Received September 3, 2020, accepted September 13, 2020, date of publication September 18, 2020, date of current version September 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3024953

Energy-Efficient Formation Morphing for Collision Avoidance in a Swarm of Drones

JAWAD NAVEED YASIN[1], SHERIF ABDELMONEM SAYED MOHAMED[1], MOHAMMAD-HASHEM HAGHBAYAN[1], (Member, IEEE), JUKKA HEIKKONEN[1], HANNU TENHUNEN[2], MUHAMMAD MEHBOOB YASIN[3], AND JUHA PLOSILA[1], (Member, IEEE)

[1]Department of Future Technologies, University of Turku, 20500 Turku, Finland
[2]Department of Electronic Systems, KTH Royal Institute of Technology, 11428 Stockholm, Sweden
[3]Department of Computer Networks, College of Computer Sciences & Information Technology, King Faisal University, Hofuf 31982, Saudi Arabia

Corresponding author: Jawad Naveed Yasin (janaya@utu.fi)

This work was supported by the Academy of Finland-funded research project under Grant 314048.

ABSTRACT Two important aspects in dealing with autonomous navigation of a swarm of drones are collision avoidance mechanism and formation control strategy; a possible competition between these two modes of operation may have negative implications for success and efficiency of the mission. This issue is exacerbated in the case of distributed formation control in leader-follower based swarms of drones since nodes concurrently decide and act through individual observation of neighbouring nodes' states and actions. To dynamically handle this duality of control, a plan of action for multi-priority control is required. In this paper, we propose a method for formation-collision co-awareness by adapting the thin-plate splines algorithm to minimize deformation of the swarm's formation while avoiding obstacles. Furthermore, we use a non-rigid mapping function to reduce the lag caused by such maneuvers. Simulation results show that the proposed methodology maintains the desired formation very closely in the presence of obstacles, while the response time and overall energy efficiency of the swarm is significantly improved in comparison with the existing methods where collision avoidance and formation control are only loosely coupled. Another important result of using non-rigid mapping is that the slowing down effect of obstacles on the overall speed of the swarm is significantly reduced, making our approach especially suitable for time critical missions.

INDEX TERMS Autonomous aerial vehicles, collision avoidance, multi-robot systems, formation maintenance, swarm intelligence, leader follower.

I. INTRODUCTION

Resource utilization and decision making optimization in autonomous navigation for a swarm of robots is gaining traction in the research community [1]. The motive for this is the absence of a multi-objective strategy in the traditional operation of robots, e.g., drones/UAVs, for optimally or near-optimally achieving various system goals under various mission or design constraints such as the flight time and energy payload [2], [3].

Swarms of drones have demand in vast and diverse application areas for instance in the military, commercial use, search and rescue, monitoring traffic, threat detection especially at borders, and atmospheric research purposes [4], [5]. Due to the ability to work in a collaborative manner in a 3-dimensional space, research on optimal navigation of swarms is gaining even more attention in the research community [6]. The deployment of an efficient navigation system for swarms or multiple UAVs adds significant research challenges over single UAVs. Two important raised challenges while focusing on navigation in a swarm of drones are: 1) the formation and its maintenance and 2) collision avoidance [7], [8]. Collision avoidance primarily focuses on path planning of individual drones to steer clear of possible collisions between the drones themselves within the swarm and between drones and external obstacles in the environment [8]. The responsibility of formation algorithms, in turn, is to define the location of each drone with respect to the other drones [9].

The interdependence of formation control and collision avoidance is of significant importance as collision avoidance needs to be considered in order to maintain the formation, and similarly, in order to avoid collisions, the intended formation needs to be considered. In the decision making for both

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

processes, the stabilization time and energy consumption minimization should be considered. Mainly a swarm *deviates* from the formation for collision avoidance, i.e., to avoid an obstacle, and after passing the obstacle the swarm *turns back* again to the formation. This sequential process of deviation and turning back must be safe, fast, and energy-efficient.

Several unanticipated parameters or factors may affect the optimal implementation of collision avoidance along with a dynamic formation control algorithm. For instance, a change in the formation may be forced by prioritizing collision avoidance over formation maintenance due to unaccounted objects/obstacles or narrow gaps/openings between multiple obstacles. In order to do the whole process autonomously, we need to analyse how collision avoidance and formation control can be systematically integrated together. In this paper, the proposed algorithm considers these factors by taking into account the strategy of maintaining the swarm formation dynamically with variable speeds of UAVs along with an efficient collision avoidance methodology. To make it safe, each drone should obey a maximum possible distance from the obstacle and other drones. To make it fast and energy-efficient, the shortest spatial deviation and turning back should be considered. In our proposed method, the deviation phase follows reflexive decision making due to the uncertainty of the obstacle. The main objective is to reduce the spatial deviation while keeping a *safe distance*. For the turn back phase, we propose an energy function for the swarm formation inspired by the energy function of a thin-plate spline, where the result of minimizing this energy function, determines the navigation decision for each drone to resume the formation. Our approach focuses on integrating all these features together along with reducing the total energy of the system. Once these factors have been taken into account, and the pattern has been developed for switching between the formation maintenance and collision avoidance modes autonomously, the thin-plate splines technique is integrated into the algorithm in order to optimize it further by reducing the overall energy of the system. This technique helps by optimally reducing the disturbances caused by obstacle(s) by bringing the node(s)[1] or UAVs back to their stable coordinates in a timely yet aggressive manner.

The contributions of this paper compared with our previous paper and the state-of-the-art are listed as follows:

1) Proposing a new idea to reduce the time and energy through random dispersion of drones in the first phase of detecting an obstacle.
2) Proposing a new idea to reduce the time and energy by applying the thin-plate splines algorithm in the second phase of detecting an obstacle.
3) Providing comprehensive simulation results for a swarm of drones for the proposed idea and implementing recent existing methods to show the efficiency of our technique in comparison with those ideas.

[1]Terminologies UAVs, drones, and nodes are used interchangeably in this paper.

The rest of the paper is organized as follows. Section 2 covers the related work. In Section 3, basic concepts of formation, swarm, and collisions are briefly described. The proposed algorithm and its development is given in Section 4. Optimal swarm reconfiguration is explained in Section 5. Section 6 focuses on simulation results and related discussion. Finally, concluding remarks are presented in Section 7.

II. RELATED WORK

Formation control algorithms can be categorized into three general approaches [10], [11], namely: 1) the virtual structure based approach, in which all the drones in the swarm formation are navigated as if there was a single big drone and hence the same trajectory is taken [12], [13]; 2) the leader-follower based approach, where every drone functions individually and autonomously by calibrating or altering its position according to the leader and maintaining its position in the formation as close as possible to the desired coordinates [14], [15]; and 3) the behaviour based approach, in which based on a pre-defined strategy the drone selects one of the multiple behaviours [16], [17]. The leader-follower based approach is more common out of the aforementioned approaches, due to its ease of analysis and implementation [18], [19]. In this approach, leaders are explicit, and it is assumed that all or some of the followers have access, when required, to relevant motion information such as velocities and positions of the leaders within their sensing range [20], [21].

With the integration of commercial, leisure based and military UAVs and/or aircraft, a good collision avoidance algorithm or system becomes exponentially important for their safe operation in the civilian airspace [22]. During the flight, they can encounter both stationary and moving obstacles and objects that need to be safely and reliably evaded using the collision avoidance system [23], [24]. Typically, algorithms for collision avoidance can be divided into three generic classes [25], [26]: 1) force-field methods that work on the principle of applying attractive/repulsive electric forces existing amongst charged objects; each drone in a swarm is considered a charged particle, and attractive or repulsive forces between drones and the obstacles are used to generate and choose the routes to be taken [27], [28]; 2) sense-and-avoid based methods, where the process of collision avoidance is simplified into individual detection and avoidance of the objects and obstacles, resulting in short response times and reducing the computational power needed [29], [30]; and 3) optimization based methods which focus on providing the optimal or near-optimal solutions for path planning and motion characteristics of each drone with respect to the other drones and obstacles. In order to calculate efficient routes within a finite time horizon, these methods rely on static objects with known locations and dimensions [31], [32].

In formation flight, UAVs/nodes perform varying maneuvers like accelerating, decelerating, synchronized movements, and turning in different directions, that require each member of the formation to have a specific minimum distance from other members. To successfully perform those maneuvers and missions, nodes must have the ability to avoid collisions with other nodes in the swarm and with external obstacles [27], [33].

Basically, the collision avoidance process while keeping the formation consists of two main phases, i.e., 1) reforming the swarm to avoid a collision while approaching an obstacle and 2) resuming the formation after passing the obstacle. Most of the existing works completely lack such tight integration of dynamic formation maintenance and collision avoidance strategies. They either focus on keeping the formation or avoiding collisions based on state-of-the-art collision avoidance algorithms. In our proposed methodology, we integrate dynamic formation maintenance along with collision avoidance capabilities for swarms to address an important research topic not properly covered by the state-of-the-art. For formation control of the swarm, we utilize the leader-follower based approach due to its ease of implementation, analysis, and scalability [20], [23]. The objective of our approach is to reduce the collision avoidance time and energy consumption during the reformation and resuming processes. During this formation morphing, the leadership of the swarm might be totally changed (as shown in Figure 1), and an ex-follower drone may take the role of the leader, further highlighting the novelty of the proposed method with respect to the state-of-the-art. It is important to note here that the overlapping of the UAV1 and UAV2 (shown as blue and green paths) is at different times, that is due to the fact as soon as UAV 2 detects UAV1 coming in front of it to bypass the obstacle, UAV2 slows down to maintain safe distance with UAV1 while maintaining its own trajectory for obstacle avoidance. As shown in the figure, UAV3 had minimum deviation and did not have to slow down as much, it may bypass the obstacle before UAV1 (the original leader), and hence UAV3 may take the role of the new leader as shown in Figure 1.

