IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 5, 2020, accepted April 21, 2020, date of publication May 11, 2020, date of current version June 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993727

Intelligent Search and Find System for Robotic Platform Based on Smart Edge Computing Service

AHMED BARNAWI[1], MARWAN ALHARBI[1], AND MIN CHEN[1,2], (Senior Member, IEEE)
[1]Faculty of Computing and Information Technology, King Abulaziz University, Jeddah 21589, Saudi Arabia
[2]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Min Chen (minchen2012@hust.edu.cn)

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant RG-1-611-39.

ABSTRACT In recent years, artificial intelligence has been widely used in the field of robotics. However, these robot-related tasks are difficult to migrate from the cloud to edge nodes due to the large computing and storage resource requirements. In this project, we develop a platform for a heterogeneous robotic system. The platform is built to facilitate the development of advanced robotic applications with minimal human interactions, where search and find system traversal is the main application. To support robots performing tasks in near field in real time, in this paper we introduce traversal and task division algorithms which are introduced to perform a cooperative search mission by a group of robotic agents to achieve intelligent search and find as an edge service. We evaluate the performance against the algorithm's parameters using data obtained in controlled field experiments. The aim was to identify and study some key performance parameters impacting the traversal function in the application.

INDEX TERMS Unmanned aerial vehicles, UAV controller, edge computing, embedded systems, multilayered architecture.

I. INTRODUCTION

Searching process involves the routine and systematic activity of traversal where the searching agent must visit and sense each point in the assigned area. Thus regardless to the detection technology, this function of the system should be treated with great care. The design of a sound traversal algorithm is a nontrivial process as it involves various factors related to the searched area geometry, the vehicle mechanical fault tolerance, the camera optical specifications and the environmental parameters. On the other hand, if we aim to employ more than one agent in the search mission, traversal process must take into account the fairness to distribute the task among the agents based on their capabilities.

The Advanced Search And find System (ASAFS) is an application developed to improve automatic search and find processes of objects in a vast areas that consists of multiple robotic agents. This application is being deployed over a heterogeneous state-of-the-art testbed developed by our research group named Multiple UAV Experimental Testbed (MUAVET). In this application, a group of robotic agents are assigned to take part in the search process. Those agents are to perform coordinated search missions. The search missions are initially planned, controlled and monitored at the base station (BS) through developed interfaces. However, when the robot agents perform some complex search tasks, the base station cannot load the resources required by the intelligent algorithm, so it will request from the cloud computing platform, which will greatly affect the execution efficiency of the search task.

Edge computing is a near-device network architecture with real-time data processing and lower latency. Robot groups often need to work together to perform search tasks, which requires highly responsive network solutions. Therefore, combining edge computing and artificial intelligence technology, this paper introduces a traversal algorithm and its components which constitute the main part of the Search Planner (SP) part of the system. The algorithm is designed at the edge of the network as an intelligent edge service to achieve highly available search. The SP program resides in

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

the base station (or the controlling agent),it calculates (and recalculates) the agents path during the cooperative the multi agents search campaign.

The main motivation of this work is to introduce a region traversal and task partitioning algorithm that can run on edge networks for a group of robotic agents performing a cooperative mission to quickly and intelligently search and find objects in a predefined area. We also analyze aspects of the system performance based on the developed algorithms on our testbed. In order to study the performance of the proposed algorithm, an analysis of the trajectories data of the flying drone is being conducted in order to quantify actual error resulted from GPS and mechanical aspects of the robotic agents. The aim is to identify optimization problems in the path traversal and suggesting solutions for optimized version of the algorithm. Finally, we conduct some real life experimental scenarios to demonstrate system operation and integration.

The structure of this paper is as follows; Section II is related work. Section III highlights the characteristics of ASAFS architecture. Section IV introduces the system components, testbed and setup. Section V presents the traversal algorithm and its components. Section VI 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.

II. RELATED WORK

Robotics is a domain that has been contributing to science since the seventies. It has driven the large scale manufacturing industries and underpinning the new generation of automation technologies [1]. This branch of technology will experience the highest growth in near future to address the design challenges for multi agents based design methods. There are various general purpose methodologies developed during the last decade, for example, Prometheus, Gaia, MaSE, Ingenias, etc. [2]. Still, the multi-robot system adds interactions between agent software and autonomous robots. In particular, agents should be built to represent physical robots in a management system [3].

In spite of the availability of many design methods, cognition is still under extensive consideration. The development of multi-robotic systems with cognitive abilities is largely studied in the swarm robotics field. In [4], the issue is tackled explicitly in the context of swarm robotics, advocating the need for a ''swarm engineering''. As a consequence, various design methodologies have been proposed, but are often somehow limited in their scope [5]–[8] and [9].

The process of terrain coverage using a single robot is to be performed using various traversal algorithms. In case of multiple robots being used to cover the terrain, the given area should be decomposed into multiple polygons. Polygon decomposition has been shown to be NP-hard [10]. Such polygon decomposition could be performed using triangulations, convex decompositions, quadrilateralizations, etc [10].

Studies have also discussed covering nonconvex and non-simply connected polygons. This coverage will have higher processing complexity. Hert and Lumelsk [10] studied the process of terrain covering with multiple robots. The given polygon is considered to be convex and has no holes to partition into n polygonal pieces, where n represents the number of robots available. Polynomial-time polygon partition is performed using sweep lines and divide-and-conquer techniques.

Maza and Ollero [11] divided the given area into different parts based on the capabilities (speed of flight, altitude, sensitivity to wind conditions, etc) of the UAVs and their initial locations. Then the UAVs cover their designated areas in zigzag pattern considering how to minimize the number of turns. Real time operation is considered when designing the complexity of the proposed algorithms.

Traversing a region requires some preparations such as detecting polygon shape, converting and rotating in order to get less turns. Reference [12] starts with solving non-convex polygon by converting it into a convex shape. The second step is rotating the whole region in order to find maximum height between two points. The region gets divided using area distribution of each UAV according to its capabilities. If we have two UAVs with variant capabilities, the one with higher capabilities will be assigned to the larger area while the other will be assigned to remaining smaller area.

Duckham *et al.* in [13] provides algorithm to characterize a shape of a set of input points in a plane. For a given shape, it takes outer points, then generates sub-triangles and vectors in order to identify the shape. This characterization is a useful measure we utilized in our strategy for performance evaluation.

The main novelty of this project lies in the cooperative integration of possibly a large number of entities in tight cooperation in one single networked system with options to feature a distributed control. The work in this project is divided into stages, at this stage, the focus is on system development based on a default scenario and core system functionalities where fundamental system features are materialized.

In recent literature, the work on UAV platform in a project Dronemap [14] was centered around developing a modular cloud proxy that acts as a moderator between drones and users. The communication between drones, users and the Dronemap Planner achieved through the ROS and MAVLink protocol [15]. The idea is to offload computations to cloud, and extending ROS for cloud usage as a mean to that end. The Robot Operating System (ROS) is robotics middleware that includes collection of software frameworks for robot software development [16]. In [17] and [18], ROS limitations are listed, the authors identified few points that limited the scalability of the multiple robotic system mainly due to bandwidth and synchronization consideration with respect to the ROS design.

The work by Dronemap group addresses shortcoming in ROS by offloading the processes to the cloud rather than depending on onboard processing thus enhancing

computational process and connectivity between users and robotic agent. It is worth mentioning that a recent work by Lee *et al.* [19] describes also an ROS based platform for multi UAV Control testbed developed to demonstrate task handover from weary to capable agents.

