Survey of Robot 3D Path Planning Algorithms

Article *in* Journal of Control Science and Engineering · January 2016

DOI: 10.1155/2016/7426913

Liang Yang
City College of New York
13 PUBLICATIONS **57** CITATIONS
SEE PROFILE

Juntong Qi
Chinese Academy of Sciences
57 PUBLICATIONS **422** CITATIONS
SEE PROFILE

Dalei Song
Chinese Academy of Sciences
28 PUBLICATIONS **260** CITATIONS
SEE PROFILE

Jizhong Xiao
City College of New York
152 PUBLICATIONS **1,593** CITATIONS
SEE PROFILE

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2016, Article ID 7426913, 22 pages
http://dx.doi.org/10.1155/2016/7426913

Review Article

Survey of Robot 3D Path Planning Algorithms

Liang Yang,[1,2] Juntong Qi,[1] Dalei Song,[1] Jizhong Xiao,[3] Jianda Han,[1] and Yong Xia[4]

[1] *Shenyang Institute of Automation, Nanta 114th Street, Shenhe District, Shenyang 10016, China*
[2] *University of Chinese Academy of Sciences, 19 Yuquan Road, Beijing 100049, China*
[3] *The City College, City University of New York, Convent Avenue at 140th Street, New York, NY, USA*
[4] *State Grid Liaoning Electric Power Company, Benxi, Liaoning 117000, China*

Correspondence should be addressed to Jizhong Xiao; jxiao@ccny.cuny.edu

Received 25 November 2015; Accepted 3 April 2016

Academic Editor: Petko Petkov

Robot 3D (three-dimension) path planning targets for finding an optimal and collision-free path in a 3D workspace while taking into account kinematic constraints (including geometric, physical, and temporal constraints). The purpose of path planning, unlike motion planning which must be taken into consideration of dynamics, is to find a kinematically optimal path with the least time as well as model the environment completely. We discuss the fundamentals of these most successful robot 3D path planning algorithms which have been developed in recent years and concentrate on universally applicable algorithms which can be implemented in aerial robots, ground robots, and underwater robots. This paper classifies all the methods into five categories based on their exploring mechanisms and proposes a category, called multifusion based algorithms. For all these algorithms, they are analyzed from a time efficiency and implementable area perspective. Furthermore a comprehensive applicable analysis for each kind of method is presented after considering their merits and weaknesses.

1. Introduction

Advances in commercial grade technology and advanced research make it possible for robots to appear in everyday life. Modern robots of different varieties include industrial robots, service robots, have seen bright future. For robots, such as Google Self-Driving Car [1] and iRobot Vacuum Cleaning robot [2], one of the most basic and important abilities is path planning, that is, autonomous routing. This requires both time efficiency to react to any emergency situation and safety consideration for implementation.

Path planning targets for moving robots from their initial locations to the goal location by their own actuators and strategies, and during the process, robots must always be able to avoid obstacles to maintain safety. Robots such as underwater robots [3, 4], wall-climbing robot [5], and micro air vehicles [6–8] have already been tested purposely with different kinds of methods. These methods can be basically perceived from [9–12], where algorithms were already detailed synthesized. These works contributes a lot; however they are analyzed in a general view without comparison and analyzed in specific perspectives as well as covering the latest works. Thus, we need to provide a comprehensive analysis on 3D path planning methods.

By reviewing the latest works, it is not hard to see that most works concentrate only on two dimensions (2D), thus limiting the behaviors of the robots only to surface or each iteration considering the height as a constant to achieve a 2.5-dimensional (2.5D) method. Choset [9] summarized a lot of his own and others' works; however he mainly concentrates on 2D environment and therefore algorithms like bioinspired algorithms were paid no attention. LaValle et al. [10] focused mainly on sampling based algorithms. None of these researchers had analyzed all the algorithms in 3D planning area. Surveys such as [13] almost all analyze the path planning problems under 2D condition. Reference [14] only focuses on micro air vehicle's path planning problems, and algorithms like mathematic optimal algorithms were ignored. Lately, authors in [13] did a thorough survey on coverage path planning, which is classified by distinguishing decomposition methods. They provided a basic understanding of 3D path planning methods and then narrowed their discussion

(a) Forest [15] (b) Urban [16] (c) Underwater [17]

Figure 1: Examples of 3D complex environments.

to sampling based algorithm to explain the 3D planning idea.

Facing the more and more challenging environment that robots faced, where environments tend to be unstructured and full of uncertain factors, 3D path planning algorithms are urgently needed nowadays. Several typical kinds of 3D environments are presented in Figure 1, including forest, urban, and underwater environments. When planning in these complex situations, a simple 2D algorithm will not be qualified; thus 3D path planning algorithms are needed.

Path planning in 3D environment shows great prospect, but unlike 2D path planning, the difficulties increase exponentially with kinematic constraints. In order to plan a collision-free path through the cluster environment, the problem of how to model the environment while taking the kinematic constraints into consideration needs to be solved. From the optimization point of view, finding a 3D optimal path planning problem is NP-hard; thus there exist no common solutions.

3D path planning algorithms include visibility graph [18] which works by connecting visible vertexes of polyhedron, random-exploring algorithms such as rapidly exploring random tree [19], Probabilistic Road Map [20], optimal search algorithms (such as Dijkstra's algorithm [21], A* [22], and D* [23]), and bioinspired planning algorithms. What must be emphasized is that this paper only pays attention to broadly applicable methods proposed. Algorithms such as manifold based algorithms [24], which generate smooth path, have been ignored due to reason that it is only applicable to rigid body robots without aerodynamical or hydromechanical influences. Synthesizing all these methods, a two-step procedure of 3D path planning is summarized as follows.

Step 1. The first step is environment perception and modeling, usually using a grid map [25] (with occupancy probability), or a convex or nonconvex region in combination of polynomial form [26].

Step 2. Then path planning algorithm is employed to find the best path according to the cost function, with the ability to achieve both time efficiency and cost minimum.

Steps above are not absolute procedures we must obey, and Step 1 can be divided into two parts, or Step 2 integrates with Step 1. According to Step 2, this paper discusses the efficiency of each algorithm and particularly concentrates on time complexity and applicability. Path planning in 3D environments may face much more uncertainties; thus all this should be taken into consideration to achieve the best path. Local minimal and global optimal are two contradictive points; this paper will analyze these two properties of the algorithms. This paper mainly answered several important questions: (a) what is the taxonomy of these 3D path planning algorithms? (b) How do these algorithms work in order to find the path? (c) Why is it suitable or not suitable for such environment?

This paper is an extension of [27], with more comprehensive explanation on each kind of algorithms.

The main contributions of this paper are listed below.

(1) The first contribution is proposing a taxonomy method for 3D path planning algorithms and puts forward a new category called multifusion based algorithms.

(2) The second contribution of this paper is providing a comprehensive analysis of 3D path planning algorithm which contains almost all of the current most successful methods.

The following sections are arranged as follows. Section 2 discusses some controversial points which need to be declared and gives definitions to these problems for further discussion. Section 3 explains the taxonomy of 3D path planning algorithms and gives a detailed analysis of the taxonomy's reason and also lists elements of each category. This section includes the basic concept of each kind of algorithms. A series of sampling based algorithms are discussed particularly, and each element is put forward by comparison in Section 4. Node based optimal algorithms' common properties are analyzed in Section 5, and this kind of algorithms shares the same merits. In Section 6, this paper discusses a special kind of planning algorithms, that is, mathematic model based algorithms, and confirms it to be path planning algorithm. Section 7 concentrates on the working mechanism of bioinspired algorithms, and three typical algorithms are analyzed elaborately. Multifusion based algorithms are discussed in Section 8, which are commonly ignored. Some typical researches are listed to prove this category. The last section outlines a conclusion and some directions for further research.

2. Preliminary Materials

This section discusses some controversial or ignored points which needed to be declared for further discussion in the following sections, namely, the difference between path planning and optimal path planning and definition of path planning and trajectory planning.

2.1. Problem Statement. Robots have the ability to operate without (or just a little guiding from) human. Ranging from

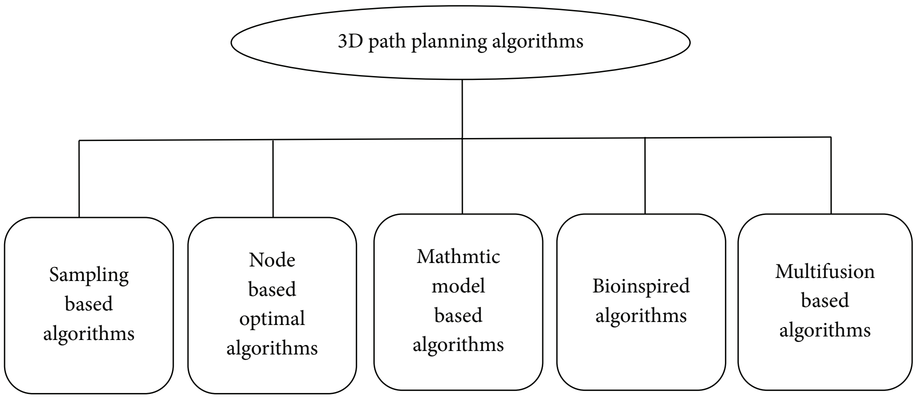

3D path planning algorithms
Sampling based algorithms
Node based optimal algorithms
Mathmtic model based algorithms
Bioinspired algorithms
Multifusion based algorithms

FIGURE 2: 3D path planning taxonomy.

underwater robots to aerial robots, when facing outdoor or indoor complex situations, they need a path planner to determine their next step movement. For path planning, the definition varies according to [8–10, 57]; this paper presents a more canonical definition based on [57].

Robots are assumed to operate in a three-dimension (R^3) space, sometimes called the workspace w. This workspace will often contain obstacles; let wo_i be the ith obstacle. The free workspace without threat of obstacles is the set of points $w_{\text{free}} = w \cup_i wo_i$; it is the space where the robots should always stay. For robots the initial point x_{init} is an element of w_{free}, and the goal region x_{goal} is also an element of w_{free}. Thus path planning problem is defined by a triplet $(x_{\text{init}}, x_{\text{goal}}, w_{\text{free}})$, and the following definitions are given.

Definition 1 (path planning). Given a function $\delta : [0, T] \rightarrow R^3$ of bounded variation, where $\delta(0) = x_{\text{init}}$ and $\delta(T) = x_{\text{goal}}$, if there exists a process Φ that can achieve $\delta(\tau) \in w_{\text{free}}$, for all $\tau \in [0, T]$, the process Φ must be a continuous process without break; then Φ is called path planning.

Definition 2 (optimal path planning). Given a path planning problem $(x_{\text{init}}, x_{\text{goal}}, w_{\text{free}})$ and a cost function $C : \sum \rightarrow R \geq 0$ ($\sum$ denote the set of all paths), if Definition 1 is fulfilled to find a path δ' and $C(\delta') = \min(C(\delta)$, δ is the set of all feasible paths), then δ' is the optimal path and Φ' is optimal path planning.

2.2. Path Planning and Trajectory Planning. Path planning and trajectory planning problems are two distinct parts of robotics, but they are intimately related. There exist several works [8, 9, 11] concerned about this problem. Synthesizing all the corresponding knowledge together, this paper supports the following definitions for further discussion.

Definition 3 (path planning (augmentation)). Find a continuous curve (no need to be smooth) in the configuration space that begins from the start node x_{init} to the goal end node x_{goal}, and the curve should obey the criteria: (a) being without time continuous consideration, (b) including stops in defined position, and (c) being made up of a number of segments, and each can be a trajectory.

Definition 4 (trajectory planning). Trajectory planning usually refers to the problem of taking the solution from a robot path planning algorithm and determining how to move along the path. Trajectory is a set of states that are associated with time; it can be described mathematically as a polynomial $X(t)$, and velocities and accelerations can be computed by taking derivatives with respect to time. It considers the kinodynamic constraints, which can be an element of path.

3. 3D Path Planning Algorithm Taxonomy

Algorithms of 3D path planning have been arising since last century; methods have different characteristics and can be applied to different robots and environments. For researchers and engineers, being stunned to swim in the algorithm sea is a common scene to start in this field. This paper reviews almost all the representative works and books and sorts out almost all the successful 3D path planning methods such as rapidly exploring random trees (RRT), Probabilistic Road Maps (PRM), Artificial Potential Field [86], and Mixed-Integer Programming [87]. The taxonomy that proposed classifying current approaches of 3D path planning is illustrated in Figure 2.

This paper divides 3D path planning algorithms into five categories; the categories are distinguished from each other by their unique properties, such as sampling based algorithms explored by applying Monte Carlo sampling (or likely approaches) to achieve visible connection. From Sections 4–8 an exhaustive discussion of each category is presented.

3.1. Sampling Based Algorithms: Elements and Analysis. This kind of methods needs some preknown information of the whole workspace, that is, a mathematic representation to describe the workspace. This kind usually samples the environment as a set of nodes, or cells, or in other forms. Then map the environment or just search randomly to achieve a feasible path. The elements of sampling based algorithms are illustrated in Figure 3.

