An autonomous, multi-agent UAV platform for inspection of civil infrastructure

Michele Bolognini[1], Lorenzo Fagiano[1], Maria Pina Limongelli[2]
[1]Department of Electronics, Information and Bioeng., Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, Italy.
[2]Department of Architecture, Built Environment and Construction Eng., Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, Italy.
email: michele.bolognini@polimi.it

ABSTRACT: UAVs (Unmanned Aerial Vehicles) are nowadays being used more and more in Structural Health Monitoring (SHM). Their versatility, speed, and manoeuvrability make them the ideal means to perform inspections autonomously and remotely, instead of relying on visual inspections carried out by human operators. Since commercial drones have limited flight times, the information collected in this short span must be maximised: to tackle the problem of gathering the maximum amount of data in the shortest possible time, we propose a platform where a central controller coordinates multiple UAVs. We address 1) the problem of generating points of interest, i.e., positions from which a sensor reading must be taken, given a 3D model of the structure, 2) the problem of assigning the points to the drones and finding the optimal traversal order of such points, in order to minimise the total flight time and make the best possible use of each drone's battery capacity. We decouple the two problems by first generating points of interest, starting from the structure's virtual model, and then feeding those points to a central mission planner that employs a linear programming formulation to find near-optimal trajectories for each agent, guaranteeing obstacle avoidance. We also address the issue of robustness of the whole system against the failure of an aircraft. We evaluate our method by applying it to the inspection of a virtual model of an existing building. We find that our approach yields good solutions in a reasonably short time, justifying its use as a robust mission planning algorithm.

KEY WORDS: Autonomous vehicles; Formation control; Coverage problem; Robustness; Swarm robotics; UAVs.

1 INTRODUCTION

Recent technological advancements have made UAVs, commonly referred to as *drones*, a powerful and relatively cheap tool in a wide variety of fields, including agriculture [1], forestry [2], parcel delivery [3] and architecture and civil engineering [4, 5, 6, 7, 8]. Their success is due to their versatility, ease of use and relatively low cost. They can host a broad range of sensors, such as standard cameras, thermal cameras, LiDAR sensors, and infrared devices. These are particularly useful for SHM because they enable essential tasks such as 3D model building [4], surface reconstruction and analysis [5], thermal profiling [6], modal analysis [7] and even some forms of contact inspection [8]. Given these capabilities, drones will likely be adopted soon as essential tools for inspection and monitoring tasks. They can be a reasonable alternative to human inspectors that is cheaper, safer, more time-effective, more repeatable and precise results. The adoption of UAVs as standard inspection tools would also be a significant step towards the automation of the SHM pipeline, allowing frequent inspection measurements to be processed by a computer and integrated into a broader framework, such as a digital twin of the inspected structure.

The push for maximum automation comes with many challenges: depending on the kind of inspection to be carried out, where should the aircraft go? How do we coordinate a fleet of UAVs? How do we minimise the total inspection time? What level of robustness to failure of a drone should be guaranteed to ensure a mission is completed? How do we ensure that no collision occurs between the vehicles and the building in question? We propose a system that addresses these issues by separating the generation of the Points of Interest (POIs) from that of the ideal trajectories. We start from a mesh model of a given building, and we use it to determine the positions of the POIs that are suitable to the task at hand (in this study, the collection of partially overlapping images to create an accurate texture of the building and analyse it in search of defects). Then, we divide the POIs in clusters to ease the computational burden of finding the optimal trajectories that reach every point. Thanks to this sub-division, we generate a simplified graph that abstractly represents the set of POIs, then we use it to solve a Capacitated Vehicle Routing Problem (CVRP [9]) that features constraints designed to add robustness to the solution. The solution of such a problem determines the trajectories for every vehicle. If one of the drones fails during the mission, an updated version of the same problem can be solved to ensure the other ones complete the task.

The novel aspects of this approach are:

1) The coordinated control of a network of independent UAVs.
2) The study of robustness of a solution concerning the failure of one aircraft.
3) The adoption of a model of an existing building for simulations. We consider one of the buildings at the University of California campus, Los Angeles (see Figure 2 and Figure 1).

In Section 2, we discuss the state of the art, in Section 3 we describe the problem in detail, and in Section 4 we present our approach to solving it. Results are presented and commented in Section 5, whereas Section 6 contains our concluding remarks.

