Journal of Network Intelligence ©2022 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **7**, Number **1**, February **2022**

GSGC:An Improved Path Planning Optimization Method Using Guided Sampling and Gradual Cutting

Zhi-Ming Cai*

School of Electronic, Electrical Engineering and Physics
Fujian University of Technology
No.33 Xuefu South Road, Fuzhou, 350118, China
caizm@163.com

Jian Lu

School of Electronic, Electrical Engineering and Physics
Fujian University of Technology
No.33 Xuefu South Road, Fuzhou, 350118, China
573843470@qq.com

Yu-Feng Ling

School of Electronic, Electrical Engineering and Physics
Fujian University of Technology
No.33 Xuefu South Road, Fuzhou, 350118, China
1504662829@qq.com

Tian-Jian Li

National Demonstration Center for Experimental Electronic Information and Electrical Technology Education
Fujian University of Technology
No.33 Xuefu South Road, Fuzhou, 350118, China
ltj@fjut.edu.cn

Lin Xu

STEM
University of South Australia
Mawson Lakes Campus, Adelaide, 5095, Australia
xuyly032@mymail.unisa.edu.au

Received September 2021; revised January 2022
(Corresponding Author: caizm@163.com)

Abstract. *Path planning of mobile robot has become a research hotspot in the fields of automatic control, computer and artificial intelligence. The sampling-based method is one of the most popular methods for path planning, among which BIT*(Batch Informed Trees), a variant of RRT, is a typical one. However, it has to traverse the space to find the first path, and will generate some redundant points which bringing a lot of redundant angles. As BIT* is not optimal and the convergence is not fast enough, a new sampling-based method, GSGC, is proposed to overcome these shortcomings in this paper. It adds an alterable guided sampling function to increase sampling efficiency. To remove redundant points, a gradual cutting function is presented to reduce the length of path and improve processing efficiency. During pruning, the elliptical area is shrunk to reduce the sampling space which improves the performance. The experimental results show that the GSGC can spend less time to get an optimal solution with faster convergence than BIT* and RRT*.*

Keywords: Path Planning, Guided Sampling, Gradual Cutting, Optimization Algorithm

1. INTRODUCTION.

In recent years, the path planning problem of mobile robot has attracted many experts to study [1, 2, 3, 4, 5]. The path planning of mobile robot aims to plan a collision-free path from the initial point to the goal point satisfying the constraints of the robot in the working environment with obstacles [6]. Path planning is divided into two aspects: local search and global search.

There are two typical local search methods: DWA (Dynamic Window Approach) and APF(Artificial Potential Field) [7]. The DWA considers robot motion performance, samples multiple sets of velocities in the velocity space and uses trajectory evaluation function to get the optimal solution. The APF, proposed by Khatib, gives mobile robot a join force which is composed of an attractive force of the goal point and a repulsive force of obstacle areas. Nevertheless, they are both easy to fall into the local minimum problem [8].

The algorithms of global search mainly include graph-based methods, metaheuristic algorithms and sampling-based methods [9]. Some graph-based methods, such as Dijkstra's algorithm [10] and A* [11], are based on greedy strategy to get the optimal solution and use dynamic programming [12] to precisely solve discrete approximation of the problem. Compared with A*, the Dijkstra's algorithm has no directionality so that it is low efficient [13]. The metaheuristic algorithms, which are based on swarm intelligence heuristic method and natural phenomenon heuristic method, are also an important branch of path planning. They can solve various optimization problems effectively without using mathematical formulas [14]. There are some classic metaheuristic algorithms, such as PSO(Particle Swarm Optimization) [15], GA(Genetic Algorithm) [16] and DE(Differential Evolution) [17]. A new algorithm, CSO(Cat Swarm Optimization) [18], imitates the behaviors of cats to improve PSO. It uses Seeking Mode and Tracing Mode to make PSO globally optimal. Shih et al. proposed Hybrid GA(Hybrid-Genetic Algorithm) [19] to improve GA. They developed an EEF(Exponential Effective Function) as a fitness function to solve the GPP(Glider Path-planning Problem). Dao et al. were inspired by the pollination of bees and proposed an algorithm to get a minimization plan [20]. Compared with the above, the DE algorithm is powerful and easy to implement [21]. QUATRE-EAR [22] is an enhanced structure of DE with less control parameters which greatly accelerates computing speed. Zhang et al. proposed a short-term traffic flow prediction algorithm, QGA-LVQ [23], to forecast the changes of traffic flow in urban networks, which was applied in smart city management. Song et al. proposed the Phasmatodea population evolution algorithm (PPE) to deal with optimization problems in the N-dimensional decision space [24], which had the potential to tackle complex path planning.

Others, known as informed algorithms, such as A* and IAFMT*(Informed Anytime Fast Marching Tree) [25], use heuristic to estimate the cost of the solution and have been widely studied. The sampling-based methods, which have the properties of probabilistic completeness and asymptotic optimality, can find the solution if there exists a feasible path. And they especially perform well in high-dimensional space. The typical representatives are PRM(Probabilistic Roadmaps) [26] and RRT(Rapidly-exploring Random Trees) [27]. In the early 1990s, Overmars et al. proposed a multiple query algorithm, PRM. It uses random points to construct a roadmap in the free space of a given map and connect them with each other. As PRM requires prior knowledge of the environment which is not always known in advance in real world, it does not always work well. The RRT, a landmark algorithm [28], is proposed as a single query algorithm which constructs a tree from the start point and explores the space by growing the tree using random points. It has better efficiency than PRM. Subsequently, many algorithms are proposed to improve RRT. RRT-connect [29], which a double tree is added to RRT, uses a greedy heuristic to speed up connection between vertexes. It builds two trees simultaneously, one from start point, and the other from goal point. Since neither RRT nor RRT-connect considers the cost of the solution found, neither of them is optimal. Karaman et al. proposed RRT* [30] which adopts the ChooseParent and Rewire procedures as an optimization model to find an optimal path. Yi et al. proposed Homotopy variant algorithm, HARRT*(Homotopy Aware RRT*) [31], which is on the basis of B-RRT*(Bidirectional RRT*) [32]. The algorithm uses artificial interference to plan the path from one topological space to another. However, when the number of nodes is too large, the memory consumption and calculation amount of the algorithm will increase exponentially [33]. Nasir et al. proposed RRT*-Smart [34] which applies intelligent sampling and path optimization function to improve the problem of slow convergence of RRT*. Another RRT# is proposed by Arslan et al. [35] The nodes with low weight in the random tree are selected through global reprogramming and the local shortest path segments are generated as part of the global path. The pure exploration is the main reason for the slow convergence rate of RRT* [36]. Therefore, the APF is advised to solve this problem. The algorithm, P-RRT*, which the APF is added to RRT* [37], gives a direction for samples generation.

Subsequently, the concept of graph search is introduced into the sample-based planner in some algorithms. The Anytime RRTs applies heuristics to improve the search by biasing sampling [38]. Inspired from above, the SBA* (Sampling-based A*) [39] is designed by improving RRT* with A*. The sampling heuristic method of node rejection has attracted a lot of attention. Informed RRT* [40] which inspired by node rejection uses the information of the current solution to create an elliptical area to constrain the sampling space and improve the performance significantly than RRT* [41]. Informed RRT*-connect [42] uses the concept of Informed RRT* and RRT-Connect to build double trees to offer low-cost solutions with fewer iterations. BIT*(Batch Informed Trees*) [43] handles samples with batches and uses the evaluation function to sort vertexes and edges. Though it can find a better solution than Informed RRT*, there are still some problems in BIT*. One is BIT* has to traverse the entire space which takes more time when finding the first path. And sometimes, it will generate a lot of bending angles. As a result, the BIT* is not optimal and the convergence is not fast enough.

