American Journal of Science and Technology
2015; 2(6): 321-328
Published online December 11, 2015 (http://www.aascit.org/journal/ajst)
ISSN: 2375-3846

American Association for Science and Technology

Keywords
RPAS,
Path Planning,
Collision Avoidance,
Risk Analysis

Received: October 12, 2015
Revised: November 7, 2015
Accepted: November 9, 2015

Operation Oriented Path Planning Strategies for Rpas

Giorgio Guglieri[1], Alessandro Lombardi[1], Gianluca Ristorto[2]

[1]Department of Mechanical and Aerospace Engineering, Corso Duca degli Abruzzi, Torino, Italy
[2]Department of Science and Technology, Piazza Università, Bolzano, Italy

Email address
giorgio.guglieri@polito.it (G. Guglieri), alessandro.lombardi@polito.it (A. Lombardi), gianluca.ristorto@unibz.it (G. Ristorto)

Citation
Giorgio Guglieri, Alessandro Lombardi, Gianluca Ristorto. Operation Oriented Path Planning Strategies for Rpas. *American Journal of Science and Technology*.
Vol. 2, No. 6, 2015, pp. 321-328.

Abstract
Due to the recent spread of RPAS into the national airfields, civil aviation authorities are actively involved in the development of regulations for RPAS, especially for small vehicles with mass less than 150 kg. These regulations often require that the RPAS operators perform a risk analysis to assess the level of risk of the operations. The paper considers the Italian regulation and describes the implementation of the RPAS risk analysis method proposed by ENAC into a 2D flight path planning software for UAV that is called JavaCube. This tool is able to generate waypoint-based paths based on graph search algorithms which incorporate the risk analysis model within their cost function so that the risk for the aircraft of occurring in catastrophic failure is minimized. The resulting paths are shown on a risk map that is generated according to UAV data, flight altitude and the population density distribution of the overflown area. This tool could provide a useful UAV path planner that meets the requirements of the current Italian regulation.

1. Introduction

Remotely piloted aircraft systems (RPAS) are showing a remarkable spread in recent years. Although UAVs were once used for military applications, now they are integrating into the national airspace of different countries to perform civil operations.

The miniaturization of electronic components (sensors, inertial measurement units, actuators and brushless motors), the improvements in battery life duration has led to a rapid and often uncontrolled spread of small electrically powered RPASs.

The civil aviation authorities are involved in the drafting of an adequate legislation to regulate the use of RPAS (typically with a mass less than 150 kg) for civil applications. These drafts often contain information concerning the risk evaluation for RPAS operations. The Italian regulation, for example, divides operations into critical and non-critical scenarios and in both cases persons who intend to operate RPAS are required to deliver a risk analysis prior to its operation to assess the overall risk [1].

As a result, a growing number of organizations and researchers are facing the problem of RPAS risk evaluation. ULTRA (Unmanned Aerial Systems in European Airspace) is a project funded by European Commission which involves different public and private stakeholders to promote the insertion of light RPAS into European Airspace in the short term. The consortium has released some deliverables that point out the gaps to fill between the current situation and the future scenarios. Deliverable 'Safety aspects of civil RPAS operations' [2] considers two current risk assessment criteria for RPAS. The first is JAA/EUROCONTROL RPAS Task Force [3], which is an effort by CAAs to establish acceptable risk procedures for RPAS operations in Europe, and defines 5 levels of hazard severity:

- severity 1: uncontrolled flight followed by an uncontrolled crash.
- severity 2: failure leading to controlled loss of the RPAS over an unpopulated area.
- severity 3: failure leading to safety reduction (e.g. communication loss with autonomous flight).
- severity 4: failure leading to slight safety reduction in safety (e.g. loss of redundancy).
- severity 5: failure leading to no safety effect.

The second risk framework is NATO STANAG 4671 [4], which uses a probability system with five classes (extremely improbable, extremely remote, remote, probable and frequent) and five severity definitions (catastrophic, hazardous, major, minor, no safety effect).

RPAS integration into national airspaces is also a hot topic at the SESAR Innovation Days, a yearly conference through which SESAR disseminates the results of its research programs. For instance, E. Pastor et al. [5] propose a brand new simulation infrastructure that will allow a real time simulation by coupling a highly capable RPAS simulation system together with a Eurocontrols ATC simulation environment (eDEP), with the possibility to perform missions using historic or forecast traffic of Eurocontrols database. This simulation framework has been tested [6] by simulating the separation maneuvers between a HALE RPAS (a Northrop Grumman Global Hawk) and a faster airplane flying at the same altitude. In a first phase, the RPAS flies without any flight intent, i.e. the communication between the RPAS and ATC is managed only via transponder and ADS-B. In a second phase the RPAS can pro-actively act providing real intentions via voice communications toward the ATC.