2 STATE OF THE ART

The application of drones to the world of SHM and constructions is a recent and growing research field. Literature reviews [10] [11] show that they are often used to extend a human operator's range of action, and in most cases, they are employed to obtain pictures of parts of buildings that would be difficult to reach otherwise, through regular RGB cameras. This is enough to harness the power of computer vision and image analysis techniques that can automatically detect surface defects such as cracks in concrete and road pavement, spalling, exposed rebars, loosened bolts and signs of moisture [12] [13] [14]. Other niches are being explored too, like video-based approaches to dynamical behaviour estimation through Digital Image Correlation (DIC) [15] or cross-correlation [16]. Regardless of the inspection's final objective, most of these approaches rely partially on human intervention. To reach full autonomy, the first problem that must be tackled is automatically generating POIs, *i.e.* points in space from which sensor readings must be taken, starting from a model of the structure. Both the type and the ideal pose of the sensors relative to the building vary significantly depending on the type of survey (building surface inspection, energy performance inspection, mapping, …) [17], but for a given task or set of tasks, it is possible to identify a region of 3D space where POIs should lie. For the problem of coverage of 3D structures, the authors of [18] and [19] developed iterative approaches to maximise uniformity of coverage of the structure and minimise the computational cost to generate a trajectory for the UAV, while also accounting for camera orientation. The resulting series of waypoints is then sent to a single drone to follow. In [20] researchers adopt sub-modular path planning based on a coarse estimation of scene geometry to obtain a single camera trajectory, solving both the viewpoint generation problem and the trajectory generation problem at once while guaranteeing obstacle avoidance and respect of a user-defined time budget. In another case [21], where photos are taken to run a Structure from Motion (SfM) algorithm, points are distributed in space to cause some overlap between consecutive images. Once the field of view of the camera is known, the desired overlap percentage can be chosen, and the POIs can be calculated. We borrow from these approaches by generating our POIs at suitable distances from the mesh model of the building we consider.

Once the POIs are established, it is necessary to determine in what order it is best to visit them, which usually means the fastest. In the case of multiple robots, points must also be distributed among them accordingly. Obstacles must also be taken into account to ensure the trajectories are feasible. This class of problem is typically addressed through graph search methods: many formulations of the Traveling Salesman Problem (TSP) and its variations exist [22], where nodes describe the POIs in a graph and edges between such nodes represent all the possible pair-wise routes and their lengths. A series of nodes represents the solution, suitably characterised through constraints in an optimisation problem depending on the desired properties. The abstract nature of graph-based representation makes it a handy and versatile tool. The approaches that rely on it usually tailor it to suit their specific needs [23], such as addressing communication constraints [24] or planning particular trajectories for fixed-wing drones that avoid certain regions of space [25]. The complexity of the trajectory generation problem increases exponentially with both the number of POIs and the number of vehicles considered, due to the exponential growth of both the search space and the number of Subtour Elimination Constraints (SECs) [22]. Therefore some heuristics must be adopted to solve even medium-sized instances in a reasonable amount of time, and the guarantee of optimality is lost. In some of those cases, theoretical lower bounds on solution quality can be found [26]. Common generalisations of the TSP include its multi-agent version (mTSP) and the Constrained Vehicle Routing Problem (CVRP), where in addition to the features described previously, each vehicle is also assigned a maximum capacity, and each node of the graph represents a *customer* with its capacity demand. In this framework, capacity refers to the quantity of a certain good that the customers require, of which each vehicle can transport a limited amount. The objective here is to ensure that each vehicle has enough capacity to serve all customers along its route. In our formulation, we exploit this concept to ensure that each drone has enough battery life to visit all of its trajectory points, adopting the same mathematical expressions but a different interpretation.

The probability of at least one drone failing increases with the number of deployed UAVs, and it is safe to address this issue at the mission planning level. Discussions of robustness and robust solutions to the CVRP are present in literature, but they focus on different aspects. Robustness is ensured against uncertainty on customer demand [27], on travel time [28] and on service times or other parameters [29], but not with respect to the failure of one of the vehicles. We study the effect of additional constraints specifically designed to take care of this issue and provide a strategy to determine what should happen in case of failure.

We adopt clustering to simplify the overall problem, breaking up the set of all POIs into smaller subsets. Clustering is also a very well-known problem to which countless solutions have been proposed in the literature [30]. Since we aim at clustering together 3D points based on spatial proximity, we adopt *k*-means clustering [31].

3 PROBLEM FORMULATION

We intend to solve a problem determined by a 3D mesh model of the building to inspect, and a set of rules to generate the POIs from such a model. The rules depend on the kind of inspection and the type of sensor employed. Alternatively, one can provide the POIs directly. Also, the number M of UAVs available for the task should be specified. Both M and the number of points n belong to $\mathbb{N}$. The objective is to establish which points should be reached by which drone. In other words, we want to obtain a number of trajectories $t \leq M$, such that each POI belongs to one trajectory only:

$$p_i \in \mathcal{P} \subset \mathbb{R}^3 \qquad i = 1, \dots, n \tag{1}$$

$$\begin{aligned} s_j \subset \mathcal{P} \mid \bigcup_{j=1}^{t} s_j = \mathcal{P} \wedge \qquad & j = 1, \dots, t \\ \wedge\ s_i \cap s_j = \emptyset \qquad & i, j = 1, \dots, t;\ i \neq j, \end{aligned} \tag{2}$$

where p_i are the points of interest, $\mathcal{P}$ is the set of all POIs and s_j is the j-th trajectory. Statement (2) imposes that the set of all

trajectories is a partition of the set of POIs $\mathcal{P}$: each trajectory is a set of points belonging to $\mathcal{P}$, they do not overlap and each point is assigned to a trajectory.

