 | 5.6569 | – | – | – |
| 5, 8 | 4.0000 | 2.7031 | 4.0000 | – | – | – |
| 6, 7 | 2.8752 | 2.8752 | 2.8712 | – | – | – |
| 6, 8 | 8.0000 | 5.7803 | 8.0000 | – | – | – |
| 7, 8 | 4.0000 | 2.9053 | 4.0000 | 2.8902 | 2.8902 | 2.8773 |

(d) Reconfiguration 4

| Nodes | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|
| $p, q$ | TPS | DOA | TOA | TPS | DOA | TOA |
| 1, 2 | 8.0000 | 8.0000 | 8.0000 | – | – | – |
| 1, 5 | 5.6569 | 5.6569 | 5.6569 | – | – | – |
| 1, 8 | 2.8284 | 2.8284 | 2.8284 | – | – | – |
| 1, 9 | 6.3246 | 6.3246 | 6.3246 | – | – | – |
| 2, 5 | 5.6569 | 5.6569 | 5.6569 | – | – | – |
| 2, 8 | 6.3246 | 6.3246 | 6.3246 | – | – | – |
| 2, 9 | 2.8284 | 2.8284 | 2.8284 | – | – | – |
| 4, 5 | – | – | – | 2.8284 | 2.8230 | 2.8284 |
| 4, 6 | – | – | – | 5.6487 | 5.0229 | 5.6487 |
| 4, 9 | – | – | – | 2.8284 | 2.0000 | 2.8284 |
| 5, 6 | – | – | – | 2.0001 | 2.0001 | 2.0001 |
| 5, 8 | 2.5412 | 2.5412 | 2.5412 | – | – | – |
| 5, 9 | 2.5412 | 2.5412 | 2.5412 | 4.0000 | 5.0169 | 4.0000 |
| 6, 9 | – | – | – | 6.3246 | 6.3246 | 6.3246 |
| 8, 9 | 4.0000 | 4.0000 | 4.0000 | – | – | – |

(e) Reconfiguration 5

**TABLE 5.** $D_C(p,q)$ **in (m) of the UAVs in the swarm after reconfiguration. To avoid collisions,** $D_C(p,q) > 2r_d$**.**