RPAS path planning combined with risk evaluation and analysis is an important topic due to recent spread of small RPAS into national airfields. Rudnik-Cohen et al. [7] highlight the tradeoff between risk and flight-time for RPASs that need to perform a task when flying over a populated area. They propose a risk assessment technique and bi-objective methods to optimize the low-risk and minimum flight time problem. In terms of performance, the best method in [7] is a network optimization used to generate initial solutions (feasible paths) that are then improved by a local and greedy approach. However, the generated paths lack of a proper automatic avoidance of no fly zones due to orography or other obstacles.

The work presented in this paper describes a risk evaluation tool embedded into a standard path planning software for RPAS that is able to create a risk map depending on flight altitude, elevation of the overflown area, population density distribution and physical parameters of the RPAS. The paths (a list of waypoints) will be generated using the following graph search algorithms:

- A*, a well-known graph search algorithm that is able to perform real time path finding.
- Θ*, which is derived from A*; it creates more realistic paths with less waypoints than A*, but it requires higher computational cost.
- RA*, a modified version of A* able to minimize the probability of occurring in catastrophic failures.

In order to better contextualize the path planner described in this article, this section will list some of the related works and off–the–shelf path planning tools.

World Wind is an open–source virtual globe first developed by NASA in 2003 for use on personal computers and then further developed with the open source community since 2004.

Beside UAV path planning, World Wind can accomplish several tasks. For example, this application is used to support the search and rescue mission operations in the definition of the most probable impact/landing area of a missing plane. Using last known positions from ground radar and other sources, mission planners are able to redefine the aircraft path and then studying its relative position to the ground by any point of view.

Mission Planner is a ground station application for the ArduPilot open source autopilot project. The autopilot works for airplane, multi–copter and rover configuration and can be used as control supplement tool for the vehicle. The mission planner can load the firmware into the autopilot board, setup vehicle parameters (such as PID controller gains), plan autonomous missions into the autopilot by point–and–clicking the waypoints on the planner map, monitor the aircraft status while in operation and record telemetry logs.

UAV Planner is a planning tool that provides automated path planning and sensor tasking for unmanned aerial vehicles. UAV Planner allows operators to model their UAV systems and perform operational scheduling and analysis. The main features include:

- Aircraft configuration
- Automated path planning
- Automated image collection planning
- 2D and 3D maps using Google Earth
- Manual path planning
- In–flight reconfiguration

The *mdCockpit* application software is developed by MicroDrones to enhance the use of small UAVs and support them during operations. It has three main functions: waypoint editor for flight planning, automated telemetric data downlink, and a module post processing data logs. In addition, it can also adjust the parameters of the aircraft. Planned operations can be saved and then loaded in the aerial vehicle at any time. Flight can be also planned using satellite images from Google Earth. To facilitate route planning, the waypoint editor allows images of maps to be displayed in the background. MDCockpit can read maps in BMP and JPEG formats and is able to generate KML files, with which routes can be displayed in Google Earth.

PCube is a 2D and 3D path planner developed at Politecnico di Torino. This software is able to create a list of waypoints to upload onto the autopilot. It features different algorithms for automated path planning but it also allows the user to manually generate paths.

Two versions of the software are available, both implemented in Matlab language: PCube 1. 1 works in 2D environment, PCube 3D performs also three dimensional path planning.

This paper aims to describe *Javacube*, a path planning software able to generate both manual and automatic routes on 2D maps. The software also incorporates a tool to perform the risk analysis of a RPAS as required by the last draft of Italian RPAS legislation [1]. Risk analysis is embedded into JavaCube's automatic path creation and is based on the previously mentioned RA* algorithm.

The work is organized as follows. Section 2 describes the new Italian RPAS regulation and the relative risk evaluation method. Section 3 presents the path finding algorithms, with particular focus on the proposed RA* algorithm. Section 4 shows the JavaCube framework and Section 5 contains the conclusions and future work.

2. Italian Normative Framework

Italian regulation divides RPAS in two categories:

- RPAS with a mass less than 25 kg.
- RPAS with a mass greater than 25 kg and no more than 150 kg.

Operations can be critical or non–critical. According to the last Italian SAPR regulation draft (July 2015), non–critical scenarios must be carried out in visual line of sight (VLOS) with RPAS not flying over urban areas, schools, hospitals, stadiums or any place that could host an even temporary crowd.

The operators who intend to fly a RPAS in a non–critical scenario must release to ENAC (Ente Nazionale per l'Aviazione Civile, the Italian civil aviation authority) a declaration that proves the non–criticality of the operation, together with a documentation evaluating the risk of flying over a specific area.

A critical operation occurs when it does not meet the above definition of non–critical scenario.

In this case the operator is asked to require and obtain an authorization from ENAC by presenting a list of documents that includes the results of the risk analysis performed to assess the level of safety of the operations. In any case, for both non–critical and critical scenarios a risk analysis is necessary and must be provided to ENAC.

