Proceedings of the 37th Chinese Control Conference
July 25-27, 2018, Wuhan, China

Asymptotical RRT-based Path Planning for Mobile Robots in Dynamic Environments

Zhuoyang Du, Shan Liu
College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027, P. R. China
E-mail: sliu@zju.edu.cn

Abstract: Rapidly-exploring random trees(RRT) is an important approach in motion planning. However, the uniform sampling strategy in conventional RRT methods leads to a low sampling efficiency in some situations and the solution is sub-optimal. Besides, conventional methods usually do not apply to dynamic environments. This paper presents an asymptotical RRT based path planning method for mobile robots in dynamic environments. To increase the sampling efficiency, a heuristic sampling algorithm is proposed on the basis of the prior knowledge of the previous planning results and the potential field of the environment. In addition, a post-processing procedure is adopted to optimize the path in consideration of path length and safety. Besides, a replanning strategy is presented to deal with the moving obstacles in dynamic environments. The simulation result shows that the heuristic sampling method has a higher sampling efficiency and faster rate of convergency than uniform sampling, a feasible path can be generated in consideration of safety and smoothness.

Key Words: Motion Planning, Mobile Robot, RRT, Artificial Potential Field

1 Introduction

Motion planning problem is a fundamental component in the research of mobile robots[1–3]. The purpose of motion planning is to generate desired trajectory for mobile robots from the current configuration to the desired configuration. Generally, the trajectory should be safe (no collision with obstacles) and feasible (satisfying kinematic and dynamic constraints). Over the past three decades, various path planning methods have been proposed, including graph based methods[4, 5], sampling based methods[6–10], optimization based methods[12], potential field based methods[13], etc.

Graph based methods, such as Dijkstra[4] and A*[5], have been widely applied in robotics. Graph based methods are easy to find the optimal solution if one exists. However, the disadvantage is that the discretization of the configuration space is required, so that the resulting path is not smooth and difficult to be tracked. In addition, the searching efficiency decreases significantly when the searching area is large.

Sampling based planning is a randomized approach which works by sampling the configuration space. Compared with graph based planning, the advantage of sampling based planning is that it doesn't need discretization of the configuration space, while the drawback of this approach is that the solution is suboptimal and the completeness of the solution cannot be guaranteed. Rapidly-exploring Random Trees(RRT) [7] is one of the most commonly used sampling based methods, which is a single-query algorithm based on incremental sampling. The advantage of RRT is its fast planning speed. Besides, kinodynamic constraints can also be considered during the planning procedure[14]. However, the solution is usually suboptimal. To overcome this shortcoming, many algorithms have been proposed. Rapidly-exploring Random Trees Star(RRT*)[9] is presented based on standard RRT by Sertaz and Karaman in 2011, which has been proven to be probabilistic completeness and asymptotically optimal. However, in this method, the rate of convergence is slow. To increase the rate of convergence and optimize the final solution, Jonathan[8] presented a modified RRT* method which makes use of the previous feasible path and restricts the sampling area. In addition, traditional sampling based planning methods are proposed with uniform sampling strategies, which lead to the fact that the sampling probability of narrow regions is much smaller than wide regions. To improve the sampling efficiency, various heuristic sampling strategies have been proposed, such as local biasing[10], goal biasing[11] and obstacle boundary biasing.

Optimization based methods usually compute trajectories by minimizing or maximizing a optimization function[12]. Different constraints can be considered in the optimization function, such as path smoothness, kinematic constraints and collision-free constraints. Wu[15] presented an improved RRT-Tailor-Spline algorithm to increase the path smoothness, in which a B-spline method is used to optimize the previous generated path.

Potential field planning was first proposed by Khatib[13] in 1986. The main idea of the artificial potential field is to construct an attractive potential field at the target point and repulsive potential fields on the obstacles. This method is popular for its efficient mathematical analysis and simplicity. However, it suffers from local minima. Besides, nonholonomic constraints of mobile robots are also difficult to be taken into consideration in potential field methods.