With ease of configurability and customization in mind, our the in-house developed MAUVET platform, on the other hand, is concerned with autonomous robotic interaction and coordination to perform specific coordinated missions. It assumes that onboard computation empowered by NUC facility as sufficient enough to enable real time services and overcome any need for offline computation. This platform assumes full control on the robotic agents by the application with great flexibility, system scalability and flexibly in reachability via remote controlling over the Internet. This full control will pay off greatly in the process of development of mobile robotic applications required to examining different scenarios and methods.

From another perspective, we have paid a lot of attention to the GUI design and implemented our system GUI interface [20], [21] to ensure flexibility and customization in the system. The GUI overcome shortcomings in similar work that we have considered [22]–[24]. We tried to enforce full control by studying the use case and putting down a modular design of the system to enhance potential system features and scalability.

III. 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 precieved 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 camera's 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.

Considering the inefficient responsiveness under the cloud architecture, we design a lightweight traversal algorithm from the perspective of computing and storage resources, so that the algorithm can be deployed in edge devices. When the UAV performs the traversal task, it sends data directly to the edge nodes. This algorithm is used in combination with lightweight deep learning to achieve efficient search and traversal.

From a design point of view, the developed framework could be characterized based on similar formworks 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.

A. COOPERATION (AUTONOMOUS VERSUS NON-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 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).

B. 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.

C. COORDINATION (CENTRALIZED VERSUS DECENTRALIZED SYSTEMS)

The coordination of the agents in the 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.

D. FRAMEWORK COMPARISONS

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

IV. ASAFS: SYSTEM COMPONENTS

The basic terrain of the experimental area is an open air space with a good coverage by Global Navigation Satellite System (GNSS) signal. The area assumed is of circular shape with a diameter of 300 meters or a compact subspace of it. The whole area is covered by a wireless radio communication signal from the single WiFi access point (AP) placed at the center of the area. It is expected that a standard WiFi access point should be able to produce a signal with sufficient strength to cover the designated area. The access point is connected to the base station by Ethernet.

Fig. 1 shows the basic system components and the interfaces among them. Over the experimentation in open air area, the agents are deployed. The UAV agents are physically connected to the Base Station (BS) via the AP in a star topology. It is noticed that different UAV's are introduced including ground/aquatic level robotic vehicles while the operator is interacting with the system via Graphical User Interface (GUI).

A. ASAFS: SYSTEM HARDWARE

The main hardware components of the MUAVET testbed are the MUAVET Base station subsystems and the MUAVET Agents subsystems. The MUAVET Agent subsystem consists of onboard PC, Flight controller, GPS receiver, Camera, WiFi Communication Module and sensors. Functionalities of the UAV Control software are provided to other modules through two interfaces. A low level interface is realized as a standard TCP/IP socket. The socket interface which provides open and portable way of connecting server and user application is written in arbitrary programming language. A C++ Application Programming Interface (API) is built above the socket interface in form of C++ functions. ASAFS application code can exploit the UAV control functionalities simply by calling functions from this API. The socket interfaces are available on the server as well as on the UAV's onboard computer, so it

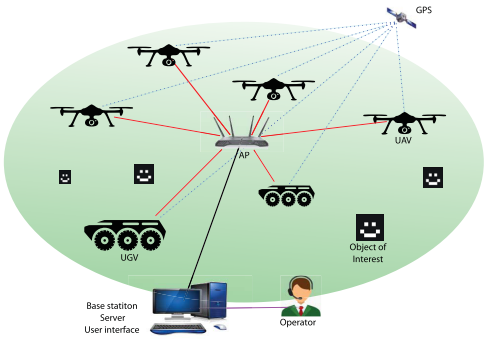


FIGURE 1. ASAFS general system configuration.

is up to user to decide the architecture of the top level system (e.g. centralized or distributed).

There is a proprietary communication link between the UAV and the BS server, this link is designed to reliably transfer data between agents and BS while controlling total data rate to avoid overloading the wireless communication channel. The communication server collects received data, provides them to modules on the Base Station and allows to control the agents from the Base Station.

The MAUVET agent is also equipped with a collision avoidance measures to mitigate the probability of mid-air agent collision. The basic collision avoidance mechanism is implemented based on continuous checking of the UAV actual positions and desired motion directions. Whenever there are two UAVs approaching each other in distance under a set threshold, collision condition is checked, which may be reacted upon and result in prompt stopping (hold position) of one or both UAVs. The collisions incidents are monitored and calculated at the server site, which collects all the necessary data about the UAVs' positions and directions. The server sends a special internal message to each UAV, telling it whether it is allowed to continue moving in the desired direction. For safety reasons, concerning possible message loss, this message is sent periodically. In [34], we have listed more details on the mechanism of collision avoidance.

B. THE DEFAULT SCENARIO AND TRAVERSAL PATTERNS

The default experimental setup of the ASAFS system is based on this scenario with flying drones only, however, the ground robotic agent are not incorporated in this scenario. Nonetheless, system design ensures that ground agents integration via system interfaces doesn't necessary impact the testbed's core hardware and software components. Fig. 2 shows a sketch of the default scenario. In this default scenario, the sequence of events can be summarized in the following steps:

1) Operator feeds the system parameters on the mission via a GUI.
2) The BS calculates a search plan and assign search areas to UAV 1, 2,3 and 4.
3) UAV 1, 2 and 3 carry on searching their assigned areas till either they find the object or assigned search is concluded.

TABLE 1. Framework comparisons.

Aspects	Mobile Robotic Systems		ASAFS
Cooperation	Autonomous	Non-autonomous	The ASAFS 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 ASAFS, the task division function is used to divide the task according to the robot capabilities. Unlike [29], [30], the ASAFS system task division function is based on centralized architecture.
	[25] [26] [27] [28]	[29] [30]	
Integration	Homogeneous	Heterogeneous	While ASAFS is a heterogeneous system that facilitates heterogeneity based on either mobile robot type (i.e. aerial, marine or ground) as in [31], [32], [33] or based on mobile robot capabilities as in [29]
	[25] [26] [27] [28] [30]	[29] [31] [32] [33]	
Coordination	Centralized	Decentralized	ASAFS 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 [25] [26] [27] [28] and the centralized system in [29]. While in [28] centralization is ensured through online guidance throughout the mission, the ASAFS system's centralized functions to plan the path for instant, can be offloaded to various system components.
	[28] [29]	[25] [26] [27] [30]	

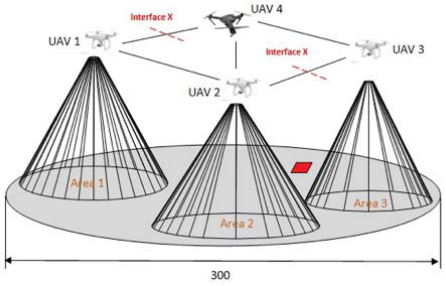


FIGURE 2. The default scenario in edge networks.

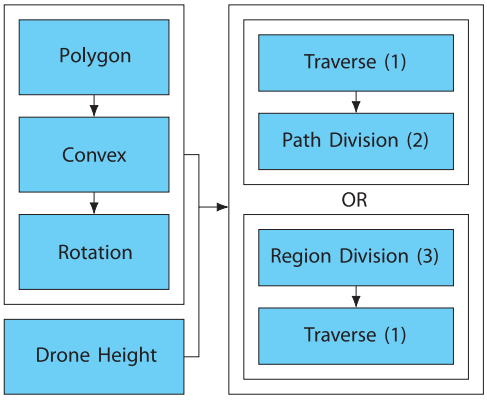


FIGURE 3. Algorithm components.

