Area Coverage with Heterogeneous UAVs using Scan Patterns

Cyrille Berger[1], Mariusz Wzorek[1], Jonas Kvarnström[1],
Gianpaolo Conte[1], Patrick Doherty[1] and Alexander Eriksson[1]

***Abstract*— In this paper we consider a problem of scanning an outdoor area with a team of heterogeneous Unmanned Air Vehicles (UAVs) equipped with different sensors (e.g. LIDARs). Depending on the availability of the UAV platforms and the mission requirements there is a need to either minimise the total mission time or to maximise certain properties of the scan output, such as the point cloud density. The key challenge is to divide the scanning task among UAVs while taking into account the differences in capabilities between platforms and sensors. Additionally, the system should be able to ensure that constraints such as limit on the flight time are not violated. We present an approach that uses an optimisation technique to find a solution by dividing the area between platforms, generating efficient scan trajectories and selecting flight and scanning parameters, such as velocity and flight altitude. This method has been extensively tested on a large set of randomly generated scanning missions covering a wide range of realistic scenarios as well as in real flights.**

I. Introduction

Unmanned Air Vehicles (UAVs) are becoming a platform of choice in many real world applications thanks to recent advances in both sensor technology and hardware platform development. Off-the-shelf UAV platforms available today are capable of carrying sufficient payloads and offer reasonable flight times. They are becoming a solution of choice in search and rescue operations covering large operational areas (several km^2) such as rescue operations after a tsunami or searching for lost persons in a mountain environment, such as described in the SHERPA project [13]. A common mission in such scenarios involves gathering information (e.g. building a 3D model) in a rescue region that will be used for planning and executing future actions taken by rescuers. In many rescue scenarios the exploration needs to be performed in a short time, as victims survival chances drop significantly a few hours after injury. Additionally, the areas involved are large thus using a team of UAVs instead of a single platform is beneficial to fulfil the mission goal more efficiently. Larger teams can be created when heterogeneous platforms are available to the rescuers decreasing the mission times even further.

In this paper we consider the exploration problem using a team of heterogeneous UAVs. The team (e.g. Figure 1) is delegated [2] with a task of gathering sensor data over a specified region. Using multiple platforms to explore an area of the environment require to coordinate the work of each agent including optimal partitioning of the region (relative to sensor capabilities of each UAV) and generation of optimal scan trajectories.

Fig. 1: The RMAX (left) and the LinkQuad (right) platforms.

The problem of exploration with a team of homogeneous agents can be solved using the frontier cell approach [3]. The world is represented as a grid with three values, unexplored, free space or obstacle. Heuristics have been developed to spread the team of robots to go and visit unexplored cells. The frontier cell approach has been extended to handle heterogeneous agents as well using integer programming to represent constraints between platforms [5], for instance if a UAV needs assistance from a ground robot to improve its localisation. The main benefit of the frontier cell approach is the ability to handle obstacles, however it requires constant synchronisation between platforms to keep the map up-to-date. A clear drawback is its scalability i.e. grid based approaches do not scale well for large areas. In outdoor environments which are considered in this paper, UAVs are expected to cover large areas and will fly over obstacles without the need to keep knowledge of the obstacles during exploration thus the frontier based approach is not the best choice.

Guaranteed search [15] is a type of exploration with multiple UAVs in which it is assumed that the searched object is trying to evade detection. Algorithms for guaranteed search assume that the object is actively trying to avoid detection. Those algorithms offer the guarantee to find the object by positioning UAVs to protect areas that have already been explored. Both 2D [15] and 3D [11], [10] flavours of the algorithms have been proposed. These algorithms require large number of platforms to work effectively thus making it very cost ineffective in the context of scenarios considered here (e.g. mountain rescue). In such scenarios we assume that victims want to be found or that the objects in the environment are static and therefore we do not have to detect whether victims are moving to areas which have already been

[1] the authors are with Linköping University, Department of Computer and Information Science, 581 83 LINKÖPING, SWEDEN, firstname.lastname@liu.se

This work is partially supported by the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT network organisation for Information and Communication Technology, and the Swedish Foundation for Strategic Research (CUAS Project, SymbiKcloud Project).

explored.

Another solution to the exploration problem is to divide the area and to assign each UAV to explore a subarea. In [8], Hert introduces an algorithm to decompose a polygon into a set of subpolygons. The surface of each subpolygon can be controlled to give larger area to more capable UAVs. This decomposition algorithm is used for exploration by a homogeneous team of UAVs in [14].

In [9], Huang suggests that a good trajectory for scanning a polygonal area follows a lawn mower pattern (Figure 2). In general scanning in straight lines yields a more uniform point density, which improves quality. The lawn mower pattern also allows the number of turns to be minimised for certain types of polygons, which is generally both faster and more convenient regardless of the type of aircraft being used. Huang also proposes an algorithm to generate a trajectory for convex polygons.

In this paper we propose a new trajectory generation algorithm which works directly on non-convex polygons, unlike [9] which requires splitting the polygonal area. With heterogeneous agents, one of the challenges is that the different platforms have different capabilities, they might be able to fly at different speeds, cover different surfaces on the ground and will need to scan areas of different sizes. We propose an extension of [14] to handle a team of heterogeneous UAVs using an optimisation technique to select the best parameters for the polygon decomposition and for the flight parameters. Our optimisation technique is able to find a set of parameters that satisfy the mission requirements given by the operator, for instance to minimise total mission time or maximise the quality of the scan output. Additionally, the proposed algorithm can also take into account constraints, such as maximum flight time.