| Nodes | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|
| $p,q$ | TPS | DOA | TOA | TPS | DOA | TOA |
| $1,2$ | 4.0000 | 4.0000 | 4.0000 | – | – | – |
| $1,3$ | – | – | – | 4.0000 | 5.6569 | 5.6569 |
| $1,4$ | 4.0000 | 4.0000 | 4.0000 | 2.7811 | 2.7811 | 2.7811 |
| $1,6$ | 5.6569 | 8.9443 | 8.9443 | – | – | – |
| $1,7$ | – | – | – | 5.6569 | 4.0000 | 4.0000 |
| $1,8$ | 8.0000 | 8.0000 | 8.0000 | 5.6567 | 5.6567 | 7.0786 |
| $1,9$ | 4.0000 | 5.3448 | 4.0000 | – | – | – |
| $1,10$ | 8.9443 | 5.6569 | 5.6569 | – | – | – |
| $1,11$ | – | – | – | 8.4371 | 8.4371 | 5.6569 |
| $1,12$ | 8.0000 | 8.9443 | 8.0000 | – | – | – |
| $2,4$ | 5.6569 | 5.6569 | 5.6569 | – | – | – |
| $2,6$ | 4.0000 | 5.6569 | 5.6569 | – | – | – |
| $2,8$ | 8.9443 | 8.9443 | 8.9443 | – | – | – |
| $2,9$ | 5.6567 | 7.0786 | 5.6567 | – | – | – |
| $2,10$ | 5.6569 | 4.0000 | 4.0000 | – | – | – |
| $2,12$ | 8.9443 | 10.6786 | 8.9443 | – | – | – |
| $3,4$ | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| $3,7$ | – | – | – | 2.7852 | 2.7811 | 2.7811 |
| $3,8$ | – | – | – | 3.9999 | 5.6462 | 5.6569 |
| $3,11$ | – | – | – | 7.0936 | 8.3581 | 7.1010 |
| $4,6$ | 8.0000 | 8.0000 | 8.0000 | – | – | – |
| $4,7$ | – | – | – | 5.6569 | 5.6569 | 5.6569 |
| $4,8$ | 4.0000 | 4.0000 | 4.0000 | 4.0000 | 4.0000 | 4.0000 |
| $4,9$ | 4.0000 | 1.7357 | 4.0000 | – | – | – |
| $4,10$ | 8.9443 | 4.0000 | 8.0000 | – | – | – |
| $4,11$ | – | – | – | 5.6569 | 5.6569 | 4.0000 |
| $4,12$ | 5.6569 | 5.3448 | 5.6569 | – | – | – |
| $6,8$ | 8.9443 | 8.9443 | 8.9443 | – | – | – |
| $6,9$ | 3.9999 | 5.6569 | 5.6462 | – | – | – |
| $6,10$ | 2.7852 | 2.7811 | 2.7811 | – | – | – |
| $6,12$ | 5.6569 | 8.9326 | 8.4374 | – | – | – |
| $7,8$ | – | – | – | 4.0000 | 4.0000 | 4.0000 |
| $7,11$ | – | – | – | 5.6569 | 5.6473 | 3.9997 |
| $8,9$ | 4.0000 | 1.8656 | 4.0000 | – | – | – |
| $8,10$ | 8.0000 | 5.6569 | 8.0000 | – | – | – |
| $8,11$ | – | – | – | 2.7488 | 2.7488 | 2.7972 |
| $8,12$ | 4.0000 | 1.7357 | 4.0000 | – | – | – |
| $9,10$ | 4.0000 | 4.0000 | 4.0000 | – | – | – |
| $9,12$ | 4.0000 | 4.0000 | 4.0000 | – | – | – |
| $10,12$ | 5.6569 | 5.6569 | 5.6569 | – | – | – |

(f) Reconfiguration 6

For comparison of the methods $\mathcal{M} = \{TPS, DOA, TOA\}$,

$$D_C\% = \frac{\min_{\mathcal{M}}(\min_{(p,q)} D_C(p,q))}{\max_{\mathcal{M}}(\min_{(p,q)} D_C(p,q))} \times 100\%, \quad (13)$$

