Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Path Planning of Redundant Series Manipulators Based on Improved RRT Algorithms

*Zhaoxing Chen, Yang Yang and Xiangrong Xu**
School of Mechanical Engineering, Anhui University of Technology
Maanshan, Anhui Province, China, 243002
xuxr@ahut.edu.cn

Aleksandar Rodic
Robotics Laboratory, Institute Mihailo Pupin
Volgina 15, Belgrade, Serbia,11060
roda@robot.imp.bg.ac.rs

***Abstract* - Aiming at the path planning problem of the redundant manipulator with seven degrees of freedom, this paper presents a new algorithm based on improved RRT (Rapidly-exploring Random Tree algorithm), Gm-RRT algorithm. The new algorithm performs path planning in the joint space of the robot to improve the efficiency of the planning. The new algorithm combines the target probability offset to improve the orientation of the growth of extended random trees; uses the method of generating random points instead of single random points to improve the computational efficiency of path planning; and introduces the method of adaptive virtual gravity of targets to ensure the balance of the new algorithm in target orientation and exploration to the surrounding space. Lastly, a simulation experiment is carried out on the MATLAB platform. The results show that the improved algorithm has advantages in time and keeps the algorithm concise.**

***Keywords*: redundant manipulator, path planning, Gm-RRT algorithm, adaptive virtual gravitational.**

I. Introduction

The robot is the key equipment of the advanced manufacturing industry, and it is also the important equipment that integrates into daily life in an intelligent age. In the new era, the industrial powers have proposed that technical innovation should lead to the upgrading of the manufacturing industry. The intelligent and individualized production process has become a trend in the manufacturing industry. So, the seven-degree-of-freedom manipulator is widely used because of its unique redundant motion characteristics and flexibility similar to the human arm. In order to develop the working potential of redundant robots, engineers have studied the kinematics and dynamics characteristics of redundant manipulators, and the path planning of the manipulators is a key direction. There are two directions for path planning, joint space planning, and mission space planning. In this paper, path planning in joint space is adopted to avoid using a large number of inverse operations to affect the speed of path planning in the workspace.

At present, the main path planning methods are artificial potential field method [1], map building method [2] and random sampling method. The artificial potential field method is carried out under the action of the gravity of the target and the repulsion of the obstacle. This method is easy to expand and can realize real-time obstacle avoidance planning. But the biggest disadvantage of this method is that the robot may fall into the local minimum, and the computational load will become large as the number of obstacles increases. The map-based method divides the workspace of the robot into different grid spaces and then finds the feasible path according to certain rules in the divided grid space, which is mainly divided into the landmark method [3-5] and grid method [6]. There are two main methods based on random sampling: Probability Road Map (PRM) [7] and Rapidly-exploring Random Tree (RRT) [8]. The main difference between the two methods is that the fast-expanding random tree method is a simple and effective path search method, which uses the number of random expanding trees to construct feasible paths in the free state space and does not have a ring structure in the process of path generation.

RRT algorithm, the Rapidly-exploring Random Tree algorithm, was proposed by Steven M. Lavalle in 1998. It is an incremental motion planning algorithm based on random sampling and a typical tree structure search algorithm. Taking an initial point in free space as the root node, the random sampling method is used to extend the tree and generate a collision-free random expansion tree. When the extended tree grows in the free space, it gradually covers the nodes near the endpoint and then generates the path from the initial path to the target state. RRT algorithm has the advantages of simple calculation and is suitable for multi-dimensional space. This method can be applied to the path planning of manipulators because it can ensure that the path planning can be realized in two-dimensional or even high-dimensional [9]. The trajectory planning of common manipulators is carried out in the workspace, and the inverse calculation is needed at each path point. When the manipulator is a redundant manipulator, its inverse calculation process will be very complex and consume a lot of time. The RRT algorithm and the improved algorithm can be used to plan the path of the manipulator in the joint space of the manipulator, which avoids a large number of inverse operations and improves the work efficiency.

