IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2022; **17**: 200–208
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI:10.1002/tee.23502

Paper

Fast-RRT*: An Improved Motion Planner for Mobile Robot in Two-Dimensional Space

Qinghua Li[*,**], Non-member
Jiahui Wang[**,***], Non-member
Haiming Li[**,***], Non-member
Binpeng Wang[**,***], Non-member
Chao Feng[*,**a], Non-member

Path planning for mobile robot aims to solve the problem of creating a collision-free path from the start state to the goal state in the given space, which is a key supporting technology for unmanned work. In order to solve the problems of the asymptotic optimal rapidly extended random trees star (RRT*) algorithm, such as its slow convergence rate, the low efficiency of planning and the high cost of path, an improved motion planner (Fast-RRT*) was proposed based on hybrid sampling strategy and choose parent based on backtracking. Firstly, the goal bias strategy and constraint sampling are combined in the sampling stage to reduce the blindness of sampling. Secondly, to obtain a path with lower cost than the RRT* algorithm, the ancestor of the nearest node is considered until the initial state in the process of choose parent for new node. To ensure the feasibility of the path, the path is smoothed by cubic B-spline curve. The effectiveness of Fast-RRT* algorithm was verified based on MATLAB and V-rep platform. © 2021 Institute of Electrical Engineers of Japan. Published by Wiley Periodicals LLC.

Keywords: backtracking; hybrid sampling strategy; path planning; RRT*

Received 8 February 2021; Revised 26 July 2021

1. Introduction

With the enhancement of complexity and dynamics of unmanned work, path planning has been received considerable attention. Roughly speaking, that is to find a safe and optimal path from the start state to the goal state in given workspace according to some optimization criteria, such as shortest distance, time. At present, common path planning algorithms include genetic algorithm [1,2], artificial potential field method (APFM) [3,4], A star(A*) [5,6], probabilistic roadmaps (PRM) [7], and rapidly exploring random trees (RRT) [8,9], and so forth.

The genetic algorithm can search the given environment globally more effective than other algorithms, but its rate of search is slow [10]. The APFM can obtain the optimal path and the path smoothly, but it is easy to fall into local optimal [11]. In addition, the APFM will oscillate in a narrow environment, resulting in failure to reach the goal point. A* can reach the goal state quickly based on heuristic function, but its path is not optimal [12,13]. PRM is probabilistic complete and not optimal [14,15]. It is remarkable that A* and Dijkstra are based on grid search to find the global optimal path. However, since the physical resources such as memory space and computation time grow exponentially, do not scale well with the state dimension or the size of problem space [16]. Compared with other algorithms, RRT has low computational complexity and can find solutions without using explicit presentation about its workspace. As a result, rapidly extended random trees (RRT) has better ability to find the path in high-dimensional environment.

RRT can finds a path quickly, it cannot ensure asymptotic optimization of path search [17] To solve this problem, S. Karaman *et al.* [17] proposed the RRT*, which introduced the process of selecting the parent node for the new node in the neighborhood and the process of rewiring to optimize the path with the increase of the number of samples. In addition to having probabilistic completeness, it also ensured the asymptotic optimization of the path search. Although the RRT* can obtain an optimal solution in theory, the calculation of the optimal solution cannot be completed due to the slow rate of convergence in a finite time.

In order to improve the performance of the RRT*, Nasir J *et al.* [18] proposed the RRT*-SMART, which introduced intelligent sampling and path optimization strategy. After finding a feasible path between the start state and the goal state, the RRT*-SMART continuously judges whether the parent nodes can be directly connected from the child nodes. In the optimization process, if there is an anchor node near the obstacle that cannot be optimized directly, sampling should be added near the anchor node to find a more optimal path. It accelerate the rate of convergence of RRT*, but the environmental adaptability of this algorithm is poor.

[a] Correspondence to: Chao Feng. E-mail: cfeng@qlu.edu.cn

[*]School of Electronic and Information Engineering(Department of Physics), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, P. R. China
[**]Jinan Engineering Laboratory of Human-machine Intelligent Cooperation
[***]School of Electrical Engineering and Automation, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, P. R. China

Noreen I *et al.* [19] proposed RRT*-AB (RRT*-Adjustable Bounds, RRT*-AB) based on connected regions, goal bounded sampling and path optimization. The goal bias bounded sampling is carried out in the boundary of the connected region to find the initial obstacle-free path. The sampling strategy reduces the sampling blindness of RRT algorithm. The planned path is gradually optimized by means of node suppression and concentrated bounded sampling.

Zhang Weimin *et al.* [20] proposed an improved RRT* based on goal constrained sampling and goal bias extension (GCSE-RRT*). In the sampling stage, the bias probability of goal is a fixed value, the goal node is selected as the sampling node according to a fixed probability, and the location of the sampling node is constrained during random sampling to improve the orientation of goal. The expansion of new nodes is determined by both random node and goal point to the rate of search of the algorithm. Relatively, GCSE-RRT* improves the utilization of node, occupies less memory, and runs in shorter time, but it planned path is not optimal.

RRT* still has some problems, such as slow rate of convergence, the low efficiency. This paper proposes the Fast-RRT*, and the main contributions are descried as follows:

1. Hybrid sampling strategy combined with the goal bias sampling and constraints sampling. The proportion of goal sampling and random sampling is balanced by the appropriate bias probability. The flexibility of searching path is improved. In constraints sampling, the sampling is successful when the constraint conditions are satisfied. Otherwise, the sampling will continue. In this way, the blindness of sampling can be reduced and the efficiency of the overall planning of the algorithm can be improved.
2. Based on the idea of backtracking, the set of possible parent nodes of the new node is expanded to reduce the cost of path. This set considers the node closest to the new node until the start state. The path is smoothed through the cubic B-spline curve [21] to ensure the feasibility of the path.

