applied sciences

MDPI

Article

Collision-Free Path Planning Method for Robots Based on an Improved Rapidly-Exploring Random Tree Algorithm

Xinda Wang [1], Xiao Luo [2,*], Baoling Han [1], Yuhan Chen [1], Guanhao Liang [3] and Kailin Zheng [1]

[1] School of Mechanical Engineering, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China; 2120170401@bit.edu.cn (X.W.); hanbl@bit.edu.cn (B.H.); 3120185244@bit.edu.cn (Y.C.); 2120170440@bit.edu.cn (K.Z.)

[2] School of Computer Science and Technology, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China

[3] School of Mechatronical Engineering, Beijing Institute of Technology, No. 5 Zhongguancun South Street, Haidian District, Beijing 100081, China; simonleungbit@hotmail.com

* Correspondence: luox@bit.edu.cn; Tel.: +86-010-6891-8856

Received: 20 January 2020; Accepted: 13 February 2020; Published: 19 February 2020

Featured Application: The new method can be used to guide the path planning of robots with any number of degrees of freedom in complex environments.

Abstract: Sampling-based methods are popular in the motion planning of robots, especially in high-dimensional spaces. Among the many such methods, the Rapidly-exploring Random Tree (RRT) algorithm has been widely used in multi-degree-of-freedom manipulators and has yielded good results. However, existing RRT planners have low exploration efficiency and slow convergence speed and have been unable to meet the requirements of the intelligence level in the Industry 4.0 mode. To solve these problems, a general autonomous path planning algorithm of Node Control (NC-RRT) is proposed in this paper based on the architecture of the RRT algorithm. Firstly, a method of gradually changing the sampling area is proposed to guide exploration, thereby effectively improving the search speed. In addition, the node control mechanism is introduced to constrain the extended nodes of the tree and thus reduce the extension of invalid nodes and extract boundary nodes (or near-boundary nodes). By changing the value of the node control factor, the random tree is prevented from falling into a so-called "local trap" phenomenon, and boundary nodes are selected as extended nodes. The proposed algorithm is simulated in different environments. Results reveal that the algorithm greatly reduces the invalid exploration in the configuration space and significantly improves planning efficiency. In addition, because this method can efficiently use boundary nodes, it has a stronger applicability to narrow environments compared with existing RRT algorithms and can effectively improve the success rate of exploration.

Keywords: Rapidly-exploring Random Tree (RRT); manipulator; motion planning; obstacle avoidance; complex environment

1. Introduction

With the development of computer technology and modern manufacturing, particularly in the context of Industry 4.0, the intelligence level requirements continue to increase and the application scenarios of multi-Degree-Of-Freedom (DOF) robots are becoming increasingly complex. These conditions pose challenges for the motion planning of manipulators. Up to now, scholars have extensively researched motion planning in high-dimensional spaces and unstructured complex environments. Sampling-based

methods mainly seek a collision-free path from the start point to the goal point by sampling in the Configuration space (C-space). It is unnecessary to model the entire space, and the methods have a probabilistic completeness [1–3]. Therefore, sampling-based methods are widely used in the motion planning of high-dimensional spaces owing to their unique advantages. For example, the Rapidly-exploring Random Tree (RRT) [4] and Probabilistic Roadmap Method (PRM) [5,6] are the most popular and commonly used techniques.

Basic-RRT, as shown in Algorithm 1, has been widely used in many fields, including robot motion planning, as a single-query planner since it was proposed by LaValle et al. in 1998. A large number of algorithms derived from RRT have been proposed to solve different problems. Improvements in RRT have mainly focused on sampling strategies and their guidance for exploring areas, the selection of extension nodes, the directions and step sizes of extensions, selection of metrics, and the collision detection algorithm and local connection method, all of which have many studies available for reference [7]. These improved methods enhance the performance of Basic-RRT from various aspects, but the obtained solutions are still highly suboptimal in most cases. Therefore, various methods of pruning and smoothing [8–11] have been proposed to implement path post processing.

Algorithm 1 Basic-RRT algorithm.

1: $T \leftarrow$ InitTree(q_{start});
2: **for** $i = 1$ **to** n **do**
3: $q_{rand} \leftarrow$ RandomSample(i);
4: $q_{near} \leftarrow$ NearestNeighbor(q_{rand}, T);
5: $q_{new} \leftarrow$ Extend($q_{rand}, q_{near}, \varepsilon$);
6: **if** CollisionFree(q_{near}, q_{new}) **then**
7: AddNewNode(T, q_{new});
8: **end if**
9: **if** Distance(q_{new}, q_{goal}) $< \rho_{min}$ **then**
10: **return** T;
11: **end if**
12: **end for**
13: **return** *Failed*;

Algorithm improvement, regardless of the method, has only two purposes: to reduce the path cost and to reduce the running time of the algorithm. For the former, the typical method is RRT*, which is an optimal planning method based on RRT and proposed by Karaman and Frazzoli [12]. It guarantees the asymptotic optimality of the algorithm by adding two processes to Basic-RRT: "near vertices" and "rewire". However, the planning process is time consuming and thus impractical for planning tasks that have certain time requirements. Many methods [13–15] have been subsequently proposed to accelerate the convergence of RRT*, but the computational time still cannot meet the requirements of the actual application.

