Optimization of Probabilistic Roadmap Based on Two-Dimensional Static Environment

Binpeng Wang[1], Houqin Huang[1(✉)], Lin Sun[2], and Chao Feng[1]

[1] Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China
1371236157@qq.com

[2] Jinan Engineering Consulting Institute, Jinan 250353, China

Abstract. To address the problems of slow planning speed and too many sharp turns in the planned route, this paper focuses on the optimization of the probabilistic roadmap by searching the neighboring nodes in the composition stage, improving its search efficiency using K-dimensional Tree (KD-TREE), smoothing the planned paths, and ensuring the safety of the planned route by expanding the map obstacles. To test the performance of the improved probabilistic roadmap algorithm, it is compared with the traditional PRM algorithm and the PRM based on the common K-Nearest Neighbor (KNN) algorithm. The simulation results show that the optimized algorithm has a significant improvement in the planning time and the final planned path is a smooth path without inflection points, which is more conducive to the actual walking of the mobile robot. The study has a wide range of applications.

Keywords: Path planning · Probabilistic Roadmap · K-dimensional Tree · K-Nearest Neighbor · Bezier Curve

1 Introduction

With the rapid development of science and technology in recent years, path planning for robots, a fundamental research topic for mobile robots, has also gained a wide scope of development. It is able to generate a collision-free trajectory between the starting point and the target point in the presence of obstacles. It has a wide range of applications in smart cars, agricultural harvesting, etc. There are many methods of path planning, classical ones are: A* algorithm [1], Artificial Potential Field method (APF) [2], Grid method [3], etc.; intelligent algorithm-based ones are: Ant Colony algorithm [4], Genetic algorithm [5], etc.; sampling-based planning methods are: Rapid-exploration Random Tree (RRT) [6], Probabilistic Roadmap (PRM) [7], etc. Compared with the classical algorithms that need to know the location of obstacles and model them accurately and intelligent algorithms that are extremely computational, PRM algorithms based on sampling techniques can effectively solve the path planning problems in high space and complex constraints.

As more scholars have researched, various improvement schemes have emerged for the PRM algorithm. Mohanta J [8] proposed to combine the PRM with a fuzzy control

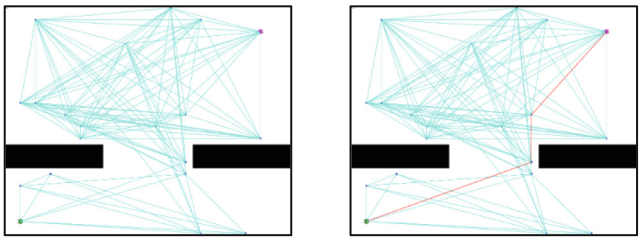
B. Wang et al. (Eds.): ICMTEL 2023, LNICST 535, pp. 392–398, 2024.
https://doi.org/10.1007/978-3-031-50580-5_35

system to make the planned path smoother and thus save the path cost; Boor and David Hsu [9] improved the planning success rate of the PRM in complex regions by this method of changing the sampling strategy. Our improved scheme mainly focuses on the construction of the undirected roadmap. The construction of the traditional undirected roadmap requires each pair of sampling points to be connected and collision detected, which is very time-consuming and most of the sampling points that are far away cannot form a valid path, resulting in wasted collision cost. We will optimize it using the K-dimensional Tree (KD-TREE) algorithm [10] to search for each sampled band of your nearest neighbor points and complete the composition with great speed. In addition, we increase the safety of the path with the expansion obstacle method and smooth our planned path with the Bezier Curve algorithm. The simulation results show that our improved scheme has a significant improvement in time cost and better rationalization of the path.

2 Related Knowledge

2.1 Probabilistic Roadmap

Probabilistic Roadmap is one of the underlying algorithms commonly used in robot path planning. PRM is usually divided into two stages: preprocessing stage and search stage. The main goal of the preprocessing stage is to construct the undirected graph G(V, E). Among them, the vertex set V represents the collision-free configuration randomly sampled in the free space C_{free}, and the connection line set E represents the collision-free path in the graph. First, sample n collision-free configurations α_i in the free space C_{free} and add them to the vertex set V, where n represents the total number of sampled configurations, and i represents the serial number of the configuration being sampled. Then, connect the configurations in the vertex set and perform collision detection. If there is no collision, add them to the connection set E. After all the sampling points complete the above steps, an undirected graph will be obtained, as shown in Fig. 1(a). The second step is to use the A* algorithm to find the optimal path, as shown in Fig. 1(b). If a collision-free path can be found between the given start point and end point, it means that the planning is successful and there is an executable motion planning scheme.



