Chapter 12

A Random Multi-Trajectory Generation Method for Online Emergency Threat Management (Analysis and Application in Path Planning Algorithm)

Liang Yang, Yuqing He, Jizhong Xiao, Bing Li and Zhaoming Liu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71410

Abstract

This paper presents a novel randomized path planning algorithm, which is a goal and homology biased sampling based algorithm called Multiple Guiding Attraction based Random Tree, and robots can use it to tackle pop-up and moving threats under kinodynamic constraints. Our proposed method considers the kinematics and dynamics constraints, using obstacle information to perform informed sampling and redistribution around collision region toward valid routing. We pioneeringly propose a multiple path planning method using 'Extending Forbidden' algorithm, rather than using variant cost principles for online threat management. The threat management method performs online path switching between the planned multiple paths, which is proved with better time performance than conventional approaches. The proposed method has advantage in exploration in obstacle crowded environment, where narrow corridor fails using the general sampling based exploration methods. We perform detailed comparative experiments with peer approaches in cluttered environment, and point out the advantages in time and mission performance.

Keywords: multiple path planning, online emergency threat management, path switching, goal biased probability, sampling based algorithm

1. Introduction

Robot path planning have been witnessed a great achievement these years with the various application of robots [1, 2]. Problems such as path planning, motion planning, and online moving obstacle management have been widely studied toward the goal of performing autonomy. Unmanned Aerial Vehicles (UAVs), an easy access robot platform, has been increasingly

applied in research and commercial areas in recent years. UAV autonomy denotes the ability of tackling with obstacle (or called no-fly zone) avoidance and trajectory planning online from a starting position to a destination while satisfying the kinematic constraints [3].

For robot path planning, emergency threat management (ETM) is one of the hardest challenges that needs to be solved, where a sudden threat may burst into view or dynamic obstacles are detected on line, especially when UAV is following the desired path. Under such conditions, UAV should consider the following attributes:

1. Time efficiency: The most important requirement for ETM algorithm is time efficiency. For general ETM, the configuration is periodically updated, such as heuristic algorithm A* [4], which it is computationally intensive if the map is represented with high resolution. In order to guarantee safety, ETM requires real-time performance.

2. Kinematic feasibility: Kinematic feasibility denotes that the output of the planner meets the kinematic constraints of the robot as well as the environment. The constraints include: (a) Path smoothness: The planner is required to output kinematic smooth path, sometimes even kinodynamically feasible as well. Thus, the path should meet the state of art tracking constraints, and enables low tracking error for UAV; (b) Minimum cost of switching: The strategy of handling the threat, especially ET, is to find the cost minimum path by generating a new path or multiple paths besides the initial one. The cost for choosing the best path should take the dynamic constraints, energy consumption and time performance into consideration.

3. Specific requirements: UAVs have already been applied to many areas, such as inspection, photography, and monitoring. They have to meet some specific requirements according to environments and system constraints. For example, best pose based illumination of tunnel inspection for crack and spalling [5], and stable tracking with obstacle avoidance as UAV photography [6] which should be able to keep stable capturing even during the flying.

Development with open robot platform [7] and field implementation [8] has witnessed the promising performance of Sampling Based (SB) methods. SB algorithms (SBA) have the advantages for planning in high dimensional space, and it is with the ability to deal with multiple classes of path or motion planning problem in both static and dynamic environment [9]. Rapidly-exploring random trees (RRTs) are single query methods which obtain Voronoi biased property and only generate homotopy paths simultaneously [12]. Although it proposes to solve the multiple degrees of freedom (DOF) operating problems in known static environments [10, 11], SBA shows great performance of dealing with any kind of path or motion planning problem in complex environments for unmanned ground robots or aerial robots.

In this paper, we introduce two biased-sampling methods, which are obstacle biased and Homologous Classes (HC) biased to perform path planning respectively. For obstacle biased path method, we have discussed in [13] with UAV demonstration. For HC classed biased approach, it aims at solving the ET problem by generating alternative paths for online dynamically switching. HC introduces an online dynamic reconfiguration approach to ensure singularity between paths, which tries to generate more paths with different obstacle reference. Thus, it can perform alternative switching online when confronted with ET. The obstacle biased planning method is called Guiding Attraction based Random Tree (GART) and HC

http://dx.doi.org/10.5772/intechopen.71410

biased is called Multi-GART (MGART). We consider the environment to be known as a priori to us, and the UAVs are with the ability to understand the clearance region. Experiments and comparative simulations are illustrated to provide the effective evaluation of the proposed methods.

2. Preliminary materials

2.1. Homology and homotopy

For path planner, the purpose is to find a feasible path p_f (cost minimum or complete) from the initial position to the goal position in the workspace $W \in R^{\{n\}}$, n denotes the dimension of space the robots locate. A general cost function can be represented as:

$$J = \int c_{energy} + c_{time} + c_{threat} dt \tag{1}$$

The c_{energy}, c_{time}, c_{threat} denote energy cost, time consumption, and threat respectively. These costs are not fixed, since the energy cost can be path length, and time consumption can change according to the velocity limitation. For cost constrained path planner, the goal is to find the asymptotic optimal rather than the completeness solution. Then, more than one path can be found during the process, and the paths can be homotopic or belong to different homotopic classes (or called homology).

It is illustrated in **Figure 1**. Given a continuous map $H: I \times I \to \Gamma$ or $H: h(s, t) = h_t(s)$, Γ denotes the topological space and $I = [0, 1]$ is the unit interval. The obstacle regions are labeled with R_{o1}, $x_{initial} = h(0, t)$ denotes start point, $x_{inter} = h(1, t)$ denotes the goal position, x_{inter} denotes an inter node for obstacle avoidance. For the continuous deformation, given $h(s, 0) = \pi_0$, $h(s, 1) = \pi_1$, the path can be continuously mapping through π_0 to π_1 with $t \in [0, 1]$. For any path deformed between, they are homotopic with π_0, π_1 if and only if they stay in the closed loop $\pi_0 \coprod -\pi_1$, where the closed loop cannot collide with any obstacle region.

Definition 1—Homotopic Paths: It denotes the equivalence class of all paths under continuous mapping $H: h(s, t) = h_t(s)$, which locates in the closed loop formed $h(s, 0) \coprod -h(s, 1)$. Any

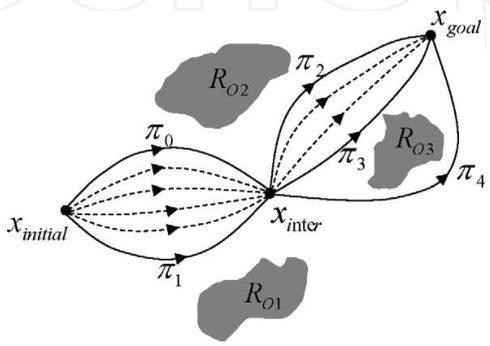


Figure 1. Homotopic and homologous classes and paths.

path in the set can be continuously deformed into any other without colliding with any obstacle in the space. For all paths in the set, they are with the same fixed end points.

We can conclude that π_2 and π_3 belong to the same homotopic class. However, we can find path π_4, which shares the same start and ending node, cannot be continuously deformed to π_3 due to the isolation of the obstacle. It means $(\pi_3 \cup -\pi_4) \cap R_{o3} \neq \Theta$. In such case, we call π_3 and π_4 are homologous, and they belong to different homotopic classes.

Definition 2—Homologous Paths: Paths, which follows the same continuous mapping $H : h(s, t) = h_t(s)$, cannot form a closed loop by $h(s, 0) \coprod -h(s, 1)$. The homologous paths belong to different homotopic classes.

2.2. Problem statement

Path planning follows a common procedure to perform trial and error process under empirical constraint to achieve completeness. The problem of path planning does not only solve a problem for exploration optimization, but also try to model the environment with a best descriptor as discussed in [13]. Let us take a look again with the problem of path planning which can be represented as:

$$H = \{h(s) | x_{initial} = h(0), x_{initial} = h(1), s \in [0, 1]\} \tag{2}$$

The path h(s) (homologous) should stay in obstacle free region R_{free}, that is, $h(s) \in R_{free}$. Usually, the path is piecewise continuously, and it can also be smoothed to obtain first order continuous thus to ensure kinematics continuous [14]. Besides the exploration to achieve completeness (in Eq. (1)), the obstacle modeling method is also important and affect the planning results.

