International Journal of Intelligent Robotics and Applications (2022) 6:119–138
https://doi.org/10.1007/s41315-021-00195-y

REGULAR PAPER

Multi-objective offline and online path planning for UAVs under dynamic urban environment

Nassim Sadallah[1] · Saïd Yahiaoui[1] · Ahcene Bendjoudi[1] · Nadia Nouali-Taboudjemat[1]

Received: 13 October 2020 / Accepted: 10 August 2021 / Published online: 15 September 2021

Abstract
This paper presents a multi-objective hybrid path planning method MOHPP for unmanned aerial vehicles (UAVs) in urban dynamic environments. Several works have been proposed to find optimal or near-optimal paths for UAVs. However, most of them did not consider multiple decision criteria and/or dynamic obstacles. In this paper, we propose a multi-objective offline/online path planning method to compute an optimal collision-free path in dynamic urban environment, where two objectives are considered: the safety level and the travel time. First, we construct two models of obstacles; static and dynamic. The static obstacles model is based on Fast Marching Square (FM^2) method to deal with the uncertainty of the geography map, and the unexpected dynamic obstacles model is constructed using the perception range and the safety distance of the UAV. Then, we develope a jointly offline and online search mechanism to retrieve the optimal path. The offline search is applied to find an optimal path vis-a-vis the static obstacles, while the online search is applied to quickly avoid unexpected dynamic obstacles. Several experiments have been performed to prove the efficiency of the proposed method. In addition, a Pareto front is extracted to be used as a tool for decision making.

Keywords UAV path planning · Dynamic urban environment · FM^2 · A* · MOHPP

1 Introduction

In the last years, we have seen an emergence of the use of Unmanned Aerial Vehicles (UAV) with a variety of structures and shapes. Their extensive use has induced the rapid growth of related research areas, both in military and in civil fields, such as, security and surveillance (Ma'sum et al. 2013), delivery (Thiels et al. 2015), search and rescues (Doherty and Rudol 2007), and fire fighting (Casbeer et al. 2005). The use of UAVs is constantly increasing, especially in urban areas (Mohammed et al. 2014). This is why the UAV should be first safely and timely designated in accordance with the target field, what is commonly known as path planning problem.

Path planning is the key element to provide autonomy to UAVs in the execution of their mission, by determining a collision-free pathway between a UAV's current position and its destination, while satisfying some optimality criterion (Goerzen et al. 2010). Path planning has been widely studied and a large number of methods have been developed in last decades. However, most of them are not efficient in real world applications because of the dynamic, uncertain and changing nature of such environments. Mobile robots motion path planning problem can be divided into two groups: Optimized Classic Approaches, and Evolutionary and Hybrid Approaches (see Fig. 1) (Khaksar et al. 2015). For the Optimized Classic Approaches, the robot is considered as a single point in the space. It includes three subgroup methods, namely, the Potential field method (Khatib 1986), the Heuristic Search method (Knuth 1977) and the Sampling Based Algorithm (LaValle 1998). The Evolutionary and Hybrid Approaches are divided into two sub-group methods, namely the Artificial Intelligence and Hybrids. In the Artificial intelligence sub-group many evolutionary algorithms are included, as the Genetic algorithm (GA) (Yun

✉ Nassim Sadallah
nsadallah@cerist.dz

Saïd Yahiaoui
syahiaoui@cerist.dz

Ahcene Bendjoudi
abendjoudi@cerist.dz

Nadia Nouali-Taboudjemat
nnouali@cerist.dz

[1] CERIST Research Center on Scientific and Technical Information, Algiers, Algeria

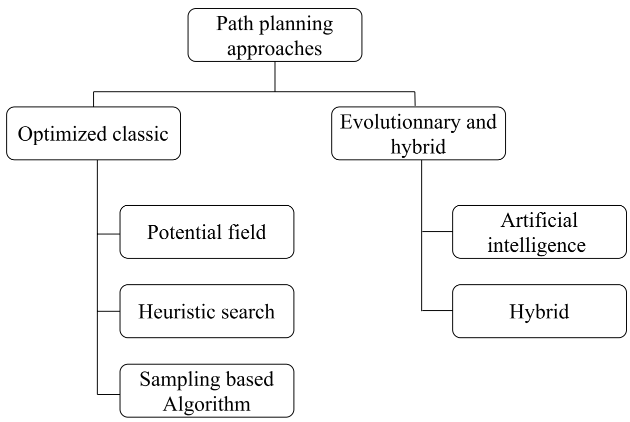


Fig. 1 Classification of path planning methods

et al. 2011), Ant Colony Optimization (ACO) (Di Caro and Dorigo 2004), Reinforced Learning (Jaradat et al. 2011) and Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995). The hybrid approaches combine different evolutionary algorithms for better result, we can cite the ACO-PSO (Gigras et al. 2015), the fuzzy logic with the PSO (PSO-Fuzzy), the genetic algorithm with fuzzy logic (GA-Fuzzy), the fuzzy logic with neural network (Neuro-Fuzzy) (Khaksar et al. 2013).

The most important criterion is the search mechanism which is responsible of defining a path that assumes the safety, optimal travel time and low energy consumption for the UAV and defines how it will navigate to the target position. It can be offline, online (Raja and Pugazhenthi 2012) or hybrid (Cao et al. 2018) that handle the optimized classic, and the evolutionary, and hybrid approaches. The offline search needs the knowledge of the static environment and the robot will execute the pre-planned path. With online search, the environment is unknown and the path is re-planned in real-time according to new information acquisition of the surrounding dynamic environment. The Hybrid search combines both offline and online search mechanisms to plan and re-plan a path within the dynamic environment.

Particularly, there are more challenges to adapt these methods to UAVs path planning. In the dynamic urban environment, a UAV would be possible to crash against the unexpected obstacles because of the dynamic and uncertain environment constraints. Although re-planning can be leveraged to mitigate the uncertainty during flying, the challenge is to generate a feasible trajectory in real-time with limited on-board computational resources and deal with the sudden changing in the surroundings within limited sensing range. In this paper, we propose to tackle this issue. Our solution is a multi-objective hybrid search path planning method to find an optimal path for a UAV flying in dynamic urbain environment, so that it can avoid any present obstacle, whether static or dynamic. The optimality of the trajectory is centered essentially around two objectives: the *travel time* and the *safety*.

The main contributions of this paper can be summarized as follows:

1. We use the graph properties and a uniform Cartesian deterministic sampling scheme to represent the urban environment in a 2D space. A grid map is constructed where each cell is mapped with a node of the graph to define the free spaces and capture both static and dynamic obstacles.
2. We design a risk map model to define the velocity and cost of each point in the environment. Specifically, a cost distribution map is constructed offline based on the Fast Marching Square [FM^2 (Valero-Gomez et al. 2013)] method which incorporates the static obstacles. And, a dynamic risk map is developed to define the unexpected obstacles that are not available in the geography map. The dynamic risk map is constructed online during the fly based on the safety distance and the perception range of the UAV.
3. A new method is proposed to solve the formulated multi-objective path planning problem where the travel time and safety are considered. Two algorithms are introduced for both search mechanisms (offline and online) to find the optimal path. The offline search is to plan a Pareto optimal path of avoiding static obstacles based on the cost distribution map. The online search is exploited to re-plan an optimal path avoiding the dynamic and unexpected obstacles based on the online constructed risk map of unexpected threats.
4. A real urban environment test is performed in addition to the simulation experiments to evaluate the efficiency of the MOHPP method in the real world.

This paper is structured as follows: in Sect. 2, we discuss the related work. In Sect. 3, we present the generic basics on which the proposed method is based. The detailed design and implementation of our method is presented in Sect. 4. We give the experimental results in Sect. 5 and conclude the paper in Sect. 6.

2 Related work

Recently, several works have been developed to address the multi-objective path planning problem for UAVs, and different path planning algorithms have been proposed. For example, in Wu et al. (2018), Yang et al. (2015), Macharet et al. (2010), Fu et al. (2012), multi-objective path planning methods (MOPP) for UAVs based on the evolutionary algorithms are proposed to find a path avoiding static obstacles in the environment. In Mittal and Deb (2007), authors proposed a 3D path planner for UAVs using multi-objective Evolutionary algorithms. Particularly, the NSGA-II algorithm with an objective function which considers the length and safety of the path. Meanwhile, a B-spline curved is applied to represent the generated path, making the control points as the decision variable of the genetic algorithm. In Roberge et al. (2013), authors attempted to compute the quasi-optimal paths in 3D complex environment using the genetic algorithm (GA) and the particle swarm optimization (PSO), reducing the execution time by adopting the "single program, multiple-data" parallel programming technique, which ensures a real-time solution. A multi-objective A^* is applied in Hernández-Hernández et al. (2014) to plan routes of UAV, where travel time, path angle, energy consumption and area deviation are considered. Authors in González et al. (2016) proposed a multi-objective approach for UAVs to find the optimal path in a 3D environment with static obstacles using the FM^2 method. In addition, authors in Chen et al. (2016) applied modified central force optimization algorithm to solve the MOPP problem in 3D environment. Furthermore, authors in Hao et al. (2017) developed a multi-objective path planner in 2D space where the elastic constraint method for multi-objective optimization is applied to find an optimal solution towards the travel time and the energy consumption.

These proposed methods (Wu et al. 2018; Yang et al. 2015; Macharet et al. 2010; Fu et al. 2012; Mittal and Deb 2007; Roberge et al. 2013; Hernández-Hernández et al. 2014; González et al. 2016; Chen et al. 2016; Hao et al. 2017) are practical to plan path but they consider only static obstacles without dynamic ones.

Other works are proposed to extend the solution to MOPP in dynamic environment. In Wu et al. (2011), a grid based multi-step A* is proposed for path planning in dynamic 4D environment (three spatial and one time dimensions) using a variable successor operator to find the least cost paths enabling the track length, angle and velocity trajectory. Furthermore, by leveraging variable successor operator to impose the multi-resolution lattices structure on the search space. Authors in Wen et al. (2017) considered uncertainties of threats, vehicles' motions and observations, and then designed an online path planning framework by integrating a sub goal selector, a sub task allocator and a local path planner. In Lin and Saripalli (2015), a multi-objective online path planning algorithm is proposed for UAVs collision avoidance. The algorithm is based on sampling intermediate way-point randomly and collision check using the reachable set.

All these methods (Wu et al. 2011; Wen et al. 2017; Lin and Saripalli 2015) focus only on the online path planning and do not explore offline informations as the geography map. Hence, the optimality of the global path is not guaranteed.

