J Intell Robot Syst (2014) 73:783–795
DOI 10.1007/s10846-013-9948-x

Collision-Free 4D Trajectory Planning in Unmanned Aerial Vehicles for Assembly and Structure Construction

D. Alejo · J. A. Cobano · G. Heredia · A. Ollero

Received: 1 September 2013 / Accepted: 13 September 2013 / Published online: 1 October 2013

Abstract This paper presents a new system for assembly and structure construction with multiple Unmanned Aerial Vehicles (UAVs) which automatically identifies conflicts among them. The system proposes the most effective solution considering the available computation time. After detecting conflicts between UAVs, the system resolves them cooperatively using a collision-free 4D trajectory planning algorithm based on a simple one-at-a-time strategy to quickly compute a feasible but non-optimal initial solution and a stochastic optimization technique named Particle Swarm Optimization (PSO) to improve the initial solution. An anytime approach using PSO is applied. It yields trajectories whose quality improves when available computation time increases. Thus, the method could be applied in real-time depending on the available computation time. The method has been validated with simulations in scenarios with multiple UAVs in a common workspace and experiment in an indoor testbed.

Keywords Aerial robotics · Trajectory planning · Real-time applications

D. Alejo · J. A. Cobano (✉) · G. Heredia · A. Ollero
Robotics, Vision and Control Group Engineering School, University of Seville, 41092 Seville, Spain
e-mail: jacobano@cartuja.us.es

D. Alejo
e-mail: dalejo@us.es

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

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

1 Introduction

Multiple UAVs present important advantages to carry out different cooperative missions such as surveillance [1], mapping, data collection [2], fire detection and monitoring, etc. In these missions is important to maintain as much as possible the trajectories planned, and meet the Estimated Time of Arrival (ETA) of each UAV to avoid collisions between the UAVs and obstacles in the environment. Therefore a system to plan collision-free 4D trajectories is required to ensure the safety of the mission.

Cooperative missions are being performed in the ARCAS FP7 European Project [3]. This project is developing a cooperative free-flying robot system for assembly and structure construction (see Fig. 1). The ARCAS system use helicopters and quadrotors with multi-link manipulators for assembly tasks [4]. The aerial vehicles carry structure parts that will be assembled at the target destination. An important part in ARCAS is cooperative assembly planning and safe trajec-

Fig. 1 Assembly and structure construction in the ARCAS project

tory generation to perform the coordinated missions, assuring that neither the aerial vehicles nor the manipulators or the objects carried collide with each other. The work presented is related to this task.

General concepts and methods on path planning could be applied in order to solve the problem. A review of these methods with a comprehensive mathematical discussion is presented in [5]. In practical applications the most popular methods include potential fields [6], graph search like A* and D* [7] and Rapidly-exploring Random Trees (RRT) [8]. Other kind of methods have been proposed such as evolutionary techniques [9–11], particle swarm optimization [12] and multi-objective evolutionary algorithms [13].

The problem of trajectory planning is NP-hard [14, 15]. Sampling-based techniques, as opposed to combinatorial planning, are usually preferred in these NP-hard problems. These planning schemes are appropriated when the solution space is hard to model or unknown a priori because of its dynamic nature.

Furthermore, planning optimal collision-free trajectories for multiple UAV leads to optimization problems with multiple local minimum in most cases and, thus, local optimization methods as gradient-based techniques are not well suited to solve it. The application of evolutionary techniques or particle swarm optimization is an efficient and effective alternative for this problem, since they are global optimization methods.

The Conflict Detection and Resolution (CDR) problem has also been studied extensively and different types of techniques have been proposed [16]. One method to resolve conflicts is based on the speed assignment [17]. In this method the speed profile for all the aerial vehicles involved in a collision is computed in a centralized way to solve the conflict. A method based on mixed-integer linear programming (MILP) is presented in [18]. It resolves the conflict by changing speed to a large number of aerial vehicles subject to velocity change constraints, but some conflicts cannot be solved. Other method resolves pairwise conflicts [19] but do not consider more UAVs. More methods based on MILP to avoid collisions are presented in [20] and [21]. The method for multiple-UAV conflict avoidance proposed in [22] assumes that UAVs fly at constant altitude with varying velocities and that conflicts are resolved in the horizontal plane using heading change, velocity change, or a combination thereof. Methods based on Ant Colony Optimization (ACO) algorithms have also been proposed [23]. In [24], the application of a game theory approach to airborne conflict resolution is presented. These techniques present a disadvantage: they are not well suited for applications that require a high level of scalability for the application to many UAVs.

This paper presents a system for collision-free 4D trajectory planning. The system automatically identifies conflicts between multiple UAVs and proposes the most effective solution considering the available computation time. The conflicts are detected by using a detection algorithm based on axis-aligned minimum bounding box. The 4D trajectory planning algorithm is based on a stochastic global optimization technique named Particle Swarm Optimization (PSO) and resolves cooperatively the conflicts. Moreover, a simple one-at-a-time strategy is considered to quickly compute a feasible but non-optimal solution, which is taken as the initial point. PSO presents two advantages with respect to other evolutionary techniques [25, 26]: its low computational overheads and faster solution convergence. The approach to solve the conflicts is to add an intermediate waypoint to the initial trajectory and/or change the speed to meet the ETA. Preliminary results were presented in [27].

On the other hand, computing time of most optimization methods is not deterministic, and most published works either do not consider computing time or simply show that in average it is much less than the available time to get a solution, thus wasting available computing time. The proposed system adopts an anytime approach to resolve the conflicts. The aim is to provide a flyable solution at any time. This solution will be more or less close to the optimum depending on the available time. Thus, valid trajectories whose quality improves when available computation time increases are yielded.