The maximum trajectory length should be minimised so that the task can be carried out in the least amount of time possible. On the other hand, trajectories should also be computed in a reasonable time, meaning a few minutes at most. These two requirements are difficult to achieve (except for trivially small problems) with an approach that seeks the global optimum, because the complexity of the trajectory generation problem increases exponentially with the number of points and vehicles, as mentioned in the previous section. Therefore, to satisfy the second requirement, we shall introduce some heuristics, significantly reducing computational time at the cost of losing the guarantee of global optimality. In addition, the length of each trajectory must be compatible with the drones' maximum flight time, meaning that it should be possible to follow it entirely without stopping. Trajectories must also start and end at a user-defined starting point $p_s \in \mathbb{R}^3$.

Finally, we want to ensure that the solution to the problem is robust with respect to the possibility of failure of one of the drones. This means that during the mission's execution, if one of the aircraft suddenly fails, it must be possible to assign to other UAVs the yet-not-visited POIs assigned to it, and still conclude the mission. In other words, there must always be a feasible solution to a new trajectory generation problem, based on the points that were not visited yet at the moment of the failure and with one less drone.

Figure 2. The building (East side), at the University of California campus, Los Angeles, considered in this study.

4 METHODOLOGY

4.1 The Building Mesh

Assuming we are inspecting the outside surface of a building, we start from a mesh model of its shell. We define a mesh model as a pair of lists: one comprising 3D points belonging to the structure, and another one containing subsets of indices of said points that are coplanar, thus defining a polygon in space. The set of surfaces in the list describes the building's external surface and is expected to be closed, *i.e.* given a point, it is always possible to know if it is inside, outside or on the surface. We shall refer to the two lists as mesh vertices and mesh faces.

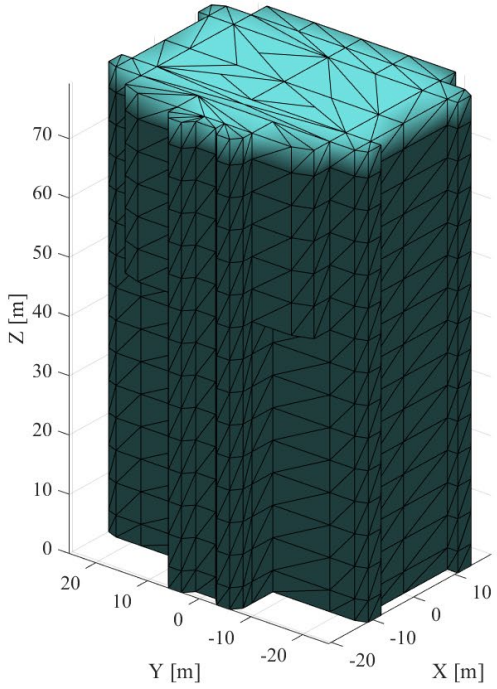

Figure 1. The mesh of the considered building (West side).

The mesh itself can either be exported from the CAD model of a building through suitable software or defined manually, provided its dimension correspond to those of the real object. To streamline calculations and without loss of generality, we use tri-meshes only, *i.e.* meshes whose faces are triangles. Furthermore, for simplicity we ensure that the normal vector to each face points towards the outside of the building. This information is encoded in the order of the subset's points, through the right-hand rule.

The first property of a good mesh is that it correctly represents the building's shape because it will be used for collision checks. To this end, it should also contain the fewest faces possible, to ease the computational cost. In particular, we will perform checks to verify if, given a pair of points in space, the segment that connects them intersects the mesh. If it does, the segment is not a feasible trajectory for a UAV. On the other hand, the mesh components can be exploited in the generation of POIs, as detailed in the next subsection. Therefore, we choose to keep two meshes representing the same building. The first has as few faces as possible and is exploited for collision checking, while the second is obtained from the first by recursively splitting in half every mesh face with a surface area greater than a user-defined threshold, and it is used to automatically generate the POIs.

If the building's CAD model is not available and the inspection goal is its generation, then the mesh of an outer bounding box might be used instead. It is also possible to add further details to the mesh, for example, encoding the building surface material in a face, which might be useful for some inspection tasks.

4.2 Point Generation and Clustering

The generation procedure for the POIs depends on the kind of inspection performed and the mounted sensors' specific properties. We simulated the planning of a photograph collection mission. To this end, the set of all points of interest $\mathcal{P} \subset \mathbb{R}^3$ is obtained by selecting a point for each face of the mesh. Each point is located at a certain distance d from the centre of the mesh face, along its normal. Consequently, all POIs are equally distant from the building's surface, and they are denser where the structure exhibits a more complex shape, conveyed by a higher number of mesh faces. This method also allows us to associate a direction to each point, representing the sensor's desired orientation in that position. The natural choice in our case is to take as orientation the vector opposite to the face normal, to align the camera plane to the building's surface. A check is also performed to ensure that the point is not inside the building mesh.