To solve this problem, this paper proposed a multi-path online switching approach, that is, the path planner can find alternative homologous-paths. Then, this paper designs an online fast switching strategy. For multiple path planner, it aims at finding as many paths as possible,

$$H_{alter} = \cup_{i=1} h^i(x(t), u(t)) \tag{3}$$

H_{alter} denotes the set of all the alternative paths $h^i(x(t), u(t))$, x(t) denotes the state, and u(t) denotes the control. However, the mission planner cannot use all the planed paths for online switching, it should find the reasonable paths without redundancy. We propose the follow rule,

$$H_{reason} = \left\{ h^i \middle| h_i \underset{\neq}{H} h_j, \nabla i \neq j \right\} \tag{4}$$

H_{reason} denotes the paths set where any two paths are not homotopic to each other, $\underset{\neq}{H}$ denotes non-homotopy. Now, we have the paths which keep distinguishable from each other with different obstacles sequence surrounding.

http://dx.doi.org/10.5772/intechopen.71410

3. Rapidly exploring random tree path planner

In this section, we try to describe the underlying research of rapidly-exploring random tree (RRT [12], upon which we propose a novel state of art approach to facilitate the active exploration in cluttered environments). SBAs are incremental search algorithms which perform random sampling with collision checking for extension, and they were first proposed to solve high dimension planning problem. They have the merits of considering the control and kinematics constraints together, and can incrementally construct a tree from the start node (or initial state) to the goal node (or goal state) with continuously sampling and expansion.

It is shown **Figure 2**, the whole tree graph by exploration is represented as G_T, the black solid dot denotes the valid state within step accessibility under kinematics constraints, and the black solid lines connect each parent state with child state for extension. Every step, a new sample g_{sample} will be generated randomly. It should be cleared to all that the initial random sampling does not mean a fixed connection, that is, the random sampling can be a direction for extending. Then, the random sample g_{sample} tries to find the nearest state in the tree for connection under minimum Euclidean metric,

$$g_{parent} = \sup \min_{g_i \in G_T} E\left(g_i, g_{sample}\right) \tag{5}$$

Where g_i is an element of all valid states set G_T.

3.1. RRT connection with kinematic constraints

For RRT planner, given a system with state $\left(\dot{x}_x, \dot{x}_y, \dot{\theta}\right)$, and a general form of system model:

$$\begin{cases} \dot{X}_x = f^x(x_x, u_x) \\ \dot{x}_y = f^y\left(x_x, u_y\right) \\ \dot{\theta} = f^\theta(\theta, u_\theta) \end{cases} \tag{6}$$

It can extend with simply random sampling with control inputs $[u_x, u_y, u_\theta]$. The random sample has to follow the kinematics constraints. Given the robot system, the differential

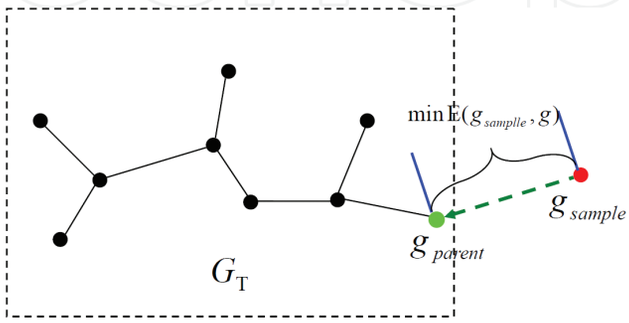


Figure 2. RRTs propagate by applying the minimal cost criterion to connect the newly sampled guard to the previous tree.

constraints can be represented as a set of implicit equations as $g(x, \dot{x}) = 0$, and it can be further represented as:

$$\dot{x} = f(x, u) \tag{7}$$

Here, x denotes the state, and $u \in U$ denotes the valid control in allowable controls set. Given the parent state $g_{parent}(t)$, the time step follows a Δt limits. Then, the control inputs vary with $u = \{u(t') \mid t \le t' \le t + \Delta t\}$. To compute $x(t + \Delta t)$, we can follow a typical procedure as [12]. It should be noted that the planner should extend toward the newly sampled g_{sample}. The planner first computes the possible region of reachability from current state x(t):

$$x(t + \Delta t) \in [x(t) + f(x(t), u(t) - \Delta t \cdot \epsilon), x(t) + f(x(t), u(t) + \Delta t \cdot \epsilon)] \tag{8}$$

where ϵ is the maximum first order factor of control input. RRT now picks a new state along the direction from parent to new sample, that is, $g_{new} \in [x(t) + f(x(t), u(t) - \Delta t \cdot \epsilon), x(t) + f(x(t), u(t) + \Delta t \cdot \epsilon)]$ and $g_{new} = g_{parent} + \delta(g_{sample} - g_{parent})$ with $\delta \in [0, 1]$.

3.2. Voronoi biased incremental search

Before discussing the Voronoi biased property of the SBAs, let first introduce some basic notation. Given a set of points $S = \{s_i \mid i = 1, 2, \ldots, n\}$ in a n-dimension space X. For any two distinct points s_p and s_q in set S, the dominant region (Voronoi region) of s_p over s_q is defined as the region where any inside point should be closer to s_p over s_q, that is,

$$R_{s_p} = \left\{ \chi \in \left| |s_p - \chi|^L < |s_q - \chi|^L \right. \right\} \tag{9}$$

Where χ is the dominant region corresponding to s_p, $||^L$ denotes the Lebesgue measurement. In a normal case, any point s_i has its own dominant region with,

$$R_{s_i} = \left\{ \chi \in \left| |s_i - \chi|^L < |s_j - \chi|^L, \text{ for all } i \neq j \right. \right\} \tag{10}$$

Normally, random sampling of RRT follows a Monte-Carlo Method [15] to perform an uniformly sampling in a n-dimensional space under Lebesgue measurement. We can look back at the beginning of Section 3, the new sampled node tries to connect to the nearest node under Euclidean metric. We can now analyze the problem in another perspective that given g_{parent} and g_s, they connect to the same origin g_o. Then, a new sample g_{sample} is generated randomly following a Monte-Carlo process. In order to explore and reach the goal, g_{sample} tries to connect to the tree following the metric defined in Eq. (5). It means that g_{parent} and g_s can be connected for expansion under minimum distance principle, then g_{sample} has to be assigned to the dominant region which subjects to a closer point (the Voronoi region). Under this principle, g_{parent} and g_s can acquire new resource for extension with the ability to keep distinct region and extending their branches.

A typical Voronoi-biased exploration using sampling can be seen in **Figure 3**, where each branch keeps distinct with each other to form a star network like structure and it behaves the

http://dx.doi.org/10.5772/intechopen.71410

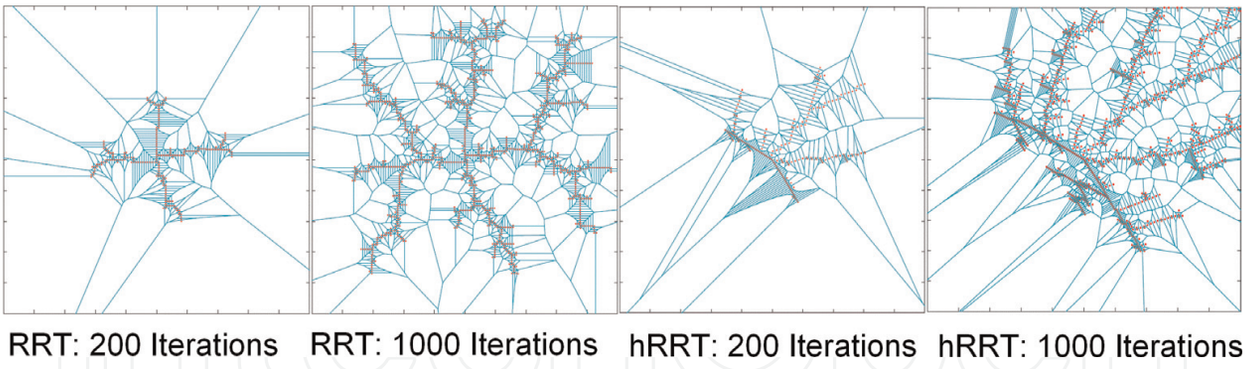


Figure 3. Results of incremental exploration of RRT and hRRT [16] after 200 and 1000 iterations, respectively.

same for heuristic informed RRT [16]. Here, unlike the dominant region of a point, RRT branch can be also treated as a distinct individual with its own Voronoi region for acquiring the extending resource.