4) The system terminates the operation either when the objects are found or search is completed.

V. TRAVERSAL ALGORITHM COMPONENTS

An overview schematics of the search planning algorithm is shown in Fig. 3. To produce a path plan for a robot in multi robotic based search mission, the process takes few steps. The shape of a selected region (a polygon) might be classified as one of the following known polygon types; a convex, concave or self-intersect. As for traversing purpose, initially our traversal algorithm is designed to deal with convex polygons, thus in case of concave or self-intersect shaped polygon we would need to convert the shape into convex shape prior to further processing.

The conversion of a concave polygon is simply achieved by ignoring the inner points. In order to do so, the algorithm starts with polygon type detection and in case a concave shape is detected, the region will get converted into convex shape. However, if the polygon is identified as a self-intersected polygon, it cannot be converted into convex and thus the region will be rejected. After that, when the region is converted into convex polygon, a region gets rotated according to its minimum height, the objective is to reduce the number of turns while traversing. The next step will be to generate the waypoints for the traversal path associated with the searched area. In the following subsection we elaborate more on the traversal path components.

A. POLYGON TYPES AND CONVERSION

The first step is detecting the polygon type using *TypeDetection* algorithm. This algorithm works by checking on the inner angles of the shape, if any of shape's inner angles was greater than 180 degrees, then the shape could be classified as either a concave or self-intersect polygon. While the selfintersected polygon is out of our scope, another test is applied to detect the shape type. This test is run by ignoring one corner point of the polygon at a time then redrawing the shape edges then checking whether that corner point was inside the new polygon or not, if at least one corner point was found inside

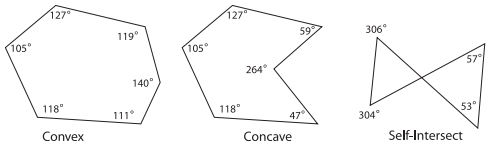


FIGURE 4. Non-convex polygons.

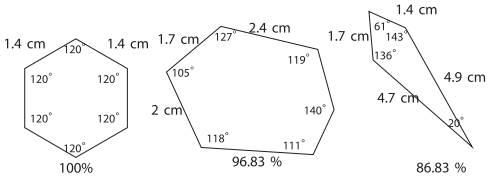


FIGURE 5. Polygon regularity.

the redrawn polygon, then its type would be identified as concave. If the polygon was found to be concave such as in Fig. 4 then it can be converted into convex by removing inner corner points. That can be done using *ConcaveToConvex* algorithm. Both *TypeDetection* and *ConcaveToConvex* algorithms complexity are $\mathcal{O}(n^2)$.

Another relevant classification of the polygon shape is to classify whether the shape is regular or irregular. In fact, traversing a regular shape provides better camera coverage and less computational effort, on the other hand, traversing irregular polygon shapes may result that some part of the region remain unvisited, thus not being sensed. This issue was investigated further in our publication [35]. For this regularity test, we implement our simple Algorithm (1), to measure degree of polygon irregularities. The algorithm uses regular polygon angle equation. The algorithm computes the difference between each angle and the average angle of the polygon inner angels. The sum of these differences of angles and edges are divided by number of edges to find the percentage of shape regularity. Fig. 5 shows different polygon with different regularities as an example.

B. ROTATION

Traversing a polygon in zigzag traversal pattern for example will result in the agent performing corner maneuvers pretty often while searching a predefined area. When a drone executes a turn, it would probably overshoot outside the traversed area to ensure inclusion of the points at the polygon's boundary. The tradeoff of this maneuver will be a wasting valuable battery and time resources of the flying agent. This problem will be exacerbated as the number of turn increases; therefore our design objective of our traversal path would be to reduce the number of turns.

In order to have less number of turns we would need to traverse a polygon horizontally where the polygon is mounted with minimum height (dimension). Fig. 6 shows how the number of turns is related to the polygon's traversal alignment. It is obvious that if the traversal alignment was made horizontally (north to south) the number of turns generated will be much more than in the case the area was traversed

Algorithm 1 Polygon Regularity Algorithm

Input: Polygon's points
$P : \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$,
$\Theta : \{\theta_1, \theta_2, \ldots, \theta_n\}$
Output: Regularity Percentage

procedure Regularity(P, Θ)
  $\overline{\Theta} \leftarrow$ Average angle
  $M \leftarrow$ Polygon perimeter
  $\overline{M} \leftarrow \frac{M}{n}$ ▷ Average perimeter
  $a \leftarrow 0$
  $m \leftarrow 0$
  for $i \leftarrow 1$ ***to*** n **do**
    $d \leftarrow \left|\overline{\Theta} - \theta_i\right|$
    $a = a + \frac{d}{360}$
    $d \leftarrow \left|\overline{M} - \sqrt{(x_{i-1} - x_i)^2 + (y_{i-1} - y_i)^2}\right|$
    $m = m + \frac{d}{M}$
  $r_{angle} = 1 - \frac{a}{n}$
  $r_{perimeter} = 1 - \frac{m}{n}$
  $r = \frac{r_{angle} + r_{perimeter}}{2}$
  return r

vertically (east to west). Thus instead of rotating the traversal alignment, we may be have to rotate the shape for simplicity. Therefore, to rotate the shape, we will need to rotate it around its center of mass. The center of the mass $(C_x; C_y)$ is calculated using the following equations:

$$C_x = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{1}$$

$$C_y = \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{2}$$

where A is the polygon's signed area, as described by

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i x_{i+1} - x_{i+1} y_i) \tag{3}$$

Moreover, finding the minimum height, which corresponds to the shortest vertical distance generated by a line perpendicular to the x-axis across the polygon, can be done by rotating the polygon by an angle θ_i. Since the irregular polygon has different and uneven angles and edges, we could not have an equation to determine the minimum height or in which angle should rotate to have minimal height. The angle increases in each iteration by a small amount in order to find minimum height. In theory, the iteration should be repeated until completing 360° to cover most of the possible positions. However, after reaching 180°, the rotated position outcome will give back the same results as in the first half cycle, thus rotating the shape by 180° is going to be enough to obtain the minimum height. Moreover, beside using on minimum height, if we had to consider the minimum width, which

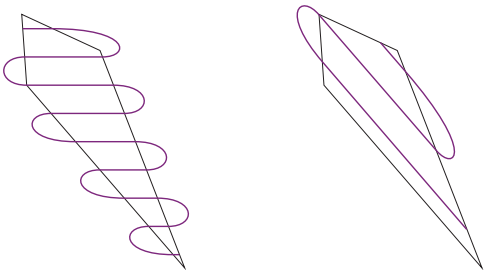


FIGURE 6. Same region but the number of turns unlike before and after applying minimum height.

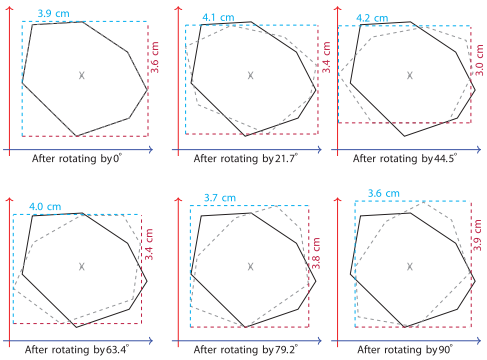


FIGURE 7. MinimumHeight method finds that the minimum height was 3.0 cm at angle 44.5°.