Since the number of points of interest n can easily be in the order of thousands, the trajectory generation problem cannot be solved over the whole set in a reasonable time. We thus introduce an heuristic, by dividing the set in an arbitrary number of clusters k, based on their spatial proximity. We then solve the problems of optimally traversing a cluster and optimally assigning clusters to drones separately. To perform the clustering step, we apply the k-means algorithm [31]. Such algorithm partitions the set of all points into a pre-defined number k of clusters. It ensures that each point belongs to the cluster with the nearest centroid, thus grouping together close points. This is a natural choice since we aim at minimising the time it takes to traverse each cluster visiting all points, and we can expect that in the global solution the optimal trajectory would be composed of consecutive points that are close to each other. We group the clusters in the set $\mathcal{C}$ and add a special cluster c_0, only comprising the starting point p_s, which will be instrumental to compute drone paths originating from and ending at such a starting location:

$$\begin{aligned} c_i \in \mathcal{C} \qquad & \text{i} = 1, \ldots, k \\ c_0 \triangleq \{p_s\}. & \end{aligned} \tag{3}$$

Finally, for each cluster, a graph is obtained. Each node in the graph represents a point in the cluster. Edges in the graph represent an obstacle-free path between two points. A pair of points is connected if they are sufficiently close, i.e. within a user-defined radius r. This reduces the number of edges in the graph without impacting the trajectory generation problem's solution quality. If an obstacle lies between the POIs, a path connecting them is found on an extended version of the same set of points: to the regular POIs in the cluster, we add a set of intermediate points, this time by calculating the normal to the vertices of the mesh and taking the point at distance d along its direction. The normal of a vertex is the average of all the normal of the faces that include such vertex. The path between the two POIs is the series of obstacle-free paths that connects them passing through intermediate points if such a path exists.

4.3 The Constrained Vehicle Routing Problem

Once the clusters have been determined, we build an undirected graph of clusters $\mathcal{G}$, which we call the *global* graph:

$$\begin{aligned} \mathcal{G} &= (\mathcal{V}, \mathcal{E}, w) \\ |\mathcal{V}| &= |\mathcal{C}| = k+1 = \text{K} \\ \mathcal{E} &\subseteq \mathcal{V} \times \mathcal{V}; \qquad |\mathcal{E}| = \text{E} \\ w &: \mathcal{V} \cup \mathcal{E} \mapsto \mathbb{R}, \end{aligned} \tag{4}$$

where $\mathcal{V}$ is the set of its vertices or nodes, and $\mathcal{E}$ is the set of edges that connect them. One node represents the starting position while the other k nodes represent one cluster each. w is a function that assigns a real value to each node, corresponding to the total length of the trajectory that optimally traverses cluster (recall that in problem (6) each node correspond to a cluster of points of the original path planning problem), and to each edge, in which case it represents the distance between two clusters. These weights are necessary to ensure that the union of all the trajectories of the clusters that will be assigned to a single drone will not be longer than the maximum span that UAV can travel. In order to assign a weight to each edge, we calculate the minimum distance between two clusters:

$$\begin{aligned} w(e_{i,j}) &= minDist(c_i, c_j) \\ e_{i,j} \epsilon \mathcal{E}; \qquad & c_{i,j} \in \mathcal{C}, \end{aligned} \tag{5}$$

where $minDist(c_a, c_b)$ returns the minimum distance between a pair of points such that one point belongs to c_a and the other to c_b. These points will be candidate entry and exit points for the cluster, meaning that if both of the clusters will be assigned to the same UAV, the aircraft will traverse cluster a leaving the exit point last and begin crossing cluster b from the entry point. At this point, though, the order of traversal between clusters has not been established yet, therefore each cluster may contain many candidates. We solve the problem of assigning the clusters to the drones first, then, given the order of the clusters for each vehicle, we optimise its trajectory inside each of the single clusters separately. To solve the first problem, though, we need to assign a weight $w(c_i)$ to each cluster, before knowing exactly how the latter will be crossed. To assign a traversing cost to each cluster we take the length of the shortest Hamiltonian path between a randomly selected pair of candidates as an estimate of the total length of the ideal trajectory. The Hamiltonian path is defined as a path that visits each node of a graph only once. To eventually determine the actual travel distance of each drone, we solve solve the problem again after determining the clusters' order, this time knowing the actual entry and exit points.

We formulate the problem of routing the aircraft through the clusters as a Constrained Vehicle Routing Problem (CVRP). Its objective is to find the optimal set of routes for a fleet of vehicles visit a given set of nodes. We introduce constraints to account for the limited flight time of the drones, expressed as a length (e.g. by assuming an average travel speed). We minimise the objective function

$$\min_x \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{m=1}^{M} l_{i,j} x_{i,j,m} \tag{6}$$

where $x_{i,j,m}$ is a Boolean optimisation variable. Its value is one if the edge connecting nodes *i* and *j* of the graph $\mathcal{G}$ is assigned to vehicle m. The value of $l_{i,j}$ contains both the weight of the edge $e_{i,j}$ and that of cluster *j*:

$$l_{i,j} = w(e_{i,j}) + w(c_j). \tag{7}$$

Note that (6) minimises the sum of trajectory lengths across all UAVs. Another reasonable choice would be to minimise the longest individual trajectory. We explore also this option and compare it with the use of cost function (6) in our simulation results (see Section 5). The following constraints are added:

$$\sum_{i=1}^{K}\sum_{m=1}^{M} x_{i,j,m} = 1 \qquad j = 2,\dots,K \tag{8}$$

$$\sum_{i=1}^{K} x_{i,p,m} = \sum_{j=1}^{K} x_{p,j,m} \qquad \begin{matrix} p = 2,\dots,K; \\ m = 1,\dots,M \end{matrix} \tag{9}$$

$$\sum_{i=1}^{K}\sum_{j=1}^{K} l_{i,j}x_{i,j,m} \le \bar{Q} \qquad m = 1,\dots,M. \tag{10}$$