For ETC, it means the improved or similar versions of corresponding algorithms; DDRRT is Dynamic Domain RRT algorithm. There is no doubt that rapidly exploring random

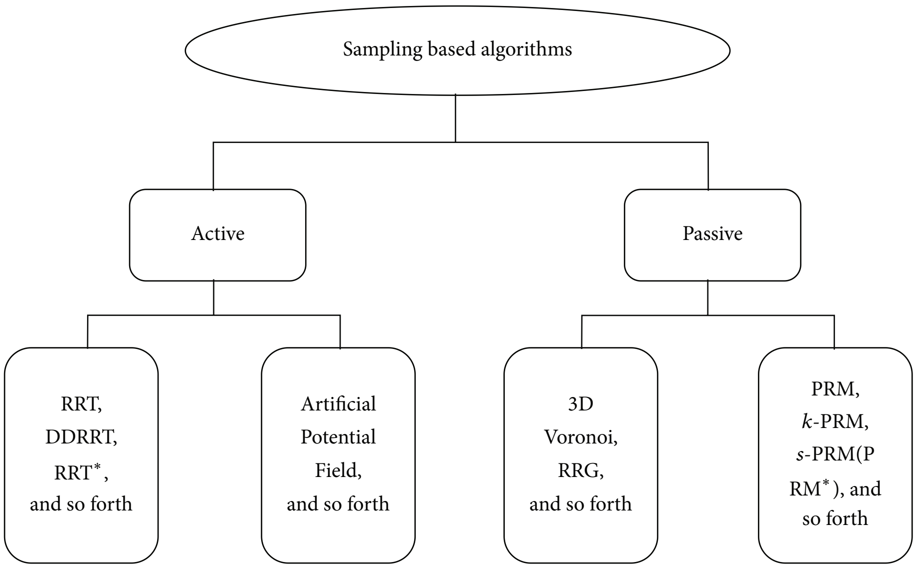

Sampling based algorithms
Active
Passive
RRT, DDRRT, RRT*, and so forth
Artificial Potential Field, and so forth
3D Voronoi, RRG, and so forth
PRM, k-PRM, s-PRM(PRM*), and so forth

FIGURE 3: Elements of sampling based algorithms.

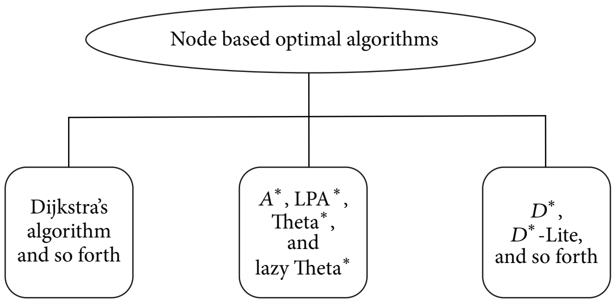

Node based optimal algorithms
Dijkstra's algorithm and so forth
A*, LPA*, Theta*, and lazy Theta*
D*, D*-Lite, and so forth

FIGURE 4: Elements of node based algorithms.

trees (RRT) and Probabilistic Road Maps (PRM) are sampling based algorithms. For 3D Voronoi, it forms a 3D obstacle free network based on preknown knowledge of the whole environment, which contains obstacle and free regions. To achieve this network, Voronoi algorithm still works in an optimization way to follow the varying edges to ensure equal distance [88]. Artificial Potential Field algorithms are sorted as sampling based algorithm due to the fact that it needs the whole workspace sampling information to escape from local minima.

It is shown in the second level of Figure 3 that this paper divides the sampling based algorithms into two more detailed parts: active and passive. Active means algorithm such as rapidly exploring random trees which can achieve the best feasible path to the goal all by its own processing procedure. Passive means algorithms such as Probabilistic Road Maps (PRM) only generate a road net from start to the goal, thus a combination of search algorithms to pick up the best feasible path in the net map where many feasible paths exist. This paper classifies the algorithms, which cannot independently find the best path or any single navigation path, as passive.

3.2. Node Based Optimal Algorithms: Elements and Analysis. Node based optimal algorithms explore through the decomposed graph [25]. Analyzing from the point of the search mechanism, node based optimal algorithms share the same property that they explore among a set of nodes (cell) in the map, where information sensing and processing procedures are already executed. This kind of methods can always find an optimal path according to the certain decomposition. Figure 4 illustrates the typical elements of node based optimal algorithms.

Where LPA* [48] represents Lifelong Planning A* algorithm which is a repeating version of A* (i.e., having the ability to handle dynamic threats), D*-Lite [56] was proposed based on LPA*, which deals with dynamic threat situations. Dijkstra's algorithms, A* and D*, are traditionally classified as discrete optimal planning [10], or road map algorithms [12], or search algorithms [9], and so on. However, they do not collide with each other and just express the same thing where algorithms like this kind deal with discrete optimization based on grid decomposition.

3.3. Mathematic Model Based Algorithms: Elements and Analysis. Mathematic model based algorithms include linear algorithms and optimal control. These methods model the environment (kinematic constraints) as well as the system (dynamic) and then bound the cost function with all the kinematic and dynamic constraints bounds which are inequalities or equations to achieve an optimal solution. The elements of mathematic model based algorithms are presented in Figure 5.

Where flatness based method was first proposed by Chamseddine et al. [63], this method employs differential flatness to ensure control flatness along the reference path; it linearizes the nonlinear kinodynamic constraints to form a rather simple form. MILP is Mixed-Integer Linear Programming [5] which has a strong modeling capability to describe almost all the information. BIP is Binary Linear Programming [62]; it is a special case of linear programming where the variables have only 0-1 integer value.

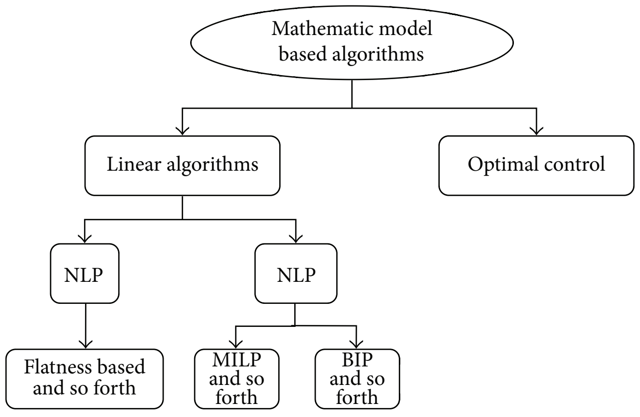


Figure 5: Elements of mathematic model based algorithms.

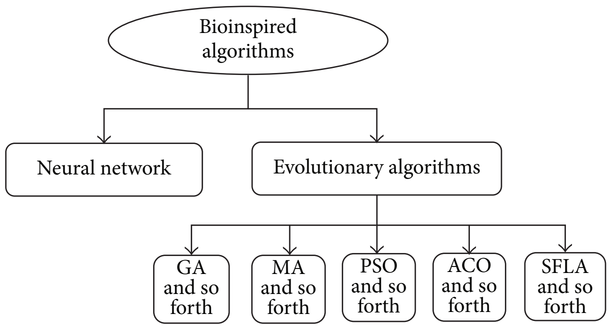


Figure 6: Elements of bioinspired algorithms.

3.4. Bioinspired Algorithms: Elements and Analysis. Bioinspired algorithms originate from mimicking biological behavior to solve problems. This kind of planning methods leaves out the process of constructing complex environment models to search a near optimal path based on stochastic approaches; it overcomes the weakness where general mathematic model based algorithms often fail (or drop into local minimum) in solving NP-hard problem with large number of variables and nonlinear objective functions [4]. Figure 6 presents a set of typical current methods belonging to this category.

Where GA is Genetic Algorithm [89], it is the most famous population-based numerical optimization method; MA represents Memetic Algorithm [90]; it is a population-based heuristic exploring approach for combinatorial optimization problems; PSO is particle swarm optimization [91]; it is a population-based stochastic optimization algorithm; ACO is Ant Colony Optimization [92] that imitates the behavior of ant in finding the shortest path by using pheromone information; and SFLA is Shuffled Frog Leaping Algorithm [93] which is the combination of MA and PSO. For bioinspired algorithms, they are divided into Evolutionary Algorithm (EA) and Neural Network (NN) due to the fact that they are analyzed at different level, which will be discussed in detail in the following sections.

3.5. Multifusion Based Algorithms: Analysis. Fusion is a lately arising approach to improving the performance of 3D path planning algorithms; algorithms can benefit each other by this way. Usually, algorithms tend to fuse in a layer by layer way and aim to plan an optimal path (with better real time, or nonlocal optimal performance). For example, Artificial Potential Field algorithms usually tend to drop into local minima without navigation function or other tricks. Probabilistic Road Maps also cannot generate an optimal single path by itself. Thus this paper classifies this kind of algorithms, which are introduced by combining several algorithms together to achieve a better performance, as multifusion based algorithms. Section 8 will give a canonical illustration to this category.

4. Sampling Based Algorithms

In Section 3.1 a list of sampling based 3D path planning algorithms is already illustrated, and this part aims to give a detailed analysis of this kind of algorithms. In order to give a clear image of this kind of algorithms, this paper defines the RRT and its improved versions as RRT series, and PRM series contains PRM and its improved versions. This definition is also applicable to Voronoi and Artificial Potential Field.

4.1. RRT Series

4.1.1. Rapidly Exploring Random Trees. Rapidly exploring random tree (RRT) method is first proposed by LaValle [19]. The method attempted to solve path planning problems under holonomic, nonholonomic, and kinodynamic constraints. RRT has the advantage of handling multi-DOF problems; thus it is widely used for PR2 and other robots [94]. Authors in [31] proposed a fast local escaping version called Dynamic Domain RRT, which will be analyzed below. Karaman and Frazzoli [30] solved nonoptimal results problem of RRT by introducing RRG and a heuristic method RRT*.

RRT rapidly searches the configuration space to generate a path connecting the start node and the goal node. In each step a new node is sampled; if the extension from the sampled to the nearest node succeeds, a new node will be added. When this kind of method is applied to 3D environment, it normally assumes that there exists a 3D configuration space $X = C$. The configuration space consists of two parts, a fixed obstacle region, $X_{\text{obs}} \subset X$, which must be avoided, and an obstacle free region, $X_{\text{free}} \subset X$, where the robots must stay. Corresponding to the configuration space, a path state (or vertex) set P includes all the sampling vertices which are generated by RRT exploration process. In order to implement the algorithm, the following steps should be obeyed (see in Figure 7).

Step 1. First add the initial state $x_{\text{init}} \in X_{\text{free}}$ in P as the first vertex. Then randomly choose a state x_{random} in X_{free}, and Figure 7 illustrates two states which are x_{random1} and x_{random2}.

Step 2. Select a nearest state x_{near} to the newly generated state x_{random} in P based on a certain metric (mostly Euclidean metric) which is already designed; regard x_{near} as the parent state of x_{random}.

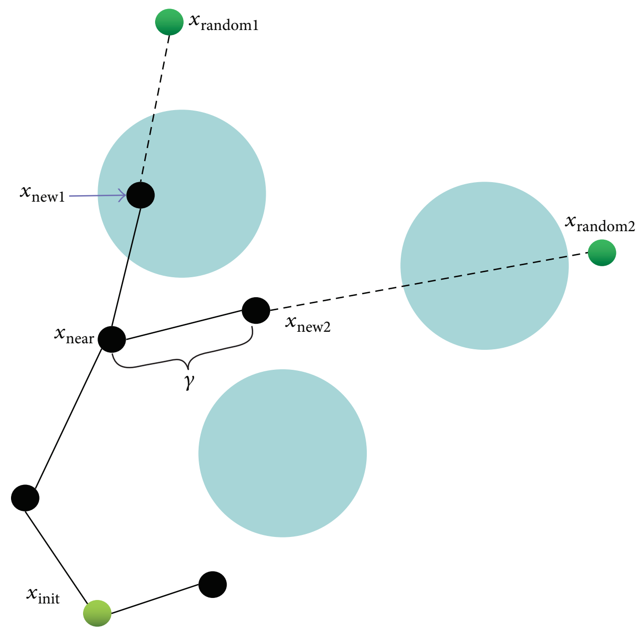


Figure 7: Exploring procedure of RRT algorithms. The cyan circles represent obstacle regions which cannot be passed. γ is maximal step extending length according to constraints and cost function stated.

Step 3. x_{random} is the state which shows the direction where the next step should go but may be beyond the robot's reachability. Thus a control input factor is added, considering the kinodynamic constraints, in a cost function $\phi = f(x, y, z)$ form. Then according to the constraints r and cost function ϕ, we get the reachable state x_{new}, judging x_{new} whether it is in X_{free}. If it locates in X_{free}, then add it to the path set P; else ignore this state. It is illustrated in Figure 7; we delete x_{new1} and repeat the whole process.

Although RRT can find a path to the goal, it is still the problem that RRT explores based on Monte Carlo random sampling, which is always biasing explored region as it will increase with the time. The method will consume much time to find a way out when the environments are cluttered, let alone converge to the optimal.