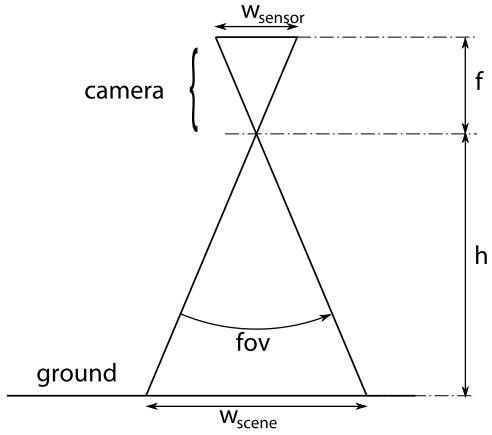


FIGURE 8. Scene width.

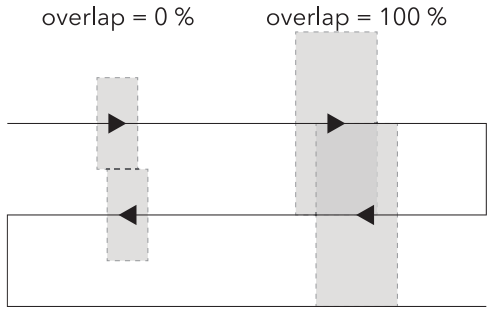


FIGURE 9. Drone height and overlap.

corresponds to the shortest horizontal distance generated by a line perpendicular to the y-axis across the polygon, we could limit the polygon rotation to 90° degrees instead. Fig. 7 shows an example of polygon rotation using both the minimum height and minimum length distances across the polygon.

It is obvious that, the number of iteration and the angular rotation increment are related. The accurate value of the minimum height would be obtained with smaller incremental values which will result on the higher number of iteration. In our example in Fig. 7 we have this increment set at 0.1 degrees.

C. ZIGZAG PATH TRAVERSE ALGORITHM

In our setup, the detection of objects of interest is achieved by a down-looking camera. Objects of interest are represented with special visual markers placed on the ground. Those objects' dimensions and camera specifications will be used to compute the maximum drone height within the range to detect the marked object. Fig. 8 shows the geometry of the problem where the camera visual specifications are related to the height of the robot.

The Traverse algorithm (2) requires region points P, pattern dimension ($W \times H$), and camera specifications C. As the drone traverse an area by following the waypoints, overlap of camera's view may occur between adjacent traversed lines. That is, the overlapped regions are scanned twice, in Fig. 11 the darker colored blue areas represent the overlapping regions that were visited twice in each direction as the robot traversed two adjacent lines. Overlapping is obvious to be a function of the height, the lower the height the less the overlapping. In fact overlapping consumes extra energy due to redundancy and may give conflicting results of the objects located in overlapped regions.

The output of algorithm (2) is the drone flying range (h_{min}, h_{max}) and the flying height based on the desired overlap between each two adjacent lines. Equation 6 presents the relationship between the drone flying height and the desired overlap. Fig. 9 shows that the relation is linear where zero overlap can be obtained when the drone is flying at h_{min} and 100% overlap is obtained when h equals h_{max}. The maximum flying height h_{max} is calculated based on the camera's optical specifications which are fed to the algorithm as input parameters C in order to ensure that the algorithm does not give flying height that is higher than the object recognition range.

$$h_{max} = \frac{w_{pattern} \times c_{res}}{2C_p \times \tan(\frac{C_{fov}}{2})} \tag{4}$$

$$w_{scene} = \frac{w_{sensor} \times h}{f} \tag{5}$$

where $w_{pattern}$ is the pattern's minimum value of width and height i.e. $\min(W, H)$, C_{res} is the camera resolution, C_{fov} is the field of view, and C_p is the minimal number of pixels.

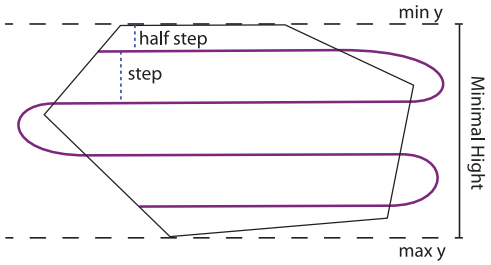


FIGURE 10. Traversing path whole region.

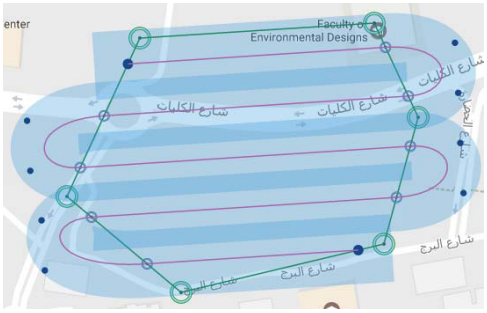


FIGURE 11. Showing path coverage with overlap.

The relation between drone flying height and overlap is given by:

$$h = \frac{1}{2} h_{max} \times (1 + v) \tag{6}$$

where v is overlap percentage

Algorithm (2) shows that once the area is selected by identifying the corner points P, of the searched area, the traversal algorithm starts with testing the polygon to determine whether convex or non-convex and convert it into convex if possible using the `testConvexAndConvert(P)` method. The polygon is then rotated to find the minimal height of the shape as explained in section V-B. Now we can apply the zigzag traversal algorithm to calculate the coordinate of the waypoints in the desired trajectory in which the polygon gets divided according to its minimal height in order to generate a path.

The path segments are generated as parallel horizontal lines (perpendicular to the minimal height) then they will be connected at the appropriate corners to construct a zigzag traversal pattern. In order to arrange the spacing between the parallel segments, we use the variable *step* which equals to half the scene width ($step = w_{scene}/2$) at maximum flying height.

The first segment of the zigzag path is spaced half a step from the minimal height top end as shown in Fig. 10. The waypoints in the drone trajectory are determined by finding the coordinate of intersected points between the generated horizontal lines and the polygon edges. Finally the algorithm checks whether it reaches the polygon boundary and ensures that the last segment is placed inside the boundary. The algorithm complexity is related to polygon minimal height and number of edges $\mathcal{O}(n^2)$.

Algorithm 2 Traversing Path Algorithm

Input: Points $P : \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$,
Pattern dimension $W \times H : \{\text{float} \times \text{float}\}$,
Camera Specifications C,
[Overlap V : where $\{0.0 \leq V \leq 1.0\}$
Drone Height H]
Output: pt path points,
h_{max} maximum flight height,
h_{min} minimum flight height,
θ rotating angle,
h drone flying height

procedure Traverse(P, W, H, C)
  if ***not*** *testConvexAndConvert(P)* **then**
    return nil
  $\theta \leftarrow$ findMinimalHeightAngleAndRotate(P) Find $min_x, min_y, max_x, max_y$ of the polygon
  $h_{max} = \frac{w_{pattern} \times c_{res}}{2C_p \times \tan(\frac{C_{fov}}{2})}$
  $h_{min} = h_{max}/2$
  $h = h_{max} \times (1 + V)/2$
  $step \leftarrow \frac{w_{sensor} \times h_{max}}{2f}$
  $i \leftarrow 0$
  $y = min_y + \frac{step}{2}$
  $pt \leftarrow \{\}$
  while $y \leq max_y$ **do**
    $L \leftarrow \{(min_x, y), (max_x, y)\}$
    $(p1, p2) \leftarrow$ PolygonLineIntersect(P, L)
    $p_L \leftarrow$ pick left point from $(p1, p2)$
    $p_R \leftarrow$ pick right point from $(p1, p2)$
    $p \leftarrow$ Generate points between p_L and p_R
    Insert $\{p_L, \{p\}, p_R\}$ into pt
    if $y + step \geq max_y$ **then**
      $y = \min(y + \frac{step}{2}, max_y)$
    else
      $y = y + step$
    $i = i + 1$
  rotatePointsBy($pt, -\theta$) **return** $\{pt, h_{max}, h_{min}, h, \theta\}$