4. Obstacle and homology biased path planner

In this Section, we propose approaches to solve two main problems, which are handling cluttered environment and online ET processing, using obstacle-biased method and homology-biased method. Collision detection during the incremental exploration is time consuming, and it follows a routine procedure to guarantee safety. It should be noted that the step validation of each new sampling state provides the directional information of obstacle distribution.

4.1. Obstacle biased path planner under kinematic constraints

SBAs mostly deploy the general idea of generating random samples for incremental exploration, and the sample locating in obstacle region will be discarded since it is time consuming and no benefits for increasing the performance of exploring. We firstly deployed a simple idea which was proved to have much higher time performance then RRT and RRT* in [17].

This paper introduces an obstacle biased algorithm, using obstacle information to help generating more samples for connection. It is shown in **Figure 4**, the newly sampled states x_1^s, x_1^s tries to connect to the nearest state in the tree. However, x_1^s leads toward the obstacle region, x_2^s locates in obstacle region. To use the obstacle information, this paper proposes an active exploring method, that is, inner propulsion and outer attraction.

For outer attraction, new sample x_1^s performs a collision checking, and find the nearest nodes ${}^o x_a$, ${}^o x_b$. We define that the further the obstacle to the sample, the more attraction it can support, that is, the attraction is proportional to the distance between obstacle and the sample using L2-norm L_2. The sample then re-allocation by add a obstacle biased attraction as:

$$x_1^{s*} = x_1^s + k\left[\left({}^o x_a - x_1^s\right) + \left({}^o x_b - x_1^s\right)\right] \tag{11}$$

Where k is a constant to adjust the shifting percentage of the attraction vector.

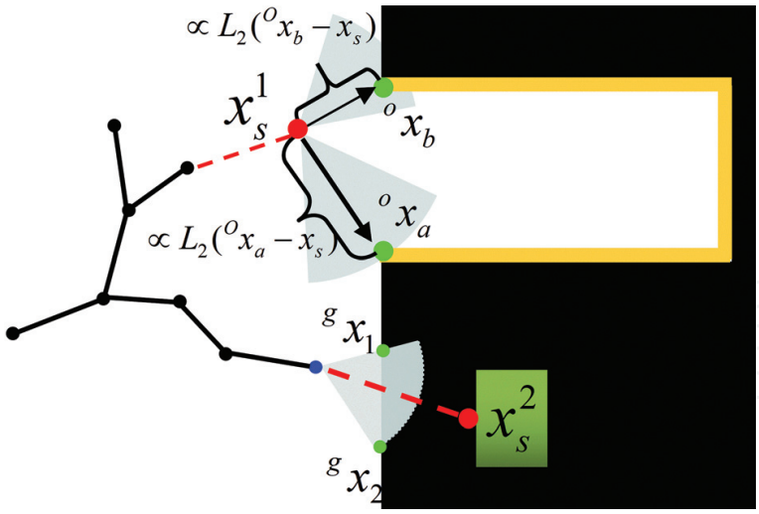


Figure 4. Obstacle biased SBA uses the obstacle location as input, with inner propulsion and outer attraction, to generate more samples for exploration. x_1^s, x_2^s are new samples, the black region denotes obstacle region.

The inner sample in collision with the obstacle is regarded to provide guiding information for the algorithm. This paper tries to find two more states ${}^{g}x_1, {}^{g}x_2$ within kinematic reachable region (discussed with Eq. (8)), it tries to find out the first two safe state with two directions which are out of obstacle region in the kinematic reachable region (the light blue fan-shaped region). Then, the two newly generated samples ${}^{g}x_1, {}^{g}x_2$ follows principle Eq. (11) to redistribute to the final position, and connect to the tree.

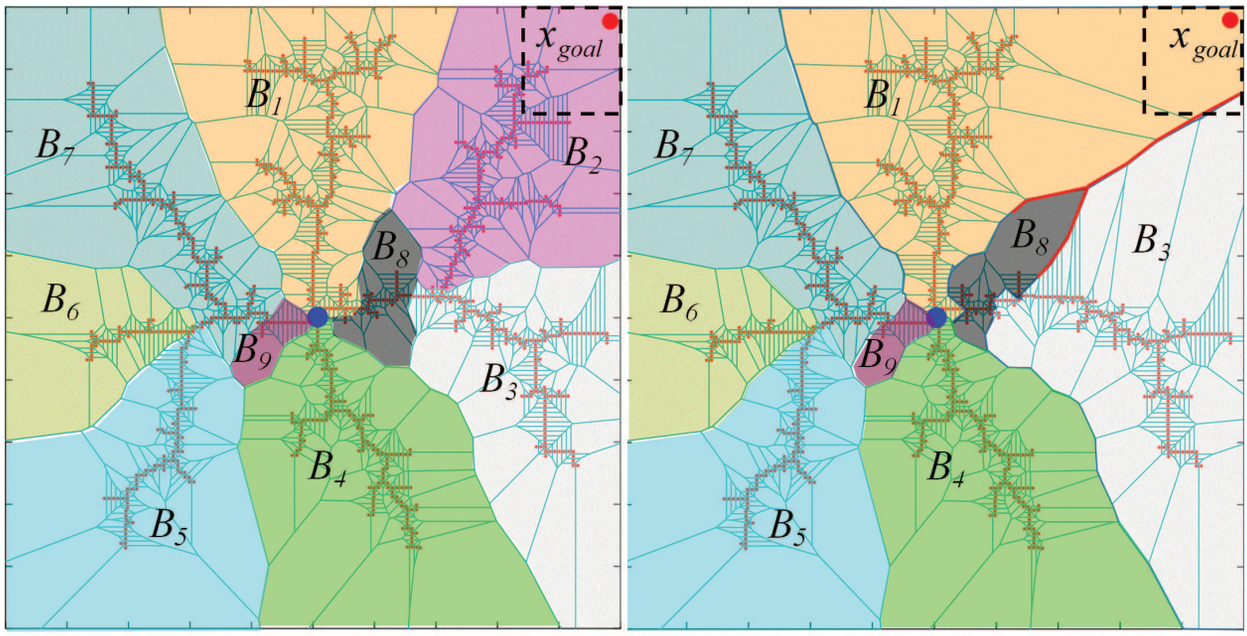
By using the two proposed approaches, we can generate more useful samples for extending, especially, the samples generate around the edge of the obstacles with the ability to perform more active exploration in cluttered environments. Besides, the outer attraction redistributes the samples toward the narrow corridor between the obstacle, which thus increases the probability of finding safe path through such obstacle crowed region.

4.2. Homology biased

We assume any path $h_t(s)$ generated using SBA is consisted by a set of nodes $h_t(s) = \{h_t | h_t(s'), s' \in [0,1]\}$, as it is illustrated in **Figure 5(a)** that exploring tree is consist of the red nodes. Each red node is regarded as distinct with other nodes in the tree, with a distinct dominant region, i.e. Voronoi region. Thus, a path $h_t(s)$, which is consisted a set of states from the initial state to the goal, can be isolated with each other with a distinct region $V(b_T)$ combined by all Voronoi regions of the states.

The region dominant property differs the path with each other, where a SB tree with multiple paths (the path here may not connect to the goal, but they keep distinct with each other from the initial position) can be described by a set of branches $B_T = \{b_T(1), b_T(2), \ldots, b_T(n)\}$. For each branch, it consists of a list of states which connect with each other to form tree structure. In the tree, the end state relies on the parent state for extending as well as trajectory tracking.

http://dx.doi.org/10.5772/intechopen.71410



(a) RRT Voronoi biased regions, each region keep distinct with each other

(b) The Extending-Forbidden method removes branch B_2 to enable exploration of B_1, B_8, B_3 toward the goal

Figure 5. The extending-forbidden algorithm (EFA) tries to find all the states along the goal reached branch at each goal reaching checking step, such as branch B_2. Then, EFA sets the flag of the states to be inactive, switching the extending probability to the nearby branches.

