JOURNAL OF AIRCRAFT
Vol. 46, No. 3, May–June 2009

Three-Dimensional Obstacle Avoidance Strategies for Uninhabited Aerial Systems Mission Planning and Replanning

Franco Persiani,* Francesca De Crescenzio,† Giovanni Miranda,‡ and Tiziano Bombardi§
University of Bologna, 47100 Forlì, Italy
and
Marco Fabbri¶ and Fabio Boscolo**
Alenia Aeronautica SpA, 10146 Torino, Italy

DOI: 10.2514/1.37244

The ability to perform autonomous mission planning is considered one of the key enabling technologies for uninhabited aerial systems. Subsequently, a big effort is made in the development of algorithms capable of computing safe and efficient routes in terms of distance, time, and fuel. In this paper an innovative 3-D planning algorithm is presented. The algorithm is based on considering the uninhabited aerial systems representation of real world systems as objects moving in a virtual environment (terrain, obstacles, and no fly zones), which replicates the airspace. Original obstacle avoidance strategies have been conceived to generate mission plans that are consistent with flight rules and with the vehicle performance constraints. Simulation test results show that efficient routes are computed in a few seconds.

I. Introduction

THE interest of both the academic and industrial worlds for the uninhabited aerial systems (UASs) has been growing for the last 10 years because of the advantages that they bring compared with the traditional habited or manned vehicles. The physical separation between the pilot and the vehicle allows the use of UASs in missions categorized as "the dull, the dirty, and the dangerous" [1]. Moreover, UASs save the space and weight traditionally allocated to the crew and relevant ancillary systems, which can be used for greater payload and more fuel. Further advantages concern the reduction of costs associated with development, use and maintenance, pilot/operator training, and the loss of human lives. It is believed that a country could receive benefits in economic, political, and social fields from a growing use of the UASs both in military and civilian missions [1–3].

Although the range of mission types in which the UASs could be used is wide and the estimated benefits are significant, such vehicles are not yet fully reliable and the number of incidents and accidents is still high. Several researchers have demonstrated that the human factor implications and the poor vehicle decisional capacity are among the most important causes of mishaps [4–7]. Currently, UASs which are operating in military missions mostly get direct or indirect input commands from the human operators and execute those commands by following reference flight parameters. Assuming a higher level of decisional autonomy, the UAS should be able to deal with more complex and evolving scenarios while providing the operator with adequate situation awareness. An increase in UAS intelligence could lead to a further extension of its capabilities, with a consequent reduction of workload required of the human during all phases of a mission. To achieve such autonomy, the vehicle should be capable of comprehending the operator commands and computing a mission plan that is consistent with its capabilities [8–18]. This would also permit the UAS to operate in coordination with other unmanned vehicles or other actors engaged in the same scenario.

The ability to autonomously perform mission planning is a key factor for the development of unmanned aerial vehicles (UAVs). Several methods have been proposed and are mainly based on the construction of visibility graphs [14,15,19,20]. Following this approach the nodes of the graph are candidate path points that the vehicle will fly through, and each arc represents an approximate path between them. To build the visibility graph, the set of nodes and the arcs that join them without intersecting obstacles have to be determined. Among such methods, Kuwata and How [15] present a method used to calculate the path that the vehicle must fly when commanded to fly close to the surface of a 3-D terrain of a complex environment, represented by cube obstacles, to avoid threats as radars. The method consists of building the visibility graph, using Dijkstra's algorithm to find the approximate shortest paths from each node to the goal, and, finally, implementing a mixed integer linear programming receding horizon control to calculate the final trajectory.

The research presented in this work focuses upon the mission planning capability of the vehicle. We describe an alternative planning algorithm that allows the vehicle to autonomously and rapidly calculate 3-D routes, following a heuristic approach. The calculus of the route is based both on obstacle avoidance strategies and on the specific vehicle performance constraints. Moreover, such routes are efficient in terms of distance traveled, time required, or fuel consumed. The algorithm is developed and verified in a virtual world that replicates the real world in which the vehicle is flying. The elements of the real world, such as the terrain and the obstacles, are represented as mesh-based models. Such an algorithm is also based on a graph, but aims at increasing the portion of space explored identifying several nodes in the 3-D space with an iterative approach.

The level of automation to be implemented to cater for the use of the algorithm on the UAS is first discussed. The structure of the virtual scenario is then described. Subsequently, a description of the planning algorithm and of the simulation test results concerning both the computation time and the effectiveness and reliability of the algorithm are provided. Finally, conclusions and future works are discussed.

Presented as Paper 8962 at the 8th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), Anchorage, Alaska, 14–19 September 2008; received 21 February 2008; revision received 13 August 2008; accepted for publication 18 August 2008.

*Senior Professor, II Faculty of Engineering; franco.persiani@unibo.it.
†Assistant Professor, II Faculty of Engineering; francesca.decrescenzio@unibo.it.
‡Ph.D. Student, II Faculty of Engineering; giovanni.miranda@unibo.it. Member AIAA.
§Computer Programmer, II Faculty of Engineering; tiziano.bombardi@unibo.it.
¶Engineer, Simulation and Avionics Integration; mfabbri@aeronautica.alenia.it.
**Engineer, Simulation and Avionics Integration; fboscolo@aeronautica.alenia.it.

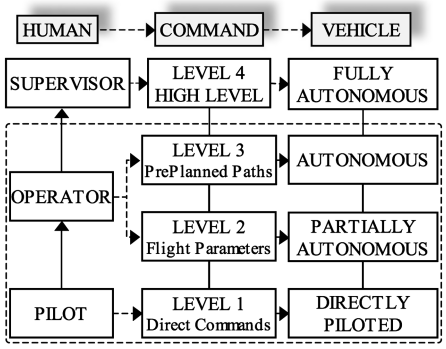


Fig. 1 UAS levels of automation.

II. System Architecture

UASs can be remotely operated or commanded by humans in different ways depending on the level of decisional autonomy implemented on the vehicle. As depicted in Fig. 1 at least four levels of automation can be defined for these systems. At the bottom level, the operator remotely pilots the vehicle by means of remote controls (level 1). Moving upward in the diagram, the level of automation increases and the operator is asked to interact with the vehicle by following different approaches. At level 2, the operator sets flight parameters (altitude, speed, heading) to be achieved and maintained during the flight and, at level 3, the operator plans a flight route that will be a reference for the vehicle during the mission. In most cases, the level of automation advocated for UASs is based on the autonomous execution of mission segments (level 4). At this level, the operator supervises the mission execution setting the goals that should be achieved by the vehicle.

Several considerations have been made for each level mentioned. Low levels of automation (levels 1 and 2) require the operator to continuously share his limited cognitive resources between sending the commands to the vehicle and analyzing the information coming from the payload [4–7,21]. Furthermore, due to the sensory isolation, the operator is more prone to errors caused by spatial disorientation (potentially leading to the failure of the mission and the loss of the vehicle). At level 3 we suppose that the human acts as a passive monitor of the preplanned mission. In this case, the human is not actively involved in the operation and could have low levels of situation awareness [22].

Level 4 is the supervisory control level and can be defined by observing how a supervisor of people acts with his subordinates. In the field of the human relationship, a supervisor sends directives to his subordinates that understand and translate them into detailed actions, gather the information about the results, and inform him. Thus, he can infer the state of the system and make a decision for further actions [23]. Concerning UASs the human operator is the supervisor whereas the intelligent vehicle can be considered to be his subordinate.

Figure 2 shows the system architecture defined to implement the supervisory control concept in this paper. Specifically, the system model is composed of two main modules: the human and the vehicle.

The human module has two macroblocks as follows:

1) Information perception and processing: the cognitive task that the human is required to perform with his deductive and reasoning skills/capabilities.

2) High level commands: the commanding task that depends on the man–machine interface features and on the capabilities of the intelligent planning algorithm.

The vehicle has three macroblocks as follows:

1) Command translation module:

a) Planning algorithm: translates the operator commands into routes that the vehicle has to fly to execute the commands.

b) Payload manager: translates the operator commands into payload operations.

2) Flight module: receives the routes and is responsible for the trajectory and stability of the vehicle.

3) Payload: "sees and senses" the environment and communicates the gathered information to the command translation module and to the human.

The information flow generated in the UAS between the human, the vehicle, and the environment is described next. The human perceives the information about the vehicle state and the environment features, processes them, and, based on the mission objective, acts by sending high level commands, such as goals, constraints, if-then rules, and specific plans. Such commands should derive from natural human interaction, that is, from simple and instinctive human gestures and vocal utterances. In this way, the human may spend few resources, in terms of time and cognitive and physical effort, to command the vehicle, and focus his attention mainly on the management of the mission and analysis of the information coming from onboard systems.

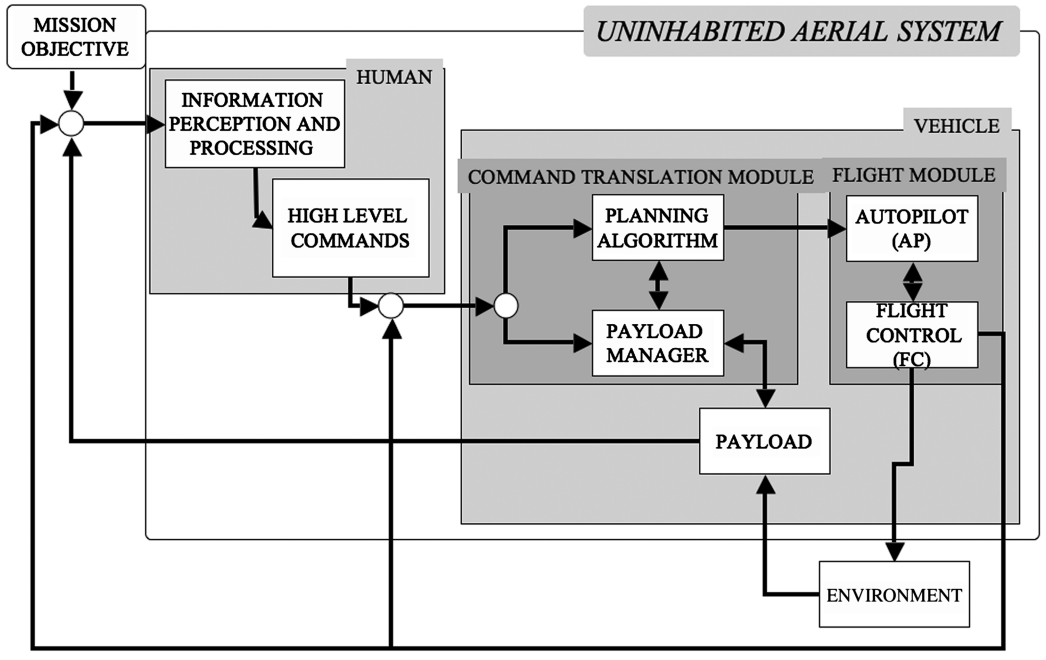


Fig. 2 The information flow in the UAS.

As previously noted, the vehicle becomes the human subordinate and can work alone in some phases of the mission. Hence, the vehicle

1) receives the commands and translates them into detailed actions as path planning actions, payload management actions, and, at a lower level, operations of the flight control system during the flight;

2) gathers information about its functioning and from the environment, processes the collected data, decides whether a replanning action is required by the contingent situation and, eventually, replans the mission;

3) informs the human about its own state and that of the environment.