This paper focuses on an asymptotical RRT-based path planning method for mobile robots in dynamic environments. The algorithm consists of three sub-processes: preprocessing, RRT-based path searching and post-processing. The core of this method is RRT-based path searching with heuristic sampling, which is to increase the sampling efficiency and the rate of convergence. Then, the previous generated path is optimized in consideration of path smoothness and safety in post-processing procedure. Finally, in order to deal with the moving obstacles and path tracking error in the control process, the path is replanned periodically. At the beginning of every replanning period, a pre-processing procedure is adopted to update the RRT tree of the previous iteration.

This work is supported by the Public Welfare Technology Application Research Project of Zhejiang Province, China (Grant No. 2017C31047).

2 Task Definition

Denote $\mathcal{X}$ as the configuration space, which describes all possible configurations in the workspace. Denote $\mathcal{X}_{free}$ and $\mathcal{X}_{obs}$ as the free and obstacle spaces, respectively. Due to the moving obstacles in the environment, the obstacle space is time-varying, as well as the free space. Let $x_{init} \in \mathcal{X}_{free}$ and $x_{goal} \subseteq \mathcal{X}_{free}$ be the initial configuration and the goal configuration, respectively. The task of path planning is to find a collision-free path which is defined as $\sigma : [0, 1] \to \mathcal{X}$, with $\sigma(0) = x_{init}$ and $\sigma(1) \in x_{goal}$. The path optimization considers not only path length and smoothness, but also the safety of the path.

Since the environment is usually time-varying in real applications, the path planned in the previous may collide with dynamic obstacles during the moving process of the robot. In addition, due to limited control effort, the tracking error is inevitable. Therefore, it is necessary to replan the path iteratively according to the change of environment and current state of the robot.

3 Heuristic RRT-based Path Searching

Conventional RRT methods sample the workspace with uniformly distributed probability, which takes no account of the environment information and the previously planned path in dynamic environment. This leads to a low sampling efficiency in several situations, e.g., narrow regions. To increase the sampling efficiency, the environment information and the previous planning results are used to construct a probabilistic map to direct the sampling process. In the heuristic RRT based path searching, the uniform sampling procedure is replaced by a heuristic sampling procedure on the basis of a probabilistic map.The algorithm is introduced in detail in this section.

3.1 Sampling Probabilistic Map

The sampling procedure randomly select a configuration to explore the workspace efficiently. Traditionally, the sampling distribution is uniform, therefore, the selection of sample takes no account of the environment information, and the sampling is inclined to be in the wide region in the workspace rather than the narrow region. Besides, once feasible paths have been generated, if the sum of the distances between a new node and the initial state and between the node and the target state is larger than the length of the current shortest path, The new path would not be a shorter path than the current shortest one.

To overcome the above shortcomings, a probabilistic map is proposed in this paper for heuristic sampling, which is constructed by an attractive artificial potential field and the prior knowledge of the current shortest path length.

3.1.1 Attractive Potential Field

To increase computational efficiency and take environment information into consideration in the searching process, an artificial potential field is adopted to construct the probabilistic map. In this paper, only an attractive potential field is used for heuristic sampling, which is constructed based on the target and the Voronoi graph of the environment. As a result, the random sampling is biased in the regions near the target and far from the obstacle.

The attractive potential field $U_{art}(x)$ is constructed as follows:

$$U_{art}(x) = U_{goal}(x) + U_{vor}(x) \tag{1}$$

where $U_{goal}(x)$ is the attractive potential field of the target and $U_{vor}(x)$ is the attractive potential field of the Voronoi graph.

$U_{goal}(x)$ is defined as follows:

$$U_{goal}(x) = \frac{1}{2} k_{goal} \left(x - x_{goal}\right)^2 \tag{2}$$

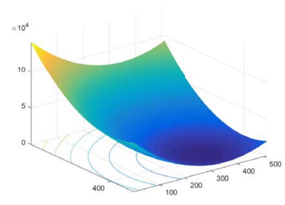
with k_{goal} being the constant weighting factor.

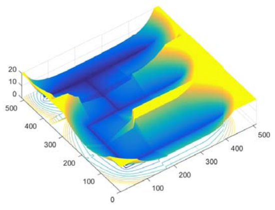
$U_{vor}(x)$ is defined as follows:

$$U_{vor}(x) = \frac{1}{2} k_{vor} \Sigma_{x_v \in \chi_v} \left(x - x_v\right)^2 \tag{3}$$

with k_{vor} being the constant weighting factor, χ_v being the set of points on the voronoi graph in the workspace.

