Performance Evaluation and Simulation of the Traversal Algorithms for Robotic Agents in Advanced Search and Find (ASAF) System

Ahmed Barnawi and Marwan Alharbi
Faculty of Computing and IT, King Abulaziz University, Saudi Arabia

Abstract: *ASAF is a multiple agent robotic system where Unmanned Aerial Vehicles (UAV) and Ground Vehicles (UGV) agents perform coordinated tasks. Our research group built this system based on Multiple Unmanned Autonomous Vehicle Experimental Testbed (MAUVET), an in-house platform that we have introduced as well. The challenge in the development of a mobile robotic system is that performance in real time deployment differs from the original plan. This case is clearer when planning the traversal path of an agent, where error happens because of mechanical and environmental factors. The aim of this paper is to investigate the agent traversal execution via system experimentation and computer simulation. The outcome of this investigation is understanding this behavior under different sets of circumstances and finding some optimization factors*.

Keywords: *Unmanned aerial vehicles, UAV controller, embedded systems, multilayered architecture*.

Received June 18, 2019; accepted June 18, 2020
https://doi.org/10.34028/18/4/14

1. Introduction

The Advanced Search and Find (ASAF) system is a multi- agent robotic system that assigns several heterogeneous robots searching for an object in an open area, each is to follow a predefined traversal path. Our research group is developing this system on top of a heterogeneous state-of-the-art testbed developed by our research group named Multiple Unmanned Autonomous Vehicle Experimental Testbed (MUAVET) [5]. We have also implemented and documented the algorithm to generate traversal paths in submitted paper [7]. In that paper, we introduced several traversal patterns like zig-zag traversal, and two strategies to divide the searched area among several flying robotic agents. We also studied the behavior of the agents' autopilot and found that due to systems' mechanical limitations and Global Psitioning System (GPS) signal accuracy, the agent does not follow the path between two waypoints (start to end) exactly, and an error happens. We also saw that this error, in real life experiments, is a function of the speed of the agent. The problem is that such an error will affect the accuracy of the target detection process as an agent's onboard camera should capture each point in the area at least once to find the target we are looking for.

In this paper we further investigate the behavior of the drone and the optimization of the overall performance by altering the traversal strategy to reduce the gaps in the traversed area. We have built a simulator based on the statistical data from the real experiments to simulate a drone traversing an area.

The simulation results shown in this paper have enhanced our understanding of the agents' behavior and gave us insights on how changing some traversing parameters may affect performance. It also helped us to compare different traversal strategies and get first impression prior to actual deployment on the experimental testbed.

The structure of this paper is as follows; Section 2 is related work. Section 3 highlights the characteristics of ASAF architecture. Section 4 introduces the system components, testbed and setup. Section 5 presents the traversal algorithm and its components. Section 6 presents error estimation analysis based on field experimental data along with discussion about performance analysis and optimization aspects. The last sections are conclusions and acknowledgments.

2. Related Work

On cooperation and coordination of multiple Unmanned Aerial Vehicles (UAVs), Wenjing and Shenghong [43] discuss several types of architectures for multi UAV cooperation, namely:

1. Multi-agent.
2. Basestation.
3. Central agent.

Our MUAVET [5] in theory supports all these modes, but for this set of experiments, we have implemented it to run in base station mode. The paper lists drawbacks of those methods, such as a single point of failure in (2) and (3) and communication difficulties in (1), then proposes a work-flow based approach for cooperation:

it gives agents autonomousity in generating decisions but unifies their parameters that would lead to a decision, leading to agents taking similar and coordinated decisions. This approach is initially for military uses but is adaptable to search and find applications as well.

Moreover, Wenjing and Shenghong [42], the same authors perform a more thorough and systematic comparison between the cooperation architectures, comparing time and data consumption of the different models. They found that Multi-Agent model is worse than the model they proposed 'workflow-based'.

Chen *et al*. [12] propose a reciprocal decision approach for the sensing of multi-UAV swarms. The approach is self-organized, distributed, and autonomous without the need for optimal parameters to be determined through repeated experiments.

More on search path design, Cabreira *et al*. [11] compare between spiral and zig-zag patterns in terms of energy efficiency, arguing that both are computationally cheap, saving energy in that regard. They claim-based on simulation results in many different areas-that spiral uses less energy overall, by up to 16.

On a similar note, Andersen [2] presents and analyses multiple path patterns: two zig-zag patterns: one with longer side length and a shorter one, spiral, and triangular 'sector' pattern. Andersen [2] selects spiral as the best in multiple metrics but without backing from real-world experiments. Both this and [11] lack real world experiments to prove their findings as opposed to our research. Thus the claim in [2, 11] should be examined as an objective of our study. Traversing a region needs some preparations such as detecting polygon shape, converting, and rotating to get less turns. Araujo *et al*. [3] starts with solving concave polygon by converting it into a convex shape. The second step is rotating the whole region to find maximum height between two points.

In [27], while being general for both rotor-based and multi-rotor craft, concludes that decomposing complex areas to their components and searching them individually in convex patterns yields best results. Researcher also used rotation of areas to improve performance of patterns. They conclude that while spiral pattern yields shorter paths, they sometimes produce very sharp turns and are harder to implement in code. Zig-zag patterns, on the other hand, have better coverage despite being longer. They propose mixing both patterns in different subareas based on desired metrics.

Otte *et al*. [28] discuss the limitations of spiral strategies. If the search area is a disk, spiral strategies work well. If an environment is 'roughly' disc-shaped, a near-optimal spiral strategy is as follows: create a closed path by connecting the two branches of a double spiral with short path segments. i.e., one near the search space edge and the other near the source in the spiral center. However, if the environments are not disk-like, Otte *et al*. [28] believe that for the following reasons, any spiral strategy will work poorly:

1. If we use a single spiral, we will sweep a large amount of area outside the search space.
2. Alternatively, if we use multiple spirals to cover space, we need a large overlap between different spirals for each covering. Our aim should be then to reduce the computing and power overhead by converting the non-disk like area into disk like shape i.e., convex shape.

Held and de-Lorenzo [19] introduce a simple and easy-to-use algorithm for computing polygonal spirals to cover planer shapes with straight line segments and circular arcs. However, their algorithm generates a spiral path with extra spacing.

When factoring wind effects on path following and trajectory planning, Liu *et al*. [21] list two common approaches: correcting path while in-flight by detecting deviations with GPS or similar utilities, and preplanning the path with wind in mind - which works when wind is constant, known, or predictable (which is also our approach-although, since our MUAVET [5] architecture is pluggable, extendable, and open source, we can bake any afore-mentioned approach in it). Paper then continues to propose an approach where an external apparatus measures wind then feeds the agent or a controller with necessary information for them to make corrections in real-time. Their work - again - is targeting fixed wind craft but applies to multi-rotor UAVs as well.

Also, Guerrero *et al*. [17] discusses the problem in terms of multi-rotor UAVs and proposes using wind to operators' advantage by turning areas into the wind to save energy (by making long legs downwind and short ones against it). This can be useful when flying a path. They, however, do that with complex algorithms to change agent headings and power settings, as opposed to our novel solution which just shifts waypoints and lets agents simply fly 'straight'.

