Journal of System Simulation

Volume 36 | Issue 9 Article 19

9-15-2024

An Improved Path Planning Algorithm for Mobile Robots

Haijie Sun
Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China; Key Laboratory of Advanced Equipment Intelligent Manufacturing Technology of Yunnan Province, Kunming 650500, China

Hongjun San
Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China; Key Laboratory of Advanced Equipment Intelligent Manufacturing Technology of Yunnan Province, Kunming 650500, China

Le Xiao
Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China; Key Laboratory of Advanced Equipment Intelligent Manufacturing Technology of Yunnan Province, Kunming 650500, China

Dexin Yao
Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China; Key Laboratory of Advanced Equipment Intelligent Manufacturing Technology of Yunnan Province, Kunming 650500, China

See next page for additional authors

An Improved Path Planning Algorithm for Mobile Robots

Abstract

Abstract: To solve the problems of invalid sampling and non-optimal paths of the RRT, the quasi-stream avoidance algorithm is proposed. The RRT algorithm is introduced to specify the sampling interval to limit the sampling points and enhance the goal-oriented nature of sampling. The quasi-stream avoidance algorithm incorporating the A* algorithm (QSA*) is used to quickly bypass the obstacle when it is encountered. A path optimization algorithm is used to smooth the searched path. The simulation results show that compared with the RRT algorithm, the computation time of the RRT-QSA* algorithm is reduced by 96.83%~99.88%, the number of search nodes is reduced by 86.62%~96.01%, the number of path lengths is reduced by 9.9%~16.7%, and the turning angle decreased by 80.93%~93.04%. The RRTQSA* algorithm shows a more intense improvement in computational efficiency than the RRT algorithm as the map size increases.

Keywords

mobile robots, path planning, RRT, path optimization, Turtlebot2

Authors

Haijie Sun, Hongjun San, Le Xiao, Dexin Yao, Jiupeng Chen, and Xiaoyuan Yang

Recommended Citation

Sun Haijie, San Hongjun, Xiao Le, et al. An Improved Path Planning Algorithm for Mobile Robots[J]. Journal of System Simulation, 2024, 36(9): 2193-2207.

第 36 卷第 9 期 系统仿真学报© Vol. 36 No. 9
2024 年 9 月 Journal of System Simulation Sept. 2024

一种改进的移动机器人路径规划算法

孙海杰[1,2]，伞红军[1,2*]，肖乐[1,2]，姚得鑫[1,2]，陈久朋[1,2]，杨晓园[1,2]
(1. 昆明理工大学 机电工程学院，云南 昆明 650500；2. 云南省先进装备智能制造技术重点实验室，云南 昆明 650500)

摘要： *为解决快速随机扩展树算法(RRT)无效采样以及路径不最优等问题，提出一种基于RRT和A*算法的拟水流避障算法RRT-QSA*。在采样上引入RRT算法规定采样区间来限制采样点，增强采样的目标导向性；遇到障碍物时采用融合了A*算法的拟水流避障算法迅速绕过障碍物；采用路径优化算法对搜索到的路径进行路径优化。*仿真结果表明：与RRT算法相比，RRT-QSA*算法的计算时间减少了96.83%~99.88%，搜索节点数减少了86.62%~96.01%，路径长度数减少了9.9%~16.7%，转折角度减少了80.93%~93.04%。随着地图的增大，RRT-QSA*算法比RRT算法计算效率的提升更加明显。

关键词： *移动机器人；路径规划；快速随机扩展树；路径优化；*Turtlebot2
中图分类号：TP242.6 文献标志码：A 文章编号：1004-731X(2024)09-2193-15
DOI: 10.16182/j.issn1004731x.joss.23-0585
引用格式: 孙海杰, 伞红军, 肖乐, 等. 一种改进的移动机器人路径规划算法[J]. 系统仿真学报, 2024, 36(9): 2193-2207.
Reference format: Sun Haijie, San Hongjun, Xiao Le, et al. An Improved Path Planning Algorithm for Mobile Robots[J]. Journal of System Simulation, 2024, 36(9): 2193-2207.

An Improved Path Planning Algorithm for Mobile Robots

Sun Haijie[1,2], *San Hongjun*[1,2*], *Xiao Le*[1,2], *Yao Dexin*[1,2], *Chen Jiupeng*[1,2], *Yang Xiaoyuan*[1,2]
(1. Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China;
2. Key Laboratory of Advanced Equipment Intelligent Manufacturing Technology of Yunnan Province, Kunming 650500, China)

Abstract: To solve the problems of invalid sampling and non-optimal paths of the RRT, *the quasi-stream avoidance algorithm is proposed. The RRT algorithm is introduced to specify the sampling interval to limit the sampling points and enhance the goal-oriented nature of sampling. The quasi-stream avoidance algorithm incorporating the A* algorithm* (*QSA**) *is used to quickly bypass the obstacle when it is encountered. A path optimization algorithm is used to smooth the searched path.* The simulation results show that compared with the RRT algorithm, the computation time of the RRT-QSA* algorithm is reduced by 96.83%~99.88%, the number of search nodes is reduced by 86.62%~96.01%, the number of path lengths is reduced by 9.9%~16.7%, and the turning angle decreased by 80.93%~93.04%. The RRT-QSA* algorithm shows a more intense improvement in computational efficiency than the RRT algorithm as the map size increases.

Keywords: mobile robots; path planning; RRT; path optimization; Turtlebot2

收稿日期：2023-05-18 修回日期：2023-07-08
基金项目：云南省科技厅重大专项(202002AC080001)；云南省基础研究计划(202301AU070059)
第一作者：孙海杰(1998-)，男，硕士生，研究方向为移动机器人路径规划。
通讯作者：伞红军(1976-)，男，副教授，博士，研究方向为机器人技术理论及应用、机电产品设计与开发、现代农业装备研发等。

0 引言

自主导航是智能车辆最重要的功能之一。机器人导航是在避开障碍物的同时自主移动到目标位置的过程。该过程有4个基本组成部分：感知、定位、路径规划、运动控制。机器人的路径规划可以看作是在工作环境中避开障碍物的同时从起始位置到目的地的平移和旋转序列[1]。有2种技术涵盖了机器人路径规划的所有方法：全局路径规划(离线路径规划)和局部路径规划(在线路径规划)[2]。全局路径规划是指根据已知的环境地图生成路径。对未知或动态障碍的反应不足。局部路径规划基于机载传感器的信息，在全局路径的一个片段上提供路径，它可以在动态环境中有效地工作。当目标距离较远或环境杂乱时，该方法效率较低[3]。全局路径规划算法主要有基于图搜索的Dijkstra[4]、A*[5]、D*算法[6]，以及基于采样的RRT等[7]。基于采样的算法是最流行的路径规划算法之一，它是一种通用的求解方法，不受受控对象复杂性的约束[8]。其中，RRT算法具有灵活强大的搜索能力，能用于各种复杂环境下的路径规划[9]，但是其随机采样存在的大量无效采样以及路径不最优等问题亟待解决[10]。

为了解决上述问题，文献[11]通过引入偏向目标的随机点来改进算法采样的无目标性，从而使算法能够加速收敛。文献[12]提出了一种基于目标约束采样和目标偏置扩展的改进算法(goal-bias constrained sampling and goal biased extending RRT*, GCSE-RRT*)，首先，用目标偏置的有约束采样替代原来的均匀采样，接着，改变了传统RRT朝着采样点扩展的思想，通过分配不同权重的方式使新节点的扩展方向由采样点和目标点共同决定，最后，采用3次B样条曲线对改进算法得到的路径进行了平滑处理。文献[13]提出了一种新的最优路径规划算法PQ-RRT*，通过采用向着目标点方向采样的策略，扩展了选择父节点过程的搜索范围。文献[9]提出了一种基于RRT*的最优路径规划算法F-RRT*，基于三角形不等式创建节点，以降低每个采样点的代价。文献[14]提出了建立在双向RRT算法基础上的KB-RRT*算法，在路径规划过程中，KB-RRT*考虑了运动约束，生成平滑的路径并保持运动连通性，利用有效的分支剪枝策略获得更好的路径，以高代价去除边缘。在寻找最近邻点时，提出了广义距离作为距离的度量，用于加速现有树节点的遍历。

对于RRT算法，只要可行解存在，就一定能被找到，但是RRT算法的收敛速度问题一直存在，且并不能保证所得出的可行路径是相对优化的。因此，许多学者一直在尝试解决路径优化的问题，例如RRT*算法[15]。RRT*算法的主要特征是先找出一条初始路径，之后随着采样点的增加，在给定的循环次数下不断地对路径进行优化。由此可知RRT*算法是渐进优化的，永远不可能在有限的时间中得出最优的路径[16]。所以，RRT*算法只是得到了一条相对优化的路径而没有解决收敛速度的问题[17]。而在机器人实际应用中是强调实时性的，因此，本文通过改善传统RRT算法的不足，并将RRT算法和A*算法与拟水流避障算法(quasi-stream avoidance algorithm, QS)结合在一起，形成了适用于机器人实际应用的(quasi-stream avoidance based on RRT algorithm and A* algorithm, RRT-QSA*)算法，来提高传统RRT算法的运行效率。