In the first part of this paper, we present the flight and sensor parameters used to model the scenario when using UAVs equipped with LIDAR sensors. In Section III, we present how the scan area is divided among the team of UAVs. Scanning trajectory generation is described in Section IV. In Section V, we present a heuristic and two optimisation algorithms used to select the parameters. Finally in Section VI we show results on how a mission is generated and a statistical analysis over a large number of randomly generated missions.

II. Parameters for scanning with a LIDAR sensor

The size of the area that can be scanned in a particular amount of time depends on the flight envelope of the aircraft performing the mission, as well as various parameters related to the sensors being used. This in turn affects both the optimal partitioning of a region and the way in which scan patterns should be generated. In this paper, we considered the use of LIDAR sensor, but very similar parameters can be used for other sensor types, such as cameras.

For many of the scan types we are interested in here, such as LIDAR scans, flying in straight lines yields more uniform results. It is also important to reduce the number of turns [9], as this reduces the amount of time the platform spends turning instead of actively scanning a previously unscanned part of the region, especially when flying with fixed-wing aircraft.

Typical parameters related to scan missions with a LIDAR sensor mounted downwards onboard a UAV are illustrated in Figure 2. Those include sensor parameters which can be set directly depending on the LIDAR manufacturer. For example, in the case of the LIDAR sensor used on our RMAX platforms: *Max Range* (R_{max}), *FOV*, β and *f*. Where, the R_{max} is the cut-off range for a single ray originating in the LIDAR sensor. Distance measurements higher than this value are discarded due to insufficient precision and the risk of misdetection. The *FOV* is the angular field of view of the scanner. The β is the angular resolution and f is the scanning frequency of the sensor, which corresponds to the number of complete scan lines per second. The Ground line in Figure 2 can be said to represent a single such scan line, consisting of a number of samples.

In addition to the sensor parameters discussed above, both coverage and scan quality depend on UAVs velocity (v) and its flight altitude relative to the ground (h).

Several additional parameters of a LIDAR scan can be derived which cannot be set or configured directly, but they indirectly affect the range of permissible values for the true parameters discussed above. Those will be used in the proposed optimisation algorithm and include *distance between flight lines* (d) and *average point density* ($avDensity$).

The parameter d specifies the distance between two flight lines as illustrated in Figure 2:

$$d = sw \cdot (1 - ov); sw = 2 \cdot \sqrt{R_{max}^2 + h^2} \qquad (1)$$

where, the *swath* (sw) is the theoretical "width" of each scan line. The swath is also equal to the width of each *strip* of LIDAR or image data being collected. This provides an upper bound on the permitted distance between *flight lines* in the final flight trajectory and altitude. The distance d must not exceed the swath, but can be lower in order to ensure some overlap ov between two consecutive scan strips.

The parameter $avDensity$ is the average density of measurements on the ground, measured in points per m^2. This represents the level of detail of the point cloud generated by the scan mission. The value depends on the sensor configuration, flight velocity v and flight ground altitude h:

$$avDensity = \frac{eFov \cdot f}{\beta \cdot d \cdot v}; eFov = 2 \cdot \arctan(\frac{d}{2 \cdot h}) \qquad (2)$$

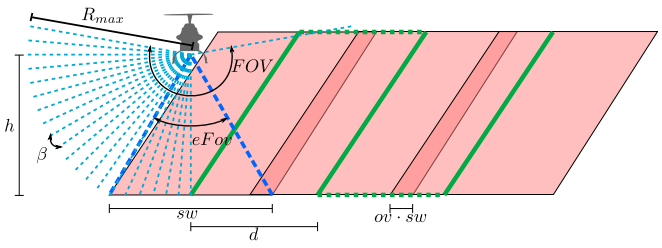


Fig. 2: Parameters for a 2D LIDAR sensor mounted downwards

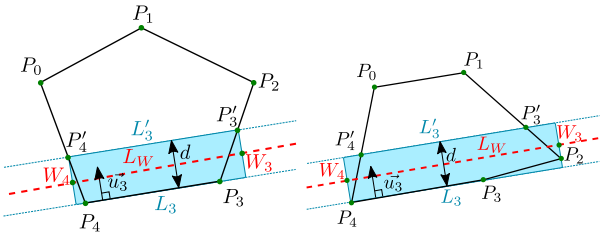


Fig. 3: Examples of scan strip generation along the longest edge of a polygon $\mathcal{P}$.

where, the *effective field of view* ($eFov$) is directly related to the swath and the ground altitude h.

In some cases, these parameters are calculated under the assumption that the ground is flat. This is a necessary approximation for the case where we do not yet know the elevation of the ground that we are scanning. True values may therefore differ somewhat from the theoretical calculations. This can be taken into account by strengthening requirements, for example by reduction of the maximum permitted ground sample distance before calculating the required parameters.

III. Partitioning a Scan Region

The *Polygon Area Decomposition* algorithm [8] is a state-of-the-art algorithm for partitioning a polygon to a fixed set of regions. Each subpolygon (partition) is anchored to a site, a specific location that corresponds to the starting position of a participating UAV. The algorithm was developed with the intention to be used for partitioning a work area between different robots and has been used successfully for UAV scanning missions [14]. The input to the algorithm consists of the following parameters:

- A polygon $\mathcal{P}$ corresponding to the mission region selected by the rescuer. The polygon does not necessarily have to be convex.
- A set of weights u_i, each representing the proportion of the mission region assigned to participant i.
- The site S_i for each UAV, which anchors the subpolygon P_i on the polygon $\mathcal{P}$.

