SEMFIRE

Segurança, Exploração e Manutenção de Florestas com Integração de Robótica Ecológica

E4.2b: Initial Deployment of Scouts under Connectivity Constraints

Cofinanciado por:

Document Identification

Title	**SEMFIRE: Segurança, Exploração e Manutenção de Florestas com Integração de Robótica Ecológica**
Reference	CENTRO-01-0247-FEDER-032691
Duration	2018-10-01 - 2021-09-30
Head of Consortium (*Chefe do Consórcio*)	Ingeniarius, Lda. (Ingeniarius)
Copromoters (*Copromotores*)	Instituto de Sistemas e Robótica da Universidade de Coimbra (ISR-UC), SFera Ultimate, Lda. (SFera)
Coordinator	Micael S. Couceiro (Ingeniarius)

Work Package/Deliverable

Work Package (*Atividade*)	**A4. Multi-Robot Coordination in the Forestry Environment** (*Coordenação Multi-Robô em Ambiente Florestal*)
Deliverable (*Entregável*)	**E4.2b**: Initial Deployment of Scouts under Connectivity Constraints (*Implantação de Scouts com Restrições de Conetividade*)
Lead Copromoter	Ingeniarius
Due Date (*Mês*)	2020-07-31 (*M22*)

Change Record

Version	Date	Author	Summary of Changes
v0.9	2021-09-30	Ali Ahmadi	Contribution for Ch. 4
v1.0	2021-11-23	João Filipe Ferreira	Final Version

Contents

List of Figures

1 Introduction

The SEMFIRE project [16] encompasses several challenges within the field of forestry robotics and field robotics in general. This deliverable addresses one of these challenges in the context of the activity #4 of the project: multi-robot coordination in the forestry environment. More specifically, this deliverable addresses the initial deployment of scouts according with the heterogeneous multi-robot system proposed in the project [16]. The complete definition of the use-case scenario and its requirements is presented in the deliverable E.1.1 [61].

Fig. 1 depicts the use-case scenario envisioned in the SEMFIRE project. The SEMFIRE solution involves a heterogeneous multi-robot team comprising a Ranger and a set of Scouts [16]. The Ranger is a large-sized powerful multi-purpose tracked robotic mulcher based on the Bobcat platform. It can operate autonomously or semi-autonomously to clear forestry areas, being equipped with a forestry mulcher attachment to cut down thin trees and shred ground vegetation to grind them into mulch. The Scouts are small unmanned aerial vehicles (UAVs) with self-organizing capabilities to explore and supervise wide forestry areas, as well as aiding the Ranger in its navigation and perception capabilities. For instance, Scouts can make more reliable and precise the Ranger's localization within the field and augment the Ranger's perception capabilities with their airborne view of the operating area.

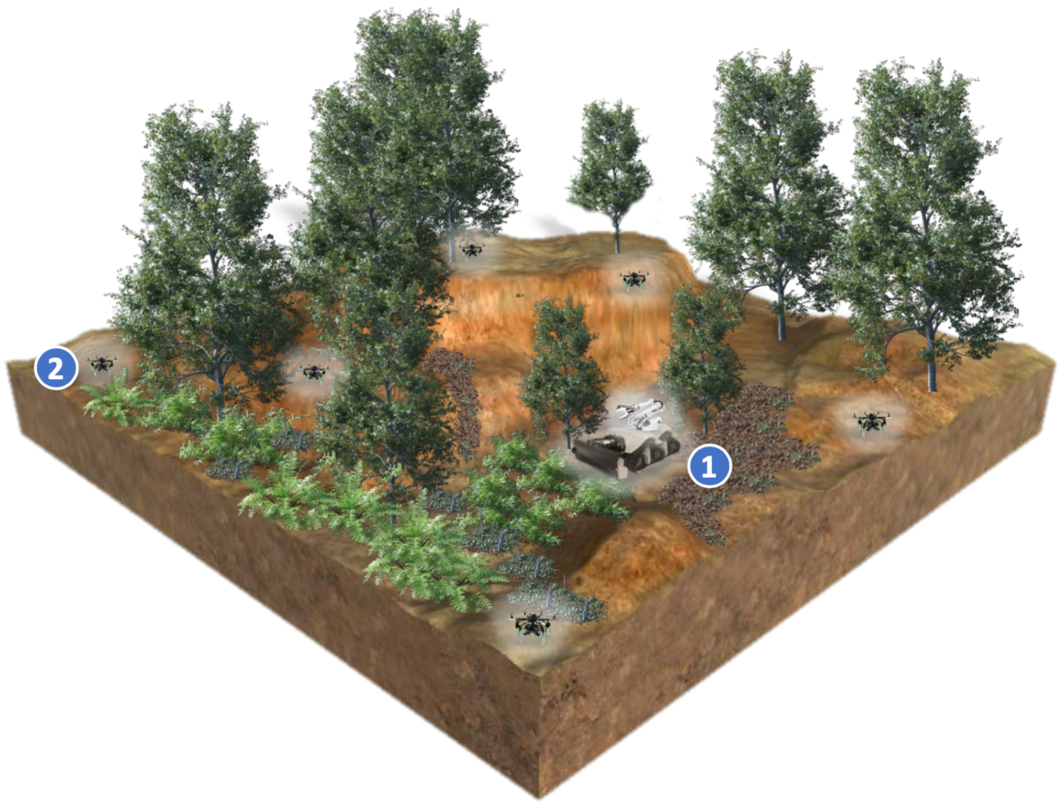


Figure 1: Illustration of the heterogeneous multi-robot system deployment in the SEMFIRE project: 1) the Ranger, a large-sized powerful tracked mobile mulcher; 2) a Scout a small assistive flying robot and member of a fleet of such robots, that can help in exploring and supervising wide forestry areas and aid in the Ranger's navigation capabilities. Figure reproduced from [16].

In the beginning of a mission in a forestry area, the Ranger is moved by an operator – be it on-site or over the Internet in a remote location – to the target site via remote control. At this stage of the mission, the Scouts are parked on top of the Ranger and are thus transported to the target site by that larger platform before being deployed. This type of heterogeneous multi-robot system is denoted in literature as a marsupial team [15]. After the multi-robot system reaches the target site, the operator issues the command for the landscaping maintenance mission to initiate. The first stage of the mission is denoted as reconnaissance and involves using the set of Scouts to collectively explore and map the surrounding environment with the intent of finding new regions of interest containing forest debris and defining the area of the operation for the Ranger to start working in its mulching and forest debris removal task.

Before the Scouts can start exploring and mapping the target site, they have to be initially deployed by the Ranger within the target area. Since a communication network infrastructure is not usually present in the forestry environment, the team of Scouts (and the Ranger) have to rely on a mobile ad hoc network (MANET) created and managed by the own team from the beginning of its autonomous task [13]. This raises important constraints in the spatial distribution of the Scouts when initially deployed so that the heterogeneous team becomes properly connected through the MANET to start the reconnaissance task. Besides connectivity constraints, the initial deployment of Scouts must avoid areas of no interest, including obstacles, and should be efficient in scattering the available Scouts throughout the target site so as to take advantage from space distribution provided by the multi-robot system [17]. This problem is denoted in literature as the initial deployment problem [14, 17, 15] and it is the main focus of this deliverable.

This deliverable is organized as follows. After this introduction, the next section (Section 2) surveys literature on the initial deployment problem. Section 3 presents the design of the initial deployment strategy that is going to be used in the SEMFIRE project. Section 4 presents preliminary work towards the implementation and validation of that strategy, resorting to a 3D multi-robot simulator. Finally, Section 5 summarizes the conclusions of current work on the initial deployment problem and points out future actions to be taken in the project within that topic.

2 Literature review

The initial deployment problem considers the number of robots to be deployed, *i.e.* the teamsize, and the choice of the robots' initial locations. Typically the problem is addressed through the concept of marsupial teams [45] whereby larger robotic platforms carry a set of smaller robotic units to be deployed, which are denoted as marsupial robots. The problem is relevant because a non efficient deployment may greatly jeopardize the mission [66]. Also, in multi-robot systems whose control policy is ruled by iterative optimization algorithms, robots' initial locations correspond to initial estimates of such algorithms which, being of good quality, can lead to faster convergence [41].

One of the first works that addressed the effect of different initial deployments was presented in [12]. The authors evaluated their coverage algorithm using both centralized and random initial deployments and concluded that the algorithm convergence was slower using a random initial deployment but tends to lead to better overall coverage for sparse topologies. Minimalist requirements on the robotic hardware, namely knowledge of the number of wireless links and bumper sensors for collision avoidance, are assumed. Most works in the literature present a random initial deployment in which robots are scattered [35, 25].

In [35], a three-dimensional (3D) deployment strategy was explored. The authors did focus on a deployment strategy in which the initial distribution of all robots is arbitrary and their positions are distinct. The main difference with other works resides in the fact that robots autonomously move in a 3D space (*e.g.*, coordinated formation flight and reconfiguration of unmanned aerial vehicles [27]) instead of a planar scenario. Therefore, the authors provide a coverage and connectivity strategy using a self-configuration process to enable robots to form a 3D tetrahedron shape. In terms of sensing and communication capabilities, robots do not share any common coordinate system, and do not retain any memory of past actions. They can detect the positions of other robots only within their limited sensing range. In addition, each robot does not communicate explicitly with other robots. Despite the positive results inherent to a random deployment, in real situations, it is necessary to ensure several constraints of the system. For instance, if the MANET supports multi-hop connectivity, these constraints may significantly increase the complexity of the random distribution since it would depend not only on the communication constraints but also on the number of robots and their own position. Moreover, a random deployment may lead to unbalanced distributions, therefore increasing the number of needed robots and energy throughout the scenario.

In [47], the authors described an approach for deployment of a swarm of heterogeneous autonomous vehicles based on descriptor functions. Similarly to the work presented in [14], each robot is treated as an agent of the network in which repulsive forces are computed as a function of the distance between agents, to spread the network throughout the environment. In the simulation experiments conducted, it was simply assumed that agents are capable of performing the area coverage task that is assigned to them and no sensing details were provided. The authors chose an initial deployment in which robots started from a compact configuration. Although this kind of initial deployment strategy works well when the main purpose is to spread the robots within area coverage scenarios, no other deployment strategy was taken into consideration by the authors. Also, despite being similar to the deployment of military units, it requires for exploring robots to find energy-efficient paths to avoid jeopardizing the success of the mission.

In [25], the authors presented a strategy to assign starting points and orientations of robots within circles of different radius around a prey. Hence, using a team of 16 robots, the authors assign 16 different positions and 4 different orientations which are randomly assigned at each trial. The robots used are reconfigurable, being equipped with infrared sensors, a camera, and grippers, which enable them to form chains of swarm robots through the use of local communication. Despite the apparent advantages of this deployment strategy in this context, no other strategies were evaluated, thus being hard to predict if the number of unsuccessful trials is somehow related with the initial deployment of robots.

Several works have focused on marsupial teams where robots are deployed in a unique and compact unloading location [18, 30, 66, 47]. For instance, in [66], the authors address a multi-robot coverage task and deal with the problem of determining the number and size of robot groups that need to be unloaded from a carrier, and the initial robot locations. A solution that can cover the deployment area within the maximum coverage time allowed is iteratively determined by varying the number and sizes of groups based on heuristics. In order to compute their algorithm, the authors assume that the density of obstacles is available and simulations modeling PPRK and Pioneer-3DX robots with little communication requirements are assumed. In addition, besides only considering a scanline deployment strategy, the authors also assume to have a unique unloading location for the whole team of robots. In other words, the carrier robot transports the smaller robots into the field and the latter robots need to autonomously move from the unloading location to their individual starting locations.

Human-marsupial robot teams for urban search and rescue were firstly studied by Murphy *et al.* [46, 45] The team members were divided in three roles: Human, Dispensing Agent (a.k.a., “mother”), and Passenger Agent (a.k.a., “daughter”), similarly to kangaroo societies. The mother robot provides not only transportation to the daughters, but also power (a.k.a., “food”) and help. The latter refers to communicating suggestions, warnings, or to rescue the daughters, which are responsible for exploring remote locations and are equipped with a camera, a microphone, two headlights, and a video transmitter to send images directly to the human. The role of the human rescuers is to supply decision-making capabilities, remotely speak with victims, and collect information about their state, number of victims, location, and presence of hazards, like gas leaks. Heuristics were proposed for the deployment of micro-rovers.

In [52], the authors divided the population of real robots into two different types of platforms: rangers and scouts. The rangers consisted of large platforms used to transport the scouts over distances of several kilometers and deploy the scouts rapidly over a large area. The scouts consisted of small and expendable robotic platforms with cylindrical shape, able to roll and jump over obstacles, and were used to sense the environment, act on their sensing, and report their findings. They were endowed with magnetometers and tiltometers, a CMOS camera, a passive infrared sensor, a microphone, a vibration sensor, a gas sensor, audio transmitters/receivers, and are able to receive remote commands. The authors did not focus on the cooperation among robots. Also, the deployment strategy was accomplished through a launcher system equipped on the ranger that was able to throw the scouts up to a range of 30 m. However, in most cooperative applications in unknown and unstructured scenarios (e.g., forestry exploration), this would require robots to be able to measure the relative distance between themselves or to be equipped with global localization systems (*e.g.*, GPS) to allow an efficient processing of the exchanged information.

Howard *et al.* [30] proposed a strategy whereby the exploring robots deploy themselves in the unknown environment in an incremental way and assure line-of-sight contact with teammates. In their strategy, no carrier robots are considered. Robots have the ultimate goal of mapping the environment while using teammates as landmarks. A greedy deployment algorithm is presented, which aims at maximizing the coverage area by exploring robots. The work has been tested using four Pioneer 2DX mobile robots equipped with Sick laser rangefinders. Other works, like [53] and [9], also follow self-deployment strategies for military, search and rescue and exploration missions.

Similarly to [52], Couceiro *et al.* [14, 17, 15] handles the initial deployment problem hierarchically by considering a heterogeneous multi-robot system comprising rangers and scouts. However, in contrast, they make use of a marsupial multi-robot system [45] wherein each ranger handles the initial deployment of scouts in a distributed and autonomous fashion. The ranger successively deploys the scouts, instructing them of their initial pose while maintaining a maximum communication range between scouts, thus guaranteeing the full connectivity of the MANET. Hence, each scout is both an exploring agent and a mobile node of a MANET that performs packet forwarding using multi-hop communication. In [14], the initial location of a scout is chosen randomly within a circumference with radius equal to the maximum communication range between two nodes of the MANET, centred in the previous scout's location, but ensuring that the chosen location did not lie on an obstacle. This randomized choice may cause an unbalanced deployment, which increases the number of scouts needed to cover an area. To overcome this shortcoming, the authors proposed an alternative initial deployment strategy [13] whereby the scouts are spatially distributed accordingly with a Spiral of Theodorus [24], a.k.a. square root spiral [26]. In [17, 15], the strategy was further refined by proposing the Extended Spiral of Theodorus (EST), which does not have a fixed central point as in [13]. Instead, the central point will vary over time depending on the scouts previously deployed, *i.e.*, the number of scouts already deployed and the distance between successive scouts. In this extended version, the strategy takes into consideration the fact that the signal propagation may vary throughout the environment due to signal fading in obstacles. Extensive simulation experiments revealed that the merits of the EST strategy compared with the randomized initial deployment. The strategy was also validated in real experiments involving 3 rangers deploying 5 scouts each in a 200 sq. m planar area.

More recently, the strategy proposed in [13] for planar environments was extended to 3D scenarios, more precisely to 3D exploration with unmanned underwater vehicles, by adding another dimension to the Spiral of Theodorus, thus creating a helix [23]. Since the unknown underwater scenario is usually less deep than wide or long, the helix travels from the surface to the seabed. The angle formed by the helix with the horizontal plane can be adjusted to the depth of the scenario and the number of swarm members. This 3D strategy was tested in a simulation environment created with Gazebo.

Metiaf *et al.* proposed a modification of particle swarm optimization to move the sensors of a wireless sensor network from random initial positions to a set of locations provided by the optimization algorithm [42]. The algorithm aims at optimizing coverage of the area while minimizing energy consumption and avoiding obstacles.

In [59], the authors propose two low-complexity algorithms for the joint transmit power and UAV trajectory design in the context of a UAV team-enabled network. This problem has similarities with the aforementioned initial deployment problem of a set of exploring scouts.

3 Initial deployment strategy

The initial deployment strategy adopted in the SEMFIRE project for the marsupial team comprising a Ranger and a team of Scouts is based on the Extended Spiral of Theodorus (EST) previously developed by some of the project team members to be used in planar environments [17, 15], with the necessary adaptations to the 3D forestry scenario of the project that will be presented in this section.

The EST follows the geometrical arrangement depicted in Fig. 2 wherein distance d is not necessarily constant. In the simplest case, this distance is a function of the maximum communication range in line-of-sight which could be assumed constant in an area not containing obstacles. However, in the presence of obstacles (*e.g.* trees, shrubs, *etc.*) the signal propagation model becomes much more complex and the maximum communication range will vary accordingly with the density and type of obstacles. Therefore, the arrangement depicted in Fig. 2, though being useful to understand the essence of the strategy, may be too simplistic for the forestry scenario if d varies throughout the area. In this case, the Spiral of Theodorus will not have a fixed central point as pointed out in [15], being denoted as the *Extended* Spiral of Theodorus (*i.e.* EST). The reader is referred to [17, 15] for details about the mathematical formulation of the EST.

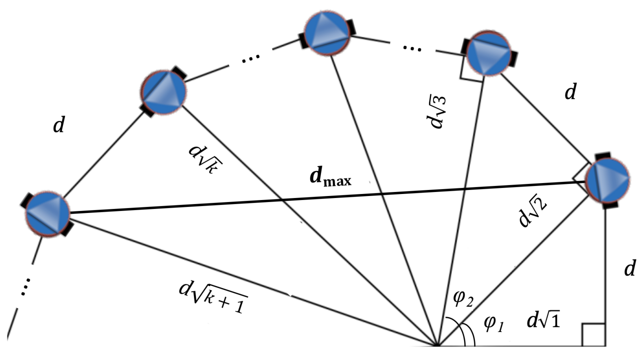


Figure 2: Extended Spiral of Theodorus. Figure reproduced from [17].

As mentioned in Sec. 1, the Ranger is a large-sized tracked robotic mulcher (see Fig. 3) that carries the Scouts on top in the beginning of the mission, before deploying them in the target area for the reconnaissance task. The Scouts are UAVs of the quadrotor (a.k.a. quadcopter) type, thus being able to take off and land vertically and hovering in the air.

Before the mission starts, a given nominal working altitude, h_n, is selected for the Scouts, which must be within the physical limitations of the aerial platforms, including the maximum communication range w.r.t. ground, should fit the regulations for the use of UAVs and, most importantly, should optimize the airborne perspective of the area according with the range and resolution of sensors on board each Scout. The EST is planar and computed onto a

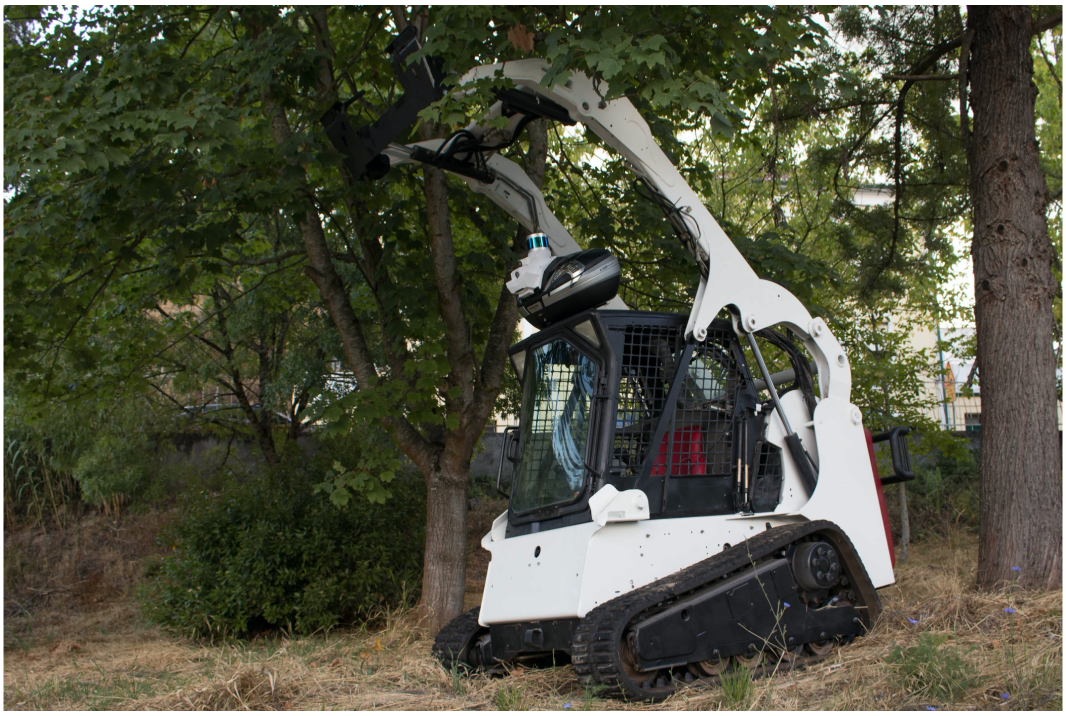


Figure 3: The Ranger.

horizontal plane located at the height h_n. The Scouts are deployed in locations indicated by the EST, which assure the connectivity of the MANET and the avoidance of obstacles, namely crests of hills and trees canopy. From an algorithmic point of view, these types of artifacts are just obstacles, as in [15], but they have to be detected not only by the Ranger based on its estimated traversability [38], but also by the Scouts, after taking off, to avoid and negotiate aerial obstacles. This means that the Ranger computes an initial estimate of the EST before the take off of any Scout, which is based solely on the available information about traversability on the ground around its initial position. During the successive deployment of Scouts, the EST is adjusted and updated, being recomputed to satisfy the constraints found by the Scouts during their flight and to be compliant with the traversability map incrementally built and updated by the Ranger [38].

