Efficient Trajectory Planning for WSN Data Collection with Multiple UAVs

D. Alejo, J.A. Cobano, G. Heredia, J. Ramiro Martínez-de Dios and A. Ollero

Abstract This chapter discusses the problem of trajectory planning for WSN (Wireless Sensor Network) data retrieving deployed in remote areas with a cooperative system of UAVs (Unmanned Aerial Vehicles). Three different path planners are presented in order to autonomously guide the UAVs during the mission. The missions are given by a set of waypoints which define WSN collection zones and each UAV should pass through them to collect the data while avoiding passing over forbidden areas and collisions between UAVs. The proposed UAV trajectory planners are based on Genetics Algorithm (GA), RRT (Rapidly-exploring Random Trees) and RRT* (Optimal Rapidly-exploring Random Trees). Simulations and experiments have been carried out in the airfield of Utrera (Seville, Spain). These results are compared in order to measure the performance of the proposed planners.

1 Introduction

Research and development of Unmanned Aerial Vehicles (UAVs) and aerial robots have been increasing in the last years due to the advantages that UAVs present over ground vehicles in terms of maneuverability and accessibility to remote areas. Different kinds of missions with UAVs have been addressed such as: surveillance [1], structure assembly [2], fire detection and monitoring [3], data collection [4], etc.

D. Alejo (✉) · J.A. Cobano · G. Heredia · J.R. Martínez-de Dios · A. Ollero
University of Seville, Camino de Los Descubrimientos S/N, Seville, Spain
e-mail: dalejo@us.es

J.A. Cobano
e-mail: jcobano@us.es

G. Heredia
e-mail: guiller@us.es

J.R. Martínez-de Dios
e-mail: jdedios@us.es

A. Ollero
e-mail: aollero@us.es

A. Koubâa and J.R. Martínez-de Dios (eds.), *Cooperative Robots and Sensor Networks 2015*, Studies in Computational Intelligence 604,
DOI 10.1007/978-3-319-18299-5_3

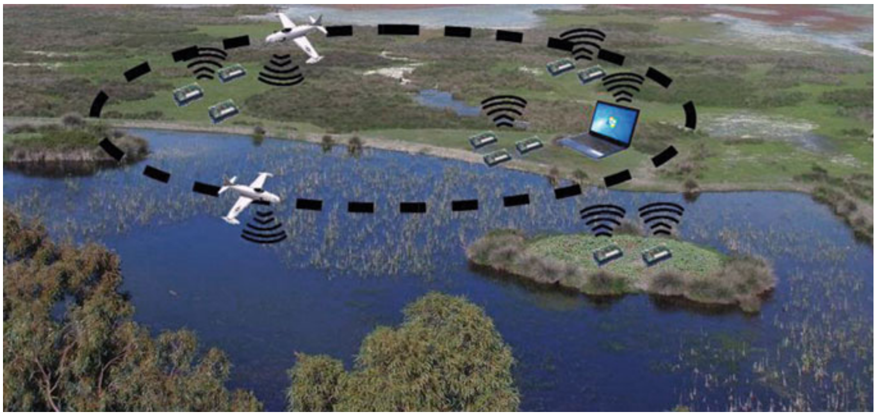

Fig. 1 Environmental monitoring in the PLANET project. WSN distributed over a large area have to be visited by a group of UAVs for data collection purposes

The work presented in this chapter has been developed within the framework of the UE-funded PLANET project (http://www.planet-ict.eu). The main objective of PLANET is the design, development and validation of an integrated platform to enable the deployment, operation and maintenance of large-scale/complex systems of heterogeneous networked Cooperating Objects, including Wireless Sensor and Actuator Networks and mobile robots. The platform support optimal and adaptive deployment and operation by means of mobile cooperating objects, i.e. vehicles, networked with static nodes. The platform has been validated in the monitoring of the Doñana Biological Reserve with very high ecological value and very sensitive to the impact of pollution.

The motivation of this work is the environmental monitoring in areas with difficult accessibility (see Fig. 1). A scenario in which a high number of nodes that have been deployed at known locations is considered. Particularly, this chapter addresses the planning of collision-free trajectories for data collection with UAVs in these zones. Firstly, the deployed nodes are grouped into groups taking into account the location of the nodes, their transmission ranges and message rates [4]. Each group has its WSN collection zone. The centers of the collection zones are considered as UAV trajectory waypoints. Then, a trajectory planning algorithm is used to ensure that the UAVs pass through the WSN collection zones taking into account the UAV kinematic constraints while avoiding collisions with among UAVs.

Therefore, trajectory planning algorithms should adapt to possible changes in the mission. Moreover, the algorithms should also be computationally efficient in order to ensure a suitable solution in a given time. Four different planners have been implemented to address this problem.

The main difficulty that has to be considered in order to develop the planners derives from the fact that the problem of trajectory planning is NP-hard [5, 6]. In addition, differential constraints given by the model of the UAV should be considered to make the problem tractable. Sampling-based techniques, as opposed to combinatorial planning, are usually preferred in these NP-hard problems. These planning schemes are suitable when the solution space is hard to model or unknown a priori because of its dynamic nature. Furthermore, planning optimal collision-free trajectories for UAVs leads to optimization problems with multiple local minima in most cases and, thus, local optimization methods as gradient-based techniques are not well suited to solve it. The application of sampling-based techniques is an efficient and effective alternative for this problem.

The first of the proposed path planners is based on genetic algorithms (GA). These algorithms are randomized algorithms that can give quasi-optimal solutions and that can be adapted to multiple different problems, such as patter recognition [7], control system design [8], shortest path routing [9] and collision avoidance [10]. They are a particular kind of evolutionary techniques that randomly generate and evolve a population of individuals. The goodness of the individuals is calculated by means of a criteria function.

The second path planer is based on Rapidly-exploring Random Tree (RRT) planning algorithm [11]. RRT planning algorithms have moderate computational requirements and can be easily adapted to changes in the environment conditions. Therefore, these algorithms are used to quickly compute a collision-free solution. The drawback of this method is that the generated path is not optimized.

It would be desirable to refine the solution obtained with RRT algorithm in the available computational time. The third path planner, based on RRT* (Optimal Rapidly-exploring Random Trees) planning algorithm [12], is designed to fulfill this purpose. The main advantage of RRT* is that it computes smoother trajectories than the ones obtained with RRT, it can optimize the length of the paths and the generated paths are more predictable.

The main contributions of this chapter are summarized in the following points:

- The chapter proposes four UAV trajectory planning methods for WSN data collection. The proposed methods adopt sampling-based approaches in contrast to traditional schemes that rely on heuristic-based reasoning. Besides, three of the four proposed methods are classified into the class named "anytime techniques", i.e. methods that ensure a valid solution in a very short time and uses the remaining time to refine the solution searching for a better solution.
- The chapter compares the four UAV trajectory planning methods and evaluate them in massive simulation experiments. The RRT_i^* planner shows the best performance and therefore it was selected for experimentation in real hardware experiments.
- The chapter validates the RRT_i^* method in real experiments performed with a Megastar fixed-wing UAV. The validation shows that the distance between UAVs and center of the WSN collection zones are lower than the collection zone radius, enabling correct data retrieval.

The chapter is organized into seven sections. The state of the art is presented in Sect. 2. The background of the implemented planners is given in Sect. 3. The addressed problem is described in Sect. 4. Section 5 presents the three path planners implemented. The simulations and experiments performed are shown in Sects. 6 and 7, respectively. Finally, the conclusions are detailed in Sect. 8.

