Spiral-like Coverage Path Planning for Multiple Heterogeneous UAS Operating in Coastal Regions

Fotios Balampanis, Ivan Maza and Anibal Ollero

Abstract— **This paper addresses area coverage in complex non concave coastal regions for an arbitrary number of heterogeneous Unmanned Aircraft Systems (UAS). The space is discretized with the constrained Delaunay triangulation and the Lloyd optimization is applied to the computed mesh. The paper presents an algorithm to compute a waypoint list for each UAS such that each sub-area is covered with its sensor on-board following a pattern that goes from the borders of the sub-area to the inner regions. In addition, the resulting paths lead to a uniform coverage pattern that avoids visiting some regions more times than others. Different sensitivity parameters of the algorithm are compared based on the average angles between the waypoints and the total length of the paths. Results show that these parameters support the optimization of the computed waypoint lists and the proposed algorithm produces feasible coverage paths while increasing their smoothness.**

Index Terms— **Coverage Planning, Multiple UAS, Area Partitioning, Lloyd smoothing**

I. Introduction

Coastal regions have complex geographical attributes and the increasing need of UAS usage for remote off-shore location activities has raised challenges for marine environment protection and sustainable management. More specifically, the European region has a vast coastline and economic zones which go far into the Atlantic and Arctic oceans. In addition, the European Strategy for Marine and Maritime Research states the need to protect the vulnerable natural environment and marine resources in a sustainable manner. The use of UAS in coastal regions provides increased endurance and flexibility while reducing the environmental impact, the risk for human operators and the total cost of operations. This study has been carried out in MarineUAS [1] framework, an European Union funded doctoral program which strategically strengthens research training on Unmanned Aerial Systems for Marine and Coastal Monitoring.

In particular, this paper tackles the problem of complex coastal area segregation and partitioning for a team of heterogeneous UAS, based on their kinematic characteristics and on-board sensing capabilities. The proposed strategy involves the use of computational geometry algorithms along with graph search algorithms which manage to segment any complex region for a number of UAS. This decomposition also takes into account their relative autonomy capabilities. Hence, each configuration space for the number of UAS is decomposed in a sum of sensor-projection sized cells and a coverage waypoint path strategy is computed in parallel for each of the sub-regions.

The paper is organized as follows. A short introduction including related work is presented in Sect. II, presenting decomposition strategies and algorithms for path planning of single or multiple UAS. Section IV introduces the grid decomposition strategy, constrained by the characteristics of the UAS, its on-board sensor and the area itself. Section V presents the algorithm which manages to extract coverage waypoint plans following a spiral-like pattern taking into account the shape of the areas. Section VI presents the quantitative and qualitative results in a simulation environment and Section VII closes the paper with the conclusions and future steps.

II. Related Work

Literature provides many studies for a network of robots executing complete area coverage tasks. There is also a number of surveys which tackle individual aspects of the problem. The authors in [1] provide an comprehensive survey regarding the coverage path planning problem for both continuous and discrete approaches. Their survey categorizes area decomposition, partitioning and planning algorithms.

A review on multi-robot task allocation and distribution is presented in [2], where the authors propose a taxonomy for this area, named iTax. This taxonomy categorizes the problems by complexity; the first category includes problems that can be solved linearly whereas the other three include NP-Hard problems. On multi-robot task allocation, the study in [3] gathers the recent approaches on solving the problem and presents the challenges in each approach.

In general, the problem can be separated in three parts and each one of the previously mentioned surveys, as well as the following studies, contributed to each of these aspects: area segregation, area partition and coverage path planning for multiple autonomous vehicles.

Regarding the discrete configuration spaces and the partition process, using a Voronoi tessellation or a Delaunay triangulation is common; in [4] a Voronoi partitioning of an area to be painted is performed in order for the involved robotic arms not to overlap or collide. The process is optimized by minimizing the completion time for the task and the sum of torques of the arms in order to expand their

This work is partially supported by the MarineUAS Project funded by the European Union's Horizon 2020 research and innovation programme, under the Marie Sklodowska-Curie grant agreement No 642153 and the MINECO Retos AEROMAIN DPI2014-C2-1-R Spanish project.

Fotios Balampanis, Ivan Maza and Anibal Ollero are with the Robotics, Vision and Control Group, University of Seville, Avda. de los Descubrimientos s/n, 41092, Sevilla, Spain. Email: fbalampanis@us.es, imaza@us.es, aollero@us.es

[1] http://marineuas.eu

lifespan. The authors in [5] perform a Delaunay triangulation in order to obtain a free space schema in an area and produce a shortest path algorithm based on the extracted edges. The repeated coverage issue is the main focus of the study in [6]. The authors initially tackle the problem by maximizing the visibility of the robots in the area, also commonly known as the "art gallery problem". These positions of maximum visibility along with a Constrained Delaunay Triangulation (CDT), produce a graph for creating coverage waypoint lists. On coverage path planning and coverage patterns, one of the most used motion pattern in literature is the boustrophedon movements, also known as lawnmower or zig zag pattern [1] [7] [8]. This strategy is applied either in a continuous space or in a square grid decomposition overlay. However, complete coverage is not always achieved and usually simple non convex areas are chosen as a test case [9]. In case of non convex areas, the usual strategy is decomposing that region into a sum of convex polygons. Unfortunately sometimes this leads to repeated coverage paths as it can be seen in [10] when the vehicle traverses to the next polygon. Furthermore, spiral paths might give a better coverage in comparison with boustrophedon movements [11].

