Control Engineering Practice 75 (2018) 1–16

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

2D visual area coverage and path planning coupled with camera footprints☆

Sina Sharif Mansouri *, Christoforos Kanellakis, George Georgoulas, Dariusz Kominiak, Thomas Gustafsson, George Nikolakopoulos

Robotics Group, Control Engineering Division of the Department of Computer, Electrical and Space Engineering, Luleå University of Technology, Luleå SE-97187, Sweden

ARTICLE INFO

Keywords:
Area coverage
Path planning
Visual inspection
UAVs
Camera footprint

ABSTRACT

Unmanned Aerial Vehicles (UAVs) equipped with visual sensors are widely used in area coverage missions. Guaranteeing full coverage coupled with camera footprint is one of the most challenging tasks, thus, in the presented novel approach a coverage path planner for the inspection of 2D areas is established, a 3 Degree of Freedom (DoF) camera movement is considered and the shortest path from the taking off to the landing station is generated, while covering the target area. The proposed scheme requires a priori information about the boundaries of the target area and generates the paths in an offline process. The efficacy and the overall performance of the proposed method has been experimentally evaluated in multiple indoor inspection experiments with convex and non convex areas. Furthermore, the image streams collected during the coverage tasks were post-processed using image stitching for obtaining a single overview of the covered scene.

1. Introduction

Unmanned Aerial Vehicles (UAVs) equipped with remote sensing instrumentation have been emerging in the last years due to their mechanical simplicity, agility, stability and outstanding autonomy in performing complex maneuvers (Kanellakis & Nikolakopoulos, 2017; Kendoul, 2012). Furthermore, UAVs have the ability to offer numerous opportunities in a variety of applications, such as mapping (Zongjian, 2008), landslides (Niethammer, Rothmund, James, Travelletti, & Joswig, 2010), search and rescue missions (Doherty & Rudol, 2007), forest fire inspection (Alexis, Nikolakopoulos, Tzes, & Dritsas, 2009) and aerial manipulation (Kanellakis, Terreran, Kominiak, & Nikolakopoulos, 2017). One of the most common remote sensors is the visual sensor, either monocular or stereo, while the acquired set of images from the UAV's mission can be analyzed and used to produce sparse or dense surface models, hazard maps, investigate access issues, and other area characteristics (Valente et al., 2011). However, the main problem in these approaches is to guarantee the full coverage of the area, a fundamental problem that is directly related to the autonomous path planning of the aerial vehicles. In order to guarantee the full coverage, the problem of the coverage path planning should be mathematically formulated to be coupled with the camera frustum, while maximizing the area coverage, in relation to the camera movement and the corresponding orientation. This problem is well-known in the literature to be NP-hard and thus there is a need of a numerical solution to provide a close to optimal solution. Moreover, in all the coverage path planning methods, there are constraints on the length of the path, as it is desired to follow the shortest one and this can directly affect the overall mission, mainly due to the UAV's limited flight time (Mansouri, Karvelis, Georgoulas, & Nikolakopoulos, 2017). Finally, the coverage path planning approach should be evaluated in real-life scenarios, which would add an important overall technological contribution of the established approach. In the presented approach, it is assumed, without loss of generality, that during the operation, the UAV has the ability to retain a closed loop fixed altitude, while having a downward-looking camera. In this specific case, the camera frustum can be modeled by fixed size rectangles, of size $w_i \times h_i$, $(w_i, h_i) \in \mathbb{R}^2$ as it is indicated in Fig. 1.

In the related literature, the problem of covering the target area with fixed size rectangles has been addressed multiple times, while this typical problem is known to be NP-complete (Daniels & Inkulu, 2001; Heinrich-Litan & Lübbecke, 2005) and one of the several computationally difficult decomposition problems (Franzblau & Kleitman, 1984). Additionally, apart from the coverage approaches for visual inspection, it has several important practical applications, such as in VLSI layout design, pattern recognition, computer graphics, databases, image processing, etc. Thus, inspired by this vision, the main objective

☆ This work has received funding from the European Union Horizon 2020 Research and Innovation Programme under the Grant Agreement No. 644128, AEROWORKS.

* Corresponding author.
E-mail address: sinsha@ltu.se (S.S. Mansouri).

https://doi.org/10.1016/j.conengprac.2018.03.011
Received 20 September 2017; Received in revised form 22 January 2018; Accepted 13 March 2018
Available online 21 March 2018
0967-0661/

Fig. 1. Schematic of the field of view in the case of UAV.

of this article is to establish an algorithm to segment the target area in relation to the camera's position and orientation (x, y, ψ) in an offline approach; in the sequel the UAV will be able to plan its shortest possible path through all segments with a fixed take-off/landing area.

On the specific topic of covering polygons with rectangles, most of the works consider a varying size of the rectangles' area (Stoyan, Romanova, Scheithauer, & Krivulya, 2011) or the target area is considered to be convex, rectilinear or a union of convex polygons (Stoyan et al., 2011). Furthermore, most of the previous contributions formulate the problem mathematically, without presenting the numerical solution to the problem, while it should be highlighted that the problem is proven to be NP-hard for the case of the fixed size rectangles (Culberson & Reckhow, 1988).

For the Coverage Path Planning (CPP) problem, there have been many works that address 2D spaces and fewer approaches that address coverage of 3D spaces (Mansouri, Kanellakis, Fresk, Kominiak, & Nikolakopoulos, 2018). A survey on CPP methods in 2D can be found in Galceran and Carreras (2013). In most of the CPP methods for covering the target area, the underlying algorithms decompose the area in sub-regions. Thus, coverage algorithms can be categorized by the type of decomposition into: (a) classical exact cellular decomposition (Lingelbach, 2004), where the decomposition of the area breaks down to simple non-overlapping regions and a robot sweeps these regions, (b) Morse-based cellular decomposition (Acar, Choset, Rizzi, Atkar, & Hull, 2002) where the area is decomposed based on critical points of Morse functions (Milnor, 2016) and a motion planner algorithm guarantees to encounter all the critical points in the target area, (c) landmark-based topological coverage (Wong, 2006), with the area decomposed based on natural landmarks, (d) contact sensor-based coverage of rectilinear environments (Butler, Rizzi, & Hollis, 1999), where the robot follows a cyclic path while building up a cellular decomposition of the area, (e) grid-based methods (Shivashankar, Jain, Kuter, & Nau, 2011) where the target area is decomposed into a collection of uniform grid cells, and (f) graph-based methods (Xu, 2011) for environments that can be presented as a graph, where the graph can be updated based on robot sensors, while performing coverage. The aforementioned methods decompose the area without consideration of the mobile agent's sensor, which can affect the coverage quality. Towards aerial coverage and visual inspection, it is assumed that a top level procedure handles the area segmentation (Avellar, Pereira, Pimenta, & Iscold, 2015; Valente et al., 2011), while there has been no related work to consider decomposition of the target area with relation to the camera footprint. Thus, this decoupling of coverage task and segmentation of the area can reduce the generality of these approaches. Moreover, in case of grid-based methods or lawn-mower problems, the target area is decomposed into a collection of uniform grid cells. As a result, most grid-based methods completeness depends on the resolution of the grid map. Although it is easy to create a grid map and grid-based representations are the most widely used for coverage algorithms, grid maps suffer from exponential growth of memory usage, while the resolution does not depend on the complexity of the area. Moreover this type of algorithms does not consider rotation of the robot, and may yield into suboptimal paths. Furthermore, in the case of the art gallery problem, the problem is to determine the minimum number of guards for observing the whole gallery. This results to static coverage problems, which are about finding a good placement of the sensors, and is categorized under surveillance problems. However, in this article, coverage path planning coupled with area decomposition is studied, which differs from the previous approaches and solutions. In the presented novel approach, the camera footprint is coupled with the UAV position and yaw angle and the area is decomposed while maximizing the covered area. This approach is directly inspired by real life applications of UAVs and the mathematical formulation of the overall problem consists of a novel consideration of the coverage problem.

Preliminary and limited results from the proposed framework have been presented in Mansouri, Georgoulas, Gustafsson, and Nikolakopoulos (2017), while this work has been extended in this article with the following fundamental additional contributions: (1) 3 Degrees of Freedom (DoF) (x, y, ψ) instead of 2 DoF (x, y) camera motion as a result of the natural movement of the UAV, (2) integration of a path planner to provide the shortest path among the centers of the identified rectangles, and (3) multiple experimental verifications of the proposed methods with a comparison and analysis of the results. Based on the aforementioned state of the art, the main contribution of this article is three-fold. Firstly, the problem of covering the polygonal target area, while considering the camera's footprint with 3 DoF is mathematically formulated and solved approximately by three different well-known metaheuristic techniques: the Pattern Search (PS), the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO). Secondly, this article addresses the problem of the segmentation of a polygonal region with fixed size rectangles for the coverage and inspection tasks for the first time, to the best of the author's knowledge. Thirdly, it should be noted that the coverage path planner is evaluated through multiple experimental trials, where the overview of the covered area is visually represented using an image stitching technique. Finally, it should be highlighted that the main novelty of the proposed scheme stems from the establishment of an overall framework for the path planning for convex and non-convex 2D areas. Thus, a novel mathematical framework for solving the coverage problem by segmentation of the target area and calculating the shortest path will be established. As a fundamental difference, in the proposed approach the cities that the TSP should visit are calculated in order to maximize the covered area. The established theoretical framework has the novelty of providing a path for maximizing the coverage of the area, while considering for the first time, to the authors best knowledge, the camera footprint position and orientation, in contrast to many existing approaches that simplify the effect of the camera footprint in the target area segmentation. Moreover, this article provides an near optimal solution for the well-known NP-hard problem of covering polygons with rectangles. Finally, one of the fundamental technological contributions of this article is the fact that the proposed framework has demonstrated the direct real life applicability and feasibility of coverage in an indoor experiment. The established coverage framework is able to integrate and adapt fundamental principles from the areas of control, image processing, and computer science, in a fully functional and efficient approach that enables the penetration of aerial robotics in real life applications and more specifically in the field of aerial inspection. It should be highlighted that in this approach the camera movement and yaw orientation is coupled with the UAV, while in cameras with gimbals, the camera remains horizontal, regardless of the motion around them and there is no closed loop controller between gimbals and the UAV orientations, a fact that limits the DoFs of the camera motion in relation to the UAV. Additionally, the usage of gimbal is limited, as lightweight UAVs have strong payload constraints.