This paper proposes an improved algorithm, GSGC (Guided Sampling and Gradual Cutting), for robot path planning. The algorithm introduces guided sampling, gradual cutting function and tunable parameters of the ellipse area. The attractive potential field is used to let the goal point guide the direction of the new sampling in the tree model. The GSGC can find the first path with fewer iterations and find the path quickly in the complex environment. Meanwhile, it can quickly converge to the optimal state, especially in high dimensions.

The rest of this article is organized as follows. Section II gives some problem definitions; Section III introduces some prerequisites of the algorithm and describes the process of the algorithm; Section IV proves some properties; Section V presents the simulation results; Section VI makes a summary and explains the future work.

2. **PROBLEM DEFINITIONS.**

To explain the principle of the algorithm, some definitions are given similarly to [30]: Let $X \subseteq R^n$ be the configuration space of the path planning problem, where $n \in N$, $n \geq 2$. Let $X_{obs} \subset X$ be the obstacle area, and $X_{free} = X \backslash X_{obs}$ represent the obstacle-free area. Let $x_{init} \in X_{free}$ be the initial point, $x_{goal} \in X_{free}$ be the goal point, $\sigma : [0,1] \rightarrow X$ be a sequence of states, called a path.

A path planning is to find a feasible path satisfying $\sigma : [0,1] \rightarrow X$, if and only if $\sigma(0) = x_{init}$ and $\sigma(1) = x_{goal}$. The problem is described as follows.

Problem 1(Feasible Planning). For a problem of path planning, if a feasible path can be found, there is feasible planning, or reports failure.

Problem 2(Optimal Planning). The optimal planning is to find a path with minimal cost.

Problem 3(Fast Planning). The fast planning is to find a feasible path with the minimal time.

A discrete set of points in the X can be seen as an implicit graph. A probabilistic model called RGG (random geometric graph) is used to describe the properties of the graph when these points are randomly sampled. The relative geometric positions of vertexes determine the edges between vertexes in an RGG. Sampling-based methods can be regarded as algorithms to build an implicit RGG model and explicit tree [44].

3. **GUIDED SAMPLING AND GRADUAL CUTTING(GSGC).**

3.1. **Overview of the proposed algorithm.**

In this paper, we define an RGG model with edges using random samples including the initial point and goal point in the free space. The parameter, r, in the RGG model is chosen to reduce the complexity of the graph and keep asymptotic optimality [45]. A heuristic is used to build an explicit tree from the initial point to goal point (Figure 1(a)). The edges in the explicit tree are collision-free. When the feasible path is found or no longer expanded, it will conclude a batch (Figure 1(b)). Subsequently, more samples are added to construct a denser RGG model, which will get a better solution than that has been found. Next, the style of LPA* [46] is used to update the explicit tree(Figure 1(c)). The tree model stops growing when there are no better solutions, no more collision-free edges, or reaches maximum iterations (Figure 1(d)).

In Figure 1, the initial point is shown as green, while the goal point is red. The gray lines and dots in the ellipse area contain better solutions. The feasible path is shown as a red line in the current state. (a) shows the first batch of samples that the GSGC deals with. (b) shows the first feasible path that the GSGC has found. Then in the ellipse area, the GSGC begins to deal with the next batch of samples and finds the better feasible path in the current state as shown in (c) and (d).

3.2. **Preliminaries.**

Let the function $g'(x)$ represent the admissible estimate of cost-to-come between the initial point x_{init} and the current point $x \in X$. Let the function $h'(x)$ represent the admissible estimate of cost-to-go between the current point x and the goal point x_{goal}. The function $f'(x)$ denotes an estimate of the movement cost of the path from the initial point x_{init} through the current point x to the goal point x_{goal} , namely the

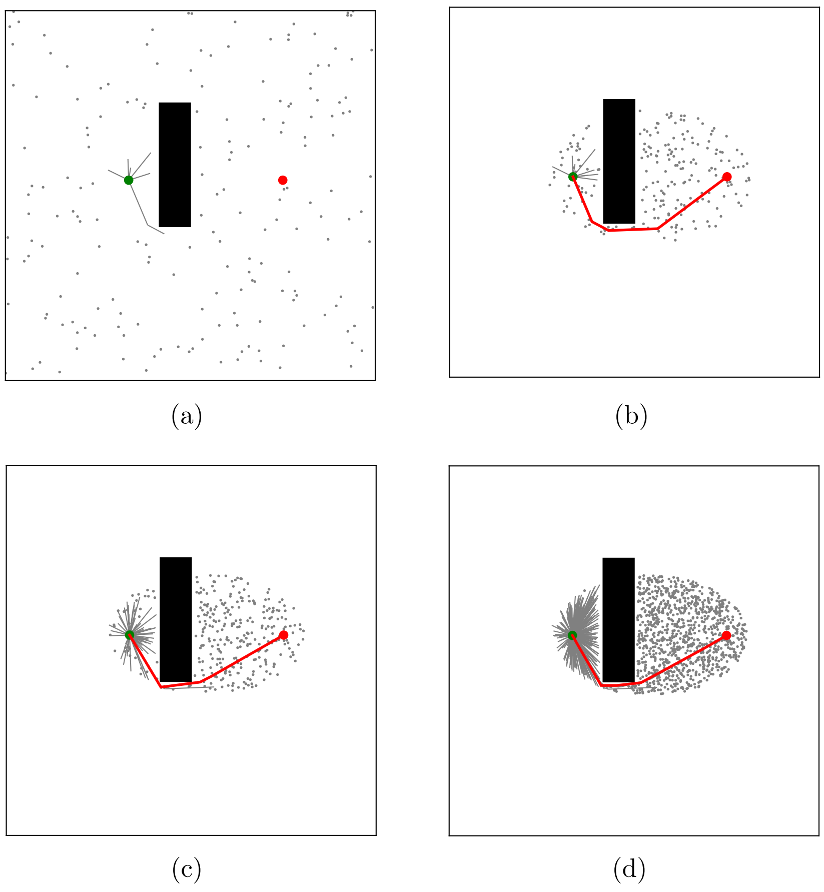


FIGURE 1. The major process of GSGC.

sum of $g'(x)$ and $h'(x)$. A subset of vertexes which can improve the current best solution cost, c_{best}, is given as follows

$$X_{f'} = \{x \in X \mid f'(x) \leq c_{best}\} \tag{1}$$

The explicit tree model is defined as $T = (V, E)$. $V \subset X_{free}$ are vertexes in the tree model, and $E = \{(v, w)\}$ are edges that connect two points $v, w \in V$ in the tree model.

The function $g_T(x)$ represents the actual cost-to-come between the initial vertex and the current state vertex $x \in X$ in the current tree model T. We suppose that a vertex not in the current tree or unreachable has a cost of infinity.

Any state vertex satisfies the formula as follows

$$g'(x) \leq g(x) \leq g_T(x) \tag{2}$$

where $g(x)$ is actual optimal cost of the current $x \in X$.

The function $c'(x, y)$ and $c(x, y)$ represent the estimate of the cost and the actual cost of the edge between the state vertexes $x, y \in X$, respectively. If the edge intersects the obstacle, it has a cost of infinity.