An hybrid path planning (offline and online) is proposed in Yang and Yoo (2018), authors developed a new optimal flight path planning mechanism for UAV data acquisition in wide IoT sensor networks by using multi-objective bio-inspired algorithms. In Pehlivanoglu (2012), authors proposed an evolutionary algorithm based Offline/Online path planner for UAV navigation in 3D rough terrain environment. In addition, in Primatesta et al. (2018) a hybrid approach is proposed to find minimum risk path for UAVs. A risk-map with assessment method combining layers related to the population density, the sheltering factor, no-fly zone and obstacles is generated to quantify the risk of a specific area. A risk-aware path planning based on the well-known Optimal Rapidly-exploring random tree is applied to retrieve the optimal path combining offline and online search. Furthermore, A multi-objective path planning (MOPP) framework is proposed in Yin et al. (2018) to explore a suitable path for a UAV operating in a dynamic urban environment.

Evolutionary algorithms cited in Yang and Yoo (2018), Pehlivanoglu (2012) suffer from the premature convergence problem, since the first population generates randomly the individuals (solutions) which lead to large quantities of unfeasible paths, and they cause more computational time or meaningless work. The computational time of the method in Primatesta et al. (2018) is high due to the Rapidly-exploring Random Tree (RRT*) method's characteristics. In Yin et al. (2018) the resulted path is zigzagging and not smooth, this causes the loss of efficiency of the method in the real world.

It can be seen that these proposed methods do not fully address the multi-objective path planning problem under the real urban environment. In order to remedy this, in our approach, multi-objective cost-determinant variables are taken into account (safety, travel time, and distance for online search). An hybrid (offline/online) path search mechanism is developed to derive the set of optimal paths considering both the static and dynamic obstacles of the real world.

3 Preliminaries

In this section, we define some generic bases on which the proposed method is based. We explain the model of the environment, the multi-objective path planning problem formulation and the Fast Marching Method, which is a base for the definition of the optimal path.

3.1 Environment modeling

The environment is described by a 2-D space graph represented by a three-tuple $\langle G, P_{start}, P_{target} \rangle$, where $G = (N, A, c)$ represents the space graph, N denotes a set of nodes (points) and $P_{start} \in N$ and $P_{target} \in N$ are, respectively, the set of the start nodes and target nodes. A is the set of arcs between

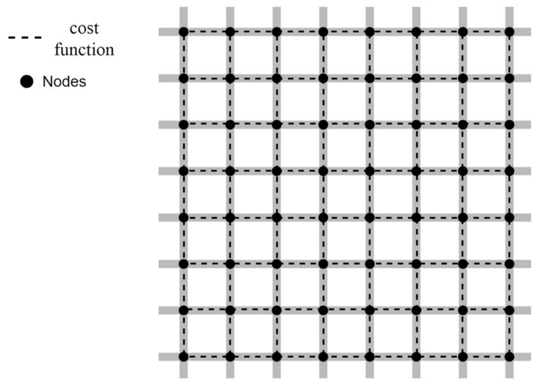


Fig. 2 2-D space graph

nodes of G, and $c : A \longrightarrow R^k$ represents the path cost function, where K is the number of objective functions. A uniform Cartesian deterministic sampling scheme is utilized to construct the graph G. As a result, each node $p \in N$ is mapped uniquely to a cell of the grid. Hence, the p refers simultaneously to both the cell and to a point located in the center of the cell (see Fig. 2).

3.2 Multi-objective path planning problem

The path planning under dynamic urban environment is constrained by a set of internal conditions (energy consumption, sensor's capacities,...) and a set of external conditions (buildings, no-fly zones, other UAVs, Birds,...). In our case, we chose to deal with these issues targeting two key objectives: **safety** and **travel time**.

Safety The safety objective is defined in this paper with the collision avoidance criterion which needs a separation distance from obstacles to provide a safe path to the UAV. So, the closer the point is from the obstacle, the lowest the safety level is. Furthermore, there is uncertainty in the position, velocity and direction of the dynamic obstacle because of potential sensor error which may lead to collision risk.

Travel time This is another crucial objective for the mission itself. The plan of the mission should be the shortest in time. The travel time is restricted typically by the cruise velocity at each node.

The solution to the multi-objective optimization problem (with two objectives) is to find a path p ,in a graph $G = (N, A, c)$, between a source and target points with the minimal total cost W. Let $c_{i,1}$ and $c_{i,2}$ be, respectively, the safety and travel time cost functions at arc $i \in A$. Let $l_j(p) = \sum_{i \in p} c_{i,j}, j \in \{1, 2\}$, be the total cost in a path for the j*th* objective.

For p to be an optimal path compared to any other path q in G, the following conditions should be verified:

$$\begin{aligned} &\forall j \in \{1,2\}, l_j(p) \leq l_j(q), \textit{and} \\ &\exists i \in \{1,2\}, l_i(p) < l_i(q) \end{aligned} \tag{1}$$

Solutions that are not dominated by any other solutions are Pareto Optimal (denoted by a Pareto set P). p is a Pareto optimal path of the Pareto optimal set P, which has the minimal total cost W with a weight coefficient α:

$$W = \min_{p \in P} \alpha . l_1(p) + (1 - \alpha) . l_2(p) \tag{2}$$

3.3 The fast marching method (FMM)

The fast marching method (FMM) is a particular case of the Level Set Methods developed initially by Osher and Sethian (1988). It is an efficient computational algorithm for modeling and tracking the motion of a physical wave interface (front) denoted as Γ. Within FMM, the front is called *interface*. The interface can be a flat curve [two-dimensional (2-D)] or a three-dimensional (3-D) surface, but the mathematical model can be generalized to n dimensions.

Considering a gridmap representing the real environment where obstacles are labeled 0 and free spaces 1. The FMM calculates the time T required for a wave to reach each point on the gridmap. The wave can be originated from more than one point, and each point generates a wave. The point where the wave is originated has a time $T = 0$.

In the context of Fast Marching, the front Γ is supposed to move in the normal direction with a non-negative speed value which can vary over time. At each instant, the front's motion is described with the Eikonal equation (Osher and Sethian 1988):

$$1 = F(x)|\nabla T(x)| \tag{3}$$

where x is the position, $F(x)$ is the expansion speed at that position, and $T(x)$ is the time the wave interface needs to reach x.

The magnitude of the gradient of the time function is inversely proportional to the velocity

$$\frac{1}{F} = |\nabla T| \tag{4}$$

The $T(x)$ function generated by a wave that expands from one source point has one global minima at the source and no local minima. As the expansion speed is positive ($F > 0$), the wave only grows. Hence, points farther from the source have a greater T (T is single valued as $F > 0$).

Osher and Sethian (1988) proposed a discret solution for the Eikonal equation. In 2-D, the area is discretized using a grid map. The intersection of row i and column j of the grid corresponds to a point $p(x_i, y_j)$ of the real environment, and the discretization of the gradient ∇T, according to Osher and Sethian (1988), leads to the following equation:

$$\left\{ \begin{array}{l} max\left(D_{ij}^{-x}T, 0\right)^2 + min\left(D_{ij}^{+x}T, 0\right)^2 \\ +max\left(D_{ij}^{-y}T, 0\right)^2 + min\left(D_{ij}^{+y}T, 0\right)^2 \end{array} \right\} = \frac{1}{F_{ij}^2} \tag{5}$$

According to Osher and Sethian (1988), a simpler but less accurate solution of (5) can be expressed as follows:

$$\left\{ max\left(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0\right)^2 + max\left(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0\right)^2 \right\} = \frac{1}{F_{ij}^2} \tag{6}$$

where

$$\begin{aligned} D_{ij}^{-x} &= \frac{T_{i,j} - T_{i-1,j}}{\delta x} \quad D_{ij}^{+x} = \frac{T_{i+1,j} - T_{i,j}}{\delta x} \\ D_{ij}^{-y} &= \frac{T_{i,j} - T_{i,j-1}}{\delta y} \quad D_{ij}^{+y} = \frac{T_{i,j+1} - T_{i,j}}{\delta y} \end{aligned} \tag{7}$$

δx and δy are the spacing grid in the x and y directions. Substituting (7) in (6) and letting

$$T = T_{i,j} \quad T_x = min\left(T_{i-1j}, T_{i+1j}\right) \quad T_y = min\left(T_{ij-1}, T_{ij+1}\right) \tag{8}$$

We can rewrite the Eikonal equation in a discretized 2-D space as follows:

$$max\left(\frac{T - T_x}{\delta x}, 0\right)^2 + max\left(\frac{T - T_y}{\delta y}, 0\right)^2 = \frac{1}{F_{ij}^2} \tag{9}$$

As we assume that the speed of the front is positive ($F > 0$), T must be greater than T_x and T_y whenever the front wave has not already passed the coordinates i, j.

We can solve (9) in three steps. First, we solve the following quadratic

$$\left(\frac{T - T_x}{\delta x}\right) + \left(\frac{T - T_y}{\delta y}\right) = \frac{1}{F_{ij}^2} \tag{10}$$

If $T > T_x$ and $T > T_y$ [taking the greater value of T when solving (10)], the obtained value is the correct solution for equation (9). Else, if $T < T_x$ or $T < T_y$, from equation (9), the corresponding member of $(\frac{T-T_x}{\delta x}, 0)$ is 0, and hence, (9) is reduced to

$$\left(\frac{T - T_x}{\delta x}\right) = \frac{1}{F_{ij}} \tag{11}$$

$$\left(\frac{T - T_y}{\delta y}\right) = \frac{1}{F_{ij}} \tag{12}$$

depending on the final value of T.

The equation (9) can be solved iteratively over a gridmap. To do so, the cells of the gridmap must be labeled as one of the following types:

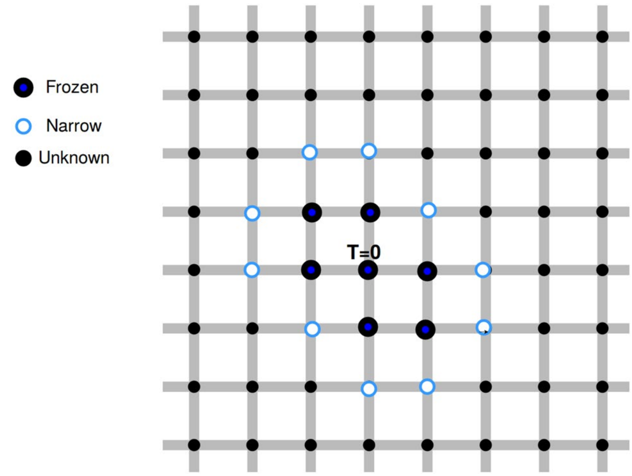


Fig. 3 Iterative wave expansion with one point source

- **Unknown** Cells with unknown value of T (the wave has not reached the point).
- **Narrow or Narrow band** Candidate cells to be part of the front wave in the next iteration. They are assigned a T value that can still change in the future iterations of the algorithm.
- **Frozen** Cells that have been passed over by the wave and, hence, their T value is fixed.

