A Robust Real-Time Path Planner for the Collision-Free Navigation of Multirotor Aerial Robots in Dynamic Environments

Jose Luis Sanchez-Lopez[1] and Jesus Pestana[1] and Pascual Campoy[1]

***Abstract*— The development of deliberative capabilities is required to achieve an intelligent fully autonomous behavior of unmanned aerial systems. An important deliberative capability is the generation of collision-free paths in complex environments.**

This paper presents a robust real-time collision-free path planner used for the horizontal 2D navigation of multirotor aerial robots in dynamic environments. Its design, using geometric primitives to describe the environment combined with a launching time generation of a probabilistic roadmap graph, permits an efficient management of dynamic obstacles. The use of an A* discrete search algorithm, together with a potential field map as the cost function, allows to speed up the collision-free path computation ensuring that it never falls in local minima. Additionally, the velocity and acceleration along the collision-free planned path is calculated.

The performance of the proposed path planner is evaluated in this paper with two simulations with complex environments including a labyrinth and dead ends, and with a real flight experiment where three fully autonomous aerial robots executed an emulated search and rescue mission.

The proposed path planner has been released to the scientific community as an open-source software included in Aerostack[2]. In addition, it has extensively been used in multiple research projects with real flights, demonstrating its good performance.

I. INTRODUCTION

To achieve a fully autonomous behavior of unmanned aerial systems, the development of deliberative capabilities is essential. This deliberation capability permits to work in future time, generating executive actions to be performed, taking into account the current state of the robot and environment, and predicting the consequences of the planned actions in the future.

A. Planning System of Aerostack

Aerostack, presented in [1], [2], is a modular multipurpose software framework for aerial robotic systems, specially focused on multirotor platforms. Aerostack proposes a multi-layered system architecture divided in seven subsystems.

The Planning System of Aerostack, shown in Fig. 1, has the responsibility to generate goals, represented as executive actions and skills, to accomplish a particular complex mission. It has to react to changes in the operation provided by the lower-level layers and to unexpected operation events, generating new goals that modify the previously produced ones.

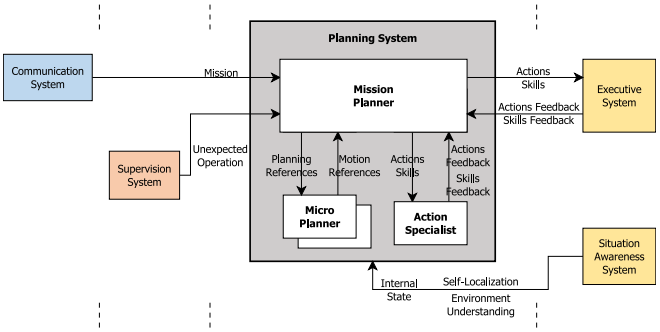


Fig. 1. General description of the Planning System of Aerostack.

The Planning System has to be able to generate goals fast enough to make possible an efficient reaction to changes in the mission, in the environment, or in the state of the robot. This is specially critical in aerial robots, since their unstable nature disqualifies them to passively wait a slow response of the Planning System.

The Planning System is formed by three components:

- *Mission planner*. It is a deliberative component that receives as input a mission to be performed and generates as output a sequence of executive actions to be executed together with their corresponding required skills.
- *Action specialist*. It helps the mission planner during the deliberation to anticipate whether a tentative action is feasible.
- *Micro planners*. They generate motion references that can be followed by the robot, given the current (self-localization, internal state, and environment understanding) and desired (planning references) state of the robot and the environment map. A trajectory planner (Section I-B) is a kind of micro planner, that generates trajectory references.

B. Trajectory Planner

The trajectory planner generates deliberative motion references for the aerial robot. These motion references might include pose references, velocity references and acceleration references (among others) with or without time constraints.

In common under-actuated multirotor aerial robots, only four degrees of freedom are controllable, normally their heading (yaw angle) and their position. Additionally, common under-actuated multirotor aerial robots are normally symmetric in their horizontal dimensions, and therefore,

This research work has been partially supported by the Spanish Ministry of Economy and Competitiveness through the project VA4UAV (Visual autonomy for UAV in Dynamic Environments), ref. DPI2014-60139-R.

[1] Jose Luis Sanchez-Lopez, Jesus Pestana, and Pascual Campoy are with Computer Vision and Aerial Robotics (CVAR), Centre for Automation and Robotics (CAR), UPM-CSIC (Technical University of Madrid), Calle Jose Gutierrez Abascal 2, 28006 Madrid, Spain jl.sanchez@upm.es

[2] www.aerostack.org

changing their heading angle is not needed to reach a particular position point, and consequently, the commanded heading and position can be decoupled, so the pose path planner might be divided in:

- *Position path planner*. It generates collision-free position references that can be followed by the aerial robot, given its current and desired position, and the environment description.
- *Heading path planner*. It generates heading motion references that must be followed by the aerial robot. For example, the yaw value can be specified as a point to look or as a yaw to look. This can be used with different objectives, like maximizing the performance of the on-board sensors (for example a camera), or to achieve a particular goal (for example to watch over some target).

It is important to note that common under-actuated multirotor aerial robots are holonomic vehicles, so they do not present any path planning constraint.

C. State of the Art

Path planning for robotic applications in general, is a very well studied topic since the beginning of the robotics, some of them focusing on aerial robotics [3].

Algorithms that try to search the collision-free path over a graph (discrete search algorithms), initially a grid map, have been used a lot. Examples of these algorithms are the famous single-query A* algorithm, [4] [5], and multi-query variants like the D* algorithm, [6].

As the discrete search algorithms have high computational requirements, other works proposed to use a potential field map (also called Artificial Field Maps, AFM) approach, where the final point has an attractive potential and the obstacles have a repulsive potential, [7]. The main disadvantage of this method is that the search easily falls into local minima, being unable to find a solution. Many works are based on AFM, trying to avoid their limitations. [8] proposed a simulated annealing in AFM; [9] combined AFM with genetic algorithms (GA); [10] merged AFM with optimal control theory; and [11] joined the AFM with model predictive control for fixed-wing UAV path planning

Other approaches used visibility graphs to reduce the computational cost. [12] defined a variant called Dynamic Visibility Graphs (DVG). [13] proposed the combination of a visibility graph with a multi-population genetic algorithm.

Rapidly-exploring random trees (RRT), presented in [14], proposed a novel single-query planning method that tries to cover the search space using random samples that are connected to the tree.

Probabilistic Roadmaps (PRM), presented in [15], designed a multi-query method that create a collision-free graph of random nodes on the search space. Multiple improvements over this method have been done [16]: [17] presented the Lazy-PRM; and [18] presented the C-PRM.