Eq. (8) ensures that for every cluster except the starting node, only one vehicle enters it, while eq. (9) imposes that if a vehicle enters a node, then the same vehicle must exit. Finally eq. (10) limits the total length of the tour for each vehicle to a maximum pre-defined capacity $\bar{Q}$. At this point one should add Sub-tour Elimination Constraints, in one of the forms that exist in the literature [22]. These are necessary as they exclude from the set of feasible solutions those that contain sub-tours, i.e. loops that do not pass from the starting point. Since this would mean adding at least hundreds of constraints, with the risk of significantly increasing the solution time, we chose a different approach to eliminate subtours from the solution: we solve the problem once as stated and, if at least one subtour is present, solve the problem again adding constraints to remove that specific subtour from the feasible solutions. This approach is applied by identifying the set $N_{sub} \subset \mathcal{V} \setminus \{v_0\}$ of nodes belonging to the subtour (v_0 is the node corresponding to the starting cluster c_0) and imposing that within that subset of nodes at most $|N_{sub}| - 1$ edges are taken:

$$\sum_{i \in N_{sub}}\sum_{j \in N_{sub}} x_{i,j,m} \le |N_{sub}| - 1 \qquad m = 1,\dots,M. \tag{11}$$

The solution of this CVRP essentially assigns clusters to drones and orders them. Once the ordering is known, the entry and exit points to each cluster are also known. Therefore it is possible to find the optimal trajectory by solving once again for the shortest Hamiltonian path, as already anticipated. The final result is a set of M loop trajectories leaving from the starting point p_s and crossing all POIs.

4.4 Robustness

To add robustness to the solution against the failure of a single aircraft, we introduce new constraints. In principle, we want to ensure that, should an UAV fail, the remaining ones have enough combined capacity to cover the trajectory section that the failed drone did not manage to cover. Let us then define the residual capacity of drone m as

$$Q_m^* = \bar{Q} - \sum_{i=1}^{K}\sum_{j=1}^{K} l_{i,j}x_{i,j,m} \tag{12}$$

which represents the capacity left at the end of the planned mission, assuming without loss of generality that the total capacity $\bar{Q}$ is the same for all the aircraft. We shall then impose, for each drone,

$$\sum_{\substack{m=1 \\ m \ne p}}^{M} Q_m^* \ge \sum_{i=1}^{K}\sum_{j=1}^{K} l_{i,j}x_{i,j,p} \qquad p = 1,\dots,M. \tag{13}$$

If the assumption of equal maximum capacity across drones holds, applying the definition of residual capacity (12) yields M identical copies of the same constraint:

$$\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{m=1}^{M} l_{i,j}x_{i,j,m} \le (M-1)\bar{Q} \tag{14}$$

essentially asking that the sum of all residual capacities is at least equal to $\bar{Q}$. If a drone fails and stops following its trajectory the same CVRP can be solved after imposing $M = M - 1$, removing the robustness constraints and updating the graph by eliminating the nodes representing clusters that the failed drone had previously visited entirely. In the worst-case scenario, the UAV fails at the start of the mission, before visiting any cluster, but the new problem still has lower complexity than the original one, as reducing the number of drones also decreases the number of optimisation variables. In order to ensure that the second problem has at least a feasible solution, though, it might be necessary to add more conservative constraints on capacity. This is because the residual capacity is distributed among multiple aircraft, instead of a single one. On the other hand we know that in the worst-case scenario, the UAVs left functioning can at least complete the trajectory assigned to them in the first iteration and then visit the nodes previously assigned to the failed drone. In that case, we must also make sure that they have enough capacity to travel back to the starting point. One way to solve this issue is to take the maximum distance L between the starting point and a cluster, across all clusters, and add that to the minimum residual capacity each drone must have. To this end we replace $\bar{Q}$ with $\overline{Q_L} = \bar{Q} - L$ in the first formulation of the problem. This way we ensure that

$$Q_m^* \ge L \qquad m = 1,\dots,M. \tag{15}$$

and each drone can travel back to the starting point after completing the new mission.

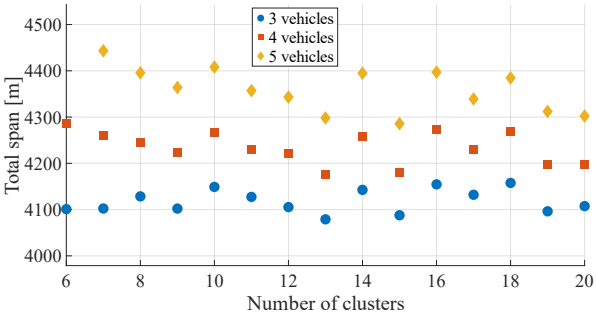

Figure 4. Sum of the lengths of all trajectories as a function of both the number of clusters and the number of drones.