4.1.2. Dynamic Domain RRT. Dynamic Domain RRT (DDRRT) aims to solve the shortcoming where RRT considers none of the obstacles region in the configuration space, especially local trap region. RRT does not consider the information of local environments and thus causes inappropriate sampling which may lead to low time efficiency. An intuitive comparison is given in Figure 8, where the red arrow means the guiding to possible next extending direction.

DDRRT and RRT differ in Step 1; DDRRT introduces a n-dimensional sphere (based on the dimension of the environment) of certain radius r at the center x_{near} to represent the reachable region. At each time if the distance of new sampling node x_{random} to the nearest node x_{near} is within radius r, the sampling node will be chosen; otherwise it will be neglected. Then try to connect x_{near} and x_{random}; if the extension is succeed, then set $r = \infty$; otherwise set $r = R$, where R is the boundary radius of dynamic domain.

DDRRT solves the Voronoi bias problem of general RRT; it can ensure fast exploration. However, it is the same as RRT where no post-smooth-processing is included; the path which is generated by DDRRT will never tend to be optimal.

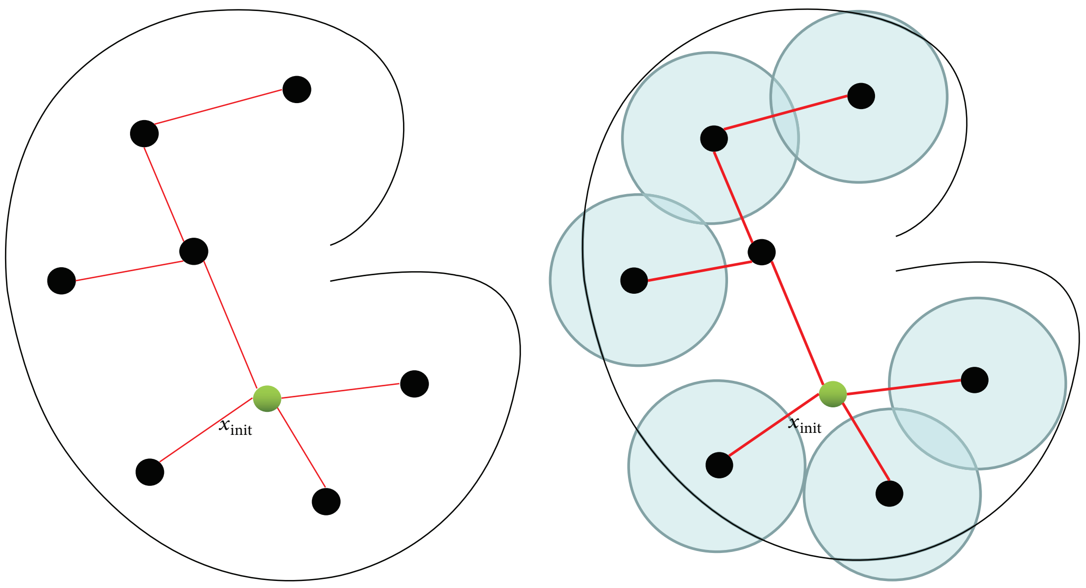
4.1.3. Rapidly Exploring Random Graph (RRG). RRT performs well in practice and can guarantee completeness, but it pays almost no attention to the quality of the results, and it is proved that RRT algorithms are not asymptotically optimal [30]. To ensure asymptotic optimality, authors in [30] introduced rapidly exploring random graph (RRG). The work used a structure $k = (S, S_0, T, L)$ to represent the performance of the system, where S is the set of states, $S_0 \subseteq S$ is the initial states sets, $T \subseteq S \times S$ is the transition relation, and L is the labeling function which is used to map each state to the set of atomic propositions.

It is illustrated in Figure 9, where "current new node" denotes the new nodes generated at current steps. Unlike RRT, RRG tries to extend to every state returned by Near() (line 7 of Algorithm 2), which is illustrated in Figure 9. In Figure 9(a), $|p|$ is the radius stated to choose the neighbor states around the center of "current new node"; then x_{new} connects all these nodes to form a graph as illustrated in Figure 9(b) if the connection is collision-free (in Algorithm 2, from line 7 to line 12). The pseudocode is given in Algorithms 1 and 2.

Line 7 shows the main difference between RRT and RRG, where RRG also connects to satisfaction vertices which lie in the circle region; thus it forms a graph. RRG holds the advantage of extending all the vertices returned by the Near() process and then connects them to present a more complex map. Although it seems to be more complex in some way, it almost surely can converge to an optimal path. Reference [29] showed experimental results of RRG, and the comparison to PRM proves the advantage of RRG. Also, the conclusion suggested that RRG enables navigating multiple robots simultaneously. However, it generates a complex network like PRM and thus cannot find the optimal path by itself.

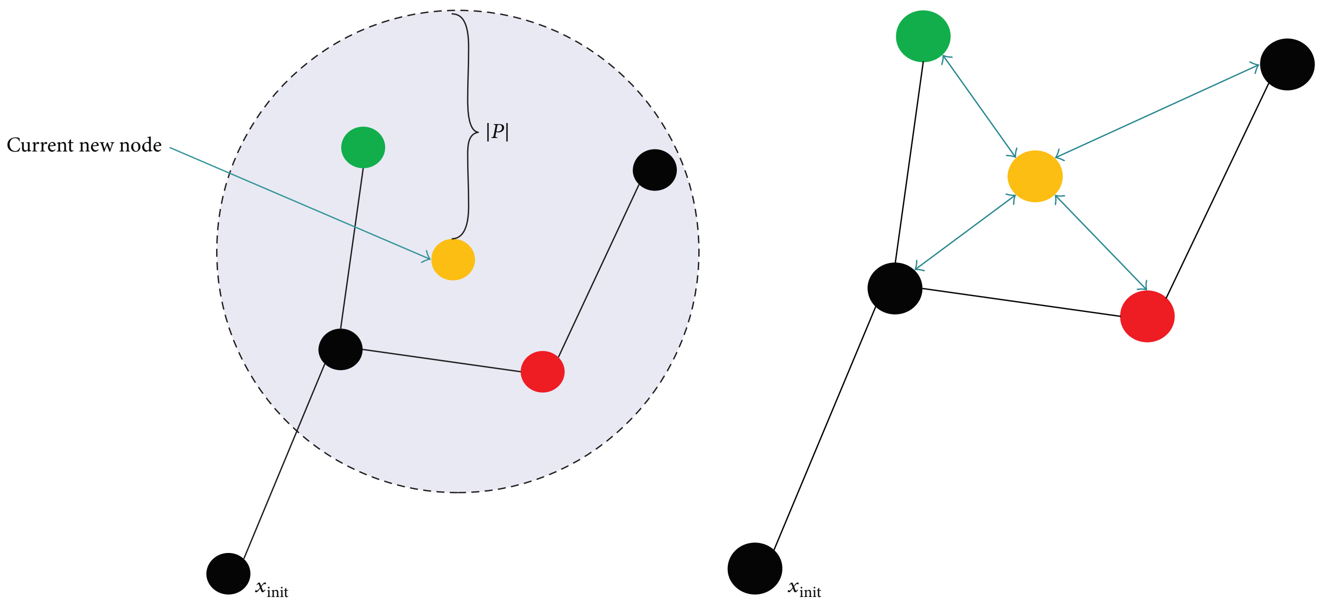
4.1.4. RRT-Star (RRT*). RRT* [30] is the tree version of RRG which also preserves the asymptotic optimal property as RRG; it is proposed to tackle differential constraints. RRT* removes probable bad connection which works in a refining way; thus it optimizes the solutions to be less expensive than they used to be. Here a cost-function cost(p) is defined to represent the cost of the unique path from x_{init} to an arbitrary state $p \in P$ and gives the cost(x_{init}) equal to zero initially. RRT* differs RRG in postprocessing process, compared to Algorithm 2; RRT* pseudocode is given in Algorithm 3 with intuitive illustration Figure 10.

Let us take the same example case as illustrated in Figure 9(a). RRT* first finds the nearest state as well as the neighbor states X_{near} and then adds the nearest state to the tree as well as the minimal cost connection if it exists (from line 4 to line 12). Further, RRT* tries to eliminate the connections which have a larger cost by connection via x_{new} state (from line 14 to line 21). Because of the pruning and



(a) It is hard for general RRT to deal with bug trap problem (b) DDRRT tackles trap problem by limiting the extension near the margin

Figure 8: DDRRT compares with general RRT.

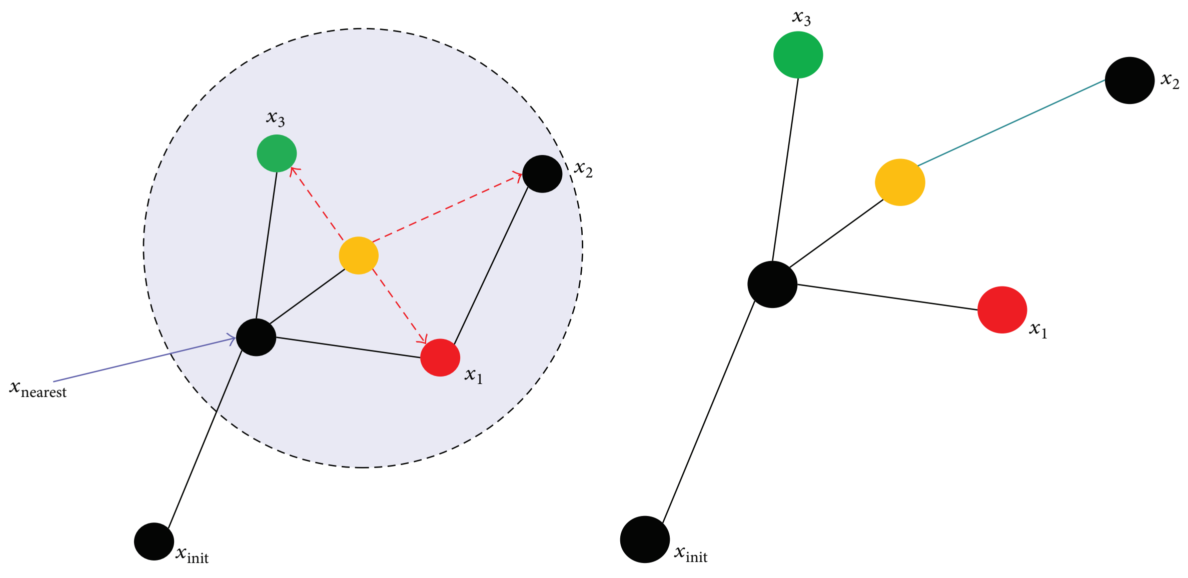


(a) The near nodes returned (b) x_{new} connects all near nodes to form RRG

Figure 9: Typical operation of RRG.

reconnecting, the tree turns to be more compact and dense as illustrated in Figure 10(b). Consequently the overall minimal cost can be obtained. However, the whole time consumption increases, and also it cannot work to generate multipath. Choudhury et al. [28] assumed that if the nearest parent already has children nearby; then the second best parent will be chosen. This method solves the problem of general RRT and RRT* by supporting an on-line fast replanning method.

4.2. PRM Series. Unlike RRT, Probabilistic Road Map (PRM) considers different choices for the set of states to which connections are attempted. PRM [20] is the first popular multiple-query method for building a road map by using sampling approach. When applied to 3D space, it first defines the configuration space X and an obstacle free space X_{free}. The method samples a random state in X_{free}. Then it tries to connect to nearest k neighbors (K-PRM) or connect to states within a δ-ball region or connect to states under



(a) Pruning process of RRT* (b) The final RRT* graph

FIGURE 10: RRT* postrefining process, that is, pruning of heavy connect.

```
E is the set of edges
(1) $V \leftarrow x_{\text{init}}$
(2) for $i = 0$ to $n$
(3)     $G = (P, E)$
(4)     $x_{\text{rand}} \leftarrow i$th randomly chosen state
(5)     $(p, E) = \text{Explore}_{\text{RRG}}(G, x_{\text{rand}})$
(6) End
```

ALGORITHM 1: Body of RRG.

TABLE 1: Elements of PRM.

Name of each sub-PRM	Criteria of corresponding algorithm
K-PRM	Meaning each step choose nearest k neighbors to be the states which are under consideration.
S-PRM	The expected states are to be chosen within a ball (or circle) region, for all states included in the radius r will be chosen to connect to the vertex.
K-S-PRM	A combination of above two, where for a given radius r the upper bound of the vertices to be chosen is k.

computational burden which is a combination of the above two, that is, K-PRM $\cap$ δ-ball. It should be declared that each connection is collision-free. After the road map is absolutely formed, a node based search algorithm (Dijkstra, A*, D*, etc.) is used to find the least cost path from initial state to the goal state. The pseudocode of PRM is illustrated in Algorithm 4.