In this paper a different coverage strategy is proposed. With a triangulation it is not possible to follow straight lines between neighboring cells, creating straight line paths by visiting the centroids of each triangle; hence an inward, from the borders of the area to the center, spiral-like path which will be agnostic of the complexity or the constraints of the region is chosen. In that manner, the convexity of an area is not a problem, while at the same time an algorithmic strategy is introduced to minimize the angles of the produced path.

III. Problem Statement and Adopted Framework

Let us consider a cooperative team of autonomous and heterogeneous UAS and an area monitoring task which has to be accomplished by the team in a coastal region A with no-fly zones inside. Let $U_1, U_2, \ldots, U_n$ be a team of n UAS to perform the monitoring task and let $A_1, A_2, \ldots, A_n$ be a partition of A in n sub-areas where A_i is the sub-area assigned to U_i. The size of the sub-areas should be computed accordingly to the relative capabilities of the involved UAS. The goal of this paper is to compute a waypoint list for each U_i such that the whole sub-area A_i is covered with the sensor on-board following a pattern that goes from the borders of A_i to the inner regions. In addition the resulting paths should lead to a uniform coverage pattern that avoids visiting some regions more times that others.

As it is discussed in [12], area triangulation is specially well-suited for the particular characteristics of the coastal regions among the possible space discretization approaches. In this work, we propose to apply a variant of the so-called Delaunay triangulation. A Delaunay triangulation $DT(H)$ is a collection of triangles in which all points $H = \{h_1, h_2, \ldots, h_m\}$ that belong to H are not inside the circum-circle of any triangle in this set. A variant called Constrained Delaunay Triangulation $CDT(H)$ [13] introduces forced edge constraints which are part of the input, and are useful in this study as they actually define the boundaries of the polygonal area and the holes inside the polygon. In order to obtain even more homogeneous triangles, Lloyd optimization [14] can be applied on the resulting triangulation. This algorithm improves the angles of each cell, making them as close as possible to 60 degrees. In addition, it is possible to consider the sensor on-board characteristics as a constraint of the CDT method. By using the largest side of the largest projected FoV of the team of UAS as a constraint in the CDT, an initial mesh is obtained that is a sum of FoV-sized triangles.

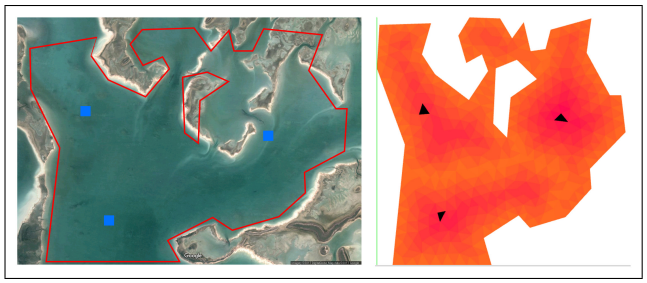

Fig. 1. On the left, a region in Great Abaco island near Miami. The red polygons define the area constraints. The blue boxes indicate the initial positions of each UAS. On the right, the same region partitioned for the three UAS. Their initial positions are the black cells, whereas each shade of orange is the border-to-center cost. Their coverage percentage in relation to the whole area is 30%, 40% and 30%. The CDT is a result of 10 Lloyd iterations.

Then, by applying the CDT discretization, each sub-area A_i is decomposed into $v_1, v_2, \ldots, v_{m_i}$ FoV-sized cells as close as possible to the size of the footprint of the respective UAS. Thus, area A_i has a total of m_i cells depending on the FoV and coverage capabilities of the respective U_i UAS. Figure 1 shows an example of the area partition process for three UAS operating in a region in Great Abaco island near Miami.

IV. Area Segregation Based on Sensor and Vehicle Constraints

A. Sensor and platform constraints

Let us consider that function $P(t)$ calculates the area projection of the on-board sensor Field of View (FoV) to the ground at a specific moment t during the flight. This function is dependent on the relative rotation matrices among the UAS frame F_U, sensor frame F_S and ground F_G coordinate frames, as it can be seen in Fig. 2.

The usual convention in aviation is used for the UAS movements; roll (γ), pitch (β) and yaw (α). The UAS coordinate frame has its x-axis pointing forwards in relation with movement, the y-axis is given by the right-hand rule, while the z-axis points downwards. There are also cases where a gimbal is used in order for the UAS attitude to become irrelevant and the gimbal to compensate for the UAS movements, or its specific orientation to translate the sensing angle.

1) Simple sensor scenario: In a simple scenario where an UAS is equipped with a camera, mounted in a gimbal and facing downwards like in Fig. 2, the FoV projection will be similar to the one shown in Fig. 3.

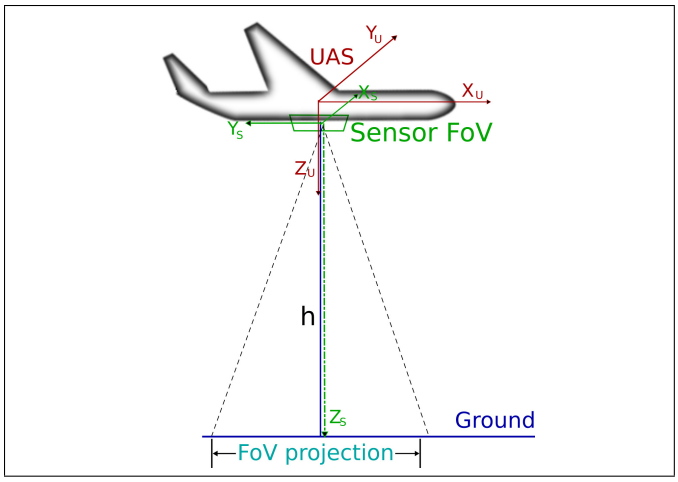

Fig. 2. The coordinate frames of the system. In a presence of a gimbal which compensates for the movements of the UAS, the sensor always points down on a given angle with respect to the ground.