Using the above definition of the artificial potential field, the sampling probability is biased at the regions near the target and far from the obstacles, as shown in Fig. 1.


(a) Attractive potential field of the target


(b) Attractive potential field of Voronoi graph

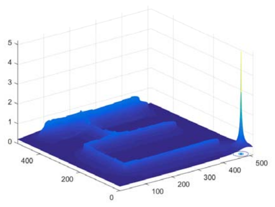

(c) Probabilistic map

Fig. 1: Artificial potential field

3.1.2 Path Prior

In the searching process, the feasible paths are searched out iteratively. Once a feasible path is found, the following searching process should ignore those path candidates that are longer than the current shortest path, so that the path searching could be more effective to achieve asymptotical path searching.

An ellipse region $\mathcal{C}$ is defined to constrain the sampling candidates, as follows:

$$\mathcal{C} = \{x \in \chi | \|x - x_{init}\| + \|x - x_{goal}\| < length(\sigma_{min})\} \tag{4}$$

Then the probabilistic map is generated as follows:

$$P(x) = \begin{cases} \frac{1}{U_{art}(x)+\epsilon} & \text{if} \quad x \in \mathcal{C} \\ 0 & \text{if} \quad x \notin \mathcal{C} \end{cases} \tag{5}$$

where ϵ is a small value to avoid singularity.

3.2 RRT-based Path Searching

```
Algorithm 1: RRT Searching
Input: $x_{current}, x_{goal}, \mathcal{X}_{obs}, \mathcal{X}_{free}, \sigma_{min}, \mathcal{T}$
Output: $\mathcal{T}, \sigma_{min}$
1  for iteration = 1 ... N do
2      $P \leftarrow$ GenerateProbabilisticMap($\mathcal{X}, x_{goal}, \sigma_{min}$);
3      $x_{rand} \leftarrow$ RandomSample($P$);
4      $x_{nearest} \leftarrow$ FindNearestNode($\mathcal{T}, x_{rand}$);
5      $x_{new} \leftarrow$ Steer($x_{nearest}, x_{new}$);
6      if CollisionCheck($x_{nearest}, x_{rand}$) then
7          $V \leftarrow \cup x_{new}$;
8          $E \leftarrow \cup(x_{nearest}, x_{rand})$;
9      if Distance($x_{new}, x_{goal}$) $> \varepsilon$ then
10         $\sigma_{min} \leftarrow$ GeneratePath($x_{new}, \mathcal{T}$);
11         if length ($\sigma_{new}$) $<$ length ($\sigma_{min}$) then
12             $\sigma_{min} = \sigma_{new}$;
13 return $\mathcal{T}, \sigma_{min}$;
```

The RRT based path searching method is presented in Algorithm 1. Given an initial state as the initial vertex, the RRT based searching generates a tree $\mathcal{T} = (V, E)$ incrementally, which is composed of a vertex set V of states from X_{free} connected by edges $E \subseteq V \times V$.

The searching process is detailed in Algorithm 2. In each iteration, the algorithm first calls *GenerateProbabilisticMap* function to generate a heuristic probabilistic map P using the method in Section 4.1 (Line 2). Based on the probabilistic map P, a sample x_{rand} is generated (Line 3). Then, the function *FindNearestNode* is called to search for the nearest feasible node $x_{nearest}$ to this sample x_{rand} in the current tree(Line 4). In addition, the nonholonomic constraint is taken into consideration during this procedure, only the node satisfying the angle constraint will be chosen. Afterwards, the function *Steer* is called and returns a new node x_{new}(Line 5). If the edge connecting $x_{nearest}$ and x_{new} does not collide with obstacles (Line 6), the vertex x_{new} and its corresponding edge $(x_{nearset}, x_{new})$ is added to the tree $\mathcal{T}$ (Line 7-8). On the other hand, if the distance between x_{new} and x_{goal} is small enough then it is regarded to have reached the goal (Line 9). Then a new path σ_{new} is generated (Line 10). If the length of the new path σ_{new} is shorter than the current shortest path σ_{min}(Line 11), the current shortest path is updated (Line 12).