2 State of the Art

This section briefly summarizes the state of the art in the main topics involved in the chapter: collection of WSN data using UAVs and UAV path planning.

Data collection using UAS has been an attractive research topic due to its wide potentialities. Some works have proposed theoretical and/or simulated analysis and architectures, see e.g. [13], which describes an architecture for the integration of WSNs and aerial vehicles or [14].

In the simplest approach, the deployed nodes gather and buffer the readings. When the UAS flies near the nodes it sends a beacon message and the nodes send the readings in reply. The UAS collects the data. Experiments with the mentioned approach are described in [15, 16]. In [4] the scalability of the basic scheme is increased grouping the deployed nodes. The groups and their collection zones are pre-computed taking into account the nodes locations and radio coverage, among others. This approach has been extended for multi-UAV cooperation purposes in this paper.

Work [17] proposes a UAS-WSN cooperation scheme where the results of the WSN operation are used to update the UAS flight plan and, at the same time, the UAS trajectory is considered in the operation of the WSN in order to improve the data collection performance. These UAS-WSN cooperation strategies require advanced UAS planning methods. In contrast, the work presented in this paper focus on the safe trajectory planning for multi-UAV systems.

UAV path planning with safety requirements has been largely investigated. A* and Theta* algorithms are efficient classic alternatives, but they are not suitable for multi-UAV planning and require a discretization of the state space that could be prohibitive in open 3D spaces.

This paper focus on multi-UAV planning, in particular in solving a Conflict Detection and Resolution (CDR) problem. Considering this problem with multiple mobile UAVs is NP-hard [18]. Sampling-based techniques match particularly well when the solution space is hard to model or unknown a priori because of its dynamic nature.

A large number of algorithms have been developed to avoid applying the brute force approach. In [19, 20] a detailed survey on CDR techniques and planning algorithms is presented, respectively. CDR methods can be coarsely classified in deterministic and stochastic.

Deterministic CDR methods perform an exploration of the feasible solution set using search procedures usually directed by the local behavior of the optimization function. These methods include non-linear programming (NLP) [21], integer programming [22, 23], dynamic programming if the problem can be broken down into simpler sub-problems [24], collocation methods reducing the number of dimensions of the problem [25, 26], among many others. These algorithms do not adapt well to changes of the environment.

Stochastic CDR methods perform a random exploration of the problem by using random variables [27]. This involves random objective functions or random constraints. These methods include random search methods [28], evolutionary computation methods [29, 30], particle swarm optimization [31], ant colony optimization methods [32], simulated annealing [33], among others. Rapidly-exploring Random Tree (RRT) planning algorithm is also commonly used for path planning [12, 34]. RRT is constructed incrementally in a way that quickly reduces the expected distance of a randomly-chosen point to the tree. This approach is suitable for path planning problems that involve obstacles and differential constraints.

3 Background

This paper is focused on path planning. It aims to apply state of the art planning methods to real systems and make replanning procedures online. In this section, the basic applied algorithms are explained.

The planification problem consists in searching for a path between an initial configuration of one or more vehicles q_{init} to a desired configuration q_{goal} while avoiding collisions with static obstacles and between vehicles. Let C be the set of possible configurations of the system, C_{free} be the set of configurations that are collision-free, while $C_{obs} = C \setminus C_{free}$ is the set of colliding configurations.

In this paper, GA optimization method has been applied in order to solve the path planning problem, as well as three randomized sampled-based planners: RRT, RRT and RRT_i^*.

3.1 Genetic Algorithm

Genetic algorithm (GA) is a heuristic search that mimics the process of natural selection, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

Figure 2 represents the basic flowchart of the GA. First, a population of candidate solutions to an optimization problem (called individuals or phenotypes) is created and then evolves toward better solutions. This evolution starts from an initial population generated randomly and is an iterative process. The population in each iteration is a generation. The fitness of every individual in the population is evaluated in each generation. The fitness is defined by the value of the objective function in the optimization problem being solved. Moreover, each individual has a set of properties which can be randomly mutated and recombined (crossover) to generate a new individuals, which are usually called offspring. Then, the offspring and old

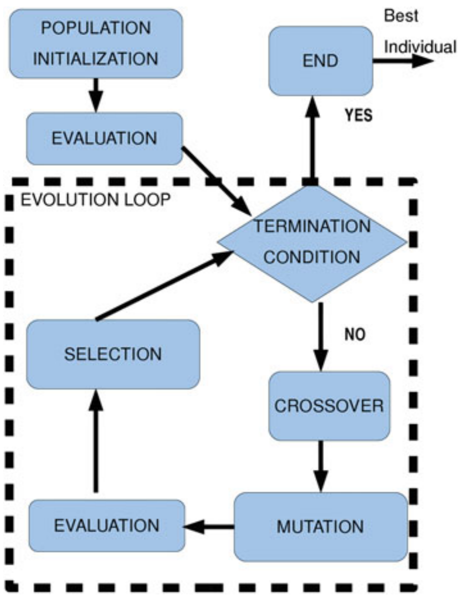

Fig. 2 Flowchart of the planning algorithm based on genetic algorithms

population compete in order to generate a new generation with the same size. This new generation will be used in the next iteration of the algorithm.

Commonly, the algorithm terminates when either a maximum number of generations has been produced, a satisfactory fitness level has been reached for the population, or a maximum allowed computation time is reached.

3.2 RRT

RRT is a planning algorithm first proposed in [34]. The basic RRT algorithm is shown in Algorithm 1. Note that some procedures are necessary for the algorithm to be run. Below you can find the list of procedures.

- **Nearest**(G, q). Searches for the closest vertex in the graph G to the configuration q.
- **Steer**(q_1, q_2). Obtains the configuration q_3 that is the closest to q_2 integrating the model from q_1 one step.
- **CollisionFree**(q_1, q_2). Returns **true** if the path that unites q_1 and q_2 is collision free.
- q_{rand} = **SampleFree**(). Returns a configuration $q_{rand} \in C_{free}$.

RRT starts a tree by creating the root in the starting configuration (q_{init}) and extends the tree by generating random samples (x_{rand}) of the configuration space and by making the tree extend to that new point. When the new sample is generated, the closest node to it is selected and the tree is extended from this sample and a new node is added (x_{new}). This new node is generated by integrating the model proposed in [35] from v_{near} with a random control signal. If the path between v_{near} and q_{new} is collision-free this node is added to the tree. This procedure is repeated until the new node is sufficiently near from the final state q_{goal}. Note that this algorithm ensures that the generated paths are flyable because they are generated by integrating the UAV model.

Many different variants of the RRT algorithm have been proposed over the years, in particular the variants that propose the growth of two trees, one starting from the goal point and one from the starting point claim to outperform basic RRT [36]. These variants are called bi-RRT. Another common improvement is to make a bias in the sampling procedure towards the goal, i.e. taking the goal as the sampled stated with a configured probability (usually 10 %).