(a) (b)

Fig. 1. (a) Undirected graphs of PRM, (b) The final path of the PRM

2.2 K-Nearest Neighbor and K-Dimensional Tree

In 1968, Cover and Hart proposed the K-Nearest Neighbor (KNN) algorithm [11], which is a relatively simple and mature class of machine learning algorithms. The main use in PRM is the KNN for the current sampling point nearest neighbor sampling point finding, and by optimizing the number of connections between sampling points when building the undirected graph to reduce the time required for collision detection, so that the planning route can be completed faster. The following are some of the KNN algorithms that we use.
Algorithm 1: Partial KNN algorithm
Input: K-dimensional data
Output: n samples of the nearest neighbors of the current node
Steps:

1. calculate the distances between all sampled points and the current point.
2. sorting in increasing order of distance.
3. select the k points with the smallest distance from the current point.

K-dimensional Tree (KD-TREE) is an improvement under the original KNN, which mainly divides the K-dimensional data space and constructs an index tree for K-dimensional data, thus avoiding the complicated practice of the original KNN to calculate the distances of all other nodes in the domain from the current node when finding the nearest neighbor node. The following algorithm is used to construct the KD-TREE index.
Algorithm 2: KD-TREE index fast construction algorithm
Input: K-dimensional data
Output: Successfully constructed KD-TREE index
Steps:

1. Determine the spilt domain: Calculate the variance of all dimensional coordinates in the k-dimensional data, and select the dimension with larger variance as the spilt domain, the variance is to describe the degree of dispersion between a set of data, the larger the bivariate means that the nodes in this dimension cover a wider range.
2. Determine the root node: In the traditional KD-TREE, the coordinate values on the split domain of K-dimensional data are arranged in ascending order, and the middle data point is taken as the root node of the KD-TREE, while we take the current node as the root node of the KD-TREE, these will make the connection between the sampling points more reasonable.
3. Assign the data on the left side of the root node to the left subtree and the data on the right side to the right subtree, and then repeat steps (2) and (3) for the left and right subtrees until only one data point remains for each partitioned space.

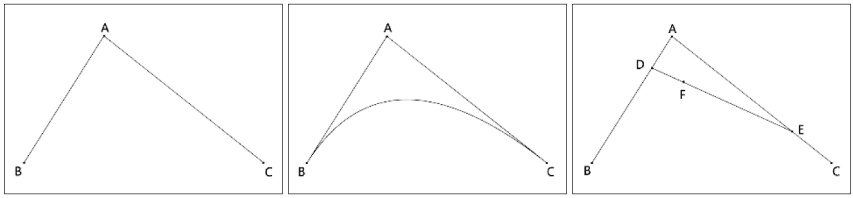
2.3 Bezier Curve

Bezier Curve is a mathematical curve applied to 2D graphics applications, and commonly used in path planning for path smoothing. Bezier Curve has some characteristics: 1. n control points are used to control the shape of the curve, here we use the sampling points in the planned completed path as control points. 2. The curve passes through the start and end points, close to but not through the middle point. In order to avoid collision

between the smoothed path and the obstacle, we will first expand the obstacle by 1/3 times, which is a more reasonable value for our many experiments. A quadratic Bezier curve is used in our improvement.

If a curve is smoothed between points A, B, and C, the exact procedure of the quadratic Bezier curve is as follows.

1. Find the points D and E on the lines AB and BC such that AD/DB = BE/EC, as shown in Fig. 2(a).
2. Connect DE and find the point F on DE such that it satisfies DF/FE = AD/DB = BE/EC, as shown in Fig. 2(b).
3. Find all the points that meet the above conditions, as shown in Fig. 2(c).



(a) (b) (c)

Fig. 2. The process of Bezier Curves.

We can obtain, according to the formula of Bezier curve of the first degree:

$$D = (1 - t)B + tA \tag{1}$$

$$E = (1 - t)C + tA \tag{2}$$

$$F = (1 - t)D + tE \tag{3}$$

Bringing the equations of (1), (2) into the equation of (3) yields the quadratic Bezier curve as:

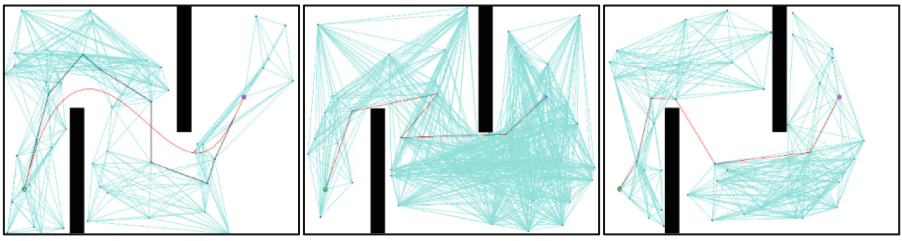
$$B(t) = (1 - t)^2 b + 2t(1 - t)A + t^2 C, t \in [0, 1] \tag{4}$$

3 Simulation Experiments

In order to evaluate the effectiveness and superiority of our improved scheme, we did sufficient experimental validation of our improved scheme and compared it with the original PRM and the PRM with basic KNN. The experimental results proved that our improved scheme has a significant improvement in planning time and the path is smoother and more reasonable.