5 RESULTS

All simulations were run in Matlab, installed on Ubuntu 20, on a machine with an i7-9750H 2.6 GHz processor. Hamiltonian pathfinding instances were solved by adding a temporary node to the considered graph with only two connections, the chosen starting and ending point, and solving the Traveling Salesman Problem (TSP) starting from the temporary node through the LKH 3.0 heuristic [32]. CVRPs were solved with IBM CPLEX solver for Matlab.

We considered a building from the University of California Campus, in Los Angeles (see Figure 2), with mesh model shown in Figure 1 and 1279 POIs. First of all, we analysed the behaviour of total length of the trajectories, which is minimised through the cost function, with respect to the number of clusters. Figure 4 shows that the solution's quality is roughly independent of the number of clusters from a certain point on. This means there is no gain and more importantly, no loss in increasing the number of clusters from this standpoint. It also suggests that clustering still leaves globally near-optimal solutions feasible in this framework. Attempting to solve the intra-cluster trajectory planning problem first and the inter-cluster drone assignment second is a faster approach, as it only entails solving the TSP once but solving it without knowing the optimal entry and exit points of each cluster yields worse solutions, which also get worser as the number of clusters increases. On the one hand, keeping the number of clusters K low reduces the complexity of the CVRP, but on the other, it makes the trajectory planning within the clusters more time consuming, as the clusters contain more points. The computational time required to solve the CVRP and the TSP increases exponentially as the number of nodes in their graph grows. Figure 3 illustrates this behaviour. The same can be deduced from Figure 5, since the number of nodes per cluster decreases as the number of cluster increases. Furthermore, a high number of smaller clusters is beneficial for their distribution among drones, because it allows the path planner to better exploit the available capacity of each unit and to obtain a more robust solution.

Solving the problem with an imposed number of drones, by extending constraint (8) to apply to the starting node with the desired number of vehicles, shows that the problem gets more complex with respect to the number of vehicles M, as expected. This is compatible with the observation that the number of optimisation variables is $2EM$. The total length also increases with M, as the path length from the starting point to the first and last clusters visited by each drone is non-negligible.

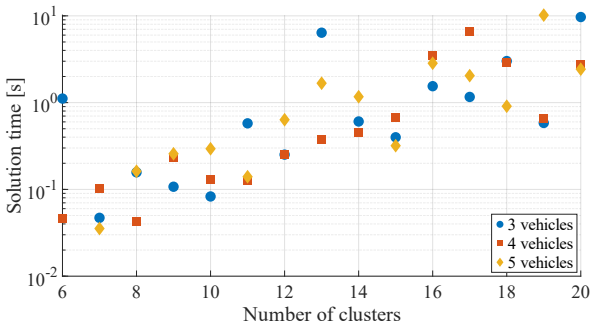

Figure 3. CVRP solution time as a function of both number of clusters and number of drones. The y-scale is logarithmic.

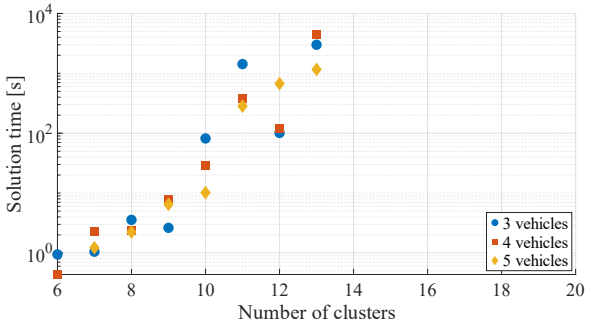

Figure 5. CVRP solution times in min-max formulation. Note how they rise much quicker with respect to the number of clusters. Simulations with high number of clusters were not carried out.

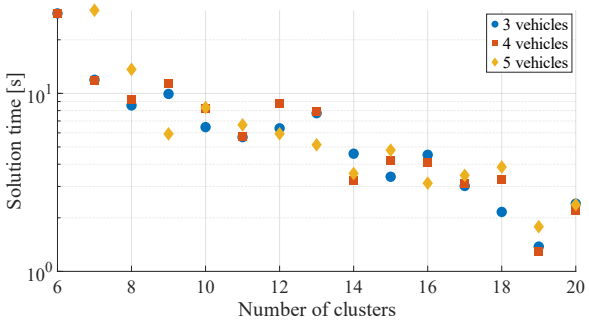

Figure 6. Total of solution time of TSP problems, as a function of both number of clusters and number of drones.

We also attempted minimising a different cost function, i.e. the length of the longest among trajectories:

$$\min_{x} \max_{m} \sum_{i=1}^{K} \sum_{j=1}^{K} l_{i,j} x_{i,j,m} \tag{16}$$

This yields solutions with much more evenly distributed workloads, thus also minimising mission length (see Figure 8). Unfortunately, for min-maxing to work, it is necessary to add a threshold variable in the optimisation, transforming a Binary Integer Linear Problem in a Mixed Integer Linear Problem, which severely impacts the solution (see Figure 5). To decrease

problem complexity, it is also useful to limit the number of arcs in the graph by only connecting clusters whose distance is below a certain threshold or completely separated by obstacles.