Algorithm 1 Basic RRT algorithm

```
Require: RRT($q_{init}, q_s$)
1: $V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$
2: repeat
3:   $x_{rand} \leftarrow SampleFree()$
4:   $v_{nearest} \leftarrow Nearest(G = (V, E), x_{rand})$
5:   $x_{new} \leftarrow Steer(v_{nearest}, x_{new})$
6:   if $CollisionFree(v_{nearest}, x_{new})$ then
7:     // Add the new vertex and the connection
8:     $V \leftarrow V \cup \{x_{new}\}$
9:     $E \leftarrow E \cup \{(v_{nearest}, x_{new})\}$
10:  end if
11: until $q_s \in G = \{V, E\}$
12: return $G = \{V, E\}$
```

3.3 RRT*

The main drawback of the RRT algorithm, when applied to mobile robots, is that the basic RRT yielded to randomized like motions that were not properly optimized and difficult to forecast. In order to overcome these drawbacks, RRT* planning algorithm makes two main modifications to the original algorithm [12] as shown in Algorithm 2.

First, when a new sample is generated, the algorithm attempts to connect it not only to the nearest neighbor but also to a set of neighbors that are close enough. Only the connection that minimizes the length of the path between the new sample and the starting configuration is added to the tree (steps 9–16).

The other modification is called the *rewiring step*. In this phase, the current cost of the neighbors of the new sample is compared to the cost that would be obtained by traveling through the new sample. If this new cost is lower than the current cost, the graph is rewired (steps 20–23).

Some extra functions are necessary for RRT* algorithm to work. These are:

- **Cost** ($n \in V$). Associates the node n with its calculated cost.
- **c(Path)**. Gives a cost to a calculated past. In the basic version the cost is the distance of the path.
- **Near (G, q, d)**. Returns a set of vertices $N = \{n \in V \backslash dist(n, q) < d\}$.

Algorithm 2 RRT* algorithm

```
Require: RRT($q_{init}, q_s$)
1: $G = \{V, E\}$
2: $V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$
3: repeat
4:   $x_{rand} \leftarrow SampleFree()$
5:   $v_{nearest} \leftarrow Nearest(G, x_{rand})$
6:   $x_{new} \leftarrow Steer(v_{nearest}, x_{new})$
7:   if $CollisionFree(v_{nearest}, x_{new})$ then
8:     $V \leftarrow V \cup \{x_{new}\}$
9:     // Connect along a minimum-cost path
10:     $U \leftarrow Near(G, x_{new}, \eta)$
11:     $v_{min} \leftarrow v_{nearest}$;$c_{min} \leftarrow Cost(v_{nearest}) + c(Path(v_{nearest}, x_{new}))$;
12:     for $all\ u \in U$ do
13:       if $CollisionFree(u, x_{new})\ and\ Cost(u) + c(Path(u, x_{new})) < c_{min}$ then
14:         $v_{min} \leftarrow u$; $c_{min} \leftarrow Cost(u) + c(Path(v_{nearest}, x_{new}))$
15:       end if
16:     end for
17:     $E \leftarrow E \cup \{(v_{min}, x_{new})\}$
18:     // Rewire vertices
19:     for $all\ u \in U$ do
20:       if $CollisionFree(x_{new}, u)\ and\ Cost(x_{new}) + c(Path(x_{new}, u)) < Cost(u)$ then
21:         $v_{parent} \leftarrow Parent(u)$
22:         $E \leftarrow (E \backslash \{(v_{parent}, u)\}) \cup \{(x_{new}, u)\}$
23:       end if
24:     end for
25:   end if
26: until $q_s \in G$
27: return $G$
```

4 Description of the Problem

The scenarios considered in PLANET project consist of several UAVs in a common airspace. The scenario contains forbidden flight zones, for example due to presence of protected species. These forbidden flight zones will be modelled as static obstacles that the UAV trajectories must avoid. The rest of UAVs in the same common airspace are considered as mobile obstacles. A mission is considered safe if during the flight, the separation between each UAV and obstacles is greater than a given safety distance (minimum separation). If a collision is detected, the trajectory of the UAVs should be updated to avoid the collision.

Therefore, the goal is to compute collision-free trajectories while minimizing the changes of the trajectory of each UAV involved in WSN data collection. Assume that the data collection zones corresponding to the groups assigned to one UAV are denoted as $\{W_1, W_N\}$, so a UAV trajectory is defined by this sequence of waypoints. The proposed method computes safe trajectories for UAVs passing through $\{W_1, W_N\}$ in presence of static and mobile obstacles. The solution only considers the addition of intermediate waypoints. The trajectory computed should ensure that the UAV passes through the WSN collection zones to a minimum distance from the center of the zone.

Although the trajectories are planned to be free of collisions, UAVs can deviate from these trajectories due to wind or other disturbances. Thus, a collision detection and avoidance method is needed to check that the actual trajectory is free of collisions. A collision detection module should check if the actual UAV trajectory is free of collisions. If a collision with a static obstacle or with another UAV is detected, a new collision-free trajectory should be computed.

The information that the system needs in order to solve the problem is the following:

- Initial trajectory of each UAV
- Parameters of the model of each UAV
- Initial location and goal location of each UAV
- Look-ahead time to know the available computation time.

5 Collision-Free Trajectory Planning Algorithm

This section describes the blocks of the proposed algorithm to plan collision-free trajectories. First, a detection algorithm is implemented to detect possible collisions. Then, a collision-free trajectory planning algorithms based on genetic algorithms (GA), RRT and RRT* are implemented to solve the detected conflicts.

Each UAV is supposed to be surrounded by a cylinder which cannot be entered by the other UAVs or obstacles (see Fig. 3). A collision is detected if the horizontal separation between UAVs, the Euclidean distance in the XY plane, is lower than the Dxy_{min} and at the same time the vertical separation is lower than Dz_{min}. This technique presents as advantages the low execution time and the need for few parameters to describe the system.

The resolution block is executed when an alert takes place. The alerts are produced for two main reasons:

1. Detection algorithm: a collision of one UAV with a static or mobile obstacle is detected.

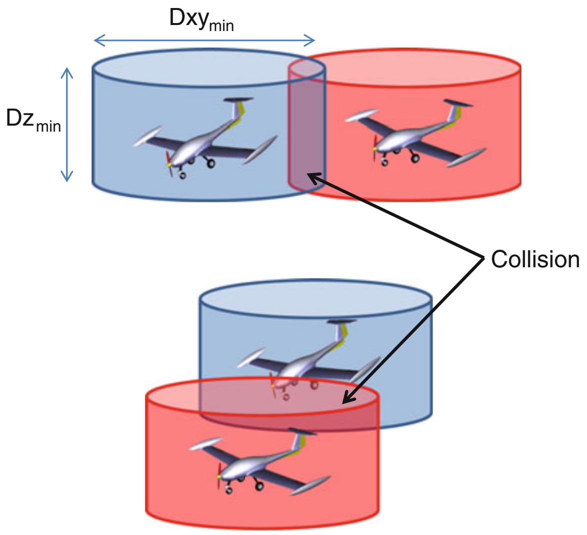

Fig. 3 Detection algorithm: each aerial vehicle is described by a *cylinder*

2. Cooperation WSN-UAV: there is a change in the WSN collection zones and so the UAV flight plan changes.

In both cases the trajectory needs to be updated. This update is carried out by the collision-free trajectory planning algorithms based on GA [30], RRT [34] and RRT* [12]. The planning algorithm adds intermediate waypoints between WSN data collection zones in order to compute collision-free trajectories for each UAV by considering the kinematics constraints. Three planners have been implemented. Each planner is described in detail in next sections.