3.1 Comparison of Simulation Experiments

Compared with the classical PRM, we use the KD-TREE in the process of building the undirected graph to find the nearby sampling points and connect them more quickly and reasonably, which can greatly reduce many invalid collision detections and improve the efficiency of our planner. On the other hand, we smooth the completed path to avoid inflection points in the path and allow the robot to move smoothly to the end point, while we use the method of expanding obstacles to avoid collision between the smoothed path and the obstacles. The following figure shows the experimental results of our improved PRM, the classical PRM and the PRM using the basic KNN algorithm (Fig. 3).



(a) (b) (c)

Fig. 3. (a) KDTREE-PRM (b) Classical-PRM (c) KNN-PRM

We can clearly see from the simulation result graphs of the three algorithms that the connectivity of the undirected map is significantly reduced after the introduction of KNN or KD-TREE, which greatly reduces the number of collision detection and improves the time efficiency of the algorithm. The black path in Fig. (a) is the path before smoothing, we can obviously see that the smoothed path is more reasonable, and the path before smoothing is generated in the expanded map, which will also avoid the collision between the smoothed path and the obstacle.

3.2 Simulation Experimental Data Analysis

We did 100 experiments for the three experimental schemes, where the number of sampling points is 50 and 100, and the number of proximity points in knn and kdtree is set to 20, and the specific planning time and path cost are compared in the following table (Table 1).

From the table, we can see that at 50 sampling points, the planning time of our scheme is improved by 19% compared to the classical PRM, 15% compared to the KNN-PRM, and there is also an improvement of about 10% in the path cost. When the sampling point is 100, the planning time of our scheme is improved by 47% compared to the classical PRM, and 12% compared to the KNN-PRM, and the path cost is not much different.

In summary, our improved PRM uses the KD-TREE algorithm to reduce the number of collision detections when constructing undirected maps and reduce the time cost. The completed paths are optimized by using the method of inflated maps and Bessel curves to make them smoother and more reasonable, achieving an effective optimization of the classical PRM.

Table 1. Algorithm simulation results.

Category	Number of sampling points	Path Costs	Planning time
Classical-PRM	50	221.48bits	0.57 s
	100	201.20bits	1.75
KNN-PRM	50	224.77bits	0.54 s
	100	204.07bits	1.04 s
KDTREE-PRM	50	201.38bits	0.46 s
	100	204.69bits	0.92 s

4 Conclusion

In this paper, based on the traditional PRM algorithm, we use KD-TREE to find the nearest neighbor part of the sampled points for linking instead of all sampled points when constructing the undirected graph, which greatly reduces the collision detection needed for this process and improves the efficiency of the planner, while using Bessel curves for the completed planned paths based on the inflated obstacles to make the paths smoother and more reasonable. The simulation experiments prove the feasibility of our improved algorithm. For the current trend of robotics, the research of PRM algorithm in high latitude space is our next direction.

References

1. Nannicini, G., Delling, D., Liberti, L., Schultes, D.: Bidirectional A* search for time-dependent fast paths. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 334–346. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68552-4_25
2. Ma'Arif, A., Rahmaniar, W., Vera, M.A.M., et al.: Artificial potential field algorithm for obstacle avoidance in UAV quadrotor for dynamic environment. In: 2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), pp. 184–189. IEEE (2021)
3. Zeng, N., Zhang, H., Chen, Y., et al.: Path planning for intelligent robot based on switching local evolutionary PSO algorithm. Assem. Autom. (2016)
4. Brand, M., Masuda, M., Wehner, N., et al.: Ant colony optimization algorithm for robot path planning. In: 2010 International Conference on Computer Design and Applications, vol. 3, pp. V3-436–V3-440. IEEE (2010)
5. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Queen's University Belfast (2010)
6. Wei, K., Ren, B.: A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. Sensors **18**(2), 571 (2018)
7. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
8. Mohanta, J.C., Keshari, A.: A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. Appl. Soft Comput. **79**, 391–409 (2019)

9. Hsu, D., Latombe: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robot. Res. (2006)
10. Ram, P., Sinha, K.: Revisiting kd-tree for nearest neighbor search. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1378–1388 (2019)
11. Farin, G.: Algorithms for rational Bézier curves. Comput. Aided Des. **15**(2), 73–77 (1983)