Another important difference compared with the original EST method proposed in [17, 15] is that Scouts take off from the Ranger (and eventually fly from that place), rather than just being "unloaded" in the place by the Ranger as in the deployment of terrestrial Scouts [15]. Moreover, the take off may happen not necessarily at the projection on the ground of the intended position for the Scout, so that the initial deployment can be faster, as described below.

When the Ranger arrives the target area (with the Scouts parked on top of it), which means that it is at the projection on the ground of the first position indicated by the EST, the initial deployment of Scouts begins according with the following procedure:

1. The Ranger issues the command for taking off of the first Scout to initiate. The first Scout takes off and tries to place itself over the Ranger at altitude h_n, or as close as possible to that position (also at altitude h_n) if some tree canopy prevents that exact position. The Scout remains hovering at the acquired position. The Ranger updates the EST based on constraints found during the flight of the first Scout deployed.

2. The Ranger starts moving on the ground towards the projection on the ground of the location prescribed to the second Scout and keeps updating its navigation map and

estimating traversability [38]. During the deployment of the second Scout, and also of the remaining Scouts, the projection on the ground of the locations prescribed for the Scouts by the current EST are used by the Ranger as navigation waypoints. As the Ranger is much slower than Scouts, the next Scout (*e.g.* the second Scout) is likely to receive the take off command before the Ranger reaches the corresponding position on the ground to the prescribed position for that Scout. This makes the Scouts deployment faster. Even so, the Ranger keeps moving throughout those waypoints in order to make shorter the distance between the take off location and the hovering position of the Scouts that were not deployed yet.

3. The Ranger issues the command for taking off of the second Scout to initiate. The second Scout takes off and flies towards its prescribed position in the EST at altitude h_n, or as close as possible to that position (also at altitude h_n) if some tree canopy prevents that exact final position, or if the minimum communication signal strength (w.r.t. the previous Scout, *i.e.* the preceding node of the MANET) cannot be satisfied due to signal fading. The Scout remains hovering at the acquired position. The Ranger updates the EST based on constraints found during the flight of the second Scout deployed.

4. The previous step is repeated until all Scouts take off and are deployed in the target area.

5. While the ranger keeps updating the traversability map [38] and reaches successively the waypoints, the EST is iteratively updated which means that the position acquired by a Scout already deployed may change; *e.g.* the Ranger may find unreachable a position where a Scout is already hovering. In this case, the Ranger instructs that Scout to reposition itself, and eventually instructs the Scouts deployed after that Scout to also reposition themselves, as the position of each Scout depends on the position of previously deployed Scouts in the EST strategy.

In the end of the initial deployment process, the Ranger will have confirmed that all positions where Scouts are deployed and hovering are traversable on the ground. All Scouts are hovering in locations that optimize the coverage of the target based on the EST strategy while, at the same time, assure the MANET connectivity, avoid places of no interest that are unreachable by the Ranger, and avoid aerial obstacles created by trees canopy. The Ranger does not need necessarily to navigate to and reach every waypoint, because it may sense traversability at a distance using its sensors and can also receive percepts from the Scouts hovering in the air.

The heterogeneous team is then ready to start the reconnaissance task so as to map the vegetation within the target area, with data acquired on the ground and from the air, and identify regions to be cut and cleared by the Ranger.

4 Fault-Tolerant Formation Control, Collision and Obstacle Avoidance Under Connectivity Constraints

4.1 The MRS-UAV system in the context of the SEMFIRE project

Experiments with teams of robots, whether aerial, underwater or ground robots, in the physical world often represent a challenging task due to the complexity involved. One has to make sure that the robot hardware configuration, the software integration and the interaction with the environment is thoroughly tested so that the deployment of robot teams runs smoothly. This usually requires long preparation time for experiments and takes the focus away from what is essential, i.e. the cooperative task performed by the robots.

With proper simulation tools, roboticists can primarily focus on the specific challenges within robotic collaborative missions, run exhaustive tests in different scenarios and with

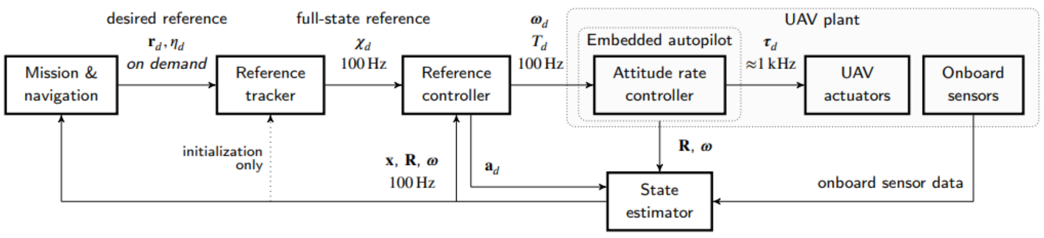


Figure 4: A diagram of the MRS UAV system architecture [6].

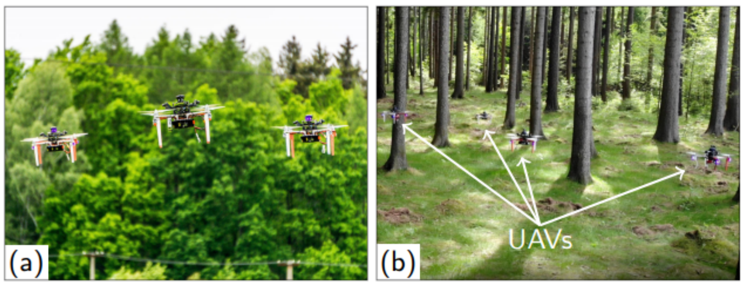


Figure 5: Swarms of multirotor UAVs testing novel flocking algorithms while localized (a) by a GNSS system, or (b) by onboard sensors only within a forest environment. Figure reproduced from [6].

different team sizes in a fairly realistic environment, and ultimately execute quicker experiments in the real world by mimicking the setting up of simulated experiments.

The scalability of multi-robot simulators has always been a known issue. Most existing 3D simulators normally fail in modern day computers to keep up the frame rate and the simulated time versus real time ratio with, even for relatively small teams, e.g. 3 or 4 mobile robots, having advanced navigation and perception capabilities. Clearly, in order to be able to simulate at least half a dozen robots under the above mentioned conditions, a balance between computation load and fidelity/realism is crucial.

Recently, the Multi-Robot Systems group at Czech Technical University (CTU) in Prague has proposed the MRS UAV simulation framework[1], which uses the Robot Operating System (ROS)[2] and the Gazebo simulation engine[3] at his back-end.

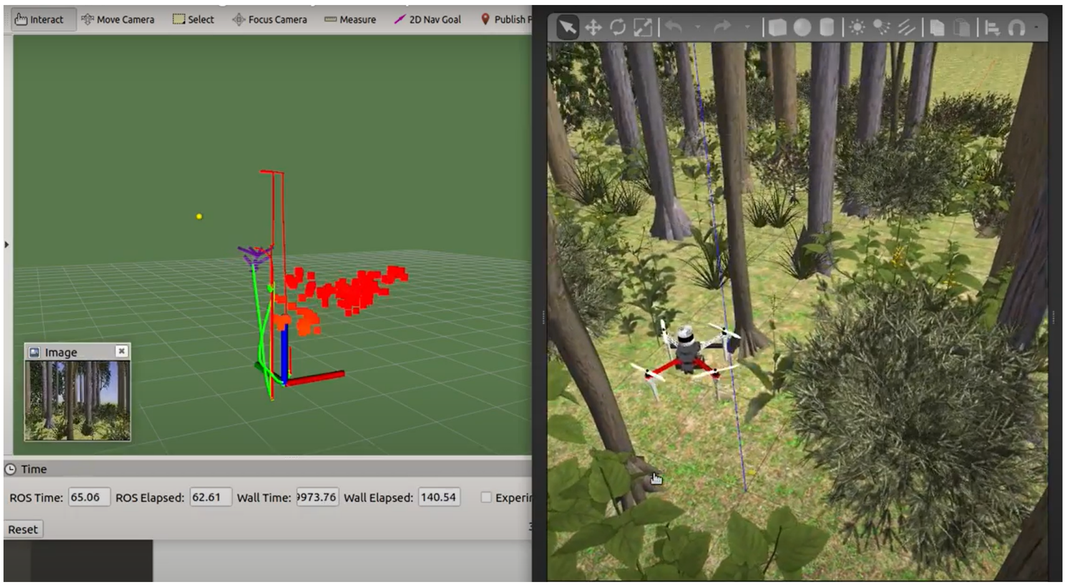
According to its authors, the MRS UAV [6] provides a multirotor Unmanned Aerial Vehicle (UAV) control and estimation system for supporting replicable research through realistic simulations, which can also be extended to real-world experiments. It follows a multi-frame localization paradigm for estimating the states of a UAV in various frames of reference using multiple sensors simultaneously. The system enables complex missions in GNSS and GNSS-denied environments, including outdoor-indoor transitions and the execution of redundant estimators for backing up unreliable localization sources. Two feedback control designs are presented (cf. Fig. 4): one for precise and aggressive maneuvers, and the other for stable and smooth flight with a noisy state estimate. The proposed control and estimation pipeline is constructed without using the Euler/TaitBryan angle representation of orientation in 3D. Instead, it relies on rotation matrices and a novel heading-based convention to represent the one free rotational degree-of-freedom in 3D of a standard multirotor helicopter.

MRS UAV provides an actively maintained and well-documented open-source imple-

[1]`https://ctu-mrs.github.io/`
[2]`https://wiki.ros.org/`
[3]`http://gazebosim.org/`

Table 1: Specifications for SEMFIRE UAV and respective sensor setup.

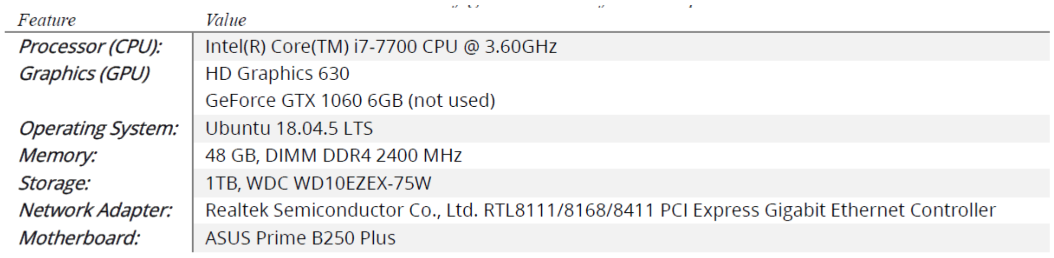
Feature	**Value**
Drone type	Drovni platform[2]
Camera unit	Intel Realsense camera d435i
CPU unit	Intel NUC
Flight controller	Pixhawk Cube 2.1
GPS unit	GPS/RTK
OS	Ubuntu 18.04



ROS Time: 65.06 ROS Elapsed: 62.61 Wall Time: 973.76 Wall Elapsed: 140.54

Figure 6: MRS UAV system in simulated scenario designed and implemented for the SEMFIRE project.

Table 2: PC configuration used for the experiments.



Feature	*Value*
Processor (CPU):	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
Graphics (GPU)	HD Graphics 630
	GeForce GTX 1060 6GB (not used)
Operating System:	Ubuntu 18.04.5 LTS
Memory:	48 GB, DIMM DDR4 2400 MHz
Storage:	1TB, WDC WD10EZEX-75W
Network Adapter:	Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller
Motherboard:	ASUS Prime B250 Plus

mentation, including realistic simulation of UAVs, sensors, and localization systems. The proposed system is the product of years of applied cutting-edge research on multi-robot systems, aerial swarms, aerial manipulation, motion planning, and remote sensing, and their use in various branches of autonomous robotics, including forestry (see Fig. 5). The system has been shaped by real-world system deployment, and utilized prestigious UAVs and robotics competitions and challenges. The proposed architecture allows reliable deployment of UAVs outside laboratory conditions using only onboard sensors.

Our preliminary study showed that the MRS UAV simulation framework provides the needed balance between computation load and fidelity/realism. Therefore, ongoing work within the SEMFIRE project has been focused in setting up the system for simulation of Scouts behavior to implement the initial deployment, following the strategy referred in Section 3, and additional behaviors such as regrouping, formation control, and more.

Besides the setup of simulation experiments as illustrated in Fig. 6, we have been working towards incorporating the Scouts multimodal sensors in the simulation environment. Namely, we have incorporated the Intel Realsense depth camera, and simulated the acquisition of aerial point clouds with the Scouts to be used for 3D mapping and path planning (see Table 1). The MRS-UAV tutorial can be found in the internal SEMFIRE technical report TR-SEMFIRE-2-vA [3].

4.2 Tests and parameter calibration for a single agent using MRS-UAV

4.2.1 Testing scenario definition

Prior to being able to use the MRS-UAV system to simulate coordinated operations with multiple UAVs, this system needed to be thoroughly tested and its parameters calibrated for the operation of a single UAV.

A testing scenario was defined using Gazebo in which the UAV navigated from point $A(0, -4, 4)$ to point $B(12, 4, 4)$ while avoiding two obstacles located on positions $O_1(4, 0, 0)$ and $O_2(8, 0, 0)$ (see Fig. 7).

All the experiments are performed on a single PC with the specification listed in Table 2.

4.2.2 Parameter list and evaluation for behaviour calibration

The parameters in Table 3 are going to be modified and evaluated separately. Each parameter has a default, minimum, and maximum value. For each experiment, 11 values are going to be generated and tested within the range of $[min - max]$ for each parameter (i.e. a test set of $+i \times \frac{max-min}{10}$, with $i = [0 - 10]$). The priority order for each parameter is also shown in Table 3.

The parameters with the order of 1 to 4 are related to Search tree generation. 5 to 9 are used in the cost function that is used for best-path selection. As it is observed in the

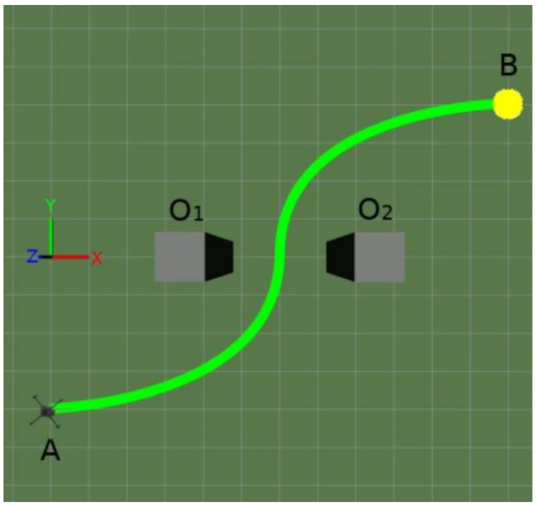


Figure 7: Proposed single UAV testing and calibration scenario for MRS-UAV in simulation. Gray squares : Obstacles (width=1m, depth=1m, height=10m); Green line : Desired path; Drone safety volume : width=0.696m , depth=0.696m , height=0.245m.

Table 3: Drone parameters for MRS-UAV.

Order	Name	Description	default	min	max
1	tree_node_distance	Distance between nodes	2	0.5	15
2	children_per_node	Branching factor of the search tree	8	1	100
3	n_expanded_nodes	Number of nodes expanded in complete tree	40	1	200
4	smoothing_margin_degrees	smoothing radius for obstacle cost in cost histogram	40	5	75
5	tree_heuristic_weight	Weight for the tree heuristic cost	35.0	0	50.0
6	obstacle_cost_param	Approximate distance from obstacles (m) when the obstacle distance term dominates the cost function	8.5	0.5	30.0
7	yaw_cost_param	Cost function weight for constant heading	3	1	20.0
8	pitch_cost_param	Cost function weight for goal-oriented behavior	25.0	5	30.0
9	velocity_cost_param	Cost function weight for path smoothness	6000	0	50000
10	max_point_age_s	maximum age of a remembered data point	20	0	100
11	min_num_points_per_cell	minimum number of points in one area to be kept if lower they are discarded as noise	1	1	100
12	smoothing_speed_xy	response speed of the smoothing system in xy (set to 0 to disable	10	0	30
13	smoothing_speed_z	response speed of the smoothing system in z (set to 0 to disable)	3	0	30
14	max_sensor_range	Data points farther away will be discarded	15.0	5	20

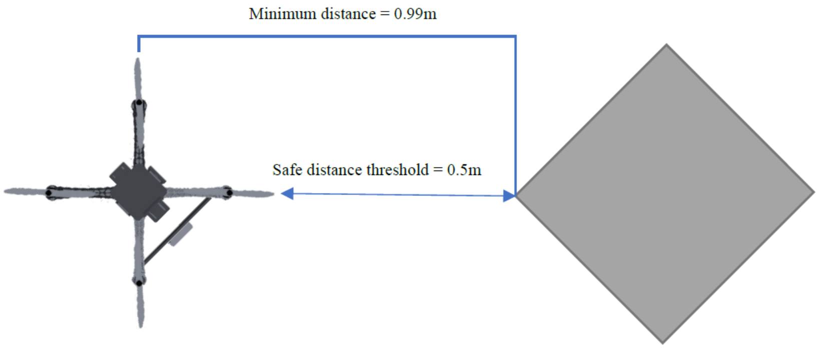


Figure 8: Safe distance and minimum distance threshold definition.

preliminary test, modifying these results in diverse output for the Search tree generation task. (i.e. result in wide/narrow search tree generation)

10 and 11 are related to local map generation. The parameter `max_point_age_s` indicates that obstacle points older than this number are deleted from the local map. The parameter `min_num_points_per_cell` discards points that are considered as noise in local map generation. So, changing these values would result in different Search tree generation.

12 and 13 are related to smooth trajectory generation. 14 is related to the maximum distance for depth point cloud data to be considered in local map generation. Modifying this parameter, as well as `tree_node_distance`, results in a deeper search tree and having a more reactive attitude for drone.

As an assumption for the experiments, the safety minimum distance to obstacle is considered as 0.5m which is used as a threshold to prevent collision with obstacles (Fig. 8). The following metrics will be used to evaluate parameter calibration quality:

1. Traversed path distance (meters). This metric is the overall traversed distance of the drone which starts at A, and ends at B. The measurement unit used for this metric is "meter". The drone will arrive at B while there is at least 5cm around B, in other words $||drone_{position} - B_{position}|| \leq 5cm$.

2. Flying time from A to B (seconds).

3. Minimum distance (meters). This metric is defined as the minimum distance between the drone position and obstacles (O_1, O_2) while navigating from A to B. According to the safety minimum distance threshold and according to the drone max extension size (diameter) which is 0.98m, the minimum distance to the obstacles should be $0.5 + 0.98/2 = 0.99$m (see Fig. 8). The minimum distance is calculating from the geometric centre of the robot with the corners of the obstacles. Therefore, the considered threshold should be more than 0.99m which is considered 1m for the experiments. Any parameter modification that results in lower than this threshold is not valid.

4. Tree calculation time (milliseconds). This metric is the average generation time of the search tree generation while traversing.

5. CPU load (percentage). This metric is the average local planner node CPU utilization while traversing.

6. RAM usage (MB). This metric is the average local planner node memory usage while traversing.

The results in each experiment are compared to the result of the baseline experiment (Experiment #1). In every experiment, the best possible values are highlighted (when relevant) in respective tables to be considered as the best value or considered as a new test set. A single highlighted row implies the best possible value for the respective parameter, and a set of highlighted rows implies a new test set for the respective parameter, and no highlighting rows implies that the last test set should be considered again.

4.2.3 Testing experiments

Experiment #1 –

In the baseline experiment, none of the parameters are modified to observe the performance of the planner with default values. Fig. 9 presents the overall traversed path from point A to point B for 10 iterations. Overall results for 10 iterations of default parameters value are presented in Table 4.

Experiment #2 –

In this experiment, the `tree_node_distance` parameter is modified. The default value of this parameter is 2. Fig. 10 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 5 and Fig. 11.

Experiment #3 –

In this experiment, the `children_per_node` parameter is modified. The default value of this parameter is 8. Fig. 12 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 6 and Fig. 13.

Experiment #4 –

In this experiment, the `n_expanded_nodes` parameter is modified. The default value of this parameter is 40. Fig. 14 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 7 and Fig. 15.

Experiment #5 –

In this experiment, the `smoothing_margin_degrees` parameter is modified. The default value of this parameter is 40. Fig. 16 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 8 and Fig. 17.

Experiment #6 –

In this experiment, the `tree_heuristic_weight` parameter is modified. The default value of this parameter is 35. Fig. 18 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 9 and Fig. 19.