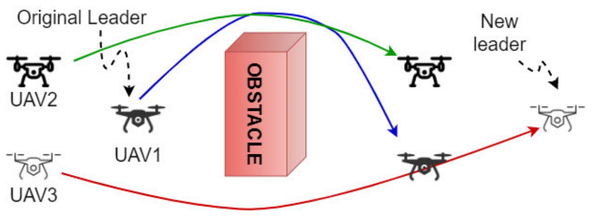


FIGURE 1. **Formation and collision avoidance.**

III. PRELIMINARIES

This section describes the basic concepts essential for the work presented in this paper.

Swarm robotics can be defined as the study of how a system consisting of multiple collaborating robots can be designed by analyzing the local interactions between the robots themselves and the robots and their environment [34]. It is strongly inspired by real-life phenomena such as swarms of insects and flocks of birds. It is also referred to as distributed robotics [35], robot colonies [36], and collective robotics [37].

A *formation* in swarm robotics refers to a desired arrangement of the robots in a swarm, a particular arrangement or shape of positions the multiple robots aim to maintain with respect to each other. The swarm can be ordered to maintain a certain shape of formation to perform a given mission [38]. In a *queue* formation, drones or robots form a simple line or sequence, following each other and maintaining the distance between each other within a given range. Its key benefit is that it enables a swarm to pass through obstacles without breaking the formation. Any arbitrarily shaped formation may have to reorganize itself into a queue formation in order to avoid and navigate through multiple obstacles while maintaining connectivity and tracking between the nodes.

A drone is defined to have a *collision* with an object, i.e., another drone within the swarm or an external obstacle, if the distance of the drone to the object is less than a predetermined *collision radius*, R_c. Expressing this mathematically, a collision is considered taking place when the following condition is true:

$$||r_u - r_o|| < R_c \tag{1}$$

where r_u and r_o are the position vectors of the UAV and the object, respectively.

Similarly, an obstacle is detected by a drone, when the following condition holds:

$$||r_u - r_o|| < d_{Range} \tag{2}$$

where d_{Range} is the detection range radius of the UAV, which varies and is dependent on the characteristics of the on-board sensor system.

For ease of simplicity, the following assumptions and initial conditions are used in this study:

1) All obstacles are stationary and/or can be introduced in front of the UAVs at any given time.
2) UAVs have variable speeds and accelerate or decelerate as needed. Using an on-board sensor system, every UAV obtains its own position and velocity vectors. The on-board sensor system could include lidar, sonar, radar, and GPS to mention a few.
3) There is no information loss in the communication channels between the UAVs.

IV. THE PROPOSED APPROACH

In this section, we describe the proposed Energy-efficient Formation morphing for Collision Avoidance (EFMCA) algorithm for a swarm of drones. The overall strategy is to combine swarm formation control and collision avoidance mechanism to facilitate the process of autonomous swarm navigation, Figure 1. To accomplish this, a novel top-level algorithm is developed, composed of two partial feedback-based algorithms: one for formation control and one for collision avoidance. The feedback for each drone's controller comprises both collision radius and formation distance, and the goal is to minimize their errors, i.e., differences between the observed values and the reference values. The angular error is the difference of the required angle from the observed angle, indicating how much the node should turn to maintain its position w.r.t. its neighbour. Correspondingly, the distance error is the difference of the measured distance from the reference distance, indicating how much the node should get closer or farther to or from its neighbour.

If there is no feedback for an object detected by the on-board sensor system, indicating there is no external object in the vicinity, the algorithm maintains the formation by dynamically checking and adjusting the distance of the drone to its neighbours. The goal is to keep the distance greater than the collision radius and close to the pre-specified formation distance.

Upon detection of an obstacle, the algorithm raises the priority of the collision avoidance part. The collision avoidance part of the algorithm gets the highest priority once the UAV approaches the minimum safe distance from the obstacle. After bypassing the obstacle(s), a Failsafe/Fault-Tolerance check is executed to see if the UAV has lost its connection or if it still has a connection with its respective leader.

A. FORMATION-COLLISION CO-AWARENESS

Algorithm 1 gives the general pseudo-code of the top-level formation-collision co-awareness algorithm. As our initial setup, we presume that the UAVs are spawned at random coordinates, and also the IDs to the nodes/UAVs are assigned before the mission is started. Every node executes this top-level algorithm locally by deploying the algorithm on its on-board processing unit. The path planning of the leaders and the navigation of the swarm is out of the scope of this work. Formation maintenance is initiated based on the feedback on the distance the node has to its neighbours (Line 2 in Algorithm 1). Then $E_{formation}$, i.e., the formation error, is calculated. To achieve this, a threshold value is added to the absolute value of the relative angular and distance position of a drone with respect to its neighbours (Line 4). The degree of freedom that each drone can have in the formation of the swarm is determined by the threshold value. Depending on whether an object is detected or not, the formation error is processed accordingly. In case of an observed obstacle, i.e., the obstacle comes within the detection range, the collision error is calculated, specifying the difference between the obstacle distance and the collision radius (Line 5).

There are three cases on which the algorithm works (Lines 6-13). If there is a positive formation error, i.e., $0 < E_{formation}$, but no probable obstacle in the vicinity, then, in order to decrease the error, the algorithm utilizes the *Formation* function to reconfigure the relative angular and distance positions of the drones (Lines 8-9). The *Collision-aware formation* function is executed in the case where an obstacle is detected by the sensors, but there is no immediate danger of a collision. Then the formation of the swarm is reconfigured by considering the location of the obstacle under observation (Lines 10-11). However, the *Collision avoidance* function is launched, if the condition $0 < E_{collision}$ is true

Algorithm 1 Formation-Collision Co-Awareness

```
    procedure FORMATION-COLLISION CO-AWARENESS
2:      formation ← Initiate Formation;
        while True do
4:          $E_{formation}$ ← Calculate distance and angular formation error (formation);
            $D_{obstacle}, E_{collision}$ ← Calculate obstacle distance and collision error if any;
6:          STATE ← (0 < $E_{formation}$, Obstacle is probe-able by local sensors, 0 < $E_{collision}$);
            switch STATE do
8:              case (True, False, False)
                    Formation ($E_{formation}$);
10:             case (-, True, False)
                    Collision-aware formation ($E_{formation}$, $D_{obstacle}$);
12:             case (-, -, True)
                    Collision avoidance ($E_{collision}$);
14:                 TPS_Flag = True;
            end switch
16:         if TPS_Flag == True; then
                Turn-Back(TPS_Flag);
18:         end if
            if MyLeader is out of range then
20:             MyLeader ← Self;
            else
22:             MyLeader ← Leader(Self);
            end if
24:     end while
    end procedure
```

indicating the obstacle is too close to the drone (Lines 12-13). This operation is oblivious of the formation, meaning that formation protocols are completely ignored; only the distance of the drone w.r.t. the obstacle matters. In this mode, only *Collision avoidance* of the above three functions can be executed; the two formation-adjusting functions are disabled, independently of the value of $E_{formation}$.

Once a collision has been avoided successfully, the control is transferred to the *Turn-Back* function to minimize the disturbance caused by the evasive maneuvering, i.e., to bring the nodes back into the formation as efficiently as possible (Lines 16-17). Since the disturbance caused by collision avoidance might totally reform the swarm, this process might be accompanied by selecting a new leader for the swarm. As a fail-safe check, i.e., a special scenario in case the leader is lost or undetectable by the follower, the follower drone temporarily sets itself as its own leader, broadcasts this to its followers, and starts navigating towards the destination (Lines 19-20). However, as soon as the leader is detected, i.e., gets back in the visible range, the drone immediately comes back into the formation and the starts to follow the leader (Lines 21-22). All the functions in Algorithm 1 are explained in detail in the following subsections.

Algorithm 2 Formation

```
   procedure FORMATION ($(E_{distance}, E_{angular}) \leftarrow E_{formation}$)
2:     $(distanceError, angularError) \leftarrow R_{formation} - E_{formation}$
       if $0 < ABS(distanceError)$ then
4:         Calculate Distance From Respective Leader;
           Accelerate/decelerate Accordingly;
6:     end if
       if $0 < ABS(angularError)$ then
8:         Set Angular Direction ($R_{angular}$);
       end if
10: end procedure
```

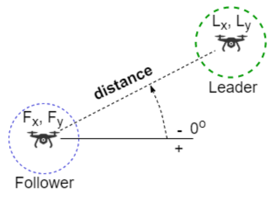


FIGURE 2. Distance and direction calculation.

