A Risk-Aware Path Planning Strategy for UAVs in Urban Environments

**Journal of Intelligent and Robotic Systems manuscript No.**
(will be inserted by the editor)

# A Risk-aware Path Planning Strategy for UAVs in Urban Environments

**Stefano Primatesta · Giorgio Guglieri · Alessandro Rizzo**

Received: date / Accepted: date

**Abstract** This paper presents a risk-aware path planning strategy for Unmanned Aerial Vehicles in urban environments. The aim is to compute an effective path minimizing the risk to the population, enforcing safety of flight operations over inhabited areas. To quantify the risk, the proposed approach uses a risk-map that associates discretized locations of the space with a suitable risk-cost. Path planning is performed in two phases: first, a tentative path is computed off-line on the basis on the information related to static risk factors; then, using a dynamic risk-map, an on-line path planning adjusts and adapts the off-line path to dynamically arising conditions.

Off-line path planning is performed using riskA*, an ad-hoc variant of the A* algorithm, which aims at minimizing the risk. While off-line path planning has no stringent time constraints for its execution, this is not the case for the on-line phase, where a fast response constitutes a critical design parameter. We propose a novel algorithm called *Borderland*, that uses the *check and repair* approach to rapidly identify and adjust only the portion of path involved by the inception of relevant dynamical changes in the risk factor. After the path planning, a smoothing process is performed using Dubins curves. Simulation results confirm the suitability of the proposed approach.

**Keywords** path planning · unmanned aerial vehicles · risk-map · risk-aware path planning · riskA* · Borderland algorithm

S. Primatesta
Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy

G. Guglieri
Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy

A. Rizzo
Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
E-mail: alessandro.rizzo@polito.it

## 1 Introduction

Unmanned Aerial Vehicles (UAVs) are gaining momentum in recent years and their extensive use has induced the rapid growth of related research areas. UAVs have been successfully applied in several applications, both in military and in civil fields, such as security surveillance, search and rescue, environmental monitoring, mapping, to name a few [2]. In particular, the use of UAVs in urban environments is growing and, in the very near future, UAVs will be involved in Smart Cities [31, 20]. For this reason, it will be mandatory to enhanced safety of flight operation.

Path planning is one of the key elements to provide autonomy to UAVs in the execution of the missions, by defining the set of waypoints to reach a destination, while satisfying some optimality criterion [13]. Path planning has been widely studied and a large number of methods have been proposed in the last decades. Starting in the late 50s with the Dijkstra algorithm [9], graph search algorithms have been widely used in path planning. One of the most commonly used algorithms is A* [17], which complements the logic of graph search with a heuristic component for the computation of the cost function. Several A*-based algorithms have been developed for dynamic [40] and anytime [27] path planning, and in high dimensional environments [18,19].

Another popular family of path planning algorithms comprises sample-based techniques, which explore the search space through a sampling scheme. Widely diffused approaches are Probabilistic Roadmaps (PRM) [24] and Rapidly-exploring Random Trees (RRT) [25]. Especially RRT is very popular, and many RRT-based algorithms have been developed. The most popular is RRT* [22], a near-optimal version of RRT, also used to perform kinodynamic path planning [23]. Sample-based algorithms have been also used for UAVs [44].

A wide variety of path planning algorithms specifically dedicated to UAVs has been presented in the literature. In [29], the authors propose a kinodynamic path planning algorithm with collision avoidance using the Closed-loop RRT. Other techniques are based on evolutionary algorithms (EAs) [36], as well as reinforcement learning approach [46]. Often, path planning for UAVs uses techniques based on an explicit 3D description of the environment [8,45].

When autonomous vehicles move in uncertain or populated environments, risk should be accounted for to produce an effective path planning. In [43] the authors propose a dynamic path planning for UAVs taking into account static and dynamic threats. In [7,42] risk-maps are used to compute the path with minimum risk. Risk-aware path planning is a common problem and concerns also mobile robots [12] and autonomous underwater vehicles (AUVs) [33]. In general, risk-aware path planning takes into account the risk from the UAV point of view, i.e. it minimizes the risk of collision with obstacles and other vehicles [38,43]. However, when UAVs operate in urban environments it should be mandatory to consider the risk to people on the ground. In [39] the authors propose a path planning algorithm for emergency landing, in order to avoid populated areas. Another interesting work in [37] introduces a path planning technique minimizing the trade off between risk to the population and flight time. In [14], Guglieri *et al.* propose a path planning strategy for UAVs using the RA* algorithm, a modified version of A* that takes into account the risk of flying over a specified area.

The risk to the population is defined by risk analyses. A common risk assessment approach defines the risk as the probability of lethal incidents when the UAV flies over a populated area [3,5,6,15].

### 1.1 Current work

This work introduces a new perspective in path planning, whereby the definition of optimal path accounts for both static and dynamic risk factors for the population on ground along the route. The proposed path planning algorithm takes a so-called risk-map in input. A risk-map is a dynamical location-based map, where each location is associated with a specific risk-cost that quantifies the risk of flying over that location [6,15]. The proposed path planning approach is composed by an off-line and an on-line path planning phase.

First, the off-line path planning aims to solve an optimal path planning problem. Given a risk-map, a starting and a final point, the off-line path planning seeks for an optimal global path that avoids obstacles and no-flight areas, computed minimizing the risk-cost defined by the risk-map. The off-line path is computed before the mission starts and, in general, when the UAV is still on the ground. Thus, the off-line path planning is not, in general, a time-critical activity.

The on-line path planning aims to repair and adapt the off-line path in real-time, according to relevant information emerging from a dynamic risk-map. During the trajectory execution, in fact, the risk-map changes when unanticipated obstacles and other risk factors appear. For this reason the on-line path planning only focuses on the portion of path not already executed, i.e., from the current position of the UAV to the target point. Unlike off-line planning, the on-line one is time-critical, since the UAV is already flying en-route and needs to react promptly to dynamically changing conditions.

After path planning, a fast Path Smoothing procedure based on Dubins curves [10] is applied to the obtained path. This procedure is necessary, in order to transform the theoretically-obtained path in a flyable one. Then, the path is handed over to a UAV Control System [41]. Figure 1 illustrates the architecture of the proposed approach.

Guidelines for the development of our approach have been introduced for the first time in [34], as a part of a Cloud-based framework for risk-aware intelligent navigation for UAVs in urban environments.

This paper is organized as follows. Section 2 provides an overview on risk assessment and risk-maps. Then, in Section 3, the off-line path planning is illustrated in detail, focusing on the problem formulation and on riskA*, our proposed algorithm. Section 4 describes the on-line path planning, achieved through the novel Borderland algorithm. Then, in Section 5 the Path Smoothing procedure using Dubins curves is described. Finally, in Section 6 the proposed method is validated by numerical results. Our conclusions are drawn in Section 7.

## 2 Risk-map

This section provides an overview of basic concepts about the definition of the risk-map.

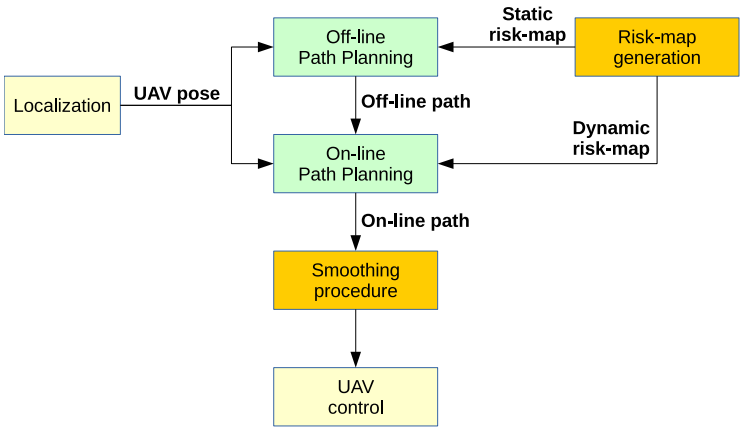



**Fig. 1** The main architecture of proposed risk-aware path planning approach.

In path planning problems, the construction of a map has a critical role, since it defines the search space, i.e., the space with all the possible configurations where the path planning algorithm searches for the solution [26]. The risk-map used in this work is a dynamic one that quantifies the risk of flying over a given position, including the presence of obstacles. The risk-map describes a discretized 2D space, i.e. flight at constant and fixed altitude is considered. A location-based map is assumed, where the 2D space is sampled in a grid fashion, and each location is associated with a number, quantifying the risk level [15]. Thus, a map $M$ consists of a $n \times m$ matrix of locations:

$$M = \begin{Bmatrix} r_c(p_{1,1}) & r_c(p_{1,2}) & \cdots & r_c(p_{1,m}) \\ r_c(p_{2,1}) & r_c(p_{2,2}) & \cdots & r_c(p_{2,m}) \\ \vdots & \vdots & \ddots & \vdots \\ r_c(p_{n,1}) & r_c(p_{n,2}) & \cdots & r_c(p_{n,m}) \end{Bmatrix}, \tag{1}$$

where each element $r_c(p_{i,j})$ is a square cell centered in the location $p_{i,j}$ and with a specific dimension. The dimension of the cell, i.e. the risk-map resolution, is defined considering the quality of the source data used to generate the risk-map. Each cell $r_c(p_{i,j})$ has a specified risk-cost expressed by a real number in the range 0-1, according to its risk value and specific to the spatial co-ordinates $p_{i,j}$. Notably, 0 corresponds to a zone with no risk, while 1 defines a zone with the highest flight risk, i.e. a no-flight zone.