So any $x, y \in X$ satisfies the following formula

$$c'(x, y) \leq c(x, y) \leq \infty \tag{3}$$

However, it is expensive to calculate $c(x, y)$, because it needs differential constraints, collision detection and so on.

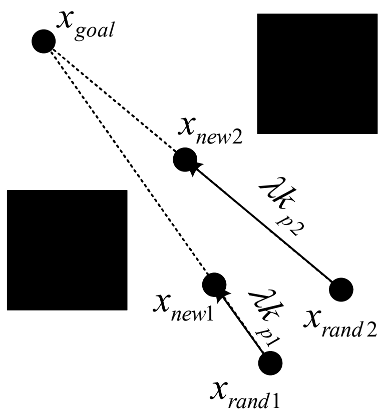


Figure 2. The diagram of guided sampling.

3.3. **Procedure of the proposed algorithm.** The proposed algorithm, GSGC, is presented in Algorithm 1-3. It is similar to BIT* as presented in [43], with the updated lines 5,6,34 in Algorithm 1 and the addition of Algorithm 2-3. For simplicity, the proposed algorithm selects the model of an $r - disc$ RGG. The purpose of the algorithm is to find a feasible path connecting the goal point from a single initial point, and the path can converge to the optimal state.

The algorithm adopts the basic structure of BIT*. It improves the method of sampling, reduces the sampling area and cuts the path to get better performance of planner. It sets a tunable parameter to reduce the sampling area (Algorithm 1; Line 5) and introduces guided sampling function (Algorithm 1, Line 6; Algorithm 2) to change the method of sampling to get more efficient. It can also improve the performance of getting the first solution. When the algorithm stops, $GradualCutting()$ function will remove the redundant vertexes to improve the final solution (Algorithm 1, Lines 34; Algorithm 3).

3.3.1. *Initialization.* At first, the initial point x_{init} and the goal point x_{goal} are put into the set of points V and the set of unconnected samples X_{sample} respectively (Algorithm 1, Line 1). The tree model is grown from x_{init} to x_{goal} by an orderly queue of edges Q_E in RGG model. These edges are composed of a vertexes expansion queue Q_V. The edges queue Q_E, and vertexes queue Q_V are initialized (Algorithm 1, Line 2).

3.3.2. *Guided sampling.* To improve sampling efficiency, a guided sampling function, $GuidSample()$, is constructed. The new sample points are generated by $GuidSample()$ function. The detailed procedure is presented in Algorithm 2.

In the algorithm, we introduce the attractive potential field [7] for sampling. New sample points are generated randomly. Only the points in the ellipse, $eclip$, will be selected and the search direction is guided by the function $DirectGuid()$. Therefore, it can avoid traversing the entire ellipse. We set and increase attractive parameter, k_p, when the obstacle is far from the current position so that the current point can approach the goal quickly. If the obstacle is close, the attractive parameter is reduced. The algorithm can guide the new sample points to move toward the goal, as shown in Figure 2.

$RandGrow(x)$ is random growth function for generating new sample point

$$x_{rand} = RandGrow(x) \tag{4}$$

We introduce the attractive potential field, defined as

$$U = \frac{1}{2} \cdot k_p \cdot \|x_{rand} - x_{goal}\|^2 \tag{5}$$

Algorithm 1 : GSGC($x_{init} \in X_{free}$,$x_{goal} \in X_{free}$)

```
1: $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset; X_{samples} \leftarrow \{x_{goal}\};$
2: $Q_E \leftarrow \emptyset; Q_V \leftarrow \emptyset; r \leftarrow \infty;$
3: repeat
4: if $Q_E \equiv \emptyset$; and $Q_V \equiv \emptyset$; then
5:     $Prune(g_T(x_{goal}));$
6:     $X_{samples} \overset{+}{\leftarrow} GuidSample\,(x_{\text{goal}}\,, m);$
7:     $V_{old} \leftarrow V;$
8:     $Q_V \leftarrow V;$
9:     $r \leftarrow \text{radius}\,(|V| + |X_{samples}|);$
10: end if
11: while $BestqueueCost(Q_V) \le BestqueueCost(Q_E)$ do
12:     $ExpandVertex(Bestinqueue(Q_V));$
13: end while
14: $(v_m, x_m) \leftarrow BestInqueue(Q_E);$
15: $Q_E \overset{-}{\leftarrow} \{(v_m, x_m)\};$
16: if $g_T\,(v_m) + c'\,(v_m, x_m) + h'\,(x_m) < g_T\,(x_{goal})$ then
17:     if $g'\,(v_m) + c\,(v_m, x_m) + h'\,(x_m) < g_T\,(x_{goal})$ then
18:         if $g_T\,(v_m) + c\,(v_m, x_m) < g_T\,(x_m)$ then
19:             if $x_m \in V$ then
20:                 $E \overset{-}{\leftarrow} \{(v, x_m) \in E\};$
21:             else
22:                 $X_{samples} \overset{-}{\leftarrow} \{x_m\};$
23:                 $V \overset{+}{\leftarrow} \{x_m\}; Q_V \overset{+}{\leftarrow} \{x_m\};$
24:             end if
25:         end if
26:     end if
27:     $E \overset{+}{\leftarrow} \{v_m, x_m\};$
28:     $Q_E \overset{-}{\leftarrow} \{(v, x_m) \in Q_E \mid g_T(v) + c'\,(v, x_m) \ge g_T\,(x_m)\};$
29: else
30:     $Q_E \leftarrow \emptyset; Q_V \leftarrow \emptyset;$
31: end if
32: until STOP;
33: $T = (V, E);$
34: return $T' \leftarrow GradualCutting(T);$
```

subject to the constraint

$$k_p = \begin{cases} k_{p1} & if\,(\|x_{obs} - x_{rand}\|) \le d \\ k_{p2} & if\,(\|x_{obs} - x_{rand}\|) > d \end{cases} \tag{6}$$

where $k_{pi}(i = 1, 2)$ are attractive parameters, which are selected according a threshold value, d, the distance between the current position and the nearest obstacle (Algorithm 2, Line 5). The $\|.\|$ is the distance between two points.

The gradient of U is $k_p \cdot (x_{rand} - x_{goal})$, which is the fastest direction to the goal for x_{rand}. Therefore, we define a direction guidance function of vertex x as follows

$$DirectGuid(x) = \lambda \cdot k_p \cdot (x_{rand} - x_{goal}) \tag{7}$$

where λ is random step size.

Algorithm 2 : $GuidSample(x_{goal}, m)$

```
1: $K = 1$;
2: while $K \leq m$ do
3:     $x_{rand} \leftarrow RandGrow(x)$;
4:     if $PointInEllipse(x_{rand}, eclip)$ then
5:         $d \leftarrow dis(x_{rand}, x_{obs})$;
6:         $\vec{F}_{att} \leftarrow DirectGuid\,(x_{rand}, x_{goal}, d)$;
7:         $x_{new} \leftarrow x_{rand} + \vec{F}_{att}$;
8:         $K = K + 1$;
9:     end if
10: end while
```

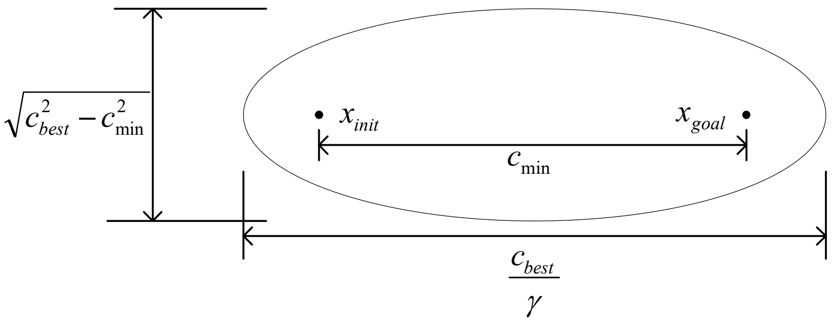


FIGURE 3. The ellipse area.