B. FORMATION ALGORITHM

A unique ID that is given to each node in the swarm, is used by the other nodes and the leader for recognition purposes. Then the leader starts navigating towards the destination; in the meantime, the followers try and maintain the formation by keeping the required distances from the respective neighbours. The general pseudo-code of the formation function is given in Algorithm 2. The input of the algorithm comprises the pre-specified formation information that is considered the reference, i.e., $R_{formation}$ (consisting of the reference distance $R_{distance}$ and angle $R_{angular}$), and the instantaneous captured formation information, i.e., $E_{formation}$, that is the estimated distance and angle each drone has w.r.t. its neighbours (e.g. the leader and follower in a queue formation) at a given time. First, the deviation of each drone w.r.t. its pre-defined position is determined by calculating the angular and distance errors (Line 2 in Algorithm 2). The algorithm then starts to change the state of the swarm by manipulating the node's mechanical actuators, in the case the angular or distance error is greater than the defined threshold indicating that the variation is significant and affects the formation negatively. The *distanceError* is calculated as follows:

$$distanceError = \sqrt{(L_x - F_x)^2 + (L_y - F_y)^2} \quad (3)$$

where F_x, F_y and L_x, L_y are, respectively, the follower's and leader's x and y-coordinates. If the distance error is positive, indicating that the distance between the node and its neighbour is too high, the drone accelerates to catch up and attain the desired formation distance. Similarly, in the case of a negative distance error, meaning that the distance of the node to its neighbour is too low, the drone starts to decelerate in order to decrease the absolute value of the distance error.

Algorithm 3 Collision-Aware Formation Algorithm

```
   procedure COLLISION-AWARE FORMATION($E_{formation}, D_{obstacle}$)
2:     $newTemporaryFormation \leftarrow$ determineNewFormation ($E_{formation}, D_{obstacle}$);
       Formation ($newTemporaryFormation$);
4: end procedure
```

To maintain the desired distance and orientation, each drone employs a local speed control for the possible manoeuvres.

Each follower determines the angle of the leader as shown in Figure 2 and calculates as follows:

$$Angle(y, x) = \begin{cases} arctan(\frac{y}{x}) & x > 0, \\ arctan(\frac{y}{x}) + \pi & x < 0 \; and \; y \geq 0, \\ arctan(\frac{y}{x}) - \pi & x < 0 \; and \; y < 0, \\ +\frac{\pi}{2} & x = 0 \; and \; y > 0, \\ -\frac{\pi}{2} & x = 0 \; and \; y < 0 \end{cases} \quad (4)$$

The result is always between -π and π. Using the plane vector from the origin to the target, the angle is formed w.r.t. the positive X-axis. Since the signs of both inputs are known and taken into consideration, the correct quadrant of the computed angle, i.e., *angularError*, and the corresponding direction values are calculated by using the result from the above equation as follows:

$$angularError = (dx = cos(Angle), dy = sin(Angle)) \quad (5)$$

C. COLLISION-AWARE FORMATION ALGORITHM

If it is not possible to maintain the formation due to the detection of an obstacle by one of the nodes, Algorithm 1 calls the proposed collision-aware formation function. This function is specified in Algorithm 3. The main plan of action in situations, where $D_{obstacle}$ (obstacle's distance) is within the range of a node but not small enough to justify the actual collision avoidance measures, is to reshape the formation while symmetrically shifting the swarm at the same time in order to divert the observer node, and thereby its followers, from the potential collision course. Based on this, the algorithm first determines a new formation which is close to the original one, making it easier to bypass the obstacle (Line 2 in Algorithm 3). After this decision, the information regarding the new formation is fed to the formation function in order to reshape the swarm (Line 3). Then the obstacle distance and the collision error are recalculated in the next iteration of Algorithm 1 to check if the obstacle is still within the detection range of the observer node or not. If the obstacle is outside the detection range of the node, the formation returns back to the original one (Lines 8-9 in Algorithm 1).

D. COLLISION AVOIDANCE ALGORITHM

The pseudo-code of the collision avoidance function is shown in Algorithm 4. Collision avoidance function is invoked if an obstacle is encountered critically close to the node. First, it starts to detect the edges of the obstacle, at which point the angles between the node's and obstacle's positions as well as the exact position of the obstacle can be determined (Line 3 in Algorithm 4). If the edges of the obstacle are in the visible range of the on-board sensor(s), it is checked if there is only one obstacle or multiple obstacles in the vicinity (Lines 4-5). In the case of multiple obstacles, the gap between the obstacles is calculated as shown in Figure 4(a) (Line 6). Then the algorithm checks and decides, based on the calculations, if it is safe and possible for the drone to go through the detected gap (Lines 7-8). If the gap is not wide enough for the drone to go through, the algorithm treats the obstacles as a single entity and guides the drone around the obstacles to avoid collisions (Lines 9-11). Moreover, if only one obstacle is detected (Line 13), the short-term path planning is done accordingly by calculating in which direction the node should navigate to bypass the obstacle with minimal deviation from its original path (Lines 13-14). However, in case neither edge of the obstacle is visible/detected by the on-board sensor system of the drone (Line 17), a tangent line is drawn from the drone's current position to the destination coordinates, and the path is chosen accordingly by navigating towards the destination while avoiding colliding with the obstacle and staying as close to the tangent line as possible (Lines 18-19), see Figure 4(b). The value of $E_{collision}$ is updated to check for successful collision avoidance (Line 21).

Algorithm 4 Collision Avoidance Algorithm

```
    procedure COLLISION AVOIDANCE($E_{collision}$)
2:      while $0 < E_{collision}$ do
            $edges$ ← detect edges ();
4:          if $edges$ contains visible edges then
                if More than one obstacle is detected then
6:                  $D_{safe}$ ← Calculate gap between obstacles ($edges$);
                    if $D_{safe} > R_c$ then
8:                      Short-term path planning ($edges$);    ▷ Align UAV to pass through the obstacles
                    else
10:                     $pathPlan$ ← Calculate path plan ($edges$);
                        Short-term path planning ($pathPlan$);
12:                 end if
                else
14:                 $pathPlan$ ← Calculate path plan ($edges$);
                    Short-term path planning ($pathPlan$);
16:             end if
            else
18:             $tangentLine$ ← Calculate tangent line();
                Short-term path planning ($tangentLine$);
20:         end if
            $E_{collision}$ ← Update collision error;
22:     end while
    end procedure
```

To put in a nutshell, the proposed collision avoidance function, i.e., Algorithm 4, operates based on three different cases, namely: 1) there is a single obstacle in the vicinity and its edges are detected, Figure 3; 2) there are multiple obstacles and their edges and gaps are detected, Figure 4(a); and 3) the edges of the obstacle(s) are not visible, indicating the obstacle is very large; in this scenario, the path is determined based on a tangent line, Figure 4(b). If $D_{obstacle}$, i.e., the distance from the drone to the obstacle, is within the middle ground of the detection range and safe distance f_s (Table 1, Figure 3), the algorithm starts decelerating the drone while calculating the angle at which the detected obstacle lies in order to deviate from that path. The collision avoidance algorithm takes complete control of the system as soon as $D_{obstacle}$ gets closer to f_s and guides the drone to go around the obstacle safely.

The situation in which the edges of a single obstacle are detected by the nodes is illustrated in Figure 3. Table 1 lists the used variables and their explanations.

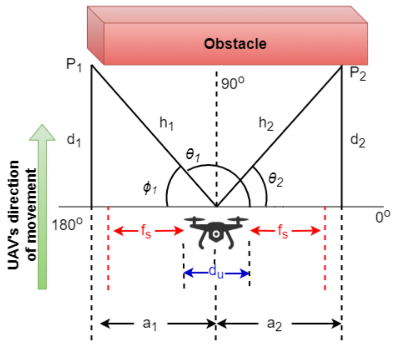
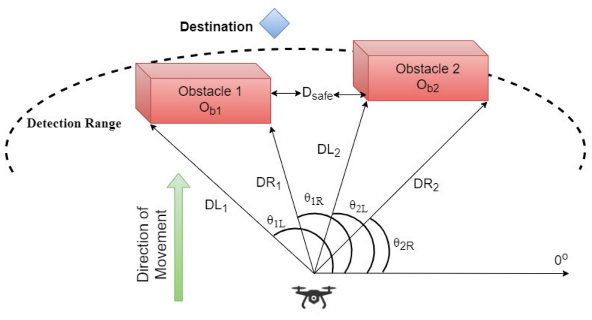


FIGURE 3. Obstacle detection and avoidance using geometric guidance law.

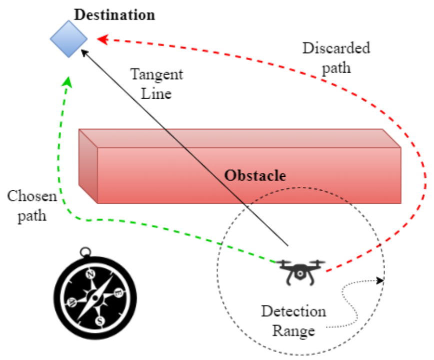
TABLE 1. Description of variables from Fig.

Variables	Description
d_u	UAV's width
f_s	minimum safe distance allowed from from either side of UAV
h_1 h_2	detected distance of the object's edges on the left and right sides
a_1 a_2	calculated from the centre of the UAV, linearized distance of the object on the left and right side of the UAV respectively
d_1 d_2	linearized distance of the object straight ahead of the UAV
P_1 P_2	point where the edge(s) (if any) were detected by the on-board sensor
θ_1 θ_2	angles of the edges
ϕ_1	linearized angle

All possible combinations of the object's location are analysed once the exact coordinates of the obstacle have been calculated, and the decision is made accordingly. It is then decided by the algorithm if it is safe to continue along the current path or if the drone must be diverted to a certain extent to avoid colliding with the object. For instance, if $a_2 < (d_u/2) + f_s$ (in Figure 3), the algorithm reroutes the drone to left



(a) Multiple Obstacles with opening between them



(b) Local decision making in case of unknown length of obstacle

FIGURE 4. Multiple objects and local decision making figures.

from its current route as continuing along the same path may result in a collision with the obstacle on the right side of the drone. However, in case $a_2 \geq (d_u/2) + f_s$, no rerouting or deviation from the original path is needed due to the fact that the obstacle is on the right side of the drone and is not in the critical collision radius.