In this paper we assume that

1) the human is not required to directly control the vehicle;

2) the vehicle translates the operator commands into routes to fly to perform the mission;

3) the vehicle is warned of new obstacles by a change in the scenario database represented by a digital world that replicates the real one.

III. Environment Modeling

A computer generated replica of the environment, at different levels of detail, is needed as input to the planning algorithm. Hence, maps, charts, and a digital database of the essential features of the scenery in which the mission is being executed have to be provided. For the scope of the activity relevant to this paper, the digital world has been constructed modeling the features of the real world as terrain, obstacles, tall buildings, and airspace boundaries. Such features are represented by three-dimensional mesh-based models. Different primitives represent the digital world: spheres, extruded polygons, and polygon meshes.

The terrain surface has been modeled as a quadrangular mesh based on digital elevation maps data (Fig. 3a). The other features have been modeled using triangular meshes. In particular, the mesh models that represent tall buildings and airspace boundaries, identified as no fly zones, where the UAS is not allowed to fly, are obtained extruding plane profiles on the ground in the vertical direction. Such models have the vertex number and their coordinates, latitude, longitude, and altitude as input parameters. The polyhedron height can be the actual height of the building (if known) or, in the case of no fly zones, it is virtually infinite because it is greater than the vehicle flight ceiling (Fig. 3b). Hazardous obstacles have been modeled in the form of semispheres. The mesh models that represent such semispheres have the radius and center latitude, longitude, and altitude of the semispheres as parameters (Fig. 3c).

IV. High Level Commands

The scenarios in which the UAS missions are executed are dynamic and can change continuously. This could drive the need to modify part of the mission or, in the worst case, the initial mission objectives. Based on the particular situation, the operator could command the vehicle to fly toward a destination point, to monitor a set of objectives or a specific rectangular area, or to survey a target of interest. The destination point, the objectives, and the target are defined by their coordinates (latitude, longitude, altitude) whereas the area is defined by the coordinates of two opposite vertices.

For example, the vehicle may be commanded to survey a new target in a military mission, or commanded to monitor an area in a civilian mission. In this paper, a high level command is defined as the instruction command that the human sends to the vehicle to execute a mission or part of it.

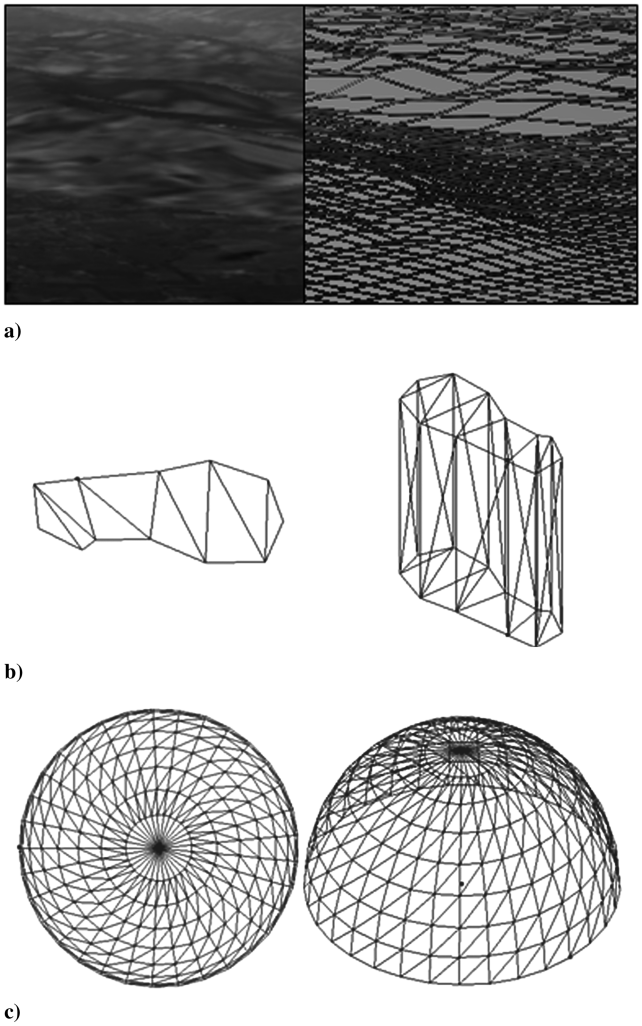


Fig. 3 The digital world.

High level commands are modeled by referring to a syntax that, in concept, is similar to the one used by the operator to command the vehicle. Generally, the command is composed of two blocks defining "what to do" and "where to do it." Furthermore, the operator may define an index of priority such as the mission time, the distance to be traveled, or the fuel to be consumed. In this way the syntax of the high level command can be

"*MISSION TASK to TARGET at time/distance/fuel PRIORITY*."

Deploying this syntax we derive the instructions as seen in Table 1.

Some examples of high level command may be "Fly to RESPU at time priority" (waypoint RESPU: 43° 41′ *1N- 11°* 33′ 36*E*), "Survey Forlì downtown at fuel priority," Monitor WP_1, WP_2, WP_3 at distance priority.

When the human commands the vehicle to monitor an area or to survey a target, he has to set the payload parameters and the altitude of the maneuver. The choice of the altitude may depend upon the onboard payload, the meteorological conditions, the position of the target on the ground, and on Air Traffic Control constraints.

Table 1 The high level commands

1) Mission task	2) Target type	3) Priority	4) Parameters
a) Fly to	a) Destination waypoint.	a) Distance	
b) Monitor	b) $WP_1, \ldots, WP_N$: set of sparse objectives	b) Fuel	Monitor or survey altitude
c) Survey	c) Area on the Earth surface	c) Time	
	d) Target		

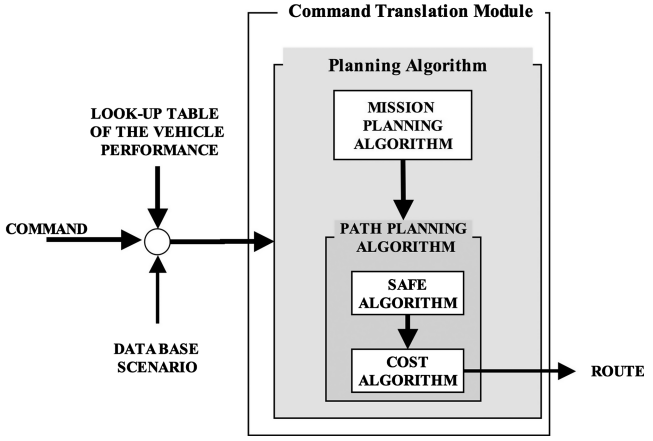


Fig. 4 The planning algorithm.

V. Planning Algorithm

The planning algorithm computes the mission routes based on the command set by the operator, the vehicle and environmental state, and look-up tables of the vehicle performance, which provide performance characteristics of the specific vehicle precomputed by means of computer simulations. In Fig. 4, the architecture of the planning algorithm is shown.

The planning algorithm is composed of two subalgorithms as follows:

1) Mission planning algorithm: It runs if a monitor or a survey task is commanded. It identifies a series of ordered point locations called primary mission waypoints, which will be included in the route to accomplish the mission objectives.

2) Path planning algorithm: If a monitor or a survey task is commanded, it runs assuming subsequent couples of primary mission waypoints (PMWs) as input data. If a fly to task is commanded, input data consist of the current vehicle position, as the starting point, and the commanded position as the destination point. The algorithm calculates the path between each couple of primary mission waypoints, called a macroleg, generating an appropriate sequence of route waypoints and corresponding speed data in a 3-D space. Each macroleg can be composed of climb, cruise, and descent phases depending on the relative altitude and position of the primary points and the obstacles.

The global route is the connected sequence of the macrolegs. Such a route is safe and efficient and is provided as a set of waypoints to pass through at a specified speed.

The path planning is composed of two subalgorithms as follows:

1) The *safe* algorithm that calculates a set of merely geometric safe paths from the first to the second primary waypoint for each couple of waypoints.

2) The *cost* algorithm that manipulates the safe paths and generates the path that fulfils the vehicle performance and mission priority.

An important feature of the algorithm is its adaptability to different vehicles. To calculate the route for a specific vehicle one has only to use the look-up table of that vehicle. In the following paragraphs, the application of the planning algorithm to each high level command previously defined is described in relationship to the blocks which compose it.

A. Path Planning Algorithm

As mentioned previously, the calculus of the path between two point locations in the 3-D space is executed in two steps: the safe planning algorithm and the cost planning algorithm.

1) The safe planning algorithm: The safe algorithm restricts the 3-D flight space to a discrete number of planes, and then calculates a set of safe paths that allow the vehicle to fly toward a destination point, avoiding the obstacle profiles on each plane. Then, each geometric safe path is sent to the cost algorithm as a list of waypoints characterized by three coordinates: latitude, longitude, and altitude.

In Fig. 5 the safe algorithm procedure is represented in a simple scenario. The obstacles are represented in the form of semispheres on the ground. The actions made by the safe algorithm are as follows:

1) Generation of a sheaf of n planes obtained by discrete angle rotations of a reference plane about the line connecting the vehicle position (START) and the destination point (END). The reference plane is the plane that contains the START point and the END and is perpendicular to the vertical plane (Fig. 5). The angle rotation is an input parameter to be set before running the algorithm. Thus, by assuming the reference plane as the plane corresponding to $\alpha = 0$, and setting the angle rotation value to $\Delta\alpha$, the sheaf will be composed

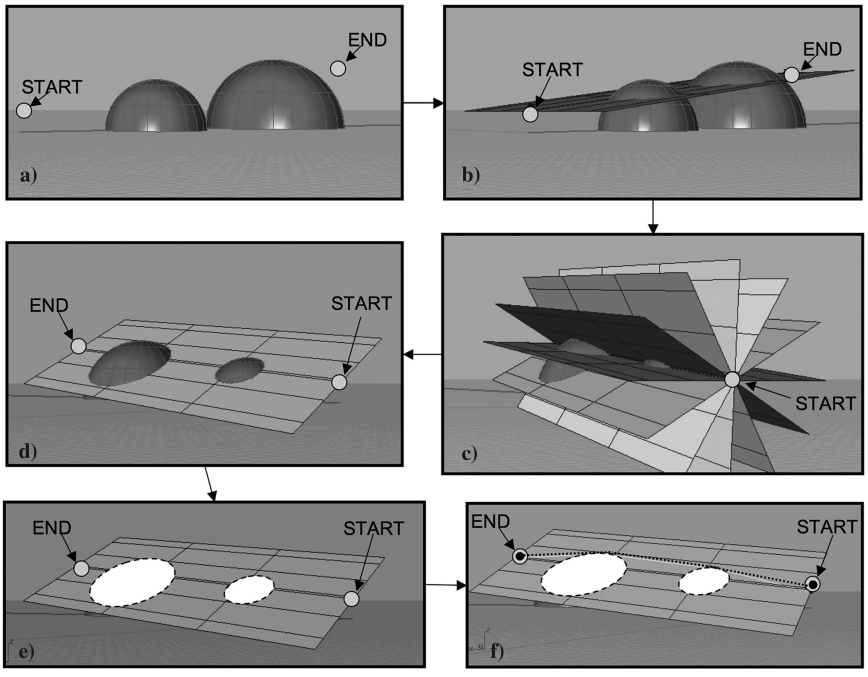


Fig. 5 a) Initial scenario; b) reference plane; c) sheaf of *n* planes; d) selection of a single plane; e) section profiles computation; and f) geometric safe path.

of n planes where $n = 180/\Delta\alpha$, the rotation value plane $i = \Delta\alpha^* i$, and $i = 0, \ldots, n$. Setting $\Delta\alpha = 30$ deg, the flight space will be sectioned by means of a sheaf of six planes (Fig. 5c).