D. TASK DIVISION

As explained earlier in the default scenario, it is expected that in order for the search mission to be carried out by multiple robotic agents, each of them are assigned to traverse a specific region. As for mission division, we propose two strategies; Path Division and Region Division. Initial location for each drone is counted in both strategies. Starting from the home location H, each drone flies to its assigned starting point in the traversed area and return to the home location H from its last assigned point. Thus the input capability C in both algorithms (3, 4) should account for the capability to fly back and forth from home location C_h and the capability needed

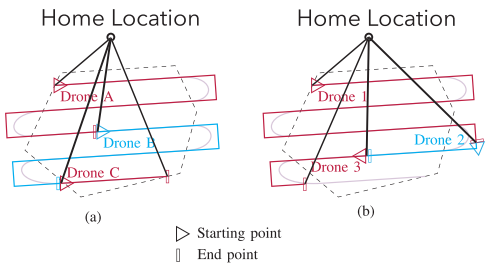


FIGURE 12. Each drone assigned path. (a) whole path covered, however; (b) shows small part of the path uncovered due to the low capability of drones 2 and 3.

to traverse the area C' where $C' = C - C_h$. The method `findFlyFromToHomeCost` is responsible for calculating C_h which depends on the farthest distance between the home location H and any point in the traversed area. The farthest distance in the Path Division algorithm (3) is calculated based on the last waypoint in the traversal path. However, in the Region Division algorithm (4), the farthest distance is calculated based on the farthest polygon edge from home location H.

1) PATH DIVISION

The algorithm (3) will divide the total traversal distance and return the set of waypoints for each drone. The algorithm inputs requires the following; the traversal waypoints of the whole region, which was already generated by Traverse algorithm (2), each drone capability c_i as percentage, and home location H. Drone's capability is computed according to battery level, maximum working time as fully charged, and drown maximum average speed. Using these information we can compute the maximum distance (d_i) that a drone is capable to cover. After the distances (w) for all drones are calculated, the capability factor c_i can be calculated by dividing each drone's maximum flight distance over group total distance.

In `compPathLenAndDroneMaxPathLen` the actual distance that each drone needs to traverse will be generated as a fraction out of the whole traversing path T. This distance should be less than or equal to the maximum distance that a drone can traverse. For example if the total length of the traversal path over the coverage area is 870 meters for drone 1 which has capability factor of $c_1 = 0.3$ and maximum flight distance 300 meters (capability), then the distance assigned to this drone to traverse starting from point (x_1, y_1) in the traversal path T is $d_1 = 261$ meters. Therefore, several waypoints from the traversal path T will be assigned to drone 1, where the path length stating from (x_1, y_1) to (x_j, y_j) is less than or equal to 261 meters. This scenario is repeated for each drone until reaching the end of traversing path as shown in Fig. 12(a) or when the last drone is assigned with its path as shown in Fig. 12(b). The complexity of Path Division algorithm (3) is $\mathcal{O}(m \times n)$ where m is the total number of waypoints and n is number of drones.

Algorithm 3 Divide Traversing Path for Each Drone

Input: Each Drone capability $C : \{c_1, c_2, \ldots, c_n\}$
where $0 < c_i \leq 1$,
Traversing path
$T : \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
Home Location: H
Output: A list of Path for each Drone

```
procedure PathDivision(C, T)
    C_h ← findFlyFromToHomeCost(H, T)
    C' ← recalculateDronesCapability(C, C_h)
    D ← compPathLenAndDroneMaxPathLen(T, C')
    R ← {} empty list for each drone path
    Add H point to R
    for each drone d_i inD do
        R_i ← extract path with d_i length from T
    Add H point to R as last point to back home
    return R
```

2) REGION DIVISION

Region Division algorithm (4) uses divide and conquer strategy. In `eachDroneMaxMinAreaWithHY` method, the polygon's minimal height is divided by number of drones (n) as shown in Fig. 13. The boundary of the polygon segment will be a horizontal line going through the polygon. Therefore, the number of horizontal lines will be $(n - 1)$. An area will be computed for each drone according to each drone capability.

The algorithm calculates the maximum and minimum areas by adding and subtracting the tuning factor (Δ) The whole region area is used to decide whether the actual partition is reached. The coordinates of the segments will result from finding the intersection points of the cross-sectional lines with the polygon segments. The initial step of recursion function in the algorithm is to prepare the division value of y for each partition and area range. The area range is computed by equations (7) and (8). This strategy provides range margins to find actual partition for each drone.

$$\text{area}_{\text{min}} = \text{area}_{\text{segment}} - \Delta \tag{7}$$

$$\text{area}_{\text{max}} = \text{area}_{\text{segment}} + \Delta \tag{8}$$

The recursion construct in the algorithm depends on four parameters (R, i, G, Q) those can be identified as Rwhich represent the remaining polygon area after each slicing process i the indexing of the current drone, G as the range assigned height of each drone, and Q coordinates of the sliced polygon points. The complexity of the region division algorithm is approximated as $\mathcal{O}(n \log n)$.

VI. PERFORMANCE ANALYSIS

In order to study and evaluate the efficiency of the proposed algorithm, the impact of developed task division strategies

Algorithm 4 Divide Region for Each Drone Algorithm

Input: Region as polygon points
$P : \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$,
Rotation angle in radian A
where $0 \leq A \leq 2\pi$,
Each Drone capability $C : \{c_1, c_2, \ldots, c_m\}$
where $0 < c_i \leq 1$,
Home Location: H
Output: A list of Paths for each Drone

procedure) RegionDivision($P, A, C, H, [R, i, G, Q]$)
  if $(R, i, G, Q) \in null$ **then**
    ▷ Recursion Base Case
    $C_h \leftarrow$ findFlyFromToHomeCost(H, T)
    $C' \leftarrow$ recalculateDronesCapability(C, C_h)
    ;
    $G \leftarrow$ eachDroneMaxMinAreaWithHY(P,C')
      ▷ Polygon height gets divide by number of drones which called Horizontal Y
    return RegionDivision($P, A, C', H, P, 1, G, \{\}$)
  $T \leftarrow$ extract polygon that above $Q_i y$ from R
  if $T > G_i min\ area and T < G_i max\ area$ **then**
    $Q_i = T$
    Remove polygon Q_i from R
    if *Current Drone is second last one* **then**
      $Q_{i+1} = R$
      return Q ▷ Recursion end point
    else
      return RegionDivision($P, A, C, H, R, i + 1, G, Q$)
  else if $T > G_i area$ **then**
    $T \leftarrow$ extract polygon that above $Q_i y$ from R
    decrease G_i area by half of assign area
    return RegionDivision(P, A, C, H, R, i, G, Q)
  else if $T < G_i area$ **then**
    increase G_i area by half of assign area
    return RegionDivision(P, A, C, H, R, I, G, Q)

and the effect on the actual flight path execusion by the UAV are studied.