In Algorithm 4, PRM adds all the connections which are without collision (from line 4 to line 9). Authors in [95] first implemented PRM into 3D environment and showed fast exploration performance. However, with the expanding of the exploring graph, the expense on collision checking increases. Amato et al. [32] put it forward by proposing an obstacle-based nodes generation strategy; at each step the road map candidate points are selected on the obstacle surface. This trick reduces the total processing time by ignoring the useless points in obstacle region. Hsu et al. [33] solved the "dynamic threat weak" problem of PRM by introducing a concept of "milestone," that is, state × time which creates a real time graph connecting the initial point and the goal point. It was implemented in ground robots and proved of having quick convergence ability. Yan et al. [6] proposed an octree to build 3D grid-structure which weakens the effect of randomness, thus guaranteeing fast searching ability. The work gave a great thought of random sampling in bounding box array. Reference [30] introduced PRM* to guarantee asymptotical optimality. Authors in [34] concentrated on the problem that typical PRM cannot tackle dynamic threats; they imported the idea of potential to avoid this, which is called Reactive Deformation Road Maps (RDR).

PRM has the bottleneck of not being able to determine an efficient near vertex choosing and connection principle. Based on the difference of vertex chosen criteria, this paper combines the idea of [10, 57] and divides PRM into three parts in Table 1.

Although K-PRM has the advantage of adopting enough samples to ensure smoothness by adjusting parameter k, the extension will be biased if a fixed direction tends to be much more dense. S-PRM can keep out the shortcoming of K-PRM, but an unreasonable radius r may result in an excessive computational burden. K-S-PRM likes an adaptable S-PRM; it can guarantee all direction connection and also limit the computational burden to a certain degree.

4.3. Voronoi.

Shamos and Hoey [96] introduced the Voronoi diagram into the field of computational geometry; it is firstly

```
Steer($x$, $y$) returns a point closer to $y$ but within the reach of $x$ by a certain value;
Near($G, x_{\text{rand}}, |P|$) return all vertices in a ball center at $x_{\text{new}}$ and radius is $r_n$;
Freeobsta($x$, $y$) means no obstacle between $x$, $y$;
(1) $P' = P; E' = E$
(2) $x_{\text{nearest}} \leftarrow$ Nearest($G, x$)
(3) $x_{\text{new}} \leftarrow$ steer($x_{\text{nearest}}, x$)
(4) if Freeobsta($x_{\text{nearest}}, x_{\text{new}}$)
(5)      $P' = P' \cup x_{\text{new}}$
(6)      $E' = E' \cup \{(x_{\text{new}}, x_{\text{nearest}}), (x_{\text{nearest}}, x_{\text{new}})\}$
(7)      $X_{\text{near}} \leftarrow$ Near($G, x_{\text{new}}, |P'|$)
(8)      while $x_{\text{new}} \in X_{\text{near}}$
(9)      if Freeobsta($x_{\text{new}}, x_{\text{near}}$)
(10)        $E' = E' \cup \{(x_{\text{new}}, x_{\text{nearest}}), (x_{\text{nearest}}, x_{\text{new}})\}$
(11)      End
(12)    End
(13) End
Return $G' = (P', E')$
```

Algorithm 2: Explore RRG (G, x).

```
Steer($x$, $y$), Near($G, x_{\text{rand}}, |P|$), Freeobsta($x$, $y$) are the same as RRG;
Parent($x_{\text{near}}$) returns the parent vertex of $x_{\text{near}}$;
(1) $P' = P; E' = E$
(2) $x_{\text{nearest}} \leftarrow$ Nearest($G, x$)
(3) $x_{\text{new}} \leftarrow$ steer($x_{\text{nearest}}, x$)
(4) if Freeobsta($x_{\text{nearest}}, x_{\text{new}}$)
(5)      $P' = P' \cup x_{\text{near}}$
(6)      $E' = E' \cup \{(x_{\text{new}}, x_{\text{nearest}}), (x_{\text{nearest}}, x_{\text{new}})\}$
(7)      $X_{\text{near}} \leftarrow$ Near($G, x_{\text{new}}, |P|$)
(8)      while all $x_{\text{new}} \in X_{\text{near}}$
(9)      if Freeobsta($x_{\text{new}}, x_{\text{near}}$) and
(10)       cost($x_{\text{new}}$) > cost($x_{\text{near}}$) + $c$(Dist($x_{\text{new}}, x_{\text{near}}$))
(11)         $x_{\text{min}} = x_{\text{new}}$;
(12)       End; END;
(13) $E' = E' \cup \{x_{\text{min}}, x_{\text{new}}\}$
(14) while all $x_{\text{new}} \in X_{\text{near}} \backslash \{x_{\text{min}}\}$
(15)  if Freeobsta($x_{\text{min}}, x_{\text{new}}$) and
(16)  cost($x_{\text{near}}$) > cost($x_{\text{new}}$) + $c$(Dist($x_{\text{new}}, x_{\text{near}}$))
(17)    $x_{\text{parent}} \leftarrow$ Parent($x_{\text{near}}$);
(18)    $E' = E' \backslash \{x_{\text{parent}}, x_{\text{near}}\}$
(19)    $E' = E' \cup \{x_{\text{new}}, x_{\text{near}}\}$
(20)  End
(21) End
Return $G' = (P', E')$
```

Algorithm 3: Explore$_{\text{RRT}*}(G, x)$.

used for finite points in the Euclidean plane and now widely used with a series of improved forms in the field of path planning. Voronoi diagram generates topological connection; the distances from the edges to the nearby obstacles are the same.

3D Voronoi first selects an initial site; this site's coordinates hold the property that the minimal distance to the obstacles nearby is the same. Then it calculates new Voronoi sites based on *calculation of the Voronoi channel* [88] and defines the Voronoi net bounds. The whole process stops when all the sites are recorded and all the channels are obtained. The *calculation of the Voronoi channel* is used to choose a Voronoi site, and the equation of Voronoi channel for the triplet of objects $\{i, j, k\}$ is as follows:

$$d_i(r) = d_j(r) = d_k(r), \quad (1)$$

where $d_i(r)$ is the minimal distance from a give point (or robot as a mass point with 3D coordinate) to the surface of ith obstacle in 3D space. Equation (1) extends the characteristics of general Voronoi, if there exists a small shift v along the

```
U is the vertices set according to the node choose method claimed ahead;
Freeobsta($x_{\text{rand}}, u$) means no obstacle between $x_{\text{rand}}$ and $u$;
P is the path vertex set, E is the path edges set;
(1)  $P' = \varnothing$; $E' = \varnothing$;
(2)  for $i = 0$ to $n$
(3)      $x_{\text{rand}} \leftarrow$ the $i$th sample state in $X_{\text{free}}$
(4)      $U \leftarrow (K\_\text{PRM}$ or $\delta$-ball or $K \cap \delta) \backslash \{p\}$
(5)       $P = P \cup x_{\text{rand}}$
(6)      for each $u \in U$
(7)       if Freeobsta($x_{\text{rand}}, u$)
(8)         $P = P \cup \{(x_{\text{rand}}, u), (x_{\text{rand}}, u)\}$
(9)       End; End;
(10)  End;
Return $G = (P, E)$
```

Algorithm 4: Whole PRM.

channel and the point $r' = r + v$ should satisfy the constraint equation,

$$d_i(r+v) = d_j(r+v) = d_k(r+v). \tag{2}$$

With respect to (1), linearization of this system should obey

$$(\nabla d_i \cdot v)\big|_r = \left(\nabla d_j \cdot v\right)\Big|_r = (\nabla d_k \cdot v)\big|_r. \tag{3}$$

This ensures finding the direction of the displacement v; thus we can get a new point r' along the vector.

An estimation function (4) is used to improve the computer-based accuracy, that is, controlling the deviation of the trajectory from the Voronoi channel:

$$\Phi = \left(d_i - d_j\right)^2 = \left(d_i - d_k\right)^2 = \left(d_j - d_k\right)^2. \tag{4}$$

Let $\Phi = 0$ ensure that the point is on the Voronoi channel. If $\Phi > \delta^2$ (δ is desired value), then it returns to the Voronoi channel by applying gradient decay procedure. However, in order to ensure fast convergence, the preliminary sampling of each state is very important; thus this kind of algorithm is classified as sampling based (even though it introduced information adjustment feedback to correct the initial sampling).

Voronoi generates a global graph or local graph, but almost the same as RRG and PRM; it also cannot generate the shortest path at the same time. Thus it seeks help from Dijkstra's algorithm, A*, D*, and so forth. This paper defines three steps to Voronoi path generation methods: (a) sampling the environment or just seeking help from other environment construction methods, (b) generating a 3D Voronoi graph, and (c) employing a search algorithm to find the minimal cost path globally.

Luchnikov et al. [88] first proposed an elaborate 3D Voronoi diagram construction method, which solves 3D complex system path planning problem. Reference [35–37] improved it to a further stage by proposing a radial edge like data structure which is capable of dealing with topological characteristics of Euclidean Voronoi diagram of spheres. The topological characteristics are *region*, *face*, *edge*, *vertex*, *oop*, and *partial edge*, which contain the geometric information and adjacency relationship. When implemented in reality, Lifeng and Shuqing [38] introduced the method that uses geographical information system to generate the environment nodes and combined with Dijkstra's algorithm to find the shortest path. Sharifi et al. [97] assigned Voronoi region to a group of UAV to solve the problem of coverage planning for an environment. Voronoi was combined with potential field method in [39, 40] which is efficient to guarantee fast convergence. Voronoi channel is built mainly based on static obstacles; authors in [41] tackle dynamic threats by adding a bound to the Voronoi channels, and the idea can be introduced to 3D environment.

4.4. Artificial Potential Algorithms.

Since first proposed by Khatib [86], potential field methods have been widely researched and implemented universally because of its low computational complexity. Potential field methods are based on the idea of assigning a potential function (relationship between obstacle free and obstacle space) to free space and simulating the vehicle as a particle reacting to force due to the potential field. It computes the goal attraction and obstacle repulsion simultaneously and guides the robot along the total force gradient. The potential field can be represented as

$$U_t(X) = U_a(X) + U_r(X), \tag{5}$$

where, in formula (5), $U_t(X)$ is the total potential at state X, $U_a(X)$ is the attractive potential at state X, and U_r denotes the repulsive potential at state X. Given $\nabla U_r(X) = -\sum^{\text{all}} f_{r,i}(X)$, that is, the virtual attractive force (from all neighbor obstacles, also please keep in mind the relation between the gradient of potential field and virtual force can be negative or positive for your consideration), then the force field at state X can be represented as

$$f_t(X) = -\nabla U_t = f_a(X) + \sum_{i=1}^{\text{all}} f_{r,i}(X). \tag{6}$$

For the attractive potential,

$$U_a = k_a \left\| X_G - X \right\|, \tag{7}$$

$$f_a(X) = -\nabla U_a = -k_a \frac{(X_G - X)}{\left\| X_G - X \right\|}, \tag{8}$$

where X_G is the goal node, $X_G - X$ is the error vector with respect to the goal node, and $f_a(X)$ is the attractive force at X. Hence, attractive force increases with the distance between current node and goal node. For the repulsive potential,

$$U_{r,i} = \begin{cases} 0 & \text{if } \eta_i(X) > \eta_{0,i} \\ K_{r,i}\left(\dfrac{1}{\eta_i(X)} - \dfrac{1}{\eta_0(i)}\right) & \text{if } \eta_i(X) \le \eta_{0,i}, \end{cases} \tag{9}$$

$$\begin{aligned} f_{r,i}(X) &= -\nabla U_{r,i}(X) \\ &= \begin{cases} 0 & \text{if } \eta_i(X) > \eta_{0,i} \\ \dfrac{K_{r,i}}{\eta_i^2(X)}\left(\dfrac{1}{\eta_i(X)} - \dfrac{1}{\eta_0(i)}\right)\nabla\eta_i(X) & \text{if } \eta_i(X) \le \eta_{0,i}, \end{cases} \end{aligned} \tag{10}$$

where $\eta_i(X)$ is the distance from x to obstacle region which has been partitioned in convex components, the distance can be defined flexibly, $\Delta\eta_i(X)$ is a polar value, $f_{r,i}$ is the ith obstacle component repulsive force to X, and $\eta_{0,i}$ is the safe margin of ith convex component which is defined by certain obstacle. For field implementation, APF tends to be more complicated as the bounder of the obstacles is not constant; thus the time efficiency is constructed based mainly on initial guess; thus we classify it as sampling based algorithm.

However, such methods are incomplete because they are prone to drop into local minima area. Many studies have been done to help potential field algorithms to overcome the local minima by generating navigation functions or computing the potential with constraints. Reference [42] proposed a harmonic potential method to solve the local minima problem by using Laplace's Equation to constrain the generation of a potential function; Rimon and Koditschek [43] proposed a Morse function with a single minimum at the desired destination strategy to form a strong stable robot navigation method to jump out of the local minima, and this is the first formally proposed navigation method. Authors in [44] imported the idea of Hamilton-Jacobi-Bellman to form a HJB function:

$$\min_u \langle V(x), u \rangle = -1, \tag{11}$$

where u is the control and $V(x)$ is the potential factor. This function generates a unique global minimum value; thus it is able to yield a global feasible path that jumps out of local minima, and [45, 46] put it to a higher stage. Ge and Cui [47] solved the goals nonreachable with obstacle nearby (GNRON) problem by introducing a new repulsive:

$$\begin{aligned} U_{r,i} &\\ &= \begin{cases} 0 & \text{if } \eta_i(X) > \eta_{0,i} \\ K_{r,i}\left(\dfrac{1}{\eta_i(X)} - \dfrac{1}{\eta_0(i)}\right)^2 \rho^n(X, X_G) & \text{if } \eta_i(X) \le \eta_{0,i}, \end{cases} \end{aligned} \tag{12}$$

where $\rho^n(X, X_G)$ is the minimal distance between the robot X and the goal X_G. $\rho^n(X, X_G)$ ensures the total potential $U_a(X)$ arrives at its global minimum, that is, 0, if and only if $X = X_G$.