The new vertex is generated as follows

$$x_{new} = x_{rand} + DirectGuid(x) \tag{8}$$

So far, m new sampling points are generated in a batch. The ellipse is defined in $Prune()$ function, which is detailed as follows.

3.3.3. *Building of Ellipse in Pruning.* A new batch (Algorithm 1, Lines 4-10) begins when the queues are empty. During pruning, the configuration space is limited to an ellipse area to improve performance obviously [40]. The ellipse is built as shown in Figure 3.

The x_{init} and x_{goal} are focal points of ellipse. In this paper, a tunable parameter, γ, is added to reduce ellipse area. The major axis of ellipse is defined as

$$2a = \frac{c_{best}}{\gamma} \tag{9}$$

where $\gamma = 2k_p$, k_p is attractive parameter.

The minor axis of ellipse is defined as

$$2b = \sqrt{c_{best}^2 - c_{min}^2} \tag{10}$$

where c_{min} is the distance between x_{init} and x_{goal}.

3.3.4. *Edge selection and processing.* The major procedure of edge selection is similar to BIT*. When the movement cost of vertexes and edges are obtained, they will be sorted in ascending order and added to the Q_V and Q_E respectively. While the estimated cost of the best vertex in the queue Q_V is less than the estimated cost of the best edge in the queue Q_E, the vertex will be expanded into the edge queue. The best edge in the queue, (v_m, x_m), is removed for processing. Then the edge processing compares the cost and judges whether the edge (v_m, x_m) is optimal or not. If it does, it needs further processing for x_m and the edge to x_m. If x_m is in the tree model, it needs to remove the edge to x_m. Otherwise, it requires moving x_m from $X_{samples}$ to Q_V and Q_E for expansion.

The process will stop if there are no better solutions, no more collision-free edges or reach maximum iterations. Otherwise, repeat the process above until the algorithm stops. (Algorithm 1, Line 32).

3.3.5. *Gradual Cutting (Algorithm 1, Line 33-34; Algorithm 3).* The process above often generates a lot of bending angles, which brings redundant vertexes in the path. Therefore, we design Algorithm 3 to remove the redundant vertexes and get an optimal solution.

Algorithm 3 : *GradualCutting*()

```
1: $i = 1$;
2: while $i \leq num(V)$ do
3:     $j = num(V)$ ;
4:     while $j \geq i$ do
5:         if $obstacle.intersects(x_i, x_j)$ true then
6:             $j = j - 1$;
7:         else
8:             for each $k = i + 1$ to $j - 1$ do
9:                 if $x_k \notin Q_{RM}$ then
10:                    $Q_{RM} \overset{+}{\leftarrow} x_k$;
11:                end if
12:            end for
13:        end if
14:        $i = j$;
15:    end while
16: end while
17: $E = reconnect(V)$;
18: $T = (V, E)$;
19: return $T$;
```

The gradual cutting operation starts forward from the first vertex, denoted as x_i, and backward from the last vertex, denoted as x_j, to check whether the line between the two vertexes collides with the obstacles or not, as shown in Figure 4. If a collision occurs, the vertex x_j is retained and scans next one x_{j-1}, then set $x_i = x_{j-1}$, as shown in Figure 4(a). If there is no collision, the vertex x_j is retained and all the vertexes between x_{i+1} and x_{j-1} will be cut and moved to a set, Q_{RM} (Algorithm 3, Line 10), then set $x_i = x_j$, as shown in Figure 4(b). The operation will be repeated till all the vertexes in V are traversed. Finally the remaining vertexes will be reconnected and put to tree model T.(Algorithm 3, Lines 17-18). After gradual cutting, redundant vertexes in the path can be effectively removed. It can reduce the length of path and improve processing efficiency.

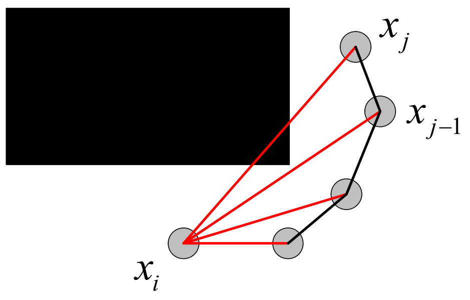
4. ANALYSIS.

In this section, we will prove some properties of the algorithm GSGC.

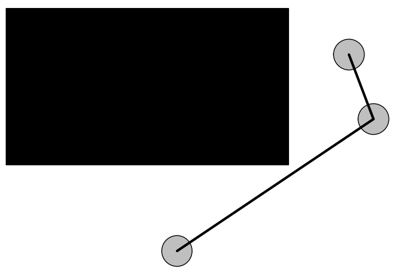
Definition 1(Probabilistic Completeness) Suppose there are a set of parameters x_{init}, x_{goal} and x_{obs}. For any path planning algorithm which has a feasible solution, we can say the algorithm has the property of probabilistic completeness if n samples satisfy:

$$\lim_{n \to \infty} P\left(\left\{V_n^{\text{algorithm}} \cap x_{\text{goal}}\right\} \neq \emptyset\right) = 1 \tag{11}$$

It means the algorithm can return the graph includes a path from x_{init} to x_{goal} without x_{obs}.



(a) Collision detection



(b) After gradual cutting

FIGURE 4. The gradual cutting process of path.

Theorem 1(Probabilistic Completeness of GSGC). For GSGC which has feasible solution, if it has n samples satisfying $\lim_{n\to\infty} P\left(\left\{V_n^{GSGC} \cap x_{goal}\right\} \neq \emptyset\right) = 1$ in current solution, there must exist a feasible path that connects the x_{init} and x_{goal} given enough time and iterations.

Proof: The RRT* has been proved to have the property of probabilistic completeness by Karaman et al. [30] This property is inherited by BIT* which is based on RRT* and Gammell et al. have proved that [43]. In the sample points selection, GSGC adds a guide strategy that changes the way of generation, but it keeps the connectivity of the tree like RRT* and BIT*. So GSGC also has the property of probabilistic completeness.

Definition 2(Asymptotically Optimality) Given a set of parameters x_{init}, x_{goal} and x_{obs}. For any path planning algorithm which has a feasible solution, if the algorithm returns a minimum movement cost of the solution with n samples satisfying:

$$P\left(\lim_{n\to\infty} \sup c_{best}^{\text{algorithm}} = c^*\right) = 1 \tag{12}$$

where $c_{best}^{algorithm}$ represents the movement cost of the best solution found by the algorithm, c^* is the movement cost of the optimal solution, we say the algorithm has the property of asymptotic optimality.

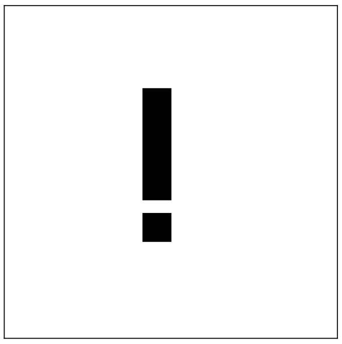
Theorem 2(Asymptotically Optimality of GSGC). For the GSGC which has a feasible solution, if it has n samples satisfying $\left(\lim_{n\to\infty} \sup c_{best}^{GSGC} = c^*\right) = 1$, namely the algorithm GSGC will always find a feasible path approximating the optimal solution from x_{init} to x_{goal}.