II. The Basic RRT Algorithm

RRT algorithm, as a spatial search algorithm based on random sampling, is simple and efficient. On the base of determining the initial state and the end state, the random tree expands continuously in the free space through random sampling, and finally finishes the search in the free space when the goal state is obtained by the tree. So, there will be a safe and feasible path from the initial state to the target state. RRT algorithm is simple to implement, does not need to accurately model obstacles, and works well in high-dimensional space. This algorithm is widely used in the path planning of mobile robots and multi-degree-of-freedom serial manipulators. Its pseudocode is displayed in algorithm 1

In the algorithm, q_S is the starting point, q_G is the ending point, λ is the growth step, T is the random tree.

```
Algorithm 1: Generate RRT
Input: $q_S, q_G, \lambda$
Output: $T$
 1. $T.init = q_S$
 2. for $k$ = 1:K do
 3.    $qrand \leftarrow GetRandomNode()$
 4.    $q\_closest \leftarrow NearNeighbor(T, q_{rand})$
 5.    $qnew \leftarrow SteerNode(q_{rand}, q_{closed}, \lambda)$
 6.    if $CollisionFree()$ then
 7.      $T.add\_vertex(q_{new})$
 8.      $T.add\_vertex(q_{new})$
 9.    end if
10.    if $GettheGoal$ then
11.      return T
12.    end if
13. end for
```

Where K is the set maximum number of cycles. Several main functions of the above algorithm are important parts of the RRT algorithm, and they are still used in the improved algorithm introduced later. These functions are described below:

1. $GetRandomNode()$, Sampling: A sampling node, q_{rand}, is randomly selected in the workspace or configuration space. Sampling points are independent.
2. $NearNeighbor()$, Get the closest node: For a known search random tree, the nearest node $q_{closest}$ to the function input node q in the random tree is obtained. Using the following formula:
$$NearNeighbor(T,q) = \arg\min_{t \in T} \|t - q\| \qquad (1)$$
3. $SteerNode()$, Expanding: A new node is generated by extending step length lambda from one configuration space node q_1 to another node q_2. The function is:
$$Steernode(q_1, q_2) = q_1 + \lambda \frac{(q_2 - q_1)}{\|q_2 - q_1\|} \qquad (2)$$
4. $CollisionFree()$, collision detection: Collision detection is carried out on the paths generated between the nodes in the two configuration spaces to determine that the paths between the two nodes are safe. If no collision occurs, a path can be generated between two nodes and the feasible logic can be returned.
5. $GettheGoal$, Arrival judgment: When the new node of the random tree grows near the target node, it will return the logic true. Otherwise, the cyclic operation will continue. The decision formula is:
$$\|qnew - qgoal\| < \varepsilon \qquad (3)$$

III. The Improved Path Planning Algorithm, Gm-RRT

There are many improved RRT algorithms, a considerable number of which are applied to planar mobile robots, but many of them have high reference value for serial robots. RRT-blossom algorithm for generating new nodes by using the regression constraint function proposed by Kalisiak et al. [10]. Frazzoli et al. proposed the RRT* algorithm by reconstructing the path so that the obtained path has the characteristics of asymptotic optimization [11]. Furthermore, on the basis of this algorithm, Nasir et al. proposed RRT*-smart algorithm with intelligent sampling and path optimization function [12], to solve the problem of the slow convergence speed of RRT* algorithm. There is also a multi-directional random tree extension improvement for improving planning efficiency in narrow and complex environments [13-15]. With the development of machine learning, sampling improvement combined with intelligent algorithms has also appeared. Chen et al. used a self-learning offense to improve the quality of selected nodes [15]. The RLT* (Rapidly-Exploring Learning Trees) algorithm proposed by Shiarlis et al. [16] reduced the computational overhead of reverse reinforcement learning. At the same time, there are some algorithms for path planning by combining the learning ability of neural networks with the fast search random tree algorithm [17-18].

The Gm-RRT algorithm proposed in this paper is not only suitable for two-dimensional space but also has a fast running speed and operation effect in high-dimensional space. The improvements of this algorithm include the following three parts.

A. *Setting the target as the growth direction with a certain probability*

Firstly, the improved algorithm proposed in this paper adopts the probabilistic bias strategy as the common improved fast search random tree algorithm. That is to say, the target point will be set to random sampling node q_{rand} with a certain probability to accelerate the expansion of the search random tree to the target point and improve the efficiency.