The risk value is defined according to [3,5,6,15]. In particular, the risk is the probability of lethal incidents, defined as succession of three conditional events: *(i)* the loss of control of the vehicle with uncontrolled crash on the ground, *(ii)* the impact with someone, and *(iii)* a fatal injury to the person that has been hit. This probability is therefore defined as follows:

$$P_{\text{casualty}}(p_{i,j}) = P_{\text{crashing}} \times P_{\text{impact}}(p_{i,j}) \times P_{\text{fatality}}(p_{i,j}) \tag{2}$$

The probability of crashing $P_{\text{crashing}}$ is the probability of loss of control of the UAV with the uncontrolled crash on the ground. This term is expressed as a rate

per hour ($h^{-1}$) and it generally coincides with the failure rate of the aircraft. Its value depends on the vehicle type and its reliability. In [6], a constant value in the range $[10^{-6}, 10^{-9}]$ $h^{-1}$ is used for generic unmanned aircraft systems, while in [4] values in the range $0.5 - 10$ $h^{-1}$ are established for Micro Air Vehicles (MAVs).

The probability of impact with someone $P_{\text{impact}}(p_{i,j})$ considers the crashing area $A_{\text{crash}}$ and the population density $\rho$ relative to the position $p_{i,j}$:

$$P_{\text{impact}}(p_{i,j}) = \rho(p_{i,j}) \cdot A_{\text{crash}}. \tag{3}$$

The crashing area is computed according to the method used in [6], taking into account the dimensions of the UAV and of the average human being.

The probability of fatal injury to someone $P_{\text{fatalty}}(p_{i,j})$ is defined using the approach proposed in [6], accounting for the sheltering factor relative to the position $p_{i,j}$ and the kinetic energy at impact. The sheltering factor defines how the population is exposed to the impact of the UAV. In fact the presence of obstacles in the impact area can reduce the probability of fatal injuries, because obstacles can absorb the impact energy and shelter to debris.

In order to ensure an appropriate level of safety, a maximum value of $P_{\text{casualty}}$ is specified. According to the "Equivalent Level of Safety" defined in [6,15], an appropriate and conservative value of *maximum acceptable risk* is $10^{-6}$ $h^{-1}$. The risk value is computed for each location $p_{i,j}$ in the risk-map, then it is normalized in the range 0-1, whereas 1 defines locations with a risk-value greater then the maximum acceptable risk. Obstacles at the flight altitude and no-fly zones enforced by National aviation agencies are accounted by forcing the corresponding risk-cost to 1.

## 3 Off-line Path Planning

The off-line path planning takes as input the risk-map and solves a risk-aware path planning problem before the beginning of the flight mission. Since the risk-map is defined at fixed flight altitude, a two-dimensional path planning problem is here tackled.

Here, off-line path planning is realized through an A*-based [17] graph search algorithm, called riskA*, which optimizes the path by trading off risk-cost and path length.

### 3.1 Problem formulation

Let $C \subseteq \mathbb{R}^2$ be a continuous search space of a path planning problem. As in many other applications [32], $C$ is discretized into a discrete space $X$, on which the risk-map will be constructed, and taking into account the risk-map resolution. Each state $x \in X$ is a discrete location in the discrete search space. With a slight abuse of notation, here and henceforth we will refer to $x$ as a state of search space, location of risk-map $M$ or a node of a search grid graph.

The obstacle region $X_{\text{obs}} \subseteq X$ is the set of locations in which flight is forbidden. Thus, the associated cost is equal to 1, as described in Section 2. The set $X_{\text{free}} = X \setminus X_{\text{obs}}$ contains the remaining navigable locations. The initial and final locations are denoted $x_{\text{start}}, x_{\text{goal}} \in X_{\text{free}}$. Let $\Sigma$ be the set of all paths, where a

single path $\sigma$ is a sequence of connected locations $x$ in the search space $X$. The path planning algorithm searches for an optimal path $\sigma^*$ from $x_{\text{start}}$ to $x_{\text{goal}}$ in $X_{\text{free}}$ that minimizes a given cost function $f : \Sigma \to \mathbb{R} \geqslant 0$. Hence, the optimal path is the solution of the following program:

$$\begin{aligned} & \sigma^* = \arg\min_{\sigma \in \sum} f\big(\sigma(s)\big) \\ \text{subject to} \quad & \sigma(0) = x_{\text{start}} \\ & \sigma(1) = x_{\text{goal}} \\ & \forall s \in [0,1]\,,\ \sigma(s) \in X_{\text{free}}. \end{aligned} \tag{4}$$

### 3.2 RiskA* algorithm

The riskA* algorithm is based on well-known A* [17]. Similarly to A* algorithm, the input of riskA* is a graph $M$ composed by nodes and edges. Here, we specialize the algorithm by considering a two-dimensional grid graph, where each location corresponds to a graph node, and each portion of path between two nodes corresponds to a graph edge. The output of riskA* is a back-pointer path, which is a sequence of nodes starting from the goal and tracing back to the start.

Similarly to A*, the general idea of riskA* is to move through the graph minimizing the cost function $f(x)$:

$$f(x) = g(x) + k \cdot h(x), \tag{5}$$

where $g(x)$ is the motion cost of the path from node $x$ to the start node $x_{\text{start}}$, the constant $k$ is the adjustment variable, and $h(x)$ the heuristic cost, i.e., the estimated motion cost of the best path between $x$ and $x_{\text{goal}}$. Thus, $f(x)$ is the estimated motion corresponding to the shortest path from $x_{\text{start}}$ to $x_{\text{goal}}$, passing through node $x$.

Differently from traditional A*, riskA* considers the risk-cost in the cost function $f(x)$. Given a generic node $x_{\text{n}}$, the motion cost $g(x_{\text{n}})$ is:

$$g(x_{\text{n}}) = \int_{x_{\text{start}}}^{x_{\text{n}}} r_{\text{c}}(x)dx, \tag{6}$$

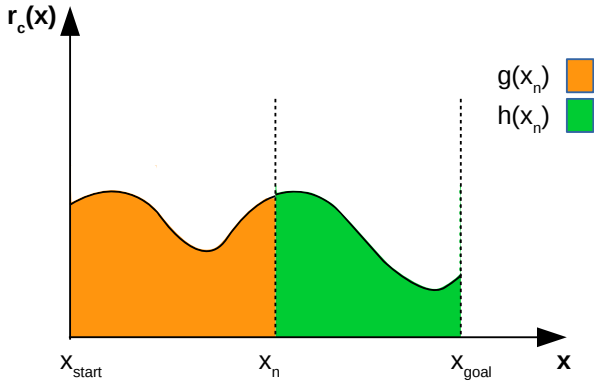
where $g(x_{\text{n}})$ is the integral of the risk-cost between the initial state $x_{start}$ and state $x_n$. Function $r_{\text{c}}(x)$ is the risk-cost associated with the path. Similarly, the heuristic function cost $h(x_{\text{n}})$ is:

$$h(x_{\text{n}}) = \int_{x_{\text{n}}}^{x_{\text{goal}}} r_{\text{c}}(x)dx, \tag{7}$$

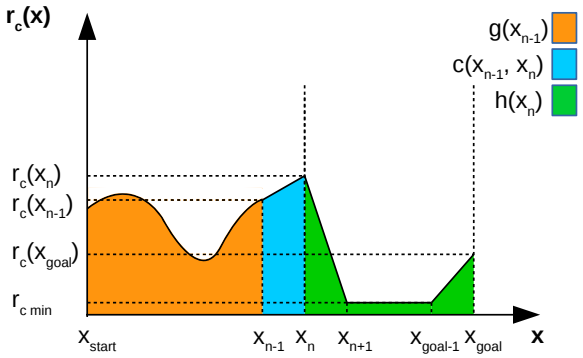
where $h(x_{\text{n}})$ is the integral of the risk-cost between state $x_n$ and the final state $x_{goal}$.

The risk-cost function takes values in $0 < r_{\text{c}}(x) \leq 1$. As a consequence, the motion cost is never equal to zero. Moreover, conservative considerations lead us to never consider the risk equal to zero.

Figure 2(a) shows the cost function $f(x)$ at step $n$. Note that $g(x)$ is the effective motion cost from the initial node, while $h(x)$ is the estimated motion cost until the final node. Hence, $g(x)$ and $h(x)$ are complementary along the path.




(a)




(b)

**Fig. 2** Graphical representation of the cost function $f(x)$. Given a generic state $x_{\mathrm{n}}$, in (a) the cost function is composed by the motion cost $g(x_{\mathrm{n}})$ and the heuristic cost $h(x_{\mathrm{n}})$. Similarly, in (b), the incremental step defined in Equations (8), (9), (10), (11) is illustrated.