Other techniques use optimization and search algorithms like the Particle Swarm Optimization (PSO) algorithm to calculate the optimum path planning. [19] used a multi-objective PSO, whereas [20] fused a genetic algorithm with a PSO for UAV path planning. [21] used a multi-objective evolutionary algorithm on a grid map.

D. Contributions and Outline

In this work, we present a robust real-time collision-free path planner used for the horizontal 2D navigation of multirotor aerial robots in dynamic environments. The proposed path planner is capable to find collision-free paths more efficiently than the deterministic sampling-based approaches. It provides real-time capabilities when used in changing dynamic environments, unlike its probabilistic sampling-based counterparts. Moreover, it never falls in local minima, providing a robust solution.

The performance of the proposed path planner has been widely demonstrated thanks to its intensive usage in multiple research projects with real flights. It is integrated in Aerostack[3], [1], [2], and it is available to the scientific community as an open-source software.

The remainder of the paper is organized as follows: in Section II, the proposed path planner algorithm is presented, while in Section III its performance is evaluated both with simulations and with a real experiments. In Section IV, the main weaknesses of the proposed path planner are discussed, analyzing the way to overcome them and pointing out some lines of future work. Finally, Section V concludes the paper.

II. PROPOSED PATH PLANNER DESCRIPTION

This section presents our proposal for the fast generation of collision-free paths, able to interact with dynamic environments.

There exist multiple cases where the environment might be simplified to a 2D one. For example, the robot might be commanded to navigate maintaining a certain altitude, being therefore the environment considered as 2D for this flying altitude.

A. Environment description

The first step required to create a collision-free path is the adequate definition of the environment. The environment is described by means of its boundaries (Section II-A.1), the objects it has (Section II-A.2), and the moving agents that are navigating there (Section II-A.3).

The generation of this environment description is carried out by the Situation Awareness System of Aerostack [1], [2], but the presented planning component imposes the kind of descriptor required.

Whenever a path planner query is requested, the proposed path planner generates an internal obstacle map based on the environment description, as the junction of the static objects map and the moving agents map.

1) Environment boundaries: The environment boundaries might have any shape and therefore the arena is not required to be rectangular.

[3] www.aerostack.org

2) Objects of the environment described with geometric primitives: The objects of the environment are described as a set of uniquely labeled geometric primitives. The advantages of using geometric primitives instead of, for example a grid square cell map, point clouds, or surface representations, is that the environment is more compactly described but without loss of resolution.

Reducing the information used to describe the environment limits the number of queries to the cost function that evaluates the distance to an obstacle, speeding up the planning algorithm.

For simplicity, our proposed approach only uses two kinds of geometric primitives for the description of the objects of the environment: rectangles, and ellipses. The parameters that describe the geometric primitives are:

- Position of the center of the reference frame attached to the object in world coordinates, $\boldsymbol{t}_{O\star}^{W}$.
- Angle of the reference frame attached to the object in world coordinates, $\theta_{O\star}^{W}$.
- Dimensions of the object, $\boldsymbol{r}_{O\star} = [r_x,\ r_y]^T$. For ellipses, their radii are used; while for rectangles, the half of their sides are used.

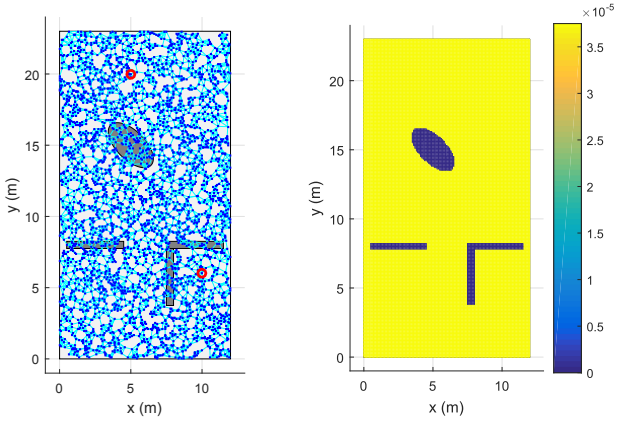
In case that the environment has objects that do not match any of the previously described geometric primitives shapes, the objects of the environment might be described, with a loss of resolution in the representation, by using either a circumscribed rectangle or ellipse of the generic object, or a combination of rectangles and ellipses to match the shape of the generic object.

3) Moving agents: Similarly to the static objects map, the moving agents (like other aerial robot agents) are added to a moving agents map using the previously described geometric primitives, but considering them as temporary objects. The size of the geometric primitives of the moving agents includes, not only the actual size, but also a fattening that depends on the movement (e.g. the velocity in module and direction) of these agents. In addition, if the moving agents are far enough (distance is greater than a threshold) from the current position of the agent that is planning the path, they are not added to the moving agents map, since they will most likely leave their current position before the agent reaches that point.

B. Arena sampling using a Probabilistic Roadmap

To reduce the time that the planner requires to plan a collision-free path, the arena is probabilistically sampled using a probabilistic roadmap (PRM). This probabilistic roadmap (also called graph) samples the arena (considering only its previously known boundaries), creating n random points (also called nodes or vertices) following a uniform distribution in the arena boundaries. The n sampled points are connected by means of edges to their nearest m neighbors (called m-neighborhood). The connection between any of two points of the probabilistic roadmap has to fall completely inside the arena.

The probabilistic roadmap is built at launching time and therefore no knowledge of the obstacles of the environment or the moving agents is incorporated to this probabilistic roadmap (see an example on Fig. 2a), unlike [15]. Creating a probabilistic roadmap that does not incorporate the knowledge of the fixed obstacles of the environment slows down the planning time because it will check points that are in collision, but it allows to handle previously unknown obstacles with a dynamic mapping. The previously known knowledge of the static objects of the environment can be incorporated to the probabilistic roadmap by modifying the uniform sampling function of the node generation into a custom probability density function (as depicted in Fig. 2b).



(a) Probabilistic roadmap without any previous knowledge about the obstacles. The red circles represent the initial and final points.

(b) Example of a probability density function for the PRM generation incorporating previous knowledge of the environment.

Fig. 2. Probabilistic roadmap (PRM).

The number of nodes n of the PRM must be representative of the environment. If the environment does not have a large number of obstacles, and the obstacles do not create corridors where the aerial robot must enter, the number of nodes n might be reduced. In the case of an environment populated with many obstacles, the number of nodes must be high. The higher the number of nodes, the higher the path planning time.

Whenever a planning query is requested, the initial point and the final point of the query are temporary added to the probabilistic roadmap, removing them after the creation of a collision-free path.