The paper is organized into six sections. The formulation of the problem is presented in Section 2. The proposed system is described in Section 3. Sections 4 and 5 present the simulations and experiments performed, respectively. Finally the conclusions are detailed in Section 5.

2 Multi-UAV Trajectory Planning

The problem of collision-free 4D trajectory planning of multiple UAVs to perform the coordinated missions proposed by the ARCAS project is considered in this paper. First the initial spatial trajectories are computed to implement the assembly planning. The deviations of UAVs from these trajectories could be significant due to perturbations (i.e. wind). The system should compute collision-free 4D trajectories when a conflict is detected during the execution of the mission. A trajectory is defined by a sequence of waypoints and the ETA to the last waypoint. UAVs share a common workspace and the separation between them should be greater than a given safety distance. It is assumed that heading and speed changes are allowed to solve the conflicts and to meet the ETA. This latter is important to meet it in coordinated missions. The proposed system detects potential conflicts and computes cooperatively a collision-free 4D trajectory for each UAV. The information that the system needs in order to solve the problem is the following:

1) Initial spatial trajectory of each UAV.
2) Parameters of the model of each UAV
3) Location of each UAV
4) ETA of each UAV
5) The available computation time to solve the conflicts

The objective is to find collision-free 4D trajectories that minimize the probability of having a collision while minimizing the changes of waypoints and speed for each UAV. Moreover, the ETA will be met to perform the mission of the ARCAS project.

3 Description of the System

This section describes the blocks of the proposed system to resolve the detected conflicts (see Fig. 2).

3.1 Axis-Aligned Minimum Bounding Box for Detection

The detection algorithm is based on axis-aligned minimum bounding box presented in [26] with

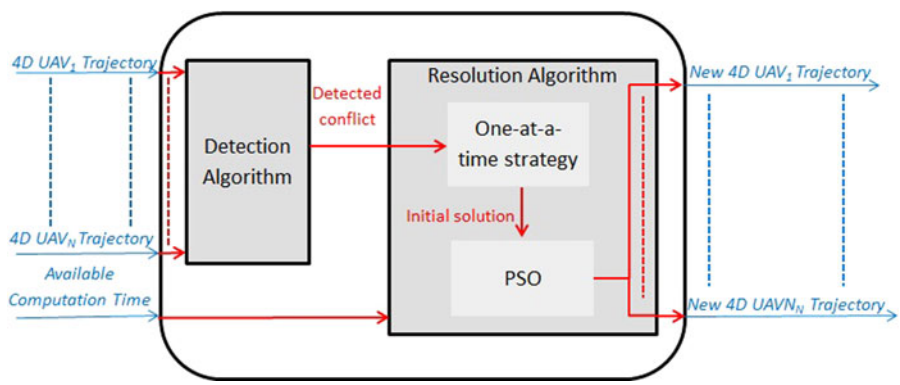

Fig. 2 Description of the proposed system to identify and resolve conflicts

some modifications to adapt to the characteristic of the ARCAS project. This technique presents as advantages the low execution time and the need of few parameters to describe the system.

The ARCAS project uses autonomous quadrotors or helicopters with a multi-link arm attached to the bottom of the aerial robot to carry out the manipulation and assembly tasks. These UAVs should grasp different structural parts and transport them to the corresponding assembly location. To avoid collisions between the UAVs (including the arms and the grasped objects), a security envelope is defined around each one. Each UAV security envelope is approximated by two boxes joined together: a horizontal box that covers the aerial robot and its rotors and a vertical box, which surrounds the arm and the transported object (see Fig. 3). Each box is defined by the intersection of three intervals, one by axis. The measurement of the horizontal box is related to the minimum horizontal separation between UAV considering the arm and the grasped object, and the vertical box is related to the vertical separation. Therefore, the minimum separation, S, between two UAVs is defined by the dimension of both joined boxes. A collision is detected when there is an overlapping between the intervals that define each box (see Fig. 3). Thus, the 3D problem is reduced to three problems of overlapping, one in each coordinate axis. Let us consider the intervals in one coordinate $A = [A_i, A_e]$ and $B = [B_i, B_e]$.

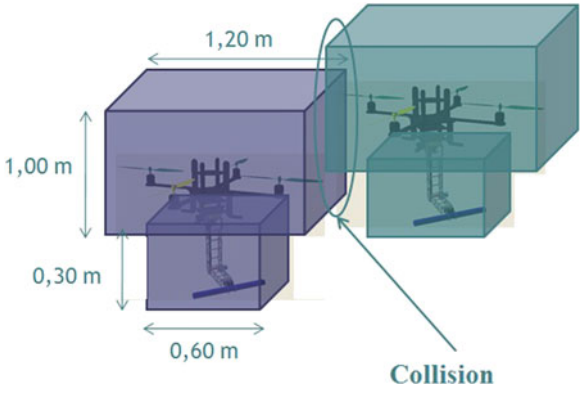

Fig. 3 Detection algorithm based on axis-aligned minimum bounding box: A and B overlap (*collision*)

The condition of overlapping for this coordinate is given by:

$$(A_e > B_i) \wedge (A_i < B_e) \tag{1}$$

3.2 4D Trajectory Planning Algorithm

The collision-free 4D trajectory planning algorithm is based on obtaining a feasible initial solution very fast, and then optimizing the solution using the PSO method, which is an heuristic global optimization algorithm. PSO is iterative, and the solution improves with time. Thus, it is guaranteed that a feasible solution is available at any time, and that this solution will improve its quality if there is more execution time available.

