Quadrotor-UAV optimal coverage path planning in cluttered environment with a limited onboard energy

Yasser Bouzid, Yasmina Bestaoui, Houria Siguerdidjane

Quadrotor-UAV Optimal Coverage Path planning in cluttered environment with a limited onboard energy

Y. Bouzid*, Y. Bestaoui, H. Siguerdidjane

***Abstract*— In this paper, a quadrotor optimal coverage planning approach in damaged area is considered. The quadrotor is assumed to visit a set of reachable points following the shortest path while avoiding the no-fly zones. The problem is solved by using a two-scale proposed algorithm. In the first scale, an efficient tool for cluttered environments based on optimal Rapidly-exploring Random Trees (RRT) approach, called multi-RRT* Fixed Node (RRT*FN), is developed to define the shortest paths from each point to their neighbors. Using the pair-wise costs between points provided by the first-scale algorithm, in the second scale, the overall shortest path is obtained by solving the Traveling Salesman Problem (TSP) using Genetic Algorithms (GA). Taking into consideration the limited onboard energy, a second alternative based on the well-known Vehicle Routing Problem (VRP) is used. This latter is solved using the savings heuristic approach. The effectiveness of this proposed two-scale algorithm is demonstrated through numerical simulations and promising results are obtained.**

I. INTRODUCTION

Nowadays, autonomous robots have been efficiently employed in military as well as in civil missions and especially in cases representing a real risk for humans as for instance: fire emergencies, natural disasters, Nuclear-Biological-Chemical (NPC) contaminations, etc. In such events, localization, search and rescue missions in damaged areas are usually lengthy involving large staff and sophisticated equipment. Therefore, the implication of robots brings a notable benefit where the challenge then is to ensure an efficient, optimal and scalable coverage planning in a highly constrained space.

A broad range of motion planning strategies has been extensively studied. Most of them focus on the optimality guarantees and the obstacles avoidance capabilities. Sampling-based search methods such as: Expansive Space Trees (EST) [1], Probabilistic Roadmap Methods (PRM) [2] and Rapidly exploring Random Trees (RRT) have been used to find collision-free trajectories in cluttered environments [3-5].

RRT based algorithms are efficient to single-query motion planning problems in constrained workspace. However, a feasible solution is obtained only and no optimal solution is ensured. Therefore, an alternative optimal RRT (RRT*) method is proposed in order to tackle this problem by introducing incremental rewiring of the graph [6]. Nonetheless, this is inconsistent with their nature, single-query, and so becomes very expensive especially in high dimensions. In addition, for multi-target path planning problem, this technique requires a huge memory to store the complete nodes. Recently, an extended promising algorithm is proposed that retains the same probabilistic guarantees on optimality and completeness as the RRT* while reducing the required memory for storing nodes in the tree. This technique is called Fixed Nodes RRT* (RRT*FN) [7].

Most of the existing path planning strategies steer to find a feasible optimal path from a starting point to one target point. However, some robotic applications such as coverage or security patrolling require other path planning techniques, which are capable to find a set of optimal paths from one starting point to many target points. In this paper, we contribute by introducing and studying an optimal coverage planning strategy in constrained planar map using a quadrotor UAV. This map is considered as a set of Points Of Interests (POIs) cluttered by obstacles of different natures and shapes as for instance: radar detection areas, forbidden areas, hazardous zones, physical obstacles, etc. These points are automatically generated or manually specified. On each POI, the quadrotor covers a region called Region Of Interest (ROI). Thus, the challenge is to follow the shortest path that links all the POIs while avoiding the obstacles. Our strategy is composed of two main scales:

1- The multi-directional RRT*FN based algorithm is developed to compute the inter-costs from each POI to its neighbors following the optimal paths with obstacles avoidance.
2- The traveling Salesman Problem (TSP) is resolved via genetic algorithms (GA) to compute the overall shortest route that allows the quadrotor to visit all the POIs once and then return to the starting point. Besides, due to the limited onboard energy, the Vehicle Routing Problem (VRP) is considered. This latter is solved using a modified savings heuristic approach.

The efficiency of the approach is demonstrated through extensive numerical simulations considering many scenarios.

The remaining of the paper is organized as follows: problem statement is presented in Section II. Then, the algorithms are introduced in Section III. The POIs generation and the map decomposition are highlighted in Section IV. The application of the proposed algorithm as well as simulation results are shown in Section V. Finally, some concluding remarks are made in the last section.

II. PROBLEM STATEMENT

A- Preliminary

The paper addresses a coverage path planning algorithm using a quadrotor that autonomously navigates in a 2D cluttered area. The quadrotor visits the overall specified and reachable points and moves then to the starting point

Y. Bouzid, Y. Bestaoui are with IBISC, Universitéd'Evry, Université Paris-Saclay, Evry, France; H. Siguerdidjane is with L2S, CentraleSupélec,Université Paris-Saclay, Gif sur yvette, France (Emails:Yasser.Bouzid@ufrst.univ-evry.fr)

following the shortest path while avoiding obstacles. Due to the limited onboard energy, the quadrotor may perform several rounds in order to complete the mission. The proposed algorithm can be used even for large dimension and complex shaped areas. Involving a quadrotor equipped with sensor for coverage mission is certainly a promising application. This flying sensor can play the same role as a set of static sensors providing fast and adaptable low cost solution. Moreover, unlike mobile robots, quadrotors have less operating constraints and their dynamic response makes them more efficient in such applications.

We firstly give an overview about the environment considered in this study. The considered map is composed of a set of non-connected no-fly zones of different shapes and sizes and a free connected space. The map also contains a set of non-connected target areas that are defined by the user and need to be covered by visiting a set of scattered points namely the Points Of Interest (POIs). At each point, the quadrotor covers a region, called Region Of Interest (ROI), whose size and shape are defined according to the range of the sensor and the altitude of the quadrotor. The union of ROIs may cover the target zones totally or partially according to the nature of the mission, the nature and the shape of the target zone, the range and the shape of the sensor's range and the user's specifications.