IV. Trajectory Generation and Time Estimation

The trajectory generator is a recursive algorithm which calculates a flight line along the longest side of the remaining region to be scanned in each iteration. For rectangles, this turns into a lawn mower pattern which yields the best results in our scenarios.

A polygon $\mathcal{P}$ is defined in terms of a sequence of vertices $\langle P_0 ... P_{n-1} \rangle \in (\mathbb{R}^2)^n$. Figure 3 shows two examples, both of which have $n = 5$ vertices. We introduce the following notation:

- $\overline{k}$ denotes k mod n. Thus, assuming $n = 5$, we have $P_{\overline{n+1}} = P_1$.
- L_i is the infinite line going through the points P_i and $P_{\overline{i+1}}$.

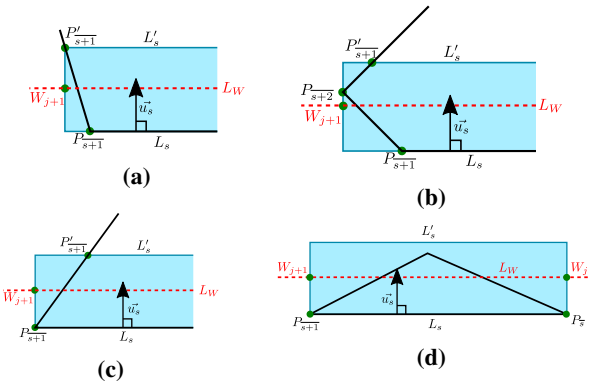


Fig. 4: Waypoint generation process depending on where the line L'_i intersects with the polygon $\mathcal{P}$.

- $d_i = dist(P_{\overline{i}}, P_{\overline{i+1}})$ is the linear distance between $P_{\overline{i}}$ and $P_{\overline{i+1}}$. For example, d_4 is the linear distance between P_4 and $P_{\overline{5}} = P_0$.

The trajectory generation algorithm $generate - trajectory(\mathcal{P})$ returns a list of flight line coordinates by executing the following steps:

Step 1: Select the scan direction to be parallel to a longest segment L_s of the polygon $\mathcal{P}$, so that $\forall i. d_i \leq d_s$. Selecting new direction in every iteration minimises the number of turns.

Step 2:Derive the outer boundary of the scan line L'_s. Let $\vec{u_s}$ be a vector perpendicular to L_s, pointing toward the interior of the polygon $\mathcal{P}$, as illustrated in Figure 3. The L'_s is the translation of L_s in the direction $\vec{u_s}$ by the distance d (see Equation (1)). The L_s and L'_s provide two of the edges for the non-overlapping part of a scan strip.

Let the line L_W be the translation of L_s in the direction $\vec{u_s}$ by the distance $\frac{d}{2}$. This line is positioned in the middle of the swath and a finite segment of the line will be followed by the UAV while scanning.

Step 3: Find the next two waypoints W_j and W_{j+1} which represent the start and end of the flight line that is being generated.

Given a convex polygon, there are two possible cases.

CASE I: Suppose the line L'_s intersects $\mathcal{P}$ in *two* points, as is the case for L'_4 in both polygons in Figure 3. We will call these points $P'_{\overline{s}}$ and $P'_{\overline{s+1}}$ (specifically, P'_3 and P'_4 in Figure 3).

We can find W_{j+1} by considering the "leftmost" point of $P_{\overline{s+1}}$, $P'_{\overline{s+1}}$ and any vertices that may intervene between these two points on the polygon (see Figure 4a, 4b and 4c). A similar situation applies at P'_s, the other end of L'_s. We therefore introduce the following notation in order to gather all such points and vertexes at *both* ends of the scan strip:

- $\mathcal{P}'_{s \to s+1}$ is defined as a set containing P'_s, $P'_{\overline{s+1}}$, and all vertices located between these two points on the polygon in the direction of increasing vertex indexes. For example, in the first polygon in Figure 3, $\mathcal{P}'_{s \to s+1} = \mathcal{P}'_{3 \to 4} = \{P'_3, P_3, P_4, P'_4\}$. Similarly, in the second polygon, $\mathcal{P}'_{s \to s+1} = \mathcal{P}'_{3 \to 4} = \{P'_3, P_2, P_3, P_4, P'_4\}$.
- For each point $\hat{p} \in \mathcal{P}'_{s \to s+1}$ (where the hat indicates that

the point may or may not be a vertex), we define $W(\hat{p})$ as its orthogonal projection onto the infinite flight line L_W. The first polygon in Figure 3 uses $W_5 = W(P'_5)$ to illustrate the orthogonal projection of P'_5 onto L_W. The second polygon instead uses $W_5 = W(P_5)$ to illustrate the orthogonal projection of P_5 onto L_W.

- $\mathcal{W} = \{W(\hat{p}) \mid \hat{p} \in \mathcal{P}'_{s \to s+1}\}$ contains the orthogonal projections of all points in $\mathcal{P}_{s \to s+1}$ onto the infinite potential flight line L_W.

The two waypoints W_j and W_{j+1} are selected in $\mathcal{W}$ so that they are maximally distant from each other: $\forall A, B \in \mathcal{W} \times \mathcal{W}, dist(W_j, W_{j+1}) \geq dist(A, B)$. This results in a new flight line $[W_j, W_{j+1}]$ for the aircraft in question. When the flight line is flown, a strip corresponding to the rectangle of width d highlighted in blue in each figure will be effectively covered. The rectangle of width d should now be removed from the polygon $\mathcal{P}$, resulting in a new polygon $\mathcal{P}'$ representing the region that remains to be covered.