5.1 Genetic Algorithm

In this chapter, GA has been applied to path planning. Therefore, an individual will represents possible trajectories of all the UAV in the system. In this case, the individuals are coded by sequences of waypoints that represent a possible trajectory for each UAV.

Firstly, an initial population is randomly generated with an uniform distribution in the search space. It is important to point out that the initial location and the goal location are always the same for each trajectory of the UAV. Figure 4 (left) represents an example of 2D flight plan that could be generated in the initial population. The UAV starting location is (0, 0) and the goal location is (5, 0). The genome is given by the vector $V = (1, 1, 3.5, 2)$ that yields to two intermediate waypoints: $WP1 = (1, 1)$ and $WP2 = (3.5, 2)$.

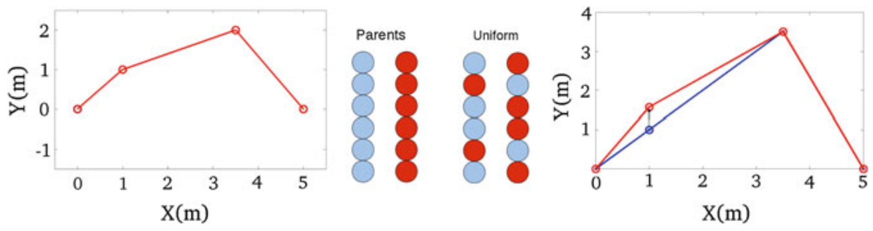

Fig. 4 *Left* Example of flight plan given by genome $G = (1, 1, 3.5, 2)$. *Middle* Crossover operator that has been used in this paper. *Right* Mutation of the second gene of the genome G (*blue line*) yields to a new genome $G_{mutation} = (1, 1.57, 3.5, 2)$ (*red line*)

Once the initial population has been generated, the following step is to evaluate the fitness of each candidate trajectory (individual). The proposed cost function that will define the fitness value to each candidate trajectory is proportional to the length of each trajectory. In addition a penalty is added if the trajectories yield to collision. So, the proposed cost function is:

$$J_i = L_i + P_{i,collision}, \tag{1}$$

where i indicates the ith iteration, L_i is the sum of the length of each UAV trajectory and $P_{i,collision}$ is the penalty added when a collision is detected.

In our implementation, the uniform crossover operator has been used. This operator randomly selects each gene from one parent as seen in Fig. 4 (middle). The mutation operator considered in the implemented algorithm randomly changes the mutated gene with a normal distribution centered in the original gene value and with configurable standard deviation. Figure 4 (right) represents an example of modification of the second gene of genome G (in blue), the new genome $G_{mutation}$ is represented in red.

By iteration of the selection and reproduction processes, the genetic algorithm ends up computing a near-optimal trajectory that ensures a collision-free trajectory, provided it exists.

*5.2 RRT and RRT**

Planning algorithms to compute collision-free trajectories based on RRT and RRT* planning algorithms have been also implemented. They can be considered as a planning algorithm based on two steps:

1. RRT computes efficiently a first collision-free trajectory that solves the problem. This solution ensures safety but might have low performance.

2. RRT* is run to refine the solution from step 1 trying to improve its performance. This method generates a time-dependent improved trajectory while safety is maintained.

Firstly, a collision-free trajectory is generated by using an efficient algorithm based on RRT planning algorithm that ensures reactive fast timely reaction. RRT are greedy-based purely randomized techniques that can efficient find a collision-free trajectory.

Once a safe solution has been computed, the solution is refined trying to improve its performance (time-dependent improved trajectory). The concept of time-dependent improved trajectory means that other solution will be computed during the available time to improve the performance. This stage exploits this time to find a better trajectory. Therefore, the algorithm provides collision-free solutions even in case of requiring very short responses. In case of having more time the solutions are improved as the computation time increases. A RRT* planning algorithm is used to refine the initial solution. RRT* will improve the solution in the available horizon time. These algorithms provide a relatively good solution in short times and allow a simple implementation. These methods have been shown to have a potential to solve large-scale problems efficiently in a way that is not possible for deterministic algorithms.

However, RRT* planning algorithm uses a simple interpolation step in the steering procedure (see Sect. 3). Because of this, the new connections are checked by considering the kinematics constraints: curvature and maximum climb or descent rate. The connection attempts that do not fulfill this are discarded. The following kinematics constraints are considered from the parameters of the Megastar UAV:

$$R > 25\,\mathrm{m}, \tag{2}$$

$$-5\,\mathrm{m/s} < C < 5\,\mathrm{m/s}, \tag{3}$$

where R is the turning radius and C is the climb or descent rate.

Last but not least, some improvements of the original versions of RRT and RRT* have been introduced in order to reduce the computational time of the planning algorithm and to generate paths with better clarification. In first stages of the algorithm, we propose the use of a non-uniform random distribution in order to explore first the zones that are near the WSN collection zones. In this case, a multivariate normal distribution has been used to produce the new samples. By using this sampling distribution, the explored space by the tree is much more oriented to the interesting areas. In addition, we bias the sampling towards the goals in the first stages of the algorithm.

Finally, some improvements proposed in [37] can be applied when a solution has been found. Firstly, the $localbias$ when using the RRT* algorithm is used. The main idea is to sample in the surroundings of a random point of the solution path in order to encourage rewiring steps of the RRT* algorithm. Also, the *node rejection* technique has been implemented. In this case, a node is rejected if the sum of its cost and the distance to the goal node is greater then the cost of the current solution. This technique is inspired in the A* algorithm [38]. The algorithm with the proposed additions is called RRT_i^* in order to distinguish it from the basic RRT* algorithm.

6 Simulations

Several simulations have been performed to validate the proposed algorithms. The planning algorithm based on genetic algorithm has been implemented in Matlab and C++ language and compiled with gcc-4.4.1. RRT and RRT* planning algorithms have been implemented in C++ by extending the Open Motion Planning Library (OMPL [39]) with the aforementioned improvements.

Taking into account the characteristics of the aerial vehicles involved in the simulations, the following dimensions of each cylinder are considered: $Dxy_{min} = 50$ m and $Dz_{min} = 20$ m.

The main objective of this section is to compare the proposed planners and justify the election of the final planner.

6.1 GA Versus RRT

First simulation considers a UAV which should pass through four WSN collection zones (see Fig. 5). The generated trajectory with RRT is represented with a blue line. Note that the RRT planning algorithm generates several intermediate waypoints in order ensure the correct data collection.

RRT planning algorithm is compared to genetic algorithm, both implemented in Matlab. Twenty simulations have been performed considering one UAV, four WSN groups and different obstacles. RRT ensures a fast initial solution and presents less computational load, 3.73 ± 0.21 s, than genetic algorithms, 6.89 ± 0.40 s. Moreover, RRT could add several intermediate waypoints to ensure that UAV passes through each WSN collection zone without collision. Genetic algorithms add only one intermediate waypoint between two consecutive waypoints. The computational time increases as more intermediate waypoints are added by Genetic algorithms.

6.2 RRT Results

Another scenario is considered with two UAVs (see Fig. 6). One collision is detected and RRT planning algorithm computes intermediate waypoints (green circles) to avoid it.