4.5. Analysis. Sampling based algorithms exploring depends much on initial guess, and this guess repeats every step. The algorithms may take advantage of nearby collision detection or potential adjustment. Although this leads to loss of completeness and inexplicit construction of the environment sometimes, this can somehow reduce the dependence on environment model and thus enable them to be implemented in various environments. We summarize all the advantage and weakness of this kind of algorithms and analyze each subcategory in detail as illustrated in Table 2.

Except for algorithms analyzed in Table 2, algorithms such as visibility graphs [18] and corridor map [98] also belong to sampling based algorithms. Visibility graph method likes a simple version of obstacle-based PRM which is proposed by Amato et al. [32]. Corridor map method likes octree-structure PRM introduced by Yan et al. [6]. They both defined a cell decomposition method to construct the workspace. However, such as PRM and Voronoi, these methods need a node search algorithm to achieve the best path.

5. Node Based Optimal Algorithms

Node based optimal algorithms are classified by the reason that they deal with nodes' and arcs' weight information (but not limited to these, or sometimes called grid); they calculate the cost by exploring through the nodes, thus to find the optimal path. This kind of algorithms is also called network algorithms [99] which means they search through the generated network, and it means the same with node based algorithms.

5.1. Dijkstra's Algorithm. Named after Dijkstra [21], Dijkstra's algorithm targets for finding the shortest path in a graph where edges'/arcs' weights are already known. Dijkstra's algorithm is a special form of dynamic programming and it is also a breath first search method. It finds shortest path which depends purely on local path cost. When applying 3D space, a 3D weighted graph must be built first; then it searches the whole graph to find the minimum cost path. A generalization of Dijkstra's algorithm is shown below with pseudocode in Algorithm 5.

Authors in [100] proposed an improved Dijkstra's algorithm by adding a center constraint; it works well in tubular objects, which is first proposed by [101]. The work [102] showed that 3D GIS environment combined method may be a practical way to implement in real outside world, and experimental results are provided to prove that Dijkstra's algorithm acts well enough. But Dijkstra's algorithm relies much on the priority queue Q data structure type, which influences the total exploring time.

5.2. A-Star (A).* A-star (A*) [22] is an extension of Dijkstra's algorithm, which reduces the total number of states by introducing a heuristic estimation of the cost from the

Table 2: Analysis of sampling based algorithms.

Method type	Shortcoming		Advantages
	Shortcomings	Improvement	
RRT	Single path	[28, 29]	Low time complexity, fast searching ability
	Nonoptimal	[29, 30]	
	Static threat only	[31]	
PRM	Expensive collision check	[6, 32, 33]	Appropriate for complex environments and replanning situations
	Static threat only	[34]	
	Nonoptimal	[30]	
Voronoi	Incomplete representation	[35–38]	Easy to implement on-line and ignoring collision checking
	Nonconvergence	[39, 40]	
	Static threat only	[41]	
Artificial potential	Local minima	[42–47]	Fast convergence

```
Q is the priority queue; $x_G$ is the goal;
$l(x,u)$ return the cost to apply action $u$ from $x$;
$C'$ is the best cost-to-come known so far;
$U_x$ is the finite action space;
$x' = f(x,u) \in X$ is a state transition function;
(1)  $C(x) = \infty \mid (x \neq x_{\text{init}}); C(x_{\text{init}}) = 0$
(2)  Q.Insert($x_I$);
(3)  while Q not empty
(4)   $x \leftarrow$ Q.GetFirst()
(5)   for all $u \in U(x)$
(6)    $x' \leftarrow f(x,u)$
(7)    if $C(x) + l(x,u) < \min\{C(x'), C(x_G)\}$
(8)    $C'(x) = C(x) + l(x,u)$
(9)    if $x' \neq x_G$
(10)     Q.Insert($x'$)
(11)  END; END
(12) END; END;
```

Algorithm 5: Dijkstra's algorithm.

current state to the goal state. The heuristic function can be designed to obtain the constraints, while the estimation must never overestimate the actual cost to get the nearest goal node. By applying this guiding like heuristic, A* can converge very fast and ensures optimality as well.

A* is proposed by introducing an evaluation function (13), which consists of postcalculation toward the initial state and heuristic estimation toward the goal,

$$f(n) = g(x) + h(x), \quad (13)$$

where $g(x)$ is the cost from initial state x_{init} to current state x, which is the same as $c(x)$ in Algorithm 5. $h(x)$ is the heuristic estimation of the cost of an optimal path from current state x to goal state. The estimation $h(x)$ of each state tends to be close to the real cost; thus A* has a faster speed to converge based on comparison of the cost of neighbors. Compared with Dijkstra's algorithm, A* obtains a faster speed to converge.

For 3D environment, A* has been widely implemented. Amato et al. [32] constructed the feasible road map by applying PRM and then adopted A* to execute best route exploration. Authors in [6] implemented A* with UAV platform with an octree based PRM. Niu and Zhuo [53] introduced "cell" and "region" conception to enhance the environment understanding of A*, thus enabling flexible representation of 3D environments. Koenig and Likhachev proposed an environment-representation varying adaptive A*, that is, Lifelong Planning A* (LPA) [48]. LPA* can adapt to environment changes by using previous information as well as iterative replanning. Williams and Ragno [49] propounded a conflict-direct A*; it accelerates the exploring process by eliminating subspaces around each state that are inconsistent. It is proposed in [50] that A* can choose any state to be parent state, thus resulting in a more flat turning angle, named Theta*. The algorithm has the ability to be able to obtain system constraints; thus it can find shorter and more realistic path. De Filippis et al. [103] implemented both Theta* and A* in 3D environment, and an experimental comparison is given to prove that Theta* reduces the searching compared to A*. Although Theta* acts well compared to A*, but when applied to 3D environment, it consumes much time to check unexpected neighbors. Line-of-sight check method, called lazy Theta* [51], is proposed to avoid unnecessary check of unexpected neighbors. Authors in [54] introduced a method to reuse information from previous explorations and update information through the affected and relevant portions of the exploring space. This may cause extra computational consumption, but it can tackle dynamical threat and converge fast.

5.3. *D-Star (D*).*

D-star (D*) [23, 52], short for dynamic A*, is famous for its wide use in the DARPA unmanned ground vehicle programs. D* is a sensor based algorithm that deals with dynamic obstacles by real time changing its edge's weights to form a temporal map and then moves the robot from its current location to the goal location in the shortest unblocked path.

D*, As well as A*, evaluates the cost by considering the postcalculation and forward estimation. D* maintains a list of states which is used to propagate information about changes of the arcs cost function. The evaluation function is represented as

Table 3: Analysis of node based optimal algorithms.

Method type	Shortcoming		Advantages
	Shortcomings	Improvement	
Dijkstra's algorithm	High time complexity	[22, 23, 48–52]	Easy to implement for various environments
	Static threat only	[23, 52]	
A*	Heavy time burden	[48, 50, 51]	Fast searching ability, enabling implementation on-line
	Nonsmoothness	[50, 53]	
	Static threat only	[23, 52, 54]	
D*	Unrealistic distance	[55, 56]	Fast searching ability, dealing with dynamic environments

$$f'(n) = g'(x) + h'(x). \quad (14)$$

However, differing from A*, $h'(x)$ is not necessarily the shortest path length to the goal compared to A*; also computation of $h'(x)$ assumes that the robot can pass through obstacles. At each time it updates a minimum heuristic function when it encounters new obstacles and the whole graph, thus enabling efficient searching in dynamic environments.

Smirnov [55] considered the worst travel distance case of D* and restrained its lower bound as $\Omega((\log|V|/\log\log|V|)|V|)$ steps and upper bound as $O(|V|^2)$. The bound is used to solve the problem that D* uses unrealistic distance in its graph, and there is a large gap between the lower bounds and upper bounds. Based on [55], Tovey et al. [104] put it forward by adding more tighter bounds for D*. Koenig and Likhachev [56] extended LPA* to the case where the goal changes between replanning episodes; it is like a simplified version of D*, called D* Lite, but it is proposed based on LPA*.

5.4. Analysis. Table 3 provides a straightforward summarization of node based optimal algorithms. As the name implies, node based optimal algorithms deal with node and arc information. However, as the nodes and arcs provide an incomplete structure of the configuration space, this kind of method can only achieve the best result limited by the representation of environment. This kind of algorithms is single search methods. It cannot generate multipath for fleet. For real time implementation, Dijkstra's algorithm's time complexity is $O(n^2)$ (n is the number of the nodes); A* and D* reduce the complexity to a lower degree, which enables on-line implementation.

6. Mathematic Model Based Algorithms

Node based optimal algorithms use grids to represent configuration space; meanwhile this kind of methods assumes the robot as a point and only considers the acceleration and velocity constraints. Thus it is not complete to represent the environments as well as consider system dynamics. Mathematic model based algorithms optimize by describing kinodynamic constraints in combination of polynomial forms. They can model the environment as a time variant system (sometimes even a model and time variant system), which is synchronization to the current location of the robot.

Due to the fact that this kind of algorithms can handle dynamics constraints to achieve cost optimum, some researchers regard mathematic optimization methods as trajectory planning method. However, mathematic model based algorithms are proposed to solve the problem of finding a path or trajectory locally or globally [45, 105]. In order to distinguish from trajectory planning methods, we provide a stronger definition compared to Definition 4.

Definition 5. Trajectory planning assumes output to be time continuous and must be able to ensure control limitation, which is still a part of the whole path.

According to Definition 5 above, local mathematic optimization methods also partly belong to path planning field. Authors in [105] transformed kinodynamic constraints into a bound of linear or nonlinear constraints and then used mathematic programming methods to find the bounded optimal path. Miller et al. [45] modeled the path planning problem in an optimal control form, which combines cost criterion and Hamiltonian function to form a boundary value problem (BVP) to find a realistic optimal path globally. Chamseddine et al. [63] introduced flatness based method to linearize the nonlinear bounds into polynomial form, and a bang-bang control is employed to generate a global optimal path. Other mathematic optimization methods such as level set method and support vector machine method also belong to this kind.

6.1. Linear Algorithms. For linear algorithm form based path planning problem, this kind of algorithm tends to have the following form:

$$J_{\text{cost-goal}} = f(u, x, y, z) + \varphi(x, y, z) + R(x, y, z) \quad (15)$$

subject to

$$\psi_0 = \psi[u(t_0), x(t_0), y(t_0), z(t_0)], \quad (16)$$

$$\psi_f = \psi[u(t_f), x(t_f), y(t_f), z(t_f)], \quad (17)$$

$$g_{\text{lower}} \le g(u, x, y, z) \le g_{\text{upper}}, \quad (18)$$

$$u_{\text{lower}} \le u(t) \le u_{\text{upper}}, \quad (19)$$

where $J_{\text{cost-goal}}$ is a cost function that takes into kinodynamic constraints as as well as the expected properties, such as minimum distance, energy, and threat. $f(u, x, y, z)$ represents

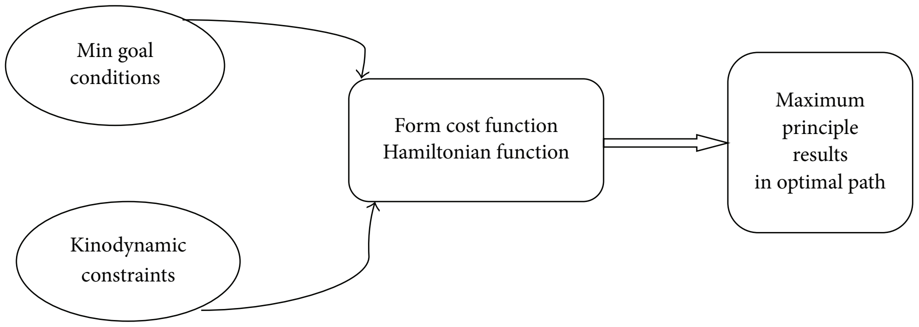

Min goal conditions
Kinodynamic constraints
Form cost function Hamiltonian function
Maximum principle results in optimal path

Figure 11: Basic optimization problem steps.

the control factor, $\varphi(x, y, z)$ is the kinodynamic constraint which acts as penalty function, and $R(x, y, z)$ is the exploring step reachable region function to ensure a strong convergence ability to the goal. Equation (16) is the initial condition, (17) is the final condition, (18) is algebraic path constraints, and (19) is the control constraints.

Linear algorithms have the ability to describe the environment completely; meanwhile they can model the kinematics and dynamics constraints. Furthermore, linear algorithms can handle control disturbance or model uncertainty. Mixed-Integer Linear Programming (MILP) methods combine binary and integer logical constraints and are most commonly used because they closely represent the environment and system. Reference [3, 5, 106] implemented MILP, respectively, in aerial, underwater, and ground robots. Bhattacharya [64] directly supported a free MATLAB toolbox, which is called OPTRAGEN, to solve the MILP problem. Masehian and Habibi [62] used 0-1 binary integer to represent the path length operator and then solved the path planning problem using binary integer programming.