The rest of the article is structured as follows. The mathematical establishment of the proposed problem is presented in Section 2, followed by the presentation of the selected algorithms for solving the problem in Section 3. Then the path generation is explained in 4, which is followed by the image stitching technique in Section 5. In Section 6 multiple simulation and experimentation results are presented with a corresponding comparison and discussion. Finally the article concludes in Section 7.

2. Problem statement

The UAVs should be able to understand the area to be inspected, ensure complete coverage and an accurate reconstruction of the inspected area. Relying on accurate state estimation, as well as on dense reconstruction capabilities, algorithms for the autonomous inspection planning should be designed to ensure full coverage. In the following, the problem is formulated for the coverage of 2D target areas with a camera that has 3 DoF in x, y and ψ and the camera frustum is modeled by fixed size rectangles, of a $w_i \times h_i \in \mathbb{R}^{2+}$.

The given region can be presented by $\Omega \subset \mathbb{R}^2$. Additionally the finite set of rectangles is denoted as:

$$\boldsymbol{\Lambda} = \{R_i : i \in I_n = \{1, 2, \dots, n\}\} \tag{1}$$

where rectangles can be defined by:

$$R_i = \left\{ (x_i, y_i) \in \mathbb{R}^2 : \frac{-w_i}{2} \le x_i \le \frac{w_i}{2}, \frac{-h_i}{2} \le y_i \le \frac{h_i}{2} \right\}. \tag{2}$$

Each rectangle can be identified by its center position (x_i, y_i), orientation of the center ψ_i, the width w_i and the length h_i. More specifically the position of the camera is at the center of the rectangle. The width and the length of the rectangle can be obtained through (3) from the altitude, and the Horizontal and Vertical Fields of View (HFOV and FOV) of the camera as depicted in Fig. 2.

$$\begin{aligned} w_i &= 2H_i \tan^{-1} \alpha, \\ h_i &= 2H_i \tan^{-1} \beta \end{aligned} \tag{3}$$

where H_i, α and β are the height, horizontal and vertical FOVs of the camera. The installed camera on the UAV has three dimensions of orientation, however, without loss of generality, in the case of a downward looking camera, only one orientation ψ around the z-axis is considered. Additionally, the positions of the vertices of the ith rectangle, as depicted in Fig. 2, can be obtained from the center (x_i, y_i) and orientation (ψ_i) of the camera by (4) as it follows:

$$\begin{aligned} \begin{bmatrix} x_i^1 \\ y_i^1 \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{1}{2} T(\psi_i) \begin{bmatrix} w_i \\ h_i \end{bmatrix} \\ \begin{bmatrix} x_i^2 \\ y_i^2 \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{1}{2} T(\psi_i) \begin{bmatrix} -w_i \\ h_i \end{bmatrix} \\ \begin{bmatrix} x_i^3 \\ y_i^3 \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{1}{2} T(\psi_i) \begin{bmatrix} -w_i \\ -h_i \end{bmatrix} \\ \begin{bmatrix} x_i^4 \\ y_i^4 \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{1}{2} T(\psi_i) \begin{bmatrix} w_i \\ -h_i \end{bmatrix} \end{aligned} \tag{4}$$

where $[\cdot]^1$, $[\cdot]^2$, $[\cdot]^3$ and $[\cdot]^4$ are the positions of the four vertices of the rectangles and $T(\psi_i)$ is the rotation matrix with respect to the orientation of the camera:

$$T(\psi_i) = \begin{bmatrix} \cos\psi_i & -\sin\psi_i \\ \sin\psi_i & \cos\psi_i \end{bmatrix} \tag{5}$$

The placement of the rectangles R_i in a constant altitude is defined by the translation vector $u_i = (x_i, y_i)$ and the orientation vector ψ_i, $i \in I_n$, while the set of the translated and orientated rectangles $R_i(u_i, \psi_i)$ is expressed by $\Lambda(u, \psi)$, where $u = \{u_1, u_2, \dots, u_n\} \in \mathbb{R}^{2n}$ and $o = \{\psi_1, \psi_2, \dots, \psi_n\} \in [0, 2\pi]^n$.

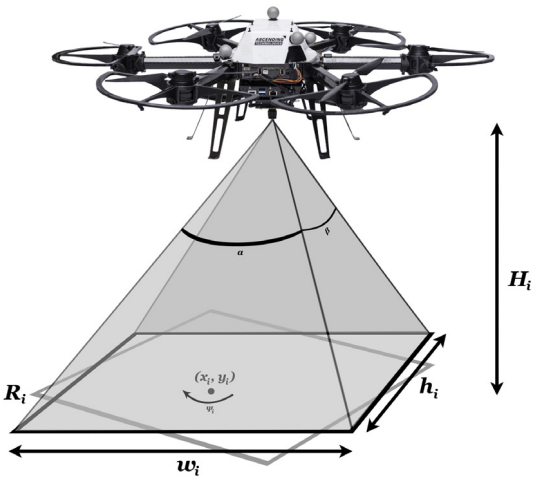

Fig. 2. Schematic of the relation between the rectangle and altitude, horizontal and vertical Field of View (FOV) of the camera.

The polygonal

$$P(u_i, \psi_i, n) = \bigcup_{i \in I_n} R_i(u_i, \psi_i) \tag{6}$$

represents the region covered by the union of the rectangles R_i, while Λ^* is a cover of Ω if there exist vectors such that:

$$\Omega \subset P(u_i, \psi_i, n) = \bigcup_{i \in I_n} R_i(u_i, \psi_i) \tag{7}$$

Therefore, the search procedure tries to find the position and orientation of each rectangle through the translation, the orientation vectors u_i, ψ_i and the total number of rectangles n. Thus, the goal is to define whether there exist vectors $u \in \mathbb{R}^{2n}$ and $o \in [0, 2\pi]^n$, such that $\Lambda(u, o)$ is a cover of Ω. The general schematic concept of the problem is presented in Fig. 3.

In order to mathematically formulate the described problem, the following optimization problem is defined:

$$\min_{u,\psi,n} \underbrace{\left| A\left(P(u_i, \psi_i, n) \cap \Omega\right) - A(\Omega) \right|}_{\text{maximizing coverage}} + \underbrace{\gamma n}_{\text{minimizing \# rectangles}}$$

s.t.

$$\begin{aligned} & u_l \le u \le u_u \\ & 0 \le \psi \le 2\pi \\ & P(u_i, \psi_i, n) = \bigcup_{i=1}^{n} R_i(u_i, \psi_i) \\ & n \in \mathbb{N} \end{aligned} \tag{8}$$

where A is the function that calculates the area of polygons, u_u and u_l are the upper and lower bounds of the translation vector that is defined by the boundaries of the target area Ω, γ is a scalar weight to penalize the number of rectangles in the objective function and n is the number of rectangles. Therefore the optimization is a mixed integer programming problem. However, the number of rectangles n can be provided by a rough estimation, e.g. by calculating a relation between the rectangle's area and the target area ($\lceil A(\Omega)/A(R_i) \rceil$). In this way, the first part of optimization will be solved with an initial guess of the number of rectangles. In the next step the number of rectangles will be updated based on the remaining area until the "optimal" number in relation to the covered area is obtained. Hence in this article the second term of optimization is omitted to reduce complexity. Moreover, several cases arise in the interaction between two rectangles $R_i(u_i, \psi_i)$ and $R_j(u_j, \psi_j)$ with $i \neq j$, $u_i = (x_i, y_i)$ and $u_j = (x_j, y_j)$, that are determined by:

$$\begin{aligned} & R_i(u_i, \psi_i) \cap R_j(u_j, \psi_j) = \emptyset \\ & R_i(u_i, \psi_i) \cap R_j(u_j, \psi_j) \neq \emptyset \end{aligned} \tag{9}$$

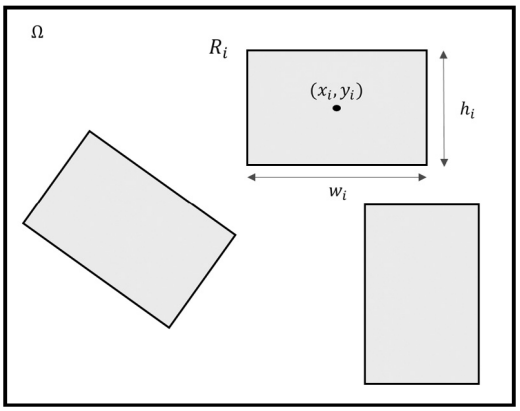

Fig. 3. Schematic of the problem statement.

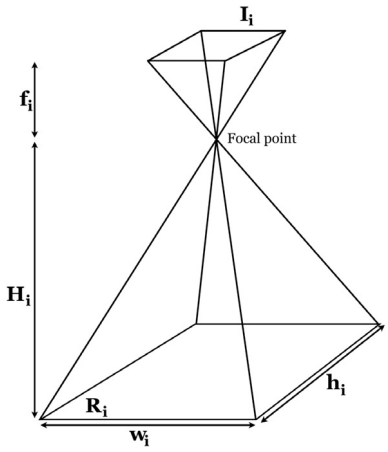

Fig. 5. Schematic relation between image frame I_i and camera footprint R_i.

While solving the problem, based on the shape of the target area Ω, different overlapping scenarios occur that can be any of the two cases in (9). However, it should be highlighted that the cases of $R_i(u_i, \psi_i) \subset R_j(u_j, \psi_j)$ and $R_j(u_j, \psi_j) \subset R_i(u_i, \psi_i)$ are not considered when dealing with the coverage problem and should be omitted. In general, the proposed problem statement is considered to be an NP-hard problem thus in the sequel metaheuristic optimization algorithms will be utilized for deriving a solution. At this point, it should be highlighted that the metaheuristics do not guarantee to locate the optimal solution and thus it is not suggested to use metaheuristics for solving problems where efficient exact algorithms are available. However, for NP-hard problems as the once considered in this article, metaheuristics (or heuristics) are the only viable option for finding a "good enough" (or near optimal) solution. Thus in this paper, the notation "optimal" is used to highlight that solution is not always the actual (unknown) optimal solution.