A. PATH ERROR ESTIMATION AND ANALYSIS

The flight path determined by the algorithm is passed to the UAV. The actual path followed by the drone is obtained using GPS localization coordinates. The precision of both the actual and required path is based on GPS accuracy. In order to obtain the relationship between the error and the speed of the trajectory, data from flight trajectory is obtained for different drone's maximum average velocities like 0.5 m/s, 1 m/s, and 5 m/s.

Fig. 14 displays the relationship between the projected and the actual flight path for a selected trajectory segments over

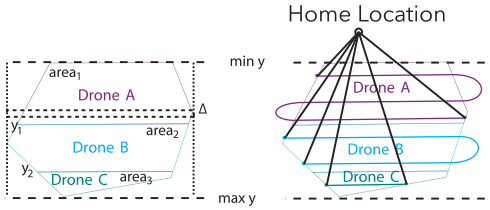


FIGURE 13. Region division and traversing path (Drone 1: 53.3%, Drone Two: 32.97%, Drone 3: 13.74%).

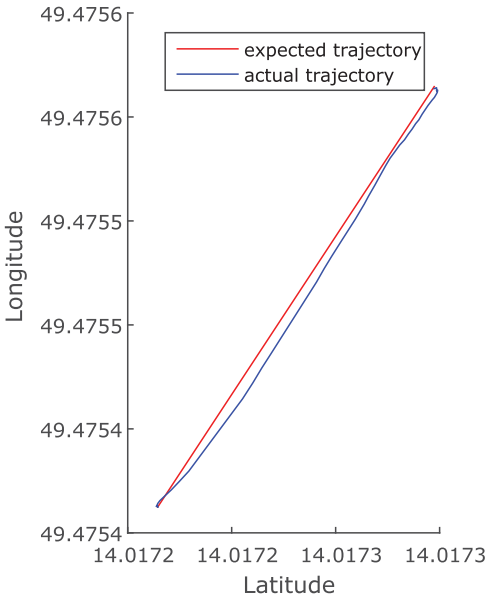


FIGURE 14. Planned Trajectory and Actual Flight Path.

several sections in multiple flight trails. The selected section is a straight lines without turns and thus theoretically a very minimal error is expected.

The expected trajectory is indicated using red color while the actual trajectory is specified using blue color. It could easily be observed that the actual and the expected path are not quite similar considering limitations from GPS accuracy. Fig. 15 shows how drone average speed changes over the straight line path. It is also observed that drone speed changes over the executed path determined by two points fed to the autopilot controller. From Fig. 15, we can summarize the following:

- The drone starts with minimal speed and accelerates towards the maximum allowed speed. Then while moving to the end to the trajectory or towards a turning point, the speed is reduced so as to enable tilting without falling.
- Increase in speed causes the difference between the expected and actual path, denoted as error, to increase. Data analytics reveal that the average speed obtained by excluding the outlier is 1.92 m/s, with corresponding error of 0.48 m.

The relationship between speed and error were further studied using curve fitting as shown in Fig. 16 Data was

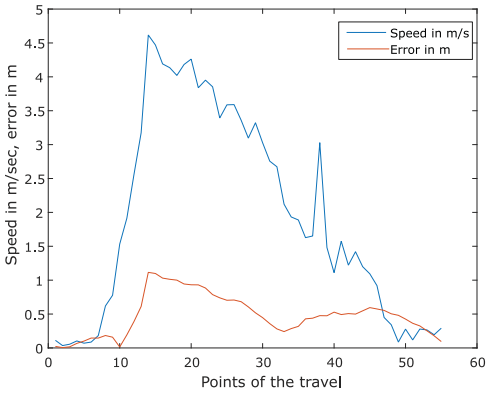


FIGURE 15. Speed and error mapping.

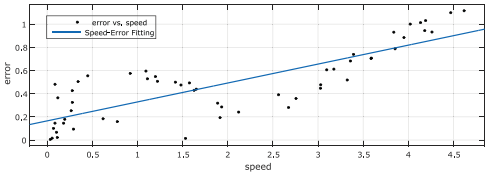


FIGURE 16. Error - speed relationship.

obtained from the drone for seven trajectories. The actual and expected path of the trajectory was compared with regards to the speed of the drone. The maximum drone speed allowed was 5 m/s. The data obtained is fitted using a linear curve. Statistics obtained from the figure below reveals that error increases with proportion to the increase in speed. The curve fits reasonably well with an adjusted R^2 value of 0.6767, which indicates that about 67% of variance in the deviation is explained by the equation (9).

$$\text{error} = a \times \sin(\text{speed} - \pi) + b \times (\text{speed} - 10)^2 + c \quad (9)$$

where

$$\begin{aligned} 0.04381 &\leq a \leq 0.2275 \\ -0.01048 &\leq b \leq -0.005706 \\ 0.8921 &\leq c \leq 1.21 \\ a = 0.1356, &\quad b = -0.008092, \; c = 1.051 \end{aligned}$$

Increasing the speed shows increase in error. Similar study [36] concluded that there is deviation from the planned trajectory at the beginning and the end of the trajectory. Also, deviations were observed while decelerating from higher speeds. Strong aerodynamic effects was stated as one major factor for such deviation. Our experiments deviates only from the planned trajectory only at higher speeds and considering GPS precision, such an error could be considered negligible.

Considering the relation between starting point and end point with the speed and error of the trajectory, it is obvious that this path error would create gaps in the searched area and hence compromise the accuracy of the search mission.

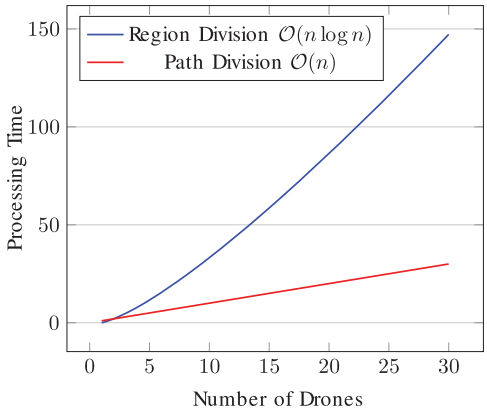


FIGURE 17. Division cost.

As counter measures we might either 1) ensure the camera field of view is wide enough to compensate for the expected error, which could be realized based on flying height of the UAV, as explained in section V-C, or 2) apply further segmentation on the straight lines in the paths. Those two strategies will be further studied and simulated in the future by our group as we have identified this issue at this stage.

B. IMPACT OF PATH DIVISION AND REGION DIVISION

In section V-C, we suggested two approaches that can be used for task division among several drones performing a cooperative searched mission, Fig. 12. In the path division approach in algorithm (3), each drone's task is assigned its portion that is calculated as a fraction of the total path length according to each drone's capability. The second approach using the Region Division algorithm (4) divides the region first according to each drone capability then generate the waypoints by the traversal algorithm (2).

Fig. 17 explains the difference in performance between the two approaches in terms of processing time. It is obvious that Region Division algorithm (4) costs more processing time than Path Division algorithm (3) as the number of drones (or divisions) increases. The increase in processing time between the two approaches can be in the order of three folds. We can conclude that the path division might suite computation on the drone itself while the most computationally intensive area division approach suites calculation on the base station.