The algorithm (see Algorithm1) has the following steps (an illustration is associated to it in Fig. 3):

- **Initialization** The algorithm defines $T = 0$ for the set of cells where the wave is originated, these cells are labeled **Frozen**. Afterward, it labels all the Manhattan neighbors as **Narrow**, computing T for each of them.
- **Main loop:** In each iteration, the algorithm solves the Eikonal equation (9) for the Manhattan neighbors of the narrow cell (that are not Frozen) with the lesser T value, then label this cell **Frozen**. The narrow band maintains an ascendent ordered list of its cells according to the value T.
- **Finalization** When all the cells are **Frozen** (the narrow band list is empty), the algorithm finishes.

Algorithm 1: FMM (Fast Marching Method) algorithm

```
input  : Gridmap G of size n ∗ m; The set of origin cells (Ori) where the wave is
         originated
output: The gridmap G with the T value set for all cells
/*                        Initialization                          */
for g_ij ∈ Ori do
    g_ij.T ← 0;
    g_ij.state ← Frozen;
    for g_kl ∈ g_ij.neighbours do
        if g_kl.state NOT Frozen then
            g_kl.T ← SolveEikonal(g_kl); // Calculate the arrival time value for
               node g_kl
            if g_kl.state = Narrow then
                NarrowBand.updatePosition(g_kl);// update the g_kl position in
                   the list according to its T value
            else
                g_kl.state = Narrow ;
                NarrowBand.InsertInPosition(g_kl);// insert g_kl in the list
                   according to T value
/*                        Main loop                               */
while NarrowBand NOT EMPTY do
    g_ij ← NarrowBand.PopFirst() ;
    for g_kl ∈ g_ij.neighbours do
        if g_kl.state NOT Frozen then
            g_kl.T ← SolveEikonal(g_kl) ;
            if g_kl.state = Narrow then
                NarrowBand.UpdatePosition(g_kl) ;
            else
                g_kl.state = Narrow ;
                NarrowBand.InsertInPosition(g_kl) ;
```

4 Proposed method

The MOHPP method introduces first the cost distribution map which defines the velocity and the cost of each point in the environment. This map is used to plan a path that meets two objectives: **the safety** and **the travel time**. The cost distribution map is exploited by the offline search mechanism. The dynamic risk map is constructed online when detecting unexpected obstacles concerning both the range perception and the safety margin of the UAV. The dynamic risk map is used with the online search mechanism.

The solution for the multi-objective (Safety and Travel time) optimization problem is to find an optimal path p between two points on the graph $G(N, A, c)$. Every arc of A has two non-negative costs denoted $c_{i,t}$ and $c_{i,s}$ representing the travel time and the safety respectively. This is possible by exploiting the FM^2 and the A* algorithm for offline search and online search respectively.

4.1 Cost distribution map

The safety of the travel is an important criterion to ensure the completion of the mission. For this aim, we need to define the different obstacles that appear in the map considering the static ones. Hence, we attribute the allowed velocity and the cost at each point of the grid map. To solve this problem, we use a variant of the Fast Marching Method over the original map (grid map) which is the Fast Marching Square method (FM^2) (Valero-Gomez et al. 2013).

The result of FMM is a gridmap with T assigned for each point, but the value of the speed of each point remains zero (0) at obstacles, and one (1) in free spaces. However, we need different velocities on the different points: if the UAV passes near an obstacle, it requires a lower speed for the security. The Fast Marching Square (FM^2) assumes the relation between security distance and velocity.

The operating principle of the FM^2, as given in Valero-Gomez et al. (2013), lies in the fact that it applies the FMM twice. In the first application, over the gridmap, the FM^2 attributes a speed value for each point in the map (the line 1–2 of Algorithm 2). All the vertices of the obstacles are the wave source, and the T value represents the velocity. As result, a grid map with a relative speed value assigned to each point of the graph.

We need to define a separation distance from obstacles where the velocity should be reduced to guarantee the safety of the UAV. To do, we use a saturation coefficient α ($0.00 < \alpha < 1.00$) to define the safety level. By varying the value of α, the separation distance varies proportionally to the farthest distant point from obstacles (point with highest T value in the first execution of FMM). When $\alpha = 1.00$, the separation distance should be the same as the farthest distant point (the highest safety level considered). When $\alpha = 0.00$ implies that the safety level consideration is zero.

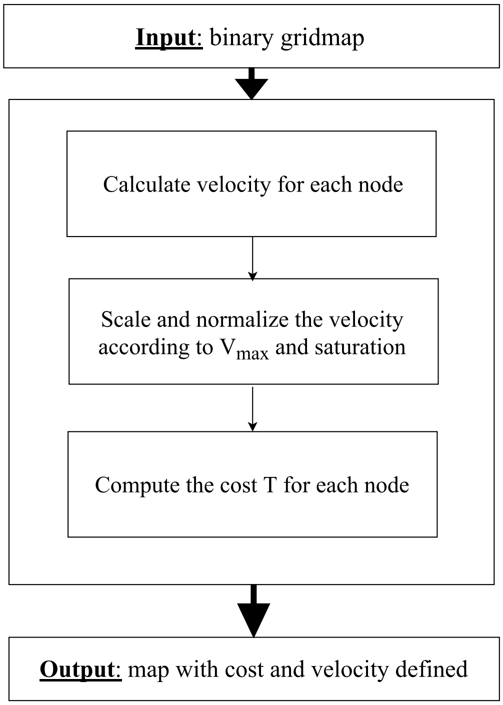


Fig. 4 Process of CDM construction

If we suppose that the T value of each point in the obtained map is the corresponding velocity (V), the latter should be scaled and normalized according to the maximum allowed speed (V_{max}) and the safety level. The scale formulation is given in Eq. (13) and the normalization in Eq. (14).

$$\forall p_{i=0,N} \in P, \qquad V_{pi} = \frac{T_{pi}}{T_{max}} \tag{13}$$

$$Vel_{pi} = \begin{cases} \dfrac{V_{pi}}{\alpha} * V_{max} & \text{if } V_{pi} < \alpha \\ V_{max} & \text{otherwise} \end{cases} \tag{14}$$

where P is the set of the points in the map, pi a point in P, T_{pi} the related defined velocity, V_{pi} the scaled velocity and Vel_{pi} is the normalized velocity.

The code implementation is given in line 3–11 of Algorithm 2.

The second application is to assign the arrival time (denoted T') for each point over the gridmap, after the saturation process, from a source point ps until reaching the destination pd (line 12 in Algorithm 2). The arrival time value (T') of obstacles is infinity. The hole process of cost distribution map (CDM) construction is illustrated in Fig. 4.

4.2 Dynamic risk map

The computational time of the process of defining static obstacles is relatively high for real time treatments. Hence, it is not suitable in case of dynamic and unexpected obstacles that need a real time reaction of the UAV. The solution

Algorithm 2: construction of the velocity and cost distribution map **CostDistributionMap**

input : Gridmap G of size $n * m$; the set of origine cells (Ori) where the wave is originated, set containing the obstalces of the map **obstacleSource**, start point **Start**, goal point **Goal**, saturation coefficient **Alpha**, maximum speed of the UAV **VELOCITY**

output: the grid map with velocities point defined

```
1  obstacleSource = set countaining the obstacles of the gridmap;
2  MapVelocity = FMM(obstacleSource, −1, G;) // set velocity at each point
   /* The saturation process */
3  T_max = the maximum T value of the grid map;
4  for node in the map do
5  |   velocity = ( node.T / T_max );
6  |   if velocity < Alpha then
7  |   |   node.velocity = (velocity/Alpha) * VELOCITY;
8  |   else
9  |   |   node.velocity = VELOCITY;
10 |   node.T′ = INFINITY;
11 |   node.state = UNKNOWN;
12 CDM = FMM(Goal, Start, MapVelocity;)// CostDistributionMap after
   saturation
13 return CDM;
```

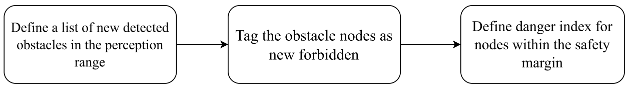


Fig. 5 Process of DRM construction

is to construct a dynamic risk map in real time, with low computation time, where the new obstacles will be spotted.

We construct a dynamic risk map (DRM) according to two criteria: the **perception range** and the **safety margin**. We define the dynamic risk map as a set of nodes, within the perception range of the UAV, that are identified as new threats by the embedded sensors. In addition, we defined a safety margin from the new obstacles at which the UAV should brake to avoid the collision. The process is illustrated in Fig. 5

1. **Perception range** UAV needs to detect surrounding environment while performing its mission. Let us denote R the perception range radius of the embedded sensors centered at the UAV. It can take the following form

$$\left(x_c - x\right)^2 + \left(y_c - y\right)^2 \leq R \tag{15}$$

where (x, y) is an arbitrary point within R, and (x_c, y_c) the current position of the UAV.

2. **Safety Margin** when an unexpected obstacle is detected the UAV takes an emergency brake as avoiding measurement. We call the emergency brake distance *Safety-Margin* (denoted d_{sm}). Next, the dynamic risk map is constructed according to the security distance:

$$I = \begin{cases} +\infty & d_c \leq d_{sm} \\ 0 & else \end{cases} \tag{16}$$

where d_c is the straight distance between the UAV and the unexpected obstacle, I is the **danger index** (points that have distance $d_c \leq d_{sm}$ take the highest value).

4.3 Offline search

To compute the global optimal path between p_{start} and p_{goal} over the cost distribution map, the maximum gradient direction should be followed from p_{start} to p_{goal}. As every cell has the lowest possible value T_i assigned, the path returned will be optimal. The maximum gradient direction is computed applying the Sobel operator over the grid map, and this from the start point until reaching the global minima which is the goal point.

Starting at the initial point, the path is calculated iteratively, the Sobel Operator calculates approximations of the gradients on the horizontal and vertical orientations (Eq. 17).