The rest of this paper is organized as follows. Section 2 gives the definition of the problem and the RRT* algorithm. Section 3 introduces the basic principle of Fast-RRT* algorithm in detail and analyzes its probabilistic completeness and complexity. In section 4, experiments and discussions are presented. Section 5 summarizes the work of this paper and looks forward to the future work.

2. Background

This section formalizes the problem of path planning and introduces the RRT* which is the basis of the proposed algorithm.

2.1. Problem definition

Let the given workspace be denoted by a set $W = (0,1)^d$, where d represents the dimension of workspace. Obstacle is denoted by $W_{obs} \subset W$, and obstacle-free is denoted by $W_{free} = W/W_{obs}$. $\left(W_{free}, q_{init}, q_{goal}\right)$ defines the problem of path planning, where $q_{init} \in W_{free}$ is the start point and $q_{goal} \in W_{free}$ is the goal point. Let a continuous function $\sigma : [0,1] \mapsto W$ of bounded variation be a path. The path σ is obstacle-free if $\forall\tau \in [0,1]$, $\forall\tau \in [0,1]$. The algorithm builds a tree $T = (V, E)$, where V is the set of vertices in the random tree, and E is the set of edges that connect these vertices. This paper only considers Euclidean distance. When $d = 2$, the cost of path $c(q_1, q_2)$ between $q_1 = \left(q_{1x}, q_{1y}\right) \in W_{free}$ and $q_2 = \left(q_{2x}, q_{2y}\right) \in W_{free}$ in this tree is defined as:

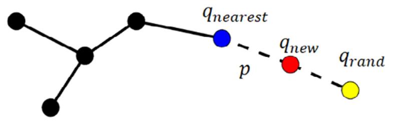


Fig. 1. The growth process of random tree

$$c(q_1, q_2) = \sqrt{(q_{1x} - q_{2x})^2 + \left(q_{1y} - q_{2y}\right)^2} \tag{1}$$

The definition of path cost in 3D workspace is similar to the above.

Assume that the feasible path from the start point q_{init} to state q is $\sigma_q = (q_{init}, q_1, q_2, \ldots, q_i, q)$. Then the cost of path σ_q is defined as:

$$c\left(\sigma_q\right) = c(q_{init}, q_1) + c(q_1, q_2) + \cdots + c(q_i, q) \tag{2}$$

Definition 1 (Feasible path solution). Find a feasible path σ : $[0,1]$, if one exists in $W_{free} \subset W$ such that $\sigma(0) = q_{init} \in W_{free}$ and $\sigma(1) = q_{goal} \in W_{free}$.

Definition 2 (Optimal path solution). Find an optimal path σ^* which connects q_{init} and q_{goal} in $W_{free} \subset W$ such that the cost of path σ^* is minimum, that is, $c(\sigma^*) = \min\{c(\sigma)\}$, where $c(\sigma)$ is the cost to reach to q_{goal} along with a path σ.

Definition 3 (Convergence to optimal solution). Find an optimal path σ^* in $W_{free} \subset W$ in the least possible time.

2.2. RRT*

Based on the RRT, one adds the process of finding the best parent node and rewiring to generate a better path in the given state space which is abbreviated as RRT*. The pseudocode of the RRT* algorithm is given in Algorithm 1. It starts with a tree rooted in vertex q_{init} and gradually extends the tree in obstacle-free region. Firstly, a node q_{rand} is randomly generated in the given state space. Then, all vertices in the random tree T are traversed to find the node $q_{nearest}$ closest to the random node, and the new node q_{new} is obtained after extending a step length p to the direction of the random node at the nearest node. The growth process of random tree is shown in Fig. 1.

Algorithm 1 *RRT**(q_{init}, q_{goal}, W_{free}, n)

```
Input: the start point $q_{init}$
  the goal point $q_{goal}$
  the obstacle-free workspace $W_{free}$
  the number of iterations $n$
Output: $T$
  $T.init$
  for $i = 0$ to $i = n$ do
    $q_{rand} \leftarrow Sample(i)$
    $q_{nearest} \leftarrow Nearest(T, q_{rand})$
    $(q_{new}, \sigma) \leftarrow Steer(q_{nearest}, q_{rand}, p)$
    if $\sigma \in W_{free}$ then
      $Q \leftarrow Near(T, q_{new}, r)$
      $q_{\min} \leftarrow Chooseparent(Q, q_{new}, q_{nearest}, \sigma)$
      $T \leftarrow Connect(T, q_{min}, q_{new}, \sigma)$
      $T \leftarrow Rewire(T, Q, q_{new})$
    end if
  end for
  return $T$
```

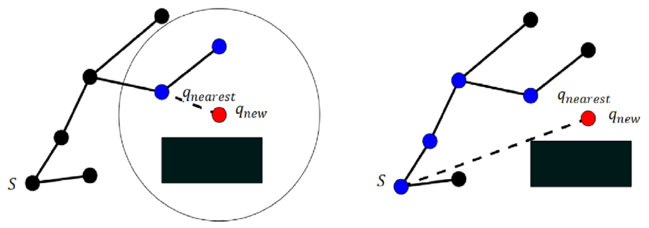


Fig. 2. The choosing parent node for q_{new} ((a) RRT* (b) Fast-RRT*)