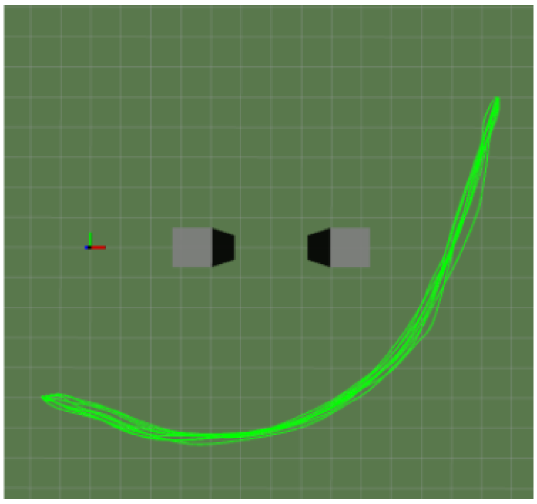


Figure 9: Traversed paths for the 10 trials of Experiment #1.

Table 4: Results for Experiment #1.

Test Value (m)	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
1	18.329	35.938	2.007	27.448	25.482	62.719
2	18.516	38.283	2.014	22.14	22.913	63.66
3	18.615	36.406	2.088	23.456	24.903	58.889
4	18.197	34.262	1.913	22.543	23.852	61.004
5	17.995	33.056	1.838	20.717	22.397	58.459
6	18.287	34.28	2.001	24.333	25.592	60.516
7	18.145	34.754	1.871	20.555	26.395	58.643
8	18.497	34.313	1.986	21.634	24.995	60.512
9	18.367	31.505	2.165	22.355	27.074	56.135
10	18.515	35.874	2.266	22.287	25.041	58.555

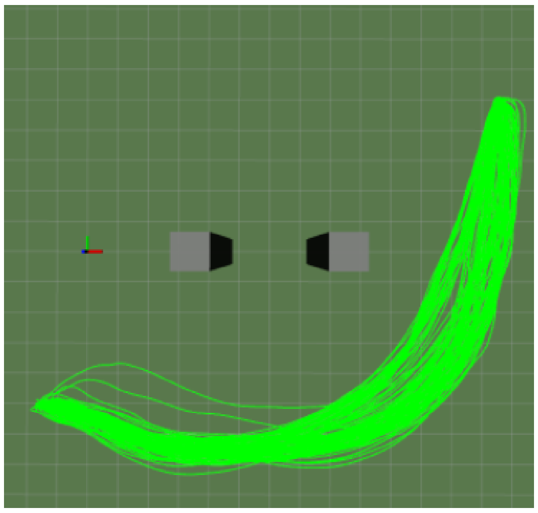


Figure 10: Traversed paths for Experiment #2.

Table 5: Results for Experiment #2.

Test Value (m)	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
0.5	18.465	34.322	1.91	13.659	22.825	62.422
1.95	18.109	33.538	2.039	21.15	24.401	61.494
3.4	18.497	34.856	2.179	21.235	23.869	62.584
4.85	19.138	36.079	2.783	20.426	24.02	62.805
6.3	19.387	35.861	2.882	22.906	24.433	60.188
7.75	19.641	36.539	2.874	20.108	23.646	62.794
9.2	20.319	36.241	3.141	8.544	20.331	62.986
10.6 5	20.19	37.052	3.391	6.729	20.124	61.148
12.1	20.274	37.535	3.335	6.654	19.711	61.248
13.5 5	20.303	36.749	3.335	5.792	19.808	61.6
15	19.849	35.036	3.194	3.548	20.261	60.515

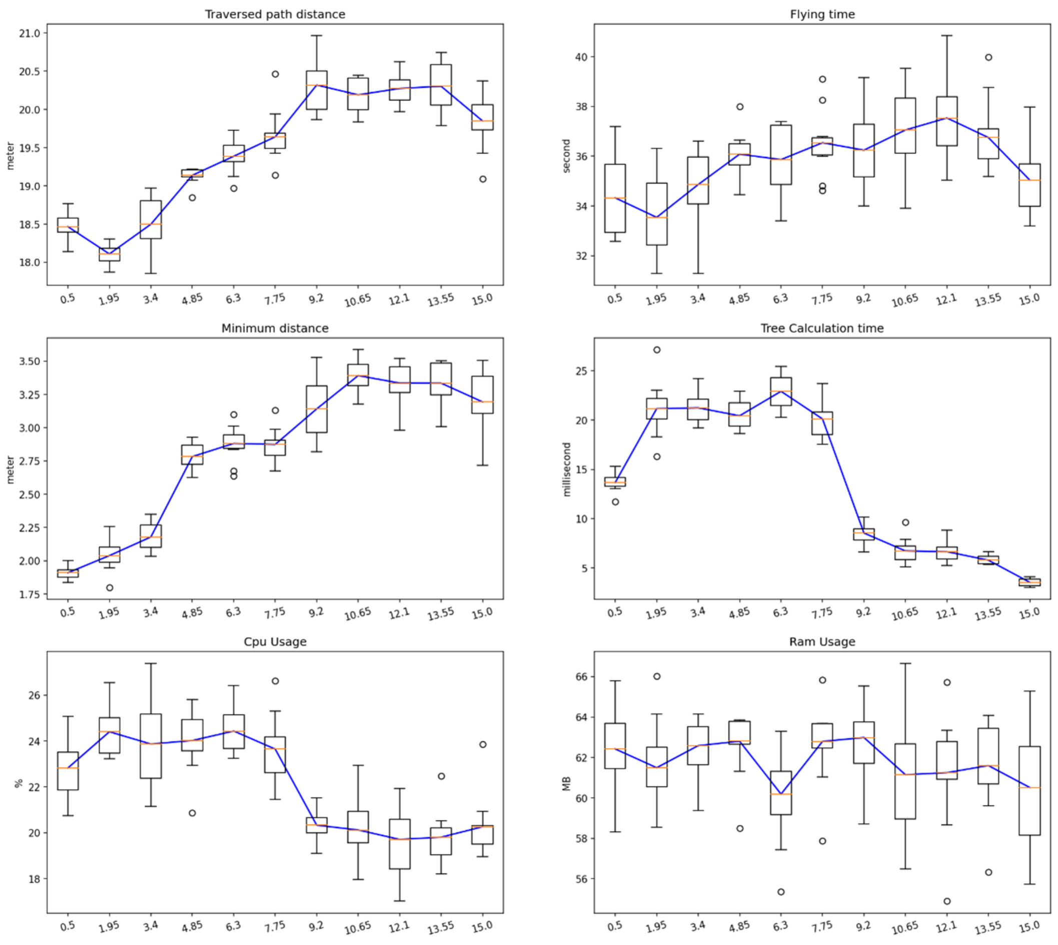


Figure 11: Result plots Experiment #2.

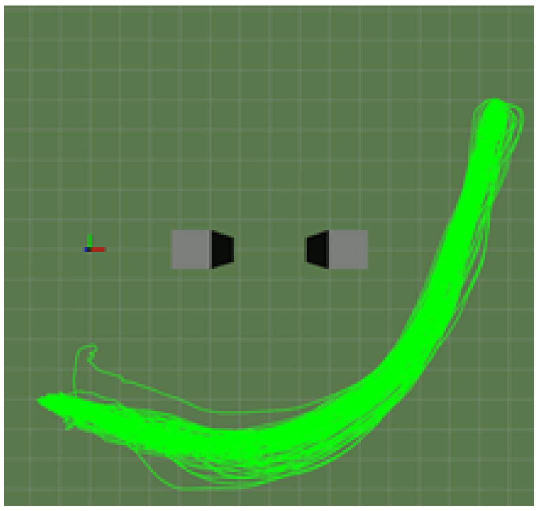


Figure 12: Traversed paths for Experiment #3.

Table 6: Results for Experiment #3.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
1	18.381	34.516	1.937	7.374	20.378	61.237
10.9	18.58	34.875	2.057	21.378	24.169	63.202
20.8	18.536	34.696	2.12	22.915	26.137	61.627
30.7	18.732	35.36	2.194	22.628	25.896	59.987
40.6	19.139	36.463	2.371	23.702	26.254	60.773
50.5	19.195	38.359	2.352	25.29	25.45	62.791
60.4	18.942	35.234	2.297	25.725	26.305	62.537
70.3	19.169	36.472	2.339	28.317	28.348	60.075
80.2	19.237	37.108	2.425	28.75	27.895	62.06
90.1	19.369	38.371	2.327	30.098	27.842	61.153
100	18.926	36.658	2.325	31.487	28.91	60.762

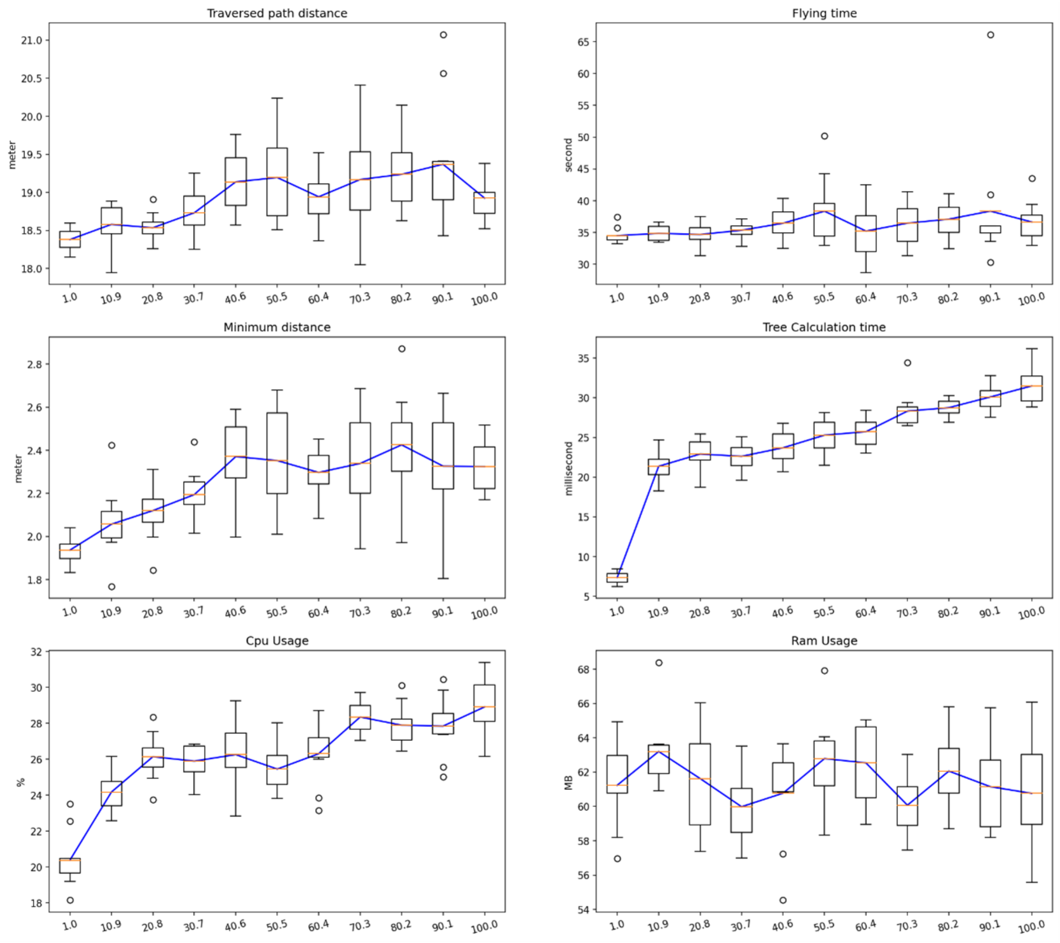


Figure 13: Result plots Experiment #3.

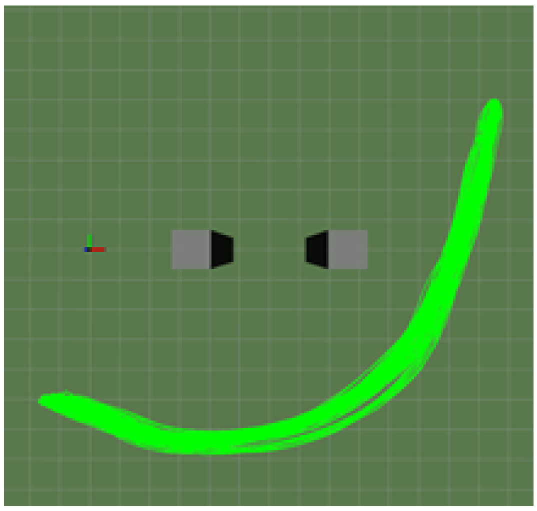


Figure 14: Traversed paths for Experiment #4.

Table 7: Results for Experiment #4.

Test Value	***Traversed path distance (m)***	***Flying time (s)***	***Minimum distance (m)***	***Calculation time (ms)***	***CPU load (%)***	***Ram usage (MB)***
1	19.159	35.461	2.486	0.862	18.453	63.213
20.9	18.352	34.287	2.002	11.996	21.265	62.028
40.8	18.147	33.656	2.024	21.386	25.114	63.468
60.7	18.156	33.656	1.959	29.867	28.36	60.946
80.6	18.43	34.316	2.144	37.143	30.511	61.947
100.5	18.285	33.988	2.165	43.655	33.003	62.272
120.4	18.307	33.228	2.241	50.751	36.476	61.15
140.3	18.455	33.393	2.278	54.12	37.917	61.99
160.2	18.471	33.228	2.33	58.748	39.637	62.554
180.1	18.513	34.71	2.308	66.98	42.908	62.5
200	18.422	34.832	2.239	70.592	43.333	60.109

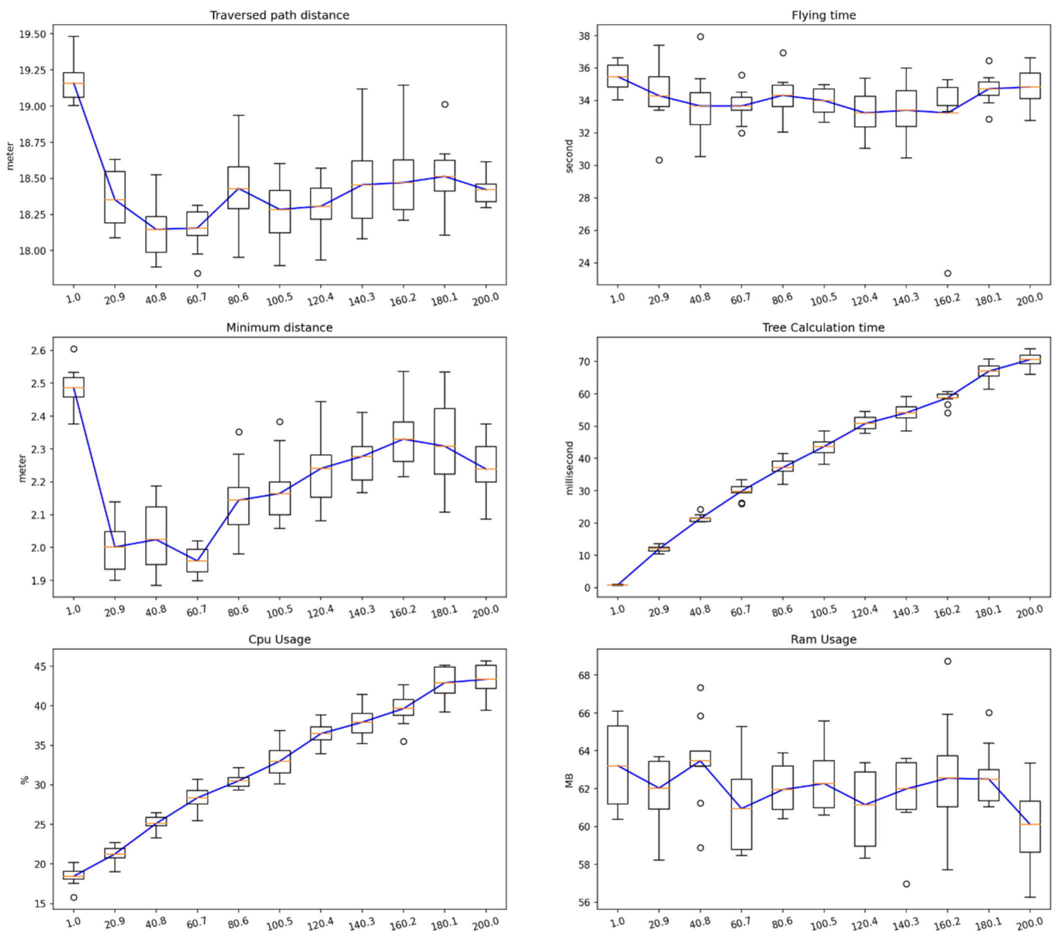


Figure 15: Result plots Experiment #4.

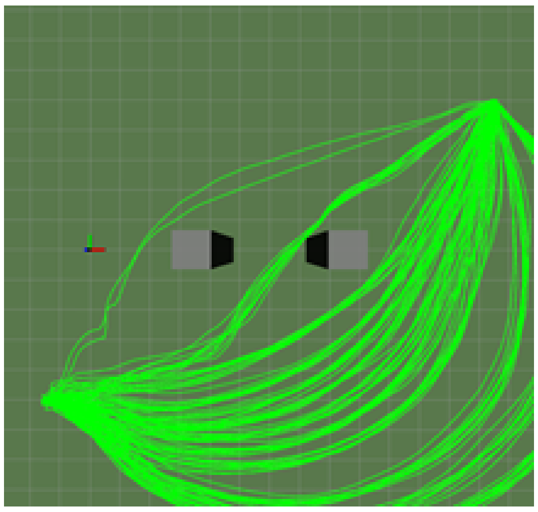


Figure 16: Traversed paths for Experiment #5.

Table 8: Results for Experiment #5.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
5	15.287	26.863	0.327	20.241	25.951	62.137
12	15.93	29.254	0.47	22.125	26.635	62.825
19	16.461	31.086	0.9	20.85	26.216	62.206
26	17.181	31.275	1.305	21.249	25.385	63.039
33	17.436	32.779	1.498	21.245	24.913	61.967
40	18.221	34.044	2.058	21.195	25.549	60.805
47	18.961	34.059	2.467	20.809	24.836	62.235
54	19.89	35.625	2.976	21.052	24.563	61.792
61	22.52	39.825	4.313	21.768	24.019	61.558
68	24.693	44.392	4.758	20.103	23.237	62.25
75	26.801	45.862	4.829	22.33	22.586	61.749

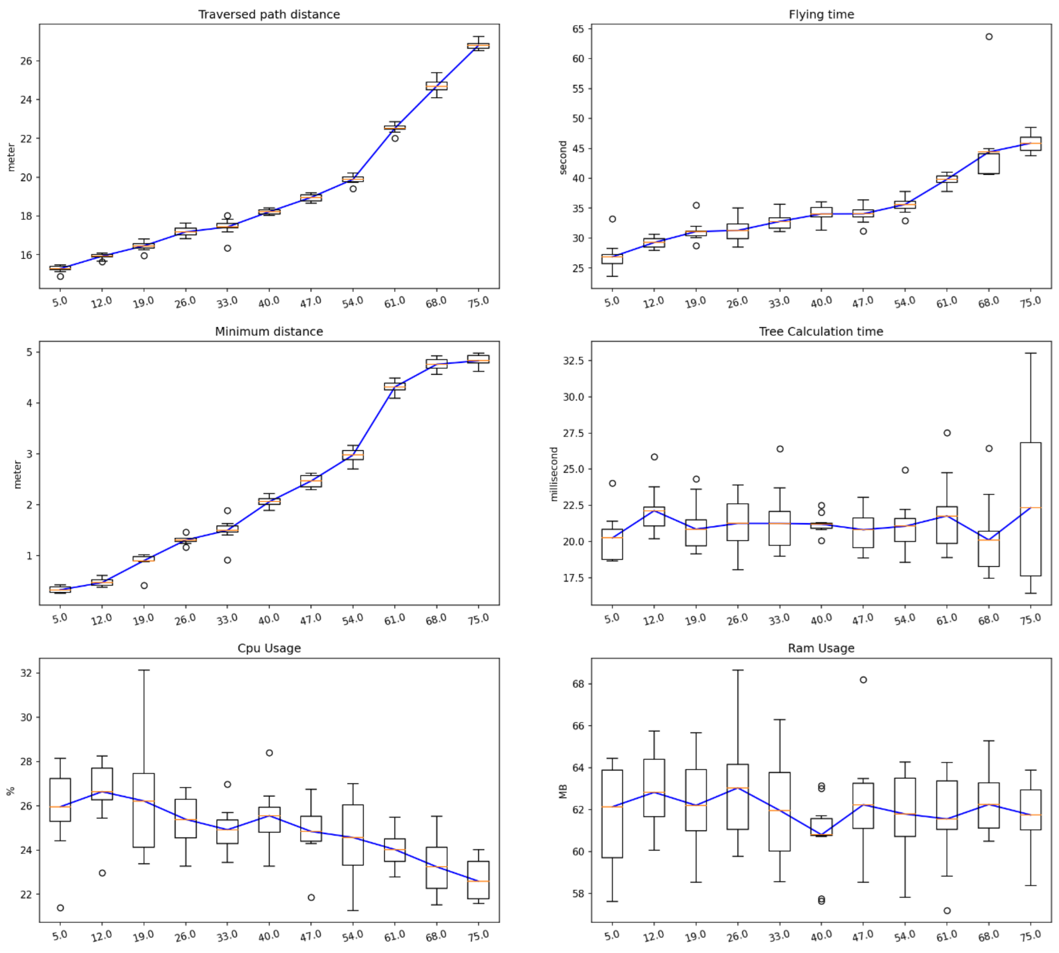


Figure 17: Result plots Experiment #5.