$$D_T\% = \frac{\min_{\mathcal{M}} D_T}{D_T} \times 100\%, \quad (14)$$

**TABLE 6.** **Overall performance of the UAVs in the swarms after reconfigurations.**

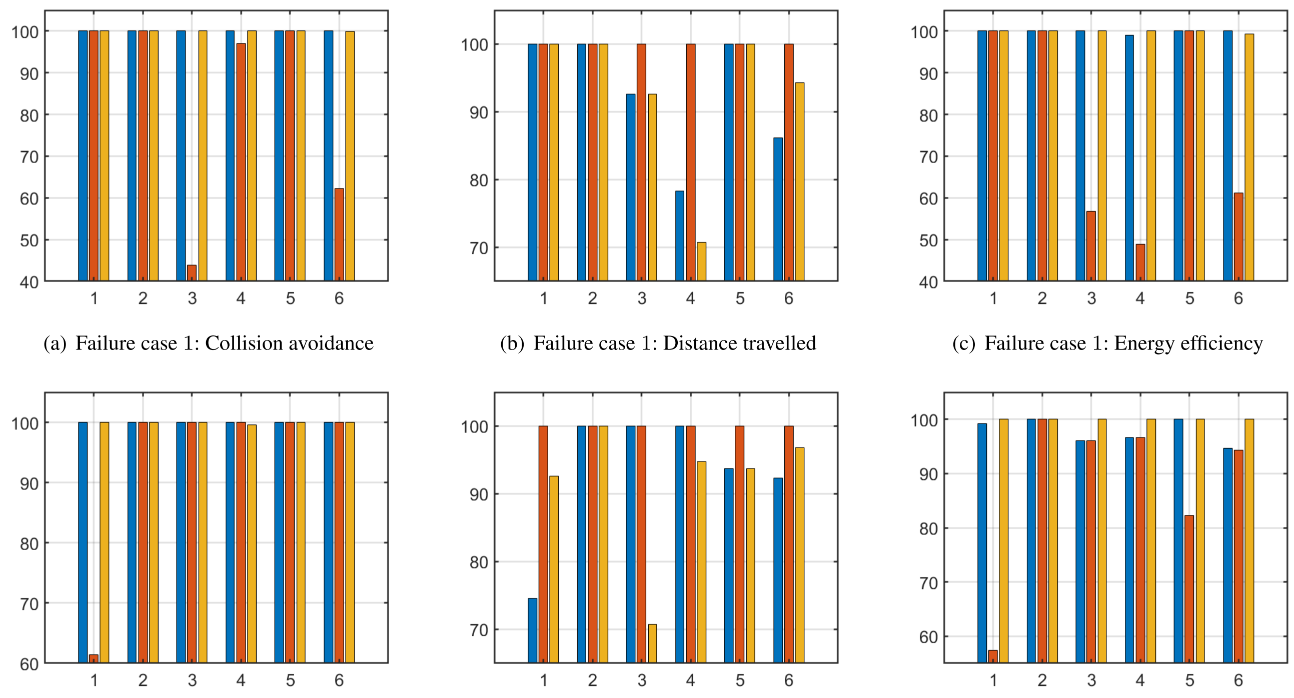
| Swarm # | Algorithm | Failure case 1 | | | Failure case 2 | | |
|---|---|---|---|---|---|---|---|
| | | $D_T$ (m) | $KE$ (J) | $E$ (J) | $D_T$ (m) | $KE$ (J) | $E$ (J) |
| 1 | TPS | 5.6569 | 0.8742 | 0.4301 | 8.4853 | 1.3113 | 0.6451 |
| | DOA | 5.6569 | 0.8742 | 0.4301 | 6.3246 | 2.2279 | 1.1136 |
| | TOA | 5.6569 | 0.8742 | 0.4301 | 6.8284 | 1.3113 | 0.6397 |
| 2 | TPS | 12.0000 | 2.6597 | 1.3434 | 12.4853 | 5.0333 | 2.6359 |
| | DOA | 12.0000 | 2.6597 | 1.3434 | 12.4853 | 5.0333 | 2.6359 |
| | TOA | 12.0000 | 2.6597 | 1.3434 | 12.4853 | 5.0333 | 2.6359 |
| 3 | TPS | 13.6569 | 2.6460 | 1.3257 | 4.0000 | 0.8866 | 0.4478 |
| | DOA | 12.6491 | 4.5091 | 2.3322 | 4.0000 | 0.8866 | 0.4478 |
| | TOA | 13.6569 | 2.6460 | 1.3257 | 5.6569 | 0.8742 | 0.4301 |
| 4 | TPS | 21.6569 | 5.2958 | 2.6425 | 12.9443 | 5.5572 | 2.9748 |
| | DOA | 16.9706 | 9.6129 | 5.3612 | 12.9443 | 5.5572 | 2.9748 |
| | TOA | 24.0000 | 5.2740 | 2.6175 | 13.6569 | 5.4684 | 2.8742 |
| 5 | TPS | 12.6491 | 4.5091 | 2.3322 | 18.3060 | 5.3812 | 2.7623 |
| | DOA | 12.6491 | 4.5091 | 2.3322 | 17.1530 | 6.3828 | 3.3577 |
| | TOA | 12.6491 | 4.5091 | 2.3322 | 18.3060 | 5.3812 | 2.7623 |
| 6 | TPS | 27.3137 | 7.0636 | 3.5170 | 30.6274 | 13.1444 | 7.1312 |
| | DOA | 23.5454 | 11.0216 | 5.7515 | 28.2843 | 13.1675 | 7.1563 |
| | TOA | 24.9706 | 7.0853 | 3.5420 | 29.2023 | 12.8404 | 6.7491 |