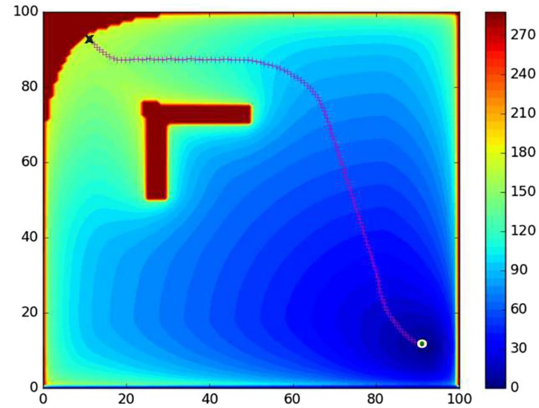


Fig. 6 generated path by applying gradient descent

$$grad_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * T \qquad grad_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * T \tag{17}$$

where T is the cost, $grad_x$ and $grad_y$ are the gradient values on the two dimensions x and y respectively.

At each point p_i, the gradient is computed on the X-direction ($grad_{ix}$) and Y-direction ($grad\ iy$), thus the magnitude and the direction (mod_i and α_i in Eq. 18) can be computed. From p_i is computed p_{i+1} successively until reaching the goal following the direction in which the time is increased the most.

$$mod_i = \sqrt{grad_{ix}^2 + grad_{iy}^2} \qquad \alpha_i = \arctan\left(\frac{grad_{iy}}{grad_{ix}}\right)$$
$$p_{(i+1)x} = p_{ix} + step.\cos(\alpha_i) \quad p_{(i+1)y} = p_{iy} + step.\sin(\alpha_i) \tag{18}$$

A highlight result is given in Fig. 6 where the maximum gradient is followed from a start point marked with black "X" in direction to the global minima marked with white "O".

4.4 Online search

The offline search mechanism guarantees the optimality of the path whatever the type of the static obstacle in the environment. However, when an unexpected obstacle appears in the detection area, the path will be ineffective and should be re-planned. The primary aim in this case is to generate quickly a path to avoid the unexpected obstacles. So, the best way to get the UAV out of the danger zone is to generate a local optimal path of avoiding. The offline search is done over the global map (without new obstacles), and, its computational time is relatively high which is not suitable for the real time treatments. As a solution, we developed an online search mechanism to generate a path of avoiding the dynamic threats within the detected area. The online search is based on the A^* algorithm, the cost distribution map informations to guide the search direction instead of the straight-line search, and a cut down mechanism to accelerate the search efficiency.

The A^* algorithm uses a cost heuristic function denoted $f(p)$ to identify the order in which the search treats the points in a tree, which is computed with

$$f(p) = g(p) + h(p) \tag{19}$$

where $g(p)$ is the real cost from the start to the current point, $h(p)$ is a heuristic cost from the current to the goal. The classic A^* utilizes a straight-line to the goal for the heuristic $h(p)$. In our case, we attribute to $h(p)$ the costs of nodes generated by the offline process. In addition, a search space cut-down mechanism is proposed to guarantee the efficiency of A^* within the ONPS mode even when the geography map is large, and the start point is far from the goal point.

We use a two-steps cut down mechanism: If the straight-line between the current point and the goal does not intersect with the dynamic threat zones generated by unexpected obstacles, then check if there is no dynamic obstacles within the UAV's safety range (d_{sm}). Hence, terminate the online search process. If the UAV's safety range (d_{sm}) contains the dynamic threats, the ONPS mode will re-plan a path to get far from that zone. When the goal point is far from the start point, the search space is reduced and the efficiency of online search process is enhanced. The pseudo-code of the online search process is summarized in Algorithms 3 and 4.

Algorithm 3: A^* algorithm **(AStar)**

```
input: start point Start, goal point Goal, cost distribution map CDM, new
       unexpected obstacle zone DynamicRiskMap, cut-down mode
/*        Initialization of different parameters and lists.              */
ClosedList = empty set;
OpenedList = priority queue containing START;
InvertedPath = empty set;
// List of path's nodes
G[Start] = 0;
F[Start] = Start.T;
/*        mode of cut down mechanism                                     */
Function cut_down_condition(mode):
  | if mode =1 then
  |   | return Intersect(current, Goal, DRM);
  | else
  |   | return within_dynamic_threats(current, Safety margin, DRM);
/*        the main loop of the A*                                        */
while OpenedList is not empty do
  | current = pop lowest rank item from OpenedList;
  | if current = Goal or cut_down_condition(mode) then
  |   | return inverted InvertedPath;
  | add current to ClosedList;
  | for n in neighbors of current do
  |   | if n is not an obstacle then
  |   |   | TentativeG = current.T + cost(current, n) ;
  |   |   | n if n in OpnenedList and TentativeG < G[n] then
  |   |   |   | remove n from OpenedList;
  |   |   | else if n in ClosedList and TentativeG < G[n] then
  |   |   |   | remove n from ClosedList;
  |   |   | else if n not in OpenedList and not in ClosedList then
  |   |   |   | G[n] = TentativeG;
  |   |   |   | F[n] = G[n] + CDM[n];
  |   |   |   | add n to OpenedList;
  |   |   |   | InvertedPath[n] = current;
return Failure;
```

Algorithm 4: Online Path Search algorithm **(ONPS)**

```
input : start point Start, goal point Goal, cost distribution map CDM, new
        unexpected obstacle zone DRM, detected new obstacle tragger
        isDetected, the global path PathGlobal, security margin SecMar
output: current position
OnlinePath = empty set;
current = Start;
OnlinePath = AStar(current, Goal, CDM, DRM, mode = 1);
while DRM not empty do
    if OnlinePath not empty then
        current = pop the first element from OnlinePath;
        add current to PathGlobal;
        DRM, isDetected = TheDetectedArea(current, SecMar);
        if isDetected and OnlinePathcrossed by a dynamic threat then
            OnlinePath = AStar(current, Goal, CDM, DRM, mode = 1);
    else
        OnlinePath = AStar(current, Goal, CDM, DRM, mode = 2);
        if OnlinePath is empty then
            DRM = empty set;
            isDetected = false;
            break;
return the current position;
```

The global method (as given in Algorithm 5) which groups the different algorithms to plan the path takes as inputs the geographic map, the start and goal points' coordinates, the maximum velocity of the UAV, the desired saturation coefficient, and the safety margin for the dynamic risk map. The first step consists of generating the cost distribution map by the first execution of the FMM method. After that, the offline search generates the global optimal path, the start point is added to the executed path, current position is initialized to start and the boolean variable which defines if an unexpected obstacle is detected is set to *false*. Then the first movement is done by the UAV (lines 2–4 in Algorithm 5). The while loop is the main process that allows the UAV to move to the target position (lines 7–19 of Algorithm 5). In this loop, a process of checking the presence of unexpected obstacles in the sensed environment is done. If no unexpected obstacle appears, the next position is the one given by the offline search. Else we construct the corresponding dynamic risk map and switch to the online search, hence the UAV will follows the different positions calculated in real time until it avoids the new obstacles or reaches the goal.

Algorithm 5: Main MOPP method

input: start point **S**, goal point **G**, binary grid map **MAP**, the maximum velocity **VMax**, saturation to obstacles **SAT**, safety margin **SafMar**

```
CDMAP=CostDistributionMap(MAP, ObstacleSource, S, G, SAT, VMax);
PlanPath = OFPS(S, G, CDMAP);
ExecutedPath = queue containing Start;
current = S;
isReplanned= False;
while current is not G do
    if isReplanned = False then
        current = pop from PlanPath;
    else
        isReplanned = False;
        PlanPath = OFPS(current, G, CDMAP);
        continue;
    append current to Executed;
    DynamicRiskMap = empty set;
    isDetected = False;
    DynamicRiskMap, isDetected = TheDetectedArea(current, SafMar);
    if isDetected then
        current =
          ONPS(current, Goal, CDMAP, ExtendedNewObstalces, SafMar);
        isReplanned = True;
```

5 Experimental results

In this section, we discuss results obtained from different experiments that have been done as part of the evaluation of our approach. Namely, the synthetic simulation on the graphical interface, and the Software-In-The-Loop experiment on a quadri-rotors. In addition, a real test is done with a real quadri-rotors model X (video available in[1]). We use specific parameter setting for each environment and a quadri-rotors model X for real tests.

5.1 Parameter setting

We conducted two types of experiments to evaluate our method. The Algorithms are programmed with Python language in both cases. The implementation steps are as follows:

Setup for simulated scene

- **Hardware** We used a computer desktop with Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz and 16 GB of memory.
- **Simulated environment ad map generation** The operating system used was Linux Ubuntu 16.04 LTS. We used TKinter library from Python to generate the simulated environments, where two binary maps with 200×150 lattices are generated with resolution of Dx:Dy = 1:1 pixel (corresponding to 1:1 m). We used matplotlib

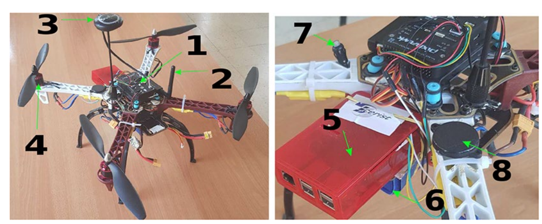


Fig. 7 Quadri-rotors model X

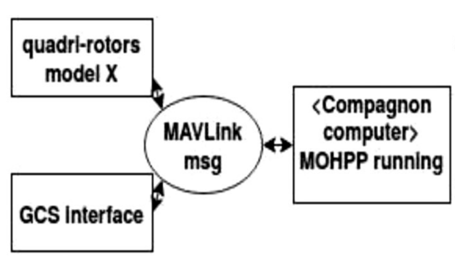


Fig. 8 Communication scheme

[1] https://www.youtube.com/channel/UCnO1PwAHU-SP2jDYca36nww

library in Python to visualize the different simulation results (velocity profile, travel time and safety level).

- **Parameters setup for the MOHPP method** Our method takes as parameters a saturation coefficient to define a safety level ($0.00 < \alpha < 1.00$), maximum velocity as the property of the maximum speed of the UAV (set to 1 m/s) and a safety margin that the UAV should take when detecting an unexpected obstacle (set to 4 m).