1 问题的描述

1.1 环境的表示方法

本文采用的环境地图建模方法是网格法。该方法因其简单、有效、对障碍物的适应性强而被认为是路径规划中最常用的方法之一[17]。一般来说，在栅格地图中，0用来表示自由区域空间，用白色表示，1用来表示障碍物，用黑色表示。本文采用矩形坐标系结合序列号法来确定网格[18]。如图1所示，移动机器人的工作环境图是一个10×10的网格。

DOI: 10.16182/j.issn1004731x.joss.23-0585

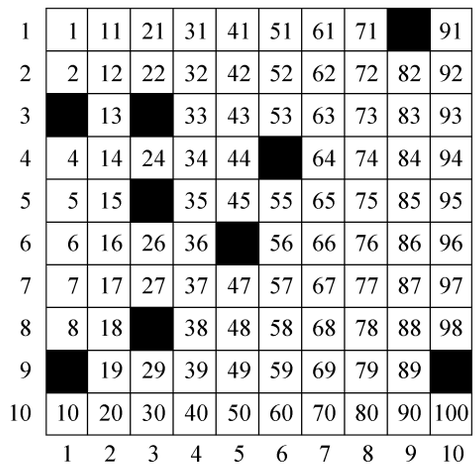


图1 网格地图
Fig. 1 Grid map

网格序号和坐标之间的转换关系为

$$i_k=\begin{cases}R, & \mathrm{mod}(\mathrm{ind}(i_k,j_k),R)=0\\ \mathrm{mod}(\mathrm{ind}(i_k,j_k),R), & \text{否则}\end{cases} \tag{1}$$

$$j_k=\begin{cases}\mathrm{int}\left(\dfrac{\mathrm{ind}(i_k,j_k)}{R}\right), & \mathrm{mod}(\mathrm{ind}(i_k,j_k),R)=0\\ \mathrm{int}\left(\dfrac{\mathrm{ind}(i_k,j_k)}{R}\right)+1, & \text{否则}\end{cases} \tag{2}$$

$$\mathrm{ind}(i_k,j_k)_{\mathrm{transfer}}=R(j_k-1)+i_k \tag{3}$$

式中：ind用于计算当前节点的索引号；R为行数的总数量；mod()为余数函数，即返回除法的余数部分，int()为取整函数，即返回除法的整数部分。比如，对于10×10网格，对于索引号60，其坐标为(10, 6)，对于坐标(3, 2)，其索引号13。

1.2 RRT算法

RRT原理为维护一棵由路径点组成的随机树：从起点开始，在空间中随机采样，并找到路径树上与采样点最接近且能与它无障碍连接的点，连接这个点与采样点，将采样点加入路径树，直至终点附近区域被探索到。但是，这种方式无法保证得到的路径是最优的。

在基本RRT算法中，P用来表示一个节点的状态信息，P_{init}表示起点，在算法开始寻路时，将起点P_{init}添加到随机树。在地图中进行随机采样，获得一个随机采样点P_{rand}，再在随机树中搜索出距离随机采样点最近的点P_{near}，在P_{near}和P_{rand}的连线上距离P_{rand}步长u(具体长度自定义，在本文网格环境中定义为2个单元格，即$2r$，r为网格比例，本文网格比例r取1，即网格的长度与机器人外接圆直径的比值为1:1)的位置生成一个新节点P_{new}，判断P_{new}和P_{near}之间是否受到障碍物影响，若不影响则将P_{new}加入到随机树中，重复这个过程，直到找到目标点。

RRT算法的具体步骤如图2所示。

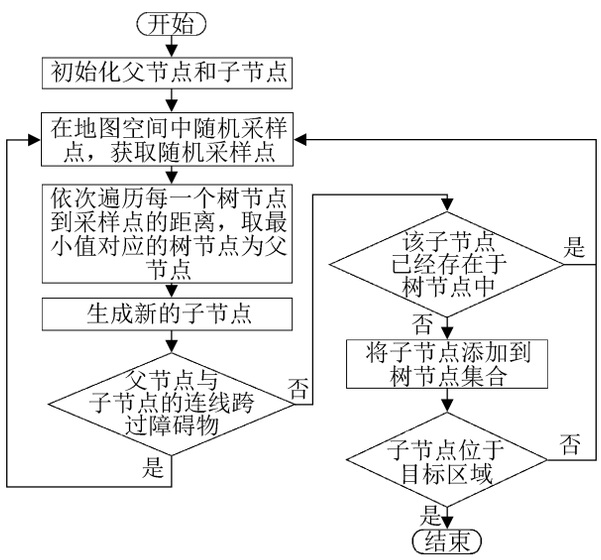


图2 RRT流程图
Fig. 2 Flowchart of RRT algorithm

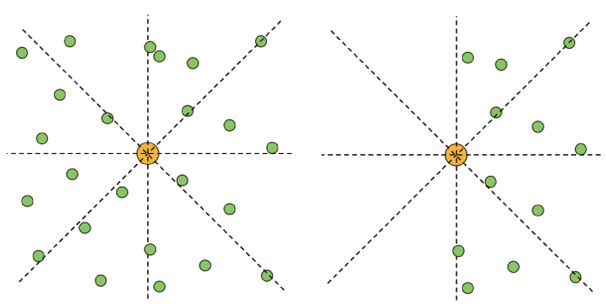
2 RRT-QSA*算法

2.1 改进RRT算法

在传统RRT算法中，由于采样点的随机性，RRT算法在进行路径规划时，其搜索空间(随机树)总是能遍布整张地图，因此，在小型较为复杂的地图场景中实用性较强。但是，将该算法应用于大型地图时，由于RRT算法盲目地随机搜索会导致状态空间中有很多不必要的节点会被搜索到，其中大量被搜索到的节点与生成的路径无关，会使搜索速度变慢，使RRT算法在大型场景中的适应性不强，基于这个问题，本文将采取一种限制采样点的方法来限制传统RRT算法对地图进行随机搜索的行为。

由于采样点的随机性，可以认为采样点的选取范围是[0°,360°]，共有8个采样区间，如图3(a)所示，对于每一个树节点，其采样点的选取空间总是遍布四周。将8个采样区间划分为2个限定采样区间E_{L_1}和E_{L_2}。

$$E_L=\begin{cases}E_{L_1}\begin{cases}E_1=[0°,45°)\\E_2=[45°,90°)\\E_7=[270°,315°)\\E_8=[315°,360°)\end{cases}\\E_{L_2}\begin{cases}E_3=[90°,135°)\\E_4=[135°,180°)\\E_5=[180°,225°)\\E_6=[225°,270°)\end{cases}\end{cases}\tag{4}$$



(a) 采样总体区间 (b) 采样限制区间

图3 限制采样区间

Fig. 3 Limit sampling interval

如图4所示，在每一次进行搜索采样之前，先确定当前树节点相对于目标点G的位置，计算当前采样角θ_{Sample}(对于刚开始搜索时，其树节点即起点S)，再判断当前采样角θ_{Sample}属于哪一个限定采样区间E_L，根据不同的采样角，把采样点的采样范围限制在其所对应的限定采样区间内，采样点将在采样区间内选取，如图3(b)所示。

限制了采样区间后的RRT算法极大地提高了搜索效率，然而，当采用限制采样区间的RRT算法应用于一些障碍物比较多的地图时，很容易被障碍物挡住，无法继续向目标点扩展搜索，因此，本文提出了一种用于避开障碍物的算法，并在每次子节点生成后增加了一个障碍物检测步骤，如果检测到障碍物，就基于当前子节点执行避障算法。

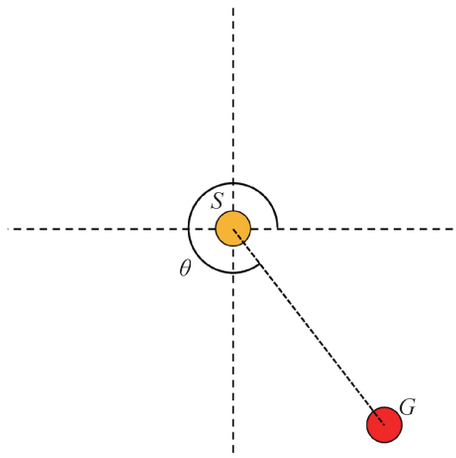
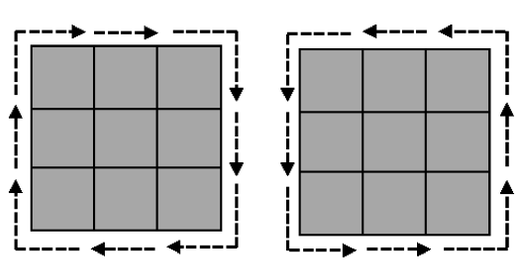