In Section 3, three different numerical methods are presented for solving (8). Additionally, it should be highlighted that the optimization problem of the area segmentation does not consider overlapping between rectangles, as finding the optimal ratio of the overlapping area further increases the computational complexity. Nevertheless, omitting the overlapping factor from the optimization step, does not restrict the performance of the stitching algorithm, which requires substantial overlapping, since without a loss of generality, the aerial agent sequentially records the video streaming from the visual sensor, while following the generated coverage path. In this manner, the post-processing of the data will provide the necessary overlapping (according to the application requirements) between the rectangles for the coverage problem, as it is presented in Fig. 4. To be more specific, after the position (x_i, y_i) and orientation ψ_i of each rectangle is obtained, the"shortest" path is generated which passes through the center of each rectangle as described in Section 4. The generated path provides overlapping between rectangles as it is shown in Fig. 4, which depicts an intermediate acquired image with a red rectangle. Furthermore, Fig. 5 provides the schematic relation between image frame I_i and camera footprint R_i, which is described in Section 5. This figure depicts that the distance between footprint R_i and frame I_i includes two components, the camera focal length f, where f is the distance from the image plane to the camera lens, and H_i, where H_i is the distance from the camera to the ground plane.

3. Numerical methods

In engineering practice many optimization problems arise, which are proven to require a non-realistic amount of time to be solved. For these cases, engineers do not have the luxury to search for the overall optimal solution and they are willing to trade optimality for "good enough" solutions within a realistic amount of time and computational recourses spent. To this end, a number of "global" metaheuristic optimization algorithms have been proposed (Talbi, 2009). These algorithms either rely on engineering intuition or are inspired by optimization mechanisms encountered in nature. One of the first such algorithms is the GA optimization procedure (Holland, 1975), and since its introduction in the 70's, a plethora of other algorithms, either completely novel or variations of existing ones, have been proposed (Yang, 2010).

Among the different algorithms, this article investigates the use of three very popular and successful paradigms for tackling the hard

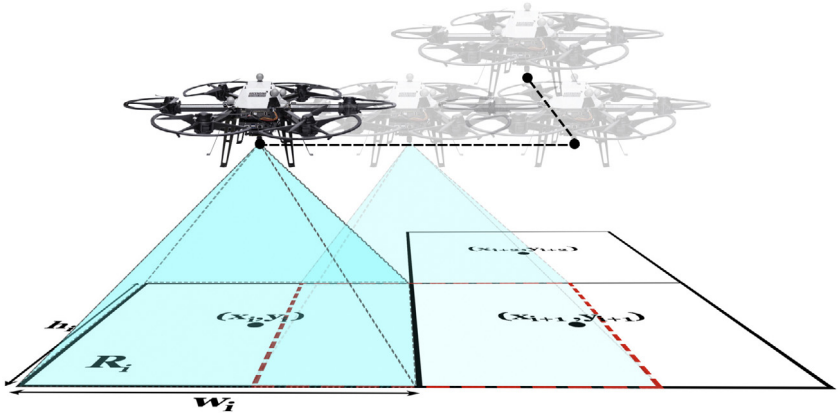

Fig. 4. Schematic of the overall proposed method.

optimization problem described in the previous section. All of them can deal with non-differentiable functions (derivative free) and have mechanisms to escape local minima (maxima). The first investigated algorithm is the Pattern Search (PS), which is the oldest among the three methods (Hooke & Jeeves, 1961). Due to its simplicity, this algorithm is used as a benchmark for comparison of the other two more recent and more advanced methods. It belongs to the family of single-solution based algorithms, which means that at each iteration a single solution is retained and used to seed the search for the next iteration. The other two algorithms are the GA, and the PSO, which rely on the evolution of a set of solutions, thus belonging to the family of population-based methods. Retaining a pool of candidate solutions can help these algorithms search larger portions of the search space and also escape local minima, utilizing an exploration–exploitation mechanism. GA performs the evolution of solutions using a set of operators (selection, mutation and combination) directly inspired by the operations that are taking place in the cells of living organisms. GA is probably the most used method from the family of evolutionary methods and has found numerous applications in engineering optimization problems. Therefore, GA is usually "the method to beat", for every new proposed algorithm. PSO, the third of the tested algorithms, is the newest among the three, and many variants have been proposed since its initial introduction (Parsopoulos, 2010). It has proven to be both efficient and easy to implement. Although initially devised to simulate the flying of birds, it was soon turned into a method for efficiently searching over complex function landscapes. Unlike GA, in PSO solutions tend to interact with each other communicating information about the quality of search areas, guiding the search of the whole "swarm" of solutions. Each of the algorithms is briefly presented in the following subsections.

3.1. Pattern search

PS (Hooke & Jeeves, 1961) belongs to a family of numerical optimization methods called direct search methods (Kolda, Lewis, & Torczon, 2003) (e.g. the Nelder Mead simplex method also belongs to this family (Nelder & Mead, 1965)) that do not require the gradient of the problem. These methods, although they are over half a century old, are still popular within the engineering community for one specific reason, they work well in practice (Lewis, Torczon, & Trosset, 2000). For the case of PS, which is based on exploratory moves around a point, the building block is the pattern that is a predefined set of vectors, which controls the search directions of the algorithm.

In PS each iteration is characterized by an optional global *search step* and *poll step*. In the *search step*, the objective function f is evaluated at a finite number of points, lying on the current mesh M_k, in an attempt to try to find a new point with a better function value than the incumbent. In the *poll step*, a positive spanning set $D_k \subset D$ is chosen in order to construct the poll set. The poll set $P_k = \{u_k + \Delta_k d \in U : d \in D_k\}$ is constructed, as the neighboring mesh points in each of the directions surrounding the current solution u_k, with $\Delta_k > 0$ depending on the fitness of the mesh. Then the function f is evaluated at the points in P_k, until all the points have been evaluated, or until one with a lower objective function value is found.

In case that the *search* or the *poll* step is successful, then the Δ_{k+1} is updated as:

$$\Delta_{k+1} = \tau^{m_k^+} \Delta_k \tag{10}$$

where $\tau > 1$ and m_k^+ is the fixed integer, while if the *search* and *poll* both fail to find an improved mesh point then the following rule applies:

$$\Delta_{k+1} = \tau^{m_k^-} \Delta_k \tag{11}$$

where m_k^- satisfies $m_{min} \leq m_k^- \leq -1$. The Algorithm for PS is presented in Algorithm 1. Moreover a full reference on pattern search mathematics and algorithms can be found in Abramson (2002).

Algorithm 1 Pattern Search

Require u_0

1. Calculate $M_0 \subset U$ by $\Delta_0 > 0$ and u_0
2. Repeat for $k = 0, 1, 2, \ldots k_{max}$
   (a) **Mesh Step** create a mesh of points around the current point
   (b) **Poll Step** poll the mesh points looking for an improved location
   (c) **Update** if a better position is found (successful polling), move to that position and expand, otherwise (unsuccessful polling) contract the mesh centered at the current location

3.2. Genetic algorithm

The GA (Davis, 1991) is an optimization and search technique based on the principles of genetics and natural selection and belongs to the broader family of evolutionary algorithms. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (Haupt & Haupt, 2004). First, the set of candidate solutions/chromosomes (population) is evaluated against a given fitness function. Based on their fitness function, solutions are selected for mating, i.e. for the creation of new candidate solutions through a process that is usually called crossover. The produced solutions are further altered through the mutation process. Finally part of the generated offsprings are used to update the genetic pool of the GA. This procedure is repeated for quite many iterations (generations in the GA literature) till either a maximum number of iterations is reached or an individual that meets a predefined criterion emerges. Overall, the algorithm is presented in Algorithm 2.

Algorithm 2 Genetic Algorithm

1. Randomly initialize the population with u candidate solutions.
2. Repeat until best individual is good enough or maximum number of generations has been reached
   (a) **Population evaluation** determine the fitness of each member of the population $f(u)$
   (b) **Selection** select parents from population stochastically according to their fitness for recombination
   (c) **Mating** perform mating using the crossover operator on parents creating new offsprings/candidate solutions
   (d) **Mutation** with a mutation probability mutate the new offspring
   (e) **Population updating** update the population using the newly created solutions and the old ones, using a predefined selection mechanism

3.3. Particle swarm optimization

PSO belongs to the swarm intelligence family (Engelbrecht, 2006) and is a derivative-free optimization method (Eberhart & Kennedy, 1995). PSO is also a population based method, which uses a set (swarm) of particles (potential solutions) that search for the "optimal" location of the complex function landscape. The movement of the particles is influenced by the particle's own memories as well as the memories of its peers. Each time the particle is attracted by its best position so far (exploitation) and by the best position so far (exploration) found by the particles in its neighbor. Depending on the formation of the neighborhood, different variants arise. If the neighbor consists of the whole population, then the PSO variant is known as the *gbest* (global

best) PSO. The basic *gbest* PSO algorithm is provided by the following equations:

$$\begin{aligned} v_i(n+1) &= \zeta(n)v_i(n) + \eta_1 c_1[p_i - v_i(n)] + \eta_2 c_2[p_{g(n)} - v_i(n)] \\ u_i(n+1) &= u(n) + v_i(n+1) \end{aligned} \tag{12}$$

where i is the particle index, $v_i(n)$ is the current velocity of the ith particle, $\zeta(n)$ is an inertia function (usually a linearly decreasing one), $v_i(t)$ is the current position of the ith particle, p_i is the position with the best fitness value, found by the ith particle so far, $g(n)$ is the particle with the best fitness among all particles (best position found so far — global version of the PSO (Kennedy, Kennedy, Eberhart, & Shi, 2001)), c_1, c_2 are positive constants that control the cognitive and social behavior of the particle and η_1, η_2 are random numbers uniformly distributed in $[0, 1]$, which introduce stochasticity in the algorithm. The algorithm is presented in Algorithm 3.

Algorithm 3 Particle Swarm Optimization

1. Initialize the swarm (usually randomly but uniformly across the search space) positions and velocities u
2. Repeat until best individual is good enough or maximum number of iterations has been reached
   (a) **Velocity** update the velocities of all particles
   (b) **Position** update the positions of all particles
   (c) **Best** update the personal best positions and the global best position of the swarm

4. Path planner

In the specific application of aerial vehicles, the vehicle should have a mission trajectory with a fixed initial and final position and pass through the center (x_i, y_i) of all obtained rectangles. This problem can be treated as a Traveling Salesman Problem (TSP) (Dorigo & Gambardella, 1997), where the goal is to find the shortest path to visit all rectangles once and return to the origin position.