Remark 1 : Assume that the overall mechanical structure of the quadrotor is enclosed by a fictive circle of radius r. Therefore, if we enlarge all the obstacles with radius r, then the quadrotor can be considered as compact flying point (see Fig. 1).

Remark 2 : In fact, avoiding obstacles using the altitude requests more thrust, then more energy is consumed compared to the longitudinal and lateral motions, so the 3D navigation problem is reduced to 2D one, in order to reduce the consumed energy.

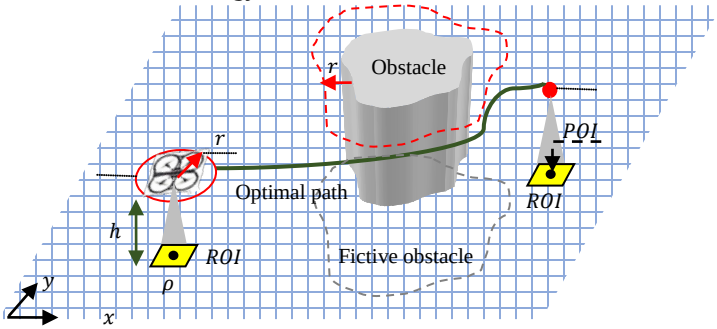


Fig. 1. Optimal path planning problem with obstacles avoidance in 2D map using quadrotor.

B- Mathematical formulation

Let $\mathfrak{D} \subseteq \mathbb{R}^2$ be the overall state space of the path-planning problem. Let $\mathfrak{D}_{free}$ be the resulting set of permissible states considered as connected space domain. Therefore, $\mathfrak{D}_{obs} \subseteq \mathfrak{D} \backslash \mathfrak{D}_{free}$ represents the states in collision with obstacles or no-fly zones. $\mathfrak{D}_{obs} = \bigcup_{i=1}^{l} \partial\Delta_i$ may be non-connected space domain where l denotes the number of sub-connected obstacles or non-fly zones (see Fig. 2). $\partial\Delta_i$ are a priori enlarged (see Remark 1). Let $\mathfrak{D}_{Target} = \bigcup_{i=1}^{m} \mathfrak{D}_{T_i} \subset \mathfrak{D}$ a set of m non-connected target zones that may have complex geometric structures.

Let $p_0 \in \mathfrak{D}_{free}$ be the initial location (base-station) and $P = \{p_i \in \mathfrak{D}_{free} | i = 1, \dots, N-1\}$ be the unordered disjoint locations of the POIs. Each point belongs to a partition of permissible states ROI (see Fig. 2). In other words,

$$W = \{w_i \subseteq \mathfrak{D}_{free} \,|p_i \in w_i,\ i = 1, \dots, N-1\} \quad (1)$$

represents the set of ROIs. Usually p_i represents the geometric center of $w_i\ i = 1, \dots, N-1$. These POIs are distributed on the free space according to the specified mission using an automatic generation algorithm (see Section IV) or defined by the user. The ROIs may have different shapes and sizes according to the range of the onboard sensor such as square shape (of dimension ρ), rectangular (of width ρ and length L) or circular shape of radius ρ. These sizes may change in function of the altitude h of the quadrotor. Notice, in the ideal case, for a complete coverage problem that $W = \mathfrak{D}_{Target} / \mathfrak{D}_{obs}$.

Let $\sigma_{ij}|_{\substack{i=0,\dots,N-1 \\ j=0,\dots,N-1 \\ i \neq j}} : [0,1] \mapsto \mathfrak{D}_{free}$ be a feasible path and $\mathcal{P}_{ij}|_{\substack{i=0,\dots,N-1 \\ j=0,\dots,N-1 \\ i \neq j}} \subseteq \mathfrak{D}_{free}$ be the set of all nontrivial feasible paths that join a point p_i by a point p_j.

The optimal path planning problem is then arranged in two scales:

- Scale 1

The first scale is the optimality inter-points. In other words, we seek to find an optimal path from point p_i to point p_j, $i = 0, \dots, N-1$ and $j = 0, \dots, N-1$ while avoiding obstacles. This optimal planning problem is then defined as the search for the paths, σ_{ij}^*, that minimizes a given cost function, $\mathcal{J}_{ij}: \mathcal{P}_{ij} \mapsto \mathbb{R}_{\geq 0}$, while connecting p_i to p_j through the free space.

$$\sigma_{ij}^* = \arg\min\{\mathcal{J}_{ij}(\sigma_{ij}) | \sigma_{ij}(0) = p_i, \sigma(1) = p_j, \forall s \in [0,1], \sigma_{ij}(s) \in \mathfrak{D}_{free}\} \quad (2)$$

$i = 0, \dots, N-1, j = 0, \dots, N-1$ and $i \neq j$.

where $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers.

Let $c_{ij}|_{\substack{i=0,\dots,N-1 \\ j=0,\dots,N-1 \\ i \neq j}}$ be the cost of an optimal path σ_{ij}^* from p_i to p_j where $c_{ij} = c_{ji}$ and $\sigma_{ij}^* \equiv \sigma_{ji}^*$.

The solution of this problem results in a matrix called inter-cost matrix $\mathcal{C} \in \mathbb{R}^{N \times N}$. Notice that this matrix is symmetrical while the diagonal elements are null ($c_{ii} = 0, i = 0, \dots, N-1$).

The solution of such a problem requires a multi-directional based path planning technique (see Fig. 2). Therefore, we propose a multi-directional RRT*FN based algorithm. This latter returns a global tree $\mathcal{T} = (V, E)$ consisting of $N-1$ sub-trees. From these trees, we distinguish $\sum_{i=1}^{i=n}(N-i)$ shortest path σ_{ij}^* with minimal costs c_{ij}. The cost quantities of the optimal paths can be interpreted as a travel distance, a travel time or the number of nodes. The fact that the quadrotor is considered, in our work, flying in a holonomic

way, the cumulative Euclidean distance between nodes is suitable as metric of optimality.