Several researches discuss wind modes such as across wind, upwind, downwind, and diagonal. The mode of wind travel usually consists of a series of 180^o turns, with an average distance across the wind [31]. The upwind mode can be diagonal or directly upwind based upon stored angles and is more efficient than the across wind mode [1, 29, 35]. Mir *et al*. [26] present wind shear model in both linear and nonlinear cases. They find the challenges and the positive impact they have met. In another word, it obvious that wind force could hinder the sensing process and thus must be addressed.

For simulating UAVs, researchers have used several types of simulation environments. One of such is X-plane, which is originally a general-purpose flight simulator more focused on gaming [41]. But its pluggable nature enables its use for simulating UAVs,

such as Garcia and Barnes [16] and Babka [4] did. In [4], researchers pointed out that it is easy to simulate several types of terrain, visibility, and weather in X-plane. Moreover, work in [16] used several PCs with modest specs-for their time in 2010-, X-plane has limitations such as mentioned by the researchers themselves, having to use one machine per agent, as well as scalability issues [16]. Also, work in [4] points out another drawback of similar approaches (using X-plane, Microsoft's Flight Simulator X, or some other similar products) which is some lack of accuracy when agents are on the smaller side, although this can be mitigated by developing own flight model, as Sorton and Hammaker [33] did and also the model presented in [9] which is standalone. Another way is building a simulator over some graphics engine that has a physics model like Unity (which is the approach used to simulate drones in our MUAVET project). This was also present in [23] where they discuss that unity is more helpful for simulating some complicated aspects of UAVs like collision avoidance and object detection in real-time fashion.

Various methods where used to improve UAV localization and increase precision when following paths assigned to them. Our segmented paths method is one of them. More discussion about segmented paths is present in a later section. Other methods include using different methods of localization in addition to GPS signals, like work in [32] which uses recognition of some scenery features that have a known location to significantly boost localization precision. Ben *et al*. [8] and Merino *et al*. [24] showed diverse ways of following the same techniques. Other ways of navigation include internal based ones like using inertia, image, air sensors and gyroscopic data to sense how much an agent has moved with respect to a known point [24]. The most precise method (although it is overkill for our purpose in this project) is a hybrid of traditional GPS navigation in addition to one or more other methods [18]. Such work is also present in work by [14].

3. A Framework for Search and Find System

In this paper we present algorithms for task division and path traversal for multi robotic mobile agents. The novelty of this work can be perceived as follows:

1. The path traversal algorithm developed, incorporates the camera's pacifications in order to ensure that process of target detection and to control the overlapping of cameras view thus all points in the searched area are covered evenly in an efficient manner unlike many similar applications where trajectory overlapping is somewhat tolerated in favor of decentralized control of the system which is not the case in our system since we assume centralized control.
2. We introduce two task division algorithms (region and path division), in either algorithm we incorporate the home location coordinate to be accounted for within the offered drone capabilities. Moreover, though the region division algorithm is straight forward, the path division algorithm introduced improves the coverage of scanned area, in particular in the adjacent sub areas boarders. Our analysis also has shown that the path division strategy is more energy efficient in terms of processing power.
3. We also provide a thorough study of the drone behavior while executing the trajectory using the developed algorithms onboard of drone in real time experiments. We have established the relationships between the localization errors and drone speed in order to lie the ground for further optimization in path planning for the system.

From a design point of view, the developed framework could be characterized based on similar frameworks that are found in the literature from the following perspective aspects explained in the following subsections. We then end this section by presenting some comparison between our framework and some frameworks of similar applications.

3.1. Cooperation (Autonomous Versus No Autonomous Systems)

A strategic decision concerning the suggested framework was to tackle the search and find application based on multiple agents cooperating among themselves where the task of each of them is determined prior to the mission start. This strategy, "non-autonomous strategy" has been used in several studies such as [13, 45] this strategy is quite different from the strategy where the task is executed by multiple autonomous agents where each of them performs the mission independently with minimum interaction or coordination (no task division) [10, 15, 30, 44].

3.2. Integration (Homogenous Versus Heterogeneous Agents)

Another design goal in our framework is to deal with agent diversity to enable the integration of different types of mobile agents (i.e., aerial, ground and marine) and/or agents with different capabilities (i.e. battery lifetime, image processing power or size), having this in mind, the system developed considers scenarios where heterogeneous mobile robots are able to perform methodological search and find tasks tailored to the needed operation. The MAUVET platform, explained in following section, ensures mobile robotic agent heterogeneity as the developed centralized system uses

standardized communication interfaces and open software architecture, this aspect of heterogeneity can be found in several studies such as [20, 25, 34, 45].

Table 1. Framework comparisons.

Aspects	Mobile Robotic Systems		ASAF
Cooperation	Autonomous	Non-autonomous	The ASAF system is considered as a none-autonomous system where the task for search and find process is divided over a team of cooperative mobile robots. In ASAF, the task division function is used to divide the task according to the robot capabilities. Unlike [13, 45], the ASAF system task division function is based on centralized architecture.
	[10, 15, 22, 30, 44]	[13, 45]	
Integration	Homogeneous	Heterogeneous	While ASAF is a heterogeneous system that facilitates heterogeneity based on either mobile robot type (i.e. aerial, marine or ground) as in [20, 25, 34] or based on mobile robot capabilities as in [45]
	[10, 13, 15, 30, 44]	[20, 25, 34, 45]	
Coordination	Centralized	Decentralized	ASAF is a centralized system where the controlling software reside on a central node. The centralized coordination enables better management of the resources and ensures avoidance for trajectory overlapping unlike the case in the decentralized systems in [10, 15], [30, 44] and the centralized system in [45]. While in [30] centralization is ensured through online guidance throughout the mission, the ASAF systems centralized functions to plan the path for instant, can be offloaded to various system components.
	[30, 45]	[10, 13, 15, 44]	

3.3. Coordination (Centralized Versus Decentralized Systems)

Similar to system developed in [30, 45], the coordination of the agents in the MAUVET system has been developed to be centralized where the task planning, monitoring and control functions for the whole mission are deployed on a single entity. Although the Single Point of Failure (SPF) is an obvious disadvantage of such architecture, the centralized approach ensures consistency and stability throughout a system of heterogeneous mobile robots performing a search and find mission where strict methodology of workflow must be followed. Moreover, with regard to the SPF, we would like to distinguish between the Physical Single Point of Failure (PSPF) and Logical Single point of Failure (LSPF). While the PSPF is associated with the hardware hosting the control software, the LSPF is associated with the central software bugs, attacks or errors. In fact, our system can be made more resilient against the PSPF by enabling offloading the running of the control software to backup sites (or even onboard of one of the mobile agents with enough computing resources) since the communication protocols are based on standardized TCP/IP interfaces and technologies. On the other hand, although the main advantage of decentralized system architecture is appreciated with dealing with LSPF, the decentralized approach doesn't really suite our need to follow a strict workflow by multiple mobile agents with limited onboard resources performing a choreographed maneuver.

3.4. Framework Comparisons

In order to compare ASAF framework with others developed, Table 1 summarizes our arguments where multi agent systems were developed to deal with similar applications.

4. ASAF: SYSTEM

We have developed the 'Advanced Search and Find' (ASAF) system as an application on top of MUAVET [5] which is a programmable experimental testbed consisting of multiple robotic agents performing pre-assigned tasks. A comprehensive GUI interface [6] monitors and manages the tasks where search parameters are set. In the following subsections we briefly highlight both the MUAVET and ASAF system.