There are many formulations of TSP available in the literature and the recent survey (Orman & Williams, 2006) presents a detailed analysis. In this article the mathematical formulation for symmetric TSP of Dantzig et al. (Matai, Singh, & Mittal, 2010) is used. Let the center of rectangles $V = \{(x_i, y_i) \in \mathbb{R}^2 : i \in I_n\}$ from $\Lambda = \{R_i : i \in I_n = \{1, 2, \dots, n\}\}$ be a set of areas that should be visited, $E = \{\big((x_i, y_i), (x_j, y_j)\big) : (x_i, y_i), (x_j, y_j) \in V, i \neq j, \{i, j\} \in I_n\}$ be the edge set, and $c_{i,j} = c_{j,i}$ be a cost measure associated with edge $(x_i, y_i), (x_j, y_j) \in E$. Thus the TSP can be defined on a complete undirected graph $G = (V, E)$ and the cost $c_{i,j}$ is the Euclidean distance between edges, defined as:

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{13}$$

The formulation of TSP is presented in (14).

$$\min \sum_{i \neq j} c_{i,j} x_{i,j}$$

subject to:

$$\begin{aligned} &\sum_{j=1, i \neq j}^{n} x_{i,j} = 1 \quad (i \in V) \\ &\sum_{i=1, j \neq i}^{n} x_{i,j} = 1 \quad (j \in V) \\ &\sum_{i,j \in S, i \neq j} x_{i,j} \leq |S| - 1 \quad (S \subset V, 2 \leq |S|) \end{aligned} \tag{14}$$

where $x_{i,j}$ is a binary variable for each edge (i, j), which is equal to 1 if and only if the edge appears in the optimal tour. Additionally, the first set of equalities requires ($\sum_{i=1, i \neq j}^{n} x_{i,j} = 1$) that guarantees that only one path leads to each area and the second set of equalities ($\sum_{j=1, j \neq i} x_{i,j} = 1$) requires that only one path departures from each area or:

$$x_{i,j} = \begin{cases} 1 & \text{if path goes from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

It should be noted that the generated path provides overlapping while moving from the center of one rectangle to the next one as it is shown in Fig. 4. The resulting waypoints are then converted into position–velocity–yaw trajectories, which can be directly provided to the utilized linear model predictive controller cascaded (Alexis, Nikolakopoulos, & Tzes, 2012) over an attitude-thrust controller.

5. Image stitching

The achieved coverage of the proposed algorithm is visually assessed through the stitching (Brown & Lowe, 2007) of acquired images, creating a global image of the covered workspace. All resources described in the previous Sections lead to a trajectory that the aerial platform is following above the area of interest. During the flight, the aerial platform visits specific waypoints where image frames, with substantial overlapping, are captured from the downward-looking camera. Processing the obtained frames in a joint manner, a single overview of the covered area is provided and can be further used for inspection purposes. Overall, the mission planner computes the coverage path for a target area and the image stitching component processes the visual information obtained along the designed path to generate the overview of the area in a single image.

In a nutshell, the stitching algorithm provides the global image using feature-based image registration technique, including (a) feature extraction and matching between image pairs, (b) projection transform estimation and (c) image warp and blending to map and overlay images in a single representation. The implemented algorithm adopts a keyframe-based approach, that processes specific frames along the path, assuming the camera footprint (Section 2) as depicted in Fig. 4. Assuming simple pinhole camera model for each frame I_i holds that $I_i = g(R_i)$, where g is the projection from footprint to camera frame.

Moreover, the generated trajectory is designed in a way to keep the height of the platform constant during the operation, to preserve the overlap ratios of the acquired frames, assuming planar scene. Considering the aforementioned assumptions, a pair of overlapping frames I_1, I_2 are connected through a linear projective transformation, expressed with matrix H known as homography. This matrix transfers two corresponding points b_1 and b_2 from one frame to the other respectively, as shown in the following equation.

$$b_1 = {}^1H_2 \, b_2 \tag{16}$$

where $b_i = [s_1, s_2, 1]^T$ are homogeneous coordinates in the image frame. The homography is used to align consecutive frames into a single image. Initially, distinct features are extracted and matched across the image pairs. The estimation of the homography H is performed using the corresponding points identified in the pair with M-estimator Sample Consensus (MSAC) (Raguram, Frahm, & Pollefeys, 2008) algorithm, which is robust to outliers. The process selects a root frame for the stitching outcome and sequentially processes the remaining frames in pairs. Once all images are registered by concatenating the relative homographies (17), the final stitched frame is obtained.

$${}^1H_j = {}^1H_2 \, {}^2H_3 \, \cdots \, {}^{j-1}H_j \tag{17}$$

Finally, the resulting stitched image is processed with a multi-band blending algorithm (Burt & Adelson, 1983) to remove visual artifacts induced from the combined images. The following algorithm provides an overview of the implemented image stitching method.

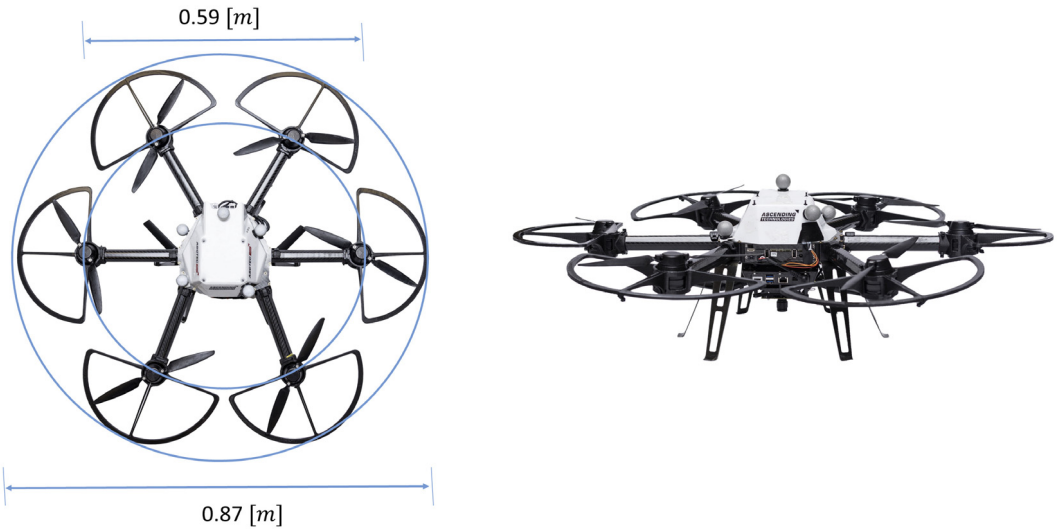

Fig. 6. AscTec NEO platform with the downside monocular camera attached.

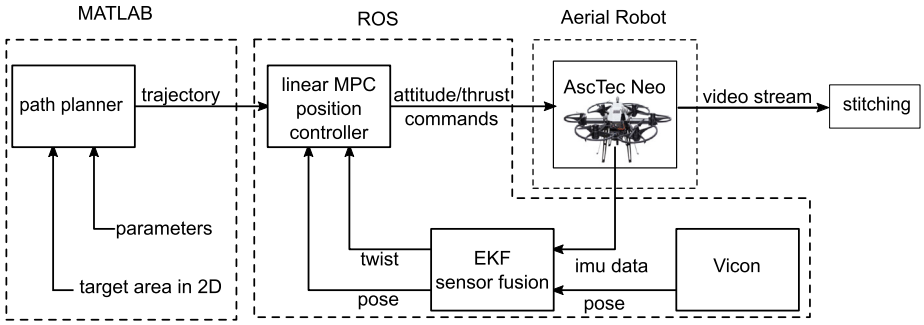

Fig. 7. Software and hardware components used for conducting inspections.

Algorithm 4 Stitching workflow

1. Acquire Image frames
2. Image Description (Feature extraction)
3. Root frame selection
4. Calculation ${}^{i}H_{j}$ for image pair (I_i, I_j)
5. Relative homographies concatenation
6. Image blending to process the stitching artifacts

6. Results

This section presents the experimental verification of the proposed scheme, followed by the necessary discussion, analysis and comparisons that prove the efficiency of the proposed framework.

6.1. Experimental setup

The proposed method has been evaluated with the utilization of the Ascending Technologies NEO hexacopter, depicted in Fig. 6. The platform has a diameter of 0.59 m and height of 0.24 m. The length of each propeller is 0.28 m as depicted in Fig. 6. This platform is capable of providing a flight time of 26 min, which can reach a maximum airspeed of 15 m/s and a maximum climb rate of 8 m/s, with maximum payload capacity up to 2 kg. It has an onboard Intel NUC computer with a Core i7-5557U and 8 GB of RAM. The NUC runs Ubuntu Server 14.04 with Robotic Operating System (ROS) installed. ROS is a collection of software libraries and tools used for developing robotic applications (Quigley, Morgan et al., 2009). Additionally, multiple external sensory systems (e.g. cameras, laser scanners, etc.) can be operated in this setup.

Regarding the onboard sensory system, the monocular camera (weight of 0.05 kg.) (Fig. 6) is attached below the hexacopter with a 90° tilt from the horizontal plane. The monocular camera provides footprint of $.5 \times .6\,\mathrm{m}^2$ in the height of 1 m, while it has been operated in 20 fps with a resolution of 640×480 pixels.

The proposed generated path, established in Sections 2 and 4 ((8) and (14)), has been entirely implemented in MATLAB. The inputs for the method are the target area Ω, the number of rectangles n and the footprint size (w_i, h_i). The generated paths are sent to the NEO platforms through the utilization of the ROS framework. The stitching process has been implemented in OpenCV library with input the captured frames from the UAV. All the simulations have been performed in MATLAB on a computer with an Intel Core i7-6600U CPU, 2.6 GHz and 8 GB RAM.