The riskA* algorithm is described in Algorithm 1. Its inputs are $x_{\mathrm{start}}$ and $x_{\mathrm{goal}}$, and a grid graph $M$ related to the risk-map. In the same way of A*, two data structures are used, called $O$ (Open set) and a $C$ (Closed set). The Open set is used to store currently discovered nodes waiting to be evaluated. Set $O$ is a priority queue defined according to the estimated cost $f(x)$, such that an element with low cost is served before an element with high cost. The Closed set is the set of nodes already processed or invalid. The algorithm uses the same logic of A*. Important variants with respect to the original algorithm are in lines 6, 7, where if more than one node $x_{\mathrm{best}}$ with the same motion cost exist, the algorithm selects the node with the lower risk-cost. The riskA* algorithm generates a search tree, which, by definition, has no cycles.

For each node $x$ explored by riskA* the estimated cost function $f(x)$ is computed as described in Equations (5), (6), (7). Considering the discrete grid, the integral is calculated with an approximative and incremental method. Thus, $g(x_{\mathrm{n}})$

**Algorithm 1** riskA* algorithm

**procedure** RISKASTARSEARCH($x_{\text{start}}, x_{\text{goal}}, M$)
  Add $x_{\text{start}}$ to $O$
  Add all invalid nodes $x_{\text{invalid}} \in M$ to $C$
  **repeat**
    Pick $x_{\text{best}}$ from $O$ with $f(x_{\text{best}}) \leqslant f(x), \forall x \in O$
    **if** $\exists$ multiple $x_{\text{best}}$ **then**
      Pick the $x_{\text{best}}$ with lower $r_{\text{c}}(x_{\text{best}})$
    **end if**
    Remove $x_{\text{best}}$ from $O$ and add to $C$
    **if** $x_{\text{best}} = x_{\text{goal}}$ **then**
      **return** ReconstructPath($x_{\text{goal}}, x_{\text{start}}$)
    **end if**
    Expand $x_{\text{best}}$: for all $x_{\text{adj}} \in \text{Near}(x_{\text{best}})$ and $x_{\text{adj}} \notin C$
    **if** $x_{\text{adj}} \notin O$ **then**
      Add $x_{\text{adj}}$ to $O$
    **else if** $g(x_{\text{best}}) + c(x_{\text{best}}, x_{\text{adj}}) < g(x_{\text{adj}})$ **then**
      Update $x_{\text{adj}}$'s backpointer to point to $x_{\text{best}}$
    **end if**
  **until** $O$ is empty
  **return** ReconstructPath($x_{\text{goal}}, x_{\text{start}}$)
**end procedure**

is the sum of the motion cost at previous step $g(x_{\text{n}-1})$ and the trapezoidal area described by the motion from $x_{\text{n}-1}$ and $x_{\text{n}}$, denoted with $c(x_{\text{n}-1}, x_{\text{n}})$:

$$\begin{aligned} g(x_{\text{n}}) &= \int_{x_{\text{start}}}^{x_{\text{n}-1}} r_{\text{c}}(x)dx + \int_{x_{\text{n}-1}}^{x_{\text{n}}} r_{\text{c}}(x)dx \\ &= g(x_{\text{n}-1}) + c(x_{\text{n}-1}, x_{\text{n}}), \end{aligned} \tag{8}$$

with

$$c(x_{\text{n}-1}, x_{\text{n}}) = \frac{r_{\text{c}}(x_{\text{n}-1}) + r_{\text{c}}(x_{\text{n}})}{2} \text{dist}(x_{\text{n}-1}, x_{\text{n}}), \tag{9}$$

where $\text{dist}(x_{\text{n}-1}, x_{\text{n}})$ is the Euclidean distance between two nodes. Figure 2(b) illustrates the discretization of the cost function.

The term $h(x_{\text{n}})$ indicates the area described by the estimated motion from $x_{\text{n}}$ and the final state $x_{\text{goal}}$. With A*-based algorithm, if an admissible heuristic is adopted, the algorithm is able to search for the optimal solution in the graph. The heuristic $h(x_{\text{n}})$ is admissible if $h(x_{\text{n}}) \leq h^*(x_{\text{n}})$ for each node $x_{\text{n}}$ and with $h^*(x_{\text{n}})$ being the effective optimal motion cost from $x_{\text{n}}$ to the goal. If the heuristic is not admissible, the A*-based algorithm can overestimate the motion cost to reach the goal, and it can overlook nodes that would lead to the optimal solution. Then, in order to have an admissible heuristic, we estimate the motion considering the minimum risk-cost between the node $x_{\text{n}}$ and the node $x_{\text{goal}}$. Then, as depicted in Figure 2(b), the heuristic cost can be computed as follows:

$$\begin{aligned} h(x_{\text{n}}) = &\frac{r_{\text{c}}(x_{\text{n}}) + r_{\text{c min}}}{2} \text{dist}(x_{\text{n}}, x_{\text{n}+1}) + \text{dist}(x_{\text{n}+1}, x_{\text{goal}-1}) r_{\text{c min}} + \\ &+ \frac{r_{\text{c}}(x_{\text{goal}}) + r_{\text{c min}}}{2} \text{dist}(x_{\text{goal}-1}, x_{\text{goal}}), \end{aligned} \tag{10}$$

where $r_{\text{c min}} > 0$ is the minimum value of the risk-cost function. Assuming $\text{dist}(x_{\text{n}}, x_{\text{n+1}}) = \text{dist}(x_{\text{goal}-1}, x_{\text{goal}}) = \text{dist}_{\text{min}}$, with $\text{dist}_{\text{min}}$ the minimum distance between two adjacent nodes, the heuristic function becomes:

$$h(x_{\text{n}}) = \frac{r_{\text{c}}(x_{\text{n}}) + r_{\text{c}}(x_{\text{goal}})}{2}\text{dist}_{\text{min}} + (\text{dist}(x_{\text{n}}, x_{\text{goal}}) - \text{dist}_{\text{min}})r_{\text{c min}} \tag{11}$$

The A*-based algorithms are able to find the optimal solution in the graph. However, due to the the discrete space, the outcome of A*-based algorithms may not be the optimal one in the continuous space. In fact, the path computed with riskA* is constrained to turn angles multiple of 45°. A Post-Optimization phase is performed after the execution of riskA*.

Post-Optimization is described in Algorithm 2. The algorithm explores the path (lines 3 to 12). Then, considering two states $x_{\text{j}}, x_{\text{k}} \in \sigma$, the algorithm verifies if the line of sight segment $\text{LOS}(x_{\text{j}}, x_{\text{k}})$ improves the path. The $\text{LOS}(\cdot)$ segment connects two nodes $x_{\text{j}}$ and $x_{\text{j}}$ with a straight line in the continuous space, inserting additional nodes using a linear interpolation (line 4). The interpolation step is comparable with the risk-map resolution. Then, the motion cost $\text{cost}(\cdot)$ of the $\text{LOS}(\cdot)$ segment is computed using the incremental method described in Equations (8) and (9). If the $\text{LOS}(x_{\text{j}}, x_{\text{k}})$ segment has a lower motion cost than the path from $x_{\text{j}}$ to $x_{\text{k}}$, the algorithm updates the path (line 6). Moreover, after the replacement of the $LOS(\cdot)$ segment, the $k$ index needs to be updated in respect to the updated path (line 7). These iterative procedure continues until the $x_{\text{goal}}$ is reached. A simple example of the Post-Optimization procedure is depicted in Figure 3.

We remark that the Post-Optimization procedure aims to optimize the path locally, since the global optimization is provided by the riskA* algorithm.

**Algorithm 2** Post-Optimization algorithm

```
1: procedure PostOptimization(σ)
2:     j = 1, k = 3
3:     while j ≠ length(σ) do
4:         Interpolate(LOS(x_j, x_k))
5:         if cost(LOS(x_j, x_k)) ⩽ cost(segment(x_j, x_k)) then
6:             Replace segment(x_j, x_k) ∈ σ with LOS(x_j, x_k)
7:             Update k
8:             k = k + 1
9:         else
10:            j = j + 1
11:        end if
12:    end while
13:    return σ
14: end procedure
```

## 4 On-line Path Planning

### 4.1 Problem formulation

The on-line path planning is based on a *check and repair* approach [11], in which the path is dynamically adapted in accordance to a dynamic risk-map.