4.2.1. Extending-forbidden algorithm

The path planner performs exploration following the Monte-Carlo approach. Given a configuration space C in a topological space (the rectangle region as illustrated in **Figure 5**), we denote the Lebesgue Measurement of the configuration space as $L^*|C|$. Then we can get the Lebesgue measurement of each branch $b_T(i)$ of the tree using the same metric. Authors in [18] proved that the dispersion of n random nodes sampled uniformly and independently in $V(b_T(i))$ is,

$$D = \sup_{b_T(i)} \left| \frac{\psi(b_t(i) : n)}{n} - \frac{L^*|V(b_T(i))|}{L*|C|} \right| = o\left(\left(\frac{\log(n)}{n} \right)^{\frac{1}{d}} \right) \tag{12}$$

Where $\psi(b_t(i))$ denotes the number of samples m, $1 \le m \le n$, that lies in the sub-branch $b_T(i)$, n is the number of all the sampling, d is the dimension of the configuration space. D denotes the difference between the ration of sampling probability and ration of space Lebesgue measurement, which follows the knowledge that Monte-Carlo method performs a uniform sampling. It means the sampling probability approaches the ratio of Lebesgue measurements, that is, the exploration probability can be represented as:

$$P_{b_T(i)} = \frac{L^*|V(b_T(i))|}{L^*|C|} \tag{13}$$

However, the probability of exploring in the configuration space does not benefit the extending bias toward the goal. Let us still take a look at **Figure 5(a)**, the branch B_2 dominant

the near-goal region, and other region are not able to extend toward the goal as the samples will not connect to the branches if it locates in the near-goal region. To solve this problem, this paper proposes an Extending-Forbidden Algorithm (EFA), it shifts the source for extending to other branches by forbidding the goal reached path.

Definition 3—Goal-biased Probability: Given a configuration space C, the exploring tree T and all its branches which are main branches B_T and its corresponding sub-branches. The goal-biased event denotes a branch can exploring toward the goal. If a goal region can be represented as G_r, and Voronoi(G_r2B_T(i)) is the region that belongs to goal region and the Voronoi region of branch $B_T(i)$. Then, the nominal goal biased probability of branch $B_T(i)$ toward G_r is:

$$P_G^*(B_T(i)) = \frac{L^*|voronoi(G_r2B_T(i))|}{L^*|C|} \tag{14}$$

And the real goal biased probability is normalized value of all branches, that is,

$$P_G(B_T(i)) = \frac{P_G^*(B_T(i))}{\sum_{j=1}^{n} P_G^*(B_T(j))} \tag{15}$$

Definition 4—Long Sub-branch (LSB) and Short Sub-branch (SSB): Given a tree T and all its Voronoi distinct main branches B_T. Then, we can define a length threshold δ_B. For all end vertices in each main branch, we calculate the length l_{sb} from end to the goal reach reached path (any state which firstly reached). If the length $l_{sb} > \delta_B$, then we call it Long Sub-branch (LSB). If $l_{sb} \leq \delta_B$, we call the sub-branch as Short Sub-branch (SSB).

It should be noted that the threshold is very empirical in Definition 4, and it is decided based on the configuration space and the kinematic constraints. In **Figure 6**, we set it as 15 meters, then we have SSB_1, SSB_2, SSB_3 as SSB, and LSB_1 as LSB. The reason why we have this definition is that we cannot shift all the extending resource to the neighbor branches, and we have the hypothesis that the SSB must has lower probability of finding a new path even given the resource for extending. For example, the main Voronoi dominant branches B_T keep distinct with each other, and each main branch has the probability $P_{b_T(i)}$ for exploration in the

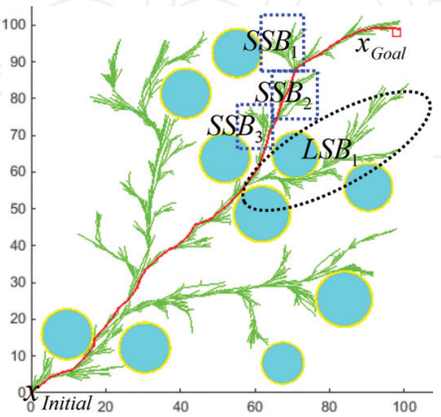


Figure 6. An intuitive example of SSB and LSB in an exploring tree, which are generated using threshold principle as defined in Definition 4.

configuration space. After the tree stops extending at a certain iteration, we can have the results as illustrated in **Figure 5(a)**. Since goal region is in Voronoi region of branch B_2, then we know that we have the goal biased probability as $P_G(B(2)) = 1$. One branch B_2 reached the goal region which is represented with dotted back rectangle, EFA searches the SSBs and LSBs based on Definition 4, and it labels states and executing forbidden. Then, we have a resource shifted Voronoi graph as illustrated in **Figure 5(b)**, where we can see that branch B_1 and B_3 obtains the Voronoi region which belongs to branch B_2. The two branches also obtain the rectangle goal region, that is, their goal biased probabilities are bigger than zero, $P_G(B(1)) > 0, P_G(B(3)) > 0$.

Since EFA can shift the goal biased by extending resource to other branches, while not all paths can obtain such resource. The following truths are hold:

1. The increasing of goal biased probability ensures the generation of a feasible path toward the goal, but not all branches with goal biased probability can reach the goal at the same time. Only one can reach the goal because of Voronoi dominant probability, thus the general SBA cannot find multiple paths.

2. The efficiency of generating multiple paths mainly depends on the environment adaptability of random exploring algorithms. For random exploring algorithm, their merits of generating multiple branches enable the generation of multiple paths.

4.2.2. Reasonable alternative reference chosen

The proposed MGART is able to perform extending-forbidden toward multiple paths, as the random exploring property guarantees completeness and diversity. However, the quality of explored paths cannot be guaranteed, particularly a large number of homotopic paths are generated. This paper proposes an approach to generate the reasonable path, and we analysis under the hypothesis that the environment is highly cluttered and it is not practical to set threshold for path planner to choose the best homological paths.

Definition 5—Reasonable Alternative Paths: Consider two homotopic paths $h(\pi_1)$ and $h(\pi_2)$ in a configuration space C. The surrounding obstacle information along each path are $\beth(h(\pi_1))$ and $\beth(h(\pi_2))$. The reasonable alternative path exists if and only the surrounding information of the two paths are not the same, such that, $\beth(h(\pi_1)) \neq \beth(h(\pi_2))$.

Given the sensing range of a robot as Υ, and the obstacles set $O = \{o_1, o_2, \ldots, o_n\}$. For path $h(\pi_1)$, it consists of a set of discrete states $X(\pi_1) = \{x(\pi_1)_1, x(\pi_1)_2, \ldots, x(\pi_1)_n\}$. In this paper, we assume that any obstacle o_i can be described with a circle or ellipse centered at o_i^c, then we can build a Delaunay Triangulation [19] connection (DTc) using the obstacle centers, the initial state, and the goal state. For DTc, it can generate a network like structure, and each two states have at most one connection. It is illustrated in **Figure 8(a)** that the green edges are the valid connections, with labels to distinct with each other. For any path, if the path intersects with an edge, the edge information should be added to the information factor, such as the solid red path intersects with edge L7, then $L7 \in \beth(h(\pi_1))$. The edge labeling method can guarantee the uniqueness toward homology, while we note that homotopic paths can also be used to perform emergency threat management. It is represented in **Figure 7(b)**, the solid red path h_s and the

dotted red path h_D are homotopic to each other. Given the sensing range Υ, we have the sensing envelop which are dotted purple lines h_1^L, h_1^R for h_D and solid black lines h_2^L, h_2^R for h_s, indicating the maximum detection range for emergency threat. Then, we have the $\{o_1\} \subset \beth(h_D)$ and $\{o_1, o_2, o_3, o_4\} \subset \beth(h_S)$, thus we have h_s and h_D both regarded as reasonable alternative paths for online threat management.