The platform contains three main components to provide autonomous flight, which are a Vicon motion capture system, a Multi-Sensor-Fusion Extended Kalman Filter (MSF-EKF) (Lynen, Achtelik, Weiss, Chli, & Siegwart, 2013) and a linear Model Predictive Control (MPC) position controller (Alexis, Nikolakopoulos, & Tzes, 2011;

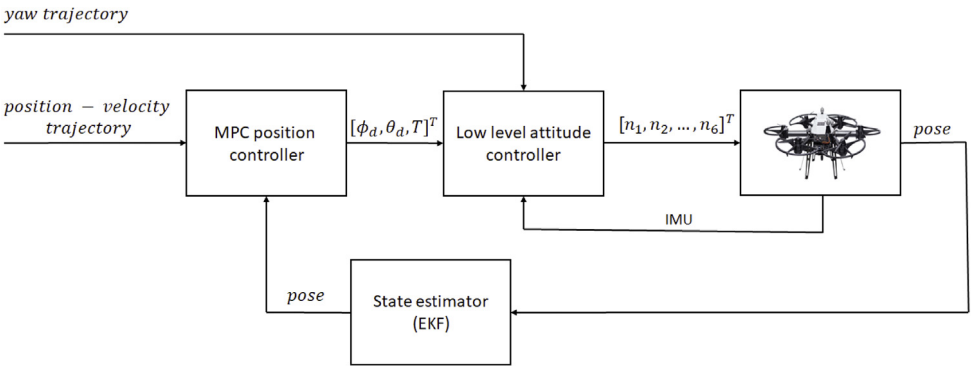

Fig. 8. Controller scheme for NEO.
Source: Regenerated from Kamel et al. (2018).

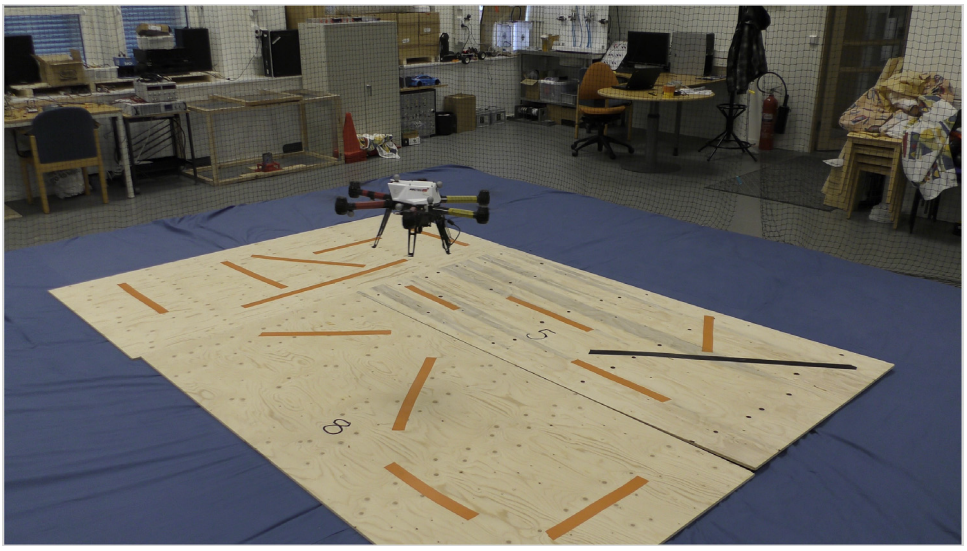

Fig. 9. Experimental target area.

Alexis et al., 2012). Vicon provides pose information and the MSF-EKF component fuses the obtained pose information and the NEO IMU (Fresk, Mansouri, Kanellakis, & Nikolakopoulos, 2017) measurements. This consists of an error state Kalman filter, based on inertial odometry, performing sensor fusion as a generic software package; it has the unique feature of being able to handle delayed and multi-rate measurements, while staying within computational bounds. The linear MPC position controller (Kamel, Stastny, Alexis, & Siegwart, 2017) generates attitude and thrust references for the NEO predefined low level attitude controller. In Fig. 8 the components of the controller are shown. Positions, velocity, and trajectory are sent to a Linear MPC position controller, which provides roll ϕ_d, pitch θ_d and mass normalized thrust T commands for the inner loop. From the initial control tuning experiments, it was proven that the low level attitude controller is able to track the desired roll, pitch and yaw trajectory and to calculate the corresponding $n_1, n_2, \ldots, n_6$ rotor speeds for the vehicle. More details about the controller scheme and parameters are provided in Kamel et al. (2017).

The image stream from the overall experiment is processed using the discussed method in Section 5, while the overall schematic of the experimental setup is presented in Fig. 7.

6.2. Experimental results

In this Section different scenarios are presented in order to evaluate the performance of the proposed method and for comparison of the three different metaheuristic algorithms adopted. The initial positions of the rectangles are generated randomly, while in order to reduce the complexity of the problem, the integer part of the objective function is omitted and the number of rectangles is provided to the algorithm and updated based on the covered area. Furthermore, in all scenarios the landing and taking off position are fixed and the TSP is solved in order to pass from all the centers of rectangles in an optimized manner. After the path is obtained, the waypoints are converted to trajectory and sent back to the NEO. In the presented experimental results, the desired trajectory has been indicated with blue color. The agent takes a video stream during the mission; in this case without loss of generality

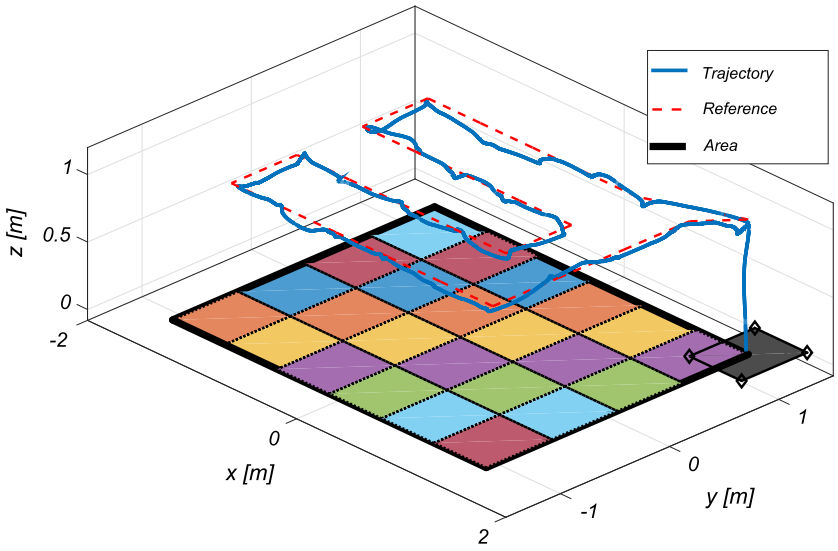

Fig. 10. Covering and generated path for a convex polygons with 24 rectangles.

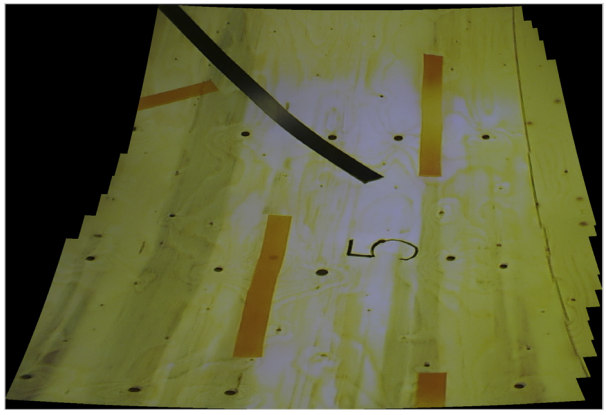

Fig. 11. Visual overview of the covered scene with 24 rectangles.

the overlapping between camera footprints is guaranteed. In all the examined cases the algorithm stops if the average relative change of the best solution is less than a threshold and the landing spot is shown by black rectangle. Due to the stochastic nature of the algorithms, each experiment was repeated ten times. However in the following figures only one execution of the experiment will be presented without loss of generality.

For demonstrating the applicability of the method, an indoor artificial 2D target area was assembled as depicted in Fig. 9 that has been visualized through the resulting stitched image. The flying arena covers a volume of $5\times5\times3$ [m^3] and the initial and final position of the UAV is $[1.5, 1.2, 0]$. In all the experimental trials the position tracking error for the UAV is below 0.1 m, 0.08 m. 0.15 m in x, y, z axises respectively. The stitching results cannot be merged to the followed path as there is no unit in the presented pictures. It should be highlighted that sequential frames captured during flight are used to provide stitching, however, more investigation should be done to improve the final stitching result and reduce the tracking error.

In the first scenario, the target area is considered to be a rectangle with a size of 3×2.4 m that can be filled by 24 rectangles with a fixed size

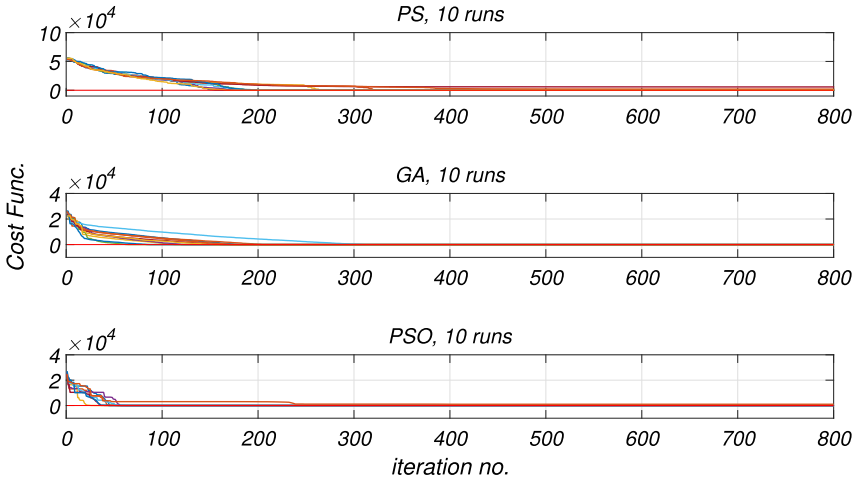

Fig. 12. Objective function value for each method and for the 10 runs in the case 1.

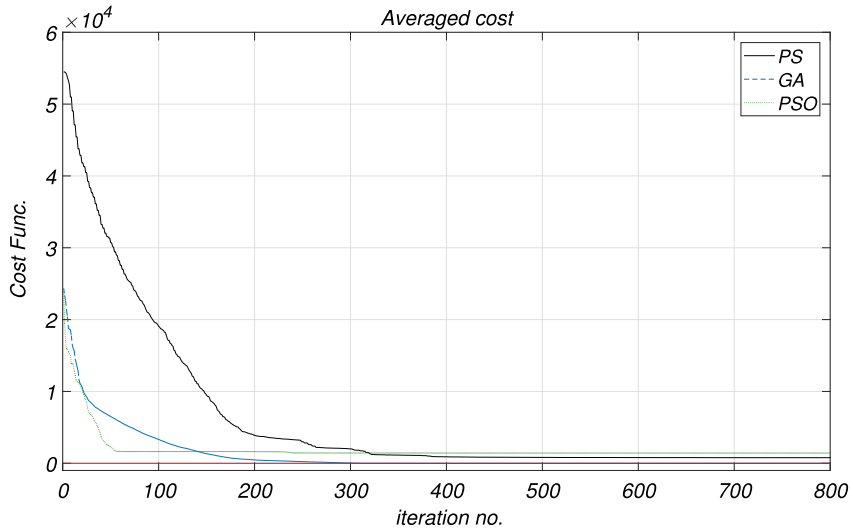

Fig. 13. Average objective function for each metaheuristic method in first scenario.