图4 确定位置

Fig. 4 Determine location

2.2 拟水流避障算法QS

自然界中小溪的水在流动中会穿越各种石头、浅滩、树木等障碍，这与自主移动机器人的寻路过程非常相似。基于这种启发，本文提出了一种在对移动机器人进行路径规划时模拟水流流动轨迹避开障碍物的方法，称为拟水流避障算法(quasi-stream avoidance algorithm, QS)。拟水流在避障时有2种基础流动方式：顺时针或者逆时针，如图5所示。在顺时针流动方式下，由↑↗→↘↓↙←↖8种流动方法按照顺时针运行规律排列组成；在逆时针流动方式下，由↓↘→↗↑↖←↙8种流动方法按照逆时针运行规律排列组成，其流动条件在不同流动模式下不同。在选择了流动方式之后，拟水流将按照该流动方式下的3种流动模式流动，如图6~7所示。



(a) 顺时针 (b) 逆时针

图5 两种流动方式

Fig. 5 Two types of flow

DOI: 10.16182/j.issn1004731x.joss.23-0585

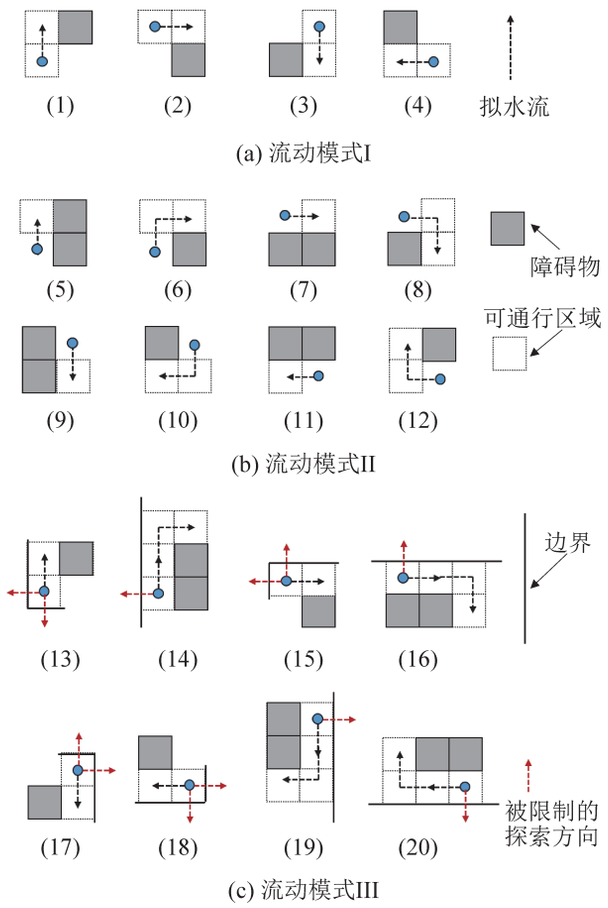


图6　顺时针流动方式
Fig. 6　Clockwise flow types

在拟水流进行避障时，增加一个限定条件：把障碍物看作是可以支撑拟水流流动的石头，且拟水流必须依附障碍物流动，不能离开障碍物。

流动模式I：确定拟水流与障碍物之间的位置关系，如果拟水流位于障碍物的左上、右上、左下、右下4个对角位置，拟水流需贴合障碍物。

流动模式II：拟水流靠近障碍物后，就会流过障碍物。一般采用常规的流动模式。

流动模式III：将拟水流即将流出栅格地图(超出地图的索引值)时所处的位置定义为极端位置，即边界，当拟水流在极端位置流动时很可能会溢出地图区域，超出地图的索引值，这时需要采用极端流动模式。当出现了边界后，不能向超出边界的方向进行探索，所以对应的探索方向会被限制。且在执行具体的流动方法前要先满足对应的极端限定条件，再判断是否满足流动条件。表1为算法中不同流动方法对应的流动条件，表2~3为2种基础流动方式下，3种流动模式下不同流动方法与流动条件的对应关系。

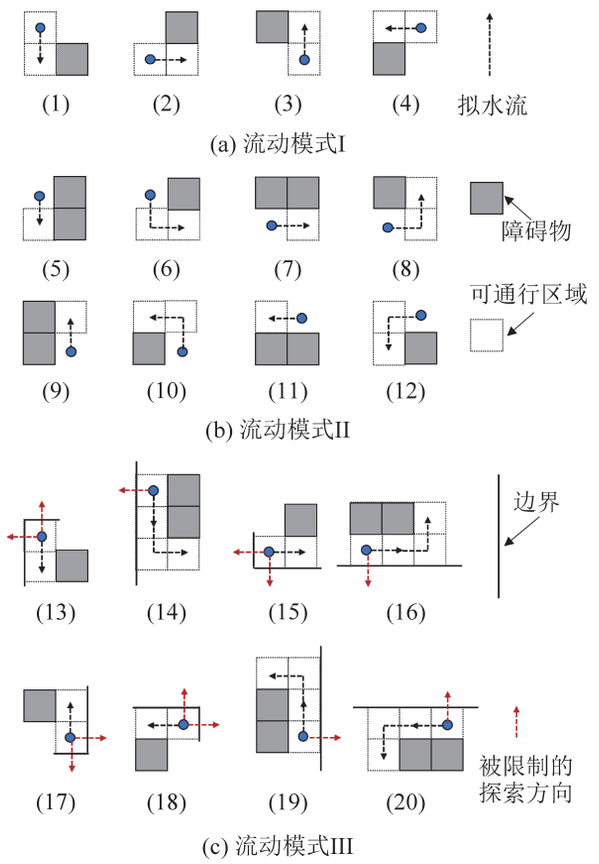


图7　逆时针流动方式
Fig. 7　Counter-clockwise flow types

表1　代码含义关系对照表
Table 1　Correspondence table of code meaning

流动条件	含义	符号
field [i, j+1]	向右搜索	R
field [i+1, j]	向下搜索	D
field [i, j-1]	向左搜索	L
field [i-1, j]	向上搜索	U
field [i+1, j+1]	向右下搜索	RD
field [i-1, j+1]	向右上搜索	RU
field [i+1, j-1]	向左下搜索	LD
field [i-1, j-1]	向左上搜索	LU
= 0	搜索结果为可通行区域	0
= 1	搜索结果为可障碍物	1

表2 顺时针流动条件对应关系表

Table 2 Correspondence table of flow conditions (clockwise)

流动	模式	极端限定条件	流动条件
↑	M1-Fig.6(1)	\	U0, RU1, R0
	M2-Fig.6(5)	\	R1, RU1, U0
	M3-Fig.6(14)	j=0	R1, RU1, U0
	M3-Fig.6(13)	j=0 & i=R	RU1, U0
↗	M2-Fig.6(6)	\	R1, U0, RU0
	M3-Fig.6(14)	j=0	R1, U0, RU0
→	M1-Fig.6(2)	\	R0, RD1, D0
	M2-Fig.6(7)	\	D1, RD1, R0
	M3-Fig.6(16)	i=0	D1, RD1, R0
	M3-Fig.6(15)	i=0 & j=0	RD1, R0
↘	M2-Fig.6(8)	\	D1, RD0, R0
	M3-Fig.6(16)	i=0	D1, R0, RD0
↓	M1-Fig.6(3)	\	D0, LD1, L0
	M2-Fig.6(9)	\	L1, LD1, D0
	M3-Fig.6(19)	j=C	L1, LD1, D0
	M3-Fig.6(17)	i=0 & j=C	LD1, D0
↙	M2-Fig.6(10)	\	L1, D0, LD0
	M3-Fig.6(19)	j=C	L1, LD0, D0
←	M1-Fig.6(3)	\	L0, LU1, U0
	M2-Fig.6(4)	\	U1, LU1, L0
	M3-Fig.6(20)	i=R	U1, LU1, L0
	M3-Fig.6(18)	i=R & j=C	LU1, L0
↖	M2-Fig.6(12)	\	U1, L0, LU0
	M3-Fig.6(20)	i=R	U1, LU0, L0

注：\代表并不存在边界，不需要极端限定条件。

表3 逆时针流动条件对应关系表

Table 3 Correspondence table of flow conditions (counter-clockwise)