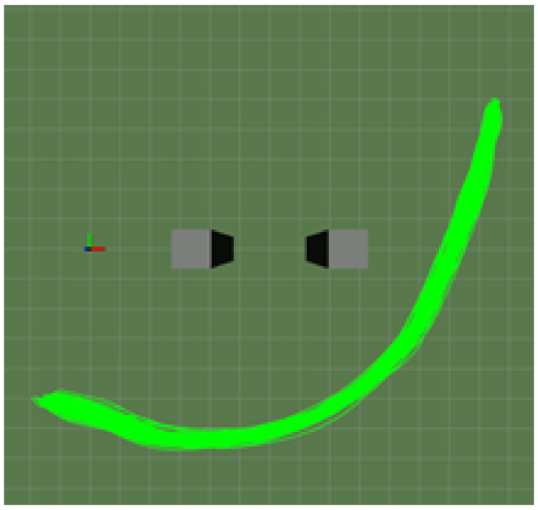


Figure 18: Traversed paths for Experiment #6.

Table 9: Results for Experiment #6.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
0	18.328	34.366	1.993	22.299		
5	18.239	34.132	1.962	21.615		
10	18.389	34.899	2.013	22.157		
15	18.449	35.496	2.053	22.353		
20	18.257	34.747	2.045	22.48		
25	18.39	35.154	1.993	21.571		
30	18.322	34.552	1.999	22.1		
35	18.366	37.635	2.006	21.969		
40	18.276	33.072	2.04	22.276		
45	18.205	33.168	2.059	21.965		
50	18.156	34.59	1.982	21.572		

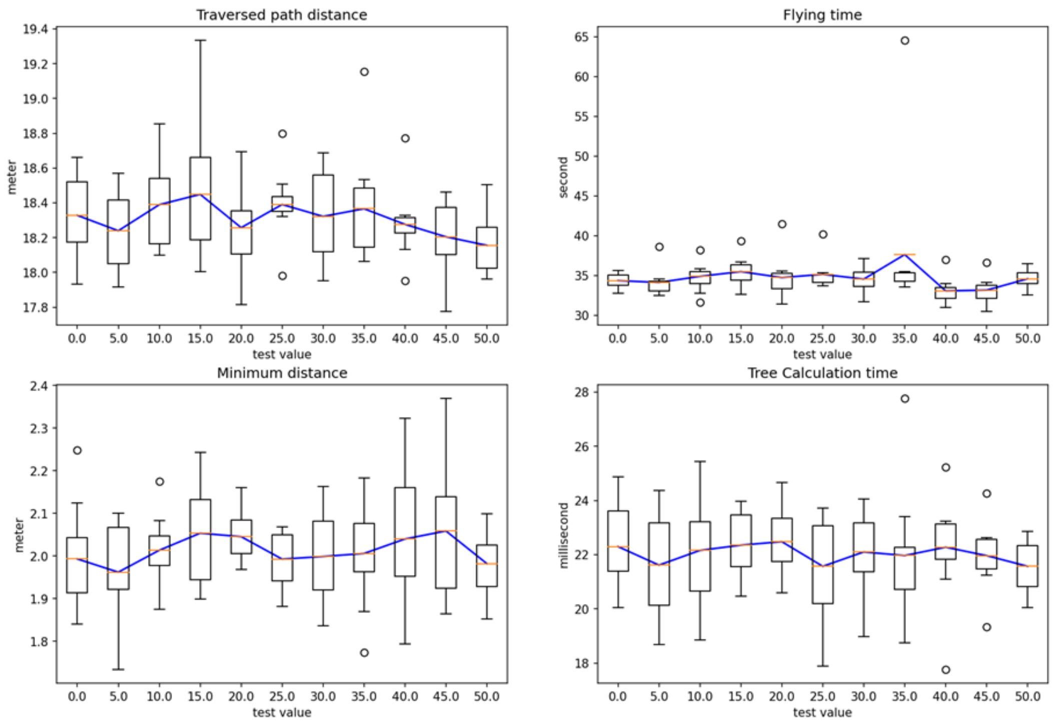


Figure 19: Result plots Experiment #6.

Experiment #7 –

In this experiment, the `obstacle_cost_param` parameter is modified. The default value of this parameter is 8.5. Fig. 20 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 10 and Fig. 21.

Experiment #8 –

In this experiment, the `yaw_cost_param` parameter is modified. The default value of this parameter is 3. Fig. 22 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 11 and Fig. 23.

Experiment #9 –

In this experiment, the `pitch_cost_param` parameter is modified. The default value of this parameter is 25. Fig. 24 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 12 and Fig. 25.

Experiment #10 –

In this experiment, the `velocity_cost_param` parameter is modified. The default value of this parameter is 6000. Fig. 26 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 13 and Fig. 27.

Experiment #11 –

In this experiment, the `max_point_age_s` parameter is modified. The default value of this parameter is 20. Fig. 28 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 14 and Fig. 29.

Experiment #12 –

In this experiment, the `min_num_points_per_cell` parameter is modified. The default value of this parameter is 1. Fig. 30 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 15 and Fig. 31.

Experiment #13 –

In this experiment, the `smoothing_speed_xy` parameter is modified. The default value of this parameter is 10. Fig. 32 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 16 and Fig. 33.

Experiment #14 –

In this experiment, the `smoothing_speed_z` parameter is modified. The default value of this parameter is 3. Fig. 34 presents the overall traversed path from point A to point B

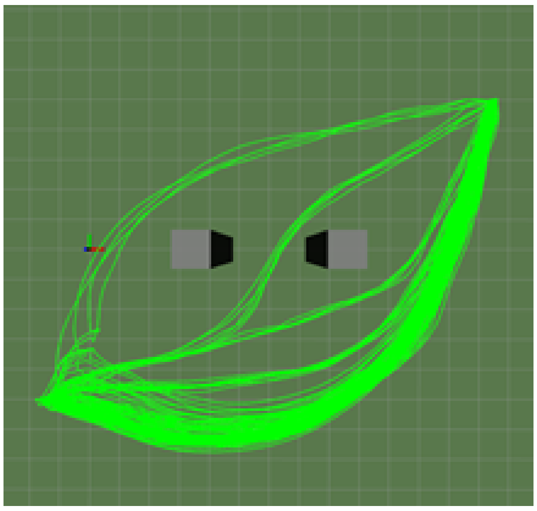


Figure 20: Traversed paths for Experiment #7.

Table 10: Results for Experiment #7.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
0.5	15.505	26.067	0.699	22.878		
3.45	16.674	29.24	1.461	22.585		
6.4	17.72	32.006	1.864	21.812		
9.35	18.338	35.395	1.991	22.37		
12.3	18.129	35.9	1.798	20.684		
15.25	18.185	35.1	2.003	21.685		
18.2	18.213	36.407	1.859	21.888		
21.15	18.335	35.096	2.094	21.243		
24.1	18.1	35.366	2.013	21.798		
27.05	18.075	33.262	1.952	22.069		
30	18.203	35.39	1.954	21.621		

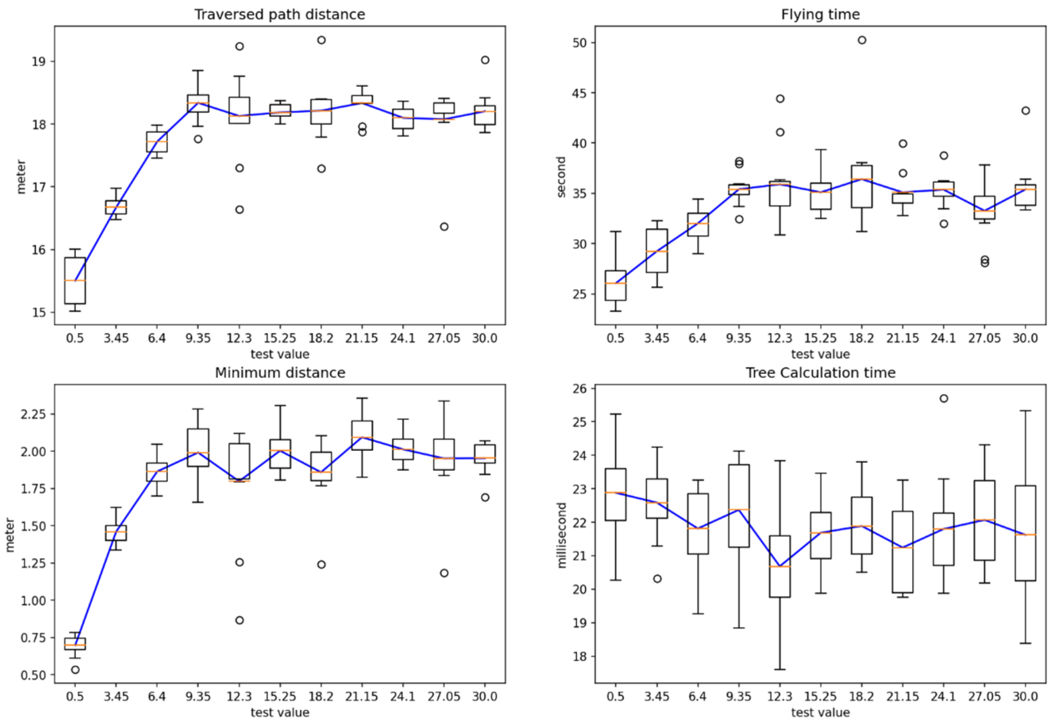


Figure 21: Result plots Experiment #7.

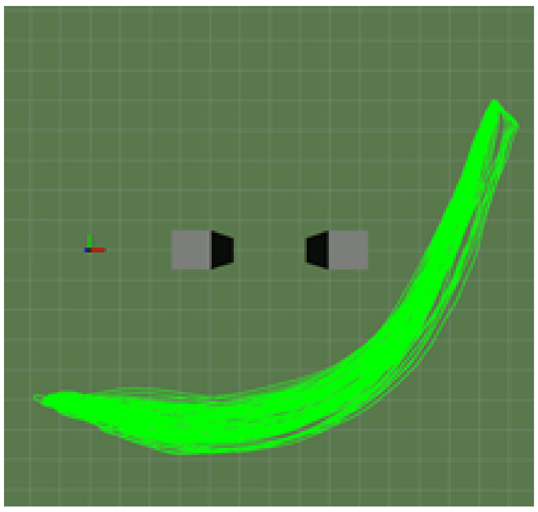


Figure 22: Traversed paths for Experiment #8.

Table 11: Results for Experiment #8.

Test Value (deg)	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
1	18.946	34.105	2.329	22.542	24.943	61.691
2.9	18.143	33.923	2.04	22.126	26.331	60.442
4.8	18.091	33.492	1.959	21.563	25.269	63.03
6.7	17.891	32.524	1.797	21.895	24.019	61.858
8.6	17.983	33.117	1.886	22.043	25.978	62.134
10.5	17.713	32.578	1.813	23.322	25.553	60.621
12.4	17.646	31.963	1.689	22.094	25.336	61.982
14.3	17.511	31.835	1.652	21.502	24.907	63.106
16.2	17.522	31.267	1.647	22.027	25.234	62.218
18.1	17.501	31.587	1.578	22.021	24.939	60.781
20	17.458	31.245	1.544	22.232	25.853	62.712

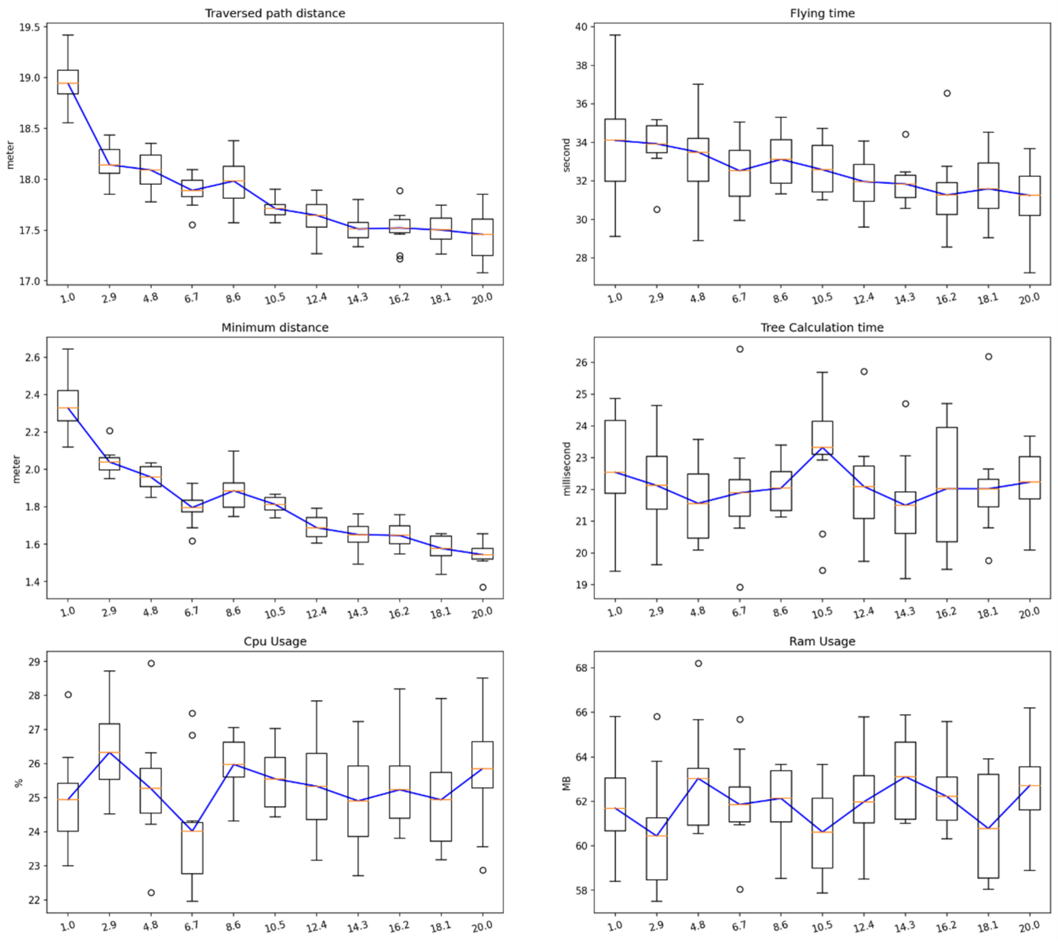


Figure 23: Result plots Experiment #8.

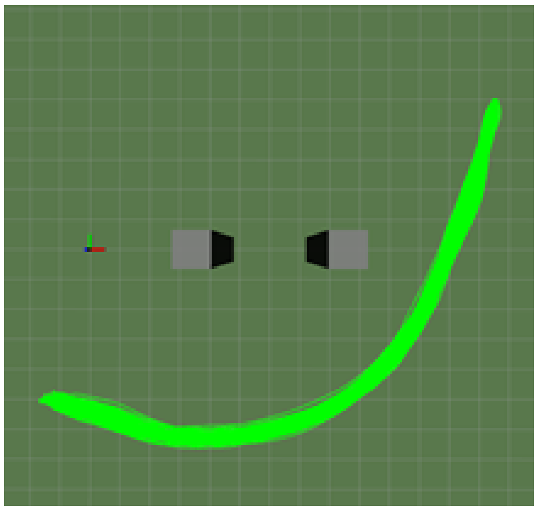


Figure 24: Traversed paths for Experiment #9.

Table 12: Results for Experiment #9.

Test Value (deg)	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
5	18.15	33.886	1.888	21.262	24.934	61.832
7.5	18.122	33.335	1.95	21.197	25.616	61.913
10	18.128	33.309	1.937	21.556	24.876	61.721
12.5	18.287	34.646	1.98	21.218	25.055	62.482
15	18.156	33.111	1.996	21.608	24.824	62.724
17.5	18.232	33.824	2.026	22.378	25.761	61.22
20	18.173	33.522	1.997	22.63	25.082	60.678
22.5	18.312	33.855	1.994	22.269	25.856	63.095
25	18.303	34.546	1.987	22.75	25.868	60.885
27.5	18.098	32.959	1.99	21.945	24.746	60.394
30	18.191	33.203	2.039	21.988	25.435	61.578

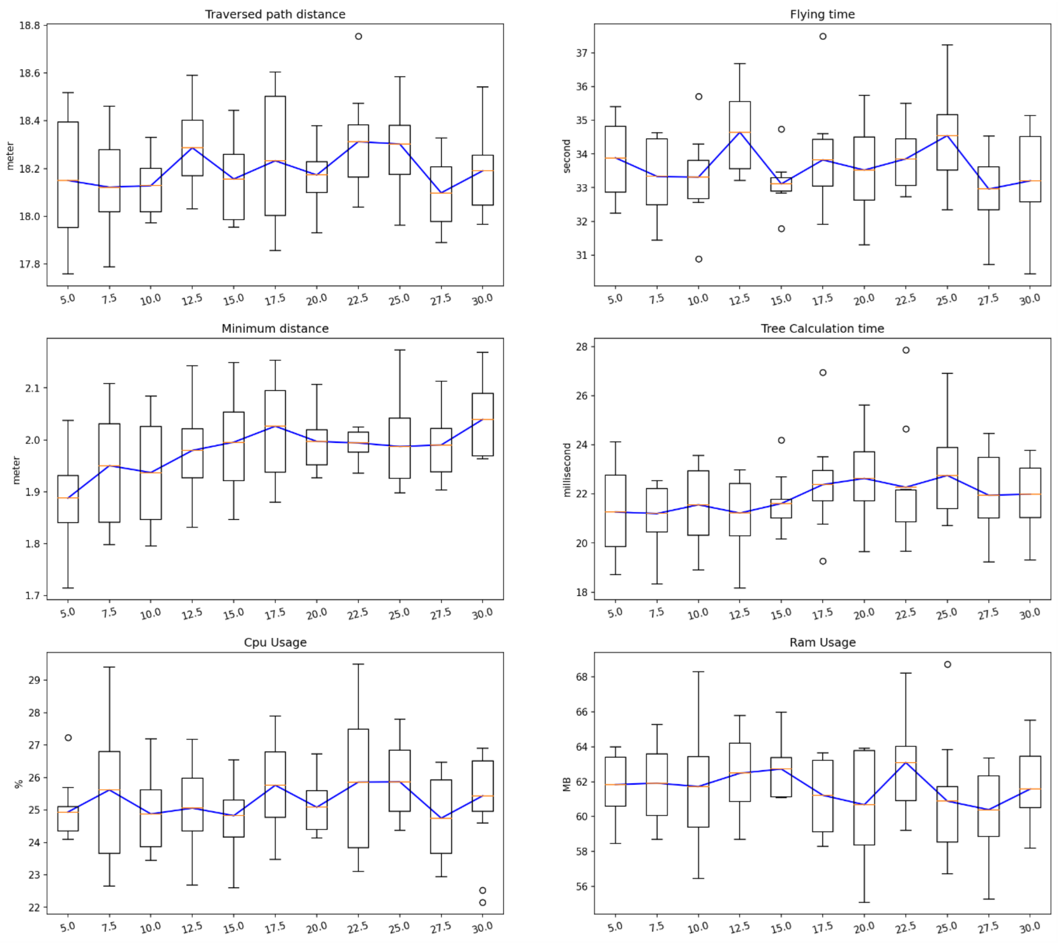


Figure 25: Result plots Experiment #9.

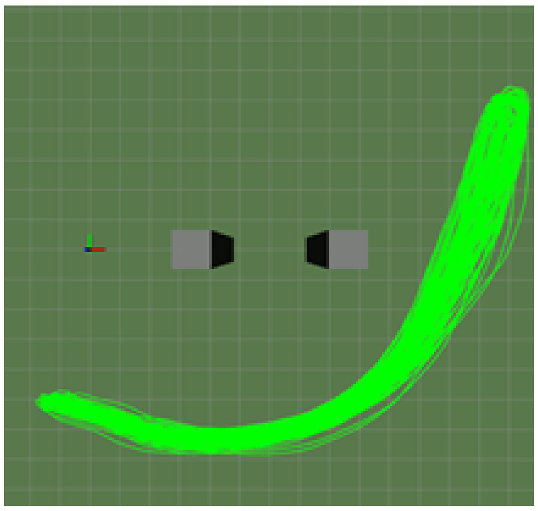


Figure 26: Traversed paths for Experiment #10.