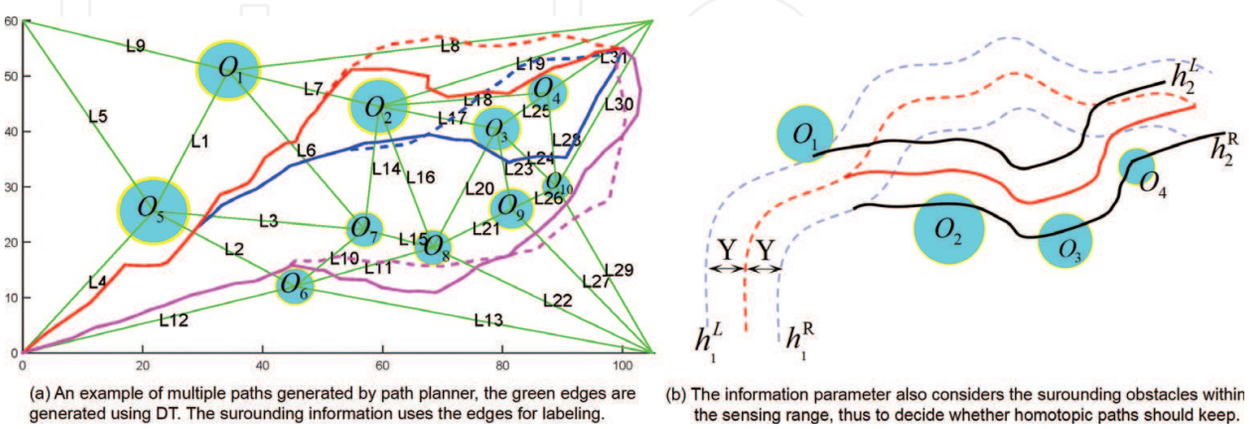
The informative approach discussed thus can help to label each path in a configuration space, such as the results listed in **Table 1** of paths in **Figure 7(a)**. Then according to Definition 5, we can find the label of each path. For any several paths which have the same label, we choose the shortest path and use as the candidate for online fast switching.

4.3. Emergency threat management

The reasonable alternative path set H_{RAP} provides a network with cluttered environment adaptivity. The concept of visibility was discussed in [20], where the cycle information is used to enable fast deformation for motion planning. Visibility is defined as:

Path	**Path surrounding information**	
	Edge information	**Obstacle information**
Dotted red	L2.L3.L6.L7.L8.L19.L31	O_5, O_2
Solid red	L2.L3.L6.L7.L8.L19.L31	O_5, O_2, O_3, O_4
Dotted blue	L2.L3.L6.L14.L16.L17.L18.L31	O_5, O_2, O_3, O_4
Solid blue	L2.L3.L6.L14.L16.L20.L23.L24	O_5, O_2, O_3
Dotted pink	L2.L10.L11.L22.L27.L29.L30	O_6, O_7, O_8
Solid pink	L2.L10.L11.L22.L27.L29.L30	O_6, O_9, O_{10}, O_3

Table 1. The path information parameter is consist of two parts, which are edge information using DTc and obstacle information using sensing envelop, respectively.



(a) An example of multiple paths generated by path planner, the green edges are generated using DT. The surrounding information uses the edges for labeling.

(b) The information parameter also considers the surounding obstacles within the sensing range, thus to decide whether homotopic paths should keep.

Figure 7. An illustration of surrounding information used to find reasonable alternative path for online emergency threat management. The information parameter consists of edge information and obstacle surrounding information within sensing envelop.

http://dx.doi.org/10.5772/intechopen.71410

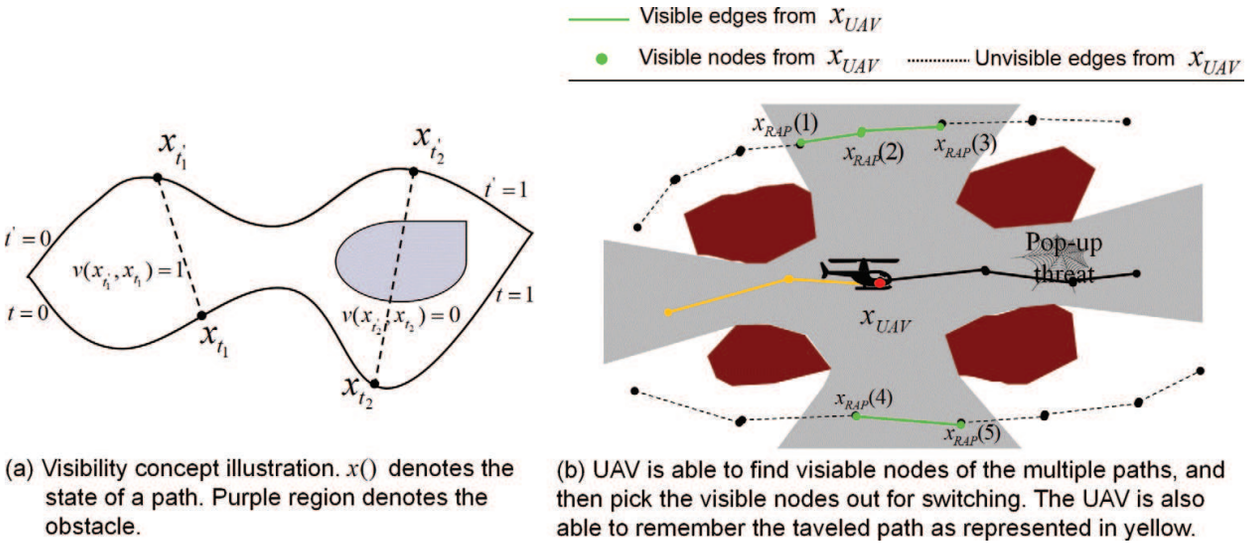
$$\mathrm{V}: \begin{cases} [0,1]\times[0,1]\rightarrow\{0,1\} \\ V(x_t, x_{t'}) = \quad 1, if\ \mathcal{L}_{link}(x_t, x_{t'}) \in C_{free} \subset C \\ V(x_t, x_{t'}) = \quad 0, if\ \mathcal{L}_{link}(x_t, x_{t'}) \in C_{obs} \subset C \end{cases} \tag{16}$$

Where $x_{()}$ denotes a state of a path, $\mathcal{L}_{link}$ is the connection of two states, C_{free} is the free space and C_{obs} is the obstacle region. A visual illustration is provided in **Figure 8(a)**, where visibility can only in obstacle free region.

It is noted that the visibility in this paper means a possible connection to switch from one path to another for emergency threat management. For switching with visibility, given all the reasonable alternative paths H_{RAP}, the algorithm performs exploration for visibility state at each UAV state x_{UAV} among H_{RAP}. The algorithm then outputs the visible guards (states) x_{RAP} as illustrated in **Figure 8(b)**. To avoid the pop-up threat (or dynamic threat), UAV must select one entry guard from the visible guard set to reconnect to another pre-planned path to the goal. To validate the best connection, that is, the entry point and the entry connection, this paper applies the heuristic:

$$x^* = \underset{i,j}{\operatorname{argmin}}\ C_{FE}\left(x_{UAV}, H_{RAP(i)}\left(x_{RAP(j)}\right)\right) + C_{TC}\left(H_{RAP(i)}\left(x_{RAP(j)}\right)\right) \tag{17}$$

Using a simple cost based metric, where C_{FE} is the forward energy cost which is the distance and the turning cost from UAV position x_{UAV} to entry state $x_{RAP(j)}$ and the path from entry state to the goal $H_{RAP(i)}(x_{RAP(j)})$. The turning cost is the integration of heading angle difference at each state, which denotes the smoothness of the planned path. C_{TC} is the threat cost, the integration of inverse distance between state and obstacles. Using such approach, the algorithm can find five visible states $x_{RAP} = \{x_{RAP}(1), \ldots, x_{RAP}(5)\}$ as illustrated in **Figure 8(b)**. Then, it tries to find the best entry state using the minimum cost principle (Eq. (17)).