C. EXPERIMENTAL RESULTS FOR ZIGZAG TRAVERSAL PATTERN

The Goals of this experiment is to study the behavior of an UAV flying in zigzag trajectory points generated by the algorithm. In this experiment multiple targets are placed in the area. UAV is placed on the predefined position. The launching and landing procedure is demonstrated within this setup. The input to this experiment is a predefined trajectory for a UAV at different speed and flying height. The outputs are post-processed data providing comparison using the given trajectory and its real execution onboard logged flying record.

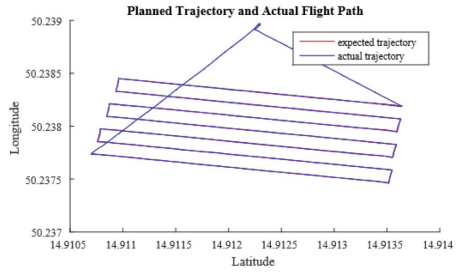


FIGURE 18. Experiment result shows a plot of the zigzag trajectory execution at speed 3m/s and height 15m.

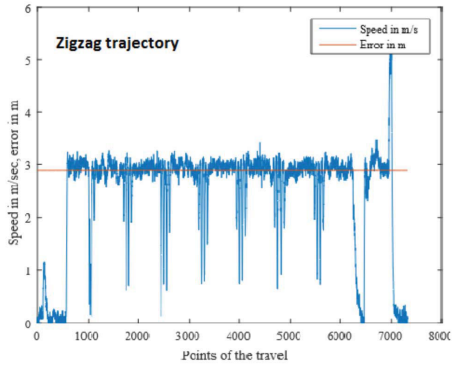


FIGURE 19. Experiment result shows a plot of speed variation along the trajectory for zigzag trajectory.

TABLE 2. Performance comparison between zigzag and segmented zigzag trajectory at maximum speed 3m/s.

Aspect	Zigzag	
	@ 3m/s	@ 6 m/s
Time laps, min	13.0	8.24
Avg. speed, m/s	2.7	4.27
Battery consumption, %	85%	42%
Localization error, m	2.8	3.1

Fig. 18 shows an example of the plotted trajectory points of a zigzag traversal run.

In Fig. 19, we show the recorded changes of the speed along the flight trajectories where you can notice that variation over the zigzag trajectory. To compare the performance of different trajectory plan performance, we have chosen to compare the zigzag to spiral trajectory parameters at different speed 3 and 6 m/s. Table 2 shows the comparison results.

With respect to the effect of speed on the localization error, it is evident that if we compare the zigzag @ 6 m/s maximum speed with the zigzag @ 3 m/s maximum speed, we can find that error increased while the battery lifetime is improved in the latter case. As for the object detection, we can report that objects were detected in all cases with good precision of object localization less than the average accumulated error by 50%.

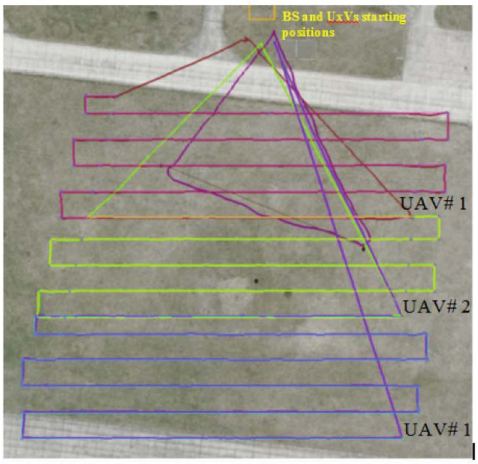


FIGURE 20. Multi-drones experiment (Zig-Zag Traversal).

To verify the developed system integration where a group of heterogeneous robots perform a joint search task, we have considered a more detailed scenario as shown in Fig. 20. Three different UAVs were used to traverse the area looking for tagged objects. The events of the scenario unfold as soon as the operator enters the search area coordinates through the GUI interface, the three UAVs are eventually launched from the initial locations which is located on north of the searched area. The drones simultaneously travel to their calculated starting points in the searched area using both the task division and the traversal algorithms then they execute the waypoints based on their assigned paths. As soon as there are no more waypoints to execute, i.e. the assigned area is traversed, the drone flies back to the home position.

VII. CONCLUSION

In this paper we have presented several findings with relation to the ongoing work to develop a heterogeneous platform for multi-agent robotic testbed. We laid out the fundamental agent's traversal algorithms with different functional components. Using real data from trajectory execution by the drones, we have identified that the behavior of the drone is subject to autopilot controller and localization accuracy. A compromised path adherence accuracy will badly impact the camera scan coverage by creating gaps in between the traversed lines of the zigzag trajectory thus increase the probability of missing a target. To overcome this problem, we may induce camera scanning overlapping between the adjacent traversed lines by varying the drone height, we have shown how our algorithm takes care of this issue. We have also identified that path adherence error is subject to the maximum average speed set by the operator. We have statistically analyzed this relationship based on our setup. Basically, we found out that the higher the maximum execution speed, the more erroneous the traversal path. Regarding to the developed task division algorithms, the analysis of the

processing cost indicates that the path division strategy is more efficient than the region division strategy while both of them take into consideration the robotic agents capabilities in order to assign the task to each of them proportionally. Finally we have presented some field experiments using the developed platform to provide overall realization of system performance and system integration. In future work, further performance enhancement and optimization of the algorithm swill be proposed, developed, analyzed and tested.

ACKNOWLEDGMENT

This work was supported by the Deanship of Scientic Research (DSR), King Abdulaziz University, Jeddah, under Grant RG-1-611-39.

REFERENCES

[1] (2020). *SPARC's Strategic Research Agenda*. [Online]. Available: http://sparc-robotics.eu/roadmap/

[2] A. S. O. Shehory, *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*. Berlin, Germany: Springer, 2014.

[3] E. Lavendelis, A. Liekna, A. Nikitenko, A. Grabovskis, and J. Grundspenkis, "Multi-agent robotic system architecture for effective task allocation and management," in *Proc. Signal Process., Robot. Autom. (ISPRA)*, 2012, pp. 167–174.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intell.*, vol. 7, no. 1, pp. 1–41, Mar. 2013.

[5] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *Int. J. Robot. Res.*, vol. 28, no. 3, pp. 357–375, Mar. 2009.

[6] C. A. C. Parker and H. Zhang, "Cooperative decision-making in decentralized multiple-robot systems: The best-of-N problem," *IEEE/ASME Trans. Mechatronics*, vol. 14, no. 2, pp. 240–251, Apr. 2009.

[7] S. Berman, V. Kumar, and R. Nagpal, "Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 378–385.

[8] G. Sartoretti, M.-O. Hongler, M. E. de Oliveira, and F. Mondada, "Decentralized self-selection of swarm trajectories: From dynamical systems theory to robotic implementation," *Swarm Intell.*, vol. 8, no. 4, pp. 329–351, Dec. 2014.

[9] M. Vigelius, B. Meyer, and G. Pascoe, "Multiscale modelling and analysis of collective decision making in swarm robotics," *PLoS ONE*, vol. 9, no. 11, Nov. 2014, Art. no. e111542.

[10] S. Hert and V. Lumelsky, "Polygon area decomposition for multiple-robot workspace division," *Int. J. Comput. Geometry Appl.*, vol. 08, no. 04, pp. 437–466, Aug. 1998.

[11] I. Maza and A. Ollero, *Multiple UAV Cooperative Searching Operation Using Polygon Area Decomposition and Efficient Coverage Algorithms*. Tokyo, Japan: Springer, 2007, pp. 221–230.

[12] J. F. Araújo, P. B. Sujit, and J. B. Sousa, "Multiple UAV area decomposition and coverage," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Apr. 2013, pp. 30–37.