Another scenario with four WSN collection zones (WSN1, WSN2, WSN3 and WSN4) and two obstacles (red circle) is considered (see Fig. 7). UAV should pass

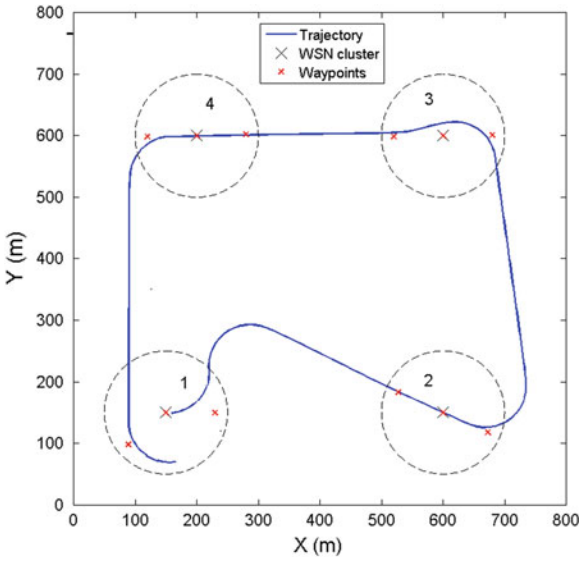

Fig. 5 Trajectory computed by RRT planning algorithm by considering four WSN collection zones

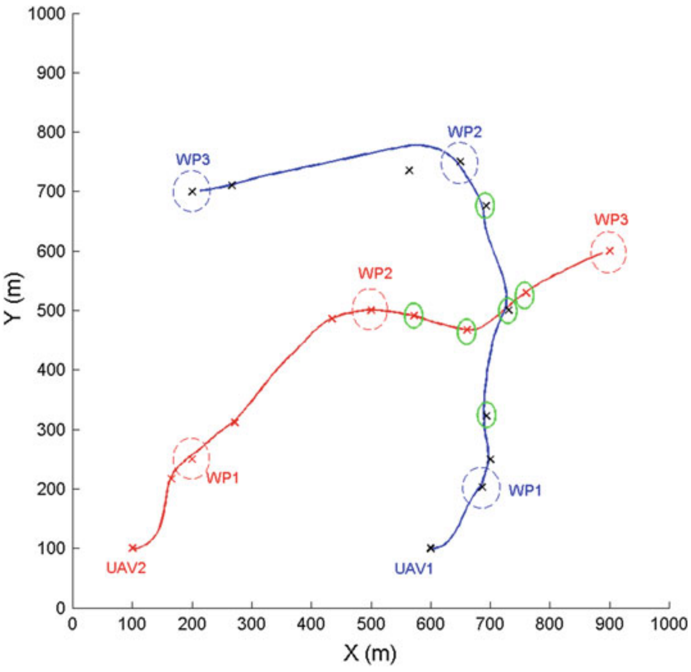

Fig. 6 Scenario with two UAVs. UAV1 (*blue lines*) should pass through the WP1, WP2 and WP3 (*black*) and UAV2 (*grey lines*) should pass through the WP1, WP2 and WP3 (*red*)

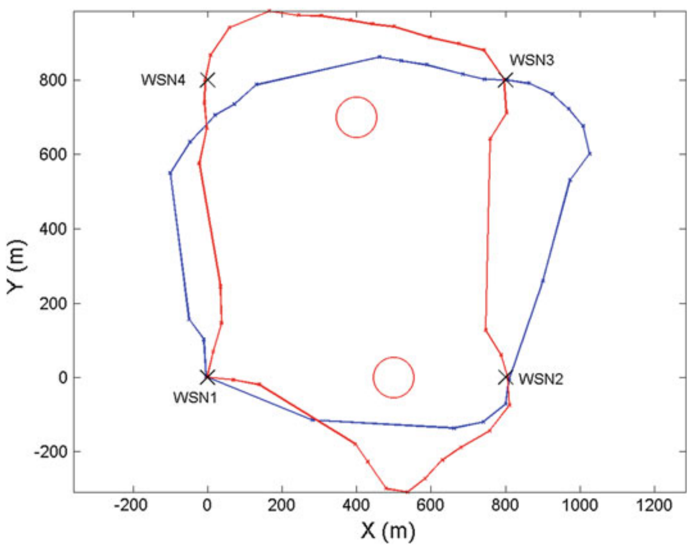

Fig. 7 Collision-free trajectories computed by RRT considering four WSNs and two obstacles

through the WSN collection zone considering a maximum distance. This distance considered is 50 m. Two different collision-free trajectories are shown in Fig. 7. Each trajectory passes through all the WSN groups. Note that the paths can change significantly, even though they are obtained with the same method.

The tree of the RRT planning algorithm is generated randomly, so every time the algorithm is executed a different collision-free trajectory is obtained from these trees. In addition, RRT lacks of an optimization method. In the original RRT algorithm the first obtained solution is not improved by any means, the remaining time is wasted in further exploring the search space.

6.3 RRT Results and RRT Comparison*

RRT* algorithm is able to improve the quality of the solution by rewiring the tree once the solution is found, overcoming the main issue of RRT algorithm. Thanks to this technique, the quality of the solution improves with the available time of execution. Figure 8 shows four different solutions considering two WSNs and five obstacles (red circle). The initial flight plan is defined by WSN1 and WSN2. Each solution is computed by considering different times of execution. The first solution (clear blue line) is computed in 5 s; the second one (blue line) in 10 s; the third one (red line) in 20 s; and the fourth one (green line) in 30 s. The improvement of the solution can be observed as the time of execution increases.

A comparison between RRT and RRT* planning algorithm has been performed. Figure 9 shows the solution trajectories computed by each one. Note that the trajectory

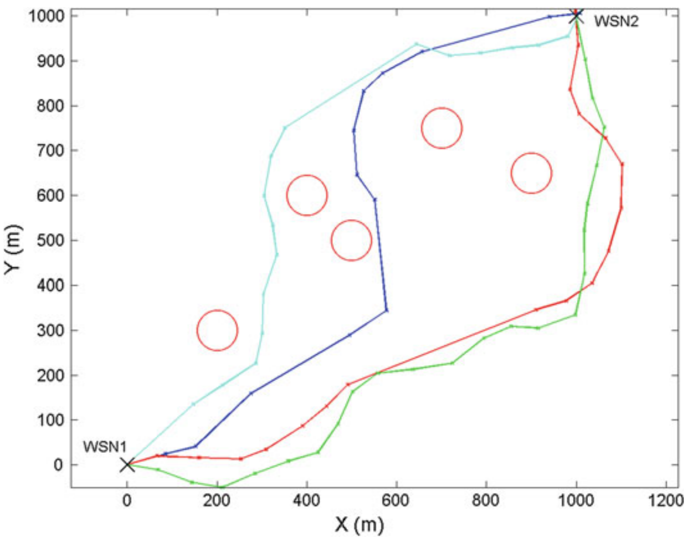

Fig. 8 Collision-free trajectories computed by RRT* considering different time of execution

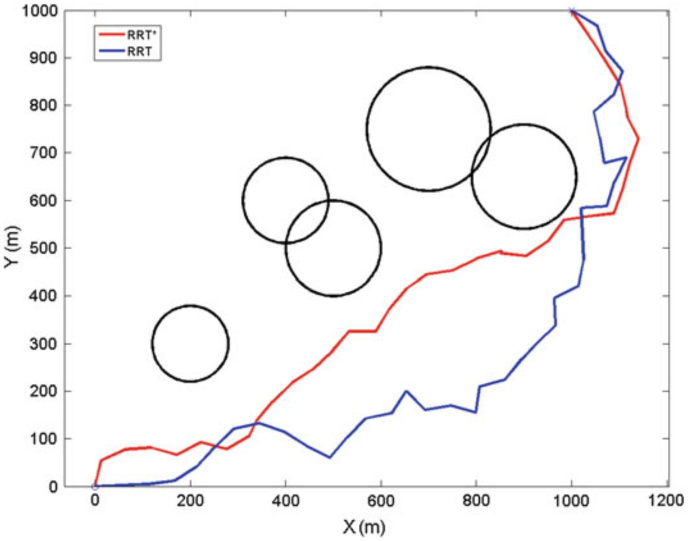

Fig. 9 Comparison between RRT (*blue line*) and RRT* (*red line*) generated trajectories

computed by RRT* planning algorithm is better and smoother that the computed by RRT.