Setup for the real scene

- **Hardware** We used a quadri-rotors model X equipped with a Pixhawk card controller and Ardupilot system as firmware. A Raspberry-Pi (RPi3 Model B) as a companion computer. We use the DroneKit-API on the RPi3 to communicate commands with the Pixhawk controller via MAVlink messages. The MOHPP method is implemented on the companion computer. To visualize and track the UAV movements, we used the telemetry radio to send messages to the Mission Planner ground control station. Pictures of the UAV are given in Figs. 7 and 8 illustrates the communication scheme, and Table 1 gives the details of the hardware components.
- **Realistic environment definition** In this setup, the environment is a region on the Satellite Map with resolution of 200 ×150 m. We extracted a binary map from the Satellite map and divided it into disjoint lattices with resolution set to Dx:Dy=1:1 pixel (corresponding to 1:1 meter of the Satellite map). All the buildings are pointed out as static obstacles.
- **Parameters setup for the MOHPP method** Our method takes the same number of parameters as in the simulated scene; a saturation coefficient to define a safety level ($0.00 < \alpha < 1.00$), maximum velocity as property of the maximum speed of the UAV (set to 1 m/s) and a safety margin at which the UAV should brake when detecting unexpected obstacle (set to 5 m). The coordinates of the start and end points are fixed, respectively to (75, 99) and (50, 20).

5.2 Evaluation

In the following, we evaluate the ability of our approach to deal with the static and dynamic obstacles of the urban environment. First, we evaluate the influence of the saturation coefficient on the safety level and the travel time over the static obstacles. Second, we evaluate the effect of the perception range on the online search mechanism to avoid the unexpected obstacles.

5.2.1 The simulated scene

In this setup, we generate two binary maps of scale 200 lattices × 150 lattices, the lattice resolution is 1:1 m.

- Effects of the saturation In this experiment we constructed a cost distribution map with different saturation values α (the original map is shown in Fig. 9a) in order to evaluate the influence of the saturation on the velocity and the length of the path. The result shown in Fig. 10 illustrates the final result of the first wave propagation of FM^2 and the saturation process (formulated in Eqs. 13 and 14) to define the relative speed at each point in the map according to the value of α (we precise that the lighter the color is, the closer the velocity is to the maximum allowed value). In Fig. 10a with α=0.15, we can observe that the points closest to the obstacles in the map have a reduced velocity and the others at the maximum velocity. Compared to results in Fig. 10c with $\alpha = 0.75$, more points have reduced velocity, this is due to the larger separation distance defined by α where the velocity should be reduced.

We called the MOHPP method hundred (100) times with different α values to obtain the safety level and the travel time tradeoff curve shown in Fig. 11 where the X-coordinates are the safety level and the Y-coordinates represents the travel time profile. The blue curve in Fig. 11 is the efficient path set P got from the execution of the MOHPP, and the red curve is the Pareto-optimal path set. The safety index of a point n (denoted I_n) is calculated according to the first propagation of the wave. Indeed, as the obstacles are the wave source expansion, the value of I_n is 0 and increases proportionally to the wave propagation. Thus the farthest point has the highest safety index. The total safety index of a path $p \in P$ is expressed as follows:

$$I_p = \sum_p I_n$$

Furthermore, we calculate the risk index as: $\log_{10}(I_p)$.

The total travel time t of a path p (with distance $d = 1$) is expressed as follows:

$$t_p = \int_p \frac{1}{Vel_n} dn$$

where Vel_n is the cruise velocity at point n.

Figure 12 shows the final Cost Distribution Map (after the second application of the FM^2 over the velocity map illustrated in Fig. 10) with the arrival time T set for each point. The coordinates of the start point (marked with the red 'X') and the goal (marked with pink 'X') are respectively (90, 10) and (70, 120). We can see the effect of the defined velocity on the wave propagation speed. Indeed, when α==0.15 (shown in Fig 10a), most of the points are at their maximum velocity, this allows a high wave propagation speed which

Table 1 UAV components details

1-Pixhawk controller	3-GPS module	5-RPi3 model B	7-Safety switch
2-Telemetry radio	4-Brushless motors 920KV	6-LiPo battery	8-Buzzer

makes the arrival time cost T low. Compared to results in Fig. 12c, the propagated wave takes a longer time to arrive at the destination, this is due to the reduced cruise velocity assigned to the points (Fig. 10c) which reduces the speed of the wave propagation.

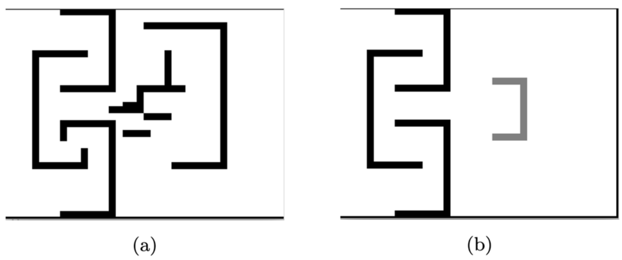


Fig. 9 On (**a**), a binary map with static obstacles (black). On (**b**), a binary map with both static and dynamic (grey) obstacles

The results of the execution of OFPS algorithm over the illustrated cost distribution maps in Fig. 12 are plotted in Fig. 13. The blue curve in the figure is the trajectory and the cyan circle is the detection range of the UAV. The corresponding travel time for the different saturation coefficient is given in Table 2. As shown, different trajectories with their velocity profiles are computed depending on the value of α. Indeed, the aim of the OFPS is to follow the maximum gradient direction from a start point to the destination with the global minima $T = 0$. To do so, the direction that reduces the T value should be chosen at each step, which is referred, simultaneously, to the neighbor with the highest cruise velocity. We can see in Fig. 13a, with $\alpha = 0.15$, the path goes near obstacles as the maximum gradient is achieved at a low separation distance. In addition, the cruise velocity profile (shown in Fig. 13d) is almost at the maximum allowed (1 m/s) due to the high consideration of travel time criterion in relation to safety level. That leads to shorter

Fig. 10 Saturated velocity map according to values of α

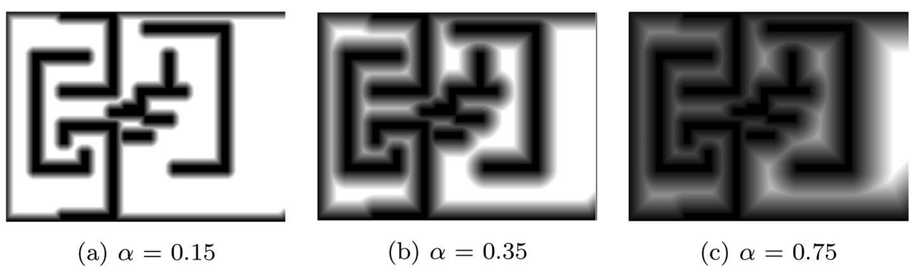
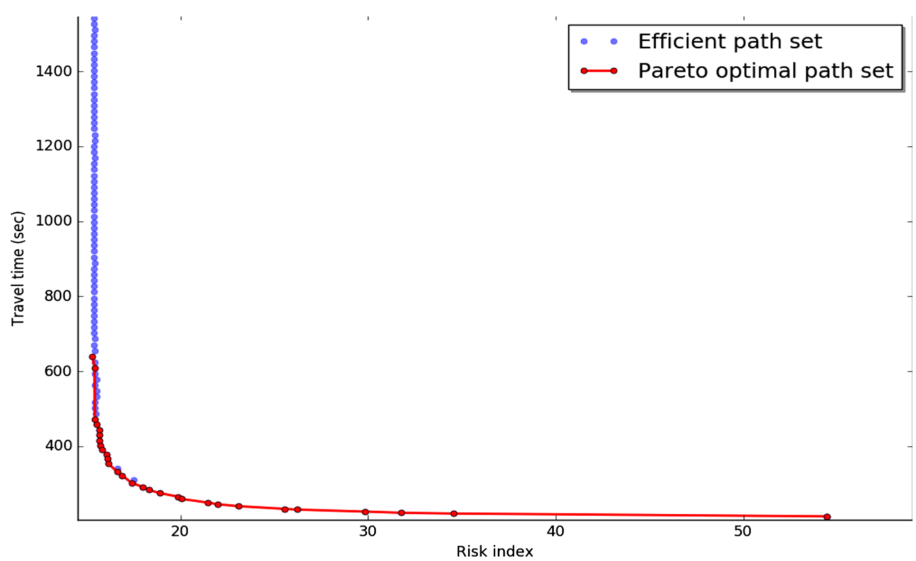
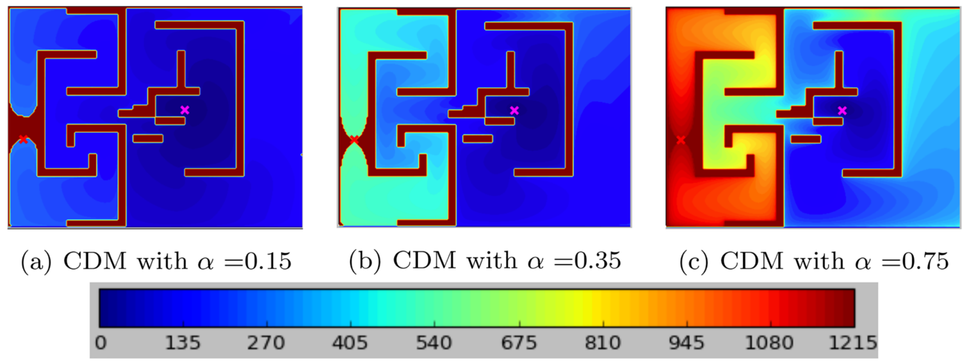


Fig. 11 Travel time and safety level tradeoff curve under a synthetic obstacles scenario

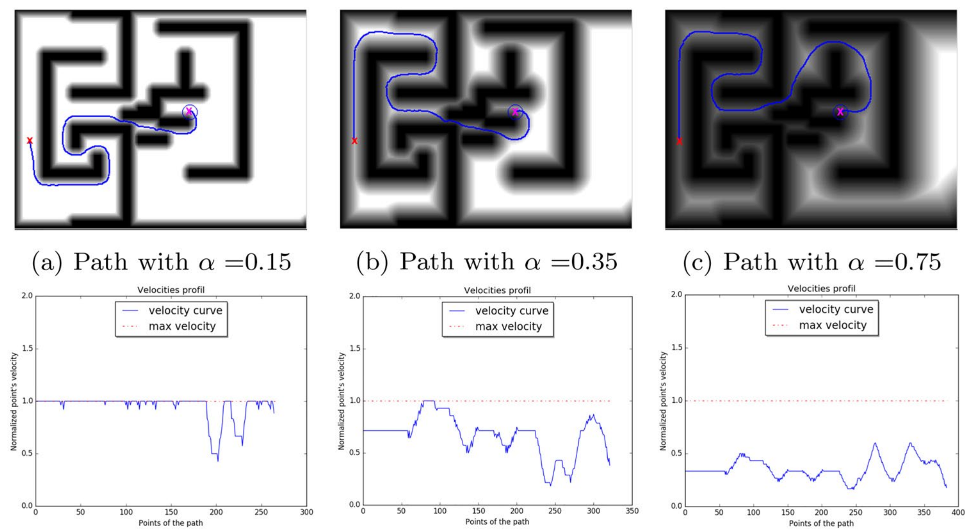




(a) CDM with α =0.15 (b) CDM with α =0.35 (c) CDM with α =0.75

(d) colors reference into values (seconds)

Fig. 12 CDM map with arrival time T assigned at each point



(a) Path with α =0.15 (b) Path with α =0.35 (c) Path with α =0.75

(d) ($\alpha = 0.15$) (e) ($\alpha = 0.35$) (f) ($\alpha = 0.75$)

Fig. 13 OFPS results and the corresponding velocity profiles

travel time (290s) and high risk index (17.70). Compared to Fig 13c, the path goes far from obstacles, as α defines a large safety distance which implies the maximum gradient to be achieved only with points that are far from obstacles as they have the lowest T and the highest cruise velocity. Furthermore, the cruise velocity profile, in Fig 13f, is almost under the maximum allowed (1 m/s), due to the high safety level consideration. This results in longer travel time and path length (1167 s) with a low risk index (5.86).