In scientific literature there are different methods to determine the level of risk of an aircraft. Most of them are based on the computation of a maximum acceptable probability, a quantity that estimates the time a UAV can fly without any catastrophic failure (i.e. causing serious damage to people or buildings).

In recent times ENAC adopted a method used by FAA [8] to compute the risk of space debris during reentry. This method computes the maximum acceptable probability as:

$$P = \frac{E_c}{A_c D_p} \tag{1}$$

where:

- E_c is the expected casualty (10^{-6} as a safety objective) and it measures the number of victims per flight hours.
- A_c is the impact area and depends on RPAS dimensions, standard human size and the glide angle of the RPAS at impact.
- D_p is the local population density.

This method is used for space debris reentry and provides results too far from the behavior of a RPAS. The reciprocal of the maximum acceptable probability is expressed in hours and represents the maximum time an aircraft can fly without occurring in catastrophic failures. The FAA method provides a minimum time that is over a thousand hours and this is not plausible for small UAVs that are kept in service with uncertified spare parts and require a more frequent maintenance (about in the range 1–10 h).

The method presented by Dalamagkidis et al. [9, 10] takes into account the aircraft kinetic energy and the sheltering factor. This latter estimates the presence of natural obstacles or buildings that could mitigate the damage caused by the crash of an RPAS. In this case the maximum acceptable probability is calculated as:

$$P = \frac{N}{A_{exp} D_p P_f} \tag{2}$$

where:

- N is the safety objective (10^{-6} victims per flight hour).
- D_p is the local population density.
- A_{exp} is area at impact.
- P_f is probability of fatal injuries to people exposed to the crash.

P_f is the term depending on kinetic energy and sheltering factor $P_s \in (0, +\infty)$ (for complete details, see [9]).

This method shows values more compatible to an RPAS: the reciprocal of the maximum probability varies in the range $10 \div 100$ h depending on the population density of the overflown area.

To increase the maximum acceptable probability of the FAA method, Guglieri et al. [11] propose to calculate the maximum acceptable probability as:

$$P = P_{FAA} \cdot \frac{1}{G} \tag{3}$$

This solution introduces the G probability factor (in case of nonhomogeneous population density areas) that takes into account that RPAS may crash in a specific area. The introduction of the probability factor G has halved the term 1/P.

The sheltering factor in P_f is an absolute real number. Guglieri et al. [11] propose to write this value as a percentage term according to Table 1:

In this paper the maximum acceptable probability is computed as in [9] with the percentage sheltering factor proposed in [11]. Next sections will focus on the path planning strategies adopted in Javacube together with the implementation of the risk analysis.

Table 1. *Sheltering factor.*

Sheltering	Area
0 %	No obstacles
25 %	Sparse trees
50 %	Trees and low buildings
75 %	High buildings
100 %	Industrial area

3. Path Finding Algorithm

Path planning strategies are based on the optimization of some parameters by using different approaches.

Probabilistic algorithms are effective when the optimization parameters evolve with time due to uncertainties of flight conditions, environment or mission tasks.

These algorithms generate a probability distribution depending on the parameter to be optimized and they implement statistic techniques to find the most probable path that optimizes this parameter [12].

Algorithms based on graph search theory come from the field of computer science where they have been extensively exploited to optimize the exchange of data in computer networks.

Nowadays they are also used for path planning of mobile robots and more recently for the flight planning of UAVs.

The graph search algorithms divide the domain of the territory into a grid of nodes (or cells). The global solution (the optimum path) consists of a subset of these nodes and it is found by minimizing a cost function that often depends on the distance from the starting node and the final node.

The founder of such algorithms is the Dijkstra algorithm [13]. More efficient algorithms, as D*, A*, and the more recent Theta* [14], are all based on the Dijkstra algorithm.

The graph search algorithms are very efficient in terms of computational cost and convergence towards the global optimum solution. However, as the probabilistic algorithms they do not take into account the physical parameters of the aircraft (mass, size, speed) and then can lead to solutions too far from reality as they do not respect the physical constraints of the vehicle.

Finally, there are path planning methods based on evolutionary algorithms (EA).

EAs start from a set of possible solutions (possible paths) obtained using a greedy approach (for example with a greedy version of graph search algorithms) that evolves into new generations in order to optimize a suitable fitness function. EAs do not always find the global optimum, and generally have a computational cost and execution time higher than graph search algorithms. In recent years, multi–objectives EAs [15] and parallel EAs [16] have been developed: they improve the solution in terms of optimization of the fitness function but the execution time still remains high.

Graph search algorithms have very short execution times. The Javacube software implements two algorithms for 2D path planning: Theta* (for details on the latter, see [14]) and RA*, a modified version of A* that takes into account the risk analysis described in section 2.