The initial solution can be obtained with a simple one-at-a-time strategy: when there is a possible conflict between several UAVs, one of them moves to its destination point while the others stay hovering at the initial position, then the next UAV goes to its target position, and so on. By moving only one UAV at a time, the conflict is avoided. On the other hand, the solution is far from the optimum since the total time will be much higher than the ETA. Other velocity planning methods to quickly compute initial solutions have been also implemented using the two velocities and the greedy approach described in [28].

There are some situations in which the conflict cannot be solved only by changing the speeds of the involved UAVs. This is the case when a frontal collision is detected, for example. In these cases, the path must be changed and the roundabout technique can be a good candidate [29]. The main idea is to make the involved aircrafts circle the conflict in the same direction. The radius of the circle should be long enough to ensure collision-free trajectories and to provide flyable trajectories.

The PSO algorithm was first proposed in [30]. It is developed from swarm intelligence and is based on the research of bird and fish flock movement behavior. It works by maintaining a swarm of particles that move around in the search-space influenced by both the improvements discovered by the other particles (social behavior) and the improvements made by the particle so far (greedy behavior). Its main advantages are its simplicity,

easy implementation and the existence of few parameters to tune when comparing with other evolutionary algorithms.

In this paper, we consider that the initial position is known and the final waypoint of each UAV should be the same as the one in the initial trajectory. Moreover, the ETA to the final waypoint should be met. Therefore, the goal of this algorithm is to obtain collision-free 4D trajectories by adding one intermediate waypoint in the trajectory of each UAV and changing the speed to meet the ETA while minimizing the following cost function:

$$J = \sum_{i=1}^{N} (L_i + k\Delta v_i^2) + \omega_c \tag{2}$$

where N is the total number of UAVs in the system, L_i is the total length of ith trajectory, Δv_i is the change of speed of each UAV to meet its ETA, k is a factor to convert to distance, and ω_c is the collision penalty that will be added if the new trajectories still lead to collisions in the system or if at least one unfeasible plan is generated. This function can be easily modified in order to take into account energy analysis and other operational costs. Note that the approach considered is centralized.

The implemented algorithm is based on [31]. Let S be the number of particles in the swarm, each particle is defined by a state vector x_i in the search-space and a velocity vector v_i. This state vector contains the information about the location of the intermediate waypoint and the velocity in the first sector of the trajectory of each UAV. Note that the speed in the second sector is calculated so the ETA to the final waypoint is the same as in the original trajectory.

In first place, the swarm is initialized by randomly assigning initial locations and velocities with an uniform distribution. Then a special particle containing the initial solution is added to ensure the existence of one conflict-free solution at any time.

Let p_i be the best known state vector of particle i and let g be the best known state vector of the entire swarm. These are recalculated whenever a new iteration is obtained.

Then the exploration loop is executed. In each iteration, both the state vector and the velocity of each particle are updated by applying the expressions indicated in steps 10 and 11 (see Algorithm 1).

The most important parameters in this formula are the social weight, ϕ_g, and the local weight, ϕ_p. ω is the inertia weight. r_g and r_p are vectors where each component is generated at randomly with an uniform $U(0, 1)$ distribution. Local and global best state vectors are also updated if necessary (steps 13–15).

The exploration loop can be finished by using many different termination criteria. Among these criteria a timeout condition and a convergence condition (most of the individuals lay in to a tight

Algorithm 1 Basic PSO algorithm

1. **for** Each particle **do**
2. Initialize each particle's state vector x_i with the desired probability function
3. Initialize particle best state vector $p_i \leftarrow x_i$
4. If $f(p_i) < f(g)$ update the swarm best state vector $g \leftarrow x_i$
5. Initialize each particle's velocity vector v_i. An uniform distribution is usually used.
6. **end for**
7. **repeat**
8. **for** Each particle **do**
9. Pick random numbers r_g r_p with $U(0, 1)$
10. Update the particle's velocity: $v_i \leftarrow \omega v_i + \phi_p r_p(p_i - x_i) + \phi_g r_g(g - x_i)$
11. Update the particle's state vector: $x_i \leftarrow x_i + v_i$
12. **if** $f(x_i) < f(p_i)$ **then**
13. Update the particle's best known state vector
14. **if** $f(x_i) < f(g)$ **then**
15. Update the swarm's best known state vector $g \leftarrow x_i$
16. **end if**
17. **end if**
18. **end for**
19. **until** A termination criterion is met

region of the search space) are the common approaches. In this paper, the algorithm concludes when the available computation time is reached.

The parameters ϕ_g and ϕ_p have been tuned by performing several tests with the same conditions and only changing one parameter at a time. These parameters are usually selected in the interval $[0, 1]$. In our case, the best values found were $\phi_g = 0.9$ and $\phi_p = 0.1$.

4 Simulations

A comprehensive set of simulations have been carried out to check the properties the proposed system. Also, a random generation process of the test considered has been performed to evaluate the system.

The definition of a metric plays an important role to evaluate the results. In this paper, a test set has been developed in a given scenario to validate the system and to provide a way to measure the properties of the system regarding time of execution, optimization and level of scalability with number of vehicles. The considered scenario has a base of 10×10 dimensional units. The design methodology is based on [26].

The test set consists of 40,000 different problems grouped by the number of vehicles involved, from 2 to 5, in subsets of 10,000 tests. This classification, using the number of vehicles, is useful to study the scalability characteristics of the method. The number of vehicles is from 2 to 5 because is the number used in the experiments of ARCAS project. Moreover, the tests have been performed in the same computer and under the same conditions.

