# Two-scale geometric path planning of quadrotor with obstacle avoidance: First step toward coverage algorithm

Yasser Bouzid, Yasmina Bestaoui, Houria Siguerdidjane

# Two-scale geometric path planning of quadrotor with obstacle avoidance: first step toward coverage algorithm

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

***Abstract*— In this paper, a quadrotor path-planning problem while avoiding obstacles, in different shapes and sizes, is considered. The quadrotor is assumed to pass by a set of scattered points or regions defined according to the desired mission following the shortest path. The problem is solved by using a two-scale proposed algorithm. In the first scale, multi-directional optimal Rapidly-exploring Random Trees (RRT) based algorithm is used to define the shortest paths from each point to its neighbors. By means of the inter-costs provides 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). The effectiveness of the proposed algorithm is verified with numerical simulations.**

## I. INTRODUCTION

Nowadays, quadrotors have been efficiently employed in military as well as in civil missions and especially in cases that represent a real risk for humans as for instance: traffic accidents, fire emergencies, natural disasters, Nuclear-Biological-Chemical (NPC) contaminations, etc. In such events, the inspection as well as the coverage missions of damaged areas are usually long processes that involve large staff and wide sophisticated equipment. Moreover, these critical missions that cannot be ensured by ground mobile robot are solved easily using quadrotor bringing a notable benefit. Thus, it has to cover (inspect) a given set of points or regions of interest (either a POI or ROI), and then sends the collected information to the ground control station for deep processing and analysis. So, the challenge is to achieve such mission by following the shortest path that ensures the link between all the POIs (ROIs) while avoiding the obstacles.

In the last decades, coverage problems have attracted the interest of researchers particularly for the mobile robots. Furthermore, involving quadrotors in such scenarios is always under current investigating where many research fields are requested especially obstacle avoidance and path planning. Numerous path planning methods, in the robotic field, have been extensively studied such as potential field methods [1], as well as cell decomposition methods [2][3]. Moreover, sampling-based search methods such as: Expansive Space Trees (ESTs), Probabilistic Roadmap Methods (PRM) [4] or Rapidly exploring Random Trees (RRT) have been used to find collision-free trajectories in cluttered environments [5][6].

RRT based approach is very promising and efficient to single-query motion planning problems in constrained workspace [5]. However, only a feasible solution is obtained and no optimal solution is insured. Therefore, an alternative optimal RRT (RRT*) is proposed in order to tackle this problem by introducing incremental rewiring of the graph [7]. New states are considered as replacement parents for existing nearby states in the tree. However, this is inconsistent with its nature, single-query, and so becomes very expensive especially in high dimensions. In addition, for multi-directional path planning problem, this technique requires a huge memory to store the complete nodes. Recently, an extended algorithm has been 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) [8].

Many published papers consider Point-To-Point optimal path planning. However, some scenarios as for instance the coverage and inspection missions involve what we call Point-To-Points path planning problem. Therefore, in this paper, we contribute by introducing a 2D optimal strategy in constrained workspace using a holonomic quadrotor UAV. The workspace is considered as a set of Points Of Interest (POIs) (resp. Regions of Interest (ROIs)) cluttered by obstacles of different natures and shapes as for instance: radar detection areas, forbidden areas, hazardous zones and physical obstacles. Our strategy is composed of two scales:

(1) Multi-directional RRT*FN based algorithm is developed to compute the inter-costs from each POI to its neighbors following the optimal path considering obstacles avoidance.

(2) Traveling Salesman Problem (TSP) is resolved via genetic algorithms (GA) to compute the overall shortest

Y. Bouzid, Y. Bestaoui are with IBISC, Université d'Evry, Université Paris-Saclay, Evry, France; H. Siguerdidjane is with L2S, CentraleSupélec, Université Paris-Saclay, Gif sur yvette, France (Emails:yasseremp@gmail.com, yasmina.bestaoui@ufrst.univ-evry.fr and Houria.Siguerdidjane@centralesupelec.fr).

route that allows the quadrotor to visit all the POIs once and then returns to the starting point.

The efficiency of the approach is demonstrated by numerical simulations.

The remaining of the paper is organized as follows: the problem formulation is presented in Section II. Then, the algorithms are introduced in Section III. The application of the proposed algorithm as the simulation results are shown in Section IV. Finally, some concluding remarks are made in the last section.