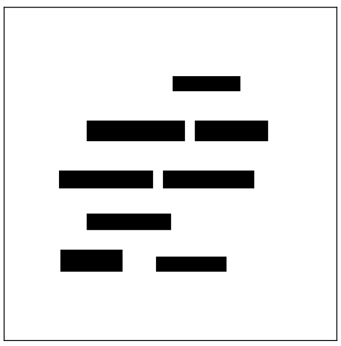
Proof: Karaman et al. have proved in their work that the RRT* can converge asymptotically to the optimal feasible solution in n random distribution samples. In the process of RRT*, it handles each sample independently and considers all the edges that less than length r. The BIT* handles samples with batches, however it processes the edges same to RRT*. The GSGC follows the same process as the BIT*. So the GSGC also has the property of asymptotically optimality.

5. EXPERIMENT.

The experiments were carried out in the simulated scenarios on a laptop with 8 GB of RAM and an Intel i5-9300H processor. The GSGC was compared with the BIT* and RRT* using evaluation benchmarks: length of path, run time, the effectiveness of gradual cutting and the convergence of iteration. The simulation platform is PyCharm with python 3.8. We used three different types of maps to simulate the real environment in Figure 5. Figure 5(a) is a narrow passage. Figure 5(b) is a simple environment. Figure 5(c) is a complex environment.

To keep the experiments consistent, the simulation parameters were set same for the three algorithms. The size of map is 240*240. The k_{p1} is 0.6 and k_{p2} is 6. The three


(a) MAP1: The single narrow passage


(b) MAP2: The simple environment

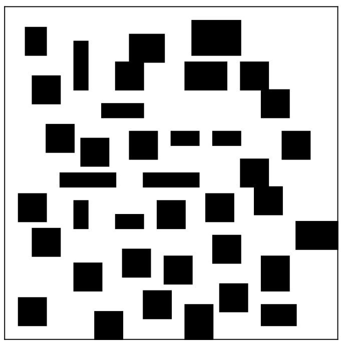

(c) MAP3: The complex environment

FIGURE 5. The different types of map in the experiments.

TABLE 1. The time cost and path length when passing narrow passages in Figure 6.

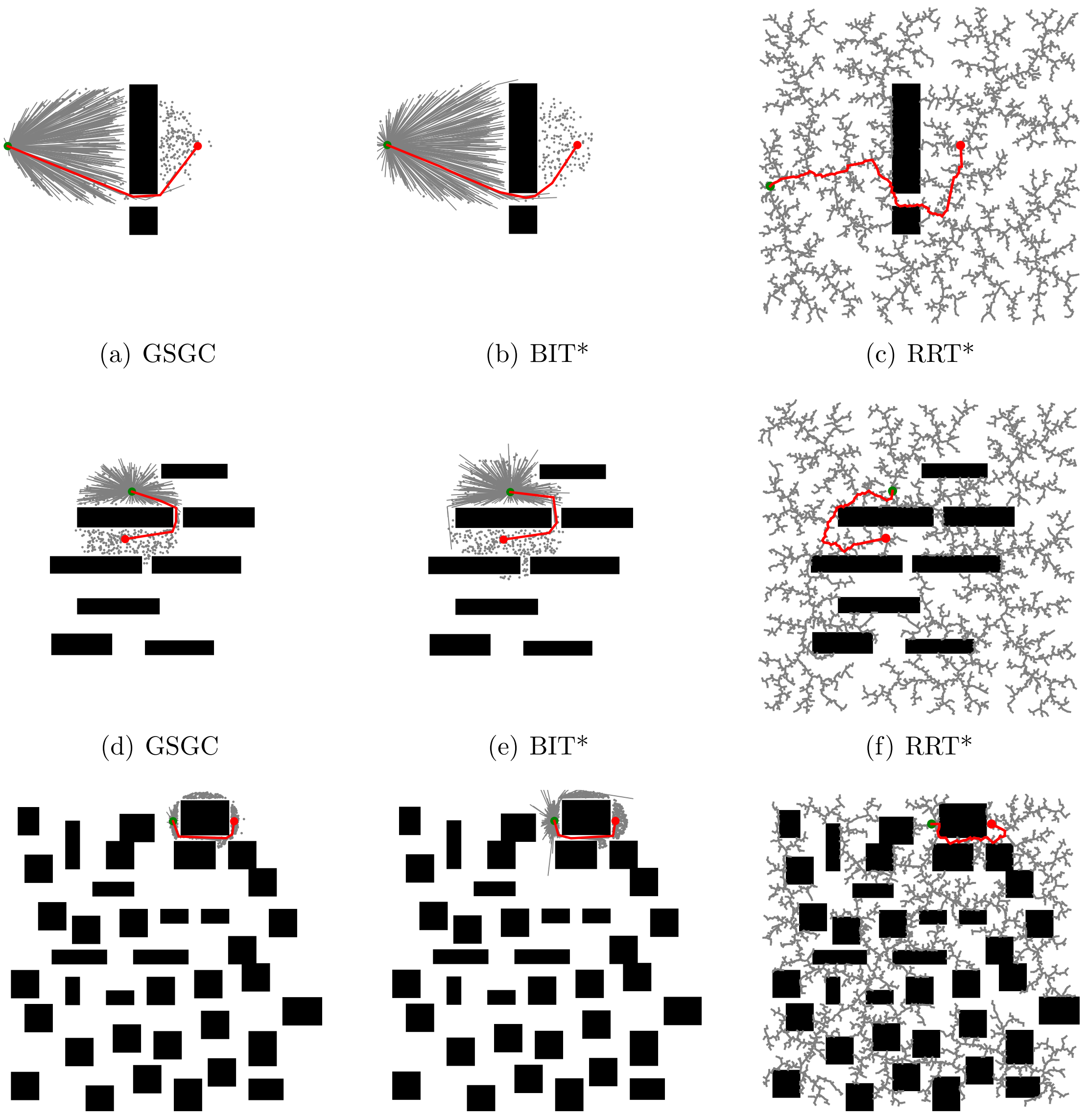
MAP	Algorithm	time(s)	pathlength
MAP1	GSGC	8.17	165.21
	BIT*	10.4	167.27
	RRT*	25.32	235.35
MAP2	GSGC	4.81	88.21
	BIT*	8.45	91.39
	RRT*	23.49	119.48
MAP3	GSGC	7.87	63.59
	BIT*	10.06	65.54
	RRT*	34.21	93.06

algorithms have identical initial points and goal points. For each experiment, we randomly selected 10 groups of points, and the experimental results were averaged.

5.1. **Fundamental performance.** The time cost and path length are two basic indexes in path planning for a robot. The better performance means using less time to find a shorter path. We considered some specific scenarios. The first one is the narrow area, which is difficult for a robot to pass. There are narrow passages in the three maps. The second one is the robot starting from the inside of the environment. The third one is the robot crossing the entire environment. The results were shown in Table 1-3 and Figure 6-8. The GSGC spent the shortest time to find the goal point with the shortest length of path among the three algorithms.

5.2. **The effectiveness of gradual cutting.** Generally, path planning tends to generate redundant points so that it will not be optimal. The GSGC adds a gradual cutting function to obtain local optimization. It can remove the redundant points efficiently and reduce bending angles. The experiments showed that the GSGC achieved local optimization since it has the least number of bending angles compared with BIT* and RRT* , as shown in Figure 9-10.