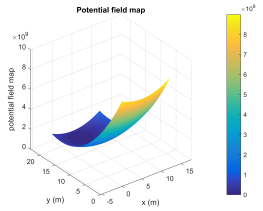
C. Potential Field Map as a cost function

The potential field map (PFM), see [7], is adopted as the cost function (see Section II-D) used by the planner to guide the search of the collision-free path in the probabilistic roadmap (as described in Section II-E).

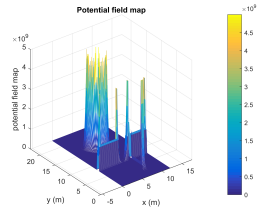
The total potential field map, $p_T = p_T\,(P)$, in a point P of the arena, is given by (see Fig. 3c):

$$p_T = p_q + p_O = p_q + \sum_{\forall i} p_{O_i} \tag{1}$$

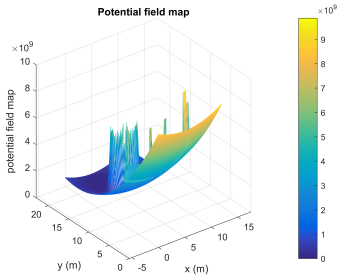
where p_q is the contribution of the planning query, p_O is the contribution of all the obstacles; and p_{O_i} is the contribution of the obstacle (or agent) O_i.



(a) Contribution of the planning query to the potential field map.rst



(b) Contribution of the obstacles to the potential field map.



(c) Total potential field map.

Fig. 3. Potential field map for a query from $t_{P_0} = [10,\ 6]^T$ and $t_{P_f} = [5,\ 20]^T$, with two rectangular obstacles, and an elliptical obstacle.

1) PFM of the planning query: To guide the search in the direction from the initial point P_0 to the target point P_f, a potential field map cost function is created, p_q. Both initial and target points are given in world coordinates, $\boldsymbol{t}_{P_0}^W$ and $\boldsymbol{t}_{P_f}^W$, respectively.

The contribution of the query to the potential field map, $p_q = p_q(P, P_0, P_f)$, in a point P of the environment, is given by a parabolic expression (see Fig. 3a):

$$p_q = \frac{\left(x_P^W - x_{P_f}^W\right)^2}{c_a} + \frac{\left(y_P^W - y_{P_f}^W\right)^2}{c_b} + k_f \tag{2}$$

being

$$c_a = \frac{k_r \cdot \left(x_{P_0}^W - x_{P_f}^W\right)^2 + \left(y_{P_0}^W - y_{P_f}^W\right)^2}{k_r \cdot (k_0 - k_f)} \tag{3}$$

$$c_b = k_r \cdot c_a \tag{4}$$

where the coefficient k_r determines the priority of the planned movement in an specific direction; the coefficient k_0 determines the altitude (cost) of the potential field map in the initial point; and the coefficient k_f determines the altitude (cost) of the potential field map in the final point. All these coefficients determine the behavior of the planner.

2) PFM of the obstacles: The contribution of the obstacle $O\star$ to the potential field map, $p_{O\star} = p_{O\star}(P, O\star)$, in a point P of the environment (see Fig. 3b), is calculated using its implicit equation v as:

$$p_{O\star} = \begin{cases} \infty, & \text{if } v \leq 0 \\ 0, & \text{if } v > v_{ig} = \frac{1}{k_2} \cdot \log\left(\frac{1}{c_{ig}} - 1\right) \\ \frac{k_1}{1+e^{k_2 \cdot v}}, & \text{otherwise} \end{cases} \tag{5}$$

being k_1 and k_2 two coefficients that determine the behavior of the path planner (in getting close to the obstacles); and c_{ig} a percentage value that indicates the contribution of the object $O\star$ in the potential field map, being ignored if its contribution to the cost is lower than $c_{ig} \cdot k_1$, as it's considered to be far enough.

The implicit equation $v = v(P, O\star)$, in a point P of the environment described in world coordinates by $\boldsymbol{t}_P^W = [x_P^W,\ y_P^W]^T$, generated by the obstacle $O\star$, is given by:

- Implicit equation of an ellipse:

$$v = \left(\frac{x_P^{O\star}}{\hat{r}_x}\right)^2 + \left(\frac{y_P^{O\star}}{\hat{r}_y}\right)^2 - 1 \tag{6}$$

- Implicit equation of a rectangle:

$$v = \left|\frac{x_P^{O\star}}{\hat{r}_x} + \frac{y_P^{O\star}}{\hat{r}_y}\right| + \left|\frac{x_P^{O\star}}{\hat{r}_x} - \frac{y_P^{O\star}}{\hat{r}_y}\right| - 2 \tag{7}$$

where Eq. (6) and Eq. (7) are transformed from object-relative coordinates to world coordinates by means of:

$$\boldsymbol{t}_P^{O\star} = \left(\boldsymbol{R}_{O\star}^W\right)^T \cdot \left(\boldsymbol{t}_P^W - \boldsymbol{t}_{O\star}^W\right) \tag{8}$$

and

$$\boldsymbol{R}_{O\star}^W = \begin{bmatrix} \cos\theta_{O\star}^W & -\sin\theta_{O\star}^W \\ \sin\theta_{O\star}^W & \cos\theta_{O\star}^W \end{bmatrix} \tag{9}$$

The value $v = v(P, O\star)$ of the implicit equations in the point P has the following meaning: if $v < 0$, the point P is inside the geometric primitive $O\star$; if $v = 0$, the point P is in the boundaries of the geometric primitive $O\star$; and if $v > 0$, the point P is outside the geometric primitive $O\star$.

To consider the dimensions of the robot in the planned path, the dimensions of the geometric primitives are augmented following the equation $\hat{\boldsymbol{r}}_{O\star} = \boldsymbol{r}_{O\star} + \max(\boldsymbol{r}_R)$, being $\boldsymbol{r}_R$ the dimensions of the robot, and $\max(\boldsymbol{r}_R)$ a vectorial representation of the maximum dimension of the robot.

It is important to note that the use of the implicit equation, is a simplification that allows to have an intuition of the distance to the objects in a very fast way, but since its value differs from their actual distance, the shape of the potential field map will be different than if it would be calculated using the distance value.

D. Cost of moving between two points

Given a potential field map p (as the one mentioned in Section II-C), it generates a surface S given by $S = \left[\boldsymbol{t}_P^W,\ p\left(\boldsymbol{t}_P^W\right)\right]^T$, for all points P of the arena.

The cost of moving between points A and B following the curve $C_{A\to B}$, is given by the line integral for the unit scalar field along the curve $C_{A'\to B'}$, that is the projection of the curve $C_{A\to B}$ over the surface S.

$$c(A, B) = \int_{C_{A'\to B'}} 1 \cdot \|\mathrm{d}\boldsymbol{l}\| \approxeq \sum_{C_{A'\to B'}} \|\Delta\boldsymbol{l}\| \tag{10}$$

where points A and B are given in world coordinates as $\boldsymbol{t}_{P_A}^W = [x_A,\ y_A]^T$ and $\boldsymbol{t}_{P_B}^W = [x_B,\ y_B]^T$ respectively.