The FoV angles of θ_x and θ_y, draw the projected rectangle $P(x_p, y_p)$ in relation with the flight altitude h:

$$x_p = 2h \tan\left(\frac{\theta_x}{2}\right) \tag{1}$$

and

$$y_p = 2h \tan\left(\frac{\theta_y}{2}\right) \tag{2}$$

Then, the biggest inscribed circle C_P of P is the one that has a radius r equal to the smallest side of $P : P_{min}$. Hence, the largest triangle W that can be inscribed in this circle is the equilateral triangle W_P and its side W_d is

$$W_d = \sqrt{3}\frac{P_{min}}{2} . \tag{3}$$

Thus, as it will be described in the next section, the desired triangulation for the mesh generation is a CDT with maximum side equal to

$$W_d = \frac{2\sqrt{3}h \tan\left(\frac{\theta_{min}}{2}\right)}{2} . \tag{4}$$

If the camera takes a picture in an dω interval, which depends on its frame rate, in the area constrained by the

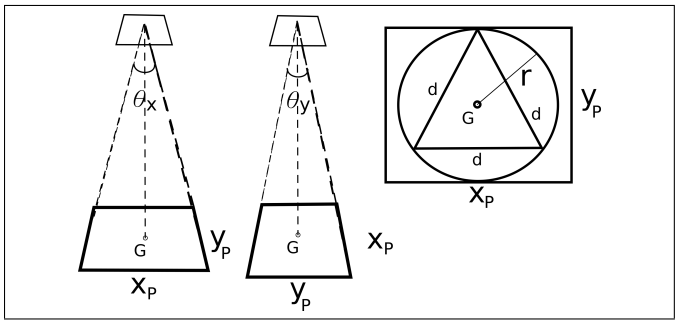

Fig. 3. Calculating the CDT side constraint in an simple scenario. In the existence of a gimbal, the FoV projection is in a fixed angle with respect to the ground. Thus, the maximum triangle that can be inscribed, is the inscribed equilateral triangle of the trapezoid's inscribed circle.

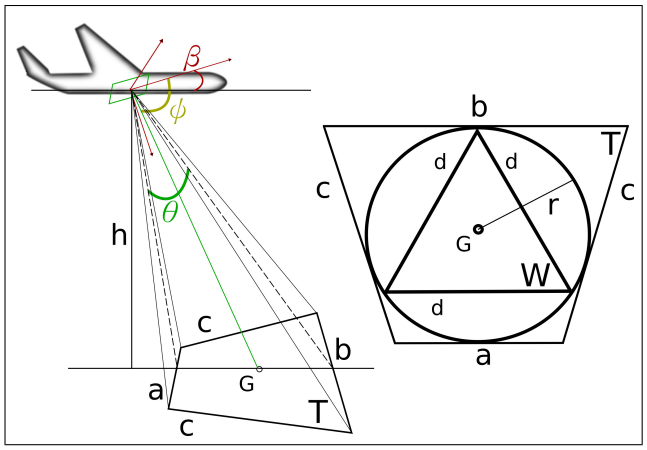

Fig. 4. Calculating the CDT side constraint in case the sensor is tilted in one or more axis [15]. The angle θ is the FoV angle as in the previous examples. Pitch angle β along with the bisector of θ compose the front mounted angle ψ. Then, the ground projection is a trapezoid and the maximum triangle that can be inscribed, is the inscribed equilateral triangle of the trapezoid's inscribed circle.

previously mentioned triangle, then by the time Ω it will have finished or stopped its task and the covered area is

$$A = \int_0^{\Omega} P(\omega)d\omega = \int_0^{\Omega} \left[\frac{\sqrt{3}}{4}W_d^2\right] d\omega = \int_0^{\Omega} \left[\frac{2\sqrt{3}h \tan\left(\frac{\theta_{min}}{2}\right)}{2}\right] d\omega \tag{5}$$

in relation with the FoV smaller angle and altitude. This coverage metric measures only the area covered by the triangles and not by the UAS nor the complete FoV projection.

2) More complex scenarios: In case the sensor is tilted in one or more axis, then the resulting FoV projection shape is either a trapezoid or an oblique projection. In the first case, the pitch angle β of the platform along with the bisector of the FoV angle θ, compose the front mounted angle ψ, as it has been adopted by [15] and shown in Fig. 4.