The functions contained in the Algorithm 1 are described in detail as follows:

Generate Probabilistic Map: The function GenerateProbabilisticMap($\chi, x_{init}, x_{goal}, \sigma_{min}$) returns a probabilistic map P, which contains the prior knowledge of current shortest path σ_{min}, as well as the artificial potential field calculated by the configuration of the goal x_{goal} and the Voronoi graph of the environment.

Heuristic Sampling: The function Sampling(P) returns a sample $x_{rand} \in \mathcal{X}$ depending on the probabilistic map P.

Nearest Neighbor: Given a tree $\mathcal{T} = (V, E)$, the function of Nearest returns a vertex $v \in V$ which is the closest point to the sample x_{rand}. Usually, we use Euclidean distance to represent the distance between nodes. However, the node with the nearest Euclidean distance might lead to a large orientation change which is difficult for a nonholonomic robot to realize. Therefore, in this algorithm, the nearest node is chosen from those nodes whose orientation change is less than $\pi/2$.

Steering: In the steering procedure, x_{rand} and $x_{nearest}$ are connected to get a new node x_{new}.

CollisionChecking: Given a new node x_{new}, collision checking is performed to ensure that the path between x_{new} and its parent node $x_{nearest}$ will not collide with obstacles. If the path is collision-free, the node x_{new} is added to the tree $\mathcal{T}$.

PathGeneration: If the path between x_{new} and $x_{nearest}$ is collision free and $x_{new} = x_{goal}$, a feasible path σ_{new} is obtained. σ_{min} is updated by σ_{new} if the length of σ_{new} is smaller than σ_{min}.

4 Pre-processing

```
Algorithm 2: Pre-processing
Input: $\mathcal{T}$
Output: $\mathcal{T}_{new}, \sigma_{min}$
1 $x_{current}, x_{goal}, \mathcal{X}_{obs}, \mathcal{X}_{free} \leftarrow$ UpdateStates()
2 $\mathcal{T}_{new} \leftarrow$ UpdateTree($\mathcal{T}, x_{current}$)
3 $\sigma_{min} \leftarrow$ UpdateFeasiblePath($\mathcal{T}_{new}, x_{goal}$)
4 return $\mathcal{T}_{new}, \sigma_{min}$;
```

Due to the varying environment caused by moving obstacles, it is necessary to replan the path periodically. The previous RRT-tree contains abundant information about the environment, therefore, it is reserved and updated according to the change of environment and the current state of the robot. Besides, in the path tracking process, control errors are unavoidable. Thus the previous RRT-tree is also updated according to the current state of the robot.

The pre-processing consists of three procedures as shown in Algorithm 2. First of all, the function *UpdateState* is called to update the current state $x_{current}$ and target state x_{goal} of the mobile robot, as well as the environment information $\mathcal{X}_{free}$ and $\mathcal{X}_{obs}$(Line 1), so that the change of the environment, i.e., dynamic obstacles, will be taken into consideration during the replanning procedure. Then, to make use of the previous planning result, the function *UpdateTree* is performed according to the current state of the robot $x_{current}$ and the environment information X_{obs}(Line 2). The current state $x_{current}$ will be the initial vertex of the new tree $\mathcal{T}_{new}$, and the previous vertexes are connected to the initial node $x_{current}$ if the nonholonomic constraint is satisfied, the child nodes of those vertexes are reserved as well. Further more, the vertexes which collide with the obstacles will be removed from the tree along with their child nodes. At last, the function *UpdateFeasiblePath* is called to search for the current shortest feasible path in the tree (Line 3), which will be used as the prior knowledge to construct the probabilistic map.

5 Post-Processing

The result of RRT searching is a sub-optimal path in terms of path length and smoothness, which is a main drawback of most RRT methods. Therefore, a B-spline based post processing method is proposed to further optimize the resulting path in terms of path length, smoothness, and safety.