2) Computation of the intersection profiles between each plane and the obstacles and no fly zone volumes (Figs. 5d and 5e).

3) Determination of geometric safe paths that avoid these profiles (Fig. 5f).

In such a way, the 3-D problem of the calculus of paths in a three-dimensional space is reduced to a 2-D problem following obstacle avoidance strategies based on turning around or flying over them, depending on the inclination of the single planes. Routes are computed within a network made of segments that do not intersect the obstacles. Such segments are iteratively selected connecting the START point, the END point, and obstacle polygon vertices as described in Fig. 6.

As an example, Fig. 7 shows the partial generation of the paths in a generic plane limited to the solution obtained by turning around the left side of the first obstacle.

1) The section profiles of each obstacle are included in a polygonal area, which is described by an ordered list of vertices. The algorithm calculates the straight line joining the START and the END points on the plane as a first solution. In this case the path is the sequence {START, END}. Then, it determines if the END is visible from the START, that is, the algorithm determines if the straight line intersects any edge of the polygons. In the case shown in the figure, the algorithm finds four ordered intersection points: I_1, I_2, I_3, I_4 (Figs. 7a–7d).

2) Because such intersections occur, the algorithm searches for alternative paths. At first, it identifies the extreme points of the first polygon edge intersected. In Fig. 7e such vertices are, respectively P_1 and P_A, because I_1 is a vertex itself. The path branches off in {START, P_1, END} and {START, P_A, END}. The algorithm checks if P_1 and P_A are visible from START. If they are visible the algorithm searches for further visible points in both directions on the same polygon. Turning clockwise around the obstacle, the vertices P'_1 is the last visible vertex on the current obstacle from START. The partial paths at this stage are {START, P'_1, END} and {START, P_A, END}, obtained in the same way (Fig. 7g).

3) The algorithm runs again twice with these new inputs. For the path to the left side of the first obstacle, the straight line joining P'_1 and END is analyzed to determine whether END is visible from P'_1 (Fig. 7g). In this case, the END is not visible because the line intersects both the first and the second obstacles. The algorithm identifies three intersection points: I_1, on the first obstacle, and I_2 and I_3 on the second one (Fig. 7h). The algorithm joins the last waypoint of the partial path before {END, P'_1}, with the vertices on the left and right of the first intersection point, I_1, obtaining the segments {P'_1, P_2} and {P'_1, P'_2} (Fig. 7i). Again, it determines if such segments intersect some polygons. If there are some intersections, the algorithm searches for visible vertices on the obstacle by turning clockwise and counterclockwise around the obstacle. Following this criteria and remaining on the left side of the branch, a new partial path is {START, P'_1, P_2, END}. This procedure is repeated until the algorithm quits or the last point inserted in the array is visible to END. The two routine branches are {START, P'_1, P_2, P'_3, P_4, END}, which turns around the first and the second obstacle clockwise, and {START, P'_1, P_2, P''_3, P'_4, END}, which turns around the first obstacle clockwise and around the second one counterclockwise (Figs. 7k–7p). The dashed lines in the figure are discarded segments.

4) The algorithm repeats the same procedure to find a set of candidate safe paths in the selected plane of the sheaf.

2) The cost planning algorithm: The cost algorithm receives the geometric safe paths from the safe algorithm, analyzes them to verify that they are compatible with the vehicle performance, displaces the waypoints (position and speed) to meet the efficiency criteria

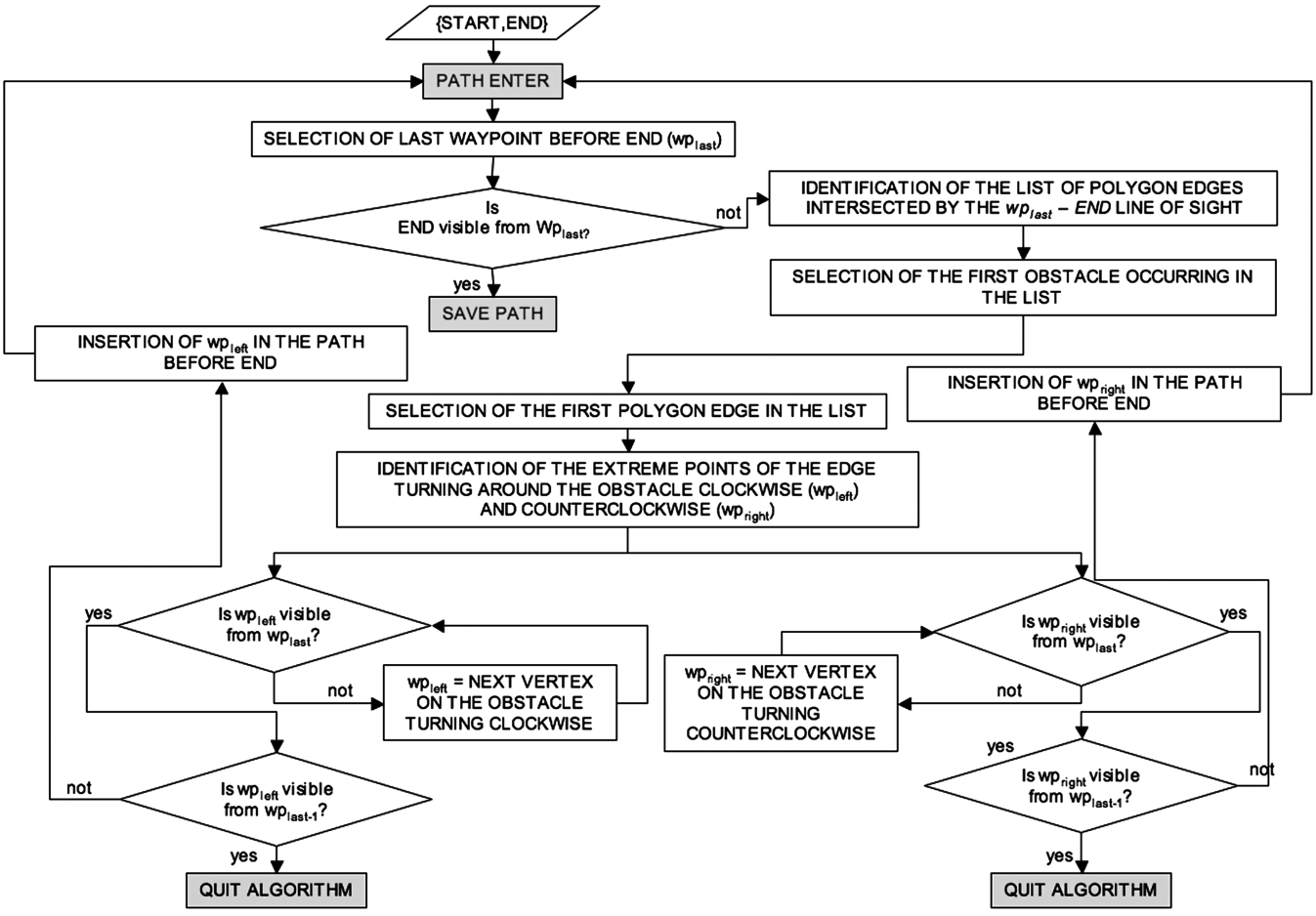


Fig. 6 Safe algorithm flow chart.

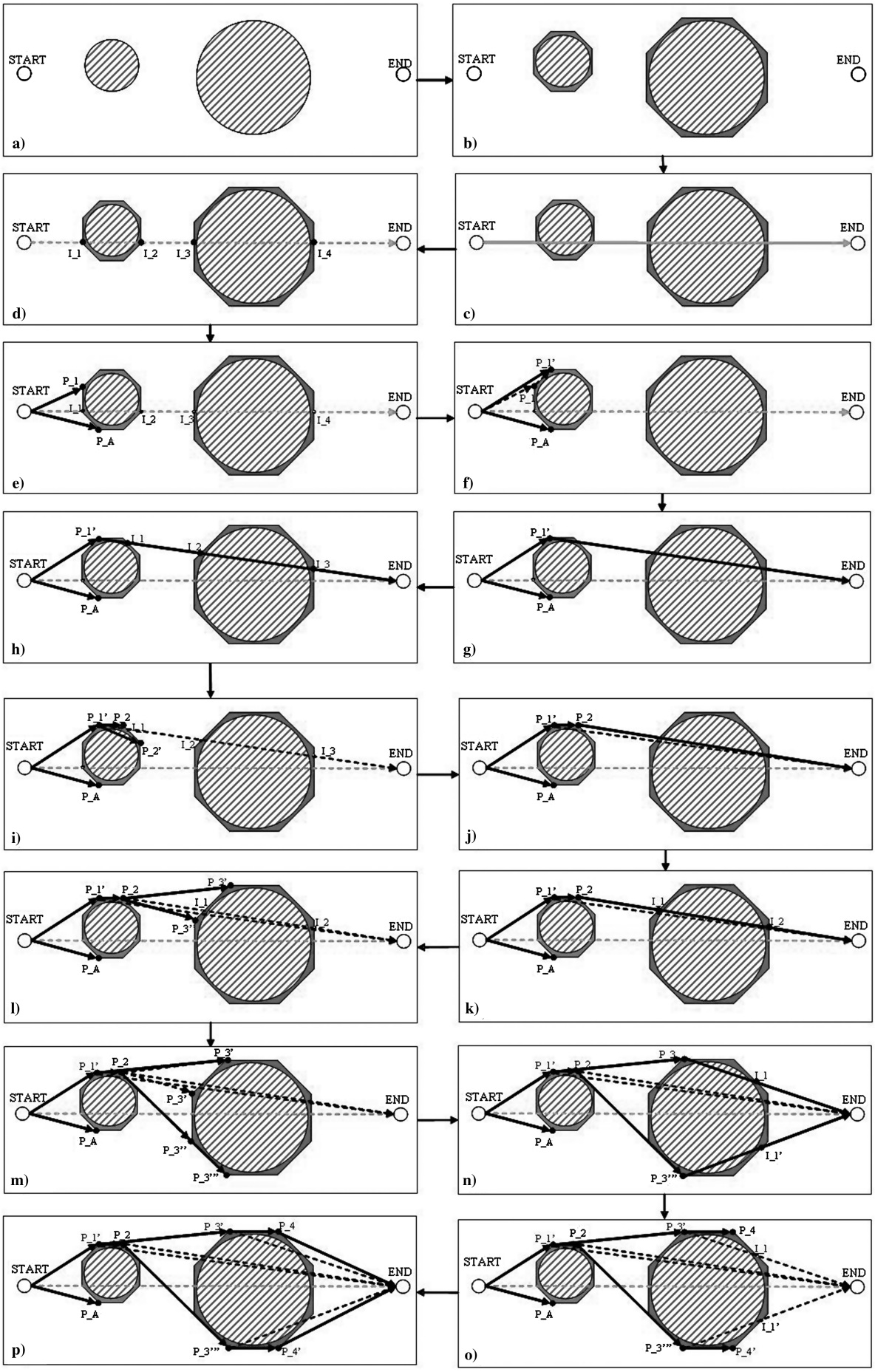


Fig. 7 Procedure for generation of the safe paths in a generic plane.

selected, and identifies the most efficient route to be used by the vehicle.