4.1. MUAVET

The MUAVET project is an experimental testbed of novel multi robotic system that supports communication and task distribution in real-life conditions of open-air area. The verification scenario targets a setup of multiple Autonomous Vehicles (AV) including Unmanned Aerial Vehicles (UAV), Ground Vehicles (UGV) or marine vehicles, performing coordinated tasks such as a search scenario for objects of interest over a given area.

The designed system infrastructure applies to all types of robots, but UAVs are the primary actor in the platform. We can incorporate UAVs (or other robots) with unique functionalities in the system.

Figure 1 shows the basic system components and the interfaces between them. Over an openair area, the robots are situated. The UAV agents physically connect to the Base Station (BS) via the access point in a star topology. Note that different unmanned aerial vehicle types are present here including ground and marine robotic vehicles. A graphical user interface runs and manages this system.

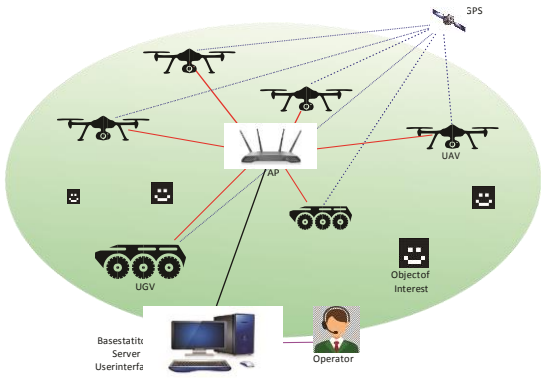


Figure 1. MUAVET.

The MUAVET Agent subsystem consists of an onboard PC, a flight controller, a GPS receiver, a camera, a Wi-Fi Communication Module, and sensors. A low-level standard TCP/IP socket interface supplies functionality of the UAV Control software. We have built a C++ Application Interface (API) above the socket interface in form of C++ functions encapsulated by MUAVET Interface class. The application code can exploit UAV control functionalities simply by calling function from this API. Both these interfaces are available on the server as well as on the UAV onboard PC, so it is up to the user to decide the strategy of controlling the agent, i.e., centralized control via BS or autonomous control via onboard PC.

4.2. ASAF

ASAF system is an application we developed to automate search and find process. 'Search and find' of an object is an incredibly routine process that consumes a lot of human and machine resources. Automation of such an operation is a perfect application for using robotic agents. Such an application will be widely beneficial. Figure 2 shows the default scenario in ASAF, the agent, i.e., an UAV, will have to fly along the pre-planned trajectory, collect sensor data (namely camera images to detect objects of interest) and send information about its own position and sensor data to the ground base station.

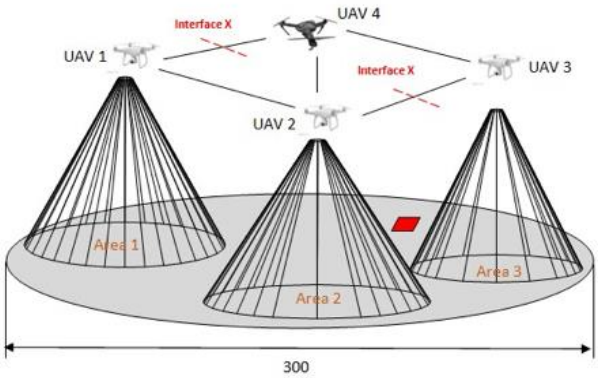


Figure 2. The default scenario.

Path planning is a part of ASAF that generates the traversal path for an agent. The traversal algorithms ensure that agents scan each point in the searched area at least once. Otherwise, search might not be successful. In section 4, we introduce various traversal algorithms to produce paths for robotic agents.

5. Proposed Traversal Algorithm for ASAF

A selected region may result in several types of polygons such as convex, concave or self-intersect.

We provide in [7] a traverse algorithm that generates a Zig-zag path (Figure 3). Here we describe another traversal approach: is the spiral path Algorithm 1. Both Zig-zag and spiral algorithm process convex polygons; however, they can give a solution for concave by ignoring inner points. They start with polygon type detection. In case of concave shape, the algorithms will convert a given region into convex shape.

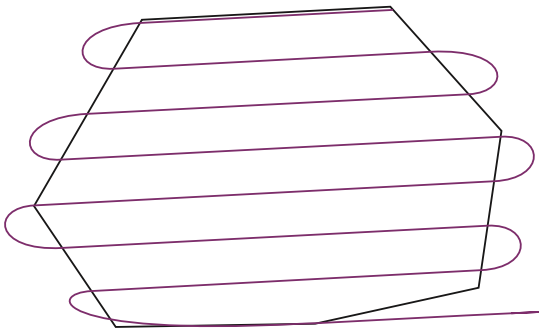


Figure 3. Zig-Zag traversing.

The algorithm rotates a convex region according to its minimum height. That results in less turns when generating traversing paths. Now, Zig-zag algorithm is ready to generate traversing paths which are lists of *x*-*y* points.

Moreover, Spiral Algorithm 1 need a convex polygon to reduce number of turns. However, there is no need for region rotation according to the minimum height since this has no effect when generating the traversal path.

To improve drone flying experience and optimize the performance, there are two approaches we add to Zig-zag paths which are: segmented and curved paths.

5.1. Spiral Algorithm

Similar to work in [11], we developed our spiral algorithm, Algorithm 1, the inputs to our algorithm are polygon points (*P*), camera properties (*F*), and overlap percentage (*V*). Thus this algorithm should produce a spiral path, Figure 4. The first step is converting the polygon into convex if possible; otherwise, we did not provide solution for nonconvex polygon. Then compute polygon boundary (x_{min},y_{min},x_{max},y_{max},*width* and *height*).

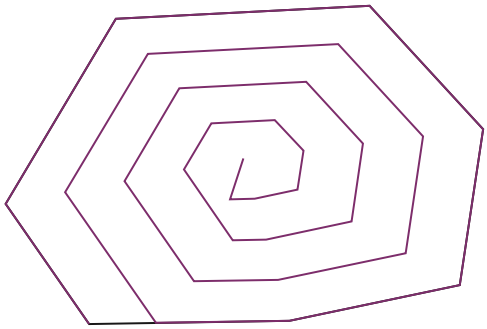


Figure 4. Spiral path traversal.

We developed Equation (1) to compute the factors for both height and width to find number of inner polygons that required to be scaled down. The polygon boundary variables and those that computed from the Equation (1) (n_h, n_w) are used to divide the polygon. Each iteration in algorithm, polygon gets scaled down by (per_h, and per_w) which per_h is for scaling each y in polygon points and per_w is for x in polygon points.

$$n_h = \frac{height}{step}, n_w = \frac{wedth}{step} \quad (1)$$

Moreover, this process results several polygons placed inside each other. These polygons are connected to each other using last point of larger polygon to the first point of next polygon. This connection process will be repeated until connecting the smallest polygon which width equal the step size or less. Last point of the smallest polygon is connected to the center.