Furthermore, upon detecting multiple obstacles and their corresponding edges, two different actions can be performed, i.e.: 1) extending the collision envelope when $D_{safe} \leq R_c$, and 2) activating the detection and gap calculation mode when $D_{safe} > R_c$. Here the distance between the inner edges of the obstacles is indicated by D_{safe} and the collision radius is denoted by R_c. Based on action 1, both obstacles lying in close proximity are considered a single entity, by extending the collision envelope, while calculating the avoidance maneuver parameters (Lines 9-10 in Algorithm 4). In this case, as shown in Figure 4(a), θ_{1L} and θ_{2R} are taken into account due to the fact that both obstacles are treated as a single obstacle. Then the calculations are performed to bypass the combined obstacle by flying from either side (left or right) of the obstacle.

In the latter case, i.e., the action 2, the algorithm analyses the detected gap between the obstacles to determine if the width D_{safe} satisfies the condition $D_{safe} \geq 2f_s + d_u$, where d_u is the width of the drone and f_s is the safe distance (Table 1, Figure 3), indicating that the opening/gap between the obstacles is wide enough for the drone to go through them. However, if the condition is not satisfied, it indicates that it is physically not possible for the drone to go through the gap between the obstacles, i.e., the gap between the obstacles is too narrow and smaller than the width of the drone and hence does not provide sufficiently large safety margin. Subsequently in this case, the algorithm switches back to the action 1, i.e., instead of considering the angles θ_{1R} and θ_{2L} like in the action 2, the algorithm opts for the angles θ_{1L} and θ_{2R} (Figure 4(a)), and the obstacles are regarded as a single obstacle which is then bypassed either from its left or right side.

In the scenario where the obstacle extends beyond the visible range of the on-board sensor system and its dimensions cannot be computed, the algorithm uses the data provided by the local GPS unit to draw a tangent line to the destination, that is the line defining the shortest path between the drone and the destination, see Figure 4(b). The node, based on the angle of the tangent, decides which direction it should opt for. For example in Figure 4(b), in order to select from the only possible choices that are west/left or east/right, using the angle of the tangent line, it is determined that the destination is towards the north-west of the drone. Consequently, to avoid the obstacle and to stay as close to the tangent line as possible while keeping the minimum safe distance from the obstacle for safe maneuvering, the green path is chosen.

V. OPTIMAL SWARM RECONFIGURATION

After observing an obstacle in its flight path, a UAV needs to maneuver around it according to rules set by the collision avoidance algorithm. Such maneuvers generally distort the shape of the swarm's formation from the originally planned shape that may, sometimes, be crucial to the success of its mission. It is the intent of our submission to continually guard the collision avoidance maneuvers such that the disturbance from the planned, i.e. optimal, formation is kept at the minimum during the course of the maneuver(s) and that, after navigating past the obstacle(s), the swarm is returned back to its initial formation. This process raises a formation construction problem that is widely covered in the literature [4], [39]. However, in our case, the formation algorithm, or in other words the *disturbance rejection* of a swarm, must be compatible with our obstacle avoidance algorithm whose main target is to reduce the overall settling time and energy of the system. It is worth mentioning that we deploy a non-rigid mapping function for efficiency reasons. That is to say that the process of returning the swarm formation to its original shape is not required to re-establish initial neighbouring states among the drones since all the drones are considered to be identical. For example, in the original state drone *2* has two neighbours drones *1* and *3*, after reconstructing the formation its new neighbours may be drones *4* and *5*, it may even become the new global leader. In the following text, we refer to the original i.e. the desired formation shape as the *model* formation, while the shape at any instant during the flight is referred to as the *scene*. In the process of returning from the *scene* to the *model*, there are two main questions to be addressed. Firstly, what is the optimal alignment of *nodes* in the *scene* to node positions in the model? We name this as the *mapping* problem. Secondly, what is the optimal trajectory of each node in the scene so that it is *mapped* into the desired node position in the model? For the first issue, we apply the well-know concept of point set registration [40]–[42], which is based on thin-plate splines formulation (TPS) that is commonly used to solve data interpolation and smoothing problems [43]. After determining the mapping strategy, for the second problem, the proposed collision avoidance algorithm utilizes the shortest path scheme for deciding trajectories of individual nodes. Though a more efficient solution for the second part may be possible, our current focus is on designing an optimal mapping strategy, thus, it suffices to indicate, here, that search for an efficient trajectory of each node is one avenue for future work. In the following, we first explain the concept of thin-plate splines (TPS), and then we propose an algorithm based on the same.

A. THIN-PLATE SPLINES (TPS)

A spline is a function defined by polynomials in a piecewise manner. Spline curves are popular and are used for approximation of complicated shapes via curve fitting due to their ease of use and non-complicated construction [43]. We analyse the algorithm in 2D to make it simpler; consequently, two sets of correspondence points, i.e., data sets, are assumed X i.e. $x_i, i = 1, 2, 3, \ldots, n$ and V i.e. v_i, i $= 1, 2, 3, \ldots, n$. Here, the locations of a point in the scene and model are given by x_i and v_i respectively, where $x_i = (1, x_{ix}, x_{iy})$ and $v_i = (1, v_{ix}, v_{iy})$. A mapping function, i.e., $f(v_i)$, can be acquired while keeping the shape of the disturbed formation/function under consideration, by minimizing the energy function, E_{TPS}, given by the following equation:

$$E_{TPS}(f) = \sum_{i=1}^{n} ||x_i - f(v_i)||^2 + \lambda \iint [(\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})] dxdy \quad (6)$$

The amount of formation disturbance is evaluated by the energy function E_{TPS}. The scaling factor is denoted by λ. If we do not intend to keep the shape of the disturbed swarm under consideration and only intend to map one point set over the other, we set λ to zero then only the closest points are mapped without the shape being considered. When the λ is set to zero, E_{TPS}, i.e., disturbance, is given by:

$$E_{TPS}(f) = \sum_{i=1}^{n} ||x_i - f(v_i)||^2 \quad (7)$$

The mapping of points to the corresponding point sets while considering the intended *formation* is represented by the integral part of the equation. The mapping process from the highest point in disturbance formation to the original formation shape, i.e., the scene and the model, is determined by minimizing of the temperature function. Once the desired mapping is calculated, each drone in the scene starts following the shortest path to reach its hypothetical position in the model.

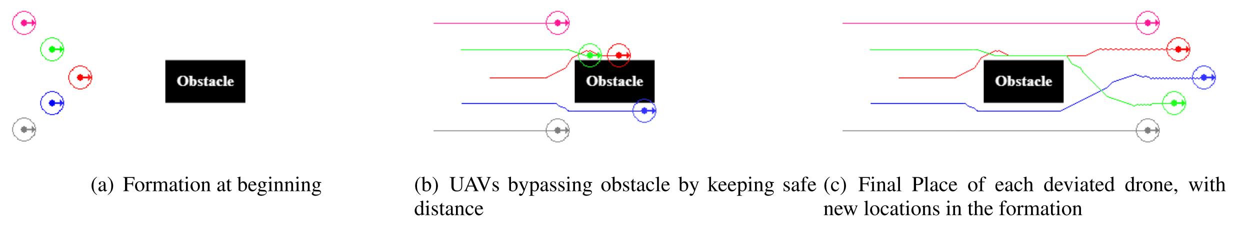
Algorithm 5 Turn-Back

```
    procedure TURN-BACK (TPS_Flag)
2:      while TPS_Flag == True do
            $F_{Location}$ = Determine the future location of the swarm;
4:          Determine the new coordinates for each drone;
            Temperature function minimization: TPS($F_{Location}$);
6:          if All nodes REACH new coordinates then
                TPS_Flag ← False;
8:          end if
        end while
10: end procedure
```

B. TURN-BACK FUNCTION

Algorithm 5 gives an overview of our TPS-based turn-back function. First, the future location of each node in the swarm is determined based on the current location of the nodes (Line 3). Based on each node's location, the new coordinates are determined for each drone (Line 4). Next, these values are fed to the TPS-based temperature minimization function, to bring the drones to their new coordinates as optimally as possible (Line 5). Once all drones have reached their respective new coordinates (Line 6), TPS_Flag is set to False and the control is returned to the main function (Line 7).

Figure 5 shows the example scenario where the swarm comes across an obstacle and starts reshaping formation while avoiding collision with the obstacle. Figure 5(a) shows the initial phase of formation and the locations of all the drones at the starting point. Figure 5(b), shows the distorted formation and the locations of the drones at halfway stage. In order to avoid the obstacle, UAV1 (red) had to slow down and deviate from its original path, thus, it comes in front of UAV2 (green). As soon as UAV2 detects UAV1 in its way, it slows down to maintain safe distance from objects ahead of itself. Whereas UAV3 (blue) needs only a small deviation from its original path, thus, it gets ahead of the rest of the formation and becomes a candidate for going to the position of UAV1 instead of slowing down in order to retain its earlier position in the model. UAVs 4 and 5, i.e., pink and grey respectively, maintain their own trajectories as they never came within colliding range of the obstacle. Figure 5(c), shows the final reformation after collision avoidance. After successfully avoiding the collision, in the reformation phase, UAV3 moves to the original position of UAV1 in the model, while UAV1 moves to the previous position of UAV2 and UAV2 moves to the position which was originally occupied by UAV3; all the while utilizing the shortest path to reach their respective final locations.