B. *Generating multiple random sampling nodes at a time*

Secondly, if a collision occurs after each new node is generated, a new random sampling node q_{rand} will be selected and the distance between each node and the random sampling node will be recalculated, which will generate a lot of calculation. Therefore, in order to ensure that the sampling process is random, a group of random sampling nodes are sampled at one time, and the distance from the sampling node to the target point is sorted from small to large. When a collision occurs and the random tree can not be extended, the sampling node of the sub-distance is extended to repeat the above work. New sampling can be initiated when new nodes can be extended or when the random sampling node group is empty.

C. *Introducing Self-adaptive Gravitational Potential Field*

Finally, an adaptive gravitational potential field is introduced. The improved algorithm combines the advantages and disadvantages of the RRT algorithm and artificial potential field method, then proposes setting gravitational potential field at the target point. By setting the virtual gravitational potential field at the target point, the random tree tends to grow toward the target node. The gravitational potential field which affects the generating node will also be affected by the distance between the end pose and the target pose of the manipulator and the collision of the manipulator. Therefore, the algorithm has the characteristics of self-adaptation.

Due to the introduction of a self-adaptive hierarchical potential field and generating multiple random sampling nodes at a time in the new algorithm. Therefore, the new algorithm is called the Gm-RRT algorithm.

The pseudocode of the Gm-RRT algorithm is as follows:

```
Algorithm 2: Gm-RRT algorithm
Input: $q_S$ , $q_G$ , $\lambda$ , $k_d$
Output: $T$
 1.  $T.init = q_S$
 2.  for $k$ = 1:K do
 3.     $q_{rand} \leftarrow GetRandomNodeList(m)$
 4.     $q_{closest} \leftarrow NearNeighbor(T, q_{rand}(1))$
 5.     for I = 1:m do
 6.        $q_{new} \leftarrow SteerNodewithG(q_{rand}(m), q_{closest}, \lambda, k_p)$
 7.        if $CollisionFree()$ then
 8.           $T.add\_vertex(q_{new})$
 9.           $T.add\_vertex(q_{new})$
10.           break
11.        else
12.           $ChangeAttraction(k_p)$
13.        end if
14.     end for
15.     if $GettheGoal$ then
16.        return T
17.     end if
18. end for
```

Where K is the set maximum number of cycles. In pseudo-code, k_p is the proportion coefficient of the gravitational attraction of the target source to the node. Combining formula 2, the node expansion formula of the new algorithm is obtained as follows:

$$p_{new} = p_{near} + \lambda \frac{F(n)}{\|F(n)\|} \qquad (4)$$

In the function:

$$F(n) = (p_{rand} - p_{near}) + k_p(p_{goal} - p_{near}) \qquad (5)$$

IV. Experimental Simulation

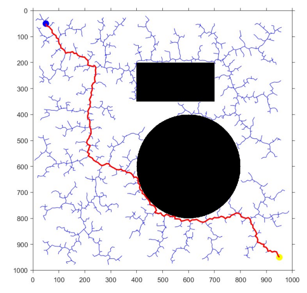
In this section, we will use the new algorithm for path planning simulation experiments to verify the feasibility of the new algorithm.

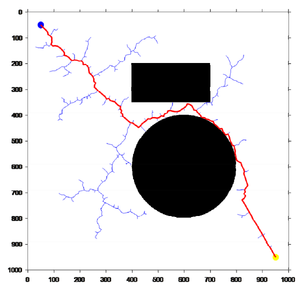
A. *Simulation Analysis on Two-Dimensional Plane*

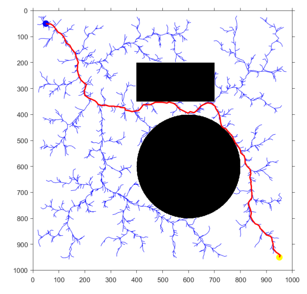
The experiment in this paper is based on the MATLAB platform. RRT algorithm, p-RRT algorithm, RRT* algorithm, and improved Gm-RRT algorithm are used to simulate and analyze the path planning in the plane respectively, which proves the effectiveness of the algorithm. Hardware information of simulation environment: Computer CPU is Intel Core i7-7700, GPU is GTX1060, memory capacity is 8G. Setting of experimental parameters: bias probability P = 0.2, step length 10, starting point coordinates (50, 50), ending point coordinates (950, 950). Twenty experiments have been carried out under the simple environment, and the time and the number of path nodes required for these algorithms are counted. The statistical results are shown in Table.1.