Algorithm 1: Spiral Traversing Path algorithm

Input: Points
$P: \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$,
$camera\ Properties\ F$,
$Overlap\ V: 0.0 \le V \le 1.0$
Output: pt path points,
h_{max} *maximum flight height,*

$procedureSprialTraversal\ (P, F, V)$
if not test Convex And Convert(P)then
  return nil
Find
min_x; min_y; max_x; max_y; width; height of the polygon
$(w_{pattern}, w_{sensor}, C_{res}, C_{fov} C_p) \leftarrow Camera\ (F)$
$h_{max} = \frac{w_{pattern} \times C_{res}}{2C_p \times tan\frac{C_{fov}}{2}}$
$h_{min} = h_{max}/2$
$h = h_{max}(1 + V)/2$
$Step \leftarrow \frac{w_{sensor} \times h_{max}}{2f}$
$i \leftarrow 0$
$n_h \leftarrow \left\lceil \frac{height}{step} \right\rceil, n_w \leftarrow \left\lceil \frac{width}{step} \right\rceil$
$pt \leftarrow P$
for
  $(i, j) in\ range\ ((n_h - 1.0), (n_w - 1.0))$
  do
    $per_h \leftarrow \left\lceil \frac{i}{n_h} \right\rceil, per_w \leftarrow \left\lceil \frac{j}{n_w} \right\rceil$
    $P_s \leftarrow resizePolygon(P, per_w, per_h)$
    $pt \leftarrow connectPathToPolygon(pt, P_s)$
Insert region center point as end path point to pt
return{pt, h_{max}}

Algorithm 2: Segmented Traversing Path algorithm

Input: Path
$P: \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$,
$Number\ of\ segments\ S: Number$
Output: bp whole path points with break points

$procedureSegmentedPath\ (P)$
  $bp \leftarrow \{\}$
  For each line$\in Pdo$
    $q \leftarrow\ Divide\ line\ into\ S\ segments$
    $Add\ first\ point\ of line\ into\ bp$
    $Add\ generated\ q\ points\ into bp$
    $Add\ last\ point\ of line\ into\ bp$
  Return bp

5.2. Segmented Zig-Zag Path Traversal

When a drone reaches maximum speed, it may cause the drone to drift away of actual path. This finding has been explained in section 7. The algorithm generates several points to break down the path which optimize tracking the path correctly. See Figure 5.

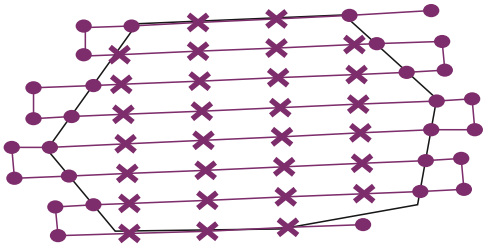


Figure 5. Segmented Zig-zag path traversal.

Segmented traverse path Algorithm 2 needs zigzag path (P) in order to divide each horizontal line in that path into (S) segments. That generates (S−1) points on each horizontal line. The algorithm inserts these points between the begin and end of the first horizontal line. This process repeated for each line in the traversal path.

Algorithm 3: Curved Traversing Path algorithm

Input: Path
$P: \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$,
$Shift\ Percentage\ S: Numberwhere - 2 \le S \le 2$
$Step\ Step: Number$
Output: cp whole path points with shifted points

$procedureCurvedPath\ (P)$
  $cp \leftarrow \{\}$
  For each line$\in Pdo$
    $q \leftarrow\ Divide\ line\ into\ 2\ segments$
    $q_y \leftarrow q_y + S \times Step$
    $Add\ first\ point\ of line\ into\ cp$
    $Add\ shifted\ q\ points\ into cp$
    $Add\ last\ point\ of line\ into\ cp$
  Return cp

5.3. Curved Path Algorithm

In case of wind, the drone could miss the correct path as the wind force may shift the drone off its path. The algorithm generates a curved path to compensate against the wind force direction since wind causes the drone to drift away from the path. Figure 6 shows the wind direction and a curved path against wind.

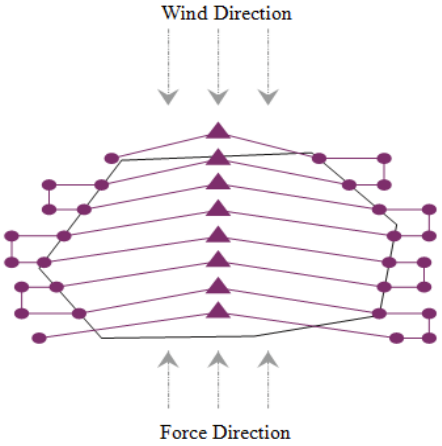


Figure 6. Wind force compansting path traversal.

Curved Path Algorithm 3 generates shifted points into the wind, which lets drones fly against wind to correct the actual path it follows.

$$step = \frac{\omega_{scene} \times h_{max}}{2f} \quad (2)$$

Algorithm also need zig-zag path (*P*), *step* (Equation (2)) [7] which is half scene width related to h_{max}, and the percentage (*h*) of shifting the point either negative or positive. In each iteration, algorithm find the midpoint and shift it up or down according to value of *h* which is between−2 and 2. Each midpoint of horizontal lines shifted in oppsite direction of the wind. Figure 6 shows wind direction and midpoint shifting toward wind.

6. ASAF Object Detection

A fundamental part of ASAF is the process of object detection. This process is relevant to the traversal height and then the object detect ability. Thus the developed traversal algorithms take the flight height into consideration as explained above. Indeed it is not of our scope to came up with detection technique and thus for the purpose of the system proof of concept, we limit the scope of target detection to AprilTag detection.

ASAF camera subsystem consists of a downlooking camera mounted on flying agents. The objects of interest are special visual markers (AprilTags) placed on the ground. The detection software can recognize the position of the marker and report the information back to the ASAF system to take a proper action.

AprilTag is a 2D label designed to encode between 4 to 12 bits [36]. The library for detection of AprilTags is freely available under the Berkeley Software Distribution (BSD) license it works with a plethora of tag families with different amounts of information encoded and resistance to sensor noise. A recommended family is Tag16h5 which allows to distinguish between 30 tags, Figure 7 shows few examples.

Figure 8 shows the geometry of the camera's field of view (fov). The agent's height (h) determines the dimensions of the width of the scene visible to the camera. It is obvious that as the height decreases, the visible width by the camera decreases as well. In [7], we have presented some equations relating these optical relationships. For example, for a specific camera subsystem, we calculated that at a height of 20m, the camera can view a rectangular area with dimensions of 24m by 18m.

In case of AprilTag markers, for a given camera pixel resolution, the detection of tags will depend on the resolution of the size of the printed AprilTag. In [36], authors show that the AprilTag with width of 0.5m and 50-pixel resolution is detectable at 21m.

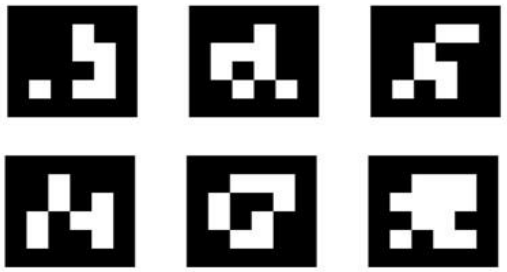


Figure 7. Examples for Tag16h5 [36].