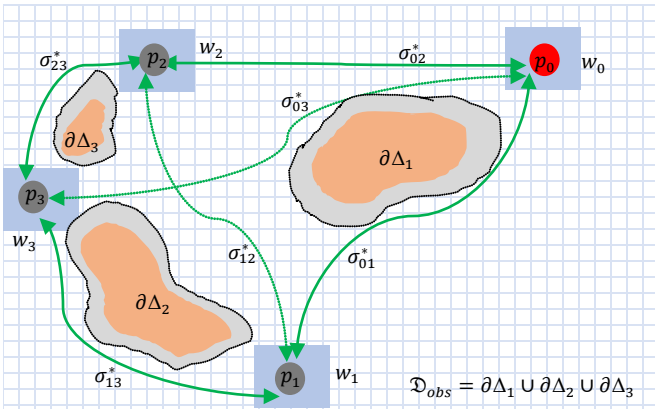


Fig. 2. Example of path planning: 2D map with three POIs $P = \{p_1, p_2, p_3\}$, three ROIs $W = \{w_1, w_2, w_3\}$, three non-connected obstacles $\mathfrak{D}_{obs} = \cup_{i=1}^{i=3} \partial\Delta_i$ and multi-directional optimal paths. Bold green line represents the shortest path.

- Scale 2

Herein, we seek to determine the minimum distance circuit starting from p_0, passing through all points $p_i|_{i=1,\dots,N-1}$ once and only once and then return back to the starting point. The objective is to find a permutation $\wp$ of the sequences: $s_0, s_1, \dots, s_{N-1}$, that minimizes the sum of distances,

$$D = \sum_{i=0}^{N-2} d(s_i, s_{i+1}) + d(s_{N-1}, s_0) \quad (3)$$

where the inter-distances $d(s_i, s_j)$ are defined from the first scale problem. Notice that the sequences s_i are chosen from the set of POIs P. This problem is known as Traveling Salesman Problem (TSP), which is a well known NP-hard optimization problem. We propose to use Genetic Algorithms (GA) in order to solve this problem and to get an optimal path according to the considered cost function.

It is impossible to achieve long missions in one round because of the limited onboard energy. Therefore, multiple rounds are suggested. Herein, we seek to determine the minimal set of circuits starting from p_0, passing through all points $p_i|_{i=1,\dots,N-1}$ once and only once and then turn back to the starting point taking in consideration the total energy without redundancy of point's passage. The main objective is to take in consideration the fact that the quadrotor must recharge or change batteries several times during the coverage mission, which is a more realistic scenario.

Let's assume that the aerial vehicle flies at moderate speed without aggressive maneuvers neither strong disturbances and at a constant altitude. This allows us to consider constant energy consumption. Then, the energy consumption can be measured in terms of time. Let us consider that the quadrotor has autonomy of C minutes called capacity. The flight time needed to cover one point is denoted by $q_i \forall i = 1, 2, \dots, n-1$. Notice that $C > max\ q_i$. The goal is to find the minimum set of routes that visit all POIs while minimizing the traveling distance (shortest path) and respecting the constraint of limited energy. The new constraint is stated as follows

$$C \geq T_R = \sum_{k=1}^{N_R-2} q_k + \sum_{k=0}^{N_R-2} t(p_k^R, p_{k+1}^R) + t(p_{N_R-1}^R, p_0^R) \quad (4)$$

where T_R is one route required time, q_k is the time required to cover one point p_k^R in the route R that contains N_R point. $t(p_k^R, p_{k+1}^R)$ is the duration to flight between two successive points in the route where $t(p_{N_R-1}^R, p_0^R)$ denotes the required time to turn back to the base-station. These points p_k^R are selected arbitrary from the set of points P.

This problem can be considered as a Capacitated-Vehicle Routing Problem (C-VRP). To solve this problem, we have adapted the Capacitated-VRP savings method to include the maximal traveling time constraint T_R.

III. ALGORITHMS DESCRIPTION

As stated in the previous section, our approach is composed of two separate scales. The first scale is used to obtain the pair-wise costs that allow the second scale to find the overall optimal path.

A. First scale RRT based algorithm

For this first scale, we propose to use the RRT based approach. This latter performs efficient searches even in high dimensional spaces. For the sake of clarity, a pseudo-code of basic RRT based path planning is summarized in Algorithm 1. The algorithm grows a tree of feasible trajectories from the initial state p_0 and returns a collision-free trajectory that reaches the goal $p_{goal} \in w_{goal}$. This basic algorithm attempts to find a first feasible solution as quickly as possible. Explicitly, from the initial state p_0, at each iteration, a random sample p_{rand} that belongs to the free workspace $\mathfrak{D}_{free}$ is chosen (line 3). The main routines used are:

- **Nearest** function (line 4): finds the nearest node $p_{nearest}$ in the tree to p_{rand}.
- **Steer** function (line 5): finds, in the free domain, the single reachable state p_{new} located at a random distance D_r from $p_{nearest}$, which is closest to p_{rand}.

If the path from the $p_{nearest}$ to p_{new} is collision free (line 6), p_{new} is added to the tree τ as a child of $p_{nearest}$ (line 7). The algorithm holds until iteration K_{loop} or when a first path is found (line 9).

Algorithm 1: Basic path planner

Function: τ =**RRT**$(K_{loop}, p_0, p_{goal})$
1: $\tau \leftarrow$**InitializeTree**();
2: **repeat**
3: $p_{rand} \leftarrow$**RandomState**$(\mathfrak{D}_{free})$
4: $p_{nearst} \leftarrow$**Nearest**(τ, p_{rand})
5: $p_{new} \leftarrow$**Steer**$(p_{nearest}, p_{rand})$
6: **If ObstacleFree**$(p_{nearest}, p_{new})$**, then**
7: $\tau \leftarrow$**InsertNode**$(p_{nearst}, p_{new}, \tau)$
8: **end if**
9: **until** $(i++ > K_{loop} || p_{new} \in w_{goal})$
10: **Return** τ

Using RRT, no optimal solution is ensured as no metric is used to measure the optimality of the trajectories. Therefore, RRT* algorithm is introduced in [6] with theoretical guarantees of optimality. RRT* converges to the optimal solution as the number of samples reaches infinity. This is by incrementally rewiring the tree as lower cost trajectories

become available with the addition of new nodes to the tree. RRT* uses additional functionality compared to RRT:

- *Cost function*: provides the total cost from the initial state to a node in the tree.
- *Rewiring procedure*: rewires the local neighborhood of the newly added node such that the cost to the initial state decreases.