The main modules of RRT* is described as follows:

- *Sample*: It returns a randomly sampled state in W_{free}.
- *Nearest*: The function $Nearest(T, q_{rand})$ returns the vertex closest to q_{rand} in the random tree $T = (V, E)$.
- *Steer*: The function $Steer(q_{nearest}, q_{rand}, p)$ returns a path σ from $\sigma(0) = q_{nearest}$ to $\sigma(1) = q_{new}$ at a distance p, where p is the incremental distance.
- *Near*: The function $Near(T, q_{new}, r)$ returns a set Q of vertices in V that are contained in a sphere of a given radius r centered at q_{new}.
- *Connect*: Given a tree $T = (V, E)$, two states $q_{min}, q_{new} \in V$, and a path σ from q_{min} to q_{new}, it adds a vertex representing q_{new} to V and an edge representing σ to E.

If the path σ between the nearest node $q_{nearest}$ and the new node q_{new} is in the obstacle-free region, then take new node q_{new} as the center of the circle and r as the radius to obtain the set Q of nodes in the sphere. The nodes in the set Q are shown as the blue dots in Fig. 2(a). The radius r is a user-defined constant and $p < r < 2p$. If the radius is too large, the number of vertices in set Q will increase, and the running time of choose parent will be longer. The radius is too small, the effectiveness of path optimization will become sacrificed. In the set of Q, select a node $q_{\min}$ to ensure that the cost $c\left(\sigma_{q_{new}}\right)$ is quasi-optimal. Algorithm 2 gives the pseudo-code for *Chooseparent*.

Algorithm 2 *Chooseparent*(Q, q_{new}, $q_{nearest}$, σ)

```
Input: the set of neighboring nodes $Q$.
  the new node $q_{new}$
  the nearest node $q_{nearest}$
  the path $\sigma$ between $q_{new}$ and $q_{nearest}$
Output: $q_{\min}$
  $q_{\min} \leftarrow q_{nearest}$
  $\sigma_{\min} \leftarrow \sigma$
  $c_{\min} \leftarrow c(T, q_{nearest}) + c(q_{nearest}, q_{new})$
  for all $q_{near} \in Q$ do
     $(q, \sigma) \leftarrow Steer(q_{near}, q_{new})$
     $c \leftarrow c(T, q_{near}) + c(q_{near}, q_{new})$
     if $c < c_{\min}$ then
        if $\sigma \in W_{free}$ then
           $q_{\min} \leftarrow q_{near}$
           $\sigma_{\min} \leftarrow \sigma$
           $c_{\min} \leftarrow c$
        end if
     end if
  end for
  return $q_{\min}$
```

Once the new node q_{new} is connected to the random tree T, the node in Q tries to reconnect new node q_{new} if the cost of the path through q_{new} is lower than the current path. The process of *Rewire* is described in Algorithm 3.

Algorithm 3 *Rewire*(T, Q, q_{new})

```
Input: the random tree $T$
  the set of neighboring nodes $Q$
  the new node $q_{new}$
Output: $T$
  for $q_{near} \in Q$ do
     $\sigma \leftarrow Steer(q_{new}, q_{near})$
     if $c(T, q_{new}) + c(q_{new}, q_{near}) < c(T, q_{near})$ then
        if $\sigma \in W_{free}$ then
           $T \leftarrow Reconnect(T, q_{new}, q_{near}, \sigma)$
        end if
     end if
  end for
  return $T$
```

3. Fast-RRT*

This section describes the Fast-RRT* and analyzes its probability completeness and computational complexity. Two proposed key concepts of Fast-RRT*, hybrid sampling and choose parent based on backtracking, are introduced in detail.

3.1. Proposed algorithm

Initially, Fast-RRT* searches the state space as RRT* does. In RRT*, sampling nodes are randomly generated. When selecting a vertex with a lower cost of path for new node q_{new}, consider vertices in the sphere centered on the new node. But in the Fast-RRT* algorithm, we proposed hybrid sampling strategy to improve goal-orientation. The set of vertices that traceback to the start point at the nearest node $q_{nearest}$ is considered. The pseudocode describing of Fast-RRT* algorithm is shown in Algorithm 4.

Algorithm 4 *Fast − RRT**(q_{init}, q_{goal}, W_{free}, n)

```
Input: the start point $q_{init}$
  the goal point $q_{goal}$
  the obstacle-free workspace $W_{free}$
  the number of iterations $n$
Output: $T$
  $T.init$
  for $i = 0$ to $i = n$ do
     $q_{rand} \leftarrow Hybridsampling(q_{goal}, \lambda_{bias})$
     $q_{nearest} \leftarrow Nearest(T, q_{rand})$
     $(q_{new}, \sigma) \leftarrow Steer(q_{nearest}, q_{rand}, p)$
     if $\sigma \in W_{free}$ then
        $A \leftarrow BackTracking(T, q_{new}, q_{nearest}, q_{init})$
        $q_{\min} \leftarrow ChooseParent(A, q_{new}, q_{nearest}, \sigma)$
        $T \leftarrow Connect(T, q_{\min}, q_{new}, \sigma)$
     end if
  end for
  return $T$
```

3.2. Hybrid sampling

The pseudocode of *Hybrid–sampling* is shown in Algorithm 5. Assume the threshold of sampling is λ_{bias}, and the probability of sampling is $\lambda_{rand} \in (0, 1)$ uniformly. The λ_{bias} is a user-defined value, and $0 < \lambda_{bias} < 1$. If λ_{rand} is less than the threshold, the sampled node is the goal. Otherwise, it is sampled with constraints in the given state space. The position of the sampling node with constraints is subjected to:

$$\begin{aligned} |q_{randx} - q_{goalx}| &< |q_{pre-randx} - q_{goalx}| \\ |q_{randy} - q_{goaly}| &< |q_{pre-randy} - q_{goaly}| \end{aligned} \tag{3}$$

where, $(q_{pre-randx}, q_{pre-randy})$, (q_{randx}, q_{randy}), and (q_{goalx}, q_{goaly}) are respectively the position of the previous node of random sampling, the current node of random sampling and the goal point. In the state space, if the sampling node is closer to the goal than the previous sampling node in the direction of x or y, the sampling is successful. Otherwise, the sampling will continue until the above constraints are satisfied.