(a) Visibility concept illustration. $x()$ denotes the state of a path. Purple region denotes the obstacle.

(b) UAV is able to find visiable nodes of the multiple paths, and then pick the visible nodes out for switching. The UAV is also able to remember the taveled path as represented in yellow.

Figure 8. For online switching, UAV should follow the visible-node selection algorithm (a) to explore the possible switching route, then it switches to the cost minimum path for ETM (b).

We further consider a situation that there may have no visible states at current location. The paper proposes to use a long-term memory approach to handle this problem, that is, the travel path should be stored in memory, such as the orange edges and states illustrated in **Figure 8(b)**. In the meanwhile, the method stores the visible state along the traveled path. Then, the UAV has to fly back to find a cost minimum path toward the goal if it confronts with pop-threat and has no visible states.

5. Experiment and discussion

In this section, we highlight the performance of the obstacle biased and homology biased path planner with the ability of emergency threat management (avoiding pop-up and dynamic threat online). In the section, we will discuss the following points: (1) How the threshold of EFA affects the performance of MGART. (2) The time performance and reliability of reasonable alternative chosen algorithm. (3) The online emergency threat management performance. The algorithm is implemented using MatLab 2016b on a laptop computer with a 2.6 GHz Intel Core I5 processor.

5.1. Comparative simulation of multiple path exploration

We design three different scenarios, which are non-obstacle scenario, rounded obstacle crowed scenario, and irregular polygons crowed scenario, to perform comparative simulations. All the scenarios are 2D with $100^{*}65$ m^2 space, and obstacles randomly generated.

For scenario 1, it is a non-obstacle environment, and we set the variable threshold as a set with value {3, 5, 7, 8, 11,13,15,18,20,25,28} for representation. As we know the length of EFA threshold affects the goal biased probability, which directly decide the area of the newly obtained Voronoi region of the neighbor branches, we design a set of comparative experiments to study the effects between EF length and RAPs. An intuitive result of the relationship between planned paths and EF length after 10,000 iterations are provided in **Figure 9(a)–(d)**. MGART can find 37 paths after 10,000 iterations if EF length is set as 3 step-length, and the number decreases to 22 if the EF length is set as 28. The reason is that the longer the EF length, the further the neighbor branches can obtain the goal biased resource. Thus, the neighbor branches need more steps to exploring toward the goal, that is, less paths will be achieved with better homology performance. As we can see that the paths in **Figure 9(d)** have a better homology performance than **Figure 9(a)**. The same EF length variation experiment is also deployed in scenario 2, and results are shown in **Figure 9(e)–(h)**. For RAPs, it is the same with the results in scene 1 that RAPs decrease with the increasing of the EF length. However, as the increasing of EF length enables more branches to explore toward the goal as well as increasing the homologous paths (see in **Figure 9(e)–(h)**), the number of the RAPs increasing with the increasing of the EF length. The statistic relation between the RAPs and the EF length is illustrated in **Figure 10**, which further proves the conclusion.

The EFA can be used to any SBAs by shifting the goal biased resource to achieve multiple RAPs for online switching. This paper compares the performance between MGART and MRRT* in three scenarios with 10,000 iterations. We compare the efficiency of generating a path, RAPs,

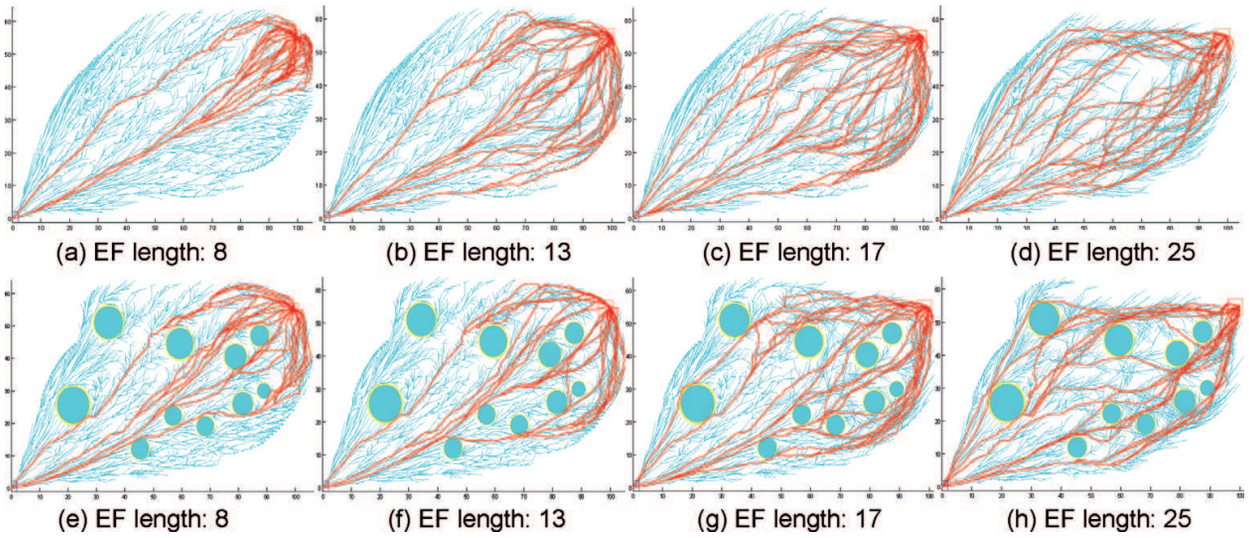


Figure 9. Illustration of alternative paths generated by MGART vary with representative backward EF length. (a)–(d) denotes the results in non-obstacles scenario, (e)–(h) denotes the results in obstacle crowed.

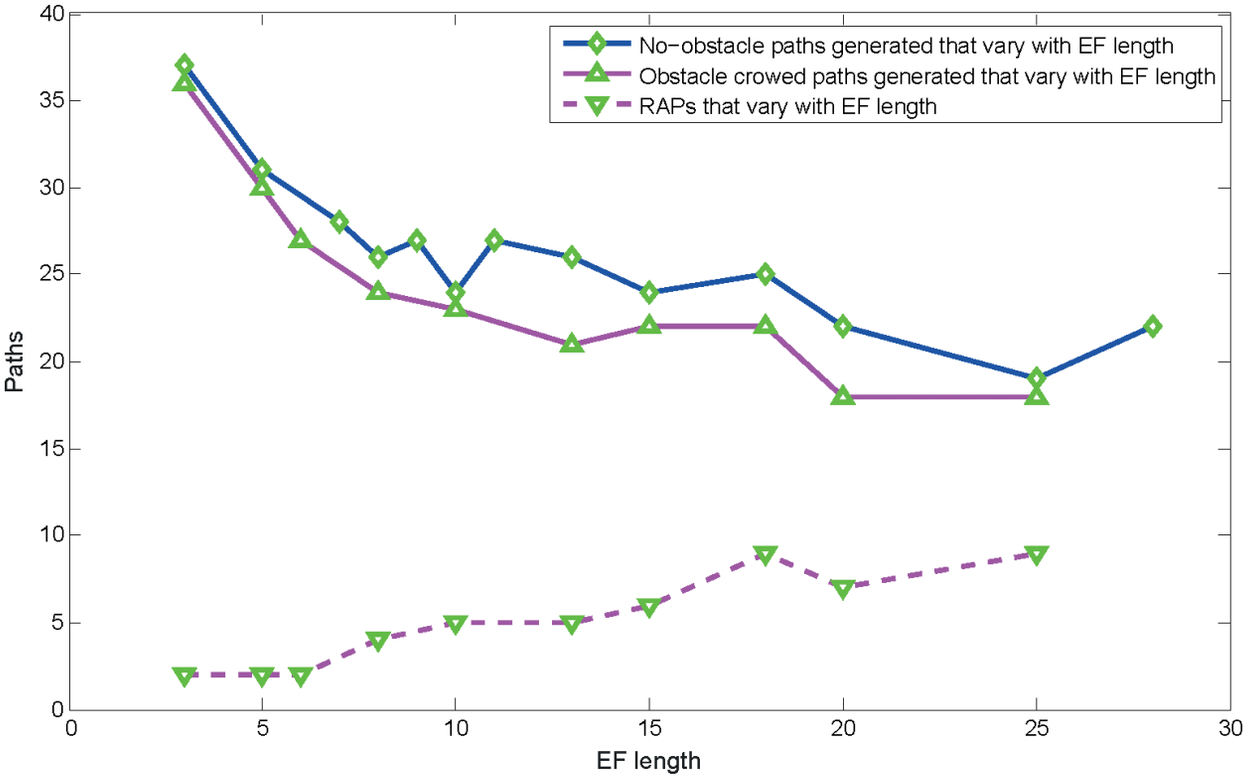


Figure 10. Relation between EF backward length and APs and RAPs with two scenarios. Here the solid diamond line denotes the relation of scene 1, and the triangle lines denotes the results of scene 2.