Table.1 Route Planning Time and Number of Route Nodes for Several RRT Algorithms in Simple Planar Environment

	average time/s	time frame/s	Average path nodes
RRT	0.441	[0.242,1.161]	171.8
bias-RRT	0.072	[0.064,0.121]	161.6
RRT*	10.263	[2.182,27.932]	96.4
Gm-RRT	0.070	[0.052,0.92]	161.7


(a)base on RRT algorithm


(b)base on bias-RRT algorithm


(c)base on RRT* algorithm

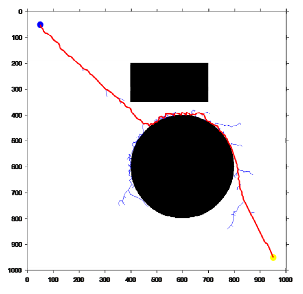

(d)base on Gm-RRT algorithm

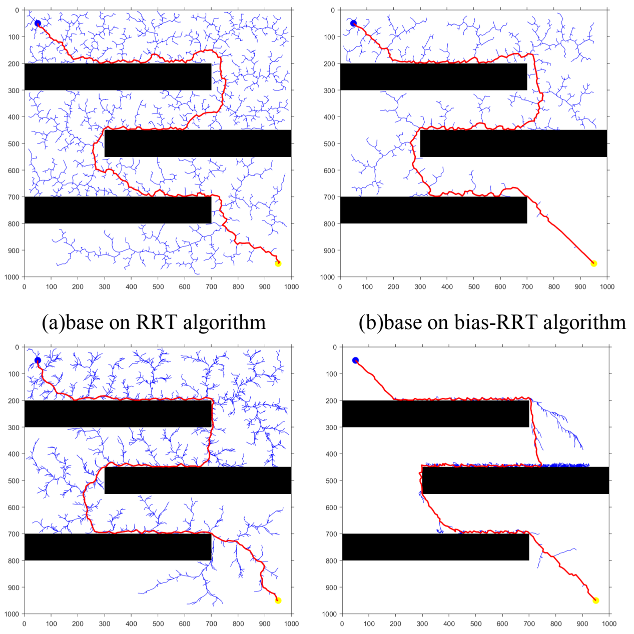
Fig1. Path Planning of Several RRT Algorithms in Simple Planar Environment

In a complex environment, a set of simulation experiments is re-conducted. The simulation results are shown in Fig.2. The RRT algorithm, bias-RRT algorithm, RRT* algorithm, and Gm-RRT algorithm are used successively in Fig.2. Twenty experiments have been carried out under the complex environment, and the time and the number of path nodes required for these algorithms are counted. The statistical results are shown in Table.2.

The tables and figures show that the proposed algorithm can guarantee the effect of path planning on the basis of less computation. It has good performance in planning time-consuming and smoothness of path.

Table.2 Route Planning Time and Number of Route Nodes for Several RRT Algorithms in complex Planar Environment

	average time/s	time frame/s	Average path nodes
RRT	0.668	[0.501,0.967]	284.2
bias-RRT	0.584	[0.445,1.022]	289.0
RRT*	11.667	[5.296,16.093]	180.6
Gm-RRT	0.548	[0.427,0.668]	280.6


(a)base on RRT algorithm (b)base on bias-RRT algorithm

(c)base on RRT* algorithm (d)base on Gm-RRT algorithm

Fig2. Path Planning of Several RRT Algorithms in Complex plane environment

B. Path planning simulation of the redundant manipulator

The robot for path planning simulation in this paper is Kuka LBR iiwa14, whose model is shown in Fig.3. The robot is a seven-degree-of-freedom redundant serial robot, which is widely used in the industry. The specific DH parameters of the robot are given in Table 3. The model shown in Fig.3 is obtained by importing its URDF file into MATLAB.

Table.3 The DH parameters of LBR iiwa14

	$\theta_i/^\circ$	d_i/m	a_i/m	$\alpha_i/^\circ$	Joint range $/^\circ$
1	θ_1	0.202	0	0	(-340~340)
2	θ_2	0.2	0	-90	(-240~240)
3	θ_3	0	0	90	(-340~340)
4	θ_4	0.409	0	-90	(-240~240)
5	θ_5	0	0	90	(-340~340)
6	θ_6	0.4	0	-90	(-240~240)
7	θ_7	0	0	90	(-360~360)
8	θ_8	0.124	0	0	fixed

In the calculation of path planning, we need to use the kinematics of the manipulator to compute the position and posture of the manipulator at the end of the workspace and each arm. The DH parameters of the robot given in Table 3 are calculated.