The ground FoV projection forms a trapezoid $T = a, b, c, d$, having the bottom side a equal to

$$a = \frac{2h \tan(\frac{\theta_y}{2})}{\sin(\psi - \beta + \frac{\theta_x}{2})} \tag{6}$$

and upper side b equal to

$$b = \frac{2h \tan(\frac{\theta_y}{2})}{\sin(\psi + \beta + \frac{\theta_x}{2})} , \tag{7}$$

where θ_y and θ_x are the FoV angles of the camera on the Y_S and X_S axess respectively, of Fig. 2. Then, largest triangle W that can be inscribed in the trapezoid circle is the equilateral triangle W_P and its side W_d is

$$W_d = \frac{\sqrt{3ab}}{2} = \frac{\sqrt{3\frac{2h \tan(\frac{\theta_y}{2})}{\sin(\psi-\beta+\frac{\theta_x}{2})}\frac{2h \tan(\frac{\theta_y}{2})}{\sin(\psi+\beta+\frac{\theta_x}{2})}}}{2} \tag{8}$$

Once again, as it will be described in the next section, the desired triangulation for the mesh generation is a CDT with maximum side equal to (8). The calculation of the coverage as in (5) is done in the same way, replacing the size of W_d.

Even more complicated scenarios are tackled by the authors in [16] and [17], while [15] provides a detailed calculation method. In this paper, the downward facing scenario is used, where a gimbal compensates for the UAS movements.

B. Sensor specific - orientation agnostic segregation cells

The aforementioned calculations manage to provide a complete coverage formalization in the continuous configuration space. Having specific information about the sensor, namely its FoV and its sampling rate, the problem can be reduced to a discrete sampling scenario, by creating a grid where every cell is a sample. By performing a discretization of the area according to the sensing capabilities and characteristics of the sensor, a Constrained Delaunay Triangulation (CDT) is used as in our previous work [12]. Then, a maximum triangle side constraint can be calculated from the FoV of the sensor as described in the previous subsection. Since the calculated triangle is inscribed in the projection's inscribed circle, if every center of each circle is considered as a waypoint, the configuration manages to produce segregation cells that are always covered by the sensor FoV if all waypoints are visited, no matter the flight direction; thus, this strategy is identified as *orientation agnostic*.

Then, the integral function in (5) for coverage calculation in continuous space can be simplified as the sum of every segregated cell v_i of the triangulation; since we take a sample in every discretized cell, the sum of all cells provides a complete coverage metric $\sum_{v=0}^{V} P(v)$, where the area of each cell v can be calculated as $\sqrt{s(s-v_a)(s-v_b)(s-v_c)}$, being s is the semiperimeter of the triangle and a,b,c are the sides; hence the decomposition cells are platform and sensor specific. Please notice that this calculation rule also applies in case that the resulting FoV projection is not a square or a trapezoid, but a tangential quadrilateral. It is always possible to obtain the inscribed circle and the respective inscribed triangle as a function of the FoV sides. In case the FoV projection is a square, the inscribed triangle calculation is trivial.

C. Grid optimization

The CDT decomposition strategy discussed in the previous section, does not guarantee that all the segregation cells are going to be equilateral triangles, thus providing smaller triangles and larger sample overlapping. Its constraints are limited to either maximum triangle side or minimum angle and the area itself. Thus the choice of optimizing the resulting grid is merely a task defined choice: if a varying overlapping schema is not critical, then the grid can be maintained. On the other hand, if a normalized schema is preferred, a series of Lloyd optimization [18] iterations can be performed in the resulting grid. This operation is computationally expensive, and thus it is recommended mainly for offline computation. Nevertheless, having the size of each cell of the triangulation as well as the FoV of the sensor, the magnitude of sample overlapping can be calculated in the initial coverage planning. It is clear that the more regular the triangles are, the less overlapping will occur. Since increasing the Lloyd iterations the distribution of triangles closer to equilateral is increased, this operation can be added in the path generation cost functions.

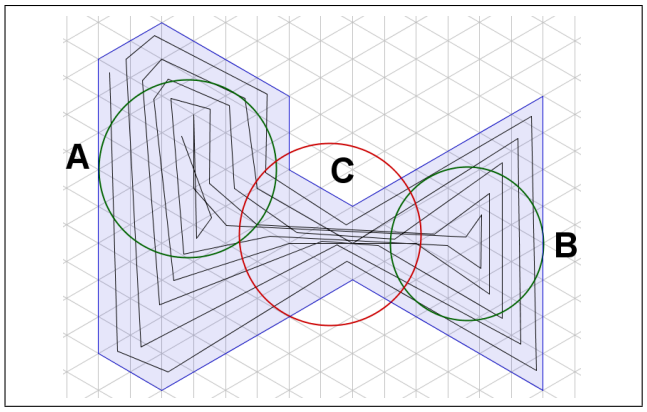

Fig. 5. A waypoint list of a triangulated area computed with the coverage algorithm in [12]. The regions in A and B create valleys in the borders-to-center cost attribution schema as their surrounding region as also as region C have higher costs. Since the algorithm initially visits cells according to their distance from the borders, a lot of back and forth paths are created in region C.

V. Coverage waypoint planning based on area properties

Since all the cells of the CDT triangulation are connected, this discretized space can be represented as an undirected graph $G = (V, E)$, where the set V of vertices represents the triangles and E is the set of edges such that there is an edge from v_i to v_j if the corresponding triangles are neighbors. Two triangles are neighbors if an UAS can move freely between them.

Each sub-area A_i is either adjacent to the sub-area of another UAS or to the borders of the whole area. In order to compute the coverage waypoint list, each vertex will be given a transition cost $D(v_i)$ of proximity from the borders of A_i. The adjacent vertices to the borders are given a high $D(v_i)$ cost and they are the root nodes of a tree. The neighbors of these vertices are given a smaller cost and they are the direct children of the parent nodes. By using the recursive algorithm presented in [12] for the rest of the vertices, a border to center pattern is generated. However, it has been found that for certain shapes of A_i the waypoint lists contain many back and forth motions over the same cells (see Fig. 5).