The last simulation scenario is a multi-UAV scenario where two UAVs (UAV1 and UAV2) must fly over two collection nodes WSN1 and WSN2 respectively (see Fig. 10). This scenario has been also solved with RRT, RRT* and RRT_i^* algorithms.

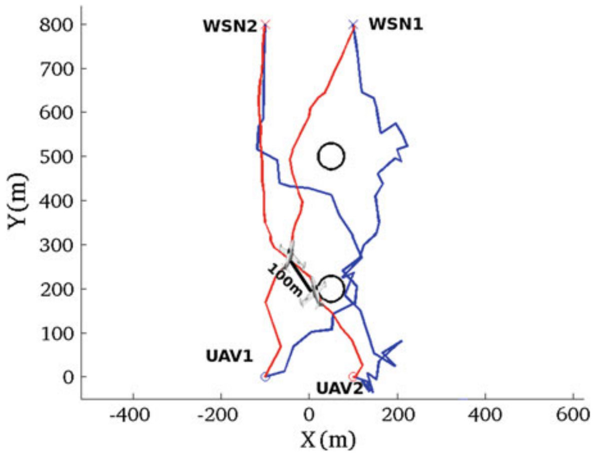

Fig. 10 Last simulation scenario with WSN1 and WSN2 to be visited by UAV1 and UAV2 respectively. Comparison between RRT (*blue*) and RRT_i^* (*red*) generated trajectories. Static obstacles are represented with black circles. The minimum distance between UAVs in RRT_i^* is represented

Figure 10 also represents the trajectory obtained with the RRT method in blue line and with RRT_i^* in red line. The minimum distance between UAV1 and UAV2 is represented and is greater than the safety distance ($Dmin_{xy} = 50\,\text{m}$).

Note that case the RRT_i^* is the final proposed planner. It starts with uniform sampling until a solution is found. When this happens, it automatically changes to local sampling. That is, sampling in the surroundings of a random point of the solution path with a Gaussian distribution with $\sigma = 5\,\text{m}$. Also, the node rejection technique (see Sect. 5.2) is active.

Table 1 represents the mean and standard deviation obtained by the three methods in generating the first solution and the cost of the best solution after 30 s of execution. Each method is applied twenty times. It is noticeable that RRT_i^* outperforms both RRT* and RRT algorithms by a great margin when comparing the cost of the best obtained solution. However, RRT is the method that generates a faster first solution, while RRT* and RRT_i^* have similar performance.

Lastly, we compare RRT* and RRT_i^* in terms of optimization of the first solution. In fact, the improvement (final cost minus initial cost) obtained with RRT* has mean 10.0 and standard deviation 12.9. In contrast, the improvement of RRT_i^* has mean 113.2 and standard deviation 48.4. Note that RRT* does not improve the solution significantly, while RRT_i^* makes a better job. In addition, no significant improvement is done by RRT* when $t > 10\,\text{s}$.

Table 1 Comparison of the execution time of the first solution (t) and cost of the best (c) solution when applying RRT, RRT* and RRT_i^*

Method	$\|t\|$	σ_t	$\|c\|$	σ_c
RRT	1.25	0.42	2301.3	175.1
RRT*	5.14	2.05	1931.4	138.6
RRT_i^*	4.88	2.37	1770	59.7

6.4 Simulation Remarks

As conclusions, this section shows that the RRT_i^* planner is the one that gives smoother and shorter trajectories when compared to the proposed RRT*, RRT and GA methods. However, RRT planner is able to plan in the control space of the model, so flyable trajectories are ensured. And additionally, RRT spends less computational time when searching for the first solution. Thus, a good combination of both methods is to use the proposed RRT planner to find a first solution and then refine this solution by using the proposed RRT_i^* method.

7 Experiments

An experiment has been performed by using Megastar UAV (see Fig. 11), at the airfield of Utrera (Sevilla, Spain), in order to test and demonstrate the correct operation of the proposed planning algorithm in a real environment.

7.1 Hardware Setup

Figure 11 shows the Megastar and the Piper fixed-wing UAVs that were used in the experiments as well as the WSN Nodes deployed on the ground and on-board the UAVs. Both fixed-wing UAVs are propelled with gasoline motors and both of them are equipped with an embedded computer PC104 that executes the UAS controller. This controller communicates with the Control Station through a Radio Modem link at 433 MHz, where all the processing is carried out through. The Radio Modem B connects to the ground station via USB. This connection offers the localization

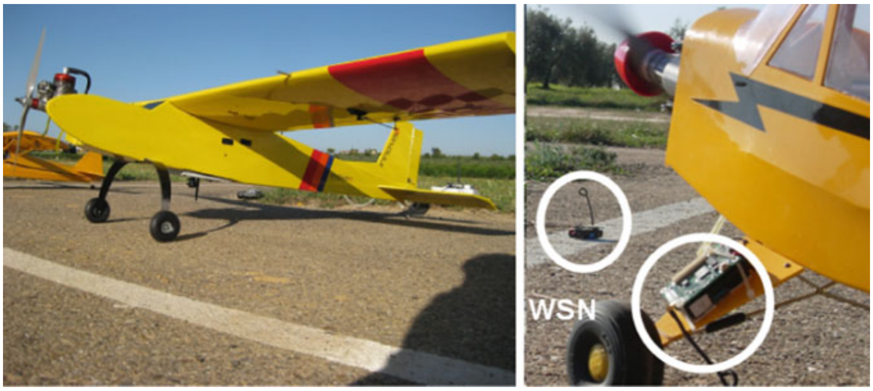

Fig. 11 *Left* Megastar fixed-wing UAV used in the experiments. *Right* Piper fixed-wing UAV used in experiments. Detail of the on-board and deployed WSN

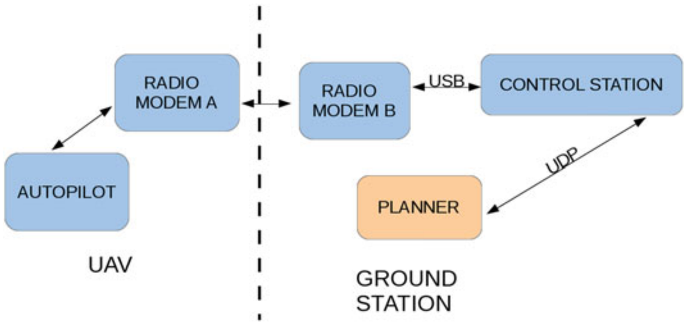

Fig. 12 *Left* Radio modem of the ground station. *Right* Communication diagram of the system

and status information of the UAV. Path planning is computed on the ground on an auxiliary computer that is linked to the Control Station using UDP protocol. The Control Station receives the list of waypoints computed by the path planning method and transmits them as the flight plan to the on-board controller. The on-board controller uses the list of waypoints as reference for the UAV controller. Figure 12 shows a scheme which represents the communications between the modules used in the experiments.