This paper focuses on the latter purpose of algorithm improvement, namely to decrease the computational time of the algorithm on the premise that the algorithm can be applied to complex environments. The usual approach involves reducing the number of nodes in the tree and the number of collision detections. RRT-biased [16] is a simple way to improve efficiency. As shown in Algorithm 2, it improves the performance of the algorithm through goal bias sampling with a certain probability (usually 5–10%). However, although the number of nodes in the random tree is appropriately reduced, the computational cost remains large. Gitae Kang et al. [17] proposed a method based on goal-oriented sampling for the motion planning of a manipulator; this method can limit the distribution of sampling points to improve the search speed, but the invalid exploration area is large. Brendan Burns et al. [18] proposed a utility function for selecting the extension node and

direction; this function, which determines the maximum expected extension step of the planner on the basis of the obtained state space information, has improved efficiency to some extent. In addition, Dong-Hyung Kim et al. [19] described a method of adaptive body selection based on the complexity of planning; this method provides another description of the variable-dimensional C-space for high-DOF articulated robots, thereby improving efficiency from the perspective of dimensionality reduction.

Algorithm 2 GoalBiasedSample()

1: $num \leftarrow$ RandomNumber; RandomNumber $\in [0, 1]$
2: **if** $num <$ k **then**
3: $q_{rand} = q_{goal}$;
4: **else**
5: $q_{rand} =$ RandomSample();
6: **end if**
7: **return** q_{rand};

These methods and many existing RRT variants that guide sampling and expansion have improved efficiency from different aspects, but there is no good general solution for complex environments, such as "narrow" ones. Haojian Zhang et al. [20] proposed an improved method combining the regression and boundary expansion mechanisms; this technique improves efficiency by providing a corresponding solution process for a specific problem. However, although the boundary node can be identified during the exploration process, this method also limits the use of boundary nodes. In addition, some methods [21–23] dedicated to improving the success rate of the algorithm by changing the sampling strategy have been proposed specifically to solve the problem of narrow environments, but their universality is also weak.

It can be seen from the above that an efficient and universal algorithm that is suitable for complex environments is needed to ensure the completion of planning tasks. Therefore, this paper proposes an autonomous path planning algorithm of Node Control based on the architecture of RRT (NC-RRT). Firstly, a method of gradually changing the sampling area is proposed to guide exploration, thereby effectively improving the search speed. In addition, unlike existing methods, which aim to explore new sampling strategies, a node control mechanism is proposed to constrain the extended nodes of the tree and, thus, enhance the applicability of the environment. The results reveal that the algorithm greatly reduces the invalid exploration in the C-space and significantly improves planning efficiency. Moreover, compared with most existing RRT algorithms, NC-RRT is universal for different environments by appropriately adjusting the parameters. For the convenience of explanation, the algorithm will be first described and verified in a two-dimensional space and then applied in a high-dimensional space.

The rest of the paper is organized in the following manner. Section 2 introduces the proposed improved RRT algorithm in two parts. The method of gradually changing the sampling area is proposed in the first part, and we present the node control mechanism in the second part. Section 3 explains the simulation process and results of the algorithm. The proposed algorithm is simulated in a two-dimensional space and then applied to a 6-DOF manipulator. Finally, the conclusions and future work of this paper are provided in Section 4.

2. The NC-RRT Method

2.1. The Method of Gradually Changing the Sampling Area

We propose the method of gradually Changing the Sampling Area based on RRT (CSA-RRT) to guide exploration; the process is shown in Algorithm 3. As in [17], we need to calculate the distance (the Euclidean distance is used in this paper) between the goal configuration point q_{goal} and the configuration point that is initially the farthest from the goal q_f in the C-space, which is represented by

$D_{farthest}$ here, to ensure the completeness of the solution space. Assuming that the dimension of the C-space is s, $D_{farthest}$ is solved as follows in Equation (1):

$$D_{farthest} = \sqrt{(q_{goal(1)} - q_{f(1)})^2 + ... + (q_{goal(s)} - q_{f(s)})^2}. \quad (1)$$

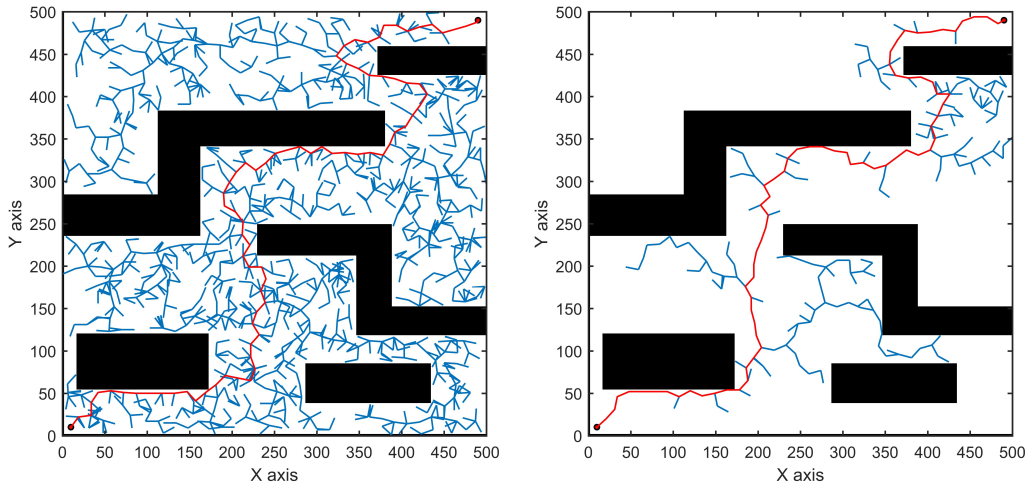
Algorithm 3 CSA-RRT algorithm.