## II. PROBLEM STATEMENT

The work takes into consideration some simplifying assumptions that are:

(A1) There is no wind or at least its effect is neglected.

(A2) The onboard energy is sufficient to accomplish the mission without returning to the starting point.

The covered area is represented as a discretized planar map where some critical points are considered as POIs according to the mission of the quadrotor, the nature of the area and the specifications fixed by the user. Thus, to accomplish a mission, the holonomic quadrotor has to pass through all the defined POIs on the planar map following the shortest path while avoiding the obstacles.

***Remark 1:*** *In fact, the quadrotor at each POI covers a region (ROI) where its size is defined according to the range of the used sensor ($\rho$) and its altitude from the ground ($h$). Usually, the each POI represents a center of ROI.*

***Remark 2:*** *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 by a radius $r$, then the quadrotor can be considered as a flying point (see Fig. 1).*

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 admissible or feasible states considered as connected space domain. Therefore, $\mathfrak{D}_{obs} \subseteq \mathfrak{D} \backslash \mathfrak{D}_{free}$ represents the set of forbidden states in collision with obstacles or no-fly zones. $\mathfrak{D}_{obs} = \bigcup_{i=1}^{l} \partial\Delta_{\mathrm{i}}$ may be a non-connected space domain where $l$ denotes the number of sub-connected obstacles or non-fly zones.

Let $p_0 \in \mathfrak{D}_{free}$ be the initial location 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 feasible states called ROI. In other words, $W = \{w_i \subseteq \mathfrak{D}_{free} \, | p_i \in w_i, \; i = 1, \dots, N-1\}$ represents the set of ROIs.

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 joins a point $p_i$ by a point $p_j$.

The optimal path planning problem is then decomposed 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 $p_j$, $i = 0, \dots, N-1, j = 0, \dots, N-1$ and $i \neq j$.

This optimal planning problem is then defined as the search for the paths, $\sigma_{ij}^*$, 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.

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

$i = 0, \dots, N-1, j = 0, \dots, N-1$ 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 $\sigma_{ij}^*$ from $p_i$ to $p_j$ where $c_{ij} = c_{ji}$ and $\sigma_{ij}^* \equiv \sigma_{ji}^*$.

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

The solution of this problem is complex and requires a multi-directional based path planning technique. Therefore, we propose a new multi-directional optimal RRT based algorithm. The path planning algorithm returns a global tree $\mathcal{T} = (V, E)$ consisting of $(N-1)$ sub-trees. From these trees we find $\sum_{i=1}^{i=n}(N-i)$ shortest path $\sigma_{ij}^*$ with the minimal costs $c_{ij}$.

The cost quantities of the optimal paths can be interpreted as a travel distance, travel time or the number of nodes... As the quadrotor is considered, flying in holonomic way, the Euclidean distance between nodes is suitable as a metric of optimality. However, this cannot measure the true distance between two states for a non-holonomic-like navigation way.

- **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 turn 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,

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

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 well-known as Traveling Salesman Problem (TSP) and can be solved using Genetic Algorithms (GA).

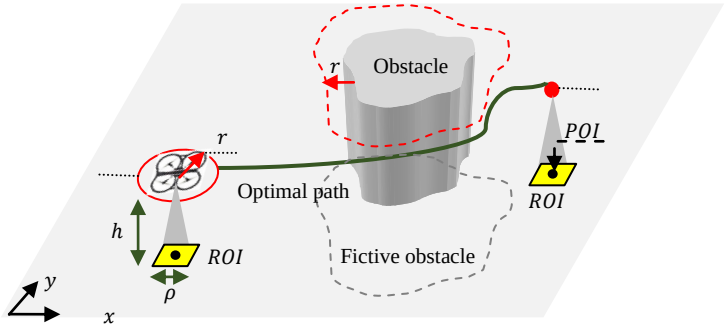



Fig. 1. Optimal path planning problem with obstacles avoidance in 2D map using quadrotor.

## III. ALGORITHMS DESCRIPTION

As stated in the previous section, our approach is composed of two distinct scales. The first scale is used to obtain the inter-costs allowing the second scale to find the overall optimal path.