5.1 B-spline Based Path Parameterization

In this process, the path is fitted by a B-spline curve, which is defined by a set of control points $\{c_{j-2}\}_{j\in[1,M]}$. Initially, the control points are chosen as the nodes in the sub-optimal path. Denoting Φ_x and Φ_y as the x and y coordinates of the control points $\{c_{j-2}\}_{j\in[1,M]}$:

$$\begin{cases}\Phi_x = [\phi_{x,-1} \quad \phi_{x,0} \quad \cdots \quad \phi_{x,m-1}] \\ \Phi_y = [\phi_{y,-1} \quad \phi_{y,0} \quad \cdots \quad \phi_{y,m-1}]\end{cases} \tag{6}$$

Denote $s \in [0,1]$ as the normalized path length starting from the initial state to the target state, then the approximate functions are defined as:

$$\begin{cases}x(s) = \sum_{k=0}^{3} f_k(t)\cdot \phi_{x,k+l} \\ y(s) = \sum_{k=0}^{3} f_k(t)\cdot \phi_{y,k+l}\end{cases} \tag{7}$$

where:

$$t = \frac{s}{\mu} - \left\lfloor \frac{s}{\mu} \right\rfloor, l = \left\lfloor \frac{s}{\mu} \right\rfloor - 1 \tag{8}$$

with $\lfloor\cdot\rfloor$ denoting the round down function, and $\{f_k(t))\}_{k=[0,3]}$ are basis functions[16].

5.2 Path Optimization

In order to optimize path length and smoothness under the collision free constraint, the optimization function is defined by the length of the path and a potential energy along the path, which is generated by the repulsive potential field of the obstacles:

$$E(\Phi) = L(\Phi) + \lambda \cdot P(\Phi) \tag{9}$$

where $L(\Phi)$ is the path length, and $P(\Phi)$ is the potential energy along the path. $\lambda \in R^+$ represents the tradeoff between path length and safety, a larger λ enforces the path to be farther away from the obstacles, while in the meantime the path length may be larger. Owing to the B-spline based curve parameterization, a shorter path generally leads to a smoother one. And a path with smaller potential energy generally leads to a safer one. The cost term for path length is defined as follows:

$$L = \int_0^1 \left[\dot{x}^2(s) + \dot{y}^2(s)\right] ds \tag{10}$$

The cost term for potential energy $P(\Phi)$ is defined as:

$$P(\Phi) = \int_0^1 P\left(x(s), y(s)\right) ds \tag{11}$$

For computational efficiency, the following approximate form is used for optimization. Pick a set of points $\{O_i\}_{i\in[1,N]}$ uniformly along the path as $O_i\left(s_i, x_i, y_i\right)$, then the objective function is defined as:

$$L = \sum_{i=1}^{N} \left(\dot{x}_i^2 + \dot{y}_i^2\right)\cdot d \tag{12}$$

where d is the interval between the control points.

$$P = \sum_{i=1}^{N} P\left(x_i, y_i\right) \tag{13}$$

$$P(x_i, y_i) = \begin{cases}\frac{1}{2}\eta(\frac{1}{\rho_i} - \frac{1}{\rho_0})^2 & \text{if} \quad \rho_i \le \rho_0 \\ 0 & \text{if} \quad \rho_i > \rho_0\end{cases} \tag{14}$$

where ρ_i represents the shortest distance of the point (x_i, y_i) to the obstacle, ρ_0 represents the distance limit of the potential field influence and η is a constant gain.

Then the coordinates of the points can be expressed as follows:

$$\begin{cases}X = [x_1, \cdots, x_N]^T = F\cdot\Phi_x \\ Y = [y_1, \cdots, y_N]^T = F\cdot\Phi_y\end{cases} \tag{15}$$

where F is $N\times M$, with F_i defined as:

$$F_i = \begin{bmatrix}\underbrace{0 \quad \ldots \quad 0}_{i-1} & f_0(t_i) & f_1(t_i) & f_2(t_i) & f_3(t_i) & \underbrace{0 \quad \ldots \quad 0}_{M-i-3}\end{bmatrix} \tag{16}$$

Besides, the derivatives with respect to the arc length are expressed as follows:

$$\begin{cases}\dot{X} = [\dot{x_1}, \cdots, \dot{x_N}]^T = F'\cdot\Phi_x \\ \dot{Y} = [\dot{y_1}, \cdots, \dot{y_N}]^T = F'\cdot\Phi_y\end{cases} \tag{17}$$

where F' is $N\times M$, with its i-th row defined as:

$$F_i = \begin{bmatrix}\underbrace{0 \quad \ldots \quad 0}_{i-1} & \frac{f_0'(t_i)}{\mu} & \frac{f_1'(t_i)}{\mu} & \frac{f_2'(t_i)}{\mu} & \frac{f_3'(t_i)}{\mu} & \underbrace{0 \quad \ldots \quad 0}_{M-i-3}\end{bmatrix} \tag{18}$$

with $f'(t_i) = \frac{\partial f(t_i)}{\partial t_i}$. Then the curve is parameterized by $\Phi = \begin{bmatrix}\frac{\partial L}{\partial \Phi_x} & \frac{\partial L}{\partial \Phi_y}\end{bmatrix}^T$