Table 13: Results for Experiment #10.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
0	18.016	34.226	1.825	20.52	24.956	61.524
5000	18.143	33.517	1.969	21.467	24.557	61.864
10000	18.29	32.848	2.048	21.745	25.178	61.26
15000	18.368	34.348	2.025	22.309	24.887	60.986
20000	18.437	34.417	2.118	21.658	24.662	61.62
25000	19.285	33.064	2.251	20.826	24.536	62.978
30000	19.481	34.855	2.455	21.569	24.733	61.061
35000	19.408	35.403	2.379	21.21	25.089	61.682
40000	19.477	33.561	2.467	21.966	24.058	61.779
45000	19.275	35.592	2.381	21.453	25.167	61.22
50000	20.02	34.057	2.736	21.941	25.259	59.785

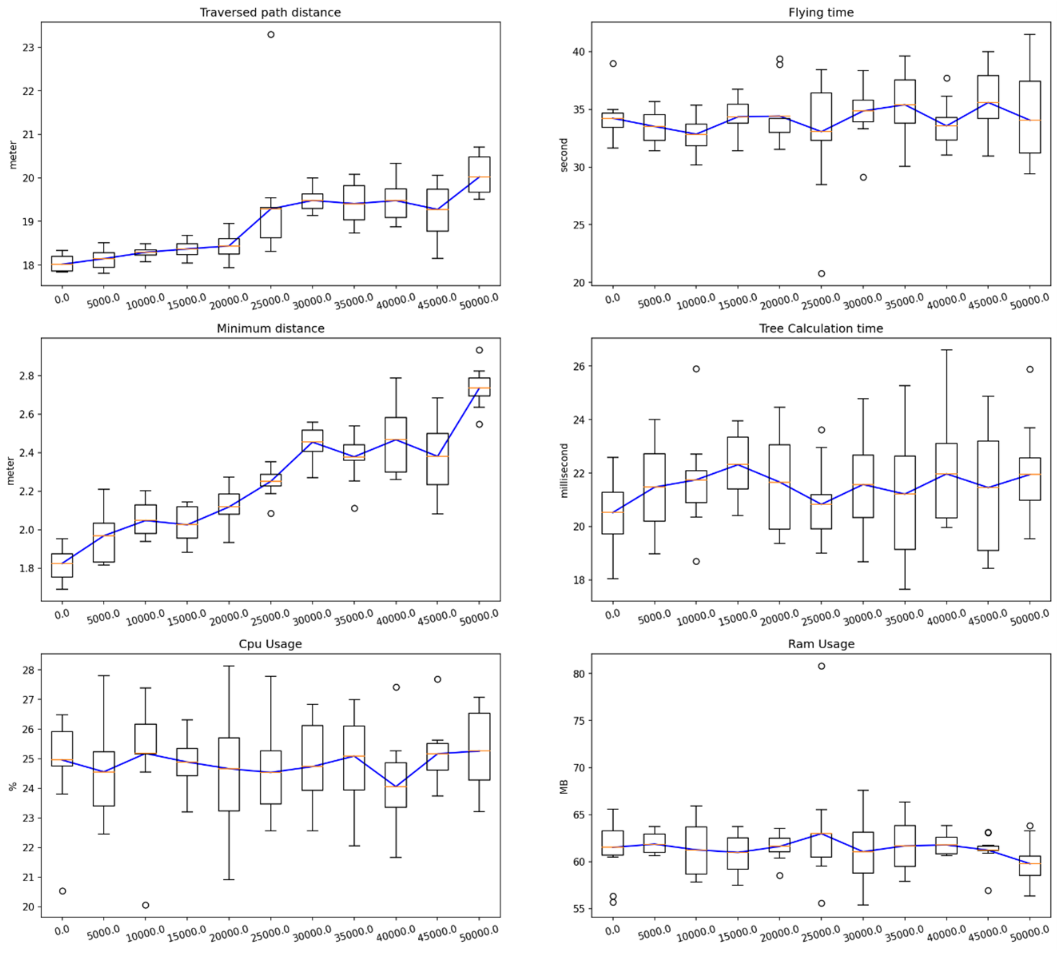


Figure 27: Result plots Experiment #10.

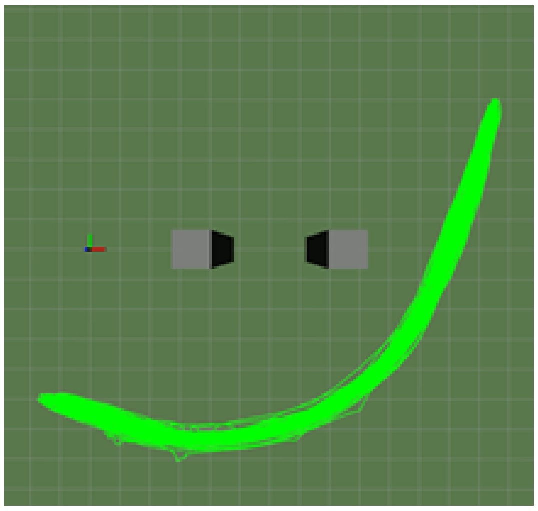


Figure 28: Traversed paths for Experiment #11.

Table 14: Results for Experiment #11.

Test Value (s)	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
0	19.065	49.409	1.811	20.686	22.714	61.77
10	18.101	34.908	1.941	24.731	23.599	60.694
20	18.134	33.453	1.977	22.278	25.794	60.876
30	18.433	34.441	1.985	21.984	24.917	62.629
40	18.331	34.217	2.016	22.123	24.96	61.103
50	18.088	33.474	2.025	22.534	24.52	61.394
60	18.109	33.923	1.982	21.361	24.955	62.368
70	18.382	34.768	2.004	21.487	24.638	61.593
80	18.195	33.749	1.982	22.737	25.321	61.926
90	18.165	34.105	2.023	22.502	25.533	60.271
100	18.216	34.908	1.933	20.864	23.877	62.287

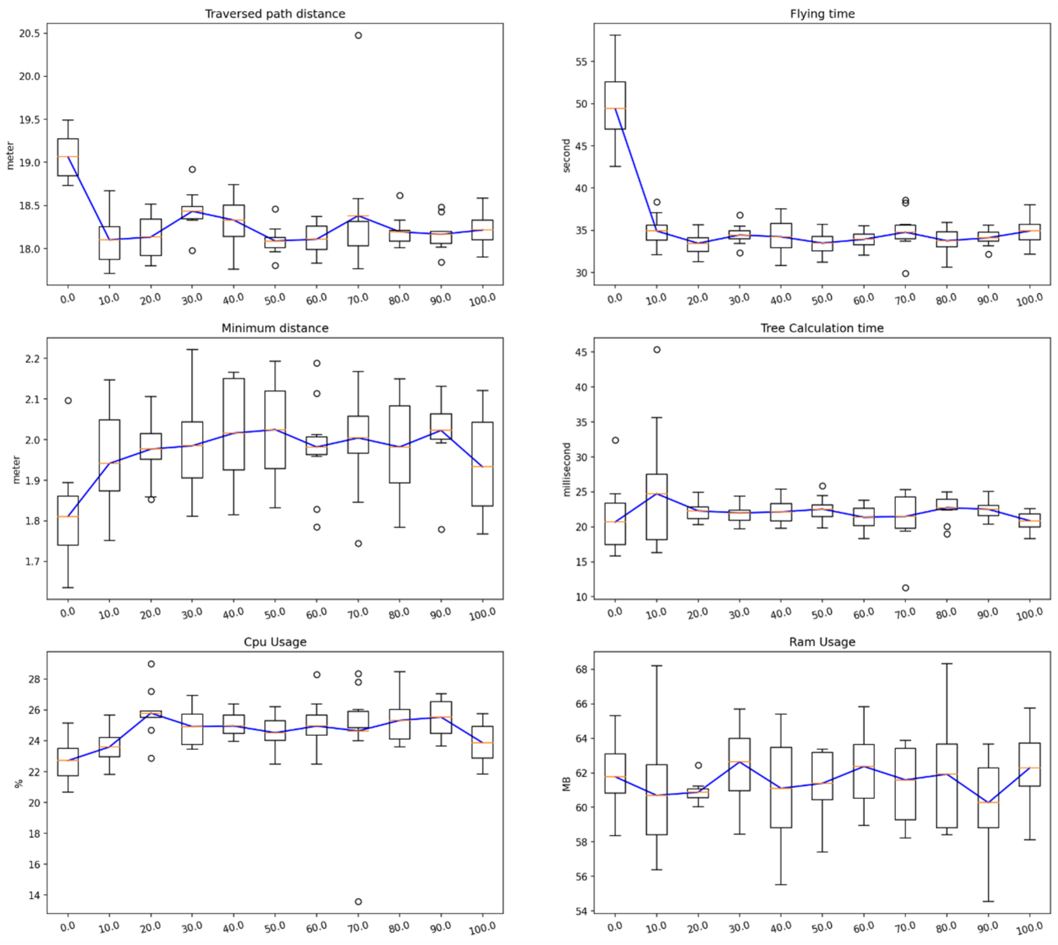


Figure 29: Result plots Experiment #11.

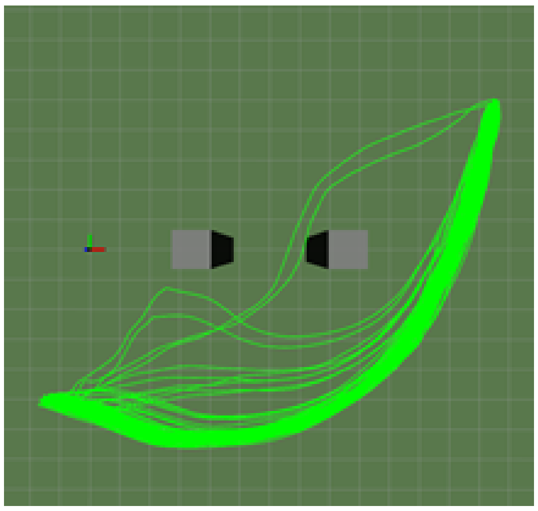


Figure 30: Traversed paths for Experiment #12.

Table 15: Results for Experiment #12.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
1	18.179	33.791	2.008	21.424	25.423	61.83
10.9	18.153	33.877	2.08	21.929	24.655	60.944
20.8	18.2	33.575	2.048	21.65	25.534	61.981
30.7	18.261	34.391	2.108	21.599	25.182	61.621
40.6	18.261	34.074	2.063	23.311	26.502	61.205
50.5	18.243	34.945	2.079	21.79	25.176	61.816
60.4	18.225	34.232	2.051	22.201	25.334	61.049
70.3	18.251	34.453	2.003	20.76	25.225	59.958
80.2	18.18	33.697	1.992	21.352	24.522	62.485
90.1	17.918	32.58	1.976	21.7	25.203	62.972
100	16.904	29.488	1.324	20.94	25.8	64.126

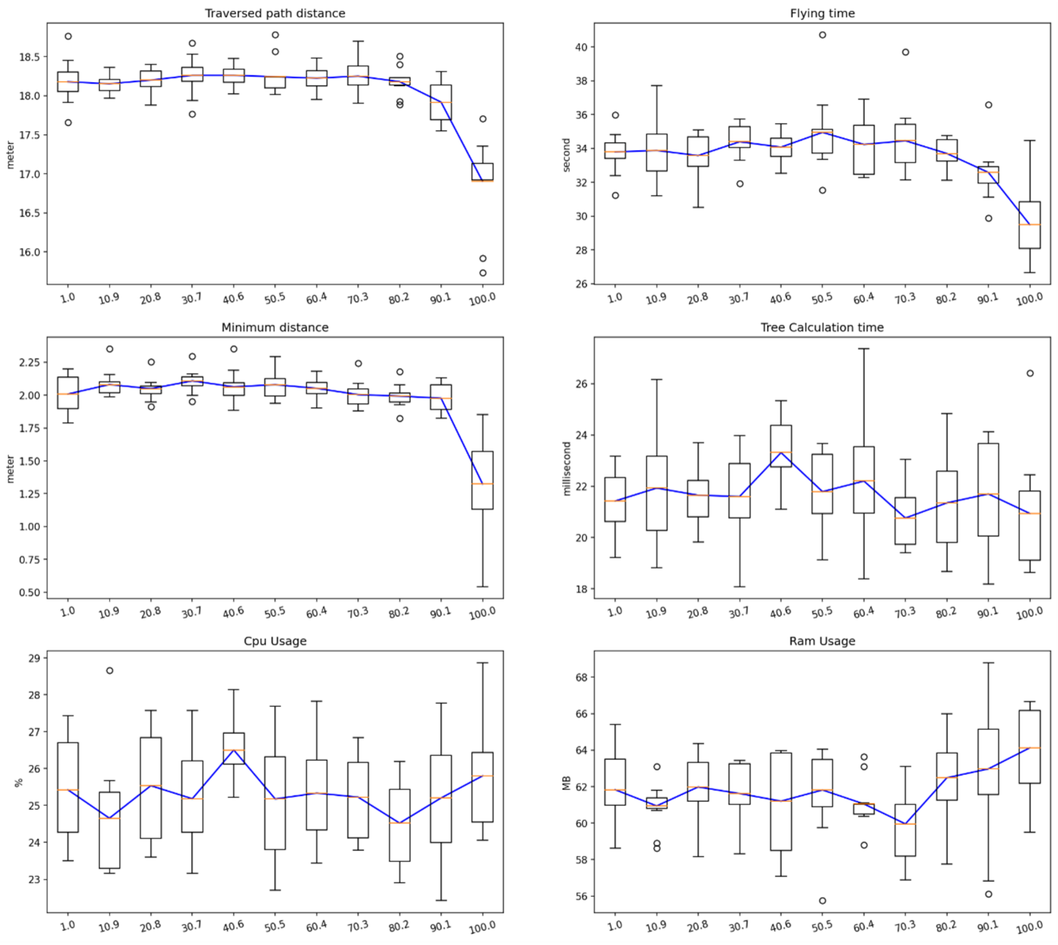


Figure 31: Result plots Experiment #12.

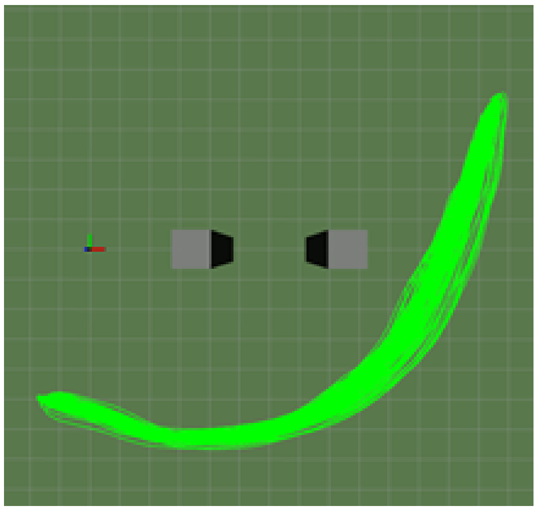


Figure 32: Traversed paths for Experiment #13.

Table 16: Results for Experiment #13.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
0	18.337	32.891	1.578	20.913	24.135	61.894
3	19.001	36.124	2.553	22.025	24.204	61.413
6	18.376	35.398	2.26	23.031	26.247	61.11
9	18.272	34.763	2	21.642	25.132	62.711
12	18.211	34.093	1.973	21.763	24.625	62.066
15	18.127	33.665	1.863	22.06	25.504	62.616
18	18.116	33.449	1.881	21.732	25.117	60.756
21	17.98	32.459	1.813	21.544	25.431	62.739
24	18.012	33.554	1.827	22.093	25.48	61.525
27	18.052	34.719	1.779	21.628	24.956	62.091
30	17.995	33.63	1.744	21.505	25.73	61.908

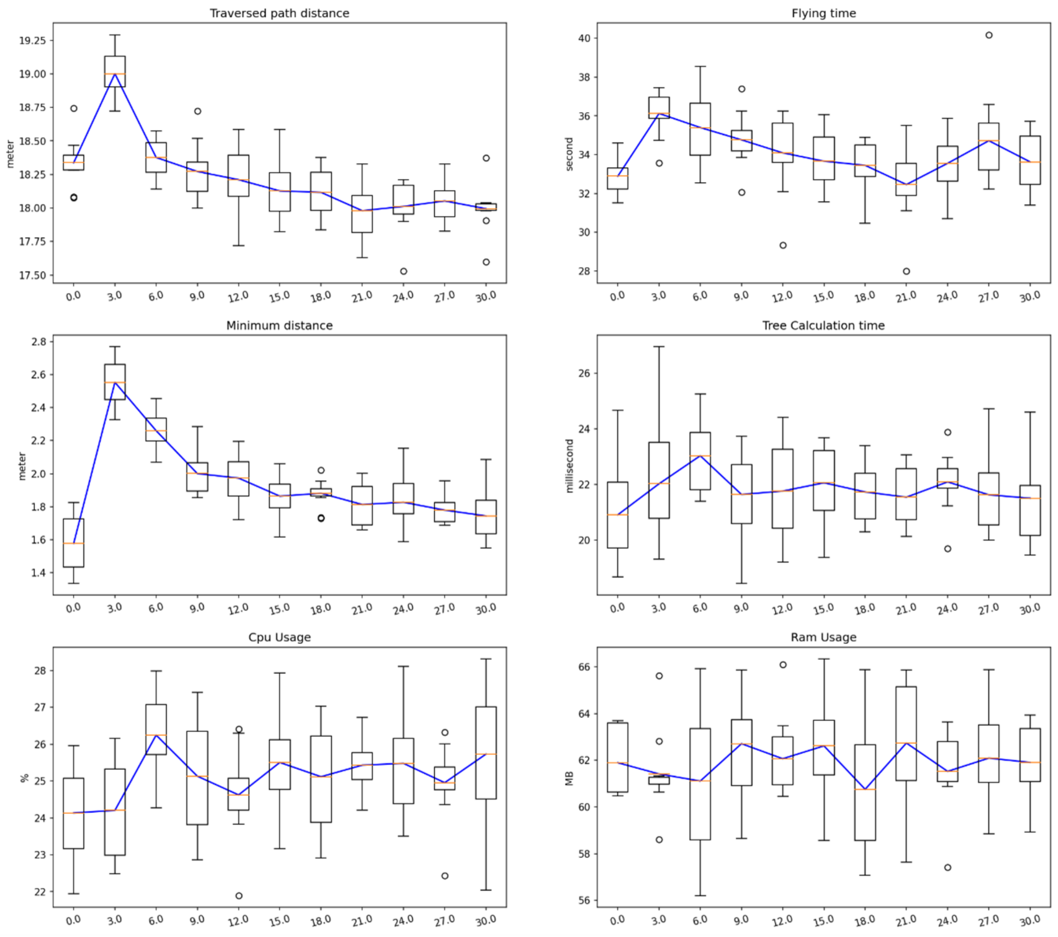


Figure 33: Result plots Experiment #13.

for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 17 and Fig. 35.

Experiment #15 –

In this experiment, the `max_sensor_range` parameter is modified. The default value of this parameter is 3. Fig. 36 presents the overall traversed path from point A to point B for 10 iterations per 11 values. Overall results for the 11 parameter values are presented in Table 18 and Fig. 37.

4.2.4 Final parameter combination experiments and parameter set choice

According to the results of the follow-up experiments on parameter combinations, in line with our safety and performance requirements, the final choice for the operational parameter set is presented in Table 19.

4.3 Formation control strategies – implementation using MRS-UAV

4.3.1 Introduction

In most applications, it is expected autonomous aerial swarms be more capable than a single-vehicle due to offering significantly enhanced adaptability, scalability, and maintainability, reliability, survivability, and fault-tolerance [11]. In this section, advances in aerial swarm robotics are briefly reviewed. Aerial robots swarming is supposed to autonomously operate in a complex 3D world including dynamic obstacles that is getting increasingly crowded with drones and other locomotives. The success of aerial swarms flying in such an environment is predicated on the distributed and synergistic capabilities of individual controlling as well as collective motions of UAVs with limited resources for on-board computation power. In this application, three factors should be considered to use as an aerial swarm solution. i) obstacle avoidance, ii) collision avoidance, and iii) formation control. A swarm generally refers to a group of similar agents that exhibits emergent behaviour arising from local interactions among the agents. Local interaction can be competitive or cooperative, which typically requires a large group of agents (10 to 100 or more). A formation consists mostly of cooperative interactions, and the relationship between the states of the agents is a well-defined form for goals (e.g., triangle, square, circle, hexagon, etc.) [11].

Authors in [51] have classified formation control problems as follows:

- Formation producing problems: The agents' objective is to achieve a predefined desired formation shape. In the literature, these problems have been addressed through matrix theory-based approach, Lyapunov based approach, graph rigidity approach, and receding horizon.
- Formation tracking problems: Agents' reference trajectories are defined, and they are supposed to be tracked by the agents. These problems have been studied through matrix theory-based approach, potential function-based approach, Lyapunov based approach, and some other approaches.

According to fundamental control ideas in [10, 55], formation control schemes are generally classified into three categories as follows.

- Leader-follower approach: At least one agent plays the role of a leader, and the rest of the agents are designated as followers. The followers track the position of the leader with some prescribed offsets while the leader tracks its desired trajectory. The main challenges in this approach are the number of leaders to be chosen and the nomination criteria for leaders.

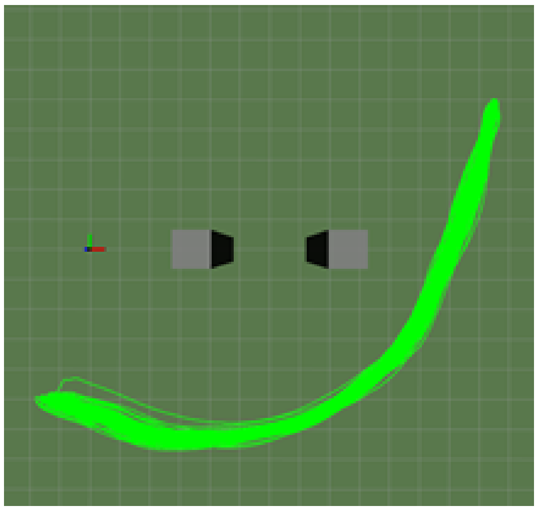


Figure 34: Traversed paths for Experiment #14.

Table 17: Results for Experiment #14.