5.3. **The convergence of iteration.** As BIT* has a higher likelihood to find a better solution and converge faster towards the optimum than RRT* [43], we only carried out



(a) GSGC (b) BIT* (c) RRT*

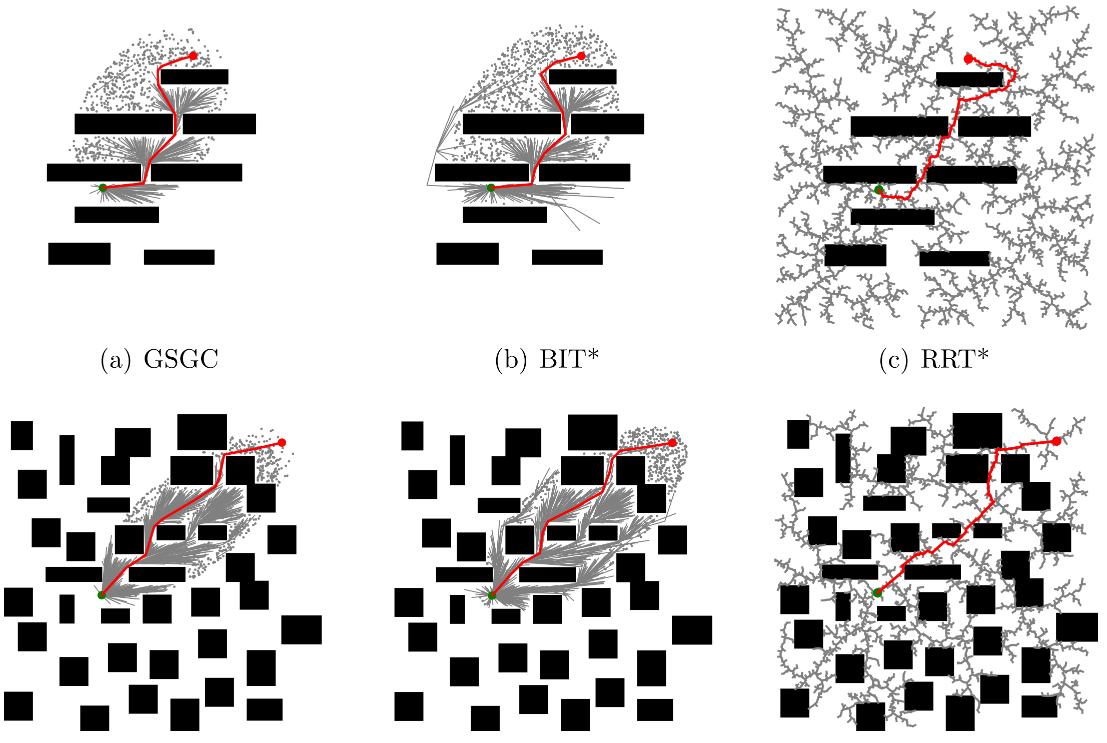
(d) GSGC (e) BIT* (f) RRT*

(g) GSGC (h) BIT* (i) RRT*

FIGURE 6. The experiments that robot passing narrow passages in different environments.

TABLE 2. The experimental results that the robot starting from the inside of the environments in Figure 7.

MAP	Algorithm	time(s)	pathlength
MAP2	GSGC	14.33	150.56
	BIT*	16.8	154.87
	RRT*	45.99	174.84
MAP3	GSGC	19.53	183.81
	BIT*	23.45	185.73
	RRT*	56.87	220.27



(a) GSGC (b) BIT* (c) RRT*

(d) GSGC (e) BIT* (f) RRT*

FIGURE 7. The experiments that the robot starting from the inside of the environments.

TABLE 3. The experimental results that the robot crossing the entire environments in Figure 8.

MAP	Algorithm	time(s)	pathlength
MAP2	GSGC	30.2	255.6
	BIT*	60.33	257.32
	RRT*	82.85	343.2
MAP3	GSGC	45.63	270.38
	BIT*	65.21	272.07
	RRT*	75.26	387.56

the comparisons between ours, GSGC, and BIT* in this section. We selected four kinds of scenarios of MAP2 and MAP3 to compare the convergence of GSGC and BIT* in Figure 7-8. As shown in Figure 11-12, the convergence curve represents the path length versus iteration times.We can see that the GSGC converges faster than BIT* and obtains the optimal path with less iterations. We also can find that BIT* is more unstable than GSGC since the convergence curves of BIT* fluctuate during iteration. Therefore, GSGC has faster and more stable convergence performance than BIT*.

6. CONCLUSION.

This paper presents a new path planner, GSGC, that introduces the attractive potential field to guide the new sample points to move toward the goal. The sampling ellipse area is shrunken according to the attractive parameters, which improves the path searching performance. Ultimately, the gradual cutting is used for trimming the redundant vertexes in the path. As demonstrated on simulated experiments, GSGC achieves the shortest path with least time cost, compared with BIT* and RRT*. After

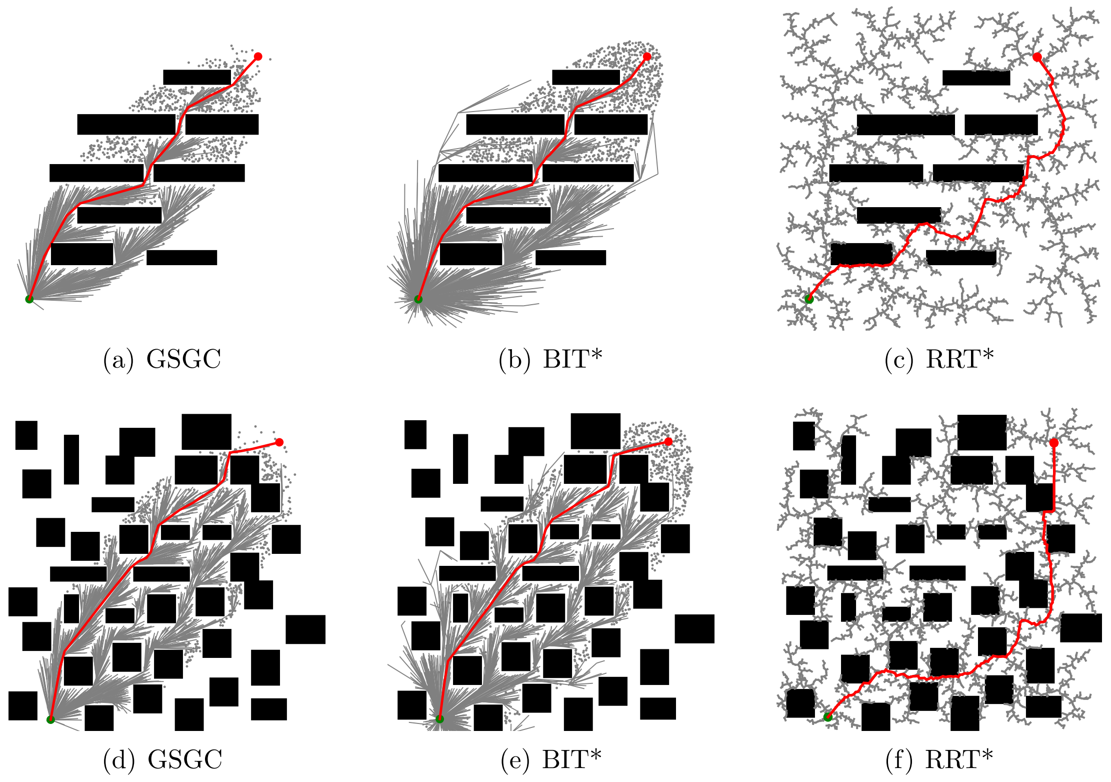


Figure 8. The experiments that the robot crossing the entire environments.