Classic A* cost function can be computed for each node of the graph and can be written as:

$$f_{A^*}(i,j) = [\,g(i,j) + h\,(i,j)] * \tau + (1-\tau) * e(i,j) \qquad (4)$$

where:

- $g(i,j)$ is a measure of the distance between the (i,j) node and the start node.
- $h(i,j)$ is a measure of the distance between the (i,j) node and the final node.
- $e(i,j)$ is a measure of the risk of the (i,j) node; black tiles form the no–fly zone as they represents areas with higher altitude than the aircraft altitude. The black tiles $e(i,j)$ value is the l_1 norm (Manhattan norm) between the start and the final node. The $e(i,j)$ value for non–black tiles is null since they represents feasible areas.
- τ is a weight used to balance the effect of the high risk zones and it is set to 0.1. This means that the cost function of a black tile is 0.9 times higher than the cost of a white tile and the UAV should avoid high risk (black) zones.

See [14] for further information and pseudocode of A* algorithm.

The modified cost function presented in this paper is written as:

$$f_{RA^*}(i,j) = f_{A^*}(i,j) + w * r(i,j) \qquad (5)$$

Where:

- $f_{A^*}(i,j)$ is the classic A* cost function.
- $r(i,j)$ is the reciprocal of the maximum acceptable probability computed in previous paragraph.
- w is a weight used to tune the effect of the density population risk in the evaluation of the cost function.

The new additional term in (5) depends on the density distribution of the overflown area and takes into account the risk analysis evaluation described in section 2.

Next section will show the user interface of the JavaCube path planner with particular focus on the automatic path algorithms discussed so far.

4. Path Planning Software

Figure 1 shows the main frame of the program.

The left window contains a map that can be loaded by the user. When the map is opened a dialog box appears and this allows the user to set the flight parameters.

The right window contains the risk map: as the altitude of the black area is higher than the altitude of the aircraft, this zone represents an obstacle and should not be overflown.

Currently the software is able to only 2D path planning. This means that the aircraft is assumed to fly at a constant altitude along the whole route; future developments of the software will include also a 3D path planning strategy.

The definition of the non–feasible region is possible thanks to the presence of a digital elevation model (DEM) file associated with the map. The remaining feasible area is colored according to a yellow–colored scale depending on the level of maximum acceptable probability computed using equation (2). The reciprocal of the maximum acceptable probability represents the hours a UAV can operate without incurring in catastrophic failures.

The yellow scaled map is built within a window where the map is superimposed on a 10x10 grid (Figure 2). On the right panel there are text fields in which the user can enter data about the UAV (mass, size) and the highest population density of the overflown map (i.e. urban zones). Other areas (suburbs, rural, industrial areas) have a density which is a fraction of the highest density. The panel contains also a legend with five radio–buttons that can be selected by the user in order to color the map according to the different values of population density. Each option is also associated with a percentage of sheltering factor, according to table 1.

By confirming the operation the windows is closed and the main frame reappears together with the yellow–color scaled risk map in the right panel.