流动	模式	极端限定条件	流动条件
↓	M1-Fig.7(1)	\	D0, RD1, R0
	M2-Fig.7(5)	\	R1, RD1, D0
	M3-Fig.7(14)	j=0	R1, RD1, D0
	M3-Fig.7(13)	j=0 & i=0	RD1, D0
↘	M2-Fig.7(6)	\	R1, RD0, D0
	M3-Fig.7(14)	j=0	R1, D0, RD0
→	M1-Fig.7(2)	\	R0, RU1, U0
	M2-Fig.7(7)	\	U1, RU1, R0
	M3-Fig.7(16)	i=R	U1, RU1, R0
	M3-Fig.7(15)	i=R & j=0	RU1, R0
↗	M2-Fig.7(8)	\	U1, R0, RU0
	M3-Fig.7(16)	i=R	U1, R0, RU0
↑	M1-Fig.7(3)	\	U0, LU1, L0
	M2-Fig.7(9)	\	L1, LU1, U0
	M3-Fig.7(19)	j=C	L1, LU1, U0
	M3-Fig.7(17)	i=R & j=C	LU1, U0
↖	M2-Fig.7(10)	\	L1, U0, LU0
	M3-Fig.7(19)	j=C	L1, LU0, U0
←	M1-Fig.7(3)	\	L0, LD1, D0
	M2-Fig.7(4)	\	D1, LD1, L0
	M3-Fig.7(20)	i=0	D1, LD1, L0
	M3-Fig.7(18)	i=0 & j=C	LD1, L0
↙	M2-Fig.7(12)	\	D1, L0, LD0
	M3-Fig.7(20)	i=0	D1, LD0, L0

注：\代表并不存在边界，不需要极端限定条件。

3 种流动模式的关系：先执行流动模式 I，后执行流动模式 II 或流动模式 III。流动模式 I 为贴合模式，当拟水流(当前子节点)处于障碍物的对角位置时先贴合(靠近)障碍物，否则无法继续执行接下来的流动模式 II 或流动模式 III。拟水流避障算法的运行流程如图 8 所示。

探索过程：流动模式 II 下，需要对 8 种流动方法的每一种方法的流动条件依次进行判断，这个过程即为探索，满足哪种流动方法的条件就执行哪种流动方法。而流动模式 III 下，只对满足边界条件的流动方法进行探索。

如图 9(a)(d)所示，在进行拟水流避障的时候，同时从顺时针和逆时针 2 个方向出发，在到达目标点后，比较 2 个方向上的路径长度，选取较短的路径进行保留。如图 9(b)(c)所示，如果在流动的过程中拟水流被边界或障碍物阻断了，则直接舍弃这一侧的路径，结果如图 9(e)(f)所示。在拟水流避障时，为了避免已成功搜索到的路径点与之前搜索到的路径点重复，把每次成功搜索到的路径点定义为 $P(k)$，把顺时针搜索成功的路径点集合定义为 temp_path_left，把逆时针搜索成功的路径点集合定义为 temp_path_right。如果 $P(k)\notin$ temp_path_left 则把搜索到的路径点 $P(k)$ 存储在 temp_path_left 中，同理，如果 $P(k)\notin$ temp_path_right 则把搜索到的路径点 $P(k)$ 存储在 temp_path_right 中。

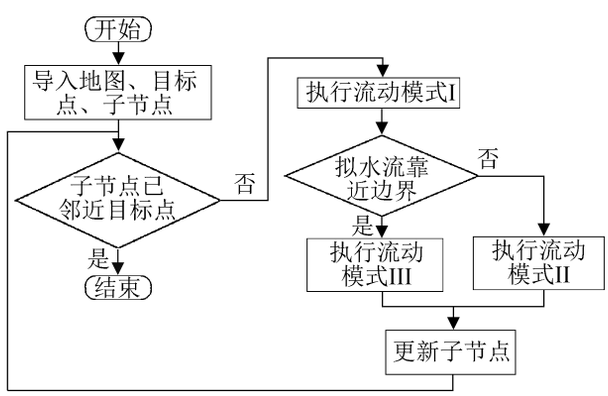


图8 QS流程图

Fig. 8 Flowchart of QS

DOI: 10.16182/j.issn1004731x.joss.23-0585

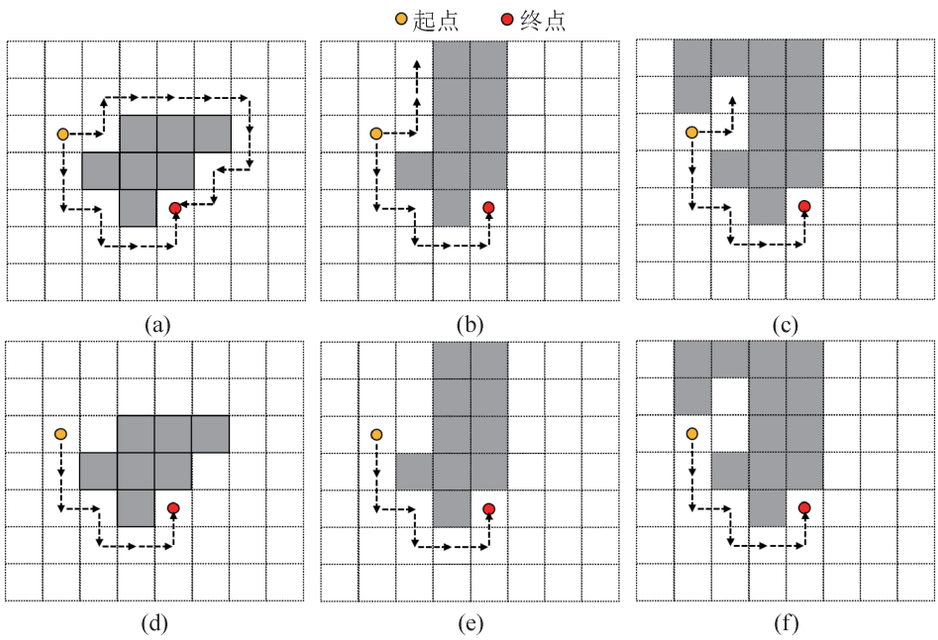


图9　拟水流避障示意图
Fig. 9　Schematic diagram of QS

拟水流避障算法具有很强的自主搜索能力，然而，如果找不到退出的出口拟水流将会一直沿着障碍物流动，所以，本文将给出了2种退出条件。

2.3 QSA*算法

A*算法是静态环境中用于求解最优路径的有效的直接搜索算法，结合了Best-first Search和Dijkstra算法的思想，在保证得到最优路径的基础上，采用启发式搜索[19]。A*算法通过一个估价函数来确定搜索方向，从起点开始向周围扩展，通过估价函数计算得到周围每个节点的代价值，选择最小代价节点作为下一个扩展节点，重复这一过程直到到达目标点，生成最终路径[20]。在搜索过程中，由于路径上的每个节点都具有最小代价的节点，因此，得到的路径代价是最小的。A*算法的估价函数为

$$f(n)=g(n)+h(n) \tag{5}$$

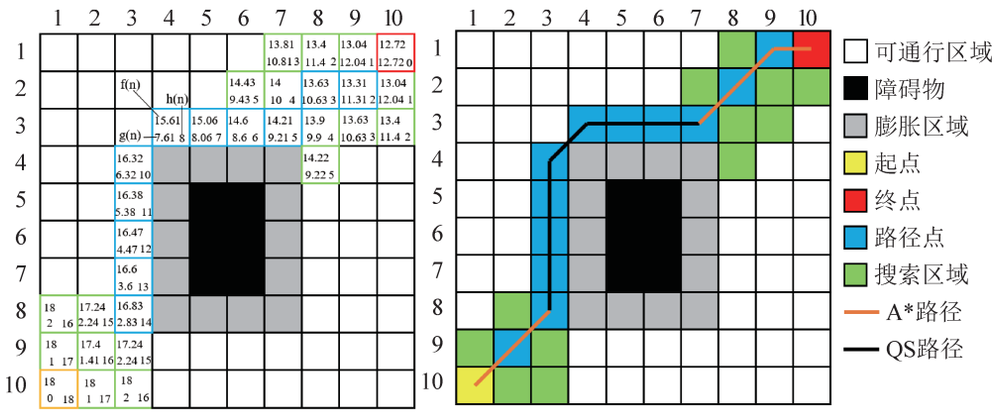
式中：$f(n)$为从起始点经由任意节点n到达目标点的估价函数；$g(n)$为起始点S到节点n的实际代价；$h(n)$为节点n到目标点G的估计代价。本文采用欧式距离度量两点之间的实际代价，采用曼哈顿距离度量两点之间的估计代价。

$$g(n)=\sqrt{(x_S-x_n)^2+(y_S-y_n)^2} \tag{6}$$

$$h(n)=|x_G-x_n|+|y_G-y_n| \tag{7}$$

式中：(x_n,y_n)、(x_G,y_G)分别为节点n和目标点G的坐标。

由拟水流避障算法QS的原理可知，在进行避障时需要找到地图中2个接近障碍物的点(临时起点和临时目标点)，在实际应用时机器人只能获取临时起点，并不清楚临时目标点的位置，因此，本文在拟水流避障算法中融入了A*算法中的启发函数的思想，在寻路过程中计算每一个成功搜索到的路径点的F值(估计函数)并进行储存，并实时监测F值的变化情况，将融合后的算法称为QSA*算法。如图10(a)所示，在一般情况下，在A*算法和QS算法的共同运作下，整个搜索过程是一直向着目标方向进行的，所以，其F值是一直减小的，例如从节点(8, 3)到节点(3, 7)时其F值一直再减小，而当F值增大时，例如，从节点(3, 7)到节点(4, 8)时其F值开始增大，说明当前已经脱离了障碍物，此时从当前节点的上一个节点退出QSA*算法，即从节点(3, 7)退出。