(a) Formation at beginning

(b) UAVs bypassing obstacle by keeping safe distance

(c) Final Place of each deviated drone, with new locations in the formation

FIGURE 5. Flexible formation through obstacle scenario.

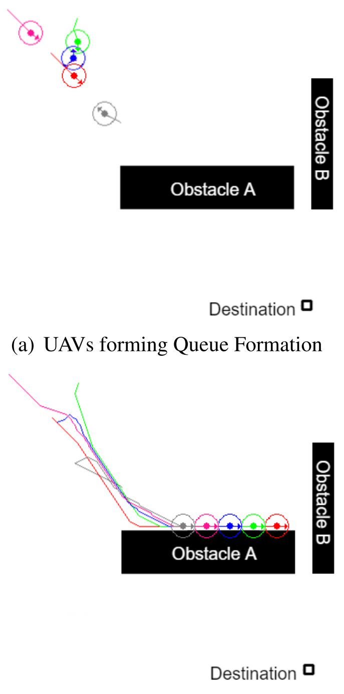
VI. SIMULATION & RESULTS

Simulation setup is as follows:

- area defined for simulations was setup as 700 × 500m 2D-XY plane
- all the UAVs are at the same altitude
- UAVs fly in horizon (self-leveling) mode instead of space mode, since horizon mode offers stabilized flight as the drone will self-level utilizing gyro and accelerometer
- five UAVs are launched from random locations in close proximity
- unique IDs are assigned to each UAV, incrementally from 1 to 5
- a simple queue formation is chosen whereby each UAV follows its immediate leader, while UAV with ID = 1, is chosen as the global leader.

The point mass particle model is used for simulating and visualizing the UAVs. By keeping the vertical axis constant, the UAVs navigate only in the XY-plane. Therefore, the equations of motion applicable for a point-mass particle moving in a 2D space are utilized here, and thus *6dof* movements are not considered in this work. Based on this, the dynamics of the drone utilized in this work is only the mass of the drone that has some initialized velocity vector and change in the velocity, i.e., acceleration. Furthermore, it is important to note here that algorithm is not restricted for rectangular objects as it mainly depends on the perception precision of the drone. Similarly, it can be perceived as during the perception phase, each obstacle is encapsulated into a bounding box and the algorithm takes into consideration those bounding box. Increasing the precision of the perception, refines the bounding box shape closer to the actual shape of the detected obstacle.

Upon spawning, UAV1, the global leader, starts navigating towards the destination, whereas the rest of the UAVs start to maintain the desired formation by accelerating/decelerating to reach their desired positions and maintain the distance from their respective leaders. To summarize, each UAVi+1 starts to track and follow UAVi, where $i = 1, .., 4$. Figure 6 shows the scenario where five UAVs are spawned at random coordinates in close proximity and come into formation by finding their respective leaders. The positions and movement/trajectories of the drones upon deployment and during early stages of queue formation are depicted in Figure 6(a). The situation emerging soon after the first obstacle, namely Obstacle A, comes within the visibility range of the leader is shown in Figure 6(b). Here, the leader navigates along the obstacle by keeping safe distance from it following *Algorithm 4*, the remaining UAVs follow the leader while maintaining the formation and keeping the safe distance from the obstacle as well, as per *Algorithm 3*.

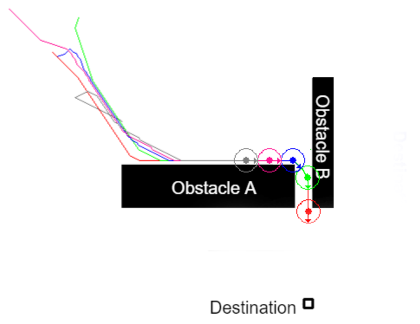


(a) UAVs forming Queue Formation

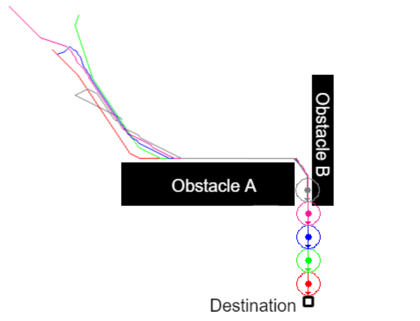
(b) UAVs in queue formation bypassing obstacle by keeping safe distance

FIGURE 6. UAVs at random coordinates coming into formation by finding their respective leaders and bypassing an obstacle. Here, the solid circle in the center of each drone shows the Collision Radius, the outer hollow circle depicts the Detection Range and the arrow indicates the direction of movement.

The trajectories of the drones while maintaining the formation and navigating through multiple obstacles are illustrated in Figure 7. The situation whence the main leader observes the second obstacle, namely Obstacle B, while traversing along obstacle A is shown in Figure 7(a). Since there are two obstacles with a large enough gap between them, the swarm's leader decides to go through the space between the two obstacles, and the other UAVs follow the leader (multiple obstacles case in Algorithm 4). Notice that in such a situation, the shape of the queue formation is not a rigid line. Instead,

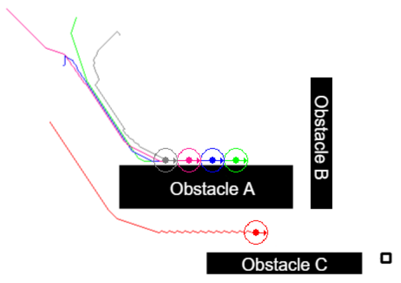


(a) UAVs in formation going through opening between obstacles

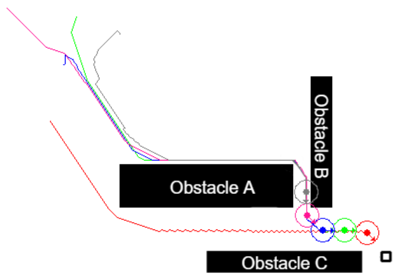


(b) UAVs coming out of the obstacles and reaching the destination point

FIGURE 7. Flexible queue formation through obstacles.

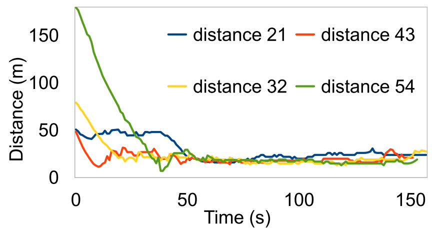


(a) Leader moves past the obstacle before its follower reaches it

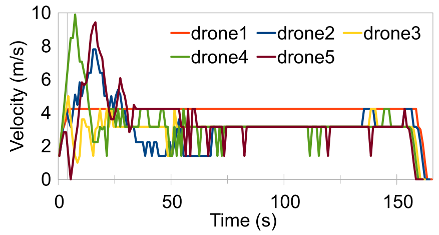


(b) Follower start navigating towards the destination and comes into ordered formation when the leader comes in its range

FIGURE 8. Leader moves past the obstacle before formation.



(a) Distance of each drone from its respective leader



(b) Velocities of all UAVs

FIGURE 9. Distance and velocity graph of the UAVs.

the drones reshape the formation into a bent/arched line e.g. when passing between the obstacles, and return to a straight line formation when they have passed the obstacles. This demonstrates the robustness and agility/adaptability of the proposed algorithm. The reformation process after coming out of the obstacles and next to the destination is shown in Figure 7(b).

An interesting situation, namely, the lost drone scenario is illustrated in Figure 8, where one of the drones wanders too far off from its immediate follower. Basically, it moves past the obstacle before other drones come into formation and it is, therefore, not in the range of its follower drone In the instant scenario, the global leader, namely UAV1, is hidden by an obstacle, therefore, its follower, namely UAV 2, loses connection to it. Now, till such time that UAV1 comes in the visible range of UAV2, the rest of the swarm continues its journey toward the destination with UAV2 as the temporary leader, as shown in Figure 8(a). When UAV1 becomes again visible to UAV2, Figure 8(b), the swarm immediately resumes the original formation and UAV2 and its followers accelerate towards the original leader UAV1. If UAV1 remains invisible/lost, UAV2 and its followers continue their journey without UAV1 until they reach the destination.

In order to analyse the efficiency and the performance of the proposed algorithm, we report and compare some acquired measurements of the system that is based on our first experiment as shown in Figure 6(b) above.

The velocity of each UAV and the relative distance between each UAV and its respective leader are shown in Figure 9. It is evident from the figure that, after the warm-up phase, the relative distances and velocities remain within close range avoiding considerable fluctuations in either parameter, please see Figures 9(a) and 9(b) respectively. This validates our claim that the proposed algorithm reliably maintains tracking and keeps safe distance between each leader and its follower in the swarm. The occurrence of momentary peaks in velocity of some drones happens due to the collision avoidance scene illustrated in Figure 7.

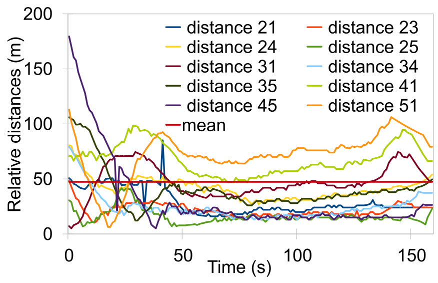


FIGURE 10. Relative distance of all drones from each other.

TABLE 2. Calculated σ of UAVs before and after coming into formation.