The cost algorithm comprises three routines: the climb feasibility routine, the efficiency routine, and the global cost routine.

If the first leg of the geometric safe path requires one to climb at a certain altitude, the climb feasibility routine compares the flight angle γ_s, set by the safe algorithm to reach the point Wp_1 from the START point, and the maximum flight angle, $\gamma_{\max}$, which the vehicle can assume based on its state variables (Fig. 8). If $\gamma_s > \gamma_{\max}$ the safe path is discarded. Otherwise, if $\gamma_s \leq \gamma_{\max}$ the safe path is sent to the efficiency routine. The subsequent pairs of Wp requiring an altitude transition are iteratively evaluated in the efficiency routine.

The efficiency routine elaborates each safe and climb feasible (in the initial segment) path to obtain a fly path, which is consistent with the vehicle performance and the mission priority. The flight path angles and the turning radius required by the safe path are adapted, if necessary, to the vehicle performance and mission priority by moving the waypoints defined by the safe path process in the 3-D space. Repositioning the waypoints is a strategy adopted to provide routes composed of ascent, cruise, and descent phases, as in the manned flight routes. The fly path generation process is differentiated for the three cases considered in this paper which are the time priority, whose aim is to provide a fast path, the fuel priority, whose aim is to provide a reduced fuel consumption solution, and the distance/length priority. Finally, the routine calculates the route cost

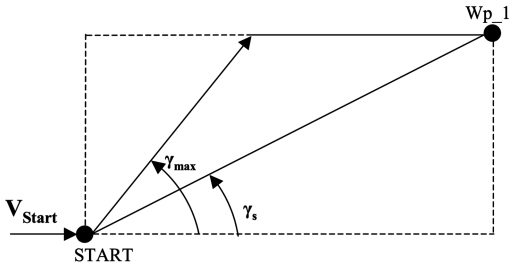


Fig. 8 Comparison between the flight angle required by the safe path and the vehicle $\gamma_{\max}$.

in terms of reduced total mission time, total fuel consumption, or total path length.

1) Case time priority: In this case the input is a safe path in a given inclined plane α and the output is the fly path composed of climb, cruise, and descent legs. A main assumption is that the cross-sectional area of 3-D obstacles in horizontal planes is constant or decreases with increasing altitude. Moreover, the cruise phase altitude is set at the altitude of the highest waypoint of the safe path. All the waypoints are initially projected on the horizontal plane α' at the cruise altitude defined above (Fig. 9). Afterward, a temporary geometric route is obtained as follows:

$$\{\text{START},\ \text{WP}_{1,\ldots,}\text{WP}_n,\ \text{END}\} \rightarrow \{\text{START},\ \text{WP}'_{1,\ldots,}\text{WP}'_n,\ \text{END}\}$$

n is the number of waypoints that compose the safe path except START and END and WP'_i is the vertical projection of point WP_i onto the plane α'. At this stage START is a known vehicle state defined by its position (lat, lon, alt), its speed **v**, and its weight. The only parameters currently defined for the other waypoints in the array are latitude, longitude, and altitude, whereas speed and weight will be iteratively assigned as described in detail in the next paragraphs.

Because the objective of this routine is to find a fast route, the climb phase is forced to be performed at the speed for the best rate of climb at a given altitude and at a given weight (V_{rap}). This parameter is available in the look-up tables of the specific vehicle. The climb angle γ_{rap} required to climb at V_{rap} speed is compared with the climb angle $\gamma_{\text{geom}_{\text{climb}}}$ corresponding to the {START, WP'_1} segment (Fig. 10).

If $\gamma_{\text{rap}} > \gamma_{\text{geom}_{\text{climb}}}$, the climb phase will be composed of one single leg at γ_{rap} climb rate. This leg is contained in the vertical plane β passing through START and WP_1 and intersects the horizontal plane α' in WPs. For the assumptions made on the shape of obstacles, no intersection occurs in this plane above the inclined plane α. Otherwise, if $\gamma_{\text{rap}} < \gamma_{\text{geom}_{\text{climb}}}$ the climb phase will be composed of two or more segments inclined at the angle γ_{rap}. In the example in Fig. 10 the first segment {START, WP''_1} is contained in the vertical plane β. The subsequent segment {WP''_1, WPs} is contained in the vertical plane β' passing through WP_1 and WP_2 and is inclined at the angle γ_{rap}. WPs is the intersection between the leg and the plane α'.

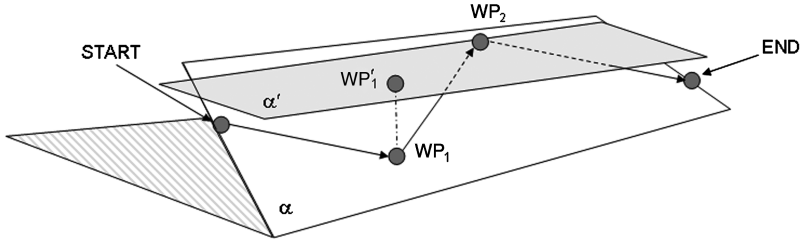


Fig. 9 Temporary geometric route for a four-points path.

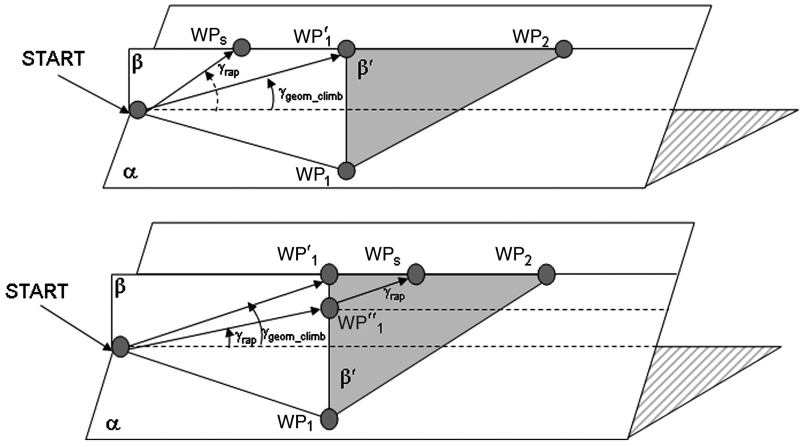


Fig. 10 The climb phase.

Except for the case $\gamma_{\text{rap}} = \gamma_{\text{geom}_{\text{climb}}}$, additional points are inserted in the original array and the temporary route will be

$$\{\text{START, WPs, WP}'_{1,\ldots}, \text{WP}'_n, \text{END}\} \quad \text{if } \gamma_{\text{rap}_{\text{climb}}} > \gamma_{\text{geom}_{\text{climb}}};$$

$$\{\text{START, WP}''_1, \text{WPs}_{\ldots,}, \text{WP}'_n, \text{END}\} \quad \text{if } \gamma_{\text{rap}_{\text{climb}}} > \gamma_{\text{geom}_{\text{climb}}}$$

Concerning the cruise phase, we assume that, because the objective in this case is to minimize the time needed to fly the given route, the waypoints which compose the cruise phase are iteratively assigned with the maximum speed at the given altitude and weight. At each waypoint, the weight is approximated from the specific fuel consumption (look-up tables) depending on the length of the previous leg. A feasibility algorithm estimates if a vehicle trajectory that simulates the flight on the given route intersects the obstacles in the turning maneuver performed at the set speed V for a given waypoint. The turning radius r_t of such a trajectory (Fig. 11) is derived from Eq. (1):

$$r_t = \frac{V^2}{g * \sqrt{n^2 - 1}} \tag{1}$$

where V is the speed temporarily assigned to WPij and

$$n = n^*; \quad \text{if } V^* < V < V_{\text{ne}} \qquad n = n(V); \quad \text{if } V_S < V < V^*$$

n^* is the load factor selected for normal flight conditions in the V–n diagram of the specific aircraft and V_{ne} is its speed which is not to be exceeded. We define r_o as the radius of the arc of the smaller circle containing the obstacle vertices in a narrow area around the waypoint and tangent to the segment_i and segment_j in Fig. 11. The algorithm compares r_o and r_t. If $r_t < r_o$ the speed in the WPij is set to V. If $r_o < r_t$ the speed is iteratively decreased to reduce the radius r_t until it is equal to r_o while the vehicle speed is greater than the stalling speed. If this cannot be achieved with a vehicle speed greater than stalling speed, the route is discarded.

The descent phase will be contained in the vertical plane δ passing through WP'_n and END. By estimating the vehicle weight in the last segment of the path, the descent speed can be derived, and we define γ_D as the descent angle from the cruise plane to END altitude at this speed. γ_D is compared to the descent angle $\gamma_{\text{geom}_{\text{desc}}}$, which the segment $\{\text{WP}'_n, \text{END}\}$ forms with the ground. If $\gamma_D < \gamma_{\text{geom}_{\text{desc}}}$ the descent phase is set at $\gamma_{\text{geom}_{\text{desc}}}$. Otherwise, if $\gamma_D > \gamma_{\text{geom}_{\text{desc}}}$ the descent phase of the route is set on the segment $\{\text{WP}_D, \text{END}\}$, which is the segment inclined by an angle γ_D, passing through END, and contained in the plane δ. δ is a safe plane above the α plane. The temporary route will be

$$\{\text{START, WPs, WP}'_{1,\ldots,}, \text{WP}'_n, \text{END}\} \quad \text{if } \gamma_{\text{rap}} > \gamma_{\text{geom}_{\text{climb}}} \quad \text{and} \quad \gamma_D < \gamma_{\text{geom}_{\text{desc}}};$$

$$\{\text{START, WP}''_1, \text{WPs}_{,\ldots,}, \text{WP}'_n, \text{END}\} \quad \text{if } \gamma_{\text{rap}} < \gamma_{\text{geom}_{\text{climb}}} \quad \text{and} \quad \gamma_D < \gamma_{\text{geom}_{\text{desc}}};$$

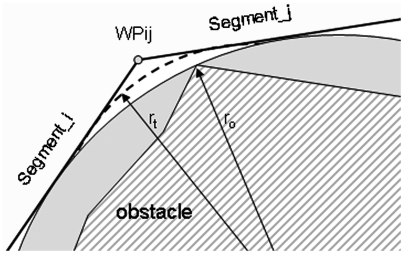


Fig. 11 Turning feasibility.

$$\{\text{START, WPs, WP}'_{1,\ldots,}, \text{WP}'_n, \text{WP}_D, \text{END}\} \quad \text{if } \gamma_{\text{rap}} > \gamma_{\text{geom}_{\text{climb}}} \quad \text{and} \quad \gamma_D > \gamma_{\text{geom}_{\text{desc}}};$$

$$\{\text{START, WP}''_1, \text{WPs}_{,\ldots,}, \text{WP}'_n, \text{WP}_D, \text{END}\} \quad \text{if } \gamma_{\text{rap}} < \gamma_{\text{geom}_{\text{climb}}} \quad \text{and} \quad \gamma_D > \gamma_{\text{geom}_{\text{desc}}}$$

2) Case fuel priority: The planning algorithm can be set to find a safe route with reduced fuel consumption. In this case the algorithm follows same procedure as the time priority case except in the cruise phase the speed in each waypoint will be set at the minimum fuel consumption speed at a given altitude and at a given weight. This parameter is available in the look-up tables.