Instead of considering the nearest node as the parent of p_{new}, we take a set of nodes P_{near} in the local neighborhood, of radius D_n of p_{new}. Then **ChooseParent** routine selects the best parent for p_{new}. p_{new} is connected to the parent, which minimizes the total cost from the initial state as first instance where RRT* optimizes the complete trajectory. As a second instant of optimization, the local neighborhood of p_{new} is optimized with the **ReWire** routine. If the assignment of p_{new} as the parent node of the nodes in P_{near} decreases the total cost from p_0 to $p_{near} \in P_{near}$, then the **ReConnect** function removes the edge between p_{near} and it's parent node and creates an edge between p_{near} and p_{new}, the new parent node.

Because there is no prior limits of the number of nodes in tree generated by the RRT* algorithm, the multi-directional RRT* consumes many computational capacities where supplementary memory is required to store the added nodes. The number of nodes can be fixed to alleviate this problem. Therefore, an extension of RRT*, called RRT* Fixed Nodes (RRT*FN), that employs a node removal procedure from the skeleton of the RRT*is used. The philosophy of the strategy comes from the fact that: If a node does not have children, it also means that it is not on a path reaching the goal state. This approach is summarized by Algorithm 2 with the main rules:

- The tree is grown as the RRT* until a maximum number of nodes M is reached where the algorithm must be restarted if a feasible path to the goal region is not reached.
- A new node is added, if and only if there exists, at least, one node in the tree with one or no child (this node is deleted from the tree) and the cost to the present state from the initial state has decreased.
- In the case of multiple nodes present without children, one of them is selected randomly.

The main functionality of this algorithm besides those ensured by RRT* is the removal procedure. We distinguish ordinary and forced removal functions that are:

- **ReWire** function (line 14): If a node in P_{near} (line 11) is the only child of another node in P_{near} and the cumulative path cost to this child node is lower through p_{new} , then the child node is reconnected as a child to p_{new} (line 42) and the parent is deleted (line 40).
- **ForcedRemoval** function (line 15): There are cases, where a node cannot be added since there is no nodes with only one child to remove in P_{near}. In order to alleviate this situation, a global **ForcedRemoval** searches the whole tree for nodes without children and removes one randomly. For more details about the algorithm the reader may refer to [7].

Remark 3 : RRT and RRT*FN do not terminate when the goal region is reached, but the algorithm still executed for the number of iterations assumed for the convergence. This is done to improve the path's total length.*

Algorithm 2: RRT*FN path planner

Function: τ =**RRT*FN**($K_{loop}, p_0, p_{goal}, M$)
1: $\tau \leftarrow$**InitializeTree**();
3: **repeat**
4: **If** $M <$ **NodesAdded**(τ) **then**
5: $\tau_{old} \leftarrow \tau$
6: **end if**
7: $p_{rand} \leftarrow$**RandomState**$(\mathfrak{D}_{free})$
8: $p_{nearst} \leftarrow$**Nearest**(τ, p_{rand})
9: $p_{new} \leftarrow$**Steer**$(p_{nearest}, p_{rand})$
10: **If ObstacleFree**$(p_{nearest}, p_{new})$,**then**
11: $P_{near} \leftarrow$**Neighbors**(τ, p_{new})
12: $p_{min} \leftarrow$**ChooseParent**$(P_{near}, p_{nearest}, p_{new})$
13: $\tau \leftarrow$**InsertNode**(p_{min}, p_{new}, τ)
14: $\tau \leftarrow$**ReWire** $(\tau, P_{near}, p_{min}, p_{new})$
15: $\tau \leftarrow$**ForcedRemoval**(τ, w_{goal})
16: **end if**
17**: If NoRemovalPerformed**(), **then**
18: $\tau \leftarrow$**RestoreTree**()
19: **until** $(i + +> K_{loop})$
20 : **Return** τ

Function: p_{min}=**ChooseParent**$(P_{near}, p_{nearest}, p_{new})$
21: $p_{min} \leftarrow p_{nearest}$;
22: $c_{min} \leftarrow$**Cost**$(p_{nearest})+c(p_{new})$
23:**for** $p_{near} \in P_{near}$ **do**
24: $\dot{x} \leftarrow$**Steer**(p_{near}, p_{new})
25: **If ObstacleFree**$(p_{near}, \dot{x})$ **and** $\dot{x} = p_{new}$, **then**
26: $\dot{c} \leftarrow$**Cost**$(p_{near})+c(p_{new})$
27: **If** $\dot{c} <$**Cost**(p_{new}) and $\dot{c} < c_{min}$ **then**
28: $p_{min} \leftarrow p_{near}$
29: $c_{min} \leftarrow \dot{c}$
30: **end if**
31: **end if**
32: **end for**
33 : **Return** p_{min}

Function: τ =**ReWire**$(\tau, P_{near}, p_{min}, p_{new})$
34:**for** $p_{near} \in P_{near} \backslash p_{min}$ **do**
35: $\dot{x} \leftarrow$**Steer**(p_{new}, p_{near})
36: **If ObstacleFree**$(p_{new}, \dot{x})$ **and** $\dot{x} = p_{near}$, **and**
37: **Cost**$(p_{new})+c(\dot{x}) <$**Cost**(p_{near}) **then**
38: **If OnlyChild**(**Parent**(p_{near})) **and**
39: $M <$**NodesAdded**(τ) **then**
40: **RemoveNode(Parent(**p_{near}**))**
41: **end if**
42: $\boldsymbol{\tau} \leftarrow$**ReConnect** $(p_{new}, p_{near}, \tau)$
43: **end if**
44: **end for**
45: **Return** τ

Finally, the multi-directional RRT*FN searches the optimal paths from each point to their neighbors (point-to-points problem). At each iteration, Algorithm 3 uses the same tree as the Algorithm 2 to find the optimal paths from a one starting point to multiple goal points located in multiple disjoint regions. However, the tree is still growing until the entire goal regions are reached while the algorithm must be

restarted even if one feasible path to one goal state is not reached (line 22). The algorithm returns a complete tree as well as a vector of costs, from the starting point to the goal points (line 25).