If the curve $C_{A\to B}$ is parametrized by $s \in [s_{min}, s_{max}]$, then, any point P of this curve is given by $\boldsymbol{t}_P^W(s)$, and for a $\Delta s = s_i - s_{i-1}$:

$$\Delta \boldsymbol{l} = \begin{bmatrix} \Delta \boldsymbol{t} \\ \Delta p \end{bmatrix} = \begin{bmatrix} \boldsymbol{t}_P^W(s_i) - \boldsymbol{t}_P^W(s_{i-1}) \\ p\left(\boldsymbol{t}_P^W(s_i)\right) - p\left(\boldsymbol{t}_P^W(s_{i-1})\right) \end{bmatrix} \tag{11}$$

and sampling the s interval, the cost of moving between points A and B, is calculated by combining equations 10 and 11, getting:

$$c(A,B) \cong \sum_{s_i=s_{min}}^{s_{max}} \sqrt{\|\Delta \boldsymbol{t}\|^2 + \|\Delta p\|^2} \tag{12}$$

In the case that the curve, $C_{A\to B}$ is a straight line connecting points A and B, the following parametrization can be used:

$$s \in [s_{min}, s_{max}] = \left[0,\ \left\|\boldsymbol{t}_{P_B}^W - \boldsymbol{t}_{P_A}^W\right\|\right] \tag{13}$$

being a generic point P of the world over the curve $C_{A\to B}$ given by:

$$\boldsymbol{t}_P^W(s) = \boldsymbol{t}_{P_A}^W + s \cdot \boldsymbol{n}_C \tag{14}$$

where

$$\boldsymbol{n}_C = \frac{\boldsymbol{t}_{P_B}^W - \boldsymbol{t}_{P_A}^W}{\left\|\boldsymbol{t}_{P_B}^W - \boldsymbol{t}_{P_A}^W\right\|} \tag{15}$$

and for a $\Delta s = s_i - s_{i-1}$:

$$\Delta \boldsymbol{t} = \Delta s \cdot \boldsymbol{n}_C \tag{16}$$

and therefore, sampling the s interval, the cost of moving between points A and B, is calculated by:

$$c(A,B) \cong \sum_{s_i=0}^{s_{max}} \sqrt{\Delta s^2 + \left(p\left(\boldsymbol{t}_P^W(s_i)\right) - p\left(\boldsymbol{t}_P^W(s_{i-1})\right)\right)^2} \tag{17}$$

where the number of sampling points of the s interval, n_{s_i}, is given by $n_{s_i} \geq \frac{s_{max}}{\min_{\forall O\star}(\hat{\boldsymbol{r}}_{O\star})}$.

E. A⋆ to find a collision-free path in the PRM graph

Our proposed approach uses a simple but efficient widely used A⋆ algorithm, [4], as a graph search algorithm, to find the collision-free path in the probabilistic roadmap (calculated in Section II-B), guided by the potential field map as the cost function (as explained in Section II-D).

This informed search algorithm searches among all possible paths to the solution for the one that incurs in the smallest cost, and among these paths it first considers the ones that appear to lead most quickly to the solution. At each iteration of its main loop, it needs to determine which of its partial paths to expand into one or more longer paths. It selects the path that minimizes:

$$f(n_i) = g(n_0, n_i) + h(n_i, n_f) \tag{18}$$

where n_i is the working node on the current iteration, $g(n_0, n_i)$ is the smallest cost of the path from node n_0 to node n_i, and $h(n_i, n_f)$ is an heuristic that estimates the cheapest path from node n_i to node n_f.

In our planner, the cost $g(n_0, n_i)$ is given by the sum of the cost needed to move between node n_0 and node n_i in the probabilistic roadmap (see Section II-D):

$$g(n_0, n_i) = \sum_{n_j=n_0}^{n_{j+1}=n_i} c(n_j, n_{j+1}) \tag{19}$$

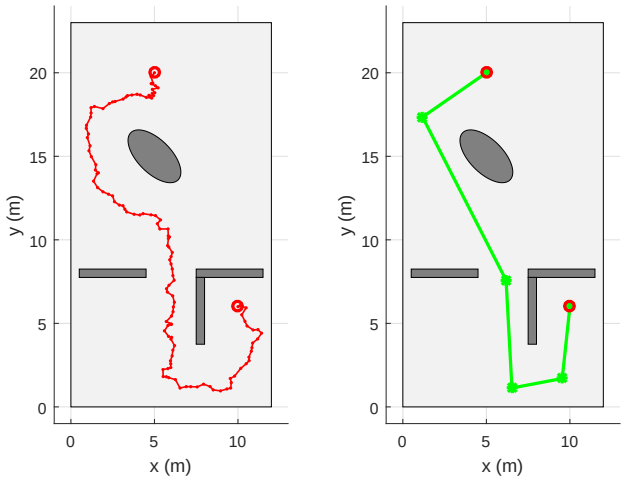
where the cost function $c(n_j, n_{j+1})$ corresponds to the total potential field map, incorporating the contribution of the query and the obstacles.

The heuristic cost $h(n_i, n_f)$ is given by the cost needed to move from node n_i to node n_f in a straight line in the environment (not in the graph):

$$h(n_i, n_f) = c(n_i, n_f) \tag{20}$$

where the cost function $c(n_i, n_f)$ corresponds to the potential field map with only the contribution of the query.

After the exploration of the graph, the path t_p (also called plan) has to be created by revisiting the explored path. An example of the collision-free path computed might be seen in Fig. 4a.



(a) Raw path generated with the A⋆ algorithm.

(b) Shortened path.

Fig. 4. Collision-free path from $t_{P_0} = [10,\ 6]^T$ to $t_{P_f} = [5,\ 20]^T$.

F. Path shortening

Once the collision-free path has been found, a post-processing is applied to shorten it. An example is shown in Fig. 4b.

In the presented algorithm, the path is shortened, t_s, by taking into account the obstacles of the environment. The initial node n_0 and the final node n_f are always the beginning and ending points of the shortened path. To obtain the middle points of the shortened path, Algorithm 1 is used.

G. Velocity and acceleration planning

In order to achieve a faster traversal time, the sharp corners of the trajectory are substituted, when possible, by circumference arcs and a velocity and acceleration plans are

Algorithm 1 Path shortening algorithm.