### A. *First scale RRT based algorithm*

For this first scale, we use the RRT-based approach. This latter performs efficient searches even in high dimensional and very constrained spaces. For the sake of clarity, we explain the basic RRT based path algorithm. It 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 7). The main routines are:

- **Nearest** function (line 8): finds the nearest node $p_{nearest}$ in the tree to $p_{rand}$.
- **Steer** function (line 9): 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 10), $p_{new}$ is added to the tree $\tau$ as a child of $p_{nearest}$ (line 13). The algorithm holds until iteration $K_{loop}$ or when a first path is found (see a part of Algorithm 1).

Using RRT, no optimal solution is ensured due to the fact that no metric is used to measure the optimality of the trajectories. Therefore, RRT* algorithm is introduced in [7] 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 functionalities 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 to 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 the 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 its parent node and creates an edge between $p_{near}$ and $p_{new}$, the new parent node. For more details about the RRT* algorithm and the additional routines, the reader may refer to [7].

Because there is no prior limits to the number of nodes in the 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 1 where the main rules are:

1- 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.
2- A new node is added, if and only if there exists, at least, one node in the tree with one or no child (is deleted from the tree) and the cost to the present state from the initial state has decreased.
3- 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 node with only 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 [8].

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

**Algorithm 1** RRT*FN path planner

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

**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**: $\tau$ =**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**$(\tau)$ **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** $\tau$

Finally, the multi-directional RRT*FN searches the optimal paths from each point to its neighbors (Point-To-Points problem). At each iteration, Algorithm 2 grows the same tree as the Algorithm 1 to find the optimal paths from one starting point to multiple goal points located in multiple disjoint regions. However, the tree still grows until all the goal regions are reached where 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 24).

**Algorithm 2** 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**($\tau$) **then**
5 : $\tau_{old} \leftarrow \tau$
6: **end if**
7: $p_{rand} \leftarrow$**RandomState**$\left(\mathfrak{D}_{free}\right)$
8: $p_{nearst} \leftarrow$**Nearest**$(\tau, p_{rand})$
9: $p_{new} \leftarrow$**Steer**$(p_{nearest}, p_{rand})$
10: **If ObstacleFree**$(p_{new})$ **then**
11: $P_{near} \leftarrow$**Neighbors**$(\tau, p_{new})$
12: $p_{min} \leftarrow$**ChooseParent**$(P_{near}, p_{nearest}, p_{new})$
13 : $\tau \leftarrow$**InsertNode**$(p_{min}, p_{new}, \tau)$
14: $\tau \leftarrow$**ReWire** $(\tau, P_{near}, p_{min}, p_{new})$
15 : $\tau \leftarrow$**ForcedRemoval**$(\tau, p_{goal})$
16: **end if**
17**:** **If NoRemovalPerformed**(), **then**
18 : $\tau \leftarrow$ **RestoreTree**()
19: $W_{goal} \leftarrow$**ReachedROI**$(\tau)$
20: $W_{remain} \leftarrow W_{remain}/W_{goal}$
21: **until** $\left(i++> K_{loop}\right)$
22: **If** $W_{remain} \neq \emptyset$, **then**
23: **Go to** line 4
24: $d =$**CostDist(**$p_0, W, \tau$**)**
24 : **Return** $\tau$, $cost$

*B. Second scale GA-TSP algorithm*

As said 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 then return to the initial point. The locations of POIs are known a priori in a bounded two-dimensional space. This problem is well-known as TSP NP-hard optimization problem where a broad range of algorithms and solutions are employed as for instance [9] [10]. Herein, a heuristic algorithm based on Genetic Algorithms (GA) is proposed to deal with this problem under the name TSP-GA (see Algorithm 3). It uses the inter-costs $d$ provided by Algorithm 2 as input. **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 $C$ as well as the optimal order of points $O$ (line 11).

**Algorithm 3** GA-TSP optimal path

**Function**: $(\mathrm{T}, O, C)$ = **GA-TSP**( $K_{loop}, K_{TSP}, N_p, p_0, P, W, M$)

***%Scale one***

$L = length\ (P)$
:$(T(1), D(1))$ = **Multi-RRT*FN**$(K_{loop}, p_0, W, M)$
$W_{remain} \leftarrow W/w_1$
**for** $i = 1$ **to** $i = L - 1$ **do**
$(T(i+1), D(i+1))$ = **Multi-RRT*FN**$(K_{loop}, p_i, W_{remain}, M)$
$W_{remain} \leftarrow W_{remain}/w_{i+1}$
**end for**

***%Scale two***

**InitializePop**$(N_p)$
**for** $i = 0$ **to** $i = K_{TSP}$ **do**
**PopEvaluation**$(N_p, D)$
$(O, C) \leftarrow$ **BestRoute()**
**GeneticOperator()**
**end for**
**Return** $O, C$.