$\mathcal{P}'$ is created by removing all the vertices in $\mathcal{P}_{s \to s+1}$, replacing them with P'_s and $P'_{\overline{s+1}}$. The first polygon in Figure 3 is then defined as $\langle P_0, P_1, P_2, P'_3, P'_4 \rangle$, while the second polygon is defined as $\langle P_0, P_1, P'_3, P'_4 \rangle$, having fewer vertices than before.

The *generate-trajectory* algorithm is then called for the new polygon (unless it is empty).

CASE II: Suppose that the line L'_s has no intersection with the polygon $\mathcal{P}$, which indicates that the remaining part of the polygon will be completely covered after generating this last scan strip (see Figure 4d). In this case, $\mathcal{W} = \{W(\hat{p}) \mid \hat{p} \in \mathcal{P}\}$ contains the orthogonal projections of all the vertexes from the polygon $\mathcal{P}$ onto the infinite potential flight line L_W. The last two waypoints W_j and W_{j+1} are selected in $\mathcal{W}$ so that they are maximally distant from each other: $\forall A, B \in \mathcal{W} \times \mathcal{W}, dist(W_j, W_{j+1}) \geq dist(A, B)$.

This also applies when L'_s intersects the polygon in exactly one point: It corresponds to the case where the "topmost" vertex in Figure 4d is reached, but does not extend beyond, L'_s.

This generates a set of flight lines $\{(W_0, W_1), ...(W_{m-1}, W_m)\}$, which the aircraft can cover in any order. Typically, a helicopter would follow them in the order they are defined, but for a fixed wing it might be better to skip a flight line and then come back to it later in order to accommodate a need for a larger turning radius.

The flying time $t(W_0, \ldots, W_m)$ can be estimated assuming a constant acceleration and deceleration model and constant yawing rate.

V. Parameters Selection

The challenging part of generating high-quality results using the proposed algorithm is setting the weights u_i and the sites S_i, as the choice of those parameters will affect the mission time and the density of the generated point clouds. If the platforms had homogeneous capabilities it would be possible to split the area equally (EQ) between platforms.

A. Heuristic

A straightforward approach is to use a heuristic (HEURI) that allocates larger regions to faster platforms and to platforms with a larger swath. For each platform the altitude must also be selected within the platform's safety limits. For a fixed desired point density average and a fixed sensor, the altitude will directly define the platform's velocity according to Equation (2). The weights are then selected as follows:

$$u_i = \alpha \cdot v_i \cdot sw \tag{3}$$

Here α is a normalization factor ensuring that $\sum_{i=1}^{n} u_i = 1$. Finally, the site S_i for each participant i can be selected as a point on the border of the original scanning region that minimises the distance to the current location of the UAV: $S_i \in \mathcal{P}$ and $\forall P \in \mathcal{P}$, $dist(UAV_i, S_i) \leq dist(UAV_i, P)$.

B. Optimisation algorithm - Covariance Matrix Adaptation Evolution Strategy

The heuristic approach described above is approximate and does not offer a guarantee to find an optimal solution. It also does not take into account constraints such as the flight endurance of a platform, or if the platform is only available for a limited amount of time due to other commitments. A better approach would be to use an optimisation technique to estimate the area weights, which would allow us to take constraints into account and to optimise an objective.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [7] is an evolutionary optimisation technique that has grown in popularity in recent years. It has been successfully applied to many problems where the gradient of the cost function could not be computed directly and where numerical computation would be expensive. CMA-ES spreads the search of the parameters, so that problems with local minima can be avoided. The strategy can estimate the correlation between parameters and detect when the parameters are independent.

CMA-ES can be used to search for a parameter $\mathbf{x}$ among a set of possible parameters $\chi \subseteq \mathbb{R}^n$ minimising the value of the cost function $f : \chi \longrightarrow \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$ without any specific knowledge or restrictions on the function itself. For example, f can be non-convex, multimodal, non-smooth, discontinuous, noisy, or ill-conditioned.

The cost functions $f(\mathcal{X})$ considered in our scenarios represent either the mission total time or the point density average (PDA). Both functions are not known to be continuous or derivable and their evaluation is done by running the actual partitioning and trajectory generation algorithms. Additionally, solutions should satisfy given constraints (e.g. maximum velocity) and since some of those are non-linear, the general form for the constrained minimisation problem is used:

$$\begin{aligned} \underset{\mathcal{X}}{\text{minimise}} \quad & f(\mathcal{X}) \\ \text{subject to} \quad & g_i(\mathcal{X}) = c_i, \qquad i = 0, \ldots, n \\ & h_j(\mathcal{X}) \leq d_j, \quad j = 0, \ldots, m. \end{aligned}$$

A penalty function used to solve the constraint optimisation problem is defined as:

$$\pi(\mathcal{X}) = \sum_{i=0}^{n} |g_i(\mathcal{X}) - c_i| + \sum_{j=0}^{m} max(0, h_j(\mathcal{X}) - d_j) \quad (4)$$

The constraint problem is a regular optimisation problem:

$$\underset{\mathcal{X}}{\text{minimise}} \quad f(\mathcal{X}) + (\pi_c \cdot \pi(\mathcal{X}))^2$$

We assume n UAVs, where UAV $k \in \{1, \ldots, n\}$ is located at position x_k^0. For each platform four parameter values need to be found by the optimisation procedure:

- $u_k \in \mathbb{R}$ is the percentage of the mission area that should be allocated to this platform.
- $s_k \in \mathbb{R}^2$ is the starting point of the platform (represented as curvilinear coordinates along the polygon).
- $h_k \in \mathbb{R}$ is the scanning altitude.
- $v_k \in \mathbb{R}$ is the UAV flight velocity.