and

$$E_{\%} = \frac{\min_{\mathcal{M}} E}{E} \times 100\% \quad (15)$$

are defined as performance indices for collision avoidance, distance travelled, and energy efficiency respectively. The obtained results are depicted in Fig. 7. For both failure cases 1 and 2, it is evident from the results that all the algorithms work well for the post-failure reconfiguration of UAVs. Both TPS and TOA techniques fulfil the collision avoidance constraint by maintaining better separations from the respective nodes in comparison to DOA. Secondly, the amount of total distance travelled is minimum using DOA as opposed to TPS and TOA. On the other hand, using TPS and TOA, systems are more efficient in terms of the consumption of less energy while reconfiguring, and their results are quite close to each other. However, for the failure case 2, TOA wins over TPS with respect to energy efficiency.

The TPS algorithm, as discussed in Section IV-A, is tested with different weights of $\lambda$ on both failure cases 1 and 2. For example, in the failure case 2, possible collision is seen for the reconfiguration of the swarm 1 with $\lambda > 0.4062$, see Fig. 8. In all cases, there is an upper limit on $\lambda$, after which the TPS gives incorrect reconfiguration coordinates or results in collisions. The borderline values of $\lambda$ are reported in Table 7. As can be seen, smaller values on $\lambda$ are required in the second failure cases that are more challenging for TPS to handle. It seems that the behaviour of TPS reconfiguration is sometimes unpredictable as mentioned in [36].



(a) Failure case 1: Collision avoidance (b) Failure case 1: Distance travelled (c) Failure case 1: Energy efficiency

(d) Failure case 2: Collision avoidance (e) Failure case 2: Distance travelled (f) Failure case 2: Energy efficiency

**FIGURE 7. Performance indices for the swarm reconfigurations where 🟦 is TPS, 🟥 is DOA, and 🟧 is TOA. The omitted axis labels are swarm # horizontally, and y (%) vertically.**

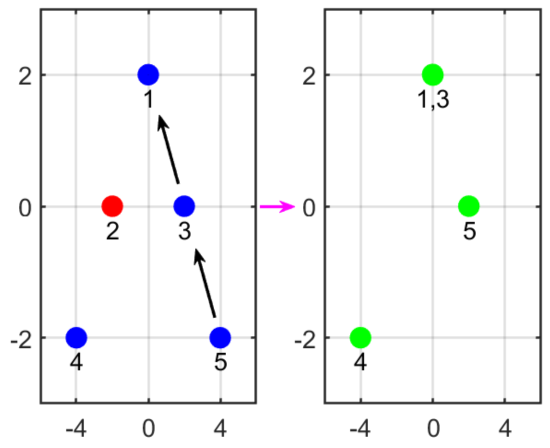


**FIGURE 8. With TPS λ > 0.4062: Failure case 2 – Reconfiguration of the swarm 1 in 2D where ● is a failed node, ● is an active node, and ● is position after reconfiguration. The omitted axis labels are x (m) horizontally, and y (m) vertically.**

**TABLE 7. The upper limit of the TPS regularization parameter λ in different cases.**

| Swarm # | Upper limit of $\lambda$ | |
|---|---|---|
| | Failure case 1 | Failure case 2 |
| 1 | 8.8748 | 0.4062 |
| 2 | 11.0355 | 0.0530 |
| 3 | 6.2639 | 0.4805 |
| 4 | 1.0838 | 0.3597 |
| 5 | 1.2950 | 0.9239 |
| 6 | 2.3437 | 0.0785 |