$T \leftarrow$ InitTree(q_{start});
$R \leftarrow D_{farthest}$;
for $i = 1$ **to** n **do**
  $q_{rand} \leftarrow$ RandomSample(i);
  if Distance(q_{rand}, q_{goal}) $> R$ **then**
    continue;
  end if
  $q_{near} \leftarrow$ NearestNeighbor(q_{rand}, T);
  $q_{new} \leftarrow$ Extend($q_{rand}, q_{near}, \varepsilon$);
  if CollisionFree(q_{near}, q_{new}) **then**
    AddNewNode(T, q_{new});
    $R =$ Distance(q_{new}, q_{goal});
  else
    $R = R + k \times \varepsilon$;
    continue;
  end if
  if Distance(q_{new}, q_{goal}) $< \rho_{min}$ **then**
    return T;
  end if
end for
return *Failed*;

We initialize the sampling radius R to $D_{farthest}$, and the sampling range in the subsequent sampling process is limited to the area within R of the goal configuration point. If a new node q_{new} is successfully added in a certain iteration, that is there is no collision with obstacles during the generation of q_{new}, then the value of R is changed to the distance between q_{new} and q_{goal}. On the contrary, if there is a collision, then the sampling radius is increased as follows:

$$R = R + k \times \varepsilon, \quad (2)$$

where k is the coefficient used to change the range of the sampling domain; its value is a positive integer and can be adjusted according to the complexity of the environment. ε is the step size of the extension. We can find that, if there is no obstacle in the environment or there are obstacles, but no collision, the sampling domain will be reduced to an area $\| q_{new} - q_{goal} \|$ away from the goal node each time a new node is added to the tree; once obstacles are encountered during expansion, the sampling domain will be expanded to find a new sampling point and the corresponding q_{near} node. When q_{new} is successfully added, the random tree will be restored to the no obstacle state and continue to explore. This gradual change in the sampling area will drive the tree to grow continuously toward the goal node. Figure 1 shows the performance comparison of Basic-RRT and CSA-RRT in the same environment. Obviously, the latter has much fewer nodes. The specific experiment and analysis are provided in Section 3.



(**a**) (**b**)

Figure 1. Performance comparison between Basic-RRT and CSA-RRT in the same environment. (**a**) Performance of Basic-RRT. (**b**) Performance of CSA-RRT.

2.2. The Node Control Mechanism

When the CSA-RRT algorithm selects the extended nodes, the nodes in the tree should be traversed to find the node with the smallest distance from the sampling configuration point in the C-space, but this traversal is usually redundant and time consuming. Under the premise of maintaining the performance of the CSA method, a node control mechanism is introduced in this part to further reduce the extension of invalid nodes and extract the boundary nodes (or near-boundary nodes), thereby further improving the speed of the algorithm and enhancing the adaptability of the environment. On the basis of this mechanism, the "local trap" phenomenon in the process of random tree expansion is proposed. We update and record the expansion state of each node when the random tree expands in the C-space. The value of the node control factor is changed according to whether the "local trap" phenomenon occurs. Then, the selection of the extension node is adjusted, such that the tree is expanded by the boundary nodes only or by nodes close to the boundary in most cases. The specific process is as follows.

We represent the extended state value of the node as δ and redefine the following concept that is different from the past: the path that each leaf node of the tree traces back to the initial node q_{start} is called a branch. Then, the change strategy of the node state value is as follows. Its δ value is set to zero each time a new node is successfully added to the tree. Then, starting from the parent node and following the reverse path, one is added to the state value of each node in the branch where this new node is located until it traces back to the initial node. As shown in Figure 2, any node q_n in the process of random tree exploration is regarded as an example. Assuming its δ value is recorded as m at this time, if a child node q_{n+1} is generated with q_n as the parent node in a certain iteration, then the value of δ for q_{n+1} is set to zero; then, starting from the parent node (i.e., q_n) of q_{n+1} and following the reverse path, one is added to the value of δ for all nodes in the branch where q_{n+1} is located until it traces back to the q_{start} node. At this time, the δ value of q_n becomes $m+1$. If the new node q_{n+2} generated in a subsequent iteration is still a child node of q_n, the above process is repeated, and the δ value of q_n becomes $m+2$. However, because the q_{n+1} node is not in the same branch as the q_{n+2} node, the δ value of the q_{n+1} node remains unchanged in this iteration.

According to the state value of each node, a node control factor *control* can be introduced to guide the selection of the expanded node. For each extension, only the node that has a state value less than *control* and lies closest to the sampling node is selected as the extension node. To reduce the invalid exploration in the C-space sufficiently, we set *control* to one by default. However, the introduction of this control factor creates a problem. Because only the δ value of the leaf nodes in the random tree is zero and the value of *control* remains one, it means that the last node in the tree will always be used as

the extension node. Combined with the proposed CSA sampling method, this technique will drive the tree to reach the target configuration extremely rapidly if there are no obstacles in the environment. As shown in Figure 3a, we can see that all nodes in the tree exist as "valid nodes" in the final query path. However, once there are obstacles in the environment, the tree easily falls into the "local trap" state, as shown in Figure 3b. The random tree will keep this state, which can be expanded in local areas only until it reaches the set maximum number of failures.