The parameter space is $\chi = \mathbb{R}^{4n}$, where each member of $\mathcal{X}$ is a tuple $\langle u_1, ..., u_n, s_1, ..., s_n, h_1, ..., h_n, v_1, ..., v_n \rangle$. The objective function is a function from χ to $\mathbb{R}$. Minimising this function entails finding a single *combination* of percentages, starting points, scanning altitudes and flight velocities for all involved UAVs that optimises a specific aspect of the mission.

Optimising for time with a fixed PDA (OTIME)

To minimise the scanning time given a fixed PDA, assume the following functions and constants as given:

- PDA is the desired point density average.
- t_k^{max} is the maximum flight time for platform k.
- v_k^{min} is the minimum velocity for platform k. This is mainly necessary for fixed-wing aircraft.
- v_k^{max} is the maximum velocity for platform k.
- $v_k(PDA, h_k)$ is the velocity at which platform k achieves the given PDA at the given scanning altitude. The selected velocity must not exceed this, but may be lower. This ensures that specifying a low PDA does not result in a commanded velocity that is greater than the maximum permitted, simply because a sufficient number of points could have been captured even at a very high speed. Instead, specifying a very low PDA results in data being captured at a greater density than required.
- $t_k(\mathbf{x})$ is the time required for platform k to fly the scan trajectory it is assigned when the partitioning algorithm and scan pattern generator are applied to the parameters specified in $\mathbf{x}$ (which includes the selected velocity of platform k). This function is assumed to include the time required to fly from the current location of the UAV to the beginning of the scan pattern.
- h_k^{min} and h_k^{max} are the maximum altitude for platform k and the maximum altitude is constrained by the sensor parameters: $h_k^{max} < R_k^{max}$

The optimisation problem to be solved is the following, where $\mathbf{x} = \langle u_1, ..., u_n, s_1, ..., s_n, h_1, ..., h_n, v_1, ..., v_n \rangle$:

	MAX Velocity	MAX Acceleration	MAX Yaw Rate
RMAX	$5.0m/s$	$1.2m/s^2$	$40°/s$
LinkQuad	$2.0m/s$	$1.0m/s^2$	$20°/s$
	MAX Fly time	MIN Altitude	MAX Altitude
RMAX	$45min = 2700s$	$30m$	$50m$
LinkQuad	$15min = 900s$	$5m$	$30m$

TABLE I: Platform parameters

	MAX Range	Frequency	Angular Resolution
SICK LM-511	$80m$	$50Hz$	$0.5°$
Hokuyo UTM-30LX	$30m$	$40Hz$	$0.25°$

TABLE II: LIDAR sensor parameters for each platform

$$\begin{aligned} \underset{\mathbf{x} \in \mathcal{X}}{\text{minimise}} \quad & \max_{k=1}^{n}(t_k(\mathbf{x})) \\ \text{subject to} \quad & t_k(\mathbf{x}) \leq t_k^{max},\ k = 1, \ldots, n \\ & v_k^{min} \leq v_k \leq v_k^{max},\ k = 1, \ldots, n \\ & v_k \leq v_k(PDA, h_k)\ k = 1, \ldots, n \\ & h_k^{min} \leq h_k \leq h_k^{max},\ k = 1, \ldots, n \end{aligned}$$

OTIME can be initialised with an equal parameter selection (OTIME-EQ) or using the heuristic (OTIME-HEURI).

Optimising for PDA with constrained time (OPDA)

To maximise the achieved PDA given a specific time limit assume the following functions and constants are given, in addition to the ones specified above:

- τ^{max} is the time limit for the entire mission.
- $PDA_k(\mathbf{x})$ is the point density average that platform k can achieve given the specified parameters.

The optimisation problem to be solved is the following, where $\mathbf{x} = \langle u_1, ..., u_n, s_1, ..., s_n, h_1, ..., h_n, v_1, ..., v_n \rangle$:

$$\begin{aligned} \underset{\mathbf{x} \in \mathcal{X}}{\text{maximise}} \quad & \min_{k=1}^{n}(PDA(\mathbf{x})) \\ \text{subject to} \quad & v_k^{min} \leq v_k \leq v_k^{max},\ k = 1, \ldots, n \\ & t_k(\mathbf{x}) \leq \tau^{max},\ k = 1, \ldots, n \\ & t_k(\mathbf{x}) \leq t_k^{max},\ k = 1, \ldots, n. \end{aligned}$$

OPDA can be initialised with an equal parameter selection (OPDA-EQ) or using the heuristic (OPDA-HEURI).

VI. Results

Two types of platforms are used for the experimental evaluation, a RMAX helicopter and a LinkQuad quadcopter (Figure 1). Table I presents each platform characteristics. The RMAX helicopter is equipped with a SICK LM-511 LIDAR sensor and the LinkQuad with a Hokuyo UTM-30LX, whose characteristics are given in Table II.

In our experiments, the cost functions $f(\mathcal{X})$ used in the optimisation typically have values below 10000. We therefore selected $\pi_c = 1000000^2$, which strongly discourages the optimisation from violating the constraints.