Test Value	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
1	18.22	34.24	2.014	23.492	25.837	61.891
3.9	18.188	33.547	1.989	22.32	26.097	62.197
6.8	18.216	33.481	2.058	21.356	24.878	61.551
9.7	18.327	34.494	2.018	21.263	25.042	62.319
12.6	18.247	33.196	2.018	21.718	24.787	60.309
15.5	18.16	33.499	1.973	21.379	25.137	61.273
18.4	18.201	33.766	1.977	22.317	25.321	61.089
21.3	18.398	34.064	1.994	22.616	25.542	61.496
24.2	18.269	33.834	2.025	21.622	25.259	62.249
27.1	18.246	34.922	2.02	21.844	24.115	60.857
30	18.18	33.275	2.019	21.907	25.709	61.314

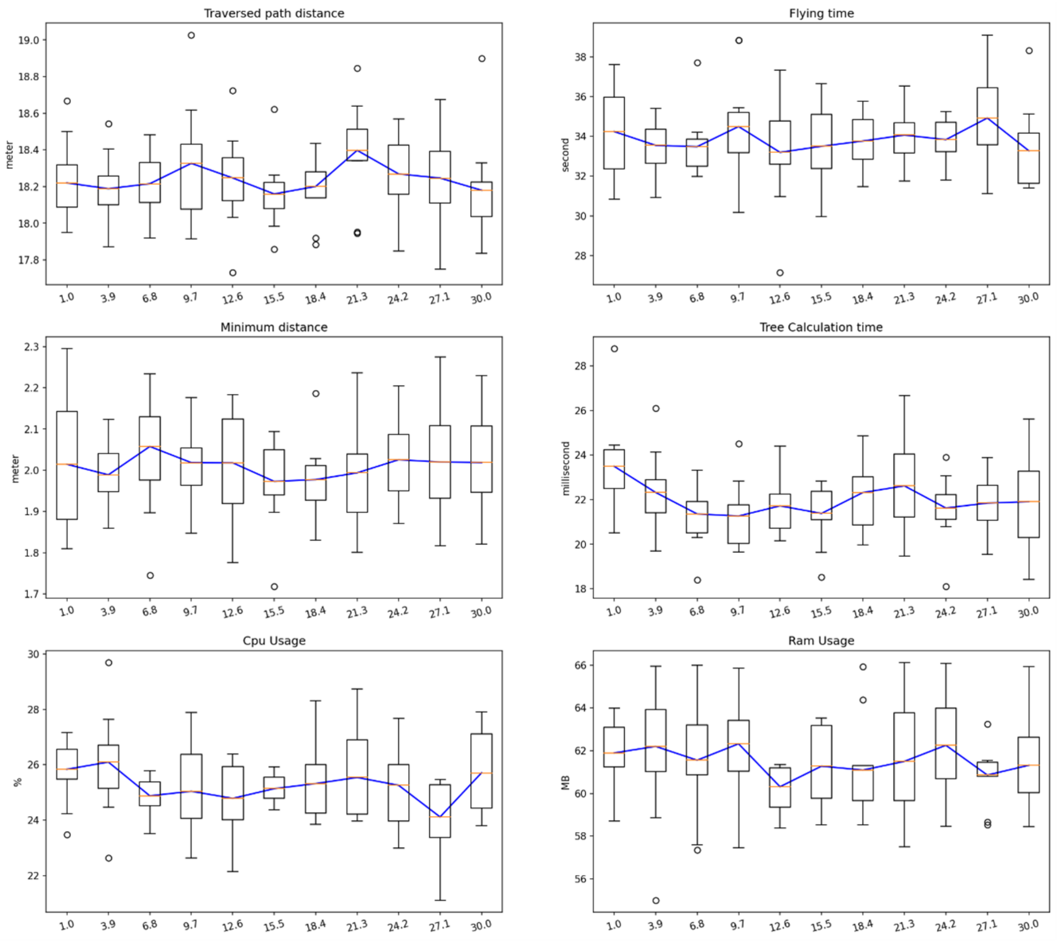


Figure 35: Result plots Experiment #14.

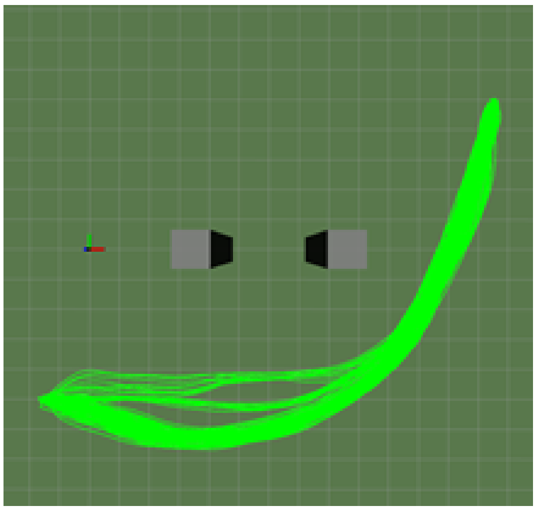


Figure 36: Traversed paths for Experiment #15.

Table 18: Results for Experiment #15.

Test Value (m)	*Traversed path distance (m)*	*Flying time (s)*	*Minimum distance (m)*	*Tree Calculation time (ms)*	*CPU load (%)*	*Ram usage (MB)*
5	17.482	30.526	2.03	17.512	22.774	61.597
6.5	17.252	29.472	1.765	20.973	24.835	61.434
8	17.831	32.758	1.882	21.053	23.885	62.327
9.5	18.291	34.091	2.026	21.57	25.25	61.866
11	18.32	34.084	2.042	23.058	26.466	61.154
12.5	18.198	33.771	2.017	21.524	24.541	62.497
14	18.283	33.718	2.061	22.115	24.992	61.47
15.5	18.502	37.157	2.105	24.326	24.886	61.514
17	18.155	32.89	1.989	21.127	25.19	62.182
18.5	18.11	33.836	1.974	21.442	25.788	62.342
20	18.174	33.363	1.916	21.885	25.896	61.304

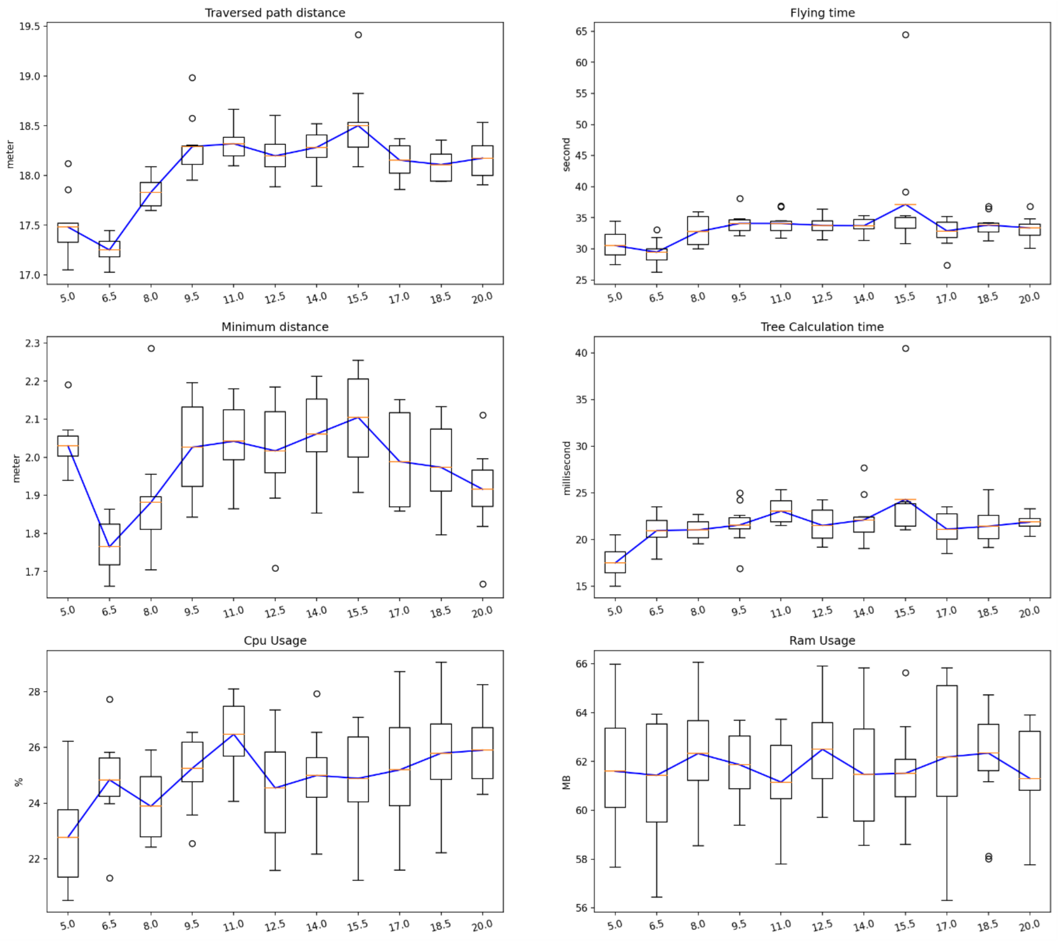


Figure 37: Result plots Experiment #15.

Table 19: Final operational parameter set choice.

Name	*Description*	*Default*	*Value*
tree_node_distance	Distance between nodes	2	0.8
children_per_node	Branching factor of the search tree	8	9
n_expanded_nodes	Number of nodes expanded in complete tree	40	20
smoothing_margin_degrees	smoothing radius for obstacle cost in cost histogram	40	33
obstacle_cost_param	Approximate distance from obstacles (m) when the obstacle distance term dominates the cost function	8.5	3.7
yaw_cost_param	Cost function weight for constant heading	3	20
tree_heuristic_weight	Weight for the tree heuristic cost	35	35
pitch_cost_param	Cost function weight for goal-oriented behavior	25	25
velocity_cost_param	Cost function weight for path smoothness	6000	6000
max_point_age_s	maximum age of a remembered data point	20	20
min_num_points_per_cell	minimum number of points in one area to be kept if lower they are discarded as noise	1	1
smoothing_speed_xy	response speed of the smoothing system in xy (set to 0 to disable	10	10
smoothing_speed_z	response speed of the smoothing system in z (set to 0 to disable)	3	3
max_sensor_range	Data points farther away will be discarded	15.0	15.0

- Virtual structure approach: In this approach, the formation of agents is considered as a single object, called a virtual structure. The desired motion for the virtual structure is given. The desired motions for the agents are determined from that of the virtual structure.

- Behavioural approach: Several desired behaviours are prescribed for agents, such behaviours as cohesion, collision avoidance, obstacle avoidance, etc.

There are three concepts used for describing formation control features, which are local and relative, and absolute.

- Relative: this term refers to the UAV's perception of variables respective to the own local coordination system.

- Absolute: this term refers to the UAV's perception of variables respective to a multi-agent global coordination system.

- Local: this term can be described from two aspects. Considering an interaction topology, a formation control system that let agents to interact with all the others can be considered non-local. On the contrary, as a formation control system performs fewer interactions, it can be considered more local. Considering sensing topology, local means relative. The term local can be described as a variable that is sensed with respect to a local coordinate system. Consequently, this term implies the non-existence of a global coordinator in both definitions.

Relative variable sensing can be associated with decentralized approaches. In this respect, distance-based control can be considered more decentralized than position-based and displacement-based control [48]. According to this, distance-based control only requires relative localization which is faster and easier than global localization. Absolute variable sensing can be associated with centralized approaches. In this respect, position-based control can be considered more centralized than distance-based and displacement-based control. Thus, position-based control requires that each agent is able to localize itself respecting to the global coordinate system, or access to some external global localization system providing that information (e.g., GPS).

A formation control scheme can be classified into centralized or decentralized according to whether it requires a global coordination system or not. Most formation control schemes found in the literature fall into decentralized control and do not explicitly require a global coordinator. A global coordinator, which implies centralized control, is an entity that gathers information from all UAVs, computes formation, makes decisions, and distributes commands to the UAVs in a cooperative manner. Considering the global coordinator's requirements for gathering required data, decentralized control is contrarily more compatible with local control, which requires less computational resources [48]. For formation control purposes, agents are required to sense variables respective to either their local coordination system or a multi-agent global coordination system. Authors in [48] believed that the characterization of formation control schemes in terms of the sensing capability and the interaction topology of agents are linked to the essential features of multi-agent formation control. Considering this characterization, the sensed variables and the actively controlled variables are the most important terms to be considered in order to achieve a desired formation in multi-agent systems. The sensed variable type implies the requirement of agent capability in sensing while the controlled variables are related to the interaction topology used by agents. Respectively, actively controlling agents' positions would let the agents move to their desired positions without any specific interaction. Meanwhile, controlling agents' inter-distances would result in more interactions among agents to obtain a desired formation as a rigid body.

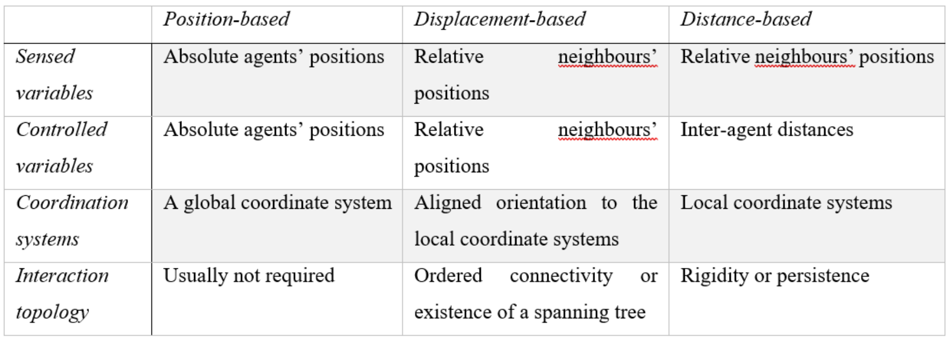
Based on observations, authors in [48] categorized the existing results on formation control into the following approaches according to types of sensed and controlled variables:

- Position-based control: It is required that the agents sense their absolute positions with respect to a global coordinate system and actively control positions to achieve the desired formation. Therefore, interactions between the agents are not essentially required as the desired formation is acquired by controlling the individual agents' position. Meanwhile, interactions can be engaged between agents for enhancing the formation control performance or addressing additional purposes. This approach is mostly considered as centralized as a global coordinator is required to get feedback from the agents and provide them with appropriate coordination commands. Additionally, interactions among the agents have turned out to be beneficial. The desired formation can be achieved without any interactions among the agents under ideal conditions. In comparison to the displacement-based and distance-based control, this approach might be costly according to the requirement of advanced sensing equipment such as GPS receivers. However, it could provide more effective solutions in practical applications [48].

- Displacement-based control: It is required that each agent senses relative positions (displacement) of its neighbouring agents with respect to the global coordinate system. Therefore, this imposes on the existence of agent interaction. Agents are actively controlling the displacements of neighbour agents to achieve the desired formation with respect to a global coordinate system. Achieving the desired formation can be described by either graph connectivity or the existence of a spanning tree.

- Distance-based control: It is required that each agent senses relative positions of neighbour agents respecting to own local coordinate system. Therefore, this imposes on the existence of agent interaction. It is not necessarily required that the orientations of local coordinate systems are aligned with other agents. To achieve the desired formation, the distances between each pair of agents are actively controlled which can be treated as a rigid body. This formation is invariant to translation, rotation, or even a combination of which applied to the correspondent positions. The interaction graph needs to be rigid and can be directed or undirected. This approach only requires relative localization which is easier than global localization implementation. In

Table 20: Formation control taxonomy according to [48].



	Position-based	*Displacement-based*	*Distance-based*
Sensed variables	Absolute agents' positions	Relative neighbours' positions	Relative neighbours' positions
Controlled variables	Absolute agents' positions	Relative neighbours' positions	Inter-agent distances
Coordination systems	A global coordinate system	Aligned orientation to the local coordinate systems	Local coordinate systems
Interaction topology	Usually not required	Ordered connectivity or existence of a spanning tree	Rigidity or persistence

comparison to position-based and displacement-based, the advantage of this control is the less need for global information. But, nonlinearity in multi-agent systems under distance-based control laws, complicates stability analysis [48].

- Other approaches: Some approaches that do not fit into the previous categories are as follows:
  - Flocking
  - Estimation based formation control
  - Pure distance-based control
  - Angle-based control
  - Containment control
  - Cyclic pursuit

Table 20 summarizes the formation control distinctions based on the previous concepts.
In terms of the interaction topology, position-based control is particularly beneficial. Although it requires agents to be equipped with more advanced sensors. In terms of the sensing capability, distance-based control is advantageous, but it requires more interactions among agents. In terms of both sensing capability and interaction topology, displacement-based control is moderate compared to the other approaches as illustrated in Fig. 38. Therefore, this reveals a trade-off between the number of agents' interactions and the agents' requirement for the sensing capability [48].
Depending on the explicitly predefined shape of the formation, it is also possible to categorize formation control as follows [48]):

- Morphous formation control: this classification implies a formation that is explicitly predefined by desired positions of agents, desired inter-agent displacements, desired inter-agent distances, or etc.
- Amorphous formation control: this classification implies a formation without explicitly predefined formation. The desired behaviours such as cohesion, collision avoidance, etc., are specified for the agents. this is related to behavioural approach discussed before.

4.3.2 Literature review

Several surveys on formation control of multi-agent systems are found in [48]. According to the amount of literature on formation control, it would be challenging to exhaustively review all the existing research on formation control. In this report, the latest solutions are reviewed based on the categorization introduced in [48].

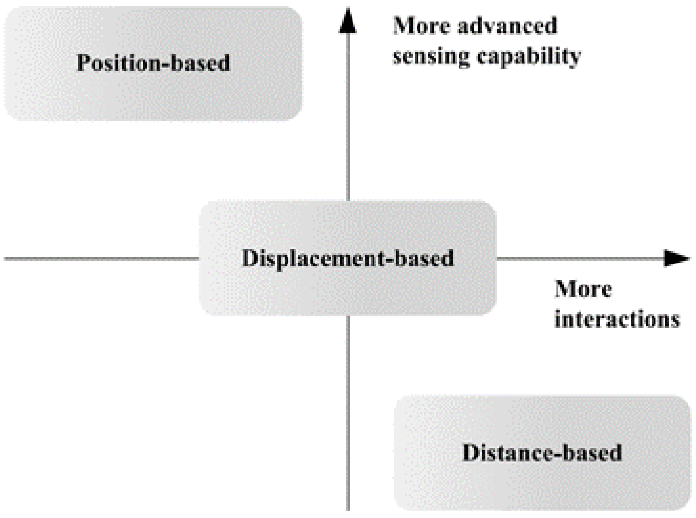


Figure 38: Sensing capability vs. interaction topology (adapted from [48]).

Position-based control – In [5], the authors presented a distributed method for formation control of a team of aerial for navigating in environments with static and dynamic obstacles. In this approach, a team of networked robots is considered in which each robot only communicates with its neighbours. It is concluded that using a constrained non-linear optimization combined with consensus, navigation of distributed teams of robots in a formation among static and dynamic obstacles can be achieved. The communications between UAVs were assumed noise-free and without packet losses (ideal communications). Using computation of an obstacle-free convex region and optimizing formation parameters, authors achieved a consensus formation control among the robots. Using consensus on convex obstacle-free regions, the robots do not need to exchange the position of all the obstacles. Instead, they compute, and exchange, the joint free spaces. Every communication process will not continue only one Δt, but several Δt due to the communication equipment. The authors successfully did the experiments in simulations with up to sixteen drones, and in experiments with up to four drones. Authors also indicated that since the approach is local, deadlocks may still occur. They have used an external motion capture system, that provides precise position information at a high update rate, to track the drones and obstacle positions.

Authors in [31] had proposed a distributed formation control and collision avoidance method based on Voronoi partition and conventional artificial potential field (APF). Using partitioning the whole space into non-overlapping regions based on Voronoi partition theory, collision avoidance is achieved. The conventional APF is used for designing the general motion control law. Using a proposed switch scheme of destinations, drones avoid collision when they reached the local equilibrium caused by the potential field. For the simulations, authors had considered eight UAVs to form a formation from random initial positions using MATLAB R2016a. For the real-world experiment, authors had used three drones , produced by Intel, as the testing platform. In the experiment, when quadrotors violate the predefined safe distance which causes the collision, the destinations are switched. The agents communicated with neighbours every Δt among a communication range less than rc, and it was practical for agents to switch their roles with others if there is a switch request from other agents within rswitch. Each agent broadcast its position and destination position to its neighbour every Δt. They have concluded that the delay will not influence the final formation structure, but it influences the duration from start to the time when a final formation is constructed. The time delay will not influence the final formation structure, but it only influences the duration from start to the time when a final formation is achieved as the collision avoidance is guaranteed by Voronoi partition and potential field.

Authors in [34] had proposed a swarm of agile micro quadrotors using an external localization system in a centralized manner. In this work, they had focused on scaling down the quadrotor groups to smaller ones to develop a truly small micro-UAV. They have believed that the most important benefit of scaling down (in size) is the ability of the quadrotor to operate in tightly constrained environments in tight formations. They have used a Vicon motion capture system to sense the position of each vehicle at 100Hz. Using mixed-integer quadratic programming techniques had let them coordinate up to twenty micro quadrotors in known three-dimensional environments including obstacles.