We can resume that shorter travel time leads to higher risk level, and high coefficient value results to safer trajectory with longer travel time which is due to the contradiction of two objectives. Fortunately, the MOHPP method can effectively return an optimal path that corresponds to the needed criteria weighting.

- Effects of the perception range In this experiment, we evaluate the ability of our approach to deal with unexpected obstacles. The original map is given in Fig. 9b where an unexpected obstacle is involved. The coordinates of the start and the end points are respectively (85,50) and (85,180). In addition, we set α to 0.3.

Table 2 Travel time and Risk index numerical results

Saturation	α= 0.15	α= 0.35	α= 0.75
Travel time	290 S	546 S	1167 S
Risk index	17.70	12.62	5.86

Figure 14, Fig. 15 depict intermediate results of avoiding known and unexpected obstacles by MOHPP method. In these figures, the circle with cyan color is the perception range, with safety margin of 4 m. The black color indicates the presence of static obstacles, the grey color is the unexpected obstacle, and the maroon color is the detected

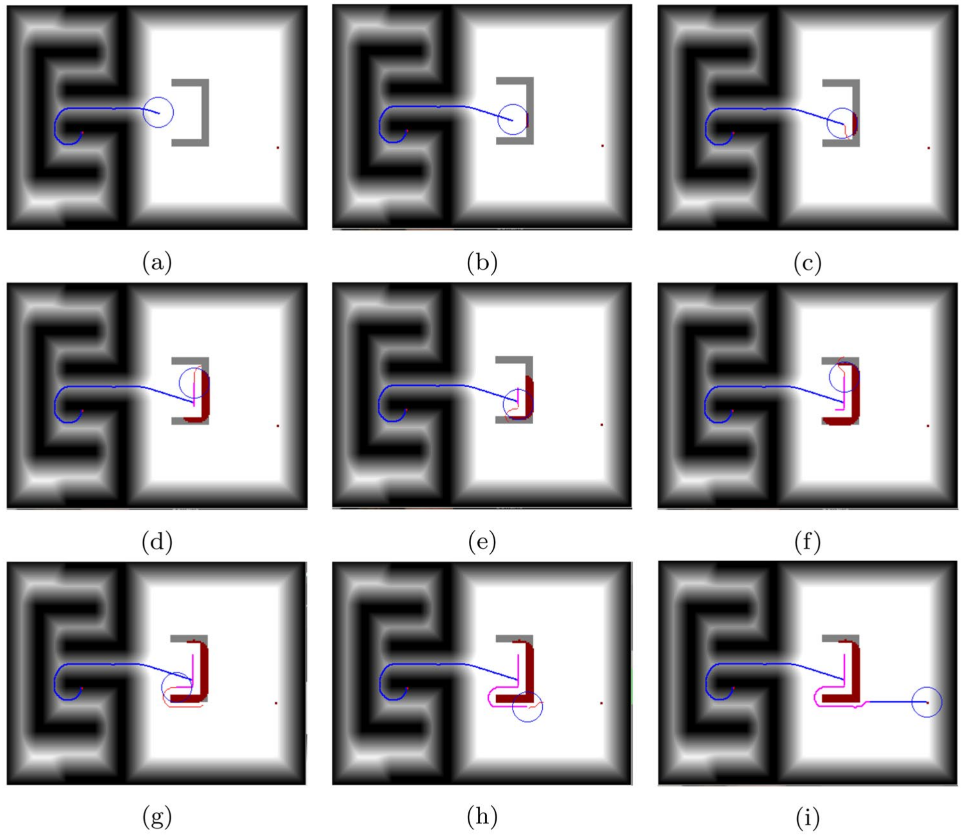


Fig. 14 Process of exploring a path with perception range = 10 m

dynamic threat zone. The blue curve is the offline generated path and the magenta color is the online path followed by the UAV. The red lines denote the computed trajectory by A^* algorithm.

The results show that the method can effectively avoid known and unknown obstacles, as we can see, the UAV succeeds to detect the unexpected obstacle and re-plan a path that keeps a safety distance which is defined as safety margin.

We note that the surface of the detected area influences directly the online search process. Actually, the ONPS algorithm computes a local optimal path of avoiding the obstacle depending on the perception range radius. When the sensed area is large, more nodes are updated and defined. If they are obstacles, then, the ONPS mechanism computes a path of bypassing with few iterations by executing A^* algorithm (see Fig. 15). With a small sensed area, more iterations are necessary to re-plan the bypassing path which is became unfeasible when new the threats are detected (see Fig 14).

Table 3 highlights the correlation between the number of iterations and the computation time of the A*, and the computation time of constructing the DRM map within the ONPS mode when the perception range changes. We can see that the number of iterations decreases and the computation time of A* and DRM increases depending on the perception range. This is due to the number of nodes in the sensed area on which ONPS invokes A*. From range of 30 m, the number of iterations and the run-time of A* remains the same, because the whole threat is identified. In addition, the computation time of A^* in one iteration within the ONPS mode is less than 3 ms.

In summary, the ONPS mode depends on the perception range of the embedded sensors. The higher the radius, the better we detect the unexpected obstacle and walk around it with few iterations. This implies more computation time. The smaller the radius, the better we reduce the computation time. This implies, more iterations as a small surface can be detected. The efficiency of the cut-down mechanism in ONPS is that the user can choose the desired sensors regardless the real-time constraint since the computation time is always low. That proves the efficiency of our method to deal with the dynamic obstacles in real-time.

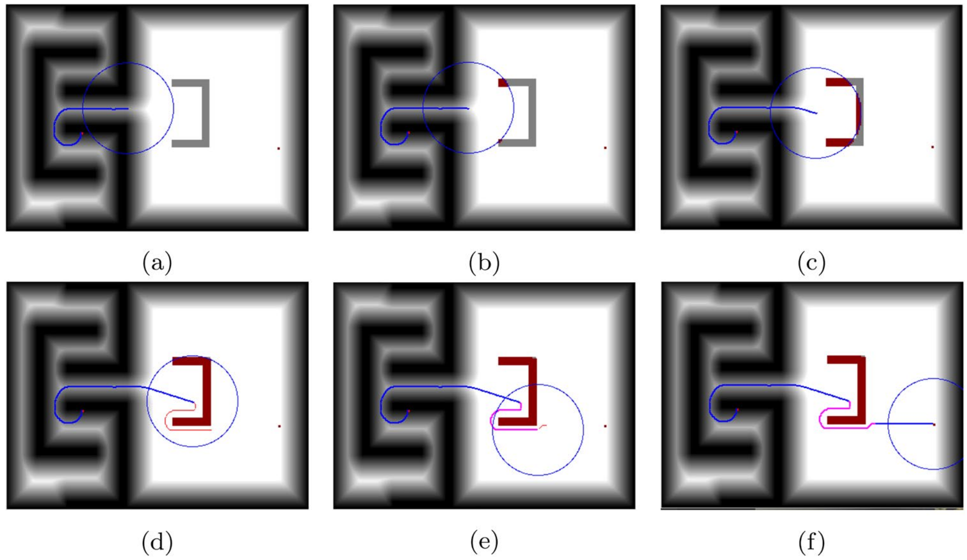


Fig. 15 Process of exploring a path with perception range = 30 m

Table 3 Running time of the A* and DRM process

Perc. range	Max runtime DRM	Calling times(A*) in ONPS	Max runtime(A*)
10 m	0.037 ms	38	0.31 ms
20 m	0.062 ms	8	1.02 ms
30 m	0.071 ms	3	2.79 ms
40 m	0.212 ms	3	2.79 ms
50 m	0.371 ms	3	2.79 ms
60 m	0.480 ms	3	2.79 ms
70 m	0.532 ms	3	2.79 ms
80 m	0.617 ms	3	2.79 ms
90 m	0.807 ms	3	2.79 ms
100 m	0.898 ms	3	2.79 ms

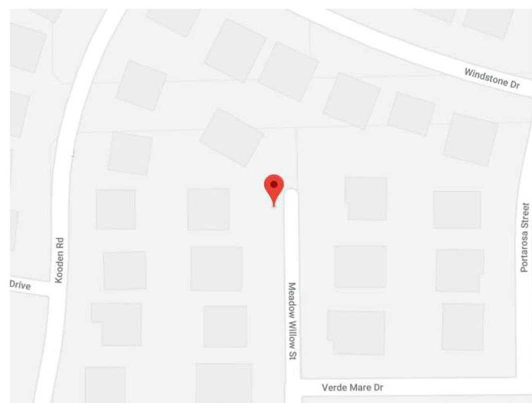


Fig. 16 Map from Google maps

5.2.2 Evaluation of MOHPP in a real scene

In this setup, we conducted this experiment under an emulated urban scene using SITL (Software In The Loop) technology, as illustrated in Fig. 16, to certify the feasibility of the MOHPP method in the real world. This scenario is a region's plan of Satellite Map (33°36′56.9″ N 117°06′47.9″ W). The map is divided into disjoint lattices with resolution of 1:1 m and all the buildings are extracted. We built a scheme from the real map as shown in Fig. 17. We set the coordinates of the starting point of the UAV to (75*th* vertex, 99*th* vertex) and the target point of the UAV at (50*th* vertex, 20*th* vertex).