Algorithm 5 *Hybrid sampling*(q_{goal}, λ_{bias})

```
Input: the goal point $q_{goal}$
  the threshold of sampling $\lambda_{bias}$
Output: $q_{rand}$
  $\lambda_{rand} \leftarrow rand()$
  if $\lambda_{rand} < \lambda_{bias}$ then
      $q_{rand} \leftarrow q_{goal}$
  else
      $q_{rand} \leftarrow Consample()$
  end if
  return $q_{rand}$
```

- *rand*: It returns a randomly generated sampling probability $\lambda_{bias} \in (0, 1)$ in the given state space.
- *Consample*: It returns a sampling node that satisfies the position constraint.

3.3. Choose parent

The subroutine *BackTracking* is shown in Algorithm 6. When the new node q_{new} is in the obstacle-free region, the possible parent nodes set A of the new node is obtained by tracing from the nearest node to the start point. The nodes in the set A are shown as the blue dots in Fig. 2(b). The node with the lowest cost in this set is regarded as the best parent node of the new node. The subroutine *Chooseparent* is shown in Algorithm 7.

Algorithm 6 *BackTracking*(T, q_{new}, q_{init}, $q_{nearest}$)

```
Input: the random tree $T$
  the new node $q_{new}$
  the start point $q_{init}$
  the nearest node $q_{nearest}$
Output: $A$
  $node \leftarrow q_{nearest}$
  while $node \neq q_{init}$ do
      $q_c \leftarrow nodeparent$
      $A \leftarrow insertnode(A, q_c)$
      $node \leftarrow q_c$
  end while
  return $A$
```

Algorithm 7 *Chooseparent*(A, q_{new}, $q_{nearest}$, σ)

```
Input: the set of possible parent nodes $A$
  the new node $q_{new}$
  the nearest node $q_{nearest}$
  the path $\sigma$ between $q_{new}$ and $q_{nearest}$
Output: $q_{\min}$
  $q_{\min} \leftarrow q_{nearest}$
  $\sigma_{\min} \leftarrow \sigma$
  $c_{\min} \leftarrow c(T, q_{nearest}) + c(q_{nearest}, q_{new})$
  for all $q_{node} \in A$ do
      $(q, \sigma) \leftarrow Steer(q_{node}, q_{new})$
      $c \leftarrow c(T, q_{node}) + c(q_{node}, q_{new})$
      if $c < c_{\min}$ then
          if $\sigma \in W_{free}$ then
              $q_{\min} \leftarrow q_{node}$
              $\sigma_{\min} \leftarrow \sigma$
              $c_{\min} \leftarrow c$
          end if
      end if
  end for
  return $q_{\min}$
```

3.4. Complexity analysis

This section proves that the complexity of the Fast-RRT* algorithm is the same as that of the RRT* algorithm in time and space. The space and time complexity of RRT* algorithm are $O(n)$ and $O(n \log n)$, respectively [17].

Lemma 1. The space complexity of the Fast-RRT* algorithm is $O(n)$.

Proof. The space complexity of the random tree T_n is the sum of the space complexity of the vertices V_n and the space complexity of edges E_n. Therefore, the space complexity of the random tree T_n is $O(n)$:

$$O(n) \approx O(n) + O(n-1) \tag{4}$$

■

Lemma 2. The time complexity of the Fast-RRT* algorithm is $O(n \log n)$.

Proof. In the Fast-RRT* algorithm, the process of calling collision detection is the process of selecting the optimal parent node for the new node. Assuming that the number of possible parent nodes of the new node in the set A is K, the process complexity of selecting the optimal parent node for the new node is $O(K \log n)$. Therefore, the overall complexity of the algorithm is $O(nK \log n)$, where $K \ll n$, the space complexity of the Fast-RRT* algorithm is $O(n \log n)$■

3.5. Probabilistic completeness

Assume M^{AL} denotes the vertices of the tree generated by an algorithm AL after n iterations. S. Karaman *et al.* have proved that RRT algorithm provides probabilistic completeness and RRT* algorithm inherits this property of RRT [17]. It satisfies:

$$\lim_{n \to \infty} P\left(\left\{M^{RRT*} \cap q_{goal} \neq \emptyset\right\}\right) = 1 \tag{5}$$

Similar to RRT*, we claim that holds for Fast-RRT* as well, which is stated formally in Lemma 3 as follow.