3) Case distance/length priority: In each plane of the sheaf, we select the shortest path within the set of geometric paths returned by the safe algorithm. In this case, the fly path is generated by recursively assigning the speed of the safe path waypoints as follows:

$$V_{i+1} = V_i \quad \text{if } \gamma_{\text{geom}_i} = 0; \qquad \text{(cruise phase)}$$

$$V_{i+1} = V_i^* \sqrt{\cos \gamma_{\text{geom}_i}} \quad \text{if } \gamma_{\text{geom}_i} > 0; \qquad \text{(climb phase)}$$

$$V_{i+1} = V_i^* \sqrt{\cos \gamma_{\text{geom}_i}} \quad \text{if } \gamma_{\text{geom}_i} < 0; \qquad \text{(descent phase)}$$

where 1) γ_{geom_i} is the angle between the horizontal plane passing through the generic waypoint WP_i and the segment joining WP_i and waypoint WP_{i+1}; and 2) V_i and V_{i+1} are the speed value previously assigned in Wp_i and the speed value temporary assigned to WP_{i+1}, respectively.

Initially the climb feasibility routine verifies that γ_{geom_i} is compatible with the maximum climb or descent angle allowed. As in the above cases (time priority and fuel priority), the algorithm verifies that the speed V_{i+1} is compatible with the turn geometry and vehicle performance at the given altitude and weight. To exploit the look-up tables and estimate the feasibility of the route with the assigned velocity value, we assume that the vehicle is in a cruise phase at the altitude of WP_{i+1}. The geometric distance between subsequent waypoints is then computed, and the weight at each point is recursively computed based on the specific fuel consumption at given conditions. This procedure calls for the value of V_{i+1} to be confirmed, reduced, or the path to be discarded.

B. Mission Planning for Monitor Command

It enables the operator to command the vehicle to monitor a set of objectives or a specific area on the ground.

Set of objectives: The operator is asked to select the following instructions/parameters:

1) monitor,
2) $\text{WP}_1, \text{WP}_2, \ldots \text{WP}_N$,
3) priority.

If the vehicle is required to execute the mission getting the reduction of fuel consumption as first priority, the following command shall be sent: *Monitor WP_1, WP_2, WP_3, WP_4 at fuel priority*.

Such locations are the primary mission waypoints of the route. The path planning algorithm calculates the macroleg between each couple of objectives as described in the previous section, and then connects the sequence of macrolegs to obtain the global routes.

Figure 12 provides a simple example where there are no obstacles to avoid. Thus, the global route is the connected sequence of the straight lines between the objectives.

Area: To command the vehicle to monitor an area the operator is asked to select the following instructions/parameters:

1) monitor,
2) area location as the coordinates of two opposite vertices of it,

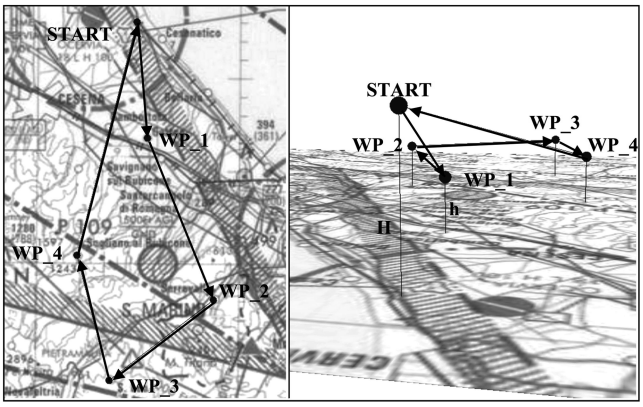


Fig. 12 The monitor command: set of objectives (START: initial vehicle position; WP_i: primary mission waypoint; h: monitor altitude; H: start altitude).

3) priority.

Besides these instructions, the operator can also indicate the point where the vehicle has to loiter after monitoring the area. If he does not indicate such a point the vehicle will return to the initial point. Figure 13 shows the route the vehicle flies to monitor an area where there are no obstacles to avoid. In the figure, WP_7 is the waypoint where the vehicle flies after having executed the mission.

In this case the mission planning algorithm

1) Computes the length of the sides that bound the rectangular area. The two sides containing the input vertex which is closest to the vehicle current position are defined as side_1 and side_2.

2) Computes the ratios side_1/scan_width and side_2/scan_width. The scan_width is also an input parameter and depends on the pan parameter of the payload sensor and on meteorological conditions. It defines the width of a single scanning strip. Such ratios define the number of scanning strips the vehicle has to run to cover the entire area by, respectively, entering from side_1 or from side_2.

3) Finally, it computes the set of primary mission waypoints (PMWs), which are the midpoint of the edge of each scanning strip, for both cases, as

$$(\text{no. of PMWs}) = 2^*(\text{no. of scanning strips}) \qquad (2)$$

Afterward, the path planning algorithm calculates the global routes the vehicle should fly by entering the area from side_1 or side_2. Finally the cheapest route according to the set priority is selected. A sample path is depicted.

C. Mission Planning for the Survey Command

Two different modes to survey a target have been considered and each of them can be of two types as follows:

1) *Spot survey:*

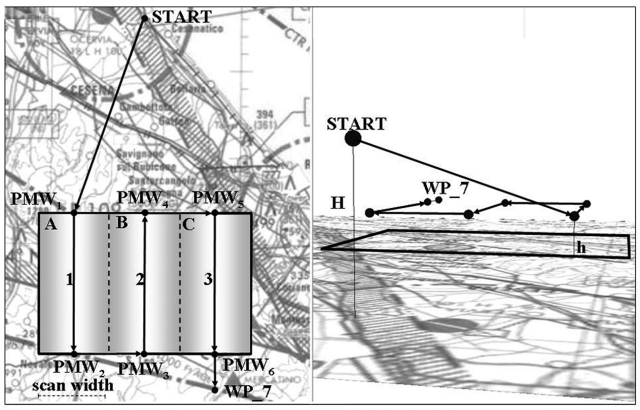


Fig. 13 The monitor command: rectangular area (START: current vehicle position; PMW_i: primary mission point; h: monitor height; H: START altitude; A, B, C: portion of area for each scan).

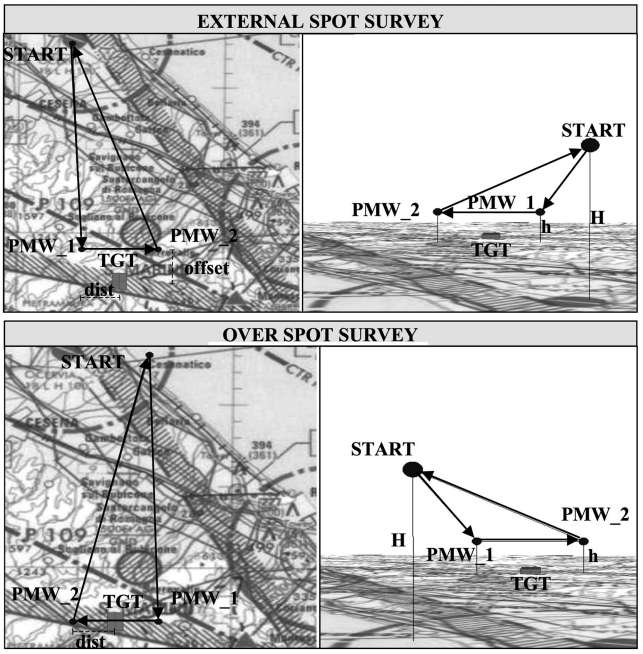


Fig. 14 The spot survey command (START: initial vehicle position; PMW_i: primary mission waypoint; h: survey altitude; H: start altitude; TGT: target; offset: payload parameter; dist: survey parameter).

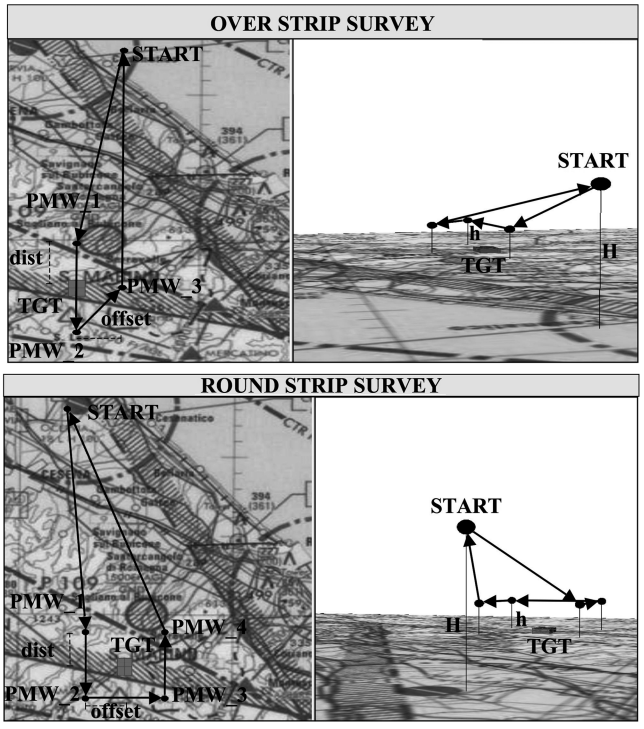


Fig. 15 The strip survey command (START: initial vehicle position; PMW_i: primary mission waypoint; h: survey altitude; H: start altitude; TGT: target; offset: payload parameter; dist: survey parameter).

a) External spot survey: a command to perform a survey of a known target without overflying the point itself.

b) Over spot survey: a command to perform a survey of a known target by overflying it.

2) *Strip survey:*

a) Over strip survey: a command to perform a survey of a predefined land strip by overflying it.

b) Round strip survey: a command to perform a survey of a predefined land strip without overflying the strip itself.

The following instructions are

Survey; target or land strip; priority; payload parameter: offset and distance.

In each case the mission planning algorithm calculates a set of primary mission waypoints. The relative position of such waypoints to the target is computed in order to follow the four specific observation strategies. The parameters to locate the primary mission waypoints, such as the observation altitude, the horizontal image capturing distance, and the scanning strip length, can be related to the sensor payload capabilities. The path planning algorithm is then applied to calculate the macrolegs between each couple of primary mission waypoints.

In Figs. 14 and 15, the positions of the primary mission waypoints with respect to the target or the land strip to be surveyed are represented.

VI. Simulation Results

The reliability of the algorithm, defined as the ability to correctly accomplish the commands, and the computation time required to compute the routes have been verified with different operational scenarios and different flight priorities in the simulation-based test environment. Several simulations have been run to test the algorithm while controlling a simulated UAS model. The obstacle avoidance strategy described in this paper has been running in the experimental UAS control station at the Alenia Aeronautica Simulation Centre in Torino (Italy). This system (Fig. 16) has been designed to be a test bed to evaluate functional aspects such as graphic formats and mission management functions and configuration aspects such as interface layout or ergonomic and interaction features of the UASs man–machine interfaces.

The simulations reveal the following:

1) The obstacle avoidance strategies provide routes that do not collide with existing obstacles.