UAV No.	Standard Deviation before formation (σ_1)	Standard Deviation after formation (σ_2)
UAV 2	10.5430785734	7.0531634715
UAV 3	5.0412687286	3.1362290768
UAV 4	11.1664821141	3.128475562
UAV 5	36.9613886179	3.5387336333

The relative distances amongst all UAVs are shown in Figure 10. This is a metric to show how well the formation is being kept. As the graph shows, the UAVs are spawned from random locations, then the formation algorithm is initiated resulting in UAVs moving closer to their respective leaders while maintaining the minimum required distance. Table 2 shows the standard deviation calculated before and after the UAVs have reached queue formation. Here, σ_1, denotes the standard deviation of the drones' distances from the starting point when the nodes are at random coordinates and until they come into formation. Whereas, the standard deviation of the distances calculated from the point the drones reach the formation until they arrive in the final destination is denoted by σ_2. It is evident from the values shown in the table that the proposed algorithm maintains the formation tightly, without any significant fluctuations in distances, especially after the UAVs have reached the desired formation.

For comparing our proposed technique with the state-of-the-art algorithms, we implemented the formation and collision avoidance algorithm presented in [23] and [27] and set it side by side with our proposed method. Figures 11, 12, and 13 show the simulation results for distance maintenance between the first three nodes. It can be seen that our proposed algorithm maintains the distances between the drone-pairs within a tight range as compared with the reference methods presented in [23], [27].

The reference algorithm [23] takes action when both edges of an obstacle are visible, whereas the algorithm in [27] starts taking collective measures as soon as an obstacle comes within the detection range, and our algorithm is an extended version of the implementation of [27] and with the help of Thin-Plate Splines technique it maintains the formation even more aggressively and reducing the overall energy of the system. Furthermore, the authors in [23] have not considered alternative measures for two or more closely placed obstacles. If two obstacles are in close proximity to each other,

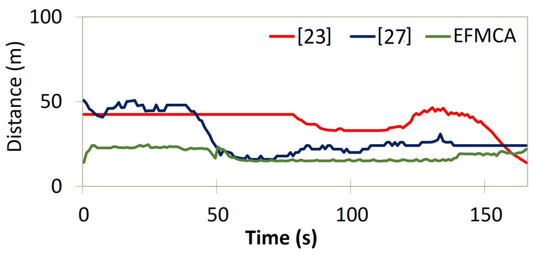


FIGURE 11. Comparison of distance maintenance from UAV2 to UAV1.

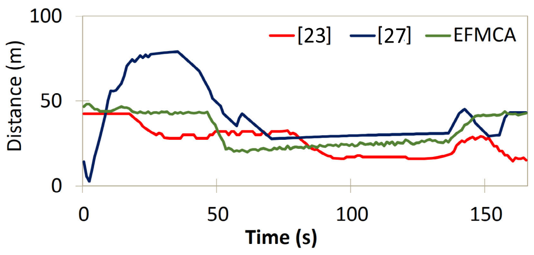


FIGURE 12. Comparison of distance maintenance from UAV3 to UAV1.

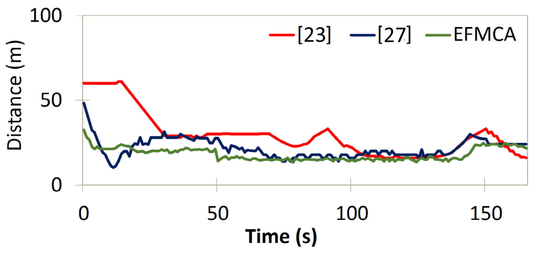


FIGURE 13. Comparison of distance maintenance from UAV3 to UAV2.

the method always considers them a single object, even in the case where a clear gap exists between the obstacles. This leads to sub-optimal flight paths. Since collision range is around the obstacles, so the UAVs can go through the obstacles rather than going around them. In our algorithm, on the other hand, the detected gaps are taken into consideration. The algorithm determines if there is sufficient space between the obstacles for the UAV to go through and takes action accordingly.

The change in temperature, i.e., the instantaneous value of the TPS energy function E_{TPS}, and the sum of this parameter that represents total disturbance suffered by the system of five drones during the complete flight from the launch of individual drones to their arrival at the destination of our proposed approach and the compared works are shown in Figures 14 and 15.

There are a few interesting things to note from Figure 14. Firstly, the initial few seconds show almost zero temperature for [23], the reason being that this algorithm assumes that drones are launched in proper formation. The other two algorithms show a large variation in temperature owing to the fact that drones are assumed to have been launched from random locations in close proximity. Here, EFMCA shows

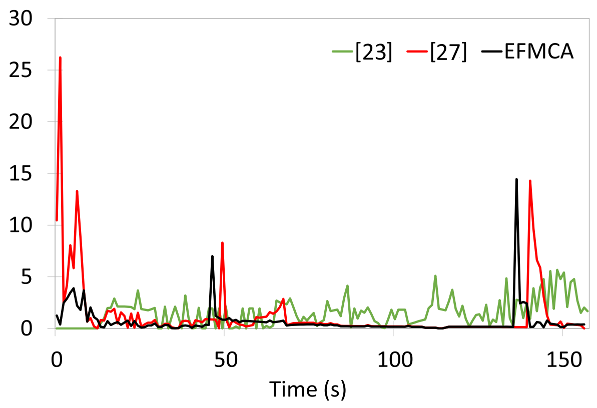


FIGURE 14. Change in temperature of the system as a whole.

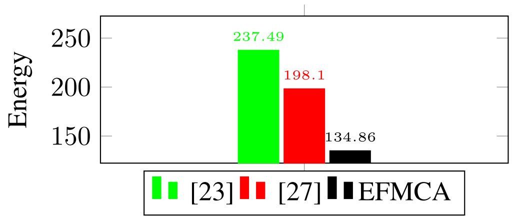


FIGURE 15. Total energy of the system.

better management of formation as is evident from reduced overshoots of temperature. The peaks around 40 seconds on the timeline represent gradual deviation in route to avoid Obstacle A with minimum disturbance in formation. Finally, the comparison of widths of peaks around 140 seconds shows that EFMCA resumes the formation considerably quickly as compared with the algorithm in [27]. This is a direct consequence of our approach of integrating TPS with formation and collision avoidance algorithm that brings the drones back in formation in a timely manner after disturbances caused by the obstacles. It is to be noted that algorithm in [23] does not try to bring the drones back into formation after splitting, as is evident from Figure 14 where the disturbance as measured by temperature does not return to zero. Thus, our method is more aggressive in balancing and bringing the system back to its stable state after disturbances caused by any obstacles. Also, as explained earlier we employ a non-rigid mapping function for point set registration of individual drones to formation locations. Therefore, in case a drone goes ahead of its leader during a maneuver, rather than slowing down to retain its original formation position as required by earlier approaches, we make this drone the new leader. Consequently, the speed of the formation as a whole increases, resulting in faster completion of the mission. This effect may be seen by *EFMCA* peaks occurring gradually earlier than the peaks of [27] in Figure 14. In order to show the overall efficiency of our scheme vis-a-vis the competing algorithms, we calculated the sum of all disturbances suffered by the system of drones during the complete mission. The result is shown in Fig. 15 where energy of the system for each of the three schemes refers to the area under the respective curve in Figure 14. It is evident from Fig. 15 that a swarm under our proposed EFMCA algorithm suffers much less overall disturbance as compared with other known schemes.

A. VALIDATION OF OUR SIMULATION RESULTS VIS-A-VIS INDUSTRY STANDARD

In order to validate our results further, we chose to deploy our proposed algorithm on **SwarmLab: a MATLAB Drone Swarm Simulator** [44], which is an open-source environment developed by *Laboratory of Intelligent Systems (LIS), Ecole Polytechnique Federale de Lausanne, Switzerland.* This simulation environment reflects the behaviour of the industrial drones, and also with the least amount of redundancy. Furthermore, the physical constraints (e.g. mass, inertia) are also supported and modelled in this environment.

Figure 16 shows the snapshots of the simulation output, taken at different intervals by implementing the EFMCA algorithm in the *SwarmLab* for observing the behaviour and effectiveness of the proposed algorithm in a different environment. Figures 16(a) and 16(b) show the initial placements of the nodes in the 3D and 2D views, with cylindrical obstacles placed in the environment. The nodes then accelerate and