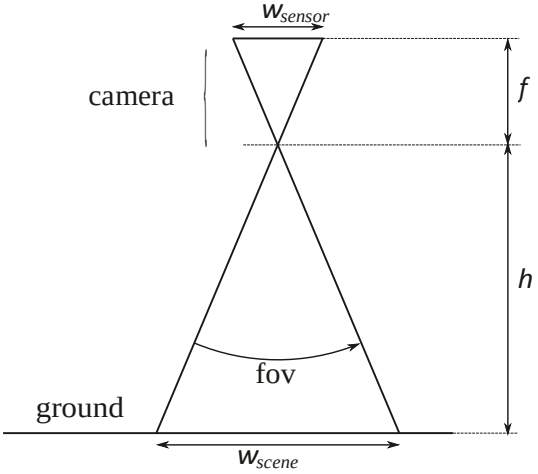


Figure 8. Camera parameters.

6.1. Traversed Area Regularity

Figure 9, shows some experimental results conducted using a camera of 640×480 pixels resolution and a fov= 60°. The tag size is 0.5*m* ×0.5*m*. The blue curve describes the theoretical limit for detection height and the dotted curve shows the actual height for detection as experimented. Distance estimation error rises with

the distance from the pattern. For distances under 30m, the error is sufficiently low. For distances above 40m, the detection fails completely, which defines the maximal detection height for the setup used in simulation. The size of the pattern in the camera image is about 20 pixels, which denotes the minimal detection size in image under best conditions.

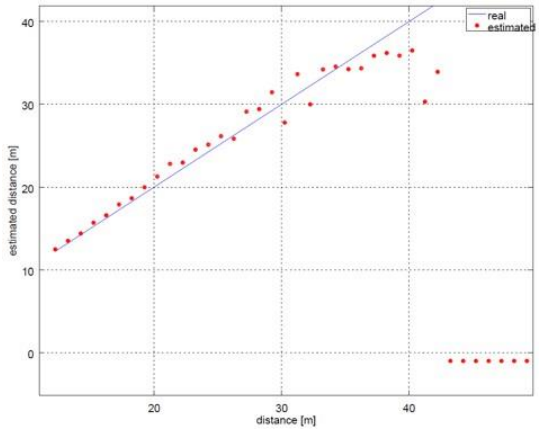


Figure 9. Camera subsystem height simulation results.

7. Navigation along the Trajectories

The basic mode of a UAV from user's point of view is flying along the trajectory specified by GPS coordinates. In the trivial case, it includes flight to a single destination position. A trajectory is a list of waypoints. Each waypoint consists of the geographical latitude, longitude, and altitude (above sea level). The navigation algorithm (autopilot) controls the UAV to fly the shortest way possible to the first waypoint on the trajectory, which is ideally a straight line [37]. A pre-set maximal velocity constraint limits the flight speed. The ideal flight velocity should be constant during the flight to the actual trajectory point, except an acceleration initially and a final deceleration to be able to reach the last point with zero velocity.

In real conditions, errors happen in the trajectory execution, due to:

- The limited precision of the conventional GPS localization.
- External Environmental effects (e.g. wind).
- Limited precision of the regulator tuning (e.g. drone mechanics).

Those effects will cause errors between designated and executed trajectory, as is clear from Figure 10. In this figure, we present experimental results based on recorded positioning data of actual flight trajectory (20 minutes) with respect to the *x-y* plan. Because performing experiments under stronger wind is out of scope, we can say that effects of external environmental effect, sensor noise and others will be negligible compared to the error due to GPS localization. Short-term (in units to tenths of seconds) variation of the GPS- estimated position, considering unobstructed view to sky (which is typically true for a flying UAV), may be below 1m. From practical experience, the real error is often below 5m [36].

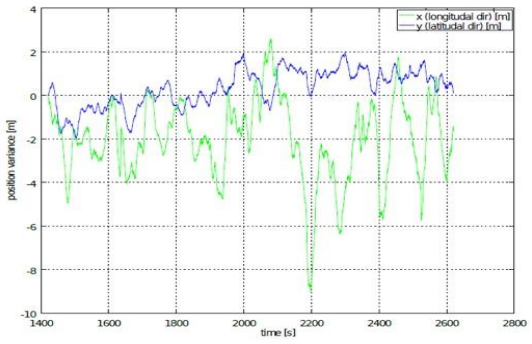


Figure 10. Error measurement during 20 minutes experiments.

Considering the previously mentioned minimal error and estimated position variance of the UAV, the autopilot cannot aim to reach an exact waypoint position (which would not be precise anyway), so the navigation to that point stops whenever the UAV reaches the vicinity around the actual waypoint. According to common GPS variance, this region should have a radius of about 0.1 to 0.5 m.

Most of drone flight parameters (except multirotor inner dynamics) are adjustable to achieve desired performance. These parameters include maximal allowed velocity, maximal acceleration, and gains in the position/velocity regulator. When agents should follow a trajectory more precisely, they fly at lower speeds to gain more control than when flying at higher speeds. This observed behavior coupled with the observation that flight velocity cannot be constant during the flight: This involves braking and speeding up near trajectory waypoints. We further explain this aspect in the following subsection.

7.1. A Real UAV Flight Behavior

For further clarifications in our investigation of UAV behavior considerations with respect to navigation errors incurred to the flying agent, we have executed a few experiments to draw conclusions. We designed a trajectory of four waypoints arranged to execute a square path with a 10 m edge length. We repeated the experiment twice at speeds of 1 and 5m/s.

Figures 11 and 12 shows the actual flight path where we see errors in each corner of the square edges, the precision to follow a straight line between the square's corners worsens as the speed of the drone increases.

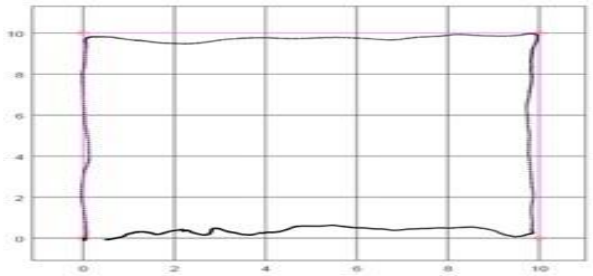


Figure 11. x-y position error for square trajectory experiment at 1m/s maximal speed.

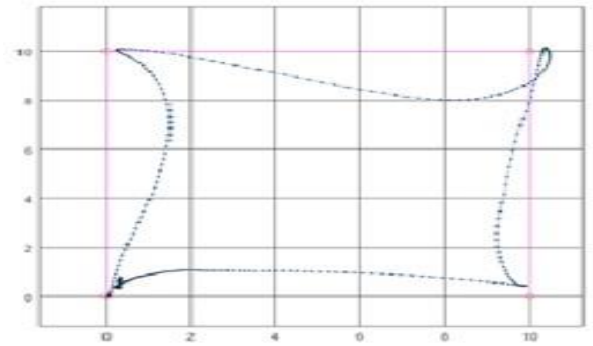


Figure 12. x-y position error for square trajectory experiment at 5m/s maximal speed.

Figures 13 and 14, enforces our observation that UAVs change their speeds during path execution.

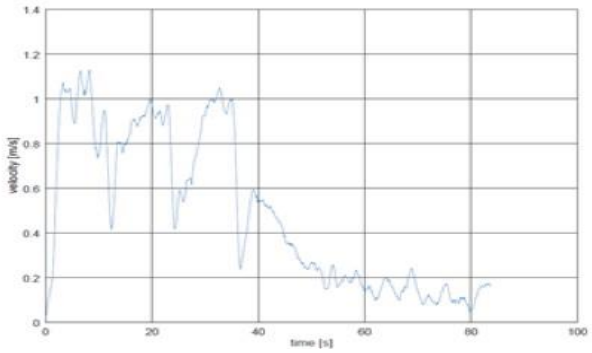


Figure 13. Flight data on speed variation during the execution of square trajectory experiment at 1m/s maximal speed.