We executed MOHPP hundred (100) times with different α value (α from 0.01 to 0.99) to test the effect of the saturation coefficient through the realistic obstacles. In this experiment, the offline search process is tested. We extracted the results to the tradeoff curve in Fig. 18. Two paths are illustrated in Figs. 19 and 20, while the numerical results are reported in Table4.

Figures 19 and 20 show the experimental test results versus the simulated scene with saturation value of 0.2 and 0.4 respectively. Figure 19a and 20a are the outputs (from the GCS interface) of the followed paths by the quadri-rotors, whilst Figs. 19b and 20b are the outputs of the simulation tests.

The OFPS search produces interesting results. The real UAV succeeds to follow the same trajectory as the one generated from the schematic diagram for the simulated scene according to the saturation coefficient. In addition, the defined cruise speed at each point is almost respected (with negligible errors) as illustrated in Fig 21. With α=0.2, a short path is generated at a small distance from buildings with the most of points at their maximum velocity (Fig. 21a). With α

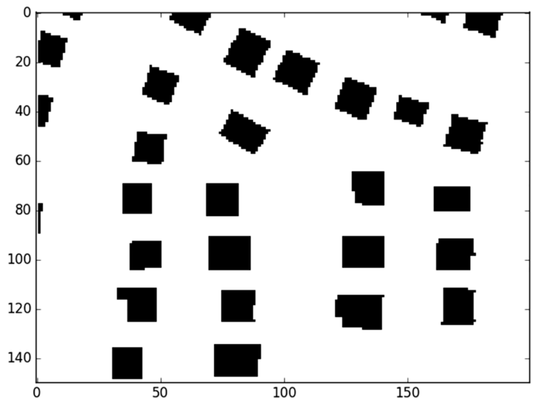


Fig. 17 schematic diagram of the realistic obstacles

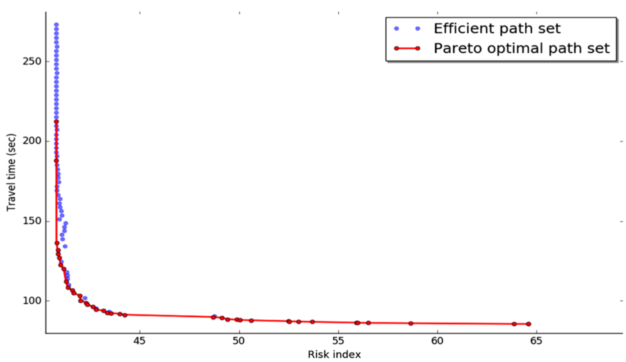


Fig. 18 Travel time and safety level tradeoff curve under a realistic obstacles scenario

=0.4, a longer path is computed far enough from buildings, and a low cruise velocity is set (Fig. 21b).

Numerical results in Table4 support that a small value of α generates a path with a minimum travel time and a higher risk index. In other hands, a minimum risk index leads to a longer travel time. In addition, the UAV flight time is little higher than the theoretical computed one, this is due to the accumulated small errors of the UAV's cruise speed (see Fig 21), also, the data exchange process between the companion computer and the flight controller.

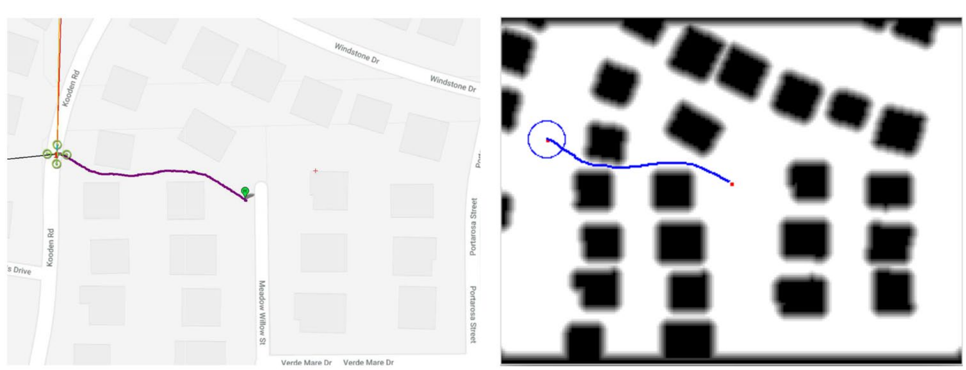
Comparing our solution with the proposed one by Yin et al. (2018) it's clear that our method is best in term of correlation between the set of criteria and the smoothness of the path. Indeed, the method in Yin et al. (2018) uses the simple bi-variate Gaussian model and the FM method to define the cost of points. The omitted results show that their solution is unable to define a sufficient distance from obstacles even when the safety is highly considered. Moreover, the generated path is zigzagging near obstacles which make it unfeasible. Our approach clearly assumes the needed safety and travel time thanks to the saturation mechanism and the FM^2.

5.2.3 Discussion of the results

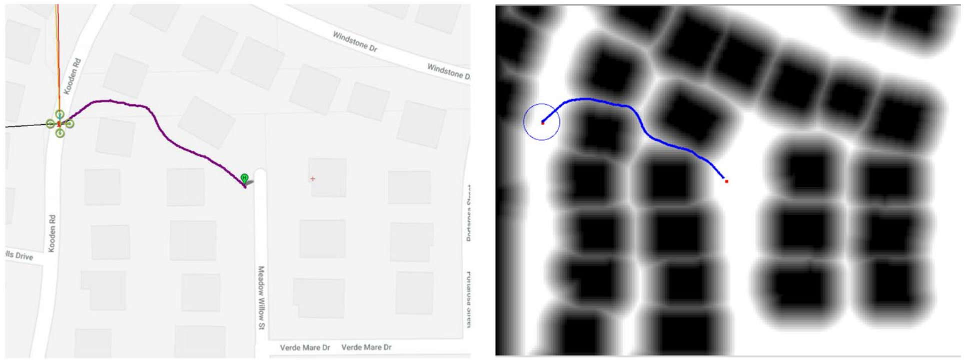
As a summary of the tests done, we can note the following observations:

- MOHPP method needs the coefficient α to determine the non-dominated solution over the travel time and the safety level criteria according to Eqs. 1 and 2. Explicitly, high value of α implies high security and longer travel time. Low value of α corresponds to shorter travel time and low security level.
- According to the results of the real experiment done with a micro quadrotor (the most used UAVs' category in smart cities) with low acceleration, the theoretical calculated paths and the followed ones are restrictively the same in term of positioning and localisation (as shown in Figs. 19 and 20). A small error is observed for the real flight in relation to the cruise velocity which is due to the embedded material estimation error. Nevertheless, embedding MOHPP into a micro quad UAV guarantees better experience within real smart cities as the similarity=98.3% between the real flight time and the simulated one.
- The introduction of the enhanced cut-down mechanism to reduce the space search, also by exploring the offline

Fig. 19 Generated path: Experimental test vs. Simulation test with α=0.2



(a) (b)



(a) (b)

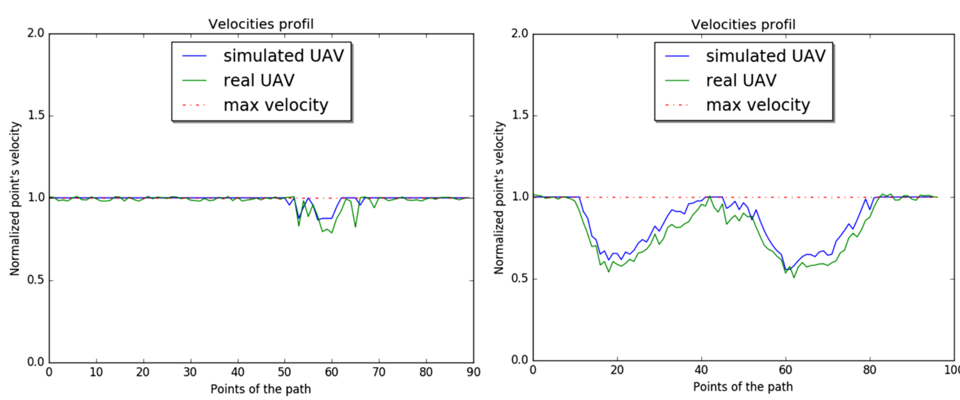
Fig. 20 Generated path: Experimental test vs. Simulation test with α=0.4

Table 4 Numerical results of the experimental tests

α	Solve time(FM^2)	Path length	flight time	UAV flight time	risk index
0.2	1.59 s	88	88.5 s	91.3 s	44.69
0.4	2.17 s	97	121.2 s	124.7 s	40.78

cost to orient the search within the ONPS (discussed in Sect. 5.2.1), can lead the UAV rapidly finding the optimal local path with less memory occupation and low computation time. As supported by results in Table 3, with max run-time of DRM (< 1 ms) and A* (< 3 ms) the agility of the UAV will be in real-time with the unexpected threats.

- The method relies heavily on the on-board sensors to avoid the unexpected obstacles. Indeed, the detection area depends on the type of sensors and their perception range. If the perception range is larger, then the detected area is large and the probability to find an optimal local path with few online process iterations is high.
- The changing of the perception range has no effect on the UAV path obtained through the CDM because the OFPS mechanism uses the prior complete knowledge of known obstacles to generate a global optimal path. Nevertheless, the computational time of the online search mechanism is affected by the perception range; as the number of iterations is reduced when the detection range is high, and vice versa.
- The MOHPP method can always determine an optimal path whatever the types of obstacles (static and dynamic). For example, when the UAV carries a sensitive or dangerous product, a safe trajectory is suitable; thus, our method can find an optimal path set with high safety level (Pareto-optimal path set with risk index value under 42 in Fig. 18). When the delivery concerns a bag of emergencies, the UAV must reach the destination soon while guaranteeing a minimum safety level. So, our method can also satisfy this requirement as it offers a Pareto-optimal path set with the shorter travel time (Pareto-optimal path set with travel time less than 80 seconds in Fig. 18). In other hands, when both high safety level and optimal travel time are required, our approach can supply a Pareto-optimal path set that meets these expectations (Pareto-optimal path set in Fig. 11 where risk index is between 42 and 45 and travel time is between 80 and 120 s).