6.2. Optimal Control. Path planning problem can be posed in an optimal control form, where optimal control can find the state and control oriented path based on a set of differential equations [107]. Optimal control can be interpreted as an extension of the linear algorithms to an infinite number of variables' condition, with the ability to model uncertainty as linear chance constraints much more easier.

A basic optimization problem is depicted in a flowchart form in Figure 11, where the initial state (or current state) and goal state are included in the constraints to ensure completeness.

For optimal control case, a general system model is considered in time continuous form,

$$\dot{x} = f(x(t), u(t)), \tag{20}$$

where $x(t)$ represent the evolving state and $u(t)$ is the control parameters. The minimizing criterion is

$$L = \phi[x(t), u(t)]. \tag{21}$$

With the necessary constraints,

$$\psi(x(t_0), t_0) = a, \tag{22}$$

$$\psi(x(t_f), t_f) = b. \tag{23}$$

Then cost function (an recommended performance index) can be derived by including all constraints above:

$$J = \phi[x(t), u(t)] + \int_{t_0}^{t_f} \lambda^T(t)\{f[x(t), u(t)] - \dot{x}\}. \tag{24}$$

Then we can achieve the Hamiltonian:

$$H = \lambda^T(t) f[x(t), u(t)], \tag{25}$$

where (22) is the initial condition and (23) is the final condition. Hamiltonian is used to solve the optimal problem based on the maximum principle and then follow the typical optimal solving procedure to generate an optimal path globally. This paper only explains in a general form; each equation can be extended to contain much more constraints.

Tisdale et al. [58] implemented discrete receding-horizon control (RHC) in UAV, which is a variant of optimal control. Chen and Schwartz [65] described the optimal control problem in detail and proposed a free MATLAB toolbox RIOTS_95, and in [108] they further implemented it to solve model predictive problem.

6.3. Analysis. This kind of methods maintains a complete form to describe the states and environmental variables, and when applied in 3D cluttered environments, these methods can adapt themselves by employing much more constraints according to the environments. These algorithms tend to have a much more complex formulation, that is, a heavy computational burden. There exists a method that can solve the weakness, that is, discrete decisions in an optimization procedure. The method allows problems to be solved on-line flexibly by devising the environment [58–60]. For example, Bellingham et al. [61] only used MILP to justify the local states and thus increase the whole exploring process a lot. Based on the analysis above, this paper provides a summarization in Table 4.

Table 4: Analysis of mathematic model based algorithms.

Method type	Shortcoming		Advantages
	Shortcomings	Improvement	
Mathematic model based algorithms	High time complexity	[58–62]	Containing almost all the information to generate an optimal path indeed
	No analytic solutions	[63]	
Free tool	OPTRAGEN [64], RIOTS_95 [65] DIDO [66], SeDuMi [67], and so forth		

7. Bioinspired Algorithms

Path planning is attributed to the top layer of a robot control process, which enables robots to work without (or with little) help from man. The motivation of planning a path in real cluttered environments is that the robots should obtain the ability to accomplish the mission by itself without supervision. For bioinspired algorithms, they originate from imitating the way how humans or other natural creatures behave or think, and they form a family of a series of algorithms which can solve NP-hard problems to generate a near optimal path.

There are two subcategories of bioinspired algorithms: one is Evolutionary Algorithm, which stems from analyzing the behavior of a certain species; another is Neural Network algorithm which follows the way how inner neuron processes the information. They belong to different level accordingly, and this paper discusses them, respectively. For Evolutionary Algorithms, they work almost with the same mechanism; thus this paper mainly analyzes two relative popular algorithms: Genetic Algorithm and Ant Colony Optimization Algorithm.

7.1. Evolutionary Algorithm. Evolutionary Algorithm is an umbrella name which includes Genetic Algorithm (GA), Memetic Algorithm (MA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Shuffled Frog Leaping Algorithm (SFLA). EA was propounded to solve the problem where traditional linear programming and dynamic programming often fail to solve NP-hard problems with large number of variables. EA is a stochastic search approach that imitates the metaphor of natural biological evolution and social behavior. The firstly proposed and now widely implemented Evolutionary method is GA; then inspired by different natural process four other methods developed.

Figure 12 illustrates a typical flowchart of Evolutionary Algorithm, which is proposed by Dong and Juris [109]. Evolutionary Algorithms start by selecting randomly feasible solutions as the first generation. Then it takes the environment, robot's capacity (dynamic ability), goal, and other constraints into consideration, to evaluate the fitness of each individual. In the next step, a set of individuals is selected as parents for the next generation according to their fitness. The last step is a mutation and crossover step. The whole process is performed in an iterative way and stops the process when a preset goal is achieved. The best fitness individuals are decoded as the optimal path nodes.

7.1.1. Genetic Algorithms. Holland [89] firstly introduced GA, and now it is the most popular population-based optimization method. The basic version of GA defines a cost function to evaluate the potential solutions. Then a partly random crossover operator takes two parents from the population set and recombines them in some way. The mutation operator tries to modify the solutions and aims to achieve a valid solution in order to escape local optimality.

GA holds the process that all individuals can exchange information; by doing this way, this nest generation can converge very fast by using this information. However, if the population becomes too similar and loses population diversity, it often leads to premature convergence. If there exists too much population diversity, it may result in a heavy computational burden to investigate the poor solutions. GA needs to repeatedly evaluate the fitness of the current generation, which also causes a high computational burden.

Fonlupt et al. [70] proposed a cooperating GA to solve the premature convergence problem of typical GA. The method holds the concept that when a certain GA is stuck in local minimum, then another GA might provide a feasible path point to allow it to search again. Hacioglu and Ozkol [71] and Hacioĝlu and Özkol [72] solved the problem by introducing an idea, that is, periodically applying a vibrational mutation operator to the whole population. Therefore, it becomes possible to escape from local optimums and then to obtain a global optimal path. To avoid time consumption with a high burden, authors in [68] proposed a combination of GA and Voronoi method. The Voronoi diagrams are constructed by using fuzzy c-means clustering method to generate the first generation, which accelerates the convergence speed. RRT obtains the merits of random exploring, which can enable escaping of local optimum. Authors in [69] employed RRT to generate the first generation of chromosomes to achieve global optimality.

7.1.2. Ant Colony Optimization. Animals such as ants could manage to establish shortest path from their colony to the feeding source and back home by group cooperation; researchers mimic the behavior and proposed Ant Colony Optimization (ACO) method. ACO introduces two basic concepts, which are "intensity of trail" and "visibility" to form the transition probability which at last decides which way to go, thus to formulate the shortest path.

"Intensity of trail" on edge (i, j) at time $t + n$ is expressed by the following formula:

$$T_{ij}(t + n) = \rho \cdot T_{ij}(t) + \Delta T_{ij}, \quad (26)$$

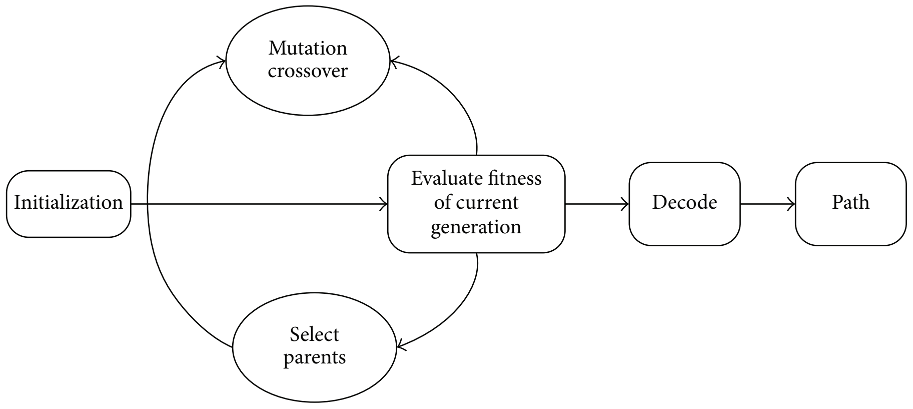


Figure 12: Evolution process for path planning.

where $T_{ij}(t+n)$ represents the information about how many ants in the past have chosen the same edge (i, j) and ρ is a weight to represent how much information is left between t and $t+n$:

$$\Delta T_{ij} = \sum_{k=1}^{m} \Delta T_{ij}^{k},$$
$$\Delta T_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if } k\text{th ant uses the edge } (i, j) \\ 0 & \text{if otherwise}, \end{cases} \quad (27)$$

where ΔT_{ij} is the sum of all m ants' "pheromone" laid on the edge (i, j) between time t and $t+n$, ΔT_{ij}^{k} is the kth ant "pheromone" laid, Q is a constant, and L_k is the kth ant tour length.

"Visibility" on edge (i, j) can be described as

$$\eta_{ij} = \frac{1}{d_{ij}}, \quad (28)$$

where d_{ij} is the Euclidean distance between i and j and η_{ij} determines the degree of how close the state i is to state j.

ACO aims to find the best path by evaluating the pheromone density in each step. The process runs flexibly in dynamic environments only needing to change the representation of "intensity of trail" of a certain edge. The algorithm is proved to be able to deal with multiobjective path planning problems, and it is also able to tackle continuous planning problems [110]. ACO is now widely implemented in 3D environment [7, 73] for path planning. However, it must be emphasized that the basic ACO is not applicable to handle vast size of pheromone matrix in practical time with computer memory limitation. Thus it is validated in simulation, but not practical in real time planning situations in most cases.

Authors in [7] proposed a differential evolution (DE) ACO method by applying differential evolution to optimize the pheromone tail. DE is a simple population-based algorithm; it tends to be more efficiency; thus the work takes advantage of DE crossover operation to achieve faster convergence. Method proposed in [73] started with relative coordinates, which can avoid rotation transformation for UAV, to reduce time consumption for generating an optimal path. Saber and Alshareef [74] came up with using A* to increase the local searching ability and introduced probabilistic nearest neighbor method to estimate the pheromone intensity; it is proved to be effective.

The other three Evolutionary Algorithms, Memetic Algorithm [90], Particle Swarm Optimization [91], and Shuffled Frog Leaping Algorithm [93], all almost share the same exploring process, and their shortcomings and advantages are also almost the same. Although compared to GA and ACO they have a lower degree of implementation, still a lot of works [4, 111, 112] have been done with these methods.

7.2. Neural Network. Another subcategory of bioinspired path planning method is Neural Network (NN). NN was introduced first introduced by Glasius et al. [113] to avoid obstacle as well as navigation and then became popular and implemented to path planning in various areas [114–116]. NN approach aims to generate a dynamic landscape in a neural-like form. It shares some merits as Artificial Potential Field method; the unsearched areas attract the robot in the entire space globally. A typical shunting equation is used to express the dynamics of a robot in neuron network, which is represented in the following form:

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^-, \quad (29)$$

where x_i denotes the neuron activity of the ith neuron. Parameters A, B, and D are nonnegative constants representing the passive decay rate and the upper and lower bounds of the neural activity, respectively. $[I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+$ and $[I_i]^-$ are the excitatory and inhibitory inputs. I_i is the external input to the ith neuron, and it is defined as $I = E$ if it is a safe unexplored area; $I = -E$ if it is an obstacle area; $I = 0$ means other cases. Here $E \gg B$ is a very large positive constant.

For NN, in each step it first chooses the maximal neural activity among the neighbor neurons; then the next location

Table 5: Analysis of bioinspired algorithms.

Method type	Shortcoming		Advantages
	Shortcomings	Improvement	
GA	High time complexity	[68, 69]	Able to solve NP-hard and multiobjectives problems
	Premature convergence	[70–72]	
ACO	High time complexity	[7, 73, 74]	Able to deal with multiobjectives and continuous planning problems
PSO	High time complexity Premature convergence	[75]	It acts faster than GA and can deal with a low number of individuals problems
	Parameter sensitive	[76]	
SFLA	High time complexity	[77]	It is more efficient than PSO and can achieve global convergence faster
	Parameter sensitive	[78]	
MA	High time complexity	[79]	It is more efficient than GA in path smoothness and with low computational complexity
NN	High time complexity	[80–82]	Stable under sudden changes in the network
	Relying on suitable rules and organisms	[83]	

is determined by this maximal neural activity. Equation (29) guarantees that the positive neural activity can propagate to all the free unexplored space with the maximal neural activity, but negative activity only stays locally. Although NN attracted so much attention, it shares the weakness as other bioinspired algorithms that it cannot form canonical rules and model; thus the reliability and time consumption are not guaranteed. Even though Hopfield model is introduced to solve the weakness [83], this is only applicable to certain problems.

Kassim and Kumar [80] propounded a new "wave expansion neural network (WENN)" which introduces grid potential for path planning; the method was implemented to 3D workspace and proved to have time efficiency. Recently Kroumov et al. [81, 82] improved the method by combination with potential field methods; this method accelerates the speed of basic NN. But according to [80], it clearly can be seen that when extended to 3D environments, the number of neighbor neurons will increase to 26, which means the computation complexity will explode simultaneously. Thus it is applicable to implement real time.