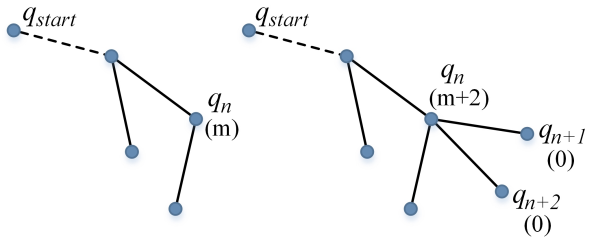
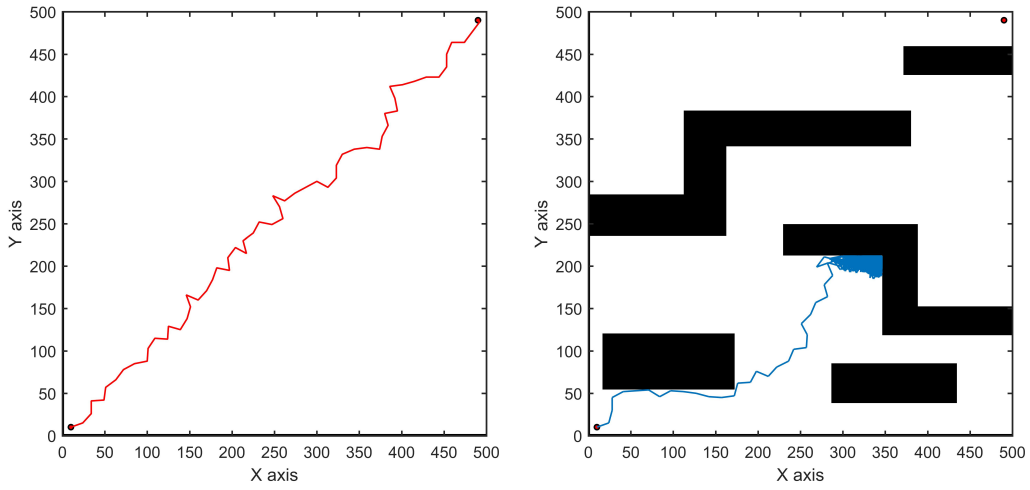


Figure 2. The change strategy of node state value δ in the node control mechanism. q_{start} is the initial node, and q_n is any node in the tree. q_{n+1} and q_{n+2} are child nodes of q_n. The values in parentheses are the node state values δ we recorded.



(**a**) (**b**)

Figure 3. Under a *control* value of one and the combination of the exploration with the CSA method, the random tree exhibits different behaviors when it does and does not encounter obstacles. (**a**) Efficient exploration in an environment without obstacles. (**b**) The "local trap" phenomenon during exploration in an environment full of obstacles.

Therefore, we need to change the value of *control* at the appropriate time to increase the number of optional extension nodes to prevent this problem. This phenomenon is likely to occur when the tree encounters obstacles. Thus, we consider the collision occurrence in the expansion process approximately as the judgment condition for the occurrence of the "local trap" phenomenon, that is the condition for changing the value of the node control factor, similar to the change condition of the sampling radius in the CSA-RRT method. When collision occurs, the *control* value is increased, and the tree can effectively escape from this area. After the new node is successfully added, the *control* value is restored to one to continue the exploration. At this point, the NC-RRT algorithm proposed in this paper is obtained, and its pseudocode is shown in Algorithm 4.

Algorithm 4 NC-RRT algorithm.

$T \leftarrow$ InitTree(q_{start});
$\Delta_T \leftarrow \{\delta(q_{start}) = 0\}$;
$R \leftarrow D_{farthest}$;
$control \leftarrow 1$;
for $i = 1$ **to** n **do**
  $q_{rand} \leftarrow$ RandomSample(i);
  if Distance(q_{rand}, q_{goal}) $> R$ **then**
    continue;
  end if
  $T_{ctrl} \leftarrow$ LessThanControl($\Delta_T, control, T$);
  $q_{near} \leftarrow$ NearestNeighbor(q_{rand}, T_{ctrl});
  $q_{new} \leftarrow$ Extend($q_{rand}, q_{near}, \varepsilon$);
  if CollisionFree(q_{near}, q_{new}) **then**
    AddNewNode(T, q_{new});
    $\Delta_T \leftarrow \{\delta(q_{new}) = 0\}$;
    $\Delta_T \leftarrow$ BranchNodeUpdate(Δ_T, q_{new}, T);
    $R =$ Distance(q_{new}, q_{goal});
    $control = 1$;
  else
    $R = R + k \times \varepsilon$;
    $control =$ c; (c $= 2, 3, ...$)
    continue;
  end if
  if Distance(q_{new}, q_{goal}) $< \rho_{min}$ **then**
    return T;
  end if
end for
return *Failed*;

3. Simulation and Analysis

Computation in a two-dimensional space is small and yields an intuitive effect. Therefore, several algorithms were firstly simulated in a two-dimensional environment in this section to evaluate the performance of the proposed algorithm. Basic-RRT, CSA-RRT, and NC-RRT were analyzed and evaluated by comparing their simulation results in various environments. Subsequently, each algorithm was applied to a 6-DOF serial manipulator and simulated in an environment full of obstacles. Simulations of all algorithms were performed using MATLAB 2015b on a Windows 10 system with an Intel Core i7-8750H 2.2 GHz CPU, and 8 GB of RAM. In addition, a virtual prototype experiment of the 6-DOF manipulator was completed by using ADAMS. The simulation results in the two-dimensional space and the virtual prototype experiments of the 6-DOF manipulator were the average values of 50 runs.