Algorithm 3: Multi-RRT*FN path planner

Function: $(\tau, cost)$ = **Multi-RRT*FN**(K_{loop}, p_0, W, M)
1:$\tau \leftarrow$**InitializeTree**();
2: $W_{remain} \leftarrow W$
3:**repeat**
4: **If** $M <$ **NodesAdded**(τ) **then**
5 : $\tau_{old} \leftarrow \tau$
6: **end if**
7: $p_{rand} \leftarrow$**RandomState**$(\mathfrak{D}_{free})$
8: $p_{nearst} \leftarrow$**Nearest**(τ, p_{rand})
9: $p_{new} \leftarrow$**Steer**$(p_{nearest}, p_{rand})$
10: **If ObstacleFree**(p_{new}) **then**
11: $P_{near} \leftarrow$**Neighbors**(τ, p_{new})
12: $p_{min} \leftarrow$**ChooseParent**$(P_{near}, p_{nearest}, p_{new})$
13: $\tau \leftarrow$**InsertNode**(p_{min}, p_{new}, τ)
14: $\tau \leftarrow$**ReWire** $(\tau, P_{near}, p_{min}, p_{new})$
15: $\tau \leftarrow$**ForcedRemoval**(τ, p_{goal})
16: **end if**
17**: If NoRemovalPerformed**(), **then**
18 : $\tau \leftarrow$**RestoreTree**()
19: $W_{goal} \leftarrow$**ReachedROI**(τ)
20: $W_{remain} \leftarrow W_{remain}/W_{goal}$
21:**until** $(i + +> K_{loop})$
22:**If** $W_{remain} \neq \emptyset$, **then**
23: **Goto** line 4
24:$cost =$**CostDist**(p_0, W, τ)
25 : **Return** τ, $cost$

B. Second scale: GA based TSP algorithm

As mentioned in Section II, this paper addresses an optimal path planning problem for a quadrotor that has to pass by a finite number of points and return back to the initial point. The locations of POIs are known a-priori in a bounded two-dimensional space. This problem is well-known as the TSP NP-hard optimization problem where a broad range of algorithms and solutions are employed as for instance [8]. Herein, the heuristic algorithm based on Genetic Algorithms (GA) is proposed to deal with this problem under the name TSP-GA (see Algorithm 4). It uses the inter-costs $\mathcal{C}$ provided by Algorithm 3. **PopEvaluation** computes the fitness function for each member of the population (line 10). Then new individuals of the population are created using mutation to add randomization to the process, similar to that of the natural genome (**GeneticOperator**) (line 12). Finally, **BestRoute** points out the total optimal cost C_t as well as the optimal order of points O (line 11).

Algorithm 4: GA-TSP optimal path

Function: (T, O, C_t) =**GA-TSP**($K_{loop}, K_{TSP}, N_p, p_0, P, W, M$)
%Scale one
1: $L = length\ (P)$
2 :$(T(1), \mathcal{C}(1))$ = **Multi-RRT*FN**(K_{loop}, p_0, W, M)
3: $W_{remain} \leftarrow W/w_1$
4:**for** $i = 1$ **to** $i = L - 1$**do**
5: $(T(i+1), \mathcal{C}(i+1))$ = **Multi-RRT*FN**($K_{loop}, p_i, W_{remain}, M$)
6: $W_{remain} \leftarrow W_{remain}/w_{i+1}$
7:**end for**
%Scale two
8: **InitializePop**(N_p)
9:**for** $i = 0$ **to**$i = K_{TSP}$ **do**
10: **PopEvaluation**(N_p, D)
11: $(O, C_t) \leftarrow$**BestRoute()**
12: **GeneticOperator()**
13:**end for**
14:**Return** O, C_t.

C. Second scale: Savings based VRP algorithm

The major constraint in quadrotor-based coverage planning is the limited onboard energy. Therefore, we develop a methodology, based on the well-known Vehicle Routing Problem (VRP), to minimize the flight path considering energy limitation. VRP finds the minimal set of shortest routes for either single or multiple starting points. In this case, the quadrotor leaves from a starting point (base station) and returns back to that point after finishing each assigned tour to download collected data and of course recharge batteries until achieving the overall mission. We proceed by the method developed in [9] and called savings-VRP approach. Notice that the standard savings algorithm considers finding the set of shortest paths while guarantying the load capacity constraint. In our adapted algorithm, the load capacity is transformed into flight time capacity. This flight time capacity contains in addition to the required time to cover each point, the traveling time from a point to another point as expressed by (4). The savings is a heuristic algorithm that often yields a relatively good solution. The basic savings concept expresses the cost savings obtained by joining two routes into one route as illustrated in Fig. 3, where point 0 represents the base station.

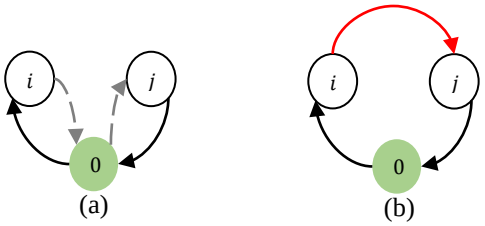


Fig. 3. Saving principle.