[13] M. Duckham, L. Kulik, M. Worboys, and A. Galton, "Efficient generation of simple polygons for characterizing the shape of a set of points in the plane," *Pattern Recognit.*, vol. 41, no. 10, pp. 3224–3236, Oct. 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320308001180

[14] A. Koubaa and B. Qureshi, "DroneTrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the Internet," *IEEE Access*, vol. 6, pp. 13810–13824, 2018.

[15] (2018). *MAVLINK*. [Online]. Available: http://qgroundcontrol.org/mavlink/start

[16] (2018). *Robot Operating System*. [Online]. Available: http://www.ros.org

[17] (2018). *Analysing ROS Distribution Capabilities*. [Online]. Available: http://www.dcs.gla.ac.uk/research/rosie/ros-limits-2016-08-11.html

[18] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Auto. Robots*, vol. 33, nos. 1–2, pp. 21–39, Aug. 2012, doi: 10.1007/s10514-012-9281-4.

[19] B. H.-Y. Lee, J. R. Morrison, and R. Sharma, "Multi-UAV control testbed for persistent UAV presence: ROS GPS waypoint tracking package and centralized task allocation capability," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2017, pp. 1742–1750.

[20] A. Barnawi, A. Al-Barakati, and O. Alhubaiti, "A GUI interfaces for a multiple unmanned autonomous robotic system," in *Proc. Int. Conf. Softw. Eng. Res. Pract.*, 2018, pp. 36–41.

[21] A. Barnawi, A. Al-Barakati, A. Khan, F. Bajaber, and O. Alhubaiti, "A proposed architecture for a heterogeneous unmanned aerial vehicles system," *Int. J. Electr. Electron. Eng. Telecommun.*, vol. 7, no. 3, pp. 119–126, Jul. 2018.

[22] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "MASON: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, Jul. 2005.

[23] J. C. del Arco, D. Alejo, B. C. Arrue, J. A. Cobano, G. Heredia, and A. Ollero, "Multi-UAV ground control station for gliding aircraft," in *Proc. 23rd Medit. Conf. Control Autom. (MED)*, Jun. 2015, pp. 36–43.

[24] J. Tisdale, A. Ryan, M. Zennaro, X. Xiao, D. Caveney, S. Rathinam, J. Hedrick, and R. Sengupta, "The software architecture of the berkeley UAV platform," in *Proc. IEEE Int. Conf. Control Appl.*, Oct. 2006, pp. 1420–1425.

[25] L. F. Bertuccelli and J. P. How, "Search for dynamic targets with uncertain probability maps," in *Proc. Amer. Control Conf.*, 2006, p. 6.

[26] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Decentralized cooperative search by networked UAVs in an uncertain environment," in *Proc. Amer. Control Conf.*, vol. 6, 2004, pp. 5558–5563.

[27] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand, "Cooperative control for multiple autonomous UAV's searching for targets," in *Proc. 41st IEEE Conf. Decis. Control*, vol. 3, Dec. 2002, pp. 2823–2828.

[28] M. Polycarpou, Y. Yang, and K. M. Passino, "A cooperative search framework for distributed agents," in *Proc. IEEE Int. Symp. Intell. Control*, Mexico City, Mexico, vol. 16, 2001, pp. 1–6.

[29] K. Zhang, E. G. Collins, Jr., and D. Shi, "Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction," *ACM Trans. Auto. Adapt. Syst.*, vol. 7, no. 2, p. 21, 2012.

[30] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[31] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung, D. F. Wolf, G. S. Sukhatme, and D. C. MacKenzie, "Adaptive teams of autonomous aerial and ground robots for situational awareness," *J. Field Robot.*, vol. 24, nos. 11–12, pp. 991–1014, 2007.

[32] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires," *J. Field Robot.*, vol. 23, nos. 3–4, pp. 165–184, 2006.

[33] A. Viguria, I. Maza, and A. Ollero, "Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions," *Adv. Robot.*, vol. 24, nos. 1–2, pp. 1–23, Jan. 2010, doi: 10.1163/016918609X12585524300339.

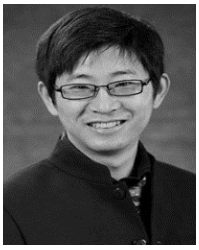
[34] A. Barnawi, "An advanced search and find system (ASAFS) on IoT-based mobile autonomous unmanned vehicle testbed (MAUVET)," *IEEE Internet Things J.*, vol. 45, pp. 3273–3287, Feb. 2018.

[35] Y. Cao, R. Wang, M. Chen, and A. Barnawi, "AI agent in software-defined network: Agent-based network service prediction and wireless resource scheduling optimization," *IEEE Internet Things J.*, Oct. 2019, doi: 10.1109/JIOT.2019.2950730.

[36] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 1485–1491, Jan. 2011.

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.



MIN CHEN (Senior Member, IEEE) has been a Full Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), since February 2012. He is the Director of the Embedded and Pervasive Computing (EPIC) Lab, HUST. He is Chair of the IEEE Computer Society (CS) Special Technical Communities (STC) on Big Data. He was an Assistant Professor with the School of Computer Science and Engineering, Seoul National University (SNU). He worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of British Columbia (UBC) for three years. Before joining UBC, he was a Postdoctoral Fellow at SNU for one and half years. His research interests include cognitive computing, 5G networks, embedded computing, wearable computing, big data analytics, robotics, machine learning, deep learning, emotion detection, the IoT sensing, mobile edge computing, and so on. He received the Best Paper Award from QShine 2008, the IEEE ICC 2012, ICST Industrial IoT 2016, and the IEEE IWCMC 2016. He serves as a Technical Editor or an Associate Editor for the IEEE Network, *Information Sciences*, *Information Fusion*, and IEEE Access, and so on. He served as a leading Guest Editor for the IEEE Wireless Communications, the IEEE *Network*, and the IEEE Transactions Service Computing, and so on. He is a Series Editor of the IEEE Journal on Selected Areas in Communications. He is Co-Chair of the IEEE ICC 2012-Communications Theory Symposium and the IEEE ICC 2013-Wireless Networks Symposium. He is General Co-Chair for IEEE CIT-2012, Tridentcom 2014, Mobimedia 2015, and Tridentcom 2017. He is Keynote Speaker for CyberC 2012, Mobiquitous 2012, Cloudcomp 2015, IndustrialIoT 2016, Tridentcom 2017, and the 7th Brainstorming Workshop on 5G Wireless. He has over 300 publications, including over 200 SCI articles, over 100 IEEE Transactions/Journal articles, 32 ESI highly cited articles and ten ESI hot articles. He has published eleven books: *OPNET IoT Simulation* (2015), *Big Data Inspiration* (2015), *5G Software Defined Networks* (2016), *Introduction to Cognitive Computing* (HUST Press 2017), *Big Data: Related Technologies, Challenges, and Future Prospects* (2014), *Cloud Based 5G Wireless Networks* (Springer, 2016), *Cognitive Computing and Deep Learning* (China Machine Press, 2018, and *Big Data Analytics for Cloud/IoT and Cognitive Computing* (Wiley, 2017). His Google Scholar Citations reached 21 300 with an H-index of 73 and i10-index of 221. His top article was cited 2600 times. He was selected as Highly Cited Research at 2018. He received the IEEE Communications Society Fred W. Ellersick Prize, in 2017, and the IEEE Jack Neubauer Memorial Award, in 2019.