2) The selected priority is respected.

3) The computation time required by the algorithm to calculate the route is reasonable. How to scale such results with respect of real equipment to be installed on board UAS will be part of future activities.

In the following sections the results of some simulations executed for the fly-to command, the monitor command, and the survey command are reported.

A. Fly-To Command

In the case of the fly-to command the simulations have been executed by considering different missions characterized by different distances between the vehicle START position and the destination point and different number of known obstacles. A number of routes have been computed corresponding to a different selected priority in each scenario, and by recording the value of the mission length, the estimated fuel to be consumed, and the time required to accomplish the mission.

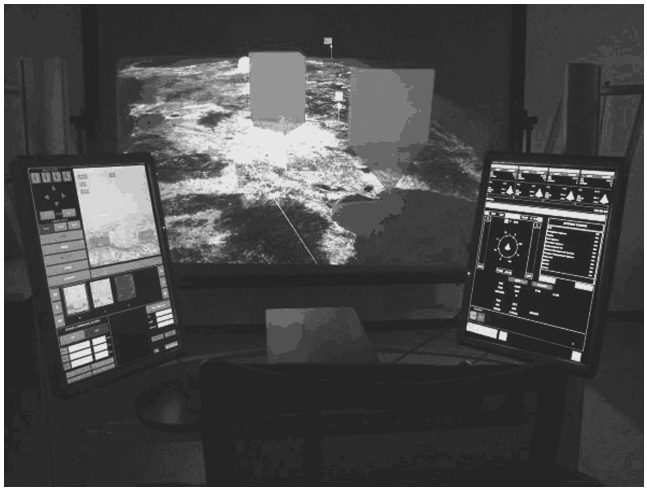


Fig. 16 The experimental UAV control station at Alenia Aeronautica.

Table 2 reports the results of the simulations relative to three different missions performed by a specific vehicle in scenarios with different numbers of obstacles. The first mission consists of flying toward a destination point (end in table) which is 45 km far (in line of sight) from the starting point (start in table). The second mission consists of flying toward a destination point which is located 125 km from the starting point of the mission, and the third mission consists of flying toward a destination point at 265 km from the starting point of the mission. The results reported in the table are the mission length, the estimated fuel to be consumed, and the time required to accomplish the mission. Moreover, in each row of the table, the minimum value of the length, estimated fuel, and the mission time between those computed for distance, fuel, and time priorities for the same mission, are presented in bold face type.

Fig. 17 shows on the map the routes computed to fly to the destination point (END) which is about 45 km farther from the START point in a scenario populated with four obstacles (Fig. 17a), six obstacles (Fig. 17b), and eight obstacles (Fig. 17c). Figure 18 shows on the map the routes computed to fly to the destination point (END) which is about 125 km farther from the START point in a scenario populated with four obstacles (Fig. 18a), six obstacles (Fig. 18b), and eight obstacles (Fig. 18c). Figure 19 shows on the map the routes computed to fly to the destination point (END) which is about 256 km farther from the START point in a scenario populated with four obstacles (Fig. 19a), six obstacles (Fig. 19b), and eight obstacles (Fig. 19c).

Figures 17–19 and Figs. 22, 24, and 26 show a 2-D representation of the routes as their projection on the ground. The 3-D routes computed in the same scenarios are depicted in a perspective view in Fig. 20.

In addition to the results reported in Table 2, the time needed to calculate each route has also been computed. In Fig. 21 the time required to calculate routes in several mission scenarios, differing from each other in the number or position of the obstacles, is shown.

It is evident that the computation time depends on the number of obstacles in the scenario. However, the computation time can evidently be different for the same number of obstacles. It could be due to the relative position of the obstacles in the scenario. In Fig. 22 a comparison between the time required to calculate the route corresponding to time priority in different scenarios in terms of position of the obstacles is reported. In Fig. 22a the mission scenario is displayed and the computation time needed to compute the route is 3.76 s. In Fig. 22b the obstacle identified with number 1 is translated

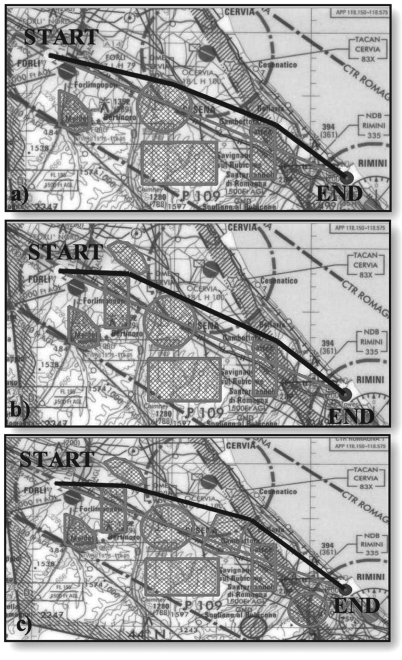


Fig. 17 The time route for a fly-to command with the distance (START, END) = 45 km.

Table 2 The reliability of the algorithm: case fly-to command

		Fly-to command								
		Dist(Start-End) = 45 km			Dist(Start-End) = 125 km			Dist(Start-End) = 265 km		
		Priority			Priority			Priority		
No. of obstacles	Results of simulation	Distance	Fuel	Time	Distance	Fuel	Time	Distance	Fuel	Time
1	Length, m	**46, 814**	47,268	47,042	**128, 683**	129,484	129,186	**270, 256**	274,989	274,990
	Fuel, kg	41	**32**	35	90	**70**	97	340	**148**	148
	Time, s	327	158	**146**	642	426	**402**	2,513	829	**827**
2	Length, m	**47, 271**	48,997	47,285	**130, 836**	132,360	130,836	**290, 877**	293,547	293,547
	Fuel, kg	54	**30**	35	168	**63**	98	375	**110**	110
	Time, s	410	186	**147**	1,182	439	**407**	2,637	889	**893**
3	Length, m	**47, 271**	48,997	47,285	**131, 096**	132,426	131,105	**292, 183**	294,947	294,947
	Fuel, kg	54	**30**	35	111	**63**	98	377	**110.62**	111
	Time, s	410	186	**147**	897	438	**408**	2,651	893.32	**897**
4	Length, m	**47, 271**	48,997	48,997	**131, 096**	132,426	131,105	**292, 183**	294,947	294,947
	Fuel, kg	54	**30**	30	111	**63**	98	377	**110**	111
	Time, s	410	186	**186**	897	438	**408**	2,651	893	**897**
5	Length, m	**47, 271**	48,997	47,285	**130, 776**	132,126	130,825	**292, 183**	294,947	294,947
	Fuel, kg	54	**30**	35	110	**64**	98	377	**110**	111
	Time, s	410	186	**147**	788	439	**407**	2,651	893	**897**
6	Length, m	**47, 688**	48,129	47,687	**133, 211**	135,619	133,211	**302, 679**	305,123	305,123
	Fuel, kg	61	**33**	35	171	**64**	100	390	**113**	114
	Time, s	430	161	**148**	1,204	422	**414**	2,748	920	**924**
7	Length, m	**48, 238**	48,923	48,236	**133, 562**	143,298.30	133,562	**305, 846**	305,846	305,846
	Fuel, kg	62	**34.22**	36	172	**58.22**	100	394	**230**	230.60
	Time, s	435	169.77	**150**	1,208	441.52	**415**	2,775	953	**952.32**
8	Length, m	**48, 238**	48,923.18	48,236	**133, 562**	143,298	133,562	**305, 846**	305,846	30,5846
	Fuel, kg	62	**34**	36	172	**58**	100	394.93	**230**	230.60
	Time, s	435	169	**150**	1,208	441	**415**	2,775.52	953	**952.32**

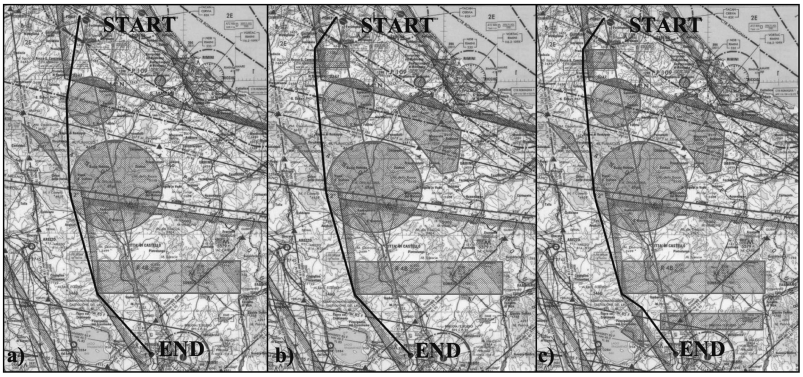


Fig. 18 The time route for a fly-to command and distance (START, END) = 125 km.

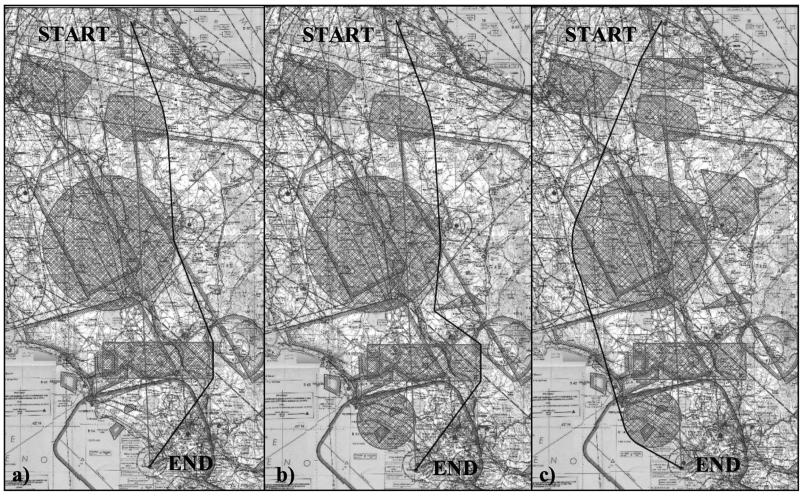


Fig. 19 The time route for a fly-to command and distance (START, END) = 265 km.