Initially in Fig. 3(a) locations i and j are visited on separate routes. An alternative to this is to visit the two locations on the same route, for example in the sequence $i - j$ as illustrated in Fig. 3(b). Because the traveling required durations are given, the savings that result from driving the route in Fig. 3(b) instead of the two routes in Fig. 3(a) can be calculated. Denoting the traveling time between two given points i and j by t_{ij} and the time required for covering q_i, the total time T_a in Fig. 3(a) is:

$$T_a = t_{0i} + q_i + t_{i0} + t_{0j} + q_j + t_{j0} \quad (5)$$

Similarly, the total traveling time T_b in Fig. 3(b) is:

$$T_b = t_{0i} + q_i + t_{ij} + q_j + t_{j0} \quad (6)$$

By combining the two routes, one obtains the savings S_{ij}:

$$S_{ij} = T_a - T_b = t_{i0} + t_{0j} - t_{ij} \quad (7)$$

Relatively large values of S_{ij} indicate that it is attractive, with regard to time traveling, to visit points i and j on the same route such that point j is visited immediately after point i. This approach is described by Algorithm 5.

Algorithm 5: Savings-VRP optimal path

1: execute Algorithm 4 until line 7
2: Calculate the savings S_{ij} for every pair (p_i, p_j).
3: Creates the "*savings list*" in descending order.
4: For the savings S_{ij} under consideration, include link (p_i, p_j) in a route following the rules of savings principle (see [9]).
5: If the savings list has not been exhausted, return to line 4, process the next entry in the list; otherwise, *stop*.
6: Any points that have not been assigned to a route during Step 4 must each be served by a vehicle route that begins at the base-station p_0 visits the unassigned point and returns to p_0.

IV. Space decomposition and POIs generation

The POIs may be moved, removed or added in specific locations by the user on the map. However, the main objective of this section is to generate automatically the set of POIs that allow the quadrotor to cover the overall target zones. Many strategies are used such as the map decomposition techniques where each target zone is decomposed into unified sub-regions, of equal size, called cells. These latter may cover partially or totally, precisely or approximately the overall target area. Therefore, the center of each cell that represents one ROI is considered as POI. Each ROI is covered when the quadrotor crosses it. This decomposition is related essentially to the shape and the size of the sensor's range. Usually, the nature of the mission defines the type of the used sensor such as infrared camera, gas sensor, chemical sensor, ultrasonic sensor, etc. A good review about the directional sensors is made in [10]. Therefore, we distinguish many regular and irregular shapes.

In fact, all the existing algorithms considering circular and rectangular cells have limited efficiency especially in the presence of complex shaped obstacles where the borders are generally ignored by the algorithm. The presence of the human operator is thus necessary to find the best locations of POIs nearby the obstacles and moves them to more advantageous positions. Moreover, the POIs are generated after obtaining the grid covering the map according to the shape of the sensor's range. Therefore, we seek to find a generic and scalable technique that allows to generate directly the POIs and still be applied whatever the shape and the size of the range.

We propose to discretize the closed bounded non-connected domains $\mathfrak{D}_{Target}$ using triangulation methods. This is referred to a mesh generation that provides a collection of vertices and edges constituting triangles covering-up the whole map. Unlike, the existing approaches, the vertices represent the set of POIs P. It is better than the choice of the centers of the cells. Then, any sensor can be used. The most available meshing software is just used as "black boxes" and difficult to be integrated with other codes. In reference [11] a simple and high quality-meshing open source was developed. It gives a specific attention to the borders of obstacles as well as the target zones with adaptation possibility of POIs density. These points are used directly whatever the shape or the size of the sensor. The density of points may be adjusted as well as the coverage rate using a scale factor. For the complete MATLAB code and further significations about this meshing algorithm, the reader may refer to [11].

V. RESULTS AND DISCUSSION

Before presenting the obtained results, we iterate firstly, our overall approach in order to achieve a coverage mission using quadrotor. It is summarized in six steps as follows:

1- Definition of the map by fixing the target zones (shape and boundaries) and obstacles.
2- Definition of the mission and the sensor (range, altitude).
3- Apply the mesh generation algorithm and get target zone decomposition.
4- From the discretized map, get the localization of the resulting POIs (vertices in the mesh model). The user can add, remove or move some POIs.
5- Apply Algorithm 4 (scale 1), based on multi-RRT*FN strategy, using the obtained POIs in order to obtain the pair-wise cost matrix and the real optimal connection graph.
6- Apply Algorithm 4 (scale 2) considering the matrix cost, to find the shortest path that relates the overall points and then turn to the initial point. If the onboard energy is not enough for the mission, a Savings-VRP can be used (Algorithm 5) to find the minimal set of shortest paths.

In this example of application, we assume usage of an AR-drone quadrotor for coverage planning of bounded and constrained area to demonstrate the effectiveness of our proposed approach. In order to reduce the consumed energy, the quadrotor is supposed flying at a constant altitude and seek to avoid the obstacles using only the lateral and longitudinal motions. Besides, bottom camera is used as sensor for photos shooting. The captured pictures have rectangular form (range of the sensor).

The workspace is considered in this case of study as a 2D bounded map $\mathfrak{D}(x,y) = [0m\ 5m] \times [0m\ 5m]$. The map is composed of one target region, $\mathfrak{D}_{Target} = \mathfrak{D}(x,y)$. In this target region, we distinguish one connected obstacle. The quadrotor can be enclosed by a circle of radius $r = 0.25\ m$. Therefore, the obstacle is enlarged by a distance of $r = 0.25m$. Therefore, the resulting fictive obstacle is $\mathfrak{D}_{obs}(x,y) = \partial\Delta_1$. This obstacle is delimited by the following coordinates (x,y): (1,1), (3.5,1), (4.5,2), (4.5,3) and (2.5,3), thus $\mathfrak{D}_{free} = \mathfrak{D} \backslash \mathfrak{D}_{obs}$. The overall algorithms are running using Matlab tool where four scenarios are proposed.