(a) saturation = 0.2 (b) saturation = 0.4

Fig. 21 Real UAV's cruise speed VS. theoretical cruise speed: Fig. 21a corresponds to the test illustrated in Figs. 19, and 21b to the one in Fig. 20

6 Conclusion

This paper deals with the path planning problem for UAV under dynamic urban environment. A method, called MOHPP, concerning two objectives, namely travel time and safety level, has been proposed. In this method, a cost distribution map is established offline to indicate the main static obstacles in the geography map, and define the cost of each point in the map. A dynamic risk map is constructed when detecting unexpected obstacles during the fly. Then a joint offline and online search algorithm has been developed to plan a path to the destination. The offline search is exploited to find the shortest path with static obstacle avoidance based on the offline cost distribution map, and the online search is exploited to avoid unexpected obstacles according to the temporarily online dynamic risk map. The different results of synthetic and realistic experiments (under SITL) show that the MOHPP method is efficient to return always the optimal path, which meets the safety level and the shortest travel time.

As future work, we will focus on the extension of the MOHPP method to the dynamic 3D environment. Given the large number of nodes to be treated in such an environment, it would be necessary to optimize the response time of the method. In addition, we will work on enhancing the accuracy of the wave propagation along the diagonal direction to reduce the errors of the Eikonal equation. Furthermore, we will try to integrate other optimization criteria, such as, the wind direction and fuel consumption, to allow the method to handle more environmental factors and improve its efficiency in a real world environment.

References

Cao, X., Yang, P., Alzenad, M., Xi, X., Wu, D., Yanikomeroglu, H.: Airborne communication networks: a survey. IEEE J. Sel. Areas Commun. **36**(9), 1907–1926 (2018)

Casbeer, D.W., Beard, R.W., McLain, T.W., Li, S.-M., Mehra, R.K.: Forest fire monitoring with multiple small uavs. In: Proceedings of the 2005, American Control Conference, vol. 5, pp. 3530–3535. IEEE (2005). https://doi.org/10.1109/ACC.2005.1470520

Chen, Y., Yu, J., Mei, Y., Wang, Y., Su, X.: Modified central force optimization (mcfo) algorithm for 3d uav path planning. Neurocomputing **171**, 878–888 (2016)

Di Caro, G., Dorigo, M.: Ant colony optimization and its application to adaptive routing in telecommunication networks. Dissertation, PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium (2004)

Doherty, P., Rudol, P.: A uav search and rescue scenario with human body detection and geolocalization. In: Orgun M.A., Thornton J. (eds.) AI 2007: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol. 4830. Springer, Berlin, Heidelberg (2007).https://doi.org/10.1007/978-3-540-76928-6_1

Eberhart, R., Kennedy, J.: Particle swarm optimization. In: Proceeding of Ieee International Conference on Neural Network. Perth, Australia pp. 1942–1948 (1995)

Fu, Y., Ding, M., Zhou, C.: Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for uav. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **42**(2), 511–526 (2012)

Gigras, Y., Choudhary, K., Gupta, K., Vandana.: A hybrid aco-pso technique for path planning. In: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1616–1621 (2015)

Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. J. Intell. Rob. Syst. **57**(1–4), 65 (2010)

González, V., Monje, C.A., Moreno, L., Balaguer, C.: Fast marching square method for uavs mission planning with consideration of dubins model constraints. IFAC-PapersOnLine **49**(17), 164–169 (2016)

Hao, Y., Li, B., Shao, L., Zhang, Y., Cui, J.: Multi-objective path planning for unmanned aerial vehicle based on mixed integer programming. In: Chinese Automation Congress (CAC), 2017, pp. 7035–7039. (2017). https://doi.org/10.1109/CAC.2017.8244046

Hernández-Hernández, L., Tsourdos, A., Shin, H., Waldock, A.: Multi-objective uav routing. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 534–542 (2014). https://doi.org/10.1109/ICUAS.2014.6842295

Jaradat, M.A.K., Al-Rousan, M., Quadan, L.: Reinforcement based mobile robot navigation in dynamic environment. Robot. Comput.-Integr. Manuf. **27**(1), 135–149 (2011)

Khaksar, W., Hong, T.S., Khaksar, M., Motlagh, O.R.E.: A genetic-based optimized fuzzy-tabu controller for mobile robot randomized navigation in unknown environment. Int. J. Innov. Comput. Inf. Control **9**(5), 2185–2202 (2013)

Khaksar, W., Vivekananthen, S., Saharia, K.S.M., Yousefi, M., Ismail, F.B.: A review on mobile robots motion path planning in unknown environments. In: 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), pp. 295–300. IEEE (2015). https://doi.org/10.1109/IRIS.2015.7451628

Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Cox, I.J., Wilfong, G.T. (eds.) Autonomous Robot Vehicles. Springer, New York, NY (1986). https://doi.org/10.1007/978-1-4613-8997-2_29

Knuth, D.: A generalization of Dijkstra's algorithm. Inf. Proc. Lett. **6**, 1–7 (1977)

LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. pp. 98–111 (1998)

Lin, Y., Saripalli, S.: Collision avoidance for uavs using reachable sets. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 226–235 (2015). https://doi.org/10.1109/ICUAS.2015.7152295

Macharet, D.G., Neto, A.A., Campos, M.F.M.: Feasible uav path planning using genetic algorithms and bézier curves. In: da Rocha Costa, A.C., Vicari, R.M., Tonidandel, F. (eds.) Advances in Artificial Intelligence – SBIA 2010. SBIA 2010. Lecture Notes in Computer Science, vol. 6404. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16138-4_23

Ma'sum, M.A., et al.: Simulation of intelligent unmanned aerial vehicle (uav) for military surveillance. In: 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 161–166 (2013). https://doi.org/10.1109/ICACSIS.2013.6761569

Mittal, S., Deb, K.: Three-dimensional offline path planning for uavs using multiobjective evolutionary algorithms. In: 2007 IEEE Congress on Evolutionary Computation, pp. 3195–3202 (2007). https://doi.org/10.1109/CEC.2007.4424880

Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., Jawhar, I.: Uavs for smart cities: opportunities and challenges. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 267–273 (2014). https://doi.org/10.1109/ICUAS.2014.6842265

Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988)

Pehlivanoglu, Y.V.: A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous uav. Aerosp. Sci. Technol. **16**(1), 47–55 (2012)

Primatesta, S., Cuomo, L.S., Guglieri, G., Rizzo, A.: An innovative algorithm to estimate risk optimum path for unmanned aerial vehicles in urban environments. Transport. Res. Proc. **35**, 44–53 (2018)

Raja, P., Pugazhenthi, S.: Optimal path planning of mobile robots: A review. Int. J. Phys. Sci. **7**(9), 1314–1320 (2012)

Roberge, V., Tarbouchi, M., Labonté, G.: Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. IEEE Trans. Ind. Inf. **9**(1), 132–141 (2013)

Thiels, C.A., Aho, J.M., Zietlow, S.P., Jenkins, D.H.: Use of unmanned aerial vehicles for medical product transport. Air Med. J. **34**(2), 104–108 (2015)

Valero-Gomez, A., Gomez, J.V., Garrido, S., Moreno, L.: The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories. IEEE Robot. Autom. Mag. **20**(4), 111–120 (2013)

Wen, N., Su, X., Ma, P., Zhao, L., Zhang, Y.: Online uav path planning in uncertain and hostile environments. Int. J. Mach. Learn. Cybern. **8**(2), 469–487 (2017)

Wu, P.P.Y., Campbell, D., Merz, T.: Multi-objective four-dimensional vehicle motion planning in large dynamic environments. IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) **41**(3), 621–634 (2011)

Wu, J., Wang, H., Li, N., Yao, P., Huang, Y., Yang, H.: Path planning for solar-powered uav in urban environment. Neurocomputing **275**, 2055–2065 (2018)

Yang, P., Tang, K., Lozano, J.A., Cao, X.: Path planning for single unmanned aerial vehicle by separately evolving waypoints. IEEE Trans. Rob. **31**(5), 1130–1146 (2015)

Yang, Q., Yoo, S.J.: Optimal uav path planning: Sensing data acquisition over iot sensor networks using multi-objective bio-inspired algorithms. IEEE Access **6**, 13671–13684 (2018)

Yin, C., Xiao, Z., Cao, X., Xi, X., Yang, P., Wu, D.: Offline and online search: Uav multiobjective path planning under dynamic urban environment. IEEE Internet Things J. **5**(2), 546–558 (2018)

Yun, S.C., Parasuraman, S., Ganapathy, V.: Dynamic path planning algorithm in mobile robot navigation. In: 2011 IEEE Symposium on Industrial Electronics and Applications, pp. 364–369 (2011). https://doi.org/10.1109/ISIEA.2011.6108732

Nassim Sadallah received his Master degree in Computer Project Management in computer science in 2018. He is a research assistant at the Research Center on Scientific and Technical Information (CERIST, Algiers). His research interests include Multi-Objective Path Planning Optimization (MOPP), Drones (UAVs, SUVs, and Rover), Mobility in Smart Cities, Graph Algorithms.

Saïd Yahiaoui is a full-time researcher with the Research Center on Scientific and Technical Information (CERIST, Algiers). He received his Magister degree in networking and distributed systems in computer science from Abderrahmane Mira University (Bejaia) in 2005 and his Ph.D. in computer science from Claude Bernard Lyon 1 University in 2013. Between 2007 and 2010, he worked as a research assistant at CERIST. His research interests include Wireless Networking, Parallel and Distributed Computing, Graph Algorithms, Big Data, and Mobility in Smart Cities.

Ahcene Bendjoudi received his Master and Ph.D. degrees in Computer Science from the University of Bejaia, Algeria, and his HDR and state engineering degrees in Computer Science from the High School of Computer Science (ESI). He is senior researcher at the Research Centre in Scientific and Technical Information (CERIST, Algiers) within Theory and Engineering of Computer Systems division (DTISI). His major research interests include Parallel Combinatorial Optimization, Cluster, Multi-core and GPU-Computing, and recently, Big Data, Big Graphs, Intelligent Transportation Systems, and Smart Cities. He conducted and he was involved in several research projects on these topics. He has more than 35 international publications including journal papers, book chapters and conference proceedings.

Nadia Nouali-Taboudjemat is a permanent full-time senior researcher (Director of Research) at the Research Center on Scientific and Technical Information (CERIST) in Algiers, where she is leading the Theory and Engineering of Computer Systems Division (DTISI) and the Ubiquitous Systems Group (UbiSys). She obtained the Engineer degree, the Magister degree, and the Ph.D. in computer science from the University of Science and Technology (USTHB), the Advanced Technologies Research Center and USTHB respectively, all in Algiers, Algeria. Her recent focus includes Big Data, Big Graphs and HPC. Her background includes pervasive/ubiquitous computing, context-aware computing, distributed/parallel and mobile computing, data and transaction processing, wireless networks, Cloud computing and security. ICT-based crisis and disaster management, cybersecurity, IoT, smart cities and sustainable environments are her favorite application domains.