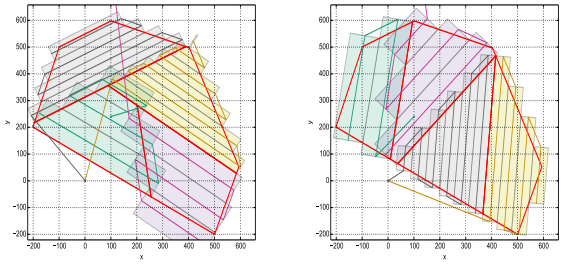
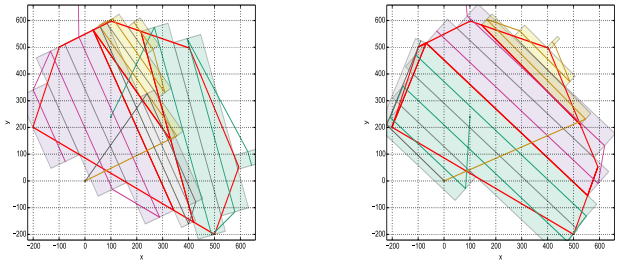
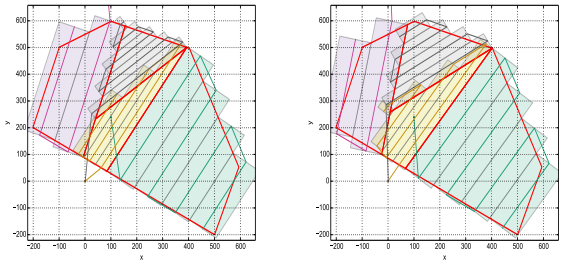


Fig. 5: The scanning mission results: equal areas (EQ, left), heuristic (HEURI, right). The red lines represent the subdivision. The coloured polylines represent the trajectory of each UAVs. The coloured rectangles correspond to the area scanned on the ground for each scanline.



(a) After 5 iterations **(b)** After 10 iterations



(c) After 100 iterations **(d)** After 1000 iterations

Fig. 6: Result of OTIME at different iterations

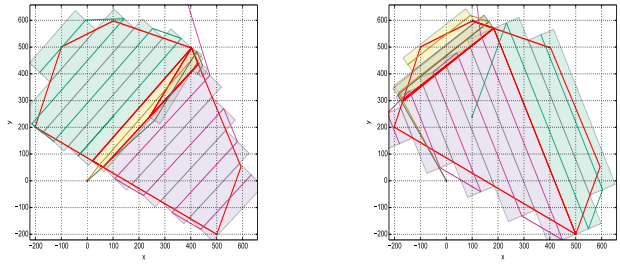
First, an example scenario with in-depth analysis of the proposed algorithms is presented. Then, statistical analysis of results for 700 random experiment runs is described.

A. In-depth analysis of an example experiment

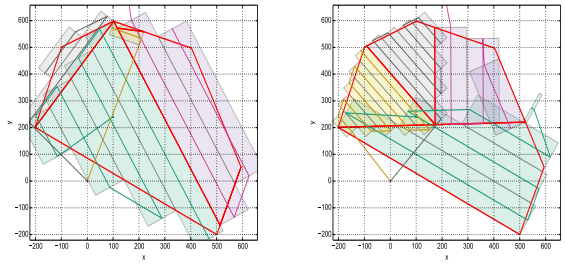
In this example two RMAX helicopters and two LinkQuad quadcopters (LQ) are used. One of the RMAX helicopters starts the mission $2km$ away from the scan area, the second RMAX is near the middle, and the two quadcopters are at its origin.

Results of partitioning and trajectory generation when using EQ or HEURI are shown in Figure 5. Figure 6 and Figure 7 depicts different stages of the optimisation for OTIME (PDA set at 60) and for OPDA ($33min$ time limit) respectively.

Table III shows the percentage of area assigned for each UAV as well as the expected execution time and the resulting PDA. For the equal area division (III-a), the mission is



(a) After 5 iterations **(b)** After 10 iterations



(c) After 100 iterations **(d)** After 1000 iterations

Fig. 7: Result of OPDA at different iterations

not feasible due to flight time limits of a small LinkQuad platform (41 and 46 minutes required). Using heuristic approach (III-b) flight times for each UAV are reduced and feasible but only a fixed PDA of 60 is achieved. More optimal solutions are found when using OTIME and OPDA optimisations (III-c, III-d). For OTIME, the mission will be finished within 21 minutes with PDA of 60 and more than 110 for RMAX and LinkQuad platforms respectively. Optimising for PDA, results in a mission of up to 33 minutes duration and higher PDA per platform. Taking into account practical considerations of mission execution (e.g. flight time) the two latter cases OTIME, OPDA are the most optimal. The choice between the two depends on the set priority, for example on other tasks the UAVs are involved in.