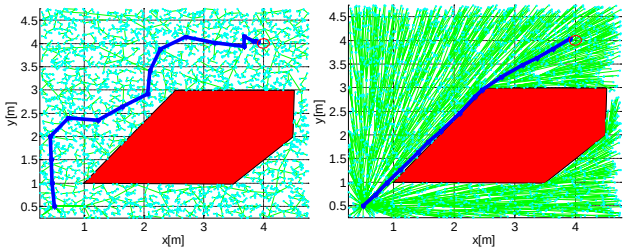
Scenario 1:

In this first scenario, we assess the performance and the efficiency of the RRT based algorithms, namely RRT, RRT* and RRT*FN, through a deep comparison. Thus, the quadrotor starts from an initial state: $p_0(x,y) = (0.5\ m, 0.5\ m)$ and flies toward the goal state $p_{goal}(x,y) = (4m, 4\ m)^T$. Herein, p_{goal} represents the unique POI, $P =$

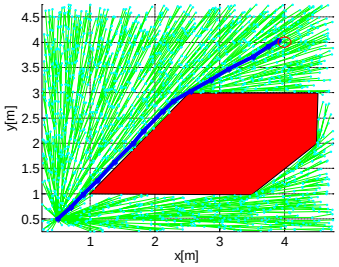
$\{p_{goal} \in \mathfrak{D}_{free}\}$, that represents the center of the ROI w_{goal} delimited by a square of width $L = 1\ m$. The parameters used in the algorithms are:

1. Max of iterations: $K_{loop} = 4000$
2. Max of nodes: $M = 2000$
3. Radius of local neighborhoods: $D_n = 1{,}5m$

The obtained results are summarized in Table 1 for the overall strategies where the corresponding paths are depicted in Fig. 4 (a), Fig. 4 (b) and Fig. 4 (c) for RRT, RRT* and RRT*FN respectively.



(a) RRT (b) RRT*



(c) RRT*FN

Fig. 4. One goal point path planning problem.

Table 1 : Summary of comparative analysis.

	RRT	RRT*	RRT*FN
Path cost (m)	6.4736	4.9492	4.9589
Run time (sec)	2.9373	11.9042	13.1847
Tree density (Number of nodes)	2993	2993	2000
Optimality	No	Yes	Yes

Obviously, from Fig. 4, RRT exhibits the worst performance with respect to optimality of the path. This is because no optimality criteria are considered. Both RRT* and RRT*FN converge towards an optimal path even though the rate of convergence is slower for the RRT*FN (13.1847 sec) where the final solution can be improved with respect to the number of iterations (see Table 1). We observe also that the trees looks denser, in the case of RRT*, than the case of RRT*FN. We should note that the required memory increases linearly as iterations increase in the case of RRT* (2993 nodes are generated). It adds nodes to improve the path without removing procedural. This latter makes the RRT*FN requires much less memory (fixed memory i.e. a fixed number of nodes M= 2000). RRT*FN still optimizes the path by adding better nodes and removing the ones with no children or one child.

Scenario 2:

In this second scenario, we show the result from the application of the multi-directional RRT*FN (see Algorithms 3). Using the same parameters as those mentioned in Scenario 1, the algorithm finds all the optimal paths from the starting state $p_0(x, y) = (0.5\ m, 0.5\ m)$ to the POIs $P = \{p_1(0.25, 4.75)\ ,\ p_2(4.75, 2)\ , p_3(2.5{,}4.75),\ p_4(4.5, 3)\} \in \mathfrak{D}_{free}$. These POIs represent the centers of ROIs $W = \{w_1, w_2, w_3, w_4\}$ that are delimited by squares of width $L = 1\ m$. The obtained results are displayed in Fig. 5.

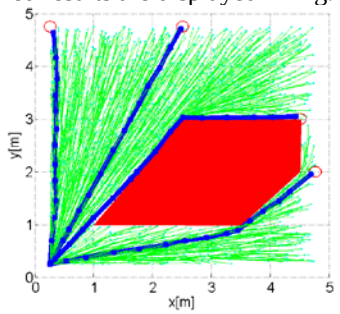


Fig. 5. Multi-RRT*FN based path planning.

We observe that the algorithm grows a tree from the starting point (one root) and find multiple optimal paths in the same tree. The optimal paths have lengths as: $c_{01} = 4.3918$, $c_{02} = 4.8831$, $c_{03} = 4.9831, c_{04} = 5.4956$.

Scenario 3:

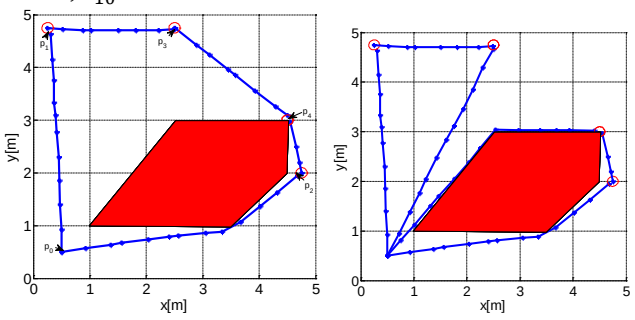
Considering the same map and the same set of POIs as in the previous scenarios, the quadrotor has to pass by all the points and then return back to the starting point by following the shortest path. To achieve this objective, Algorithm 4 is used. In this particular example, four sub-trees are grown where:

Tree 1: starting point: p_0 goal points: p_1, p_2, p_3, p_4
Tree 2: starting point: p_1 goal points: p_2, p_3, p_4
Tree 3: starting point: p_2 goal points: p_3, p_4
Tree 4: starting point: p_3 goal points: p_4

Once, the inter-costs for the set of points P are obtained, we start computing the shortest route that connects all the points. To do this, we use the second scale of Algorithm 4 by setting up the following conditions:

1. GA population size: N_p =60
2. Max of iterations $K_{TSP} = 4000$

As shown in Fig. 6 (a), good results are obtained. The global path, of cost $14.55m$, contains five sub-optimal paths of costs $c_{02} = 4.58\ m, c_{24} = 0.99\ m, c_{43} = 2.59\ m, c_{31} = 2.24\ m, c_{10} = 4.34m$.