In this paper the coverage algorithm presented in [12] is extended and improved by introducing a path smoothing parameter. The new algorithm identifies the existence of *valleys* (as in Fig.5) in the decomposition schema and, according to a sensitivity parameter, visits isolated areas first before continuing to the rest of the area as in Algorithm 1. The sensitivity metric is calculated by identifying if the next

step is further than the smaller distance between neighboring cells times the sensitivity setting.

Algorithm 1: Coverage waypoint list with valley sensitivity. D_{v_c} is the border-to-center cost of cell v as it has been assigned by the method in [12]. D_c is an auxiliary variable with the current border-to-center cost. S_m is the valley smoothing metric which is compared for every next step. S_s is the valley sensitivity setting. Cell v_{I_k} is the starting position of UAS U_k. W is the produced waypoint list of vertices. Function $maxNeighborDistance$ gets the maximum distance between neighboring cells in all CDT_k, where CDT_k is the sub CDT for UAS U_k. Function $findClosest$ finds the closest vertex to the current one that has its same border-to-center cost. Function $calculateDistance$ measures the distance between two vertices. Vector S_v is the placeholder for valley vertices that are to be visited in the future.

```
D_c ← ∞;
S_m ← maxNeighborDistance(CDT_k) ∗S_s;
v ← findClosest(v_{I_k}, D_c);
W.insert(v);
foreach v_i ∈ CDT_k do
    if ∃v, D_{v_c} = D_c then
        v_j ← findClosest(v, D_c);
        if CalculateDistance(v, v_j) > S_m then
            S_v ← v_j;
            D_c ← D_c − 1;
        end
        else
            W.insert(v_j);
            v ← v_j;
        end
    end
    else if S_v = ∅ then
        D_c ← D_c − 1;
    end
    else
        v_j ← findClosest(v, S_v);
        W.insert(v_j);
        S_v.remove(v_j);
        D_c ← D(v_{j_c});
    end
end
```

While this algorithm manages to compensate for area properties like the aforementioned valley regions, an extended solution is needed in order to also compensate for sharp turns, which violate the kinematic constraint of minimum turn rate of the UAS.

VI. Simulation Results and Comparisons

The proposed strategies and algorithms have been tested in a simulated environment. Since the initial segmentation, partitioning and waypoint path planning processes are performed offline before flight, interesting quantitative metrics

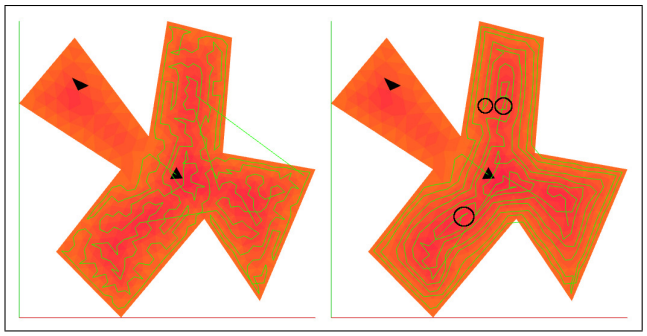

Fig. 6. A sample of different valley sensitivity setups. In the left case, the algorithm identifies a lot of next steps which are far from the sensitivity setting, making a lot of turns. On the right, the paths are smoother. In the black circles, some points where the algorithm identified a far next movement and chose the closest cell instead.

are extracted which will help to validate the solution and the comparison with the actual flight data. The experimental setup is based on a Software In The Loop (SITL) simulation environment and the algorithms have been implemented in a ROS[2] (Robot Operating System) node. The CDT has been performed by using the CGAL library [19]. The ROS node also performs the visualization functions and the communication of the extracted waypoint lists to the UAS instances, by sending MAVLINK[3] messages. Each simulated flight is monitored by the APM Planner 2 ground control station (APM Planner 2[4]) which receives emulated data from the Arduplane[5] autopilot software. The UAS model used in the simulated flights is a fixed wing Rascal11 model airplane. Its on-board controller is the open source autopilot Ardupilot and has been combined with the JSBSim flight dynamics model simulator. The Arduplane controller used is the Pixhawk Flight Management System [20].

A. Segregation, partition and coverage waypoint planning

In order to identify the geometrical properties and spot the qualitative differences in different sensitivity settings of Algorithm 1, two areas have been chosen. The first is a sample area with several "valley" areas. The second is a realistic scenario in an existing area, the same that has been presented in Fig. 1. On the latter, a simulated flight has been performed in order to obtain results of the real trajectory followed.

1) Sample area: This area has been segregated and partitioned for two UAS; one of them visits three out of four "valley" regions, as it can be seen in Fig. 6.