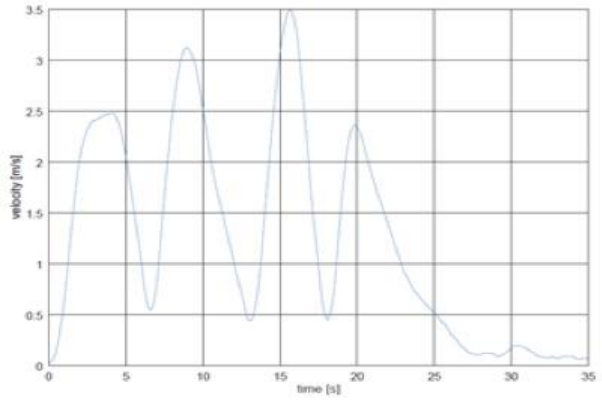


Figure 14. Flight data on speed variation during the execution of square trajectory experiment at 5m/s maximal speed.

During the traversal of an edge between 2 waypoints

1. The UAV accelerates gradually at the beginning then decelerates at the end of the edge.
2. The maximal speed may not be reachable especially with the higher speed scenario i.e., at 5m/s.
3. Speed transition is smoother at higher maximal speed.

8. Problem Simulation

We ran a simulation using Unity3D Version 5.6.1 [40], to evaluate the traversal algorithms. Further, we studied the impact of various parameters on the coverage and overlap. We fed the simulator with information about the traversals. Coverage shows the percentage of the area that the agent has traversed [7]. Overlap shows the amount of area that agents traversed (scanned) more than once.

The architecture of the simulator has two main components:

1. UAV package, which controls the flight of the UAV and generation/traversal of the path.
2. Scene Manager object handles simulating environment options and user inputs.

We used two levels of the UAV package:

1. Stabilizing level to handle UAV stabilization algorithms and PID controllers.
2. Trajectories level to handle the logic of the circuit (path) to follow.

Scene Manager has five manager scripts, namely Input Manager, File Manager, Drone Manager, Camera Manager and Wind Manager. Input Manager controls the configuration of the traversal path input (File or Manual based), traversal type (Zigzag or Spiral), overlap percentage, frames per seconds and many other configurations. Using Shoelace Formula, we calculate the area covered by the traversal [39].

$$A = \frac{1}{2}[\sum_{u-1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{u-1}^{n-1} x_{i+1} y_i - x_1 y_n] \qquad (3)$$

File Manager [40] stores the outcome of the traversal of the UAVs. File Manager also controls how often to store data like speed of the drone, position of the drone, coverage percentage, overlap percentage, screenshots, ..., etc.

Drone Manager [40] manages and instantiates drones. Camera Manager manages cameras in the scene. In our simulation, we only use the top view camera of the scene from the UAV.

Wind Manager [40] helps us to simulate and study the impact of wind on the UAV path traversal and area coverage. The Wind Manager controls the Speed, direction, and the zone of the wind.

Poly Ops class handles trajectory lines and overlapping boundaries calculations. This class depends on the Slope-Intercept equation to calculate the necessary information. The Waypoint Circuit script automatically applies a curve between any two points.

We do not need this feature in our traversal and thus we read each point twice into the circuit. We calculate the distance between two trajectory lines using the point (0,b) where b is the y-intercept variable from the slope-intercept equation for one of the trajectory lines as our starting point:

$$y = mx + b \quad (4)$$

We calculate coverage percentage with the help of Unity's built-in function Physics. OverlapBox().

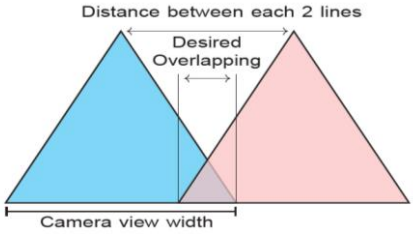


Figure 15. Desired overlapping.

Figure 15 gives information about the desired overlapping regions. The traversal lines of an UAV are separated by a distance, which could cause the visible width of the camera (camera view width) to overlap. We determine the traversal path based on the desired overlap percentage. Increasing the desired overlapping is achieved by reducing the flight height, which is also considered in traversal algorithm. The OverlapBox() function captures the field of view of the UAV based on the filling of the region.

The Desired overlap percentage is based on the boundaries and to check whether the captured point is above the current lower boundary and below the current upper boundary. Such an identification is based on the slope-intercept equation [38] as;

$$(LB_m \times P_x + LB_b) < P_z < (UB_m \times P_x + UB_b) \quad (5)$$

Where LB_m, LB_b, UB_m, UB_b are the slope and *y*-intercept of the current lower bound line equation and the slope and *y*-intercept of the current upper bound line equation respectively. While P_x and P_z are the *x* and *z* values of the point to be checked.

9. Results and Discussions

We studied the outcome of the experiment based on several factors like: coverage percentage, overlap percentage, number of turns, segmented paths, and zigzag or spiral algorithm. Regular polygons are of equal angles and equal edges. Irregular polygons have different and uneven angles and edges. Coverage shows what is the covered area's percentage compared to the whole area. Overlap shows the amount of area that agents traverse more than once.

Impact of desired overlap on coverage: In this experiment, the UAV traversed two areas with 100% and 86% regularities, respectively. The maximum speed of the UAV tops at 5m/s. Figure 16 shows that with a regular area traversed, the coverage is close to 100% when the desired overlap percentage is 50%, while irregular areas need to increase the desired overlap to reach close to 100% coverage. This observation can be attributed to the fact that irregular areas requires more turns to be covered fully. This probably shows limitation of our algorithms and is an issue needs to be addressed as well.

Impact of speed on coverage: As it is noticed in our results in section 7 that, as the speed of the drone increases, accuracy decreases, and thus localization errors increase. Thus the drone speed has significant impact on the search coverage. In order to do examine the limitation of our system, we calculated the coverage and the overlapping percentage using PolyOps class described in section 8. Figure 17 also shows the results that increasing speed causes the coverage percentage to drop significantly. The performance degradation is more obvious in the case of irregular areas. This outcome results point out to the fact that the performance will be determined by the average speed parameters.

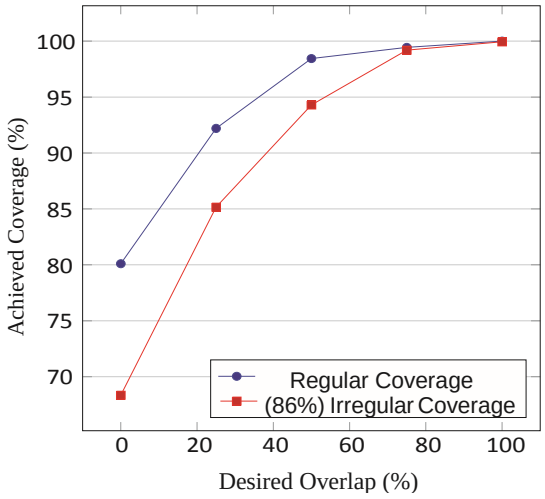


Figure 16. Desired overlap vs achieved coverage.