## VII. CONCLUSION

In this paper, the failure recovery architecture that considers the simultaneous failures of multiple nodes is proposed. This architecture is divided into two subsystems; post-failure reconfiguration of UAVs and their control, having the overall purpose of formation maintenance with collision avoidance. For the evaluation of the reconfiguration mechanism, three different algorithms comprising of TPS, DOA, and TOA are presented. These methods are tested on six different swarms having two different failure cases each, in a total of twelve cases, which have a different number of failing nodes. For the tracking control of each node, LQR with an integral action technique is used. The different reconfiguration algorithms are compared on collision avoidance, total distance travelled, kinetic energy, and energy efficiency. It is evident from the results that for both failure cases 1 and 2, all three algorithms i.e. TPS with $\lambda = 0$, DOA, and TOA are effective and work well for the post-failure reconfiguration of UAVs. The UAVs are dynamically and reliably reconfigured as fast as possible without collisions and maintaining the desired formation. Furthermore, another important contribution is the trade-off between the total distance travelled by the swarms and their corresponding energy efficiency. From the performance indices for the swarm reconfiguration, as can be expected, it is clear that the DOA is most efficient when it comes to travelled distance. On the other hand, TPS and TOA perform clearly better than DOA when it comes to collision margins and energy efficiency. There is a minor difference between the results of TPS and TOA, but one can say that TOA is slightly more energy efficient than TPS. Despite the suggested heuristics, DOA and TOA are combinatorial in nature and might be hard to use in practice. The performance of the TPS is quite well and can be considered for the reconfiguration of swarms. Furthermore, it is also found that the use of too large regularization parameter $\lambda$ will result in the mapping of nodes to incorrect locations, including multiple nodes on the same location (i.e. collisions), which is of course not desired. In fact, it is found that the use of $\lambda = 0$ worked in all tests, so this can be recommended.

## APPENDIX

$$Q = \text{diag}\left(\begin{bmatrix} 0.5, 0.8, 1000, 0.5, 0.5, 0.5, 1, 1, \\ 1, 1, 1.5, 2000, 0.5, 0.8, 1000 \end{bmatrix}\right)$$

$$R = I_4$$

## ACKNOWLEDGMENT

The authors would like to thank FinELib consortium, Finland for the APC funding.

## REFERENCES

[1] J. Bjerknes and A. Winfield, "On fault tolerance and scalability of swarm robotic systems," in *Distributed Autonomous Robotic Systems* (Springer Tracts in Advanced Robotics), vol. 83. Berlin, Germany: Springer, 2013, pp. 431–444.

[2] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles—A survey," *J. Ind. Inf. Integr.*, vol. 16, Dec. 2019, Art. no. 100106, doi: 10.1016/j.jii.2019.100106.

[3] X. Wang, V. Yadav, and S. N. Balakrishnan, "Cooperative UAV formation flying with obstacle/collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 4, pp. 672–679, Jul. 2007.

[4] C. Zhuge, Y. Cai, and Z. Tang, "A novel dynamic obstacle avoidance algorithm based on collision time histogram," *Chin. J. Electron.*, vol. 26, no. 3, pp. 522–529, 2017.

[5] Y. Singh, "Cooperative swarm optimisation of unmanned surface vehicles," Ph.D. dissertation, Univ. Plymouth, Plymouth, U.K., 2019, pp. 2–4. [Online]. Available: https://pearl.plymouth.ac.uk/handle/10026.1/13700 and https://pearl.plymouth.ac.uk/bitstream/handle/10026.1/13700/2019Singh10511788PhD_Full.pdf?sequence=2&isAllowed=y

[6] J.-C. Laprie, "Dependable computing and fault tolerance: Concepts and terminology," in *Proc. 25th Int. Symp. Fault-Tolerant Comput. Highlights From 25 Years*, 1995, pp. 2–11.

[7] J. O'Keeffe, D. Tarapore, A. G. Millard, and J. Timmis, "Adaptive online fault diagnosis in autonomous robot swarms," *Frontiers Robot. AI*, vol. 5, p. 131, Nov. 2018.

[8] L. Qin, X. He, and D. H. Zhou, "A survey of fault diagnosis for swarm systems," *Syst. Sci. Control Eng.*, vol. 2, no. 1, pp. 13–23, Dec. 2014.

[9] R. Xue and G. Cai, "Formation flight control of multi-UAV system with communication constraints," *J. Aerosp. Technol. Manage.*, vol. 8, no. 2, pp. 203–210, 2016.

[10] E. Khalastchi and M. Kalech, "Fault detection and diagnosis in multi-robot systems: A survey," *Sensors*, vol. 19, no. 18, p. 4019, Sep. 2019.

[11] E. Khalastchi and M. Kalech, "On fault detection and diagnosis in robotic systems," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–24, Jan. 2019.