Table I. The experiment of parameter values

Parameter	Value
Size of experimental environment	200×200 pixels
Number of experiments	500
Number of iterations	2000
Step size p	5 pixels
Radius r	7 pixels

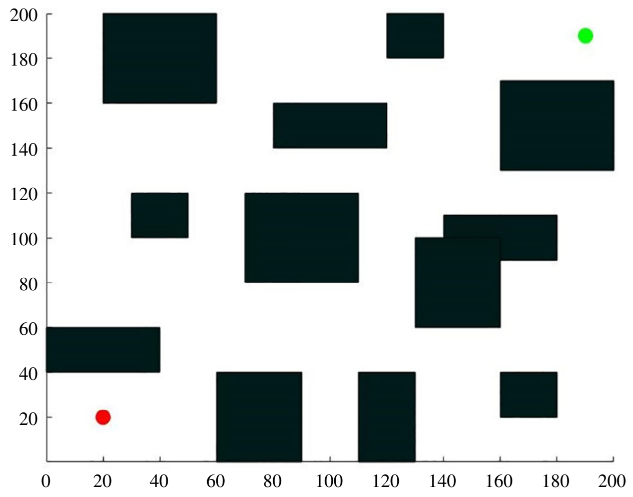


Fig. 3. The general environment of experiment (red dot: Start point, green dot: Goal point)

Lemma 3. For any given robustly feasible path planning problems $(W_{free}, q_{init}, q_{goal})$, if a feasible path solution exists, then

$$\lim_{n\to\infty} P\left(\left\{M^{Fast-RRT*} \cap q_{goal} \neq \emptyset\right\}\right) = 1 \tag{6}$$

Proof . By convention, we have defined $M_0^{RRT*} = M_0^{Fast-RRT*} = q_{init}$ (Algorithm 1 to 4). Like RRT* algorithm, the random tree generated by Fast-RRT* contains q_{init} as one of its states; the Fast-RRT* algorithm generates a random sample, which is connected to the state of the nearest node in the tree. Therefore, the tree generated by Fast-RRT* is also a connected tree; Fast-RRT* points the random sample to the goal point q_{goal}, so when the number of iterations approaches infinity, the probability of the tree generated by Fast-RRT* finding the goal point is close to 1. In general, the probability of Fast-RRT* finding a feasible path solution approaches 1 as the number of iterations approaches infinity. ■

4. Experiments and Discussions

Compared with RRT* [17] and GCSE-RRT* [20] in different environment, and the performance analysis is given in detail. The size of experimental environment is 200×200 pixels, it including general environment and cluttered environment. Statistical data were derived from 500 independent runs for each algorithm. Experiments were run on an Intel i5-8265U CPU with 8G of RAM. The parameter values in the experiment are described in Table I.

4.1. General environment

The general environment is shown in Fig. 3. The black area is the obstacle, and the red line represent the final path. Figure 4 shows the generated path for related algorithm with 2000 iterations in the same environment.

The comparison between three algorithms is shown in Fig. 5 and Fig. 6. The figure clearly demonstrates the convergence rate of three algorithms. After many experiments, Fast-RRT* algorithm found the initial obstacle-free path at 480 iterations with a higher probability. The iterations of Fast-RRT* is lower than other two algorithms. Figure 6 shows that Fast-RRT* algorithm can converge to the optimal path faster. Fast-RRT* not only generate better solution but also take less time and iterations to converge. The path cost cannot be calculated before the initial path is found. The iterations/time for each algorithm to find the initial path is different, so the iterations/time of algorithm cannot start with zero in the comparative experiment of Figs. 5 and 6.

Table II shows the comparison of the cost of path and time of the three algorithms. Compared with RRT* and GCSE-RRT*, the path cost of Fast-RRT* is reduced by 11.56% and 4.45%, and the running time is shortened by 14.54% and 12.18%.

In order to verify the feasibility and effectiveness of the Fast-RRT* algorithm for mobile robot path planning, a 3D path planning experiment was carried out in V-rep with a dual-wheel differential drive robot Pioneer P3Ddx as the experimental object. The environment size is 5 m × 5 m, and the origin is in the center. Where, the start point is $(-1.5, -2, 0.1)$, the goal point is (2.0, 1.8, 0.1). The red line is the path planned by the robot.

The virtual simulation is performed in different environments with dual-wheel differential drive robot Pioneer P3Ddx, as shown