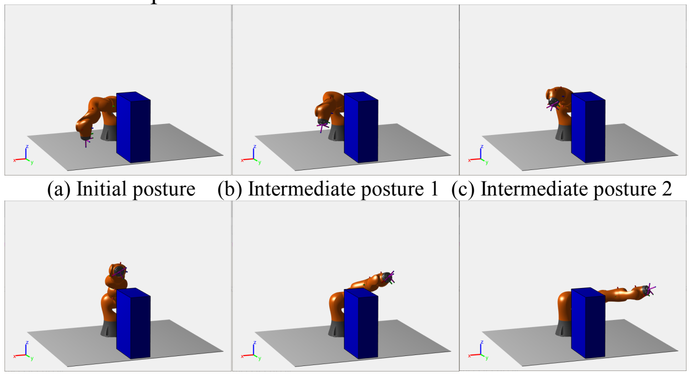
$$^{n}T_{n+1} = A_{n+1} = Rot(z,\theta_{n+1})\times Trans(0,0,d_{n+1}) \times Trans(a_{n+1},0,0)\times Rot(x,\alpha_{n+1}) \tag{6}$$

$$A_{n+1} = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Where $^{n}T_{n+1}$ is the homogeneous transformation matrix of the first n + 1 joint relative to the nth joint.

The pose of the end of the manipulator is $x=\left[r^T,\psi^T\right]^T \in R^6$, where r represents the position of the end relative to the inertial reference frame, and ψ is the attitude of the end corresponding to the inertial reference frame, where ψ is expressed in the order of *X-Y-Z* Euler angles. The known position and posture of the terminal point of the initial state is x_1 = [-1.40, 0.04, -3.05, 0.5, 0.61, 0.18], and the terminal position and posture of the final state is x_2 = [1.02, 0.58, -1.06, -0.61, 0.68, 0.58]. The obstacle is a blue cube with a size of 0.2 *0.4 *0.6, the set step size is 0.8

When the position and posture of the endpoint of the start and end states are known, the inverse solution of the Robotics System Toolbox in MATLAB is used to obtain the joint values of the manipulator under the two states. Then the joint values obtained are used as the starting point and the endpoint of the configuration space. The starting point is q_{start} = [0.95,1.36,-0.2,-0.6,0.18,1.1,-0.9], and the endpoint is q_{goal} = [2.3,1.38,-0.16,0,0.17,-0.23,-0.94], the two states are shown in Fig.3.a and Fig.3.f, respectively. The Gm-RRT algorithm is used to plan the path of the manipulator in joint space. The simulation process is shown in Fig.3. From a series of drawings, we can clearly see that the manipulator starts from the initial position, follows the trajectory obtained by the new algorithm, bypasses the blue obstacle, and finally reaches the designated position. The average time consumed is about 5.6 seconds, which shows the feasibility of the new algorithm for path planning in high-dimensional space.


(a) Initial posture (b) Intermediate posture 1 (c) Intermediate posture 2

(d) Intermediate posture 3 (e) Intermediate posture 4 (f)Final pose

Fig.3 Simulation Environment of Redundant Manipulator

Fig.4 shows the simulation track of bias RRT algorithm and GM RRT algorithm in the same environment. Fig.4.a is the planning trajectory of bias RRT algorithm; Fig.4.b is the planning trajectory of the Gm-RRT algorithm. We can see that the path of the two is basically the same after crossing the obstacle. However, compared with the Gm-RRT algorithm, bias- RRT algorithm is not so smooth and has obvious shaking motion in the path part before crossing the obstacle. This is because the former is random when expanding random tree growth, so it needs multiple planning to find a more appropriate path. In terms of time, the average time of multiple experiments of the bias-RRT algorithm needs 8.7s, and the average time of multiple experiments is about 5.6s when using the Gm-RRT algorithm. The two algorithms need a similar time, but the success rate of the Gm-RRT algorithm is higher than that of the bias-RRT algorithm. Therefore, the Gm-RRT algorithm in high-dimensional space path planning in the exploration of effective space has a good performance.