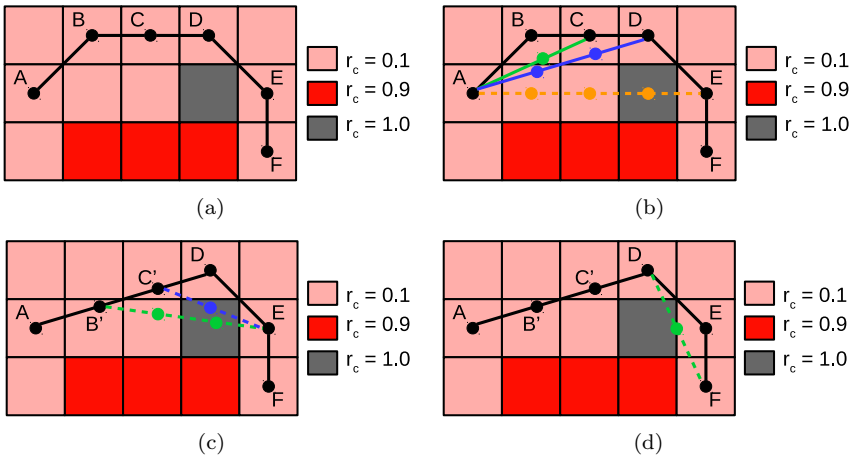



**Fig. 3** Simple example of the Post-Optimization procedure. In (a) the path is computed with riskA*, as a sequence of nodes from A to F. In (b) the Post-Optimization procedure searches for LOS(·) segments that improve the path. Starting from node A, it considers at first the LOS(A, C), then the LOS(A, D). On the contrary, it discards the LOS(A, E), because it crosses a high risk area. In (c), the path is updated with the segment A-B'-C'-D, with B' and C' being the interpolated nodes of the LOS(A, D). Then, the Post-Optimization procedure discards the LOS(B′, E) and the LOS(C′, E), as well as the LOS(D, F) in (d).

Recalling the notation of the off-line path planning problem described in Section 3.1, the off-line path $\sigma$ is a sequence of locations $x \in X_{\text{free}}$ from $x_{\text{start}}$ to $x_{\text{goal}}$. Considering the dynamic risk-map, the search space $X(k)$ changes at each discrete-time step $k$, then the path is considered as a sequence of states $x(k)$. Hence, at each time step, the differential search space $X_{\text{diff}}(k)$ can be defined as follows:

$$X_{\text{diff}}(k) = X(k) - X(k-1), \tag{12}$$

Moreover, the on-line path planning algorithm aims to repair the path only when it is necessary, i.e. when the path is involved by an area with increased risk-costs. For this reason, a differential risk-map $M_{\text{diff}}$ related to $X_{\text{diff}}$ is defined as:

$$M_{\text{diff}}(x_{\text{n}}(k)) = \begin{cases} 1 & \text{if } \Delta r_{\text{c}}(x_{\text{n}}(k)) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

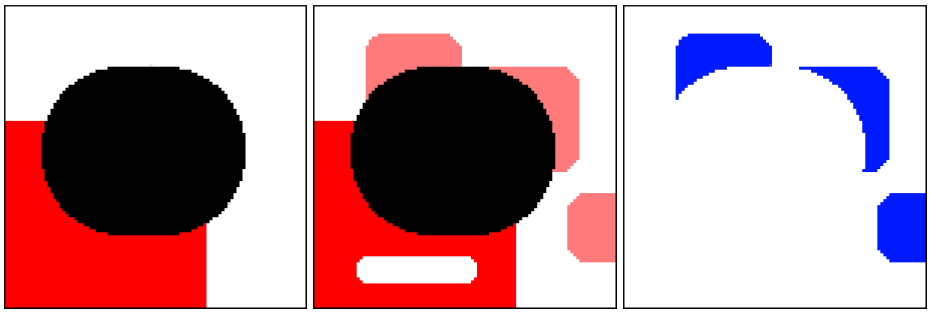
with

$$\Delta r_{\text{c}}(x_{\text{n}}(k)) = r_{\text{c}}(x_{\text{n}}(k)) - r_{\text{c}}(x_{\text{n}}(k-1)) \tag{14}$$

The differential risk-map $M_{\text{diff}}$ has the same dimension of the risk-map $M$. The notation $M_{\text{diff}}(x_{\text{n}}(k))$ defines the state of the generic node $x_{\text{n}}$ in the differential map $M_{\text{diff}}$ at time $k$. Figure 4 illustrates an example of risk-maps at time $k$ and $k-1$, and the corresponding differential map.

The *check* routine verifies if $M_{\text{diff}}(x(k)) > 0, \ \vee \ x(k) \in \sigma$, i.e., it verifies which part of the path has to be updated because of a change in the risk-map.

The *repair* routine tries to adjust the path with a fast algorithm, in order to tackle the dynamic risk-map.



(a) (b) (c)

**Fig. 4** Example of the risk-map in which the risk areas are identified: white areas are with minimum risk-cost, black areas are with maximum risk-cost, and shade of red areas are with middle cost, in which darker red areas involving more risk than bright red ones. In (a), the risk-map at time $k-1$. In (b), the risk-map at time $k$. In (c), the differential risk-map defined according to Equation (13).

The riskA* is not suitable to solve this problem, because of the computational time, since it requires a time proportional with the dimension of the map. Unlike off-line path planning, the on-line algorithm needs to adapt the path in short time, because the UAV is performing the mission. As a consequence, the on-line path planning needs to trade off between computational burden and optimality.
In this work, we propose the Borderland algorithm as a solution to the on-line path planning problem.

### 4.2 Borderland algorithm

Here, we introduce the Borderland algorithm. It is an extension of Bug algorithms [30] applied to grid graphs and with generic motion cost. Bug-based algorithms are widely used in on-line path planning to tackle complex and dynamic environments [1,21]. In our approach the idea is to follow the contour of each risk area involved and circumnavigate it, in order to adjust the path minimizing the combination of risk-cost and path length.

The pseudo code of the Borderland algorithm is described in Algorithm 3. The input are: *(i)* the current UAV position at time $k$, *(ii)* the most recent path computed at time $k-1$, *(iii)* the grid graph search space $M$ at time $k$, related to $X$, and *(iv)* the differential space $M_{\text{diff}}$ related to $X_{\text{diff}}$. The Borderland algorithm always considers the path from the current UAV position to the goal, because the previous portion of the path is already executed.
First, the algorithm visits each location $x$ in the path $\sigma$ and it checks if each $x$ is involved in the differential risk-map. If this is the case, it adds $x$ to a differential set $D$ (lines 2 to 5). The differential set $D$ is a set of nodes members of the path $\sigma$ and involved in the differential risk-map $M_{\text{diff}}$.
From all locations in $D$, Borderland detects path segments $S[x_{\text{a}}, x_{\text{b}}]$ as a sequence of locations (line 7). The segments $S[x_{\text{a}}, x_{\text{b}}]$ are the portions of path need to be repaired.
Here, the algorithm checks if the current position $x_{\text{pos}}$ is involved in the segment

**Algorithm 3** Borderland algorithm

```
1: procedure BorderlandSearch(x_pos, σ, M, M_diff)
2:     for each x ∈ σ do
3:         if M_diff(x) > 0 then
4:             Add x to D
5:         end if
6:     end for
7:     Detect segments S[x_a, x_b] as a sequence of x ∈ D
8:     for each S[x_a, x_b] do
9:         if x_a ∈ S[x_a, x_b] = x_pos then
10:            Search nearest x_esc with M_diff(x_esc) = 0
11:            Search for σ_seg[x_a, x_b] through x_esc
12:        else
13:            σ_seg = Circumnavigate area(M_diff > 0)
14:            if ∄ σ_seg then
15:                if r_c(area(M_diff > 0)) < 1 then
16:                    Reduce area(M_diff > 0) until ∃ σ_seg
17:                else if r_c(area(M_diff > 0)) = 1 then
18:                    Search σ_seg[x_a, x_b] in M
19:                end if
20:            end if
21:        end if
22:        if ∃ σ_seg then
23:            if cost(σ_seg) > cost(S[x_a, x_b]) then
24:                Discard σ_seg
25:            end if
26:        else
27:            ∄ solution
28:            return
29:        end if
30:    end for
31:    Reconstruct Path σ_new with σ and every σ_seg
32:    Simplify σ_new
33:    return σ_new
34: end procedure
```

$S[x_a, x_b]$. In the affirmative case, it searches for an escape location $x_{esc}$ outside of the differential risk-map and seeks for an alternative segment $\sigma_{seg}$, passing through $x_{esc}$ (lines 9 to 11).

Hence, for each $S[x_a, x_b]$, the Borderland algorithm tries to circumnavigate the differential area in $M_{diff}$ with a new portion of path $\sigma_{seg}$ (line 13) as an alternative segment.

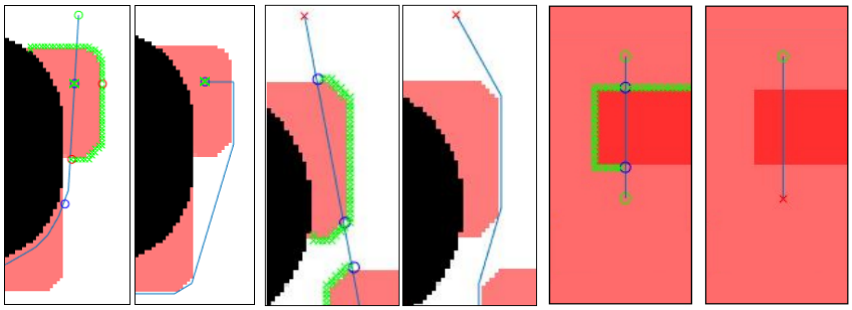
If $\sigma_{seg}$ does not exist and the involved area in $M_{diff}$ has a risk-cost lower than 1 (obstacle or no-flight area), the algorithm searches for an alternative path, by reducing the involved area $M_{diff}$ until $\sigma_{seg}$ exists (lines 15, 16). This means that the algorithm searches for an alternative path in the differential map (see Figure 6).