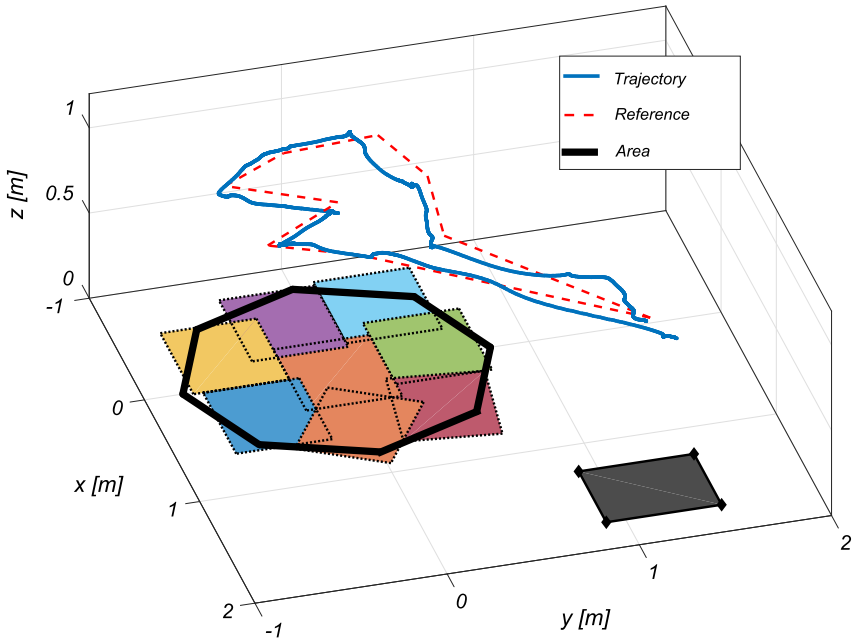

Fig. 14. Covering an octagonal polygons with 9 rectangles.

of .5 × .6 m. Fig. 10 presents the obtained results and the corresponding path is depicted for the inspection of the area, while Fig. 11 depicts the resulting image after stitching together the corresponding captured frames. Additionally, Figs. 12 and 13 present the values of the ten runs and the average of the first term of the objective function respectively.

In the second scenario, an octagonal polygon with an area of 1.8 m^2 is covered by 8 rectangles. The result of the coverage and the individual values and the average of the objective function are depicted in Figs. 14, 16 and 17 respectively. Likewise, a generated path through the center of the rectangles is obtained. The visual overview of the covered scene is demonstrated in Fig. 15.

Furthermore, Figs. 18, 20, 21 and 19 present the final placement and the individual values and average of the objective function and overall stitched image respectively. In this case a complex non-convex polygon is considered and it is filled by 14 rectangles. Additionally, the *H* and *C* shape environments are studied and they are filled by 22 and 14 rectangles respectively. The Figs. 22 and 23 present the final placement of the camera footprints, reference path and trajectory which is followed by the UAV correspondingly.

Table 1 summarizes the mean percentage of coverage of the target area. As it can be seen the coverage is higher than 96% for all five cases and all three metaheuristic methods.

Additionally, the Root-Mean-Square Error (RMSE) between the reference and the trajectory which is followed by the UAV, is provided in Table 2.

The proposed method provides an offline path planner in the case of a downward looking camera for a single agent, where Fig. 24 displays a comparison for each method's average computational time

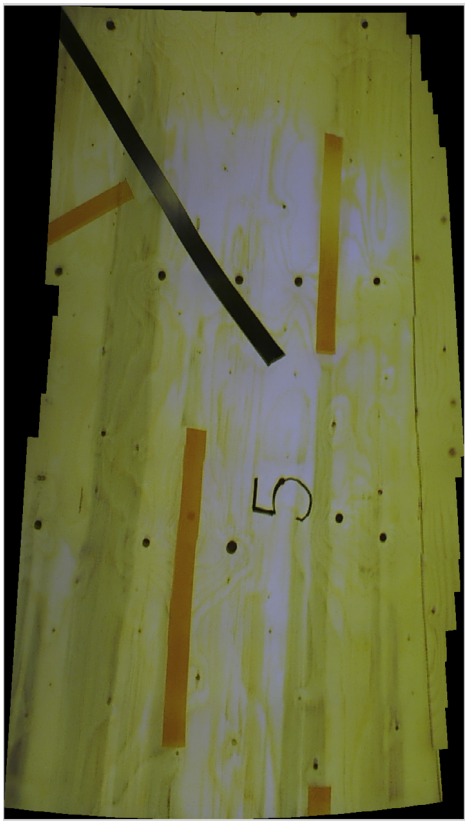

Fig. 15. Visual overview of the covered scene with 9 rectangles.

Table 1
Coverage percentage for the different scenarios and methods.

Methods	Cases				
	1	2	3	4	5
Pattern Search	99.9%	96.3%	98.6%	99.9%	99.5%
Genetic Algorithm	99.9%	97.5%	97.7%	99.8%	99.1%
Particle Swarm Optimization	99.9%	97.0%	98.0%	99.7%	99.2%

Table 2
RMSE of the trajectory followed by UAV in each case.

RMSE	Case 1	Case 2	Case 3	Case 4	Case 5
x m	0.04	0.03	0.05	0.08	0.10
y m	0.03	0.04	0.06	0.05	0.08
z m	0.15	0.060	0.8	0.11	0.12

and for each scenario. From the obtained results, it is observed that the computational time of the GA is significantly higher than the PSO and PS methods in all the examined cases, due to the more involved mechanism applied for generating a new set of solutions.

For comparison purposes, Table 3 summarizes the produced path length between the proposed path planner, the path generated by the grid based method and the method in Mansouri et al. (2017). In the cases of the grid-based method and the method in Mansouri et al. (2017), the UAV maneuvers with a fixed altitude of 1 m were considered, while the UAV sweeps the grids with a minimum number of turns, to cover the target area. As it has been presented, the proposed path planner reduces the length of the path in all these cases, except for the case 1, which is a rectangle polygon and the result is the same in all the compared methods.

7. Discussion and conclusions

This article presented a holistic approach to the problems of Area Coverage and Path Planning with UAVs for Polygon Areas. The area coverage of polygons is (approximately) solved using metaheuristics and the path planning problem is treated as a TSP. TSP algorithm optimizes the path through the target area. All five investigated methods can lead to satisfactory coverages of different polygon shape areas (both convex and non convex) and with an achieved coverage level above 96%. It should be noted that the total coverage could be achieved using more conservative approaches (using more rectangles). On average, the PS method seems to slightly outperform the other two as PSO and GA exploration mechanism can sometimes lead them outside of the boundaries, However all of them provide quite similar results with small variations because the objective function has many (good) local minima corresponding to quite different positioning. Among the three, the GA

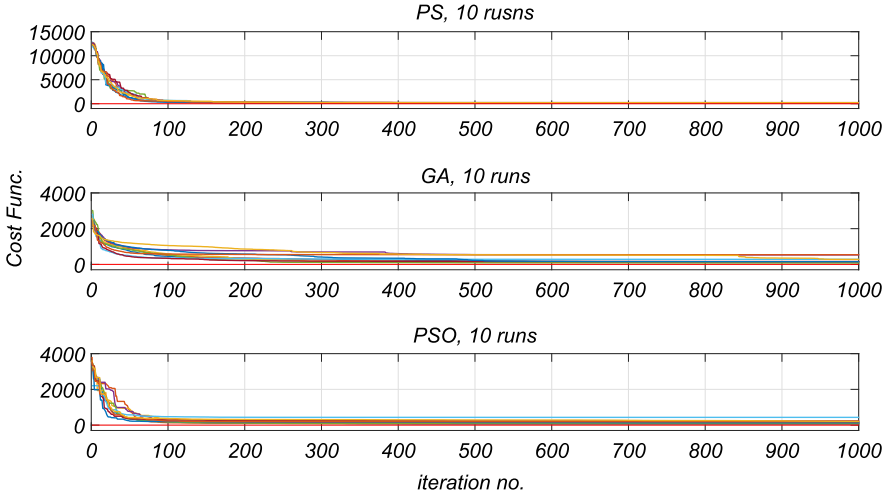

Fig. 16. Objective function value for each method and for the 10 runs in the case 2.

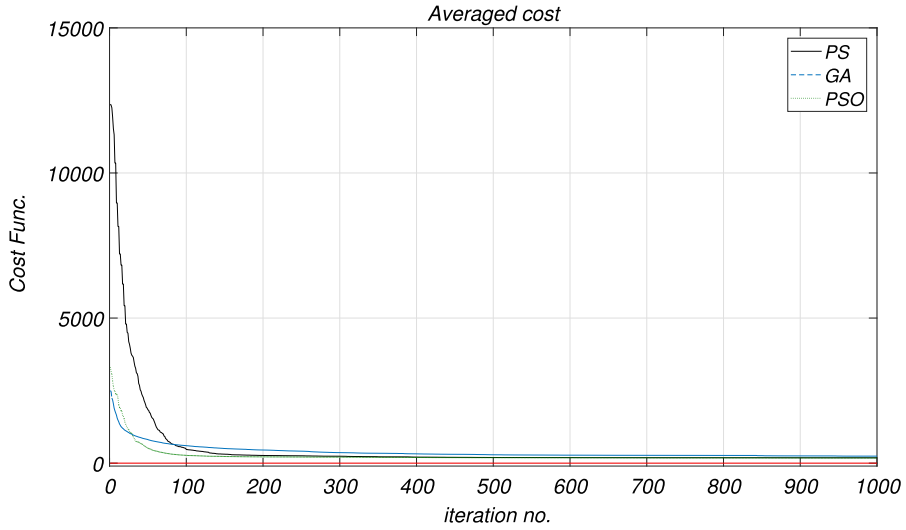

Fig. 17. Average of Objective function value for each metaheuristic in the second case.