(a) TSP problem (b) C-VRP problem

Fig. 6. Shortest path.

In the case of limited energy, many rounds must be supposed. In this example, we assume that the quadrotor has a capacity of 35 seconds flying at moderate speed with average value of 0.5 m/sec. At each point, the quadrotor hovers about 3 seconds. The obtained sequences and the shortest paths are

depicted in Fig. 6 (b) where the quadrotor passes by all the points in two rounds that are:

Sequence 1 that links the points p_0, p_2 and p_4 has a length of $10.69\,m$ and comprises three optimal sub-paths of costs: $c_{02} = 4.58\,m$, $c_{24} = 0.99\,m$ and $c_{40} = 5.12\,m$.

Sequence 2 that links the points p_0, p_3 and p_1 has a length of $21.70\,m$ and comprises of three optimal sub-paths $c_{03} = 4.14\,m$, $c_{31} = 2.24\,m$ and $c_{10} = 4.62$ m. This example is just to demonstrate the effectiveness of Algorithm 5. Because in real life, the quadrotors have an autonomy at least of 10 minutes.

Scenario 4:

In this scenario, the quadrotor should cover completely the map described in the previous scenarios. For this reason, we have to generate the POIs in order to achieve the given mission. The discretized map model using mesh generation is displayed in Fig. 7.

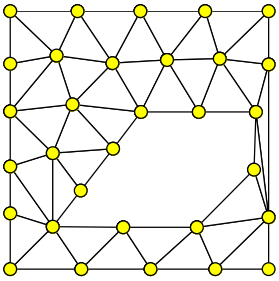
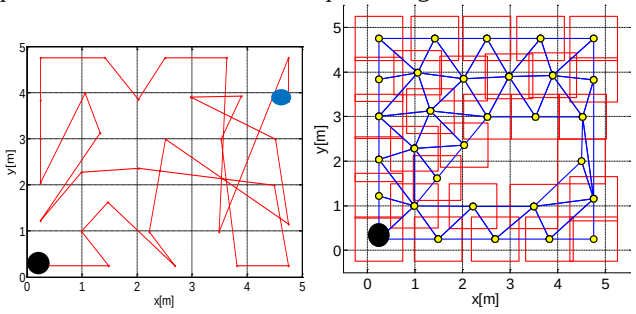


Fig. 7. Map meshing model.

The vertices of the triangles are considered as POIs. They are uniformly distributed on the map and represented by the yellow points (see Fig. 7). In order to cover totally this map, the quadrotor has to pass by all these points. The shortest path sequences is displayed in Fig. 8 (a). The footprints of the quadrotor passage are displayed in Fig. 8 (b). Notice that the quadrotor uses a camera with square range of width $L = 1\,m$.



(a) Shortest path (b) Camera footprints

Fig. 8. Coverage planning mission.

This technique ensures a coverage planning. The quadrotor starts from the base station (black disk) and finishes at the blue point then turns back to the base station. The developed approach allows covering almost all the map with a coverage rate about 94%.

VI. CONCLUSION & FUTURE IMPROVEMENTS

We have presented a coverage path-planning algorithm for VTOL quadrotors flying in 2D workspace while avoiding obstacles. The algorithm is executed in two steps: the first one is a RRT*-FN based algorithm that uses removal procedurals to maintain a fixed number of nodes, limiting the size of memory. This algorithm allows finding optimal paths from one point to their neighbors. The second stage allows connecting these points following the shortest path using GA. We provided also a scalable solution with respect to the consumed energy using savings technique. For a complete coverage scenario, a discrete model of the map composed of a collection of vertices and edges is obtained through a mesh generation. Then, from this discretized map, we define the set of POIs. Regarding future work many extensions to this paper can be made:

- Heuristics can be used to improve the convergence rate of our algorithm as in [4] by involving the Artificial Potential Fields.
- The coverage mission cannot be completed if the quadrotor is stopped by a threat. Therefore, a team of quadrotors will be used to cover the target area in our near future work.

REFERENCES

[1] J. M. Phillips, N. Bedrossian, and L. E. Kavraki, "Guided Expansive Spaces Trees: a search strategy for motion- and cost-constrained state spaces," *in IEEE International Conference on Robotics and Automation*, 2004, vol. 4, p. 3968–3973 Vol.4.

[2] 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, 1996.

[3] S. Lavalle "Planning Algorithms / Motion Planning." [Online]. Available: http://planning.cs.uiuc.edu/. [Accessed: 07-Jan-2017].

[4] P. Pharpatara, B. Hérissé, and Y. Bestaoui, "3-D Trajectory Planning of Aerial Vehicles Using RRT*," *IEEE Transactions on Control Systems Technology*, vol. PP, no. 99, pp. 1–8, 2016.

[5] P. Pharpatara, R. Pepy, B. Hérissé, and Y. Bestaoui, "Missile trajectory shaping using sampling-based path planning," *in IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2533–2538.

[6] S. Karaman, and E. Frazzoli, "Optimal Kinodynamic Motion Planning using Incremental Sampling-Based Methods," *in IEEE Conference on Decision and Control*, 2010.

[7] O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree based memory efficient motion planning," *in IEEE International Conference on Mechatronics and Automation*, 2013, pp. 354–359.

[8] Q. Zhu and S. Chen, "A New Ant Evolution Algorithm to Resolve TSP Problem," *in Sixth International Conference on Machine Learning and Applications*, 2007, pp. 62–66.

[9] Copyright (c) 1994-2016 by Michael G. Kay Matlog Version 17 21-Jan-2016 (http://www.ise.ncsu.edu/kay/matlog).

[10] M. Amac Guvensan and A. Gokhan Yavuz, "On coverage issues in directional sensor networks: A survey," Ad Hoc Networks, vol. 9, no. 7, pp. 1238–1255, 2011.

[11] Per-Olof Persson, *Mesh Generation for Implicit Geometries*, thesis, 2005.