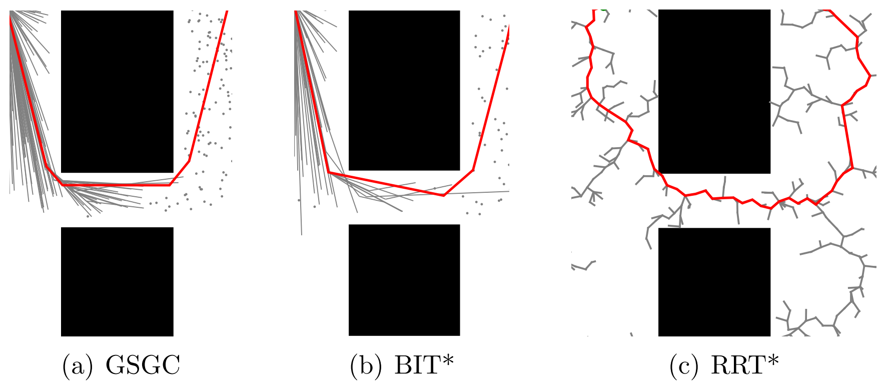


Figure 9. The turning at corners in MAP1.

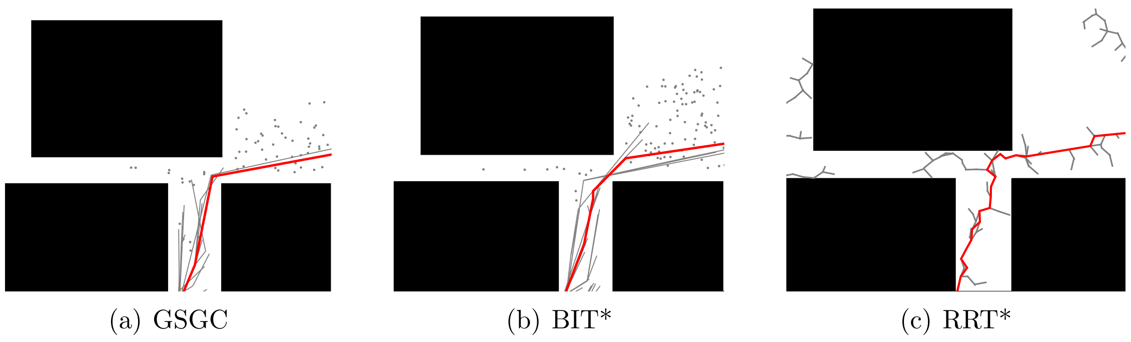
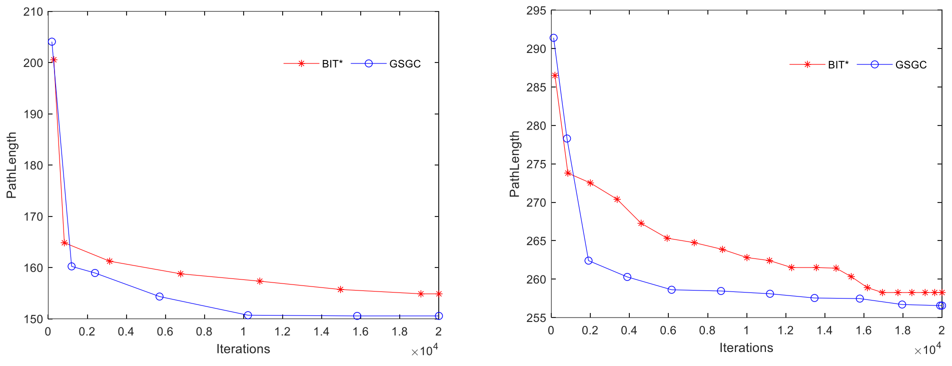
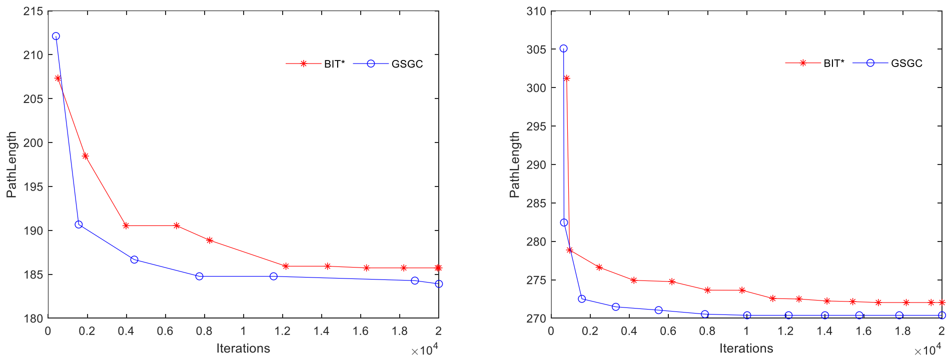


Figure 10. The turning at corners in MAP2.



(a) (b)

FIGURE 11. The convergence curve: (a)The process of iteration in Figure 7 (a), (b); (b)The process of iteration in Figure 7 (d), (e).



(a) (b)

FIGURE 12. The convergence curve: (a)The process of iteration in Figure 8 (a),(b); (b)The process of iteration in Figure 8 (d),(e).

gradual cutting, the path is further optimized. In the aspect of convergence of iteration, GSGC converges faster and more stably than BIT* which outperforms RRT*. Although the GSGC is an effective improvement, it also has some limitations. The GSGC has not been applied to a real robot, so it lacks consideration of kinematics constraints of robots. In the future, we will test the algorithm in robots and work more to improve it.

Acknowledgment. This work is partially supported by the Start-up Research Project under the Grant[GY-Z21064] of Fujian University of Technology, the New Engineering Research Project (Construction of Electronic and Electrical Practice Education System and Practice Platform Oriented to New Engineering) of Fujian University of Technology, the Research Project of Education&Teaching Reform of Fujian Province under the Grant [FBJG20190145] and the Research Project of Experimental Teaching Reform of Fujian University of Technology under the Grant [SJ2018002]. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] R. Fareh, M. Baziyad, T. Rabie and M. Bettayeb, Enhancing path quality of real-time path planning algorithms for mobile robots: A sequential linear paths approach, *IEEE Access*, vol.8, pp.167090-167104, 2020.

[2] I Sung B Choi and P Nielsen, On the training of a neural network for online path planning with offline path planning algorithms, *International Journal of Information Management*, vol.57, Article ID 102142, 9 pages, 2021.

[3] Y. Q. Huang , Z. K. Li, Y. Jiang and L. Cheng, Cooperative path planning for multiple mobile robots via HAFSA and an expansion logic strategy, *Applied Sciences*, vol.57, no.4, pp.1-10, 2019.

[4] R. Sandström, A. Bregger, B. Smith, S. Thomas and N. M. Amato, Topological nearest-neighbor filtering for sampling-based planners, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp.3053-3060, 2018.

[5] P. C. Song, J. S. Pan and S. C. Chu, A Parallel Compact Cuckoo Search Algorithm for Three-Dimensional Path Planning, *Applied Soft Computing*, vol.94, Article ID 106443, 2020.