3.1. Simulations in a Two-Dimensional Environment

Each algorithm was simulated in three different environments in a two-dimensional space. As shown in Figure 4, three typical maps were used: a map cluttered with obstacles, a trapped environment, and a narrow passage. The environment dimensions were 500×500 in all cases. The black areas in the map indicate obstacles, and the start and goal points were set separately in each map. The step size ε was set to 15, and the maximum number of failures nwas set to 2000. In addition, the k and c values in all the algorithms could be adjusted appropriately according to the different environments.

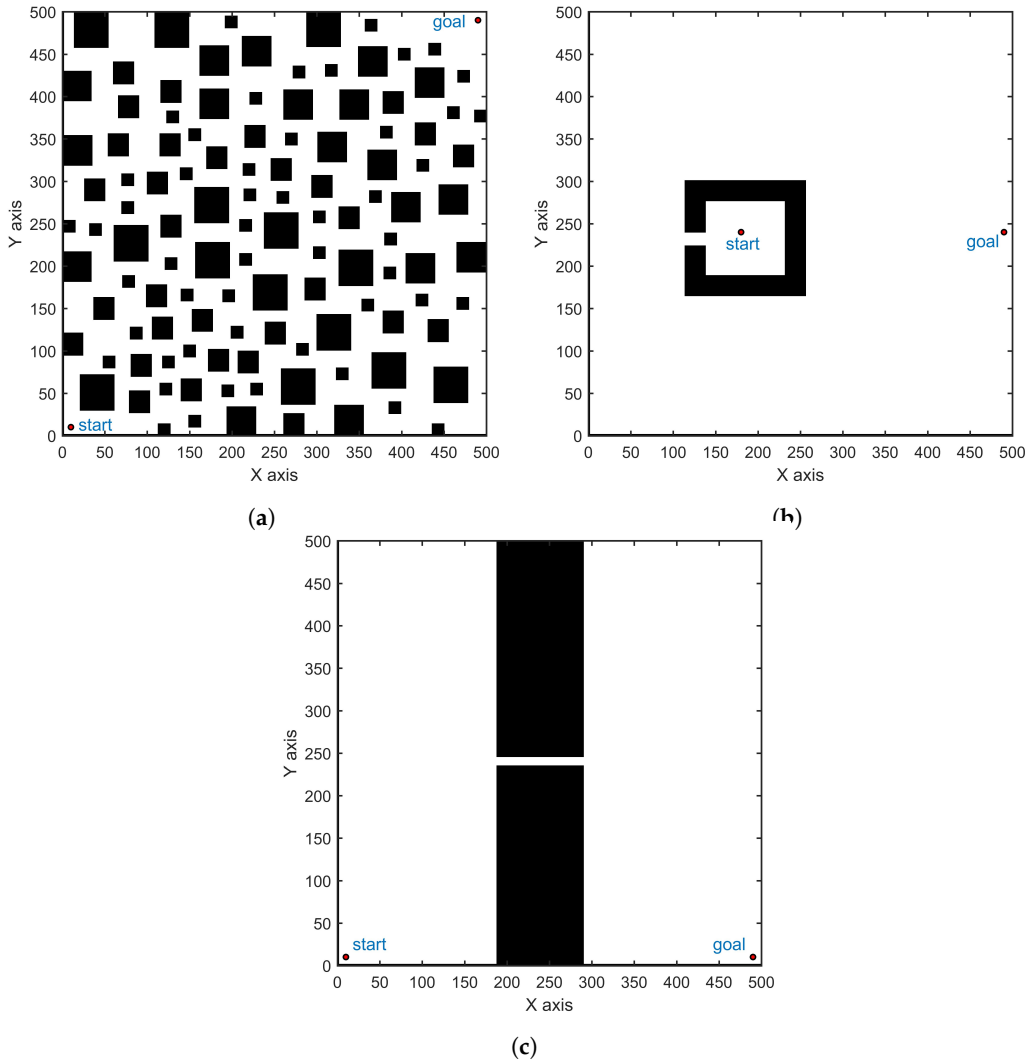
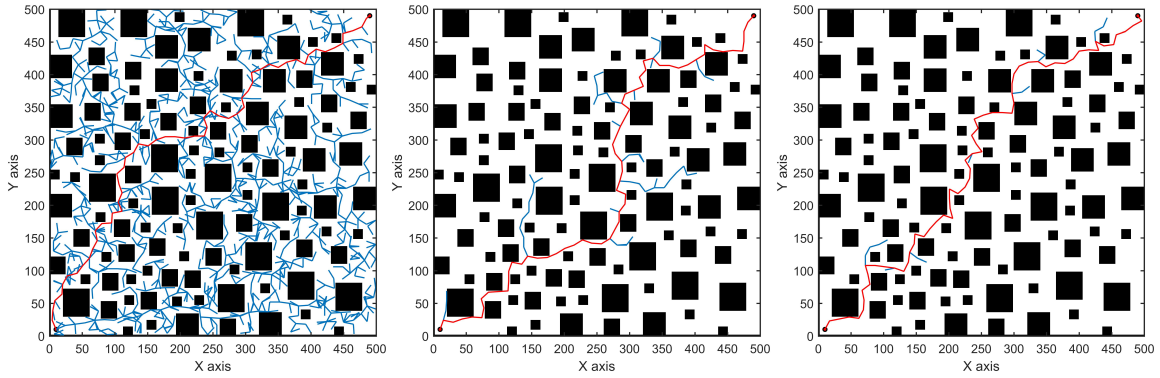


Figure 4. Three typical environments in a two-dimensional space. (**a**) The cluttered environment. (**b**) The trapped environment. (**c**) The narrow environment.