For the optimization of the objective function defined in (9), gradient based methods can be used. Then the partial derivative of the objective function E with respect to the parameter Φ can be derived as follows:

$$\frac{\partial E}{\partial \Phi} = \frac{\partial L}{\partial \Phi} + \lambda\frac{\partial P}{\partial \Phi} \tag{19}$$

$\frac{\partial L}{\partial \Phi}$ is computed as follows:

$$\frac{\partial L}{\partial \Phi_x} = \Sigma_{i=1}^{N} 2\cdot\dot{x}_i\cdot\frac{\partial \dot{x}_i}{\partial \Phi_x} \tag{20}$$

$$\frac{\partial L}{\partial \Phi_y} = \Sigma_{i=1}^{N} 2\cdot\dot{y}_i\cdot\frac{\partial \dot{y}_i}{\partial \Phi_y} \tag{21}$$

$\frac{\partial P}{\partial \Phi}$ is computed as follows:

$$\frac{\partial P}{\partial \Phi_x} = \sum_{i=1}^{N}\frac{\partial P}{\partial x_i}\cdot\frac{\partial x_i}{\partial \Phi_x} \tag{22}$$

$$\frac{\partial P}{\partial \Phi_y} = \sum_{i=1}^{N}\frac{\partial P}{\partial y_i}\cdot\frac{\partial y_i}{\partial \Phi_y} \tag{23}$$

The update rule for Φ is given by:

$$\Phi_{k+1} = \Phi_k - \alpha\nabla E(\Phi_k) \tag{24}$$

where α is the step size and $-\nabla E(\Phi)$ is the negative gradient of E at Φ_k .

6 Simulation Evaluations

To evaluate the performance of the asymptotical RRT-based path planning algorithm, simulations are conducted in MATLAB and V-REP[17]. Firstly, we implement RRT based planning with uniform sampling and heuristic sampling methods to show the benefit of the probabilistic map. Then, the result of post-processing is shown. Finally, moving obstacles are taken into consideration, and the asymptotic RRT-based path planning in dynamic environments is implemented.

In the following figures, the solid black blocks represent obstacles in the environments, while the blue and red points represent the initial state x_{init} and the target state x_{goal}, respectively. The paths are represented by the red lines, while the trees expanded in the procedure is represented by the blue lines.

6.1 Performance Comparison between Uniform Sampling and Heuristic Sampling

To verify the efficiency of the heuristic sampling proposed in this algorithm, we compared our method with standard uniform sampling method. Fig. 2(a) and fig. 2(b) show the situations when the first path is computed via uniform sampling and heuristic sampling, respectively. It can be seen from the result that the heuristic sampling method in this paper has a faster searching speed. This is due to the attractive potential field used for heuristic sampling, which increases the sampling probability near the target and in the safe region, especially the narrow region.

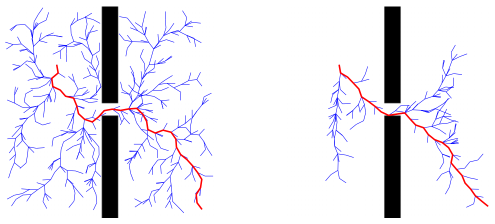
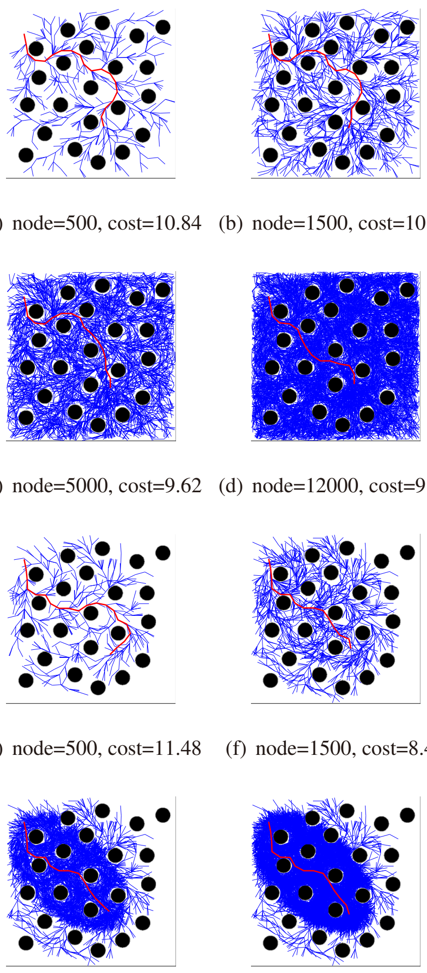