Initialization: $t_s = \{\}$, $n_i = n_0$, $n_j = n_{i+1}$
Loop:
while $n_j \neq n_f$ **do**
  if $c(n_i, n_j) \geq \sum_{n_k=n_i}^{n_{k+1}=n_j} c(n_k, n_{k+1})$ **then**
    $t_s = \{t_s, n_j\}$, $n_i \leftarrow n_j$, $n_j \leftarrow n_{j+1}$
  else
    $n_j \leftarrow n_{j+1}$
  end if
end while
Note: The cost function $c(n_i, n_j)$ corresponds to the potential field map, incorporating only the contribution of the obstacles.

calculated over the resulting path. Therefore, the trajectory executed by the controller is a sequence of straight segments and circumference arcs. The velocity and acceleration plans, inspired in [22], are calculated over the trajectory as described in our previous work in [23], [24].

In the circumference arcs calculation, the radius are limited to ensure that the trajectory does not deviate from the shortened path, further than a parametrized confidence distance. The velocity plan is calculated considering and fulfilling the following constraints: the equations of the uniformly accelerated motion and the parametrized maximum velocity, v_{max}, and acceleration constraints, a_{max}. The maximum velocity for each turn is calculated as $v_{turn} = \sqrt{a_{max} R_{turn}}$, where R_{turn} is the arc radius. When a turn is too sharp, it stays as a concatenation of two straight segments. Such turns are almost stall-turn maneuvers, and therefore, the velocity plan in such sharp turns is preconfigured with a parameter named $v_{stall\,turn}$.

The velocity and acceleration plans resemble a bang-bang type control on the commanded acceleration, constrained by the maximum velocity and the turn velocities, which are themselves constrained by the maximum acceleration.

H. Collision-free path check

The planned collision-free path t_s has to be checked whenever the environment changes (a new static object has been mapped, or a moving agent has moved) to determine if it is still collision-free.

If the cost of the planned collision-free path t_s is lower than a threshold, the path is considered to be collision-free, and it is calculated by:

$$c(n_0, n_f) = \sum_{n_j=n_0}^{n_{j+1}=n_f} c(n_j, n_{j+1}) \tag{21}$$

where the cost function $c(n_j, n_{j+1})$ corresponds to the potential field map, incorporating only the contribution of the obstacles.

III. EVALUATION

A. Methodology

The proposed path planner, incorporated as an open-source component in Aerostack, has been intensively used in multiple research projects, including two international competitions, IMAV 2013 [25], [26], and IARC 2014 [27]; in several experiments [28], [29], [30], [2], and multiple public demonstrations.

Its performance has been demonstrated when used in a system formed by a single aerial robot, and in a system with multiple moving aerial agents. The environments tested included both previously known and previously unknown objects, with different shapes, as a house with a door and windows, column areas, and narrow passages. The tested obstacles were both fixed and moving. In all these challenges, the planner was able to calculate collision-free paths in real-time, re-planning them fast enough when needed.

An intensive isolated evaluation of the performance of the presented path planner is done in Section III-B, by means of two different simulations with very challenging environments. These simulations has been run several times, showing that the random nature of the PRM leads to different results.

To complement the simulations, an experiment with a real flight is presented in Section III-C. In this experiment, the performance of the presented path planner is evaluated within Aerostack [1], [2].

B. Simulation in complex static environments

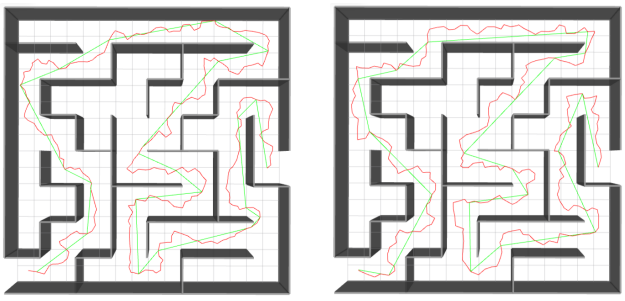
In this section, the performance of the presented path planner is evaluated by means of two simulations. The environments of these simulations, formed only by static objects, are very challenging and unrealistic compared with the environments that an aerial robot normally faces.

In these tests, no previous knowledge about the objects of the environment is incorporated to the planner. The reader must note that in these environments, the heuristic given by the potential field map is not contributing to speed up the search process and therefore the graph is needed to be visited almost completely by means of the A* algorithm to be able to find the solution. The aerial robot is configured as an ellipse of $0.5 \times 0.5\ m^2$.

In both tests, shown in Figs. 5 and 6, the raw collision-free path over the PRM is displayed in red, while the green line is the shortened collision-free planned path.

Fig. 5 shows the collision-free paths calculated for an environment that is a labyrinth of dimensions $16 \times 16\ m^2$, to move from point $t_{P_0} = [1,\ 1]^T$ (bottom left) to point $t_{P_f} = [15,\ 7]^T$ (middle right). Two different runs of the algorithm are shown. As the obstacles density is very high, the PRM requires a large number of nodes, set to 3000 in the presented case (approx. $11.72\ nodes/m^2$, same number of elements in the graph than an equivalent 2D cell map with a resolution of $29\ cm/cell$). The neighborhood of the nodes of the PRM has been set to 6 (higher than the 4-neighborhood of a 2D cell map).

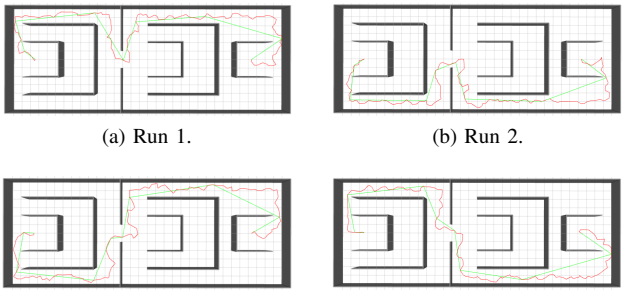
Fig. 6 shows the collision-free path calculated for an environment of dimensions $32 \times 12\ m^2$, that include five dead end areas (creating local minimum in the potential field map) and a narrow passage, to move from point $t_{P_0} = [3,\ 6]^T$ (left) to point $t_{P_f} = [28,\ 6]^T$ (right). Four different runs are



(a) Run 1. (b) Run 2.

Fig. 5. Collision-free path calculated for a labyrinth of dimensions $16 \times 16\ m^2$, to move from point $t_{P_0} = [1,\ 1]^T$ (bottom left) to point $t_{P_f} = [15,\ 7]^T$ (middle right).

shown. The configured number of nodes of the PRM is set to 3000 (approx. 7.81 $nodes/m^2$, same number of elements in the graph than an equivalent 2D cell map with a resolution of 36 $cm/cell$), with a 6-neighborhood.



(a) Run 1. (b) Run 2.