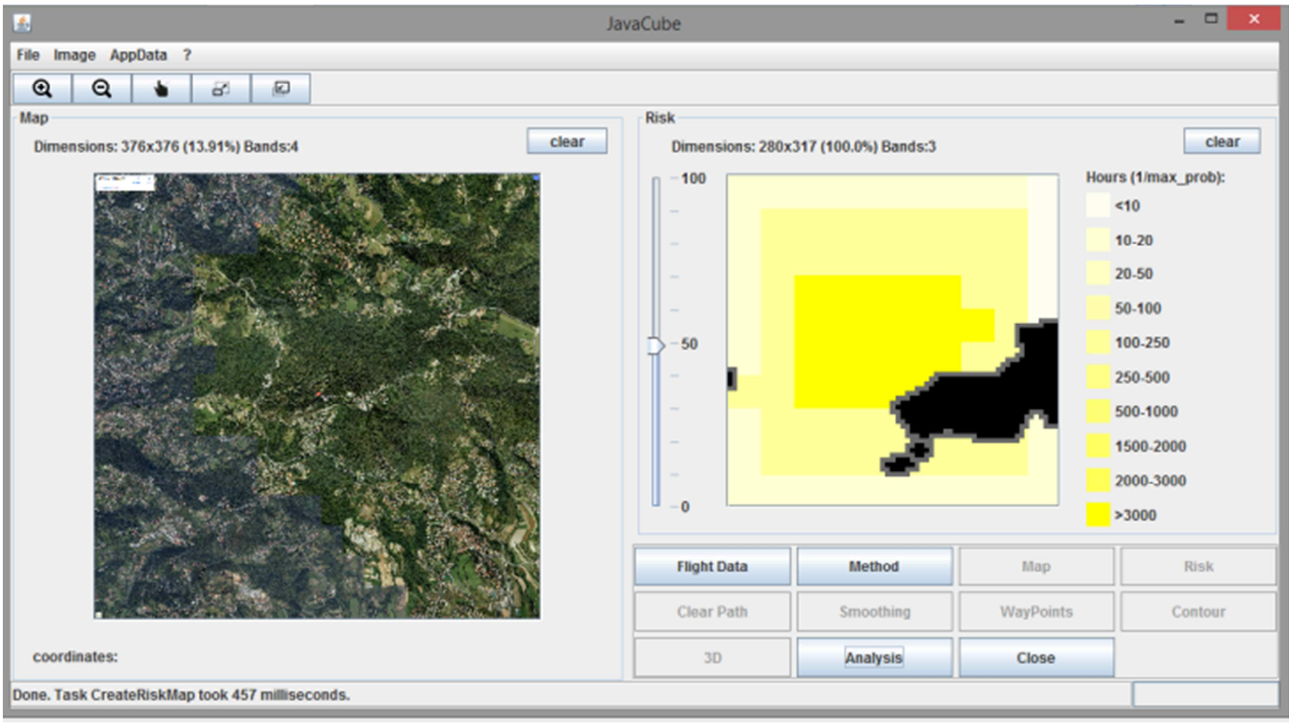

Figure 1. *JavaCube main frame.*

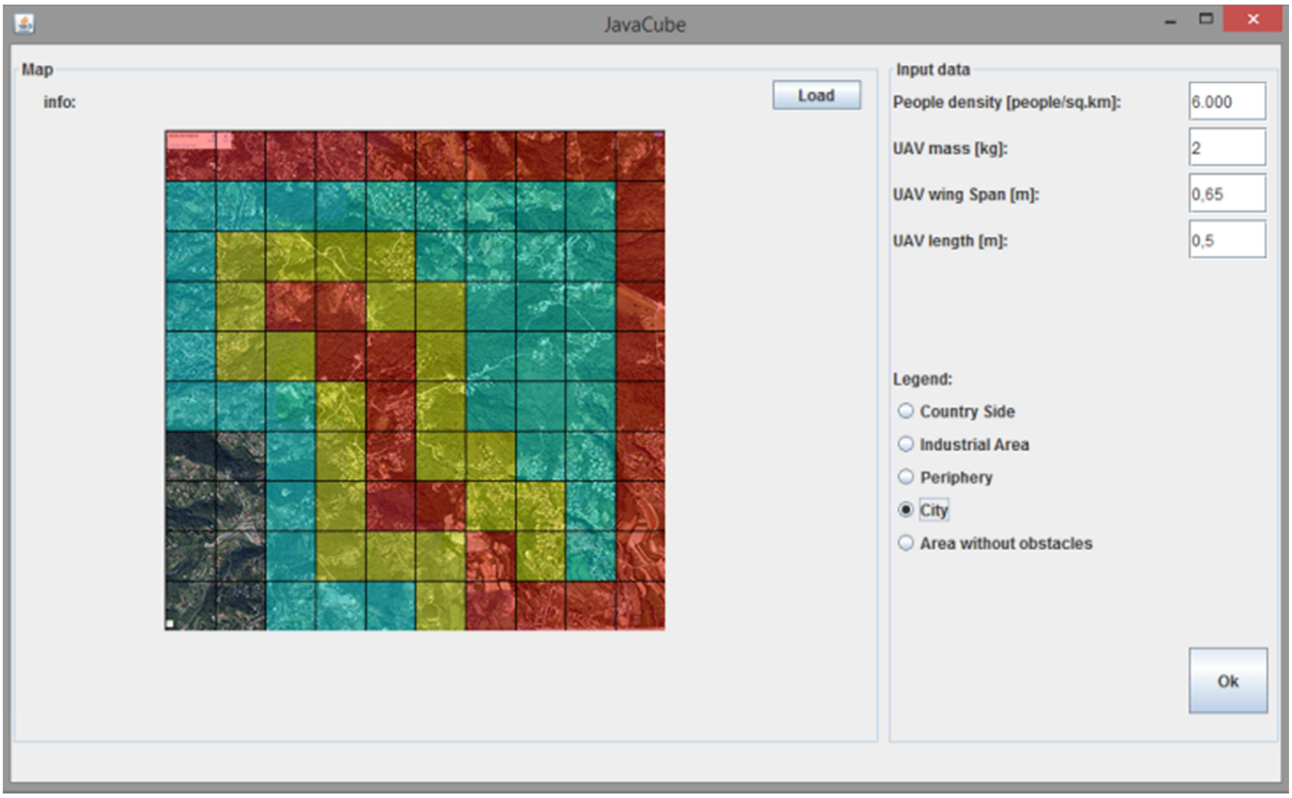

Figure 2. *Window for the definition of the risk analysis.*

Table 2. *Path methods.*

Methods	Paths
Loop	Geometric (square, rectangle, butterfly), Point & Click
Point to Point	Point & Click, Minimum Risk (RA*, Theta*)
Strip Mode	Grid paths

The "Method" button allows the operator to choose among different paths (see Table 2), including:

- geometric loop path (square, butterfly);
- grid flight paths ("*Strip mode*");
- manual paths ("*Point and click*"): the user chooses the waypoints that will form the path;
- automatic paths: the RA* and Theta* algorithm ("*Minimum risk*").