Otherwise, if the involved area is a no-fly zone, it is impossible to find a local solution considering the differential risk-map. Thus, the algorithm seeks for a solution in the risk-map $M$, circumnavigating no-fly zones (lines 17, 18).

If a feasible portion of path $\sigma_{seg}$ exists, the Borderland algorithm compares the motion cost of the new segment $\sigma_{new}$ with the old one $S[x_a, x_b]$. If the former has a greater cost, then it discards $\sigma_{seg}$ (lines from 22 to 25).

On the contrary, if a solution doesn't exist, the algorithm reports it.

Once all segments have been examined, the new path $\sigma_{new}$ is reconstructed using



(a) (b) (c)

**Fig. 5** Examples of Borderland scenarios. After the update of the risk-map, in (a), the current position is in a high risk area. Thus, an escape route is computed, finding an alternative path with lower cost. In (b), a common scenario, whereby the algorithm circumnavigates the risk area with a path with lower motion cost. In (c), the algorithm tries to circumnavigate the risk-area. The alternative path has a greater cost then the original one, then the route doesn't change.

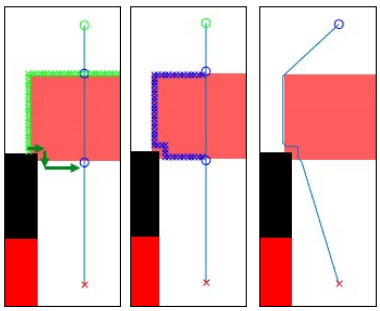
the old path $\sigma$ and segments $\sigma_{\text{seg}}$ (line 31). Finally, the algorithm simplifies the path $\sigma_{\text{new}}$ with the Post-Optimization procedure described in Algorithm 2 (line 32).

In order to clarify how the Borderland algorithm works, in Figures 5 and 6 some simple scenarios are illustrated. In Figure 5(a), the current position is in the high risk area due to a dynamic update of the risk-map. The algorithm searches for an escape location outside of the updated differential space and it finds an alternative path.
Figure 5(b) shows the typical scenario, in which the algorithm circumnavigates the high risk area and it finds a path with lower cost. On the contrary, in Figure 5(c), the new route computed by Borderland has a greater cost than the previous one, then it discards it.
In the particular scenario of Figure 6, the Borderland algorithm searches for an alternative path. In order to find a solution, the algorithm reduces the involved area and the resulting path minimizes the motion cost.

## 5 Path Smoothing using Dubins Curves

After the path planning procedure, the theoretical path is not suitable for UAVs. Even if the Post-Optimization procedure solves the path coinstraining to grid edges, the resulting path can't be performed by aerial vehicles because of kinematic constraints. In order to achieve a more suitable and realistic path, a smoothing procedure is required. Due to their simplicity and performance, Dubins curves are the suitable solution.

The Dubins curves are introduced in [10] and they refer to the shortest curve between two poses in the two-dimensional plane considering constraints in the radius of curvature. Assuming a constant speed of the vehicle and the state of the



(a) (b) (c)

**Fig. 6** Example of Borderland scenario. After the update of the risk-map, in (a), the path crosses the area with high risk-cost. The algorithm searches for an alternative path. As there is no solution, the algorithm searches for the solution in the differential map by *reducing* the involved area, until an alternative path is found (b). In (c) the final solution.

UAV $q = (q_x, q_y, q_\theta)$, the differential equation of Dubins curves are:

$$\dot{q_x} = cos(q_\theta) \tag{15}$$

$$\dot{q_y} = sin(q_\theta) \tag{16}$$

$$\dot{q_\theta} = u \tag{17}$$

with $u$ normalized between $-1$ and 1 with respect to the maximum curvature. According to [10], the shortest path between two poses is always expressed as a combination of no more than three motion primitives. For this reason only three values of $u$ are used $u \in \{-1, 0, 1\}$. For simplicity, the value 0 describes a straight motion (S), whereas $-1$ and 1 are the right (R) and left (L) turn, respectively. As a consequence, only six combination of curves are possible:

$$\{LRL, RLR, LSL, LSR, RSL, RSR\} \tag{18}$$

In Figure 7 is illustrated the Path Smoothing procedure using Dubins curves.

Often, the smoothing using Dubins curves is performed during the path planning phase, whereby the path is defined considering the curvature radius of the vehicle [16,28]. Sometimes, this latter approach is preferable because the smoothing procedure can compromise the optimality of the path. On the contrary, the disadvantage is the introduction of more complexity in the algorithm, increasing the computation time. However, we prefer to perform the smoothing procedure after the path planning phase for two reason: *(i)* the resolution of the risk-map is compared with the curvature radius of the vehicle, then the path remains optimal; *(ii)* the smoothing is performed in very short time, useful for the on-line adaptation of the path.

## 6 Simulation Results

This section reports preliminary simulation results, obtained through Matlab simulations and using a laptop with a 2-core with 1.9 GHz CPU.

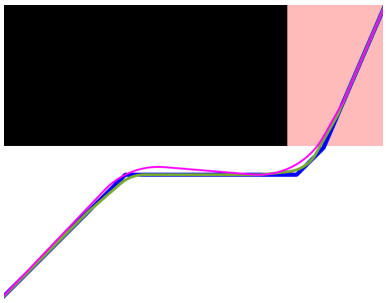
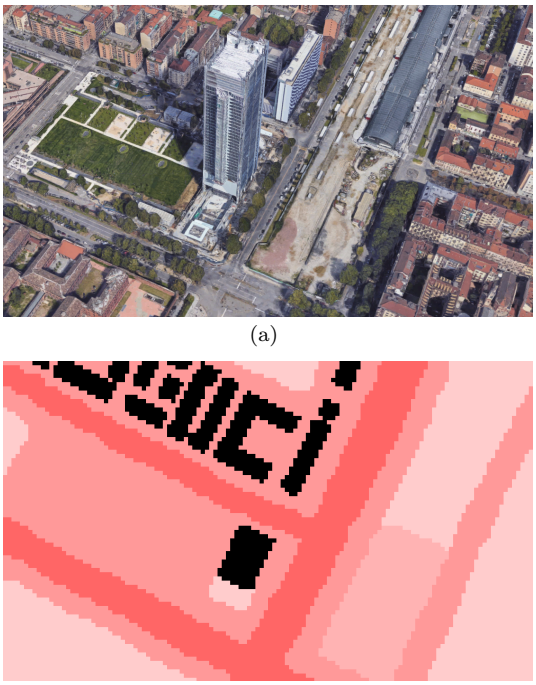


**Fig. 7** Example of the Path Smoothing procedure using Dubins curves. In blue the path before the smoothing procedure. In green the smoothed path with a curvature radius of 10 m, while in magenta the path with a curvature radius of 20 m.



(a)

(b)

**Fig. 8** Risk-map related to the Torino's neighbourhood. In (a) the urban area from Google Maps. In (b) the realistic risk-map at 20 meter of altitude. Black pixels describe the occupied areas ($\mathbf{r_c} = 1$), while in shade of red areas are with other risk-costs ($0 < \mathbf{r_c} < 1$), where darker red areas have a greater risk-cost than bright red ones.

**Table 1** Results of riskA* algorithm with different values of $k$.

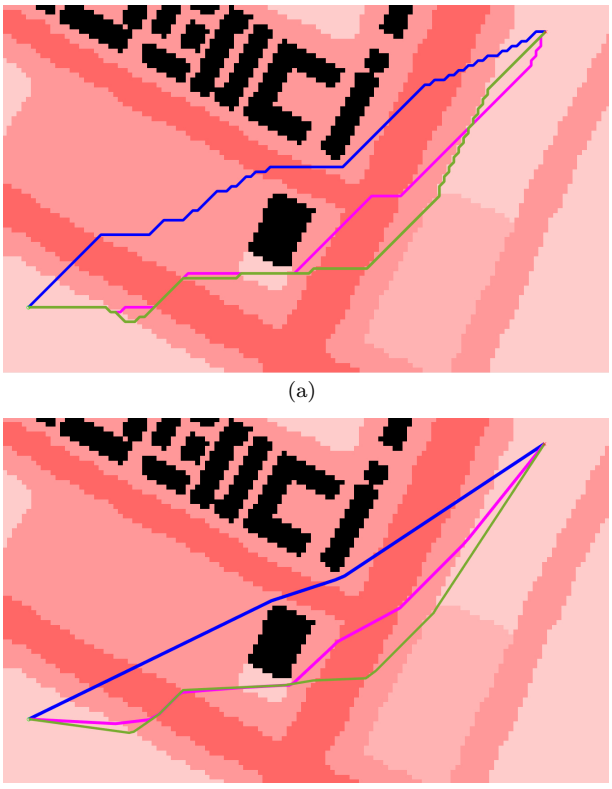
| Map | k | solve time [s] | path length [m] | cost | average risk-cost |
|---|---|---|---|---|---|
| Map 1 $126 \times 76$ cells (500 simulations) | 0.0 | 0.3010 | 333.7345 | 107.1005 | 0.3220 |
| | 0.5 | 0.2623 | 333.7345 | 107.1005 | 0.3220 |
| | 0.75 | 0.2432 | 333.7345 | 107.1005 | 0.3220 |
| | 1.0 | 0.2256 | 333.7525 | 107.1010 | 0.3219 |
| | 1.25 | 0.2066 | 333.7525 | 107.1030 | 0.3220 |
| | 1.5 | 0.1864 | 333.9410 | 107.1150 | 0.3218 |
| | 2.0 | 0.1472 | 334.2295 | 107.1945 | 0.3219 |
| | 2.5 | 0.1126 | 334.2990 | 107.5485 | 0.3227 |
| | 3.0 | 0.0814 | 333.7855 | 108.2410 | 0.3252 |
| Map 2 $339 \times 131$ cells (200 simulations) | 0.0 | 3.3292 | 769.4265 | 220.5930 | 0.2864 |
| | 0.5 | 2.8039 | 769.4265 | 220.5930 | 0.2864 |
| | 0.75 | 2.5165 | 769.4265 | 220.5930 | 0.2864 |
| | 1.0 | 2.3007 | 769.5300 | 220.5940 | 0.2863 |
| | 1.25 | 2.1237 | 769.5300 | 220.6045 | 0.2864 |
| | 1.5 | 1.9884 | 769.7800 | 220.6225 | 0.2864 |
| | 2.0 | 1.6323 | 771.2785 | 220.7955 | 0.2865 |
| | 2.5 | 1.1328 | 773.3660 | 222.0745 | 0.2872 |
| | 3.0 | 0.6551 | 774.1210 | 223.5875 | 0.2888 |