average time for finding a path, and average time for any RAP are compared in **Table 2**. GART has a better performance in both path exploration and RAP generation, such that MGART can find at least 3 times of the number of paths toward the goal than MRRT*. Because GART introduces the environmental information to speed up the exploring process, the results prove

that MGART is more efficient in finding RAPs, which is almost 100% faster than MRRT*. For time performance, we can see that MGART also outperforms MRRT* with at least 3 times advantage.

Besides comparison of the time performance of finding online switching paths (that is RAPs), we also pay attention to the quality of the path generated. The average lengths and standard deviation of the length of all paths in each scene are illustrated in **Figure 11**. The average length of the paths that generated by MGART and MRRT* are illustrated in **Figure 11(a)**, we can see that MGART has a strong convergence performance than MRRT*. The standard deviation of the lengths is shown in **Figure 11(b)**, results demonstrate that MGART is more likely to find paths with smaller fluctuation as well as smaller cost.

5.2. Performance of reasonable alternative path chosen

We also test the path labeling algorithm, that is, the surrounding information pursuing using DTc and sensing envelop, which is used to obtain the reasonable alternative paths under Definition 5. It is should be noted that the under the definition, any two paths do not have the same information parameter, which enables fast switching when facing pop-up threat. As the path label method guarantees the unique labeling of all the paths, only the paths which stretch in a parallel way and within the same sensing envelop have the same labels.

The results of simulation after 10,000 iterations in scenario 2 and 3 are provided in **Table 3**. For each single path, the time needed for labeling the path mainly depends on the area, dimension,

Scenario	Algorithm	Paths after 10,000 iterations	RAPs after 10,000 iterations	Average time for a feasible path	Average time for a RAP
Scenario 1	MGART	48	/	2.081	/
	MMRRT*	12	/	7.825	/
Scenario 2	MGART	43	9	2.093	8.213
	MRRT*	11	5	7.169	15.772
Scenario 3	MGART	34	10	2.257	7.674
	MRRT*	11	5	6.789	14.935

Table 2. Detailed comparison of planning efficiency between MH-GART and MH-RRT* in all three scenarios.

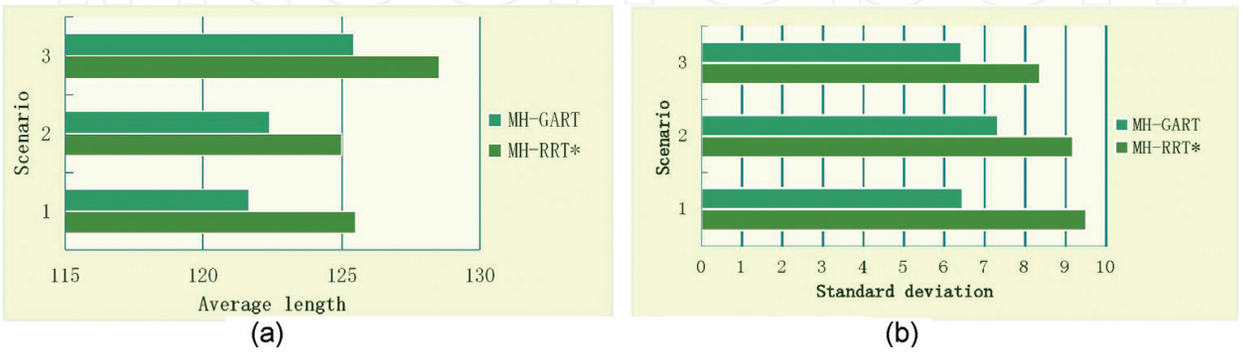


Figure 11. Comparison of (a) average length and (b) standard deviation of the APs generated by MGART and M-RRT* in three scenarios.

the complexity of the configuration space. For our tested with area 100*60, the average time for acquiring the information for labeling 0.078 s (see in **Table 3**). The average time needs for RAP pursing of our cases is 0.139 s.

5.3. Experiments of emergency threat management

MGART can be used for 3D and 2D pop-up threat management, and the 3D environments can be easily segmented by DT. We evaluate the performance of our method in both 2D and 3D environments, and we also compared the time performance.

For 2D environments, we implement three tests with different number of dynamic threats. The RAP chosen algorithm works when robot realizes that the path will collide with the pop-up threat, that is, robot at position x_{UAV} detects the moving threat (see in **Figure 12(a)**). The simulation setting is illustrated in **Table 4**, where the robot speed is 10 m/s and the moving threat can be detected within 10 m detection range. Thus, the robot has less than 1 s to re-plan a path and executing to avoid the obstacle. RAP chosen algorithm first evaluates all its neighbor RAPs (the green lines) around the robot, and chooses the cost minimal and collision free path based on principle Eq. (17) (the dotted green path in **Figure 12(b)**). It is noted that **Figure 12(b)–(d)** are results of using MGART to avoid one, two, and three moving threats, respectively. The black parts along the navigation path denote the position where threat is detected by robot. We also execute test in 3D environment (see in **Figure 13**) with pup-up and moving threats. The on-line switching is supposed to be used for aerial robots in 3D, thus Dublin's Curves is used when switching from current position to safe path.

For all the experiments, we study the time efficiency of each switching to escape from current dangerous situation. For one moving threat avoiding (see in **Figure 12(b)**), the time needed to switch to other RAP is 0.0507 s, and the whole navigation duration is 13.14 s with 10 m/s

Scenario 2		**Scenario 3**	
Time for labeling (s)	**Time for RAPs (s)**	**Time for labeling (s)**	**Time for RAPs (s)**
0.0721	0.146	0.0842	0.132

Table 3. Time performance of proposed method in two scenarios.

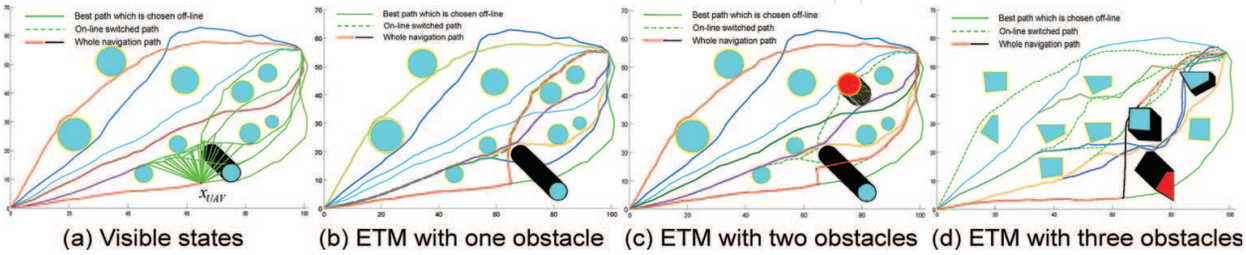


Figure 12. Tests of on-line switching to avoid dynamic threats using MGART in 2D scenarios. (a) Robot detects moving threat at position x_{UAV}, then it evaluates all its visible neighbor RAPs (the green lines) to choose the switching path. (b) Complete navigation of avoiding one moving threat, the red path is the navigation path. (c) Test of avoiding two moving threats, the black and red circles are threats. (d) Tests of avoiding three moving threats.