The two settings that have been chosen demonstrate the effects that this algorithm might have: on lower sensitivity settings, some vertices might be identified as valleys while they are not. As a result, repetitive turns are produced and the paths are far from smooth. On higher sensitivity settings, the paths are smoother and the valleys are identified correctly, but the total length of the path is larger, as it

[2]http://www.ros.org/
[3]http://mavlink.org
[4]http://ardupilot.org/planner2//
[5]http://ardupilot.com/

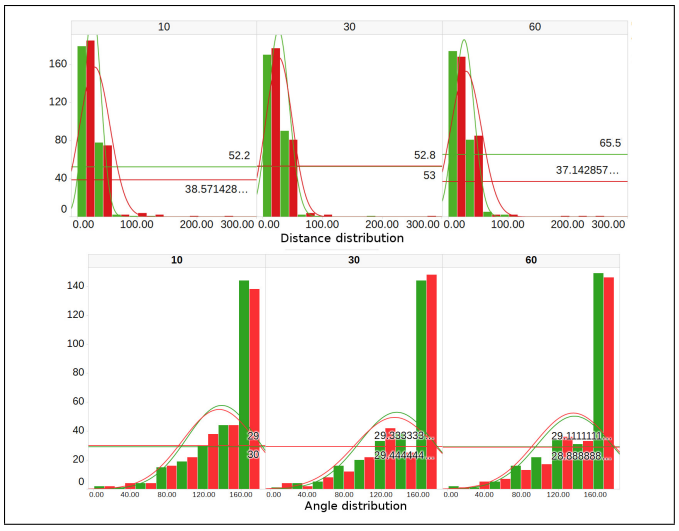

Fig. 7. A comparison between the algorithm in [12](depicted in red color) and the one presented in this paper (green). In the upper panel the average distance between each cell might be larger but as the distribution curve shows, also clear in Table II, the total path length is smaller. In the lower panel, the angle distribution is shown. While there is no big difference, the distribution curves show that with the new algorithm, there is a shift towards larger angles, thus smoother paths. Both panels are in columns which represent the different Lloyd smoothing settings (10, 30 and 60)

TABLE I

METRICS ON VALLEY SMOOTHING SENSITIVITY SETTING CORRESPONDING TO FIG. 6

	Valley smoothing sensitivity	
	2	**4**
Total Path(m)	7139.51	8515.79
Min distance(m)	9.87	10.63
Max distance(m)	226.51	191.72
Min angle(degrees)	20.60	5.09
Max angle(degrees)	179.889	179.99

can be seen in Table I. In order to compare these results with a previous algorithm presented in [12], tests have been conducted for three different Lloyd smoothing settings: 10, 30 and 60 iterations. The algorithm has been tested with a valley sensitivity setting of 6. The upper panel of Fig. 7 shows that the distribution dispersion of distances between each cell is wider. This results in longer total path lengths, as it is also shown in Table II. In the lower panel, the distribution curve shows that in average the new algorithm also manages to produce smoother paths, by increasing the angles between each three waypoints.

2) Realistic scenario: The area presented in Fig. 1 has been selected as a realistic scenario for a mission. In Fig. 8, the qualitative difference in the calculated waypoint list path can be identified and Fig. 9 shows the angle distribution in the two different cases; it shows that the dispersion of angles is greater in the smaller sensitivity but the produced total paths are smaller. On the other hand, the bigger sensitivity setting provides wider angles, thus smoother paths. Then, the total metrics of the produced paths can be seen in Table III.

As it can be seen from Fig. 9, both increased Lloyd

TABLE II

COMPARISON BETWEEN THE ALGORITHM IN [12] AND THE ONE IN THIS PAPER, CORRESPONDING TO FIG. 7

Algorithm	**Lloyd**	**Total path (m)**	**Distance Min (m)**	**Distance Max (m)**
Previous	10	7736,95	5,77	255,15
	30	7322,59	6,02	290,29
	60	7502,25	12,99	233,97
Current	10	6958,26	10,41	144,13
	30	7144,34	12,99	174,32
	60	7098,88	12,99	94,11

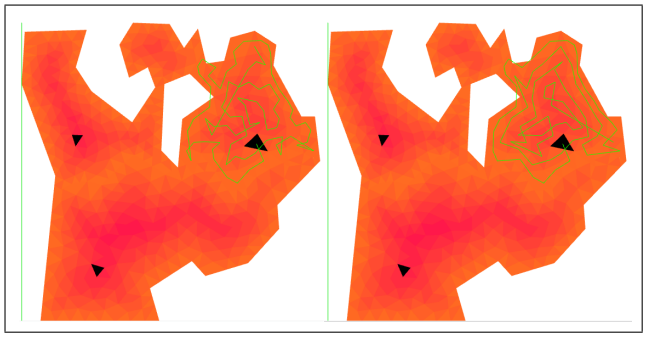

Fig. 8. The difference in coverage path planning can be identified mainly in the number of sharp turns, repeated paths and total path length. The image on the left represents a sensitivity setting of two whereas the one on the right a sensitivity setting of four. The total metrics of the example can be seen in Table III.

iterations and valley sensitivity metrics manage to increase the average angle metric. The actual simulated trajectory can be seen in Fig. 10 and a screenshot of the simulated experiment in Fig. 11.

B. Configuration comparison

Regarding the use of Lloyd optimization for creating a smoother grid, results show that the angle distribution of the grid is more coherent with increased number of iterations. Nevertheless, this operation is strictly dependent on the area input as well as the constraints of the CDT. In the distribution shown in Fig. 12, in the 100 iterations case, the algorithm did not manage to overcome the 65 iterations thus not been able to smooth the grid any more.