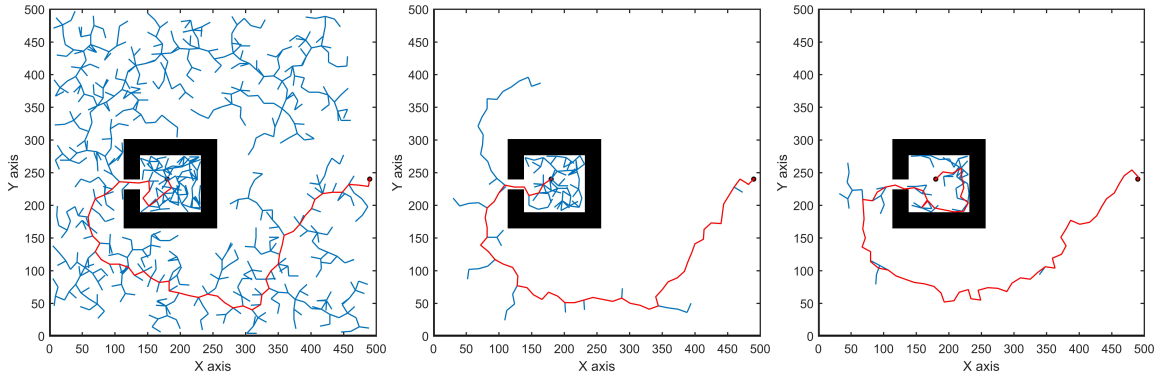
Figures 5–7 present the performance of the three algorithms in the cluttered, trapped, and narrow environments, respectively. The entire exploration process is denoted by blue lines, and the resulting query path is denoted by red lines. When the Basic-RRT algorithm was simulated, the nodes of the random tree almost filled the entire free space in each map. By contrast, the number of nodes was greatly reduced after the proposed CSA-RRT and NC-RRT algorithms were used. In addition, as seen in Figures 6c and 7c, the extended nodes in the tree were distributed more around the obstacles because of the node control mechanism in the NC-RRT algorithm.

Tables 1–3 present the simulation results of the three algorithms with respect to the average computational time, average number of nodes in the tree, average number of collision detections, average path length, and success rate in each environment. The results revealed that the proposed CSA-RRT and NC-RRT had good effects in each environment. Compared with that of Basic-RRT, the running times of the proposed algorithm were greatly reduced, and the numbers of nodes in the random tree and the numbers of collision detections were decreased.



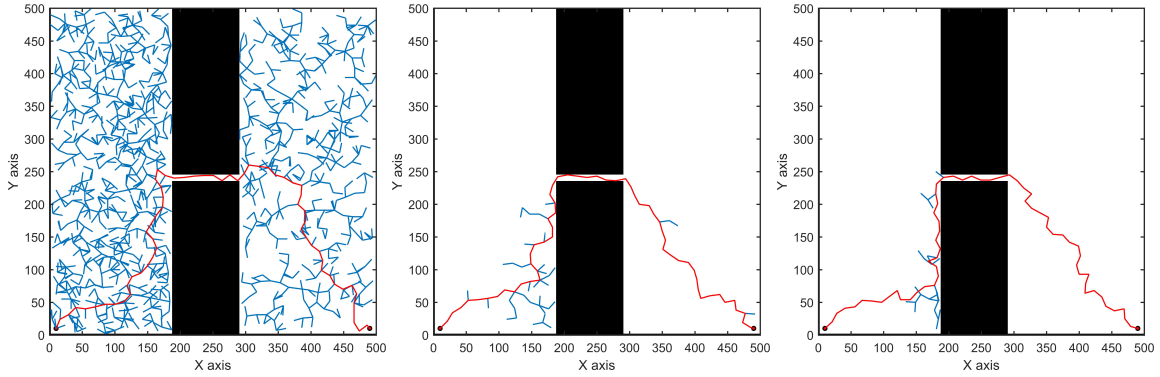
(**a**) Basic-RRT (**b**) CSA-RRT (**c**) NC-RRT

Figure 5. Performance of the three algorithms in the cluttered environment. $(k = 1,\ c = 2)$.



(**a**) Basic-RRT (**b**) CSA-RRT (**c**) NC-RRT

Figure 6. Performance of the three algorithms in the trapped environment. $(k = 3,\ c = 2)$.



(**a**) Basic-RRT (**b**) CSA-RRT (**c**) NC-RRT

Figure 7. Performance of the three algorithms in the narrow environment. $(k = 1,\ c = 2)$.

However, as can be seen from Tables 1–3, the average computational time of CSA-RRT and NC-RRT in the cluttered environment was 0.058 and 0.055 s, respectively; the average computational time in the trapped environment was 0.118 and 0.159 s, respectively; and the average computational time in the narrow environment was 0.106 and 0.117 s, respectively. Thus, the computing efficiency of the two algorithms in each environment was nearly the same. Moreover, the CSA method could reduce the path length to some extent, but the path length increased slightly after the node control mechanism was added. The reason was that the random tree in the NC-RRT method usually expanded along the boundary of obstacles. However, it was irrelevant for our purposes because the node control mechanism proposed in this paper was concerned more about how to improve the environmental adaptability of the algorithm by using boundary nodes than about the path cost. Furthermore, all

the above-mentioned algorithms would eventually need to complete path pruning and smoothing when used in practice, so this cost could be ignored. In addition, we could see that the success rates of NC-RRT in the cluttered and trapped environments were not significantly different compared with those of the other two algorithms. However, in the narrow environment, the success rate was effectively increased, and the occurrence of pathological cases was reduced due to the excellent boundary property of the NC-RRT algorithm. Therefore, the proposed node control mechanism was necessary. In other words, the NC-RRT algorithm, which combined the CSA method and the node control mechanism, could effectively improve the efficiency of planning and was more suitable for complex environments than other algorithms.