Motes from the *Mica2* family from *Crossbow Inc.* have been used. These systems have been selected because of requirements in size, weight and energy consumption. These motes use a 916 MHz transceiver at a rate of 9.6 Mbps. Their microprocessor is a *ATMega128* 8-bit @ 7 MHz. *Mica* family has different sensor boards such as the MTS400 with accelerometers and temperature and light sensors; or the MTS420 with GPS. However, in this experiment we are not interested in the real data provided by the WSN, but rather in establishing communications between deployed and on-board nodes.

7.2 Results

Figure 13 shows the location where the experiments have been carried out. Four different WSN collection zones were used $\{WP1, WP4, WP7, WP10\}$ and 5 WSN nodes were deployed inside each of them. Therefore, UAV should pass through the collection zones centered at $\{WP1, WP4, WP7, WP10\}$, see Fig. 13.

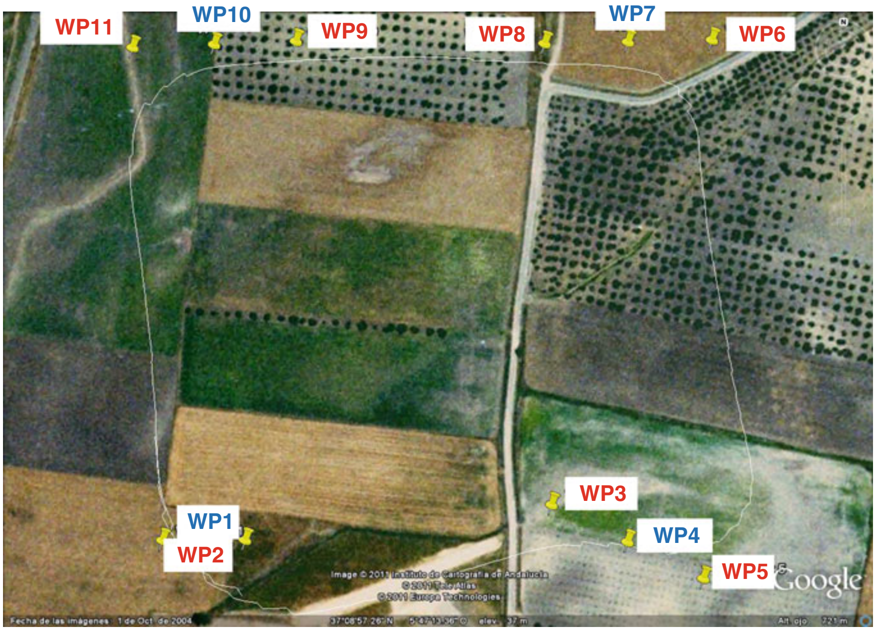

Fig. 13 Experiment performed at the airfield of Utrera (Seville, Spain). UAV should pass through WSN groups: WP1, WP4, WP7 and WP10

The RRT* method was executed and as result it provided a list of intermediate waypoints that forced that the UAV trajectory actually passed $\{WP1, WP4, WP7, WP10\}$. The list of intermediate waypoints were: $WP2$, $WP3$, $WP5$, $WP6$, $WP8$, $WP9$ and $WP11$, see Fig. 13. The complete list of waypoints was then used to generate the UAV flight plan, which was received by the UAV on-board controller. The UAV executed the flight plan and Table 2 shows the resulting distance between the UAV trajectory and the center of each WSN collection zone: the UAS passed through all collection zones at a distance suitable to perform WSN data collection. These distances obtained in all the UAV passes were similar and in all of them the UAS passed through all collection zones at a distance suitable for WSN data collection.

Table 2 Distance of pass of the UAV trajectory through each WSN collection zone

WSN collection zones	WP1	WP4	WP7	WP10
Distance (m)	13.00	6.04	13.28	11.23

8 Conclusions

Several collision-free trajectory planners based on a genetic algorithm, RRT and RRT* planning algorithm to perform applications of WSN (Wireless Sensor Network) data collection with Unmanned Aerial Vehicles have been presented. These kind of stochastic methods are good candidates in these applications because they are efficient computationally and can be easily adapted to changes in the environmental conditions. These algorithms compute trajectories between WSN collection zones to ensure that the UAV passes through the WSN collection zones.

This chapter shows that RRT and RRT* planning algorithm are more suitable in these scenarios. They quickly compute a collision-free solution and then the solution can be refined by RRT* considering the remaining time. RRT* computes smoother trajectories than the ones obtained with RRT planning algorithm. Also RRT_i^* algorithm has been presented that includes several improvements from the original RRT* algorithm.

Several simulations have been performed to check the validity of the proposed algorithms. In particular, several scenarios have been solved by using the path planning algorithm based on RRT* and RRT_i^*. The execution time is in the order of seconds demonstrating the efficiency of the proposed algorithm. RRT_i^* and RRT^* algorithms are compared and the results show that RRT_i^* greatly outperforms RRT* in terms of the optimization of the initial solution.

Last but not least, a field experiment has also been done with the Megastar UAV in Utrera airfield to validate the approach in a real environment. The results of the experiment show that the UAV were able to collect data of four WSN nodes since the distance to the WSN nodes was lower than the maximum allowed deviation.

Future work will include performing a scalability analysis when applied to WSN deployed in a sparse area and with a higher number of UAVs in the system. Also, the use of parallelized planning algorithms such as *C-Forest* [40] seems convenient in order to take advantage of modern multi-core processors. To conclude, path pruning techniques such as the proposed in [41] can be used in a post-processing step to the paths obtained with RRT* method in order to get rid of unnecessary waypoints. However, this has to be studied carefully in multi-UAV applications, because it can produce unexpected collisions.

Acknowledgments This work was supported by the European Commission FP7 ICT Programme under the Project PLANET (European Commission FP7-257649-ICT-2009-5) and the RANCOM Project (P11-TIC-7066) funded by the Junta de Andalucía (Spain). David Alejo is granted with a FPU Spanish fellowship from the Ministerio de Educación, Cultura y Deporte (Spain).