(c) Run 3. (d) Run 4.

Fig. 6. Collision-free paths calculated for an environment of dimensions $32 \times 12m^2$, to move from point $t_{P_0} = [3,\ 6]^T$ (left) to point $t_{P_f} = [28,\ 6]^T$ (right).

Despite the high complexity of the simulated environments, the path planner is able to find a solution without falling in any local minima, validating its robust behavior. Moreover, as compared, it is more efficient than the 2D cell map based equivalent planner, as it uses a smaller number of nodes for a particular resolution.

C. Real experiments with a multi-robot system

This section presents a real experiment with multiple (three) aerial agents performing a fully autonomous indoors mission. An emulated search and rescue mission, similar to the one presented in [2] is executed. In this mission, an emergency situation is emulated inside a house (right side of Fig. 7). Two aerial robots (labeled as red in Fig. 7) take-off from the emergency team base (left side of Fig. 7) and enter the building searching for a subject (labeled as green in Fig. 7). Once detected, the target is found, the two search robots return to the rescue team base while another aerial robot (labeled as blue in Fig. 7) perform a rescue task (i.e. a visual servoing interaction [31]) to the subject. After the rescue, the aerial robot returns to the rescue team base.

Three Parrot AR.Drone 2.0 are used as aerial platforms, individually connected to three average laptops using a Wi-Fi interface. The three laptops are connected to a LAN

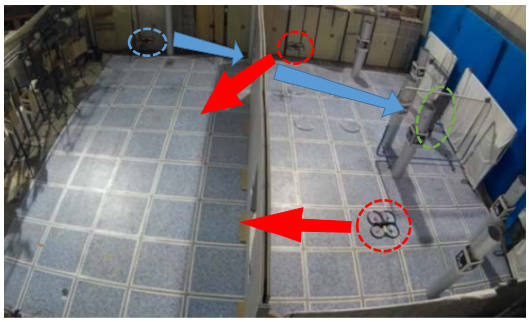


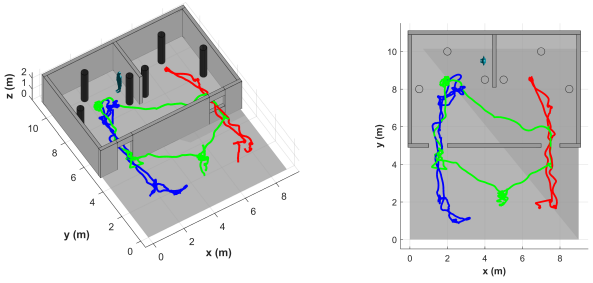
Fig. 7. A frame of the search and rescue mission, after the detection of the target.

by means of a switch. Every laptop is running all the components of Aerostack required for every robotic agent to perform the fully autonomous mission, including a separate instance of the presented path planner per aerial robot.

The dimensions of the arena are $9 \times 11\ m^2$. The configured number of nodes used in the PRM is inflated to 3000 (approx. 30.30 $nodes/m^2$, same number of elements in the graph than an equivalent 2D cell map with a resolution of 18 $cm/cell$) The neighborhood of the nodes of the PRM has been set to 5 (higher than the 4-neighborhood of a 2D cell map). The aerial robot is configured as an ellipse of $0.7 \times 0.7\ m^2$.

Whenever the proposed planner is unable to find a collision-free path, an empty path is commanded to the path following controller, meaning that it has to hover in the current position until a new collision-free path is found.

Fig. 8 shows the trajectories followed by the aerial robots. The two search aerial robots, hereinafter designated as *drone1* and *drone2*, are labeled with the red and green trajectories respectively. The rescue robot, designated as *drone3*, is labeled with the blue trajectory.

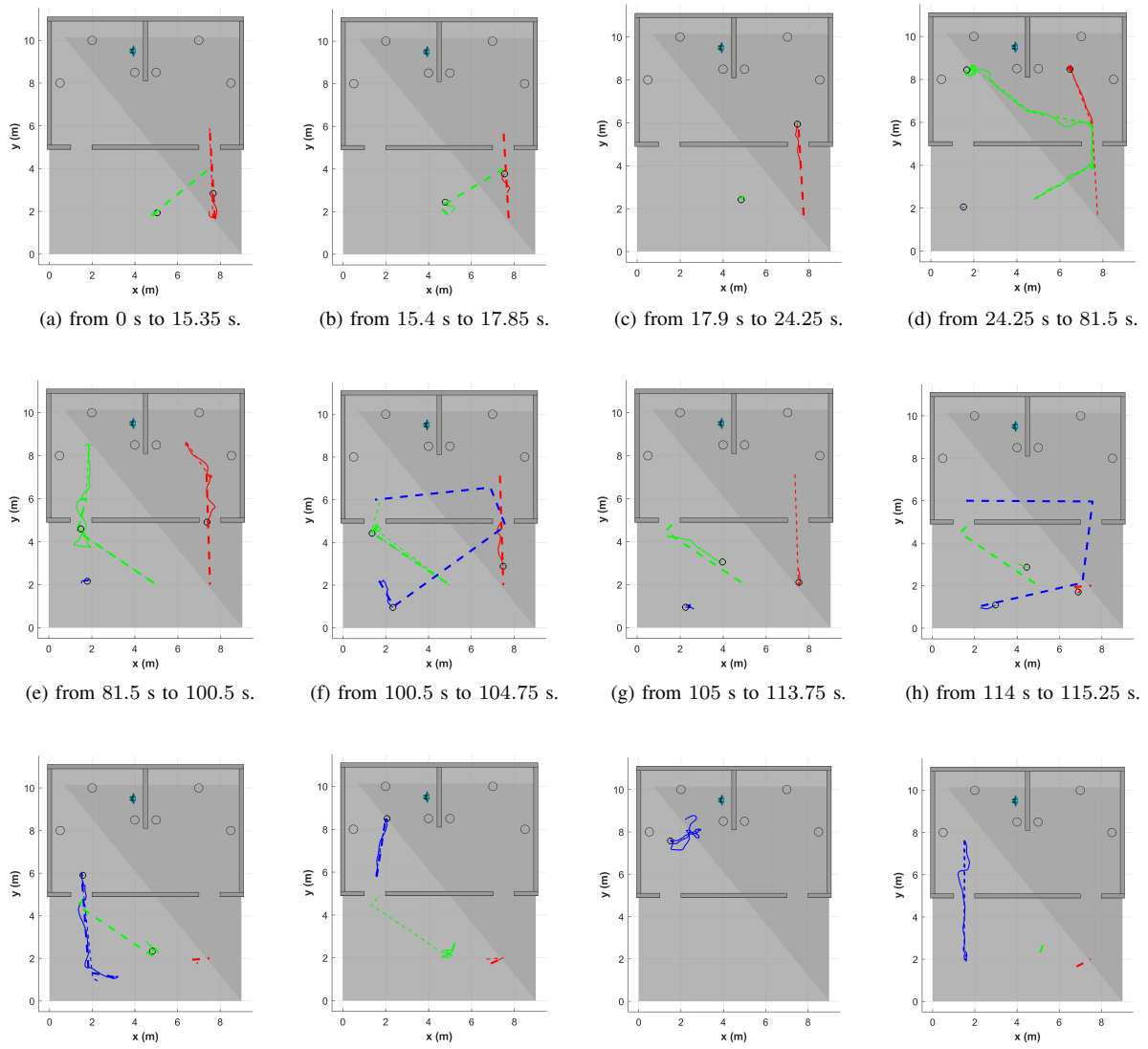


(a) 3D view (b) 2D view

Fig. 8. Trajectories followed by the aerial robots.