(a) 搜索节点 (b) 搜索路径

图 10 QSA*算法避障

Fig. 10 Algorithm obstacle avoidance of QSA*

QSA*流程如下：

(1) 开始导入地图、起点、终点和子节点；

(2) 判断子节点是否已临近终点，若是，算法结束运行，否则，执行流动模式I，并统计F值；

(3) 判断拟水流是否靠近边界，若是，执行流动模式III，并统计F值，否则，执行流动模式II，并统计F值；

(4) 判断F值是否增大，若是，算法结束运行，否则，重复(2)。

如果在执行QSA*算法过程中被迷宫类型的障碍物挡住了，此时机器人找到的节点因为A*算法的限制不能朝着与目标点相反的方向扩展，容易造成死锁，所以在这个时候增加一个补救措施：基于当前子节点 Childnode 向着目标点方向搜索，把搜索到的第一个非障碍物节点作为临时目标点，不再记录节点的F值，单独执行拟水流避障算法，如图 11 所示。

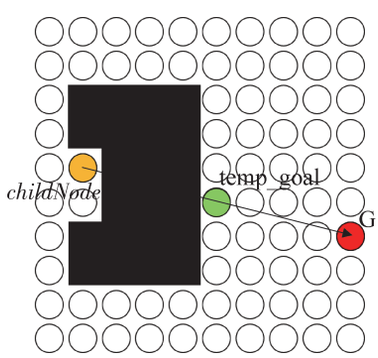


图 11 搜索临时目标点

Fig. 11 Search for temporary target

2.4 RRT-QSA*算法

将限制了采样点的RRT算法和加入了A*算法的拟水流避障算法QSA*融合在一起，组成新的算法RRT-QSA*。当该算法开始运行时，以路径的起点作为初始节点进行搜索。在搜索过程中，如果遇到了障碍物就调用QSA*算法进行避障，当脱离障碍物后继续执行RRT算法进行搜索，重复迭代，直到找到目标节点。

融合算法RRT-QSA*进行路径搜索的具体流程如下：

(1) 初始化父节点和子节点；

(2) 在地图中随机采样，获得随机采样点；

(3) 依次遍历每个树节点到采样点的距离，取最小值对应的树节点为父节点；

(4) 并生成新的子节点；

(5) 若子节点遇到障碍物，则调用QSA*算法避障并将生成的路径点添加到树节点中，判断子节点是否位于目标区域，如果是，则算法结束运行，否则重复(2)；

(6) 若子节点没有遇到障碍物，判断该子节点是否在树节点中，若是，则重复(2)，若不是，则将子节点添加到树节点中，并判断子节点是否位于目标区域，若是，则算法结束运行，若不是则重复(2)。

DOI: 10.16182/j.issn1004731x.joss.23-0585

2.5 路径优化

一方面，由于RRT算法自身特点，其规划出来的路径总是存在很多转折点，非常的不平整，另一方面由于拟水流避障算法是基于障碍物寻路的算法，容易产生贴着障碍物的路径，在移动机器人实际应用时效果不好，因此，本文提出了一种路径优化算法，在RRT-QSA*算法规划好路径后对路径进行平滑优化处理。优化流程如下：

(1) 从路径集{P}中取出第1个路径点P_1，作为第一次平滑处理的起点；

(2) 找到第2个路径点P_2；

(3) 在地图中从P_1开始向P_2扩展；

(4) 检测扩展区域是否有障碍物，如果没有，从P_1向下一个路径点P_k继续扩展，如果检测到扩展区域中有障碍物，那么将第一次平滑处理的起点P_1和P_{k-1}保存在平滑路径集{P_{Smooth}}中；

(5) 开始第二次平滑处理，直到扩展到终点。

具体路径平滑处理过程如图12所示，其中，扩展区域用浅绿色表示，正在被检测的路径点用橙色高亮显示。导入如图12(a)所示的地图和路径集，第1次平滑处理开始，从路径集中的第1个点(1, 1)开始，向第2个点(2, 1)扩展，没有检测到障碍物，继续向第3个点(2, 2)扩展，扩展区域如图12(b)所示，没有检测到障碍物，继续向下一个点(3, 2)，(4, 2)，(4, 3)扩展，扩展区域如图12(b)~(e)所示，直到扩展到点(4, 4)时检测到障碍物，扩展区域如图12(g)所示，此时只保留第一个点(1, 1)和当前扩展点的前一个点(4, 3)，第一次平滑处理结束；第二次平滑处理从上一次平滑处理的最后一个点(4, 4)开始向后扩展，如图12(h)~(i)所示，保留点(4, 4)和点(4, 6)；第二次平滑处理如图12(k)所示，保留点(5, 6)和点(8, 8)；后续路径点之间存在障碍物无法进行平滑处理，至此平滑处理结束，得到如图12(l)所示的路径。

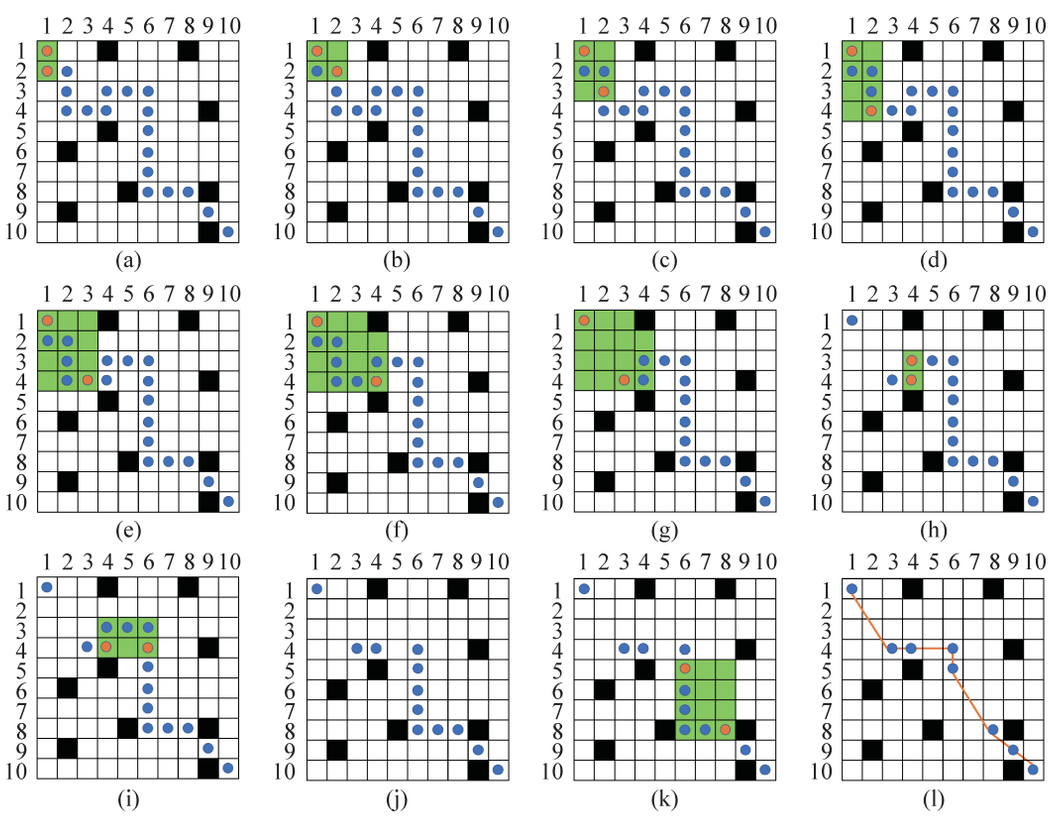


图12　路径优化示意图

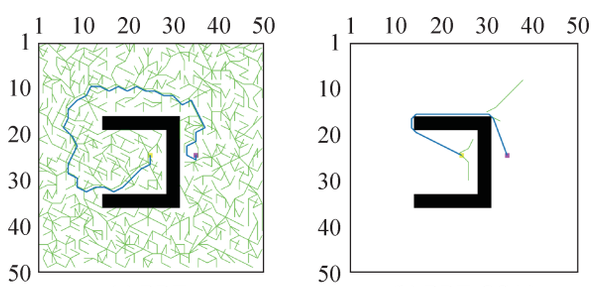
Fig. 12　Schematic diagram of path optimization