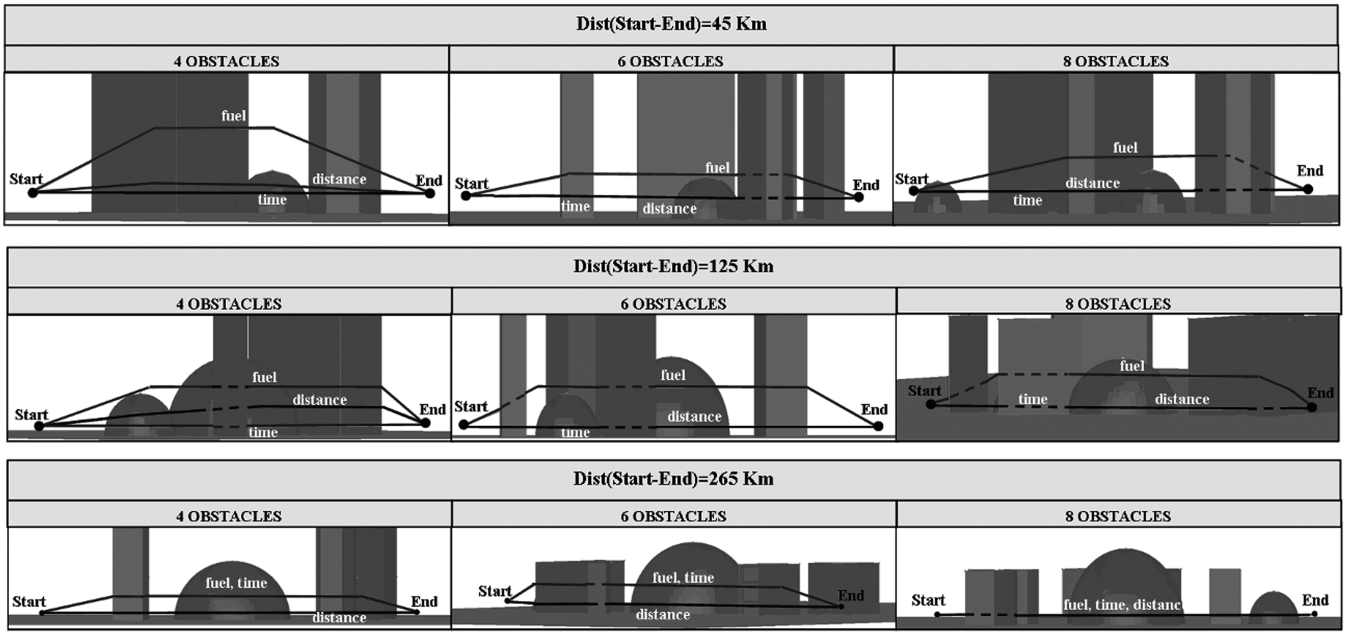


Fig. 20 A perspective view of 3-D routes for fly-to command.

until it intersects the line joining the start and the end. In this case the computation time increases and becomes 5.47 s. In Fig. 22c, the obstacle identified with number 2 is translated and the computation time becomes 7.51 s. Finally, in Fig. 22d the obstacle identified with number 2 is translated closer to the straight line and the computation time becomes 6.43 s.

The computation time seems to be influenced by scenarios in which the obstacles are close to each other. The obstacles that are far from the start–end line of sight have very little influence on the total computation time.

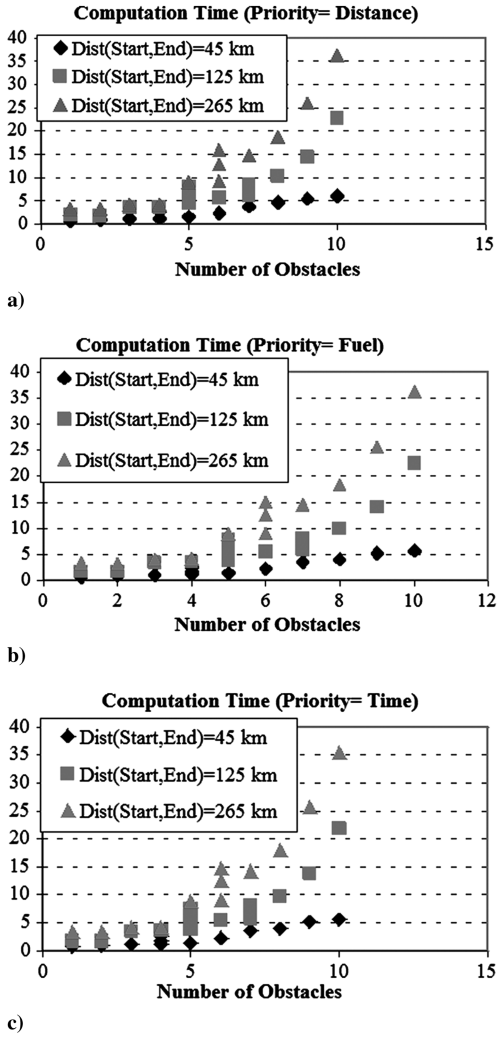


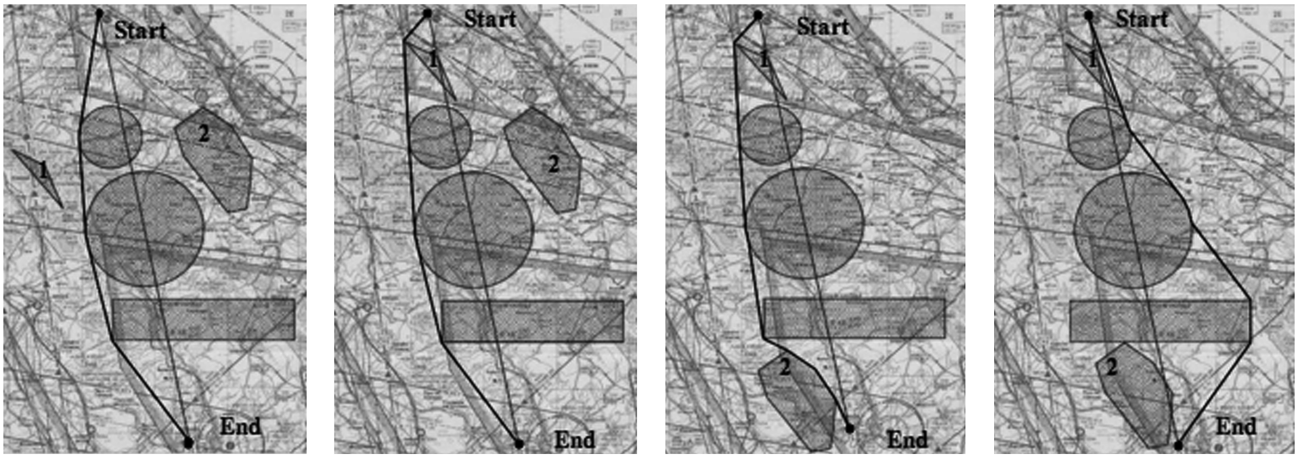
Fig. 21 The computation time.

B. Monitor Command

Set of objectives: In the case of monitoring a set of waypoints the simulations have been run in scenarios characterized by different numbers of objectives, the distance between them, and the number of

Table 3 Monitor set of objectives command

No. of obstacles		Distance	Priority Fuel	Time
0	Length, m	**303, 897**	303,897	303,897
	Fuel, kg	391	**229**	229
	Time, s	2,741	945	**945**
1	Length, m	**313, 072**	313,862	313,269
	Fuel, kg	340	**216**	236
	Time, s	2,386	1,032	**974**
2	Length, m	**313, 916**	316,592	314,113
	Fuel, kg	341	**213**	236
	Time, s	2,394	1,060	**977**
3	Length, m	**324, 571**	328,848	324,768
	Fuel, kg	355	**187**	244
	Time, s	2,494	1,103	**1, 010**
4	Length, m	**329, 741**	345,121	329,938
	Fuel, kg	362	**182**	248
	Time, s	2,541	1,184	**1, 026**
5	Length, m	**325, 448**	343,707	325,691
	Fuel, kg	358	**180**	245
	Time, s	2,542	1,167	**1, 013**
6	Length, m	**330, 853**	346,066	331,096
	Fuel, kg	365	**185**	249
	Time, s	2,591	1,194	**1, 030**
7	Length, m	**330, 853**	346,809	331,096
	Fuel, kg	365	**193**	249
	Time, s	2,591	1,211	**1, 030**
8	Length, m	**330, 853**	346,809	331,096
	Fuel, kg	365	**193**	249
	Time, s	2,591	1,211	**1, 030**



a) Computation Time = 3.76 s b) Computation Time = 5.47 s c) Computation Time = 7.51 s d) Computation Time = 6.43 s

Fig. 22 The computation time as a function of obstacle positions.

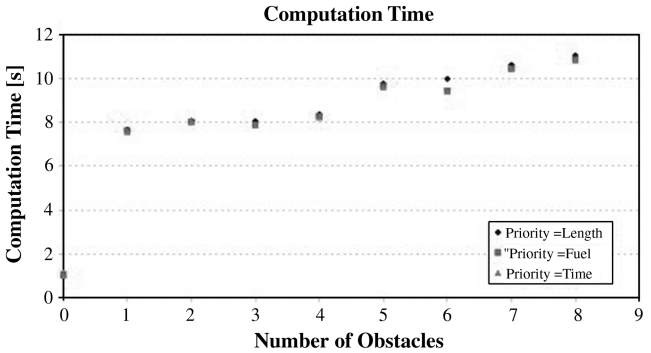


Fig. 23 The computation time for the monitor command.

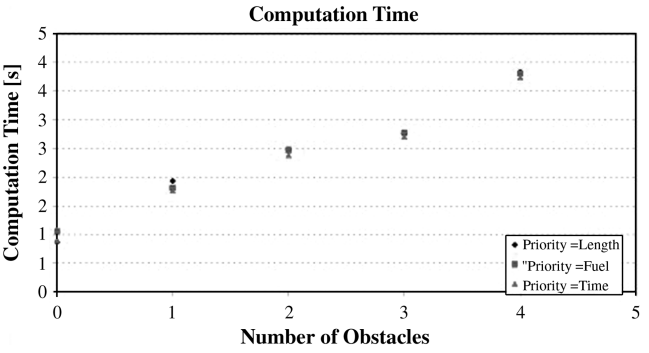


Fig. 25 The computation time for the survey command.

obstacles. The routes corresponding to each priority in each scenario have been computed. Table 3 reports the results of the simulations relative to a scenario characterized by three objectives and a different number of obstacles. The distance between the vehicle position (start in table) and the first objective (Wp_1) is 45 km, the distance between the first and the second objective (Wp_2) is 105 km, and the distance between the second and the third waypoint (Wp_3) is 150 km. Figure 23 reports the computation time. Figure 24 reports the routes calculated by setting the mission length, the fuel to be consumed, and the mission duration time as priority in eight scenarios characterized by an increasing number of obstacles.

C. Survey Command

In the case of the survey command the simulations have been run by considering different missions characterized by the distance between the vehicle and the target or the area to survey in the presence of a certain number of obstacles.

Table 4 Over spot survey command

No. of Obstacles		Priority Distance	Priority Fuel	Priority Time
0	Length	**167, 652**	167,982	167,982
	Fuel	197	**148**	149
	Time, s	1,526	771	**765**
1	Length	**167, 652**	167,982	167,982
	Fuel	197	**148**	147
	Time, s	1,526	771	**765**
2	Length	**168, 206**	171,701	168,535
	Fuel	200	**114**	137
	Time, s	1,550	679	**624**
3	Length	**168, 206**	171,701	168,535
	Fuel	200	**114**	137
	Time, s	1,550	679	**624**
4	Length	**168, 206**	171,701	168,535
	Fuel	200	**114**	137
	Time, s	1,550	679	**624**

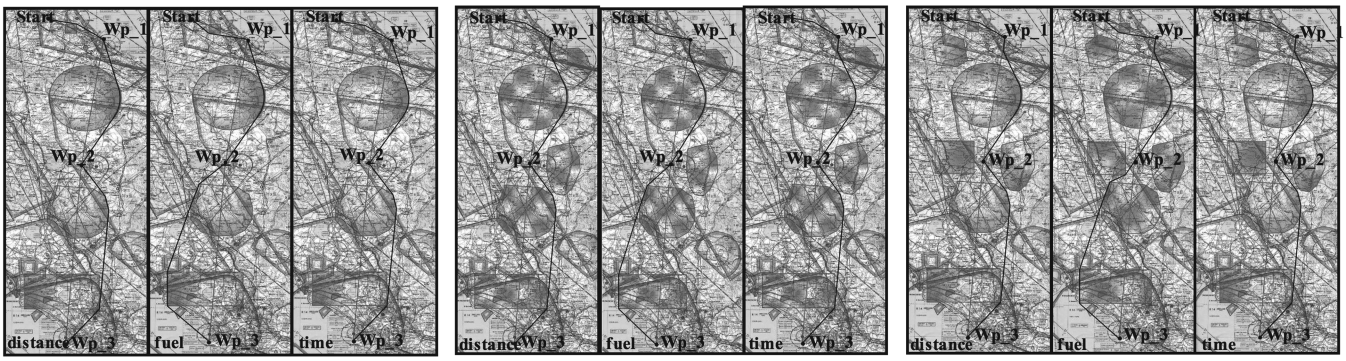


Fig. 24 Monitoring the set of three objectives.