The algorithms have been run in a PC with a 2GHz Dual Core processor and 2 GB of RAM. The operating system used was Kubuntu Linux with kernel 2.6.32. The code was written in C++ language and compiled with the gcc-4.4. 1.

The minimum horizontal and vertical separation between UAVs is shown in Fig. 3. These separation distances have to be calculated according to the dimension of each vehicle and their localization and control errors. The dimensions of the scenario are 10 m $\times$ 10 m. Two hundred tests have been performed for each subset. The number of intermediate waypoints, M, is set to 1. Therefore, each solution trajectory is composed of two segments. Each UAV should meet its ETA to perform the coordinated mission. The allowed speed for each UAV is between 0.3 m/s and 2 m/s, and $k = 5$. The speed in the first segment is chosen randomly and in the second one is computed to meet the ETA. If a particle does not meet the ETA, it is penalized.

Figure 4 shows the relation between the time of execution and the number of iterations per-

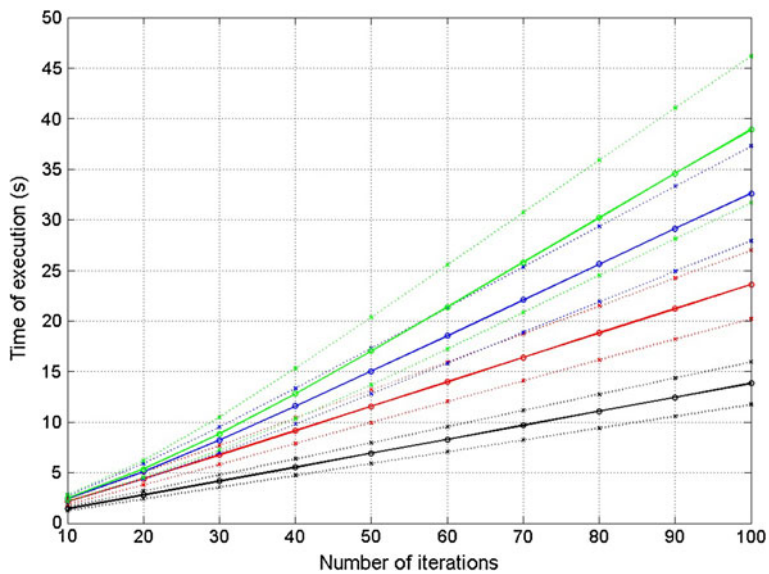

Fig. 4 Time of execution vs. number of iterations depending on the number of vehicles: two UAVs (*black*), three UAVs (*red*), four UAVs (*blue*) and five UAVs (*green*). *Solid line* shows the mean time and *dotted line* shows the standard deviation

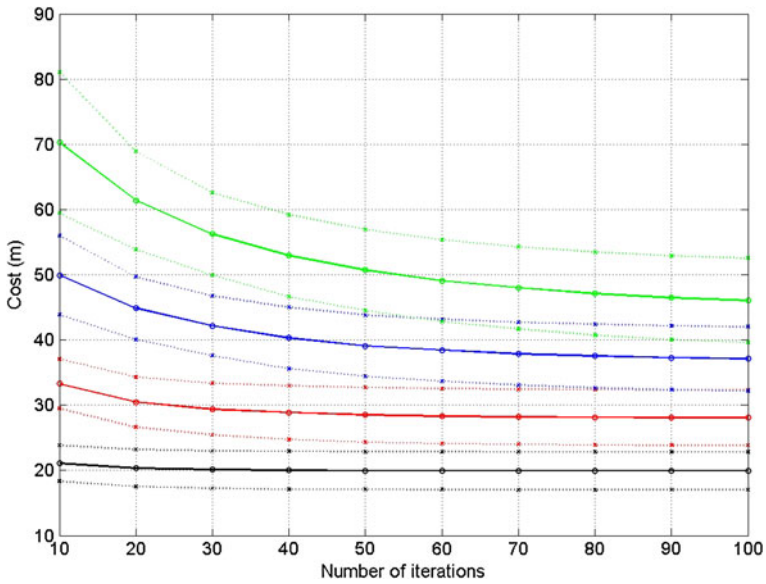

Fig. 5 Median of minimum cost throughout successive iterations depending on the number of vehicles: two UAVs (*black*), three UAVs (*red*), four UAVs (*blue*) and five UAVs (*green*). *Solid line* shows the mean time and *dotted line* shows the standard deviation

formed. Mean time and its standard deviation are shown each ten iterations. One hundred iterations are considered in each test. The dependence with the number of UAVs shows the scalability characteristics of the method.

Figure 5 shows the evolution of the mean cost and its standard deviation with the number of iterations considering different number of UAVs. The proposed system finds a better solution as time passes.

The speed should be changed in order to meet the ETA of each UAV. Figure 6 show the information on the speed of each UAV involved when five UAVs are considered. Each box of the figure depicts statistics of the two hundred tests performed for a given number of UAVs. The central mark is the median, the edges of each box are the 25th and 75th percentiles, and the whiskers extend to the extreme data points.

Note that the mean change of speed of each UAV is low and similar.

The median of the minimum costs computed in all the tests has been chosen as statistical indicator. This indicator indicates how much time it would cost to achieve a solution with certain level of optimality. This relates the cost in a given iteration to the obtained minimum cost in the corresponding problem.