In [4], arbitrary target patterns were represented with an optimal robot deployment, using a method that was independent of the number of robots. The proposed approach has focused on pattern formation based on Voronoi partitioning. The aim was to generate both visually convincing final formations by optimizing the robots' goal positions, as well as simple and smooth robot motions at the transitions of patterns. To keep the formation a set of goal positions was computed. Then based on the distances to the goal positions, new goal positions were assigned to each agent. A centralized version of the Hungarian algorithm, introduced in [33], was used to find the optimal assignment. For the experiments using real robots, they have done the experiment in a flat area where 10 robots operate using an overhead camera for localization of infrared LEDs and a centralized computing unit for controlling the robots. For the simulation, they had increased the number of agents to 50.

Distance-based control – Another approach based on distributed vision-based nonlinear formation control has been proposed in [62]. The authors have presented a solution for the formation control of three UAVs while tracking a human worker in a power line tower using a distributed vision-based nonlinear formation control approach (leader-follower). This approach results in an adaptable formation where the controller minimizes the error in observation always maintaining the visualization of the human by the whole formation. Authors have proposed a formation control architecture composed of a human blob detector, a multi-UAV sensor fusion, and a formation controller. They proposed a multi-UAV sensor fusion that is based on a modified Kalman filter with combined states and covariances in the prediction step. The formation controller finds the optimal position for each UAV to track the human in formation while maintaining the desired distance and minimizing the covariance of observation.

In [60], the authors proposed a control algorithm that consists of leader-follower principle and consensus algorithm and artificial force based on APF. According to the simulation results, they had proved that the system will achieve formation while avoiding collisions between agents and obstacles while keeping the maximum distance between agents.

Using a proposed improved APF method for formation control with obstacle avoidance in a complex environment in [72], authors were able to maintain a more flexible formation obstacle avoidance while many static and dynamic obstacles exist. UAVs convergence to the desired formation is ensured by using structural constraints of the formation configuration.

In [64], Using an improved consensus algorithm (ICA) for formation control and using a combination of ICA and the particle swarm optimization (PSO) algorithm for static obstacle avoidance and using a combination of model predictive control (MPC) with PSO for dynamic obstacle avoidance. The ICA-PSO algorithm and MPC-PSO algorithm can be adaptively switched when avoiding different types of obstacles. An APF is added along the axis OZ to avoid the collision among UAVs without changing their motion in the XOY plane. The authors mentioned that the communication topology among UAVs is not the main concern in this study so that it is assumed that each UAV in the formation can communicate with others without any problem.

Authors in [19] proposed a fully distributed control strategy for agents with higher-order dynamics (i.e. drones). They proved that the fully distributed proposed control can be implemented locally on agents using the relative position measurements. Thus, agents are not supposed to communicate or have a common sense of orientation. The robustness property of SDP design allows agents to move along a rotated control direction. This property can be used to prevent collision among agents. They presented a distributed control strategy for planar formations of agents with a variety of dynamics. They considered agents with linear or input-to-state linearizable dynamics. The approach is based on the barycentric-coordinate based (BCB) control, which is fully distributed, does not require interagent communication or a common sense of orientation, and can be implemented using relative position measurements acquired by agents in their local coordinate frames.

Authors in [65] have proposed a design of distributed UAV formation control. A consensus controller, a task assignment strategy, and an obstacle avoidance algorithm had designed. A hierarchical and modular multi-UAV simulation platform called XTDrone has been developed. The authors proposed a consensus formation control algorithm based on the leader-follower principle. The simulation validation had done containing 9 UAVs in the simplified simulator, and 6 UAVs in Gazebo simulation, which validates the consensus control algorithm. In this approach, for simplification, the relative position ground truth is sent to each UAV.

The work presented in [1] shows a Leader-referenced behavioural-based approach to formation control. Additionally, it extends the work of Balch and Arkin to incorporate a formation switching strategy and a decentralized approach [7]. Using a decentralization would require transmitting a substantial amount of state between robots. Using an implementation of a motor schema, formation maintenance, as well as obstacle avoidance, was obtained. Considering other robots as dynamic obstacles, collision avoidance was performed between robots. If robots are approximately adjacent, the robot with the lowest ID halts while the other can move, preventing deadlock. They have implemented a reactive formation switching strategy which determines the safest formation for the robots considering the current environment. They have implemented centralized and decentralized formation control. Each approach visibly maintains formation while the robots traverse the arena. They concluded that having a centralized approach allows reaching the goal in a shorter time as robots are aware of other robot positions. They used a variant of the rapidly-exploring random trees (RRT) algorithm, called RRT*, to generate paths for robots. The centralized algorithm relies on knowledge of all robot positions as well as their combined view of the world. The decentralized solution achieves this with limited message passing between robots to communicate the required state. Communication links are defined between certain robots in each formation.

In [57], the authors considered the challenge of a decentralized control strategy for controlling a group of quadrotors able to measure relative bearings in their own body frames. They have simulated the purposed formation control strategy using six quadrotors in V-REP as well as having an experiment with four real quadrotors using onboard computers and onboard cameras for relative localization of the quadrotors. The proposed control/estimation scheme has not necessarily required a special topology for the interaction graph. The bearing controller and the localization algorithm have the same decentralized expression for all agents as a function of the measured bearings and body-frame linear/angular velocities. The proposed control strategy has relied on an extension of the rigidity theory. This extension has allowed them to devise a decentralized bearing controller that was needless of the presence of a common reference frame or of reciprocal bearing measurements for all the agents.

The authors proposed a decentralized gradient-based controller in [56], which was able to enforce bearing rigidity maintenance for a group of UAVs equipped with onboard cameras. The proposed control tried to localize other drones using onboard cameras and the bearing measurements. They had presented an algorithm for bearing rigidity maintenance to keep the formation of drones and the strategy is inspired by the connectivity rigidity maintenance controllers.

The authors of [29] have proposed a novel and feasible path planning technique based on a multiple-objective optimization algorithm for a group of UAVs called “angle-encoded swarm optimization”. The goal was to minimize the travel distance of UAVs while simultaneously avoiding obstacles and maintaining altitude constraints as well as the shape of the UAVs' formation. The formation was modelled as a virtual rigid body and had controlled to maintain a desired geometric shape among the generated path which was based on the centroid of the formation. They have experimented with the formation using three 3DR Solo drones equipped with a proprietary Mission Planner software, and the Internet-of-Things (IoT) for multi-directional communication between the UAVs to share position data.

A fully decentralized strategy for maintaining the formation rigidity of a multi-robot system using range measurements was proposed in [68] by authors. In this approach, the graph topology was allowed to change freely over time. A distributed algorithm for estimating a common relative position reference frame amongst a team of robots was proposed in this work. The purposed algorithm was performing only range measurements in addition to one agent which was endowed with the capability of measuring the bearing to two other agents. They have used an estimation of the rigidity eigenvalue and eigenvector which was finally used to generate a local control action for each agent to maintain the rigidity property and to enforce additional constraints such as collision avoidance and neighbour sensing or limiting communication range. In this approach, the communication and sensing links among the robots were left free to change over time while preserving rigidity. The proposed approach was experimentally validated with a robotic testbed consisting of 6 quadrotors (five real robots and a simulated one) UAVs operating in a cluttered environment.

A decentralized connectivity maintenance strategy for the teleoperation of a team of UAVs was presented in [2], an extension of the previous work reported in [22] and [58]. They have addressed the problem of connectivity maintenance for a team of drones as well as collision avoidance and obstacle avoidance using a leader-follower approach. Furthermore, they have addressed the problem of airflow avoidance by introducing an airflow-avoidance technique to prevent the influence of airflow generated by the drones on others. They also have implemented a consensus-based velocity control which enabled all follower robots to track the leader's velocity. Using this feature let the drones' fast movements while maintaining high flexibility and minimal change of topology of the formation. They have also had another improvement which enabled the automatic decrease of the connectivity eigenvalue minimum asymptote, with the objective of achieving a dynamic expansion of the formation which enabled the formation to cover ground as much as possible which can be useful for surveillance and mapping applications. They have also addressed the problem of deadlock by improving the automatic detection and resolution of deadlock configurations which was caused by conflicts between the connectivity force. The result of this work was simulated in V-REP using five to nine drones and was experimented with real drones using three drones (one leader, two followers).

In [70], cooperative control of UAV cluster formation based on distributed consensus has been studied. The formation control and speed tracking control of UAV based on distributed consensus control algorithm were implemented considering a determined communication topology. The simulation of four UAVs has been done such that the UAVs in front of the formation was broadcasting a formation change command after encountering obstacles and selecting the appropriate formation by calculation and then broadcasting the new formation to other UAVs. In this work, consensus stability could be achieved if and only if the multi-agent communication network contains directed spanning trees, or the undirected communication network topology map which was a fully connected graph. Using simulation results, they have proved that the distributed consensus-based control algorithm was able to perform the operation of the UAV cluster formation, maintenance, and transformation.

In [36], a solution for formation and obstacle avoidance of UAVs has been proposed that was able to achieve cluster situational awareness, autonomous formation control and intelligent collaborative decision making. An algorithm for the cooperative formation of multiple UAVs has been proposed that avoided the obstacles simultaneously while keeping the formation. They have integrated the improved artificial potential field algorithm and the formation algorithm to form the total virtual force by modifying the Reynolds clustering rule [49]. Using this improvement, the speed of the formation members, as well as the expected distance between agents, were maintained. the weight coefficient of consensus theory has been used to prioritize UAVs. Considering multiple UAVs as a cluster resulted in keeping formation and avoiding internal collision by exchanging state information inside the formation. They had presented the result of the proposed approach via simulations using MATLAB in different scenarios.

A formation maintenance algorithm, based on the leader-follower approach, for multiple UAVs, integrated with a collision avoidance capability was developed and simulated in [67]. The simulations have shown satisfactory results that the UAVs were able to dynamically bypass obstacles without colliding with them while maintaining the given swarm formation. The formation control was done by accelerating and decelerating on demand according to the distance with the leader drone. Using distributed approach had led the UAVs to take fast local decisions when approaching obstacles. Moreover, the algorithm has also considered the lost UAVs by routing them towards the destination and making a temporary formation when they have been lost. They have done the simulations using 5 drones in a constant altitude. The first drone was responsible for obstacle avoidance and path planning to the goal and other drones were following.

A novel onboard relative localization approach, the UVDAR system for visual relative localization with application to leader-follower formations of multirotor UAVs, was proposed in [63]. This approach, which is based on ultraviolet light, is used for real-time control of a leader-follower formation of multirotor UAVs. A new sensor, called UVDAR, is employed in an innovative way. It does not require communication and is extremely reliable in real-world conditions. This sensing system provides relative position and yaw measurements independently of environmental conditions such as changing illumination and the presence of undesirable light sources and their reflections. The maximum distance for reliable detection by the UVDAR is 15m. To validate the performances of the proposed algorithm, they conducted real-world outdoor flights experiment with two DJI f550-based hexarotors, equipped with an Intel NUC7 computer, and a PixHawk flight controller.

The authors of [40] introduced a unified formation flying pipeline with distributed formation control and task assignment solutions that run onboard the vehicles and use VIO for localization. They used the alternating direction method of multipliers (ADMM) solver to address the scalability issue of general solvers for obtaining formation gains in a computationally efficient manner. They had started to investigate scalability by comparing the runtime of the ADMM-based solver with the interior-point method used in CVX to solve the SDP formulation. They had used software-in-the-loop simulations and hardware demonstrations and implemented the pipeline in C++ using Robot Operating System. To evaluate the approach, they have experimented with the approach in simulation with up to 100 UAVs, and six UAVs in real-world by cycling through the three formations. A base station was used only to dispatch the desired formation graph to the UAVs during the experiment. For each trial, the pipeline was tested in three main configurations: with centralized assignment, with distributed assignment, and without assignment. Finally, they have concluded that every trial using assignment was successful.

Displacement-based control – In [50], the authors proposed a Sliding mode control (SMC) for trajectory tracking and control formation based on the leader-follower principle for multiple UAVs. They conclude that the implemented SMC as the formation control approach shows the desired performance and maintains the proposed formation. For the experiments, they have used bebop 2 drones consisting of the defined trajectory tracking, using 3 drones in a triangle type formation (a leader and two followers). The experimental results show that the proposed method can move the quadrotors to any position.

In [54], the authors proposed a novel concept of motion planning and stabilization of formations of micro aerial vehicles based on a dynamic virtual-leader-follower scheme. This concept is suited for utilization of onboard visual relative localization of neighbouring MAVs, which can be considered as a novel approach for GPS denied environments navigation. In this approach, Wi-Fi communication was employed for sharing the map updates. For localization purposes, they have relied only on onboard sensors (like IMU, PX4flow, and the relative visual localization system) and no external positioning system was used.

Considering a leader-follower principle, a new approach based on stress matrices to achieve formation manoeuvre control in arbitrary dimensions was proposed in [71]. Authors have proposed distributed control laws for single-integrator, double-integrator, and unicycle agent models. Any target formation can be tracked by the proposed control laws. As a result, the centroid, orientation, scales in different directions, and other geometric parameters of the formation can be changed continuously.

Other approaches – Authors in [21] had addressed the problem of motion controlling for a group of UAVs to keep a formation in terms of only relative angles, called bearing formation control. A bearing-only formation controller was proposed which was requiring only bearing measurements and trying to maintain bounded inter-agent distances despite the lack of direct metric information. In the case of a human operator in charge of steering the formation, formation maintenance was achieved by employing two force-feedback devices to provide haptic cues informative of group performances with respect to human instructions. The relative bearings that are needed by the controller were obtained from an onboard monocular camera with a horizontal/vertical FOV of about 88/60° and by detecting a coloured sphere equipped on the top of every other drone. Due to the restrictions of the limited FOV on the drones, they have forced every UAV to rotate with some rotation speed to scan the environment and periodically detect all the other UAVs. The task started with forcing every UAV to perform a complete 360° scan to estimations of all the relative azimuths. Then, during normal motion, the UAVs are supposed to rotate towards the neighbour UAV that was not seen for the longest time. Depending on the formation configuration, if a UAV was able to measure the bearings relative to all the other UAVs, it was supposed to simply rotate to keep them in the most centred way to keep the data up to date. In the simulations, they have assumed an unlimited FOV capability for the UAVs so that relative bearings could always be retrieved. They had performed the simulation of a human/hardware-in-the-loop (HHIL) involving 12 UAVs starting far from the desired bearing formation. They had estimated that the communication complexity between N drones is $2(N-1)+1$ exchanged messages over the network per unit of time.

The formation maintenance of multiple UAVs based on proximity behaviour have been studied in [37]. According to the expected UAV formation structure and the predicted position, velocity, and attitude information of other UAVs in the azimuth area, an adaptive distributed formation flight strategy was established by exploiting proximity behaviour observations. The proposed method has considered the azimuth area relative to the UAV to capture the state information of other proximal UAVs. The dependency degree factor was introduced to update the state equation based on proximity behaviour. The positions, speeds, and attitude errors were used to form an adaptive dynamic adjustment strategy in formation. In the simulation, six UAVs were distributed on three levels with a height difference of 10m and they were communicating with neighbour UAVs.

A formation control law was provided in [44]. It has combined the control and communication constraints in a balanced manner. Using consensus-based laws for double integrated dynamics, introduced in [32], and concepts of social potential function proposed in [49, 43], a leader-follower formation approach was addressed that has enabled collision avoidance while maintaining agents' network connectivity. This approach has prevented the usage of force on multiple probable collisions among agents and furthermore avoided handling too much information. The control law was simulated using MATLAB to generate a V-shaped formation.

As a case study in [20], the authors have addressed the problem of multi-agent formation control. They have proposed a distributed control strategy to stabilize a formation that was described with bearing constraints that only required bearing measurements and parallel rigidity of the interaction graph. They have considered the possibility of having a more refined model of the multi-agent network by allowing a multiple-graph representation to explicitly consider the conceptual difference between sensing, communication, control, and parameters stored in the network. They have also addressed the required interaction network by explicitly considering the conceptual difference between, sensing, communication, control, and parameters stored in the network, and by exploiting modelling analyzed the connection between scalability, minimality and rigidity.

The authors of [68] addressed a formation control problem for a team of agents that are only able to sense the relative bearings from their local body frame to neighbouring agents. Each agent has been tasked with maintaining predetermined bearings in respect to their neighbours. Using the developed rigidity theory for SE(2) frameworks, introduced in [69], they have proposed a gradient-type controller to stabilize the formation that was directed bearing rigidity matrix. The simulation had contained 6 agents.

A novel algorithm for representing static and deforming shapes with a multi-agent system, called Shaped Flocking, has been described by authors in [28]. They have demonstrated a system that allows a user to perform real-time directioning of a swarm of robots via a drawing interface for the field of entertainment robotics. For the experimental setup, they have used a workspace for robots and an overhead camera for localization and a central computer for drawing interface and robot control.

In [39], the authors proposed a reinforcement learning approach using a DeepSarsa based path planning and obstacle avoidance method for helping UAVs to avoid moving obstacles as well as finding a path to a target based on a deep neural network. To verify the performance of the proposed approach, two UAVs want to pass a small terrain, where two flying obstacles cut their paths and one static obstacle stays in the front of the exit. After performing experiments and simulation, the authors conclude that the Deep-Sarsa model performance in the application of path planning and obstacle avoidance, especially in a dynamic environment, is noticeable. The trained model can provide a reliable path for UAVs without collisions. The proposed approach has been trained in a simplified environment and tested in a ROS-Gazebo simulation platform.

Overview – The list for the state of the art on formation control is presented in Table .

For comparison purpose and summarization of the approaches, used terms, acronyms and abbreviations are defined next:

- SC: availability of the source code;
- CA: capability of collision avoidance between drones;
- OA: capability of avoiding obstacles for each drone;
- MND: the maximum number of drones used in the simulation experiment or in the real environment experiment;
- CLT: This column indicates the communication loss tolerant of the approach, i.e. how tolerant is the agents' communication while there would be a communication loss;
- RU: capability of using the approach in real-time;
- "N/A" means not enough information is available;

- Replicability: this indicates the feasibility of implementing new software based on the description of a computational model or method provided in the original publication. Replicating a published result means developing new software based on the description of a computational model or method provided in the original publication and obtaining results that are similar enough to be considered equivalent [8]. The 5-star ranking implies a percentage from $0\% - 100\%$ likelihood of a solution being replicable.

- ULCR: usability under limited computational resources (reference provided by Tables 1 and 2).

4.3.3 Selecting the right solution for the SEMFIRE use case scenarios

According to the proposed drone configuration, presented in Tables 1 and 2, and SEMFIRE project scenario, some concepts should be considered to select the most adequate approach to the SEMFIRE use case. The SEMFIRE use case includes four stages: initial deployment, reconnaissance, clearing and aftermath. In the first three stages, the Scouts are supposed to autonomously spread while maintaining multimodal connectivity among each other and the Ranger through distributed formation control, thus leading to a certain degree of spatial compactness above the target area. Scouts are collectively exploring the target area with the goal of finding the regions of interest (ROIs) within the target area. To Achieve a formation among UAVs, it could be done using a centralized or decentralized approach. Using a centralized approach requires a reliable connection among UAVs and the coordinator (possibly Ranger) and it also may require an absolute localization as well. In a decentralized approach, both absolute and relative localization could be used. It is more efficient to use a relative localization to be more resistant to possible communication loss among UAVs and Ranger.

By reviewing the literature on formation control in previous sections, some challenging main concerns regarding cooperative formation control should be considered to achieve the SEMFIRE goal. Thus, two concepts are important and should be considered. I) Communication and II) Localization.

To maintain communication, it is required to have a threshold for the distance that would limit how far the scouts could be distanced apart according to the wireless communication infrastructure between UAVs. To achieve this, it is required to have a good and reliable communication infrastructure. According to the SEMFIRE goal, it is required to have a semantic map as the result of the UAVs' explorations. Therefore, there should be good and reliable communication between the UAVs and the Ranger. Also, it is required to select a proper formation control strategy requiring less communication bandwidth.

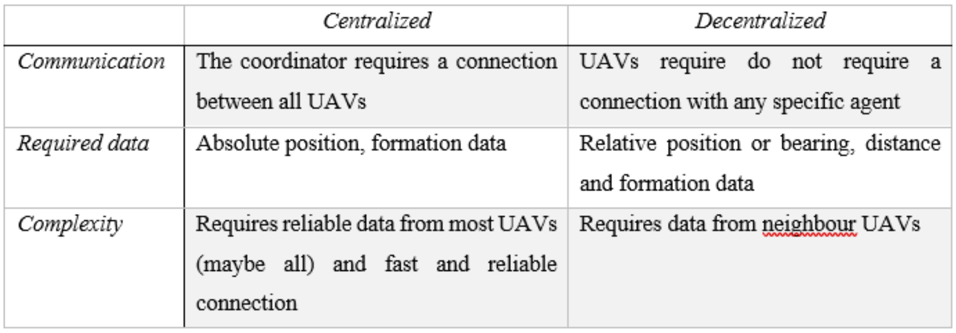
To maintain the formation, it is required to have a localization system. According to the possible errors in absolute localization using GPS below the canopies inside a forest, it is most efficient to use a relative localization system among UAVs and Ranger. In most research, it is assumed that relative localization is available to all the UAVs. Most of the approaches are done in GPS denied environment. According to the configuration of Scouts in the SEMFIRE project, it is feasible to have a localization using onboard cameras that let them maintain a distributed formation.