The *Loop* method is used for the generation of standard closed paths (square, rectangular or butterfly paths) or manually closed paths.

Strip Mode method allows the user to create grid–shaped paths. This algorithm depends on the size of the camera sensor onboard the UAV, its focal length and the overlap percentage of two subsequent pictures taken at a specific flight altitude (Figure 3).

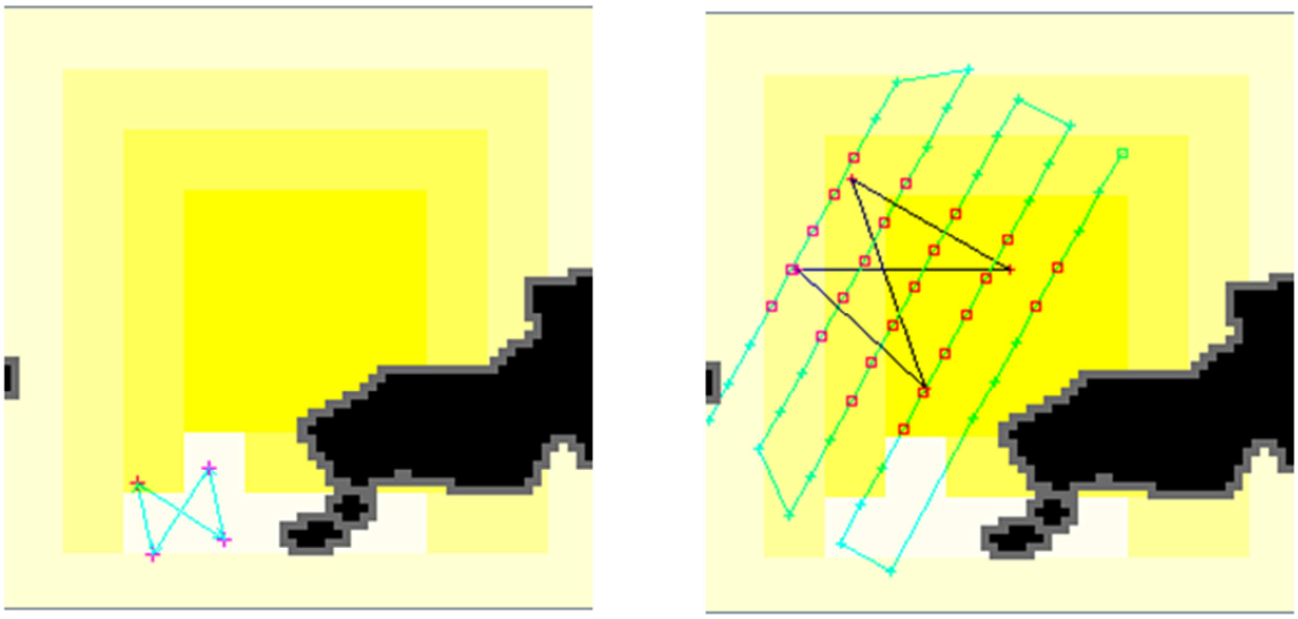

Figure 3. *Butterfly path (left). Strip mode path (right).*

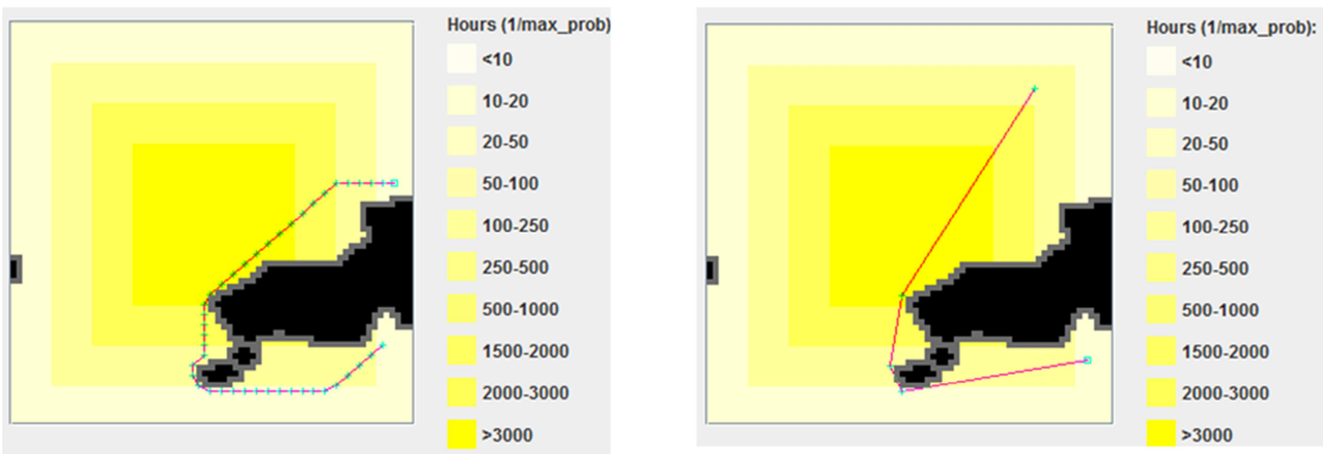

Figure 4. *A* path (left). Theta* path (right).*

Figure 4 (left) shows a path computed using classic A*. The algorithm consists of a main loop in which, at each iteration, the eight neighbor cells of a specific cell are expanded (Figure 5). The cost function is computed using (4) for each expanded cell that are now stored in an *open list*. At the following iteration the minimum cost cell is extracted from the open list and put into a *close list*. The other cells within the open list are expanded, i.e. the cost function of the eight neighbor cells is evaluated for each cell within the open list. The loop ends when the goal cell is finally expanded. The path is generated backwards extracting the cells from the close list.

Figure 4 (right) shows a Theta* path. Although similar to