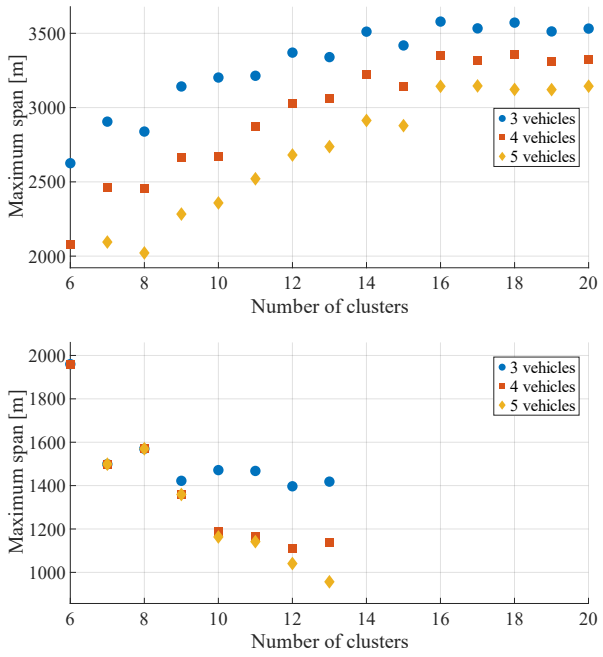

Figure 8. Maximum trajectory length as a function of number of drones and clusters. Objective function (6) (Top) and (16) (Bottom).

6 CONCLUSIONS AND FUTURE WORK

We provide a framework to coordinate multiple aircraft to visit a set of points of interest related to a building, with the goal of carrying out a given data-collection task. The presented approach makes the problem tractable in reasonable time. The cost we pay is losing the possibility of reaching the globally theoretically optimal solution of the problem in the general case, but we deem the quality of the solutions we obtain reasonable. This result is obtained by clustering the points of interest and solving the trajectory generation and point assignment problems separately, making the approach more scalable and reducing the computational time by several orders of magnitude. Both problems are stated as integer optimisation problems. Collision avoidance is ensured through space discretisation and path planning over an undirected graph at a local level. Robustness is addressed through a unique formulation of the CVRP where some constraints are added.

While the solution time in this formulation is not low enough for real-time use with a finite horizon control technique, we deem it reasonable for a dynamic mission planner with occasional on-line replanning. To impact such time, we are now considering heuristic solvers for the CVRP problem, where a feasible solution is built iteratively from a carefully selected starting point to find a sufficiently good solution in a much shorter time. We are also studying smart clustering mechanisms, such as the one proposed in [26], where clusters are identical, and they appear in an ordered and less connected graph structure, thus allowing the solution of the local trajectory generation problem to be found once and applied to all clusters.

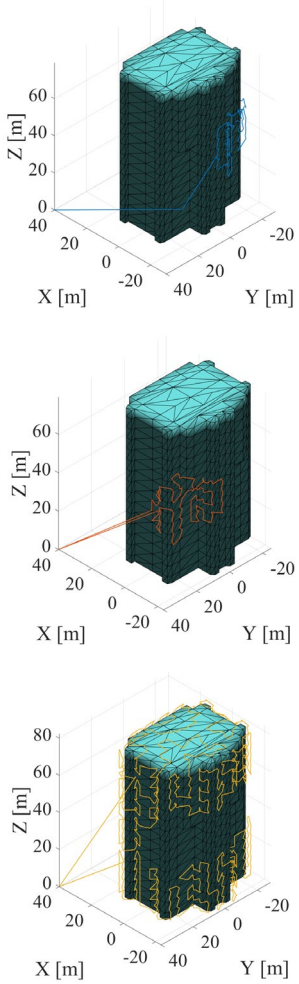

Figure 7. Solution of the problem with 3 drones, 14 clusters.

ACKNOWLEDGMENTS

The authors would like to acknowledge professor F. Amigoni for his insight and fruitful discussions, which contributed to this project's development.

BIBLIOGRAPHY

[1] U. R. Mogili and B. B. Deepak, “Review on Application of Drone Systems in Precision Agriculture,” in *International Conference on Robotics and Smart Manufacturing*, Chennai, 2018.

[2] L. Tang and G. Shao, “Drone remote sensing for forestry research and practices,” *Journal of Forestry Research,* no. 26, pp. 791-797, 2015.

[3] K. Dorling, J. Heinrichs, G. G. Messier and S. Magierowski, “Vehicle Routing Problems for Drone Delivery,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 47, no. 1, pp. 70-85, 2017.

[4] C. Kim, H. Moon and W. Lee, “Data Management Framework of Drone-Based 3D Model Reconstruction of Distaster Site,” in *International Archives of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences, Prague, 2016.

[5] J. Seo, L. Duque and J. Wacker, "Drone-enabled bridge inspection methodology and application," *Automation in Construction,* vol. 94, pp. 112-126, 2018.