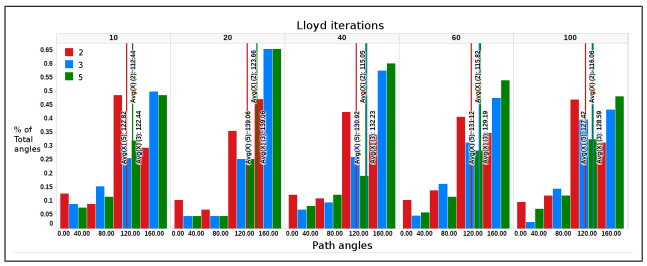

Fig. 9. The angle distribution in the produced path of the Abaco region. Red, blue and green colors represent the different valley sensitivity settings of 2,3 and 5 respectively. Each panel shows the different Lloyd iteration settings. The horizontal axis distributes the angles in 5 bins, whereas the vertical axis show the percentile rank of each bin. Vertical lines show the average angle of each case.

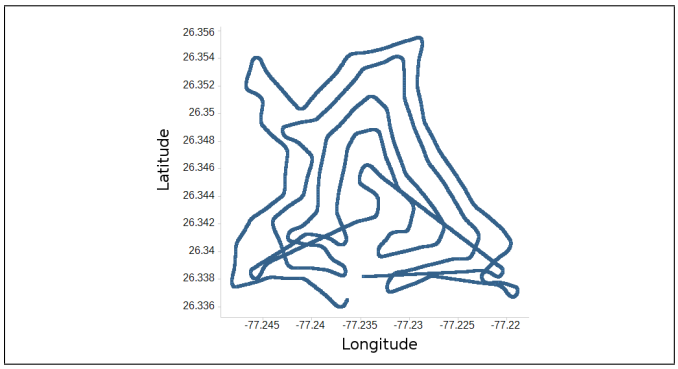

Fig. 10. The actual trajectory when the simulated model followed the produced waypoint plan.

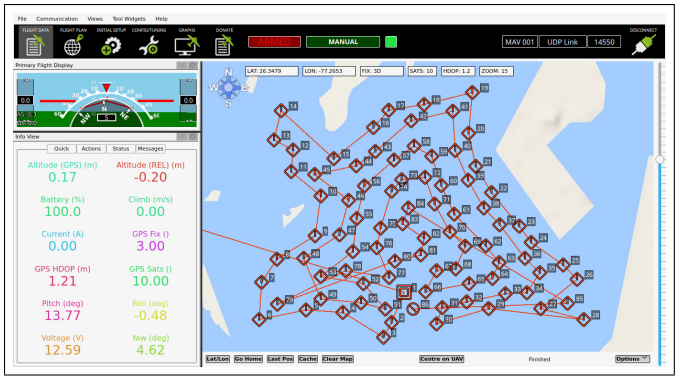

Fig. 11. The APM Planner 2 ground control station showing the loaded waypoints. The mission had a flight setup of 100 meters altitude.

TABLE III

METRICS ON VALLEY SENSITIVITY SETTING CORRESPONDING TO FIG. 8

	Valley smoothing sensitivity		
	2	3	5
Total Path(m)	3807.88	4185.54	4344.05
Min distance(m)	12.90	12.90	12.90
Max distance(m)	216.87	127.52	118.24
Min angle(degrees)	11.5629	11.69	0.8535
Max angle(degrees)	179.667	179.667	179.913

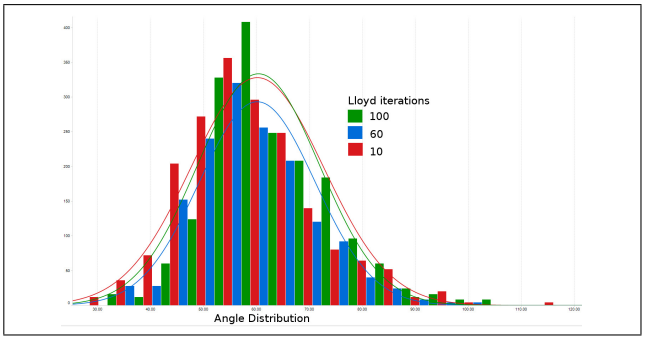

Fig. 12. Angle distribution in different Lloyd iteration trials. The red Gaussian curve which represents the lesser number of iterations tested, shows that there is a larger dispersion in angles, resulting in more "sharp" cells. While the best performance is achieved by the 100 iterations, where there are more angles reaching the 60^o ideal case, the required computation time should be considered. Depending on the application, a solution closer to 50 iterations could be competent enough.

TABLE IV

TIME IN SECONDS FOR LLOYD OPTIMIZATION IN THE WHOLE AREA AND IN EACH UAS CONFIGURATION SPACE

	Lloyd Iterations		
CDT	10	60	100
All	0.2	0.86	1.008
UAS1	0.076	0.36	0.32
UAS2	0.064	0.316	0.392
UAS3	0.156	0.552	0.6

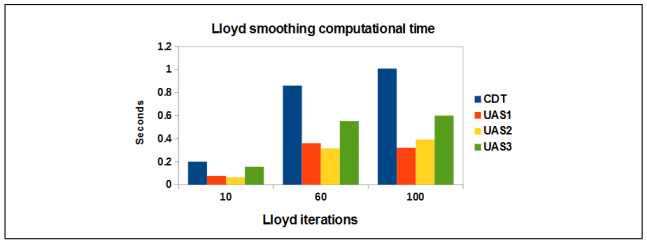

Fig. 13. Time for Lloyd smoothing to complete. CDT is the initial triangulation before the partitioning process. Each one the configuration spaces might have different optimization times due to the area complexity.