Figure 7 shows a normalization of the cost against the number of iterations. A line that marks the required number of iterations to compute for a 90 % level of optimality is drawn. If the test

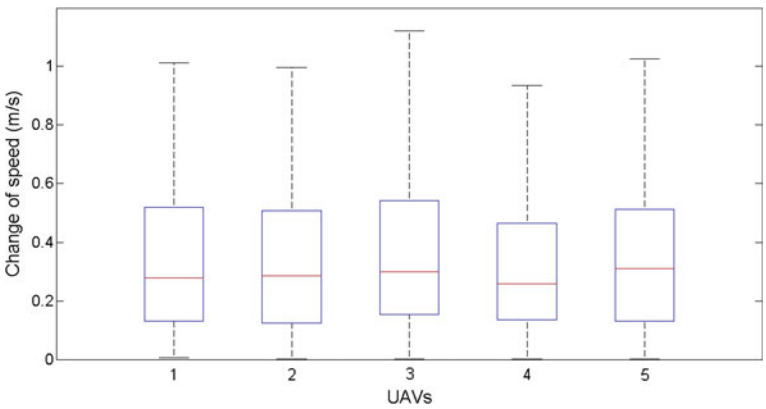

Fig. 6 Speeds with five UAVs by considering 200 tests

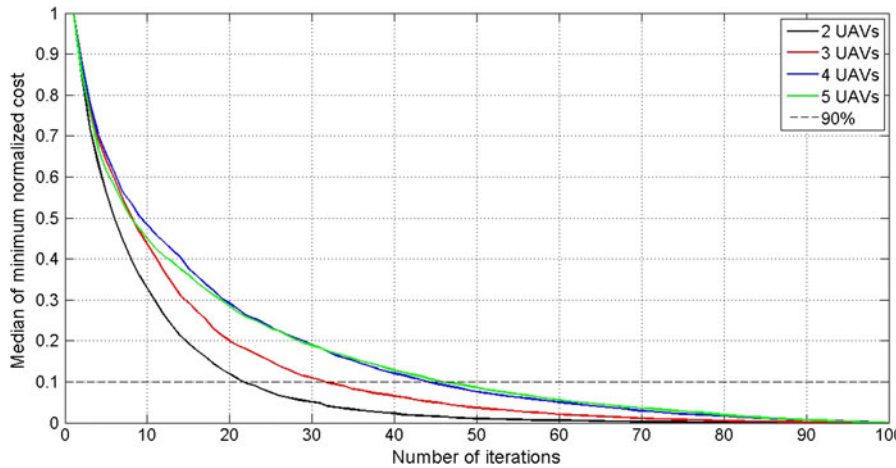

Fig. 7 Normalized cost throughout successive iterations. The line marks the 90 % optimality

set is executed in the same computer where the user has installed the proposed method, Fig. 4 will provide an estimation of the time needed for that number of iterations, and therefore, that level of optimality.

For the cost normalization, a linear transformation, $f(x) = ax + b$, is applied to the actual cost values to set them in the range [0,1]. Therefore a and b are chosen in such a way that the maximum cost equals to 1 and the minimum cost equals to 0. Therefore,

$$a = \frac{1}{Cost_{\max} - Cost_{\min}} \tag{3}$$

$$b = \frac{Cost_{\min}}{Cost_{\min} - Cost_{\max}} \tag{4}$$

Table 2 Evolution of the solution for three UAVs: time of execution and cost

Time (s)	Improvement of the cost (m)
0.010	60.47
0.135	47.56
0.353	40.01
0.804	38.34
1.246	36.23
2.157	33.27
3.021	32.15
4.403	30.01

Table 1 Evolution of the solution for two UAVs: time of execution and cost

Time (s)	Improvement of the cost (m)
0.009	38.01
0.251	23.44
0.416	23.01
0.826	22.24
1.439	21.08
1.622	20.98
1.987	20.78
2.798	20.36

Table 3 Evolution of the solution for four UAVs: time of execution and cost

Time (s)	Improvement of the cost (m)
0.014	79.89
0.198	66.73
0.390	63.71
0.771	61.12
1.190	58.26
2.410	49.94
3.780	47.35
5.132	44.85

Table 4 Evolution of the solution for five UAVs: time of execution and cost

Time (s)	Improvement of the cost (m)
0.016	115.34
0.215	103.15
0.413	99.97
0.680	96.02
1.284	87.54
2.495	70.27
3.327	67.70
5.364	61.40

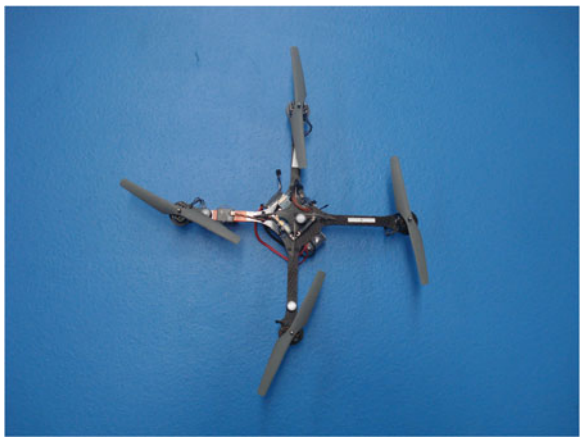

Fig. 8 Hummingbird quadrotor from Ascending technologies used in the experiments

Depending on the number of UAVs, a solution of great quality, 90 %, is computed between 20 or 47 iterations. This characteristic is important to apply this collision-free 4D trajectory planning algorithm in real-time applications. Moreover, the algorithm based on PSO presents better results than the algorithm based on genetic algorithms presented in [26].

The method can be applied in real-time applications by considering an anytime approach. It yields trajectories whose quality improves when available computation time increases. Tables 1, 2, 3, and 4 show how the quality of the solution improves as the time increases by using PSO. The advantage of this approach can be noted because a good quality of solution is achieved on two seconds. Obviously, the quality of solution improves as the time increases. Moreover, a first solution is computed in few milliseconds.