Table 1. Simulation results of the three algorithms in the cluttered environment. ($k = 1,\ c = 2$).

Algorithm	Average Computational Time (s)	Average Number of Nodes in the Tree	Average Number of Collision Detections	Average Path Length	The Success Rate of the Algorithm
Basic-RRT	0.463	844.520	38,026	954.369	96%
CSA-RRT	0.058	92.900	4860.1	915.982	100%
NC-RRT	0.055	82.860	3941.7	977.667	100%

Table 2. Simulation results of the three algorithms in the trapped environment. ($k = 3,\ c = 2$).

Algorithm	Average Computational Time (s)	Average Number of Nodes in the Tree	Average Number of Collision Detections	Average Path Length	The Success Rate of the Algorithm
Basic-RRT	0.471	928.680	35,821	865.236	84%
CSA-RRT	0.118	122.909	9705.3	805.751	88%
NC-RRT	0.159	126.817	9214.6	992.595	88%

Table 3. Simulation results of the three algorithms in the narrow environment. ($k = 1,\ c = 2$).

Algorithm	Average Computational Time (s)	Average Number of Nodes in the Tree	Average Number of Collision Detections	Average Path Length	The Success Rate of the Algorithm
Basic-RRT	0.550	1145.600	40,159	871.825	78%
CSA-RRT	0.106	172.732	8867.2	868.609	82%
NC-RRT	0.117	126.245	6746.4	942.615	98%

3.2. Simulation of the 6-DOF Manipulator

In this section, the above algorithms are applied to a 6-DOF serial manipulator. As presented in Figure 8, the manipulator was surrounded by obstacles (blue solid blocks) mainly distributed in the upper and lower parts of the manipulator workspace. These two parts restricted the path of the manipulator through a tunnel. Each algorithm was implemented and applied to the manipulator to complete the task of passing through the tunnel from the initial configuration to the target configuration. Because the success rate of using uniform sampling for planning in high-dimensional spaces was almost zero, all the algorithms in this experiment adopted the sampling strategy with a 10% goal bias to maintain the consistency of the conditions. The step size ε was set to 2°, and the maximum number of failures n was set to 2000. $k = 15,\ c = 2$ were set in the NC-RRT and CSA-RRT algorithms, and the average time obtained after 50 runs of each algorithm is presented in Table 4. It could be seen that compared with RRT with the 10% goal bias, the algorithm proposed in this paper significantly enhanced the efficiency.

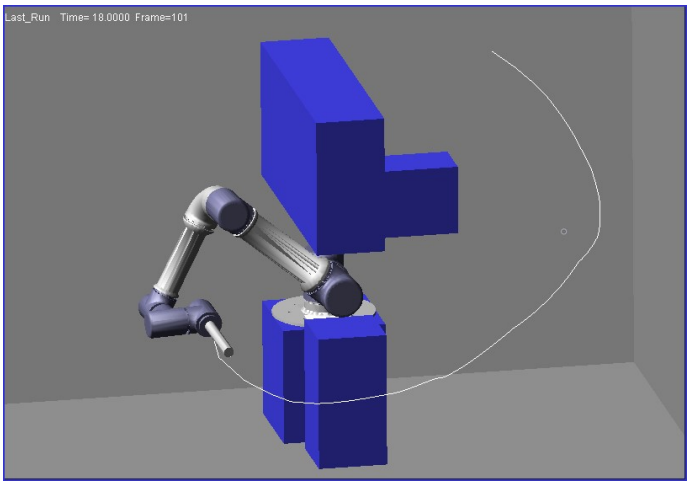


Figure 8. Simulation of a 6-DOF manipulator. The three algorithms are separately applied to the manipulator to allow it to pass through a tunnel and reach the target configuration.

Table 4. Average computational time for each algorithm applied to the 6-DOF manipulator (10% goal bias, 100% success rate).

Algorithm	Basic-RRT	CSA-RRT	NC-RRT
Average computational time (s)	3.658	1.541	1.469

4. Conclusions and Future Work

To address the problem of existing sampling-based planners having low exploration efficiency and poor environmental adaptability and the increasingly sophisticated level of intelligence requirements in the Industry 4.0 era not being met, this study proposed a path planning algorithm that was based on the architecture of the RRT algorithm and suited complex environments. The algorithm included a method of gradually changing the sampling area and a node control mechanism, which were used to guide the random tree exploration and reduce the expansion of invalid nodes, respectively. Furthermore, the node control mechanism could extract boundary nodes to improve the environmental adaptability. The algorithm was tested in three scenarios in two-dimensional space and was applied to a 6-DOF manipulator. The results revealed that the algorithm was effective and universal. It could significantly improve the planning efficiency and had a stronger applicability to complex environments, particularly narrow environments, compared with the traditional RRT algorithm.