A*, at each iteration the algorithm verifies the line of sight between the expanded cell and the parent cell. In this way Theta* is able to generate a more feasible path with less waypoints even though the execution time is higher than A* algorithm.

Even in this case the loop ends when the goal cell is expanded and the path is created backward from the goal cell to the start cell.

The algorithm RA* has an additional weighted term to the A* cost function. The weight can be changed using the slider in the right window of the main frame (Figure 1). When the slider is at 0% the additional term is null and the path is created through classic A*; the slider at 20% (figure 6 – left) makes the additional term of the same order of magnitude of the classic cost; at 100% the weighted term is one order of magnitude greater than the A* cost function: in this case (figure 6 – right) the path is forced to avoid the high density population zones thus minimizing the risk of the mission.

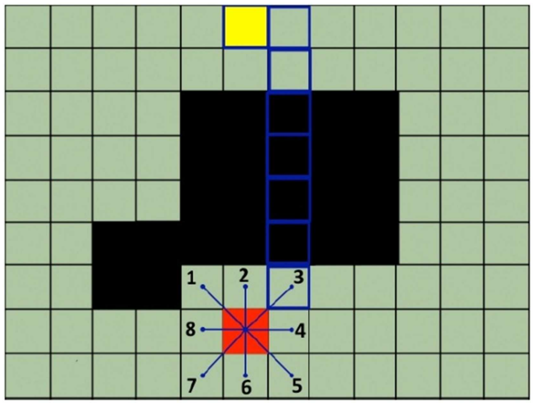

Figure 5. *Cell expansion in A* algorithm.*

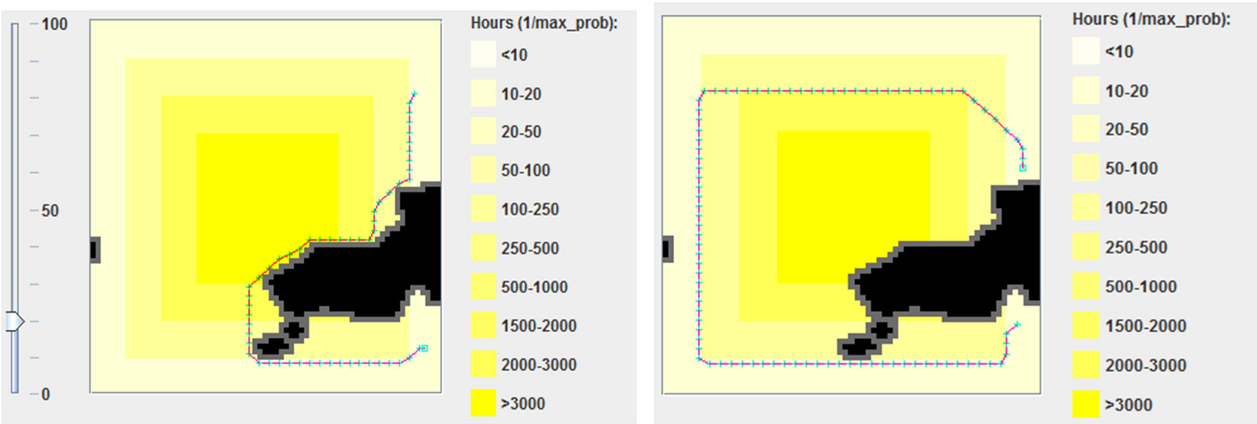

Figure 6. *RA* Algorithm.*

5. Conclusions

This work is intended to meet the requirements of the RPAS Italian legislation by proposing a risk analysis tool to evaluate the level of risk of RPASs. The risk analysis is necessary to obtain the permission to operate in both critical and non–critical scenarios.