B. Massive random experiment

This section describes aggregated results for a large number of experiments. Statistical analysis for 700 randomly generated missions is presented. Missions are generated as follows. A random number of points (between 3 and 20) $\mathcal{P}$ is choosen between $-300m$ and $300m$, the scanning area is taken as the convex hull of $\mathcal{P}$. Up to 3 RMAX and 3 LinkQuad platforms can be selected to be part of the team. The probability to be near the origin of the world, which is assumed to be their home base is set to 50%. RMAX helicopters have a 40% probability to be within $300m$ of the origin and 10% to be within $900m$. LinkQuad platforms have 50% probability to be within $300m$ of the origin. This is to model typical usage of different types of UAV platforms, i.e. smaller LinkQuads are more likely to carry out missions close to their base while RMAX helicopters can also execute long range missions.

platform	area	time	PDA	velocity (m/s)	altitude (m)
RMAX 0	25%	10min	39.1	2.5	35
RMAX 1	25%	22min	39.1	2.5	35
LinkQuad 0	25%	46min	255.6	1.0	17
LinkQuad 1	25%	41min	255.6	1.0	17

(a) Results using equal areas

platform	area	time	PDA	velocity (m/s)	altitude (m)
RMAX 0	22.54%	12min	60.0	1.7	35.0
RMAX 1	22.54%	29min	60.0	1.7	35.0
LinkQuad 0	27.46%	10min	60.0	4.115	17.0
LinkQuad 1	27.46%	11min	60.0	4.115	17.0

(b) Results using the heuristic

platform	area	time	PDA	velocity (m/s)	altitude (m)
RMAX 0	49.77%	21min	60.0	1.939	20.04
RMAX 1	18.88%	21min	60.0	1.939	20.0
LinkQuad 0	15.30%	15min	117.6	2.0	19.83
LinkQuad 1	16.04%	14min	152.6	2.0	5.205

(c) Results of OTIME with $PDA = 60$

platform	area	time	PDA	velocity (m/s)	altitude (m)
RMAX 0	45.21%	33min	88.3	1.316	20.28
RMAX 1	24.16%	33min	88.34	1.316	20.1
LinkQuad 0	15.27%	15min	124.0	1.998	17.08
LinkQuad 1	15.36%	15min	144.6	1.998	8.116

(d) Results of OPDA with a maximum time set to $33min$

TABLE III: Area subdivision, flight time and PDA for the different parameter selection

There are two configuration parameters for the optimisation problem: the number of function evaluations and the initial solution. The optimisation algorithm is initialised with either EQ or HEURI and stopped after 10000 function evaluations to study the evolution of the cost function.

The average time to compute the scan division and generate the trajectory for each UAV is 39ms, using a single thread program implemented in Python running on a 8-core Intel Xeon at 3.70Hz.

The ratio for the cost function is defined as:

$$\rho^i = \frac{f^i_{best} - f^{i_0}_{best}}{f^n_{best} - f^{i_0}_{best}} \tag{5}$$

where f^i_{best} is the best cost function at iteration i, i_0 is the first iteration where a solution that satisfied the constraints is found and n is the last iteration (in this paper $n = 1000$). ρ^i is a normalised representation of the quality of a solution at iteration i.

We define the cost benefit function for OTIME as:

$$f^i_b = f^i_{best} + \mathcal{N}_i * 0.039 \tag{6}$$

where f^i_{best} is the best value for a cost function achieved at time i (which corresponds to the mission time in seconds) and $\mathcal{N}_i$ is the number of function evaluations that were needed to reach iteration i. This function allows to evaluate the benefit of running further iterations compared to the computational cost. The minimum value for this function indicates the optimal number of iterations: for less iterations, it is beneficial to keep running the optimisation to get even better results, for more iterations, the cost of computing a better set of parameters is higher than the reduction in mission time.

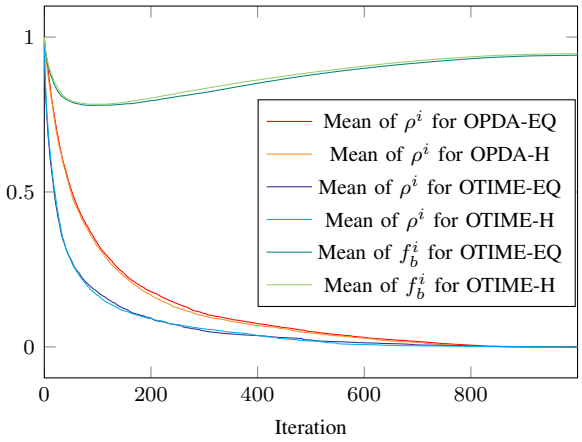


Fig. 8: The evolution of ρ^i and f^i_b for each iteration.

The average evolution of ρ^i for OTIME and OPDA and the average of f^i_b for OTIME over all generated missions is shown in Figure 8. In the worst case for OPDA, it took 437 iterations to get a result that satisfies the constraints and for OTIME up to 693.

Figure 9 shows the gain in mission time for OTIME and in PDA for OPDA compared to using EQ or HEURI. When the gain is negative it is due to EQ and HEURI providing a solution that violates the constraints.

83% of the solutions provided by EQ and 46% by HEURI resulted in violation of the flight time constraint. While for 25% of the missions OTIME and OPDA could not find any solution that respected the constraints. In most cases, it is because the team of UAVs was too small to cover the area.

VII. Discussion

As expected, the selection of the parameters using an optimisation process gives better results than a heuristic approach or an equal partition, as shown in Table III and Figure 9. More importantly the results show the optimisation process is capable of respecting constraints, such as flight time limits.

Results from the cost benefit function presented in Figure 8 show that for OTIME the average optimal number of iterations is between 69 and 126 (which corresponds to 690-1260 function evaluations). There is no good metric to evaluate when a good result has been achieved in case of OPDA. Nevertheless, after 200 iterations the improvement seems small. In the worst case 500 iterations were needed in order to find a solution satisfying the constraints.

While our algorithm is not capable of telling if the problem has a solution that satisfies the constraint, a system would notify the rescuer that the optimisation is taking too long. The penalty function can be used to specify which constraints are causing the most problems, then the rescuer can modify the mission specification to relax some of the constraints.