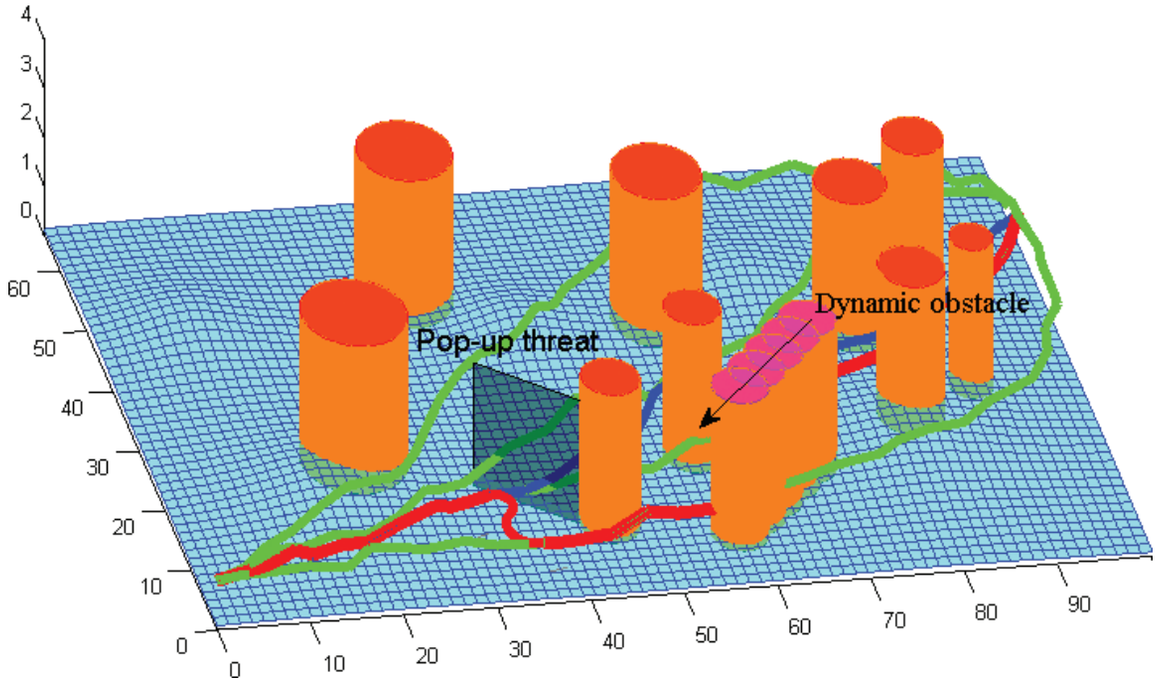


Figure 13. The experiment of using MGAT for 3D emergency threat management case, where pop-up threat and moving threat are exist in the environment.

Robot speed	10 m/s
Detection range	10 m
Speed of cyan obstacle (in **Figure 12(b)–(c)**)	1.4 m/s
Speed of red obstacles (in **Figure 12(c)–(d)**)	0.8 m/s
Speed of cyan obstacle (in **Figure 12(d)**)	0.8 m/s
Online switching range	20 m

Table 4. Setting of simulation for ETM.

speed. For two dynamic threats avoiding case (see in **Figure 12(c)**), the whole navigation time is 13.32 s, and the time spend to avoid the second threat is 0.0912 s. In scene 3, we designed a long duration for threat (see in **Figure 12(d)**). The two cyan threats disable the blue-path, thus robot has to switch for more times while tracking the dark path. The average time is no more than 0.15 s which can be decreased when implemented in robot's platform with C++ implementation, and the whole navigation time is 13.5 s.

6. Conclusion

The main contribution of this paper is that an online EMT planner is proposed, where pop-up threat and moving obstacle happen during tracking the pre-planned path. We propose a new multiple path planning approach called MGART, which is improved based on GART, by introducing an 'Extending Forbidden' algorithm to shift the goal biased probability to neighbor branches around goal reached branch. The algorithm is shown to inherit the merits of

GART and the ability of exploring in cluttered environments, and it guarantees asymptotically optimal and completeness. It is also shown that the algorithm can generate multiple paths without using variant cost principles, but only relying on the EFA threshold, thus it enables selection for online dynamical switching.

In the future, we would like to research on online visual positioning and environment perception topic, which is lack of discussion in this paper. We would like to enable cognitive sensing and autonomous for robots.

Acknowledgements

This work was supported by NSFC under grant No. 61528303.

Author details

Liang Yang[1,2,3], Yuqing He[1,3], Jizhong Xiao[1,2,3]*, Bing Li[2] and Zhaoming Liu[1,3]

*Address all correspondence to: jxiao@ccny.cuny.edu

1 State Key Laboratory of Robotics, Chinese Academy of Sciences, Shenyang, China

2 The City College/City University of New York, New York, USA

3 University of Chinese Academy of Sciences, Beijing, China

References

[1] Goerzen C, Kong Z, Mettler B. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. Journal of Intelligent and Robotic Systems. 2010; **57**(1–4):65

[2] Yang L, Qi J, Song D, Xiao J, Han J, Xia Y. Survey of robot 3D path planning algorithms. Journal of Control Science and Engineering. 2016;**5**(2016):1-22

[3] Chen H, Chang K, Agate CS. UAV path planning with tangent-plus-Lyapunov vector field guidance and obstacle avoidance. IEEE Transactions on Aerospace and Electronic Systems. 2013;**49**(2):840-856

[4] Davoodi M, Panahi F, Mohades A, Hashemi SN. Multi-objective path planning in discrete space. Applied Soft Computing. 2013;**13**(1):709-720

[5] Rathbun D, Kragelund S, Pongpunwattana A, Capozzi B. An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments. In: Proceeding of IEEE Digital Avionics Systems Conference. Vol. 2; 2002. pp. 8D2-1-8D2-12

[6] Valenti RG, Jian Y-D, Ni K, Xiao J. An autonomous flyer photographer. In: Proc. of 2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER); 2016. pp. 273-278

[7] Coleman D, Sucan I, Chitta S, Correll N. Reducing the barrier to entry of complex robotic software: A moveit! case study. arXiv preprint. 2014;**1404**(3785):1-14

[8] Yang K, Gan SK, Sukkarieh S. A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV. Advanced Robotics. 2013;**27**(6): 431-443

[9] Yoshida E, Kanehiro F. Reactive robot motion using path replanning and deformation. In: IEEE International Conference on Robotics and Automation (ICRA); 2011. pp. 5456-5462

[10] Lindemann SR, LaValle SM. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In: IEEE International Conference on Robotics and Automation. Vol. 4; 2004. pp. 3251-3257

[11] Kavraki L, Latombe J-C. Randomized preprocessing of configuration for fast path planning. In: Proc. of IEEE International Conference on Robotics and Automation; 1994 May 8. pp. 2138-2145

[12] LaValle SM, Kuffner JJ Jr. Randomized kinodynamic planning. The International Journal of Robotics Research. 2001;**20**(5):378-400

[13] Yang L, Xiao J, Qi J, Yang L, Wang L, Han J. GART: An environment-guided path planner for robots in crowded environments under kinodynamic constraints. International Journal of Advanced Robotic Systems. 2016;**13**(6):1-18

[14] Yang L, Song D, Xiao J, Han J, Yang L, Cao Y. Generation of dynamically feasible and collision free trajectory by applying six-order Bezier curve and local optimal reshaping. In: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2015 Sep 28. pp. 643-648

[15] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research. 2011;**30**(7):846-894

[16] Urmson C, Simmons R. Approaches for heuristically biasing RRT growth. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 2; 2003, October. pp. 1178-1183

[17] Pan J, Manocha D. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. The International Journal of Robotics Research. 2016;**35**(12):1477-1496

[18] Yang L, Qi J, Jiang Z, Song D, Han J, Xiao J. Guiding attraction based random tree path planning under uncertainty: Dedicate for UAV. In: IEEE International Conference on Mechatronics and Automation (ICMA); 2014 Aug 3. pp. 1182-1187

[19] Levcopoulos C, Krznaric D. Quasi-greedy triangulations approximating the minimum weight triangulation. Journal of Algorithms. 1998;**27**(2):303-338

[20] Jaillet L, Siméon T. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. The International Journal of Robotics Research. 2008;**27**(11–12):1175-1188