The analysis is computed by evaluating the maximum acceptable probability, or alternatively its reciprocal *1/P* that expresses the hours an aircraft can fly without occurring in catastrophic failures. Acceptable 1/P values for RPAS lie in the 10–100 hours range. This work proposes a risk analysis tool embedded in a Java flight planner that allows the user to perform different tasks: it is possible to load a map and its relative DEM file in order to generate a risk map containing information on non–feasible areas (as at higher altitudes than the aircraft altitude) and feasible areas through the computation of the maximum acceptable probability of section 2.

The user can choose from different manual or automatic paths. This work is mostly focused on the implementation of an A* based algorithm whose cost function incorporates a term that is an estimate of the aircraft level of risk. The algorithm has been tuned, and now the user can choose whether to minimize the aircraft distance between start and final cells by creating a classic A* path or to perform a risk analysis oriented path estimation by using the new RA* algorithm.

Future works will include the implementation of the risk analysis in other types of algorithms (e.g. Theta*), together with a benchmark that will consider the algorithm execution time and global optimum solution. 3D path planning algorithms will be also implemented to get paths with variable altitude. Finally, JavaCube will be tested on real missions to verify its accuracy and effectiveness.

References

[1] Ente Nazionale per l'Aviazione Civile. *Mezzi Aerei a Pilotaggio Remoto.* 2015.

[2] Unmanned Aerial Systems in European Airspace. *Safety Aspects of Civil RPAS Operations*. 2013.

[3] JAA/EUROCONTROL RPAS Task Force. *A concept for European Regulations for Unmanned Aerial Vehicles (RPAS)*. 2004.

[4] NATO. *UAV Systems Airworthiness Requirements*. 2007.

[5] E. Pastor, M. Perez-Batlle, P. Royo, R. Cuadrado and C. Barrado. *Preparing for an Unmanned Future in SESAR. Real-time Simulations of RPAS Missions*. Proceedings of III SESAR Innovation Days, Stockholm, Sweden, 2013.

[6] E. Pastor, M. Perez-Batlle, P. Royo, R. Cuadrado and C. Barrado. *Real-time Simulations to Evaluate the RPAS integration in Shared Airspace*. Proceedings of IV SESAR Innovation Days, Madrid, Spain, 2014.

[7] E. Rudnik-Cohen, J. W. Hermann, S. Azarm. *Risk-based Path Planning Optimization Methods for UAV over Inhabited Areas*. ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Boston, USA, 2015.

[8] Federal Aviation Administration. *Expected Casualty Calculations for Commercial Space Launch and Reentry Missions*. FAA–AC431.35, 2000.

[9] K. Dalamagkidis, K.P. Valavanis and L.A. Piegl. *Evaluating the Risk of Unmanned Aircraft Ground Impacts.* In Proceedings of the 16th Mediterranean Conference on Control and Automation, Ajaccio, France, 2008.

[10] K. Dalamagkidis, K.P. Valavanis and L.A. Piegl. *On Integrating Unmanned Aircraft Systems into the National Airspace System.* Springer ed., 2012.

[11] G. Guglieri, F. Quagliotti and G. Ristorto. *Operational Issues and Assessment of Risk for Light UAVs.* Springer ed., 2012. Journal of Unmanned Vehicle Systems, Vol. 2, n. 4, 2014.

[12] L.F. Bertuccelli and J.P. How. *Robust UAV Search for Environmentas with Imprecise probability Maps*. Proceedings of IEEE Conference of Decision and Control, Seville, Spain, 2005.

[13] D. Ferguson and A. Stentz. *Using interpolation to improve path planning: The Field D* algorithm.* Journal of Field Robotics, Vol. 23, n. 2, pp. 79–101, 2006.

[14] K. Daniel, A. Felner, S. Koenig and A. Nash. *Theta*: Any–Angle Path Planning on Grids.* Journal of Artificial Intelligence Research, Vol. 39, pp. 533–579, 2010.

[15] S. Mittal and K. Deb. *Three–Dimensional Offline Path Planning Strategies for UAVs Using Multi–objective Evolutionary Algorithms.* Congress on Evolutionary Computation, Singapore, 2007.

[16] D. Jia and J. Vagners. *Parallel Evolutionary Algorithms for UAV Path Planning.* AIAA 1st Intelligent Systems Technical Conference, Chicago, USA, 2004.