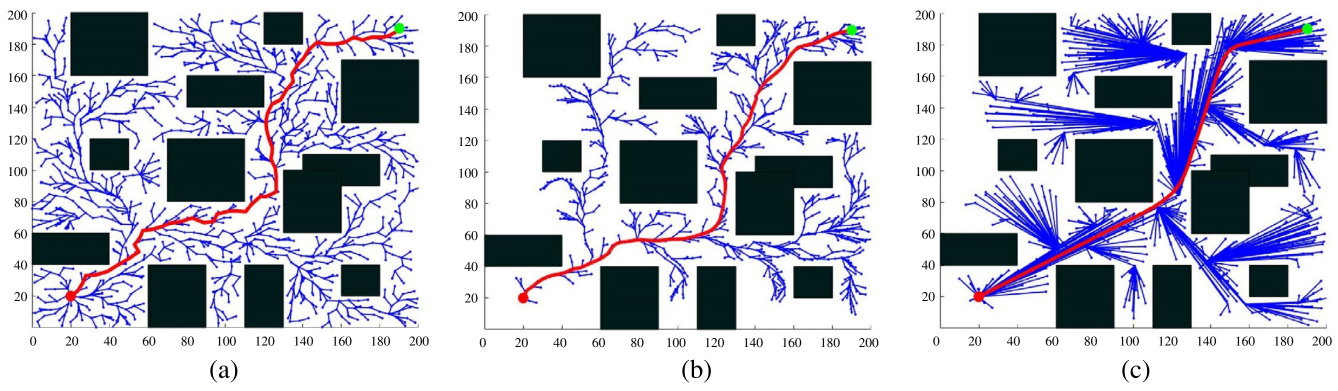


Fig. 4. Comparison of the results (a) RRT*, (b) GCSE-RRT*, and (c) Fast-RRT*

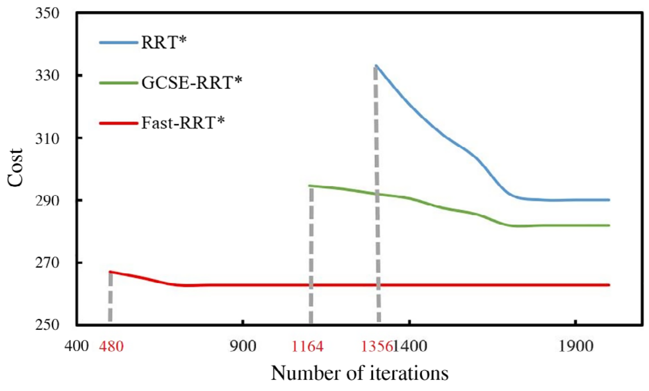


Fig. 5. The rate of convergence of RRT*, GCSE-RRT*, and Fast-RRT* (number of iterations)

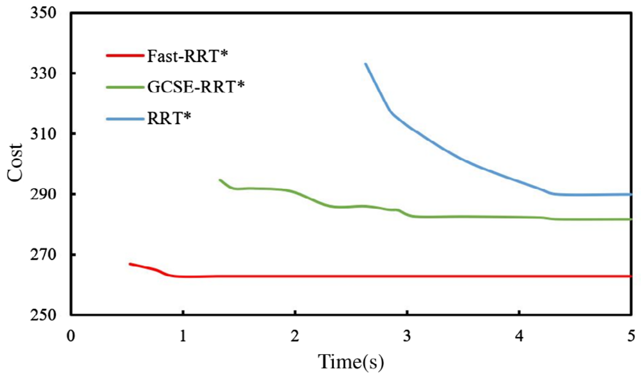


Fig. 6. The rate of convergence of RRT*, GCSE-RRT* and Fast-RRT* (time)

Table II. Compared with related work

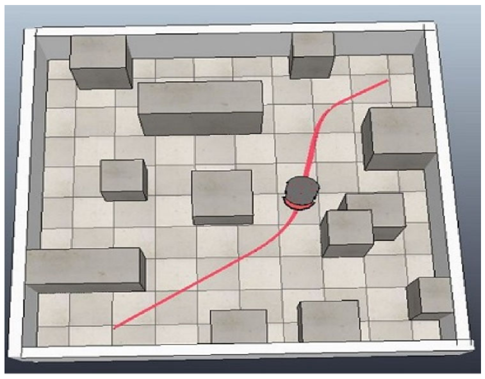
Algorithm	Cost	Time(s)
RRT*	292.12	4.47
GCSE-RRT*	275.02	4.35
Fast-RRT*	262.77	3.82

in Fig. 7. As can be seen from this, the shape of obstacles does not affect the result of path planning in virtual simulation.

The virtual simulation is performed in same environments with different mobile robot model, as shown in Fig. 8. As the shape of the robot changes, the collision detection part of this article needs to be changed. Therefore, the change of robot shape has an impact on the result of path planning. In conclusion, the method proposed in this paper can effectively implement an obstacle-free navigation plan.

4.2. Other experimental scenarios The method proposed in this paper is not only applied to the above scenario for navigation but also can be applied to cluttered environment. The RRT*, GCSE-RRT*, and the Fast-RRT* algorithm are run in different environment and the results is given in Table III. The simulation results are shown in Fig. 9.

Table III shows that the proposed algorithm can achieve a better path in a shorter time. The performance of navigation


(a)

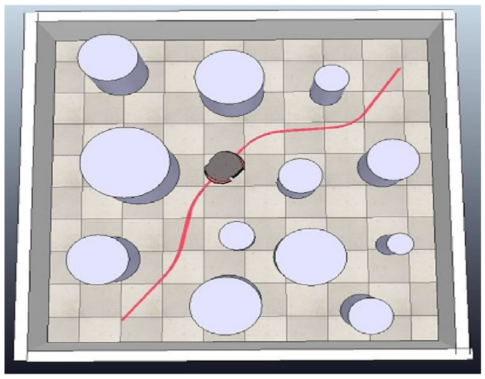
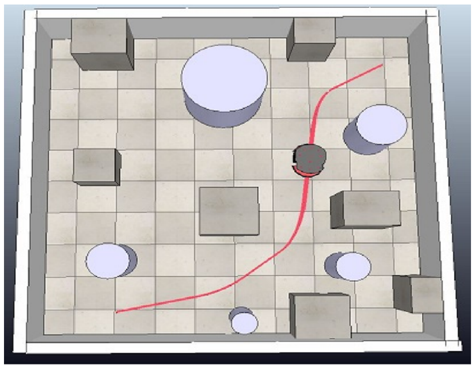

(b)

Fig. 7. Results of virtual simulation of Fast-RRT*


(a)

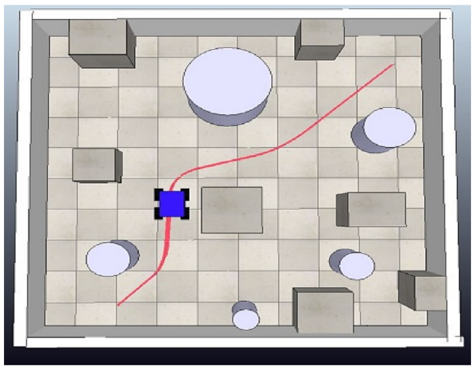

(b)