## IV. RESULTS AND DISCUSSION

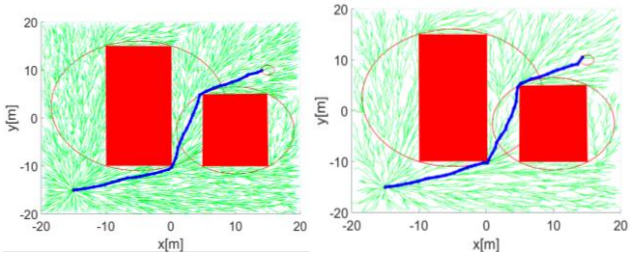
In this section, 2D RRT based motion planning strategies, using a holonomic quadrotor, are implemented for the sake of validation. The workspace considered in this case of study is described as a 2D bounded map $\mathfrak{D}(x, y) = [-20m\ 20m] \times [-20m\ 20m]$ with two non-connected static obstacles. In fact, these obstacles are enlarged by a distance $r = 0{,}25\ m$. Therefore, the resulting fictive obstacles are $: \mathfrak{D}_{obs}(x, y) = \partial\Delta_1 \cup \partial\Delta_2 = [-10m\ 0m] \times [-10m\ 15m] \cup [5m\ 15m] \times [-10m\ 5m]$. Thus $\mathfrak{D}_{free} = \mathfrak{D} \backslash \mathfrak{D}_{obs}$.

***Scenario 1:*** In this first scenario, we assess the performance and the efficiency of the RRT based algorithms, namely RRT* and RRT*FN, through a deep comparison. Thus, the quadrotor starts from an initial state: $p_0(x, y) = (-15m, -15m)$ towards the goal state $p_{goal}(x, y) = (15m, 10m)$. Herein, $p_{goal}$ represents the unique POI, where $P = \{p_{goal} \in \mathfrak{D}_{free}\}$, represents the center of the ROI $w_{goal}$ delimited by a circle of radius $\rho = 0{,}5m$. The parameters used in the algorithms are:

- Max of iterations: $K_{loop} = 9000$
- Max of nodes: $M = 3000$
- 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. 2.

Both RRT* and RRT*FN converge towards an optimal path even though the rate of convergence is slower for the RRT*FN where the final solution improves with respect to the number of iterations (see Table 1). We observe that the trees look denser, in the case of RRT* than the case of RRT*FN (see Fig. 2 (a) and Fig. 2(b)). We note that the required memory increases linearly as iterations increase in the case of RRT* (6743 nodes are generated). It adds nodes to improve the path without removing procedural. This latter makes the RRT*FN requiring much less memory (fixed memory i.e. a fixed number of nodes M= 3000). Before the maximum number of nodes is reached, both techniques behave similarly to each other. Then, RRT*FN still optimizes the path by adding better nodes and removing the ones with no children or one child. Moreover, the RRT*FN algorithm is able to get close to the optimal solution within reasonable computation time.




(a) RRT* (b) RRT*FN

Fig. 2. RRT based path planning.

Table 1 : Summary of comparative analysis.

| | RRT* | RRT*FN |
|---|---|---|
| Path cost (m) | 43.4329 | 43.9561 |
| Run time (sec) | 19.6595 | 23.5364 |
| Tree density (Number of nodes) | 6743 | 3000 |
| Optimality | Yes | Yes |

***Scenario 2:*** In this second scenario, we show the result from the application of the multi-directional RRT*FN (see Algorithms 2). Using the same parameters as those mentioned in Scenario 1, the algorithm finds all the optimal paths from the starting state $p_0(x, y) = (-15m, -15m)$ to the POIs $P = \{p_1(-5, -17), p_2(2, 2), p_3(-5, 16), p_4(16, 0)\} \in \mathfrak{D}_{free}$. These POIs represent the centers of ROIs $W = \{w_1, w_2, w_3, w_4\}$ that are delimited by circles of radius $\rho = 0{,}5m$. The obtained results are displayed in Fig. 3.