3 仿真与实验

3.1 仿真结果及分析

为了验证RRT-QSA*算法在解决机器人路径规划问题上的可行性和有效性，本文选取了在路径规划性能测试中比较有代表性的2个迷宫地图和1个狭窄通道地图，分别对RRT算法和RRT-QSA*算法进行了100次仿真模拟。仿真环境如下：Windows10 64位；处理器Intel (R) Core (TM) i7-12700U；主频2.69 GHz；内存16 GB。由于RRT算法的随机性和不确定性，导致每一次运行时算法得出的数据都是不同的，所以，针对RRT算法和RRT-QSA*算法数据波动的情况采用箱线法对2种算法性能进行进一步的对比分析。

在50×50网格迷宫下，以黄色网格坐标点为起始点，以红色网格坐标点为终点，规划结果如图13所示，图中绿色部分表示算法生成树节点分布情况，蓝色线段表示生成的路径。



(a) RRT (b) RRT-QSA*

图13 迷宫地图I

Fig. 13 Maze map-I

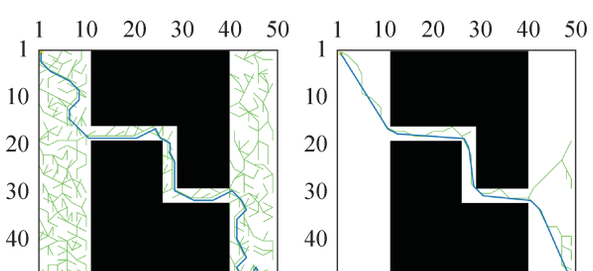
在迷宫地图I中，起始点设置为(25, 25)，目标点设置为(25, 35)，图13显示了这2种算法在迷宫地图I中的路径结果及节点搜索情况，2种算法的具体数据如表4所示。与RRT算法相比，在迷宫地图I中RRT-QSA*算法的计算时间减少了98.4%，搜索节点数减少了89.53%，路径长度数减少51.15%，转折角度减少了83.59%。因此，RRT-QSA*算法在迷宫地图I中的表现要优于RRT算法。

表4 算法数据对比(迷宫地图I)

Table 4 Algorithms statistics comparison (maze map-I)

指标	RRT			
	时间/s	路径长度	节点数	转折角度/(°)
均值	0.25	87.74	483.80	1 861.75
标准差	0.26	3.27	206.32	251.30
最大值	0.87	110.65	878	2 252.23
最小值	0.02	72.15	185	1 513.73
指标	RRT-QSA*			
	时间/s	路径长度	节点数	转折角度/(°)
均值	0.004	42.86	50.67	305.51
标准差	0.003	1.79	2.72	6.95
最大值	0.015	46.45	53	308.99
最小值	0.002	41.96	45	291.59

在狭窄通道地图中，起始点设置为(1, 1)，目标点设置为(50, 50)，图14为这2种算法在狭窄通道地图中的路径结果及节点搜索情况，2种算法的具体数据如表5所示。与RRT算法相比，在狭窄通道地图中RRT-QSA*算法的计算时间减少了85%，搜索节点数减少了80.81%，路径长度数减少8.88%，转折角度减少了79.09%。由此可知，RRT-QSA*算法在狭窄通道地图中的表现要优于RRT算法。



(a) RRT (b) RRT-QSA*

图14 狭窄通道地图

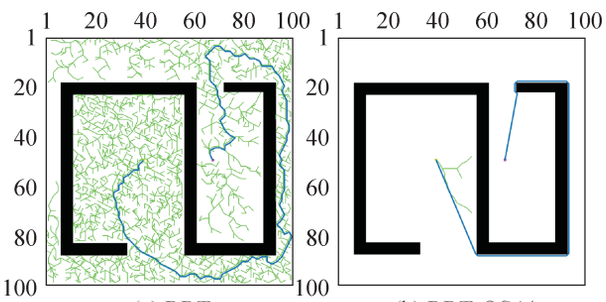
Fig. 14 Narrow passage map

在迷宫地图II中，起始点设置为(50, 79)，目标点设置为(50, 135)，图15为2种算法在迷宫地图II中的路径结果及节点搜索情况，两种算法的具体数据如表6所示。与RRT算法相比，在迷宫地图II中RRT-QSA*算法的计算时间减少了85%，搜索节点数减少了80.81%，路径长度数减少8.88%，转折角度减少了79.09%。由此可知，RRT-QSA*算法在迷宫地图II中的表现要优于RRT算法。

DOI: 10.16182/j.issn1004731x.joss.23-0585

表5　算法数据对比(狭窄通道地图)
Table 5　Algorithms statistics comparison (narrow passage map)

指标	RRT				RRT-QSA*			
	时间/s	路径长度	节点数	转折角度/(°)	时间/s	路径长度	节点数	转折角度/(°)
均值	0.20	89.89	387.80	1 804.36	0.03	81.91	74.40	377.35
标准差	0.16	3.27	81.42	114.65	0.01	2.09	47.36	42.26
最大值	0.66	95.09	547	1 988.53	0.03	86.35	216	470.11
最小值	0.05	84.61	243	1 632.59	0.02	78.52	53	321.51



(a) RRT　(b) RRT-QSA*

图15　迷宫地图II
Fig. 15　Maze map-II

为进一步验证RRT-QSA*算法的性能，将实际应用中较为常见的A*算法加入对比，并在4个不同维度的地图中来检测3个算法的性能。运行结果取100次执行下的平均值±标准差，如表7所示，可以看出RRT-QSA*算法和A*算法的性能非常接近，说明了经改进后RRT-QSA*算法可以达到主流算法的性能要求，可以应用于实际机器人的路径规划。

如图16所示，在每个地图上，绿色方块表示不同的算法在进行路径搜索时访问到的节点。

如图17所示，与RRT算法相比，RRT-QSA*算法的计算时间减少了96.83%～99.88%，搜索节点数减少了86.62%～96.01%，路径长度数减少了9.9%～16.7%，转折角度减少了80.93%～93.04%。此外，随着地图大小的增加，RRT-QSA*算法的计算效率比RRT算法提升更显著。综上所述，相比于RRT算法，RRT-QSA*算法可以得到一条搜索时间更短、搜索领域更少、路径长度更短、转折角度更小且平滑的路径。

3.2　实验验证

为了验证本文算法的有效性和可行性，将RRT-QSA*算法应用到基于ROS的移动机器人Turtlebot2上，Turtlebot2采用激光雷达A2获取实验场景信息后利用amcl和hector模块进行定位和建图。

本实验的实验场景如图18 (a)所示，为4 m×4 m的封闭区域，机器人构建的地图为5 cm×5 cm网格，因此理论上构建的地图为80×80网格地图，如图18 (b)所示。