7.3. Analysis. Bioinspired algorithms stem from mimicking the natural behavior; they specify a set of optimizing rules as well as the model of processing the information. Then the whole process executes based on the rules and models by iterative optimization. Thus the final results rely much on the rules and model proposed, and the time consumption cannot be guaranteed. Often if the environment becomes more complex, they need a large computational resource to find a solution to meet the expectation (Table 5).

EA is named as a kind of population-based algorithm; all of its subcategories share almost the same procedures: reproduction, mutation, recombination, and selection. Although this kind of algorithms tends to be time consuming and premature convergence, it can deal with multiobjectives problems and solve NP-hard problems. GA is the most popular Evolutionary Algorithm, and PSO is proved to act much faster than GA, but PSO is sensitive to parameters. PSO can be solved by hybrid approach [76] but also holds the same shortcoming as GA. ACO can find optimal path without premature convergence, but it is impossible for the basic ACO to deal with real complex environment. SFLA operates the local exploration by using PSO-like method, but it shuffles virtual frogs periodically with ensuring global exploration. The PSO-like method relies strongly on suitable parameters [78]. MA is sometimes called hybrid GA, which differs from GA in that MA shares chunk of the gene rather than a neat crossover in chromosomes. Since MA coupled with a learning procedure to perform local refinements, it still has a more heavy computational burden. NN is not sensitive to environment changes, but it relies too much on its model.

8. Multifusion Based Algorithms

Multifusion based algorithms manage with problems where usually a single approach proposed cannot work to find an optimal path individually. When faced with unknown environments, with whether dynamic threat or static threat, not all traditional single planning approaches can fulfill the task of obtaining cost minimum or fast convergence or computational efficiency. For example, PRM cannot generate an optimal path itself, which is the same as Voronoi. Potential field methods often jump into local minimum; node based optimal algorithms need preknown environmental skeleton information. Mathematic model based algorithms tend to be time consuming and unable to solve NP-hard problem with varying environments; bioinspired algorithms vary their performance with the model diagram and have heavy time complexity. Thus researchers try to introduce a combination of different approaches to form a fast searching and global optimal algorithm.

Yan et al. [6] used 3D grid to represent the environment and 3D PRM to form a road map in obstacle free space; at last A^* optimal search algorithm combined to achieve an optimal path. Masehian and Amin-Naseri [39] introduced a visibility graph, Voronoi diagram, and potential field (VVP) integrated algorithm. If the extension of VVP algorithm to 3D space is investigated, it shows effective tradeoff between the shortest and safest path. Schøler et al. [85] combined visibility graph and Dijkstra's algorithm (or Geodesics) to find an optimal solution for path planning problem in 3D. Jaishankar and Pralhad [84] planned outdoor environment path by citing GIS-MCDA approach which first synthesizes a variety of information to generate a "combined gray level image"; then an optimal searching algorithm as A^*, D^*, or EA is used to achieve an optimal path. A hybrid of mathematic model based algorithms with EA [117] is proposed to solve NP-hard problem, where general EA tend to be premature. A lot of works adopt this idea; this paper gives a pioneering work of coming up with a taxonomy of this kind of methods.

Based on the principle of each algorithm, this paper classifies all of these multifusion based algorithms into two categories: (a) Embedded Multifusion Algorithms (EMA) and (b) Ranked Multifusion Algorithms (RMA). EMA combines several algorithms' advantages together; these algorithms work in simultaneous model, thus ensuring a tightly coupled approach to achieve better performance. RMA means each algorithm works in separate level, where they work rankly. Table 6 illustrates several typical algorithms of each category.

Table 6: Typical elements of multifusion based algorithms.

Name of subcategory	Typical elements of each subcategory
Embedded multifusion algorithms	RDR [34], VVP [39],
	Voronoi potential field algorithms [39, 40],
	neural network potential field algorithms [80–82],
	hybrid bioinspired [7, 74, 76, 78], and so forth
Ranked multifusion algorithms	PRM node based optimal algorithms [6],
	GIS-MCDA algorithms [84],
	visibility graph node based optimal algorithms [85],
	visibility graph geodesics algorithm [85],
	sampling based EA [68, 69], and so forth

9. Analysis and Conclusion

This paper analyzes a certain aspect of providing a comprehensive knowledge of 3D path planning, including canonical definition and basic knowledge of each kind of algorithms as well as applicable area. By analyzing all these algorithms which have already proved to be successful, this paper classifies all the approaches into five categories: sampling based algorithms, node based optimal algorithms, mathematic model based algorithms, bioinspired algorithms, and multifusion based algorithms. This paper first lists the elements of each category and then supports a critical discussion of each algorithm. During each discussion process, this paper answers four important questions, respectively: (a) what is the taxonomy of these 3D path planning algorithms? (b) How do these algorithms work in order to find the path? (c) Why is it suitable or not suitable for such environment?

According to the analysis we supported, several conclusions can be drawn as illustrated in Table 7, with the details of time complexity, static (S) or dynamic (D) environmental applicability, and real time applicability.

(1) Sampling based algorithms all share the merits of using an initial guess; they differ in how to do postprocessing to ensure completeness or optimum. The random initial guess ensures escaping of local minimum, and this kind of algorithms does not rely much on environmental representation. This kind of algorithms can be further classified as active and passive, where active algorithm can find the optimal path by their own, but passive algorithms cannot. These approaches are appropriate for on-line implementation as they have a high time efficiency, with the ability to handle dynamic and static threats.

(2) Node based optimal algorithms are grid based exploring algorithms; they commonly tackle with node/arc information which can transform distance between nodes/arcs into weight. They originate from dynamic programming approaches and thus cannot further optimize the result beyond the decomposition of the environment. The results of this kind of algorithms rely much on the preconstructed graph and can be combined with other methods to achieve global optimal.

(3) Mathematic model based algorithms aim to describe the whole workspace in a mathematic form, with the advantage of describing all constraints with differential equations. They can easily represent dynamic constraints and kinematic constraints. This kind of algorithms gives an overall consideration to safety, reliability, and efficiency and then bounds the cost function tightly. Although this kind of methods loads a heavy computational burden on computer, it can perform well enough with the improvement of computer technology.

(4) Bioinspired algorithms import heuristic idea and can excellently deal with complex and dynamic unstructured constraints as well as NP-hard problem. This kind of algorithms optimizes the path by mutation (iterative optimization), but the process also brings problem simultaneously. The mutation process repeats until final goal is achieved; almost all bioinspired algorithms have the shortcoming of having a long dealing period. Thus this kind of algorithms is suggested to work off-line, even though they can handle dynamic threats.

(5) Multifusion based algorithms synthesize several algorithms' advantages together to achieve global optimal and cost minimum. This kind of algorithms imports the idea of complementation, that is, merging the merits of several algorithms. These algorithms have ability to achieve several goal simultaneously; thus it often happens that sometimes several simple relative methods combine to form a rather well performed method. These methods are designed to work real time, with strong environmental adaption.

3D path planning approaches explode these years, but problems such as real time planning, complete information expressing, and complex environments modeling still have not been completely solved yet. According to Section 8,

Table 7: Properties of each kind of methods.

Method	Time complexity	S/D environment	Real time
Sampling based algorithms	$O(n\log n) \leq T \leq O(n^2)$	S and (Part) D	On-line
Node based algorithms	$O(m\log n) \leq T \leq O(n^2)$	S and (Part) D	On-line
Mathematic model based algorithms	Depending on the polynomial equation	S and D	Off-line
Bioinspired algorithms	$O(n^2) \leq T$	S and (Part) D	Off-line
Multifusion based algorithms	$O(n\log n) \leq T$	Depending on the algorithms	On-line

this paper recommends a fusion of multipath planning and environment modeling methods; this may become a key direction for 3D path planning under complex situations, and also control uncertainty is recommended to be included.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is partially supported by National Science and Technology Support under Grant no. 61503369 and Grant no. 61528303, which are highly acknowledged. High acknowledgment goes also to Professor D. Theilliol [118], who really provided so much guiding of how to construct a nice work during his visiting time.

References

[1] G. INC, Google self-driving car project, https://www.google.com/selfdrivingcar/.

[2] iRobot INC, "Vacuum cleaning robot," http://www.irobot.com/For-the-Home/Vacuum-Cleaning/Roomba.aspx.

[3] N. K. Yilmaz, C. Evangelinos, P. F. J. Lermusiaux, and N. M. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.

[4] M. P. Aghababa, "3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles," *Applied Ocean Research*, vol. 38, pp. 48–62, 2012.

[5] R. Yue, J. Xiao, S. L. Joseph, and S. Wang, "Modeling and path planning of the city-climber robot part II: 3D path planning using mixed integer linear programming," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '09)*, vol. 6, pp. 2391–2396, Guilin, China, December 2009.

[6] F. Yan, Y.-S. Liu, and J.-Z. Xiao, "Path planning in complex 3D environments using a probabilistic roadmap method," *International Journal of Automation and Computing*, vol. 10, no. 6, pp. 525–533, 2013.

[7] H. Duan, Y. Yu, X. Zhang, and S. Shao, "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," *Simulation Modelling Practice and Theory*, vol. 18, no. 8, pp. 1104–1115, 2010.

[8] F. Schler, *3d path planning for autonomous aerial vehicles in constrained spaces [Ph.D. thesis]*, Department of Electronic Systems, Faculty of Engineering and Science, Aalborg University, Aalborg, Denmark, 2012.

[9] H. M. Choset, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, 2005.

[10] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[11] J.-C. Latombe, *Robot Motion Planning*, vol. 124, Kluwer Academic, Norwell, Mass, USA, 1991.

[12] Y. B. Sebbane, *Lighter Than Air Robots: Guidance and Control of Autonomous Airships*, vol. 58, Springer, 2012.

[13] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[14] H. Himawan Triharminto, A. Prabuwono, T. Adji, N. Setiawan, and N. Chong, *UAV Dynamic Path Planning for Intercepting of a Moving Target: A Review*, vol. 376 of *Book Section 18*, Springer, Berlin, Germany, 2013.

[15] F. Yan, Y. Zhuang, and J. Xiao, "3D PRM based real-time path planning for UAV in complex environment," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '12)*, vol. 6, pp. 1135–1140, Guangzhou, China, December 2012.

[16] T. Leenknegt, *Three-Dimensional Path Planning in Complex Environments*, Vakgroep Elektronica en Informatiesystemen, Universiteit Gent, 2013.

[17] A. Mustard, "Underwater environment," http://it.sohu.com/20110512/n280555394.shtml.

[18] S. Flemming, C.-H. Anders la, and B. Morten, *Configuration Space and Visibility Graph Generation from Geometric Workspaces for UAVs*, Book Section 4, American Institute of Aeronautics and Astronautics, 2011.

[19] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," Tech. Rep. 98-11, Computer Science Department, Iowa State University, Ames, Iowa, USA.

[20] 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.

[21] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

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

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

[24] M. Žefran, V. Kumar, and C. B. Croke, "On the generation of smooth three-dimensional rigid body motions," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 576–589, 1998.

[25] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Proceedings of the 20th AAAI Conference on Artificial Intelligence*, Toronto, Canada, 2012.

[26] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13)*, pp. 4054–4061, IEEE, Karlsruhe, Germany, May 2013.

[27] L. Yang, J. Qi, J. Xiao, and X. Yong, "A literature review of UAV 3D path planning," in *Proceedings of the 11th World Congress on Intelligent Control and Automation (WCICA '14)*, pp. 2376–2381, IEEE, Shenyang, China, July 2014.

[28] S. Choudhury, S. Scherer, and S. Singh, "Rrt*-ar: sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '13)*, pp. 3947–3952, IEEE, Karlsruhe, Germany, 2013.

[29] R. Kala, "Rapidly exploring random graphs: motion planning of multiple mobile robots," *Advanced Robotics*, vol. 27, no. 14, pp. 1113–1122, 2013.

[30] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proceedings of the 49th IEEE Conference on Decision and Control (CDC '10)*, pp. 7681–7687, Atlanta, Ga, USA, December 2010.

[31] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic-domain RRTs: efficient exploration by controlling the sampling domain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3856–3861, Barcelona, Spain, April 2005.

[32] N. M. Amato, O. B. Bayazit, and L. K. Dale, "OBPRM: an obstacle-based PRM for 3D workspaces," in *Proceedings of the 3rd Workshop on the Algorithmic Foundations of Robotics on Robotics : The Algorithmic Perspective: The Algorithmic Perspective (WAFR '98)*, pp. 155–168, 1998.

[33] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[34] R. Gayle, A. Sud, M. C. Lin, and D. Manocha, "Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, pp. 3777–3783, San Diego, Calif, USA, November 2007.

[35] Y. Cho, D. Kim, and D.-S. K. Kim, "Topology representation for the voronoi diagram of 3d spheres," *International Journal of CAD/CAM*, vol. 5, no. 1, pp. 59–68, 2005.

[36] D. Kim and D.-S. Kim, "Region-expansion for the Voronoi diagram of 3D spheres," *CAD Computer Aided Design*, vol. 38, no. 5, pp. 417–430, 2006.