5 Experiments

Several experiments with the Hummingbird quadrotors (see Fig. 8) in an indoor testbed have been also performed to validate the method. The quadrotors used in these experiments are from Ascending Technologies and have 200gr payload and up to 20 min of autonomy.

The useful volume of the indoor testbed where tests can be conducted is a cube with a base of 10×10 m and 3 m height. The localization system is based on 20 VICON cameras that only needs

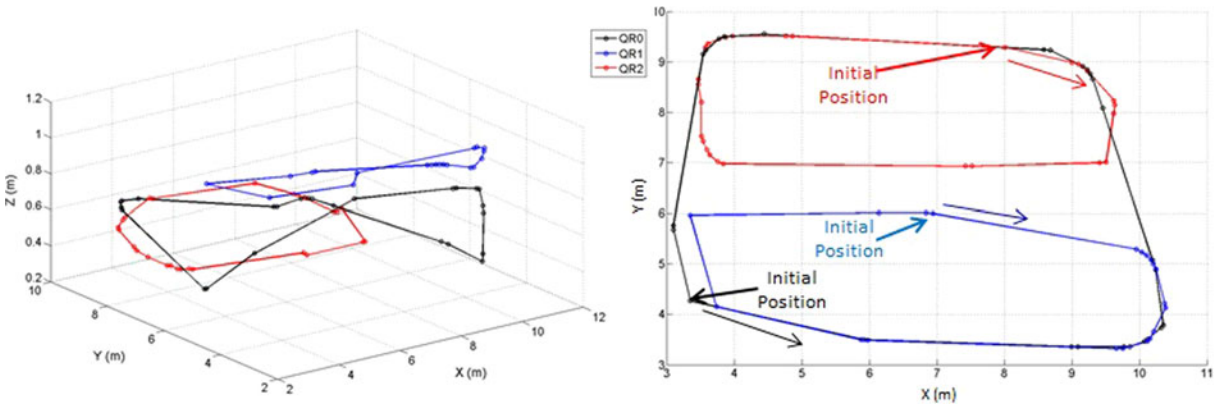

Fig. 9 Hummingbird quadrotor from Ascending technologies used in the experiments

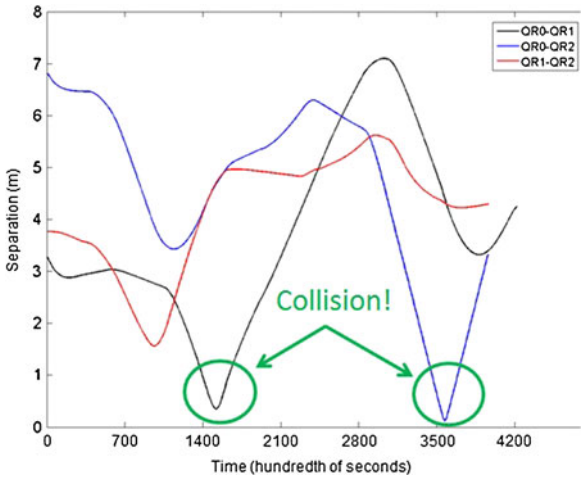

Fig. 10 Separation between the QR trajectories from the initial spatial trajectories

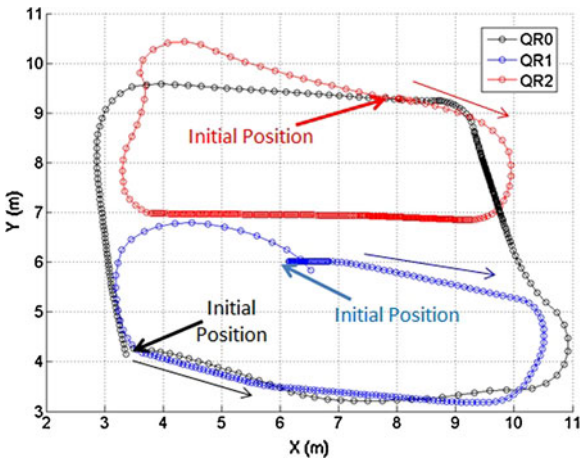

Fig. 11 Trajectories computed by the proposed system for each quadrotor in the experiment

the installation of passive markers on each of the aerial vehicles. This system is able to provide, in real-time, the position and attitude of each aerial vehicle with centimeter accuracy.

In this section, one experiment with three quadrotors is presented. Taking into account the characteristics of the scenarios of the ARCAS project, the minimum horizontal separation between quadrotors in XY plane was $S_{xy} = 1.0$ m, and the vertical separation in Z axis, $S_z = 0.45$ m.

Figure 9 shows the initial spatial trajectories of each quadrotor. The trajectories are simulated to check if collisions are detected. The quadrotors fly with constant velocity, $v = 0.5$ m/s. Figure 10 shows the separation between quadrotors. Two collisions are detected between QR0-QR1 and QR0-QR2. Therefore, the proposed system should compute collision-free trajectories.

Figure 11 shows the solution trajectories computed by the system. Different changes of velocity should be performed to avoid the collisions and meet the ETA. These changes of velocity can be observed in Fig. 11. The speed depends on the proximity of the waypoints (circles): closest waypoints indicate less speed and farther points indicate more speed.

Finally, the solution trajectories are flown. Figure 12 represent the real data from VICON system and Fig. 13 shows the horizontal and vertical separation between quadrotors. Note that the horizontal minimum separation QR0-QR1 is violated between the time instants 26.45 s and 28 s approximately but the vertical separation meets the vertical minimum separation in this interval.