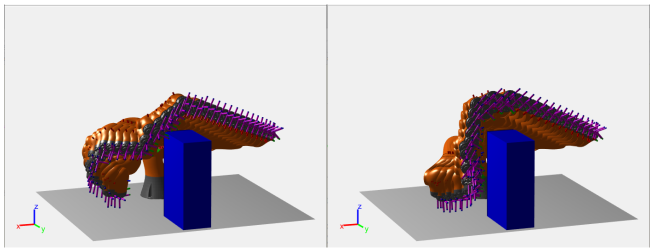

(a)Simulation trajectory-based On the bias-RRT algorithm (d) Simulation trajectory-based on Gm-RRT algorithm

Fig.4 Path Planning of Redundant Manipulators Based on Two Different Algorithms

V. Conclusion

In this paper, the path planning algorithm of the redundant manipulator is studied, and an improved algorithm Gm-RRT based on the fast search random tree algorithm is proposed. The algorithm can carry out path planning in the joint space of the robot, which saves a lot of time for inverse computation in the workspace planning. The algorithm utilizes the virtual gravity generated by the target to enhance the directivity and generates random sampling points at one time to save time. The simulation results show that the algorithm can solve the general path planning problem well. The algorithm has some shortcomings in real-time and at the narrow space environment, which is also the next improvement focus.

Acknowledgment

This research was supported by the International Science and Technology Project of Anhui Province, China (No.1604b0602018) and National Key R&D Program of China (No. 2017YFE0113200).

References

[1] Khatib O. Real-Time Obstacle Avoidance System for Manipulators and Mobile Robots[J]. International Journal of Robotics Research, 1986, 5(1):90-98.

[2] Wong S, Macdonald B. A topological coverage algorithm for mobile robots[C]// IEEE/RSJ International Conference on Intelligent Robots & Systems. IEEE, 2003.

[3] Asano T, Asano T, Guibas L, et al. Visibility-polygon search and euclidean shortest paths[J]. Foundations of Computer Science, 1975. 16th Annual Symposium on, 1985:155-164.

[4] Canny J, The Complexity of Robot Motion Planning[J]. 1988.

[5] Akahashi O, Schilling R. Motion planning in a plane using generalized Voronoi diagrams[J]. IEEE Transactions on Robotics and Automation, 1989, 5(2):150.

[6] PengLi, XinhanHuang, MinWang. A novel hybrid method for mobile robot path planning in unknown dynamic environment based on hybrid DSm model grid map[J]. Journal of Experimental & Theoretical Artificial Intelligence, 2011, 23(1):5-22.

[7] Kavraki L E, Kolountzakis M N. Latombe J C. Analysis of Probabilistic Roadmaps for Path Planning[J]. IEEE Transactions on Robotics and Automation, 1998, 14(1):166-171.

[8] Lavalle S. Rapidly-exploring random trees: a new tool for path planning[R]. Iowa State University,1998.

[9] Kun W, Bingyin R. A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm[J]. Sensors, 2018, 18(2):571-.

[10] Kalisiak M, Panne M V D. RRT-blossom: RRT with a local flood-fill behavior[C]// IEEE International Conference on Robotics & Automation. 2006.

[11] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning[J].The International Journal of Robotics Research, 2011, 30（7）:846-894.

[12] Islam F, Nasir J, Malik U, et al. RRT∗ -Smart: Rapid convergence implementation of RRT∗ towards optimal solution[C]// International Conference on Mechatronics & Automation. 2012.

[13] Akgun B, Stilman M. Sampling heuristics for optimal motion planning in high dimensions[C]// 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011. IEEE, 2011.

[14] Qureshi A H, Ayaz Y. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments[J]. Robotics and Autonomous Systems, 2015, 68:1-11.

[15] Yi D, Goodrich M A, Seppi K D. Homotopy-aware RRT*: Toward human-robot topological path-planning[C]// 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI). ACM, 2016.

[16] Cheng P. Sampling-Based Motion Planning with Differential Constraints[C]// 2005.

[17] Shiarlis K, Messias J, Whiteson S. Rapidly exploring learning trees[C]// 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.

[18] Pérez-Higueras, Noé, Caballero F, Merino L. Learning Human-Aware Path Planning with Fully Convolutional Networks[J]. 2018.

[19] Ichter B, Harrison J, Pavone M. Learning Sampling Distributions for Robot Motion Planning[J]. 2017.