Fig. 8. Results of virtual simulation of Fast-RRT*

Table III. Compared with related work

Scenario	Algorithm	Cost	Time(s)	Iteration
Scenario 1	RRT*	308.11	0.86	623
	GCSE-RRT*	285.27	0.34	499
	Fast-RRT*	268.85	0.32	228
Scenario 2	RRT*	313.67	1.95	1504
	GCSE-RRT*	295.42	0.99	694
	Fast-RRT*	267.98	0.89	545
Scenario 3	RRT*	396.42	2.37	1764
	GCSE-RRT*	388.07	2.04	1356
	Fast-RRT*	373.51	1.58	1041

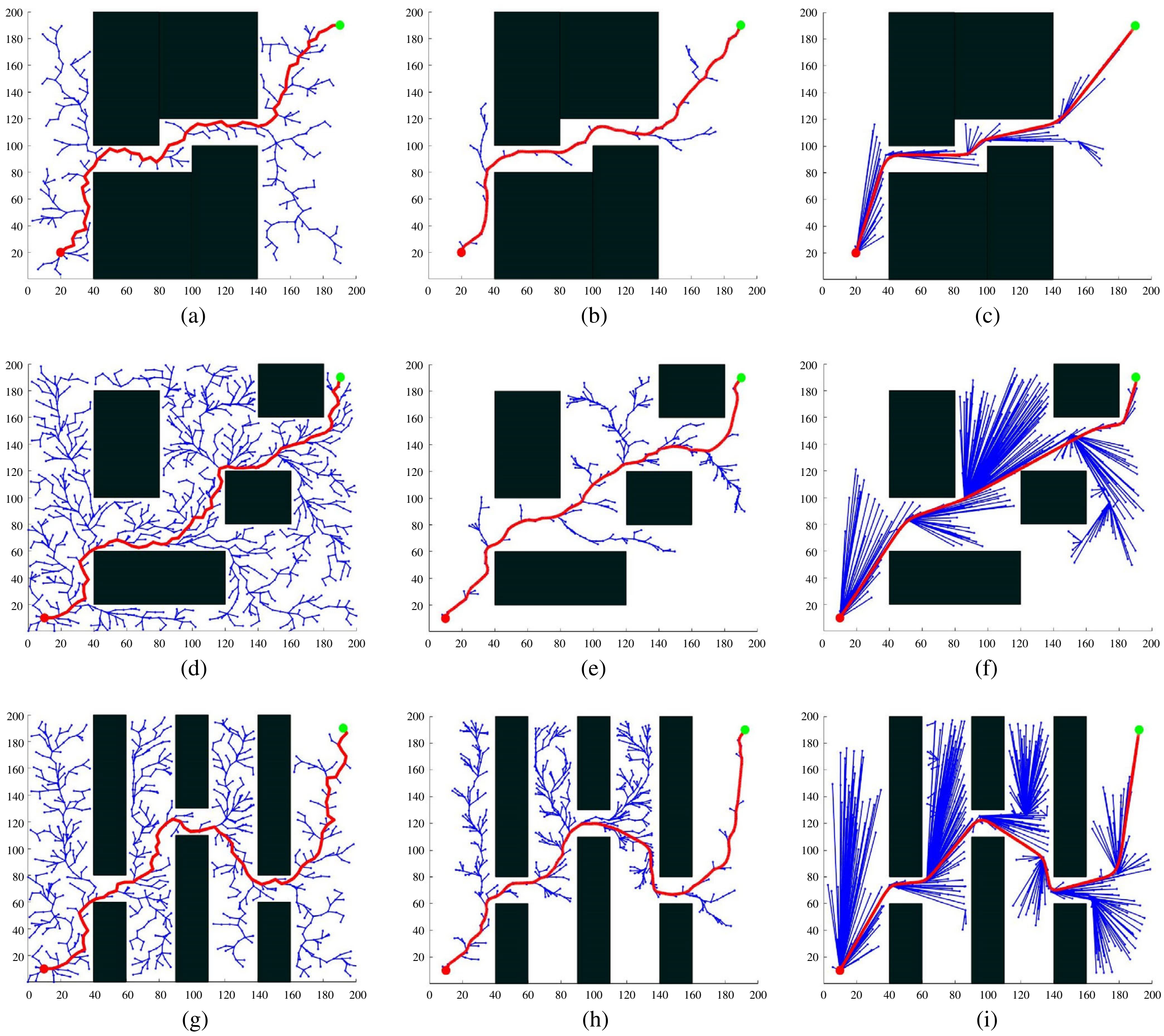


Fig. 9. Comparison of the results of RRT*, GCSE-RRT* and Fast-RRT*((a), (d), (g) show the result of RRT*, (b), (e), (h)show the result of GCSE-RRT*, and (c), (f), (i) shows the result of Fast-RRT*)

usually depends on the initial solution of path planning, since the robot will follow the initial solution at the beginning. Under the same conditions, a better initial solution means more energy savings. Furthermore, using Fast-RRT*, the approximation of an optimal solution requires fewer nodes, resulting in higher memory utilization. The quality of a path-planning algorithm depends not only on asymptotic optimality but also on the rate of convergence.