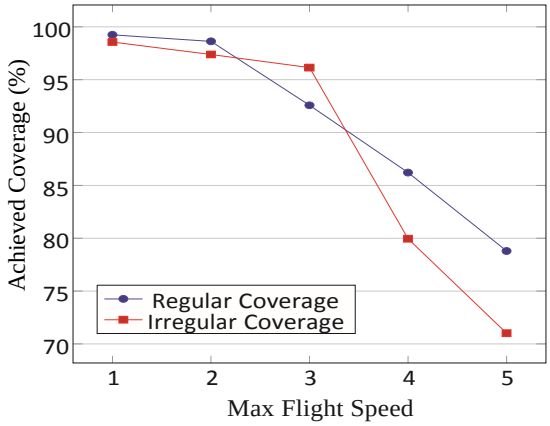


Figure 17. Impact of speed on coverage and achieved overlap.

Impact of traversal algorithms: To reduce the localization error, we have developed an algorithm to generate segmented trajectory based on the conclusion obtained in [7] that setting the drone to fly between closer waypoints decrease the error. Segmented paths

are like normal paths but with extra points on each traversal line, Figure 9 to conform to this conclusion, an experiment to appreciate the impact of segmentation is designed, we set UAV maximum speed to 5 m/s and the desired overlap percentage was set to 0%. Table 2 shows that using segmented paths dose not only increases the coverage percentage but also it increases the traversal time. This show that path segmentation limitation. In addition to that Table 2 shows also the performance of spiral algorithm, which outperform the normal zigzag in terms of coverage and outperforms the segmented algorithm in terms of traversal time, this might indicate that spiral algorithm can be optimized further for optimal performance.

Table 2. Impact of the segmented paths on traversal time.

	Normal Zig-zag	**Segmented**	**Spiral**
Coverage	80%	91%	83%
Time (seconds)	181	240	205

Along different regularities, we study the impact of different parameters. Desired overlap percentage impacts both the achieved overlap and coverage percentages. Overall, there is clearly a relationship between the coverage percentage and the traversal time. Using spiral algorithm and/or segmented paths do increase the coverage percentage but also the traversal time. Optimization of these algorithms could further enhance the resultant coverage, overlap, and time duration. The simulation experiment does not have battery constraint. But, in real experiments, battery life could be a deciding factor on which approach to use as the traversal time depends on the battery life.

We study the impact of spiral and zigzag algorithms on coverage percentages and traversal times. UAV's maximum speed was set to 5 m/s and the desired overlap percentage was set to 0%. UAVs traverse the same area using zigzag and spiral algorithm. We compare coverage percentage and time taken between them. Table 3 shows that there is about 10% improvement in the coverage percentage when traversing using spiral path but at the cost of 50% extra time as it require the drone to handle more turns than the zig-zag traversal.

Table 3. Traverse algorithm.

	Zig-Zag	**Spiral**
Coverage	71%	83%
Time (seconds)	140	205

Along different regularities, we study the impact of different parameters. Desired overlap percentage impacts both the achieved overlap and coverage percentages. Overall, there is clearly a relationship between the coverage percentage and the traversal time. Using spiral algorithm and/or segmented paths do increase the coverage percentage but also the traversal time. Optimization of these algorithms could further enhance the resultant coverage, overlap, and time duration. The simulation experiment does not have battery constraint. But, in real experiments, battery life could be a deciding factor on which approach to use as the traversal time depends on the battery life.

10. Conclusions and Future Work

In this work, we have reported on our progress. We focused on the agent's traversal pattern as it is the most essential task the agent should execute in the search mission. As for an application such as a search and find application, agents should scan each point in the traversed area at least once. Otherwise, UAVs might miss some objects. Success to cover a spot is a function of many factors such as: the number of captured frames per second, the height of the camera, the speed of the drone, the number of waypoints in an area, the width of the cameras field of view, GPS error... etc. In this work, we have investigated most of these aspects and related them using available means of simulation under different path traversal strategies or/and several factors. The aim was to find optimization problems for further improvements. We identified the limitations relevant to each traversal algorithms. In fact the new traversal algorithm developed such as segmented and spiral has shown performance improvement in many aspects. In future work, we aim to work on algorithms that produce optimized paths. We will explore whether we can exploit cognition techniques to estimate the error and suggest proper reactions.

Acknowledgment

This research work was funded by Makkah Digital Gate Initiative under grant no (MDP-IRI-12020). Therefore, the authors gratefully acknowledge technical and financial support from the Emirate of Makkah Province and King Abdelaziz University, Jeddah, Saudi Arabia.

References

[1] Abdulrahim M., "Flight Dynamics and Control of an Aircraft with Segmented Control Surfaces," *in Proceedings of 42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, pp. 128, 2003.

[2] Andersen H., "Path Planning for Search and Rescue Mission Using Multicopters," Master's Thesis, Institutt for tekniskkybernetikk, 2014.

[3] Araujo J., Sujit P., and Sousa J., "Multiple Uav Area´ Decomposition and Coverage," *in Proceedings IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Singapore, pp. 30-37, 2013.

[4] Babka D. *Flight Testingi A Simulation Based Environment*, California Polytechnic University, 2011.

[5] Barnawi A. and Al-Barakati A., "Design and Implementation of a Search and Find Application on A Heterogeneous Robotic Platfrom," *Journal of Engineering Technology*, vol. 6, pp. 235-239, 2017.

[6] Barnawi A., Al-Barakati A., Khan A., Bajaber F., and Alhubaiti O., "A Proposed Architecture for A Heterogeneous Unmanned Aerial Vehicles System," *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 7, no. 3, pp. 119-126, 2018.

[7] Barnawi A., Alharbi M., and Chen M., "Intelligent Search And Find System for Robotic Platform Based on Smart Edge Computing Service," *IEEE Access*, vol. 8, pp. 108821-108834, 2020.

[8] Ben L., Johnson E., and Vachtsevanos G., "Vision-Based Navigation and Target Tracking for Unmanned Aerial Vehicles," *IEEE Robotics and Automation Magazine*, vol. 1, pp. 63-71, 2006.

[9] Berndt J., "Jsbsim: An Open Source Flight Dynamics Model inc++," *in Proceedings AIAA Modeling and Simulation Technologies Conference and Exhibit*, Providence, pp. 4923, 2004.

[10] Bertuccelli L. and How J., "Search for Dynamic Targets with Uncertain Probability Maps," *in Proceedings American Control Conference*, Minneapolis, pp. 6, 2006.

[11] Cabreira T., Franco C., Ferreira P., and Buttazzo G., "Energy-aware Spiral Coverage Path Planning for Uavphotogrammetric Applications," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3662-3668, 2018.

[12] Chen R., Xu N., and Li J., "A Self-Organized Reciprocal Decision Approach for Sensing Coverage with Multi-Uav Swarms," *Sensors*, vol. 18, no. 6, pp. 1864, 2018.

[13] Choi H., Brunet L., and How J., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912-926, 2009.

[14] Condomines J., *Nonlinear Kalman Filter for Multi-Sensor Navigation of Unmanned Aerial Vehicles*, Elsevier, 2018.

[15] Flint M., Polycarpou M., and Fernandez-Gaucherand E., "Cooperative Control for Multiple Autonomous Uav's Searching for Targets," *in Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, pp. 2823-2828, 2002.