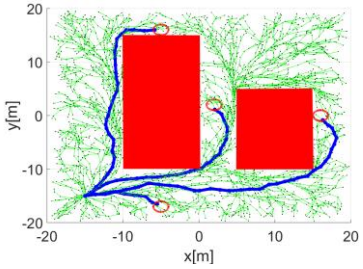



Fig. 3. Multi-RRT*FN based path planning.

We observe that the algorithm grows a tree from the starting point (one root) and finds multiple optimal paths in the same tree.

***Scenario 3:*** Considering the same map and the same set of POIs as in the previous scenario, the quadrotor has to go through all the points and then return to the starting point by

following the shortest path. To achieve this objective, Algorithm 3 is used incorporating the multi-directional RRT*FN algorithm. This latter (first scale) allows obtaining the inter-costs of the optimal paths. In this particular example, four 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: $p_4$

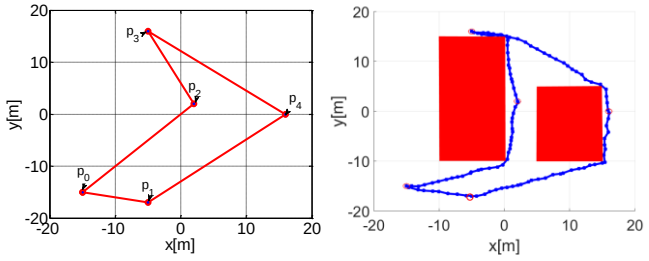
Once, the inter-costs for the set of points $P$ are obtained, we start computing the shortest route that connects all the points. To do this, we use the second scale of Algorithm 3 by setting up the following conditions:

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

Fig. 4 (a) shows the order of points that ensures the shortest path while the planned path is shown in Fig. 4 (b).

As shown in Fig. 4, good results are obtained. The global path, of cost 117.03 m, contains five sub-optimal paths where their costs are: $c_{01} = 9.844m$, $c_{14} = 31.85m$, $c_{43} = 28.58m$, $c_{32} = 18.28m$, $c_{20} = 28.46m$.

In fact, for high dimension problems, the GA does not guarantee to find the shortest path, although it approaches it. However, the use of exact techniques renders the convergence to the optimal path very slow and takes very large time.




(a) TSP solution (b) Shortest path

Fig. 4. Shortest path using GA-TSP algorithm.

## V. CONCLUSION & FUTURE IMPROVEMENTS

We have presented an efficient path planning algorithm for VTOL quadrotors flying in 2D workspace while avoiding static obstacles. The algorithm is executed in two stages: the first one is a RRT based algorithm that uses removal procedures to maintain a fixed number of nodes, which limits 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. The efficiency of the algorithm is shown through numerical simulations.

The proposed algorithm may be employed for coverage scenario where the first step is to generate a discrete map composed of a collection of vertices and edges that constitute some geometrical shapes that represent the ROIs and covering the whole map. Then, from these ROIs in the discretized map, we define the set of POIs. The coverage problem is then solved by first executing the first scale of Algorithm 3 in order to obtain the matrix of costs and second a search to determine in which order the ROIs should be covered using the second scale of Algorithm 3.

Regarding future work many expansions in this paper can be made:

- The limited onboard energy can be considered with respect to the long missions. This pushes to consider the well-known Vehicle Routing Problem (VRP) or by involving multi-quadrotors in formation.

## References


[1] R. Daily and D. M. Bevly, "Harmonic potential field path planning for high speed vehicles," in *2008 American Control Conference*, 2008, pp. 4609–4614.

[2] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, Nov. 2015.

[3] J. Carsten, D. Ferguson, and A. Stentz, "3D Field D: Improved Path Planning and Replanning in Three Dimensions," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3381–3386.

[4] 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.

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

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

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

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

[9] M. Mi, X. Huifeng, Z. Ming, and G. Yu, "An Improved Differential Evolution Algorithm for TSP Problem," in *2010 International Conference on Intelligent Computation Technology and Automation*, 2010, vol. 1, pp. 544–547.

[10] X. s Yan, H. m Liu, J. Yan, and Q. h Wu, "A Fast Evolutionary Algorithm for Traveling Salesman Problem," in *Third International Conference on Natural Computation (ICNC 2007)*, 2007, vol. 4, pp. 85–90.