[37] D.-S. Kim, Y. Cho, D. Kim, S. Kim, J. Bhak, and S.-H. Lee, "Euclidean voronoi diagrams of 3D spheres and applications to protein structure analysis," *Japan Journal of Industrial and Applied Mathematics*, vol. 22, no. 2, pp. 251–265, 2005.

[38] L. Lifeng and Z. Shuqing, "Voronoi diagram and gis-based 3d path planning," in *Proceedings of the 17th International Conference on Geoinformatics*, vol. 5, pp. 1–5, Fairfax, Va, USA, 2009.

[39] E. Masehian and M. R. Amin-Naseri, "A voronoi diagram-visibility graph-potencial field compound algorith for robot path planning," *Journal of Robotic Systems*, vol. 21, no. 6, pp. 275–300, 2004.

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

[41] T. Roos and H. Noltemeier, *Dynamic Voronoi Diagrams in Motion Planning*, vol. 553 of *Lecture Notes in Computer Science, Book Section 17*, Springer, Berlin, Germany, 1991.

[42] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using Laplace's equation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2102–2106, Cincinnati, Ohio, USA, May 1990.

[43] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[44] S. Sundar and Z. Shiller, "Optimal obstacle avoidance based on the hamilton-jacobi-bellman equation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2424–2429, San Diego, Calif, USA, 1994.

[45] B. Miller, K. Stepanyan, A. Miller, and M. Andreev, "3D path planning in a threat environment," in *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC '11)*, vol. 6, pp. 6864–6869, IEEE, Orlando, Fla, USA, December 2011.

[46] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[47] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[48] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, vol. 1, pp. 968–975, IEEE, Washington, Wash, USA, 2002.

[49] B. C. Williams and R. J. Ragno, "Conflict-directed A* and its role in model-based embedded systems," *Discrete Applied Mathematics*, vol. 155, no. 12, pp. 1562–1595, 2007.

[50] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, pp. 1177–1198, AAAI Press, MIT Press, Vancouver, Canada, 2007.

[51] A. Nash, S. Koenig, and C. Tovey, "Lazy theta*: any-angle path planning and path length analysis in 3D," in *Proceedings of the Third Annual Symposium on Combinatorial Search*, vol. 2, pp. 153–154, Atlanta, Ga, USA, 2010.

[52] A. Stentz, "The focussed d-star algorithm for real-time replanning," in *Proceedings of the International Joint Conference on AI*, vol. 95, pp. 1652–1659, Montreal, Canada, 1995.

[53] L. Niu and G. Zhuo, "An improved real 3d a* algorithm for difficult path finding situation," in *Proceeding of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, Beijing, China, 2008.

[54] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[55] Y. V. Smirnov, "Hybrid algorithms for on-line search and combinatorial optimization problems," *Carnegie-Mellon Univ Pittsburgh Pa Dept of Computer Science*, vol. 142, p. 141, 1997.

[56] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[57] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[58] J. Tisdale, Z. W. Kim, and J. K. Hedrick, “Autonomous UAV path planning and estimation: an online path planning framework for cooperative search and localization,” *IEEE Robotics and Automation Magazine*, vol. 16, no. 2, pp. 35–42, 2009.

[59] A. Richards and J. P. How, “Aircraft trajectory planning with collision avoidance using mixed integer linear programming,” in *Proceedings of the American Control Conference*, pp. 1936–1941, Anchorage, Alaska, USA, May 2002.

[60] C. S. Ma and R. H. Miller, “Milp optimal path planning for real-time applications,” in *Proceedings of the American Control Conference*, p. 6, Minneapolis, Minn, USA, 2006.

[61] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, “Multi-task allocation and path planning for cooperating UAVs,” in *Cooperative Control: Models, Applications and Algorithms*, pp. 23–41, Springer, 2003.

[62] E. Masehian and G. Habibi, “Robot path planning in 3D space using binary integer programming,” *International Journal of Mechanical System Science and Engineering*, vol. 23, pp. 26–31, 2007.

[63] A. Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol, “Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2832–2848, 2012.

[64] R. Bhattacharya, “Optragen: a matlab toolbox for optimal trajectory generation,” in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 6832–6836, San Diego, Calif, USA, December 2006.

[65] Y. Chen and A. L. Schwartz, “RIOTS 95: a MATLAB toolbox for solving general optimal control problems and its applications to chemical processes,” in *Recent Developments in Optimization and Optimal Control in Chemical Engineering*, R. Luus, Ed., pp. 229–252, Transworld Research Publishers, 2002.

[66] I. M. Ross and M. Karpenko, “A review of pseudospectral optimal control: from theory to flight,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.

[67] J. F. Sturm, “Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones,” *Optimization Methods and Software*, vol. 11, no. 1–4, pp. 625–653, 1999.

[68] Y. V. Pehlivanoglu, “A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV,” *Aerospace Science and Technology*, vol. 16, no. 1, pp. 47–55, 2012.

[69] G. Erinc and S. Carpin, “A genetic algorithm for nonholonomic motion planning,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1843–1849, Roma, Italy, April 2007.

[70] C. Fonlupt, P. Preux, and D. Robilliard, “Preventing premature convergence via cooperating genetic algorithms,” in *Proceedings of the ACM Workshop on Foundations of Genetic Algorithms*, pp. 1–6, Citeseer, 1993.

[71] A. Hacioglu and I. Ozkol, “Transonic airfoil design and optimisation by using vibrational genetic algorithm,” *Aircraft Engineering and Aerospace Technology*, vol. 75, no. 4, pp. 350–357, 2003.

[72] A. Hacioĝlu and I. Özkol, “Vibrational genetic algorithm as a new concept in airfoil design,” *Aircraft Engineering and Aerospace Technology*, vol. 74, no. 3, pp. 228–236, 2002.

[73] Y. Chen, X. Zhao, C. Zhang, and J. Han, “Relative coordination 3D trajectory generation based on the trimmed ACO,” in *Proceedings of the International Conference on Electrical and Control Engineering (ICECE ’10)*, vol. 1, pp. 1531–1536, Wuhan, China, June 2010.

[74] A. Y. Saber and A. M. Alshareef, “Scalable unit commitment by memory-bounded ant colony optimization with A^* local search,” *International Journal of Electrical Power and Energy Systems*, vol. 30, no. 6-7, pp. 403–414, 2008.

[75] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, “Particle swarm optimization: basic concepts, variants and applications in power systems,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[76] J. Tang, J. Zhu, and Z. Sun, “A novel path planning approach based on appart and particle swarm optimization,” *Advances in Neural Networks*, vol. 3498, pp. 253–258, 2005.

[77] H. Pu, Z. Zhen, and D. Wang, “Modified shuffled frog leaping algorithm for optimization of UAV flight controller,” *International Journal of Intelligent Computing and Cybernetics*, vol. 4, no. 1, pp. 25–39, 2011.

[78] A. Rahimi-Vahed and A. H. Mirzaei, “A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem,” *Computers and Industrial Engineering*, vol. 53, no. 4, pp. 642–666, 2007.

[79] L. Buriol, P. M. França, and P. Moscato, “A new memetic algorithm for the asymmetric traveling salesman problem,” *Journal of Heuristics*, vol. 10, no. 5, pp. 483–506, 2004.

[80] A. Kassim and B. Kumar, “A neural network architecture for path planning,” in *Proceedings of the International Joint Conference on Neural Networks (IJCNN ’92)*, vol. 2, pp. 787–792, IEEE, Baltimore, Md, USA, 1992.

[81] V. Kroumov and J. Yu, “3D path planning for mobile robots using annealing neural network,” in *Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC ’09)*, pp. 130–135, Okayama, Japan, March 2009.

[82] V. Kroumov, J. Yu, and K. Shibayama, “3D path planning for mobile robots using simulated annealing neural network,” *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 7, pp. 2885–2899, 2010.

[83] C. W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, “Shortest path routing algorithm using Hopfield neural network,” *Electronics Letters*, vol. 37, no. 19, pp. 1176–1178, 2001.

[84] S. Jaishankar and R. N. Pralhad, “3D off-line path planning for aerial vehicle using distance transform technique,” *Procedia Computer Science*, vol. 4, pp. 1306–1315, 2011.

[85] F. Schøler, A. la Cour-Harbo, and M. Bisgaard, “Generating approximative minimum length paths in 3D for UAVs,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV ’12)*, pp. 229–233, Madrid, Spain, June 2012.

[86] O. Khatib, *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, Book Section 29, Springer, New York, NY, USA, 1990.

[87] R. Deits and R. Tedrake, “Footstep planning on uneven terrain with mixed-integer convex optimization,” in *Proceedings of the 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids ’14)*, pp. 279–286, Madrid, Spain, November 2014.

[88] V. A. Luchnikov, N. N. Medvedev, L. Oger, and J.-P. Troadec, “Voronoi-Delaunay analysis of voids in systems of nonspherical particles,” *Physical Review E*, vol. 59, no. 6, pp. 7205–7212, 1999.

[89] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[90] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," *Parallel Computing and Transputer Applications*, vol. 1, pp. 177–186, 1992.

[91] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, section 630, pp. 760–766, Springer US, 2010.

[92] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[93] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[94] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[95] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration space for fast path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2138–2145, May 1994.

[96] M. I. Shamos and D. Hoey, "Closest-point problems," in *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, pp. 151–162, Berkeley, Calif, USA, 1975.

[97] F. Sharifi, Y. Zhang, and B. W. Gordon, "Voronoi-based coverage control for multi-quadrotor UAVs," in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE '11)*, vol. 6, pp. 991–996, Washington, DC, USA, August 2011.

[98] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '10)*, pp. 1997–2004, Anchorage, Alaska, USA, 2010.

[99] F. B. Zhan and C. E. Noon, "Shortest path algorithms: an evaluation using real road networks," *Transportation Science*, vol. 32, no. 1, pp. 65–73, 1998.

[100] L. Verscheure, L. Peyrodie, N. Makni, N. Betrouni, S. Maouche, and M. Vermandel, "Dijkstra's algorithm applied to 3D skeletonization of the brain vascular tree: evaluation and application to symbolic," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '10)*, pp. 3081–3084, Buenos Aires, Argentina, September 2010.

[101] M. Wan, Z. Liang, Q. Ke, L. Hong, I. Bitter, and A. Kaufman, "Automatic centerline extraction for virtual colonoscopy," *IEEE Transactions on Medical Imaging*, vol. 21, no. 12, pp. 1450–1460, 2002.

[102] I. A. Musliman, A. A. Rahman, and V. Coors, "Implementing 3d network analysis in 3d-gis," in *Proceedings of the 21st ISPRS Congress Silk Road for Information from Imagery*, vol. 8, pp. 113–120, Beijing, China, 2008.

[103] L. De Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for UAVS in 3D environments," *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1–4, pp. 247–264, 2012.

[104] C. Tovey, S. Greenberg, and S. Koenig, "Improved analysis of D^*," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3371–3378, Taipei, Taiwan, September 2003.

[105] C. L. Shih, T.-T. Lee, and W. A. Gruver, "A unified approach for robot motion planning with moving polyhedral obstacles," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 4, pp. 903–915, 1990.

[106] K. Culligan, M. Valenti, Y. Kuwata, and J. P. How, "Three-dimensional flight experiments using on-line mixed-integer linear programming trajectory optimization," in *Proceedings of the American Control Conference (ACC '07)*, vol. 6, pp. 5322–5327, New York, NY, USA, July 2007.

[107] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance Scenarios," *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2–4, pp. 190–216, 2010.

[108] C. Tricaud and Y.-Q. Chen, "Linear and nonlinear model predictive control using a general purpose optimal control problem solver riots 95," in *Proceedings of the Chinese Control and Decision Conference*, pp. 1552–1557, Yantai, China, 2008.

[109] J. Dong and V. Juris, *Parallel Evolutionary Algorithms for UAV Path Planning*, section 1, American Institute of Aeronautics and Astronautics, 2004.

[110] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*, pp. 36–39, Springer US, 2010.

[111] S. Kundu and D. R. Parhi, "Modified shuffled frog leaping algorithm based 6DOF motion for underwater mobile robot," *Procedia Technology*, vol. 10, pp. 295–303, 2013.

[112] F. Jung Leng, K. Jared, O. James, and W. Eliot, *Three-Dimensional Path Planning of Unmanned Aerial Vehicles Using Particle Swarm Optimization*, American Institute of Aeronautics and Astronautics, 2006.

[113] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Networks*, vol. 8, no. 1, pp. 125–133, 1995.

[114] A. A. Kassim and B. V. K. V. Kumar, "A neural network architecture for path planning," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 787–792, Baltimore, Md, USA, 1992.

[115] C. Luo and S. X. Yang, "A real-time cooperative sweeping strategy for multiple cleaning robots," in *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 660–665, Vancouver, Canada, October 2002.

[116] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 718–725, 2004.

[117] X. Zhang, H. Duan, and Y. Yu, "Receding horizon control for multi-UAVs close formation control based on differential evolution," *Science China Information Sciences*, vol. 53, no. 2, pp. 223–235, 2010.

[118] D. Theilliol, http://page-perso.cran.uhp-nancy.fr/sitejoomla/.