[12] C. Sun, J. Tang, and X. Zhang, "FT-MSTC: An efficient fault tolerance algorithm for multi-robot coverage path planning," in *Proc. IEEE Int. Conf. Real-time Comput. Robot. (RCAR)*, Jul. 2021, pp. 107–112.

[13] H. Yang, Q.-L. Han, X. Ge, L. Ding, Y. Xu, B. Jiang, and D. Zhou, "Fault-tolerant cooperative control of multiagent systems: A survey of trends and methodologies," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 4–17, Jan. 2020.

[14] A. Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol, "Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 4, pp. 2832–2848, Oct. 2012.

[15] F. Liu, Y. Hua, X. Dong, Q. Li, and Z. Ren, "Adaptive fault-tolerant time-varying formation tracking for multi-agent systems under actuator failure and input saturation," *ISA Trans.*, vol. 104, pp. 145–153, Sep. 2020, doi: 10.1016/j.isatra.2019.01.024.

[16] W. Suo, M. Wang, D. Zhang, Z. Qu, and L. Yu, "Formation control technology of fixed-wing UAV swarm based on distributed ad hoc network," *Appl. Sci.*, vol. 12, no. 2, p. 535, Jan. 2022.

[17] H. Wang, M. Chen, and P. Fu, "A distributed fault-tolerant mechanism for mission-oriented unmanned aerial vehicle swarms," *Int. J. Commun. Syst.*, vol. 34, no. 8, May 2021, Art. no. e4789.

[18] X. Wang and G.-H. Yang, "Fault-tolerant consensus tracking control for linear multiagent systems under switching directed network," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 1921–1930, May 2020.

[19] X. Yu, Z. Liu, and Y. M. Zhang, "Fault-tolerant formation control of multiple UAVs in the presence of actuator faults," *Int. J. Robust Nonlinear Control*, vol. 26, no. 12, pp. 2668–2685, Aug. 2016.

[20] G. Raja, Y. Baskar, P. Dhanasekaran, R. Nawaz, and K. Yu, "An efficient formation control mechanism for multi-UAV navigation in remote surveillance," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.

[21] A. Tahir, J. Boling, M.-H. Haghbayan, and J. Plosila, "Development of a fault-tolerant control system for a swarm of drones," in *Proc. Int. Symp. (ELMAR)*, Sep. 2020, pp. 79–82, doi: 10.1109/ELMAR49956.2020.9219027.

[22] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[23] C. Yang, Y. Liu, X. Jiang, Z. Zhang, L. Wei, T. Lai, and R. Chen, "Non-rigid point set registration via adaptive weighted objective function," *IEEE Access*, vol. 6, pp. 75947–75960, 2018.

[24] P. Guo, W. Hu, H. Ren, and Y. Zhang, "PCAOT: A Manhattan point cloud registration method towards large rotation and small overlap," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 7912–7917.

[25] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.

[26] H. Chui and A. Rangarajan, "A new algorithm for non-rigid point matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jun. 2000, pp. 44–51.

[27] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, M. M. Yasin, and J. Plosila, "Energy-efficient formation morphing for collision avoidance in a swarm of drones," *IEEE Access*, vol. 8, pp. 170681–170695, 2020.

[28] P. Gabrlik, V. Kriz, and L. Zalud, "Reconnaissance micro UAV system," *Acta Polytechnica CTU Proc.*, vol. 2, no. 2, pp. 15–21, Dec. 2015.

[29] A. Basci, K. Can, K. Orman, and A. Derdiyok, "Trajectory tracking control of a four rotor unmanned aerial vehicle based on continuous sliding mode controller," *Elektronika ir Elektrotechnika*, vol. 23, no. 3, pp. 12–19, Jun. 2017.

[30] F. Šolc, "Modelling and control of a quadrocopter," *AiMT*, vol. 5, no. 2, pp. 29–38, Dec. 2010.

[31] A. Tahir, J. M. Boling, M.-H. Haghbayan, and J. Plosila, "Comparison of linear and nonlinear methods for distributed control of a hierarchical formation of UAVs," *IEEE Access*, vol. 8, pp. 95667–95680, 2020, doi: 10.1109/ACCESS.2020.2988773.