(a) (b)

Fig. 2: (a)The initial path found by uniform sampling based RRT at 582 iterations. (b)The initial path found by heuristic sampling based RRT at 209 iterations.

Fig. 3 is a comparison of the asymptotic RRT based path planning in static environments with and without the prior knowledge of the current shortest path. Even with a less optimal first solution in fig. 3(e), the searching with prior knowledge converges to a cost of 8.40 after 1500 iterations, while the searching result without prior knowledge has a cost of 10.82 after the same iterations. In fig. 3(d), the path of the method without prior knowledge would also converge to a smaller cost of 9.41, but it takes more iterations than the method with prior knowledge. As shown in fig. 3(e)-fig. 3(h), in the method with prior knowledge, the sampling region is restricted by the current shortest feasible path to increase the sampling efficiency, so that the method with prior knowledge has a faster rate of convergency.


(a) node=500, cost=10.84 (b) node=1500, cost=10.82

(c) node=5000, cost=9.62 (d) node=12000, cost=9.41

(e) node=500, cost=11.48 (f) node=1500, cost=8.40

(g) node=5000, cost=8.20 (h) node=12000, cost=8.20

Fig. 3: A comparison of asymptotic RRT path planning with and without prior knowledge. (a)-(d) Results without prior knowledge at 500, 1500, 5000 and 12000 iterations. (e)-(d) Results with prior knowledge at 500, 1500, 5000 and 12000 iterations.

6.2 Post-processing

Simulations are conducted to illustrate the effect of the repulsive potential field used in post processing. As shown in fig. 4, the dotted line and the solid lines represent the paths before and after post-processing, respectively. The red, yellow and green lines represent the optimization results when λ is 20, 100 and 200, respectively. When λ gets larger, the influence of the potential field increases, leading to the result that the optimized path gets farther away from the obstacles, but the path length gets larger.

6.3 RRT-based Planning in Dynamic Environments

Fig. 5 shows the performance of this algorithm in dynamic environments. The black point represents current position of the mobile robot while the red point represent the goal. The moving obstacles are represented by blue solid regions and their moving directions are illustrated by arrows. The red line is the trajectory generated during the replanning period.

Because of the replanning strategy, the change of the environments is updated at the beginning of every iteration. And the RRT tree is updated on the basis of the environment

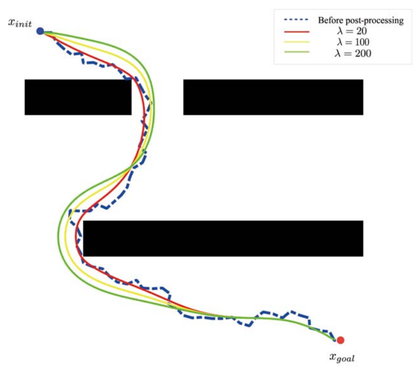
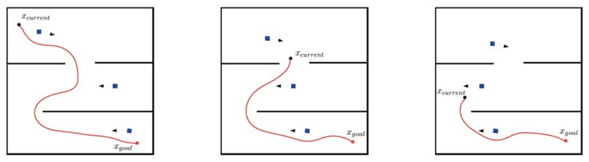


Fig. 4: Post-processing

map, that is, the edges that collide with the obstacles will be removed from the tree while the others will be remained. In this way, the planner can deal with the moving obstacles in the environments, and also, the useful part of the original tree can be reserved.