[6] E. I. Al Khatib, M. A. K. Jaradat and M. F. Abdel-Hafez, Low-cost reduced navigation system for mobile robot in indoor/outdoor environments, *IEEE Access*, vol.8, pp.25014-25026, 2020.

[7] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *The International Journal of Robotics Research*, vol.5, pp.90-98, 1986.

[8] Y. Koren and J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, *1991 IEEE International Conference on Robotics and Automation*, pp.1398-1404, 1991.

[9] I. B Jeong, S. J. Lee and J. H. Kim, Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate, *Expert Systems with Applications*, vol.123, pp.82-90, 2019.

[10] I. Sung, B. Choi and P. Nielsen, On the training of a neural network for online path planning with offline path planning algorithms, *International Journal of Information Management*, vol.57, Article ID102142, 9 pages, 2021.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions of Systems Science and Cybernetics*, vol.4, pp.100-107, 1968.

[12] C. S. Sallaberger, G. M. T. D'Eleuterio, Optimal robotic path planning using dynamic programming and randomization, *Acta Astronautica*, vol.35, pp.143-156, 1995.

[13] Y. Ma and N. N. Wu, Research on Seismic Site Emergency Rescue Traffic Path Analysis System Based on Public Image Information, *Journal of Information Hiding and Multimedia Signal Processing*, vol.9, pp.577-585, 2018.

[14] Q. W. Chai, S. C. Chu, J. S. Pan, and W. M. Zheng, Applying Adaptive and Self Assessment Fish Migration Optimization on Localization of Wireless Sensor Network on 3-D Te rrain *Journal of Information Hiding and Multimedia Signal Processing*, Vol.11, No.2, pp.90-102, 2020.

[15] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol.4, pp.1942-1948, 1995.

[16] J. H. Holland (eds.), *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, USA,1992.

[17] K. Price, R. M. Storn, and J. A. Lampinen (eds.), *Differential Evolution: a Practical Approach to Global Optimization*, Springer, Berlin-Heidelberg, Germany, 2007.

[18] J. S. Pan, P. W. Tsai, and S. C. Chu, Cat swarm optimization, *9th Pacific Rim International Conference on Artificial Intelligence*, pp.854-858, 2006.

[19] C. C. Shih, M. F. Horng, T. S. Pan, J. S. Pan, and C. Y. Chen, A genetic-based effective approach to path-planning of autonomous underwater glider with upstream-current avoidance in variable oceans, *Soft Computing*, vol.21, no.18, pp.5369-5386, 2016.

[20] T. K. Dao, J. S. Pan, T. S. Pan, and T. T. Nguyen, Optimal path planning for motion robots based on bees pollen optimization algorithm, *Journal of Information and Telecommunication*, vol.1, no.4, pp.351-366, 2017.

[21] J. S. Pan, N. X. Liu, S. C. Chu, A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning, *IEEE Access*, vol.8, pp.17691-17712, 2020.`https://doi.org/10.1109/ACCESS.2020.2968119`

[22] Z. Y. Meng and J. S. Pan, QUasi-Affine TRansformation Evolution with External ARchive (QUATRE-EAR): an enhanced structure for differential evolution, *Knowledge-Based Systems*, vol.155, pp.35-53, 2018.

[23] F. Q. Zhang, T. Y. Wu, Y. O. Wang, R. Xiong, G. Y. Ding, P. Mei, and L. Y. Liu , Application of Quantum Genetic Optimization of LVQ Neural Network in Smart City Traffic Network Prediction, *IEEE Access*, vol.8, pp.104555-104564, 2020.

[24] P. C. Song, S. C. Chu, J. S. Pan, and H. M. Yang, Simplified Phasmatodea population evolution algorithm for optimization, *Complex & Intelligent Systems*, pp.1-19, 2021.`https://doi.org/10.1007/s40747-021-00402-0`

[25] X. Jing, K. C. Song, D. F. Zhang, H. W. Dong, Y. H,Yan and Q. G. Meng, Informed Anytime Fast Marching Tree for Asymptotically Optimal Motion Planning, *IEEE Transactions on Industrial Electronics*, vol.68, no.6, pp.5068-5077, 2021.

[26] N. M. Amato and Y. Wu, A randomized roadmap method for path and manipulation planning, *1996 IEEE International Conference on Robotics Automation*, pp.113-120, 1996.

[27] S. M. LaValle and J. J. Kuffner, Randomized kinodynamic planning, *1999 IEEE International Conference on Robotics and Automation*, pp.473-479, 1999.

[28] R. Mashayekhi, M.Y.I. Idris, M.H. Anisi, and I. Ahmedy, Hybrid rrt: A semi-dual-tree rrt-based motion planner, *IEEE Access*, vol.8, pp.18658-18668, 2020.

[29] J. J. Kuffner and S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, *2000 IEEE International Conference on Robotics and Automation*, pp.995-1001, 2000.

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

[31] D. Yi, M. A. Goodrich, and K. D. Seppi, Homotopy-aware RRT*: Toward human-robot topological path-planning, *2016 11th ACM/IEEE International Conference on Human-Robot Interaction*, pp.279-286, 2016.

[32] B. Akgun and M. Stilman, Sampling heuristics for optimal motion planning in high dimensions, *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.2640-2645, 2011.

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

[34] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, RRT*-SMART: A rapid convergence implementation of RRT, *2012 IEEE International Conference on Mechatronics and Automation*, pp.1651-1656, 2012.

[35] O. Arslan and P. Tsiotras, Use of relaxation methods in sampling-based algorithms for optimal motion planning, *2013 IEEE International Conference on Robotics and Automation*, pp.2421-2428, 2013.

[36] Y. J. Li, W. Wu, Y. Gao,D. L Wang, and Z. Fan, PQ-RRT*: An improved path planning algorithm for mobile robots, *Expert Systems with Applications*, vol.152, Article ID 113425, 2020.

[37] A. H. Qureshi and Y. Ayaz, Potential functions based sampling heuristic for optimal path planning, *Autonomous Robots*, vol.40,no.6,pp.1079-1093, 2016.

[38] D. Ferguson and A. Stentz, Anytime rrts, *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.5369-5375, 2006.

[39] S. M. Persson and I. Sharf, Sampling-based A* algorithm for robot path-planning, *The International Journal of Robotics Research*, vol.33, no.13, pp.1683-1708, 2014.

[40] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, Informed RRT*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic, *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.2997-3004, 2014.

[41] B.K. Patle, Ganesh Babu L, Anish Pandey, D.R.K. Parhi, and A. Jagadeesh, A review: On path planning strategies for navigation of mobile robot, *Defence Technology*, vol.15, no.4, pp.582-606, 2019.

[42] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy, and I. Ali, Informed RRT*-connect: An asymptotically optimal single-query path planning method, *IEEE Access*, vol.8, pp.19842-19852, 2020.

[43] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs, *2015 IEEE International Conference on Robotics and Automation*, pp.3067-3074, 2015.

[44] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, Informed sampling for asymptotically optimal path planning, *IEEE Transactions on Robotics*, vol.34,no.4,pp.966-984, 2018.

[45] L. Janson, E. Schmerling, A. Clark, and M. Pavone, Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions, *The International Journal of Robotics Research*, vol.34, no.7, pp.883-921, 2015.

[46] S. Koenig, M. Likhachev, and D. Furcy, Lifelong planning A*, *Artificial Intelligence*, vol.155, no.1, pp.93-146, 2004.