5. Conclusion

In this paper, an improved RRT*algorithm is given, abbreviated as Fast-RRT*. The main idea was to use the beneficial characteristics of RRT* while adding the advantage of constraint sampling and choosing parent based on backtracking. The proposed Fast-RRT* algorithm, while keeping the complexities as RRT*, was not only find the initial solution more quickly but also converge to the better solution quickly. Finally, cubic B-spline was used to optimize the path to ensure the stability of the path. Our work is currently applicable to the static environment, and will be applied to dynamic environment in the future. We hope to further integrate sensors to realize path planning that can be used for real-time navigation in future work. In the follow-up work, the algorithm proposed in this paper is combined with POMDP model to improve the navigation ability of this algorithm in real environment. In addition, this paper does not consider kinematic constraints, we hope to continue to explore in the future work.

Acknowledgment

This work is jointly supported by National Natural Science Foundation of China (No. 61701270), Young Doctor Cooperation Foundation of Qilu University of Technology (Shandong Academy of Sciences) (No. 2017BSHZ008).

References

(1) Tsai CC, Huang HC, Chan CK. Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. *IEEE Transactions on Industrial Electronics* 2011; **58**(10):4813–4821.

(2) Lee J, Kim DW. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. *Information Sciences* 2016; **332**(1):1–18.

(3) Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research* 1986; **5**(1):90–98.

(4) Cui P, Yan W, Wang Y. Reactive path planning approach for docking robots in unknown environment. *Journal of Advanced Transportation* 2017; **2017**:1–11.

(5) Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 1968; **4**(2):100–107.

(6) DECHTER R, PEARL J. Generalized best-first search strategies and the optimality of a*. *Journal of the ACM* 1985; **32**(3):505–536.

(7) Kavraki LE, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 1996; **12**(4):556–580.

(8) Lavalle SM. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*, vol. **98**. Department of Computer Science. Iowa State University: Ames, IA; 1998; 1–4.

(9) Kun W, Bingyin R. A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors* 2018; **18**(2):571.

(10) Xin J, Zhong J, Yang F, Cui Y, Sheng J. An improved genetic algorithm for path-planning of unmanned surface vehicle. *Sensors* 2019; **19**(11):2640.

(11) Kumar PB, Rawat H, Parhi DR. Path planning of humanoids based on artificial potential field method in unknown environments. *Expert Systems* 2019; **36**(2):1–12.

(12) Qinghua L, Yue Y, Yaqi M, Zhao Z, Chao F. A combined navigation algorithm for large concave obstacles(in Chinese). *Journal of Electronics and Information* 2020; **42**(4):917–923.

(13) Singh Y, Sharma S, Sutton R, Hatton D, Khan A. A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Engineering* 2018; **169**:187–201.

(14) Li Q, Mu Y, You Y, Zhang Z, Feng C. A hierarchical motion planning for mobile manipulator. *IEEJ Transactions on Electrical and Electronic Engineering* 2020; **15**:1390–1399.

(15) Baumann MA, Léonard S, Croft EA, Little JJ. Path planning for improved visibility using a probabilistic road map. *IEEE Transactions on Robotics* 2010; **26**(1):195–200.

(16) Jeong IB, Lee SJ, Kim JH. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Systems with Application* 2019; **123**(6):82–90.

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

(18) Nasir J, Islam F, Malik U, Ayaz Y, Hasan O, Khan M, Muhammad MS. RRT*-Smart: A rapid convergence implementation of RRT*. *International Journal of Advanced Robotic Systems* 2013; **10**(7):299–311.

(19) Noreen I, Khan A, Ryu H, Doh NL, Habib Z. Optimal path planning in cluttered environment using RRT*-AB. *Intelligent Service Robotics* 2017; **11**:41–52.

(20) Weimin Z, Chuanxiong F. Path planning of mobile robot based on improved RRT* algorithm(in Chinese). *Journal of Huazhong University of Science and Technology (Natural Science Edition)* 2021; **49**(1):31–36.

(21) Farin G, Rein G, Sapidis N, Worsey AJ. Fairing cubic B-spline curves. *Computer Aided Geometric Design* 1987; **4**(1–2):91–103.

Qinghua Li (Non-member) received the Ph.D. degree in signal and information processing from Shandong University in 2009. He is currently an associate professor with Qilu University of Technology (Shandong Academy of Sciences), Jinan, P. R. China. His current research interests include machine learning, machine vision.

Jiahui Wang (Non-member) received the B.S. degree in automation from Polytechnic College of Hebei University of Science and Technology, Shijiazhuang, China, in 2018. She is currently pursuing the M.S. degree in control engineering from Qilu University of Technology (Shandong Academy of Sciences), Jinan, China. Her research interests are path planning, motion planning, and particularly the motion planning of mobile manipulators.

Haiming Li (Non-member) received the B.S. degree in Communication engineering from the Qilu University of Technology (Shandong Academy of Sciences), Jinan, China, in 2017, where he is currently pursuing the M.S. degree with the School of Electrical Engineering and Automation. His research interests include machine learning, deep learning, and obstacle avoidance.

Binpeng Wang (Non-member) received the master degree in control theory and control engineering from Jinan University in 2004. He is currently an associate professor with Qilu University of Technology (Shandong Academy of Sciences), Jinan, P. R. China. His current research interests include intelligent control, robot application, and motion planning.

Chao Feng (Non-member) received the Ph.D. degree in signal and information processing from Shandong University in 2015. He is currently a lecturer with Qilu University of Technology (Shandong Academy of Sciences), Jinan, P. R. China. His current research interests include homomorphic encryption, remote data checking, and motion planning.