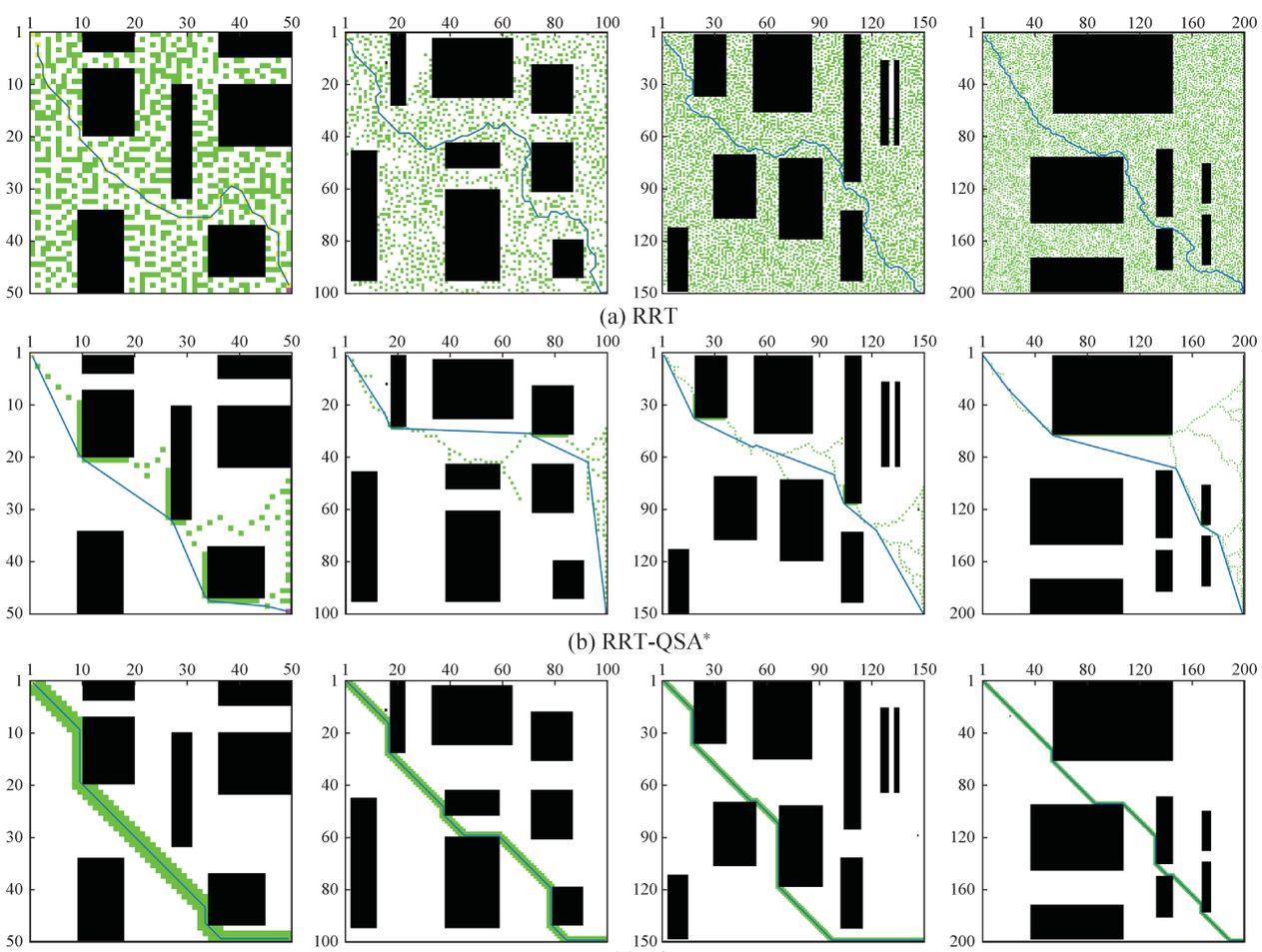
表6　算法数据对比(迷宫地图II)
Table 6　Algorithms statistics comparison (maze map-II)

指标	RRT				RRT-QSA*			
	时间/s	路径长度	节点数	转折角度/(°)	时间/s	路径长度	节点数	转折角度/(°)
均值	6.20	279.82	2 205.86	4 639.15	0.013	203.76	312.50	438.32
标准差	5.50	16.66	728.86	387.67	0.002	2.36	9.20	6.95
最大值	21.49	309.60	3 745	5 363.13	0.017	208.87	326	490.75
最小值	1.64	250.98	1 217	4 064.18	0.001	201.45	297	419.63

表7　在不同地图中各算法数据对比

Table 7　Algorithms statistics comparison (different maps)

参数	算法	50×50	100×100	150×150	200×200
时间/s	RRT	0.41±0.34	3.39±3.08	32.62±21.39	102.39±79.16
	RRT-QSA*	0.013±0.01	0.07±0.03	0.14±0.05	0.16±0.04
	A*	0.013±0.02	0.04±0.02	0.07±0.01	0.12±0.08
转折点数	RRT	553±129	1 714±674	5 074±1529	9 331±2 546
	RRT-QSA*	74±46	183±253	269±57	372±26
	A*	225±0	431±0	680±0	951±0
路径长度	RRT	94.1±6.04	183.67±10.69	283.78±10.41	349.82±18.34
	RRT-QSA*	78.3±4.31	159.39±6.19	244.71±9.70	315.18±10.68
	A*	76.9±0	173.39±0	242.93±0	299.59±0
转折角度/(°)	RRT	1 767±133.84	3 440±253.66	5 383±302.93	6 713±513.79
	RRT-QSA*	337±8.99	290±13.74	599±73.86	467±79.16
	A*	279±0	242±0	451.9±0	448.21±0



(a) RRT

(b) RRT-QSA*

(c) A*

图16　在不同地图中各算法仿真结果

Fig. 16　Simulation results of each algorithm in different maps

DOI: 10.16182/j.issn1004731x.joss.23-0585

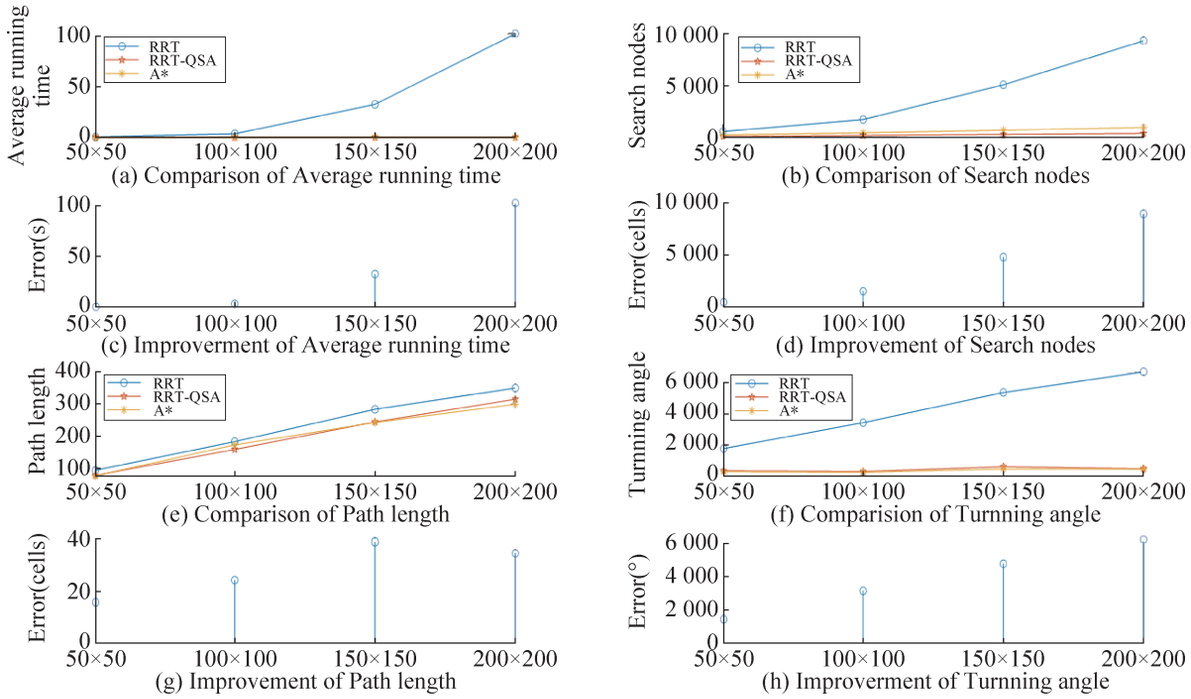


图 17　在不同地图中各算法性能对比

Fig. 17　Performance comparison of each algorithm in different maps

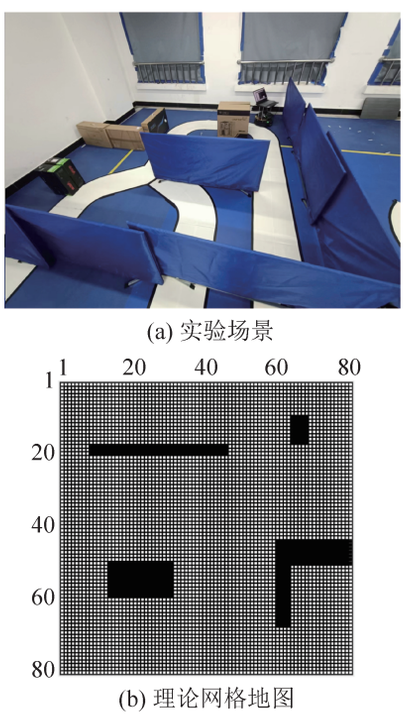
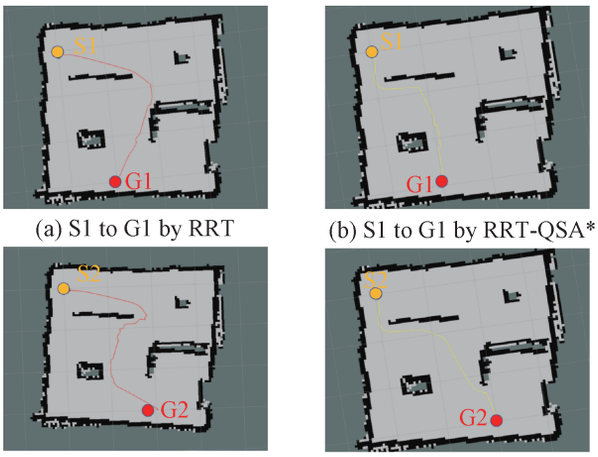