The risk-map used in the simulations is illustrated in Figure 8 and describes a Torino's neighbourhood with realistic risk-costs. The risk-map has $126 \times 76$ cells, where each element is a square cell with dimension $5 \times 5$ m. The risk-map is defined by colored coded: white areas are with minimum risk-cost, black areas are with maximum cost (obstacles and no-flight areas), while shade of red areas are with middle cost, in which darker red areas involving more risk than bright red ones. As reported in Section 2, the maximum acceptable probability is defined at $10^{-6}$ h$^{-1}$, then zones with greater risk values are defined as no-flight areas.

Regarding the off-line path planning, the proposed riskA* algorithm is implemented. In order to select the best value of the parameter $k$, a Monte Carlo simulation campaign is carried out considering a set of values of $k$ and two maps. We execute 500 independent simulations with the map illustrated in Figure 8(b) and 200 independent simulations with the map in Figure 11. Simulations are randomized with respect to start and goal positions. Observing numerical results in Table 1, we can conclude that the $k$ parameter affects the optimized path and the time required to compute the solution. High values of $k$ yields fast solution time at the cost of path optimality. An opposite effect is observed for low values of $k$. In particular, with $k = 0$, the heuristic function is not active and the behavior of the riskA* is the same of the Dijkstra algorithm. With both maps, our simulations lead to select a value of $k = 0.75$ to provide a good trade-off between optimality of the solution and computation time.

According to the results in Table 1, the time to attain the optimal solution via riskA* also depends on the map size, due to an increase of the size of the graph to be visited.

Further, we also compare our riskA* algorithm with the original A* and RA*. The RA* algorithm is proposed in [14] by Guglieri *et al.*. Similar to riskA*, RA* is based on original A* and considers the risk to the population of flying over a specified area. The cost function of RA* takes into account the risk-cost as an additive factor, whereas, in this test, the A* algorithm optimizes the path length.



(a)

(b)

**Fig. 9** Path planning with A* (in blue), RA* (in magenta) and riskA* (in green). In (a) only the path planning algorithm is executed, while in (b) the Post-Optimization procedure improves the path.

**Table 2** Numerical results of the simulation depicted in Figure 9. The percentage values refer to the values of the A* algorithm.

| **Algorithm** | **solve time [s]** | **path length [m]** | **cost** | **average risk-cost** |
|---|---|---|---|---|
| A* | 0.5621 | 653.0510 | 254.0945 | 0.3883 |
| PO A* | 0.5637 | 608.5530 (-6.81%) | 234.6380 (-7.65%) | 0.3834 (-1.26%) |
| RA* | 0.4684 | 668.9085 (+2.42%) | 209.9570 (-17.37%) | 0.3070 (-20.94%) |
| PO RA* | 0.4692 | 649.4940 (-0.54%) | 202.9355 (-20.13%) | 0.3098 (-20.21%) |
| riskA* | 0.5446 | 700.6245 (+7.2848) | 196.0440 (-22.85%) | 0.2790 (-28.14%) |
| PO riskA* | 0.5468 | 674.4790 (+3.28%) | 189.9475 (-25.25%) | 0.2781 (-28.38%) |

Figure 9 illustrates a comparison between the mentioned approaches. In the reported scenario, the advantage of riskA* is apparent. According to the numerical results reported in Table 2, the riskA* algorithm tends to compute a longer path, yet reducing the average risk-cost, as well as the motion cost.

In order to validate the proposed riskA* algorithm, 500 independent Monte Carlo simulations are performed comparing A*, RA* and riskA*, where the start and goal points are randomly sampled in the map. Test results are reported in

**Table 3** Comparison of A*, RA* and riskA*. The numerical results are the average values of 500 simulations.

| **Algorithm** | **solve time [s]** | **path length [m]** | **cost** | **average risk-cost** |
|---|---|---|---|---|
| A* | 0.0554 | 376.9025 | 140.5340 | 0.3705 |
| PO A* | 0.0755 | 356.6225 (-5.38%) | 133.1715 (-5.23%) | 0.3707 (+0.05%) |
| RA* | 0.1964 | 381.8215 (+1.31%) | 128.1155 (-8.84%) | 0.3329 (-10.15%) |
| PO RA* | 0.2066 | 367.5745 (-2.47%) | 123.2700 (-12.28%) | 0.3341 (-9.82%) |
| riskA* | 0.2812 | 392.5540 (+4.15%) | 125.5625 (-10.65%) | 0.3220 (-13.09%) |
| PO riskA* | 0.2948 | 375.1550 (-0.46%) | 120.0565 (-14.57%) | 0.3207 (-13.41%) |

Table 3. This test demonstrates that riskA* is able to find the optimal path by trading off the path length and the risk-cost. Compared with the A*-based solution, the path is longer (+4.15%), but the average risk-cost is significantly lower (−13.09%). The results show also the different behavior of the RA* algorithm. Compared with the riskA*, RA* computes shorter paths, but with higher average risk-cost.

After the path planning procedure, Post-Optimization is performed. Figure 9(b) highlights the advantage of the application of the Post-Optimization procedure. We observe that, in the original paths in Figure 9(a) the turn angles are constrained to be expressed in multiples of 45°, while Post-Optimization optimizes the path without compromising the optimality. The advantages of the Post-Optimization procedure are also reported in Tables 2 and 3, performed on A*, RA* and riskA*. With A*, Post-Optimization only takes into account the path length, while with RA* and riskA* the motion cost is used, considering both path length and risk-costs. We observe that Post-Optimization reduces the path length and the motion cost, maintaining a comparable values of average risk-cost. Moreover, Post-Optimization only slightly affect the computation time of the solution.

Regarding the on-line path planning, the Borderland algorithm is implemented. Figure 10 exemplifies a scenario in which Borderland algorithm and Path Smoothing using Dubins curves are applied. The results displayed in the figure highlights the advantages of our approach with respect to riskA* in executing the replanning phase. Numerical performance indicators are synthesized in Table 4, where the replanning phase is executed twice on different risk-maps. In Map 1, the path is computed using riskA*. The risk-map is updated (Map 2) and, as a consequence, the motion cost of the path changes and the path needs to be updated. The riskA* computes a new path from scratch, while Borderland tries to repair the oldest path. Referring to Table 4 the Borderland algorithm finds a solution that is not the optimal one, but the resulting motion cost is lower than the previous path. Moreover, the solve time is significantly less in respect of the riskA* algorithm. Similar results are obtained with the third map. The path in the new map (Map 3) is not valid and both riskA* and Borderland update the path. RiskA* and Borderland compute a path with similar features, but Borderland requires less computational time. In this test both Post-Optimization and Path Smoothing with Dubins curves are performed, providing a suitable path for UAVs.