[6] M. L. Mauriello and J. E. Froehlich, "Towards automated thermal profiling of buildings at scale using unmanned aerial vehicles and 3D-reconstruction," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 2014.

[7] V. Hoskere, J.-W. Park, H. Yoon and B. F. Spencer, "Vision-Based Modal Survey of Civil Infrastructure Using Unmanned Aerial Vehicles," *Journal of Structural Engineering,* vol. 145, no. 7, p. 04019062, 2019.

[8] B. B. Kocer, "Inspection-while-flying: An autonomous contact-based nondestructive test using UAV-tools," *Automation in Construction,* no. 106, p. 102895, 2019.

[9] P. Toth and D. Vigo, The Vehicle Routing Problem, Philadelphia: Society for Industrial and Applied Mathematics, 2002.

[10] L. Duque, J. Seo and J. Wacker, "Synthesis of Unmanned Aerial Vehicle Applications for Infrastructures," *Journal of Performance of Constructed Facilities,* vol. 32, no. 4, p. 04018046, 2018.

[11] B. F. Spencer, V. Hoskere and Y. Narazaki, "Advances in Computer Vision-Based Civil Infrastructure Inspection and Monitoring," *Engineering,* vol. 5, pp. 199-222, 2019.

[12] V. Hoskere, Y. Narazaki, T. A. Hoang and B. F. Spencer, "Vision-based Structural Inspection using Multiscale Deep Convolutional Neural Networks," in *Huixian International Forum on Earthquake Engineering for Young Researchers*, Urbana-Champaign, 2017.

[13] G. Morgenthal and N. Hallermann, "Quality Assessment of Unmanned Aerial Vehicle (UAV) Based Visual Inspection of Structures," *Advances in Structural Engineering,* vol. 17, no. 3, pp. 289-302, 2014.

[14] Y.-J. Cha and W. Choi, "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types," *Computer-Aided Civil and Infrastructure Engineering,* no. 33, pp. 731-747, 2018.

[15] M. Kalaitzakis, S. R. Kattil, N. Vitzilaios, D. Rizos and M. Sutton, "Dynamic Structural Health Monitoring using a DIC-enabled drone," in *International Conference on Unmanned Aircraft Systems*, Atlanta, 2019.

[16] H. Yoon, V. Hoskere, J.-W. Park and B. F. Spencer, "Cross-Correlation-based structural system identification using unmanned aerial vehicles," *Sensors,* vol. 17, no. 9, 2017.

[17] T. Rakha and A. Gorodetsky, "Review of Unmanned Aerial System (UAS) applications in the build environment: Towards automated building inspection procedures using drones," *Automation in Construction,* vol. 93, pp. 252-264, 2018.

[18] K. Alexis, C. Papachristos, R. Siegwart and A. Tzes, "Uniform Coverage Structural Inspection Path-Planning for Micro Aerial Vehicles," in *IEEE International Symposium on Intelligent Control*, Sydney, 2015.

[19] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics," in *IEEE International Conference on Robotics and Automation*, Seattle, 2015.

[20] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan and N. Joshi, "Submodular trajectory optimization for aerial 3d scanning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[21] S. Daftry, C. Hoppe and H. Bischof, "Building with Drones: Accurate 3D Facade Reconstruction using MAVs," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seattle, 2015.

[22] T. Bektas, "The multiple traveling salesman problem: an overviewof formulations and solution procedures," *Omega,* vol. 34, pp. 209-219, 2006.

[23] B. L. Brumitt and A. Stentz, "GRAMMPS: A Generalized Mission Planner for Multiple Mobile Robots In Unstructured Environments," in *Proceedings ofthe 1998 IEEE International Conference on Robotics & Automation*, Leuven, 1998.

[24] E. I. Grotli and T. A. Johansen, "Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP," *Journal of Intelligent Robot Systems,* vol. 65, pp. 265-282, 2011.

[25] H. Ergezer and K. Leblebicioglu, "3D Path Planning for Multiple UAVs for Maximum Information Collection," *Journal of Intelligent Robot Systems,* vol. 73, pp. 737-762, 2014.

[26] M. Salaris, A. Riva and F. Amigoni, *Multirobot Coverage of Linear Modular Environments,* arXiv preprint arXiv:2001.02906, 2020.

[27] I. Sungur, F. Ordóñez and M. Dessouky, "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty," *IIE Transactions,* vol. 40, no. 5, pp. 509-523, 2008.

[28] C. Lee, K. Lee and S. Park, "Robust vehicle routing problem with deadlines and travel time/demand uncertainty," *Journal of the Operational Research Society,* vol. 63, pp. 1294-1306, 2012.

[29] F. Ordóñez, "Robust Vehicle Routing," *INFORMS TutORials in Operations Research,* pp. 153-178, 2014.

[30] R. Xu and D. C. Wunsch, Clustering, John Wiley & Sons, 2008.

[31] A. Likas, N. Vlassis and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition,* vol. 36, no. 2, pp. 451-461, 2003.

[32] K. Helsgaun, "An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems," 2017.