Fig. 9 shows the trajectories followed by the aerial robots (solid lines), together with their planned path (dashed lines), in different instant times. As can be extracted, the proposed path planner is able to generate collision-free paths fast enough to ensure an adequate deliberative behavior.

In Fig. 9a, the mission planner of *drone1* commands it to cross the window (point $[7.5,\ 6]^T$) after take-off, *drone2* is commanded to face the window (point $[7.5,\ 4]^T$) before crossing it, to guarantee that it enters the house through the



(a) from 0 s to 15.35 s. (b) from 15.4 s to 17.85 s. (c) from 17.9 s to 24.25 s. (d) from 24.25 s to 81.5 s.

(e) from 81.5 s to 100.5 s. (f) from 100.5 s to 104.75 s. (g) from 105 s to 113.75 s. (h) from 114 s to 115.25 s.

(i) from 115.5 s to 129.75 s. (j) from 130 s to 149 s. (k) from 149.5 s to 186.25 s. (l) from 186.5 s to 205.35 s.

Fig. 9. Trajectories followed by the aerial robots in several time frames. Solid lines represent the followed paths, while dashed lines are the planned paths.

window, generating a conflict between the agents on purpose to show the performance of the proposed planner.

In Fig. 9b can be seen that *drone2* needed to plan a new path after *drone1* moved forward to avoid it. Nevertheless, as *drone1* moves forward, *drone2* is unable to plan a collision-free path to reach the commanded point, and it waits until *drone1* has cleared the commanded point (as seen in Fig. 9c).

After crossing the window, *drone1* is commanded to explore the left room (point $[6.5,\ 8.5]^T$). Once the path is clear, *drone2* is able to plan a collision-free path to face the window, and after that, it is commanded to explore the right room (point $[2,\ 8.5]^T$), as represented in Fig. 9d.

Once the target is detected by *drone2*, both exploration robots are commanded to return to their take off positions (*drone1* to point $[7.5,\ 2]^T$; and *drone2* to point $[5,\ 2]^T$). In the meantime, drone3 takes off and it is commanded to cross the doorway (point $[1.75,\ 6]^T$). Nevertheless, as it can be seen in Fig. 9e, while the doorway is occupied, *drone3* is unable to find a collision-free path.

In Figs. 9f to 9i an interesting situation where *drone3* is trying to find a collision-free path with the disturbances of *drone1* and *drone2* is shown. In Fig. 9f, as the doorway is occupied by *drone2*, *drone3* plans a collision-free path through the window, since *drone1* has already cleared it. As *drone2* moves forward, the previously planned path by *drone3* is not collision-free anymore, and as *drone2* is still very close to the doorway, *drone3* is unable to find a collision-free path, as represented in Fig. 9g. As soon as *drone1* lands, it is not considered as moving agent anymore, and *drone3* is able to find again a collision-free path through the window (Fig. 9h). Finally, in Fig. 9i, as *drone2* is moving forward, *drone3* is able to find a shorter collision-free path crossing the doorway.

In Fig. 9j *drone3* is commanded to face the target mov-

ing to point $[2,\ 8.5]^T$. Once it reaches this point, *drone3* performs the visual servoing task (Fig. 9k). Whenever the subject is labeled as rescued, *drone3* is commanded to return to its take off position (point $[1.5,\ 2]^T$), as shown in Fig. 9l.

This experiment demonstrates that the proposed path planner is able to operate in real-time, generating collision-free paths robustly without falling in any local minima in a complex environment formed by both static obstacles and moving agents. As explained, the proposed trajectory planner was the only component responsible for the collision avoidance.

IV. CONSIDERATIONS AND FURTHER IMPROVEMENTS

This section explores some of the limitations of the presented path planner and points out how to overcome them.

A. Generalization to other environment descriptions

The proposed position path planner imposes a description of the environment based in two kinds of geometric features, rectangles and ellipses. This requires an adequate Situation Awareness System that transforms the available environment description into the geometric primitive based one.

On the one hand, more kinds of geometric features like triangles should be added for a richer and more accurate description of the environment.

On the other hand, other environment descriptions such as quadtrees or cell maps might be directly used by considering each cell as a rectangular obstacle. Nevertheless, its direct usage is highly discouraged as the efficiency of the presented algorithm would be drastically reduced.

B. Generalization to 3D environments

The generalization to 3D environments is an immediate future work. For a 3D environment, the geometric primitives that might be used for environment description are prisms, ellipsoids and cylinders, among others. Therefore, new implicit equations v would need to be defined for such geometric primitives. The rest of the algorithm presented in Section II would remain equivalent.

C. Dynamic arena sampling

An important limitation of the proposed path planner is the need of sampling the environment with a probabilistic roadmap, and therefore, selecting the number of nodes used, their neighborhood, and their sampling probability density function. As discussed before, a large number of nodes lead to inefficient searches, but a small number of nodes yield an inaccurate coverage of the free environment, what is specially critical if the environment is populated with a large number of obstacles or the obstacles create small corridors.

To overcome this limitation an improvement over the PRM might be done, such as a Lazy-PRM, [17]; or a C-PRM, [18]. In this approach, whenever a plan is not found, more nodes might be dynamically added to the probabilistic roadmap to have a better coverage of the environment. A possible criteria to add these nodes might use the potential field map, adding random nodes with higher probability where the potential field map has higher values (excluding the value ∞ of equation 5, so excluding the obstacles), that is, near the obstacles and their surroundings.

D. Efficient management of moving agents

The time that the presented path planner requires to create a collision-free path mainly depends on the number of nodes of the probabilistic roadmap, the number of obstacles, the position and shape of the obstacles, and the distance between the initial point and the final point.