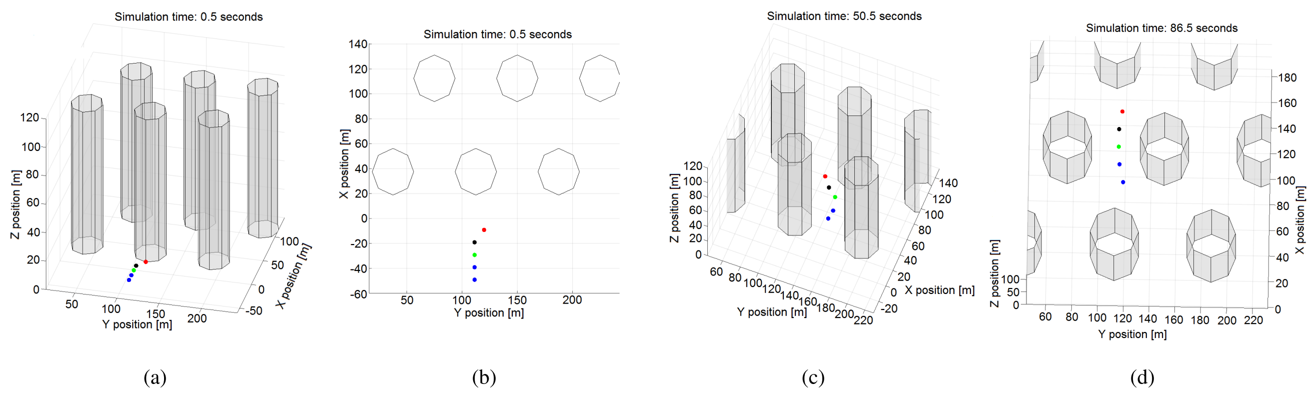
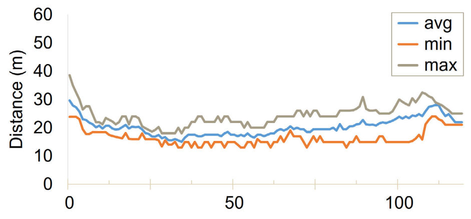
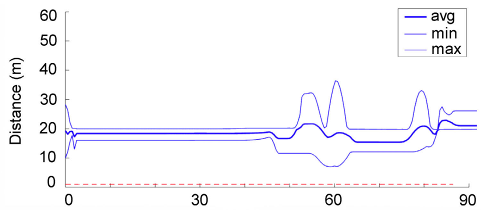


FIGURE 16. Simulation snapshots in SwarmLab. (a) 3D view. (b) at the start of simulation. (c) halfway through the simulation, navigation through the obstacles. (d) towards end of simulation.

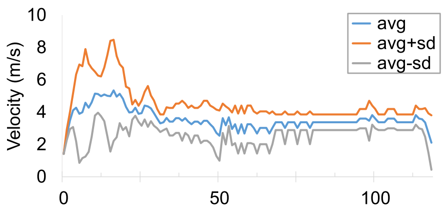


(a)

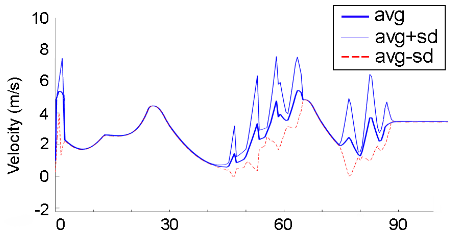


(b)

FIGURE 17. Average, minimum, and maximum distance maintained between the nodes. a) In our environment, b) implementation in SwarmLab.



(a)



(b)

FIGURE 18. Average velocity, average + standard deviation, and average - standard deviation of the swarm. a) our environment, b) SwarmLab.

decelerate to reach their desired positions w.r.t. their immediate leaders, as visible from the average velocity graph in Figure 18(b). Figure 16(c) shows the state of the swarm navigating through the obstacles at the half-way stage of the simulation. It is observed that the nodes maintain the flexible queue formation and rapidly decrease any disturbances caused due to the presence of the obstacles in the path. Figure 16(d) shows the snapshot towards the end of the simulation, where the swarm has successfully managed to avoid multiple obstacles in its path while maintaining the defined safe distance from the obstacles.

Figure 17 shows the comparison of the distance maintained between the nodes utilizing the EFMCA algorithm in our own environment and in *SwarmLab*. The analysis of the behaviour of the distance maintained by the nodes shows similar trends in the results obtained from the two environments. The difference in momentary peaks or disturbances in Figure 17(a), as compared with Figure 17(b), is due to the absence of some dynamics, such as a drone's mass and inertial movement, in our Python based environment which is still being developed constantly. However, the average distance trend comparison between the two environments provide enough evidence that the algorithm performed as expected in the state-of-the-art third party simulation environment.

Figures 18(a) and 18(b) show the overall trend of the velocity of the swarm throughout the simulation, with average, average + standard deviation, and average - standard deviation. The initial peaks in the graphs are due to the nodes accelerating to reach their desired coordinates in the formation. Moreover, the average velocity trend in Figure 18(b) showcases the smooth acceleration and deceleration, whereas the sharp momentary peaks Figure 18(a) are due to the absence of some of the dynamics in our environment. However, the average trend is similar as in the absence of the obstacles, the average velocity of the swarm is maintained at around 3.5 m/s.

VII. CONCLUSION

In this paper, we proposed a Thin-Plate Spline inspired formation maintenance algorithm for multiple UAVs, integrated with a collision avoidance capability. We theoretically investigated the behaviour of the proposed algorithm and tested it in a simulation environment. The simulations demonstrated, that the simulated UAVs were able to dynamically and reliably bypass obstacles without colliding with them while maintaining the given swarm formation very closely during maneuvers and reverting to it in a timely manner. By the ability to accelerate and decelerate on demand, the drones can efficiently reach their respective leaders or find their places in the formation when they start at random coordinates or wander off due to the presence of obstacles in their vicinity. Furthermore, the decentralized distribution of the algorithm allows the UAVs to take local decisions when in close proximity to obstacles, making the method highly robust and efficient. The collision avoidance scheme is able to flexibly handle situations with multiple detected obstacles. Moreover, the algorithm also takes care of the case where a UAV goes outside the visibility range of its leader; such lost UAVs are routed towards the destination by making a temporary formation if necessary. The multi-priority strategy works appropriately, changing the priorities of the different parts/functions of the algorithm whenever needed. As a comparison, we showed that the proposed algorithm outperforms the two known existing methods [23], [27], in terms of the response time, flexibility, and robustness. Another important contribution of our approach is that by employing non-rigid mapping between drones and formation positions, the overall speed of the swarm suffers minimum lag owing to avoidance maneuvers, as compared with the above referred earlier approaches. This effect can be quite significant for those missions where the time to reach the destination is of critical importance.

In our future work, we plan to extend the algorithm to handle the 3-dimensional movement along with the introduction of other environmental effects such as air drag on the individual nodes as well as on the overall shape of the swarm. For instance, in a queue formation, the preceding nodes will experience a lesser effect of drag and will subsequently consume lesser power as compared with the leader. Similarly, studying the effect of air drag on a multi-layered V-shaped formation will be interesting to analyse as well. This study can help in optimizing the resource management in the swarm. Furthermore, we plan on testing the proposed approach in real-time by performing practical experiments and analyzing its effectiveness.

REFERENCES

[1] M. Campion, P. Ranganathan, and S. Faruque, ''Notice of removal: A review and future directions of UAV swarm communication architectures,'' in *Proc. IEEE Int. Conf. Electro/Information Technol. (EIT)*, May 2018, pp. 0903–0908.

[2] M. Bowkett, K. Thanapalan, and E. Constant, ''Operational safety analysis and controller design of a dual drones system,'' in *Proc. Int. Symp. Comput. Sci. Intell. Controls (ISCSIC)*, Oct. 2017, pp. 82–87.

[3] Y. Cao, M. Li, I. Å Vogor, S. Wei, and G. Beltrame, ''Dynamic range-only localization for multi-robot systems,'' *IEEE Access*, vol. 6, pp. 46527–46537, 2018.

[4] 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.

[5] G. Ladd and G. Bland, ''Non-military applications for small UAS platforms,'' in *Proc. AIAA Infotech Aerosp. Conf.*, Apr. 2009, p. 2046.

[6] Á. Madridano, A. Al-Kaff, D. Martín, and A. A. D. L. de la Escalera, ''3D trajectory planning method for UAVs swarm in building emergencies,'' *Sensors*, vol. 20, no. 3, p. 642, Jan. 2020.

[7] 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.

[8] 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, May 2017.

[9] L. He, P. Bai, X. Liang, J. Zhang, and W. Wang, ''Feedback formation control of UAV swarm with multiple implicit leaders,'' *Aerosp. Sci. Technol.*, vol. 72, pp. 327–334, Jan. 2018.

[10] C. B. Low and Q. S. Ng, ''A flexible virtual structure formation keeping control for fixed-wing UAVs,'' in *Proc. 9th IEEE Int. Conf. Control Autom. (ICCA)*, Dec. 2011, pp. 621–626.

[11] W. Ren, ''Consensus based formation control strategies for multi-vehicle systems,'' in *Proc. Amer. Control Conf.*, 2006, pp. 1–8.

[12] L. Dong, Y. Chen, and X. Qu, ''Formation control strategy for nonholonomic intelligent vehicles based on virtual structure and consensus approach,'' *Procedia Eng.*, vol. 137, pp. 415–424, 2016.

[13] N. H. M. Li and H. H. T. Liu, ''Formation UAV flight control using virtual structure and motion synchronization,'' in *Proc. Amer. Control Conf.*, Jun. 2008, pp. 1782–1787.

[14] X. Wu, S. Wang, and M. Xing, ''Observer-based leader-following formation control for multi-robot with obstacle avoidance,'' *IEEE Access*, vol. 7, pp. 14791–14798, 2019.

[15] K.-K. Oh, M.-C. Park, and H.-S. Ahn, ''A survey of multi-agent formation control,'' *Automatica*, vol. 53, pp. 424–440, Mar. 2015.

[16] T. Balch and R. C. Arkin, ''Behavior-based formation control for multirobot teams,'' *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[17] J. R. T. Lawton, R. W. Beard, and B. J. Young, ''A decentralized approach to formation maneuvers,'' *IEEE Trans. Robot. Autom.*, vol. 19, no. 6, pp. 933–941, Dec. 2003.

[18] H. Su, X. Wang, and Z. Lin, ''Flocking of multi-agents with a virtual leader,'' *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 293–307, Feb. 2009.