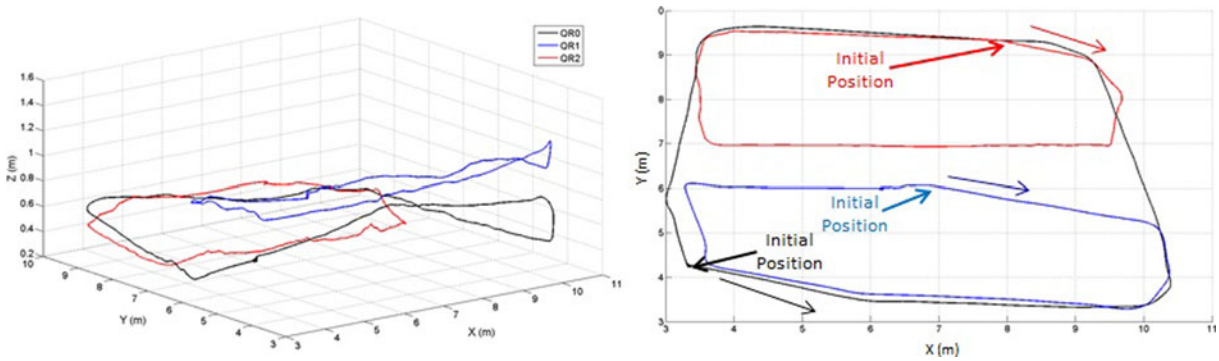

Fig. 12 Trajectories flown by each quadrotor in the experiment

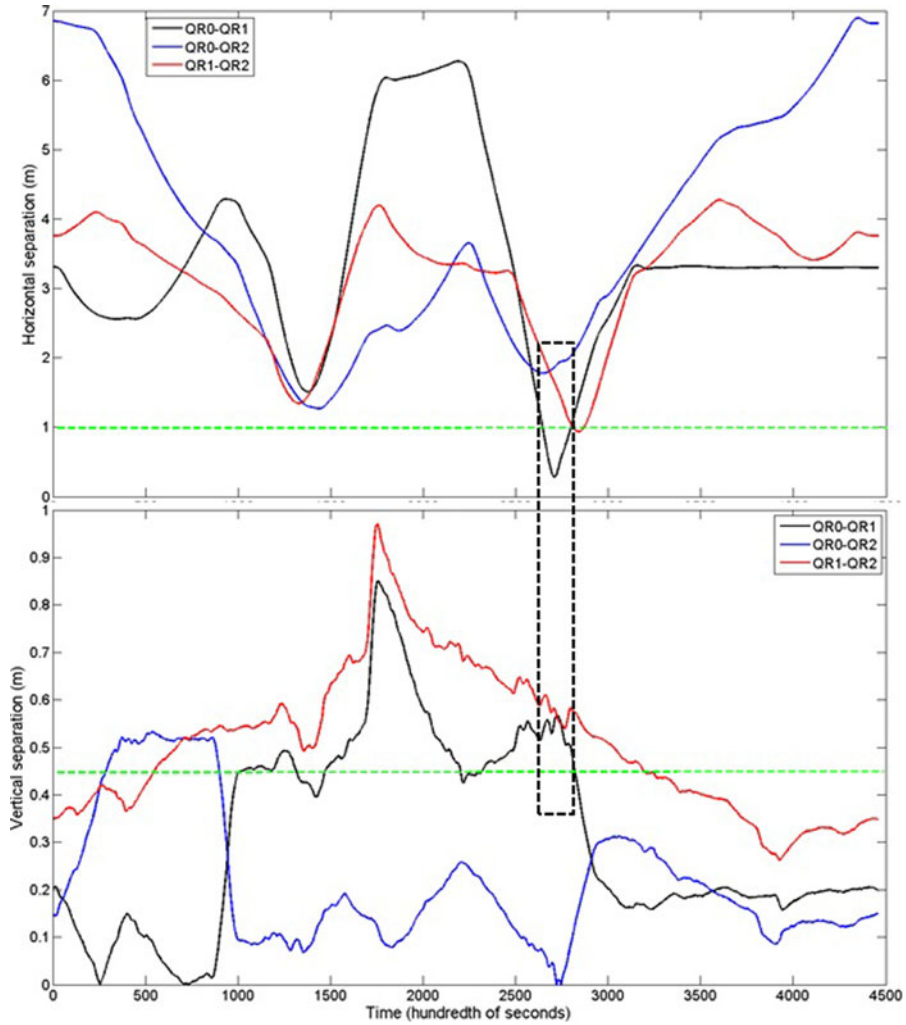

Fig. 13 Horizontaland vertical separation between the QR trajectories from the real data. *Green dashed line* shows the horizontal and vertical minimum separation

Therefore, the trajectories are safe and collisions are avoided.

6 Conclusions

In this paper, we have presented a new system for assembly and structure construction with multiple UAVs (Unmanned Aerial Vehicles) which automatically identifies and resolves conflicts among them. The planning algorithm is based on an anytime stochastic optimization approach, that is, the system proposes the most effective solution considering the available computation time. The proposed system detects conflicts in trajectories of multiple UAVs using an algorithm based on axis-aligned minimum bounding box, and resolves them cooperatively using a collision-free 4D trajectory planning algorithm based on a simple one-at-a-time strategy to quickly compute a feasible but non-optimal initial solution and Particle Swarm Optimization to improve incrementally the initial solution. The system provides solution 4D trajectories at any time, so it can be used with low available computation times, although a sub-optimal solution will be obtained. Thus, the proposed system is well suited to situations with variable available computation times, depending on the number of UAVs and the distance to potential conflicts.