According to the comparisons in Tables 21 and 22, and according to the requirements-feined above, the most feasible approach required to be decentralized and a combination of global with relative localization to reduce the risk of GPS signal loss and communication loss among UAVs. Considering these concepts, the reviewed approaches are listed in Table ... in the order of highest replicability to lowest.

4.3.4 Implementation and Simulation

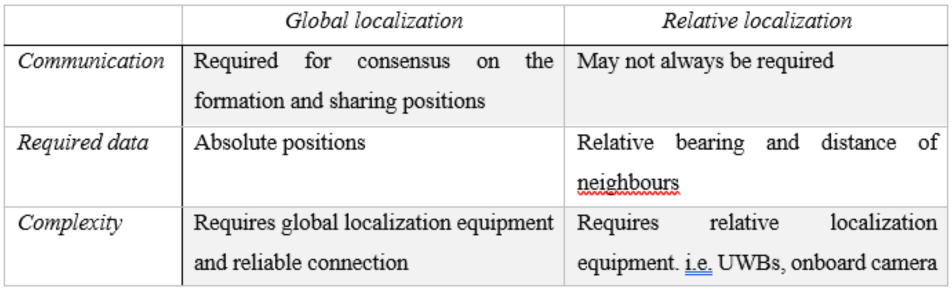
Localization error model for simulation – According to the possible errors in localization of drones inside the forest, it is required to have an additive error model for localization

Table 21: Centralized vs decentralized approach comparison.



	Centralized	*Decentralized*
Communication	The coordinator requires a connection between all UAVs	UAVs require do not require a connection with any specific agent
Required data	Absolute position, formation data	Relative position or bearing, distance and formation data
Complexity	Requires reliable data from most UAVs (maybe all) and fast and reliable connection	Requires data from neighbour UAVs

Table 22: Global vs relative localization comparison.



	Global localization	*Relative localization*
Communication	Required for consensus on the formation and sharing positions	May not always be required
Required data	Absolute positions	Relative bearing and distance of neighbours
Complexity	Requires global localization equipment and reliable connection	Requires relative localization equipment. i.e. UWBs, onboard camera

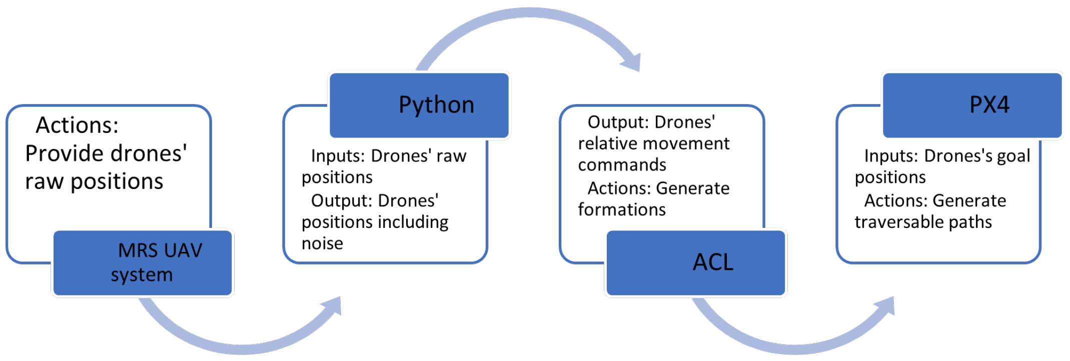
in simulation using Gazebo. In that case, three random numbers (E_x, E_y, E_z) in a uniform distribution with the maximum value of $\pm 0.5m$ is added to the current drone's position.

Integration in ROS framework – The MRS UAV system, ACLSWARM[4], and PX4-Avoidance[5] packages are used for the SEMFIRE use case. To integrate these packages, it is required to implement a middleware node to get raw localization data from the MRS UAV system and feed them to other packages as represented in Fig. 39 which presents an overview of the integration process. The drones raw position data are fed to a middle-ware python node to add the error model to them. Therefore, the noisy position data are fed to the ACLSWARM package to generate swarm positions. The output of the ACLSWARM package is fed to the PX4 avoidance package then as goals to generate trajectories considering obstacles in the FOV of the drones. In order to have the best experience in path planning, the drones are required to turn to the direction of the goals first without movement.

5 Conclusion

This deliverable addressed the deployment of the heterogeneous multi-robot team of the SEMFIRE project in simulation in order for the team to start the mission and the first task – the reconnaissance task. The problem was introduced and defined in Section 1. Section 2 surveyed relevant literature in this domain. Section 3 presented the initial deployment strategy that was designed to fulfill the specific requirements of the use-case scenarios of the SEMFIRE project. Section 4 presented extensive simulations of the deployment strategy presented in this deliverable so as to test it, introduce any adjustments found necessary, and validate it in simulated scenarios.

[4] `https://bitbucket.org/semfire-isr-uc/aclswarm/`
[5] `https://bitbucket.org/semfire-isr-uc/px4-avoidance/`



Actions:
Provide drones' raw positions
MRS UAV system
Python
Inputs: Drones' raw positions
Output: Drones' positions including noise
Output: Drones' relative movement commands
Actions: Generate formations
ACL
PX4
Inputs: Drones's goal positions
Actions: Generate traversable paths

Figure 39: System integration as a ROS framework.

In the ensuing pilot trials, the research team will validate the strategy with the real platforms in a real scenario.

References

[1] Multi-Robot-Formation-Control: A group project to implement multi-robot formation control. Github. URL `https://github.com/sumaiyah/Multi-Robot-Formation-Control`.

[2] M. Aggravi, C. Pacchierotti, and P. R. Giordano. Connectivity-Maintenance Teleoperation of a UAV Fleet With Wearable Haptic Feedback. *IEEE Trans. Autom. Sci. Eng.*, 2020. doi: 10.1109/tase.2020.3000060.

[3] Ali Ahmadi. MRS UAV System Tutorial. techreport TR-SEMFIRE-2-vA, Institute of Systems and Robotics, University of Coimbra, 2021.

[4] J. Alonso-Mora, A. Breitenmoser, R. Siegwart M. Rufli, and P. Beardsley. Multi-robot system for artistic pattern formation. In *Proc. - IEEE Int. Conf. Robot. Autom. (ICRA)*, page 4512–4517, 2011. doi: 10.1109/ICRA.2011.5980269.

[5] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus. Distributed multi-robot formation control in dynamic environments. *Auton. Robots*, 43(5), 2019. doi: 10.1007/s10514-018-9783-9.

[6] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert, and Martin Saska. The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *arXiv preprint arXiv:2008.08050*, 2020.

[7] T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Trans. Robot. Autom.*, 14(6):926–939, 1998. doi: 10.1109/70.736776.

[8] L. A. Barba. Terminologies for Reproducible Research. arXiV, February 2018. URL `https://arxiv.org/abs/1802.03311v1`.

[9] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri. Snap and spread: A self-deployment algorithm for mobile sensor networks. In Sotiris E. Nikoletseas, Bogdan S. Chlebus, David B. Johnson, and Bhaskar Krishnamachari, editors, *Distributed Computing in Sensor Systems*, pages 451–456, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-69170-9.

[10] R. W. Beard, J. Lawton, and F. Y. Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Trans. Control Syst. Technol.*, 9(6), 2001. doi: 10.1109/87.960341.

[11] S. J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar. A Survey on Aerial Swarm Robotics. *IEEE Trans. Robot.*, 34(4):837–855, August 2018. doi: 10.1109/TRO.2018.2857475.

[12] N. Correll, J. Bachrach, D. Vickery, and D. Rus. Ad-hoc wireless network coverage with networked robots that cannot localize. In *2009 IEEE International Conference on Robotics and Automation*, pages 3878–3885, 2009.

[13] M. S. Couceiro, R. P. Rocha, and N. M. Ferreira. Ensuring ad hoc connectivity in distributed search with robotic darwinian particle swarms. In *In Proc. of 9th IEEE Int. Symposium on Safety, Security, and Rescue Robotics (SSRR'2011)*, pages 284–289, Kyoto, Japan, Nov. 2011.

[14] M. S. Couceiro, C. Figueiredo, D. Portugal, R. P. Rocha, and N. M. Ferreira. Initial deployment of a robotic team: a hierarchical approach under communication constraints verified on low-cost platforms. In *Proc. of 2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2012)*, pages 4614–4619, Vilamoura, Portugal, Oct. 2012.

[15] M. S. Couceiro, D. Portugal, R. P. Rocha, and N. M. F. Ferreira. Marsupial teams of robots: Deployment of miniature robots for swarm exploration under communication constraints. *Robotica*, 32(7):1017–1038, Oct. 2014.

[16] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha. SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems. In *Proc. of 2019 IEEE/SICE Int. Symp. on System Integration*, pages 270–276, Paris, France, Jan. 2019.

[17] Micael S. Couceiro. *Evolutionary Robot Swarms under Real-World Constraints.* PhD thesis, University of Coimbra, Portugal, Sep. 2013.

[18] F. Dellaert, T. R. Balch, M. Kaess, R. Ravichandran, F. Alegre, M. Berhault, R. McGuire, E. Merrill, L. V. Moshkina, and D. Walker. The Georgia Tech Yellow Jackets: A marsupial team for urban search and rescue. In William D. Smart, Tucker R. Balch, and Holly A. Yanco, editors, *AAAI Mobile Robot Competition 2002, Papers from the AAAI Workshop, 28 July - 1 August 2002, Edmonton, Alberta, Canada*, volume WS-02-18 of *AAAI Technical Report*, pages 44–49. AAAI Press, 2002.

[19] K. Fathian, S. Safaoui, T. H. Summers, and N. R. Gans. Robust Distributed Planar Formation Control for Higher Order Holonomic and Nonholonomic Agents. *IEEE Trans. Robot.*, 37(1):185–205, February 2021.

[20] A. Franchi and P. R. Giordano. Decentralized control of parallel rigid formations with direction constraints and bearing measurements. In *Proc. IEEE Conf. Decis. Control*, 2012. doi: 10.1109/cdc.2012.6426034.

[21] A. Franchi, C. Masone, V. Grabe, M. Ryll, H. H. Bülthoff, and P. R. Giordano. Modeling and control of UAV bearing formations with bilateral high-level steering. *Int. J. Rob. Res.*, 31(12), 2012. doi: 10.1177/0278364912462493.

[22] P. R. Giordano, A. Franchi, C. Secchi, and H. H. Bülthoff. A passivity-based decentralized strategy for generalized connectivity maintenance. *The International Journal of Robotics Research*, 32(3), March 2013. doi: 10.1177/0278364912469671.

[23] N. D. Griffiths Sànchez, P. A. Vargas, and M. S. Couceiro. A Darwinian swarm robotics strategy applied to underwater exploration. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6, 2018.

[24] D. Gronau. The Spiral of Theodorus. *The American Mathematical Monthly*, 111(3): 230–237, 2004.

[25] R. Groß, R. O'Grady, A.L. Christensen, and M. Dorigo. The swarm-bot experience: Strength and mobility through physical cooperation. In Serge Kernbach, editor, *Handbook of Collective Robotics*, chapter 2. Jenny Stanford Publishing, 2013. doi: 10.4032/9789814364119.

[26] H. K. Hahn and K. Schoenberger. The ordered distribution of natural numbers on the square root spiral. *The Journal of Business*, 2007.

[27] G. Hattenberger, S. Lacroix, and R. Alami. Formation flight: evaluation of autonomous configuration control algorithms. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2628–2633, 2007.

[28] S. Hauri, J. Alonso-Mora, A. Breitenmoser, R. Siegwart, and P. Beardsley. Multi-Robot Formation Control via a Real-Time Drawing Interface. *Springer Tracts Adv. Robot.*, 92:175–189, 2014. doi: 10.1007/978-3-642-40686-7_12.

[29] V. T. Hoang, M. D. Phung, T. H. Dinh, and Q. P. Ha. Angle-Encoded Swarm Optimization for UAV Formation Path Planning. In *IROS 2018*, 2018. doi: 10.1109/IROS.2018.8593930.

[30] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 299–308, Tokyo, 2002. Springer Japan. ISBN 978-4-431-65941-9.

[31] J. W. Hu, M. Wang, Q. Pan C. H. Zhao, and C. Du. Formation control and collision avoidance for multi-UAV systems based on Voronoi partition. *Sci. China Technol. Sci.*, 63(1), 2020. doi: 10.1007/s11431-018-9449-9.

[32] S. Joshi and O. González. Consensus-Based Formation Control of a Class of Multi-Agent Systems. Technical Report NASA/TM-2014-218663, National Aeronautics and Space Administration (NASA), 2014.

[33] H. W. Kuhn. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.*, 2(1–2):83–97, March 1955. doi: 10.1002/NAV.3800020109.

[34] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar. Towards a swarm of agile micro quadrotors. *Auton. Robot.*, 35(4), July 2013. doi: 10.1007/S10514-013-9349-9.

[35] G. Lee, Y. Nishimura, K. Tatara, and N. Y. Chong. Three dimensional deployment of robot swarms. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5073–5078, 2010.

[36] Q. Lin, X. Wang, and Y. Wang. Cooperative Formation and Obstacle Avoidance Algorithm for Multi-UAV System in 3D Environment. In *Chinese Control Conference, CCC, 2018*, July 2018. doi: 10.23919/ChiCC.2018.8483113.

[37] W. Liu, X. Zheng, and Z. Deng. Adaptive distributed formation maintenance for multiple UAVs: Exploiting proximity behavior observations. *J. Cent. South Univ.*, 28(3): 784–795, April 2021. doi: 10.1007/S11771-021-4645-6.

[38] D. S. B. Lourenço, J. F. Ferreira, and D. Portugal. 3D local planning for a forestry UGV based on terrain gradient and mechanical effort. In *Proc. of 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2020), , Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020)*, Las Vegas, NV, USA, Oct. 2020.

[39] W. Luo, Q. Tang, C. Fu, and P. Eberhard. Deep-Sarsa Based Multi-UAV Path Planning and Obstacle Avoidance in a Dynamic Environment. In *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, volume 10942, page 102–111. LNCS, June 2018. doi: 10.1007/978-3-319-93818-9_10.

[40] P. C. Lusk, X. Cai, S. Wadhwania, A. Paris, K. Fathian, and J. P. How. A Distributed Pipeline for Scalable, Deconflicted Formation Flying. *IEEE Robot. Autom. Lett.*, 5(4): 5213–5220, March 2020. doi: 10.1109/lra.2020.3006823.

[41] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030.

[42] A. Metiaf and Q. Wu. Particle swarm optimization based deployment for WSN with the existence of obstacles. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 614–618, 2019.

[43] A. Mueller. Modern Robotics: Mechanics, Planning, and Control

$$Bookshelf$$

. *IEEE Control Syst. Mag.*, 39(6):100–102, 2019. doi: 10.1109/MCS.2019.2937265.

[44] S. Mukherjee and K. Namuduri. Formation Control of UAVs for Connectivity Maintenance and Collision Avoidance. In *Proceedings of the IEEE National Aerospace Electronics Conference, NAECON, 2019*, 2019. doi: 10.1109/NAECON46414.2019.9058089.

[45] R. R. Murphy. Marsupial and shape-shifting robots for urban search and rescue. *IEEE Intelligent Systems and their Applications*, 15(2):14–19, 2000.

[46] R. R. Murphy, M. Ausmus, M. Bugajska, T. Ellis, T. Johnson, N. Kelley, J. Kiefer, and L. Pollock. Marsupial-like mobile robot societies. In *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, page 364–365, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 158113066X. doi: 10.1145/301136.301236. URL `https://doi.org/10.1145/301136.301236`.

[47] M. Niccolini, M. Innocenti, and L. Pollini. Near optimal swarm deployment using descriptor functions. In *2010 IEEE International Conference on Robotics and Automation*, pages 4952–4957, 2010.

[48] K. K. Oh, M. C. Park, and H. S. Ahn. A survey of multi-agent formation control. *Automatica*, 53, 2015.

[49] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. Automat. Contr.*, 51(3):401–420, March 2006. doi: 10.1109/TAC.2005.864190.

[50] D. V. Redrovan and D. Kim. Multiple quadrotors flight formation control based on sliding mode control and trajectory tracking. In *International Conference on Electronics, Information and Communication, ICEIC 2018*, January 2018. doi: 10.23919/ELINFOCOM.2018.8330657.

[51] W. Ren and Y. Cao. Distributed coordination of multi-agent networks: Emergent problems, models, and issues. In *Communications and Control Engineering*, number 9780857291684, page 1–307. Springer International Publishing, 2011.

[52] P. E. Rybski, N. P. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson. Enlisting rangers and scouts for reconnaissance and surveillance. *IEEE Robotics Automation Magazine*, 7(4):14–24, 2000.

[53] C. Sahin, E. Urrea, M. U. Uyar, M. Conner, I. Hokelek, G. Bertoli, and C. Pizzo. Self-deployment of mobile agents in MANETs for military applications. In *Army Science Conference*, pages 1–8, 2008.

[54] M. Saska, T. Baca, and D. Hert. Formations of unmanned micro aerial vehicles led by migrating virtual leader. In *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016. doi: 10.1109/ICARCV.2016.7838801.

[55] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen. A survey of spacecraft formation flying guidance and control (Part II): Control. In *Proceedings of the American Control Conference*, volume 4, 2004. doi: 10.23919/acc.2004.1384365.

[56] F. Schiano and P. R. Giordano. Bearing rigidity maintenance for formations of quadrotor UAVs. In *Proc. - IEEE Int. Conf. Robot. Autom. (ICRA 2017)*, page 1467–1474, July 2017. doi: 10.1109/ICRA.2017.7989175.

[57] F. Schiano, A. Franchi, D. Zelazo, and P. R. Giordano. A rigidity-based decentralized bearing formation controller for groups of quadrotor UAVs. *IEEE Int. Conf. Intell. Robot. Syst.*, page 5099–5106, November 2016. doi: 10.1109/IROS.2016.7759748.

[58] C. Secchi, A. Franchi, H. H. Bulthoff, and P. Robuffo Giordano. Bilateral control of the degree of connectivity in multiple mobile-robot teleoperation. In *Proc. - IEEE Int. Conf. Robot. Autom. (ICRA 2013)*, page 3645–3652, 2013. doi: 10.1109/ICRA.2013.6631089.

[59] H. Tang, Q. Wu, and B. Li. An efficient solution for joint power and trajectory optimization in UAV-enabled wireless network. *IEEE Access*, 7:59640–59652, 2019.

[60] R. Toyota and T. Namerikawa. Formation control of multi-agent system considering obstacle avoidance. In *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE 2017*, November 2017. doi: 10.23919/SICE.2017.8105616.

[61] Sfera Utimate. Deliverable 1.1. Technical report, SEMFIRE P2020 R&D Project, December 2018.

[62] T. Uzakov, T. P. Nascimento, and M. Saska. UAV Vision-Based Nonlinear Formation Control Applied to Inspection of Electrical Power Lines. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020. doi: 10.1109/ICUAS48674.2020.9213967.

[63] V. Walter, N. Staub, A. Franchi, and M. Saska. UVDAR System for Visual Relative Localization with Application to Leader-Follower Formations of Multirotor UAVs. *IEEE Robot. Autom. Lett.*, 4(3):2637–2644, July 2019. doi: 10.1109/LRA.2019.2901683.

[64] Y. Wu, J. Gou, X. Hu, and Y. Huang. A new consensus theory-based method for formation control and obstacle avoidance of UAVs. *Aerosp. Sci. Technol.*, 107, 2020. doi: 10.1016/j.ast.2020.106332.

[65] K. Xiao, L. Ma, S. Tan, Y. Cong, and X. Wang. Implementation of UAV Coordination Based on a Hierarchical Multi-UAV Simulation Platform. arXiV (Online), May 2020. URL `https://arxiv.org/abs/2005.01125v3`.

[66] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee. Deployment of mobile robots with energy and timing constraints. *IEEE Transactions on Robotics*, 22(3):507–522, 2006.

[67] J. N. Yasin, M. H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila. Formation Maintenance and Collision Avoidance in a Swarm of Drones. In *ISCSIC 2019: 2019 3rd International Symposium on Computer Science and Intelligent Control*, 2019. doi: 10.1145/3386164.3386176.

[68] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano. Decentralized rigidity maintenance control with range measurements for multi-robot systems. *Int. J. Rob. Res.*, 34(1), 2015. doi: 10.1177/0278364914546173.

[69] D. Zelazo, P. R. Giordano, and A. Franchi. Bearing-only formation control using an SE(2) rigidity theory. In *Proc. IEEE Conf. Decis. Control*, volume 54, page 6121–6126, 2015. doi: 10.1109/CDC.2015.7403182.

[70] J. Zhang, W. Wang, Z. Zhang, K. Luo, and J. Liu. Cooperative Control of UAV Cluster Formation Based on Distributed Consensus. In *IEEE International Conference on Control and Automation, ICCA 2019*, 2019. doi: doi:10.1109/ICCA.2019.8899916.

[71] S. Zhao. Affine Formation Maneuver Control of Multiagent Systems. *IEEE Trans. Automat. Contr.*, 63(12):4140–4155, December 2018. doi: 10.1109/TAC.2018.2798805.

[72] Y. Zhao, L. Jiao, R. Zhou, and J. Zhang. UAV formation control with obstacle avoidance using improved artificial potential fields. In *IEEE 2017 36th Chinese Control Conference (CCC)*, July 2017. doi: 10.23919/ChiCC.2017.8028347.