The complexity effect of the algorithm is clear in Table IV and Fig. 13, where the time each smoothing procedure needs significantly increases, thus exposing the trade off among sample overlapping and smoother or shorter paths.

VII. CONCLUSIONS AND FUTURE WORK

This paper has introduced an offline algorithmic approach which manages to decompose, partition and produce initial coverage waypoint lists for complex coastal regions, in a heterogeneous multi-UAS context. The strategy has shown to be consistent with complex area characteristics as well as properties of the UAS and their sensors.

The continuous space can be treated as a sum of FoV projection triangular cells. The extracted lists create a waypoint flight plan which evaluates the actual path as straight lines between the waypoints. In that manner a transition to the continuous space can be introduced by a Dubins path calculation model between each pair of the waypoints.

The actual UAS trajectory is a matter of the platform itself as well as the environmental conditions. In many experimental setups, as the one in this study, these issues are treated by the on-board autopilot. However, a platform might be able to follow the produced trajectories, in case of a multirotor for example, or might not, in case of a fixed wing. These metrics should be introduced in the waypoint lists extraction, by introducing a cost function which takes into consideration even more kinematic constraints and UAS characteristics.

Regarding future work, the uncertainties in motion and sensing should will be encoded in each of the segregated cells and vertices in order to implement the algorithms in online systems through on-board computation of the trajectory. The dynamic replanning issue in real time systems is crucial since, for instance, changes in the swarm, like

UAS communication loss or in the tasks appointed, have to dynamically reevaluate the initial programming.

The algorithms presented here, could be expanded in finding the shortest path for a given goal area. To do so, common shortest path algorithms in literature have to be implemented in the solution so to increase the energy autonomy of the platforms and improve the produced trajectories. The proposed method can be compared with several other segregation, coverage and shortest path methods, in order extract useful performance metrics.

Finally, the aforementioned extension as well as the current implementation have to be tested in a real world environment with actual platforms, performing the calculations on-board. This will lead to an uncertainty analysis and comparison between the predicted trajectories and the actual ones.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.09.004

[2] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/12/1495.abstract

[3] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," A. Koubâa and J. Mart\'inez-de Dios, Eds. Cham: Springer International Publishing, 2015, ch. Multi-robo, pp. 31–51. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-18299-5_2

[4] M. Hassan, D. Liu, S. Huang, and G. Dissanayake, "Task oriented area partitioning and allocation for optimal operation of multiple industrial robots in unstructured environments," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE, dec 2014, pp. 1184–1189. [Online]. Available: http://ieeexplore.ieee.org/document/7064473/

[5] G. E. Jan, C. C. Sun, W. C. Tsai, and T. H. Lin, "An O(n log n) Shortest Path Algorithm Based on Delaunay Triangulation," pp. 660–666, 2014.

[6] P. Fazli, A. Davoodi, and A. K. Mackworth, "Multi-robot repeated area coverage," *Autonomous Robots*, vol. 34, no. 4, pp. 251–276, may 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9319-7

[7] C. Di Franco and G. Buttazzo, "Coverage Path Planning for UAVs Photogrammetry with Energy and Resolution Constraints," *Journal of Intelligent {&} Robotic Systems*, vol. 83, no. 3, pp. 445–462, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10846-016-0348-x

[8] I. A. Hameed, D. Bochtis, and C. A. Sørensen, "An Optimized Field Coverage Planning Approach for Navigation of Agricultural Robots in Fields Involving Obstacle Areas," *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, p. 231, may 2013. [Online]. Available: http://journals.sagepub.com/doi/10.5772/56248

[9] F. Balampanis, I. Maza, and A. Ollero, "Area Partition for Coastal Regions with Multiple UAS," *Journal of Intelligent & Robotic Systems*, 2017. [Online]. Available: http://link.springer.com/10.1007/s10846-017-0559-9

[10] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, aug 2016, pp. 1237–1242. [Online]. Available: http://ieeexplore.ieee.org/document/7743548/

[11] "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 801–812, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2011.06.002

[12] F. Balampanis, I. Maza, and A. Ollero, "Coastal Areas Division and Coverage with Multiple UAVs for Remote Sensing," *Sensors*, vol. 17, no. 4, p. 808, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/4/808

[13] J.-d. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec, "Triangulations in CGAL," *Computational Geometry*, vol. 22, no. 1-3, pp. 5–19, may 2002. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0925772101000542

[14] "Cgal - 2d conforming triangulations and meshes - 2.5 optimization of meshes with lloyd," http://doc.cgal.org/latest/Mesh_2/index.html#secMesh_2_optimization, 2016.

[15] Y. Li, H. Chen, M. Joo Er, and X. Wang, "Coverage path planning for UAVs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, 2011.

[16] M. Pepe and G. Prezioso, "Two approaches for dense dsm generation from aerial digital oblique camera system," in *Proceedings of the 2nd International Conference on Geographical Information Systems Theory, Applications and Management*, 2016, pp. 63–70.

[17] L. Paull, C. Thibault, A. Nagaty, M. Seto, and H. Li, "Sensor-Driven Area Coverage for an Autonomous Fixed-Wing Unmanned Aerial Vehicle," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–1, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6671976

[18] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[19] T. C. Project, *CGAL User and Reference Manual*, 4th ed. CGAL Editorial Board, 2015. [Online]. Available: http://doc.cgal.org/latest/Manual/packages.html

[20] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, may 2015.