The advantage of the Borderland algorithm is the fast adaptation of the path. Borderland searches for an alternative solution by exploring the local space near the involved area in the differential map. On the contrary, riskA* computes the optimal global path, by evaluating all the search space. For this reason, the Bor-

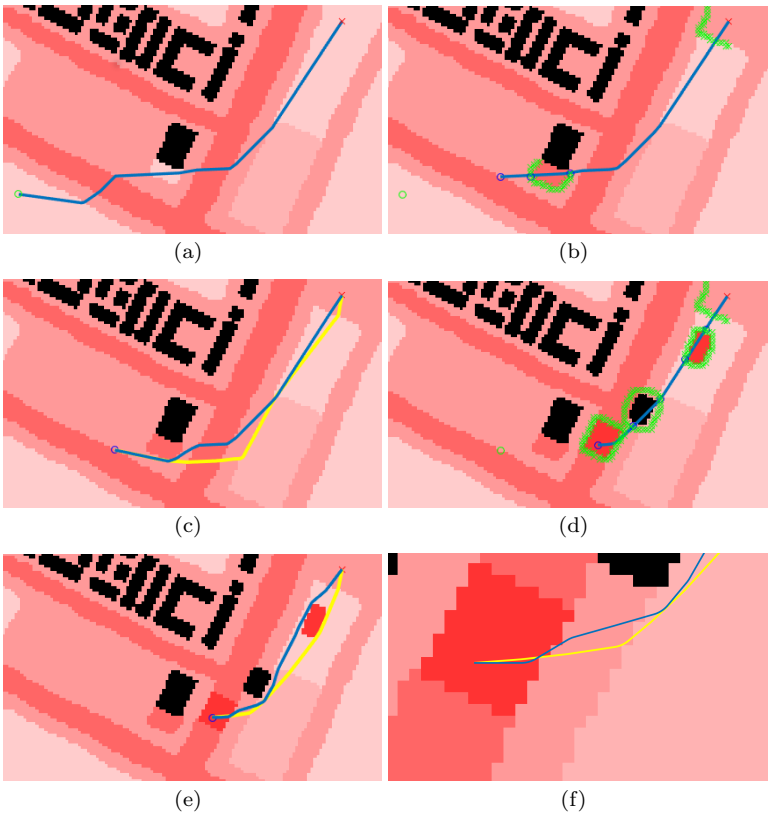
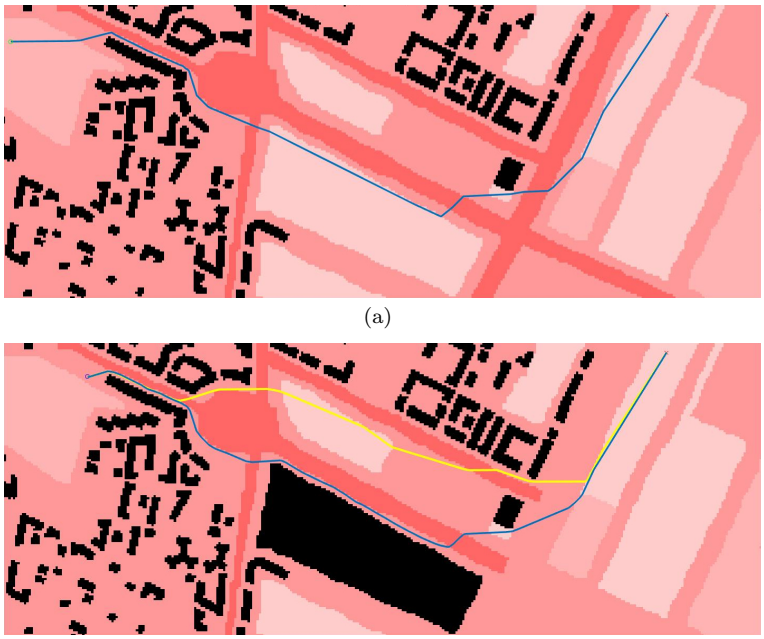


**Fig. 10** Example of the proposed risk-aware path planning approach. In (a), riskA* computes the off-line path (in blue). In (b), the risk-map changes and the Borderland algorithm checks the path exploring cells around the updated area. In (c), the path repaired by the Borderland (in blue) and the path computed from scratch with the riskA* algorithm (in yellow) are compared. Similar behavior in (d) end (e), whereby the risk-map is updated and the Borderland algorithm is able to adapt the path. In (f) a detail of the path computed, where the path is smoothed with Dubins curves.

**Table 4** Results of on-line path planning.

| **Map ID** | | **solve time [s]** | **path length [m]** | **cost** | **average risk-cost** |
|---|---|---|---|---|---|
| 1 | PO riskA* | 0.8634 | 674.4790 | 189.9475 | 0.2781 |
| 2 | previous path | | 497.3965 | 165.7020 | 0.3336 |
| | PO riskA* | 0.7902 | 530.0510 | 157.9735 | 0.2986 |
| | Borderland | 0.1932 (-75.55%) | 506.5655 (-4.43%) | 159.3680 (+0.88%) | 0.3159 (+5.79%) |
| 3 | previous path | | 336.5660 | Invalid | Invalid |
| | PO riskA* | 0.2985 | 351.5975 | 98.2045 | 0.2839 |
| | Borderland | 0.1211 (-59.43%) | 348.9715 (-0.74%) | 98.4210 (+0.22%) | 0.2856 (+0.60%) |



(a)

(b)

**Fig. 11** Simple scenario with high dimensional map. In (a) the path computed by riskA*. In (b) the path computed with Borderland (in blue) and with riskA* (in yellow) are reported.

**Table 5** Risk-aware path planning in high dimension map. The percentage values compare the Borderland with the PO riskA* algorithm.

| Map ID | | solve time [s] | path length [m] | cost | average risk-cost |
|---|---|---|---|---|---|
| 1 | PO riskA* | 10.7896 | 1813.9145 | 503.5395 | 0.2760 |
| 2 | previous path | | 1644.6530 | Invalid | Invalid |
| | PO riskA* | 6.6231 | 1512.1280 | 447.9680 | 0.2966 |
| | Borderland | 0.2931 (-95.57%) | 1638.6085 (+8.36%) | 522.9410 (+16.74%) | 0.3174 (+7.01%) |

derland algorithm cannot guarantee the global optimal solution. However, the advantage of Borderland is more visible with high dimension map. In Figure 11 a simple scenario is shown using a map with $339 \times 131$ cells. Results in Table 5 demonstrate how Borderland is faster then riskA*. The solve time in Table 5 is the amount of time to perform the replanning, as well as both the Post-Optimization and Path Smoothing using Dubins curves. Figure 10(f) shows a detail of the path after the Path Smoothing procedure.

## 7 Conclusions and Future Works

In this paper, we have presented a risk-aware path planning strategy for UAVs in urban environments, where people and crowds are a critical safety issue. The proposed approach computes a path usable in autonomous missions and it consists of two phases, off-line and on-line path planning.

The off-line path planning searches for a global path considering the risk-map as a static environment. An A*-based algorithm is proposed, called riskA*. RiskA* uses a cost function that considers both path length and risk-cost, the resulting path minimizes the combination of these two factors.
Simulation results corroborate the validity of our approach and demonstrate the good performance of riskA*. In particular, riskA* is able to seek for an optimal solution considering the risk to the population on ground, critical factor in urban environments. Results of riskA* are also compared with RA*, an A*-based algorithm proposed in [14].

The on-line path planning adjusts and adapts the path considering a dynamically updated risk-map. We propose the Borderland algorithm as a solution. Toward the adaptation of the off-line trajectory to the new operational conditions, Borderland performs *check and repair* routines. First, it checks which parts of path are involved in the dynamic risk-map, then it searches for an alternative path.
Simulation results show that Borderland is able to repair and adjust the path in short time and in a different scenarios, minimizing the risk-cost. Moreover, instead of compute new path from scratch every time, Borderland adjusts the path only when it is necessary. It is an advantage, especially in high dimensional map.

After the path planning, the Path Smoothing procedure allows a realistic and suitable path for UAVs to be obtained. The smoothing is performed using Dubins curves and provides to modify the path in a flyable one in short time and without compromising the optimality of the path.

The joint use of off-line and on-line path planning constitutes a valid risk-aware path planning strategy for UAVs. The resulting behavior demonstrates that the proposed approach is able to compute and maintain a valid, safe and reliable path, despite the evolution in the operational conditions.
The proposed risk-aware path planning approach solves the problem of compute and maintain a safe path in urban environments. In particular it guarantees to always have a risk level lower than the maximum acceptable risk defined, as well as it avoids the no-flight zones forced by National aviation agencies.

Future works will include the implementation of the proposed approach on a real robotic platform using ROS (Robot Operating System) [35]. Possible improvements involve the consideration of kinodynamic constraints of the vehicle, the presence of multiple UAVs, and the adaptation to a tridimensional environment.

## Acknowledgment

This work was supported by a fellowship from TIM, by the Siebel Energy Institute, and by Compagnia di San Paolo.