[19] W. Yu, G. Chen, and M. Cao, ''Distributed leader–follower flocking control for multi-agent dynamical systems with time-varying velocities,'' *Syst. Control Lett.*, vol. 59, no. 9, pp. 543–552, Sep. 2010.

[20] A. Mahmood and Y. Kim, ''Leader-following formation control of quadcopters with heading synchronization,'' *Aerosp. Sci. Technol.*, vol. 47, pp. 68–74, Dec. 2015.

[21] S. He, M. Wang, S. Dai, and F. Luo, ''Leader–follower formation control of usvs with prescribed performance and collision avoidance,'' *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 572–581, 2019.

[22] Y. Kim and J. Choi, ''Fuel-efficient formation flight-control design based on energy maneuverability,'' *J. Guid., Control, Dyn.*, vol. 31, no. 4, pp. 1145–1150, Jul. 2008.

[23] J. Seo, Y. Kim, S. Kim, and A. Tsourdos, ''Collision avoidance strategies for unmanned aerial vehicles in formation flight,'' *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 6, pp. 2718–2734, Dec. 2017.

[24] Y. Lin and S. Saripalli, ''Collision avoidance for UAVs using reachable sets,'' in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2015, pp. 226–235.

[25] H. Pham, S. A. Smolka, S. D. Stoller, D. Phan, and J. Yang, ''A survey on unmanned aerial vehicle collision avoidance systems,'' 2015, *arXiv:1508.07723*. [Online]. Available: http://arxiv.org/abs/1508.07723

[26] B. M. Albaker and N. A. Rahim, ''A survey of collision avoidance approaches for unmanned aerial vehicles,'' in *Proc. Int. Conf. Tech. Postgraduates (TECHPOS)*, Dec. 2009, pp. 1–7.

[27] J. N. Yasin, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, ''Formation maintenance and collision avoidance in a swarm of drones,'' in *Proc. 3rd Int. Symp. Comput. Sci. Intell. Control*, Sep. 2019, pp. 1–6, doi: 10.1145/3386164.3386176.

[28] J. Sun, J. Tang, and S. Lao, ''Collision avoidance for cooperative uavs with optimized artificial potential field algorithm,'' *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[29] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, ''Navigation of autonomous swarm of drones using translational coordinates,'' in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness*, Y. Demazeau, T. Holvoet, J. M. Corchado, and S. Costantini, Eds. Cham, Switzerland: Springer, 2020, pp. 353–362.

[30] J. N. Yasin, S. A. S. Mohamed, M. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, ''Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches,'' *IEEE Access*, vol. 8, pp. 105139–105155, 2020.

[31] N. E. Smith, R. Cobb, S. J. Pierce, and V. Raska, ''Optimal collision avoidance trajectories via direct orthogonal collocation for Unmanned/Remotely piloted aircraft sense and avoid operations,'' in *Proc. AIAA Guid., Navigat., Control Conf.*, Jan. 2014, p. 0966.

[32] X. Zhang, A. Liniger, and F. Borrelli, ''Optimization-based collision avoidance,'' 2017, *arXiv:1711.03449*. [Online]. Available: http://arxiv.org/abs/1711.03449

[33] K. D. Do, ''Synchronization motion tracking control of multiple underactuated ships with collision avoidance,'' *IEEE Trans. Ind. Electron.*, vol. 63, no. 5, pp. 2976–2989, May 2016.

[34] E. Şahin, ''Swarm robotics: From sources of inspiration to domains of application,'' in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. Berlin, Germany: Springer, 2005, pp. 10–20.

[35] A. Martinoli, ''Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies,'' Ph.D. dissertation, Dept. Comput., Swiss Federal Inst. Technol. Lausanne (EPFL), Lausanne, Switzerland, 1999.

[36] R. C. Arkin and G. A. Bekey, *Robot Colonies*. Cham, Switzerland: Springer, 1997.

[37] C. R. Kube and H. Zhang, ''Collective robotics: From social insects to robots,'' *Adapt. Behav.*, vol. 2, no. 2, pp. 189–218, Sep. 1993.

[38] G. Beni, ''From swarm intelligence to swarm robotics,'' in *Swarm Robotics*, E. Şahin and W. M. Spears, Eds. Berlin, Germany: Springer, 2005, pp. 1–9.

[39] X. Fu, J. Pan, H. Wang, and X. Gao, ''A formation maintenance and reconstruction method of UAV swarm based on distributed control with obstacle avoidance,'' in *Proc. Austral. New Zealand Control Conf. (ANZCC)*, Nov. 2019, pp. 205–209.

[40] 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.

[41] 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.

[42] A. Myronenko and X. Song, "Point-set registration: Coherent point drift," 2009, *arXiv:0905.2635*. [Online]. Available: http://arxiv.org/abs/0905.2635

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

[44] E. Soria, F. Schiano, and D. Floreano, "SwarmLab: A MATLAB drone swarm simulator," 2020, *arXiv:2005.02769*. [Online]. Available: http://arxiv.org/abs/2005.02769

JAWAD NAVEED YASIN received the B.S. degree in electrical engineering from COMSATS University, Islamabad, Pakistan, and the M.S. degree in embedded computing from Åbo Akademi University, Turku, Finland, in 2017. He is currently pursuing the Ph.D. degree in embedded and electronics engineering from the University of Turku, Turku.

In 2009, he worked as a Teaching Assistant, for one semester, with the Electrical Engineering Department, COMSATS University, Pakistan. From 2009 to 2010, he worked as a Lab and Network Engineer with the National University of Computer & Emerging Sciences, Pakistan. From 2011 to 2016, he worked as a Hardware Engineer with Nordic IT Oy, Finland. In 2018, he worked as a Teaching Assistant in advanced sensors networking with the Department of Future Technologies, University of Turku. His research interests include agent-based modeling, swarm intelligence, embedded systems, collision avoidance, and autonomous vehicles.

SHERIF ABDELMONEM SAYED MOHAMED received the B.A. degree in electrical, electronics, and communication engineering from Ain Shams University, Egypt, in 2011, and the M.S. degree in electronics and information engineering from Kunsan National University, South Korea, in 2016. He is currently pursuing the Ph.D. degree with the University of Turku, Finland. His research interests include vision-based navigation algorithms for autonomous vehicles, embedded systems, swam intelligence, and machine learning.

MOHAMMAD-HASHEM HAGHBAYAN (Member, IEEE) received the B.A. degree in computer engineering from the Ferdowsi University of Mashhad, the M.S. degree in computer architecture from the University of Tehran, Iran, and the Ph.D. degree (Hons.) from the University of Turku, Finland.

Since 2018, he has been holding a postdoctoral and lecturer positions with the University of Turku. His research interests include high-performance energy efficient architectures for autonomous systems and artificial intelligence. He has several years of working experience in industry and designing IP cores as well as developing research tools.

JUKKA HEIKKONEN has been a Professor of Computer Science with the University of Turku, Finland, since 2009. He is also the Head of the Algorithms and Computational Intelligent (ACI) research group, where his current research is related to data analytics, machine learning, and autonomous systems. He has worked at top level research laboratories and Center of Excellences in Finland and international organizations (European Commission, Japan) and has led many international and national research projects. He has authored more than 150 scientific articles.

HANNU TENHUNEN is currently a Chair Professor of Electronic Systems with the KTH Royal Institute of Technology, Stockholm, Sweden. He has held a professorship position, as a full professor, invited professor or visiting honorary professor in Finland (TUT, UTU), Sweden (KTH), USA (Cornel U), France (INPG), China (Fudan and Beijing Jiatong Universities), and Hong Kong (The Chinese University of Hong Kong), and has an Honorary Doctorate from Tallinn Technical University. He has contributed to over 850 international publications with H-index 41. He holds nine international patents granted in multiple countries. He is a member of the Academy of Engineering Science of Finland. He has served in Technical Program Committee's of all major conferences in his area, has been the general chairman or vice-chairman or a member of Steering Committee of multiple conferences in his core competence areas. He has been one of the founding editorial board members of three scientific journal, have been a quest editor for multiple special issues of scientific journals or books, and have contributed numerous invited articles to journals.

MUHAMMAD MEHBOOB YASIN received the B.Sc. degree from the University of the Punjab, Lahore, Pakistan, in 1975, the M.Sc. degree in computer science from the University of Wales, Aberystwyth, U.K., in 1983, and the Ph.D. degree from The Open University, Milton Keynes, U.K., in 1986. He served in various research and development roles in Pakistan Atomic Energy Commission for over two decades, and then moved to academia, serving as a Professor, the Department Chair, and the Dean at renowned institutions in Pakistan. In 2009, he joined the prestigious Computer Laboratory, Cambridge University, U.K., as a Visiting Professor. He has been a (Full) Professor of Computer Networks with the College of Computer Sciences & Information Technology, King Faisal University, Saudi Arabia, since 2011. His research interests include computer networks, applied cryptography, the Internet of Things, and cybersecurity.

JUHA PLOSILA (Member, IEEE) received the Ph.D. degree in electronics and communication technology from the University of Turku (UTU), Finland, in 1999. He is currently a (Full) Professor of Autonomous Systems and Robotics with the Department of Future Technologies, UTU. He is also the Head of the EIT Digital Master Programme in Embedded Systems, EIT Digital Master School (European Institute of Innovation and Technology), and represents UTU in the Node Strategy Committee of the EIT Digital Helsinki/Finland node. He has a strong research background in adaptive multi-processing systems and platforms and their design, which includes 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 the 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.