图 18　实验场景和理论网格地图

Fig. 18　Experimental scenes and theoretical grid maps

实验准备：①因为激光雷达 A2 是新接入的，所以需要在实验开始前修改雷达驱动和 slam 建图的 launch 文件，调整相关参数；②通过把 RRT-QSA* 算法通过 plugin 插件注册的方式移植进 move_base 模块中来替换 global_planner 进行全局路径规划，并通过打开 rqt 来查看话题关系图以验证插件是否注册成功；③启动键盘控制节点移动 Turtlebot2 构建出该实验场景的二维地图，再启动 map_server 模块将地图保存，随后将地图文件导入 launch 文件中。

实验流程：先启动雷达，然后连接小车，再启动导航文件，在进行路径规划前先在 rviz 中校准 Turtlebot2 的位置，起点设置为黄点，指定红色点为目标点来进行路径规划。为了更好地体现不同算法在实际应用时的性能，本文在实验中将碰撞半径设置为 0.3 m。在实验中将本文算法 RRT-QSA*与 RRT 算法作比较，寻路结果如图 19 所示，RRT 算法为红色路径，RRT-QSA*为黄色路径。由寻路结果和表 8 可知，RRT-QSA*在实际应用中得到的路径转折点更少、路径更短、搜索时间更少，实验结果证明本文算法能够有效完成移动机器人的路径规划任务。



(a) S1 to G1 by RRT (b) S1 to G1 by RRT-QSA*

(c) S2 to G2 by RRT (d) S2 to G2 by RRT-OSA*

图 19 RRT-QSA*和 RRT 算法的路径规划结果

Fig. 19 Path planning results of RRT-QSA* and RRT algorithms

表 8 搜索时间对比

Table 8 Comparison of searching time s

算法	S1-G1	S2-G2
RRT	1.5309	2.0347
RRT-QSA*	0.0326	0.0365

4 结论

RRT 算法在寻路过程中存在无效采样以及路径不最优等缺点，难以满足移动机器人在路径规划中的实时性要求。为了提高了路径规划的速度和保证路径的质量，本文提出拟水流避障算法 QSA*，对 RRT 算法加以改进，在搜索过程中优化搜索策略，在遇到障碍物时改用拟水流避障算法 QSA*进行避障，取代了原算法中随机采样的操作，从而提高寻路效率。仿真实验证明：RRT-QSA*在保证生成相对最优路径的同时，能够显著提高寻路速度，随着地图尺寸的提升，效果更加明显。将 RRT-QSA*应用在 Turtlebot2 机器人上的实验充分表明，RRT-QSA*算法可以满足实际要求，且优化效果明显。因此，在处理复杂地图或者强调实时计算的场景中，RRT-QSA*算法具有较强的应用优势。本文算法处理的是全局静态环境下的路径规划问题，对于其在动态空间或者高维地图下的应用拓展还待进一步研究。

参考文献：

[1] Patle B K, Ganesh Babu L, Anish Pandey, et al. A Review: On Path Planning Strategies for Navigation of Mobile Robot[J]. Defence Technology, 2019, 15(4): 582-606.

[2] Liu Lixing, Wang Xu, Yang Xin, et al. Path Planning Techniques for Mobile Robots: Review and Prospect[J]. Expert Systems with Applications, 2023, 227: 120254.

[3] 林韩熙, 向丹, 欧阳剑, 等. 移动机器人路径规划算法的研究综述[J]. 计算机工程与应用, 2021, 57(18): 38-48.
Lin Hanxi, Xiang Dan, Ouyang Jian, et al. Review of Path Planning Algorithms for Mobile Robots[J]. Computer Engineering and Applications, 2021, 57(18): 38-48.

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

[5] Hart P E, Nilsson N J, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[6] Koenig S, Likhachev M. Fast Replanning for Navigation in Unknown Terrain[J]. IEEE Transactions on Robotics, 2005, 21(3): 354-363.

[7] Lavalle S M. Rapidly-exploring Random Trees: A New Tool for Path Planning[J]. Research Report, 1999, 98 (11): 1-4.

[8] Wang Jiankun, Meng M Q H, Khatib O. EB-RRT: Optimal Motion Planning for Mobile Robots[J]. IEEE Transactions on Automation Science and Engineering, 2020, 17(4): 2063-2073.

[9] Liao Bin, Wan Fangyi, Hua Yi, et al. F-RRT*: An Improved Path Planning Algorithm with Improved Initial Solution and Convergence Rate[J]. Expert Systems with Applications, 2021, 184: 115457.

[10] 陈秋莲, 蒋环宇, 郑以君. 机器人路径规划的快速扩展随机树算法综述[J]. 计算机工程与应用, 2019, 55(16): 10-17.
Chen Qiulian, Jiang Huanyu, Zheng Yijun. Summary of Rapidly-exploring Random Tree Algorithm in Robot Path Planning[J]. Computer Engineering and Applications, 2019, 55(16): 10-17.

[11] 赵文龙, Abdou Yahouza M Sani. 基于改进RRT算法的移动机器人路径规划方法[J]. 计算机与数字工程, 2022, 50(8): 1733-1738.
Zhao Wenlong, Abdou Yahouza M Sani. Path Planning Method Based on Improved RRT Algorithm for Mobile Robot[J]. Computer & Digital Engineering, 2022, 50(8):

DOI: 10.16182/j.issn1004731x.joss.23-0585

1733-1738.

[12] 张伟民, 付仕雄. 基于改进RRT*算法的移动机器人路径规划[J]. 华中科技大学学报(自然科学版), 2021, 49(1): 31-36.
Zhang Weimin, Fu Shixiong. Mobile Robot Path Planning Based on Improved RRT* Algorithm[J]. Journal of Huazhong University of Science and Technology (Nature Science Edition), 2021, 49(1): 31-36.

[13] Li Yanjie, Wei Wu, Gao Yong, et al. PQ-RRT*: An Improved Path Planning Algorithm for Mobile Robots[J]. Expert Systems with Applications, 2020, 152: 113425.

[14] Wang Jiankun, Li Baopu, Meng M Q H. Kinematic Constrained Bi-directional RRT with Efficient Branch Pruning for Robot Path Planning[J]. Expert Systems with Applications, 2021, 170: 114541.

[15] Zhou Ying, Zhang Endong, Guo Hongling, et al. Lifting Path Planning of Mobile Cranes Based on an Improved RRT Algorithm[J]. Advanced Engineering Informatics, 2021, 50: 101376.

[16] 张瑞, 周丽, 刘正洋. 融合RRT*与DWA算法的移动机器人动态路径规划[J/OL]. 系统仿真学报. (2023-03-24) [2024-03-18]. https://doi. org/10.16182/j. issn1004731x. joss.22-1543.
Zhang Rui, Zhou Li, Liu Zhengyang. Dynamic Path Planning for Mobile Robot Based on RRT* and Dynamic Window Approach[J/OL]. Journal of System Simulation. (2023-03-24) [2024-03-18]. https://doi. org/10.16182/j. issn1004731x.joss.22-1543.

[17] Miao Changwei, Chen Guangzhu, Yan Chengliang, et al. Path Planning Optimization of Indoor Mobile Robot Based on Adaptive Ant Colony Algorithm[J]. Computers & Industrial Engineering, 2021, 156: 107230.

[18] 孙瑞, 张文胜. 基于改进蚁群算法的移动机器人平滑路径规划[J]. 图学学报, 2019, 40(2): 344-350.
Sun Rui, Zhang Wensheng. Smooth Path Planning of Mobile Robot Based on Improved Ant Colony Algorithm [J]. Journal of Graphics, 2019, 40(2): 344-350.

[19] Zhong Xunyu, Tian Jun, Hu Huosheng, et al. Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-scale Dynamic Environment[J]. Journal of Intelligent & Robotic Systems, 2020, 99(1): 65-77.

[20] Li Changgeng, Huang Xia, Ding Jun, et al. Global Path Planning Based on a Bidirectional Alternating Search A* Algorithm for Mobile Robots[J]. Computers & Industrial Engineering, 2022, 168: 108123.