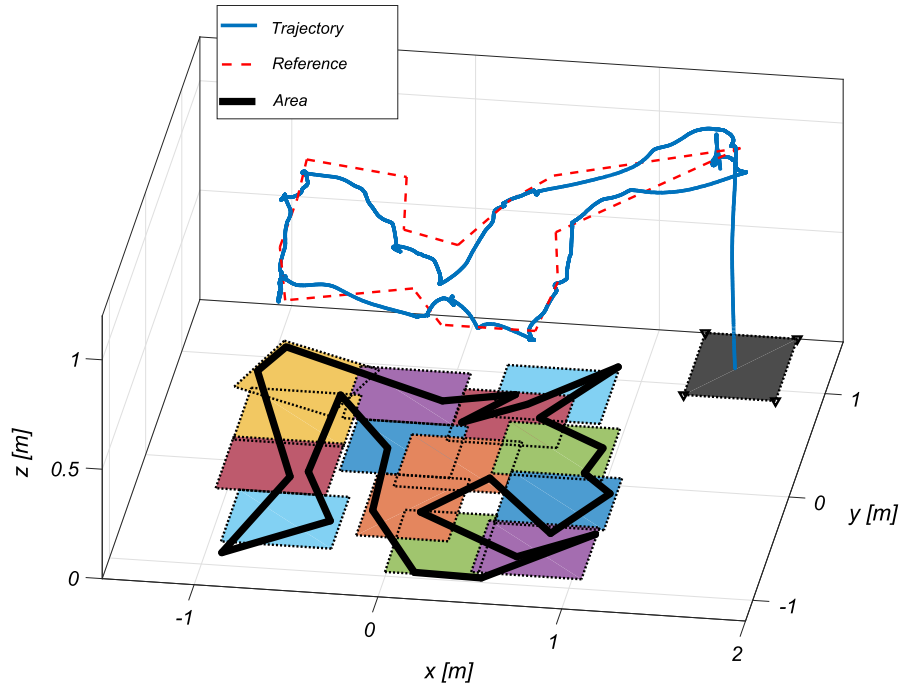

Fig. 18. Covering a complex polygon with 14 rectangles.

has the slowest convergence, which however is not such a crucial factor at this stage, since the optimization is performed offline.

Additionally, although experiments involving parallel computations were not involved, two out of three of the proposed algorithms (PSO and GA as well as any population based metaheuristics) can be implemented in parallel. On the other hand, this article considers the coverage problem without the overlapping necessity, as the problem's complexity would lead to an exponential growth of the search space.

Furthermore, in the current approach the path planner operates offline, which means that the parameter variations, such as wind or changes in the target area bounds, do not affect the planned path, however, these parameters have a significant effect on the quality of the coverage. In the case of external disturbances, such as wind gusts or significant parameter variation, these alterations have a direct affect on the followed path during the online execution of the path following and thus the better the control scheme is, the better the path following will be. However, in the case of varying target area bounds, the optimization approach should solve the problem online or closer to real time to update the path, while it should be linked with other online identification and estimation schemes for capturing the time varying nature of the parameters, a feature that currently is considered as future work.

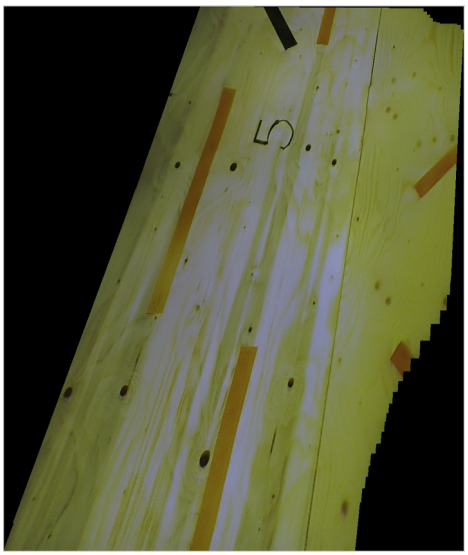

Fig. 19. Visual overview of the covered scene with 14 rectangles.

Finally, from the obtained results, the shortest path is generated for the aerial vehicles for remote visual sensing. The performed experimental trials demonstrate the covered scene using the image stitching method. For this method the captured frames from the onboard sensory system of the UAV during the coverage task were used. The provided path for the UAV led to frames with substantial overlapping, which is necessary for the algorithm to provide the visual overview of the scene. However, this specific path generation problem needs more investigation since there are multiple choices for the resulting path, based on the considered dynamics of the UAV and the corresponding constraints, e.g. minimum energy consumption, minimum time coverage, etc. Moreover, the stitching result can be improved using camera with higher resolution, alternative target areas, enhanced image registration and reducing tracking error during navigation. Future work will be focused towards the investigation of the effect of changing the UAV's altitude, which can cause time varying areas of camera frustum in an on-line approach and also investigate the possibility of online execution of the presented algorithms.

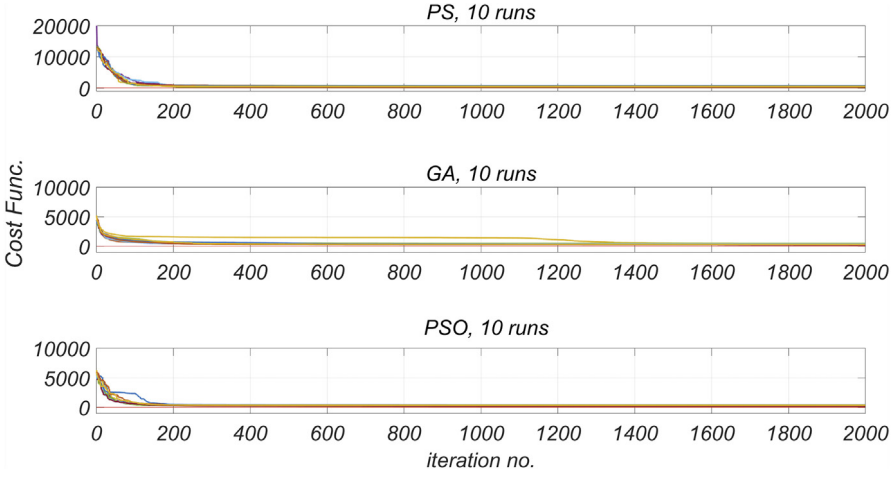

Fig. 20. Objective function value for each method and for the 10 runs in the case 3.

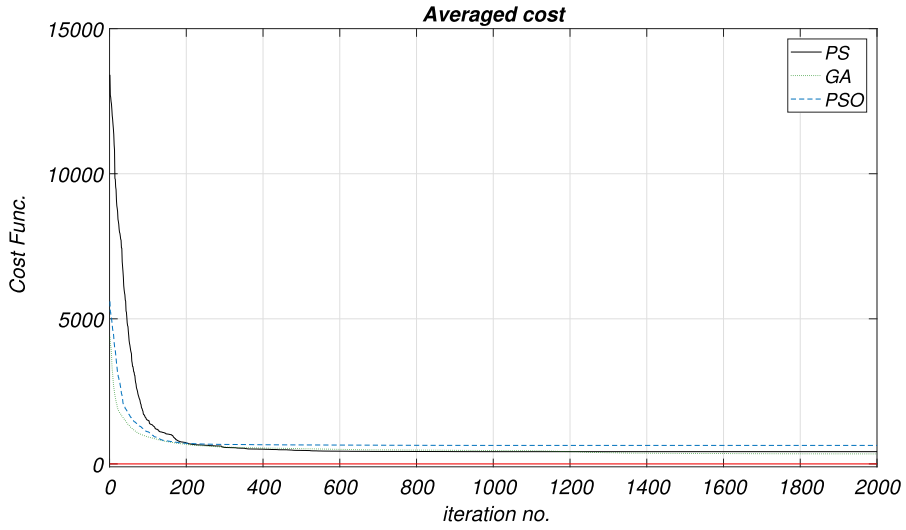

Fig. 21. Average of objective function value for each metaheuristic in the fourth case.

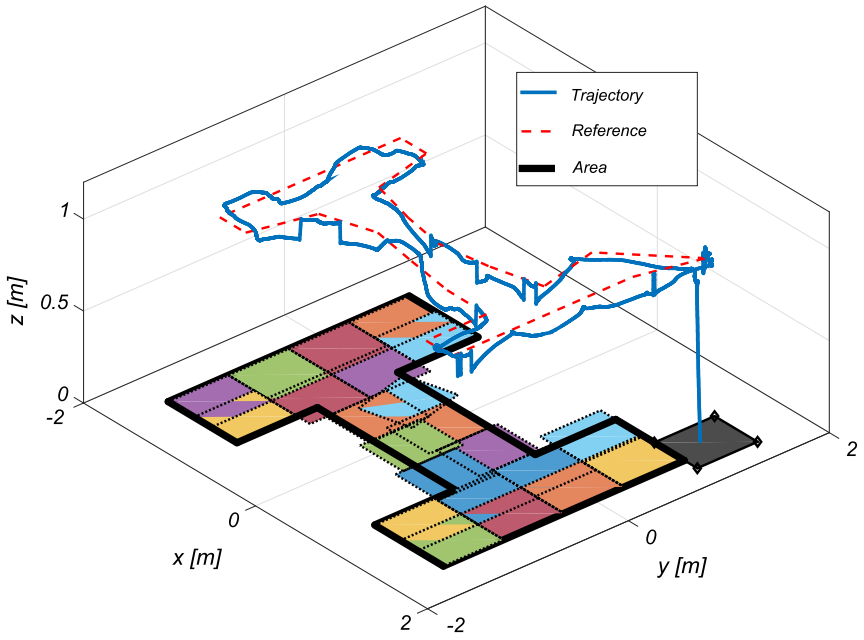

Fig. 22. Covering a complex polygon with 26 rectangles.