[32] A. Tahir, J. Böling, M.-H. Haghbayan, and J. Plosila, "Navigation system for landing a swarm of autonomous drones on a movable surface," *Commun. ECMS, Proc. 34th Int. ECMS Conf. Modeling Simulation*, vol. 34, no. 1, pp. 168–174, 2020, doi: 10.7148/2020-0168.

[33] Q. Ali and S. Montenegro, "Explicit model following distributed control scheme for formation flying of mini UAVs," *IEEE Access*, vol. 4, pp. 397–406, 2016.

[34] C. Masse, O. Gougeon, D.-T. Nguyen, and D. Saussie, "Modeling and control of a quadcopter flying in a wind field: A comparison between LQR and structured H$\infty$ control techniques," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2018, pp. 1408–1417.

[35] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *Proc. 3rd Int. Workshop Agent Theories, Archit., Lang.*, Berlin, Germany: Springer, 1996, pp. 21–35.

[36] V. Sathyanarayanan, "Evaluation of moving least squares as a technique for non-rigid medical image registration," M.S. thesis, Vanderbilt Univ., Nashville, TN, USA, 2008. [Online]. Available: https://ir.vanderbilt.edu/bitstream/handle/1803/15098/Thesis_viju11.pdf;jsessionid=42D3CEB876C252944BCBB6662492666A?sequence=1

**ANAM TAHIR** received the B.S. degree in computer engineering and the M.S. degree in electrical engineering (major: control systems) from COMSATS University, Islamabad, Pakistan, and the Master of Engineering degree in autonomous maritime operations from the NOVIA University of Applied Sciences, Turku, Finland. She is currently pursuing the Ph.D. degree with the Department of Computing, Faculty of Technology, University of Turku, Finland. Her research interests include autonomous vehicles, adaptive and nonlinear control, modelling and simulation of dynamical systems, and intelligent control.

**HASHEM HAGHBAYAN** (Member, IEEE) received the B.A. degree in computer engineering from the Ferdowsi University of Mashhad, the M.Sc. degree in computer architecture from the University of Tehran, Iran, and the Ph.D. degree (Hons.) from the University of Turku, Finland. From 2018 to 2021, he was a Postdoctoral Researcher with the Department of Computing, Faculty of Technology, University of Turku. He is an Adjunct Professor (Docent) in embedded intelligent systems with the Department of Computing, Faculty of Technology, University of Turku. His research interests include machine learning, autonomous systems, high-performance energy-efficient architectures, and on-chip/fog resource management.

**JARI M. BÖLING** received the M.Sc. degree in chemical engineering and the Ph.D. degree in control engineering from Åbo Akademi University, Turku, Finland, in 1994 and 2001, respectively. From 2003 to 2004, he was a Postdoctoral Researcher with the University of California at Santa Barbara, Santa Barbara, USA. Since 2005, he has been a Senior University Lecturer of control engineering with the Faculty of Science and Engineering, Åbo Akademi University. His research interests include systems identification, adaptive control, machine learning, and modelling and simulation of dynamical systems.

**JUHA PLOSILA** (Member, IEEE) received the Ph.D. degree in electronics and communication technology from the University of Turku, Finland, in 1999. He is currently a Full Professor in autonomous systems and robotics with the Department of Computing, Faculty of Technology, University of Turku. He is the Head of the EIT Digital Master Program in embedded systems with the EIT Digital Master School, European Institute of Innovation and Technology; and represents the University of Turku in the Node Strategy Committee of the EIT Digital Helsinki/Finland node. He has a strong research background in adaptive multiprocessing systems and platforms, and their design, including specification, development, and verification of self-aware multi-agent monitoring and control architectures for massively parallel systems; machine learning and evolutionary computing based approaches; as well as application of heterogeneous energy efficient architectures to new computational challenges in the cyber-physical systems and the Internet of Things domains, with a recent focus on fog/edge computing (edge intelligence) and autonomous multi-drone systems.