Figure 8 and Figure 9 show that there is very little difference if the optimisation problems are initialised with EQ

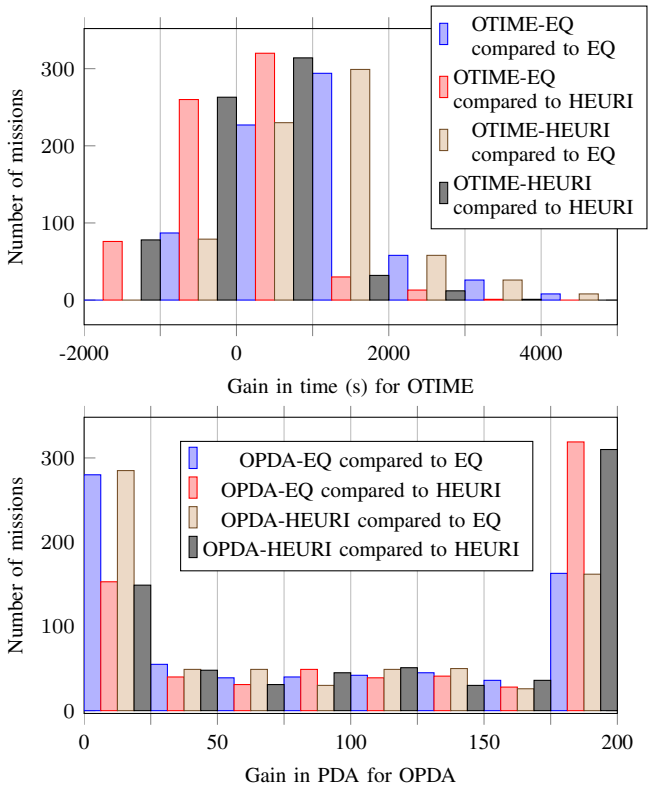


Fig. 9: Gain in time and PDA per problem.

or HEURI. This is due to the CMA-ES algorithm selecting random solutions within the boundary and exploring the space of possible solutions before converging toward the best.

VIII. Conclusion

In this paper we have presented a solution to the partitioning of a scanning area between the members of a team of heterogeneous UAVs. We have shown how to formulate the optimisation problem to suit the need of an operator, whether the mission should to be performed as quickly as possible or resulting in the highest point density within a time limit. The algorithms presented in this paper have been implemented in simulation, integrated in our delegation system and used with real platforms [6].

The main focus of the future work will be put on improving computation time in order to speed up the convergence toward the optimal values. Since the CMA-ES algorithm needs to evaluate the objective function multiple times before it can compute an update to the parameters, parallel programming techniques can be used. In our current approach, a linear bound is set on the parameters, but for non-linear constraints, a penalty function is added to the objective function. This leads to slower convergence, as the optimisation algorithm attempts to use parameters that are outside of the boundary. Variant of the CMA-ES algorithms, such as [4], [1], [12], have been proposed to handle constraints more efficiently, which should improve the convergence properties of the algorithm.

Our current approach does not take into account that scanning areas overlap which could lead to UAVs flying around the same positions at the same time. In the future, the algorithms can be extended to use a path planner and integrate this information in the computation of the total flight time.

References

[1] D. V. Arnold and N. Hansen. A (1+1)-cma-es for constrained optimisation. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 297–304, New York, NY, USA, 2012. ACM.

[2] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello. Mixed-initiative planning and execution for multiple drones in search and rescue missions. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*, 2015.

[3] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 476–481 vol.1, 2000.

[4] R. Chocat, L. Brevault, M. Balesdent, and S. Defoort. Modified covariance matrix adaptation – evolution strategy algorithm for constrained optimization under uncertainty, application to rocket design. *International Journal for Simulation and Multisciplinary Design Optimization*, 6:A1, 2015.

[5] A. Dewan, A. Mahendran, N. Soni, and K. M. Krishna. Heterogeneous ugv-mav exploration using integer programming. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5742–5749, Nov 2013.

[6] P. Doherty, J. Kvarnström, P. Rudol, M. Wzorek, G. Conte, C. Berger, T. Hinzmann, and T. Stastny. A collaborative framework for 3d mapping using unmanned aerial vehicles. In *Principles and Practice of Multi-Agent Systems (PRIMA)*, 2016. To appear.

[7] Nikolaus Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[8] S. Hert and V. Lumelsky. Polygon area decomposition for multiple-robot workspace division. *International Journal of Computational Geometry and Applications*, 8:437–466, 1998.

[9] W.H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 27–32 vol.1, 2001.

[10] A. Kleiner, A. Kolling, M. Lewis, and K. Sycara. Hierarchical visibility for guaranteed search in large-scale outdoor terrain. *Autonomous Agents and Multi-Agent Systems*, 26(1):1–36, 2013.

[11] A. Kolling, A. Kleiner, and P. Rudol. Fast guaranteed search with unmanned aerial vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6013–6018, Nov 2013.

[12] A. Maesani and F. Dario. *Parallel Problem Solving from Nature – PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, chapter Viability Principles for Constrained Optimization Using a (1+1)-CMA-ES, pages 272–281. Springer International Publishing, Cham, 2014.

[13] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, and N. Tomatis. The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–4, Nov 2012.

[14] I. Maza and A. Ollero. *Distributed Autonomous Robotic Systems 6*, chapter Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms, pages 221–230. Springer Japan, Tokyo, 2007.

[15] T. G. McGee and J. K. Hedrick. Guaranteed strategies to search for mobile evaders in the plane. In *2006 American Control Conference*, pages 6 pp.–, June 2006.