As explained in Section II-H, the planned collision-free path is checked whenever the environment changes. Whenever this planned path is not collision-free anymore, a new collision-free path calculation needs to be done. This would happen often in an environment with a large number of moving agents or if the moving agents are moving in the neighborhood of the robot. Every time a planning request is executed, the A* algorithm is executed from the beginning. A more efficient approach would reuse the information of a previous query, updating only the information that differs from the previous query, speeding up the search. This reuse of information might be done using a better graph search algorithm like a D* algorithm, [6].

V. CONCLUSIONS

In this paper, we have presented a robust real-time collision-free path planner used for the horizontal 2D navigation of multirotor aerial robots in dynamic environments.

The objects of the environment are represented as geometric primitives, simplifying the information needed to describe the environment and therefore reducing the computational cost to determine the distance to the obstacles.

The arena is sampled using a probabilistic roadmap approach, generated at launching time and therefore without adding the obstacles to it, unlike the state of the art PRMs. This has two advantages: on the one hand, the space of the arena is covered more compactly than if using other approaches such as a grid map, but without loss of resolution. On the other hand, it allows to handle moving obstacles efficiently, since they are not included in the probabilistic roadmap graph.

A discrete search algorithm, concretely an A* algorithm is employed to calculate the collision-free path on the probabilistic roadmap graph. To speedup this search, and to be able to avoid the obstacles, a potential field map is used as the cost function to guide the search, that never falls in a local minima.

Finally, a velocity and acceleration planning along the collision-free planned path is carried out, permitting the fast collision-free path following.

The performance of the proposed path planner has been widely demonstrated thanks to its previous intensive usage in multiple research projects with real flights, including two international aerial robotics competitions, several different experiments, and various public demonstrations. In this paper, its performance has been evaluated with two simulations

with complex environments including a labyrinth and dead ends, and with a real flight experiment where three fully autonomous aerial robots were executing an emulated search and rescue mission.

An analysis of the main limitations of the proposed path planner, together with our proposed future work to overcome them has been carried out.

The proposed path planner is completely available to the scientific community as an open-source software integrated in Aerostack[4].

ACKNOWLEDGMENT

The authors thank Ramon A. Suarez Fernandez, Hriday Bavle, and Carlos Sampedro their valuable help with the real experiments.

REFERENCES

[1] J. L. Sanchez-Lopez, R. A. S. Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy, "Aerostack: An architecture and open-source software framework for aerial robotics," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 332–341.

[2] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. Suárez Fernández, and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *Journal of Intelligent & Robotic Systems*, pp. 1–27, 2017.

[3] X. Yu and Y. Zhang, "Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects," *Progress in Aerospace Sciences*, vol. 74, pp. 152 – 166, 2015.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[5] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of a*," *J. ACM*, vol. 32, no. 3, pp. 505–536, July 1985.

[6] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 3310–3317 vol.4.

[7] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23–32, Feb 1992.

[8] M. G. Park, J. H. Jeon, and M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570)*, vol. 3, 2001, pp. 1530–1535 vol.3.

[9] X. Xu, J. Xie, and K. Xie, "Path planning and obstacle-avoidance for soccer robot based on artificial potential field and genetic algorithm," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 1, 2006, pp. 3494–3498.

[10] Y. bo Chen, G. chen Luo, Y. song Mei, J. qiao Yu, and X. long Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.

[11] K. Yang and S. Sukkarieh, "Model predictive unified planning and control of rotary-wing unmanned aerial vehicle," in *2012 12th International Conference on Control, Automation and Systems*, Oct 2012, pp. 1974–1979.

[12] H.-P. Huang and S.-Y. Chung, "Dynamic visibility graph for path planning," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2813–2818 vol.3.

[13] M. d. S. Arantes, J. d. S. Arantes, C. F. M. Toledo, and B. C. Williams, "A hybrid multi-population genetic algorithm for uav path planning," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16. New York, NY, USA: ACM, 2016, pp. 853–860.

[14] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[15] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996.

[16] R. Geraerts and M. H. Overmars, *A Comparative Study of Probabilistic Roadmap Planners*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 43–57.

[17] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, April 2000, pp. 521–528 vol.1.

[18] G. Song, S. Miller, and N. M. Amato, "Customizing prm roadmaps at query time," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 2, 2001, pp. 1500–1505 vol.2.

[19] Y. Zhang, D. wei Gong, and J. hua Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172 – 185, 2013.

[20] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, Feb 2013.

[21] A. Hidalgo-Paniagua, M. A. Vega-Rodrguez, and J. Ferruz, "Applying the {MOVNS} (multi-objective variable neighborhood search) algorithm to solve the path planning problem in mobile robotics," *Expert Systems with Applications*, vol. 58, pp. 20 – 35, 2016.

[22] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, pp. 1–14.

[23] J. Pestana, I. Mellado-Bataller, C. Fu, J. L. Sanchez-Lopez, I. F. Mondragon, and P. Campoy, "A general purpose configurable navigation controller for micro aerial multirotor vehicles," in *ICUAS 2013*, 2013.

[24] J. Pestana, I. Mellado-Bataller, J. L. Sanchez-Lopez, C. Fu, I. F. Mondragón, and P. Campoy, "A general purpose configurable controller for indoors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 387–400, 2014.

[25] J. Pestana, J. L. Sanchez-Lopez, P. de la Puente, A. Carrio, and P. Campoy, "A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, May 2014, pp. 617–622.

[26] J. Pestana, J. L. Sanchez-Lopez, P. de la Puente, A. Carrio, and p. Campoy, "A vision-based quadrotor multi-robot solution for the indoor autonomy challenge of the 2013 international micro air vehicle competition," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 601–620, 2016.

[27] J. L. Sanchez-Lopez, J. Pestana, J.-F. Collumeau, R. Suarez-Fernandez, P. Campoy, and M. Molina, "A vision based aerial robot solution for the mission 7 of the international aerial robotics competition," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, June 2015, pp. 1391–1400.

[28] J. L. Sanchez-Lopez, J. Pestana, P. de la Puente, R. Suárez Fernandez, and P. Campoy, "A system for the design and development of vision-based multi-robot quadrotor swarms," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, May 2014, pp. 640–648.

[29] J. L. Sanchez-Lopez, J. Pestana, P. de la Puente, and P. Campoy, "A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 779–797, 2016.

[30] C. Sampedro, H. Bavle, J. L. Sanchez-Lopez, R. A. S. Fernández, A. Rodríguez-Ramos, M. Molina, and P. Campoy, "A flexible and dynamic mission planning architecture for uav swarm coordination," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 355–363.

[31] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy, "Computer vision based general object following for gps-denied multirotor unmanned vehicles," in *American Control Conference (ACC), 2014*, June 2014, pp. 1886–1891.

[4] www.aerostack.org