An important requirement in coordinated missions of the ARCAS project is to meet the ETA.

The proposed system computes solution trajectories meeting the ETA of each quadrotor to perform the mission.

The system has been validated with many simulations performed in different scenarios and several studies to analyze the characteristics of the system. A comparative to genetic algorithms is mentioned. Moreover, real experiments have been carried out in an indoor testbed to show the performance of the system.

The main advantages of the proposed system with respect to the one presented in [12] are: it considers multiple vehicles in the space and could be applied in real-time.

Acknowledgements This work has been supported by the ARCAS Project, funded by the European Commission under the FP7 ICT Programme (ICT-2011-287617) and the CLEAR Project (DPI201 1-28937-C02-01), funded by the Ministerio de Ciencia e Innovacion of the Spanish Government. Also it has been partially funded by the Junta de Andalucia project P09-TEP- 5120.

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. 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). Online available: http://www.springerlink.com/index/10.1007/s10846-010-9414-y
3. 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
4. Jimenez-Cano, A.E., Martin, J., Heredia, G., Cano, R., Ollero, A.: Control of an aerial robot with multi-link arm for assembly tasks. In: IEEE Int. Conf. Robotics and Automation (ICRA). Karlsruhe, Germany, 6–10 May 2013
5. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006). Available at http://planning.cs.uiuc.edu/
6. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Rob. Res. **5**(1), 90–98 (1986). Online available: doi:10.1177/027836498600500106
7. Stentz, A., Mellon, I.C.: Optimal and efficient path planning for unknown and dynamic environments. Int. J. Robot. Autom. **10**, 89–100 (1993)
8. 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)
9. 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: Proc. IEEE Int Robotics and Automation (ICRA) Conf, pp. 4429–4434 (2011)
10. Vivona, R., Karr, D., Roscoe, D.: Pattern-based genetic algorithm for airborne conflict resolution. In: AIAA Guidance, Navigation and Control Conference and Exhibit. Keystone, Colorado (2006)
11. Durand, N., Alliot, J.: Optimization resolution of en route conflicts. In: First USA/Europe Air Traffic Management Research and Development Seminar (ATM1997). Saclay (France), 17–19 June 1997
12. Pontani, M., Conway, B.A.: Particle swarm optimization applied to space trajectories. J. Guidance Control Dyn. **33**, 1429–1441 (2010)
13. Pohl, A.J., Lamont, G.B.: Multi-objective UAV mission planning using evolutionary computation. In: Winter Simulation Conference, pp. 1268–1279 (2008). Available online, doi:10.1109/WSC.2008.4736199
14. Gilmore, J.F.: Autonomous vehicle planning analysis methodology. In: AIAAA Guidance Navigation Control Conference, pp. 2000–4370 (1991)
15. Szczerba, R.J.: Threat netting for real-time, intelligent route planners. In: IEEE Symp. Inf., Decis. Control, pp. 377–382 (1999)
16. Kuchar, J.K., Yang, L.C.: A review of conflict detection and resolution modeling methods. IEEE Trans. Intell. Transp. Syst. **1**, 179–189 (2000)
17. Cobano, J.A., Alejo, D., Ollero, A., Viguria, A.: Efficient conflict resolution method in air traffic management based on the speed assignment. In: Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems, ser. ATACCS '12, pp. 54–61. IRIT Press, Toulouse, France (2012). Online available: http://dl.acm.org/citation.cfm?id=2325676.2325684
18. 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, 2009 held jointly with the 2009 28th Chinese Control Conference CDC/CCC 2009, pp. 5219 –5226 (2009)
19. Erzberger, H.: Automated conflict resolution for air traffic control. In: Proceeding International Congress Aeronautical Sciences, pp. 179–189 (2006)
20. Richards, A., How, J.P.: Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: In Proc. ACC, pp. 1936–1941 (2002)
21. Pallottino, L., Feron, E., Bicchi, A.: Conflict resolution problems for air traffic management systems solved with mixed integer programming. IEEE Trans. Intell. Transp. Syst. **3**(1), 3–11 (2002)
22. Hwang, I., Tomlin, C.J.: Protocol-based conflict resolution for air traffic control. Air Traffic Control Quarterly **15**(1), (2007)

23. 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)
24. Masci, P., Tedeschi, A.: Modelling and evaluation of a game-theory approach for airborne conflict resolution in omnet++. In: Second International Conference on Dependability (2009)
25. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Nat. Comput Springer **1**, 235–306 (2002)
26. 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
27. Alejo, D., Cobano, J.A., Heredia, G., Ollero, A.: Particle swarm optimization for collision-free 4d trajectory planning in unmanned aerial vehicles. In: Proceedings of the 1st International Conference on Application and Theory of Automation in Command and Control Systems, pp. 298–307. Atlanta, USA, 28–31 May 2013
28. Alejo, D., Díaz-Báñez, J.M., Cobano, J.A., Pérez-Lantero, P., Ollero, A.: The velocity assignment problem for conflict resolution with multiple aerial vehicles sharing airspace. J. Intell. Robot. Syst. **69**(1–4), 331–346 (2013)
29. Pallotino, L., Scordio, V.G., Bicchi, A., Frazzoli, E.: Decentralized cooperative policy for conflict resolution in multi-vehicle systems. IEEE Trans. Robot. **23**(6), 1170–1183 (2007)
30. Kennedy, J., Eberhat, R.: Particle swarm optimization. IEEE Int. Conf. Neural Netw. **4**, 1942–1948 (1995)
31. Pedersen, M.E.H.: Good parameters for particle swarm optimization. Hvass Laboratories, Technical Report no. HL1001 (2010)