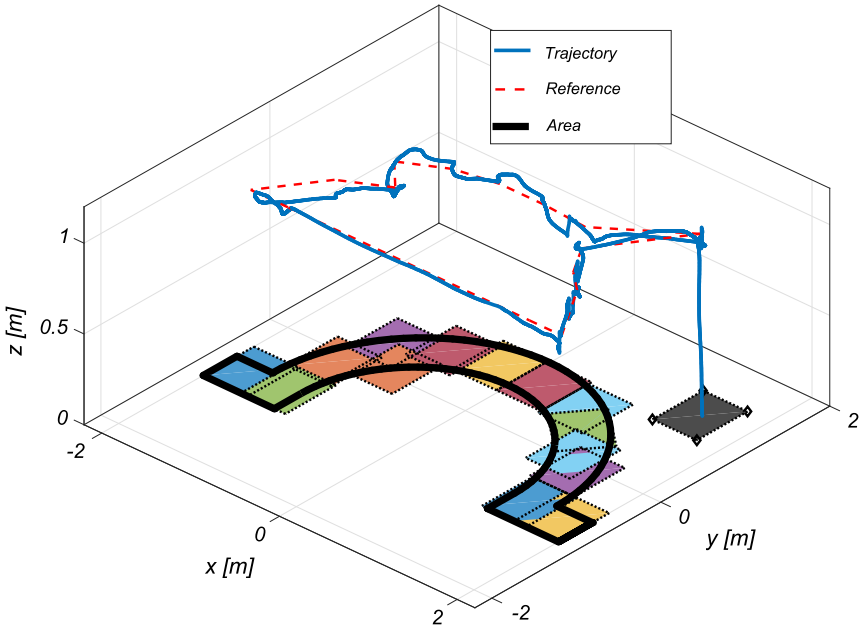

Fig. 23. Covering a complex polygon with 14 rectangles.

Table 3
The comparison of the path length between the proposed path planner and the path from grid based method and method (Mansouri et al., 2017).

	Case 1	Case 2	Case 3	Case 4	Case 5
Path from grid-based method with fixed altitude of 1 m	12.4 m	5.5 m	7.8 m	15.22 m	7.09 m
Path from method (Mansouri et al., 2017) with fixed altitude of 1 m	12.4 m	4.5 m	7.1 m	14.22 m	6.48 m
Proposed path planner	12.4 m	3.87 m	6.4 m	12.48 m	5.08 m

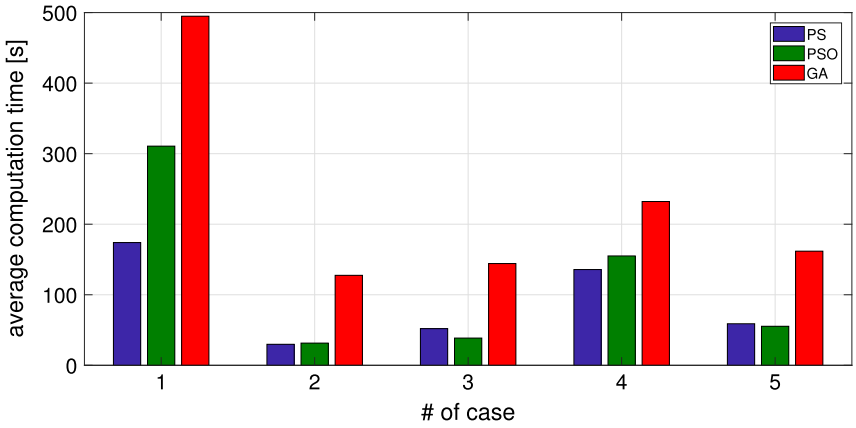

Fig. 24. Average computation time of each method for each scenario.

References

Abramson, M. A. (2002). *Pattern search algorithms for mixed variable general constrained optimization problems* (Ph.D. thesis), École Polytechnique de Montréal.

Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P. N., & Hull, D. (2002). Morse decompositions for coverage tasks. *International Journal of Robotics Research*, *21*(4), 331–344.

Alexis, K., Nikolakopoulos, G., & Tzes, A. (2011). Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, *19*(10), 1195–1207.

Alexis, K., Nikolakopoulos, G., & Tzes, A. (2012). Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory & Applications*, *6*(12), 1812–1827.

Alexis, K., Nikolakopoulos, G., Tzes, A., & Dritsas, L. (2009). Coordination of helicopter UAVs for aerial forest-fire surveillance. In *Applications of intelligent control to engineering systems* (pp. 169–193). Netherlands: Springer.

Avellar, G. S., Pereira, G. A., Pimenta, L. C., & Iscold, P. (2015). Multi-uav routing for area coverage and remote sensing with minimum time. *Sensors*, *15*(11), 27783–27803.

Brown, M., & Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, *74*(1), 59–73.

Burt, P. J., & Adelson, E. H. (1983). A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, *2*(4), 217–236.

Butler, Z. J., Rizzi, A. A., & Hollis, R. L. (1999). Contact sensor-based coverage of rectilinear environments. In *Intelligent control/intelligent systems and semiotics, 1999. Proceedings of the 1999 IEEE International symposium on* (pp. 266–271). IEEE.

Culberson, J. C., & Reckhow, R. A. (1988). Covering polygons is hard. In *Foundations of computer science, 1988, 29th annual symposium on* (pp. 601–611). IEEE.

Daniels, K., & Inkulu, R. (2001). An incremental algorithm for translational polygon covering. In *University of Massachusetts at lowell computer science technical report* (pp. 1–31).

Davis, L. (1991). *Handbook of genetic algorithms*.

Doherty, P., & Rudol, P. (2007). A UAV search and rescue scenario with human body detection and geolocalization. In *AI 2007: Advances in artificial intelligence* (pp. 1–13). Springer.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro machine and human science, 1995. Proceedings of the sixth international symposium on* (pp. 39–43). IEEE.

Engelbrecht, A. P. (2006). *Fundamentals of computational swarm intelligence*. John Wiley & Sons.

Franzblau, D. S., & Kleitman, D. J. (1984). An algorithm for covering polygons with rectangles. *Information and Control*, *63*(3), 164–189.

Fresk, E., Mansouri, S. S., Kanellakis, C., & Nikolakopoulos, G. (2017). Reduced Complexity Calibration of MEMS IMUs. In *25th mediterranean conference on control and automation*.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258–1276.

Haupt, R. L., & Haupt, S. E. (2004). *Practical genetic algorithms*. John Wiley & Sons.

Heinrich-Litan, L., & Lübbecke, M. E. (2005). Rectangle covers revisited computationally. In *International workshop on experimental and efficient algorithms* (pp. 55–66). Springer.

Holland, J. H. (1975). *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence*. Ann Arbor, MI: University of Michigan Press.

Hooke, R., & Jeeves, T. A. (1961). "Direct search" solution of numerical and statistical problems. *Journal of the ACM*, *8*(2), 212–229.

Kamel, M., Stastny, T., Alexis, K., & Siegwart, R. (2017). Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In *Robot operating system (ROS)* (pp. 3–39). Springer, Cham.

Kanellakis, C., & Nikolakopoulos, G. (2017). Survey on computer vision for UAVs: Current developments and trends. *Journal of Intelligent & Robotic Systems*, 1–28. http://dx.doi.org/10.1007/s10846-017-0483-z.

Kanellakis, C., Terreran, M., Kominiak, D., & Nikolakopoulos, G. (2017). On vision enabled aerial manipulation for multirotors. In *Emerging technologies and factory automation, 2017 IEEE 22nd international conference on*. IEEE.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, *29*(2), 315–378.

Kennedy, J. F., Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. Morgan Kaufmann.

Kolda, T. G., Lewis, R. M., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, *45*(3), 385–482.

Lewis, R. M., Torczon, V., & Trosset, M. W. (2000). Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, *124*(1), 191–207.

Lingelbach, F. (2004). Path planning using probabilistic cell decomposition. In *Robotics and automation, 2004. Proceedings. 2004 IEEE international conference on* (Vol. 1) (pp. 467–472). IEEE.

Lynen, S., Achtelik, M., Weiss, S., Chli, M., & Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to MAV navigation. In *Proc. of the IEEE/RSJ conference on intelligent robots and systems*.

Mansouri, S. S., Georgoulas, G., Gustafsson, T., & Nikolakopoulos, G. (2017). On the covering of a polygonal region with fixed size rectangles with an application towards aerial inspection. In *2017 25th mediterranean conference on control and automation*.

Mansouri, S. S., Kanellakis, C., Fresk, E., Kominiak, D., & Nikolakopoulos, G. (2018). Cooperative uavs as a tool for aerial inspection of the aging infrastructure. In *Field and service robotics* (pp. 177–189). Springer.

Mansouri, S. S., Karvelis, P., Georgoulas, G., & Nikolakopoulos, G. (2017). Remaining useful battery life prediction for uavs based on machine learning. *IFAC-PapersOnLine*, *50*(1), 4727–4732.

Matai, R., Singh, S. P., & Mittal, M. L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. *Traveling Salesman Problem, Theory and Applications*, 1–24.

Milnor, J. (2016). *Morse theory (AM-51)* (Vol. 51). Princeton University Press.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313.

Niethammer, U., Rothmund, S., James, M., Travelletti, J., & Joswig, M. (2010). UAV-based remote sensing of landslides. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *38*(Part 5), 496–501.

Orman, A., & Williams, H. P. (2006). A survey of different integer programming formulations of the travelling salesman problem. *Optimisation, Econometric and Financial Analysis*, *9*, 93–108.

Parsopoulos, K. E. (2010). *Particle swarm optimization and intelligence: advances and applications: advances and applications*. IGI Global.

Quigley, Morgan, et al. ROS: an open-source Robot Operating System. *ICRA workshop on open source software*. Vol. 3. No. 3.2. 2009.

Raguram, R., Frahm, J.-M., & Pollefeys, M. (2008). A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. *Computer Vision–ECCV 2008*, 500–513.

Shivashankar, V., Jain, R., Kuter, U., & Nau, D. (2011). Real-time planning for covering an initially-unknown spatial environment. In *Twenty-fourth international FLAIRS conference*.

Stoyan, Y. G., Romanova, T., Scheithauer, G., & Krivulya, A. (2011). Covering a polygonal region by rectangles. *Computational Optimization and Applications*, *48*(3), 675–695.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

Valente, J., Barrientos, A., del Cerro, J., Rossi, C., Colorado, J., Sanz, D., et al. (2011). Multi-robot visual coverage path planning: Geometrical metamorphosis of the workspace through raster graphics based approaches. In *International conference on computational science and its applications* (pp. 58–73). Springer.

Wong, S. (2006). *Qualitative topological coverage of unknown environments by mobile robots* (Ph.D. thesis), ResearchSpace@ Auckland.

Xu, L. (2011). *Graph planning for environmental coverage*. Carnegie Mellon University.

Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Zongjian, L. (2008). UAV for mappinglow altitude photogrammetric survey. *International Archives of Photogrammetry and Remote Sensing*, *37*, 1183–1186.