## References

1. Buniyamin, N., Wan Ngah, W., Sariff, N., Mohamad, Z.: A simple local path planning algorithm for autonomous mobile robots. International journal of systems applications, Engineering & development **5**(2), 151–159 (2011)
2. Cai, G., Dias, J., Seneviratne, L.: A survey of small-scale unmanned aerial vehicles: Recent advances and future development trends. Unmanned Systems **2**(02), 175–199 (2014)

3. Clothier, R.A., Walker, R.A., Fulton, N., Campbell, D.A.: A casualty risk analysis for unmanned aerial system (uas) operations over inhabited areas. In: AIAC12, Twelfth Australian International Aerospace Congress, 2nd Australasian Unmanned Air Vehicles Conference, pp. 1–15 (2007)
4. la Cour-Harbo, A.: Mass threshold for harmless drones. International Journal of Micro Air Vehicles **9**(2), 77–92 (2017)
5. la Cour-Harbo, A.: Quantifying ground impact fatality rate for small unmanned aircraft. Journal of Intelligent & Robotic Systems **-**, 1–18 (2018). DOI 10.1007/s10846-018-0853-1
6. Dalamagkidis, K., Valavanis, K., Piegl, L.A.: On integrating unmanned aircraft systems into the national airspace system: issues, challenges, operational restrictions, certification, and recommendations, vol. 54. Springer Science & Business Media (2011)
7. De Filippis, L., Guglieri, G., Quagliotti, F.: A minimum risk approach for path planning of uavs. Journal of Intelligent & Robotic Systems **61**(1), 203–219 (2011)
8. De Filippis, L., Guglieri, G., Quagliotti, F.: Path planning strategies for uavs in 3d environments. Journal of Intelligent & Robotic Systems **65**(1), 247–264 (2012)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
10. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics **79**(3), 497–516 (1957)
11. Ferguson, D., Kalra, N., Stentz, A.: Replanning with rrts. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pp. 1243–1248 (2006)
12. Feyzabadi, S., Carpin, S.: Risk-aware path planning using hirerachical constrained markov decision processes. In: Automation Science and Engineering (CASE), 2014 IEEE International Conference on, pp. 297–303. IEEE (2014)
13. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. In: Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009, pp. 65–100. Springer (2009)
14. Guglieri, G., Lombardi, A., Ristorto, G.: Operation oriented path planning strategies for rpas. American Journal of Science and Technology **2**(6), 1–8 (2015)
15. Guglieri, G., Ristorto, G.: Safety assessment for light remotely piloted aircraft systems. In: INAIR 2016 - International Conference on Air Transport, vol. 1, pp. 1–7 (2016)
16. Hansen, K.D., la Cour-Harbo, A.: Waypoint planning with dubins curves using genetic algorithms. In: Control Conference (ECC), 2016 European, pp. 2240–2246. IEEE (2016)
17. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
18. Islam, F., Narayanan, V., Likhachev, M.: Dynamic multi-heuristic a. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 2376–2382. IEEE (2015)
19. Islam, F., Narayanan, V., Likhachev, M.: A*-connect: Bounded suboptimal bidirectional heuristic search. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 2752–2758. IEEE (2016)
20. Jensen, O.B.: Drone city-power, design and aerial mobility in the age of" smart cities". Geographica Helvetica **71**(2), 67 (2016)
21. Karakaya, S., Ocak, H., Küçükyıldız, G.: A bug-based local path planning method for static and dynamic environments. In: International Symposium on Innovative Technologies in Engineering and Science. Valencia, Spain (2015)
22. Karaman, S., Frazzoli, E.: Incremental sampling-based algorithms for optimal motion planning. Robotics Science and Systems VI **104**, 2 (2010)
23. Karaman, S., Frazzoli, E.: Optimal kinodynamic motion planning using incremental sampling-based methods. In: Decision and Control (CDC), 2010 49th IEEE Conference on, pp. 7681–7687. IEEE (2010)
24. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation **12**(4), 566–580 (1996)
25. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning (1998). TR 98-11, Computer Science Dept., Iowa State University
26. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
27. Likhachev, M., Gordon, G.J., Thrun, S.: Ara*: Anytime a* with provable bounds on suboptimality. In: Advances in Neural Information Processing Systems, pp. 767–774 (2004)

28. Lin, Y., Saripalli, S.: Path planning using 3d dubins curve for unmanned aerial vehicles. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, pp. 296–304. IEEE (2014)
29. Lin, Y., Saripalli, S.: Sampling-based path planning for uav collision avoidance. IEEE Transactions on Intelligent Transportation Systems **18**(11), 3179–3192 (2017)
30. Lumelsky, V.J., Stepanov, A.A.: Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica **2**(1-4), 403–430 (1987)
31. Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., Jawhar, I.: Uavs for smart cities: Opportunities and challenges. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, pp. 267–273. IEEE (2014)
32. Murphy, R.: Introduction to AI robotics. MIT press (2000)
33. Pereira, A.A., Binney, J., Hollinger, G.A., Sukhatme, G.S.: Risk-aware path planning for autonomous underwater vehicles using predictive ocean models. Journal of Field Robotics **30**(5), 741–762 (2013)
34. Primatesta, S., Capello, E., Antonini, R., Gaspardone, M., Guglieri, G., Rizzo, A.: A cloud-based framework for risk-aware intelligent navigation in urban environments. In: Unmanned Aircraft Systems (ICUAS), 2017 International Conference on, pp. 447–455. IEEE (2017)
35. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol. 3, p. 5 (2009)
36. Rathbun, D., Kragelund, S., Pongpunwattana, A., Capozzi, B.: An evolution based path planning algorithm for autonomous motion of a uav through uncertain environments. In: Digital Avionics Systems Conference, 2002. Proceedings. The 21st, vol. 2, pp. 8D2–8D2. IEEE (2002)
37. Rudnick-Cohen, E., Herrmann, J.W., Azarm, S.: Risk-based path planning optimization methods for unmanned aerial vehicles over inhabited areas. Journal of Computing and Information Science in Engineering **16**(2) (2016)
38. Savkin, A.V., Huang, H.: The problem of minimum risk path planning for flying robots in dangerous environments. In: Control Conference (CCC), 2016 35th Chinese, pp. 5404–5408. IEEE (2016)
39. Silva Arantes, J.d., Silva Arantes, M.d., Motta Toledo, C.F., Júnior, O.T., Williams, B.C.: Heuristic and genetic algorithm approaches for uav path planning under critical situation. International Journal on Artificial Intelligence Tools **26**(01), 1760008 (2017)
40. Stentz, A.: Optimal and efficient path planning for unknown and dynamic environments. Tech. rep., DTIC Document (1993)
41. Sujit, P., Saripalli, S., Sousa, J.B.: Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. IEEE Control Systems **34**(1), 42–59 (2014)
42. Weiß, B., Naderhirn, M., del Re, L.: Global real-time path planning for uavs in uncertain environment. In: Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE, pp. 2725–2730. IEEE (2006)
43. Wen, N., Su, X., Ma, P., Zhao, L., Zhang, Y.: Online uav path planning in uncertain and hostile environments. International Journal of Machine Learning and Cybernetics **8**(2), 469–487 (2017)
44. Wzorek, M., Doherty, P.: Reconfigurable path planning for an autonomous unmanned aerial vehicle. In: Hybrid Information Technology, 2006. ICHIT'06. International Conference on, vol. 2, pp. 242–249. IEEE (2006)
45. Yang, L., Qi, J., Xiao, J., Yong, X.: A literature review of uav 3d path planning. In: Intelligent Control and Automation (WCICA), 2014 11th World Congress on, pp. 2376–2381. IEEE (2014)
46. Zhang, B., Mao, Z., Liu, W., Liu, J.: Geometric reinforcement learning for path planning of uavs. Journal of Intelligent & Robotic Systems **77**(2), 391–409 (2015)

**Stefano Primatesta** is currently a Ph.D. student in Computer and Control Engineering at the Politecnico di Torino, Italy. He received his B.S in Electronic Engineering end the M.S. in Mechatronic Engineering from Politecnico di Torino in 2011 and 2014, respectively. His research interests include autonomous navigation and service robotics, with applications on unmanned aerial vehicles and

unmanned ground vehicles.

**Giorgio Guglieri** is currently a Full Professor of Politecnico di Torino, Italy, in the Department of Mechanical and Aerospace Engineering. He received his M.S. degrees from Politecnico di Torino in 1989. His research interests include flight mechanics, unmanned aerial vehicles and space systems. He is Senior Member AIAA and Member AHS.

**Alessandro Rizzo** is an Associate Professor in the Department of Electronics and Telecommunications at Politecnico di Torino, Italy. He received the "Laurea" degree (summa cum laude) in computer engineering and the Ph.D. degree in automation and electronics engineering from the University of Catania, Italy, in 1996, and 2000, respectively. In 1998, he worked as EURATOM Research Fellow at JET Joint Undertaking, Abingdon, U.K., researching on sensor validation and fault diagnosis for nuclear fusion experiments. In 2000 and 2001, he worked as Research Consultant to ST Microelectronics, Catania Site, Italy, and as Industry Professor of Robotics at the University of Messina, Italy. From 2002 to 2015 he was a tenured Assistant Professor at Politecnico di Bari, Italy, where he conducted and supervised research on modeling and control of nonlinear systems, robotics, complex networks and systems, distributed estimation and control, and dynamical systems theory. In November 2015, he joined Politecnico di Torino. From 2012, he has also been a Visiting Professor at New York University Tandon School of Engineering, Brooklyn NY, USA. Prof. Rizzo is engaged in conducting and supervising research on complex networks and systems, modeling and control of nonlinear systems, cooperative robotics. He is author of two books, two international patents, and more than 100 papers on international journals and conference proceedings. Prof. Rizzo has been the recipient of the award for the best application paper at the IFAC world triennial conference in 2002 and of the award for the most read papers in Mathematics and Computers in Simulation (Elsevier) in 2009. Prof. Rizzo is also a Distinguished Lecturer of the IEEE Nuclear and Plasma Science Society.