However, the proposed method had certain limitations. The selection of the parameters k and c in the algorithm was a problem. Although the algorithm was universal, these parameters sometimes needed to be adjusted appropriately for different environments in order to obtain the best results. In addition, the path quality needed to be improved. This issue will be further researched by attempting an adaptive adjustment of parameters and considering the introduction of curvature constraints and other kinematic and dynamic constraints to improve the performance of the algorithm.

Author Contributions: Conceptualization, X.W. and X.L.; methodology, X.W.; software, Y.C.; validation, G.L. and K.Z.; formal analysis, X.L.; investigation, G.L. and K.Z.; data curation, X.L.; writing, original draft preparation, X.W.; writing, review and editing, X.W.; visualization, Y.C.; supervision, B.H.; project administration, B.H. All authors read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R & D Program of China (2016YFC0803000, 2016YFC0803005).

Acknowledgments: Thanks to Professor Qingsheng Luo for providing some suggestions to improve this manuscript. Additionally, thanks to Shanda Wang, Yan Jia and Lei Wang for the language help.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

C-space	Configuration space
RRT	Rapidly-exploring Random Tree
PRM	Probabilistic Roadmap Method
CSA-RRT	Gradually Changing the Sampling Area-RRT
NC-RRT	Node Control-RRT
DOF	Degree-Of-Freedom

References

1. Barraquand, J.; Kavraki, L.; Latombe, J.C.; Motwani, R.; Li, T.Y.; Raghavan, P. A random sampling scheme for path planning. *Int. J. Robot. Res.* **1997**, *16*, 759–774. [CrossRef]
2. Hsu, D.; Latombe, J.C.; Kurniawati, H. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robot. Res.* **2006**, *25*, 627–643. [CrossRef]
3. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
4. LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. *Comput. Sci. Dept. Oct.* **1998**, *98*. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1853 (accessed on 19 February 2020).
5. Amato, N.M.; Wu, Y. A randomized roadmap method for path and manipulation planning. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 22–28 April 1996; Volume 1, pp. 113–120.
6. 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.* **1996**, *12*, 566–580. [CrossRef]
7. Elbanhawi, M.; Simic, M. Sampling-based robot motion planning: A review. *IEEE Access* **2014**, *2*, 56–77. [CrossRef]
8. Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; et al. Stanley: The robot that won the DARPA Grand Challenge. *J. Field Robot.* **2006**, *23*, 661–692. [CrossRef]
9. Urmson, C.; Anhalt, J.; Bagnell, D.; Baker, C.; Bittner, R.; Clark, M.; Dolan, J.; Duggins, D.; Galatali, T.; Geyer, C.; et al. Autonomous driving in urban environments: Boss and the urban challenge. In *The DARPA Urban Challenge*; Springer: Berlin, Germany, 2009; pp. 1–59.
10. Yang, K.; Sukkarieh, S. An analytical continuous-curvature path-smoothing algorithm. *IEEE Trans. Robot.* **2010**, *26*, 561–568. [CrossRef]
11. Wei, K.; Ren, B. A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors* **2018**, *18*, 571. [CrossRef] [PubMed]
12. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
13. Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M.; Muhammad, M.S. RRT*-SMART: A rapid convergence implementation of RRT. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 299. [CrossRef]
14. Qureshi, A.H.; Iqbal, K.F.; Qamar, S.M.; Islam, F.; Ayaz, Y.; Muhammad, N. Potential guided directional-RRT* for accelerated motion planning in cluttered environments. In Proceedings of the 2013 IEEE International Conference on Mechatronics and Automation, Takamatsu, Japan, 4–7 August 2013; pp. 519–524.
15. Weghe, M.V.; Ferguson, D.; Srinivasa, S.S. Randomized path planning for redundant manipulators without inverse kinematics. In Proceedings of the 2007 7th IEEE-RAS International Conference on Humanoid Robots, Pittsburgh, PA, USA, 29 November–1 December 2007; pp. 477–482.
16. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA, Millennium Conference, IEEE International Conference on Robotics and Automation, Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.

17. Kang, G.; Kim, Y.B.; Lee, Y.H.; Oh, H.S.; You, W.S.; Choi, H.R. Sampling-based motion planning of manipulator with goal-oriented sampling. *Intell. Serv. Robot.* **2019**, *12*, 265–273. [CrossRef]
18. Burns, B.; Brock, O. Single-query motion planning with utility-guided random trees. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3307–3312.
19. Kim, D.H.; Choi, Y.S.; Kim, S.H.; Wu, J.; Yuan, C.; Luo, L.P.; Lee, J.Y.; Han, C.S. Adaptive rapidly-exploring random tree for efficient path planning of high-degree-of-freedom articulated robots. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2015**, *229*, 3361–3367. [CrossRef]
20. Zhang, H.; Wang, Y.; Zheng, J.; Yu, J. Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments. *IEEE Access* **2018**, *6*, 53296–53306. [CrossRef]
21. Boor, V.; Overmars, M.H.; Van Der Stappen, A.F. The Gaussian sampling strategy for probabilistic roadmap planners. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Detroit, MI, USA, 10–15 May 1999; pp. 1018–1023.
22. Sun, Z.; Hsu, D.; Jiang, T.; Kurniawati, H.; Reif, J.H. Narrow passage sampling for probabilistic roadmap planning. *IEEE Trans. Robot.* **2005**, *21*, 1105–1115.
23. Zhong, C.; Liu, H. A region-specific hybrid sampling method for optimal path planning. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 71. [CrossRef]