(a) (b) (c)

Fig. 5: Asymptotical RRT based path planning in dynamic environments

7 Conclusion

An asymptotical RRT based path planning algorithm is proposed in this paper. Instead of uniform sampling strategy in conventional RRT methods, we present a heuristic sampling method based on a probabilistic map. The probabilistic map contains an attractive potential field of the Voronoi graph of the environment and an attractive potential field generated by the goal. In addition, the prior knowledge of the previous planning result is also taken into consideration during the generation of the probabilistic map. It has been shown that the heuristic sampling method in this paper not only has a quicker searching speed than the traditional uniform sampling method, but also has a higher rate of convergency.

Furthermore, a post-processing procedure is adopted in this paper to optimize the previous generated path. An optimization function is defined by the path length and an repulsive potential field of the obstacles. Via the post-processing procedure, the sub-optimal path is optimized so that it can satisfy the path smoothness constraints and kinematic constraints of mobile robots and guarantee the safety of the robots.

In consideration of the moving obstacles in the dynamic environments and tracking error, an replanning approach is conducted. Before every iteration of the replanning, a preprocessing procedure is presented to update the previous RRT tree on the basis of the change of the environment information. In addition, the current shortest path is also found for the probabilistic map. Simulation results have illustrated the performance of this algorithm in dynamic environments.

References

[1] S. M. LaValle, *Planning Algorithms*, Cambridge, U.K.: Cambridge Univ. Press, 2006.

[2] B. Paden, S. Z. Yong, D. Yershov, and E. Frazzoli, A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles, *IEEE Trans. on Intelligent Vehicles*, 1(1): 33-55, 2016.

[3] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, A Review of Motion Planning Techniques for Automated Vehicles, *IEEE Trans. on Intelligent Transportation Systems*, 17(4): 1135-1145, 2016.

[4] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, *Numerische Mathematik*, 1(1): 269-271, 1959.

[5] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, Practical search techniques in path planning for autonomous driving, *Ann Arbor*, 1001(48105): 18-80, 2008.

[6] M. Elbanhawi and M. Simic, Sampling-based Robot Motion Planning: A Review, *IEEE Access*, 2: 56-77, 2014.

[7] S. M. LaValle, Rapidly-exploring Random Trees: A New Tool for Path Planning, 1998.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic, in *IEEE/RSJ International Conference on Intelligent Robots and Systems IEEE*, 2014: 2997-3004.

[9] S. Karaman and E. Frazzoli, Sampling-based Algorithms for Optimal Motion Planning, *The International Journal of Robotics Research*, 30(7): 846-894, 2011.

[10] B. Akgun and M. Stilman, Sampling Heuristics for Optimal Motion Planning in High Dimensions, in *IEEE/RSJ International Conference on Intelligent Robots and Systems IEEE*, 2011: 2640-2645.

[11] J. J. Kuffner and S. M. LaValle, RRT-connect: An Efficient Approach to Single-query Path Planning, in *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA IEEE*, 2000: 995-1001.

[12] N. Ratliff, M. Zucker, J. A. Bagnell, et al, CHOMP: Gradient Optimization Techniques for Efficient Motion Planning, in *IEEE International Conference on Robotics and Automation IEEE Press*, 2009: 489-494.

[13] O. Khatib, Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *Autonomous Robot Vehicles*, 1986: 396-404.

[14] Z. Yan, B. Hao, W. Zhang and S. X. Yang, Dubins-RRT Path Planning and Heading-Vector Control Guidance for a UUV Recovery, *International Journal of Robotics and Automation*, 31(3): 251-262, 2016.

[15] D. Wu, Y. Sun, X. Wang and X. Wang, An Improved RRT Algorithm for Crane Path Planning, *International Journal of Robotics and Automation*, 31(2): 84-92, 2016.

[16] S. Lee, Scattered Data Interpolation with Multilevel B-splines, *IEEE Trans. on Visualization and Computer Graphics*, 3(3): 228-244, 1997.

[17] E. Rohmer, S. P. N. Singh, M. Freese, V-REP: A Versatile and Scalable Robot Simulation Framework, in *IEEE/RSJ International Conference on Intelligent Robots and Systems IEEE*, 2013: 1321-1326.