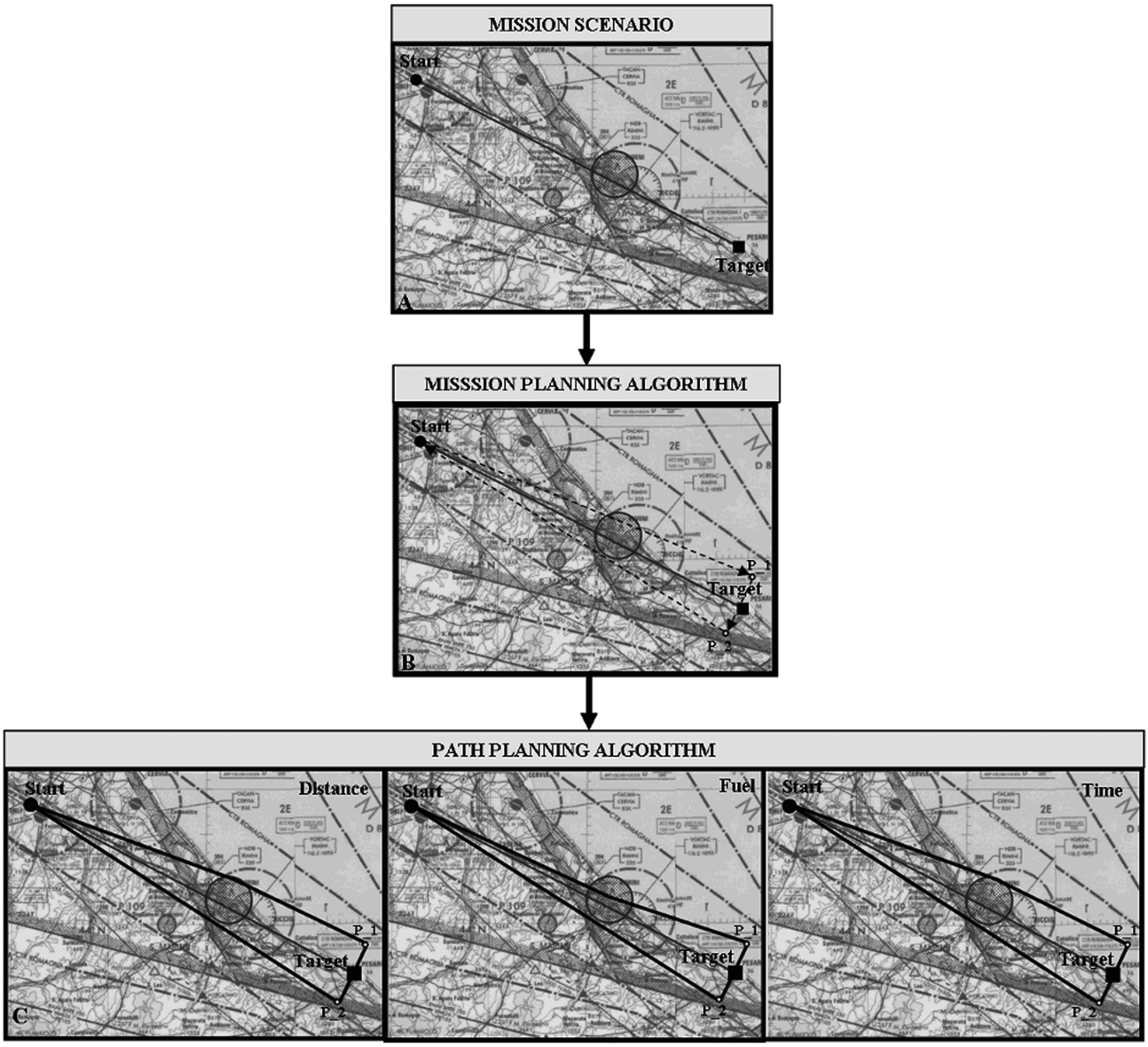


Fig. 26 The over spot survey route for a simple scenario.

Table 4 reports the results of the simulations relative to a mission consisting of the survey target far from the starting point of the mission 90 km, executed in scenarios with different numbers of obstacles. In Fig. 25, the time needed to compute the route in the considered scenarios is reported.

In the survey command, the mission planning algorithm identifies the primary mission waypoints, and the path planning algorithm finds the routes between each couple of primary waypoints. In Fig. 26 the route to survey a target 90 km far from the vehicle start is shown: starting from the mission scenario (Fig. 26a), two primary mission waypoints are computed (P_1 and P_2 in Fig. 26b) and, finally, the final routes are calculated by setting the three priorities (Fig. 26c).

VII. Conclusions

In this paper an alternative planning algorithm is described. The algorithm aims at increasing the autonomy of the vehicle by handling high level commands received from the operator, as well as deciding how to replan the mission when unforeseen situations occur. The planning algorithm is based on a graph, but differs from the others because it allows the exploration of more portions of space, replicating the approach of human pilots, and enables the vehicle to avoid obstacles of any shape while requiring a short computation time.

For what concerns the human operator, this mission planning and replanning logic aims at providing support in decision making tasks. This approach will have to be considered coherently in the foreseen future concept of the human–machine interface for UAV systems.

Future work will focus on further increasing vehicle decisional autonomy and on the management of UAS fleets.

References

[1] "Unmanned Aircraft Systems Roadmap 2005-2030," U.S. Department of Defence, http://www.acq.osd.mil/.

[2] McManus, I. A., and Walker, R. A., "Multidisciplinary Approach to Intelligent Unmanned-Airborne-Vehicles Mission Planning," *Journal of Aircraft*, Vol. 43, No. 2, March–April 2006, pp. 318–335. doi:10.2514/1.15204

[3] Rubio, J. C., Vagners, J., and Rysdyk, R., "Adaptive Path Planning for Autonomous UAS Oceanic Search Missions," AIAA Paper 2004-6228, 2004.

[4] Williams, K. W., "Human Factors Implications of Unmanned Aircraft Accidents: Flight Control Problems," Civil Aerospace Medical Inst., Federal Aviation Administration, Final Report, April 2006.

[5] McCarley, J. S., and Wickens, C. D., "Human Factors Implications of UASs in the National Airspace," TR AHFD-05-05/FAA-05-01, April 2005, prepared for Aviation Administration Atlantic City International Airport, NJ.

[6] McCarley, J. S., and Wickens, C. D., "Human Factors Concern in UAS

Flight," Inst. of Aviation, Aviation Human Factors Div., Univ. of Illinois at Urbana–Champaign, IL, 2004, www.hf.faa.org.

[7] Williams, K. W., "A Summary of Unmanned Aircraft Accident/Incident Data: Human Factors Implications," Civil Aerospace Medical Inst., Federal Aviation Administration, Dec. 2004.

[8] Barbier, M., and Chanthery, E., "Autonomous Mission Management for Unmanned Aerial Vehicles," *Aerospace Science and Technology*, Vol. 8, No. 4, June 2004, pp. 359–368. doi:10.1016/j.ast.2004.01.003

[9] Bortoff, S. A., "Path Planning for UASs," *Proceedings of the American Control Conference*, IEEE, Piscataway, NJ, June 2000.

[10] Jun, M., and D'Andrea, R., "Path Planning for Unmanned Aerial Vehicles in Uncertain and Adversarial Environments," *Cooperative Control. Models, Application and Algorithms*, edited by S. Butenko, R. Mulphey, and P. Pardalos, Kluwer, The Netherlands, 2002, pp. 95–108.

[11] How, J., King, E., and Kuwata, Y., "Flight Demonstrations of Cooperative Control for UAV Teams," AIAA Paper 2004-6490, 2004.

[12] Rathinam, S., Zennaro, M., Mak, T., and Sengupta, R., "An Architecture for UAS Team Control," *5th IFAC Symposium on Intelligent Autonomous Vehicles*, Elsevier Science, Oxford, England, U.K., 2004, pp. 1–7.

[13] Beard, R. W., McLain, T. W., Goodrich, M. A., and Anderson, E. R., "Coordinated Target Assignment and Intercept Unmanned Air Vehicle," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 6, Dec. 2002, pp. 911–922. doi:10.1109/TRA.2002.805653

[14] Gu, D., Postlethwaite, I., and Kim, Y., "A Comprehensive Study on Flight Path Selection Algorithm," *Target Tracking: Algorithms and Applications*, Institution of Electrical Engineers, Stevenage, England, U.K., 2006, pp. 77–90.

[15] Kuwata, Y., and How, J., "Three Dimensional Receding Horizon Control for UAVs," *AIAA Guidance, Navigation and Control Conference and Exhibit*, AIAA Paper 2004-5144, 2004.

[16] Kim, Y., Gu, D., and Postlethwaite, I., "Real-Time Optimal Mission Scheduling and Flight Path Selection," *IEEE Transactions on Automatic Control*, Vol. 52, No. 6, June 2007, pp. 1119–1123.. doi:10.1109/TAC.2007.899048

[17] Chakravorty, S., and Junkins, J. L., "Motion Planning in Uncertain Environments with Vision-Like Sensors," *Automatica*, Vol. 43, No. 12, 2007, pp. 2104–2111. doi:10.1016/j.automatica.2007.04.022

[18] Economou, J. T., Kladis, G. P., Tsourdos, A., and White, B., "A Node-To-Node Composite Graph and Pseudo-Boolean Modelling: An Unmanned Aerial Vehicle Energy Application," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 221, No. 5, Oct. 2007, pp. 815–830. doi:10.1243/09544100JAERO191

[19] Ben-Moshe, B., "Computing the Visibility Graph of Points Within Polygon," *SCG (Symposium on Computational Geometry) '04*, Assoc. for Computing Machinery, New York, June 2004.

[20] Nouri Baygi, M., and Ghodsi, M., "3D Visibility Graphs," *12th CSI Computer Conference (CSICC'2006)*, Computer Society of Iran, Tehran, Iran, Feb. 2007.

[21] Cummings, M. L., Platts, J. T., and Sulmistras, A., "Human Performance Considerations in the Development of Interoperability Standards for UAS Interfaces," *Proceedings of the Moving Autonomy Forward Conference*, Muretex, Lincoln, England, U.K., 2006.

[22] Endsley, M. R., "Automation and Situation Awareness," *Automation and Human Performance: Theory and Applications*, Lawrence Erlbaum, Mahwah, NJ, pp. 163–181.

[23] Sheridan, T. B., *Human and Automation*, Wiley, New York, 2002.