References

1. Beard, R.W., McLain, T.W., Nelson, D.B., Kingston, D., Johanson, D.: Decentralized cooperative aerial surveillance using fixed-wing miniature uavs. Proc. IEEE **94**(7), 1306–1324 (2006)
2. Ollero, A.: Aerial robotics cooperative assembly system (arcas): first results. In: Aerial Physically Acting Robots (AIRPHARO) Workshop, IROS 2012, Vilamoura, Portugal, 7–12 Oct 2012
3. Merino, L., Caballero, F., Martinez de Dios, J.R., Maza, I., Ollero, A.: An unmanned aircraft system for automatic forest fire monitoring and measurement. J. Intell. Robot. Syst. **65**(1–4), 533–548 (2012)
4. Cobano, J.A., Martínez-de Dios, J.R., Conde, R., Sánchez-Matamoros, J.M., Ollero, A.: Data retrieving from heterogeneous wireless sensor network nodes using uavs. J. Intell. Robot. Syst. **60**(1), 133–151 (2010)
5. Gilmore, J.F.: Autonomous vehicle planning analysis methodology. In: AIAAA Guidance Navigation Control Conference, pp. 2000–4370 (1991)
6. Szczerba, R.J.: Threat netting for real-time, intelligent route planners. In: IEEE Symposium Information, Decision Control, pp. 377–382 (1999)
7. Hruschka, E.R., Campello, R.J.G.B., Freitas, A.A., De Carvalho, A.C.P.L.F.: A survey of evolutionary algorithms for clustering. Trans. Sys. Man Cyber Part C **39**(2), 133–155 (2009). http://dx.doi.org/10.1109/TSMCC.2008.2007252
8. Li, Y., Ang, K., Chong, G., Feng, W., Tan, K., Kashiwagi, H.: Cautocsdevolutionary search and optimisation enabled computer automated control system design. Int. J. Autom. Comput. **1**(1), 76–88 (2007)
9. Chang Wook Ahn, R.S.R.: A genetic algorithm for shortest path routing, problem and the sizing of populations. IEEE Trans. Evol. Comput. **6**(6), 566–579 (2012)
10. Cobano, J.A., Conde, R., Alejo, D., Ollero, A.: Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties. In: Proceedings of IEEE International Robotics and Automation (ICRA) Conference, pp. 4429–4434 (2011)
11. Lavalle, S.M.: Rapidly-exploring random trees: a new tool for path planning. In: Computer Science Department, Iowa State University. Technical Report TR 98–11 (1998)
12. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**, 1–76 (2011)
13. Pignaton, C.P.T.L.E., Morado, A.: Middleware support in unmanned aerial vehicles and wireless sensor networks for surveillance applications. Stud. Comput. Intell. **237**, 289–296 (2009)
14. Mitchell, H.L.P.D., Qiu, J., Grace, D.: Use of aerial platforms for energy efficient medium access control in wireless sensor networks. Comput. Commun. **33**(4), 500–512 (2010)
15. Teh, S.K., Mejias, L., Corke, P., Hu, W.: Experiments in integrating autonomous uninhabited aerial vehicles (uavs) and wireless sensor networks. In: 2008 Australasian Conference on Robotics and Automation (ACRA 08). The Australian Robotics and Automation Association Inc., Canberra (2008). http://eprints.qut.edu.au/15536/
16. Valente, J., Sanz, D., Barrientos, A., Cerro, J., Ribeiro, A., Rossi, C.: An air-ground wireless sensor network for crop monitoring. Sensors **11**(6), 6088–6108 (2011). http://www.mdpi.com/1424-8220/11/6/6088
17. Martinez-de Dios, J., Lferd, K., de San Bernab, A., Nez, G., Torres-Gonzlez, A., Ollero, A.: Cooperation between uas and wireless sensor networks for efficient data collection in large environments. J. Intell. Robot. Syst. **70**(1–4), 491–508 (2013). http://dx.doi.org/10.1007/s10846-012-9733-2
18. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. J. ACM **41**(4), 764–790 (1994)
19. Kuchar, J.K., Yang, L.C.: A review of conflict detection and resolution modeling methods. IEEE Trans. Intell. Transp. Syst. **1**, 179–189 (2000)
20. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. J. Intell. Robot. Syst. **57**(1–4), 65–100 (2010)

21. Prasanna, H.M., Ghosey, D., Bhat, M.S., Bhattacharyya, C., Umakant, J.: Interpolation-aware trajectory optimization for a hypersonic vehicle using nonlinear programming. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, USA, Aug 2005
22. Vela, A., Solak, S., Singhose, W., Clarke, J.-P.: A mixed integer program for flight-level assignment and speed control for conflict resolution. In: Proceedings of the 48th IEEE Conference on Decision and Control, pp. 5219–5226, Dec 2009
23. Pallottino, L., Feron, E., Bicchi, A.: Conflict resolution problems for air traffic management systems solved with mixed integer programming. Int. Transp. Syst. IEEE Trans. **3**(1), 3–11 (2002)
24. Bauso, D., Giarre, L., Pesenti, R.: Multiple uav cooperative path planning via neuro-dynamic programming. In: 43rd IEEE Conference on Decision and Control, pp. 1087–1092. Nassau, Bahamas, Dec 2004
25. Geiger, B.: Unmanned aerial vehicle trajectory planning with direct methods. Ph.D. dissertation, The Pennsylvania State University, Pennsylvania, USA (2009)
26. Vera, S., Cobano, J.A., Heredia, G., Ollero, A.: An hp-adaptative pseudospectral method for collision avoidance with multiple uavs in real-time applications. In: IEEE International Conference Robotics and Automation (ICRA), pp. 4717–4722. Hong-Kong, China, 31 May–7 June 2014
27. Spall, J.C.: Introduction to Stochastic Search and Optimization, 1st edn. Wiley, New York (2003)
28. Chakrabarty, A., Langelaan, J.W.: Flight path planning for uav atmospheric energy harvesting using heuristic search. In: AIAA Guidance, Navigation and Controls Conference, Toronto, Canada, Aug 2010
29. Lamont, G.B., Slear, J., Melendez, K.: Uav swarm mission planning and routing using multi-objective evolutionary algorithms. In: IEEE Symposium on Computational Intelligence in Multicriteria Decision Making, pp. 10–20. Honolulu, Hawai, USA, 1–5 April 2007
30. Conde, R., Alejo, D., Cobano, J.A., Viguria, A., Ollero, A.: Conflict detection and resolution method for cooperating unmanned aerial vehicles. J. Intell. Robot. Syst. **65**, 495–505 (2012). doi:10.1007/s10846-011-9564-6
31. Alejo, D., Cobano, J.A., Heredia, G., Ollero, A.: Collision-free 4d trajectory planning in unmanned aerial vehicles for assembly and structure construction. J. Intell. Robot. Syst. **73**, 783–795 (2014)
32. Durand, N., Alliot, J.: Ant colony optimization for air traffic conflict resolution. In: Proceedings of the Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009), Napa, CA, USA (2009)
33. Xue, E.M., y Atkins, M.: Terminal area trajectory optimization using simulated annealing. In: 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, Jan 2006
34. Lavalle, S.M., Kuffner, J.J., Jr: Rapidly-exploring random trees: progress and prospects. In: Algorithmic and Computational Robotics: New Directions, pp. 293–308 (2000)
35. Alejo, D., Conde, R., Cobano, J., Ollero, A.: Multi-UAV collision avoidance with separation assurance under uncertainties. In: IEEE International Conference on Mechatronics, ICM 2009, pp. 1–6, April (2009)
36. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006). http://planning.cs.uiuc.edu/
37. Akgun, B., Stilman, M.: Sampling heuristics for optimal motion planning in high dimensions. In: International Conferences on Intelligent Robots and Systems (IROS2011), pp. 2640–2645. San Francisco, USA, 25–30 Sept 2011
38. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. Syst. Sci. Cybern., IEEE Trans. **4**(2), 100–107 (1968)
39. Şucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. IEEE Robot. Autom. Mag. **19**(4), 72–82, Dec 2012. http://ompl.kavrakilab.org
40. Otte, M., Correll, N.: 1 c-forest: parallel shortest-path planning with super linear speedup. IEEE Robot **29**(3) 798–806 June 2013
41. Yang, K., Sukkarieh, S.: Planning continuous curvature paths for uavs amongst obstacles. In: Australasian Conference on Robotics Automation, Canberra, Australia (2008)