[16] Garcia R. and Barnes L., "Multi-Uav Simulator Utilizing X-Plane," *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 393-406, 2009.

[17] Guerrero J., Escareno J., and Bestaoui Y., "Quadrotor Mav Trajectory Planning in Wind Fields," *in Proceedings of IEEE International Conference on Robotics and Automation*, Karlsruhe, pp. 778-783, 2013.

[18] Hajiyev C., Soken H., and Vural S., *State Estimation and Control for Low-cost Unmanned Aerial Vehicles*, Springer, 2015.

[19] Held M. and de-Lorenzo S., "On the Generation of Spiral-Like Paths within Planar Shapes," *Journal of Computational Design and Engineering*, vol. 5, vo. 3, pp. 348-357, 2018.

[20] Hsieh M., Cowley A., Keller J., Chaimowicz L., Grocholsky B., Kumar V., Taylor C., Endo Y., Arkin R., Jung B., and Wolf D., Sukhatme G., MacKenzie D., "Adaptive Teams of Autonomous Aerial and Ground Robots for Situational Awareness," *Journal of Field Robotics*, vol. 24, pp. 991-1014, 2007.

[21] Liu C., McAree O., and Chen W., "Path Following for Small Uavsin the Presence of Wind Disturbance," *in Proceedings of UKACC International Conference on Control*, Cardiff, pp. 613-618, 2012.

[22] Liu X., Fan J., Mao J., and Ye F., "A Low-Power Self-Service Bus Arrival Reminding Algorithm On Smart Phone," *The International Arab Journal of Information Technology*, vol. 16, no. 2, pp. 260-264, 2019.

[23] Meng W., Hu Y., Lin J., LinF., and Teo R., "Ros+ Unity: An Efficient High-Fidelity 3d Multi-Uav Navigation And Control Simulator In Gps-Denied Environments," *in Proceedings of IECON-41st Annual Conference of the IEEE Industrial Electronics Society*, Yokohama, pp. 002562-002567, 2015.

[24] Merino L., Caballero F., Forssen P., Wiklund J., Ferruz J., Martihez-de-DiosA J., Moe A., NordbergK., and Ollero A., *Advances in Unmanned Aerial Vehicles*, Springer, 2007.

[25] Merino L., Caballero F., Mart´ınez-de Dios J., Ferruz J., and Ollero A., "A Cooperative Perception System for Multiple Uavs: Application to Automatic Detection of Forest Fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165-184, 2006.

[26] Mir I., Eisa S., and Maqsood A., "Review of Dynamic Soaring: Technical Aspects, Nonlinear Modeling Perspectives and Future Directions," *Nonlinear Dynamics*, vol. 94, pp. 1-28, 2018.

[27] Ost G., "Search Path Generation with Uav Applications Using Approximate Convex Decomposition, MSC Thesis, Linköpings Universitet, 2012.

[28] Otte M., Kuhlman M., and Sofge D., "Competitive Target Search with Multi-Agent Teams: Symmetric and Asymmetric Communication Constraints," *Autonomous Robots*, vol. 42, no. 6, pp.1207-1230, 2018.

[29] Pennycuick C., "The Flight of Petrels and Albatrosses (Procellariiformes), Observed

inSouth Georgia and Its Vicinity," *Philosophical Transactions of the Royal Society B*, vol. 300, no. 1098, pp. 75-106, 1982.

[30] Polycarpou M., Yang Y., and Passino K., "A Cooperative Search Framework for Distributed Agents," *in Proceedings of the IEEE International Symposium on Intelligent Control*, Mexico City, pp. 1-6, 2001.

[31] Richardson P., "Upwind Dynamic Soaring of Albatrosses and Uavs," *Progress in Oceanography*, vol. 130, pp. 146-156, 2015.

[32] Shaw A. and Barnes D., "Landmark Recognition for Localisation and Navigation of Aerial Vehicles," *in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, pp. 42-47, 2003.

[33] Sorton E. and Hammaker S., "Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle Using Flightgear," *in Infotech@ Aerospace*, pp. 70-83, 2005.

[34] Viguria A., Maza I., and Ollero A., "Distributed Service based Cooperation in Aerial/Ground Robot Teams Applied to Fire Detection and Extinguishing Missions," *Advanced Robotics*, vol. 24, no. 1-2, pp. 1-23, 2010.

[35] Wakefield E., Phillips R., Matthiopoulos J., Fukuda A., Higuchi H., Marshall G., and Trathan P., "Wind Field and Sex Constrain the Flight Speeds of Central-Place Foraging Albatrosses," *Ecological Monographs*, vol. 79, no. 4, pp. 663-679, 2009.

[36] Wang J. and Olson E., "Apriltag2: Ecientand Robust Ducial Detection. In Intelligent Robots and Systems (IROS)," *in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, pp. 4193-4198, 2016.

[37] Webpage., Pixhawk project webpages. https://pixhawk.org, Last Visited, 2021.

[38] Webpage., Mathportal webpage, distance and midpoint calculate. https://www.mathportal.org/calculators/analyticgeometry/distance-and-midpoint-calculator.php, Last Visited, 2017.

[39] Webpage., Wikipedia website, shoelace formula. https://en.wikipedia.org/wiki/Shoelace formula, Last Visited, 2017.

[40] Webpage,. Unity website. https://unity3d.com, 2017-11-30, Last Visited, 2017.

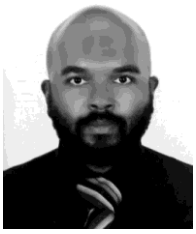
[41] Webpage., Xplane 11 flight Simulator-More Powerful. Made Usable. https://www.x-plane.com, Last Visited, 2021.

[42] Wenjing C. and Shenghong X. "Comparison of Multi-Uav Cooperation Architectures," *in Proceedings of 3rd International Conference on Information Management*, Chengdu, pp. 500-505, 2017.

[43] Wenjing C. and Shenghong X., "Workflow Based Multi-Uav Cooperation Architecture," *in Proceedings of 3rd International Conference on Information Management*, Chengdupp. 496-499, 2017.

[44] Yang Y., Minai A., and Polycarpou M. "Decentralized Cooperative Search by Networked Uavsin an Uncertain Environment," *in Proceedings of the American Control Conference*, Boston, pp. 5558-5563, 2004.

[45] Zhang K., Collins E., and Shi D., "Centralized and Distributed Task Allocation in Multi-Robot Teams via a Stochastic Clustering Auction," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 7, no. 2, pp. 1-22, 2012.



Ahmed Barnawi received the M.Sc. degree from the University of Manchester (UMIST), U.K., in 2001, and the Ph.D. degree from the University of Bradford, U.K., in 2005. He is currently a Professor of information and communication technologies with the Faculty of Computing and IT (FCIT), King Abdulaziz University (KAU). He is the Managing Director of the KAU Cloud Computing and Big Data Research Group. He acted as an Associate and Visiting Professors in Canada and Germany. He is an Active Researcher with good research fund awards track. He published near to 100 articles in peer reviewed journals. His research interests include big data, cloud computing, future generation mobile systems, advanced mobile robotic applications, and IT infrastructure architecture.

Marwan Alharbi received the Master of Science degree from the University of Denver, USA, in 2014. He is currently a Lecturer of information technology with the Faculty of Computing and IT (FCIT), King Abdulziz University (KAU). His research interests include machine learning, AI and blockchain, and the IoT.